Computer e Architettura

oltre le linee

UNIVERSITÀ DEGLI STUDI DI NAPOLI "FEDERICO II"
FACOLTÀ DI ARCHITETTURA
DIPARTIMENTO DI PROGETTAZIONE ARCHITETTONICA E AMBIENTALE

TESI DI DOTTORATO IN PROGETTAZIONE ARCHITETTONICA
E TECNOLOGIE INNOVATIVE PER LA SOSTENIBILITÀ AMBIENTALE - 22° CICLO

TUTOR: PROF. ANTONIETTA PIEMONTESE
DOTTORANDO: ROSARIO MARENA

Indice generale

Introduzione

Se col pensiero torno indietro alla mia adolescenza ricordo che, da grande, avrei voluto fare l'archeologo. Premesso che è possibile fare solo ciò di cui si ha conoscenza aggiungo che l'idea mi venne dopo aver fantasticato un po' a seguito della visione di un documentario su Heinrich Schliemann e la scoperta di Troia. Dico questo perché il seguente lavoro chiude proprio con una proposta di utilizzo del computer per ricostruire, in tempo reale, siti archeologici.

Lo scopo principale è quello di fare un po' il punto della situazione riguardo l'uso del computer nella progettazione architettonica ed in particolare mostrare, anche con esempi specifici, gli strumenti oggi a disposizione per progettare e rappresentare. Gli strumenti di ausilio a pensiero e immagine.

Il percorso comincia illustrando alcuni momenti importanti nella storia della definizione dei meccanismi logici che sottendono al pensiero umano prima e penso a Turing, a Minsky, a von Neumann e della progettazione e costruzione dei personal computer poi e penso a Wozniak, a Gates, a Jobs.

Esposizione del repertorio degli strumenti informatici dedicati alla progettazione e discussione dei loro fondamenti teorici, delle loro potenzialità e dei loro limiti.

Con l'istruzione, il gap generazionale e la pratica della trasmissione della conoscenza si fa una pausa di riflessione chiosando con l'attuale e futura diffusione dell'insegnamento su internet.

Il percorso si conclude con una serie di considerazioni su ipotetici scenari futuri ma anche e soprattutto con la concreta proposta di utilizzo di un sistema di realtà amplificata per gestire la ricostruzione in tempo reale di un'antica città come Pompei.

Per quanto concerne il dottorato di riferimento e, quindi, le tecnologie innovative per una progettazione sostenibile devo dire che non ho analizzato progetti tipo né studiato una particolare soluzione tecnologica ma ho affrontato lateralmente la questione anche dando spazio all'etica hacker riguardo la diffusione della conoscenza, nodale per la sostenibilità ambientale.

Aggiungo che, fra i software, sono trattati anche quelli maggiormente indirizzati ad una progettazione sostenibile ma, ovviamente, tutto ciò che è digitale è, potenzialmente, sostenibile.

La sostenibilità ambientale è, secondo me, intrinsecamente legata con l'intero mondo digitale che rappresenta un enorme passo avanti nell'utilizzo delle risorse e straordinariamente meno invasivo. Senza scendere, naturalmente, nel dettaglio basti pensare ai trasporti o agli oggetti.

Passata la fase dell'elettronica gigante e grossolana abbiamo cominciato ad affinarci e a sfruttare meglio ed in maniera più calibrata le risorse. Ci saranno sempre più interferenze fra l'uomo e l'elettronica ma è l'evidente legge dell'incontro per il cambiamento, processo di trasformazione. Qualcuno si lamenterà, qualcun'altro si esalterà. Fino a quando ci sarà possibile godiamoci lo spettacolo.

Un ringraziamento particolare alla prof. Antonietta Piemontese e a suo marito prof. Rolando Scarano; grazie a tutti quelli presenti in Bibliografia e Sitografia e grazie anche a tutti quelli che, presenti o meno, per mancanza di spazio non c'entrano.

Rosario Marena

Napoli, 30 novembre 2009

Capitolo 1° - Storia

Il percorso comincia illustrando alcuni momenti importanti nella storia della definizione dei meccanismi logici che sottendono al pensiero umano prima e penso a Turing, a Minsky, a von Neumann e della progettazione e costruzione dei personal computer poi e penso a Wozniak, a Gates, a Jobs.

Macchine intelligenti

"Il cervello è una macchina di carne" dichiarò Marvin Minsky alcuni anni or sono e molti, specialmente quelli che ritengono che l'intelligenza non potrà mai essere meccanizzata, rimasero estremamente turbati.

Marvin Minsky - *fonte Wikipedia*

Parto dall'affermazione di Minsky perché condivisa e perché credo serva a far compiere un salto concettuale, se non lo si è già fatto, molto importante per meglio comprendere la struttura logica alla base.

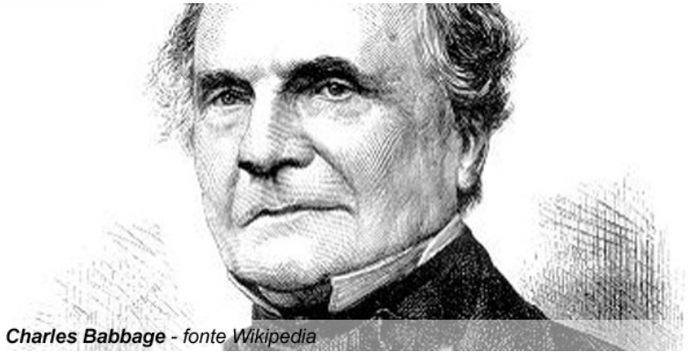
"Minsky è poco tollerante nei confronti di ciò che considera insensato, come ad esempio l'idea che sia impossibile comprendere la mente. Ma non crede neppure che la comprenderemo semplicemente catalogando le parti che la compongono". Così Jeremy Bernstein in *Uomini e macchine intelligenti* e continua citando Minsky da una delle numerose interviste fatte sulla sua vita e sul suo lavoro: "Credo che l'intelligenza non emerga da un piccolo numero di principi potenti e ben definiti, come la fisica, ma piuttosto da un centinaio di tipi di meccanismi fondamentalmente diversi, che devono interaggire nel modo giusto. Quindi, anche se per comprendere ciascun meccanismo ci volessero solo quattro anni, per sbrogliare tutta la matassa ci vorrebbero quattrocento anni".

Più cauto del grande Gottfried Wilhelm Leibniz che riteneva, trecento anni fa, di riuscire, con un drappello di uomini scelti e in soli cinque anni, a formalizzare tutto il pensiero e a rendere tutti i ragionamenti esatti. Jeremy Bernstein continua citando George Boole nel celebre *Indagine sulle leggi del pensiero* quale inventore di un nuovo tipo di algebra per arrivare a costruire il suo sistema di calcolo logico. Interessante poi è che verso il 1880 si tradusse il suddetto calcolo logico in circuiti elettrici, come descritto da Arthur W. Burks in *Logic, Biology and Automata - Some Historical Reflections*. Nel 1886 Charles Sanders Peirce in una lettera ad Alan Marquand: "Io credo che l'elettricità sia il mezzo migliore di cui servirsi. Siano A, B e C tre interruttori o comunque punti in cui il circuito può essere aperto o chiuso. Il circuito della figura 1 è chiuso solo se tutti gli interruttori sono chiusi; per quello della figura 2 basta che uno qualunque sia chiuso. Ciò corrisponde alla moltiplicazione e all'addizione della logica".

Marquand progettò una macchina logica in cui relè telefonici facevano da interruttori.

"Burks avanza la congettura, senz'altro corretta, che Peirce si fosse ispirato al lavoro di Charles Babbage; e questi è davvero una delle più singolari della scienza moderna. Nato nel 1792 in Inghilterra, nel Devonshire, ereditò dal padre banchiere un patrimonio considerevole, che spese tutto per finanziare i propri esperimenti scientifici. […] Fin da giovane, Babbage decise di dedicare la propria vita a costruire macchine che potessero sollevare gli uomini dall'ingrata fatica dei calcoli" continua Bernstein e ancora "Babbage era

un visionario, il cui massimo interesse risiedeva nelle macchine che egli immaginava di costruire e non in quelle che poteva effettivamente costruire. Dopo aver realizzato la macchina alle differenze, Babbage cercò di allestirne una versione più precisa, un dispositivo capace di fornire risultati precisi fino alla ventesima cifra decimale". Le calcolatrici attuali arrivano alla nona cifra decimale.



Charles Babbage - *fonte Wikipedia*

Molto di quanto sappiamo del lavoro di Babbage riguardo la macchina analitica lo si deve a Lady Lovelace, figlia del poeta Byron, rimasta affascinata fin da piccola quando, durante una visita, le capitò di vedere la macchina alle differenze.

Per programmare la macchina, Babbage intendeva sfruttare schede perforate come quelle del lionese J.M. Jacquard per fornire istruzioni alla macchina. Lady Lovelace: "Possiamo dire a ragione che la macchina analitica tesse configurazioni algebriche prorpio come il telaio di Jacquard tesse fiori e foglie".

La IBM ha fatto costruire, a mero scopo dimostrativo, sia la macchina alle differenze che la macchina analitica.

L'era moderna della computazione cominciò nel 1937 con Howard Aiken della Harvard e George R. Stibitz dei Laboratori Bell. Con un gruppo di ingegneri e basandosi sulle idee di Babbage Aiken e Stibitz cominciarono a progettare e costruire calcolatori elettromeccanici e impiegando, come commutatori, relè telefonici.

la macchina analitica di Babbage - *fonte Wikipedia*

Howard Aiken, che aveva cominciato la propria carriera come fisico, per sottrarsi all'improba fatica di risolvere le equazioni della tesi di dottorato, cercò di progettare una macchina che le risolvesse per lui e nel 1939, sovvenzionato dall'IBM, cominciò a costruirne una. La macchina nacque già superata perché nel maggio del 1943, complice il matematico Herman M. Goldstine, era stato messo in cantiere il primo calcolatore veramente elettronico, l'ENIAC (Electronic Numerical Integrator And Calculator).

Nell'agosto del 1944 il progetto ENIAC acquisì una nuova importante recluta: von Neumann. Goldstine ricorda che incontrandolo su una banchina della stazione cominciò a conversare con lui: "quando von Neumann si rese conto che ero impegnato nella costruzione di un calcolatore elettronico capace di eseguire 333 moltiplicazioni al secondo, il tono della nostra conversazione mutò completamente e quella che era una distesa e piacevole chiacchierata diventò una sorta di esame orale per il dottorato in matematica".

Dopo aver preso visione del progetto ENIAC i calcolatori, gli automi e il loro rapporto

con la mente umana divennero uno dei temi dominanti del lavoro di von Neumann. Pubblicò una serie di rapporti monumentali, scritti in collaborazione con Burks, Goldstine e altri. Nel 1952 le loro idee furono messe in pratica in un calcolatore costruito presso l'Institute for Advanced Study di Princeton. Von Neumann ravvisava un legame tra quelle ricerche e ciò che oggi si chiama intelligenza artificiale. Com'è confermato da Goldstine, von Neumann subì profondamente l'influenza di un articolo scritto nel 1943 da Warren McCulloch e Walter Pitts *A Logical Calculus of the Ideas Immanent in Nervous Activity* in cui venivano poste le basi dell'IA. Quest'articolo divenne notissimo grazie all'interesse provato e provocato da von Neumann che, dopo averlo letto, ne adottò le notazioni in gran parte dei lavori scritti in seguito sulla teoria degli automi.

John von Neumann *- fonte Wikipedia*

Figura eccentrica e originalissima continua Bernstein: "McCulloch passò certo il resto della vita curandosi di se stesso. Non era un matematico, e questo è sicuramente il motivo per cui si associò a Walter Pitts, che lo era; quando pubblicarono il loro primo articolo, Pitts aveva diciassette anni. Costretto ad abbandonare le scuole superiori dal padre, che voleva farlo lavorare, Pitts scappò di casa e alla fine si ritrovò a Chicago. Qui, secondo un aneddoto che McCulloch amava ricordare, il giovane Pitts passava lunghe ore, nel parco vicino all'università, conversando di filosofia con un uomo che conosceva solo come Bert, e che poi era Bertrand Russell, professore ospite alla University of Chicago nell'anno accademico 1938-39. Sempre secondo McCulloch, Russell consigliò a Pitts di leggere un libro di Rudolf Carnap, il grande logico e filosofo della scienza che stava a Chicago e che frequentava il seminario di Russell. Pitts lesse il libro e poi andò a trovare Carnap per dirgli che aveva scoperto un errore. Non sono riuscito a trovare questo episodio nella minuziosa biografia di Russell scritta da Ronald Clark, benché le date corrispondano. Fu a Chicago che McCulloch cominciò a lavorare con Pitts, che poi fu suo studente alla University of Illinois. Il loro articolo del 1943 è di ardua lettura; e il paragrafo intitolato *Nets with Circles* (un tentativo di spiegare come le reti neurali apprendano e si evolvano) è sicuramente errato".

Minsky ricorda di averlo letto quando era ancora studente: "Mi capitò di leggere piuttosto presto il loro articolo e ne capii quanto potevo capirne. George Miller, che allora era assistente alla Harvard e s'interessava di psicologia matematica, non riusciva a comprendere la seconda parte, mentre non aveva avuto alcun problema con la prima. Allora mi ci misi d'impegno e alla fine potei tranquillizzarlo, dicendogli che secondo me la notazione non era coerente e le definizioni erano sbagliate. Penso che McCulloch fosse rimasto abbagliato da Pitts, che allora, adolescente, era già un matematico prodigio. In seguito cercai di discutere l'articolo con Pitts, che era venuto al MIT con McCulloch, ma egli si rifiutò sempre di parlarne. Ne trassi la conclusione che Pitts, che morì nel 1960, avesse in mente una teoria che non riuscì mai ad elaborare compiutamente. Forse stava barando e McCulloch ci era in qualche modo cascato. Rassicurai Miller dicendogli che se non riusciva a capirlo non era colpa sua: c'era qualcosa che non andava. Ma mi ci volle un bel coraggio, per arrivare a questa conclusione".

Continua Bernstein: "nondimeno rimane da spiegare perché il saggio di McCulloch e Pitts e quelli che seguirono abbiano esercitato un'influenza tanto importante su Minsky e sui suoi contemporanei nel momento in cui tentavano di cominciare a comprendere l'intelligenza. Per rispondere è necessaria qualche nozione sulla fisiologia del sistema nervoso. Agli inizi del secolo, grazie alle ricerche avviate da Santiago Ramón y Cajal e da

altri, si potè accertare che nel cervello si trovano da dieci a cento miliardi di cellule nervose distinte, i neuroni. Ciascuno possiede fino a diecimila sinapsi, le porte d'entrata attraverso le quali gli arrivano i segnali inviatigli da altri neuroni. Ciascuno ha anche un assone, una fibra in grado di trasportare gli impulsi nervosi prodotti dal neurone quando *scarica* e può scaricare fino a un migliaio di volte al secondo. Il neurone possiede una *soglia di scarica:* se i segnali che riceve superano un certo valore critico, esso scarica, altrimenti no (i neuroni malati possono scaricare spontaneamente e non sono regolati dall'informazione d'ingresso). Certi segnali d'ingresso possono essere negativi, cioè inibitori, altri positivi; ma ciò che conta è se la somma di questi segnali d'ingresso supera o no la soglia critica. Se l'assone si ramifica, l'impulso emesso da un neurone può dividersi e può fornire così un segnale d'ingresso a parecchi neuroni vicini. La velocità di propagazione dell'impulso varia da 1 m/s negli assoni sottili fino a 150 m/s in quelli grossi. La domanda alla quale McCulloch e Pitts cercarono di rispondere è: che cosa è in grado di fare una rete neurale di questo genere? Per affrontare il problema, essi sostituirono alla rete reale una rete fittizia o ideale, in cui ogni neurone era una specie di scatola nera definita da cinque proprietà, o assiomi.

Ecco l'elenco degli assiomi di McCulloch e Pitts:

1) L'attività di un neurone è un processo *tutto o niente*. [Un neurone o scarica o non scarica].

2) In qualunque istante, perché un neurone sia eccitato è necessario che nel periodo di addizione latente venga eccitato un numero di sinapsi ben determinato; questo numero è indipendente dall'attività precedente e dalla posizione del neurone. [Il neurone ha una soglia di scarica ben determinata, che deve essere raggiunta affinché esso scarichi].

3) L'unico ritardo significativo all'interno del sistema nervoso è il ritardo sinaptico. [La velocità con cui gli impulsi si propagano lungo gli assoni è così grande che, in effetti, il tempo di ciclo della rete coincide con il tempo impiegato dal neurone a scaricare e poi a tornare allo stato normale, non eccitato, cioè circa un millesimo di secondo].

4) Se una qualunque sinapsi inibitrice è attiva, l'eccitazione del neurone in quel momento è assolutamente impedita. [Negli anni in cui McCulloch e Pitts scrissero questo articolo, si credeva che gli impulsi inibitori esercitassero un'azione incondizionata; ora sappiamo invece che essi si sommano al resto dell'ingresso sinaptico. Questo è uno degli elementi del loro articolo che richiesero una rielaborazione].

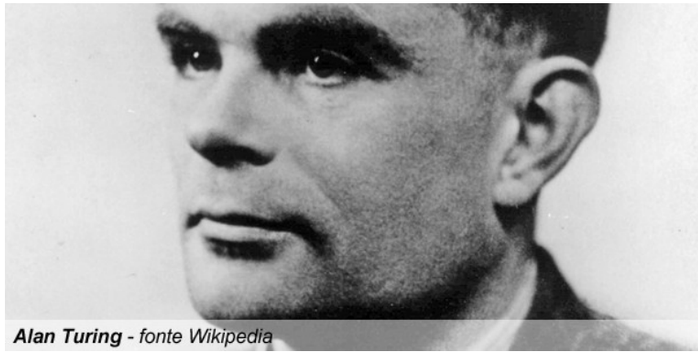
5) La struttura della rete non muta nel tempo.

L'ultimo assioma significa, presumibilmente, che i componenti di questa rete ideale sono considerati immortali. In realtà, una delle caratteristiche più sorprendenti dei neuroni è che essi muoiono e non vengono sostituiti. Nel corso della propria esistenza, ogni individuo perde circa il dieci per cento dei neuroni; e tuttavia continua a operare, mentre una radio, per esempio, cessa di funzionare se le si toglie un transistore. Nel 1962, durante un convegno sul controllo neurale e la memorizzazione del l'informazione, a McCulloch venne fatta una domanda sui neuroni morti o lesi ed egli rispose:

'Non ho mai visto il mio cervelietto, ma ho visto quello di molti individui della mia età. Ho più di cinquant'anni, e ritengo che a questa età nel mio cervelietto almeno il dieci per cento delle cellule di Purkinje siano state sostituite da graziosi buchini, ma riesco ancora a toccarmi il naso. È incredibile quanto è piccola la quantità di cervello che deve rimanere integra perché esso possa continuare a funzionare'.

Ispirandosi all'articolo di McCulloch e Pitts, von Neumann intraprese uno studio approfondito sul modo di organizzare un automa affidabile - ad esempio una rete neurale - a partire da componenti non affidabili. Egli comprese che uno dei metodi chiave è quello di introdurre nel sistema una certa ridondanza, cioè di fare in modo che vi siano due o più componenti che possano subentrare l'uno all'altro in caso di difetto. Egli considerò anche la possibilità di agganciare i sistemi fra loro, in modo che potessero controllarsi a vicenda, e diede il seguente esempio concreto: supponiamo di avere tre macchine identiche, ciascuna in grado di eseguire un calcolo molto lungo, per esempio di un milione di passaggi, con una probabilità su cento di commettere un errore in qualche punto. Ciò sarebbe inaccettabile, poiché nessuna delle tre macchine potrebbe eseguire tutto il calcolo correttamente. Se però le

macchine lavorassero di conserva, in modo che, in caso di disaccordo tra due di esse, fosse preso per buono il responso della terza e poi il calcolo continuasse, la probabilità di errore passerebbe da uno su cento a uno su trentatré milioni. Von Neumann congetturò che nel sistema nervoso centrale sia incorporato un qualche genere di ridondanza, probabilmente ogni genere di ridondanza; e oggi sappiamo che le cose stanno proprio così".



Alan Turing - *fonte Wikipedia*

Le idee di von Neumann sono state applicate nella progettazione dei calcolatori, addirittura fondamentali per le macchine di ultima generazione.

Riprendendo un'opinione di von Neumann possiamo dire che Alan Turing ha influito più d'ogni altro sulla logica degli automi e, per ironia della sorte, i suoi lavori più importanti risalgono agli anni Trenta, quando i calcolatori non esistevano ancora. Ma egli seppe concepirli con una costruzione mentale e compì un'analisi profondissima di ciò che macchine del genere potrebbero fare. La sua macchina ideale somiglia a un calcolatore concreto come la rete neurale di McCulloch e Pitts somiglia a un vero cervello.

Il Tech Model Railroad Club

II Tech Model Railroad Club (TMRC) è una organizzazione studentesca del Massachusetts Institute of Technology (MIT), uno dei più famosi club sui modellini di treni, e fonte di cultura hacker. Formatosi dopo la seconda guerra mondiale e varie vicissitudini occupò lo storico palazzo 20.

I membri del club, che hanno condiviso la passione per scoprire il funzionamento delle cose per poi controllarle, sono stati tra i primi hacker. Alcuni dei primi membri chiave del club sono stati Jack Dennis e Peter Samson, che ha compilato il Dizionario 1959 della Lingua TMRC e che alcuni dicono coniato la frase *Information wants to be free*.

Il club era composto da due gruppi: i *taglierino e pennello* più interessati a pittura e repliche di treni con valori storici ed emotivi, e coloro che comprendendo i circuiti facevano circolare i treni. Questi ultimi, i *Signal and Power*, sarebbero stati tra quelli che hanno reso popolare il termine "hacker", tra i molti termini slang che successivamente si trasferirono al computer e alla programmazione. Inizialmente attratti dall'IBM 704, il multi-mainframe da milioni di dollari del palazzo 26, il gruppo ha iniziato davvero ad essere coinvolto con i computer quando Jack Dennis, un ex membro, mostrò loro il Tx-0, un computer prestato, a lungo termine, dal Lincoln Laboratory.

Nel circolo stesso un sistema semi-automatico di controllo, basato su relè telefonici, è stato installato dalla metà del 1950. Nel 1964 circa questo è stato sostituito da un secondo sistema più efficace; lead designer di questo progetto è stato Alan Kotok, allora una stella nascente del personale di progettazione presso la *Digital Equipment Corporation*. Le attrezzature vennero donate dalla compagnia telefonica tramite la *Western Electric College Gift Plan*.

La *Digital Equipment Corporation* donò due computer Pdp-11 nel 1970. Uno dei due sostituì la tastiera di una vecchia macchina perforatrice a schede installata in origine da Richard Greenblatt.

Il palazzo 20, la casa del TMRC per 50 anni, fu demolito nel 1997. Al club è stato offerto un nuovo spazio nel palazzo N52 del *MIT Museum*.

***IBM 704** - fonte Wikipedia*

Così scrive Steven Levy in *Hackers*: "Il vero motivo per cui Peter Samson, nel cuore della notte, stava vagabondando nel palazzo 26 è una faccenda che lui stesso avrebbe trovato difficile da spiegare. Di certe cose non si parla. Se foste come quelli con cui Peter era sul punto di far conoscenza e di cui sarebbe diventato amico, in quel suo anno da matricola al Massachussetts Institute of Technology, nell'inverno del 1958-59, non ci sarebbe stato bisogno di alcuna spiegazione. Vagabondare per il labirinto di laboratori e magazzini, in cerca dei misteri della commutazione telefonica nelle stanze piene di apparecchiature, seguendo sentieri di fili o di relè nei condotti sotterranei dell'aria condizionata, per qualcuno era un comportamento normale, e non c'era bisogno di giustificare l'impulso di aprire una porta senza permesso, se dietro quella porta si fosse percepito un rumore sospetto irresistibilmente attraente. E allora, se non ci fosse stato nessuno a proibire fisicamente l'accesso a qualunque cosa stesse provocando quel rumore affascinante e a impedire di toccare la macchina, ecco che avreste cominciato a sfiorare gli interruttori e a osservare le reazioni, a girare una vite, sganciare un pannello, rimuovere qualche diodo e provare qualche connessione. Peter Samson e i suoi amici erano cresciuti in una particolare relazione col mondo, all'interno della quale le cose acquisivano significato solo se si scopriva come funzionavano. E come avrebbe potuto capirlo se non mettendoci le mani sopra?

Fu nel seminterrato del palazzo 26 che Samson e i suoi amici scoprirono la stanza *Eam*. Il palazzo 26 era un'alta struttura di vetro e acciaio, uno degli edifici più recenti del MIT, in contrasto con le venerande architetture neoclassiche che fronteggiavano l'Istituto su Massachusetts Avenue. Nel seminterrato di questo edificio, privo di personalità, c'era la stanza *Eam*, l'*Electronic account machinery*, che ospitava dei macchinari che funzionavano come computer.

Nel 1959 non molta gente aveva visto un computer, figuriamoci poi toccarne uno. Samson - un ragazzo dai capelli rossi, ispidi e ricci con una propensione ad allungare il suono delle vocali come se stesse correndo dietro ai possibili significati delle frasi, mentre si trovava nel mezzo delle parole - aveva visto i computer durante le sue visite al MIT dalla sua città natale, Lowell, nel Massachusetts, a meno di cinquanta chilometri dall'Università. Queste visite lo avevano reso un *fanatico di Cambridge*, uno dei tanti studenti di liceo della regione, pazzi per la scienza, che erano stati attratti, come da una forza gravitazionale, verso l'Università di Cambridge. Aveva persino cercato di mettere assieme il suo computer personale con pezzi di scarto di vecchi flipper: erano la migliore fonte di elementi logici che avesse potuto trovare. *Elementi logici:* il termine sembra contenere proprio quel che attraeva Samson, figlio di un riparatore di macchine per l'industria, verso l'elettronica. Era la sua storia. Quando si cresce con un'insaziabile curiosità sul funzionamento delle cose, il piacere che si prova scoprendo quanto è raffinato un circuito logico, dove tutte le connessioni devono completare i loro percorsi, è eccitante a dismisura. Peter Samson, che aveva imparato presto ad apprezzare la semplice perfezione matematica di tutto ciò, ricordava di aver assistito sul canale della Tv pubblica di Boston, la Wgbh, a una trasmissione che forniva un'introduzione sommaria alla programmazione di un computer nel suo specifico linguaggio. Bastò ad accendere la sua immaginazione: per Peter un computer era di certo come la lampada di Aladino che una volta sfregata, avrebbe obbedito ai suoi ordini. Così

cercò di imparare più che poteva in quel campo: costruì vari aggeggi per conto proprio, s'iscrisse a concorsi e competizioni scientifiche, e arrivò dove la gente della sua specie aspirava ad arrivare: il MIT. Il porto d'arrivo dei più brillanti liceali dagli occhiali da gufo e pettorali sottosviluppati, che meravigliavano gli insegnanti di matematica e che venivano bocciati in educazione fisica, che non sognavano di pomiciare dopo il ballo del diploma, ma di accedere alle finali del concorso in occasione della Fiera della scienza della *General Electric*. Questo era per lui il MIT, nei cui corridoi avrebbe vagato alle due di notte, cercando qualcosa d'interessante, e dove alla fine avrebbe scoperto quel che lo avrebbe spinto a iniziare una nuova forma di processo creativo, un nuovo stile di vita, e che lo avrebbe posto in prima linea di una società immaginata solo da qualche scrittore di fantascienza di serie B. Qui avrebbe scoperto un computer con cui *giocare"*.

Peter Samson era stato membro del Tech model railroad club fin dalla prima settimana trascorsa al Mit, nell'autunno del 1958. I suoi membri, studenti più anziani dall'aspetto distinto e dall'occhio sveglio, che parlavano col ritmo frenetico della gente che vuole togliere di mezzo le parole in un attimo, vantavano una singolare esposizione di trenini scala HO che tenevano in un locale della sede del club al palazzo 20. Peter Samson era da tempo affascinato dai treni, specialmente quelli delle metropolitane. Così si unì alla visita del palazzo, una costruzione provvisoria con il tetto di legno costruita durante la Seconda guerra mondiale. I corridoi erano cavernosi e, anche se la stanza del club era al secondo piano, l'umidità e la luce offuscata davano la sensazione di essere in un seminterrato.

La stanza del club era occupata dall'enorme plastico ferroviario. I membri del Signal and Power erano ossessionati dal modo in cui il *sistema* funzionava, dalla sua crescente complessità, dal modo in cui avrebbero reagito le altre parti a qualsiasi cambiamento, dal come usare quelle connessioni tra le parti per ottenere il massimo.

Molti pezzi del *sistema* erano stati regalati tramite il *Western electric college gift plan,* direttamente dalla compagnia dei telefoni. Il rappresentante di facoltà dei club era anche l'incaricato del sistema telefonico universitario e aveva fatto in modo che una sofisticata attrezzatura telefonica fosse a disposizione dei modellisti. Attraverso il disco combinatore dei telefoni, i "macchinisti" del TMRC potevano specificare su quale tratto del binario esercitare il controllo, e da là far partire un treno. Un'altra persona che gravitava intorno all'S&P era Alan Kotok, un tipo grassoccio del New Jersey, senza mento, un occhialuto dalle lenti spesse della classe di Samson. Alle superiori era stato in visita al Mobil Research Lab nella vicina Haddonfield, e vide il suo primo computer: l'ebbrezza di quell'esperienza lo spinse a decidersi d'entrare al MIT. Nel suo anno da matricola, si guadagnò la reputazione di uno dei più capaci S&P del TMRC.

Gli S&P erano quelli che passavano il sabato recuperando pezzi al rottamaio di Eli Heffron a Somerville, quelli che stavano sdraiati per ore su piccole sedie a rotelle che chiamavano *bunkies* per scivolare a lavorare sotto i punti difficili del *sistema*, che avrebbero faticato anche di notte per realizzare una linea totalmente abusiva tra il telefono del TMRC e il campus zona est. I più produttivi tra quelli che lavoravano all'S&P si definivano, con grande orgoglio, *"hacker"*. Dentro i confini della stanza del palazzo 20, e della *tool room* dove si svolgevano un po' di studio e molte sessioni di cazzeggio tecnico, si erano unilateralmente conferiti attributi eroici da saga nordica.

Ecco come Samson vedeva sé e i suoi amici in un poema alla Sandburg pubblicato sul foglio del club:

Installatore di commutatori al servizio del mondo intero,
tester di fusibili, creatore di piste,
giocatore delle strade ferrate e sezionatore spinto del sistema,
sciatto, spettinato, sbracato.
Macchina del punto-funzione, linea di luce:
mi dicono che sei malvagio e ci credo; poiché ho visto
le tue lampadine colorate sotto la lucite adescare i servi del sistema...
Sotto la torre, polvere ovunque,
tagliando con le pinze.

Hackerando come persino fa un'ignorante matricola che non ha mai perso il diritto
d'accesso ma è stato buttato fuori lo stesso,
Hackerando le schede madri, perché sotto i loro fermi si trovano
gli scambi e sotto il loro controllo si avanza lungo il modellino.
Hackerando!
Hackerando gli hack sciatti, spettinati e sbracati della giovinezza;
diodi sfrigolanti, non collegato, orgoglioso di essere installatore
di commutatori, tester di fusibili, creatore di piste,
giocatore delle strade ferrate e sezionatore spinto del sistema.

Appena potevano, Samson e gli altri s'infilavano nella stanza Eam con le loro *plug-board,* cercando di usare la macchina per controllare gli scambi sotto il modellino. E, cosa altrettanto importante, cercavano di vedere cosa poteva fare il calcolatore elettromeccanico, misurandone i limiti.

Sempre Levy: "in quella primavera del 1959, al MIT era stato inaugurato un nuovo corso, il primo corso di programmazione per computer che le matricole avrebbero potuto frequentare. L'insegnante era un uomo distaccato, dalla chioma selvaggia e una barba ugualmente ribelle, John McCarthy. Docente di matematica, McCarthy era il classico professore distratto: abbondano aneddoti sulla sua abitudine di rispondere improvvisamente a una domanda, ore, talvolta perfino giorni dopo che gli era stata posta. Ti si avvicinava nel corridoio e, senza salutarti, cominciava a parlare con una dizione roboticamente esatta, come se la conversazione fosse stata interrotta solo una frazione di secondo prima, e non da una settimana. Molto probabilmente però, il suo tardivo responso sarebbe stato illuminante.

McCarthy era anche uno dei rari ricercatori a lavorare sui computer con un metodo completamente nuovo. La natura mutevole e controversa del suo campo di studi affiorava dall'altisonanza del nome che aveva dato alla ricerca: *intelligenza artificiale.* Quest'uomo pensava davvero che i computer potessero essere *intelligenti.* Perfino in un luogo di intensa ricerca scientifica come il MIT, molti consideravano un tale concetto ridicolo: giudicavano i computer utili strumenti, seppur assurdamente cari, per eseguire enormi calcoli e progettare sistemi missilistici di difesa (come aveva fatto il più potente computer del MIT, Whirlwind, per il sistema d'allarme avanzato Sage), ma deridevano l'idea che i computer potessero di per sé costituire un campo di studio. Ancora alla fine degli anni Cinquanta al MIT non esisteva ufficialmente la scienza informatica e McCarthy e i suoi compagni specialisti di computer lavoravano all'istituto d'ingegneria elettrica, da cui dipendeva il corso n. 641. Kotok, Samson e pochi altri membri del TMRC quella primavera presero a frequentarlo.

McCarthy aveva lanciato un mastodontico programma sull'IBM 704, il "bestione", che gli avrebbe dato la straordinaria capacità di giocare a scacchi. Per i critici del campo nascente dell'intelligenza artificiale, questo non era altro che uno degli esempi di insensato ottimismo da parte di gente come John McCarthy. Ma McCarthy aveva una certa idea di ciò che i computer potevano fare, e giocare a scacchi era solo l'inizio".

John McCarthy *- fonte Wikipedia*

E ancora: "un giorno un vecchio socio del TMRC, che poi andò a lavorare al MIT, fece visita al club. Si chiamava Jack Dennis. Quando era studente, agli inizi degli anni Cinquanta, aveva lavorato forsennatamente sotto la struttura del modellino ferroviario, ma poco prima

aveva lavorato su un computer che il MIT aveva da poco ricevuto dal Lincoln lab, un laboratorio di applicazioni militari affiliato all'istituto. Il computer si chiamava Tx-0, ed era uno dei primi al mondo che funzionava a transistor. Il Lincoln lab lo aveva usato specialmente per mettere alla prova un gigantesco computer chiamato Tx-2, che disponeva di una memoria così complessa che soltanto questo fratellino costruito appositamente era in grado di diagnosticarne i mali. Ora che il lavoro per cui era stato progettato era finito, i tre milioni di dollari del Tx-0 avevano preso il largo verso l'istituto come "investimento a lungo termine", e apparentemente nessuno al Lincoln lab aveva segnato la data del rientro. Dennis chiese ai componenti dell'S&P del TMRC, se avessero piacere di vederlo.

Ehi voi, suore!... Vi piacerebbe incontrare il Papa?

Il Tx-0 era al palazzo 26, al secondo piano del Research laboratory of electronics (Rle), proprio sopra al primo piano del centro di calcolo che ospitava il mastodontico IBM 704".

***Tx-0** - fonte Wikipedia*

Non ci fu verso di tenere Kotok, Saunders, Samson lontani da quella macchina. Fortunatamente, intorno al Tx-0 non sembrava esserci il tipo di burocrazia che circondava l'IBM 704. Niente quadri di clero officiante. Il tecnico responsabile era un abile scozzese dai capelli bianchi di nome John McKenzie. Se da una parte si assicurava che i laureandi e quelli che lavoravano a progetti pagati - gli utenti ufficialmente registrati - avessero libero accesso alla macchina, McKenzie tollerava la piccola folla di matti del TMRC che cominciava a recarsi all'Rle, dove c'era il Tx-0.

Gli hacker venivano fuori di notte. Era l'unico modo per trarre il massimo vantaggio dagli importantissimi "tempi morti" del Tx-0. Di giorno, Saunders riusciva generalmente a comparire in due o tre corsi. Il resto del tempo lo occupava eseguendo "manutenzione generica", cose come mangiare e andare in bagno. Vedeva Marge per un momento, ma alla fine si rinchiudeva nel palazzo 26. Esaminava alcuni programmi della notte precedente stampati sulla carta da 25 centimetri usata dal Flexowriter. Faceva note e modifiche al tabulato per aggiornare il codice verso quello che riteneva lo stadio successivo. Andava al Model railroad club, a scambiare i suoi programmi con quelli di qualcun altro, controllandone simultaneamente le idee buone e gli errori potenziali. Poi tornava al palazzo 26, nella "kludge room" accanto al Tx-0, a cercare un Flexowriter libero per aggiornare il suo codice. Nel frattempo controllava che qualcuno non avesse cancellato una sessione di lavoro sulla macchina; la sua era verso le due o le tre del mattino. Aspettava che venisse il suo turno nella "kludge room", o giocava a bridge al Railroad club.

Il gruppo "taglierino e pennello" del TMRC non vedeva molto favorevolmente le infiltrazioni di tixomania nel club: lo ritenevano una sorta di cavallo di Troia per una svolta nel fulcro dell'interesse del club: dal modellismo ferroviario all'informatica. E se aveste assistito a una delle riunioni del club che si tenevano ogni giovedì alle cinque e un quarto, avreste afferrato la preoccupazione: gli hacker sfruttavano ogni possibile spunto della procedura dibattimentale per creare una riunione complicata quanto i programmi ai quali stavano lavorando sul Tx-0. Venivano fatte mozioni per fare mozioni che facevano mozioni, e obiezioni respinte come fossero bug. Samson era uno dei più recidivi e, a un certo punto, un membro del TMRC, esasperato da quanto stava accadendo, propose una mozione "per comprare un tappo e frenare la logorrea di Samson".

Hackerare le regole dell'assemblea era facile, ma la disposizione d'animo logica richiesta dalla programmazione si riversava anche nelle altre attività comuni. Se aveste posto

una domanda a un hacker avreste sentito il suo accumulatore mentale sottoporre a esame ogni bit prima di formulare la risposta. Marge Saunders andava al supermercato con la sua Volkswagen ogni sabato mattina e al ritorno chiedeva al marito: "Ti *dispiace* aiutarmi a portar dentro la spesa?" Bob Saunders rispondeva: "No!" Allibita, Marge trascinava da sola le provviste in casa. Dopo che la stessa scena si ripeté un po' di volte, esplose, imprecando contro Bob ed esigendo una spiegazione per il "no".

"È una domanda sciocca", le rispose. "È ovvio che non mi *piace* aiutarti a portare dentro la spesa. Ma se mi chiedi se t'aiuterò a portar dentro la spesa, be', quella è un'altra storia."

Era come se Marge avesse immesso un programma nel Tx-0, e il programma, come tutti quando la sintassi è impropria, si fosse impiantato. Era bastato correggere la domanda e Bob Saunders l'aveva fatto girare con successo sul suo computer mentale.

Hacker

Questo saggio è libero; è possibile distribuirlo e/o modificarlo secondo i termini della licenza GNU General Public License come pubblicata dalla Free Software Foundation; si applica la versione 2 o successiva.

Prologo: i Real Programmer

In principio furono i Real Programmer. Non è così che si definirono. Ma neanche "hacker", o qualcosa in particolare; il nomignolo "Real Programmer" fu coniato solo dopo il 1980. Fin dal 1945, ad ogni modo, la tecnologia informatica ha attirato molte delle menti più brillanti e creative del pianeta. A partire dall'ENIAC di Eckert e Mauchly in avanti, è sempre esistita cultura tecnica più o meno continua e autocosciente di programmatori entusiasti, di persone il cui rapporto col software era di puro divertimento.

I Real Programmer di solito provenivano dai settori dell'ingegneria e della fisica. Indossavano calzini bianchi, camicie e cravatte in poliestere e lenti spesse, programmavano in linguaggio macchina, in FORTRAN e in un'altra mezza dozzina di linguaggi ormai dimenticati. Si tratta dei precursori della cultura hacker, dei ben poco celebrati protagonisti della sua preistoria.

Dalla fine della Seconda Guerra Mondiale ai primi anni '70, nella grande era dei computer a linea di comando e dei mainframe chiamati "big iron", i Real Programmer dominarono la scena della cultura tecnica dei computer. Alcuni articoli del venerato folklore degli hacker risalgono proprio a quegli anni: la famosa storia di Mel (inclusa nel Jargon File), vari passi di Murphy's Laws (La legge di Murphy) e il manifesto caricatura tedesco "Blinkenlights" che ancora adorna molte sale di computer.

Molte persone formatesi nella cultura "Real Programmer", rimasero attive anche negli anni '90. Si narra che Seymour Cray, progettista della linea di supercomputer Cray, abbia una volta trasportato un intero sistema operativo di sua concezione in un computer di sua creazione. Col sistema ottale e senza un errore. Funzionò tutto alla perfezione. Macho supremo dei Real Programmer.

Su scala minore, Stan Kelly-Bootle, autore di The Devil's DP Dictionary (McGraw-Hill, 1981) e straordinario folklorista, programmò sul Manchester Mark I nel 1948, il primo computer digitale completamente operativo. Oggi scrive strisce tecnico-umoristiche per riviste di informatica, che spesso hanno il sapore di un energico e complice dialogo con la odierna cultura hacker.

Altri, come David E. Lundstrom, hanno scritto aneddoti su quei primi anni (A Few Good Men From UNIVAC, 1987).

Ciò che, comunque, ebbe origine dalla cultura "Real Programmer", è lo slancio innovativo che investì il computer interattivo, le università e le reti. Elementi che hanno avuto un ruolo fondamentale nella nascita di una tradizione tecnica che sarebbe sfociata nell'attuale cultura hacker Open Source.

I primi hacker

L'origine della cultura hacker, come oggi la conosciamo, può essere fatta risalire al 1961, anno in cui il MIT acquistò il primo PDP-1. Il comitato Signals and Power del Club Tech Model Railroad del MIT, adottò la macchina quale prediletto giocattolo-tecnologico creando strumenti di programmazione, linguaggi e quell'intera cultura che ancora oggi ci appartiene in modo inequivocabile. Questi primi anni sono stati esaminati nella prima parte del libro Hackers di Steven Levy (Anchor/Doubleday, 1984).

Pdp-1 - fonte Wikipedia

La cultura informatica del MIT sembra essere stata la prima ad adottare il termine "hacker". Gli hacker della TMRC divennero il nucleo dell'Artificial Intelligenge Laboratory (Laboratorio di Intelligenza Artificiale) del MIT, il principale centro di ricerca AI (Intelligenza Artificiale) su scala mondiale, nei primi anni '80. La loro influenza si protrasse ben oltre il 1969, il primo anno di ARPAnet.

ARPAnet è stata la prima rete transcontinentale di computer ad alta velocità. Ideata e realizzata dal Ministero della Difesa statunitense come esperimento nelle comunicazioni digitali, crebbe fino a diventare un collegamento tra centinaia di università, esponenti della difesa e laboratori di ricerca. Permise a tutti i ricercatori, ovunque essi si trovassero, di scambiarsi informazioni con velocità e flessibilità senza precedenti, dando un forte impulso allo sviluppo del lavoro di collaborazione e accelerando enormemente il ritmo e l'intensità del progresso tecnologico.

Ma ARPAnet fece anche qualcos'altro. Le sue autostrade elettroniche misero in contatto gli hacker di tutti gli Stati Uniti e questi, finora isolati in sparuti gruppi, ognuno con la propria effimera cultura, si riscoprirono (o reinventarono) nelle vesti di vera a propria tribù di rete.

Le prime intenzionali azioni di hackeraggio - i primi linguaggi caratteristici, le prime satire, i primi dibattiti autocoscienti sull'etica hacker - tutto questo si propagò su ARPAnet nei suoi primi anni di vita. (Basti come esempio la prima versione del Jargon File, datato 1973.) La cultura hacker mosse i primi passi nelle università connesse alla Rete, in particolar modo (ma non esclusivamente) nei loro dipartimenti di scienza informatica.

Dal punto di vista culturale, l'AI (Intelligenza Artificiale) Lab del MIT è da considerarsi il primo tra laboratori di pari natura a partire dai tardi anni '60. Anche se istituti come il Laboratorio di Intelligenza Artificiale dell'Università di Stanford (SAIL) e, più tardi, l'Università Carnegie-Mellon (CMU), divennero in seguito quasi altrettanto importanti. Tutti costituivano fiorenti centri di scienza dell'informazione e ricerca sull'intelligenza artificiale. Tutti attiravano individui brillanti che contribuirono al grande sviluppo del mondo degli hacker, sia dal punto di vista tecnico che folkloristico.

Per comprendere ciò che successe dopo, comunque, è necessario un ulteriore sguardo ai computer stessi, poiché sia la nascita del Laboratorio che il suo futuro declino furono fortemente influenzati dalle correnti di cambiamento nell'ambito della tecnologia informatica.

Fin dai giorni del PDP-1, le sorti dell'hacking si intrecciarono alla serie di minicomputer PDP della Digital Equipment Corporation (DEC). La DEC aprì la strada a prodotti interattivi di stampo commerciale ed a sistemi operativi time-sharing. La flessibilità, la potenza e la relativa economicità di queste macchine, portarono molte università al loro

acquisto.

L'economicità dei sistemi time-sharing, costituì l'habitat ideale per lo sviluppo della cultura hacker e anche ARPAnet fu costituita, per la maggior parte della sua durata, da una rete di macchine DEC.

La più importante fra queste fu il PDP-10 che fece la sua comparsa nel 1967. Essa rappresentò la macchina preferita dagli hacker per quasi quindici anni; il TOPS-10 (sistema operativo DEC per la macchina) e il MACRO-10 (suo assemblatore), sono ancora ricordati con passione nostalgica nell'ambito della cultura hacker.

Il MIT, pur utilizzando lo stesso PDP-10, imboccò una strada lievemente diversa; rifiutò il software DEC del PDP-10 scegliendo di creare un proprio sistema operativo, il leggendario ITS.

ITS stava per "Incompatible Timesharing System", (Sistema Time-Sharing Incompatibile), sigla che rendeva perfettamente l'idea delle intenzioni insite nel progetto: volevano fare a modo loro. Fortunatamente per tutti noi, la gente della MIT possedevano un grado di intelligenza in grado di contrastare la sua arroganza. L'ITS, strambo, eccentrico e a volte perfino pieno di difetti, portò tuttavia una brillante serie di innovazioni tecniche, e ancora detiene senza dubbio il record di sistema operativo time-sharing più a lungo utilizzato.

Lo stesso ITS fu scritto in Assembler, ma molti progetti ITS furono scritti nel linguaggio LISP dell'AI. Il LISP si rivelò il più potente e flessibile linguaggio dell'epoca e, a distanza di vent'anni, si presenta ancora meglio congegnato rispetto a molti dei linguaggi odierni. Il LISP permise agli hacker di ITS di dare libero sfogo a tutta la loro creatività. Fu forse questa la formula del successo straordinario di questo linguaggio, che resta uno dei preferiti dagli hacker.

Molte creazioni tecniche della cultura ITS, sopravvivono ancora oggi; l'editor Emacs è forse il più conosciuto. Così come molto del folklore ITS è tuttora "vivo" per gli hacker, come dimostra il Jargon File.

Non si può certo dire che il SAIL e il CMU si fossero nel frattempo assopiti. Molti nuclei di hacker, sviluppatisi intorno al PDP-10 del SAIL, divennero più tardi figure chiave nel progresso dei personal computer e delle interfacce di software finestra/icona/mouse, come oggi le conosciamo. Gli hacker di CMU, dal canto loro, stavano portando avanti ciò che avrebbe dato vita alle prime applicazioni pratiche su larga scala di sistemi esperti e di robotica industriale.

Un altro luogo che ha giocato un ruolo fondamentale per il progresso culturale fu lo Xerox PARC, il famoso Centro Ricerche di Palo Alto. Per più di un decennio, a partire dai primi anni '70 fino alla metà degli '80, il PARC produsse un'impressionante quantità di innovazioni hardware e software. Le moderne interfacce di software costituite da mouse, finestre e icone, videro la luce proprio in quell'ambito, ma anche le stampanti laser e la local area network (LAN). La serie PARC di macchine D, anticipò di un decennio i potenti personal computer degli anni '80. Purtroppo, questi profeti non ebbero né onori né gloria in seno alla loro azienda e presto diventò un'abitudine descrivere sarcasticamente il PARC come un luogo caratterizzato dallo sviluppo di brillanti idee per chiunque altro, tranne che per se stessi. L'influenza di queste menti sulla cultura hacker fu comunque a dir poco pervasiva.

Le culture ARPAnet e PDP-10 crebbero in forza e varietà nell'arco degli anni '70. I programmi per le mailing list elettroniche, utilizzati fino ad allora per incoraggiare la cooperazione tra i diversi gruppi di interesse disseminati a quattro angoli del mondo, furono sempre più impiegati per scopi sociali e ricreativi. DARPA chiuse deliberatamente un occhio di fronte alle attività tecniche "non-autorizzate", ben comprendendo come queste spese extra fossero un piccolo prezzo da pagare rispetto all'effetto di convogliare l'attenzione di un'intera generazione di menti giovani e brillanti alla causa dell'informatica.

Probabilmente, la più nota delle mailing list a sfondo "sociale" di ARPAnet fu la SF-LOVERS, per gli appassionati di fantascienza; basti pensare che ancora oggi essa continua ad

esistere in "Internet", l'erede naturale e senza confini della rete ARPAnet. In questo scenario, si contano numerosi altri pionieri di questo stile di comunicazione che più tardi venne commercializzato in servizi time-sharing a pagamento come CompuServe, Genie e Prodigy.

La nascita di Unix

Nel frattempo, comunque, nel selvaggio New Jersey, qualcos'altro era stato messo in cantiere fin dal 1969, qualcosa che avrebbe inevitabilmente adombrato la tradizione del PDP-10. L'anno di nascita di ARPAnet, fu anche l'anno in cui un hacker dei Laboratori Bell, di nome Ken Thompson, inventò il sistema Unix.

Thompson si era trovato coinvolto nella fase di sviluppo di un Sistema Operativo Time-Sharing chiamato Multics, che divideva la propria discendenza con ITS. Multics costituì un importante banco di prova su come la complessità di un sistema operativo potesse essere celata fino a essere resa invisibile all'utente e perfino alla maggioranza dei programmatori. L'idea fu quella di rendere l'uso di Multics molto più semplice e programmabile in modo da permettere di operare anche dall'esterno.

I Laboratori Bell si tirarono fuori dal progetto quando Multics iniziò a mostrare segni di crescita non giustificata (il sistema fu poi commercializzato da Honeywell, senza successo). Ken Thompson cominciò ad avere nostalgia dell'ambiente Multics, e pensò di giocare un po' miscelando alcune caratteristiche del sistema operativo naufragato con altre di sua concezione su un rottame di DEC PDP-7.

Un altro hacker, di nome Dennis Ritchie, inventò un nuovo linguaggio chiamato "C", da usare con una versione Unix di Thompson ancora allo stato embrionale. Come Unix, C fu progettato per essere piacevole e facile da usare oltre che flessibile. L'interesse per questi strumenti non tardò a crescere nell'ambito dei Laboratori Bell, e subì un'impennata nel 1971 quando Thompson e Ritchie vinsero un appalto per produrre quello che oggi chiameremmo sistema di office-automation per uso interno. Ma Thompson e Ritchie avevano in mente qualcosa di ben più ambizioso.

Per tradizione, i sistemi operativi erano stati, fino ad allora, scritti in Assembler in modo da ottenere la maggiore efficienza possibile dalle macchine host. Thompson e Ritchie furono tra i primi a capire che la tecnologia dell'hardware e dei compilatori aveva raggiunto un tale livello di maturità da poter scrivere in C un intero sistema operativo: nel 1974 l'intero ambiente operativo era regolarmente installato su numerose macchine di diversa tipologia.

Si tratta di un evento senza precedenti e le implicazioni che ne derivarono furono enormi. Se davvero Unix poteva presentare la stessa interfaccia e le stesse funzionalità su macchine di diverso tipo, era sicuramente in grado di fungere da ambiente software comune per tutte. Gli utenti non avrebbero mai più dovuto pagare per nuovi software appositamente progettati ogni volta che una macchina diventava obsoleta. Gli hacker erano in grado di utilizzare gli stessi strumenti software da una macchina all'altra, piuttosto che dover reinventare l'equivalente di fuoco e ruota ogni volta.

Oltre alla portabilità, Unix e C presentavano altri punti di forza. Entrambi si basavano sulla filosofia "Keep it simple, stupid!" letteralmente "Semplifica, stupido!". Un programmatore poteva senza difficoltà tenere a mente l'intera struttura logica di C (a differenza di molti altri linguaggi precedenti, ma anche successivi), e non dover più ricorrere continuamente ai manuali. Unix era un insieme flessibile di semplici strumenti che si mostravano complementari l'un l'altro.

Questa combinazione si rivelò adatta per una vasta gamma di operazioni, incluse alcune completamente nuove, non previste in origine dagli stessi progettisti. La sua diffusione in AT&T fu estremamente rapida, a dispetto della mancanza di programmi di supporto formale. Entro il 1980, il suo uso si era già allargato a un gran numero di università e siti di ricerca informatica, e centinaia di hacker la consideravano come la propria casa.

Le macchine da lavoro della prima cultura Unix furono i PDP-11 e il loro discendente fu il VAX. Ma, proprio per la sua caratteristica portabilità, Unix funzionava senza alcuna modifica su una vasta gamma di macchine che costituivano ARPAnet. Nessuno usava l'Assembler, i programmi creati in C erano facilmente utilizzabili su tutte queste macchine.

Unix aveva persino una propria rete non certo di qualità eccelsa: Unix-to Unix Copy Protocol (UUCP), bassa velocità, poco affidabile ma economica. Due macchine Unix qualsiasi potevano scambiarsi posta elettronica point-to-point attraverso le ordinarie linee telefoniche. Questa funzionalità era parte integrante del sistema e non solo un'opzione. Le postazioni Unix cominciarono a formare una rete a se stante, e una cultura hacker iniziò a crescere al suo interno. È del 1980 la prima Usenet board, che sarebbe rapidamente diventata più grande di ARPAnet.

ARPAnet stessa ospitò alcuni siti Unix. PDP-10 e le culture Unix e cominciarono a incontrarsi e fondersi, anche se, dapprima, senza grande successo. Gli hacker di PDP-10 consideravano la gente di Unix come

una banda di principianti che utilizzava strumenti dall'aspetto primitivo, se paragonati alla squisita e perfino barocca complessità di LISP e ITS. "Coltelli di pietra e pelli d'orso!" brontolavano.

Ecco allora che si delineò un terzo scenario. Il primo personal computer fu immesso sul mercato nel 1975. La Apple fu fondata nel 1977, e il suo progresso avvenne con impressionante rapidità negli anni che

seguirono. Il potenziale dei microcomputer era ormai chiaro e attrasse inevitabilmente un'altra generazione di giovani e brillanti hacker. Il loro linguaggio era il BASIC, talmente primitivo che i partigiani del PDP-10, e gli aficionados di Unix lo considerarono subito indegno di qualsiasi considerazione.

La fine del tempo che fu

Ecco la situazione nel 1980: tre culture, simili ma organizzate intorno a diverse tecnologie. La cultura ARPAnet/PDP-10 sposata a LISP, MACRO, TOPS-10 e ITS. Il popolo di Unix e C, con i loro PDP-11, VAX e connessioni telefoniche di modesta entità. E infine un'orda anarchica di appassionati dei primi microcomputer, decisi a portare al popolo la potenza del computer.

Tra tutte queste, la cultura ITS poteva ancora rivendicare il posto d'onore. Ma sul Laboratorio si stavano addensando nubi minacciose. La tecnologia PDP-10 dipendente da ITS, cominciava a essere datata e il

Laboratorio stesso era diviso in fazioni fin dai primi tentativi di commercializzazione della tecnologia AI. Alcune delle migliori menti del Laboratorio (e di SAIL e CMU), si erano lasciate attirare da lavori molto ben retribuiti presso società di nuova costituzione.

Il colpo di grazia fu inferto nel 1983, quando DEC cancellò la sua adesione al PDP-10 per concentrarsi sulle linee VAX e PDP-11. ITS non aveva più un futuro. In virtù della sua scarsa portabilità, infatti, l'idea di trasportarlo da un hardware all'altro era impensabile per chiunque. La variante funzionante su Unix di Berkeley VAX, divenne il sistema prediletto dagli hacker, e chiunque avesse rivolto lo sguardo al futuro, si sarebbe reso conto di quanto rapidamente crescesse la potenza dei microcomputer e con quale velocità avrebbero spazzato via tutto quello che li aveva preceduti.

Fu all'incirca in questo periodo che Levy scrisse Hackers. Una delle sue principali fonti di informazione fu Richard M. Stallman (inventore di Emacs), una figura chiave del Laboratorio e accanito oppositore della commercializzazione della tecnologia del Laboratorio.

Stallman (meglio conosciuto con le sue iniziali e login name, RMS), creò la Free Software Foundation, dedicandosi alla produzione di free software di alta qualità. Levy lo elogiò quale "ultimo vero hacker", una descrizione che si rivelò fortunatamente errata.

Il grandioso progetto di Stallman riassunse chiaramente la transizione che subì la cultura degli hacker nei primi anni '80: nel 1982 egli iniziò la costruzione di un intero clone di Unix, scritto in C e disponibile gratuitamente. Si può quindi dire che lo spirito e la tradizione di ITS furono preservati come parte importante della più nuova cultura hacker, incentrata su Unix e VAX.

Sempre in quello stesso periodo, la tecnologia dei microchip e della local area network iniziarono a fare presa sul mondo degli hacker. Ethernet e il microchip Motorola 68000 costituirono una combinazione teoricamente molto potente e solo dopo numerosi tentativi si arrivò alla prima generazione di ciò che oggi conosciamo come workstation.

Nel 1982, un gruppo di hacker Unix di Berkeley, fondò Sun Microsystems con la convinzione che Unix funzionante su un hardware con base 68000, relativamente economico, sarebbe stata la combinazione vincente per una grande varietà di applicazioni. La previsione si rivelò esatta e la loro intuizione definì il modello che l'intera industria avrebbe seguito. Sebbene i loro prezzi non erano ancora alla portata della maggior parte degli utenti, le workstation erano relativamente economiche per università e grandi aziende. Reti formate da questa nuova generazione di computer (uno per utente), sostituirono rapidamente gli ormai sorpassati VAX e altri sistemi time-sharing.

L'era del free Unix

Quando nel 1984 la AT&T iniziò ad essere svenduta e Unix divenne per la prima volta un prodotto commerciale, il mondo degli hacker si divideva in una "network nation", relativamente coesiva e centrata su Internet e Usenet, in cui venivano per lo più usati minicomputer o workstation funzionanti con Unix, e una vasta ma disorganizzata "hinterland" di appassionati di microcomputer.

La classe di macchine workstation costruite da Sun e da altri, aprì nuovi orizzonti agli hacker. Queste erano concepite per realizzare grafica di livello professionale e trasferire e gestire dati condivisi attraverso una rete. Nel corso degli anni '80, il mondo degli hacker si mostrò attento alle sfide di software e strumenti per sfruttare al massimo queste caratteristiche. Il gruppo Unix di Berkeley sviluppò un supporto integrato per i protocolli ARPAnet che offriva una soluzione al problema delle reti favorendo un'ulteriore crescita di Internet.

Numerosi furono i tentativi di semplificare l'uso degli strumenti di grafica delle workstation. Il sistema che prevalse fu l'X Window System. Uno dei fattori che determinarono il suo successo fu dato dalla disponibilità dei suoi sviluppatori a fornire gratuitamente i sorgenti, secondo l'etica hacker, e a distribuirli tramite Internet.

La vittoria di X sui sistemi di grafica proprietari (incluso quello offerto dalla stessa Sun), fu un'importante messaggio di cambiamento che, pochi anni dopo, avrebbe profondamente influenzato lo stesso Unix.

La rivalità tra ITS e Unix generava ancora qualche occasionale manifestazione di collera faziosa (per lo più proveniente dalla parte dei sostenitori dell'ex-ITS). L'ultima macchina ITS cessò comunque di funzionare per sempre nel 1990. I suoi partigiani si ritrovarono senza più un posto dove stare e furono in larga parte assimilati dalla cultura Unix non senza lamentele.

Nell'ambito degli hacker della rete, la grande rivalità negli anni '80 era tra i sostenitori della versione Unix di Berkeley e quella di AT&T. Sono ancora oggi reperibili copie di un manifesto di quel periodo che riportava un combattente, in stile cartoon, con ali a forma di X, preso in prestito dal film Guerre Stellari, in fuga da una Death Star (stella morta) in esplosione contrassegnata dal logo AT&T. Gli hacker di Berkeley amavano vedersi come i ribelli contro i crudeli imperi aziendali. La versione Unix di AT&T non riuscì mai a competere sul mercato con il concorrente BDS/Sun, sebbene si aggiudicò la guerra degli standard. Nel 1990, le versioni AT&T e BSD divennero difficili da distinguere avendo l'una adottato molte innovazioni dell'altra e viceversa.

Agli inizi degli anni '90, la tecnologia delle workstation del decennio precedente cominciava a vedersi seriamente minacciata da nuovi personal computer, a basso costo e dalle alte prestazioni, basati sul chip Intel 386 e i suoi discendenti.

Per la prima volta, ogni singolo hacker poteva finalmente permettersi di disporre anche a casa di macchine paragonabili, per potenza e capacità di memoria, ai minicomputer di un decennio prima, macchine Unix in grado di supportare un ambiente di sviluppo completo e di comunicare con Internet.

In questo nuovo scenario, il mondo MS-DOS rimase beatamente allo scuro degli sviluppi in corso. Nonostante le fila degli appassionati di microcomputer della prima ora si ingrandirono rapidamente fino a diventare una popolazione di hacker DOS e Mac di dimensioni ancora maggiori rispetto alla cultura "network nation", essi non riuscirono mai a sviluppare una cultura consapevole. Il ritmo dei cambiamenti era talmente veloce che ben cinquanta diverse culture tecniche nacquero e cessarono di esistere con la rapidità di una farfalla, senza mai raggiungere la stabilità necessaria allo sviluppo di un gergo, di un folklore e di una storia propri. L'assenza di una rete realmente pervasiva, paragonabile a UUCP o a Internet, non permise loro di diventare una network nation. Il crescere degli accessi a servizi commerciali online, come CompuServe e Genie, ma parallelamente la non diffusione in bundle di strumenti di sviluppo per sistemi operativi non-Unix, significava poco materiale su cui lavorare. Questa situazione impedì lo svilupparsi di una tradizione di collaborazione tra gli hacker.

La corrente hacker più importante, (dis)organizzata intorno a Internet, e finora largamente identificata con la cultura tecnica di Unix, non era interessata ai servizi commerciali. I suoi adepti volevano solo strumenti migliori e più Internet, cose che l'economico PC a 32-bit promise di mettere alla portata di tutti.

Ma dov'era il software? Le macchine Unix commerciali restavano comunque costose. Nei primi anni '90, numerose società fecero una prova vendendo porting di Unix BDS o AT&T per macchine PC. Il successo si rivelò elusivo, i prezzi non erano scesi di molto e (ancora peggio) non si ottenevano sorgenti modificabili e ridistribuibili per il proprio sistema operativo. Il tradizionale modello di software-business non stava affatto fornendo agli hacker ciò che volevano.

Neanche con la Free Software Foundation la situazione migliorò. Lo sviluppo di HURD, il tanto sospirato kernel Unix gratuito per hacker promesso da RMS, rimase fermo per anni e non riuscì a produrre alcunché di utilizzabile fino al 1996 (sebbene dal 1990 la FSF avesse fornito quasi tutti gli altri complicati componenti di un sistema operativo simile a Unix).

Ciò che dava davvero motivo di preoccupazione era che, con l'inizio degli anni '90, si cominciava a vedere con chiarezza come dieci anni di tentativi di commercializzare Unix stessero dopotutto fallendo. La promessa di Unix, di rendere portabili le cross-platform si perse tra mezza dozzina di versioni proprietarie di Unix. I detentori di Unix proprietario diedero prova di tanta lentezza e inettitudine nel campo del marketing, che Microsoft fu in grado di inglobare la maggior parte della loro fetta di mercato con la tecnologia del sistema operativo Windows, incredibilmente inferiore a quella Unix.

Nei primi mesi del 1993, qualsiasi osservatore pessimista avrebbe avuto tutti i motivi per decretare l'imminente fine della storia di Unix e della fortuna della sua tribù di hacker, cosa tra l'altro predetta sin dai tardi anni '70 a intervalli regolari di 6 mesi.

In quei giorni, era pensiero comune la fine dell'era del tecno-eroismo individuale e che l'industria del software e la nascente Internet sarebbero state dominate da colossi come Microsoft. La prima generazione di hacker Unix sembrava invecchiata e stanca (il gruppo di Ricerca della Scienza Informatica di Berkeley chiuse i battenti nel 1994). Il periodo non era tra i più felici.

Fortunatamente, ci furono cose che sfuggirono all'attenzione della stampa specializzata e perfino alla maggior parte degli hacker, cose che avrebbero prodotto sviluppi positivi verso la fine del 1993 e l'inizio del 1994.

In futuro, questa situazione avrebbe portato la cultura a imboccare una strada completamente nuova, disseminata di insperati successi.

I primi free Unix

Dal gap provocato dal fallimento dell'HURD, era emerso uno studente dell'Università di Helsinki di nome Linus Torvalds. Nel 1991, cominciò a sviluppare un kernel free Unix per macchine 386 usando un kit di strumenti della Free Software Foundation. Il suo rapido successo nella fase inizale, attrasse molti hacker di Internet, volonterosi di aiutarlo nello

sviluppo del suo Linux, una versione Unix con sorgenti interamente free e redistribuibili.

Anche Linux aveva i suoi concorrenti. Nel 1991, contemporaneamente ai primi esperimenti di Linus Torvald, William e Lynne Jolitz stavano sperimentando il porting di Unix BSD sul 386. La maggior parte di coloro che paragonavano la tecnologia BSD agli sforzi iniziali di Linus, si aspettavano che i porting di BSD diventassero i più importanti free Unix su PC.

***Linus Torvalds** - fonte Wikipedia*

La caratteristica fondamentale di Linux, tuttavia, non era tanto tecnica quanto sociologica. Fino allo sviluppo di Linux, era pensiero comune che qualsiasi software complicato come un sistema operativo, dovesse essere sviluppato in modo attentamente coordinato da un ristretto gruppo di persone ben collegate tra di loro. Questo modo di operare era, ed è tuttora, tipico sia del software commerciale che delle grosse cattedrali di freeware costruiti dalla Free Software Foundation negli anni ’80; così come dei progetti freeBSD/netBSD/OpenBSD, che allargarono il campo di applicazione del porting originale 386BSD dei Jolitz.

Linux si evolse in modo completamente differente. Fin quasi dalla sua nascita, fu casualmente "preda di hacking" da parte di un vasto numero di volontari collegati solo tramite Internet. La qualità fu mantenuta non da rigidi standard o autocrazia, ma dalla strategia semplice e naive di proporre settimanalmente delle idee e di ricevere opinioni in merito da centinaia di utenti ogni giorno, creando una sorta di rapida selezione darwiniana sulle modifiche introdotte dagli sviluppatori. Con stupore da parte di quasi tutti, il progetto funzionava piuttosto bene.

Verso la fine del 1993, Linux fu in grado di competere per stabilità e affidabilità, con molti Unix commerciali, ospitando una grande quantità di software. Esso stava perfino cominciando ad attirare il porting di

applicazioni software commerciali. Un effetto indiretto di questo sviluppo, fu lo spazzare via la maggior parte dei piccoli fornitori di Unix commerciali - la loro caduta fu anche determinata dalla mancanza di hacker e potenziali utenti ai quali vendere. Uno dei pochi sopravvissuti, BSDI (Berkeley System Design, Incorporated), fiorì offrendo sorgenti completi, con il suo Unix base BSD, e coltivando stretti legami con la comunità hacker.

All’epoca tali sviluppi non furono pienamente rilevati dalle comunità hacker e non lo furono affatto al di fuori di essa. La tradizione hacker, a dispetto delle ripetute predizioni su una sua imminente fine, stava proprio iniziando a riorganizzare il mondo del software commerciale a propria immagine. Trascorsero ancora cinque anni prima che questa tendenza iniziasse a palesarsi.

La grande esplosione del Web

L’iniziale crescita di Linux coincise con un altro fenomeno: la scoperta di Internet da parte del grande pubblico. I primi anni ’90 videro l’inizio di una fiorente industria dell’Internet provider, che forniva connessioni al pubblico per pochi dollari al mese. Dopo l’invenzione del World Wide Web, la già rapida crescita di Internet accelerò a rotta di collo.

Nel 1994, anno in cui il gruppo di sviluppo Unix di Berkeley chiuse ufficialmente i battenti, molte diverse versioni di free Unix (Linux e i discendenti del 386BSD) catalizzarono

l’interesse degli hacker. Linux era distribuito su CD-ROM, e andava via come il pane. Alla fine del 1995, le maggiori aziende informatiche cominciarono a promuovere i propri hardware e software giocando la carta della loro grande compatibilità con Internet!

Nella seconda metà degli anni ’90, l’attività degli hacker si incentrò sullo sviluppo di Linux e sulla diffusione di massa di Internet. Il World Wide Web era riuscito a trasformare Internet in un mezzo di comunicazione di massa, e molti hacker degli anni ’80 e ’90, intrapresero l’attività di Internet Service Provider fornendo accesso a questo nuovo mondo.

La diffusione di massa di Internet, aveva perfino portato la cultura hacker ad essere rispettata in quanto tale. Nel 1994 e 1995, l’attivismo hacker fece naufragare la proposta Clipper che avrebbe posto sotto il controllo del governo un metodo di codifica. Nel 1996, gli hacker si mobilitarono per sconfiggere il "Communications Decency Act" (CDA), e scongiurare il pericolo di censura su Internet.

L'etica hacker

L'accesso ai computer e a tutto ciò che potrebbe insegnare qualcosa su come funziona il mondo dev'essere assolutamente illimitato e completo.

L'etica hacker
Steven Levy - Hacker

Qualcosa di nuovo stava nascendo intorno al Tx-0: un nuovo stile di vita con una filosofia, un'etica e un sogno.

Non ci fu un momento preciso in cui gli hacker del Tx-0 intravidero che votando le loro abilità tecniche nell'informatica, insieme a una dedizione raramente riscontrata al di fuori dei monasteri, sarebbero divenuti l'avanguardia di un'audace simbiosi tra uomo e macchina. Avevano lo stesso fervore di quei meccanici flippati che truccano i loro *chopper* e che danno per scontato che il loro mondo sia l'unico. Mentre si andavano formando gli elementi di una cultura, cominciavano ad accumularsi leggende, e la loro padronanza delle tecniche di programmazione arrivava a superare ogni soglia di abilità fin lì raggiunta. Comunque, questo gruppetto di hacker era restio ad ammettere che la loro piccola comunità, in stretta connessione con il Tx-0, avesse lentamente e inavvertitamente costruito un corpo organico di concetti, convinzioni e costumi.

I precetti di questa rivoluzionaria *etica hacker* non erano tanto oggetto di dibattito e discussione, ma erano piuttosto tacitamente accettati: non erano stati scritti manifesti e non c'erano missionari che cercassero di fare adepti. Il computer stesso operava le conversioni, e quelli che parevano seguire più fedelmente l'etica hacker erano persone come Samson, Saunders e Kotok, la cui vita prima del MIT sembrava essere solo il preludio al momento in cui si sarebbero realizzati dietro la consolle del Tx-0. In seguito ci sarebbero stati hacker che interpretarono l'etica ancor più alla lettera degli stessi hacker del Tx-0, gente come il leggendario Greenblatt oppure Gosper, benché ci sarebbero voluti anni prima che i principi dell'hackeraggio venissero formulati esplicitamente.

Tuttavia, già dai giorni del Tx-0, le fondamenta dell'edificio erano state gettate. E questa era l'etica hacker:

L'accesso ai computer - e a tutto ciò che potrebbe insegnare qualcosa su come funziona il mondo - dev'essere assolutamente illimitato e completo. Dare sempre precedenza all'imperativo di metterci su le mani!

Gli hacker credono nella possibilità d'imparare lezioni essenziali sui sistemi e sul mondo smontando le cose, osservando come funzionano, e usando questa conoscenza per creare cose nuove, ancor più interessanti. Detestano qualsiasi persona, barriera fisica o legge che tenti d'impedirglielo.

Questo è vero soprattutto quando un hacker vuole aggiustare qualcosa che (dal suo punto di vista) è guasta o necessita di miglioramento. I sistemi imperfetti fanno infuriare gli hacker, il cui istinto primario è di correggerli. Questa è la ragione per cui in genere gli hacker odiano guidare le auto: il sistema dei semafori collocati a casaccio unitamente all'inspiegabile dislocazione delle strade a senso unico provocano ritardi così *inutili* che l'impulso è quello di riordinare la segnaletica, aprire le centraline di controllo e riprogettare l'intero sistema.

Nel mondo degli hacker chiunque s'incazzasse abbastanza da aprire una centralina dei semafori e la smontasse per farla funzionare meglio, sarebbe più che incoraggiato a tentare. Le regole che ci impediscono di prendere in mano questioni del genere sono ritenute troppo ridicole per dovervisi conformare. Questo modo di pensare spinse il Model railroad club a istituire, su una base estremamente informale, un'organismo chiamato *Midnight requisitomig committee,* il Comitato per la requisizione di mezzanotte. Quando il TMRC aveva bisogno di procurarsi un certo quantitativo di diodi, o qualche relè in più, per implementare qualche nuova funzionalità nel "sistema", un manipolo di membri dell'S&P attendeva l'oscurità e s'intrufolava nei luoghi in cui queste cose erano reperibili. Nessuno degli hacker, che erano persone assolutamente scrupolose e oneste in altre occasioni, sembrava ritenerlo un "furto". Un'intenzionale cecità!

Tutta l'informazione dev'essere libera

Se non avete accesso alle informazioni di cui avete bisogno per migliorare le cose, come farete? Un libero scambio di informazioni, soprattutto quando l'informazione ha l'aspetto di un programma per computer, promuove una maggiore creatività complessiva. Per una macchina come il Tx-0, arrivato quasi senza software, e liiunque si sarebbe forsennatamente messo a scrivere programmi di sistema per facilitarne la programmazione, *strumenti per fare strumenti,* riponendoli in un cassetto della consolle a portata di mano di chiunque volesse usare la macchina. Que-sio comportamento evita la temuta e rituale perdita di tempo per reinventare la mota; invece di stare tutti a scrivere la propria versione dello stesso programma, la migliore dovrebbe essere disponibile per chiunque, e ognuno dovrebbe essere libero di dedicarsi allo studio del codice e perfezionare proprio *quello.* Sarebbe un inondo ingioiellato di programmi completi di ogni caratteristica, che non danno problemi, corretti fino alla perfezione.

La convinzione, talora accettata acriticamente, che l'informazione dovrebbe essere libera era un conseguente tributo al modo in cui un ottimo computer o un valido programma lavorano: i bit binari si muovono lungo il percorso più logico, diretto e necessario a svolgere il loro complesso compito. Cos'è un computer se non e qualcosa che beneficia di un libero flusso di informazione? Se, per esempio, l'accumulatore si trova impossibilitato a ricevere informazioni dai dispositivi di input/output (i/o) come il lettore del nastro o dagli interruttori, l'intero sistema collasserebbe. Dal punto di

vista degli hacker, qualsiasi sistema trae beneficio da un libero flusso d'informazione.

Dubitare dell'autorità. Promuovere il decentramento

Il modo migliore per promuovere il libero scambio delle informazioni è avere sistemi aperti, qualcosa che non crei barriere tra un hacker e un'informazione, o un dispositivo di cui egli possa servirsi nella sua ricerca di conoscenza. L'ultima cosa di cui c'è bisogno è la burocrazia. Questa, che sia industriale, governativa o universitaria è un sistema imperfetto, ed è pericolosa perché è inconciliabile con lo spirito di ricerca dei veri hacker. I burocrati si nascondono dietro regole arbitrarie (agli antipodi degli algoritmi logici con cui operano le macchine e i programmi): si appellano a quelle norme per rafforzare il proprio potere e percepiscono l'impulso costruttivo degli hacker come una minaccia.

Il simbolo dell'universo burocratico è incarnato da quell'enorme società chiamata International business machine: l'IBM (tenete ben presente che i fatti si stanno svolgendo negli anni Cinquanta e Sessanta). La ragione per cui i suoi computer, i "bestioni", si basassero sull'elaborazione batch era soltanto parzialmente riconducibile alla tecnologia delle valvole elettroniche. La ragione vera stava nel fatto che l'IBM era una società goffa e mastodontica, che non aveva compreso la carica innovativa dell'hackeraggio. Se l'IBM avesse avuto mano libera (questa cosa la pensavano molti hacker del TMRC), il mondo sarebbe diventato una macchina a elaborazione batch, basata su quelle noiose schede perforate, e soltanto ai più privilegiati sacerdoti sarebbe stato concesso di interagire effettivamente col computer.

Bastava vedere qualche essere del mondo IBM, osservare il suo camice bianco ben abbottonato, l'impeccabile cravatta nera, i capelli dalla scriminatura ben curata, e il vassoio di schede perforate in mano. Nel centro di calcolo, dov'erano alloggiati il 704, il 709 e più tardi anche il 7090 - il meglio che l'IBM avesse allora da offrire - e fino alle aree riservate, oltre le quali era proibito l'accesso al personale non autorizzato, regnava un ordine soffocante. Niente a che vedere con l'atmosfera estremamente informale che circolava nell'ambiente del Tx-0, dove gli abiti consunti erano la norma e pressoché chiunque poteva entrare.

In realtà, l'IBM aveva fatto e avrebbe continuato a fare molto per l'avanzamento dell'informatica. La sua stessa dimensione e potente influenza hanno fatto dei computer un elemento permanente dello stile di vita americano. Per molta gente le parole IBM e computer sono praticamente sinonimi. Le macchine IBM erano veri e propri cavalli da tiro, degne della fiducia che gli uomini d'affari e gli scienziati avevano accordato loro. Ciò era dovuto in parte all'approccio conservatore dell'IBM: non costruiva le apparecchiature più avanzate tecnologicamente, ma faceva affidamento su concetti sperimentati e un marketing attento e aggressivo. Non appena il dominio dell'IBM s'impose nel campo dei computer, la società divenne un impero avvitato su se stesso, reticente e pieno di sé.

Ciò che faceva letteralmente impazzire gli hacker era l'atteggiamento dei sacerdoti e portaborse dell'IBM, i quali credevano veramente che soltanto l'IBM avesse "veri" computer e il resto fosse tutta spazzatura. Non era possibile parlare con quella gente: era impossibile convincerli. Loro stessi erano gente "a elaborazione batch", e lo dimostravano non soltanto con le loro preferenze in fatto di macchine ma per il concetto stesso che avevano di come mandare avanti un centro di calcolo... e il mondo. Questa gente non avrebbe mai potuto capire la naturale superiorità di un sistema decentrato, nel quale nessuno impartisse degli ordini: un sistema in cui le persone potessero seguire i propri interessi, per cui se lungo la strada avessero scoperto un punto debole nel sistema, avrebbero potuto imbarcarsi in un ambizioso intervento chirurgico. Non era necessario avere dei permessi: bastava la volontà di intervenire.

L'inclinazione antiburocratica coincideva perfettamente con la personalità di gran parte degli hacker che, sin dall'infanzia, erano stati abituati a costruire progetti scientifici, mentre i loro compagni di classe si scornavano tra loro e apprendeva-vo le astuzie dei rapporti sociali tramite lo sport. Questi giovanotti, un tempo dei tagliati fuori, avevano ora trovato nel computer un formidabile mezzo riequilibratore, provando la sensazione, come dice Peter Samson, "d'aver aperto la porta e di cominciare a camminare in quest'immenso e nuovo universo..." Una volta attraversata quella soglia e sedutisi dietro la consolle di un computer da un milione di dollari, gli hacker avevano il potere. Così diventava naturale dubitare di qualsiasi forza che potesse cercare di limitare la misura di quel potere.

Gli hacker dovranno essere giudicati per il loro operato, e non sulla base di falsi criteri quali ceto, età, razza o posizione sociale

L'immediata accoglienza del dodicenne Peter Deutsch nella comunità degli hacker (sebbene non da parte dei laureati non hacker) ne è stato un buon esempio, l'arimenti, gente che poteva esibire credenziali particolarmente appariscenti non veniva presa sul serio prima di aver dato prova di sé dietro la consolle di un computer.

Questo atteggiamento meritocratico non nasceva necessariamente dall'innata bontà di cuore degli hacker; derivava invece dal fatto che gli hacker si curavano meno delle caratteristiche superficiali di ciascuno, e prestavano più attenzione al potenziale dell'individuo di far progredire lo stato generale dell'hackeraggio, nel creare programmi innovativi degni d'ammirazione e nella capacità di contribuire a descrivere le nuove funzioni del sistema.

Con un computer puoi creare arte

Il programma per musica di Samson ne era un esempio, ma per gli hacker l'artisticità del programma non stava nei suoni piacevoli che uscivano dall'altoparlante. Il codice del programma possedeva una bellezza propria. (Samson, in effetti, era stato piuttosto misterioso rifiutandosi di aggiungere commenti al suo codice originario, spiegando cosa questo stesse facendo in un determinato momento. Un famoso programma scritto da Samson conteneva centinaia di istruzioni in linguaggio assembly con un solo commento accanto a un'istruzione che conteneva il numero 1750. Il commento era Ripjsb, e la gente naturalmente s'era lambiccata il cervello intorno ai suoi possibili significati, prima che qualcuno si rendesse conto che il 1750 era l'anno di morte di Bach, e che Samson aveva scritto un acronimo per *Requiescat in pacem Johann Sehastian Bach.)*

Stava emergendo una certa estetica dello stile di programmazione. A causa dello spazio limitato sulla memoria del Tx-0 (un handicap comune a tutti i computer di quell'epoca), gli hacker apprezzavano enormemente quelle tecniche innovative che permettessero ai programmi di fare operazioni complicate con pochissime istruzioni. Più corto era il programma, più spazio c'era a disposizione per altri programmi, e più velocemente girava il tutto. Talvolta, quando non c'erano problemi di velocità o di spazio, e non c'erano particolari necessità artistiche ed estetiche, si buttava giù un programma orribile, che aggredisse il problema con "forza bruta". "Bene, potremmo farcela aggiungendo venti righe," sembrava dirsi tra sé e sé Samson, "e potrebbe essere più veloce scrivere le istruzioni che escogitare un ciclo di iterazione, all'inizio e alla fine, per fare lo stesso lavoro con solo sette o otto istruzioni." Però, quest'ultimo programma sarebbe stato maggiormente ammirato dai compagni hacker, e alcuni programmi erano stati ridotti a così poche righe con una tale abilità che i pari dell'autore, prendendone atto, quasi si scioglievano in reverenziale rispetto.

Certe volte ridurre un programma diventava una gara, una competizione accanita per dimostrarsi padroni del sistema, tanto da far intravedere eleganti scorciatoie per eliminare un'istruzione o due o, meglio ancora, ripensare l'intero problema e approntare un nuovo algoritmo che facesse risparmiare un intero blocco di istruzioni (un algoritmo è una particolare procedura applicabile alla soluzione di un problema complesso con il computer; è una specie di passe-partout matematico). Il tutto poteva essere fatto più vigorosamente accostandosi al problema da un'angolazione insolita, alla quale nessuno avesse mai pensato prima, ma che poi sarebbe risultata logica.

C'era senz'altro un impulso artistico in quelli che potevano utilizzare questa tecnica da genio marziano: una magia nera, una qualità visionaria che li rendeva capaci di scartare gli stantii schemi dei migliori cervelli del pianeta e concepire un nuovo algoritmo totalmente inaspettato.

E quanto accadde con il programma di conversione da binario a decimale. Si trattava di una subroutine - cioè di un programma contenuto in un altro programma che poteva essere richiamato, quando necessario, da dentro molti programmi differenti - per tradurre i numeri binari elaborati dal computer in normali numeri del sistema decimale. Citando le parole di Saunders, questo problema divenne "la prova del nove del programmatore: se eri in grado di scrivere un programma di conversione al sistema decimale che funzionasse, sapevi abbastanza di computer per poterti definire, a buon diritto, un programmatore". E chi avesse scritto un *grande* programma di conversione dei numeri al sistema decimale, avrebbe potuto fregiarsi del titolo di hacker. Più che una competizione, la messa a punto definitiva di un programma di conversione al sistema decimale con il minor numero possibile di istruzioni divenne una sorta di sacro Graal degli hacker.

Da mesi circolavano varie versioni di programmi di conversione al sistema decimale. Se qualcuno fosse stato del tutto a digiuno dell'argomento o totalmente ottuso - un vero e proprio *"laser",* un "perdente"* - avrebbe potuto usare un centinaio d'istruzioni per far convertire al computer il linguaggio macchina in decimale. Ma qualsiasi hacker di un qualche valore ne avrebbe usate meno e, alla fine, adoperando il meglio dai programmi, ottimizzando un passaggio qui e uno là, il programma si sarebbe ridotto a circa cinquanta istruzioni.

Dopo, le cose si fecero serie: c'era gente che lavorava per ore, cercando una strada per fare la stessa cosa con il numero minore possibile di righe di codice. Divenne qualcosa di più di una competizione: era ricerca. Per quanti sforzi fossero prodotti, nessuno sembrava in grado di abbattere la barriera delle cinquanta righe. Qualcuno pose la domanda se fosse veramente possibile farlo. C'era una soglia oltre la quale non si poteva accorciare un programma?

Tra quelli che si arrovellavano con questo dilemma c'era un tipo di nome Jensen, un hacker del Maine, alto e taciturno che sedeva quietamente nella "kludge room" e scribacchiava sui tabulati con il contegno sereno di uno che scortecci un bastone col coltello. Jensen era sempre in cerca di sistemi per comprimere tempo e spazio nei suoi programmi: il suo codice era una sequenza completamente bizzarra di funzioni booleane e aritmetiche mischiate tra di loro, e spesso usava computazioni differenti su varie sezioni della stessa "parola" di diciotto bit. Cose straordinarie, acrobazie magiche.

Prima di Jensen, il parere comune era stato che l'unico algoritmo logico per un programma di conversione al sistema decimale avrebbe voluto far fare alla macchina ripetute sottrazioni, usando una tabella delle potenze di dieci per mantenere i numeri nelle giuste colonne decimali. In qualche modo Jensen capì che una tabella delle potenze di dieci non era necessaria; e se ne uscì con un algoritmo capace di convertire i numeri in ordine inverso, ma, con un qualche giochetto di prestigio digitale, li buttava fuori nell'ordine giusto. C'era una complessa spiegazione matematica della cosa, che fu chiara agli altri hacker soltanto quando videro il programma di Jensen appiccicato su una bacheca, il suo modo per dir loro che aveva valicato il limite del programma di conversione al sistema decimale: *quarantasei istruzioni.* Fissarono tutti il codice a bocca aperta. Marge Saunders ricorda che gli hacker furono stranamente calmi nei giorni successivi.

"Avevamo capito che l'argomento era chiuso," disse più tardi Bob Saunders, "quello era il Nirvana".

I computer possono cambiare la vita in meglio

Questo principio era solo sottilmente manifesto. Difficilmente un hacker avrebbe dato a un estraneo una panoramica della miriade di vantaggi forniti dai computer nell'aprire le vie della conoscenza.

Eppure questa premessa dominava il comportamento quotidiano degli hacker del Tx-0, e quello delle generazioni di hacker venute dopo la loro.

Sicuramente il computer aveva cambiato le *loro* vite, le aveva arricchite, aveva dato loro un perno su cui ruotare, le aveva rese avventurose. Li aveva resi padroni di una porzione del loro destino. Peter Samson dice oggi: "L'abbiamo fatto dal 25 al 30 percento per l'amore di farlo, perché era qualcosa che potevamo fare e fare bene, e al 60 percento per amore di avere qualcosa che fosse a suo modo vivo, figlio nostro, che avrebbe cominciato a camminare da solo quando noi fossimo scomparsi. E questo che rende grande la programmazione, il fascino magico che possiede... Una volta risolto un problema di errato comportamento del computer o del programma, l'hai risolto per sempre, e tutto ti appare esattamente come lo avevi immaginato".

Come la lampada dì Aladino, potevi farle realizzare i tuoi desideri.

Sicuramente tutti avrebbero potuto trarre beneficio sperimentando questo potere. Sicuramente tutti avrebbero potuto trarre beneficio da un mondo basato sull'elica hacker. Questa era l'intrinseca convinzione degli hacker che hanno irriverentemente dilatato il punto di vista convenzionale su quello che i computer avrebbero potuto e dovuto fare: guidando il mondo verso un modo nuovo di considerare e interagire con i computer.

Non fu facile. Perfino in un'istituzione così avanzata come il MIT, alcuni professori consideravano frivola e persine demenziale una spiccata attrazione per i computer. Bob Wagner, un hacker del TMRC, una volta si trovò a dover spiegare a un professore d'ingegneria *cosa fosse un* computer. Bob Wagner sperimentò questo scontro tra procomputer e anticomputer ancor più vivamente quando si trovò in un corso di analisi numerica nel quale il professore richiedeva a ogni studente di eseguire i compiti utilizzando calcolatrici elettromeccaniche che solo dal rumore sembrano lerrovecchi. Kotok era nello stesso corso, ed entrambi erano sgomenti alla prospettiva di lavorare con quelle macchine obsolete. "Perché dovremmo," chiesero, "se abbiamo questo computer?"

Così Wagner cominciò a lavorare su un programma che avrebbe emulato il comportamento di un calcolatore. L'idea era scandalosa e per qualcuno era un'indebita appropriazione di prezioso tempo-macchina. Secondo la mentalità di quel periodo a proposito di computer, quel tempo era così prezioso che si potevano solo tentare imprese che usassero al meglio la macchina, imprese che avrebbero richiesto una quantità enorme di matematici impegnati per giorni in calcoli noiosi e ripetitivi. Gli hacker la pensavano diversamente: qualsiasi cosa sembrasse interessante o divertente era pane per l'informatica e potevi agire in base a quel principio, usando computer interattivi, senza che qualcuno ti spiasse da dietro le spalle e ti chiedesse di rendere conto del tuo progetto specifico. Dopo due o tre mesi di lotte nell'intricato groviglio dell'aritmetica a virgola mobile (necessaria per consentire al programma di sapere dove collocare la virgola decimale) su una macchina che non era dotata di un metodo semplice per eseguire moltiplicazioni elementari, Wagner aveva scritto tremila righe di codice che eseguivano il lavoro. Aveva fatto svolgere a un computer incredibilmente costoso la prestazione di un calcolatore che costava mille volte di meno. In onore di questo paradosso, chiamò il programma *Expensive Desk Calculator* [calcolatore da tavolo costoso] e orgogliosamente lo impiegò per fare il suo compito.

Il suo voto: zero. "Hai usato un computer!" fu il commento del professore. "Questo compito *non può* essere giusto."

Wagner non provò neanche a dare spiegazioni. Come avrebbe potuto spiegare al suo insegnante che il computer stava trasformando in realtà quelle che una volta erano state solo incredibili possibilità? O che un altro hacker aveva addirittura scritto un programma chiamato *Expensive Typewriter* [macchina da scrivere costosa] che trasformava il Tx-0 in qualcosa in grado di scrivere testo, elaborare il documento in righe di caratteri e stamparli sul Flexowriter... potreste immaginare un professore che accetta un compito in classe *scritto con il computer?* Come poteva quel

professore e chiunque non si fosse mai immerso in questo inesplorato universo di uomini-macchina, capire che Wagner e i suoi compagni hacker usavano regolarmente il computer per simulare, secondo le parole dello stesso Wagner, "strane situazioni, altrimenti difficilmente rappresentabili"? Il professore avrebbe imparato col tempo, come tutti, che l'universo aperto dal computer è infinito.

Se qualcuno avesse bisogno di un'ulteriore prova, si potrebbe citare il progetto a cui Kotok stava lavorando al centro di calcolo, il programma del gioco degli scacchi che il barbuto professore di intelligenza artificiale [la] John McCarthy o "Zio", com'era stato soprannominato dai suoi studenti hacker, aveva cominciato sull'IBM 704. Sebbene Kotok e vari altri hacker che collaboravano con lui al programma nutrissero soltanto disprezzo per la mentalità a elaborazione batch dell'IBM che aveva pervaso la macchina e la gente intorno a essa, erano riusciti a rubacchiare qualche ora a tarda notte per provare a usarlo interattivamente, e ingaggiare così una battaglia informale con i programmatori dei sistemi del 704 per vedere quale gruppo sarebbe stato conosciuto quale più grande consumatore di tempo al computer.

La testa della classifica era incerta, e O personale camice-bianco-e-cravatta-nera era rimasto abbastanza impressionato da permettere a Kotok e al suo gruppo di toccare i comandi e gli interruttori del 704: un raro contatto fisico col tanto celebrato "bestione" IBM.

Il ruolo di Kotok nel far nascere il programma di scacchi era indicativo di quello che era il ruolo degli hacker nel campo dell'intelligenza artificiale: una testa d'uovo come McCarthy o il suo amico Marvin Minsky impostavano un progetto o si domandavano ad alta voce se una certa cosa fosse realizzabile e gli hacker, se interessati, si mettevano all'opera.

Il programma di scacchi all'inizio era stato scritto in Fortran, uno dei primi linguaggi di programmazione procedurali. I linguaggi di programmazione procedurali assomigliano più alla lingua naturale che al linguaggio assembly, sono più facili da usare e fanno più cose con meno istruzioni; comunque, ogni volta che un'istruzione viene impartita in un linguaggio procedurale come il Fortran, il computer deve prima tradurre quel comando nel proprio linguaggio macchina binario. Ciò viene eseguito da un programma chiamato compilatore che, per compiere questo lavoro, impiega tempo e spazio prezioso all'interno del computer. In effetti, usando un linguaggio procedurale ci si allontana dal contatto diretto con il computer e quindi generalmente gli hacker preferiscono - ai linguaggi meno eleganti di "alto livello" come il Fortran - il linguaggio assembly o, come lo chiamano loro, linguaggio "macchina".

Kotok, dunque, si rese conto che a causa dell'enorme mole di numeri che sarebbero stati trattati nel programma di scacchi, una parte del programma doveva essere scritta in Fortran, e una parte in linguaggio assembly. Lo hackerarono pezzo per pezzo, con "generatori di mosse", strutture dati di base, e ogni genere di algoritmi innovativi per la strategia di gioco. Dopo aver dato alla macchina le regole per muovere ogni pezzo, le diedero dei parametri con cui valutarne la posizione, considerare le varie possibilità di movimento, e poi scegliere la mossa che avrebbe posto il pezzo nella condizione più vantaggiosa. Kotok ci lavorò per anni; il programma cresceva insieme ai computer IBM del MIT, e una notte memorabile un gruppetto di hacker si raccolse per vedere il programma fare le sue prime mosse in una partita vera. L'apertura fu piuttosto dignitosa, ma dopo otto scambi o giù di lì incominciarono i guai, con il computer messo sotto scacco. Tutti volevano vedere come avrebbe reagito. Lui, il computer, prese tempo (sapevano tutti che durante quelle pause la macchina stava effettivamente "pensando", posto che il vostro concetto di pensiero includa la considerazione automatica delle varie mosse, la loro valutazione, il rifiuto di gran parte di esse e l'uso di una serie predefinita di parametri secondo i quali fare la scelta definitiva). Alla fine, il computer mosse una pedina due caselle in avanti, saltando irregolarmente sopra un altro pezzo. Un bug! Ma era un bug intelligente: liberava il computer dallo scacco matto. Forse il programma stava formulando qualche nuovo algoritmo con cui conquistare la scacchiera.

In altre università, certi professori sostennero pubblicamente che i computer non sarebbero mai stati capaci di sconfiggere un essere umano a scacchi. Gli hacker la pensavano diversamente e proprio loro sarebbero stati quelli che avrebbero condotto i computer ai livelli più alti e sarebbero stati i principali beneficiari di questa simbiosi con il computer. Non sarebbero stati comunque gli unici: tutti avrebbe guadagnato qualcosa dall'uso di macchine pensanti, in un mondo automatizzato anche a livello intellettuale. E ancor più ne avrebbe beneficiato chiunque si losse rapportato al mondo con la stessa intensità di ricerca, lo stesso scetticismo verso la burocrazia, la stessa apertura alla creatività, la generosità nella condivisione dei saperi e la mentalità tipica di coloro che seguivano l'etica hacker Bisognerebbe accettare gli altri sulla stessa base non pregiudiziale con cui i computer accettano chiunque inserisca codice in una Flexowriter. Non trarremmo forse giovamento se imparassimo dai computer i mezzi per costruire un sistema perfetto e tentassimo di emulare quella perfezione in un sistema umano? Se *tutti* potessimo interagire con i computer con lo stesso impulso creativo, produttivo e innocente degli hacker, la loro etica potrebbe spargersi attraverso la società come un'onda benefica e i computer cambierebbero davvero il mondo per il meglio.

Nei confini monastici del Massachussetts institute of technology, la gente viveva questo sogno: il sogno degli hacker. Nessuno osava sperare che il sogno si potesse diffondere, perciò la gente cominciò a costruire proprio là, al MIT, l'irripetibile esperienza di un paradiso terrestre hacker.

Nell'estate del 1961, Alan Kotok e gli altri hacker del TMRC erano venuti a sapere che una nuova società avrebbe presto consegnato al MIT, assolutamente gratis, un nuovo prodotto dell'evoluzione dell'informatica, una macchina che faceva compiere diversi passi in avanti alle caratteristiche interattive del Tx-0, una macchina che per gli hacker poteva essere anche meglio del Tx-0: il Pdp-1.

Questa avrebbe cambiato il mondo dell'informatica per sempre e avrebbe latto avvicinare alla realtà il sogno ancora confuso degli hacker.

Alan Kotok s'era fatto conoscere come un vero mago del Tx-0, tanto che insieme .1 Saunders, Samson, Wagner e qualche altro, era stato assunto da Jack Dennis per costituire un gruppo di programmazione di sistema del Tx-0. La paga sarebbe stata ili ben 1 dollaro e 60 all'ora. Per alcuni hacker, il lavoro era solo una delle tante scuse per non entrare in classe e alcuni hacker, come Samson, non avrebbero mai preso la laurea e sarebbero stati troppo occupati con l'hackeraggio per rimpiangerne veramente la perdita. Kotok, comunque, non fu capace solo di destreggiarsi tra i Mini corsi, ma anche di affermarsi come hacker "canonico". Presso il Tx-0 e il TMRC, si era conquistato una fama leggendaria. Un altro hacker, arrivato al MIT quell'anno, si ricorda di Kotok che dava dimostrazioni ai nuovi venuti di come funzionasse il Tx-0: "Mi diede l'impressione che fosse ipertiroideo o qualcosa di simile," rammenta Bill Gosper, che sarebbe anch'egli diventato un hacker canonico, "parlava molto lentamente, era un cicciottello e i suoi occhi erano sempre semichiusi. Ma mi sbagliavo completamente. [Sul Tx-0] Kotok aveva un'autorità morale infinita: aveva scritto il programma di scacchi e ne capiva l'hardware." (Quest'ultima affermazione non era un complimento da poco: "capire l'hardware" è simile a dire "penetrare il Tao della fisica".)

L'estate in cui si seppe in giro del Pdp-1, Kotok lavorava alla Western electric, un lavoro fantastico, dato che tra tutti i sistemi possibili quello telefonico era il più ammirato. Il Model railroad club si recava spesso in visita alle centrali telefoniche, proprio come gli appassionati d'arte cercherebbero di visitare un museo. A Kotok sembrò curioso che in una società telefonica così grande, soltanto pochi ingegneri avessero una conoscenza approfondita della complessità del sistema. Ciò nonostante, quegli ingegneri erano in grado di fornire dettagli su specifiche funzioni del sistema, come i commutatori di selezione e i relè a passi successivi; Kotok e gli altri braccavano questi esperti per carpire

loro informazioni, e gli ingegneri, lusingati e che probabilmente non avevano idea che questi ragazzi del college avrebbero veramente *usato* quelle informazioni, accondiscendevano prontamente.

Kotok si era messo in testa di sfruttare quelle visite, leggere tutto il materiale tecnico su cui posava mano e vedere cosa succedeva componendo numeri differenti sul complicato sistema telefonico del MIT. Era una prima esplorazione, proprio come scoprire i sentieri digitali del Tx-0. Nell'inverno precedente, gli hacker del TMRC s'erano impegnati nella realizzazione di "una schedatura della rete telefonica", registrando tutti i luoghi raggiungibili dalle linee del sistema del MIT. Anche se non collegato alle linee telefoniche esterne, il sistema portava al Lincoln lab, e da lì ai committenti del ministero della difesa di tutto il paese. Era questione di catalo-gazione e tentativi. Si cominciava con un codice d'accesso, si aggiungevano un po' di cifre, si verificava chi rispondeva, si chiedeva a chi avesse risposto dove si trovava in quel momento e poi si aggiungevano altri numeri per saltare al luogo seguente. A volte era possibile raggiungere persine le linee urbane, con il cortese ausilio dell'ignara compagnia dei telefoni. E, come avrebbe in seguito ammesso Kotok, "se c'era una qualche debolezza nel sistema telefonico, un difetto di progetto col quale poter chiamare illegalmente, non mi tiravo indietro. Era un problema loro, non mio".

Eppure, la motivazione era la ricerca, non la frode, ed era considerato non etico profittare illecitamente di queste bizzarre connessioni. Certe volte i profani non lo comprendevano.

I compagni di camera di Samson al pensionato, per esempio, che non erano hacker, pensavano fosse lecito approfittare dei bug del sistema senza la sacrosanta motivazione dell'esplorazione del sistema. Dopo averlo messo sotto pressione per giorni, finalmente Samson cedette e passò loro un numero di una ventina di cifre che, diceva, avrebbe avuto accesso a una località proibita. "Potete chiamare dal telefono del corridoio," gli disse, "ma io non ci voglio venire". Mentre cominciavano ansiosamente a comporre il numero, Samson andò al telefono di sotto, che suonò giusto mentre stava arrivando. "Qui il Pentagono" sparò col suo miglior tono di voce seriosa. "Qual è il suo livello di sicurezza, per favore?" Dal telefono di sopra Samson udì espressioni di terrore, poi il click del telefono che riattaccava.

La schedatura della rete era ovviamente un'impresa limitata agli hacker, il cui desiderio di conoscere il sistema era più forte di qualsiasi paura d'essere beccati. Ma anche se l'esoterismo legato ai telefoni affascinava Kotok, la prospettiva del Pdp-1 ebbe il sopravvento. Forse intuiva che, dopo il Pdp-1, niente, nemmeno l'hackeraggio telefonico, sarebbe stato più lo stesso. Quelli che avevano progettato e vendevano questa nuova macchina non erano i soliti "camici bianchi". La compagnia era una nuova ditta, la Digital equipment corporation (Dee) e alcuni utenti del Tx-0 sapevano che i primi prodotti della Dee erano delle speciali interfacce create appositamente per quel Tx-0. Era già eccitante che alcuni fondatori della Dee avessero una concezione dell'informatica lontana dalla mentalità a elaborazione batch dell'IBM e che i ricercatori della Dee sembrava avessero osservato lo stile della comunità del Tx-0, assolutamente informale, interattiva, capace di valorizzare l'improvvisazione e di smanettare sui comandi. Sembrava che avessero progettato un computer che avrebbe consolidato quel tipo di comportamento. Il Pdp-1 (acroni-mo di Programmed data processor, un termine considerato meno minaccioso di "computer", che portava con sé tutte le connotazioni del mastodonte) sarebbe divenuto famoso come il primo "minicomputer", progettato non per macinare grandi quantità di numeri, ma bensì per la ricerca scientifica, per l'elaborazione degli enunciati matematici e per... l'hackeraggio.

Era così compatto che l'intero impianto non era più grosso di tre frigoriferi, non richiedeva molta aria condizionata e si poteva addirittura metterlo in funzione senza un'intera schiera di schiavi necessari per attivare nel giusto ordine le varie alimentazioni, far partire il generatore di temporizzazione del sistema, o altri compiti gravosi. Il prezzo al dettaglio era sbalorditivamente basso, 120.000 dollari, cioè abbastanza economico da zittire quelli che si lamentavano di quanto prezioso fosse ogni secondo di tempo operativo. Per di più, la macchina, che era nell'ordine il secondo Pdp-1 fabbricato (il primo era stato venduto alla vicina ditta scientifica Boll Beranek and Newman, o Bbn) non costò nulla al MIT: era stata donata dalla Dee all'Eie lab. Era evidente che gli hacker avrebbero avuto persine più tempo per lavorare su questa che sul Tx-0.

Il Pdp-1 sarebbe stato consegnato con una piccola dotazione di software che gli hacker giudicarono subito completamente inadeguata. Gli hacker del Tx-0 si erano ormai abituati a utilizzare il più avanzato software interattivo, una serie favolosa di programmi di sistema, scritti dagli stessi hacker e appositamente confezionati per le loro implacabili esigenze di controllo della macchina. Il giovane Peter Deutsch, il dodicenne che aveva scoperto il Tx-0, aveva mantenuto la sua promessa di scrivere un assemblatore ancor più strabiliante, mentre Bob Saunders aveva tirato fuori una versione più piccola e veloce del programma debugger *Flit* (lo strumento software per la ricerca degli errori di un programma), chiamata *Micro-Flit.* Questi programmi avevano tratto beneficio da un set di istruzioni più esteso: un giorno, dopo un notevole lavoro di pianificazione e progettazione da parte di Saunders e Jack Dennis, il Tx-0 era stato spento e un gruppo di ingegneri ne aveva esposto le "interiora" e cominciato a implementare nuove istruzioni nella macchina. Questo formidabile lavoro espanse il linguaggio assembly con molte altre istruzioni. Quando pinze e caccia-viti furono accantonati e con precauzione venne acceso il computer, tutti iniziarono forsennatamente a debuggare i programmi e ad aggiornarli utilizzando le nuove istruzioni.

Il set d'istruzioni del Pdp-1 non era molto differente da quello del Tx-0 espanso; Kotok lo sapeva e quindi si mise, senza sforzo, a scrivere software per il Pdp-1 sin da quella stessa estate, utilizzando ogni minuto del tempo libero che aveva. Immaginando che tutti avrebbero dato l'assalto per cominciare a programmare non appena la macchina fosse installata, lavorò per portare il *Micro-Flit* sulla nuova macchina in modo che scrivere software per l'"Uno" sarebbe stato più facile. Samson chiamò prontamente il debugger scritto da Kotok *Ddt* e il nome gli sarebbe rimasto incollato, benché il programma stesso avrebbe subito più volte modifiche per opera degli hacker che volevano aggiungere caratteristiche operative o ridurre il codice.

Ma Kotok non era l'unico a prepararsi all'arrivo del Pdp-1. Come un'eterogenea e nervosa accolita di genitori prima di un parto, gli altri hacker erano occupatissimi a tessere babbucce e copertine software per il neonato in arrivo, così quest'annunciato erede al trono dell'informatica avrebbe subito ricevuto il benvenuto non appena consegnato alla fine di settembre.

Gli hacker aiutarono a trasportare il Pdp-1 nella sua nuova casa, la "kludge room" proprio la porta a fianco del Tx-0. Era una meraviglia: davanti alla consolle grande la metà di quella del Tixo, c'era un pannello pieno di interruttori e di luci; accanto c'era lo schermo incassato in un alloggiamento esagonale in stile quasi *déco* di un blu brillante; dietro c'erano alti armadi, grandi quanto un frigo e profondi tre volte tanto, con cavi, piastre, commutatori e transistor: entrarci, naturalmente, era proibito. C'era un Flexowriter collegato per l'input (il personale si lagnò così tanto per il rumore, che il Flexowriter venne poi rimpiazzato con una macchina da scrivere Tbm modificata, che non funzionava però altrettanto bene) e un lettore di nastro perforato ad alta velocità, sempre in input. Alla fin fine, un giocattolo veramente divino.

A Jack Dennis piacevano alcune parti del software scritto dalla Bbn per il prototipo del Pdp-f, specialmente l'assemblatore. Kotok, invece, quando lo vide girare si sentì male: l'operazione non sembrava coincidere con lo stile accuratissimo che prediligeva, e così lui e alcuni altri dissero a Dennis che volevano scriverne un altro. "È una cattiva idea", replicò Dennis, che voleva subito un assemblatore funzionante, e immaginava ci sarebbero volute settimane prima che gli hacker ne creassero uno.

Kotok e gli altri furono inflessibili. Quello era un programma con cui avrebbero convissuto: doveva essere perfetto. (Naturalmente nessun programma lo è mai, ma ciò non ha mai fermato un hacker.)

"Ho una proposta," disse Kotok, questo ventenne mago del computer con il corpo di Budda, allo scettico seppur comprensivo Jack Dennis, "se noi ti scriviamo questo programma durante il weekend, ce lo pagherai?"

Le retribuzioni a quel tempo s'aggiravano intorno a poco meno di cinquecento dollari tutto compreso. "Mi sembra un ottimo affare", rispose Dennis.

Kotok, Samson, Saunders, Wagner e altri due cominciarono la notte di un venerdì di fine settembre. Pensavano di partire dal programma assemblatore del Tx-0 la cui prima stesura era di Dennis e che il dodicenne Peter Deutsch, tra gli altri, aveva rimesso a nuovo. Non avrebbero cambiato input o output, né avrebbero riscritto gli algoritmi; ogni hacker si sarebbe invece preso una sezione di programma del Tx-0 e lo avrebbe convcrtito in codice per il Pdp-1. Non dormirono mai! Durante quel weekend, sei hacker lavorarono l'equivalente di duecentocinquanta ore, scrivendo codice, correggendolo e innaffiando il cibo del take-away cinese da asporto con quantità massicce di Coca-cola. Fu un'abbuffata di programmazione e quando Jack Dennis arrivò quel lunedì mattina rimase sbalordito nel trovare un assemblatore installato sul Pdp-1, pronto a dimostrare che era in grado di trasformare il codice as-sembly in binario.

Con la pura forza dell'hackeraggio, gli hacker del Tx-0 - anzi no, del *Pdp-1* - avevano, in un weekend, tirato fuori un programma che sarebbe costato, all'industria del computer, settimane o forse mesi di duro lavoro. Era un progetto che probabilmente quest'ultima non avrebbe intrapreso senza una lunga e tediosa procedura di domande scritte, studi, incontri, esitazioni burocratiche e molto probabilmente anche notevoli compromessi durante l'attuazione. E poteva anche capitare che non se ne sarebbe fatto nulla. Il progetto fu un trionfo per l'etica hacker.

Gli hacker ebbero ancor più accesso a questa macchina di quanto già non ne avessero al Tx-0, e quasi tutti trasferirono le loro attività nella "kludge room". Alcuni rimasero testardamente a lavorare col Tixo e questo, per gli hacker del Pdp-1, era motivo di benevola ilarità. Tanto per girare il dito nella piaga, gli hacker avevano preparato una piccola dimostrazione basata sui nomi mnemonici delle istruzioni di questa macchina audacemente nuova, includendo alcune istruzioni esoteriche come DAC (deposit accumulator), LIO (load input-output), DPY (deplay) e JMP (jump). Il gruppo del Pdp-1 si sarebbe messo in riga e declamato all'unisono: LAC, DAC, DIPPY DAP, LIO, DIO, JUMP!

Appena scandita l'ultima parola, "Jump!" [salto!], saltavano tutti verso destra. La scarsa qualità della coreografia era più che compensata dal loro entusiasmo: erano stati sopraffatti dalla bellezza della macchina e dei computer.

Lo stesso tipo di entusiasmo era evidente nella ancor più spontanea programmazione che aveva luogo sul Pdp-1, e che andava da seri programmi di sistema a programmi per controllare un primitivo braccio robot, fino a hack del tutto inconsue-li. Uno di questi ultimi fu la connessione tra il Pdp-1 e il Tx-0, utilizzando un cavo ni traverso cui sarebbero passate le informazioni, un bit alla volta, tra le due macchine. Samson racconta che allora gli hacker invitarono il venerabile pioniere dell'intelligenza artificiale John McCarthy a sedersi al Pdp-1: "Professor McCarthy, dia un'occhiata al nostro nuovo programma di scacchi!" Poi chiamarono un altro insegnante al Tx-0: "Venga a vedere il nuovo programma di scacchi!" Dopo che McCarthy ebbe battuto sulla tastiera la sua prima mossa e che questa apparve sul I Icxowriter del Tx-0, gli hacker nella seconda stanza dissero all'altro professore che era appena stato testimone dell'apertura del Tx-0: "Adesso faccia la sua! " Dopo qualche mossa, McCarthy s'accorse che il computer visualizzava le mosse, una lettera alla volta, con una pausa sospetta tra una e l'altra lettera. Allora McCarthy seguì il cavo finché arrivò al suo avversario in carne e ossa. Gli hacker se la risero a crepapelle. Ma non ci sarebbe voluto molto prima che tirassero fuori programmi per computer - senza scherzi - per giocare veramente gli scacchi a livello di torneo.

Il Pdp-1 spinse gli hacker a programmare senza limiti. Samson hackerava cose come il calendario maya (che funzionava con un sistema di numerazione a base venti) facendo gli straordinari su una nuova versione del suo programma musicale per il Tx-0 che si avvantaggiò delle aumentate capacità audio del Pdp-1 per creare musica a tre voci: fughe in tre movimenti di Bach, melodie che interagivano... e la musica computerizzata proruppe dalla vecchia "kludge room"! La gente alla Dec aveva sentito parlare del programma di Samson e gli chiese di completarlo sul Pdp-1, e Samson lo elaborò al punto che si poteva inserire una brano dalla tastiera con una semplice traduzione delle note in lettere e cifre, e il computer avrebbe risposto con una sonata d'organo a tre voci. Un altro gruppo codificava le operette di Gilbert e Sullivan.

Con orgoglio Samson regalò alla Dee il compilatore musicale perché lo distribuissero a chiunque ne facesse richiesta: era fiero del fatto che altra gente avrebbe usato il suo programma. Il gruppo che lavorava sul nuovo assemblatore si comportò allo stesso modo e agli hacker andava bene di tenere il nastro perforato del programma nel cassetto in modo che chiunque usasse la macchina potesse accedervi, cercare di migliorarlo, tagliare via un po' d'istruzioni inutili, o aggiungervi nuove caratteristiche operative. Furono onorati quando la Dee richiese il programma per offrirlo agli altri proprietari di Pdp-1 e la questione sui diritti d'autore non fu mai sollevata. Per Samson e gli altri l'uso del computer era una tale gioia che avrebbero pagato pur di godersela e il fatto che fossero pagati la principesca somma di 1 dollaro e 60 l'ora era per loro un regalo. Quanto ai diritti d'autore, il software non era forse un dono al mondo, qualcosa che era già ricompensa? L'idea era quella di rendere il computer sempre più utile, più stimolante per i suoi utenti, fare dei computer qualcosa di così interessante che la gente fosse tentata di giocare con loro, esplorarli ed eventualmente farci anche dei programmi. Scrivere un buon programma significa costruire una comunità e non produrre una merce. In ogni caso, nessuno dovrebbe mai pagare per il software: l'informazione dev'essere libera!

Gli hacker del TMRC non erano gli unici ad aver messo a punto dei progetti per il nuovo Pdp-1. Durante l'estate del 1961, un progetto per il programma più sofisticato realizzato fino ad allora - una specie di vetrina di quanto una rigorosa applicazione dell'etica hacker avrebbe potuto produrre - stava per essere attuato. Sede di questi dibattiti era un appartamento in Higham Street a Cambridge, e i loro animatori erano tre programmatori itineranti tra i venti e i trent'anni in circolazione da anni per vari centri di calcolo. Dei tre, due vivevano nell'appartamento e così, in onore alle pompose denominazioni che emanava la vicina Harvard university, i tre scherzosamente chiamavano casa loro l'Higham institute.

Uno degli accademici di questa pseudoistituzione era Steve Russell, soprannominato per oscure ragioni, Slug. Aveva quel modo di parlare da scoiattolo affannato, atteggiamento diffusissimo tra gli hacker, insieme alle lenti spesse, l'altezza modesta, il fanatismo per i computer, i film dell'orrore e la peggior fantascienza. Tutti e tre gli interessi erano condivisi dai partecipanti alle riunioni in Higham Street. Russell era stato anche un *"coolie"* (per usare un termine del TMRC) di Zio John McCarthy, cioè aveva svolto anche lavori di manovalanza e compilazione per il professore.

McCarthy aveva tentato di concepire e implementare un linguaggio d'alto livello all'altezza del lavoro sull'intelligenza artificiale. Pensava di averlo trovato nel Lisp. Il linguaggio era chiamato così per il suo metodo di LISt Processing. Con comandi semplici ma potenti, il Lisp poteva fare molte cose con poche linee di codice; poteva anche eseguire potenti ricorsioni (dei riferimenti a un oggetto all'interno dello stesso oggetto) che avrebbero permesso ai programmi scritti in quel linguaggio di "imparare" veramente da ciò che accadeva mentre il programma girava. Il problema con il Lisp a quell'epoca era che occupava un bel po' di spazio su un computer, girava molto lentamente, e generava voluminose quantità di codice extra, tanto da

richiedere un altro programma di "raccolta dell'immondizia" per pulire periodica-mente la memoria del computer.

Russell stava aiutando Zio John a scrivere un interprete Lisp per il "bestione" IBM 704. Era, stando alle sue parole, "un orribile lavoro d'ingegneria", principalmente a causa dalla noiosa procedura di elaborazione batch cui il 704 costringeva.

Paragonato a quella macchina, il Pdp-1 sembrava a Slug Russell la terra promessa: più accessibile del Tx-0 e senza l'elaborazione batch! Benché non sembrasse grosso abbastanza per poter usare il Lisp, aveva altre meravigliose qualità, alcune delle quali oggetto di discussione all'Higham institute. Ciò che interessava particolarmente Russell e i suoi amici era la prospettiva di venir fuori con un qualche elaborato "trucco di visualizzazione" sul Pdp-1, usando il monitor. Dopo lunghe discussioni notturne, i tre dell'Higham institute si convinsero che la miglior dimostrazione della magia del computer sarebbe stata un gioco sorprendente sotto il profilo visivo.

C'erano stati diversi tentativi di realizzare questo genere di cose sul Tx-0. Uno di questi era un hack dal titolo *Mouse in thè Maze* [topo nel labirinto]: prima l'utente costruiva un labirinto con una penna luminosa, poi un blip sullo schermo che rappresentava un topo sarebbe andato in cerca di altri blip dall'aspetto di pezzi di formaggio**.** C'era anche una versione più snob del gioco, nella quale il topo rincorreva un Martini; dopo aver trovato il bicchiere, ne cercava un altro, fino a consumare nule le sue energie, troppo ubriaco per andare avanti. Quando si faceva correre il lupo una seconda volta dentro il labirinto, esso si sarebbe "ricordato" il percorso ı«Trovare i bicchieri e come un esperto ubriacone si sarebbe precipitato a farsi i ᴍ ni bevuta. Questo era quanto, più o meno, potevano offrire i trucchi di visualizzazione del Tx-0.

Ma già sul Pdp-1, che aveva uno schermo più facile da programmare di quello «lei Tx-0, c'erano stati alcuni significativi programmini per lo schermo. Lo sforzo più ammirato era stato quello di uno dei due geni gemelli dell'intelligenza artificiale al MIT, Marvin Minsky (l'altro era, ovviamente, McCarthy). Minsky era più espansivo dell'altro esperto collega dell'Ia e più disposto a entrare nel modo di vedere le cose degli hacker. Era un uomo dalle idee grandiose sul futuro dell'informatica: credeva veramente che un giorno le macchine sarebbero state in grado di pensare e aveva più volte suscitato scalpore definendo senza mezzi termini i cervelli umani "macchine di carne" e sostenendo implicitamente che le macchine non fatte di carne avrebbero funzionato altrettanto bene, un giorno. Uomo vivacissimo, con un guizzo negli occhi dietro le spesse lenti, la testa completamente pelata e un onnipresente maglione girocollo, Minsky diceva queste cose col suo stile secco, teso simultaneamente a spingere a fondo la provocazione e ad alludere che si trattasse, in fondo, di un enorme abbaglio cosmico: *naturalmente le macchine non possono pensare, he he he.* Minsky era l'uomo giusto e gli hacker del Pdp-1 partecipavano spesso al suo corso, *Introduzione all'Ia 6.544,* non solo perché era un ottimo teorico ma anche perché conosceva bene la sua materia. Dai primi anni Sessanta, Minsky aveva cominciato a organizzare quello che sarebbe diventato il primo laboratorio d'intelligenza artificiale del mondo; e sapeva bene che per fare quel che aveva in mente gli occorrevano geni della programmazione a mo' di fanteria e quindi incoraggiò l'hackeraggio in ogni modo.

Uno dei contributi di Minsky all'incremento della disciplina degli hack fu un programma di visualizzazione sul Pdp-1, chiamato *Circle Algorithm.* L'aveva scoperto per caso; mentre cercava di tagliar via un'istruzione da un programmino che serviva a convertire linee rette in curve e spirali, Minsky inavvertitamente scambiò un "Y carattere" per un "Y numero primo" e, invece di vedersi sullo schermo le prime spirali caotiche come si aspettava, comparve un cerchio: una scoperta incredibile, che in seguito avrebbe fatto emergere profonde implicazioni matematiche. Approfondendo la ricerca, Minsky usò il *Circle Algorithm* come base di lancio per una visualizzazione più elaborata, in cui tre particelle s'influenzavano a vicenda, creando un turbinio di affascinanti disegni sullo schermo e rose con diversi numeri ili petali che si autogeneravano. "Le forze che le particelle esercitavano le une sulle altre erano incredibilmente bizzarre," ricorda oggi Bob Wagner, "e simulavamo una violazione della legge naturale!" Minsky battezzò il suo hack *Tri-Pos: Three-Position Display,* ma gli hacker lo chiamarono affettuosamente *Minskytron.*

Slug Russell venne ispirato da quei fatti. Alcuni mesi prima durante le discussioni all'Higham institute, lui e i suoi amici avevano esaminato le caratteristiche dell'hack definitivo. Dato che erano stati appassionati della fantascienza-spazzatura, particolarmente dei romanzi di E.E. Doc Smith, avevano deciso che il Pdp-1 sarebbe stata una macchina perfetta per creare un qualcosa che fosse a metà tra un film di serie B e un giocattolo da 120.000 dollari. Un gioco in cui due persone potessero affrontarsi a duello. All'Higham institute venne dunque prontamente organizzato un gruppo di ricerca sul tema delle guerre spaziali, le cui conclusioni sottintendevano che Slug Russell sarebbe stato l'autore di questo storico hack.

Ma, dopo diversi mesi, Russell non aveva nemmeno cominciato. Esaminava lo schermo del *Minskytron* comporre sequenze, girava i commutatori per veder sviluppare nuovi pattern e ogni tanto, se il sistema si impiantava, ne girava altri. Era affascinato, ma riteneva l'hack troppo astratto e matematico. "Questa demo è una schifezza," affermò alla fine, "ci sono solo trentadue istruzioni o giù di lì, e in realtà non *fa* niente."

Slug Russell sapeva che il suo gioco di guerra spaziale sarebbe stato diverso: col suo dizionario da fantascienza kitsch proclamò che sarebbe stato l'hack più *astrale* dell'universo. Ciò che aveva tirato Slug dentro i computer era innanzitutto il senso di potere che si prova usando quei cavolo di scatoloni. Puoi dire al computer cosa deve fare, e lui combatte con te, ma alla fin fine fa sempre quello che gli hai ordinato di fare. Naturalmente il tutto rispecchierà il tuo grado di stupidità, e spesso quello che gli dici di fare può produrre qualcosa di spiacevole. Ma anche in quel caso, dopo torture e tribolazioni, farà *esattamente* ciò che vuoi. La sensazione che provi allora è diversa da ogni altra al mondo... e può farti diventare un tossicomane. Questo è ciò che accadde a Slug Russell, e lui cominciava a rendersi conto che aveva avuto lo stesso effetto negli hacker che si trattenevano nella "kludge room" fino all'alba. Era questa la sensazione che viveva, e Slug Russell immaginò che fosse *potere.*

Una sensazione simile, sebbene meno intensa, Slug l'aveva avuta dai romanzi di Doc Smith. Lasciò che la sua immaginazione si scatenasse a costruire emozionanti scorribande per lo spazio a bordo di rombanti astronavi bianche... e si domandò se quella stessa eccitazione potesse essere ricreata stando seduto dietro la consolle del Pdp-1. Doveva essere *quella* la *guerra spaziale* intorno a cui aveva fantasticato. Ancora una volta si propose di realizzare quel programma... più tardi.

Slug non aveva la grinta di altri hacker. Talvolta aveva bisogno di un incentivo. Dopo che ebbe commesso l'errore di aprire la bocca su questo programma che sarebbe stato sul punto di creare, gli hacker del Pdp-1, sempre impazienti di vedere un altro programma aggiungersi alla pila di nastri di carta nel cassetto, lo pressavano perché lo *facesse.* Dopo aver cercato scuse per un certo tempo, disse che l'avrebbe fatto, ma che per prima cosa avrebbe dovuto immaginare come scrivere le sofisticate sequenze di istruzioni seno-coseno necessarie per tracciare la rotta dell'astronave.

Kotok sapeva che quell'ostacolo era facilmente superabile. In quel periodo, era in ottimi rapporti con la gente della Dee, a Maynard, a pochi chilometri di distanza. La Dee, paragonata alle industrie di computer d'allora, era senza tante pretese e non considerava gli hacker del MIT degli scapestrati o dei giocherelloni come avrebbe fatto l'IBM. Per esempio, un giorno che un pezzo dell'apparecchiatura s'era rotto, Kotok chiamò Maynard e ne parlò con la Dee; gli dissero: "Vieni qua e prenditi il pezzo di ricambio". Ma quando Kotok arrivò, erano già passate le cinque e trovò tutto chiuso. Senonché il sorvegliante gli permise di entrare e trovata la scrivania dell'ingegnere con cui aveva parlato, frugò

dappertutto finché non trovò il pezzo. Informalità: questo era ciò che piaceva agli hacker. E così non fu un problema per Kotok andar di nuovo a Maynard un giorno, dove era sicuro che qualcuno gli avrebbe fornito una routine di seno e coseno che avrebbe girato sul Pdp-1.

Trovò abbastanza facilmente chi l'aveva e, dato che l'informazione era libera, se la portò al palazzo 26.

"Ecco qua, Russell" disse Kotok, con il nastro di carta in mano. "Ora quale altra scusa hai?"

A quel punto, Russell non poteva più tirarsi indietro. Così passò le sue ore libere a scrivere questo gioco fantastico come non se n'erano mai visti prima per il Pdp-1. Ben presto spese tutte le sue ore disponibili lavorando al gioco: aveva cominciato all'inizio di dicembre e quando arrivò Natale stava ancora scrivendo; quando il calendario fu girato sul 1962 stava ancora scrivendo. Russell era riuscito a produrre un punto sullo schermo che si poteva manipolare: attivando dei piccoli interruttori sul pannello di controllo si potevano far accelerare e cambiare direzione ai punti. Si mise dunque al lavoro per creare le sagome delle due astronavi: erano tutt'e due le classiche astronavi dei cartoni animati, appuntite in alto e con gli alettoni in basso.

Per distinguere l'una dall'altra, fece la prima paffuta e a forma di sigaro, con una protuberanza al centro, mentre disegnò la seconda con un profilo affusolato. Russell usò le solite istruzioni seno-coseno per capire come far muovere quelle sagome in diverse direzioni. Poi scrisse una subroutine per lanciare un "siluro" (un altro punto) dalla prua dell'astronave con un dispositivo posto sul computer. Il computer avrebbe esaminato la posizione del "siluro" e dell'astronave nemica; se entrambi occupavano la stessa area, il programma avrebbe iniziato un'altra serie d'istruzioni che rimpiazzava la nave colpita con una pioggia casuale di punti che voleva raffigurare un'esplosione (quel procedimento era chiamato "rilevamento di collisione").

Tutto questo era un significativo passo concettuale in avanti verso la più sofisticata programmazione "in tempo reale", dove quello che accade sul computer è in perfetta sincronia con quanto l'essere umano gli sta effettivamente trasmettendo. In altri termini, Russell stava imitando lo stile di debug interattivo on-line, di cui gli hacker erano maestri: la libertà di vedere quale istruzione ha fermato il programma e usare gli interruttori o il Flexowriter per infilarne dentro una diversa, e tutto mentre il programma sta girando con il debugger *Ddt.* Il gioco *Spacewar,* un vero e proprio programma per computer, contribuì a dimostrare che tutti i giochi - e qualsiasi altra cosa, forse - funzionano come programmi per computer. Quando ti smarrivi un po', bastava modificare i tuoi parametri e ti rimettevi in sesto. Potevi inserire nuove istruzioni. Lo stesso principio era applicabile al tiro a segno, alla strategia degli scacchi, e al lavoro di un corso al MIT. La programmazione di un computer non era una mera ricerca tecnica, ma un approccio ai problemi della vita.

Negli ultimi stadi della programmazione, Saunders aiutò Slug Russell; lavoravano intensamente dalle sei alle otto ore per ogni sessione. Un giorno di febbraio Russell svelò i fondamenti del gioco. C'erano due astronavi, ciascuna dotata di trentun missili. C'era un certo numero di punti disposti a caso sullo schermo a raffigurare le stelle in questo campo di battaglia spaziale. Le navi potevano essere manovrate attraverso quattro dispositivi posti sulla consolle del Pdp-1, che rendevano possibile la virata in senso orario e in senso antiorario, l'accelerazione e il lancio dei missili.

Slug Russell sapeva che mostrando una versione approssimativa del gioco e lasciando cadere il nastro di carta nella scatola con i programmi di sistema del Pdp-1, avrebbe invogliato a compiere dei non sollecitati miglioramenti. *Spacewar* non era una simulazione con il computer ordinaria: *diventavi effettivamente* un pilota di un'astronave da guerra. Era come se la fantascienza di Doc Smith fosse diventata realtà. Ma lo stesso potere a cui Russell aveva attinto per fare il suo programma - il potere che il Pdp-1 dava al programmatore per creare il suo piccolo universo - era disponibile anche agli altri hacker, che naturalmente si sentirono liberi di migliorare il piccolo universo di Slug Russell. Lo fecero immediatamente. Il tipo di miglioramenti avrebbe potuto essere riassunto dalla reazione degli hacker alla routine originale delle istruzioni per il lancio dei missili. Sapendo che nella realtà le armi militari non sempre sono perfette, Russell pensò di creare dei missili realistici. Anziché dotarli di una traiettoria in linea retta prima che restassero senza energia ed esplodessero, inserì alcune variazioni casuali per direzione e velocità. Invece di apprezzare la verosimiglianza, gli hacker pensarono bene di denunciarlo come un limite. A loro piacevano sistemi che andavano via lisci e strumenti attendibili e, così, aver a che fare con qualcosa che *non funzionava bene* li mandava in bestia. Russell in seguito capì che "è chiarissimo: le armi o gli strumenti che non siano più che affidabili sono tenuti in poca considerazione; alla gente piace poter contare sulle proprie attrezzature".

Naturalmente il problema poteva facilmente essere risolto. Il vantaggio che un mondo creato dal computer ha nei confronti di quello reale è proprio quello di poter risolvere un problema anche arduo come la traiettoria imperfetta dei missili, modificando soltanto qualche istruzione. Ecco perché fu naturale per così tanta gente perdersi nell'hackeraggio sin dal primo momento! I missili assunsero dunque una traiettoria regolare, e la gente passava ore a ingaggiare duelli nello spazio. E ancor più a cercare di perfezionare il mondo delle guerre spaziali.

Peter Samson, per esempio, amava l'idea delle battaglie stellari, ma non poteva sopportare i puntini generati a caso che fungevano da volta celeste. Lo spazio reale aveva stelle fisse in punti ben precisi. "Avremo la stessa cosa" si ripromise Samson. Si procurò un voluminoso atlante dell'universo e cominciò a inserire in una routine istruzioni che avrebbero generato quelle costellazioni visibili a chiunque si fosse trovato sull'equatore in una notte limpida.

Tutte le stelle entro la quinta magnitudine erano rappresentate; Samson riprodusse la loro giusta brillantezza riuscendo a controllare la frequenza d'accensione del punto sullo schermo che rappresentava la stella. Inoltre attrezzò il programma in modo che, man mano che il gioco progrediva, il ciclo scorresse di conseguenza (via via lo schermo riproduceva il 45 percento della volta celeste). Oltre a incrementare la verosimiglianza, questo programma denominato "planetario costoso ", forniva alle astronavi da guerra una mappa di base con cui misurare la propria posizione. Il gioco avrebbe potuto veramente essere chiamato, come disse Samson, *Shootout-at-El-Cassiopea,* sparatoria a El Cassiopea.

Un altro programmatore, Dan Edwards, non era soddisfatto del movimento lineare delle due astronavi in duello che rendeva il gioco solo un test di abilità motorie. Pensò che l'aggiunta del fattore gravita, avrebbe creato una componente strategica di rilievo. Quindi programmò una stella centrale - un sole - al centro dello schermo; in questo modo si sarebbe potuta sfruttare la sua attrazione gravitazionale per aumentare la velocità durante l'orbita, ma se non si prestava attenzione e si prendeva un'orbita troppo stretta, si sarebbe potuti precipitarvi dentro, il che significava la morte certa.

Prima che tutte le implicazioni strategiche di questa variante potessero essere impiegate, Shag Garetz, uno del trio dell'Higham institute, calò sul banco un asso nella manica. Aveva letto nei romanzi di Doc Smith che i pirati dello spazio potevano passare da una galassia all'altra grazie a una "galleria iperspaziale", che li proiettava in un "misteriosissimo spazio a n dimensioni". Aggiunse al gioco la possibilità dell'"iperspazio", permettendo al giocatore di scappare da una situazione di estremo pericolo premendo un "pulsante antipanico" che lo avrebbe spedito nell'iper-spazio. Fuggire nell'iperspazio era possibile per ben tre volte nel corso di una partita; lo svantaggio era che non sapevate in anticipo dove sareste sbucati. Certe volte si poteva anche ricomparire proprio vicino al sole, giusto in tempo per vedere l'astronave attirata senza speranza verso una morte sicura sulla superficie rovente. In onore al programma originale di Marvin Minsky, Garetz programmò la caratteristica dell'iperspazio in modo che un'astronave che vi entrasse avrebbe lasciato "un

segnale di emissione fotonica indotta dalla spinta d'accelerazione", un alone di luce che si formava spesso sullo schermo del *Minskytron.*

Le varianti erano infinite. Commutando pochi parametri potevate trovarvi in un gioco di "guerra spaziale idraulica", in cui i missili sgorgavano a fiotti anziché uno per uno. Oppure, mentre le ore si facevano piccole e la gente s'infilava nel programma interstellare, uno gridava: "Attivare 'venti spaziali!'" e qualcun altro scriveva un fattore di accelerazione che costringeva i giocatori a fare aggiustamenti ogniqualvolta si muovevano. Sebbene ogni perfezionamento che un hacker avesse desiderato fare fosse accolto con gioia, era pessimo stile fare cambiamenti che alterassero il gioco senza darne notizia. Le pressioni sociali che facevano rispettare l'elica hacker - che incitava a manipolare per migliorare, non per danneggiare - prevenivano ogni tentazione di combinare guai. Comunque, gli hacker erano già coinvolti in un'incredibile deviazione del sistema: stavano utilizzando un computer costosissimo per il gioco, in futuro, più venerato al mondo!

A *Spacewar* si giocava moltissimo: per alcuni era diventato una droga. Sebbene nessuno potesse prenotare ufficialmente il Pdp-1 per una sessione di *Spacewar,* quella primavera, ogni momento libero della macchina pareva avere una qualche versione del gioco in corso. Con la bottiglie di Coca-cola in mano (e talora anche soldi), gli hacker disputavano tornei estenuanti. Alla fine Russell scrisse una subroutine di istruzioni per calcolare il punteggio, visualizzando in ottale (a quel punto sapevano tutti leggere quel sistema numerico in base otto) il totale delle partite vinte. Per un certo tempo lo svantaggio principale era costituito dallo smanettamento necessario per azionare gli interruttori sulla scomoda consolle del Pdp-1 (tenendo le braccia in quella particolare posizione, dopo un po' i gomiti facevano male). Così, un giorno, Kotok e Saunders si recarono al TMRC e misero insieme i pezzi di quello che sarebbe stato il primo joystick. Lo costruirono interamente con scarti abbandonati nella stanza del club e lo assemblarono in un'ora di ispirata destrezza manuale. Le scatole di controllo erano di legno, con un coperchio di masonite e avevano degli interruttori per la rotazione e la spinta, come pure un pulsante per l'iperspazio. Tutti i comandi erano, naturalmente, molto silenziosi, così da poter furtivamente aggirare il nemico o immergersi rapidamente nell'iperspazio.

Mentre alcuni hacker perdevano interesse nello *Spacewar,* una volta esaurita la fase furibonda della programmazione, altri sviluppavano un istinto micidiale nel mettere a punto tattiche per far strage di nemici. La maggioranza delle partite erano vinte o perse nel giro di pochi secondi. Wagner divenne esperto nella tattica dello "stare in attesa", che consisteva nel rimanere fermo e immobile mentre la gravita lo teneva in orbita intorno al sole, poi piombava giù e cominciava a lanciare missili addosso all'avversario. C'era poi una variante nel gioco, detta "l'apertura Cbs", per cui si cercava prima la giusta angolazione di tiro e *poi* ci si buttava in orbita intorno alla stella: questa tattica prendeva il nome dal fatto che quando entrambi le astronavi rivali la assumevano, tracciavano sullo schermo un disegno che somigliava molto all'occhio della Cbs. Saunders, che prendeva il gioco molto seriamente, usava una tattica Cbs modificata per mantenere il predominio nel corso dei tornei e, per un certo periodo, nessuno riuscì a batterlo. Comunque, dopo venti minuti di difesa della propria posizione, perfino a un campione di *Spacewar* si sarebbero annebbiati gli occhi e così tutti riuscivano ad avere la loro possibilità di giocare a *Spacewar,* probabilmente più di quanto non fosse effettivamente salutare. Peter Sam-son, nel gioco secondo solo a Saunders, lo capì una notte, mentre tornava a casa. Scendendo dal treno, alzò lo sguardo al ciclo limpido e terso. Cadde una meteora. *Dov'è la mia astronave?* Samson pensò e, istintivamente, si girò annaspando nell'aria in cerca del joystick che non c'era.

Nel maggio del 1962, in occasione dell'annuale festa del MIT gli hacker diedero in pasto alla macchina un nastro di carta con ventisette pagine di linguaggio assem-bly per il Pdp-1, installarono uno schermo extra - in realtà un gigantesco oscilloscopio - e per tutto il giorno mostrarono *Spacewar* a un pubblico che si assiepava intorno allo schermo e non riusciva a credere ai propri occhi. La sua visione - un gioco di fantascienza scritto dagli studenti e gestito da un computer - era così vicina all'incredibile che nessuno osava pensare che un giorno ne sarebbe derivata una vera e propria industria del divertimento.

Solo qualche anno dopo, quando era alla Stanford university, Slug Russell si rese conto che il gioco non era altro che un'aberrazione hacker. Una notte, dopo aver lavorato fino a tardi, Russell e alcuni amici andarono in un bar vicino che aveva dei flipper. Giocarono fino alla chiusura; poi, invece di andare a casa, Russell e i colleghi tornarono al loro computer e per prima cosa si misero a giocare a *Spacewar.* Improvvisamente Steve Russell capì: "Questa gente ha appena finito di giocare con un flipper e si è messa a smanettare su *Spacewar...* oddio, anche *Spacewar è* un flipper". Il flipper sicuramente più avanzato, creativo e costoso sulla faccia della terra.

Come gli altri prodotti degli hacker e il programma musicale, il gioco delle guerre spaziali era in vendita e, come gli altri programmi, fu messo nel cassetto per chiunque volesse prenderlo, visionarlo e riscriverlo come gli sembrava meglio. Il lavoro di gruppo che, fase dopo fase, aveva perfezionato il programma forniva un altro argomento in favore dell'etica hacker: l'impulso a entrare nei meccanismi della cosa e renderla migliore aveva portato a un consistente miglioramento. E ovviamente era anche divertentissimo. Non c'era da meravigliarsi che gli altri proprietari di Pdp-1 cominciassero ad aver notizia della cosa e che i nastri di carta contenenti *Spacewar* fossero distribuiti gratuitamente. A un certo punto il pensiero che qualcuno *avrebbe dovuto* trarre benefici economici da tutto questo passò forse per la testa di Slug Russell, ma a quel punto erano ormai decine le copie che circolavano. La Dee fu deliziata nel riceverne una e gli ingegneri la usarono come un programma diagnostico finale per i Pdp-1, prima di distribuirli sul mercato. Alla fine, senza pulire la memoria del computer, spegnevano la macchina. I rivenditori della Dee lo sapevano e, spesso, quando le macchine venivano consegnate ai nuovi acquirenti, le accendevano, si accertavano che non uscisse fumo da dietro, e digitavano "VY" sulla tastiera. Se la macchina era stata ben imballata e installata, al centro dello schermo sarebbero apparsi la stella-sole e i missili a forma di sigaro e di tubo pronti per la battaglia cosmica. Un volo inaugurale per una macchina magica.

Spacewar, come si vide in seguito, fu la durevole eredità dei pionieri dell'hackeraggio al MIT. Nei due anni successivi molti dei programmatori del Tx-0 e del Pdp-1 lasciarono l'istituto. Saunders avrebbe trovato lavoro presso un'industria di Santa Monica (dove poi avrebbe scritto un programma di *Spacewar* per il Pdp-7 che usava per lavoro); Bob Wagner se ne andò alla Rand corporation; Peter Deutsch andò a Berkeley per iniziare il primo anno del college; Kotok trovò un lavoro part-time che si trasformò in una importante carica di progettazione alla Dec (sebbene per anni non riuscì a non girare attorno al TMRC e al Pdp-1). Con uno sviluppo che avrebbe avuto un peso considerevole nella diffusione dell'hackeraggio stile MIT fuori da Cambridge, John McCarthy lasciò l'istituto per fondare un nuovo laboratorio d'intelligenza artificiale sulla West Coast, alla Stanford university. Slug Russell, lo "scrivano" del Lisp di McCarthy, si accodò.

Delle facce nuove e certe emergenti attività in campo informatico assicuravano che la cultura hacker al MIT non avrebbe solo avuto una prosecuzione, ma che sarebbe rifiorita e sviluppata più che mai. Le nuove facce erano di hacker incredibilmente audaci, destinati a diventare leggende viventi tramandate di bocca in bocca. Ma la molla che avrebbe permesso a questa gente di interpretare il loro ruolo da grandi hacker era già in tensione, spinta da persone i cui nomi sarebbero diventati noti con mezzi più convenzionali: tesi, premi accademici e in qualche caso la notorietà all'interno della comunità scientifica.

Questi individui erano i *planner.* Tra loro si annoveravano scienziati che occasionalmente si dedicavano all'hackeraggio - Jack Dennis, McCarthy, Minsky - ma che in ultima analisi erano più interessati

dagli scopi dell'informatica che dediti all'elaborazione vera e propria. Vedevano i computer come una risorsa per una vita migliore della specie umana, ma non pensavano che lavorare su un computer fosse l'elemento chiave per rendere migliore quella vita.

Tra i planner, alcuni prefiguravano il giorno in cui computer dotati di intelligenza artificiale avrebbero alleggerito l'uomo dalla fatica mentale, allo stesso modo in cui le macchine industriali già lo avevano parzialmente affrancato dal giogo di quella fisica. McCarthy e Minsky erano l'avanguardia di questa scuola di pensiero, t-il entrambi avevano partecipato nel 1956 alla conferenza della Dartmouth che aveva creato una fondazione per la ricerca in questo campo. L'attività di McCarthy con il linguaggio d'alto livello Lisp era indirizzata verso questa meta, ed era sufficientemente avvincente per stimolare hacker come Slug Russell, Peter Deutsch, Peter Samson e altri a lavorare in Lisp. D'altro canto Minsky sembrava interessato all'intelligenza artificiale su una base più teorica: era un allegro e baldanzoso Johnny Appleseed del settore che spargeva i suoi semi, ognuno dei quali capace di lar sbocciare un vero "albero di mele" di utili tecniche e progetti di Ia. I planner erano anche molto favorevoli a mettere il potere dei computer nelle mani del maggior numero possibile di ricercatori, scienziati, statistici e studenti. Alcuni di loro lavoravano per renderne più facile l'uso; John Kemeny della Dartmouth mostrò come ciò fosse possibile scrivendo un linguaggio per computer facile da usare chiamato Basic (i programmi scritti in Basic girano molto più lentamente e occupano più spazio nella memoria di quelli scritti in linguaggio assembly, ma non esigono la dedizione quasi monastica richiesta invece dai linguaggi macchina). I planner del MIT tendevano a estendere l'accesso ai computer a quanta più gente possibile. C'erano molti ordini di motivi per questo, non ultima la proiezione del livello economico, elemento questo che da solo era ovviamente preferibile al sistema allora vigente, per il quale perfino i secondi del tempo trascorso al computer erano considerati una mercé di valore (anche se non ve ne sareste accorti osservando chi giocava a *Spacewar* sul Pdp-1). Se più gente avesse usato i computer, sarebbero emersi nuovi esperti programmatori e teorici, e la scienza informatica - sì, questi aggressivi planner la chiamavano scienza - avrebbe tratto solo beneficio dai nuovi talenti. Ma c'era qualcosa di più. Era qualcosa che ogni hacker sapeva, cioè la convinzione che l'informatica, in sé e per sé, fosse positiva. John McCarthy rappresentava questa fede quando diceva che lo stato naturale dell'uomo era restare collegato al computer tutta la vita. "Quel che l'utente vuole è un computer che possa essere ai suoi ordini per lunghi periodi di tempo."

L'uomo del futuro. Le mani su una tastiera, gli occhi su un monitor, sempre in contatto con il corpo dell'informazione e di pensiero che il mondo ha archiviato dall'inizio della storia. Tutto questo, per *l'uomo informatizzato,* sarebbe stato accessibile.

Niente di tutto questo sarebbe accaduto con l'IBM 704 a elaborazione batch. Né con il Tx-O o il Pdp-1, con le loro liste di prenotazione settimanali che si riempivano completamente qualche ora dopo l'affissione al muro. Per far questo, occorreva che più persone usassero il computer contemporaneamente (l'idea che ogni persona, uomo o donna, dovesse avere un computer personale era qualcosa che soltanto un hacker avrebbe considerato sensata). Questa concezione multiutente era denominata "time-sharing" e nel 1960 i planner più influenti del MIT promossero il "Long-range computer study group" [gruppo di studio a lungo termine sul computer]. Tra i membri c'erano personalità che avevano osservato con benevolenza e compiacimento l'ascesa degli hacker del MIT, gente come Jack Dennis, Marvin Minsky e Zio John McCarthy. Essi sapevano quanto fosse importante per loro mettere le mani su quegli aggeggi; il problema non era se adottare o meno il time-sharing: era questione di come farlo.

Le fabbriche di computer, particolarmente l'IBM, non erano entusiaste ed era evidente che il MIT avrebbe dovuto andare avanti da solo (nonostante anche l'azienda di ricerca di Boll Beranek e Newman stesse lavorando sul time-sharing). Alla fine al MIT iniziarono due progetti: il primo era il tentativo solitario di Jack Dennis di scrivere un sistema time-sharing per il Pdp-1. Il secondo fu intrapreso da un professore, F.J. Corbató, che avrebbe cercato aiuto dal riluttante Golia, l'IBM, per scrivere un sistema per il 7090.

Il ministero della difesa, specialmente attraverso l'Arpa *[Advanced research projects agency,* agenzia per i progetti di ricerca avanzata], aveva sostenuto i computer fin dai tempi della guerra, attento alle loro eventuali applicazioni d'uso militare. Così all'inizio degli anni Sessanta aveva ottenuto una serie di finanziamenti a vasto raggio per il suo progetto time-sharing, che sarebbe stato chiamato "progetto Mac" (le iniziali avevano un duplice significato Multiple access computing e Machine aided cognition) [elaborazione ad accesso multiplo e conoscenza assistita dalla macchina]. Lo Zio Sam avrebbe sborsato tre milioni di dollari all'anno, Dennis ne avrebbe avuto la direzione e Marvin Minsky avrebbe anche lui avuto un ruolo di primo piano, particolarmente nell'impiego di un terzo dei fondi che non sarebbe andato allo sviluppo del time-sharing, ma per il nuovo settore dell'intelligenza artificiale. Minsky se ne rallegrò moltissimo, dato che quel milione di dollari era dieci volte più sostanzioso del suo precedente budget per l'Ia, e perché si rendeva conto

clic buona parte dei rimanenti due terzi avrebbe preso comunque la strada verso l'Ia. Era una possibilità per mettere in piedi una struttura ideale dove, con macchine sofisticate, la gente avrebbe potuto progettare la realizzazione del sogno degli hacker, al riparo dalla pazzia burocratica del mondo esterno. Nel mentre, il sogno hacker sarebbe stato vissuto giorno dopo giorno dai devoti studenti della macchina.

I planner sapevano che sarebbe occorso personale specializzato per mandare avanti il laboratorio. Marvin Minsky e Jack Dennis sapevano che l'entusiasmo di quei brillanti hacker era essenziale per portare avanti le loro *grandi idee.* Come dirà in seguito Minsky del suo laboratorio: "In quell'ambiente c'erano diverse cose che andavano avanti. C'erano le teorie più astratte sull'intelligenza artificiale su cui si discuteva e alcuni (degli hacker) erano concentrati su quello. Ma c'era anche la faccenda di come si sarebbero potuti fare i programmi che a loro volta facessero quelle cose e come attuarli".

Minsky fu proprio felice di risolvere la faccenda lasciandola agli hacker, gente per cui "i computer sono la cosa più interessante al mondo". Il genere di gente che, per burla, avrebbe hackerato programmi anche più bestiali di *Spacewar* e poi, invece di giocarli tutta la notte (come era accaduto qualche volta nella "kludge room"), ne avrebbe hackerati altri. Al posto di simulazioni spaziali, gli hacker che lavoravano al progetto Mac avrebbero affrontato sistemi più complessi: braccia robotiche, progetti di visione artificiale, indovinelli matematici e labirintici sistemi time-sharing che avrebbero superato ogni immaginazione. Le classi del MIT di questi primi anni Sessanta avrebbero dato alcuni tra gli hacker più incredibili che si siano mai seduti dietro una consolle. E nessuno di loro meriterà più di Richard Greenblatt questa onorificenza.

L'idea hacker di comunicazione

L'idea hacker di comunicazione è opposta a quella sottostante al medium televisivo: orizzontale, rizomatica, decentrata, non gerarchica né autoritaria, non controllata né censurata dove diventa possibile scambiarsi saperi in modo paritario.

L'idea hacker di comunicazione è anche una visione diversa e critica della tecnologia: non più pensata per pochi "sacerdoti" , ma comprensibile, smontabile e ricomponibile per adattarla a fini individuali e collettivi.

L'idea hacker di comunicazione prevede altresì una condivisione dei saperi e delle tecnologie: la grande opportunità rappresentata dai programmi di software libero, che creati collettivamente in rete consentono già oggi a milioni di utilizzatori di sottrarsi al giogo economico del software commerciale e possono diventare un'opportunità di occupazione e di organizzazione nuova della produzione.

L'idea hacker di comunicazione pone le basi per pratiche condivise di intelligenza collettiva, che sappiano amplificare le risorse degli individui verso finalita' di bene comune.

L'idea hacker di comunicazione è alla base del concetto di Lavoro in Rete, una pratica che non ha frontiere, né conosce etnie e che quindi per sua natura non può far altro che opporsi attivamente a qualsiasi logica di guerra.

Assemblea dei promotori dell'Hack IT 99
(Laboratorio Studentesco Deposito Bulk – Milano)

Il primo hackmeeting internazionale: L'ICATA

(International Conference on the alternative use of technology)
DICHIARAZIONE FINALE DELL'ICATA 89
Adottata il 4/8/89

In questa dichiarazione programmatica finale, l'intera scena hacker internazionale ha concordato nell'agosto 1989 su alcuni principi base, al fine di riaffermare la propria pratica e di spezzare la canea montante repressiva, in corso contro di essi in quasi tutti i paesi del mondo. E' interessante notare che la pratica dell'hackeraggio viene letta come necessaria per infrangere il monopolio statale e delle multinazionali sull'informazione. Questo dominio, difatti, suona tanto più strano, se confrontato con l'oggettiva democraticità del mezzo "computer".

Noi, cittadini planetari e partecipanti alla FESTA GALATTICA DEGLI HACKERS e dell'ICATA 89 ad Amsterdam, abbiamo confrontato, durante tre giorni, le nostre idee, le nostre esperienze, le nostre speranze e rispettivi scopi per l'avvenire. Profondamente turbati dalla prospettiva di una tecnologia dell'informazione e degli attori economici e politici scatenati da essa, senza controllo democratico né partecipazione popolare efficace, noi abbiamo risoluto che:

Lo scambio libero e senza alcun ostacolo dell'informazione sia un elemento essenziale delle nostre libertà fondamentali e debba essere sostenuto in ogni circostanza. La tecnologia dell'informazione deve essere a disposizione di tutti e nessuna considerazione di natura politica, economica o tecnica debba impedire l'esercizio di questo diritto.

Tutta intera la popolazione debba poter controllare, in ogni momento, i poteri del governo; la tecnologia dell'informazione deve allargare e non ridurre l'estensione di questo diritto.

L'informazione appartiene a tutto il mondo, essa è prodotta per tutto il mondo. Gli informatici, scientifici e tecnici, sono al servizio di tutti noi. Non bisogna permettere loro di restare una casta di tecnocrati privilegiati, senza che questi debbano rendere conto a nessuno del loro operato.

Il diritto all'informazione si unisce al diritto di scegliere il vettore di questa informazione. Nessun modello unico di informatizzazione deve essere imposto a un individuo, una comunità o a una nazione qualsiasi. In particolare, bisogna resistere alle pressione esercitata dalle tecnologie "avanzate" ma non convenienti. Al loro posto, bisogna sviluppare dei metodi e degli equipaggiamenti che permettano una migliore convivialità, a prezzi e domanda ridotti.

La nostra preoccupazione più forte è la protezione delle libertà fondamentali; noi quindi domandiamo che nessuna informazione di natura privata sia stockata, né ricercata tramite mezzi elettronici senza accordo esplicito da parte della persona interessata. Il nostro obiettivo è di rendere liberamente accessibile i dati pubblici, proteggere senza incertezze i dati privati. Bisogna sviluppare delle norme in questo senso, insieme agli organismi e alle persone interessati.

Ogni informazione non consensuale deve essere bandita dal campo dell'informatica. Sia i dati che le reti devono avere libertà d'accesso. La repressione dei pirati deve divenire senza fondamento, alla maniera dei servizi segreti.

Parallelamente domandiamo che tutte le legislazioni, in progetto o già in applicazione, rivolte contro i pirati e che non perseguono scopi criminali o commerciali, siano ritirati immediatamente.

L'informatica non deve essere utilizzata dai governi e dalle grandi imprese per controllare e opprimere tutto il mondo. Al contrario, essa deve essere utilizzata come puro strumento di emancipazione, di progresso, di formazione e di piacere. Al contempo, l'influenza delle istituzioni militari sull'informatica e la scienza in generale deve cessare.

Bisogna che sia riconosciuto il diritto d'avere delle connessioni senza alcuna restrizione con tutte le reti e servizi internazionali di comunicazione di dati, senza interventi e controlli di qualsiasi sorta.

Bisogna stabilire dei tetti di spesa, per paese, per avere accesso a questi vettori di comunicazione di dati pubblici e privati. Si deve facilitare quei paesi senza una buona infrastruttura di telecomunicazione e la loro partecipazione nella struttura mondiale.

Noi ci indirizziamo agli utilizzatori progressisti di tecnologie di informazione nel mondo affinché socializzino le loro conoscenze e specializzazioni in questo campo con delle organizzazioni di base, al fine di rendere possibile uno scambio internazionale e interdisciplinare di idee e informazioni tramite delle reti internazionali.

Ogni informazione è al contempo deformazione. Il diritto all'informazione è al contempo inseparabilmente legato al diritto alla deformazione, che appartiene a tutto il mondo. Più si produce informazione, e più si crea un caos di informazione sfociante sempre più in rumore. La distruzione dell'informazione come del resto la sua produzione, è il diritto inalienabile di ognuno.

Bisognerebbe sovvertire i canali regolamentari e convenzionali dell'informazione grazie a dei detournaments e dei cambiamenti surrealisti degli avvenimenti, al fine di produrre del caos, del rumore, dello spreco i quali, a loro volta, saranno considerati come portatori di informazione.

La libertà di stampa deve applicarsi anche alle pubblicazioni tecno-anarchiche, che appaiono in giro, per reclamare la liberazione dei popoli, la fine delle tirannie della macchina e del sistema sugli uomini.

L'Homebrew Computer Club

La prima riunione del leggendario Homebrew Computer Club si svolse nel marzo del 1975 nel garage di uno dei membri a Menlo Park, contea di San Mateo, Silicon Valley. I membri del club erano appassionati di elettronica, sebbene molti fossero ingegneri elettronici o avessero comunque un'esperienza come programmatori. Durante la prima riunione parlarono dell'Altair 8800 e di altri argomenti di carattere tecnico. Durante le riunioni si scambiavano anche schemi elettrici e discutevano di programmazione. Dai membri del club emersero alcune persone che segnarono la storia dell'informatica come Bob Marsh, Adam Osborne e Lee Felsenstein. Alle riunioni parteciparono anche Steve Jobs e Steve Wozniak, i famosi fondatori dell'Apple Computer. Infatti fu durante una delle riunioni del club che fu presentato l'Apple I.

Steve Wozniak *- fonte Wikipedia*

L'Homebrew Computer Club
Steven Levy - Hacker

La notte del 5 marzo sulla Silicon Valley pioveva. Tutti e trentadue i partecipanti al primo meeting del gruppo ancora senza nome potevano distinguere il rumore della pioggia mentre sedevano sul pavimento di cemento dell'ampio box di Gordon French.

Alcuni dei convenuti si conoscevano; altri erano entrati casualmente in contatto attraverso il volantino che Fred Moore aveva affisso. Lee Felsenstein e Bob Marsh erano arrivati da Berkeley col furgone scassato di Lee. Bob Albrecht era venuto per dare al gruppo la sua benedizione e per mostrare l'Altair 8800 che la Mits aveva prestato alla Pcc. Toni Pittman, un ingegnere free-lance che aveva costruito un improbabile *computer homebrew* [fatto in casa] intorno al preistorico chip 4004 della Intel, aveva incontrato Fred Moore a una conferenza sui computer svoltasi il mese prima e pregustava con piacere l'incontro con altri che avessero interessi simili ai suoi. Steve Dompier, ancora in attesa delle rimanenti parti del suo Altair, aveva letto il volantino appeso al Lawrence Hall. Marty Spergel aveva un piccolo commercio di componenti elettronici e pensava fosse una buona idea sentire il parere sui chip da qualche tecnico.

Un ingegnere della Hewlett-Packard chiamato Alan Baum aveva sentito dell'incontro e si domandava se si volesse ragionare sui nuovi computer a basso costo; s'era portato dietro un amico che aveva conosciuto alle scuole superiori, un dipendente dell'Hp, un certo Stephen Wozniak.

Quasi tutte le persone riunite in quel garage erano fanatici dell'hardware, con la sola eccezione di Fred Moore, che aveva ideato una specie di piccola comunità in cui le persone si sarebbero "boostrappate" [attivate] da sole alla cultura dell'hardware. Non aveva ancora capito che questo era, come avrebbe poi puntualizzato Gordon French, "il più bel raduno di ingegneri e tecnici che fosse stato possibile mettere sotto lo stesso tetto". Questa era gente seriamente intenzionata a introdurre i computer dentro le proprie case per studiare, per divertirsi e per creare... e il fatto che avrebbero dovuto costruirsi il computer non era certo un deterrente. L'avvento dell'Altair aveva dimostrato che il loro sogno era realizzabile e vedere altri con lo stesso obiettivo era un secondo brivido. E nella parte anteriore del pienissimo laboratorio-garage di Gordon French - un'automobile non ci sarebbe proprio entrata - c'era lui, un Altair. Bob Albrecht lo accese, le luci si illuminarono e ognuno di loro sapeva che dietro quell'impassibile pannello frontale stavano agitandosi piccoli bit binari, eseguendo dei LDA, JMP e ADD.

Fred Moore aveva preparato un tavolo e prendeva appunti, mentre Gordon French, che era incredibilmente orgoglioso del suo 8008 fatto in casa, faceva da moderatore. Uno per uno si presentarono e venne fuori che sei dei trentadue presenti avevano costruito un qualche tipo di computer, mentre diversi altri avevano ordinato gli Altair. Ben presto, nacque un dibattito sui meriti e demeriti dei chip, in particolare dell'8008. In effetti c'erano argomenti infiniti per alimentare il dibattito: esadecimale (numeri in base sedici) contro ottale (numeri in base otto), codice macchina dell'8080, archivi su nastro di carta contro cassette contro carta e matita... Discussero di quello che avrebbero voluto in un club e le parole che ricorrevano di più tra i presenti erano "cooperazione" e "condivisione". Ci furono anche dibattiti su quello che la gente poteva fare con i computer in casa, e certi sostenevano che avrebbero fatto giochi, controllato i servizi domestici, editato testi, educato i figli. Lee menzionò il Community memory. Albrecht distribuì l'ultimo numero di "Pcc". Steve Dompier raccontò del suo pellegrinaggio ad Albuquerque, di come la Mits stesse cercando di far fronte a più di quattrocento ordinazioni, di quanto fossero occupati a cercare di spedire i kit di base, e che erano impossibilitati anche solo a pensare di poter spedire il materiale extra che avrebbe messo la macchina in grado di fare qualcosa di diverso dall'accendere delle luci.

Fred Moore era molto entusiasta dell'energia che la riunione aveva generato. Gli sembrava di aver messo in moto qualcosa. Non si rendeva conto in quel momento che l'origine di quella vivacità intellettuale non era la contemplazione, alla maniera dei planner, dei cambiamenti sociali resi possibili dall'informatica di massa, ma del fascino al calor bianco degli hacker verso la tecnologia. Spinti dalla buona volontà, tutti sembravano ritenere necessario il lavoro collettivo; Moore suggerì di fissare le riunioni del gruppo ogni quindici giorni. A simboleggiare la concezione di libero scambio che il gruppo avrebbe incarnato, Marty Spergel, il fornitore di materiali elettrici, noto all'interno del gruppo come "thè Junk Man" [rottamaio], tirò fuori un chip 8008, proprio quando tutti se ne stavano andando. "Chi lo vuole?" chiese, e appena si alzò la

prima mano, lanciò il chip, il primo pezzo di tecnologia delle dimensioni di un'unghia che poteva procurare una buona percentuale della potenza multimilionaria del Tx-0.

Più di quaranta persone vennero al secondo meeting, che si tenne allo Stanford Ai lab ai piedi delle montagne, la tana degli hacker tolkieniani di Zio John Mc-Carthy. Gran parte della riunione trascorse con una discussione su quale nome si sarebbe dato il gruppo.

I suggerimenti erano Infinitesimal computer club, Midget brains, Steem beer computer club, People's computer club, Eight-bit byte bangers, Bay area's computer experimenters' group e Amateur computer club of America. Alla fine rimasero Bay area amateur computer club e Homebrew computer club. Le ultime tre parole diventarono di fatto la denominazione. Nel pieno spirito hacker il club non richiedeva una tessera, non occorrevano pagamenti neanche minimi (ma il suggerimento di French di dare un dollaro per coprire le spese del volantino per la convocazione delle riunioni e la newsletter aveva fruttato al terzo incontro 52 dollari e 63), e non c'erano cariche elettive.

Già al quarto incontro, divenne evidente che l'Homebrew computer club sarebbe stato il paradiso degli hacker. Ben più di un centinaio di persone ricevettero per posta l'avviso che il meeting si sarebbe tenuto quella settimana alla Peninsula School, un'isolata scuola privata rannicchiata in un'area boschiva di Menlo park.

Steve Dompier aveva nel frattempo finito di costruire il suo Altair: aveva ricevuto l'ultima spedizione di pezzi una mattina alle dieci e aveva passato le successive trenta ore a montarlo, solo per scoprire che la memoria di 256 byte non funzionava. Sei ore dopo scopriva che l'errore era stato causato da un solco sul circuito stampato. Lo riparò e cercò di capire cosa farci.

Sembra che l'unica proposta della Mits per quelli che riuscivano a finire davvero la costruzione del computer fosse un programma in linguaggio macchina che si poteva inserire solo usando la fila di piccoli interruttori sul pannello frontale. Era un programma che utilizzava le istruzioni del chip 8080: LDA, MOV, ADD, STA e JMP. Se tutto andava bene, il computer avrebbe fatto la somma di due numeri. L'utente dal canto suo doveva tradurre il codice dei led lampeggianti e convertirli mentalmente dalla loro forma ottale in numero decimale. Era la stessa sensazione del primo uomo che mise il piede sulla Luna, una pietra miliare nella storia - la risposta alla domanda che sconcertava l'umanità da secoli: cosa accade quando sommi sei e due? Otto! "Per un ingegnere a cui piacevano i computer quello era un evento esaltante", dice oggi Harry Garland, proprietario di uno dei primi Altair e membro dell'Homebrew computer club, ammettendo però che "sarebbe stato difficile spiegare a un estraneo perché era così esaltante".

Per Steve Dompier era una situazione incredibile che non si fermò lì. Fece piccoli programmi in linguaggio macchina per testare tutte le funzioni dei chip (dovevano essere programmi piccoli, dato che la memoria dell'Altair era minuscola). Andò avanti così finché i suoi dieci "dispositivi di input" - le sue dita - ebbero i calli. Il chip 8080 aveva un set d'istruzioni con ben 72 tipi fondamentali di funzioni, e quindi il lavoro si prevedeva lungo.

Da pilota dilettante, Dompier ascoltava normalmente mentre lavorava una stazione sulle onde corte che mandava le previsioni del tempo. Un giorno dopo aver esaminato un programma per ordinare dei numeri, gli accadde una cosa molto strana quando iniziò a far "girare" il programma: la radio cominciò a emettere dei rumori: zipppp! ziiip! ziiiiiiipppp! Apparentemente sembrava la reazione all'interferenza in radio frequenza causata dallo spostamento dei bit da una locazione a un'altra all'interno dell'Altair. Portò la radio più vicino e lanciò nuovamente il programma. Questa volta gli "zip" erano più forti. Dompier era raggiante: aveva scoperto il primo dispositivo input/output per l'Altair 8800.

Ora il punto era controllare il congegno. Dompier prese la sua chitarra e stabilì che uno dei disturbi che emetteva il computer fosse (all'indirizzo di memoria 075) l'equivalente di un accordo di Fa. Così fece hackeraggio sul programma finché trovò nella memoria le corrispondenti locazioni per le altre note. Dopo circa otto ore aveva completato la mappa della scala musicale e creato un programma per scrivere la musica. Sebbene si trattasse di un programma semplice, niente a che vedere con il raffinato programma musicale di Peter Samson per il Pdp-1, il suo caricamento attraverso quegli interruttori aveva sottratto a Dompier un mucchio di tempo. Ma alla fine approntò la sua interpretazione di *Pool on thè Hill* dei Beatles (il primo pezzo su spartito che aveva trovato) in tempo per il meeting delPHome-brew alla Peninsula School.

L'incontro si teneva in una stanza al secondo piano della scuola, un enorme, vecchio edificio in legno che sembrava uscito da un episodio della *Famiglia Addams.* L'Altair di Dompier, ovviamente, era oggetto di venerazione e lui moriva dalla voglia di far vedere a tutti la prima applicazione documentabile per l'Altair. Ma quando Dompier cercò di accendere l'Altair, si accorse che non funzionava. La presa elettrica non dava segni di vita. La più vicina presa funzionante era al primo piano dell'edificio e, dopo essersi procurato una prolunga della lunghezza sufficiente, Dompier poté attaccare il suo Altair, nonostante il filo fosse appena sufficiente, e la macchina dovesse sporgere un poco fuori dalla porta. Dompier cominciò allora il lungo procedimento di battitura dei bottoni giusti per immettere la canzone in codice ottale, e stava per finire quando due bambini, che stavano giocando nel corridoio, incespicarono nella prolunga elettrica, facendola uscire dalla presa. Ciò provocò la cancellazione di quanto contenuto dalla memoria del computer che Dompier aveva riempito bit dopo bit. Ripeté l'operazione e, alla fine, fu pronto per la prima dimostrazione pubblica di un'applicazione per Altair.

Battè il tasto RUN.

La piccola radio sulla sommità del grosso e minaccioso altoparlante del computer cominciò a gracchiare e a stridere. Era più o meno della musica, e quando si diffusero le prime lamentose battute della ballata di Paul McCartney, la stanza degli hacker - normalmente brulicante di chiacchiere sull'ultimo chip - cadde in un riverente silenzio. Il computer di Steve Dompier, con la pura, tremolante innocenza della prima recita di scuola, stava suonando una canzone. Come finì l'ultima nota, ci fu un assoluto e stupefatto silenzio. Avevano appena avuto la conferma che il sogno condiviso da tutti era una realtà. Un sogno che solo poche settimane prima sembrava vago e lontano.

Bene, prima che potessero riprendersi... *l'Altair riprese a suonare.* Nessuno (eccetto Dompier) era preparato a questo evento, un'interpretazione di *Daisy,* che alcuni di loro sapevano essere stata la prima canzone suonata da un computer, presso i laboratori Bell nel 1957; quell'importante evento della storia del computer stava giungendo per la prima volta alle loro orecchie. Era un bis così inaspettato che sembrava provenisse da una qualche connessione genetica della macchina ai suoi antenati "bestioni" (una nozione evidentemente implicita in 2001 di Kubrick quando il computer Hal, una volta smantellato, regredito allo stadio infantile si mette a "cantare" proprio *Daisy).*

Quando l'Altair ebbe finito, il silenzio non durò a lungo. La stanza si riempì di applausi e congratulazioni, tutti gli hacker saltavano e battevano le mani. La gente dell'Homebrew era un mélange di professionisti, troppo appassionati per considerare l'informatica solo un mestiere, di dilettanti folgorati dalle possibilità della tecnologia e di combattenti tecnoculturali votati al rovesciamento di una società oppressiva in cui governo, business e specialmente l'IBM avevano riservato i computer a una casta disprezzata. Lee Felsenstein ricorda quello come "un gruppo di fuggiaschi, evasi dall'industria, almeno temporaneamente, che per qualche ragione i boss non stavano controllando. Ci mettemmo insieme e cominciammo a sperimentare cose che non facevano scalpore perché non era ciò che stavano facendo le grandi aziende. Ma noi sapevamo che proprio questa era la nostra grande possibilità per realizzarle nel modo in cui noi pensavamo andassero fatte". Per ottenere questo obiettivo ci voleva nientemeno che una completa riscrittura della storia del computer e, in qualche modo, il semplice recital musicale dell'Altair di

Steve Dompier era stato un primo passo.

"Quello è stato uno dei più importanti traguardi della storia del computer" confessa Bob Marsh. Steve Dompier mise per iscritto quell'esperienza, assieme al codice in linguaggio macchina del programma, nel numero successivo di "Pcc" sotto il titolo *Musica, più o meno* e, in seguito, per mesi, i proprietari degli Altair gli telefonavano nel cuore della notte per fargli ascoltare le loro "fughe di Bach".

Dompier ricevette più di quattrocento telefonate di questo tipo. Là fuori c'erano molti più hacker di quanto si potesse immaginare.

Bob Marsh, il disoccupato compagno di garage di Lee Felsenstein, lasciò il primo incontro dell'Homebrew quasi stordito per l'eccitazione dell'evento a cui aveva preso parte in quel buco. Sapeva bene che fino ad allora solo un piccolo numero di persone aveva osato accostarsi al *personal computing.* Ora c'era questo capellone di Steve Dompier che diceva che questa scassata azienda, la Mits, aveva *migliaia* di ordinazioni cui far fronte. Bob Marsh capì all'istante che la confraternita hacker sarebbe cresciuta esponenzialmente negli anni successivi. Ma, come un fuoco incontenibile, aveva bisogno di essere alimentata. I led intermittenti dell'Altair erano stimolanti, ma lui sapeva che - com'è vero che gli hacker sono gli hacker - ci sarebbe stata una richiesta di ogni genere di periferiche, periferiche che questa Mits ovviamente non avrebbe potuto fornire. Quindi *qualcuno* avrebbe dovuto farlo, poiché l'Altair era la base di un fantastico sistema per costruire sistemi e mondi nuovi.

Proprio come il Pdp-1 o il Pdp-6, i computer erano arrivati al MIT come scatole magiche prive di un soddisfacente sistema operativo e, proprio come gli hacker del MIT supplirono a tale mancanza - attraverso assemblatori, programmi di debugging e ogni altra sorta di attrezzi hardware e software per rendere possibile la creazione di nuovi sistemi e qualche applicazione - slmilmente toccherà agli hacker dell'hardware, sebbene meno organizzati, apporre il loro marchio sull'Altair 8800.

Bob Marsh capì di trovarsi all'inizio di una nuova era e di avere un'incredibile opportunità. Seduto sul freddo pavimento del garage di Gordon French, decise che avrebbe progettato e costruito alcune schede di espansione che avrebbe inserito dentro uno degli slot vuoti del bus Altair. Ma non fu l'unico ad aver avuto quell'idea. Infatti proprio là a Palo Alto (la città presso Menlo park, dove s'era tenuto il meeting), due professori della Stanford, Harry Garland e Roger Melen, stavano già lavorando ad alcune schede elettroniche aggiuntive per l'Altair. Loro non sapevano del primo meeting dei fanatici dell'hardware, ma avrebbero partecipato al secondo e, da quel momento, vi avrebbero preso parte regolarmente.

I due docenti erano venuti per la prima volta a conoscenza dell'Altair quando Melen, un alto e robusto signore, il cui umorismo era un po' limitato da una persistente balbuzie, aveva fatto visita a Les Solomon, alla fine del 1974 nell'ufficio newyorchese di "Popular Electronics". Melen e Garland avevano scritto nel loro tempo libero alcuni articoli per gli hobbisti su alcuni progetti e stavano giusto mettendo a punto un articolo che descriveva come costruire un congegno di controllo per una telecamera.

Melen notò una strana scatola sulla scrivania di Solomon e chiese cosa fosse. Solomon lo informò che quella scatola, il prototipo dell'Altair che Ed Roberts aveva mandato per rimpiazzare quello smarritosi durante il trasporto aereo, era un computer da tavolo basato sull'8080 che veniva venduto a meno di quattrocento dollari. Roger Melen non riteneva possibile una cosa del genere e Les Solomon gli disse che se aveva dei dubbi, avrebbe potuto telefonare a Ed Roberts di Albuquerque. Melenge lo fece senza esitazione e organizzò una visita sulla via del ritorno a Ovest.

Il suo scopo era comprare due di quei computer. Inoltre, non aveva ancora percepito il pagamento per un progetto di cui lui e Garland avevano trattato per "Popular Electronics" e che Ed Roberts aveva messo in vendita. Così erano due gli argomenti di cui Melen intendeva parlare con Ed.

Il computer Altair era di gran lunga la cosa più importante - il giocattolo giusto al momento giusto, pensava Melen - ed era così eccitato dalla prospettiva di possederne uno, che quella notte non dormì. Quando finalmente raggiunse la modesta sede della Mits, fu deluso dal vedere che non *c'erano* Altair pronti da portarsi a casa. Ma Ed Roberts era un individuo affascinante, un ingegnere nato, che aveva una visione lucida delle cose. Discussero fino alle cinque del pomeriggio dei dettagli tecnici di quella visione. Tutto questo accadde prima che il famoso articolo uscisse su "Popular Electronics", nonostante Ed Roberts fosse preoccupato su quale tipo di risposta avrebbe potuto avere. Calcolò che non sarebbe stato male avere a disposizione qualcuno che costruisse le schede da inserire nell'Altair e s'accordò per mandare a Melen e a Garland un primo prototipo, in modo che potessero creare qualche aggeggio per connettere una telecamera alla macchina e poi una scheda per avere in uscita anche una buona immagine video.

Garland e Melen divennero quindi soci e chiamarono la loro azienda Cromem-co, in onore del pensionato studentesco della Stanford in cui un tempo avevano vissuto, il Crowthers Memorial. Erano ben lieti di aver trovato personaggi del loro stesso livello all'Homebrew club, tra cui Marsh, il quale gli aveva parlato del suo amico Gary Ingram che lo stava aiutando a metter su un'azienda chiamata Processor technology.

Marsh sapeva che la necessità più impellente per il proprietario di un Altair era una memoria più grande dei miseri 256 byte che erano in dotazione alla macchina, e così pensò di progettare una scheda che avrebbe portato la memoria a 2K (ogni K equivale a 1024 byte). La Mits aveva annunciato le sue schede di memoria e ne aveva anche spedita qualcuna ai clienti. Erano ottime schede e anche molto carine, ma non funzionavano. Marsh prese a prestito l'Altair della Pcc e lo esaminò attentamente, leggendo il manuale con la massima attenzione. Questo era necessario perché all'inizio non poteva permettersi di spendere denaro per fare delle fotocopie. Pensò di mandare avanti la ditta nello stesso modo in cui Roberts mandava avanti la Mits: annunciando prima il prodotto e raccogliendo la somma necessaria per poi progettarlo e costruirlo in serie.

Il primo di aprile perciò, Marsh e Ingram, un introverso ingegnere che non era mai stato alle riunioni dell'Homebrew ("Non era il genere di cose che faceva", confessa Marsh), inaugurarono ufficialmente l'azienda. Marsh fu capace di raggranellare abbastanza denaro da fotocopiare una cinquantina di fogli che spiegavano la linea dei prodotti proposti. Il 2 aprile Marsh presenziò al terzo meeting dell'Homebrew distribuendo i volantini e annunciando un 20 percento di sconto a quelli che avessero ordinato anticipatamente. Dopo una settimana non si era fatto ancora vivo nessuno. Marsh racconta: "Cominciò a prendermi la disperazione. Avevamo sbagliato, non avrebbe funzionato. Poi venne la nostra prima ordinazione, una scheda per [memoria] Rom. Un ordine d'acquisto con 'pagamento a 30 giorni' da questa azienda chiamata Cromemco. Pensammo 'Chi è questa Cromemco? Perché non pagano in contanti?' La disperazione continuava. *Ancora non si decolla!* Il giorno successivo arrivarono tre ordinazioni e, nel giro di una settimana, avevamo 2.500 dollari in cassa. Ne prendemmo mille, per un piccolo annuncio pubblicitario su "Popular Electronics", e poi la situazione si sbloccò. Ci vollero solo due mesi per raggiungere centomila dollari d'ordinazioni".

L'ironia era che Marsh e le altre ditte gestite da hacker non erano tagliate per i grandi affari. Cercavano solo un modo per finanziare la propria aspirazione di giocare con l'elettronica ed esplorare questo nuovo regno di piccoli computer. Per Marsh e gli altri che avevano lasciato le prime riunioni dell'Homebrew con il fervore di mettersi a costruire schede elettroniche, stava cominciando la festa: progettare e costruire il materiale, esprimere se stessi con i contorcimenti e i grovigli delle piastre per circuiti logici integrati digitali da attaccare al bus bizantino di Ed Roberts.

Come Marsh scoprì, costruire una scheda per l'Altair per un hacker dell'Homebrew era l'equivalente del tentare di scrivere un grande romanzo. Era qualcosa che i duri recensori dell'Homebrew avrebbero

esaminato con attenzione, e non avrebbero soltanto osservato se funzionava o meno, ma avrebbero anche giudicato la relativa eleganza e la solidità dell'architettura. La disposizione dei circuiti sulla scheda era una finestra aperta sulla personalità del progettista e perfino dei dettagli superficiali come la fattura dei connettori attraverso cui si montava la scheda avrebbe messo in luce le motivazioni, la filosofia e la dedizione al prodotto. Gli schemi digitali, come i programmi, "sono le migliori immagini della mente che si possano trovare", dice Felsenstein. "Posso dirti cose di certe persone dal progetto dell'hardware che vedo. Se guardi bene qualcosa puoi arrivare a dire: 'Cristo, questo tizio progetta come un verme. Va da un punto sino alla fine senza sapere cosa ha fatto nel mezzo'."

Bob Marsh voleva che la Processor technology fosse conosciuta per la qualità dei suoi prodotti e passò i mesi successivi in un costante logorio, cercando non solo di finire i suoi progetti, ma di realizzarli con cura. Ciò era importante per l'azienda come pure per la stima di se stesso.

Il procedimento non era dei più semplici. Dopo aver deciso la funzionalità della scheda, si passavano lunghe notti a progettarla. Consultando il manuale che descriveva il funzionamento del chip 8080 si annotavano i numeri delle varie sezioni desiderate, progettando questa sezione per l'input, quella per la memoria, e la griglia labirintica che stava dentro a quel pezzo di plastica nera cominciava a prendere forma nella mente. L'efficacia della scelta sulle sezioni a cui dedicarsi dipendeva da quanto bene e accuratamente veniva conservata quella visione. Si faceva uno schizzo a matita di quelle connessioni, ciò che era destinato a stare su una parte della scheda in blu, e quello sull'altra faccia, in rosso. Quindi si prendevano i fogli di Mylar, stendendoli su un tavolo luminoso con una griglia, e si cominciavano a tracciare i profili delle connessioni con del nastro di carta. Si poteva poi scoprire che gli schemi avevano qualcosa che non andava - troppa confusione da una parte o interconnessioni troppo vicine - ed era necessario correggere qualcosa. Un errore poteva far saltare tutto. Ci si doveva perciò assicurare di fare un altro schema da sovrapporre: collocandolo sopra il progetto con i nastri di carta si poteva vedere se si era commesso un grave errore, come connettere tre cose insieme. Se lo schema era sbagliato, lo si accantonava.

Si doveva creare il progetto in modo che la scheda avesse molti strati: ciò creava un differente insieme di connessioni sulla superficie e sul fondo.

Il progetto, durante il lavoro, doveva essere spostato su e giù, e talvolta il nastro si staccava, oppure c'erano dei piccoli pezzi di nastro penzolanti, oppure ancora un

capello s'era incollato da qualche parte: uno qualsiasi di questi fenomeni indesiderati sarebbe stato fedelmente riprodotto dall'eliocopia (se non si avevano abbastanza soldi, ci si doveva accontentare di un'accurata fotocopia) per poi finire in un disastroso corto circuito. Si doveva infine contrassegnare il progetto per l'azienda produttrice della scheda, indicando dove perforare, dove fosse necessaria la placcatura in oro e così via.

Alla fine, si doveva andare alla fabbrica di schede più vicina con gli schemi in mano e gliele si consegnavano. Dato che c'era la crisi, le ditte erano contente di aver ottenuto un lavoro, perfino se commissionato da un trasandato hacker dell'hardware con gli occhi da sballato. Mettevano quella roba su un *digitizer,* perforavano i buchi e, alla fine, ottenevano una base verdastra di epoxy con sopra un groviglio d'interconnessioni argentee.

Questo era il metodo di lusso. Bob Marsh infatti all'inizio non si poteva permettere di lavorare in quel modo: incideva a mano la scheda sul fornello di cucina, usando limatura da circuiti stampati, tracciando linee appena visibili, in cui il materiale si liquefava. Quel metodo era un contorto corteggiamento della dea Catastrofe, ma Marsh era un lavoratore indefesso e oculato. Spiega più tardi: "Ero veramente preso da quel lavoro. Divenni un tutt'uno con il mio progetto di scheda".

Per la prima scheda di memoria, Marsh fu messo sotto pressione in modo particolare. Ogni benedetta settimana ai meeting dell'Homebrew o tutti i giorni al telefono, i clienti furibondi reclamavano affannosamente la propria scheda di memoria come fossero palombari che boccheggiassero per carenza d'aria. Marsh così ricorda i loro lamenti: "Quand'è pronta la mia scheda? Ne ho bisogno. *Devo averla assolutamente".*

Finalmente Marsh la finì. Non c'era tempo per farne un prototipo. Aveva la sua scheda, che era un rettangolo in epoxy verde con una piccola protuberanza di connettori dorati incisi sotto, delle dimensioni giuste per essere inserita nello slot del bus Altair. Aveva i chip e i cavi che i costruttori del kit vi avrebbero saldato sopra (in un primo tempo la Processor tech avrebbe venduto solo schede da assemblare). Marsh aveva già tutto pronto, ma non aveva l'Altair su cui fare il test. Quindi, incurante del fatto che fossero le tre di notte, chiamò il buon Dompier che aveva conosciuto all'Homebrew e gli disse di portargli lì la macchina. L'Altair aveva per Dompier la stessa importanza di un figlio, se non fosse stato scapolo, e così lo avvolse con tutte le attenzioni in una coperta rossa e lo portò via in braccio. Dompier nel-l'assemblare la macchina aveva seguito il manuale, indossando perfino un braccialetto di rame al polso quando saldava (per scaricare l'elettricità statica) e stando attento a non toccare il fragile cuore 8080 della macchina. Naturalmente si prese un colpo quando, dopo aver amorevolmente posato la macchina nel laboratorio di Marsh, i due veterani dell'hardware, Marsh e Ingram, cominciarono a smanettare sui chip come un paio di meccanici che stessero installando una marmitta: li afferravano con le loro dita sudice, li estraevano, poi li riponevano, infine li mettevano di nuovo dentro. Dompier osservava terrorizzato. Alla fine la scheda fu pronta, e Ingram girò l'interruttore e fu allora che il prezioso computer di Steve Dompier cadde in uno stato comatoso: avevano inserito la scheda alla rovescia.

Ci volle un giorno per aggiustare l'Altair, ma Steve Dompier non si arrabbiò: anzi avrebbe prestato la sua macchina alla Processor technology per altri test di prova. Ciò era indicativo del comportamento dei membri dell'Homebrew. Erano questi una razza diversa di hacker dagli inawicinabili maghi del MIT, ma erano anche rispettosi dell'etica hacker che subordinava la proprietà e l'individualismo al bene comune, il che, in pratica, voleva dire che avrebbero aiutato la gente a fare hacke-raggio nel modo più efficace. Steve Dompier era nervoso a proposito del suo Altair, ma non desiderava altro al mondo che una scheda di memoria, per poter caricare qualche vero programma. E poi voleva dispositivi di i/o e dispositivi per la visualizzazione... per poter scrivere utility e rendere più potente la macchina. Erano *strumenti per fare strumenti,* per penetrare in profondità nel mondo che ruotava attorno al misterioso microprocessore 8080. Bob Marsh e gli altri dell'Homebrew, sia che stessero creando prodotti per gli altri, sia che semplicemente fossero hacker curiosi come lui, in ogni caso procedevano uniti, e insieme formarono una comunità anche se questa non era centrata geograficamente come quella del MIT intorno al Pdp-6. L'Homebrew nonostante si estendesse da Sacramento a San José, si teneva saldamente unita.

Quando, ai primi di giugno, Bob Marsh si fece vedere a un incontro dell'Homebrew con il primo carico di schede, la gente che le aveva ordinate fu così grata che si poteva credere che gliele stesse regalando. Consegnava i piccoli pacchetti di plastica contenenti le schede e i circuiti integrati, insieme al manuale di istruzioni scritto da Lee Felsenstein. "Se non sei un esperto costruttore," avvisava Lee, "non metterti a costruire questi kit".

C'era veramente poca esperienza al mondo per costruire quel genere di roba, ma gran parte dell'esperienza era concentrata in quella sala, che era ora l'auditorium dello Stanford linear accelerator (Slac). Erano passati solo quattro mesi dopo il primo fortuito incontro del club e i suoi membri erano diventati già quasi dieci volte tanto.

Il piccolo club formato da Fred Moore e Gordon French era diventato qualcosa che nessuno dei due avrebbe mai potuto immaginare. Era l'avanguardia di una razza di hacker dell'hardware che si stavano "bootstrappando" dentro una nuova industria che, ne erano sicuri, sarebbe stata diversa da ogni industria

precedente. L'industria dei computer da tavolo sarebbe stata influenzata dall'etica hacker. (Il termine *bootstrap* era significativo del nuovo gergo parlato da questi hacker: la parola letteralmente descrive il processo per cui il sistema operativo carica se stesso quando la macchina si accende per la prima volta, ovvero fa il *boot.* Una parte del programma carica il codice nella memoria del computer; questo codice istruisce la macchina a caricare il codice che ancora manca. Proprio come quando, dopo un piccolo stimolo, una persona riesce con le proprie forze a terminare un'impresa. E quello che avevano fatto i membri dell'Homebrew: si erano creati una nicchia nel mondo dei computer da tavolo, poi avevano scavato in profondità facendo della nicchia una caverna e un'abitazione permanente).

Ma i fondatori del club vennero entrambi spesso lasciati indietro dall'abilità tecnica della gente intorno a loro. French sembrava soffrire di quello che sembrava essere un latente atteggiamento burocratico. Per certi aspetti, la sua mania di far procedere il club in maniera ordinata e controllata fu senz'altro d'aiuto. Fungeva da segretario e bibliotecario, tenendo la lista dei numeri di telefono di tutti e quale attrezzatura avessero. Come poi rammenterà: "Il mio telefono non la smetteva di suonare. Era incredibile. Tutti cercavano informazioni e tutti avevano bisogno di tutti per andare avanti, perché c'era assoluta scarsità di attrezzatura. Un esempio: 'Dato che hai un terminale, potresti prestarmelo per un paio di giorni che ci metto dentro il programma che gestirà il mio lettore di nastro di carta perforata?' E cose del genere".

Per altri aspetti e particolarmente per il modo in cui moderava gli incontri, lo stile di French non si addiceva allo spirito hacker che fermentava nell'Homebrew. "Gordon era un tipo didattico", ricorda Felsenstein. "Cercava di portare la discussione dove voleva lui. Voleva che fosse un evento educativo, voleva consigliare letture, insegnare alla gente certe cose, specialmente quelle di cui era esperto. Si irritava moltissimo se la discussione si limitava a una trasmissione superficiale delle conoscenze. Si intrometteva invece nella conversazione delle persone quale che fosse l'argomento, inserendo le sue opinioni e dicendo loro: 'C'è un punto importante che avete tralasciato e io in questo genere di cose ne so molto di più'." Dopo la prima parte dell'incontro, in cui la gente si presentava e raccontava quel che stava facendo, Gordon si alzava in piedi in fondo alla stanza e cominciava un discorso che aveva tutta l'aria di una lezione, illustrando il modo in cui la macchina usa il codice caricato, e informando tutti gli altri irrequieti membri di come il fatto di imparare bene il funzionamento del codice li avrebbe salvati da preoccupazioni future... prima o poi la gente si seccò a tal punto da andarsene in silenzio dalle riunioni e cominciare a scambiarsi informazioni nel corridoio. Era una situazione delicata, quel genere di complicato dilemma umano che normalmente gli hacker odiano affrontare. Ma si faceva strada la necessità che un nuovo moderatore assumesse il controllo della situazione.

La scelta logica sarebbe ricaduta su Fred Moore, che sedeva anch'egli in fondo alla stanza durante i primi mesi dell'Homebrew con il suo registratore e il suo blocco di appunti, sintetizzando i contenuti della riunione in modo da pubblicarne i punti salienti sulla newsletter mensile. Egli dedicava una gran quantità del suo tempo al gruppo, perché vedeva che gli hacker e i loro Altair erano sul punto di trasformarsi in una significativa forza sociale. "Attraverso la condivisione della nostra esperienza e scambiandoci i consigli, noi facciamo progredire quest'arte e rendiamo possibile alla gente avere computer a costi più bassi" scriveva nella newsletter, aggiungendo poi il suo commento politico, "È evidentissimo che la gente vuole i computer, probabilmente per il proprio divertimento o per uso educativo. Perché le grandi industrie trascurano questo mercato? Sono occupate a vendersi le une con le altre (e al governo e ai militari) macchine dai costi gonfiati. Non vogliono vendere direttamente al pubblico. Sono molto contento del successo che la Mits sta avendo con l'Altair perché ciò determinerà tre cose: f) sveglierà le altre aziende sulla domanda di computer a basso costo nelle abitazioni; 2) provocherà la formazione di computer club e di gruppi di appassionati che riempiranno il vuoto delle conoscenze tecnologiche; 3) contribuirà a smitizzare l'immagine dei computer..."

Moore individuava esplicitamente l'obiettivo del club nello scambio delle informazioni. Come un inarrestabile flusso di bit all'interno di un computer perfettamente progettato, l'informazione fluiva liberamente tra gli associati dell'Homebrew. "Più di ogni altro individuo, Fred Moore sapeva cosa significava condividere" ricorda Gordon French. "Quella era una delle espressioni che lui usava sempre: condivisione, condivisione, condivisione."

Ma la maggioranza del club preferì seguire un percorso che divergeva da quello di Moore. Fred era del resto alle prese con le applicazioni. Sin dai primi incontri aveva sollecitato i membri di questo gruppo fondamentalmente anarchico a mettersi insieme per fare qualcosa, sebbene poi fosse abbastanza vago su cosa potesse essere questo qualcosa. Forse usare i computer per aiutare gli handicappati, forse compilare liste d'indirizzi postali per sostenere la resistenza alla chiamata di leva. Moore aveva visto giusto nel percepire che il dinamismo del club era in qualche modo di natura politica, ma la sua visione sembrava cozzare con il fatto che gli hacker generalmente non si propongono la trasformazione sociale: gli hacker fanno gli hacker. Moore era più attirato dall'idea di fondare un sistema sociale democratico e basato sulla condivisione piuttosto che dalle manipolazioni dei sistemi operativi; egli sembrava guardare all'Homebrew non come a una fortezza tecnologica di persone determinate all'edificazione di un potere di computer casalinghi, ma come a uno schema di rivoluzione popolare, del tipo della resistenza alla leva o dei gruppi antinucleari in cui era stato coinvolto. Aveva suggerito di vendere torte per incrementare i fondi del gruppo, oppure di pubblicare sulla newsletter delle poesiole tipo "Non lamentarti, non agitarti/ dipende da ognuno di noi/ costruire il club/ per fare quel che vuoi tu". Intanto, molti membri del club saltavano alla pagina successiva della newsletter per studiare gli schemi illustrati nell'articolo dal titolo *Generazione di funzioni logiche arbitrarie attraverso i multiplexer digitali*. *Quello* era il modo per cambiare il mondo, e molto più divertente di una vendita di torte.

Lee Felsenstein poi non credeva che Moore "perseguisse più di tanto i suoi intenti politici. A livello superficiale era rimasto fermo alla protesta e al rituale della protesta. Ma noi eravamo molto più interessati a ciò che si potrebbe chiamare la 'propaganda del fatto concreto'".

Così quando fortuitamente si presentò un'occasione per rendere gli incontri più compatibili allo spirito libero degli hacker - Gordon French, che faceva consulenze per la Social security administration, era stato temporaneamente chiamato a Bal-timora - non fu Moore quello a cui la gran parte dei membri del club si rivolsero perché facesse da moderatore, ma a Lee Felsenstein. Egli dimostrò di essere la scelta ideale, dato che era non solo capace quanto gli altri come hacker dell'hardware, ma era anche un informatico politicizzato. Considerava l'incarico di moderare questi meeting come un importante aumento di responsabilità e poteva essere il punto di riferimento della rivoluzione sul fronte dell'hardware, permettendo alle riunioni di proseguire con la giusta miscela di anarchia e ordine, continuando la sua personale guerriglia fatta di schemi per progetti hardware che si sarebbe conclusa con il trionfo del Tom Swift terminal, e contribuendo alla resurrezione del progetto della Community memory - un processo che stava cominciando proprio quell'estate con la pubblicazione del periodico ciclostilato intitolato "Journal of Community Com-munication", che avrebbe diffuso il concetto di computer da tavolo "creati e usati dalla gente nella vita di tutti i giorni in quanto membri di una comunità".

Quando, comunque, per la prima volta si alzò nella stanza del meeting dell'Homebrew, in quel giugno del 1975, era terrorizzato. Come ricorderà, qualcuno chiese chi fosse il nuovo moderatore e Marty Spergel, "the Junk Man", il proprietario della M&R Electronics, suggerì Lee e allora "ci fu un'ovazione". Fu come un'incoronazione. Apprensivo com'era, era una possibilità che non poteva perdere. Come al solito per

Lee, i rischi di fallimento erano meno pericolosi dei rischi derivanti dal non tentare affatto.

Sapeva qualcosa di come si dirige un'assemblea. Durante il suo periodo di studente radicale nel 1968, aveva ascoltato un "microfono aperto" di una radio di Berkeley che era così malamente organizzato, con le telefonate degli ascoltatori che non si sentivano e con disturbi di vario genere, tanto che era piombato nello studio brandendo la sua radio portatile e dicendo: "Ascoltate qua, idioti!". Alla fine divenne un collaboratore della trasmissione, e una parte del suo ruolo era di preparare gli ospiti prima che andassero in onda. Pensava che la sua funzione all'Homebrew potesse attingere da quell'esperienza; sollecitava le persone non abituate a rivolgersi a un pubblico più vasto di un tavolo pieno di componenti elettronici e a parlare con altri esseri umani di ciò che era di loro interesse. Come aveva ben rilevato Fred Moore, questo era il cuore del meeting, lo scambio delle informazioni. Così Lee, creando un'architettura per le riunioni come se stesse affrontando un problema di progettazione elettronica, iniziò a incasellare la discussione. Ci sarebbe stato un momento per andare in giro tra la gente della stanza a sentire a cosa stessero lavorando o cosa volessero sapere: quella sarebbe stata la sessione di *mapping,* simile al disegnare uno schema. Poi ci sarebbe stata una sessione "ad accesso casuale", dove si potevano dare o ricevere consigli su tematiche specifiche, oppure generiche, sia per avere le informazioni richieste, sia perché semplicemente interessava parlarne. Dopo di ciò, c'era talvolta un breve discorso, per esempio qualcuno che illustrava un sistema oppure mostrava un nuovo prodotto, e poi ci sarebbe stato ancora *mapping* e poi ancora una sessione ad accesso casuale. Quando Lee si accorse che la gente era riluttante a tornare dalla prima sessione ad accesso casuale -certe volte ti potevi perdere in discussioni tecniche o in qualche problema religioso come una tecnica per fare wire-wrapping' sulla scheda o qualche altra cosa - cambiò il programma includendo una sola sessione ad accesso casuale alla fine della riunione. Così corretta la struttura lavorava.

Lee scoprì che stare davanti a un gruppo di persone che lo accettavano e apprezzavano il suo ruolo di *"stack pointer"* - la funzione del computer che determina l'ordine in cui devono essere eseguite le istruzioni - aiutava il suo rafforzamento psicologico nell'uscire dal suo guscio. Dopo essersi appropriato delle qualità di moderatore, prese sufficiente confidenza per pronunciare di fronte al gruppo un discorso sul suo Toni Swift terminal; scarabocchiando sulla lavagna del piccolo auditorium dello Slac, parlava di display video, affidabilità dell'hardware, di Ivan Illich e dell'idea di prendere in considerazione l'utente nella progettazione. Questa fu una buona miscela di politica sociale ed esoterismo tecnologico, e i membri dell'Homebrew la apprezzarono. Lee si scoprì un talento per la battuta pronta e, in effetti, mise a punto un piccolo rituale all'inizio di ogni meeting. Cominciò a tenere un portamento orgoglioso di maestro di cerimonia del club: nella sua mente egli era il direttore del circo hacker, un gruppo che era cruciale nello sviluppo di una modalità di vita "a microprocessore".

Non molto tempo dopo l'assunzione del controllo da parte di Lee, un Fred Moore pieno di problemi rassegnò le dimissioni da tesoriere, segretario e direttore della newsletter. Aveva infatti anche dei guai di carattere personale: la donna con cui viveva l'aveva lasciato. Era un brutto momento per andarsene: sentiva che il club sarebbe stata la sua eredità, in un certo senso, ma era anche chiaro che le sue speranze che il club lavorasse a fini comunitari erano vane. Al suo posto s'era affermata invece la "propaganda del fatto concreto" e, ancora peggio, alcune persone venivano ai meeting, Fred ricorda, "con i dollari segnati negli occhi e dicevano: 'Ehi, qui c'è da fare una nuova industria. La metterò su io... produrrò queste schede e farò i milioni...'". C'erano altri obiettivi sociali legati ai computer che Moore voleva perseguire, ma aveva capito, come spiegherà in seguito, che "la gente del club era veramente più avanti [di lui] per le conoscenze d'elettronica o d'informatica, [e a causa di questo] s'innamorava di molte di quelle macchine, macchine che erano molto seducenti". Così Fred era infelice per come la gente accettava ciecamente la tecnologia. Qualcuno aveva raccontato a Fred della forza lavoro femminile a basso costo in Malesia o in altri paesi asiatici che fisicamente assemblava quei magici chip. Aveva sentito che le donne asiatiche venivano pagate con salari da lame, lavorando in fabbriche insicure, e che nemmeno sarebbero potute ritornare a vivere ai loro villaggi, poiché non conoscevano più le tradizionali forme di relazione sociale. Sentiva che avrebbe dovuto dirlo al club, che doveva forzare l'argomento, ma poi capiva che non era il tipo di problema che all'Homebrew club poteva interessare.

Ciò nonostante, amava il club e quando i suoi problemi personali lo costrinsero a piegarsi e ad andarsene, disse: "è uno dei giorni più tristi della mia vita". Una piccola e meditativa figura si alzò e andò alla lavagna in un incontro di metà agosto e scrisse i suoi compiti da coprire, chiedendo chi avrebbe fatto la newsletter, chi avrebbe tenuto la cassa, chi avrebbe preso gli appunti... E qualcuno si alzò e iniziò a scrivere "Fred Moore" a fianco di ogni richiesta. Gli si ruppe il cuore, sentiva che per lui era finita e, anche se non poteva metterli al corrente di tutte le sue ragioni, allo stesso tempo doveva comunicare ai suoi fratelli che non sarebbe più tornato.

"Vedo me stesso come una persona che ha aiutato questa gente a tenersi unita e a mettere in comune le proprie capacità e la propria energia", racconta Moore. E quegli obiettivi erano stati raggiunti. In effetti ogni meeting sembrava crepitare per lo spirito e l'eccitazione mentre la gente si scambiava chiacchiere e chip, proiettando se stessa in questo nuovo mondo. Nel momento del *mapping,* la gente s'alzava e diceva che aveva questo o quel problema nell'assemblare l'Altair e Lee chiedeva: "Chi può aiutare questo tizio?" e tre o quattro mani si alzavano. Meraviglioso. E poi? E poi qualcuno avrebbe detto di aver bisogno di un chip 1702. Qualcun altro di avere un ben più potente 6500, e ne sarebbe nato uno scambio.

Poi c'erano quelli che si alzavano per riportare le ultime notizie da Silicon Valley. Jim Warren, un tozzo laureando in informatica alla Stanford, era un commerciante di chiacchiere ben documentato che saltava su nel periodo della sessione ad accesso casuale e proseguiva per dieci minuti a parlare di questa o di quell'azienda, sovente infilandoci qualche visione personale sul futuro delle comunicazioni computerizzate attraverso la trasmissione digitale.

Un altro famoso approvvigionatore di questa strana forma di pettegolezzi era Dan Sokol, un ingegnere alle prime armi, che lavorava come collaudatore di sistemi in una delle più grandi aziende della Valley. Le sue intuizioni, il più delle volte, erano anticipatorie delle ricerche scientifiche (per mantenersi questa reputazione, ammette oggi Sokol, falsificava metà delle voci sul suo conto). Dan, un discepolo digitale capellone e barbuto che si era lanciato nell'I lomebrew con le energie dei nuovi convcrtiti, aderì subito all'etica hacker. Riteneva che non ci fossero informazioni che non meritassero di essere diffuse, e più importante era il segreto, più grande era il suo piacere nel rivelarlo. "C'è qualcuno qui della Intel?" chiedeva e, se non c'era nessuno, divulgava le ultime sul chip che Intel aveva fino a quel momento protetto dallo spionaggio delle altre aziende della Valley (e forse anche da qualche spia russa).

Talvolta Sokol, incallito propugnatore del baratto, infilava la mano in tasca e ne tirava fuori il prototipo di un chip. Per esempio, un giorno mentre era al lavoro, ricorda, entrarono dei signori di una nuova azienda chiamata Atari a collaudare alcuni chip. Agivano in modo assai circospetto e non dicevano di che razza di chip si trattasse. Sokol li esaminò con attenzione: alcuni portavano il marchio Syntech, altri Ami. Sokol aveva conoscenze in entrambe le aziende e loro gli dissero che erano dei pezzi fatti su ordinazione, ideati e progettati da personale Atari. Se ne portò uno a casa, lo pose su una scheda e lo collaudò. Il chip risultò contenere un programma per giocare un nuovo videogame, *Pong.* Infatti la nuova ditta Atari aveva da poco cominciato a mettere insieme un apparecchio domestico per fare quel gioco, in cui due persone avevano il controllo di due "palette" di luce collocate su uno schermo Tv e cercavano di mantenere in gioco una

"palla", o meglio una specie di pallina intermittente. Sokol riprodusse lo schema su una piastra, la portò all'Homebrew e la fece vedere. Prese con sé anche qualche chip in più, e li scambiò con altri chip, una tastiera e alcuni chip di Ram. "Si trattava di un furto vero e proprio", spiega oggi; ma dal punto di vista dell'Homebrew, Sokol stava liberando un valido hack dall'oppressione della proprietà. *Pong* era qualcosa di valido e doveva appartenere a tutti. E, all'interno dell'Homebrew, scambi come quelli erano liberi e abituali.

Anni prima, Buckminster Fuller aveva elaborato il concetto di sinergia - il potere del collettivo superiore alla somma delle parti, derivante da persone e/o fenomeni che operino assieme in un sistema - e l'Homebrew sarebbe stato un esempio da manuale della realizzazione di questo principio. L'idea di una certa persona avrebbe stimolato un'altra persona a imbarcarsi in un progetto più vasto, e forse a fondare una società per costruire un prodotto basato su quell'idea. Oppure, se qualcuno saltava fuori con qualche hack particolarmente intelligente per produrre un generatore di numeri casuali per l'Altair, avrebbe ceduto il codice in modo tale che ognuno potesse utilizzarlo e alla riunione successiva qualcun altro avrebbe escogitato un gioco che impiegava quella routine.

La sinergia continuava perfino dopo il meeting, dato che alcuni membri dell'Homebrew avrebbero proseguito le loro discussioni fino a mezzanotte all'Oasis, un buco di bar sempre pieno di fumo vicino al campus. (Il luogo era stato suggerito da Roger Melen; Jim Warren, un accanito non fumatore, tentò di dirottare il gruppo verso una sezione per non fumatori al Village Host', ma non fu mai ascoltato). Pigiati dentro questa baracca di legno i cui tavoli erano stati profondamente incisi con le iniziali di generazioni di studenti, Garland, Melen, Marsh, Felsenstein, Dompier, French e chiunque altro se la fosse sentita, erano incoraggiati a mettere in mostra le proprie idee per via dell'energia della riunione e dei boccali di birra. Intravedevano sviluppi così fantastici, che nessuno mai pensò che potessero essere più che fantasia, vaneggiando del giorno in cui i monitor degli nome computer avrebbero riprodotto programmi pornografici - *Smut-Roms,* li chiamavano - che non sarebbero stati illegali poiché "diventavano pornografici" solo nel momento in cui venivano inseriti nel computer. Come poteva il mero codice essere pornografico? Questa era solo una delle decine di riflessioni altamente improbabili che sarebbero state fortemente smentite nel giro di pochi anni.

Sinergia: Marty Spergel, "the Junk Man", sapeva esattamente come funzionava. Un uomo abbronzato di mezz'età, parsimonioso e con un largo sorriso disarmante, pensava che l'Homebrew fosse come "una piccola comunità boy scout, dove ciascuno aiuta l'altro. Ricordo di aver avuto dei problemi con la telescrivente del mio ufficio e un tizio [dell'Homebrew] disse che l'avrebbe controllata. Non solo lo fece, ma venne con un piccolo kit e vi aggiunse quattro o cinque pezzi differenti, li lubrificò, li montò, aggiustò tutte le leve. Io allora dissi: 'Quanto ti devo?'. E lui: 'Nulla'". Per "the Junk Man", quella era la vera essenza dell'Homebrew.

Spergel teneva sempre conto dei pezzi di cui la gente aveva bisogno; capitava che certe volte ne portasse qualche scatola in occasione degli incontri. Da quando si cominciò a parlare del Tom Swift terminal, aveva chiesto a Lee se volesse costruirne uno per l'azienda di Spergel, la M&R Electronics. "Be', lo Swift terminal non è pronto," disse Lee, "ma che ne direbbe di un progetto per un modem?" Il modem era un congegno che abilitava i computer a comunicare sulle linee telefoniche che Lee aveva concepito un paio d'anni prima. "Lui probabilmente sapeva cosa fosse un modem, comunque la sua reazione non lo lasciò trapelare" racconta Lee. I modem venivano venduti a un prezzo che oscillava tra i quattrocento e i seicento dollari, ma Marty fu in grado di costruire il modem "Pennywhistle", sulla base del bel progetto di Lee, e venderlo a soli 109 dollari. Spedirono una copia degli schemi a Les Solomon a "Popular Electronics" che mise una foto del modem di Lee sulla copertina.

Sinergia. Il crescente numero di membri dell'Homebrew che stavano progettando o regalando addirittura nuovi aggeggi, dai joystick ai dispositivi di i/o per gli Altair, usava il club come una fonte di idee, ma vi si rivolgeva anche per le prime ordinazioni o per eseguire dei beta test per i prototipi. Se uno aveva appena realizzato un prodotto poteva portarlo al club e sottoporlo alle più spietate critiche disponibili sul mercato; poteva poi divulgare le specifiche tecniche e gli schemi; se invece si fosse trattato di software, allora divulgava il codice sorgente. Tutti avrebbero potuto imparare e migliorarlo, se questo incontrava il loro interesse e se ne erano abbastanza capaci.

S'era instaurato un clima frizzante che funzionava così bene perché, conformemente all'etica hacker, non veniva imposto alcun confine artificiale. In effetti ogni principio di quell'etica, formulata dagli hacker del MIT, era messa in pratica in una qualche misura all'interno dell'Homebrew. Le esplorazioni e la pratica erano riconosciute come valori cardinali; le informazioni acquisite in queste ricerche e in questi progetti avventurosi erano liberamente diffuse persino a potenziali concorrenti (l'idea di concorrenza giunse assai lentamente in queste nuove ditte, dato che ci si sforzava di creare una versione hacker dell'industria, un compito che doveva vedere tutti collaborare insieme); erano banditi ruoli o atteggiamenti autoritari, le persone ritenevano che i personal computer fossero gli ambasciatori più significativi della decentralizzazione; la qualifica di membro del club era aperta a chiunque capitasse lì; la stima la si guadagnava con le capacità o le buone idee e non era difficile trovare un diciassettenne che conversava da pari con un ricco ed esperto ingegnere di mezz'età; c'era un forte grado di apprezzamento per la raffinatezza tecnologica e l'estro digitale e, soprattutto, questi hacker dell'hardware stavano vedendo, in maniera populista e vibrantemente alternativa, come i computer potessero cambiare la vita. Si trattava di macchine a buon mercato che sapevano che sarebbero divenute effettivamente utili di lì a pochi anni.

Questo, naturalmente, non li esimeva dall'immergersi totalmente nell'hackeraggio con queste macchine per il piacere dell'hackeraggio in sé, per impadronirsi dello strumento, per fare ricerca o per sognare. Le loro esistenze erano orientate al momento in cui avrebbero progettato la scheda per un congegno, oppure avrebbero connesso i cavi al bus, oppure ancora al momento in cui il programma da loro elaborato sarebbe stato lanciato per la prima volta... Una persona definì poi quel momento come simile a una locomotiva che faceva retromarcia su tratto di strada ferrata da te appena terminato, e scorresse sopra quel tratto alla velocità di 150 chilometri all'ora. Se il tuo binario non era abbastanza robusto, il treno sarebbe deragliato rovinosamente... fumo... fuoco... metallo contorto. Ma se tu lo avevi hackerato bene, il treno si sarebbe esibito in una corsa travolgente. Ti saresti esaltato per la realizzazione di migliaia di calcoli al secondo che ti sarebbero stati spalancati davanti da quel pezzo di attrezzatura che portava il tuo marchio. Tu, il padrone dell'informazione e legislatore di un mondo nuovo.

Dei planner si recarono in visita all'Homebrew *e* furono sviati dalla ferocia delle discussioni tecniche, dall'intensa fiamma prodotta nel momento in cui la gente si proiettava verso la meta hacker della costruzione. Ted Nelson, autore di *Computer Lib,* venne al meeting e ne trasse un'impressione confusa. Ricorda che la gente del-l'Homebrew vestiva in maniera trasandata, era sempre spettinata, "dei monaci del chip, delle persone ossessionate dai microprocessori. Era come andare a un incontro di gente innamorata di martelli". Bob Albrecht vi partecipava raramente e dice che: "capivo solo quattro o cinque parole di quel che dicevano quei tizi... loro sì che erano hacker". Jude Milhon, la donna di cui Lee rimase amico dopo averla incontrata attraverso il "Barb" *e* il coinvolgimento nel Community memory, vi capitò una sola volta e rimase disgustata dalla concentrazione esclusiva circa la pura e semplice tecnologia: l'esplorazione e il controllo per il puro piacere del controllo. Notò inoltre la mancanza di donne tra gli hacker dell'hardware e la fece arrabbiare l'ossessione tutta maschile per il piacere e il potere della tecnologia. Sintetizzò le sue impressioni nell'epiteto "sono ragazzini coi loro giocattoli" e, come Fred Moo-re, si preoccupava che l'innamoramento per

la tecnologia li potesse ciecamente portare all'abuso di quella stessa tecnologia.

Nessuna di queste faccende rallentò la marcia dell'Homebrew, che incrementò le proprie fila fino a raggiungere diverse centinaia di aderenti, riempiendo l'auditorium dello Slac, diventando l'incontro quindicinale culminante per la vita di più di un centinaio di fedelissimi *"brewers'".* Quello a cui avevano dato inizio era ora diventata una crociata, qualcosa che Ted Nelson, il cui libro era pieno di lunghe geremiadi anti-IBM, avrebbe apprezzato. Mentre libra e i *"big boys"* non pensavano nemmeno lontanamente a questi casuali assembramenti di hacker sotto forma di computer club - per via della loro idea di essere i padroni assoluti dei calcolatori -le persone dell'Homebrew e altre come loro non solo stavano facendo hackeraggio con arguzia sul chip 8080, ma stavano sgretolando dalle fondamenta la torre di Babele del batch. "Noi ci rafforzavamo a vicenda" dice Lee. "Ci fornivamo una struttura eli supporto. Comperavamo i nostri prodotti. Ci guardavamo le spalle l'uno con l'altro. Ecco dov'eravamo, l'establishment non ci prestava alcuna attenzione. Eppure eravamo gente che ne sapeva tanto quanto loro di questi aspetti della tecnologia, perché era un settore completamente nuovo. Potevamo spaziare liberamente ecl era proprio quello che facevamo."

Quando a un certo punto Les Solomon, il guru newyorkese di questo movimento, venne a far visita alla West Coast, l'età dell'oro dell'Homebrew computer club era al culmine. Dapprima incontrò Roger Melen e Harry Garland che avevano appena finito il prototipo prodotto dalla Cromemco e che sarebbe stato sulla copertina del novembre 1975 di "Popular Electronics": un scheda aggiuntiva per l'Altair che avrebbe permesso alla macchina di connettersi a un televisore a colori, con risultati grafici strabilianti. In effetti, Melen e Garland avevano chiamato la scheda Dazzler [strabiliante]. Les si recò nell'appartamento di Roger per vederlo, ma prima d'inserire la scheda dentro l'Altair di Roger tutti e tre cominciarono a bere, ed erano molto allegri quando misero la scheda e accesero il televisore.

C'erano due programmi che avrebbero beneficiato della Dazzler. Uno era chiamato *Kaleidoscope,* un caleidoscopio che sprizzava di luce e cambiava forma sul monitor. Era un grande momento per Solomon, che vedeva il computer che generava dei meravigliosi disegni sullo schermo televisivo.

Poi provarono un altro programma: *Life.* Il "gioco che è più di un gioco", creato dal matematico John Conway, quel gioco che il mago del MIT Bill Gosper aveva hackerato con enorme passione, al punto da crederlo potenzialmente in grado di generare la vita stessa. La versione per Altair girava molto più lentamente di quella del Pdp-6, ovviamente - e per di più senza nessuna di quelle utility elegantemente hackerate - ma seguiva le medesime regole. E *lo faceva* mentre erano seduti al tavolo della cucina. Garland vi caricò alcune sequenze e Les Solomon, non conoscendo a fondo le regole del gioco, sicuramente inconsapevole delle profonde implicazioni filosofiche e matematiche, osservava le piccole stelle blu, rosse o verdi (che erano il modo in cui la Dazzler faceva apparire le cellule) mangiare altre piccole stelle, oppure incrementarne il numero. *Che perdita di tempo,* pensò. *Chi se ne frega?*

Poi cominciò a giocare pigramente con la macchina, impostando uno schema da sviluppare. Gli accadde, preso dalla sbornia, di elaborare qualcosa che rassomigliava alla stella di David. Oggi rammenta: "Feci girare il programma e osservai il modo in cui si consumava. Ci vollero dieci minuti e poi le generazioni si estinsero. E pensai, 'Accidenti, è interessante. Significa forse che la religione ebraica si estinguerà dopo 247 generazioni?' Allora disegnai un crocefisso. 11 programma andò avanti per 121 -generazioni. Significa forse questo che il giudaismo sopravviverà al cristianesimo?" Continuò poi caricando in programma mezzelune, stelle e simboli di vari significati differenti e tutti e tre - anzi tutti e quattro, Altair compreso - si ritrovarono a esplorare i misteri delle religioni e delle nazioni del mondo. "Ma che diavolo c'entra la filosofia alle tre di notte, dopo aver bevuto?", commenta Solomon. *"Era un computer.* Nient'altro."

Ma Les Solomon aveva anche altra magia da trasmettere. Tra le storie che raccontava, storie così fuori dal mondo che solo un povero d'immaginazione si sarebbe lamentato della loro improbabilità, ce n'era una che risaliva al tempo in cui aveva svolto ricerche per soddisfare uno dei suoi hobby, cioè l'archeologia precolombiana. Ciò richiedeva di passare molto tempo nella giungla, "correndo dietro agli indiani, scavando, rotolandosi nello sporco... a cercare cose". Era da quegli indiani, ribadiva Les Solomon, che aveva imparato il principio vitale del *vril,* un potere che ti permette di muovere enormi oggetti con una forza molto contenuta. Solomon credeva che fosse stato il potere del *vril* a permettere agli egizi di costruire le piramidi (e forse il *vril* era il potere di cui parlava Ed Roberts quando capì che il suo Altair avrebbe dato alla gente il potere di diecimila costruttori egiziani di piramidi). Secondo la sua storia, Solomon avrebbe incontrato un venerabile *brujo* indiano e gli avrebbe chiesto se poteva insegnare questo potere. Poteva il *brujo* insegnarglielo? Il *brujo* acconsentì. Dopo quella sera in cui aveva bevuto e giocato a *Life,* Solomon partecipò a una riunione dell'Homebrew allo Slac, dove gli era stato conferito il titolo di ospite d'onore, per aver fatto da levatrice all'Altair di Ed Roberts. E, dopo il meeting, Solomon si mise a parlare del *vril* agli hacker dell'hardware. In sala c'era una buona dose di scetticismo.

Fuori dallo Slac c'erano degli enormi tavoli da picnic arancioni con basi di cemento. Solomon disse a qualcuno dell'Homebrew di porre le proprie mani sopra uno dei tavoli e poi lui stesso lo toccò. Ora dovevano semplicemente pensare che si sarebbe sollevato.

Lee Felsenstein descrive la scena: "Disse: 'Ehi, vi faccio vedere io...' Noi pendevamo tutti dalle sue labbra, avremmo fatto qualsiasi cosa. E così sei persone circondarono il tavolo e vi misero sopra le mani. Mise la sua mano sopra le altre, strabuzzò gli occhi e disse: 'Alzati'. *E il tavolo si alzò di circa una spanna.* Si alzò con un movimento armonioso, elegante come un'onda sinusoidale. Non sembrava pesante. *Risollevò* davvero".

Ma alla fine, perfino i partecipanti, eccetto Solomon, non erano sicuri che fosse accaduto davvero. Lee Felsenstein, vedendo chiudersi un altro capitolo in quel frammentato romanzo di fantascienza che era la sua vita, comprese il mitico impatto di questo evento. Essi, i soldati dell'Homebrew computer club, avevano portato i loro talenti e avevano applicato l'etica hacker lavorando per il bene comune. Era l'atto di lavorare insieme all'unisono, provando, senza i dubbi causati dal guardare indietro, che faceva accadere cose straordinarie. Perfino impossibili. Gli hacker del MIT lo avevano scoperto quando il desiderio di fare hackeraggio li induceva a persistere così ostinatamente nel lavoro che le barriere di sicurezza, la stanchezza e i condizionamenti mentali sembravano dissolversi. Ora, nel movimento per annientare la tradizione di controllo antihacker dell'industria, per cambiare il modo di vedere riprovevole che il mondo aveva dei computer e della loro gente, l'energia combinata degli hacker dell'hardware che lavoravano insieme poteva fare qualsiasi cosa. Se non si fossero trattenuti, se non si fossero ritirati, se non avessero ceduto all'avidità, avrebbero fatto scorrere come un'onda gli ideali dell'hackeraggio attraverso la società, come fa una perla gettata in un bacinella d'argento.

L'Homebrew club s'era installato alla sommità del potere del *vril*.

La rivoluzione digitale

Fondamentali rivoluzioni tecnologiche hanno trasformato, in diversi stadi dello sviluppo della società umana, la vita economica, sociale e culturale. Ognuna di esse ha influenzato fortemente architettura, città e territorio.

Quella agricola, dell'era neolitica, è stata la prima di queste rivoluzioni. Caratterizzata dall'invenzione della ruota e dell'aratro, dalla coltivazione dei campi e dall'allevamento del bestiame. Il suo principale risultato fu il passaggio da una vita come cacciatori e raccoglitori ad una vita basata sulla produzione sistematica di risorse alimentari. All'interno della società poté così cominciare ad affermarsi una divisione capillare del lavoro, si svilupparono classi di artigiani specializzati e la popolazione cominciò a raggrupparsi in paesi e città: nasceva l'architettura. Si trattò di una rivoluzione molto lenta ed ebbe inizio, grosso modo, 10000 anni fa.

Successivamente la rivoluzione industriale, del diciannovesimo secolo, fu molto più veloce. Cominciò in Gran Bretagna intorno al 1780 e si diffuse in gran parte del mondo in meno di duecento anni. Il suo fattore determinante fu la scoperta di modi per sostituire la forza muscolare di uomini e animali con la forza di macchine a vapore alimentato, in un primo momento, a carbone poi col motore elettrico, il motore a combustione interna, il reattore nucleare. Si sviluppò un'economia basata sull'energia. I veicoli a motore annullarono le distanze, mentre i macchinari delle nuove fabbriche producevano beni e manufatti in grandi quantità. Cambiò il modello di organizzazione del lavoro. La divisione del lavoro si intensificò e grandi masse di operai non qualificati sostituirono gli artigiani e i lavoratori specializzati. Le città divennero sempre più grandi e complesse. Sistemi meccanici ed elettrici furono introdotti nelle nuove costruzioni e il loro ruolo divenne sempre più importante. L'impiego di nuovi materiali quali l'acciaio, il cemento armato e il vetro aprì possibilità organizzative e costruttive senza precedenti. Si impose la necessità di documentare le costruzioni con disegni sempre più precisi e completi, e si cominciarono ad applicare metodi formalizzati per la previsione dì costi e funzionalità.

La rivoluzione digitale fu alimentala dai progressi, negli anni della seconda guerra mondiale, nel settore dell'elettronica, quindi accelerata in maniera esplosiva dalla comparsa di nuove tecnologie come il transistor prima, il circuito integrato poi e, finalmente, il *chip* di silicio. Questa rivoluzione si è diffusa in tutto il mondo nell'arco di pochi decenni ed è stata di un ordine di grandezza più veloce della rivoluzione industriale e di due ordini di grandezza più veloce della rivoluzione agricola.

Come la rivoluzione industriale aveva sostituito la forza muscolare dell'uomo con macchine che consumavano energia, la rivoluzione digitale sta sostituendo la forza del cervello umano con macchine che elaborano l'informazione. Non abbiamo più soltanto un'economia agricola e un'economia dell'energia, ma anche un'economia, di sempre maggiore importanza, basata sull'informazione. Stiamo entrando nell'era postindustriale, nella quale la raccolta, l'elaborazione e la diffusione delle informazioni assumono un ruolo dominante nella vita economica.

La società postindustriale ha, di fatto, cominciato a svilupparsi a partire da una base economica fondata su tre pilastri:

la produzione sistematica di beni alimentari e l'estrazione di risorse naturali;

la produzione industriale di beni e il rapido trasporto meccanico di persone e merci;

la produzione, l'archiviazione e trasmissione elettronica delle informazioni.

Il ruolo dell'intelligenza

Le ricerche tecniche per costruire calcolatori potenti, piccoli ed economici ha avuto un ruolo trainante nell'evoluzione della rivoluzione digitale. Un simile traguardo tecnologico, strettamente legato agli avanzamenti nella tecnologia dei chip può essere letto in funzione di alcuni fattori chiave. II costo unitario per unità di calcolo, sia di immagazzinamento che di elaborazione, è sceso, negli ultimi due decenni, di quasi un milione di volte. I primi calcolatori riempivano grandi stanze, consumavano enormi quantità di energia ed erano esclusiva di pochi e avanzati laboratori di ricerca. Oggi alcune *workstation* ad alte prestazioni sono più piccole dell'elenco del telefono, possono essere installate in qualunque ufficio, consumano meno energia di una lampada da tavolo e costano molto meno di un'automobile.

La conseguenza economica di tutto ciò è semplice e scioccante assieme. L'intelligenza,

considerata tradizionalmente un bene prezioso, è diventata d'un tratto così a buon mercato che arriveremo presto a considerarla gratuita. Anche perché oramai l'intelligenza è dappertutto. L'automobile è dotata di chip e così il videoregistratore, forse il tostapane.

Nel 1948, quando tutto questo si poteva soltanto intravedere all'orizzonte, Norbert Wiener nel suo pionieristico libro *Cybernetics* anticipò i profondi effetti sociali della disponibilità di intelligenza a basso costo: “Posso forse chiarire il contesto storico della situazione presente sostenendo che la prima rivoluzione industriale, la rivoluzione degli "scuri e satanici opifici", consisteva nella svalutazione del braccio umano per mezzo della competizione con la macchina. Non esiste livello di paga, per basso che sia, per il quale un operaio scavatore degli Stati Uniti possa competere con il lavoro di un escavatore a vapore. La rivoluzione industriale moderna agisce allo stesso modo nel deprezzare il cervello umano, almeno nelle sue decisioni più elementari e di routine. Certo, allo stesso modo in cui i più abili carpentieri, meccanici e sarti sono in qualche modo stati capaci di sopravvivere alla prima rivoluzione industriale, gli scienziati e gli amministratori più preparati potranno sopravvivere alla seconda. Ad ogni modo, nel momento in cui il comune essere umano di mediocre (o ancor più basso) livello arriverà a riconoscere l'avvento della seconda rivoluzione industriale, si troverà nella condizione di non aver più nulla da vendere che possa valere il denaro dì qualcuno”.

Aveva ragione. Gli sportelli automatici hanno già sostituito molti cassieri di banca, esattamente come i database gestionali hanno tolto il lavoro agli impiegati d'archivio. È facile prevedere che i prossimi candidati a questo processo saranno i lavoratori meno specializzati, considerato che il costo unitario di elaborazione delle informazioni continua a diminuire e il livello di sofisticazione del software è in crescita. Il discorso vale anche per i progettisti.

Siamo molto vicini al punto in cui il comune architetto potrebbe non aver più nulla da vendere che possa valere il denaro di qualcuno.

Nuove sfide

Come lavoreranno gli architetti, i paesaggisti, gli urbanisti nell'emergente era postindustriale, e cosa si produrrà?

Come, all'alba dell'era delle macchine, ad alcuni artigiani capitò di diventare attori centrali dell'innovazione, ai progettisti e ai designer più attenti e consapevoli si apre l'opportunità di contribuire direttamente allo sviluppo digitale e di applicare questi nuovi e straordinari strumenti alle più importanti mansioni sociali e culturali. Le trasformazioni in atto spianano la strada a nuove opportunità e a stimolanti sfide intellettuali, ma anche a nuovi e controversi problemi.

Una ragione per essere ottimisti è che la rivoluzione digitale inverte parzialmente una delle conseguenze più problematiche della rivoluzione industriale. L'introduzione di sistemi di produzione talmente vasti e complessi da escludere la maggior parte degli individui dal loro possesso e utilizzo. Le conseguenti croniche tensioni tra forza lavoro e padronato hanno contribuito all'idea che assimila le tecnologie avanzate ad atteggiamenti autoritari e a strutture di potere oppressive. Questa diffusa posizione viene tuttavia radicalmente messa in discussione dalla presenza in tutto il mondo di più di cento milioni di computer, dalla capillare disponibilità di personal computer a costi ben inferiori a quelli di una normale automobile, e dalla proliferazione e diffusione di prodotti software a basso costo. Oggi più che mai, individui e piccoli gruppi con risorse anche limitate possono sviluppare nuovi strumenti software, potenti e innovativi, e possono valersi di quelli già esistenti per raggiungere risultati che vanno ben al dì là di quelli ottenibili nel passato.

L'emergere di nuove arti popolari, basate sulla tecnologia informatica, è un secondo e incoraggiante sviluppo. La maggior parte della produzione di questi nuovi artigiani digitali fai-da-te è ovviamente spesso naif e imperfetta, e i vecchi professionisti che un tempo controllavano buona parte della produzione hanno buon gioco a farsene beffe. Ciononostante è evidente che la diffusa e consistente pratica di queste arti, nonché il livello di consapevolezza critica che ne derivano, rappresentano segnali culturali di grande peso, in

particolare se confrontati con l'alternativa di un uso di massa sostanzialmente passivo. Infine, nonostante la capacità di coordinamento occhio-mano, cruciale nella maggior parte delle arti tradizionali, tenda a perdere peso in molti dei contesti oggi supportati dalla tecnologia digitale, gli aspetti legati a una professionalità critica e ben radicata sono oggi vitali come non mai. Esattamente come l'attività del disegno non può venir ridotta a un mero fatto atletico, quando svolta a mano, non è possibile liquidarla come una semplice questione meccanica, quando a tracciare le linee è un computer, e ciò vale sia quando un determinato segno venga tracciato direttamente, con la grafite sulla carta, sia quando ciò venga causato indirettamente dalla forma del tracciato di un fascio di elettroni su uno schermo a raggi catodici: ciò che conta, in ogni caso, è la piena consapevolezza del perché stiamo posizionando una linea in una determinata posizione, e quali qualità ci permetteranno di farlo nel modo più efficace. Un buon sistema computerizzato, come il pennello di un calligrafo o un eccellente pianoforte, sono splendidi e precisi strumenti che, per essere ben impiegati, richiedono dai loro utenti una estrema cura ed esperienza. Anche se la graduale scomparsa di alcune arti tradizionali potrà dispiacere a qualcuno, assisteremo al sorgere di nuove arti che, inevitabilmente, finiranno per sostituirle.

Capitolo 2° - Strumenti e applicazioni

Esposizione del repertorio degli strumenti informatici dedicati alla progettazione e discussione dei loro fondamenti teorici, delle loro potenzialità e dei loro limiti.

Hardware e Software

Un'altra coppia che sottolinea la fusione per una condizione nuova e nuove possibilità.

Nella prima stesura gli argomenti erano separati. Accorpando e semplificando la struttura, come ben ricorda John Maeda in *Le leggi della semplicità*, i due argomenti si sono ovviamente attratti. Del resto, come abbiamo visto nel primo capitolo, le due cose sono inscindibili fin dai tempi di Babbage e oltre considerando, per esempio, le semplici operazioni aritmetiche.

Ciò che peserà di più in questo lavoro sarà ovviamente il software. Giusto qualche accenno per quanto riguarda l'hardware e specificazioni dove necessario.

Con hardware, in ingegneria elettronica e informatica si indica la parte fisica di un personal computer, ovvero tutte quelle parti magnetiche, ottiche, meccaniche ed elettroniche che ne consentono il funzionamento. Più in generale il termine si riferisce a qualsiasi componente fisico di una periferica o di una apparecchiatura elettronica.

Possiamo considerare, con William J. Mitchell e Malcolm McCullough in *Digital Design Media*, un tostapane elettrico che "prende fette di pane come *input*, esegue un processo, produce toast come *output*. La sua funzione è trasformare pane in toast. Analogamente, un computer prende informazioni come input, esegue un processo, produce nuove informazioni come output: la sua funzione è trasformare le informazioni di cui disponiamo nelle informazioni che ci servono". Continuando sviluppando ulteriormente l'analogia "il tostapane elettrico può accettare serie diverse di input (brioches, pizzette ecc.). Per ottenere risultati diversi è possibile agire sui suoi controlli così da introdurre variazioni dello stesso processo. La gamma delle informazioni che un computer può accettare come input è molto vasta, e le variazioni nel processo possono essere determinate scrivendo programmi che descrivano le operazioni da eseguire".

Quindi il diagramma funzionale di un computer lo descrive come un strumento in grado di elaborare un input di dati accessibili per ottenere un output di dati desiderati.

"Senza sostanziali modifiche a questo schema di organizzazione funzionale, i progressi più significativi della tecnologia hardware sono stati raggiunti sostituendo componenti costosi e ingombranti con dispositivi più veloci, più economici e di dimensioni più ridotte. Le celebri macchine calcolatrici progettate da Charles Babbage nel secolo scorso si servivano di apparati meccanici per archiviare e processare numeri. Alcuni dei primi calcolatori moderni impiegavano relè telefonici elettromeccanici. I primi computer degli anni Quaranta e Cinquanta erano costituiti da serie immense di valvole. Queste furono successivamente rimpiazzate da transistor, circuiti integrati e chip di silicio. Il processo continua, con lo sviluppo di tecniche per la fabbricazione di chip sempre più piccoli, densi e complessi".

Personal computer e workstation

Il personal computer è una tipologia di computer contraddistinta dall'essere *general purpose*, monoutente, destinato principalmente ad un utilizzo produttivo, e dall'avere prestazioni medie, si distingue dall'home computer in quanto il secondo è destinato principalmente ad un utilizzo ludico ed è contraddistinto da prestazioni base. Invece la workstation, pur essendo un computer *general purpose* e monoutente e destinato principalmente ad un utilizzo produttivo, è contraddistinta da alte prestazioni. Caratteristica quest'ultima che rende la workstation un computer utilizzato da professionisti in ambiti che necessitano di grandi potenze di calcolo come ad esempio il CAD, la ricerca scientifica, la postproduzione.

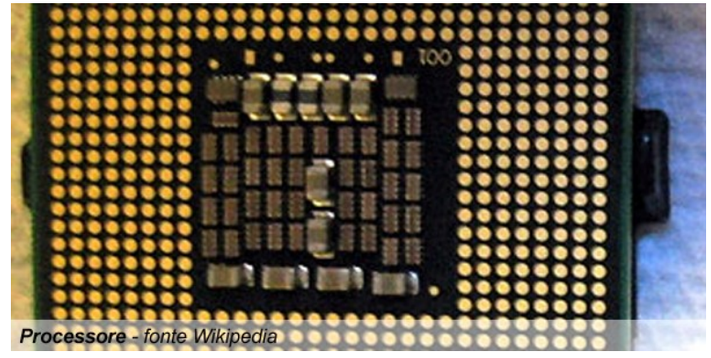
Oggi un computer portatile può essere dotato come una workstation e servire dì volta in volta da ambiente per la scrittura, per il calcolo numerico, per la composizione e l'esecuzione musicale, per la fotografia digitale, o da ambiente di progettazione superbamente attrezzato ed efficiente, che oltre a comprendere buona parte delle tradizionali funzioni del tavolo da disegno ci mette a disposizione una serie in crescita continua di strumenti e potenzialità nuove.

Scheda madre

Scheda madre - *fonte Wikipedia*

La scheda madre o scheda di sistema, anche conosciuta come motherboard o mainboard (sinonimi mutuati dall'inglese), in sigla MB, o con le abbreviazioni mobo (abbreviazione di "motherboard") e M/B (abbreviazione di "motherboard" o "mainboard"), è una parte fondamentale di un moderno personal computer: raccoglie in sé tutta la circuiteria elettronica di interfaccia fra i vari componenti principali e fra questi e i bus di espansione e le interfacce verso l'esterno. La scheda madre è responsabile della trasmissione e temporizzazione corretta di molte centinaia di segnali diversi, tutti ad alta frequenza e tutti sensibili ai disturbi: per questo la sua buona realizzazione è un fattore chiave per la qualità e l'affidabilità dell'intero computer.

Processore



Processore - *fonte Wikipedia*

L'unità centrale di elaborazione, in sigla CPU (dal corrispondente termine inglese *central processing unit*), anche chiamata nella sua implementazione fisica processore, è uno dei due componenti principali della macchina di von Neumann, il modello su cui sono basati la maggior parte dei moderni computer.

Compito della CPU è quello di leggere i dati dalla memoria ed eseguirne le istruzioni; il risultato dell'esecuzione dipende dal dato su cui opera e dallo stato interno della CPU stessa, che tiene traccia delle passate operazioni.

Durante la fase di rendering è il componente del computer più impegnato. Vale la pena ricordare che, se si necessita di ulteriore potenza di calcolo per effettuare un'animazione per esempio, attraverso la rete è possibile comandare a distanza, previa registrazione e pagamento, sistemi di calcolo molto potenti. Le renderfarm. Con un sistema portatile ben equipaggiato ed una connessione ad internet è possibile fare quasi tutto.

Disco

Disco - *fonte Wikipedia*

L'hard disk drive (termine di origine inglese), in sigla HDD, normalmente abbreviato in hard disk, anche chiamato disco rigido (traduzione letterale di "hard disk") o disco fisso (traduzione letterale di "fixed disk"), è una tipologia di dispositivo di memoria di massa che utilizza uno o più dischi magnetici per l'archiviazione dei dati.

Il disco rigido è una delle tipologie di dispositivi di memoria di massa attualmente più utilizzate. È infatti utilizzato nella maggior parte dei computer e anche in altre tipologie di dispositivi elettronici. Ha da poco tempo un serio concorrente, il disco allo stato solido, destinato probabilmente in futuro a soppiantarlo.

Un efficace sistema per velocizzare enormemente la gestione dati su disco, qualunque sia la sua tecnologia, è quello di mettere più dischi in parallelo.

Un Redundant Array of Independent Disks ("insieme ridondante di dischi indipendenti", RAID) è un sistema informatico che usa un insieme di dischi rigidi per condividere o replicare le informazioni. I benefici del RAID sono di aumentare l'integrità dei dati, la tolleranza ai guasti e/o le prestazioni, rispetto all'uso di un disco singolo.

Memoria

Memoria - *fonte Wikipedia*

La memoria ad accesso casuale, acronimo RAM (del corrispondente termine inglese *Random Access Memory*), è una tipologia di memoria informatica caratterizzata dal permettere l'accesso diretto a qualunque indirizzo di memoria con lo stesso tempo di accesso. La memoria ad accesso casuale si contrappone alla memoria ad accesso sequenziale e alla memoria ad accesso diretto rispetto alle quali presenta tempi di accesso sensibilmente inferiori motivo per cui è utilizzata come memoria primaria.

La tipologia di memoria ad accesso casuale più comune attualmente è a stato solido, a lettura-scrittura e volatile, ma rientrano nella tipologia di memoria ad accesso casuale la maggior parte delle tipologie di ROM, la NOR Flash, oltre a varie tipologie di memorie informatiche utilizzate ai primordi dell'informatica e oggi non più utilizzate come ad esempio la memoria a nucleo magnetico.

Esclusivamente l'acronimo RAM (non il termine "memoria ad accesso casuale") ha anche una seconda accezione più ristretta ma attualmente più diffusa secondo cui la RAM è una memoria ad accesso casuale della tipologia più comune cioè a stato solido, a lettura-scrittura e volatile. L'uso della memoria RAM è comune a tutte le architetture hardware, sia

a singolo processore che multiprocessore e costituisce la *memoria primaria* dell'elaboratore.

Caratteristica comune a tutti i tipi di RAM utilizzati per la memoria principale è quella di perdere il proprio contenuto nel momento in cui viene a mancare l'alimentazione elettrica. Sono allo studio altri tipi di memoria, basati su altri principi, che in futuro potrebbero consentire di superare questa limitazione.

Scheda grafica

Scheda grafica - *fonte Wikipedia*

Una scheda video è un componente del computer che ha lo scopo di generare un segnale elettrico (output) che possa essere mostrato a video (display). A seconda del tipo di computer questo dispositivo può essere più o meno potente: i primi modelli di scheda video potevano visualizzare solo testo; successivamente si sono diffuse anche schede video in grado di mostrare output grafici (immagini non testuali) e, recentemente, anche modelli tridimensionali *texturizzati* in movimento e in tempo reale. Questi ultimi tipi di scheda provvedono anche ad elaborare e modificare l'immagine nella propria memoria interna, mentre le schede 2D possono mostrare immagini 3D solo con l'aiuto della CPU che deve eseguire da sola tutti i calcoli necessari.

Reti

Ancora Mitchell e McCullough: “come due persone possono parlare al telefono, due computer possono scambiarsi informazioni per mezzo di un collegamento in telecomunicazione. La combinazione delle tecnologie dei computer e delle telecomunicazioni ha permesso lo sviluppo di reti di calcolatori sempre più estese e sofisticale.

La comunicazione di rete offre alcuni significativi vantaggi. 11 più ovvio è la possibilità di ottenere un efficiente scambio di dati tra computer distanti e l'accesso remoto a importanti risorse in linea, come il catalogo di una biblioteca. Le reti, inoltre, facilitano la collaborazione a distanza; impiegando sofisticate tecniche di comunicazione, ad esempio, progettisti che stanno in luoghi diversi possono lavorare sullo stesso file di testo, foglio elettronico o database CAD, comunicando contemporaneamente tra loro in videoconferenza.

Il *networking* permette inoltre la condivisione delle risorse. Una stessa stampante può servire efficientemente più computer a essa collegati, il che permette di evitare di dover disporre di una stampante per ogni posto macchina. Aggregata così la domanda di stampa e distribuii i costi su un numero maggiore di utenti, una stampante di rete può generalmente essere di tipo più costoso e di maggior qualità rispetto alla stampante di un singolo utente. Estendendo questo principio possiamo sviluppare reti di calcolatori sofisticate che (come le organizzazioni umane) si fondano su divisione del lavoro, specializzazione dei ruoli e una qualche forma di controllo gerarchico. Una tipica rete di tipo *”client/server”* si serve dì almeno un grande e potente calcolatore, noto *come file server,* con funzioni di archivio dati centralizzato, e di macchine più piccole che a esso si collegano per svolgere il loro lavoro. Potrà poi esserci una macchina particolarmente veloce: un server di calcolo, dedicato ai compiti più pesanti sotto un profilo computazionale. In altri casi, la rete potrà essere organizzata così che ogni macchina che ne fa parte possa svolgere alternativamente sia il ruolo di "client" che di "server". Sulle scrivanie degli uffici, nelle linee di produzione, negli studi di progettazione, nei laboratori e nelle aule didattiche, o dovunque si svolge il lavoro in una particolare organizzazione, sì troveranno diverse stazioni di lavoro: gli utenti della rete

interagiscono direttamente con queste postazioni, che potranno quindi essere configurate per compiti specifici: elaborazione del testo, grafica, composizione musicale e via dicendo. Infine, potranno venir predisposte stazioni specializzate di input/output per la produzione di stampe, il plottaggio di tavole ingombranti, la registrazione del suono o l'acquisizione di immagini.

Gli schemi di comunicazione tra i computer si sono modificati parallela mente all'evoluzione della tecnologia informatica. Agli inizi, considerati i costi elevati dei processori e della memoria, si tendeva a concentrare le risorse in grosse macchine centralizzate che servivano terminali di comunicazione dotati di poca o nessuna intelligenza locale. Era l'era dei sistemi *time-sharing* (durata grosso modo fino ai primi anni Ottanta). Successivamente, quando gli economici chip di silicio divennero ampiamente disponibili, si apri (a cavallo tra gli anni Settanta e Ottanta) l'era dei personal computer. Le organizzazioni che si basano sui personal computer realizzano un pieno trasferimento di intelligenza alle stazioni di lavoro dei singoli utenti, penalizzando però gli aspetti legati all'intercomunicazione e alla condivisione delle risorse. Più tardi. Con l'evoluzione delle tecnologie per l'intercomunicazione tra computer, siamo entrati nell'era delle reti. In una rete, le risorse di calcolo possono essere distribuite o centralizzate nel modo più conveniente.

Con il proliferare delle reti e dei collegamenti tra di esse, sono comparse enormi reti di reti: è il caso di Internet, che nel corso degli anni Novanta ha avuto una crescita esplosiva. È plausibile prevedere che le tecnologie, oggi separate, dell'informatica e delle telecomunicazioni finiranno per fondersi per dar vita a un solo ambiente unificato di media digitali. La televisione sarà parte di un computer e il computer si trasformerà in una postazione per la teleconferenza. Avremo ancora il nostro computer sulla scrivania, ma ci serviremo probabilmente anche dì dispositivi per il calcolo e per la telecomunicazione più piccoli, senza fili, trasportabili o addirittura indossabili come capi di vestiario. Assistiamo all'evoluzione di una rete di reti globale alla quale una popolazione estremamente mobile potrà in futuro collegarsi in qualsiasi momento e da qualunque luogo".

Sistemi operativi

Il sistema operativo è codice responsabile del controllo e della gestione dei componenti hardware che costituiscono un computer e dei programmi che su di esso vengono eseguiti.

Solitamente un sistema operativo mette a disposizione dell'utente un'interfaccia software (grafica o testuale) per accedere alle risorse hardware (dischi, memoria, I/O in generale) del sistema. Tale accesso dipende, sui sistemi che prevedono la multiutenza, dai privilegi di ogni utente.

Il compito principale del sistema operativo è quello di permettere all'utente, umano o non, di interagire direttamente con la macchina.

Un computer, come qualsiasi organizzazione umana, ha compiti da svolgere e risorse per realizzarli. Così Mitchell e McCullough: "il gestore di un'organizzazione umana raccoglie le istruzioni sul lavoro da svolgere e alloca conseguentemente le risorse necessarie per il suo espletamento. In un computer, lo stesso ruolo è svolto da una componente software chiamata sistema operativo. L'utente istruisce il sistema operativo (generalmente battendo comandi o puntando con il mouse). In risposta, il sistema operativo governa la macchina.

Ogni sistema operativo deve svolgere alcuni compiti fondamentali. In particolare deve occuparsi di organizzare i file, comandare l'esecuzione di programmi che su questi lavorano, allocare memoria e risorse hardware, gestire i processi di input e output; deve archiviare e riprodurre sequenze di operazioni ricorrenti, coordinare eventi come code di stampa o scambi di messaggi, garantire la sicurezza dei dati e aggiornare le versioni del lavoro archiviato, restituendo infine lo stato di queste attività in maniera complessiva e chiara. Non tutti si rendono conto della fondamentale importanza di tutto ciò: per organizzare il nostro lavoro e divenire buoni utenti di un computer dobbiamo essere in grado di padroneggiare gli elementi base di un sistema operativo.

Come nel caso dei manager umani, i sistemi operativi variano enormemente nel loro stile, nella gamma di risorse e di compiti che sono in grado di svolgere e nel grado in cui questi stessi compiti possono essere loro assegnali senza ricorrere a istruzioni specifiche. Alcuni di questi aspetti potranno coinvolgere casualmente gli utenti, mentre sarà bene che altri restino di competenza dell'amministratore del sistema.

Un'importante distinzione dev'essere fatta tra sistemi mono- e multi-utente. I personal computer sono generalmente dotati di sistemi operativi mono-utente, mentre calcolatori di classe superiore impiegano sistemi operativi multi-utente, che cioè permettono una interazione simultanea con più utenti. I sistemi operativi più semplici possono svolgere un compito per volta, mentre i sistemi *multitasking* possono svolgere contemporaneamente più funzioni (l'elaborazione di un documento durante la stampa di un altro). Tra i sistemi operativi più diffusi degli anni Ottanta e Novanta, per esempio, il DOS è un sistema operativo semplice, mono-utente, in grado di svolgere un solo compito alla volta, mentre UNIX è un più complesso e sofisticato sistema multi-utente e multitasking.

Generalmente un sistema operativo controlla un singolo computer, ma con l'evoluzione della tecnologia del *networking* si è sviluppato un interesse crescente nei confronti di sistemi più sofisticati che possono gestire in modo distribuito le risorse di un'intera rete. Un simile sistema può cioè analizzare lo stato della rete alla ricerca di un processore temporaneamente inutilizzato e attribuirgli un compito, senza che l'utente sappia o si debba preoccupare di dove il lavoro viene in effetti svolto.

Lo schema di interazione con i primi sistemi operativi si basava su serie di comandi impartiti via tastiera. Questo approccio assicura potenza e flessibilità, ma può spesso risultare, agli occhi di un utente inesperto, criptico e misterioso. Con il declino dell'era in cui l'uso dei calcolatori era dominio esclusivo di una categoria di tecnici e specialisti, si è assistito a una crescente tendenza verso l'impiego di interfacce grafiche e approcci più intuitivi e comprensibili tra l'utente e il sistema operativo. II Macintosh della Apple, per esempio, ha reso popolare l'idea di impiegare un mouse per puntare e selezionare le "icone" di una "scrivania" visualizzata sullo schermo. Alcuni dei sistemi operativi testuali che lo avevano preceduto, come il DOS e lo UNIX, hanno presto adottato questo approccio, introducendo interfacce grafiche organizzate "per finestre". Questa soluzione presenta lo svantaggio di un'interfaccia che rimane separata rispetto a un sistema non inerentemente grafico; svantaggio che è parzialmente compensato dalla possibilità di input diretto da tastiera quando necessario, ad esempio per interagire con entità che non sono immediatamente utilizzabili nella limitata porzione visibile dello schermo. È possibile standardizzare le forme di interazione impiegando uno stesso sistema di interfaccia grafica, indipendente dalle diverse piattaforme, che può essere quindi adattato a una vasta gamma di sistemi operativi ospiti. Il sistema X Windows sviluppato presso il MIT e un tipico esempio di questo approccio".

Storia dei sistemi operativi

In Informatica la Storia dei Sistemi Operativi descrive l'evoluzione di questi software durante tutto il periodo di sviluppo dei calcolatori elettronici. La Storia dei Sistemi Operativi procede a fianco della Storia del personal computer e, più in generale, della Storia dell'informatica.

Storia dei sistemi operativi
da Wikipedia, novembre 2009

In un periodo delimitabile tra il 1945 e il 1955 gli elaboratori elettronici erano ammassi di valvole termoioniche, occupavano intere stanze, erano lentissimi e così costosi che potevano permetterseli soltanto grossi centri di calcolo o Università. Inoltre questi calcolatori erano molto inaffidabili, in quanto le valvole che li componevano si rompevano spesso. In questo periodo non esisteva ancora il concetto di Sistema Operativo; infatti il programma da eseguire veniva inserito ad ogni esecuzione in codice binario attraverso dei primitivi lettori di schede perforate e dopo alcune ore il risultato veniva inviato ad una stampante.

Tra il 1955 e il 1965, grazie alla rivoluzionaria invenzione del transistor gli elaboratori (chiamati Mainframe) divennero abbastanza affidabili da poter essere costruiti e venduti in serie, anche se erano comunque macchine grosse e costosissime tanto che gli unici acquirenti possibili erano ancora una volta i Centri di Calcolo, le Università e le banche. Per eseguire dei programmi (o come venivano chiamati, *job*), un programmatore doveva scrivere il proprio programma su carta, trasferirlo su schede, caricarlo nel computer, attendere il termine dell'esecuzione e la stampa del risultato. Tale operazione era molto dispendiosa in termini di tempo e non permetteva di sfruttare la macchina durante le lunghe fasi di caricamento di dati e programmi. Non essendo stata ancora introdotta la tecnologia di accesso diretto alla

memoria (DMA) durante le fasi di input/output il processore era totalmente utilizzato per il controllo di queste operazioni. È per questo che si adottò la soluzione del sistema batch (a *lotti*): l'idea di base era quella di dividere i tre lavori, ovvero il caricamento dei dati, il calcolo e la stampa su macchine distinte. Il calcolo veniva affidato ad un calcolatore centrale costoso come l'IBM 7094 mentre gli elaboratori satelliti erano macchine più economiche come gli IBM 1401. Il Sistema Operativo di questi Mainframe doveva erogare pochi semplici servizi: gestione dell'input/output, interpretazione dei comandi contenuti nelle schede controllo e controllo dell'esecuzione di programmi, sia quelli lanciati dall'utente, sia le utilità di sistema. I sistemi operativi tipici per questi elaboratori, per lo più programmati in FORTRAN e in Assembler erano il FMS (*Fortran Monitor System*) e l'IBSYS. Dai primi anni '60 cominciò a farsi strada il concetto di dispositivo virtuale e astrazione. Prima di ciò un programmatore che avesse voluto comandare ad esempio una stampante, doveva conoscere, nei minimi dettagli, il funzionamento a basso livello della periferica, mentre a partire dall'introduzione del concetto di periferica virtuale il Sistema Operativo avrebbe fatto da intermediario tra utente e periferica. Nello stesso periodo i Sistemi Operativi iniziarono a supportare il DMA e lo SPOOL.

Il DMA (Direct Memory Access) è il sistema che permette di trasferire interi blocchi di dati da memoria secondaria a memoria centrale in modo completamente indipendente dal processore, il quale può, nel frattempo, eseguire altre operazioni. Lo SPOOL (Simultaneous Peripheral Operations On Line) è un sistema che permette di gestire in maniera efficiente le code di job di stampa.

Infine, ma non meno importante si assistette all'introduzione delle *politiche di ordinamento dei job*. Essi venivano caricati ed eseguiti in maniera sequenziale, ma l'ordine di esecuzione dei programmi era gestito da opportune politiche implementate nel Sistema Operativo. Siamo ancora lontani da ciò che sarà poi il timesharing supportato da tutti i Sistemi Operativi moderni, tuttavia è il germe che darà il via alla implementazione della multiprogrammazione negli anni successivi.

UNIX

Dagli anni sessanta esiste il concetto di timesharing: ogni utente dispone di un dispositivo di ingresso (la tastiera) e un dispositivo di uscita (un monitor o una telescrivente), ed ha la possibilità di inviare comandi al Sistema Operativo ottenendo subito una risposta. Infatti in questi sistemi con timesharing un programma resta in esecuzione fino a quando esso richiede un'operazione di I/O, oppure occupa la CPU per più di un certo

intervallo di tempo prestabilito (detto *quanto temporale*).

Nel 1962 venne realizzato al MIT il primo sistema di timesharing su un IBM 7094: il CTSS. Fu in realtà il MULTICS la vera rivoluzione. Venne sviluppato congiuntamente dal MIT, dalla General Electric e dai Bell Labs, ed era in grado di supportare centinaia di utenti in timesharing. La realizzazione fu però molto più complessa del previsto, tanto che i Bell Labs abbandonarono il progetto. Tuttavia Multics introdusse molte nuove idee che influenzarono non poco i successivi Sistemi Operativi. Se da un lato esistevano questi supercomputer dall'altro negli stessi anni vi fu lo sviluppo dei minielaboratori, dei quali un importante esponente fu il PDP-1 del 1961 che costava solo 120.000$ (cioè il 5% del prezzo di un IBM 7094) e che ebbe un gran successo. Per questi sistemi vennero progettati appositi Sistemi Operativi, il più famoso dei quali fu senza dubbio UNIX.

UNIX fu progettato a partire dal 1969 da un gruppo di ricercatori della AT&T presso i Bell Labs, tra cui erano presenti Ken Thompson (che lavorò anche al progetto Multics), Dennis Ritchie e Douglas McIlroy. Esso prese notevole spunto dal padre Multics, e grazie all'ottimo lavoro di queste persone divenne un sistema molto interattivo, affidabile e ricco di funzionalità, tanto che, nelle sue varianti ed evoluzioni, tuttora domina il mercato delle workstation. Da esso furono realizzate varianti come BSD (*Berkley Software Distribution*) e sistemi Unix-like come Minix (usato in ambito didattico) e successivamente l'ormai famosissimo Linux sviluppato dallo studente finlandese Linus Torvalds. Oggigiorno i sistemi operativi *NIX sono conformi allo standard POSIX (che uniforma l'interprete dei comandi e le API dei programmi), offrendo una compatibilità reciproca di base necessaria a non stroncarne lo sviluppo.

L'arrivo del Personal Computer (anni '80)

Verso gli anni '80 grazie alla tecnologia LSI (large scale integration) la costruzione di chip integrati divenne massiccia e portò all'abbattimento dei prezzi dell'hardware, facendo sorgere l'era dell'elaboratore personale o Personal Computer. Queste macchine erano piccole, economiche ed avevano prestazioni simili a quelle dei calcolatori medio-grandi di 10-20 anni prima. I primi modelli erano dotati di Sistemi Operativi monoutente con accesso interattivo e supporto al timesharing. Il più importante tra i primi Sistemi Operativi per Personal computer era il CP/M-80 della Digital Research per le CPU 8080 / 8085 / Z-80. Era basato sui Sistemi Operativi della Digital Equipment Corporation specialmente quelli per l'architettura PDP-1. MS-DOS (o PC-DOS quando fornito da IBM) era originariamente basato proprio sul CP/M-80.

Microsoft vs Apple

Steve Jobs era uno dei pochi che credeva nell'idea del Personal Computer. All'epoca era difficile immaginare cosa potesse farsene una persona di un computer in casa. Egli invece continuò per la sua strada fondando Apple Computer Inc. il 1° Aprile 1976 assieme a Steve Wozniak e Ronald Wayne. Jobs era convinto che il futuro del Personal Computer sarebbe stato legato all'interfaccia grafica. E così, ispirandosi a quella sviluppata da Xerox qualche anno prima, Apple lanciò nel 1984 Mac OS il primo sistema operativo per Personal Computer con interfaccia grafica. Questa fu una vera rivoluzione tanto che di lì a poco Microsoft avrebbe commercializzato Windows (20 novembre 1985) e sarebbe nato l'X Window System in ambiente Unix (1984). All'inizio Windows non era definibile Sistema Operativo: era piuttosto un'estensione di MS-DOS. Fu con il rilascio di Windows 3.0, nel 1990, che Microsoft si impose sul mercato. Oltre alle maggiori performance che garantiva alle applicazioni rispetto alle versioni precedenti, Windows 3.0 forniva un ambiente multitasking migliorato rispetto alle precedenti versioni di MS-DOS, grazie all'introduzione del supporto alla memoria virtuale, e divenne così un degno rivale del Macintosh (su cui girava Mac OS) di Apple. A partire da Windows 3.1 fu introdotto il supporto alla multimedialità (perfezionato via via nelle successive release), mentre con l'introduzione di Windows 95 si passò definitivamente dal calcolo a 16 bit a quello a 32 bit.

I sistemi operativi di rete e il fenomeno Linux

A fianco di Microsoft ed Apple il mercato delle workstation e dei grandi elaboratori era comunque dominato da UNIX. Un fenomeno interessante che iniziò a prendere piede da metà degli anni '80 fu lo sviluppo delle reti di calcolatori, fenomeno che ha poi portato all'incredibile crescita di Internet. Nacquero così i *Sistemi Operativi di rete* e i *Sistemi Operativi distribuiti*. I primi altro non sono che normali Sistemi Operativi ai quali vengono aggiunti i software per il collegamento a macchine remote e quindi alle relative risorse condivise come file e stampanti. I secondi sono Sistemi Operativi che girano su sistemi a più processori oppure che inviano i processi da elaborare ad altri computer della rete. Essi sono dotati di particolari politiche di scheduling che permettono una efficace allocazione dei processi tra le CPU disponibili. Inoltre anche il sistema di archiviazione (il *file system*) è unico, anche se in realtà è distribuito tra vari nodi della rete. Esempi della prima classe di sistemi di rete sono tutte le versioni di Windows dalla 3.1 e NT in poi, mentre UNIX e derivati fanno parte di entrambe le categorie in quanto supportano entrambe le tecnologie. Nel 1991 Linus Torvalds sviluppò un kernel Unix-like, capace però di girare sulla piattaforma x86. Così nacque Linux. Fu quando venne abbinato al Progetto GNU di Richard

Stallman, portavoce della *filosofia* del Software Libero, che iniziò la fortuna di Linux in quanto la combinazione risultava, e lo è tuttora, un sistema operativo efficiente ed affidabile anche se non sempre di facile utilizzo.

I giorni nostri

Oggigiorno è disponibile una grande varietà di sistemi di elaborazione dalle più disparate dimensioni e performance a costi contenuti; questo permette una diffusione pervasiva degli elaboratori elettronici nelle più diverse attività umane. Inoltre sono oggi sempre più diffuse le interconnessioni tra i vari dispositivi in modalità sia wired che wireless. Tutte queste innovazioni hanno portato allo sviluppo di sistemi operativi per le più svariate architetture, in particolare per *dispositivi handheld* come cellulari (tra i quali non si può non citare il Symbian OS) e PDA (con Windows Mobile e Palm OS).

Per qualunque architettura venga sviluppato un Sistema Operativo moderno esso deve fornire il supporto, oltre a quanto visto sinora, a molteplici esigenze quali:

- Streaming audio/video (trasmissione ed elaborazione continua di dati multimediali)
- supporto alle più diverse tecnologie di interconnessione (ad esempio Ethernet, Bluetooth e Wireless LAN)
- integrazione di tecnologie per la fruizione di contenuti su Internet
- una efficiente gestione dell'energia.

La linea di sviluppo dei sistemi operativi moderni per il mercato consumer è principalmente incentrato su multimedialità, connettività e risparmio energetico.

Interfaccia

Per interfaccia si intende il dispositivo fisico o virtuale che permette la comunicazione fra due o più entità di tipo diverso; ogni entità espone una sua faccia, con il suo particolare protocollo di comunicazione e il dispositivo viene interposto fra di esse.

Interfacciare significa collegare due o più dispositivi in modo che lo scambio di dati avvenga correttamente.

L'interfaccia è l'aspetto che assume ad esempio un software per far sì che l'utente riesca a comunicare e interagire con la macchina; in questo caso si parla di interfaccia utente. L'interfaccia rappresenta il componente di livello più alto di un'applicazione e, dal punto di vista dell'utente, il più critico.

È essenzialmente un collegamento tra una sorgente di informazioni e una destinazione; è la parte di sistema con cui l'utente finale interagisce. Un formato standard che consente lo scambio di dati a livello di software.

L'interfaccia del sistema operativo è il guscio più esterno, la prima cosa che incontriamo quando accendiamo il computer. La sua più comune rappresentazione è oggi una sorta di *scrivania*, sulla quale si trovano *documenti* e *strumenti*. II modello di interazione delle prime interfacce, basate su testo, si riferiva prevalentemente alla metafora del dialogo *master-servant*.

Nell'interfaccia di un sistema operativo le risorse di un computer, o di una rete di computer, sono organizzate e presentate per permetterne l'uso migliore. In funzione del lavoro da svolgere, è necessario poter accedere ai file di dati, ai programmi applicativi e a strumenti per la gestione dell'allocazione delle risorse. L'interfaccia delle singole applicazioni organizza ambienti di lavoro specializzato, dedicati allo svolgimento di compiti specifici: i programmi di videoscrittura e grafica presentano generalmente la simulazione di un *foglio di carta*, quelli per il disegno tecnico *tavoli da disegno* e *strumenti per il disegno tecnico*, i programmi per la modellazione tridimensionale presentano *mondi* tridimensionali, e così via.

L'hardware è diventato una merce indifferenziata, mentre il software è proliferato, si è diversificato ed è diventalo sempre più personalizzabile. Non c'è più motivo per cui due sistemi computerizzati debbano essere uguali, e ognuno di noi può assemblare un ambiente di lavoro che rispecchi il suo talento e i suoi particolari interessi.

Tutto questo mi fa venire in mente OpenDoc, un brillante progetto andato in fumo.

OpenDoc

OpenDoc è un framework sviluppato da Apple Computer per consentire il collegamento dinamico tra applicazioni e documenti. È ispirato alla tecnologia object linking and embedding (OLE) sviluppata da Microsoft per l'ambiente Windows.

Venne inizialmente creato da Apple Computer nel 1992 dopo una breve collaborazione tra Apple e Microsoft. Inizialmente Microsoft contattò Apple per proporle di

aderire alla tecnologia OLE 2. Apple analizzò il prototipo e le specifiche di OLE 2 e inviò a Microsoft le sue proposte di miglioramento. Microsoft non sembrò ad Apple molto ricettiva e quindi questa decise di sviluppare una tecnologia alternativa a OLE, e avviò il progetto che avrebbe portato a OpenDoc.

Il fine ultimo di OpenDoc era quello di lasciare all'utente la possibilità di disporre un ambiente di lavoro personalizzato senza doppioni e altamente integrato. Ciò sarebbe stato possibile avendo a disposizione non più le varie applicazioni bensì dei moduli specializzati. Non sarebbe stato più necessario, per esempio, avere un modulo destinato alla gestione del testo in un programma di impaginazione e un altro nel programma di disegno e un altro ancora in quello di montaggio video. Un solo modulo per il testo completamente integrato con tutto il resto. E così via per tutti gli altri moduli.

Inizialmente aveva il nome in codice "Exemplar", poi "Jedi" e in seguito "Amber" che nella release finale divenne OpenDoc. Il team di sviluppo si rese conto che per ottenere un successo nell'adesione dello standard era necessario coalizzarsi con altre società per poter spingere OpenDoc. Crearono il *Component Integration Laboratories* con IBM e WordPerfect. Nel 1996 lo standard venne adottato dall'Object Management Group.

Kurt Piersol dell'Apple Computer, il disegnatore dell'architettura di OpenDoc si scontrò con Jed Harris (futuro presidente del CILabs) che lo criticò per l'architettura dello standard. Mark Ericson di WordPerfect capo progetto del porting per Windows introdusse una interoperabilità tra OpenDoc e OLE.

OpenDoc venne inizialmente rilasciato per Mac OS System 7.5 ed era uno standard basato sui documenti e non sulle applicazioni. Alcune applicazioni furono concepite seguendo lo standard OpenDoc come, per esempio, il word processor WAV che è un parziale successo dell'architettura OpenDoc. Venne rilasciato anche Cyberdog un browser creato da Apple e utilizzante l'architettura OpenDoc e anche il software Nisus Writer sviluppato da Nisus integrava OpenDoc al suo interno. Quando IBM entrò in Taligent, decise di implementare l'architettura OpenDoc in OS/2 Warp 4.

Nonostante allo standard OpenDoc aderissero formalmente centinaia di sviluppatori, nella realtà il suo utilizzo era scarso e i pochi sviluppatori che lo adottavano non riuscirono a collaborare in modo proficuo. Apple intanto perdeva molto denaro e l'arrivo di tecnologie come Java e i JavaBeans rendevano la tecnologia sorpassata. Quando Steve Jobs ritornò in Apple decise di eliminare il progetto OpenDoc e di licenziare l'intero team di sviluppo durante la grande ristrutturazione del 1997.

Dati

Trovandosi di fronte alla scrivania di un computer si possono notare icone di file di informazioni registrate sul disco fisso. Alcune di esse rappresentano file di dati che contengono informazioni da elaborare, altre file di applicazioni che contengono i programmi per l'elaborazione stessa.

Per avere una qualche utilità pratica, una serie dì dati deve soddisfare due condizioni. Prima di tutto deve contenere le informazioni che ci servono e poi dev'essere strutturata così da permettere un accesso efficiente alle informazioni desiderate.

Le informazioni contenute in un libro, per esempio, sono accessibili, apparentemente, in almeno quattro modi diversi: in modo sequenziale; via sommario; consultando l'indice analitico; infine seguire i rimandi incrociati. Notate però che due soli elementari meccanismi sottendono tutte queste procedure: analizzare il testo sequenzialmente e seguire riferimenti alle diverse posizioni delle informazioni. Per accedere ai dati archiviati in memoria i computer si servono di questi stessi due meccanismi elementari. Organizzare le informazioni per permetterne l'elaborazione al computer consiste dunque prevalentemente nel costruire sequenze appropriate e nel creare strutture dì riferimento, chiamate puntatori, che, in maniera analoga agli indici e ai riferimenti incrociati di un libro, permettano di localizzare i dati nella memoria. Il risultante schema delle informazioni è noto come *struttura dati.* Una raccolta di informazioni archiviata in modo semi permanente nella memoria di un computer e strutturata per un particolare scopo viene chiamata *database.*

Forum

La struttura informatica contenente discussioni e messaggi, di argomento vario, scritti dagli utenti. Un Foro, una Piazza.

Forum può riferirsi all'intera struttura informatica nella quale degli utenti discutono su vari argomenti, a una sua sottosezione oppure al software utilizzato per fornire questa struttura. Un senso di comunità virtuale si sviluppa spesso intorno ai forum che hanno utenti abituali ed interessi comuni. La tecnologia, i videogiochi, la politica, l'attualità e lo sport sono temi popolari, ma ci sono forum per un enorme numero di argomenti differenti.

I forum vengono utilizzati anche come strumenti di supporto on-line per vari prodotti e all'interno di aziende per mettere in comunicazione i dipendenti e permettere loro di reperire informazioni.

Ci si riferisce comunemente ai forum anche come *board*, *message board*, *bulletin board*, *gruppi di discussione*, *bacheche* e simili.

Molti forum richiedono la registrazione dell'utente prima di poter inviare messaggi ed in alcuni casi anche per poterli leggere. Differentemente dalla chat, che è uno strumento di comunicazione sincrono, il forum è asincrono in quanto i messaggi vengono scritti e letti anche in momenti diversi.

Tutorial

Il *Computer Based Training* è un metodo di insegnamento basato sull'uso di speciali programmi didattici.

Un *tutorial* è una lezione on-line che utilizza diverse strategie per trasferire contenuti specifici. Alcune strategie di esposizione dei contenuti:

Illustrazione tramite il multimedia (animazione, video o semplici testo-immagine)
Interazioni per mantenere vivo l'interesse e consentire un apprendimento di tipo deduttivo: drag&drop, fill-in-the-blanks, quiz, ecc.
Materiali mono-mediali utilizzati per l'approfondimento.

Una lezione consente:

Apprendimento in autonomia: principalmente finalizzato all'addestramento (training).
Apprendimento on-line: stimolo per la discussione e la condivisione del sapere (collaborative learning).

Risorse

Ormai la rete, come il mondo materiale, possiede un'infinità di cose. Oggetti di qualunque fattura gratuiti o a pagamento, da soli o accompagnati. Oggetti di qualsiasi tipo veramente *dal cucchiaino alla città* e oltre. Per i siti più rappresentativi si può fare riferimento alla Sitografia allegata al presente lavoro. Mi preme qui aggiungere che Google SketchUp mette a disposizione, attraverso la 3D Warehouse, davvero un'immesità di oggetti. Esportabili in qualunque modellatore 3D considerando che il formato di esportazione verso Google Earth, il .kmz, altro non è che un file *zippato* contenente un file COLLADA al suo interno.

Per le texture vale lo stesso discorso già fatto per gli oggetti, un'infinità. Spesso anche i siti sono gli stessi. Ma poi ci sono i plug-in. Piccole sub-applicazioni in alcune circosatanze vitali. Tutto il software free.

È arrivato il momento di raccogliere un po' le idee e vedere come ci stiamo muovendo per trasmettere le informazioni da una generazione all'altra.

Programmi applicativi

Possiamo ora definire con maggior precisione la funzione principale di un calcolatore, come l'applicazione di algoritmi, sequenze di istruzioni, a file di dati e database per ottenere

i risultati richiesti. Programmare significa progettare gli algoritmi e le strutture dati su cui essi operano. Dal punto di vista dell'utente i programmi sono strumenti per elaborare file di dati e database.

Gli utenti che fanno un uso professionale del computer dovrebbero avere una comprensione critica degli algoritmi e delle strutture dati dei programmi che usano: delle loro potenzialità, dei loro limiti e dei principi teorici su cui si basano. Comprendere metodi e materiali intellettuali è per un architetto altrettanto importante di quanto sia fondamentale avere familiarità con i metodi e i materiali delle costruzioni. Ciò non significa che si debba essere degli esperti programmatori, ma almeno avere una certa familiarità con la programmazione per essere in grado di lavorare in modo efficiente.

Grafica raster

Nella grafica raster le immagini vengono descritte come una griglia di pixel opportunamente colorati. Una mappa di bit colorati, una *bitmap*.

La bitmap è caratterizzata da due proprietà: risoluzione e profondità di colore.

La prima è determinata dal numero di pixel contenuti nell'unità di misura considerata. La seconda è definita dalla memoria che si dedica ad ogni pixel, ovvero dal numero di bit dedicati ad ogni pixel per descrivere il colore, e si misura in bit per pixel; maggiore è il numero di bit, maggiore è il numero di colori che è possibile descrivere.

I dati raster possono essere memorizzati attraverso tipologie di file che, sfruttando algoritmi di compressione diversi, gravano in modo differente sul supporto di refistrazione. I formati raster più comuni sono i seguenti: bmp, dgn, gif, jpg, png, raw, tga, tif.

Grafica vettoriale

Nella grafica vettoriale un'immagine è descritta mediante un insieme di primitive geometriche che determinano punti, linee, curve e poligoni.

I principali vantaggi della grafica vettoriale rispetto alla grafica raster sono la qualità, la maggiore compressione dei dati e la più facile gestione delle eventuali modifiche.

La grafica vettoriale, essendo definita attraverso equazioni matematiche, è indipendente dalla risoluzione, mentre la grafica raster, se viene ingrandita o visualizzata su un dispositivo dotato di una risoluzione maggiore di quella del monitor, perde di definizione.

Tale sistema di descrizione delle informazioni grafiche presenta inoltre l'indubbio vantaggio di una maggiore compressione dei dati: in pratica una immagine vettoriale occuperà molto meno spazio rispetto ad una corrispondente raster, con una riduzione dell'occupazione di RAM e memoria di massa. Risulterà inoltre più facile da gestire e da modificare, essendo minore la quantità di dati coinvolti in ogni singola operazione di aggiornamento. Questo rende il vettoriale particolarmente adatto per gestire grandi quantità di dati come quelli cartografici che sono tipicamente gestiti in modalità vettoriale.

Il principale svantaggio, rispetto alla grafica raster, risiede nel fatto che la realizzazione di immagini vettoriali non è una attività così intuitiva come nel caso delle immagini raster. In realtà la *bitmap* è mutuata dal mondo reale ed è anche quella che mostra la grafica vettoriale.

Ha un notevole utilizzo nell'editoria, nell'architettura, nell'ingegneria e nella grafica realizzata al computer. Tutti i programmi di grafica tridimensionale salvano i lavori definendo gli oggetti come aggregati di primitive matematiche.

CAD

AutoCAD e friends. Il settore dell'informatica volto all'utilizzo della computer grafica per supportare l'attività di progettazione.

Con le parole di William J. Mitchell e Malcolm McCullough in *Digital Design Media*: “la moderna pietra filosofale, l'agente che muta l'intelligenza di base di un chip di silicio

nell'intelligenza d'alto livello di un sofisticato sistema *CAD*, è il software: programmi e banche dati che traducono il sapere architettonico in forma processabile da una macchina. Nell'economia dell'informazione tale *macchinario intellettuale* gioca un ruolo altrettanto importante di quello svolto dalle macchine nell'economia manifatturiera. Oggi il software è il vero e proprio strumento produttivo: in esso si deve investire per essere più competitivi. Per poterlo usare efficacemente è necessario comprenderne proprietà e limiti.

Il ruolo del software non è soltanto economico, ma anche culturale. Un programma può essere il frutto di competenza e immaginazione. Le modalità per rendere pubblico il software, distribuirlo, raccoglierlo in biblioteche diventano ogni giorno più importanti. Il software è divenuto un mezzo per accumulare conoscenza e trasmettere cultura, come i suoi predecessori: il racconto orale, il manoscritto, il libro stampato, il disco fonografico".

CAD

da Wikipedia, novembre 2009

In informatica, l'acronimo inglese CAD viene usato per indicare due concetti correlati ma differenti:

Computer-Aided Drafting, ovvero *disegno tecnico assistito dall'elaboratore*

in tale accezione indica il settore dell'informatica volto all'utilizzo di tecnologie software e specificamente della computer grafica per supportare l'attività di disegno tecnico (*drafting*). I sistemi di *Computer Aided Drafting* hanno come obiettivo la creazione di un modello, tipicamente 2D, del disegno tecnico che descrive il manufatto, non del manufatto stesso. Ad esempio, un sistema *Computer Aided Drafting* può essere impiegato da un progettista nella creazione di una serie di disegni tecnici (in proiezione ortogonale, in sezione, in assonometria, in esploso) finalizzati alla costruzione di un motore;

Computer-Aided Design, ovvero *progettazione assistita dall'elaboratore*

in questa accezione, la più comune, CAD indica il settore dell'informatica volto all'utilizzo di tecnologie software e in particolare della computer grafica per supportare l'attività di progettazione (*design*) di manufatti sia virtuale che reali. I sistemi di *Computer Aided Design* hanno come obiettivo la creazione di modelli, soprattutto 3D, del manufatto. Ad esempio, un sistema *Computer Aided Design* può essere impiegato da un progettista meccanico nella creazione di un modello 3D di un motore. Se viene realizzato un modello 3D, esso può essere utilizzato per calcoli quali analisi statiche, dinamiche e strutturali ed in tal caso si parla di Computer Aided Engineering (CAE), disciplina più vasta di cui il CAD costituisce il sottoinsieme di azioni e strumenti volti alla realizzazione puramente geometrica del modello.

I diversi criteri di classificazione dei sistemi CAD

I sistemi CAD possono essere classificati secondo differenti criteri. Guardando all'estensione del dominio, inteso come campo di utilizzo, si può distinguere tra:

Sistemi CAD orizzontali

Si tratta di sistemi CAD aventi un dominio molto ampio, utilizzabili con successo in contesti applicativi differenti, come ad esempio progettazione architettonica e quella meccanica. I comandi offerti da questi sistemi sono indipendenti da uno specifico contesto applicativo. Si avranno pertanto comandi come *traccia-linea* senza alcuna nozione se la linea rappresenta una parete di un edificio o lo spigolo di un supporto metallico.

Sistemi CAD verticali

Si tratta di sistemi con dominio ristretto, orientati ad un particolare contesto applicativo, con comandi e funzionalità specifici per quel contesto. Ad esempio, un sistema CAD verticale per la progettazione di interni offrirà comandi per creare e posizionare differenti tipi di pareti e collocarvi porte e finestre. I CAD orientati all'ambito industriale e in special modo alle costruzioni meccaniche in senso lato vengono indicati come MCAD.

Una classificazione alternativa, molto utilizzata in ambito commerciale, suddivide i sistemi CAD in tre fasce principali sulla base di prezzo e funzionalità:

Sistemi di fascia bassa

Sono sistemi CAD tipicamente limitati al disegno 2D, venduti a prezzo contenuto (indicativamente inferiore ai 300€) e rivolti ad utenti occasionali o non professionisti.

Sistemi di fascia medio-bassa

Sono sistemi CAD tipicamente limitati al disegno 2D, integrano vari moduli e permettono di gestire proprietà del disegno, venduti a prezzo contenuto (indicativamente inferiore ai 1500€) e rivolti a professionisti artigiani, piccole aziende, impiantisti e tutti coloro che non fanno della progettazione il proprio "core business".

Sistemi di fascia media

Sono sistemi CAD che integrano il disegno 2D con la modellazione 3D, venduti ad un prezzo medio (indicativamente inferiore ai 5000€). Questi sistemi sono usualmente rivolti a piccole o medie aziende e a professionisti, e vengono spesso integrati con moduli "verticali", cioè particolarmente adatti alla velocizzazione dei compiti giornalieri. Spesso sono integrati inoltre con una suite di strumenti come il PDM per la gestione dei dati riguardanti i prodotti progettati (Product Lifecycle Management).

Sistemi di fascia alta

Sono sistemi CAD complessi che integrano la modellazione 3D con il disegno 2D, e offrono una gestione avanzata dei dati supportando processi aziendali che si estendono ben oltre l'ufficio tecnico. Hanno costi elevati e sono tipicamente utilizzati dalle medie e grandi aziende, come per i sistemi di fascia media, anche con un PDM.

Pressoché tutti i sistemi CAD possono essere personalizzati ed estesi al fine di migliorare la produttività dei progettisti e la qualità e dei progetti. Le principali modalità per estendere un sistema CAD sono:

Librerie

Collezioni di modelli di oggetti e simboli da utilizzare nel progetto. Per esempio, un CAD per arredatori può contenere una libreria di mobili. Ogni mobile può essere copiato dalla libreria e posizionato nel progetto di un arredamento.

Macro

Comandi ottenuti componendo comandi più semplici tramite un linguaggio di programmazione. Per esempio, in un sistema CAD 2D per fornire la funzione di disegno di muri, una macro può chiedere all'utente di inserire il punto iniziale, il punto finale e lo spessore del muro, e inserire automaticamente nel modello due linee parallele che rappresentano il muro.

Verticalizzazioni

I CAD, in particolare quelli di alte fasce, possono gestire proprietà ed informazioni dei progetti ottenuti per personalizzarli, presentarli con video o immagini (rendering), oppure per calcolarne le proprietà fisiche e geometriche (analisi di interferenze, simulazioni dinamiche, analisi agli elementi finiti -f.e.m.- ecc.) integrandosi con strumenti CAE (Computer-aided engineering).

Settori d'impiego

Architettura, urbanistica, ingegneria civile: progettazione di costruzioni.

Arredamento: progettazione di interni.

Elettrotecnica e meccanica: progettazione di apparecchi elettrici o meccanici.

Industrial design: progettazione di oggetti di consumo, come mobili o attrezzi casalinghi, recentemente anche abbigliamento.

Impiantistica: progettazioni di tubazioni cablaggi e impianti di condizionamento.

Elettronica: progettazione di circuiti elettronici, a livello di schema elettrico, di circuito integrato, di circuito stampato, o di intero sistema.

Storia

Probabilmente, l'antenato dei sistemi di CAD è stato il sistema Sketchpad sviluppato al Massachusetts Institute of Technology nel 1963 da parte di Ivan Sutherland. Si trattava di un sistema sperimentale che consentiva al progettista di disegnare su un monitor a raggi catodici con una penna ottica.

Le prime applicazioni commerciali del CAD si ebbero negli anni 1970 in grandi aziende elettroniche, automobilistiche, o aerospaziali. Venivano impiegati computer mainframe e terminali grafici vettoriali. Questi ultimi sono monitor a raggi catodici il cui pennello elettronico, invece di scandire lo schermo come nei televisori, viene controllato dal computer in modo da tracciare le linee.

Negli anni 1980 vennero sviluppati sistemi CAD per microcomputer con monitor a grafica raster, cioè basate su frame buffer. Tali sistemi erano ancora o molto limitati o molto costosi, e comunque molto difficili da usare, per cui venivano usati solo da aziende medio-grandi o da professionisti, essendo questi strumenti tecnologicamente sofisticati.

Negli anni 1990 la semplificazione nell'uso del computer dovuto alla diffusione delle interfacce utente grafiche e l'abbassamento dei costi dell'hardware hanno reso i sistemi CAD alla portata di tutti i professionisti.

Funzionalità Principali dei Sistemi CAD 2D

I sistemi CAD per il disegno 2D offrono un insieme di comandi che, benché presentati all'utente con interfacce e nomi differenti da un sistema all'altro, sono riconducibili ad un nucleo comune. Molte di queste sono in realtà funzioni offerte anche dai sistemi CAD che operano in tre dimensioni.

Disegno

I sistemi per il disegno offrono comandi per il disegno di elementi grafici elementari e comandi più potenti che consentono al disegnatore di realizzare con rapidità elementi grafici più complessi. Questi comandi sono usualmente potenziati dall'abbinamento con modalità operative basate su sistemi di riferimento alternativi e dalla riferibilità di punti notevoli.

Disegno di entità grafiche elementari

Questi sono i mattoni di costruzione che il sistema CAD e l'utente utilizzano per costruire disegni 2D, dai più semplici ai più complessi. In tutti i sistemi CAD 2D sono presenti comandi per il disegno di semplici geometrie quali linea, segmento, arco, circonferenza, ecc.

Disegno di entità grafiche composte

Sono usualmente disponibili comandi di alto livello per la rapida realizzazione di strutture grafiche più complesse come poligoni regolari di *n* lati inscritti o circoscritti ad un cerchio, rette perpendicolari, parallele o bisettrici, raccordi, quote, ecc. Particolare attenzione viene posta dagli sviluppatori di sistemi CAD nella implementazione delle funzionalità di quotatura. I disegnatori sono molto esigenti e richiedono comandi per la quotatura che siano di facile utilizzo e al contempo fortemente personalizzabili così da adattarsi a norme, gusti estetici ed esigenze di ciascun utente o gruppo di utenti.

Utilizzo di sistemi di coordinate definiti dall'utente

Nella realizzazione di un disegno è fondamentale l'utilizzo di sistemi di coordinate alternativi come ad esempio coordinate cartesiane relative, coordinate polari, distanze da altre geometrie, ecc. Meno importante, nei sistemi 2D, è la creazione di coppie / terne cartesiane poste in vari punti del disegno ed attuabili dall'utente con il corrispondente sistema di riferimento. Nei sistemi CAD 3D, questa stessa funzionalità è considerata irrinunciabile in quanto consente al disegnatore di operare su un piano di lavoro liberamente posizionato nello spazio oppure coincidente con una faccia preesistente.

Punti notevoli

Si tratta di comandi che abilitano la selezione di punti che sono univocamente individuabili sul disegno pur non essendo rappresentati esplicitamente in memoria come entità geometriche. Ad esempio, il punto medio di un segmento pur non essendo rappresentato e memorizzato dal sistema CAD come entità geometrica può essere riferito come centro nella procedura di costruzione di una circonferenza. La selezione di punti notevoli rende più veloce ed estremamente precisa la realizzazione di un disegno. Esempi di punti notevoli sono:

il centro di circonferenze ed archi

gli estremi di un segmenti e archi

i punti medi di segmenti e archi

l'intersezione di segmenti ad archi.

I comandi per la selezione di punti notevoli non sempre sono invocati direttamente dall'utente, possono essere attivati automaticamente dal sistema CAD in corrispondenza di altri comandi che richiedono l'acquisizione di punti e/o vertici.

Attributi grafici

Per ricreare a schermo la grande varietà di linee utilizzate dal disegno tecnico i sistemi CAD consentono di selezionare gli attributi di tracciamento di ciascuna entità grafica, sia essa un segmento, un arco, o altro. Usualmente il tipo di tratto (continuo, tratteggiato, ecc.) viene visualizzato direttamente sullo schermo, mentre il differente spessore delle linee viene usualmente rappresentato graficamente sullo schermo utilizzando linee di spessore uniforme ma di colori differenti. La corrispondenza tra colore e spessore viene ripristinata al momento della stampa.

Strutturazione del disegno

I sistemi CAD non si limitano alla sola automatizzazione delle attività tradizionali del disegno ma offrono anche funzionalità di strutturazione del disegno possibili solo con l'ausilio di strumenti informatici. Il disegno, pertanto, cessa di essere un insieme uniforme di entità grafiche per divenire una struttura anche complessa di aggregazioni di entità arricchite di attributi grafici e del contesto applicativo, come ad esempio materiali, note di lavorazione, costi, ecc. Queste funzionalità vengono proposte all'utente del sistema CAD come funzionalità supplementari: egli è responsabile di deciderne il migliore utilizzo in funzione delle proprie esigenze e delle modalità di lavoro dell'ambiente professionale in cui opera. I principali strumenti di strutturazione del disegno offerti dai attuali sistemi CAD sono i seguenti:

Strutturazione in livelli (layer)

Il disegno, tipicamente 2D, può essere strutturato con la creazione di strutture *orizzontali* corrispondenti ad insiemi logici di entità grafiche. Ad esempio, in un progetto di ingegneria civile si collocano su livelli distinti: pianta dell'edificio, rete idrica, rete elettrica, rete idraulica, ecc. Ciascuno strato o livello (layer) raggruppa entità affini ma non necessariamente

appartenenti allo stesso componente dell'oggetto. I livelli sono gestiti con meccanismi che consentono di controllarne la visibilità individuale come se si trattasse di fogli trasparenti sovrapponibili.

Strutturazione in gruppi

Un'altra tecnica di strutturazione del disegno, non necessariamente alternativa ai livelli, consiste nel riunire le entità grafiche in gruppi sulla base di affinità funzionali o in base all'appartenenza ad un medesimo componente dell'oggetto. L'operazione di raggruppamento può essere iterata a comporre gruppi di gruppi. Questo comando consente di ricreare nel disegno la strutturazione tipica di un assemblato di oggetti reali, in cui ogni parte appartiene ad un sotto-assieme che a sua volta si colloca in un insieme più ampio.

Referenziazione (simboli, blocchi)

Un'altra tecnica di strutturazione consiste nell'inserimento nel disegno di riferimenti a componenti (simboli) definiti esternamente al disegno stesso e comunque modificabili separatamente da questo. Nel disegno, ciascun riferimento che rimanda ad un simbolo in libreria è detto istanza del simbolo. Con questa tecnica è possibile inserire nel disegno dei particolari standardizzati, usualmente definiti in una libreria esterna, in cui ciascuna istanza è posizionata e visualizzata come entità grafica indipendente, con la certezza dell'assoluta corrispondenza di ciascuna istanza con la descrizione primaria presente in libreria. Mentre è impossibile modificare singolarmente l'istanza di un simbolo, se non nei suoi parametri di posizionamento e scalatura, qualora si volessero modificare tutte le istanze è sufficiente modificare l'elemento originale ottenendo una propagazione automatica a tutte le istanze.

Modifica del disegno

Uno dei più evidenti vantaggi nell'utilizzo di un sistema CAD rispetto all'impiego di tecniche tradizionali, consiste nella grande facilità e rapidità con cui è possibile modificare, anche in modo radicale un disegno per correggerlo o per creare una versione. Le principali funzionalità di modifica del disegno sono:

Cancellazione di entità

Tutti i sistemi CAD consentono di cancellare le entità grafiche del disegno selezionandole individualmente, selezionando tutte le entità racchiuse in una certa area rettangolare, oppure agendo per categorie (ad esempio, tutti i segmenti gialli) o per strutture (ad esempio, tutte le entità del livello 25).

Modifica degli attributi di una entità

A volte modificare un disegno significa cambiare gli attributi grafici, come colore o tipo di linea, di alcune entità grafiche. Nei disegni strutturati è anche possibile portare una o più entità da un gruppo o da un livello ad un altro oppure modificare gli attributi grafici di tutte le entità appartenenti ad uno stesso gruppo oppure residenti su uno stesso layer.

Trasformazione

Tutte le entità grafiche e gli insiemi di entità possono essere modificati con opportune trasformazioni. Sono usualmente disponibili le consuete trasformazioni lineari di scalatura, traslazione, rotazione, specularità e le combinazioni di queste. Le modalità con cui queste trasformazioni sono rese disponibili all'utente possono essere le più varie.

Riorganizzazione della tavola

Utilizzando le funzionalità di trasformazione una tavola può essere rapidamente riordinata o modificata ad esempio, per ospitare una nuova vista. In queste operazioni il disegnatore è spesso supportato dalla presenza di più viste a diversi livelli di zoom; si concilia così l'esigenza di effettuare operazioni localmente molto precise conservando una visione globale della tavola. Un'altra possibilità di riorganizzazione della tavola consiste nella modifica di livelli o gruppi, per ottenere una strutturazione meglio aderente alle esigenze del disegnatore ed alle caratteristiche strutturali e funzionali dell'oggetto.

Gestione di parti ricorrenti

L'utente di un sistema CAD può velocizzare in modo significativo il proprio lavoro creando degli speciali archivi, detti librerie, in cui raccogliere i disegni o i particolari di utilizzo più frequente. Questa possibilità fornisce un reale riscontro in termini di benefici economici e qualitativi solo se il disegnatore opera in un contesto regolamentato da precise norme ed è supportato da un'adeguata organizzazione nonché dalla disponibilità di sufficienti risorse.

Librerie di normalizzati

L'accesso ad archivi o librerie di parti normalizzate, disponibili in più viste e in vari formati e direttamente inseribili nel disegno, consente di realizzare con rapidità e precisione anche tavole molto complesse. Le librerie di normalizzati sono realizzabili direttamente dal disegnatore oppure possono essere acquistate dal produttore del sistema CAD o da terze parti.

Librerie di parti ricorrenti

Queste librerie, del tutto analoghe alle librerie di normalizzati, sono specifiche di ciascuno studio di progettazione e pertanto sono costruite direttamente dai singoli utenti. In queste librerie si accumula un patrimonio di disegni che rappresentano parti o sotto-parti ricorrenti archiviate e catalogate in un formato che le rende facilmente reperibili e riutilizzabili con evidenti vantaggi in termini di produttività.

Riutilizzo di disegni

La possibilità di duplicare disegni esistenti per generare nuovi disegni mediante opportune cancellazioni e modifiche costituisce un'altra utile possibilità di utilizzo dei sistemi CAD in particolare qualora il disegnatore si trovi a realizzare disegni che presentano forti similarità con tavole prodotte in precedenza.

Interrogazione del disegno

Il disegno creato con un sistema CAD deve essere utilizzabile non solo come rappresentazione grafica ma anche sorgente di informazione sul progetto. È importante che sia consentito l'accesso a tutta l'informazione contenuta nel disegno, sia essa in forma esplicita o implicita. Le informazioni estraibili da un modello 2D sono limitate, soprattutto se paragonate alle informazioni estraibili da un modello 3D che rappresenta il medesimo pezzo. Un modello 2D di un ingranaggio può contenere tutte le informazioni necessarie alla manifattura della ruota, ma solamente un modello 3D del medesimo ingranaggio potrà essere interrogato per estrarre informazioni circa il volume, il baricentro, ecc. Le funzionalità di interrogazione del modello CAD sono indispensabili, ad esempio, per la generazione di programmi di lavorazione per la produzione del pezzo con una macchina utensile a controllo numerico. Le principali classi di interrogazione supportate dai sistemi di disegno sono:

Interrogazione della geometria

Tutti i sistemi CAD orientati al disegno offrono la possibilità di conoscere, per le entità grafiche nel disegno, angoli, lunghezze, distanze, raggi, coordinate, ecc., anche se non definiti esplicitamente. Ad esempio è possibile costruire una circonferenza con tre vincoli di tangenza ed una volta tracciata richiedere al sistema CAD di conoscere il valore del raggio o del diametro. Alcuni sistemi CAD offrono il calcolo automatico di aree definite da profili chiusi. Questa funzionalità può essere di rilievo non solo per il disegnatore ma anche per il progettista.

Stima dei costi e della complessità

Le capacità di interrogazione del modello possono essere utilizzate per automatizzare alcune attività, come, ad esempio, per calcolare una stima dei costi di produzione dell'oggetto e generare automaticamente la distinta base con il conteggio dei componenti presenti e della loro numerosità.

Accesso esterno al modello

Può essere considerata una forma di interrogazione anche la possibilità di accedere a tutte le informazioni contenute nel modello CAD per mezzo di programmi esterni realizzati dagli stessi utenti. A questo scopo numerosi sistemi CAD offrono delle interfacce di

programmazione dette API (Application Programming Interface). Utilizzando queste interfacce un programmatore può accedere a tutte le funzionalità del sistema CAD oppure ad un suo sotto insieme per mezzo di chiamate a funzioni nel contesto di un programma scritto in un linguaggio di programmazione.

Automatizzazione di attività ripetitive

La realizzazione di un disegno comprende operazioni particolarmente ripetitive e tediose che possono essere facilmente automatizzabili da semplici programmi. I sistemi CAD offrono alcuni comandi che consentono di sgravare il disegnatore dall'esecuzione di queste parti limitandone l'intervento umano all'impostazione di pochi parametri iniziali. Le attività più comunemente automatizzate sono:

Campiture

Tutti i sistemi CAD offrono adeguati comandi per campire automaticamente un profilo chiuso con un tipo di campitura selezionato o definito dall'utente. L'intervento del disegnatore si limita alla selezione del profilo ed alla scelta o definizione del tipo di campitura.

Pattern a Array

Un'altra attività ripetitiva è il disegno di motivi circolari o rettangolari di elementi costanti come, ad esempio, la sequenza di fori posti circolarmente su una flangia. I sistemi CAD sono in grado di posizionare automaticamente questi elementi ricorrenti, richiedendo al disegnatore la selezione dell'elemento ripetuto e le regole che governano il posizionamento.

Quotatura associativa

Alcuni sistemi supportano la creazione di quote legate dinamicamente ad entità geometriche, con aggiornamento automatico di posizione e valore al variare delle entità quotate. Queste quote, dette associative, contribuiscono a velocizzare la produzione di disegni soggetti a frequenti modifiche o riutilizzati nella generazione di varianti.

Gestione di archivi

Un aspetto, spesso sottovalutato, dell'utilizzo dei sistemi CAD sono le funzionalità di archiviazione dei disegni. Queste funzionalità frequentemente sono presenti nel sistema CAD solo con implementazioni essenziali. Versioni più estese sono disponibili mediante moduli software esterni. Con questi strumenti è possibile organizzare gli archivi così da consentire un accesso rapido ed organizzato al patrimonio di disegni di ciascuno studio di progettazione o ufficio tecnico. Le funzioni offerte dagli strumenti di archiviazione sono:

Memorizzazione

La memorizzazione di disegni, o documenti tecnici, su supporto magnetico oppure ottico, è una funzionalità di base. Alcuni sistemi consentono la semplice creazione di un file lasciando all'utente la responsabilità di organizzare da sé la gestione dell'archivio di disegni. Altri sistemi, ad un livello maggiore di integrazione, gestiscono il disegno nel contesto di una base dati; il sistema gestisce i permessi e le modalità di accesso e di riferimento al disegno, organizzando automaticamente le versioni successive e predisponendo i meccanismi di condivisione tra più utenti e di accesi multipli contemporanei. Questi archivi strutturati sono solitamente abbinati a strumenti informatici di navigazione per il reperimento rapido delle tavole e delle informazioni ad esse associate. Queste funzionalità consentono di utilizzare in modo produttivo il patrimonio di disegni esistenti e di ridurre in modo significativo lo spazio richiesto rispetto a quello richiesto dalle tecniche tradizionali di archiviazione.

Classificazione

La classificazione semiautomatica delle tavole con associazione di documenti ausiliari relativi alla documentazione tecnica, come la distinta base, consente di automatizzare e razionalizzare i meccanismi di archiviazione garantendo una reale reperibilità dei documenti.

Trasporto

I documenti, o disegni, in forma digitale possono essere inviati in luoghi diversi da quello originale in tempi ridottissimi e con degrado della qualità nullo. Le tavole possono essere inviate localmente da un ufficio ad un altro o da un edificio ad un altro in tempo reale utilizzando le reti locali di elaboratori (LAN) mentre possono essere inviate da una differente località geografica via internet.

Interscambio dati

La possibilità di scambiare dati tra sistemi CAD diversi e tra sistemi CAD e sistemi per il CAM (Computer Aided Manufacturing), costituisce un elemento fondamentale nella valutazione delle funzionalità di un sistema. Per lo scambio di dati tra sistemi diversi, sono percorribili due strade alternative: realizzare un convertitore da ciascun sistema CAD verso tutti gli altri sistemi CAD esistenti oppure concordare un formato dati neutrale e realizzare, per ciascun sistema CAD, due convertitori: uno in grado di convertire i dati dalla rappresentazione interna nel formato neutrale ed uno in grado di convertire il formato neutrale nella rappresentazione interna del sistema. Risulta evidente l'economicità della seconda soluzione rispetto alla prima. Numerosi formati neutrali di dati sono stati proposti nel corso degli anni ma nessuno di essi si è imposto con una diffusione sufficiente sugli altri. Attualmente sono utilizzati alcuni formati definiti da standard ufficiali, come IGES, VDA-FS, STEP, ecc., ed altri standard definiti da standard de facto come il formato DXF. Alcuni produttori di CAD utilizzano anche il formato PDF. Fra i nuovi formati in via di affermazione, soprattutto per lo scambio di modelli 3D con le informazioni grafiche e numeriche vi è il formato IFC Le funzionalità offerte dall'impiego di questi formati di dati consentono al sistema CAD di:

Scambiare informazioni con altri sistemi CAD

Questo scambio può avvenire per molteplici motivi: passaggio ad un sistema più evoluto o di altro produttore, scambio dati di progetto con fornitori, utilizzo di ambienti di progettazione multi fornitore, ecc.

Scambiare informazioni con strumenti per la documentazione tecnica

Diviene sempre più sentita l'esigenza di riversare i modelli prodotti dalla progettazioni verso strumenti per la produzione di documentazione tecnica automatizzando in questo modo la produzione di illustrazioni, schemi, ecc. Questo tipo di scambio non richiede una precisione particolare, come è invece per altri casi: infatti attualmente si utilizzano strumenti non specifici e spesso approssimati.

Scambiare informazioni con strumenti di analisi e verifica

Anche il trasferimento in tempi rapidi di un modello CAD a strumenti per l'analisi strutturale o per il calcolo di altro tipo è divenuta una esigenza molto sentita dai progettisti.

Scambiare informazioni con sistemi CAM

Questo è un punto fondamentale, infatti i sistemi CAM devono poter operare su dati di massima precisione e con tempi di scambio molto contenuti. Con una fedele conversione dei dati si pongono i presupposti per una corretta esecuzione della lavorazione a controllo numerico.

Personalizzazione dell'ambiente

Non è possibile produrre dei sistemi CAD che soddisfino perfettamente le esigenze specifiche ed i gusti di tutti i potenziali utenti. Per questa ragione ciascun sistema offre agli utenti la possibilità di modificare sia le modalità di interazione che lo stile del disegno. Il livello di configurabilità varia da sistema a sistema. Questa funzionalità viene sempre più considerata una caratteristica irrinunciabile. Le principali possibilità di configurazione o personalizzazione sono:

Configurazione dei parametri generali

Con la scelta di opportuni valori per i parametri di sistema, è possibile adattare le modalità di interazione

e l'aspetto del sistema ai gusti dell'utente limitatamente alle caratteristiche configurabili dal sistema utilizzato. Ad esempio è possibile associare comandi di uso frequente a combinazioni di tasti o posizionare le corrispondenti icone in zone rapidamente accessibili dello schermo.

Configurazione dello stile

Con la selezione di opportuni valori per i parametri utente, è possibile adattare lo stile di disegno adottato dal sistema CAD alle preferenze del disegnatore ed alle convenzioni interne di uno specifico studio di progettazione o ufficio tecnico. Si possono ad esempio configurare i parametri relativi allo stile di quotatura, allo stile dei testi, al cartiglio standard, ecc.

Integrazione con moduli specializzati

Tutti i sistemi CAD sono estendibili fornendo al disegnatore, entro il sistema stesso, l'accesso a moduli specializzati, usualmente realizzati da terze parti, per contesti applicativi specifici. Ad esempio, un disegnatore di impianti elettrotecnici potrà acquisire un modulo per la verifica automatica di alcune caratteristiche dell'impianto progettato, integrato nel sistema CAD.

Programmazione di funzioni specifiche

Per esigenze specifiche del singolo disegnatore o dello studio di progettazione, i sistemi CAD offrono la possibilità di estendere l'insieme dei comandi con opportuni programmi, detti comunemente macro, codificati direttamente dagli utenti o acquistati da terze parti. Questa possibilità, pur essendo teoricamente molto interessante, si scontra con la difficoltà che gli utenti, progettisti e disegnatori, incontrano nell'utilizzo dei linguaggi di programmazione, gli unici strumenti per accedere a questa capacità di potenziamento del sistema CAD.

Visualizzazione

L'attuale dimensione degli schermi per computer non è in alcun modo paragonabile alla dimensione di un tecnigrafo oppure di un foglio di formato A0; pertanto i sistemi CAD sono costretti ad offrire modalità alternative per la visualizzazione dei disegni. Le funzionalità essenziali di visualizzazione, nei sistemi 2D, sono analoghe a quello che potremmo ottenere osservando un foglio da disegno con una macchina fotografica o con una telecamera: operando sull'obiettivo si può ingrandire o rimpicciolire a piacere il disegno passando da una visione globale dell'intero disegno ad una visione locale di una sua sottoparte; inoltre spostando orizzontalmente o verticalmente la telecamera è possibile variare l'area del disegno inquadrata. Si noti che si tratta di funzioni di visualizzazione, cioè di funzioni che modificano la vista del disegno e non il disegno. Le principali funzionalità per il controllo della visibilità sono:

Zoom

L'utente del sistema CAD può ingrandire o rimpicciolire a piacere parte o tutto il disegno senza per questo perdere in precisione sia nell'immagine che appare sul video che nel risultato prodotto dai comandi impartiti al sistema.

Pan

Con questo termine si intende generalmente l'insieme di funzioni che consento all'utente del sistema CAD di muovere orizzontalmente e/o verticalmente la telecamera virtuale con cui osserva il disegno ad inquadrare i vari dettagli. Le possibili modalità operative con cui questa funzione è resa disponibile all'utente sono molto varie:

barre di scorrimento (scroll bar) poste ai lati dell'area di visualizzazione

utilizzo di comandi impartiti da tastiera o di tasti funzionali (FrecciaSu, FrecciaGiú, ecc.)

trascinamento (drag) del disegno direttamente con il dispositivo di puntamento (mouse).

Viste multiple

Alcuni sistemi CAD 2D, offrono al disegnatore l'opportunità di operare contemporaneamente sul medesimo modello da due viste differenti, usualmente corrispondenti a due finestre grafiche. Questa modalità operativa consente di controllare agevolmente parti del disegno poste a volte in punti molto distanti del medesimo foglio e che pertanto richiederebbero l'impiego di un fattore di zoom inaccettabile per essere inquadrati contemporaneamente su un unico schermo.

Disegni multipli

In numerosi sistemi CAD, il disegnatore, questo può operare contemporaneamente su più disegni, usualmente posti su finestre distinte, effettuando operazioni di copia e incolla da un disegno ad un altro. Questa è una funzionalità diffusasi solo recentemente e che consente una significativa velocizzazione delle attività di integrazione di più disegni e di modifica in generale.

Durante le fasi di progettazione esecutiva e di produzione degli elaborati progettuali si profonde grande impegno nella progettazione concettuale, nello studio della forma dell'edificio e nella determinazione più conveniente dei costi. La progettazione concettuale riveste enorme importanza per la progettazione architettonica e strutturale e si fonda su di un flusso continuo di informazioni digitali dell'edificio passando dalla modellazione concettuale alla progettazione dettagliata e collega i due ambiti con gli enormi vantaggi di questa correlazione. Lo studio concettuale del progetto è sviluppato con profonda conoscenza delle materie di analisi e calcolo strutturale che consentono di creare e modulare le forme geometriche ed i moduli costruttivi. Quando il progetto concettuale è completo, il modello viene riprodotto in formato di file CAD standard ed adoperato per la progettazione dettagliata. Quando il progetto giunge alla fase dove occorrono strumenti per la progettazione dettagliata ma si verifica una modifica imprevista del progetto concettuale (dovuta ad una revisione richiesta da parte della committenza), si rivede il modello concettuale, lo si importa nel modello di progettazione dettagliata e si sincronizzano i modelli oppure si modificano gli elementi del progetto dettagliato sulla base del progetto concettuale revisionato. Il progettista sviluppa in modo indipendente i modelli concettuali e li mappa direttamente nei componenti del modello architettonico con l'evolversi del progetto, dal concetto iniziale all'esecuzione, pervenendo così ad una soluzione finale che contempera le fasi di progettazione concettuale e di dettaglio attraverso la modellazione concettuale. In fase di sviluppo del progetto, il progettista converte le superfici delle masse che definiscono gli edifici in componenti architettoniche che, sia pure non vincolate alle superfici, mantengono le relazioni tra la geometria del modello concettuale e i componenti dell'edificio in modo da effettuare eventuali variazioni finanche negli elaborati progettuali. Il

progettista lavora in modo fluido tra modello concettuale e modello architettonico in maniera tale da definire contemporaneamente sia il modello concettuale che il modello di progettazione dettagliato al momento stesso della concettualizzazione superando così la consueta separazione tra le due fasi di progettazione.

Il progettista opera con sistemi completi e specifici per la progettazione e la documentazione che supportano tutte le fasi della documentazione di progetto e di costruzione a partire dagli studi concettuali per arrivare ai disegni ed agli abachi di costruzione più dettagliati fornendo vantaggi competitivi immediati, maggiore coordinamento e qualità e contribuendo ad una maggiore redditività del proprio operato professionale. Durante il processo di modellazione, le attività progettuali rivelano l'intento di cogliere tutti gli intenti progettuali originali e si avvalgono delle caratteristiche proprie del concetto di modello di progetto digitale e di tutte le innovazioni che conferiscono note di avanguardia ed eccellenza al mondo della progettazione.

L'essenza della progettazione è costituita dalle relazioni che possono essere incorporate nel modello dell'edificio: la creazione e manipolazione di queste relazioni costituiscono l'atto stesso del progettare e gli elementi parametrici offrono accesso diretto a queste relazioni e rappresentano un modo naturale e intuitivo di pensare agli edifici utilizzando un computer.

Le prestazioni professionali del progettista hanno l'intento specifico di includere e coordinare informazioni costruttive processabili come abachi, costi, scopi progettuali, prestazioni ingegneristiche con la produzione di tutti i dati grafici dell'edificio in una struttura logica arricchita da grafiche architettoniche 3D.

Si combina il modello di progetto, dotato di propria geometria e dati, con un modello comportamentale in cui si opera la gestione delle modifiche così che l'intero modello di costruzione e la serie completa di documenti progettuali faranno parte di un database integrato, in cui tutto è parametrico, tutto è interconnesso e sussiste l'autocoordinamento in tempo reale in ogni elaborato della struttura modellata: questa associatività bidirezionale e immediata conferisce elevata qualità, coerenza e affidabilità alla produzione degli elaborati di progetto strutturale ottenuti con l'ausilio dei processi digitali per la progettazione, l'analisi e la documentazione.

L'intento del progettista si manifesta in forme di modellazione strutturale che recepiscono la vera essenza del progetto attraverso l'esame completo e la migliore progettazione dell'edificio nella cui modellazione traspare l'ottimizzazione progettuale con lo sviluppo e lo studio simultanei delle varie alternative progettuali del singolo modello e con le opzioni di visualizzazione, quantificazione e analisi che tengono traccia di tutte le correlazioni all'interno delle versioni del progetto. Nella prassi progettuale, il progettista adotta modelli di costruzione digitale che contengono tutte le informazioni necessarie per l'analisi e la valutazione delle prestazioni dell'edificio oltre a tutti i dati necessari per supportare l'analisi progettuale durante l'esecuzione del progetto. I modelli sviluppati contengono i richiesti livelli di dettaglio e affidabilità atti a completare molto velocemente tutte le analisi durante il ciclo progettuale in modo da ottenere direttamente un immediato feedback sulle alternative progettuali fin dalle prime fasi del processo progettuale. Attraverso specifiche architetture di dati costruite intorno al modello parametrico dell'edificio il progettista elabora progetti precisi e affidabili, immediati e completamente coordinati. La coordinazione delle modifiche ed il mantenimento della coerenza degli elaborati progettuali permettono al progettista di concentrare maggiori risorse sul progetto e sulla gestione delle modifiche integrando tutte le funzionalità e la sicurezza di gestione che sono fondamentali nei propri documenti di disegno. La modellazione parametrica dell'edificio consente al progettista di incorporare decisioni progettuali e di dettagli nel modello digitale in modo che l'intento essenziale sia rappresentato negli elaborati di progetto con la maggiore efficienza possibile. Nel contempo si implementano anche strategie di accesso al modello progettuale di applicazioni quali l'analisi dell'energia, la stima e le specifiche dei costi e le tecniche per la riduzione della complessità del modello necessarie alla gestione di progetti di elevate dimensioni.

Applicazioni

Premesso che ulteriori riferimenti vanno ricercati in Sitografia ricordiamo fra i maggiori software del settore:

AutoCAD. Come poterlo dimenticare. È il programma CAD per antonomasia. Qualcuno direbbe paradigmatico. E con tutto queste cose e tante altre che si porta appresso non può certo essere più tanto competitivo. Ma l'Autodesk è ricca e riesce a comprare tutto ciò che di interessante e sensato compare sul mercato. Verso la fine degli anni '80, durante un viaggio in Inghilterra, mi capitò di leggere degli annunci di lavoro nei quali si richiedeva la conoscenza di sistemi CAD di AutoCAD. A quel tempo, quasi a digiuno di informatica, li consideravo complementari. Comunque, nel bene e nel male, AutoCAD ci ha traghettato dal cartaceo al digitale. E patisce questa condizione ibrida. Nessun tipo di problema nel gestire file molto grandi.

Dopo AutoCAD si deve necessariamente parlare di MicroStation. Salvo qualche rara eccezione il programma della Bentley è di gran lunga superiore a quello dell'Autodesk. Da anni oramai. Senza entrare nello specifico c'è da dire che MicroStation segue maggiormente le richieste dell'utenza.

ArchiCAD, già Radar, è il terzo dell'elenco. Uno fra i primi ad utilizzare un sistema complesso basato su oggetti. Non vengono più considerate le semplici primitive geometriche bensì le parti componenti l'edificio. La gestione del lavoro è più orientata ad un processo di organizzazione di tipo architettonico. Problemi nel gestire file complessi. Nato per avere un feedback veloce sulla costruzione architettonica tridimensionale si rivela più utile per produrre bidimensionali canonici.

Vectorworks, già Minicad, è il quarto. Altro CAD proveniente dal mondo Macintosh. Minicad fu concepito come programma di grafica tridimensionale, sfruttando le capacità delle macchine della Apple. Nel corso degli anni si è evoluto ma permane una certa inaffidabilità.

Rhinoceros. Intelligente, per certi aspetti rivoluzionario, abbastanza potente, necessario. La modellazione NURBS ha fornito nuove possibilità. Rhinoceros con Grasshopper, trattato più avanti, diventa veramente interessante se poi vogliamo considerare tutti i motori di rendering o i numerosi *plug-in* il prodotto diventa irresistibile. Anche dal punto di vista economico. Non è immune da *bug*.

Poi c'e stato Revit. Finalmente un CAD parametrico per l'architettura. Come ArchiCAD meglio, molto meglio di ArchiCAD. L'Autodesk lo ha fagocitato subito. Necessita di macchine dotate per poter gestire, elegantemente, situazioni complesse.

Il più elegante è senza dubbio MoI 3D. Nato da una costola di Rhinoceros è ancora giovane ma, personalmente, lo trovo fra i più intelligenti per disegnare seguendo un modello di disegno tipo CAD.

Tralasciando importanti software quali Allplan, FormZ, Catia e altri chiudiamo con Google SketchUp dicendo che è intelligente, gratuito e apre le porte del paradiso degli oggetti.

Modellazione tridimensionale

Se ci guardiamo attorno e osserviamo le forme degli oggetti, vediamo che ne esiste una varietà enorme. Il lavoro di un modellatore inizia analizzando le forme base che definiscono un oggetto. Un pallone da calcio è una sfera, una scatola di latta è un cilindro e un dado è un cubo. Questi oggetti sono costruiti da semplici forme di base, spesso definite *primitive*.

Molti oggetti sono costituiti da una varietà di forme di base: un imbuto è l'intersezione tra un cono e un cilindro stretto, mentre una libreria o un edificio è costituito da blocchi di vario spessore, larghezza e altezza. Infine, ci sono forme che sono molto più difficili da modellare: gli oggetti con un sacco di curve come una vettura sportiva o il famoso Guggenheim di Bilbao, le forme organiche di quasi tutti gli esseri viventi, gli alberi, le rose, i

gatti, le persone e in generale la maggior parte degli oggetti dalla natura intorno a noi, le montagne e le formazioni geologiche, l'acqua, le nuvole. Per queste situazioni, gli sviluppatori di software hanno dovuto creare sistemi di modellazione complessi.

Inizialmente, e in molti dei programmi di modellazione di oggi, i computer rappresentano strutture utilizzando un sistema che si basa su poligoni.

Oggi esistono altri sistemi di modellazione in cui l'utente non lavora con i poligoni ma con superfici curve definite matematicamente.

NURBS, spline, patch bezier, ecc, sono i diversi tipi di strumenti forniti da un software di modellazione per la creazione di superfici curve complesse.

Il vantaggio di spline, o curve matematiche, è che esse definiscono sempre la superficie perfettamente, non importa quanto vicino sia l'osservatore. Tuttavia sono molto più difficili da gestire e, in alcune situazioni, risolvere i problemi può essere alquanto complesso.

Con modellazione 3D si indica un processo atto a definire una qualsiasi forma tridimensionale in uno spazio virtuale generata su computer.

Modellazione 3D

da Wikipedia, novembre 2009

Con Modellazione 3D si indica un processo atto a definire una qualsiasi forma tridimensionale in uno spazio virtuale generata su Computer, questi oggetti, chiamati modelli 3D vengono realizzati utilizzando particolari programmi software, chiamati modellatori 3D, o più in generale Software 3D. Questo termine viene utilizzato in ambito informatico, e si distingue da altri tipi di modellazione tridimensionale, come ad esempio la scultura tradizionale.

Cenni storici

La storia della Computer grafica 3D è naturalmente molto recente, lo stesso termine di grafica computerizzata nasce solo nel 1960.

Una delle prime rappresentazioni tridimensionali su calcolatore è stata quella del famoso "primo uomo" o "Boeing Man" realizzata da William Fetter; un insieme di linee che descrivevano la sagoma virtuale di un pilota di aereo.

A partire dal 1959, la General Motors, in collaborazione con la IBM, sviluppa il sistema "DAC", uno dei primi sistemi CAD della storia; attraverso una penna ottica e uno schermo sensibile, gli operatori disegnavano delle curve matematiche in uno spazio virtuale, con le quali delimitavano i profili, le sezioni e le superfici delle automobili.

Della prima metà degli anni 60' è anche il il sistema chiamato "Adage", considerata da molti la prima workstation CAD indipendente.

Da quanto riportato si evince che la nascita della *modellazione 3D* avvenne in ambito industriale, primariamente come supporto alla progettazione. Da allora i campi di utilizzo della *modellazione 3D* e della grafica tridimensionale si sono enormenente ampliati, uscendo in buona parte dall' ambito tecnico.

La Modellazione nella pratica operativa dalla Grafica 3D

La *modellazione 3D* può anche essere fine a sè stessa, e in questo caso il modello generato non richiede ulteriori elaborazioni, ma generalmente la modellazione rappresenta il primo step di una serie di operazioni successive che determineranno l'elaborato finale. Questo primo step, nella specifica area della Computer grafica 3D, non può mai mancare, e ne rappresenta il presupposto di partenza.

Si prenda ad esempio un caso particolare abbastanza complesso: *la realizzazione di un'immagine statica fotorealistica di un personaggio 3D*. Questa comporta i seguenti passaggi essenziali:

- Modellazione 3D primaria
- Modellazione 3D secondaria
- Surfacing (definizione dei materiali di superficie)
- Mappatura (definizione delle coordinate di proiezione)
- Applicazione delle Texture
- Inserimento dello scheletro
- Skinning del modello
- Definizione della postura del modello
- Allestimento scenico
- Illuminazione della scena
- Rendering della scena
- Salvataggio dell'immagine in un file grafico
- Output finale (ad es. stampa su carta)

O un caso relativamente più semplice: *Corpo in alluminio di un mulinello, realizzato con macchina utensile*.

- Modellazione 3D della parte in un modellatore CAD
- Assemblaggio e verifica della parte nel modello di Assieme
- Esportazione del modello 3D in un formato macchina compatibile
- Lavorazione con macchina utensile CNC dell'oggetto
- Pulitura, rifinitura
- Anodizzazione e lucidatura dell'oggetto
- Assemblaggio nel prodotto finale

Come si nota la realizzazione del modello 3D è posta sempre all'inizio della catena operativa, ed è la base delle successive operazioni.

Campi di impiego della modellazione 3D

I sistemi di modellazione vengono impiegati in tutti i campi della Computer grafica 3D, tanto che in taluni casi modellazione 3D e grafica 3D sono sinonimi.

Applicazioni a carattere scientifico o tecnico

- Scienze matematiche, fisiche e naturali (biologia, fisica, matematica, astronomia etc.)
- Studio del territorio (Geologia, Sismologia, meteorologia etc.)
- Scienze storiche (archeologia, paleontologia, paleoantropologia etc.)
- Scienze applicate
- Medicina (Forense, ricostruttiva, indagini diagnostiche etc.)
- Ingegneria civile
- Ingegneria industriale
- Architettura
- Industrial Design

Applicazioni artistiche

- Industria cinematografica e televisiva

- Videogame e applicazioni videoludiche
- Grafica pubblicitaria
- Pubblicazioni editoriali
- Web design
- Applicazioni multimediali
- Produzione artistica

Tipologie di modellazione

Da un punto di vista tipologico, tutta la modellazione 3D, rientra in due grandi famiglie, ognuna riguardante un ben determinato genere di modelli:

La Modellazione organica

È la tipica modellazione utilizzata per realizzare gli esseri umani o le creature, animali o umanoidi. Viene usata per tutti i soggetti "naturali", come rocce, piante, alberi e per il territorio in generale, in questi casi i modelli sono tanto più riusciti quanto più sono ricchi di particolari. Anche molti oggetti di industrial design, che abbiano forme morbide e arrotondate, possono servirsi di una modellazione organica.

La Modellazione geometrica

È il tipo di modellazione meno recente, viene utilizzata per realizzare oggetti tecnici o meccanici, o comunque per qualsiasi cosa che abbia una natura artificiale, e che non rientri nella categoria precedente. Generalmente la complessità dei modelli realizzati con questo genere di modellazione è molto inferiore, se si guarda all'aspetto esteriore delle singole forme, ma non se si considerano aspetti legati alla precisione e alla corrispondenza delle parti.

Naturalmente uno stesso oggetto può contenere sia modellazione organica che geometrica, oppure può essere formato da un insieme di parti contenenti sia modelli organici che geometrici.

Tecniche di modellazione 3D

Si possono dividere in tre categorie principali:

1. Modellazione Procedurale (automatica e semi-automatica)
2. Modellazione Manuale
3. Da dati provenienti da modelli reali (scansione tridimensionale)

Che a loro volta possono venire suddivise in tre distinti generi di modellazione:

- Modellazione Solida - dove l'oggetto risultante è considerato come formato da un volume pieno.
- Modellazione Volumetrica - determina delle entità generanti una superficie implicita.
- Modellazione di superfici - l'oggetto in questo caso è determinato dalle sue superfici esterne.

In alcuni modellatori un oggetto è considerato fomato da superfici finché queste sono aperte, mentre viene riconosciuto come solido una volta che tutte le superfici siano saldate fra di loro e formino un corpo chiuso.

Il seguente elenco esamina le diverse tecniche di Modellazione Manuale. Alcune delle tecniche descritte (come ad es. le superfici patch), essendo abbastanza datate, risultano essere superate e obsolete rispetto a tecniche più recenti e avanzate. Malgrado questo taluni Modellatori 3D, mantengono al loro interno alcuni di questi strumenti come accessori o utilità.

Costruzioni di base (solidi e superfici)

- Primitive - Generalmente ogni pacchetto 3D che non si occupi solo di rendering, contiene al suo interno un set più o meno nutrito di primitive, ossia di oggetti predefiniti (solidi o superfici), direttamente impiegabili; di solito le primitive standard, cioè sempre presenti, sono: il piano, il Cubo/Parallelepipedo, la Sfera, il Cilindro, il Cono/Piramide,il Toro, e spesso la Teiera.
- costruzione per estrusione - è un semplice metodo per realizzare delle forme estruse partendo da un disegno 2d di base o da un poligono piano, e assegnandogli una certa altezza e una direzione di estrusione.
- costruzione per rivoluzione - più complessa della precedente costruzione, una rivoluzione si può considerare come una estrusione attorno a un asse, si parte sempre da un profilo o da un poligono di base, e invece della profondità viene assegnato un angolo di rivoluzione.

Superfici patch

La modellazione per patch è uno dei sistemi meno recenti utilizzati in grafica 3D, e nel corso del tempo si è molto evoluta sviluppando una serie di nuove tecniche. Nella sua forma più semplice determina delle superfici parametriche generate da quattro o più curve adiacenti che formino un perimetro chiuso.

- Superficie di Coons - determina una patch interpolata tra solo quattro curve di bordo, aventi i vertici in comune, il primo algoritmo di questa classe di superfici fu sviluppata da Steven A. Coons nel 1967.
- Patch di Bézier - è una superficie parametrica controllabile localmente mediante una griglia di punti di controllo, congiungendo assieme più patch di Bézier si ottengono superfici più complesse chiamate superfici spline, in questo caso i punti di controllo si trovano all' intersezione tra le varie patch.

Modellazione Spline (superfici)

La modellazione spline utilizza la tecnica del patching, precedentemente descritta, e le curve spline. Fondamentalmente un modello realizzato mediante questo sistema è formato da una gabbia di curve spline, intersecanti e collegate tra loro. Gli spazi compresi tra tre o quattro curve spline unite nei loro punti di controllo, vengono poi riempiti da patch. Tale sistema si presta soprattutto a realizzare modelli organici.

Costruzioni Avanzate (solidi e superfici)

Le costruzioni avanzate utilizzano lo stesso concetto alla base dell'estrusione e della rivoluzione semplice, ma vi aggiungono dei controlli molto più sofisticati.

- Estrusione Sweep - è essenzialmente una estrusione lungo un percorso. Viene sempre usato un profilo o un poligono di base, come nell'estrusione semplice, a cui viene associato un percorso che può essere una curva o delle linee spezzate.
- Costruzione per Loft - in questo caso la forma dell'oggetto è data da una serie di profili disposti in una certa sequenza, i profili possono essere considerati come le sezioni dell'oggetto.
- Rivoluzione su binario - è un sistema ibrido tra una rivoluzione e una sweep, si parte sempre da un profilo e da un asse di rivoluzione, ma vi si aggiunge anche un percorso di base (chiamato anche binario), che il profilo dovrà seguire durante la rivoluzione.

Questo genere di costruzioni, nel corso degli anni sono state dotate in realtà di molti tipi di controllo, si sono aggiunte linee guida supplementari, controlli del tipo di torsione, definizioni di tangenze etc.

Questi sistemi di modellazione, per il loro alto grado di precisione, vengono impiegati per lo più per definire oggetti tecnici o di design industriale.

Modellazione poligonale

Si tratta di tecniche basilari nell'ambito della grafica 3D.

La modellazione poligonale opera su superfici organizzate in maglie più o meno dettagliate di facce poligonali. Queste superfici possono solo approssimare l' oggetto finale se siamo in presenza di un basso livello di poligoni (in questo caso caso l' oggetto viene detto Low Poly). In altri casi un modello poligonale - a modellazione ultimata - può essere formato anche da un numero molto elevato di facce.

I seguenti sistemi procedono dai più elementari ai più evoluti:

- Per spostamento di elementi - un modello poligonale è formato da 3 elementi essenziali: facce, lati e vertici; lo spostamento arbitrario di un singolo elemento o di gruppi di essi, determina una modifica della mesh di partenza. La selezione di un componente della mesh e il suo spostamento (trascinamento, rotazione, ridimensionamento etc.), nello spazio è la tecnica più elementare di modellazione poligonale.
- Da primitive di base - Uno dei sistemi più semplici e diretti per iniziare a modellare un oggetto poligonale, è quello di partire da una primitiva poligonale di base, e iniziare a modicarla spostando, ruotando, scalando i suoi componenti, fino a ottenere la forma voluta. Questa tecnica è molto semplice, ma consente in genere di ottenere modelli poco complessi,

vincolati cioè alla complessità (anche in termini di densità poligonale della mesh) della primitiva di partenza.

- Metodo della mesh piana - oltre a modificare i poligoni di mesh esistenti (ad es. delle primitive), esiste la possibilità di creare singolarmente ogni poligono dell' oggetto e di costruire i poligoni nella posizione più comoda per realizzare il modello finale. Uno dei sistemi di disegno diretto dei poligoni viene detto Metodo della mesh piana.Si tratta in sostanza di creare una griglia di poligoni posizionati in piano e aventi la struttura il profilo e la conformazione generale dell' oggetto finale. Posizionati i poligoni sul piano, si passa a determinarne la tridimensionalità: o spostando i punti della griglia lungo la profondità del modello, o attraverso dei sistemi di estrusione.
- Metodo a tela di ragno - Si tratta di una variante della precedente tecnica. In questo caso non si costuiscono e posizionano tutti i poligoni di base del modello, ma si parte da una sua zona (centrale), e si iniziano a creare e modellare i singoli poligoni con un sistema appunto a "tela di ragno", cioè dall' interno e procedendo manmano verso le zone esterne del modello. È un sistema complesso e dispendioso in termini di tempo, utilizzato soprattutto per il suo alto grado di precisione.
- Per Rifinitura Progressiva - è il sistema più evoluto, può considerarsi uno dei paradigmi della Modellazione 3D. Adottando un qualsiasi metodo analizzato precedentemente si inizia a definire la forma in una maniera molto schematica, perlopiù approssimandone la morfologia e facendo attenzione a tenere estremamente basso il numero iniziale di poligoni. Dovendo gestire pochi poligoni è possibile modificare molto agevolmente le proporzioni e il volume generale della forma. Solo quando si è soddisfatti dell' aspetto grezzo del modello si può iniziare - adottando gli specifici strumenti di ogni pacchetto software - a definire maggiormente la forma. È importante che a ogni passaggio di rifinitura si passi a definire prima i volumi maggiori del modello, per andare poi a definire le zone sempre più piccole, la definizione e il numero di dettagli apportabili è a discrezione del grafico 3D. Il principio fondamentale da tenere a mente è che: tanto minore è il numero di poligoni presenti nel modello, tanto maggiore è la possibilità di modificarne la morfologia generale - tanto maggiore è il numero di poligoni tanto meno si potrà modificare la forma già impostata in precedenza. In pratica ogni passaggio è irreversibile, tanto più si definiscono i particolari dell' oggetto, tanto meno si potrà modificare (o correggere) il suo aspetto generale. A questo problema si può porre rimedio salvando il modello in maniera progressiva, in modo da avere a disposizione tutti i passaggi intermedi di modellazione, in caso di errore si può ripartire dal modello precedente a minore dettaglio, se il software utilizzato fa uso dei layer, è possibile conservare le varie versioni in layer separati.
- Per Displacement map - vedi la sezione relativa
- Per Scultura 3D - vedi la sezione relativa

Modellazione solida

La modellazione solida, o CSG, è un tipo di modellazione geometrica, utilizzata soprattutto in anbito tecnico e CAD, Storicamente si inizia a parlare di modellazione solida solo alla fine degli anni '60, mentre il primo modellatore solido commerciale (chiamato Romulus)risale al 1982, seguito poi da Parasolid, della Unigraphics, nel 1988.

La modellazione solida utilizza i seguenti strumenti di base:

- Primitive di base - sono le medesime primitive analizzate in precedenza.
- Costruzioni per estrusione e rivoluzione, semplici e avanzate - anche in questo caso i modelli solidi utilizzano le stesse tecniche descritte in precedenza.
- operazioni booleane - derivante dall'Algebra di Boole, questa tecnica è invece esclusiva della modellazione solida. Consente di ottenere delle forme complesse partendo dalle primitive solide, componendole tra loro attraverso tre operazioni: Unione, Sottrazione e Intersezione.
- Smussi e raccordi - sono funzioni automatiche che intervengono lungo i bordi dei solidi, consentendo di raccordarli mediante un certo raggio, o smussarli di un determinato angolo.

Metaball

Le metaball (o "blob"), sono un particolare tipo di primitive utilizzate per realizzare modelli organici, di design o simulazioni di liquidi. Sono delle entità di tipo volumetrico come i voxel, hanno un nucleo centrale che viene visualizzato come superficie implicita e un campo di forza o di "influenza" esterno. Quando due metaball vengono accostate reagiscono fra di loro attraverso il campo di forza esterno che le attrae (se è positivo) e ne determina la fusione, o le respinge (se è negativo), e provoca una sottrazione di volume.

Realizzato il modello sotto forma di superficie implicita, è possibile in genere convertirlo in una mesh poligonale vera e propria, invocando parametri come la densità finale della mesh. Sono state sviluppate diverse forme di Metaball:

- Metaball sferiche - sono le metaball nella loro forma nativa, essendo vincolate a tale geometria, presentano lo svantaggio, nel caso si debbano realizzare modelli organici complessi, dell' alto numero di entità da dover posizionare, soprattutto in presenza di forme allungate e flessuose.
- Metaball con altra geometria - pur sfuttando i medesimi principi delle metaball sferiche, queste entità possono assumere la forma di altri tipi di primitiva, e hanno il vantaggio di consentire di approssimare la forma finale facendo uso di molte meno entità.
- Metamuscoli - queste entità geometriche rappresentano una delle maggiori evoluzioni delle metaball. Furono introdotte per la prima volta nel 1997 dalla REM Infogràfica sotto forma di plug-in per 3DS Max, denominata MetaReyes in revisione 3.1. I metamuscoli sono delle metaball deformabili su percorsi spline; modificando i punti di controllo delle spline e i parametri delle metaball si ottengono delle forme allungate approssimanti un muscolo, l'interazione di vari metamuscoli, secondo le modalità tipiche delle metaball, genera la forma finale. Il pregio e il limite di tali primitive è la loro specializzazione nel definire forme quasi esclusivamente anatomiche.

Superfici NURBS

La tecnologia NURBS fu introdotta dalla Boeing nel 1975, acronimo di non-uniform rational B spline (B-Spline razionali non uniformi), viene utilizzata in grafica 3D per realizzare una vasta tipologia di modelli; è particolarmente adatta a rappresentare superfici organiche, ad esempio di creature e personaggi, e oggetti di design che richiedano superfici complesse e precise come le automobili. Le superfici NURBS sono superfici matematiche perfettamente smussate, non caratterizzate dalla tipica sfaccettatura delle superfici poligonali, sono facilmente modificabili e controllabili attraverso pochi punti di controllo, chiamati CV (Control Vertices).

Una superficie NURBS può essere generata o dalle stesse curve NURBS, attraverso operazioni di estrusione, rivoluzione, lofting, patching e altre, oppure da primitive di tipo NURBS come sfere, cilindri, tori etc. Le successive modifiche di un modello NURBS, dipendono molto dagli strumenti messi a disposizione dal Software 3D, ma in genere si passa a editare i punti della superficie, o i vertici CV, dove è necessario si possono aggiungere o rimuovere curve nelle direzioni U e V, come si possono aggiungere e rimuovere i vertici di controllo nelle curve.

Modellazione solida parametrica

È un genere di modellazione usata in ambito CAD.

La modellazione solida parametrica, nell'ambito della progettazione CAD ha colmato alcune mancanze della modellazione solida semplice. Essa consente di generare i solidi mediante l'immissione di parametri numerici, ad es. l'altezza, la lunghezza, la profondità, i raggi e le misure angolari, e di poter intervenire su questi parametri anche dopo aver realizzato il modello, per modificarne e aggiornare la geometria senza doverlo ricostruire. In aggiunta alla modellazione tramite

parametri venne introdotto anche il concetto di "feature" e di albero di costruzione: in pratica tutte le lavorazioni applicate sul modello solido vengono registrate (come feature), in una specie di albero cronologico che funziona secondo uno schema di dipendenza padre-figlio; è possibile in ogni momento della modellazione tornare indietro nell'albero di costruzione, selezionare una feature, editare e modificare i suoi parametri, e aggiornare tutto il modello con i nuovi parametri. La modellazione solida parametrica viene oggi detta "ibrida", in quanto molti modellatori CAD hanno aggiunto delle funzioni avanzate di modellazione ibrida solida e di superficie, per potere realizzare modelli più complessi.

Di seguito viene schematizzata una sequenza tipo di modellazione solida parametrica; si tratta di uno schema molto semplice, adottabile soprattutto per componenti meccaniche o che comunque non richiedano interventi complessi:

1. Selezione di un Piano di partenza (un piano di default o creato appositamente)
2. Disegno 2D sul piano (schizzo iniziale)
3. Lavorazione o Feature di base (ad es. Estrusione dello schizzo)
4. Lavorazioni secondarie (Tagli/estrusioni in modo analogo alla lavorazione di base)
5. Lavorazioni di rifinitura (Smussi, Raccordi, filettature etc.)

Bisogna ricordare che ogni parametro numerico riguardante gli schizzi di partenza e le feature di lavorazione è editabile e modificale in qualsiasi momento della modellazione, così come sono modificabili tutte le opzioni delle feature. Il sistema di lavorazione non procede in un unico senso (come nella Modellazione poligonale a Rifinitura Progressiva) ma è reversibile e modificabile all'infinito.

Superfici di Suddivisione (Subdivision Surface)

Le Superfici di suddivisione della B-spline di Catmull Clark furono Sviluppate da E. Catmull e J. Clark nel 1978. Furono utilizzate per la prima volta nell' ambito della Computer grafica 3D dalla Pixar nel film di animazione Geri's Game, del 1989.

Sono uno strumento di modellazione molto versatile, adatto soprattutto a realizzare modelli organici in maniera semplice e dettagliata. Coniugano assieme le migliori caratteristiche della modellazione poligonale e della modellazione NURBS; come le superfici NURBS sono perfettamente smussate e prive di sfaccettature, ma possono avere come base forme dalla topologia irregolare, tipiche dei modelli poligonali.

Uno dei migliori sistemi per iniziare la modellazione con le superfici di suddivisione è proprio quello di convertire un modello poligonale, l'unico requisito importante è che la mesh da convertire sia il più semplice possibile, formata cioè da un basso numero di poligoni, questo perché non servono molti poligoni per realizzare delle superfici di suddivisione perfette. Il passaggio da una superficie poligonale a una superficie di suddivisione è automatico, e questo vale anche per il processo inverso.

Fondamentalmente le superfici di suddivisione utilizzano le stesse tecniche di modellazione impiegate per le mesh poligonali, con qualche distinguo e con molte più possibilità, come ad esempio poter assegnere un maggiore o minore "peso" a ciascun punto della superficie.

Superfici implicite (Voxel)

Generalmente, più che rappresentare una tecnica di modellazione, i Voxel vengono usati come dei sistemi per visualizzare geometrie o fenomeni particolari. I voxel generano un volume attorno a un punto geometrico (cioè definito e posizionato nello spazio), tale punto viene visualizzato e renderizzato tramite la superficie implicita del voxel.

La visualizzazione volumetrica tramite voxel viene impiegata ampiamente in ambito medico, utilizzando i dati tridimensionali provenienti dalle Tomografie computerizzate (TC), e dalle risonanze magnetiche (RM), I modelli generati in tale modo rientrano nella categoria della modellazione da "scansione tridimensionale", descritta più avanti.

I voxel vengono anche utilizzati nell'ambito della animazione tridimensionale per alcuni tipi di simulazione complessa, come quella degli effetti gassossi, atmosferici e per le esplosioni, similmente possono venire impiegati per realizzare i materiali liquidi e fluidi, come acqua, lava, etc. mediante motori di generazione particellare, in questo caso il loro l'utilizzo rientra nell'ambito della "modellazione procedurale".

Per quanto concerne la modellazione manuale vera e propria, le superfici implicite possono utilizzare la struttura di geometrie esistenti. Sfruttando la caratteristica dei voxel di creare entità volumetriche attorno a dei punti geometrici, si possono costruire forme particolari, sia materiche che "immateriali" utilizzando uno dei sistemi di modellazione qui esaminati (ad es. la modellazione poligonale), e mammano verificare la forma volumetrica che si stà generando.

I modelli ottenibili possono anche essere simili a quelli realizzati tramite le metaball (che pure sono entità volumetriche), ma generalmente si sfuttano le capacità tipiche dei voxel di generare superfici molto complesse, difficilmente ottenibili in altro modo. Da quanto detto si comprende che l'ambito di utilizzo delle superfici implicite comprende quasi esclusivamente forme e strutture di tipo organico, naturale o immaginario, ma non di tipo geometrico.

Mappe di Displacement

Il displacement mapping è una tecnica di modellazione che non utilizza gli strumenti standard di modifica, ma si basa sull' elaborazione di immagini in scala di grigio.

Utilizza lo stesso principio dell'"Images mapping" (mappatura di immagini), ad es. il "Bump mapping" (rugosità), con la differenza che il displacement interviene sulla geometria del modello, modificandola. Agendo nella direzione "normale" della superficie, la mappa di displacement provoca uno spostamento in senso positivo dei punti del modello corrispondenti alle zone chiare dell'immagine, e in senso negativo di quelli corrispondenti alle zone scure. Può essere considerato come uno strumento di deformazione della mesh attraverso immagini, viene utilizzato sia su modelli organici che geometrici.

Si possono distinguere due generi di displacement:

- Displacement Geometrico - agendo direttamente sui punti della mesh poligonale, questo tipo di displacement necessita di un alto livello di tasselizzazione della mesh per produrre risultati buoni, ha quindi lo svantaggio di produrre modelli molto pesanti e difficilmente gestibili.
- Displacement per Micropoligoni (Microdisplacement) - il displacement per Micropoligoni genera in automatico un grande numero di piccole facce triangolari (anche molti milioni), ed è in grado di realizzare modelli molto dettagliati. La particolarità e il grande vantaggio di questo sistema risiede nel fatto che la tassellizzazione del modello avviene solo in fase di rendering o pre-visualizzazione (è cioè temporanea), mentre non va ainteressre la geometria di base che può mantenersi così molto semplice. Per estremo, utilizzando un solo poligono piano e un'immagine mappata, si può ottenere in fase di rendering un modello perfettamente definito (ad es. un terreno frastagliato o un bassorilievo scultoreo).

Scultura 3D

Per indicare questa tecnica si usa anche il termine "displacement painting", in quanto deriva dalla comunione di tecniche di displacement map e di tecniche di painting 3D.

È un sistema molto affine a tecniche di scultura tradizionale, opera utilizzando dei pennelli virtuali, variabili in dimensione e funzioni, che, passati sulla superficie del modello vanno a modificarne la geometria in tempo reale, provocando protusioni, avvallamenti, scalfitture e incisioni, come se si stesse lavorando su un pezzo di argilla.

I precursori di questa tecnica furono i programmi di painting diretto su mesh, che però non lavoravano sul canale del displacement. Il primo esempio di questo tipo di scultura fu il modulo "artisan", impiegato da

Maya, ma il capostipite vero e proprio dei modellatori basati su questa tecnologia è senza dubbio ZBrush di pixologic, seguito da una serie di altri pacchetti commerciali.

La scultura diretta della mesh viene utilizzata soprattutto per la rifinitura e il dettaglio in alta definizione di modelli semplici realizzati con altri metodi, ma può essere usata anche per definire da zero un modello partendo da primitive semplici come parallelepipedi o sfere. È usata in larga misura nella modellazione organica, in particolare nella modellazione e definizioni di personaggi.

Data l'estrema complessità dei modelli ottenuti con questa tecnica (che possono essere formati da molti milioni di poligoni), si rende quasi sempre necessario trasferire i dati tridimensionali della mesh in mappe di displacement o in normal map, utilizzabili in modelli molto più leggeri.

Tecniche di Rotoscoping

Non si tratta di una tecnica di modellazione 3D in senso stretto, Il rotoscoping (o ricalco), è piuttosto una tecnica di supporto alla modellazione. In molti casi può essere di aiuto iniziare la modellazione di un qualsiasi oggetto utilizzando come riferimento delle immagini di sfondo. Queste vengono posizionate e scalate nelle finestre standard di lavorazione del programma 3D, oppure, se si preferisce, possono essere mappate su dei piani paralleli alle viste di lavoro. Per taluni soggetti è sufficiente utilizzare una sola immagine di riferimento, per altri, più complessi, possono servire due o tre immagini, posizionate nelle viste: Frontale, Laterale, Superiore (o inferiore).

La tecnica del rotoscoping è utilizzabile per ogni tipologia di oggetti, da quelli realizzati in modellazione organica, agli oggetti tecnici realizzati in ambiente CAD.

Un discorso a parte meritano le successive due voci, in quanto adottano tecnologie e procedure particolari,che le pongono al di fuori della modellazione manuale semplice:

La modellazione procedurale

La modellazione procedurale è una modellazione assistita da strumenti software che generano in maniera automatica o semi-automatica la geometria voluta. La qualità dei modelli prodotti è in questo caso delegata alle maggiori o minori capacità del software impiegato.

vi sono varie categorie di softwares procedurali per quanto concerne la creazione di forme tridimensionali, si possono distinguere i seguenti simulatori e generatori:

- Simulatori fluidodinamici
- Simulatori di tessuti e soft-body
- Generatori di vegetazione
- Generatori di capelli e peluria
- Generatori di modelli 3D (volti, creature, oggetti geometrici etc.)
- Generatori frattali (terreni, forme astratte etc.)

Questi programmi generano forme tridimensionali sotto forma di mesh, volumi o superfici, impiegabili nei normali softwares 3D per le applicazioni necessarie.

Generalmente sono due i metodi di modellazione usati: o esclusivamente attraverso l'impostazione dei parametri messi a disposizione dal software e l'immissione di dati numerici, dopodiché la generazione procede in maniera automatica - o attraverso dei sistemi di modellazione guidata, che consentono un controllo maggiore di quanto si stà realizzando: in questo caso il software segue delle geometrie di guida (curve, mesh etc.), o viene limitato da vincoli esterni. I casi da anallizzare sarebbero molti e specifici per ogni tipologia di modellazione procedurale.

La scansione tridimensionale

Realizzare modelli 3D acquisendoli da oggetti reali rientra in un tipo di modellazione utilizzato in svariati settori; dall' architettura all' industria cinematografica, dalla conservazione dei beni artistici alla medicina, etc.

Esiste una vasta gamma di strumenti e procedure per ottenere della repliche virtuali di oggetti fisici:

- Per Fotogrammetria - è un sistema abbastanza semplice e economico, che permette di acquisire forme a basso dettaglio. Si utilizzano delle fotografie del soggetto prese da varie angolature (a volte con dei marcatori applicati), il software si occupa poi di ricostruire la versione tridimensionale. La precisione non è assoluta, e i modelli approssimano in maniera semplice la forma di partenza.
- Per Sonda 3D a contatto (Tastatore) - si basa sull'uso di un braccio meccanico snodato che và a "tastare" il modello negli incroci di una griglia segnata sulla sua superficie, mentre il software riproduce i punti nello spazio tridimensionale, è un sistema adatto a replicare oggetti non troppo grandi e realizzati in materiali rigidi (ad es. piccole e medie sculture).
- Per scansione Laser - è un sistema versatile che comprende molti tipi di strumenti, a seconda delle dimensione degli oggetti da scansionare, della risoluzione etc. , si và da strumenti manuali, piccoli e portatili, a apparecchiature da studio, fisse o mobili, fino a attrezzature da utilizzarsi in spazi aperti per rilevare territori o architetture.

I sistemi laser, per ogni inquadratura dell' oggetto, producono delle superfici formate da "nuvole di punti", varie inuadrature forniranno una serie di nuvole di punti che andranno a comporre il modello 3D, il dettaglio ottenibile con questi sistemi può essere anche molto elevato. Le tipologie di oggetti scansionabili è molto vasta; essendo una tecnica non a contatto e non invasiva, si possono rilevare oggetti morbidi e flessibili come ad es. i corpi umani.

- Per Proiezione di Pattern Luminosi - produce una serie di nuvole di punti che verranno trattate in maniera similare alla scansione laser. In questa tecnica sul modello viene proiettata una luce bianca, sotto forma generalmente di strisce, che viene poi catturata da dei sensori di luce (ad es. delle macchine fotografiche digitali). Rispetto alla scansione laser è un sistema molto più veloce, ma ha lo svantaggio di non poter scansionare oggetti molto grandi.
- Scansioni TAC o RMN

Questi sistemi di indagine diagnostica consentono, in una maniera non invasiva, di poter rilevare le strutture anatomiche interne di un corpo umano o animale, non ottenibili con altri sistemi.

Principi di corretta Modellazione

Per comprendere quale debba essere il giusto utilizzo dei vari sistemi di modellazione bisognerebbe introdurre il concetto di Modello 3D corretto e Modello 3D scorretto. Si deve cioè spostare l'attenzione dall'aspetto tecnico della modellazione a un'analisi attenta del modello da realizzare.

Il processo di modellazione deriva primariamente dalla tipologia del modello da realizzare. La tipologia del modello comporterà una prima scelta tra tecniche di modellazione organica e tecniche di modellazione geometrica (non avrebbe senso approcciare la modellazione di un componente meccanico con delle tecniche organiche; come sarebbe un nonsenso voler realizzare una mano umana con un sistema CAD), questo perché ogni tipologia di oggetto è associabile in maniera naturale a determinate tecniche e non a altre.

Ciò che condizionerà la scelta specifica del sistema di modellazione, saranno invece le caratteristiche richieste al modello dalla sua destinazione d'uso. Un modello 3D molto bello da vedersi non è necessariamente eseguito correttamente: perché potrebbe essere inadatto all' utilizzo cui è destinato (ad es. il modello 3D di un' automobile da usarsi in un videogame, sarà necessariamente diverso dal modello CAD della stessa automobile da utilizzarsi per la produzione di serie). Si adotterà una tecnica di modellazione corretta se sarà adeguata primariamente alla tipologia del modello e secondariamente al suo utilizzo finale.

Grasshopper

Se guardiamo l'architettura come un oggetto rappresentato nello spazio, abbiamo sempre a che fare con la geometria e un po' di matematica per capire il design di questo oggetto.

Nella storia dell'architettura, diversi stili architettonici hanno presentato diversi tipi di geometria e la loro logica di articolazione e ogni periodo ha trovato un modo per affrontare i suoi problemi geometrici e domande.

Dato che i computer hanno cominciato ad aiutare gli architetti, simulare lo spazio e articolazioni geometriche, è diventato parte integrante del processo di progettazione.

La geometria coputazionale è divenuta un argomento interessante da studiare e la combinazione di algoritmi di programmazione con la geometria ha dato luogo a geometrie algoritmiche note come *generative algorithm*. Sebbene il software 3D abbia aiutato a simulare quasi tutto lo spazio visualizzato, è l'idea di algoritmo generativo che porta le possibilità attuali del design come *design parametrico* nel campo dell'architettura.

Gli architetti hanno iniziato ad utilizzare forme libere, a progettare e verificare gli spazi al di là del limitazioni delle geometrie convenzionali dello spazio "euclideo". È stata la combinazione di architettura e digitale che ha portato *blob* sul tavolo e poi spingere ulteriormente. L'architettura contemporanea dopo l'età di *blob*, sembra essere ancora più complessa.

"La logica alla base della progettazione parametrica può essere utilizzata qui come un metodo alternativo di progetto, quello in cui il rigore geometrico della modellazione parametrica può essere utilizzato prima di integrare limiti di produzione, le logiche di montaggio e caratteristiche dei materiali nella definizione di semplici componenti, e poi al proliferare dei componenti in sistemi più grandi. Questo approccio impiega l'esplorazione di variabili parametriche per capire il comportamento di un tale sistema e quindi utilizza queste informazioni per elaborare strategie di risposta del sistema per l'ambiente le condizioni e le forze esterne" (Hensel, Menges, 2008).

Per lavorare con gli oggetti complessi, di solito un processo di progettazione parte da un livello molto semplice, prima e poi gli altri livelli si aggiungono ad esso; forme complesse sono composte da diverse gerarchie, ciascuna associata con le sue logiche e dettagli. Questi livelli sono interconnessi e dei loro membri incidere l'altro e in questo senso questo metodo chiamato 'associativo'.

In generale, la modellazione associativa si riferisce a un metodo in cui gli elementi di design sono costruiti gradualmente in più gerarchie e ad ogni livello, alcuni parametri di questi elementi sono estratti dal generatore per altri elementi del livello successivo e questo va avanti, passo dopo passo fino alla definizione di tutta la geometria. Quindi, fondamentalmente il punto finale di una curva potrebbe essere il punto centrale di un altro cerchio e qualsiasi cambiamento nella curva avrebbe cambiato il cerchio di conseguenza.

Il punto è che tutte queste geometrie sono facilmente regolabili, dopo il processo. Il designer ha sempre accesso agli elementi del prodotto di design dal punto di avvio per i dettagli. In realtà, poiché la progettazione di un prodotto è il risultato di un algoritmo, gli ingressi dell'algoritmo potrebbero essere cambiati e il risultato dovrebbe essere aggiornato di conseguenza.

"La progettazione parametrica consente il riconoscimento dei modelli di comportamento, la geometrica capacità performativa e le tendenze del sistema. In un feedback continuo con l'ambiente esterno, queste tendenze comportamentali possono informare lo sviluppo ontogenetico di uno specifico sistema attraverso la parametrica differenziazione dei suoi sub-luoghi" (Hensel, Menges, 2008).

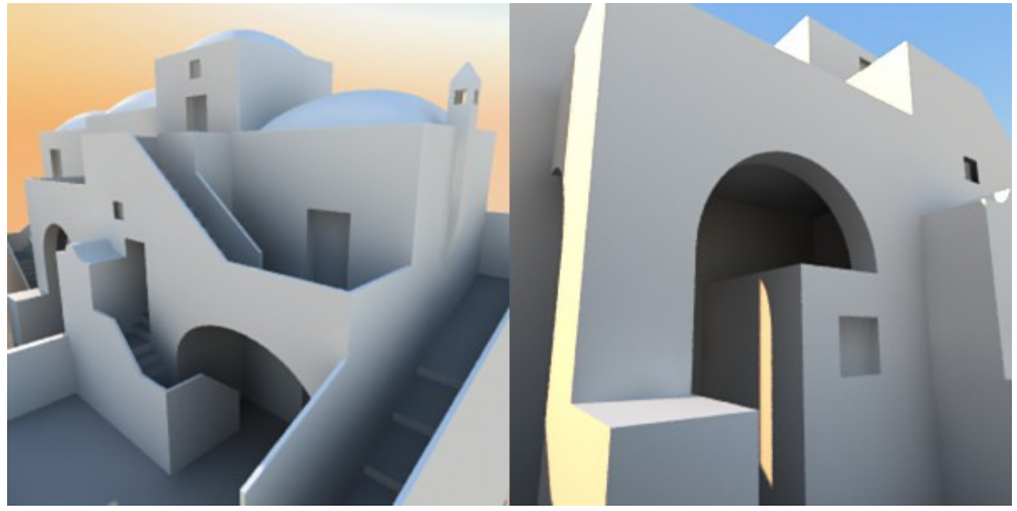
Grasshopper è una piattaforma per Rhinoceros della McNeal per far fronte agli algoritmi generativi e associativi di modellazione.

Applicazioni

Anche se esistono numerosi ed economici programmi di pura modellazione in questa sede ricorderemo soltanto:

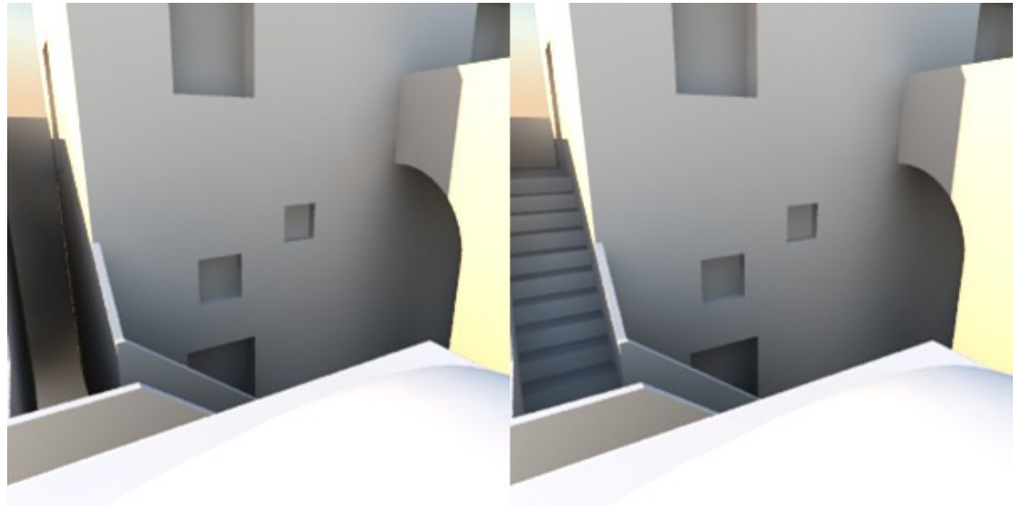
3D Studio Max dell'Autodesk, Maya dell'Autodesk, Softimage dell'Autodesk, tris d'Assi. Poi c'è Lightwave e poi Modo, Cinema 4D e Blender. Tutti fanno tutto o quasi. Dal punto al film, al videogioco, al virtuale. Sono gli stumenti di computer grafica più avanzati, coprono tutti gli aspetti delle prossime sezioni e necessitano di tempo ed esperienza per poter essere utilizzati efficacemente.

Verifica geometrica



Attraverso la modellazione solida è possibile, in ogni momento, accertarsi di tutte le possibili interazioni tra i corpi in questione. Verificare le geometrie. Aggiungere elementi di dettaglio dove necessario.

Si può percorrere, virtualmente, l'intero spazio progettato e alcuni software allertano se, in fase di modellazione, si verificano delle incongruenze.



Illuminazione e shading

Superfici

Se ci guardiamo attorno, ancora una volta, oltre alla struttura delle cose, vediamo una grande varietà di finiture superficiali. Una volta che il modello è stato capito, ad ogni parte o

pezzo dovrebbero essere assegnate diverse proprietà:

Colore: questo è probabilmente ciò che la gente percepisce più chiaramante. Tuttavia, non è sempre così semplice: di che colore è uno specchio? un bicchiere? la nostra pelle? Di solito, il colore è definito da più di una variabile. Ad esempio, controllare la quantità e il colore della luce dispersa da un oggetto, e il colore ambiente che controlla la sensibilità del materiale alla luce dell'ambiente circostante. Fondamentalmente, abbiamo il controllo della quantità di luce presente nelle ombre di un oggetto, dal momento che quasi mai appaiono in nero.

Specularità: controlla la brillantezza o scintilla di luce che produce su un oggetto. Un oggetto è molto lucido se ha specularità alta e Matt se la specularità è bassa.

Riflessione: controlla i riflessi, che provengono dall'ambiente circostante, sulla superficie dell'oggetto.

Trasparenza: il vetro di una finestra ci permette di vedere ciò che è dall'altra parte, se è pulito.

Rifrazione: queste distorsioni nelle superfici trasparenti sono il risultato del processo di rifrazione. Per questo motivo un bastone messo in acqua sembra piegarsi.

Altre proprietà: luminanza, caustiche, anisotropia, ecc, ma quelli di cui sopra sono le più importanti. Le diverse applicazioni 3D permettono di controllare questi parametri, il realismo di un materiale dipende dalla loro corretta regolazione. Anche un oggetto ben modellato può perdere tutta la sua credibilità se il colore è troppo saturo o se le superfici sono troppo lucideo e riflettenti.

Texture

Per molti oggetti non è possibile definire il solo colore di superficie: il pavimento del terrazzo, il legno nei mobili, o il modello di una camicia, che si compongono di diversi colori, con una distribuzione che è talvolta geometrica o, in altri momenti, completamente casuale. È per questo che ci rivolgiamo alle texture. Se abbiamo la scansione di un pezzo di marmo può essere applicato come finitura superficiale a qualsiasi oggetto. Questo tipo di operazione, conosciuta come texture bitmap, di solito proviene da un'immagine reale o una creata in un programma per immagini come Photoshop. Come in qualsiasi altra immagine bitmap, come una fotografia, è molto importante controllare la risoluzione, adattandola alle proprie esigenze. In caso contrario, quando si arriva troppo vicino all'oggetto, i pixel si riveleranno. Per evitare questo problema i programmatori hanno sviluppato un sistema di mappatura procedurale o *shader*. Riguardano gli algoritmi interni che il programma 3D crea, generalmente basato su strutture frattali che offrono diversi vantaggi:

La risoluzione è sempre ottimale; mai pixel show.
A causa della sua natura frattale, il programma è molto abile nell'imitare la natura caotica delle superfici, come la corteccia di un albero, le venature in marmo, o le fiamme nel fuoco.
Non si vede mai la ripetizione.

Di solito i calcoli sono più veloci che applicare una bitmap di grandi dimensioni. Tuttavia, alcuni shader possono risultare essere molto complessi.

Esistono vari sistemi di mappatura: piana, cubica, sferica, cilindrica, mappatura UV. Quest'ultima, per esempio, si applica con texture che si adattano alla forma come un guanto.

Illuminazione

Questo è uno degli argomenti più difficili in computer grafica. La difficoltà principale deriva dal fatto che la luce è emessa da un punto specifico, sia il sole, una lampadina, la fiamma di una candela.

Quando la luce si scontra con gli organismi, si illumina, ma si riflette anche in loro, così illuminante altri punti che in un primo momento non sembrano essere influenzati dalla luce.

In qualsiasi programma 3D, ci sono diversi tipi di luci per l'illuminazione di una scena. In generale, ci sono quattro classi di luce, ne esistono altri, ma non sono così importanti:

Radiale: luce che procede da un punto concreto posto nella scena, ed emette i suoi raggi in tutte le direzioni. È la luce ideale per una lampadina appesa luce o una fiamma.

Spot: le luci che possono essere orientate in una direzione specifica, sono tipicamente utilizzati in teatro o spettacoli. Siamo in grado di controllare la dimensione di apertura del cono di luce, nonché la sua diffusione e altri fattori.

Parallela: questa è la luce ideale per simulare il sole. La luce radiale può essere utilizzata per rappresentare il sole, dato che è una stella che si trova in un punto concreto e che emette luce in tutte le direzioni. Tuttavia, in relazione a noi, il sole si trova lontano, molto lontano. Tanto che, per la posizione di un punto luminoso a migliaia di chilometri di distanza non è pratico. Ecco perché la luce utilizzata si chiama parallela, poiché i raggi emessi sono paralleli, quasi come i raggi del sole, quando raggiungono la Terra.

Luci Ambient: un tipo di luce che non proviene da un punto concreto, viene da tutte le direzioni. Come abbiamo detto, la luce proviene da un punto specifico e arriva su di un oggetto da una direzione, illuminante da una certa angolazione. Ma la luce rimbalza. In una stanza con pareti bianche la luce, che penetra attraverso una finestra da una specifica direzione, rimbalza sulle pareti e gli oggetti che si trovano sul suo cammino, illuminando dolcemente un divano in una zona che dovrebbe essere in ombra. Tenere presente che in una stanza, non abbiamo quasi mai il buio totale. Un altro fenomeno si verifica all'aria aperta con la dispersione di luce che attraversa l'atmosfera, come nuvole o inquinamento. Per simulare questo tipo di effetto, è stato creata la luce ambiente.

Direttamente connesse con l'illuminazione sono le ombre prodotte dagli oggetti. Nel mondo reale ogni luce che analizziamo genera un'ombra quando entra in contatto con un ostacolo, anche le luci riflesse. D'altra parte, al fine di salvare i calcoli in un programma 3D, la luce può essere controllata in modo da non produrre ombre. Si dovrebbe pensare alla grafica al computer come un trompe l'oeil costante, ogni trucco è valido ogni volta che fa risparmiare tempo in calcoli, fino a quando la qualità dell'immagine non ne soffre in modo sostanziale.

Illuminazione e *shading*
in Computer grafica 3D, Wikipedia, novembre 2009

Lo *shading* (lett. "ombreggiatura") è il processo di determinazione del colore di un determinato pixel dell'immagine. Esso comprende in genere il processo di illuminazione (*lighting*), che ricostruisce l'interazione tra gli oggetti e le sorgenti di luce: a questo scopo sono necessari per un modello di illuminazione le proprietà della luce, le proprietà di riflessione e la normale alla superficie nel punto in cui l'equazione di illuminazione viene calcolata.

Per produrre una rappresentazione visuale dell'immagine efficace, bisogna simulare la fisica della luce. Il modello matematico più astratto del comportamento della luce è l'equazione di rendering, basata sulla legge di conservazione dell'energia. Essa è un'equazione integrale, che calcola la luce in una certa posizione come la luce emessa in quella posizione sommata all'integrale della luce riflessa da tutti gli oggetti della scena che colpisce quel punto. Questa equazione infinita non può essere risolta con algoritmi finiti, quindi necessita di approssimazione.

I modelli di illuminazione più semplici considerano solo la luce che viaggia direttamente da una sorgente luminosa ad un oggetto: questa è chiamata "illuminazione diretta". Il modo in cui la luce viene riflessa dall'oggetto può essere descritto da una funzione matematica, chiamata "funzione di distribuzione della riflessione bidirezionale" (*bidirectional reflectance distribution function*, BRDF), che tiene conto del materiale illuminato. La maggior parte dei sistemi di *rendering* semplifica ulteriormente e calcola l'illuminazione diretta come la somma di due componenti: diffusa e speculare. La componente diffusa, o Lambertiana corrisponde alla luce che viene respinta dall'oggetto in tutte le direzioni, mentre quella speculare alla luce che si riflette sulla superficie dell'oggetto come su uno specchio. Il modello di riflessione di Phong aggiunge una terza componente, ambientale, che fornisce una simulazione basilare dell'illuminazione indiretta.

Gli oggetti sono in realtà bombardati da moltissime sorgenti luminose indirette: la luce "rimbalza" da un oggetto all'altro finché non perde energia. L'"illuminazione globale" indaga questo comportamento della radiazione luminosa. Come l'illuminazione diretta, essa comprende una componente diffusa ed una speculare. La riflessione reciproca diffusa riguarda la luce che colpisce un oggetto dopo averne già colpito un altro. Dal momento che questo ha assorbito una data lunghezza d'onda dello spettro della luce che lo ha colpito, la luce che respinge ha un colore diverso da quella da cui è illuminato. La riflessione reciproca speculare si manifesta generalmente con caustiche (ovvero con la concentrazione della radiazione luminosa in un punto da parte di una superficie speculare, come quella ottenibile dalla luce solare con una lente).

Dato che gli algoritmi completi di illuminazione globale, come Radiosity e il photon mapping, richiedono grande capacità di calcolo, sono state sviluppate tecniche per approssimare l'illuminazione globale. L'algoritmo di occlusione ambientale, ad esempio, calcola da quanta luce ambientale può essere raggiunto ogni punto di un modello.

I modelli poligonali impiegati in applicazioni in tempo reale non possono avere un alto livello di dettaglio; il sistema più semplice per illuminarli è calcolare un valore di intensità luminosa per ogni poligono, basato sulla sua normale. Questo metodo è chiamato *flat*

shading, dato che rivela la forma "piatta" di ogni poligono. Per evitare questa "sfaccettatura", i valori corrispondenti ai vertici devono essere interpolati. Il Gouraud shading calcola l'intensità luminosa ad ogni vertice del modello basandosi sulla normale corrispondente, quindi esegue una interpolazione lineare su tutta la superficie del poligono. Il difetto più evidente di questa tecnica è che "perde" i riflessi speculari vicini al centro di un poligono. La soluzione data dal Phong shading è l'interpolazione su tutta la superficie del poligono delle normali ai vertici, e successivamente il calcolo dell'illuminazione pixel per pixel.

Queste equazioni si applicano a oggetti che possiedono colorazione propria, ma modellare ogni dettaglio presente sulla superficie di un oggetto sarebbe enormemente dispendioso. Col *texture mapping* si può descrivere la superficie di un oggetto senza aggiungere complessità alla scena: un'immagine (*texture*) viene "spalmata" sulla superficie di un oggetto, come un planisfero su una sfera per creare un mappamondo; durante lo *shading*, il colore del modello viene identificato in quello della texture, nel suo pixel ("texel") corrispondente.

Dato che le *texture* non possono rispecchiare l'illuminazione della scena, ma solo il colore del modello, per "perturbare" le normali ai poligoni si usa il *bump mapping*. Questo fa uso di immagini che contengono, anziché un colore, un valore usato per modificare la normale al poligono nel punto corrispondente, e modificare così la forma della superficie. Questa tecnica aggiunge "ruvidità" alle superfici con grande risparmio di poligoni.

Il *normal mapping* è una tecnica che sostituisce invece di perturbare la normale alla superficie: una *normal map* è un'immagine a 3 canali in cui ogni pixel rappresenta un vettore 3D, ovvero la normale al punto stesso.

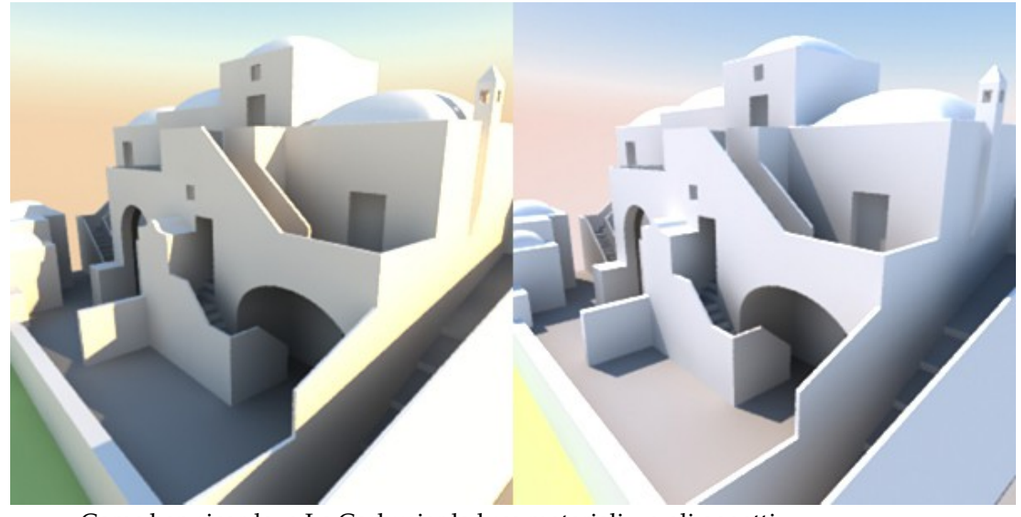
L'obiettivo di ogni algoritmo di *shading* è determinare il colore risultante di uno specifico punto sulla superficie di un oggetto. Gli *shader* programmabili offrono grande versatilità in questo, basandosi su linguaggi di programmazione specifici detti "linguaggi di *shading*". Questi linguaggi vengono sviluppati per applicazioni specifiche nella computer grafica, e includono algebra lineare e caratteristiche mirate alle problematiche di illuminazione. Gli *shader* possono includere qualsiasi tecnica di illuminazione, *texture mapping* e manipolazione geometrica. Uno "*shader* procedurale" determina il colore risultante in maniera completamente algoritmica: possono così risultare convincenti senza bisogno di grandi *texture*.

Formano una classe a sé stante i "*vertex shader*" e i "*pixel shader*", designati appositamente per funzionare insieme ad algoritmi *scanline* e per girare su una GPU. Mentre in precedenza ogni *hardware* grafico implementava una specifica *pipeline* che costringeva l'utilizzatore ad usare esclusivamente il modello di illuminazione per cui era programmato l'hardware, con questa categoria di *shader* ogni momento del *rendering* è sotto il controllo dello sviluppatore.

Applicazioni

Le applicazioni di questo paragrafo sono da ricercare nei paragrafi precedente e successivo. Escludendo qualche caso specifico di illuminotecnica tipo Dialux, l'illuminazione è di competenza di sistemi di modellazione, resa e animazione o di programmi di rendering, che vedremo poi.

Verifica illuminazione



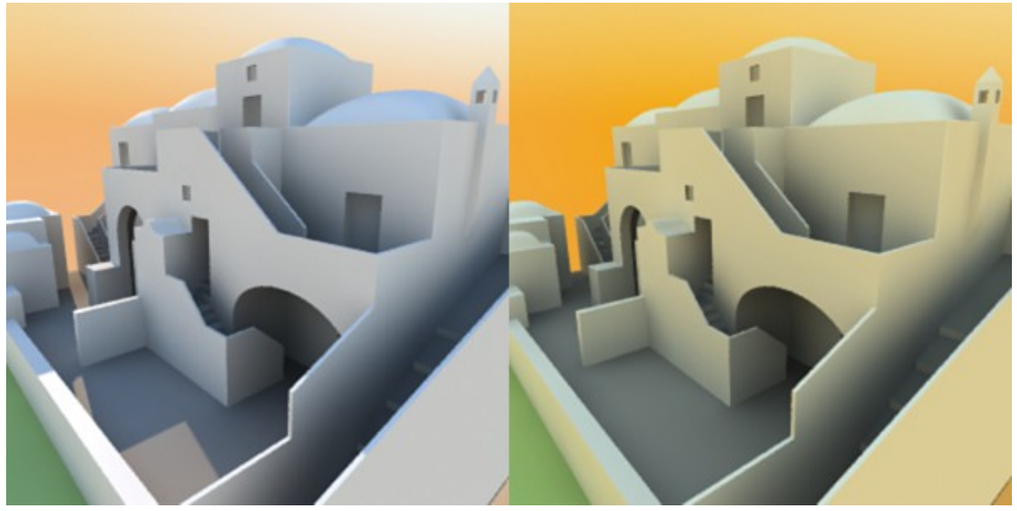
Come ben ricordava Le Corbusier la luce materializza gli oggetti.

Oltre alle geometrie possiamo valutare le ombre con estrema precisione. Percorrere, temporalmente, le fasi della giornata o dell'anno.

Scegliere sistemi e soluzioni più adatte alla rappresentazione. Possiamo, se necessario, falsare dei valori per accentuare una serie di effetti.

Possiamo, semplicemente, vedere prima. E possiamo farlo con estrema precisione e, allo stato attuale, anche abbastanza rapidamente.

Possiamo, infine, giocare con luce e corpi. E, giocando, trovare una particolare condizione, la condizione particolare che stavamo cercando.



Rendering

Fino ad ora abbiamo fatto noi quasi tutto il lavoro. Ora tocca alla macchina. In un secondo di film abbiamo 24 fotogrammi e il computer deve calcolare ognuna di queste immagini. Questo processo è chiamato rendering. Esistono svariati sistemi di rendering o algoritmi di resa ma i più importanti sono:

Wireframe: di solito usato per test di movimento, per vedere come stanno andando le cose ed evitare una sorpresa dopo. È il più veloce, in quanto mostra solo poche righe per definire i poligoni di ciascun elemento. Non c'è trama riconoscibile, solo la struttura degli oggetti come nella modellazione, ma è molto utile per testare la qualità dei movimenti in una animazione prima di passare a sistemi più lenti.

Phong: si tratta di un algoritmo piuttosto goffo diffuso in molti programmi. Non si possono rappresentare le ombre, per non parlare di una serie di altri fenomeni fisici. Viene utilizzato solo per l'animazione di prova.

Raytracing: qui, riflessi, rifrazioni e ombre proiettate sono calcolati in base ai parametri assimilati al mondo reale, dando un risultato che è abbastanza vicino alla realtà. La parte negativa è che è molto più lento del Phong, ma di solito è usato più in immagini che in animazioni. In questo sistema ogni raggio visivo che esce dalla fotocamera e raggiunge l'oggetto e, in base alle tariffe di riflessione, trasparenza e di rifrazione, passa ad altri oggetti o le luci da lì. Ogni piccolo raggio visivo che esce dalla nostra macchina fotografica sarà un pixel per la nostra immagine.

Radiosity: è il migliore di tutti i sistemi di rendering, ma è anche il più lento, con una differenza: calcola anche le interazioni tra la luce e il colore degli oggetti che sono nelle immediate vicinanze. Per esempio, se una pallina rossa è vicino ad una parete bianca, una zona della parete più vicina alla sfera apparirà di colore rosso scuro. Un altro esempio: se si illumina una parete, che riflette una parte di questa luce e fornisce una luce debole per gli oggetti che si trovano nelle vicinanze. Questo è il sistema perfetto per la simulazione realistica in materia di architettura, interni in particolare, dal momento che illustra molto bene come la luce si comporta in queste condizioni. È anche molto usata per aggiungere realismo alle scene per i giochi 3D video. Le scene vengono precalcolate e salvate su un disco, altrimenti sarebbe impossibile giocare in tempo reale.

Rendering
da Wikipedia, novembre 2009

Il Rendering è un termine dell'ambito della computer grafica; identifica il processo di "resa" ovvero di generazione di un'immagine a partire da una descrizione matematica di una scena tridimensionale interpretata da algoritmi che definiscono il colore di ogni punto dell'immagine. La descrizione è data in un linguaggio o in una struttura dati e deve contenere la geometria, il punto di vista, le informazioni sulle caratteristiche ottiche delle superfici visibili e sull'illuminazione.

Descrizione

È uno dei temi più importanti della grafica tridimensionale computerizzata e in pratica sempre in relazione con tutti gli altri. Nella "pipeline grafica" è l'ultimo importante stadio e fornisce l'aspetto finale al modello e all'animazione.

Con il crescente perfezionamento della grafica computerizzata dal 1970 in avanti è diventato un oggetto di studio e ricerca sempre più importante. È usato per: montaggio video/giochi per computer, simulatori, effetti visuali per film/serie TV, e visualizzazione di progetti. Ciascuno con una differente combinazione di caratteristiche e tecniche.

Sono disponibili in commercio un gran numero di motori di render, alcuni dei quali integrati nei più diffusi pacchetti di modellazione e animazione tridimensionale, alcuni altri indipendenti, altri ancora distribuiti come progetti open source.

Dall'interno, un renderizzatore è un programma progettato attentamente e basato su una combinazione selezionata di metodi relativi a: ottica, percezione visiva, matematica e ingegneria del software.

Nel caso della grafica tridimensionale, il rendering è un processo lento e richiede un gran numero di elaborazioni da parte della CPU, oppure è assistito in tempo reale dagli acceleratori 3D delle schede grafiche (per i giochi tridimensionali).

Uso

Quando l'elaborazione preliminare della scena (una rappresentazione wireframe solitamente) è completa, inizia la fase di rendering che aggiunge texture bitmap o texture procedurali, luci, bump mapping, e posizioni relative agli altri oggetti. Il risultato è un'immagine completa che è possibile vedere.

Nel caso di animazioni per pellicole cinematografiche, molte immagini (fotogrammi) devono essere disegnate e assemblate in un programma capace di creare un'animazione di questo tipo. La maggior parte dei programmi di elaborazione 3D sono in grado di elaborare queste immagini.

Fenomeni

Le immagini possono essere analizzate in termini di una serie di fenomeni visibili. Le ricerche e i progressi nel campo del rendering sono state in gran parte motivate dal tentativo di simularli in modo accurato ed efficiente.

- shading — ombreggiatura; variazione del colore e luminosità di una superficie a seconda della luce incidente
- texture mapping — un metodo per definire i dettagli del colore di una superficie mettendola in corrispondenza con un'immagine (texture)
- bump mapping — un metodo per simulare irregolarità nella forma di una superficie mettendola in corrispondenza con un'immagine (bump map) che definisce una perturbazione fittizia della superficie, usata solo per ricavarne una distorsione della direzione perpendicolare (normale) impiegata nei calcoli per la propagazione della luce.
- normal mapping — un metodo simile al bump mapping in cui l'immagine definisce direttamente come perturbare la normale della superficie in quel punto.
- displacement-mapping — estrusione di una superficie secondo le normali tramite un'immagine in scala di grigi, producendo una reale perturbazione della forma della superficie, (per esempio per creare una montagna a partire da una superficie piana).
- distance fog — attenuazione e dispersione della luce nel passaggio attraverso l'aria o altri mezzi; solo il vuoto è perfettamente trasparente.
- shadows — gestione delle ombre proiettate
- soft shadows — ombre parziali prodotte da sorgenti di luce estese
- reflection — riflessioni speculari o quasi
- transparency — trasmissione della luce attraverso un oggetto
- rifrazione — deviazione della luce nel passaggio da un mezzo all'altro
- illuminazione indiretta e Global illumination — tenere conto della luce riflessa più volte (il minimo è una sola riflessione, sorgente di luce -> oggetto -> camera)
- caustiche — accumulo di luce riflessa o rifratta proiettata in forme caratteristiche su altri oggetti (ad esempio la forma a cardioide della luce riflessa dall'interno di un cilindro o le forme irregolari in movimento sul fondo di una piscina)
- profondità di campo o DoF (Depth of Field) — simulazione della progressiva sfocatura degli oggetti posti a distanza crescente dalla superficie di messa a fuoco (profondità di campo).
- motion blur — simulazione della sfocatura degli oggetti in movimento rapido come in una ripresa fotografica.
- subsurface scattering o SSS — simulazione del comportamento della luce che penetra un oggetto di materiale traslucido come la cera o la pelle umana (dispersione subsuperficiale).
- ambient occlusion — simulazione del comportamento della luce in prossimità di volumi occlusi dove i raggi luminosi faticano ad entrare e uscire
- anisotropia — simulazione di un materiale che riflette la luce in modo diverso per ogni direzione tangente al punto.

Tecniche

Le principali tipologie di algoritmi per risolvere il problema sono:

- radiosity: collegata alla matematica agli elementi finiti;
- ray tracing: collegata dalla matematica probabilistica.

Questi approcci possono essere particolarmente intensi dal punto di vista computazionale, perché entrambi creano una struttura abbastanza completa per la gestione delle equazione di rendering.

Per le applicazione *real-time*, non è pensabile di eseguire una elaborazione completa. In genere si semplifica il problema con una delle seguenti approssimazioni:

- Nessuna illuminazione, solo texture mapping, poiché il colore intrinseco di un oggetto ha l'influenza maggiore sul suo aspetto.
- Illuminazione diretta: si tiente conto solo della luce che va dalla fonte di illuminazione alla superficie, non di quella riflessa da altre superfici presenti nella scena. Questa luce potrà essere tenuta in considerazione con altri casi speciali attraverso il precalcolo.

Alcuni dei principali algoritmi, sono:

- Algoritmo del pittore
- Algoritimi di tipo scanline
- Algoritmi che utilizzano lo Z-buffer
- Illuminazione globale
- Radiosity
- Ray tracing
- Volume rendering

Chi deve eseguire il rendering di grandi quantità di immagini (per esempio quelle di una sequenza cinematografica) usa una rete di computer connessi tra loro, detta render farm.

L'attuale stato dell'arte per la costruzione di scene in 3D per la creazione di film è il linguaggio di descrizione delle scene RenderMan creato dalla Pixar. (da confrontare con formati più semplici per la descrizione di un ambiente 3D come VRML o API come DirectX o OpenGL che sfruttano l'accelerazione hardware delle moderne schede grafiche).

Applicazioni

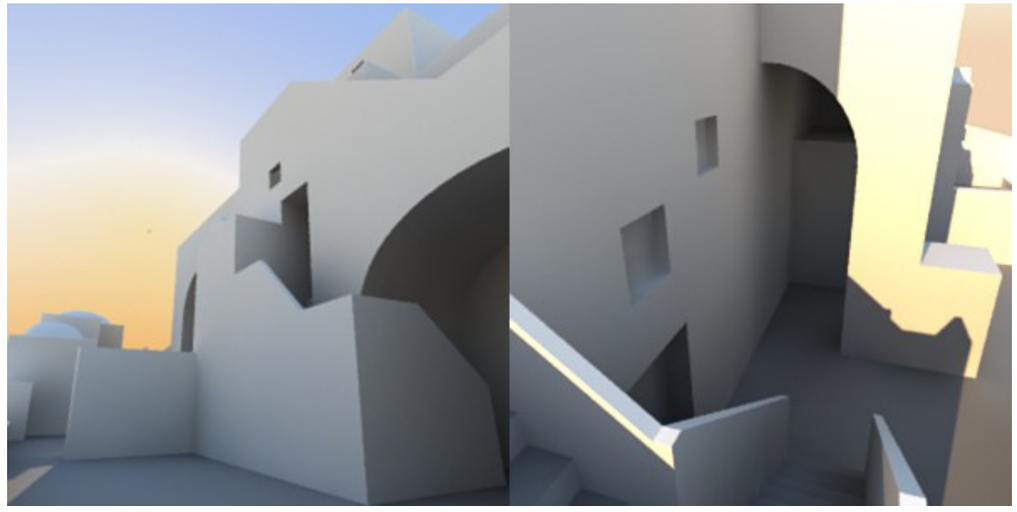
Avevo pensato di riportare un elenco in ordine di apparizione, considerando per *apparizione* il tempo impiegato dal singolo software per calcolare il risultato finale. Ma tempo

e strumentazioni, in questo momento, me lo impediscono.

Grosso modo (gli ultimi saranno i primi):

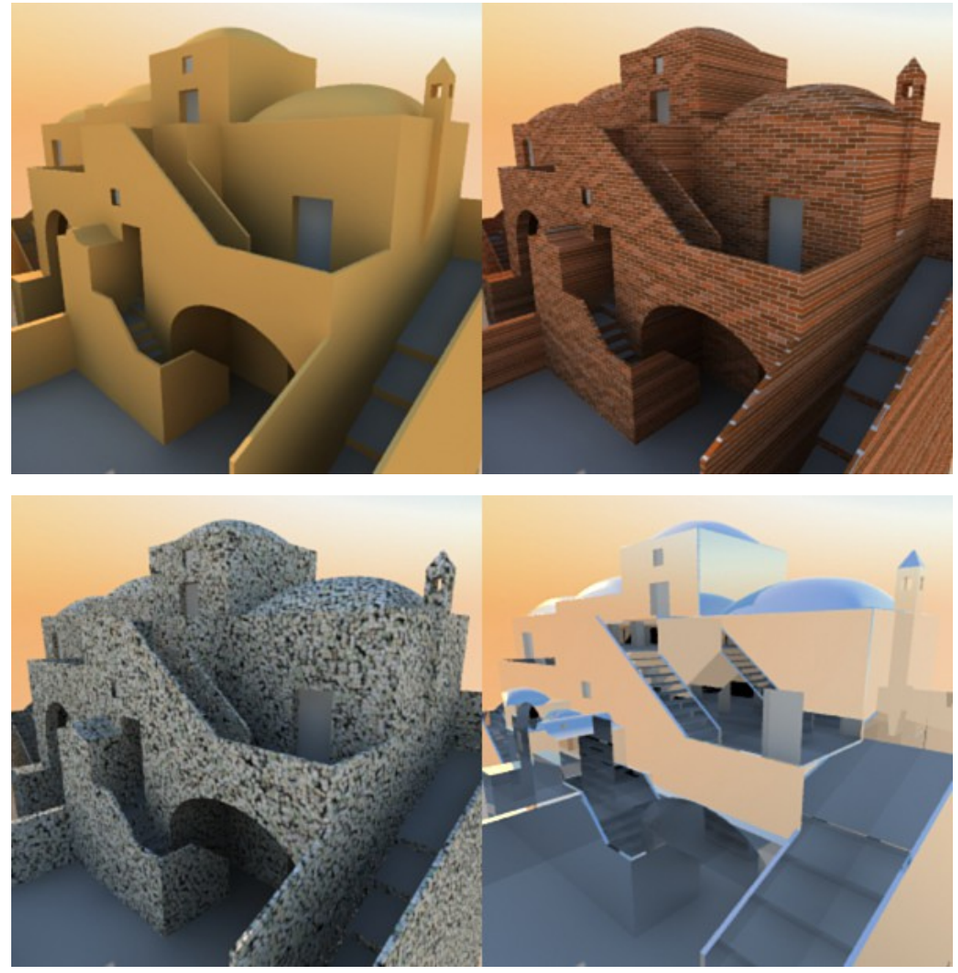
Maxwell, Fryrender, FPrime, Final Render, Mental Ray, Kray, Vray.

Verifica materiali



Valutare tagli, angolazioni. Viste impossibili. Ma anche e sopratutto verificare materiali e geometrie.

Aver tutte le caratteristiche fisiche dei materiali a portata di mouse. Tutte le interazioni, tutti gli effetti, le soluzioni, i costi.



Animazione

Questo processo è simile a un film o una fotografia: le scene sono costruite, artisticamente dipinte, e adeguatamente illuminate, ora, tutto quello che si deve fare è noleggiare buoni attori e fotografare o filmare l'azione. Tutto qui.

In alcune animazioni, il movimento è limitato a una serie di *voli* o attorno a una scena. L'esempio più evidente è la computer grafica architettonica. In un programma 3D, una macchina fotografica è un oggetto che può essere aggiunto a una scena, con alcune caratteristiche come il formato di immagine, apertura di messa a fuoco, ecc, al fine di catturare l'ambiente da un punto virtuale di vista.

La cosa comincia a complicarsi se si desidera spostare altri oggetti intorno. Raramente è semplice come lo spostamento o un elemento di svolta, la maggior parte del tempo, i pezzi vengono spostati e ruotati in relazione a altri pezzi e, nei casi più complessi, la gente si muove, cammina, corre, ride, urla o piange. Ora stiamo parlando di una disciplina estremamente complessa. Le formiche in *A Bug's Life* o i giocattoli di *Toy Story* o, ancora, i dinosauri di *Jurassic Park* rappresentano il più alto livello di perfezione che è stato raggiunto in questo campo. A questa perfezione si arriva attraverso gli sforzi di molte persone altamente specializzate, in collaborazione con le migliori macchine e programmi, per lunghi periodi di tempo.

Simulazioni fisiche

Uno degli ultimi, poco eclatanti, acquisti Autodesk riguarda appunto un software di simulazione fisica adattato all'architettura.

Per simulazione si intende un modello della realtà che consente di valutare e prevedere lo svolgersi dinamico di una serie di eventi susseguenti all'imposizione di certe condizioni da parte dell'analista o dell'utente. Un simulatore di volo, ad esempio, consente di prevedere il comportamento dell'aeromobile a fronte delle sue caratteristiche e dei comandi del pilota.

Le simulazioni sono uno strumento sperimentale molto potente e si avvalgono delle possibilità di calcolo offerte dall'informatica; la simulazione, infatti, non è altro che la trasposizione in termini logico-matematica-procedurali di un "modello concettuale" della realtà; tale modello concettuale può essere definito come l'insieme di processi che hanno luogo nel sistema valutato e il cui insieme permette di comprendere le logiche di funzionamento del sistema stesso.

Le simulazioni possono anche avere carattere ludico; oggi esistono sul mercato diversi software (videogiochi di simulazione) che consentono di simulare il comportamento di persone, veicoli, civiltà. Ovviamente il livello di approfondimento di tali simulazioni, in termini di modello concettuale sottostante è più basso.

Simulazione
da Wikipedia, novembre 2009

Nell'ambito delle simulazioni, acquisisce notevole importanza la simulazione del funzionamento dei processi produttivi e logistici. Tali sistemi sono infatti caratterizzati da elevata complessità, numerose inter-relazioni tra i diversi processi che li attraversano, guasti dei segmenti, indisponibilità, stocasticità dei parametri del sistema. Consideriamo, ad esempio, un impianto semplice per la produzione di un unico articolo, con solamente due macchine automatiche ed imballaggio manuale; in questo semplice sistema l'arrivo delle materie prime, la durata delle lavorazioni, il tempo necessario agli operatori per imballare sono tutte variabili stocastiche, in quanto il ritmo produttivo e di arrivo non è costante; inoltre, le macchine sono soggette a guasti e manutenzione, gli operatori possono non essere sempre disponibili etc.

Il progettista degli impianti industriali e il responsabile delle operations possono certamente avere interesse a valutare con anticipo l'effetto delle loro scelte su tali sistemi complessi, in termini, ad esempio, di capacità di produzione, tempo di attraversamento, scorte, blocchi. Possono inoltre avere dei problemi riguardo al dimensionamento di macchine, magazzini, flotta dei carrelli trasportatori e simili.

La simulazione, consentendo l'analisi della realtà ad un elevato livello di dettaglio e padroneggiando facilmente la complessità del sistema, fa si che alla fine sia possibile ottenere un gran numero di informazioni utili. Il prezzo da pagare per tale completezza è ovviamente il tempo; le operazioni di programmazione sono infatti assai lunghe, affinché si possano ottenere dei dati sufficientemente sensati e tali da dare la possibilità di ottenere un modello della realtà ad essa aderente.

Passi e procedure

Al fine di poter procedere correttamente per avere un modello di simulazione utile e funzionante è opportuno procedere con una serie di passi:

- *Definizione degli obiettivi e delle problematiche da esaminare*: un' attenta analisi del problema consente di circoscriverne l'esame riducendo

il successivo tempo di analisi;

- *Stesura di un modello concettuale*: consiste nella comprensione e modellazione del sistema produttivo che si intende simulare; questa fase è particolarmente importante in quanto definirà il comportamento dei diversi flussi di materiale e di informazioni che attraverseranno il modello.
- *Validazione del modello concettuale*: si tratta di un confronto con la direzione dell'impresa e con gli operatori per assicurarsi della capacità del modello di offrire un'immagine consistente della realtà.
- *Analisi dei dati in ingresso*: la raccolta e l'analisi dei dati che diverranno la base per la definizione dei parametri di funzionamento del sistema (ad esempio: i diversi tempi di lavoro di una singola macchina). Attraverso le tecniche del calcolo delle probabilità diviene possibile definire una distribuzione di probabilità per ogni parametro, da inserire all'interno del modello.
- *Scrittura del modello in termini matematici*
- *Calibrazione e valutazione*
- *Definizione di un piano degli esperimenti*: una singola iterazione ("run") di simulazione non ha alcun significato; rappresenta solo una delle possibili evoluzioni del sistema. È quindi opportuno effettuare diversi "run" per poi analizzare i parametri in uscita. La lunghezza della singola iterazione e il numero delle iterazioni vengono determinate in questa fase.
- *Analisi dei dati in uscita*: dopo aver raccolto i dati relativi ai parametri, depurati da eventuali transitori è possibile creare degli intervalli di confidenza ovvero stimare il "range" di valori in cui i parametri che analizzano il problema proposto al primo passaggio possono oscillare.

Elementi caratteristici di un modello di simulazione

- Entità - Le entità sono gli elementi "trattati" dal processo; tali "oggetti" hanno la caratteristica di essere "temporanei", e di subire passivamente le trasformazioni. Ad esempio, in un'impresa di lavorazioni meccaniche, i semilavorati e le materie prime, che devono essere fresati, spianati etc possono essere modellizzati come "entità". Naturalmente, è possibile simulare anche processi in cui la produzione non riguardi un bene fisico, ma un servizio: in questo caso, le entità rappresenteranno informazioni, documenti, clienti, a seconda delle necessità.

Le entità, all'interno del modello, possono essere considerate a loro volta come:

- Anonime - Nella maggior parte dei casi, non interessa tenere traccia del singolo pezzo in lavorazione o in generale in transito nel sistema. Pertanto le entità non sono caratterizzate, e vengono considerate come un "flusso" indistinto.
- Personalizzate - Caso duale del precedente, si presenta quando l'analista, spesso per il numero esiguo di pezzi in lavorazione, ha interesse a considerare i parametri di lavorazione del singolo pezzo.
- Operazione: rappresenta una delle trasformazioni che avranno luogo sull'entità.

Possono essere individuati due cicli di operazioni:

- Il ciclo macchina: attinente agli stati (vedi) ed operazioni che la macchina attraverserà, ovvero l'insieme di tutte le possibili successioni di operazioni e attese.
- Il ciclo pezzo: rappresenta il percorso delle entità nel sistema, le macchine visitate e le operazioni subite
- Macchine: rappresentano gli elementi "fissi" del sistema, la cui definizione degli stati definisce univocamente la situazione generale del sistema, e delle quali sono di rilevanza per l'analista soprattutto le prestazioni. Le macchine possono essere fisiche, ed in questo caso ci si riferisce a macchine realmente presenti nel sistema da modellizzare, o "logiche", ed in questo caso compiono operazioni "fittizie" fisicamente, ma presenti logicamente nel sistema (ad esempio, il controllo di quantità in ingresso nell'impianto non ne provoca trasformazioni "fisiche" ma lo "trasforma" da "lotto da controllare" a "lotto controllato").
- Stati: gli stati sono delle variabili (di tipo vario: possono essere numeri o valori logici) che descrivono lo stato del sistema e delle sue componenti, per ogni istante di tempo.
- Eventi: fenomeni che modificano lo stato del sistema (ad esempio, la fine di una lavorazione modifica lo stato di una macchina da "occupata" a "libera").
- Code: insiemi di entità che non possono accedere alle trasformazioni successive in quanto la macchina risulta occupata.
- Attributi: proprietà permanenti di un insieme di entità o di una macchina.
- Orologio locale: orologio che contiene, a livello di singola macchina, l'istante di tempo che identifica la fine della lavorazione in corso.
- Orologio generale: orologio che regola lo scorrere generale del tempo di simulazione.

Funzionamento dei simulatori

- Fase di scan
- Fase di rescan

Tipi di simulatori

- Orientati agli eventi
- Orientati ai processi
- Orientati alle attività

Programmazione del modello

Una volta costruito il modello esso va tradotto in un programma su calcolatore. È possibile usare linguaggi *general purpose* quali Pascal, C, C++, per i quali esistono delle librerie di *routines* orientate alla simulazione. Esistevano anche diversi linguaggi specializzati quali ad esempio SIMSCRIPT, MODSIM e GPSS. Una interessante alternativa è quella di ricorrere ad applicazioni di tipo interattivo per la simulazione, fra gli altri: AutoMod, Simul8, Arena Simulation, Witness, Extend e Micro Saint. Tali applicazioni sono di facile uso e quindi molto adatte a costruire rapidamente modelli, anche sofisticati, ma sono meno versatili e potenti dei linguaggi specializzati o di quelli *general purpose*. Per problemi di piccole dimensioni è anche possibile usare strumenti informatici di uso comune quali le spreadsheet. Tali strumenti possono essere utili quando si vuole rapidamente avere un'idea del funzionamento di una singola componente o di un sottosistema di un sistema complesso.

Applicazioni

Per i software di animazione si deve far riferimento al paragrafo riguardante la modellazione.

Progettazione sostenibile

In architettura l'aggettivo ecocompatibile si riferisce ai processi o prodotti che hanno la capacità di integrarsi con l'ambiente in cui vive l'essere umano e in generale con l'intero l'ecosistema. Da ciò si deduce che il concetto di ambiente, inteso fino agli anni 80 solo come ambiente naturale, si estende al "sistema di interrelazioni strutturali tra un soggetto e il suo spazio di pertinenza". L'ecocompatibilità è inoltre strettamente legata al concetto di sviluppo sostenibile introdotto nel 1987 dal Rapporto Brundtland secondo cui "lo sviluppo è sostenibile se soddisfa i bisogni delle generazioni presenti senza compromettere quelli delle generazioni future".

In quest'ottica il progetto ecocompatibile, oltre che rispondere alle complesse esigenze dell'utente, ha l'obbligo di promuovere lo sviluppo sostenibile in relazione ai tre grandi ambiti di riferimento: economico, ambientale, sociale.

In particolare la progettazione ecocompatibile dell'ambiente costruito è connotata da un rapporto con il contesto, inteso come ambiente fisico e antropizzato, tale da garantire condizioni di benessere con un ridotto consumo di risorse ambientali e un basso livello di inquinamento. L'obiettivo è oggi quello di diffondere l'approccio ecocompatibile e di integrarlo all'interno dell'iter progettuale ordinario, nelle diverse fasi progettuali e nelle diverse scale del processo edilizio.

Non esistono ancora molti software orientati specificamente verso la progettazione sostenibile ma il mercato preme e cominciano a vedersi i primi risultati. Ecotect.

Ecotect Analisys, altro prodotto acquistato dall'Autodesk, è un software di analisi che permette di valutare le prestazioni dell'edificio e del sito. Permette di effettuare un'analisi approfondita dei fattori ambientali quali irraggiamento solare, ombreggiatura, illuminazione con luce diurna e prestazioni termiche.

Irraggiamento solare: permette la visualizzazione dell'irraggiamento solare su finestre e superfici per mostrare l'irraggiamento solare incidente differenziale calcolato su un determinato periodo di tempo.

Ombre e riflessi: visualizzazione intereattiva di ombre, penetrazione solare e riflessi.

Progettazione dell'ombreggiatura: progettazione di dispositivi per ombreggiare una finestra in modo ottimale e calcolo dell'irraggiamento solare su una determinata finestra in un determinato periodo di tempo.

Illuminazione con luce diurna: calcolo dei livelli di illuminazione naturale e artificiale con tanto di analisi del fattore di luce diurna e componente cielo verticale.

Prestazioni termiche: calcolo dei carichi di riscaldamento e raffreddamento per modelli con qualsiasi numero di zone o tipo di geometria.

Accesso alla luce: analisi degli angoli di proiezione del sito e valutazione delle ostruzioni, calcolo delle componenti cielo verticale, visualizzazione della linea di ostruzione del cielo.

Autodesk Ecotect Analisys è la soluzione ottimale per integrare l'analisi nel processo di progettazione di ogni architetto.

I vantaggi derivano da un'analisi già nella fase di lavoro concettuale in modo da prendere in esame tutti i parametri prestazionali dell'edificio.

Fornisce un elevato grado di dettaglio di feedback visivo per una veloce e facile comprensione dei risultati.

Supporta il workflow BIM di Autodesk Revit in tutte le fasi di progettazione.

Capitolo 3° - Istruzione

Con l'istruzione, il gap generazionale e la pratica della trasmissione della conoscenza si fa una pausa di riflessione chiosando con l'attuale e futura diffusione dell'insegnamento su internet.

Istruzione

Gli studenti che avranno finito gli studi durante il primo decennio del XXI secolo sono l'ultimo piccolo gruppo di una generazione di transizione; quando questa transizione sarà completa la professione risulterà irrevocabilmente cambiata, diventando il culmine del processo che e in corso in maniera cosi marcata. L'applicazione dell'insegnamento dell'informatica nelle scuole di architettura rispecchia quella nella professione, con lo stesso grado di enfasi derivante dal suo uso come strumento per integrarlo nel processo progettuale.

Scrive James Steele in *Architettura e Computer*: "nei soli Stati Uniti, si stima che a causa della prosperità economica derivante soprattutto dalla rivoluzione dell'informazione e dei media, il 98 per cento della classe 2000 accederà alla professione con una media di reddito annuale di 28.000 dollari. Se da una parte questo salario iniziale non e cambiato significativamente negli ultimi 20 anni, le migliorate abilita tecniche, paragonate ai più esperti datori di lavoro, coordinatori di progetto e architetti meno abili nell'uso dei computer faranno far camera più facilmente, arrivando a raggiungere 43.000 dollari all'anno nel giro di tre anni. Nel frattempo, circa il 70 per cento del tempo sarà trascorso di fronte a un monitor e siccome e più o meno lo stesso tempo che in passato veniva dedicate al disegno a mano, vi e una notevole differenza che influenza direttamente la responsabilità degli educatori nel prossimo futuro. Le ricerche sull'uso del computer negli uffici di tutto il mondo sono inadeguate, ma una ricerca realizzata verso la meta degli anni novanta negli Stati Uniti su un campione a caso stratificato di 450 unità, ha mostrato che i tirocinanti sono molto più abili con i software rispetto agli architetti, ai coordinatori di progetto e ai datori di lavoro più esperti che dovrebbero prepararli in quella che viene ancora considerata una relazione maestro-allievo".

Il gap generazionale è evidente. La rivoluzione digitale, veloce e totalizzante, risulta essere un grosso colpo da assimilare lentamente. Ed è ciò che sta avvenendo. Non potendo riformare le scuole di architettura sull'uso del calcolatore, come auspicato anche da Aldo Loris Rossi, dobbiamo aspettare che il tempo faccia il suo lavoro. Queste ultime considerazioni riguardano l'intero mondo occidentale ma sono ovviamente più evidenti in zone periferiche quale l'attuale realtà napoletana.

La tecnologia non è neutrale

Ogni strumento possiede delle caratteristiche intrinseche, delle peculiarità. Un sistema di strumenti quale quello dell'esperienza digitale apporta una serie di modifiche nel modo di fare e di pensare e, poi, di nuovo nel fare. È nel *fare di nuovo* che si manifesta palesemente la non neutralità della tecnologia digitale. Una volta compiuto il primo passo si scopre il mondo con occhio nuovo. Si comincia a pensare diversamente. Si evitano più facilmente i vecchi errori si ha a che fare con nuovi problemi. Nuovi problemi, nuove soluzioni. Mi verrebbe da dire meno male che la tecnologia non è neutrale. Tutto sommato anche il titolo *la tecnologia non è neutrale* non ha molto senso. La tecnologia interviene e ridefinisce. La trasformazione è l'unica cosa che possiamo percepire. Continuamente.

Mantenere l'uso della mina

Riporto integralmente il piccolo paragrafo di James Steele, sempre in *Architettura e Computer*, sull'uso della mina aggiungendo che, digitalmente parlando, non esiste nulla di più veloce di un blocco per appunti e una matita per fissare un'idea.

"Coloro che prevedono la fine della matita, o che proclamano allegramente che essa fa già parte di oggetti in disuso come la frusta del cocchiere e la macchina da scrivere, dovrebbero considerare la misura in cui molti professionisti attualmente integrano efficacemente tecniche grafiche e digitali in un'ampia gamma di applicazioni, come indicate da Eric Owen Moss e Moore, Ruble e Yudell tra una miriade di altri studi che potrebbero essere menzionati. Utilizzata in una fase concettuale come avviene per Gehry e Moss, o come affermazione simbolica dell'associazione tradizionale dell'architetto con i grafici, come per la MRX o come controllo della realtà sullo spazio virtuale in varie fasi del processo digitale, come nel caso di Jon Jerde, sembra che la matita sarà utilizzata ancora per un bel po'".

L'importanza di Gehry

Frank Gehry verrà ricordato oltre che per la sua architettura anche come figura paradigmatica nel panorama della neonata architettura digitale. Pur essendo una sorta di ibrido, i prossimi penseranno con le nuove tencologie, fissa in sé lo sforzo auspicabile da un professionista considerato tale. Così James Steele: "iI contrappunto esilarante di questa crescente distanza dalla realtà materiale e la posizione difesa da Frank Gehry e meglio dimostrata nel Guggenheim Museum di Bilbao, dove CATIA ha mostrato di permettere una stretta integrazione tra progetto e costruzione. Gehry crede che la capacita del computer di convertire dettagli documentati in una forma costruita semplicemente attraverso il trasferimento di un disco ridarà all'architetto, come centro delle informazioni, un ruolo centrale nel processo costruttivo. Ciò presuppone, naturalmente, che i dettagli siano innanzitutto corretti. È una buona informazione, non la tecnica, che conquisterà il rispetto di imprenditori edili e clienti. Quello che ha impressionato Gehry, rendendolo alla fine entusiasta dell'uso del computer e stata la possibilità di costruire forme che prima non avrebbe considerate in quanto gli imprenditori edili non le avrebbero neppure toccate; egli li ha rassicurati e ha liberato la propria immaginazione. Ma, come Picasso che ha studiato l'anatomia prima di essere in grado di frammentarla in maniera convincente, Gehry ha una lunga esperienza nel mettere insieme un edificio ermetico, strutturalmente stabile e funzionalmente operativo prima che inizino i suoi voli di fantasia digitali e che la sua conoscenza acquisita con difficoltà abbia informato la sua arte. Gli studenti e i tirocinanti devono raggiungere ora questo punto con una guida meno esperta. Frattanto, un'osservazione generale, non ancora convalidata dalla ricerca metodica, e che in assenza di una pedagogia pienamente sviluppata e di un dibattito sulle questioni sollevate in questa sede la sorte di uno studente e soggetta all'interesse individuale e alla motivazione nonché all'inclinazione e abilità del suo o della sua preside e del suo istruttore di studio Pochissime scuole sembrano avere un programma integrate per l'introduzione del computer in tutti gli studi, concentrandosi su una selezione di corsi che, al contrario, insegnano la tecnica. L'idea della teoria informatica e ancora una nozione completamente aliena, in parte a causa di un sentimento di inevitabilità tecnologica che preclude la discussione, nonché la selezione del programmi, che indirizza le capacita di cui necessitano i laureati per ottenere un lavoro, completare l'internato e superare gli esami di stato, prima che si possano sollevare questioni meno *pragmatiche*. Le università che un tempo erano valutate da eventuali studenti in base alla direzione progettuale, vengono ora sempre più scelte in base al numero di guru del computer e in base a quanto siano all'avanguardia, e l'approccio di lasciarsi guidare dal computer e preferito rispetto al suo uso come strumento. Non vi sono dubbi che, malgrado la direzione preferita da qualsiasi preside, direttore o tutore, la vita dello studente e oggi molto facilitata dal computer sotto diversi aspetti. I tempi di configurazione sono lunghi e laboriosi nei progetti ortogonali, ma una volta stabilito un progetto, sezioni, prospettive, elevazioni, walkthrough e animazioni sono esponenzialmente più veloci e offrono una comprensione molto più chiara delle relazioni spaziali tridimensionali rispetto alle tecniche grafiche convenzionali.

La tecnologia della progettazione computerizzata ha anche aperto possibilità inattese per i docenti, come dimostra una lezione di storia, sviluppata da Robert Timme, preside delle School of Architecture presso l'Università della California del Sud. Come spiega Timme:

'Prima del computer, le idee ed i processi progettuali erano difficili da presentare. Combinando

vari programmi software si possono mostrare dei soggetti complessi in un formato visivo chiaro, descrittivo e animato. Questa lezione, Ordine e Progettazione, e una presentazione di un sistema di componenti architettonici tridimensionali e del modo fondamentale in cui essi possono essere ordinati per ottenere una definizione spaziale e una gerarchia. La presentazione utilizza edifici ben noti come esempi di varie strategie organizzative. Si muove attraverso un processo progettuale, illustrando lo sviluppo di un'idea semplice'.

II set di immagini iniziale descrive colonne che sono viste con diversa spaziatura. L'idea dello spazio intercolonnare era essenziale per la costruzione del tempio greco e romano. La presentazione di varie spaziature con elevazione semplice non rende l'idea dell'importanza di questo concetto. II mostrare le colonne obliquamente e in tre dimensioni illustra la vista da dove l'altare doveva trovar-si, come nel Partenone. Questa serie di immagini favorisce la discussione di questioni che vanno dalla straordinaria trasparenza ai rituali e alle sequenze professionali legate ai templi. Le immagini tridimensionali ed i diagrammi sono stati creati con Form-Z e Photoshop e presentati in PowerPoint.

Anche l'animazione e la visualizzazione del walkthrough rendono gli studenti più consapevoli delle gerarchie strutturali nei loro progetti; l'implacabile elevazione logica delle colonne e dei muri dimostra rigorosamente le implicazioni delle decisioni relative a progetti bi-dimensionali e risolve in parte il problema cronico di integrare la struttura nello studio progettuale. Le presentazioni non cartacee di PowerPoint sono in genere più organizzate e facilmente comprensibili, e danno risposte più convincenti".

I limiti del CAD

Sviluppato nella meta degli anni sessanta, fu disponibile per gli architetti nel 1970, ma nel decennio successivo, solo le grandi aziende poterono permettersi una sede guidata da un tecnico espressamente formato e del costo di 100.000 dollari, e occuparsi dell'acquisto esclusivo di hardware, software e contratti di manutenzione. L'avvento del personal computer nel 1982 ha aperto il mercato e ha reso disponibile software competitivi.

In realtà il CAD è uno strumento di transizione. Un ponte tra il disegno tecnico tradizionale e ciò che ancora non si sapeva bene sarebbe stato. Un ponte verso la modellazione 3d e quello che sarà.

Nonostante il CAD abbia continuato a dominare fino a tempi relativamente recenti il suo aspetto più importante e stata la produzione di documentazione, piuttosto che la progettazione.

Verso la metà degli anni novanta, una ricerca sul collegamento tra l'uso del CADD e la filosofia progettuale in America ha concluso che i sistemi computerizzati sono giunti giusto in tempo per risolvere i problemi progettuali diventati troppo complessi per essere trattati con i metodi tradizionali. E ancora che l'uso del computer per la progettazione non ha successo e come il disegno e la documentazione. Al momento si stima che meno del 14 per cento di esse utilizzava il CAD per la progettazione.

L'inadeguatezza del CAD ha portato Alberto Perez-Gomez ad esprimere una critica intuitiva sostenendo che esso rappresenti la continuità scientifica della razionalità modernista e leggibile nella forma costruita. Michael Benedict, che nel 1991 ha pubblicato *Cyberspace: Primi Passi nella Realtà Virtuale*, uno dei primi tentativi di comprendere il fenomeno del computer in architettura, ha espresso la sua impressione sostenendo che: "a causa del modo in cui funziona AutoCAD, posso sempre dire quando cammino in uno spazio progettato utilizzandolo. È costruito a strati di progetti in sezioni o modelli".

Dove sono finiti i maestri?

Il gap di competenza ha diverse implicazioni importanti, la prima delle quali è che la trasmissione della conoscenza pratica, la componente del mondo reale dell'istruzione di un giovane professionista che spesso manca nell'esperienza accademica, si e interrotta, facendo ricadere sul tirocinante la responsabilità di trovare le risposte alle domande relative ai

sistemi di costruzione, alle questioni procedurali e di dettaglio che un tempo venivano fornite da colleghi esperti. Continua James Steele: “negli Stati Uniti almeno, dove non è stata adottata la tradizione europea di un anno di studio trascorso a far esperienza in studio, questa trasmissione cambia fondamentalmente il tacito accordo che è esistito tra le scuole, dove l'enfasi viene posta sugli aspetti teorici della progettazione, e lo studio, dove ci si attende si debbano riempire le enormi lacune della conoscenza dei laureati. Nonostante le consistenti lamentele registrate nel corso delle ricerche dell'American Institute of Architects, circa il fatto che le scuole non preparano adeguatamente gli studenti ad essere operativi negli studi, questo tacito accordo e stato appoggiato forse perché coloro che hanno esperienza hanno sentito che fosse loro dovere sostenere e portare avanti un sistema che ha reso possibile la loro stessa crescita professionale. Ma attualmente, con i tirocinanti sempre più relegati all'informatica ripetitiva, isolati al di fuori del circuito educativo a causa della loro destrezza e abilita speciali che rappresenta l'equivalente cibernetico dell'essere assegnato ad un negozio di plastici, questo accordo deve cambiare. Occorre ispirare alternative che già esistono, come il Dublin Institute of Technology, dove il programma di studi include un anno obbligatorio presso una struttura parallela per I'apprendimento delle tecniche di costruzione, Le università americane hanno rifiutato questo tipo di istruzione ritenendo che essa le ridurrebbe allo stato di scuole commerciali, in quanto l'apprendimento delle tecniche di costruzione comprometterebbe in qualche modo I'idealismo teorico. L'esempio di Howard Roark ne *La Fonte Meravigliosa* di Ayn Rand tocca ancora un punto dolente: l'alternativa del lavoro duro nella cava dovrebbe essere presa in considerazione solo se tutte le altre possibilità di espressione creativa sono impedite.

In qualche punto del suo cammino, il principio Bauhaus di integrare gli ideali della teoria progettuale, che Roark rappresenta, con la produzione della quale egli e costretto a vivere, si biforca. II paragone gotico della cooperazione della comunità ispirata dal cielo, successivamente filtrata attraverso le sensibilità dell'Arts and Crafts trasformate poi in un'ideologia secolare del primo periodo moderno, si e perso una volta trapiantato negli Stati Uniti. L'immagine del modernismo, coincidendo con la rapida crescita dei media nel dopoguerra, e stata adottata facilmente ed evidentemente troppo superficialmente, mentre l'ideologia, soprattutto relativa al creare, e andata perduta. Con l'enfasi crescente sull'abilità digitale, e stato introdotto ancora un alto strato di astrazione tramite simulacri, con la possibilità che gli studenti si allontanino ancora maggiormente dalla realtà della costruzione”.

L'insegnamento ai tempi della rete

I maestri sono finiti nella rete.

È già da un po' che esistono sistemi di *e-learning* ma ultimamente si stanno raffinando sempre più. Con la diffusione dell'alta velocità ma più in generale con la crescita digitale che stiamo vivendo le possibilità di interagire in tempo reale sul web è oramai alla portata di tutti.

La parcellizzazione a cui assistiamo investe ovviamente anche l'insegnamento e nella stretta attuale e futura le risorse vanno ottimizzate. Scegliere oggi un corso di studi di tipo universitario potrebbe non essere la scelta più conveniente. Considerate le attuali competenze richieste dal mercato del lavoro il *gap* di cui si parlava sopra riguarda soprattutto l'inadeguatezza e la mancanza di insegnamenti adeguati. Mancanza sopperita da tutta una serie di corsi specifici e orientati al lavoro che rapidamente vanno affermandosi sul web.

Mi sembra evidente che il futuro potrebbe salvare solo le istituzioni più prestigiose lasciando in lenta e inesorabile agonia tutte le altre.

Altri scenari si intravedono all'orizzonte. Alcuni fantastici e stimolanti, altri angoscianti. Comunque diversi, molto diversi da quelli attuali.

Capitolo 4° - Scenari diversi

Il percorso si conclude con una serie di considerazioni su ipotetici scenari futuri ma anche e soprattutto con la concreta proposta di utilizzo di un sistema di realtà amplificata per gestire la ricostruzione in tempo reale di un'antica città come Pompei.

Idea e modello

Dalla metafisica alla modellazione solida. Il modello accompagna da sempre l'architetto nella prefigurazione progettuale, costituendo uno degli strumenti più antichi.

L'uso di rappresentare un edificio sotto forma di una sua copia plastica in minori dimensioni era molto diffuso in epoche passate, quando la tecnica delle rappresentazioni grafiche era meno progredita e l'esecuzione di un modello costituiva il mezzo migliore per comprendere l'opera.

Mirabili esempi di tali modelli risalgono al tardo Medioevo e soprattutto al Rinascimento, dove l'architetto, nelle botteghe associa la pratica al discorso. Proprio questa relazione tra l'idea e la sua immediata realizzazione in un modello rappresenta un momento di prefigurazione della realtà futura.

Attraverso la sensazione tattile il progettista si appropria fisicamente del sito, interiorizzandolo.

Tadao Ando usa il termine *Shintai* (corpo) per esprimere l'unione inseparabile di corpo e spirito evidenziando la sua formazione segnata da un profondo rapporto con la natura e il mestiere.

La questione dell'importanza di vedere il risultato della propria fatica è chiarito da Erich Fromm nel 1956 quando, individuando nella ricerca dell'unione la risposta dell'uomo al problema dell'esistenza umana, nell'opera *The Art Of Loving* scrive: "un terzo modo per raggiungere l'unione è l'attività creativa, sia quella dell'artista che dell'artigiano. In ogni tipo di attività creativa, colui che crea si fonde con la propria materia, che rappresenta il mondo che lo circonda… l'artefice e il suo oggetto diventano un'unica cosa: l'uomo si unisce col mondo nel processo di creazione. Questo, tuttavia, vale solo per il lavoro produttivo, per il lavoro nel quale io progetto, produco, vedo il risultato della mia fatica".

Innovativa in Jean Prouvè era la capacità di mettere alla prova le potenzialità intuitive dei suoi allievi dell'Ecole des Arts et Mètiers, presentandogli problemi da risolvere con una manipolazione diretta delle mani, considerate importanti quanto il lavoro della mente.

Nel lavoro del movimento architettonico *De Stijl* la realizzazione di modelli rappresenta l'attività fondatrice del processo progettuale. Rietveld, nella casa Schröder, arriva con un primo plastico a maturare la soluzione finale, così da poter passare ad un secondo modello infine a un terzo 1:25.

Nell'Opera House, i primi volumi, sono allineati su di un'asse principale secondo un criterio funzionale. La forma del lotto induce a una manipolazione di rientro del modello attraverso la segmentazione dell'asse principale. Così facendo l'architetto non rinuncia all'idea principale del progetto.

Il metodo di Frank Gehry, segue un iter in cui il modello costituisce il primo atto. Un modello viene realizzato a mò di scultura. Gli spigoli di questo primo modello vengono toccati da una penna ottica diventando i vettori di un sistema reticolare, íncipit di una serie di modelli.

Per la copertura della Sidney Opera House, Jørn Utzon propose una soluzione nella quale le superfici di tutti i gusci venivano estratte da una stessa sfera. Fu realizzata una calotta di legno tagliata a spicchi. I vuoti derivati dall'estrazione dei gusci suggeriscono la tecnica realizzativa.

Va chiarito l'equivoco sull'interscambiabilità del ruolo della modellistica con

l'elaborazione di modelli 3D. Queste due attività sono oggi insostituibili con ruoli e funzioni diverse e vanno utilizzate in maniera complementare.

La ricerca *Passive Downdraught Evaporative Cooling* si prefigge lo scopo di ridurre i costi di gestione e migliorare il comfort termoigrometrico riprendendo il principio della torre che capta il vento dominante. La verifica è stata condotta in galleria del vento su modelli in plexiglass in scala 1:20 e realizzando modellini speciali.

L' iter progettuale condotto da Renzo Piano nell'Auditorium di Roma coinvolge compositori, direttori d'orchestra e consulenti d'eccezione. Vennero realizzati in grande scala superfici riflettenti per grafici laser della riflessione acustica. Da questi grafici vennero creati modelli matematici al computer, attraverso i quali ne derivò la forma. Da questa prova derivò un plastico sul quale furono effettuate prove acustiche, i risultati poi furono corretti per conoscere esattamente le prestazioni della sala a dimensioni reale.

Nel progetto per la sistemazione della Piazza España ad Alcoy Calatrava realizza un modello in legno della copertura mobile della vasca d'acqua prevista al temine della piazza per verificare tecnicamente il movimento meccanico che descrive un'onda e per precisare il fondamentale effetto dinamico.

Nella Galleria d'Arte Urbana a Dundee David Chipperfield realizza un modello/sezione di notevoli dimensioni, che viene orientato all'aperto nel luogo dove dovrebbe sorgere l'edificio, per verificare come la luce naturale viene regolata dall'orientamento degli elementi in copertura.

Il modello rappresenta il veicolo con cui il progettista comunica la sua idea: in essa riassume tutti i contenuti della sua ricerca. La scelta del tipo di modello che si vuole realizzare sono in relazione a ciò che si vuole comunicare, quindi spetta di priorità al progettista. La scelta del materiale è determinata da considerazioni di ordine espressivo; se accosto due materiali, l'uno con finitura rugosa e l'altro con finitura liscia, non necessariamente definisco la natura fisica dei due materiali ma ne anticipo le caratteristiche, il primo sicuramente più naturale e scabro del secondo.

Il modello di architettura è tradizionalmente monomateriale e monocromo. Sono sconsigliati l'uso di colori e materiali eterogenei; l'obiettivo non è di rappresentare la realtà in miniatura perché ciò potrebbe distrarre il progettista nell'analisi delle caratteristiche determinabili del sito.

Il grande modello presente a Lugano, della chiesa di San Carlo alle Quattro Fontane a Roma riproduce, in scala al vero, la metà esatta della chiesa. L'intenzione è di interpretare fedelmente lo spazio, attraverso il modello si evidenzia la natura di sottrazione, di 'scavar la forma nella materia'. Il modello è realizzato attraverso la sovrapposizione di tavole di abete rosso dello spessore di 4,5 centimetri, restituendo la forma architettonica esattamente come le curve di livello restituiscono l'orografia del terreno.

In considerazione dei tanti e pressanti requisiti di qualità di cui un edificio necessita, la figura dell'artigiano sembra rivivere nell'architetto, il quale può concepire solo un lavoro di èquipe. I più importanti studi di architettura sono muniti di uno spazio/falegnameria. Il modello sembra essere uno strumento indispensabile ed insostituibile dando un notevole contributo al processo progettuale.

In ultima analisi, la modellistica dovrebbe essere intesa come gioco paziente di attesa e di decantazione, come processo circolare per capire e sperimentare, allo scopo di perseguire l'obiettivo prefissato.

Architettura virtuale

Nuove tecnologie della comunicazione stanno trasformando il nostro approccio alla realtà quotidiana. Da un lato, gli strumenti tradizionali della rappresentazione, il controllo e la costruzione fisica di una architettura proposta è molto migliorata con l'uso delle nuove tecnologie digitali. A sua volta, un nuovo mondo si apre ai nostri occhi con la nascita dell'architettura virtuale, un'architettura che non è fatta di palazzi reali, ma destinata a esistere solo in digitale.

L'uso del computer e la moderna tecnologia digitale ha permesso la costruzione opere sorprendenti. Pensate al Centro d'Arte Contemporanea Georges Pompidou, costruito a Parigi negli anni '70 da Renzo Piano e Richard Rogers, che oltre ad essere stato il manifesto dell'architettura *high tech*, è uno dei primi progetti disegnati integralmente al computer. E ancora l'ultima produzione di Frank Gehry, la cui complicata costruzione è stata resa possibile dall'uso di programmi informatici come il Catia per modellazione, resa e calcoli strutturali. Grazie al Vrml (Virtual reality modelling language) il gruppo olandese NOX a realizzato a Rotterdam un esempio di *fluid architecture,* il Padiglione dell'acqua.

Intervistata Odile Fillion: "il Padiglione è formato da due edifici divisi in due parti. Gli edifici sono divertenti perché non assomigliano a niente. Sono edifici liquidi senza porte, senza entrate, senza finestre e senza facciate. Non si sa come avvicinarlo, sembra un animale. Una volta all'interno, il nostro spostamento lo modellerà e lo modificherà sia in modo sonoro che in modo visivo. Vengono attivate proiezioni e il nostro spostamento induce a delle modifiche delle nostre percezioni. Sono attivati dei flussi di acqua che obbligano a cambiamenti di percorso".

Nella storia c'è sempre stata una architettura virtuale, destinata a non diventare realtà. Pensate alle meravigliose prospettive della città ideale del Rinascimento, alle ricostruzioni di fantastiche architetture del passato, ai disegni di Boullée, di Ledoux e degli utopisti del XIX secolo francese, alle visioni futuristiche degli anni '60 di Archigram.

Proiezioni immaginarie di futuri possibili trovano il loro nuovo mondo nel web, uno spazio virtuale per l'interazione e la comunicazione tra persone, dove il nostro avatar può camminare, muoversi.

Marcos Novak, architetto americano padre di transarchitetture, verso la fine degli anni '70 ha iniziato a sperimentare le prime forme di immagini di sintesi. Inoltre, l'opera di Palladio era un personaggio virtuale. Voglio dire, gli architetti hanno sempre avuto a che fare con il mondo virtuale in un certo senso. Ora hanno a che fare con la tecnologia virtuale.

L'architettura del virtuale realizza dunque una sorta di estensione del nostro mondo reale offrendo nuove possibilità di esperienza e di interazione. Oggi le sperimentazioni virtuali in architettura riguardano le giovani generazioni. Uno di questi giovani architetti che ha studiato e utilizza il virtuale è Ammar Eloueini: "la virtualità è un'estensione di tutto ciò che è realtà, come sostiene Gil Deleuse. C'è dunque un ambiente virtuale nel quale gli architetti possono lavorare, e si potrebbe aggiungere che tutti gli architetti sono in fondo virtuali, poiché non fanno altro che concepire le loro costruzioni, e non le costruiscono. Per quanto mi riguarda, il potenziale più interessante che la virtualità può offrire sta nell'ambiente virtuale, nel quale le nozioni classiche di geometria e di peso che non è possibile realizzare nella realtà, si possono simulare grazie appunto alla virtualità. E questo è un fatto che arricchisce enormemente l'architettura. Penso che oggi siamo ancora alle premesse di quanto potrebbe offrire questo spazio che viene detto virtuale e si muovono appena i primi passi in questo spazio: sono come i primi passi sulla luna di trent'anni fa. Oggi ci troviamo molto a disagio nella dimensione virtuale, non abbiamo ancora raggiunto un perfetto equilibrio, lo stiamo ancora cercando e credo ci siano ancora numerosissimi aspetti da sviluppare in relazione allo spazio virtuale. In un certo senso occorre che esso venga architettato: bisogna che architetti, artisti, filosofi e sociologi lavorino a questo spazio per circoscriverlo, per poterlo sfruttare. Questo spazio esiste solo che finora non si disponeva di tecnologie che ci permettessero di sfruttarlo: oggi si cominciano a intravedere queste tecnologie che si svilupperanno ancora".

L'architetto del futuro avrà dunque una personalità "bilingue", "ibrido", in grado di pensare e costruire lo spazio nel mondo reale e nel mondo virtuale. Il nostro futuro allora sarà forse ispirato da architetture interattive, entità dinamiche in movimento, misteriosi e affascinanti forme organiche.

Realtà virtuale *Intervista ad Ammar Eloueini, MediaMente*	
Domanda 1 Potrebbe dare una definizione dell'architettura virtuale?	**Risposta** E' difficile definire l'architettura virtuale. Penso non ci sia una sola definizione valida. Ce ne sono molte e si possono considerare diverse cose come architettura virtuale. La virtualità è un'estensione di tutto ciò che è realtà, come sostiene Gilles Deleuze. C'è dunque un

ambiente virtuale nel quale gli architetti possono lavorare, e si potrebbe aggiungere che tutti gli architetti sono in fondo "virtuali", poiché non fanno altro che concepire le loro costruzioni e non le costruiscono.

Per quanto mi riguarda, il potenziale più interessante che la virtualità può offrire sta nell'ambiente virtuale, ambiente nel quale le nozioni classiche di geometria o di peso non valgono necessariamente, e certe forze, che non è possibile realizzare nella realtà, si possono simulare grazie appunto alla virtualità. E questo è un fatto che arricchisce enormemente l'architettura.

Al di fuori di quest'ambito, di questo potenziale, si può parlare di un'architettura per uno spazio virtuale. Quello virtuale è infatti uno spazio che si aggiunge allo spazio reale nel quale si vive, o si è abituati a vivere. Progettare architettura per questo spazio fa parte, deve far parte ormai dell'attività degli architetti e del loro mestiere. Per quel che mi concerne, questa non è la mia occupazione primaria, ma penso che ci siano molte persone interessate a questo aspetto della virtualità nell'architettura.

Domanda 2

Generalmente, quando si pensa all'architettura si pensa a qualcosa di costruito, e sembra ci sia una contraddizione nel concetto di architettura virtuale?

Risposta

Spesso, e a torto, si mettono a confronto reale e virtuale, o attuale e virtuale. Il virtuale è l'estensione del reale, e penso che non ci sia un'architettura del virtuale così come ne esiste una del reale.

Si può creare architettura per uno spazio virtuale, ma un'architettura virtuale vera e propria non esiste.

Tutta l'architettura che non viene realizzata può essere considerata, a un dato momento, come virtuale, ma questa opposizione tra architettura virtuale e reale mi sembra basata su falsi presupposti.

Domanda 3

E qual è, secondo lei, il limite dell'architettura virtuale?

Risposta

Penso che oggi siamo ancora alle premesse di quanto potrebbe offrire questo spazio che viene detto virtuale, e si muovono appena i primi passi in questo spazio: sono come i primi passi sulla luna di trenta anni fa. Oggi ci troviamo molto a disagio nella dimensione virtuale, non abbiamo ancora raggiunto un perfetto equilibrio, lo stiamo ancora cercando, e credo ci siano ancora numerosissimi aspetti da sviluppare in relazione allo spazio virtuale. In un certo senso, occorre che esso venga architettato: bisogna che architetti, artisti, filosofi e sociologi lavorino a questo spazio per circoscriverlo, per poterlo sfruttare. Questo spazio esiste, solo che finora non si disponeva di tecnologie che ci permettessero di sfruttarlo: oggi si cominciano a intravedere queste tecnologie, e si svilupperanno ancora. Gli architetti dovrebbero interessarsi molto da vicino a ciò che accade in questo spazio.

Domanda 4

Che rapporto c'è fra le nuove tecnologie e l'architettura?

Risposta

La maniera più semplice di vedere le nuove tecnologie applicate all'architettura è nel modo di incrementare la produzione, vale a dire di lavorare più velocemente. Oggi gli architetti nel loro lavoro possono produrre una grande quantità di progetti, fino a cinquantamila progetti per un palazzo o per un aeroporto, e con grande facilità grazie a queste nuove tecnologie. Questo è l'aspetto più immediato: la rappresentazione dell'idea diventa estremamente semplice, e la produzione può lavorare molto più facilmente e rapidamente con le nuove tecnologie. Ma la cosa più importante, e che più mi interessa in relazione alle nuove tecnologie è l'impatto sulla produzione stessa dell'architettura, sul processo del design. In altri termini, l'architetto ha, o aveva, l'abitudine di lavorare con la carta, le matite, i modellini: oggi il suo ambiente di lavoro è cambiato, e gli strumenti con cui l'architetto svolge il proprio compito diventano estremamente rilevanti. L'architetto non costruisce, contrariamente all'artista che dà vita a un'opera, un testo o un'installazione: l'architetto progetta sempre un lavoro che dovrà essere realizzato da altri, di qui l'importanza degli strumenti. Lo si può vedere risalendo indietro nel tempo: gli strumenti con cui gli architetti hanno lavorato in passato hanno influenzato enormemente la produzione dell'architettura. Oggi è possibile immaginare che le nuove tecnologie, o le cosiddette nuove tecnologie, potranno influenzare moltissimo il modo di lavorare dell'architetto. Gli architetti hanno l'abitudine di lavorare con elementi statici: la carta, i modellini. Oggi con le nuove tecnologie si può cominciare a introdurre le nozioni di tempo, di movimento, di flusso, fare simulazioni che permettono all'architetto di operare diversamente, di immaginare lo spazio, di rapportarsi ad esso in un altro modo, di affrontare l'architettura in un altro modo. C'è un terzo punto che riguarda le nuove tecnologie, ed è la produzione stessa degli edifici, anch'essa molto importante. Oggi esistono macchine a comandi numerici, ossia, a partire da elementi disegnati al computer si possono ottenere direttamente parti di costruzione prodotte a un costo identico a quello di una produzione su larga scala. Perciò anche la nozione di produzione di massa si trasforma in una standardizzazione; particolari ed elementi variabili si possono realizzare sempre allo stesso costo, e questo è un aspetto importantissimo che oggi si presenta grazie alle nuove tecnologie.

Domanda 5

Pensi che gli architetti siano pronti all'utilizzo di queste nuove tecnologie, oppure sono in ritardo?

Risposta

Non penso che ci sia un ritardo, ma piuttosto un sistema che si autoconserva. Le nuove tecnologie sono arrivate molto rapidamente, e in brevissimo tempo si è realizzato un mutamento radicale nell'ambiente di lavoro dell'architetto. Questo mutamento disturba enormemente gli architetti abituati a metodi tradizionali. In genere un architetto elabora un suo modo di procedere nel corso degli anni, ci lavora, lo sviluppa ulteriormente, ma è molto difficile ottenere che egli cambi dall'oggi al domani il suo sistema di lavoro, o che si trovi subito a suo agio in un ambiente diverso: questo crea enormi scompensi, e perciò esiste una certa resistenza degli architetti a trasformare il loro ambiente di lavoro e i loro metodi, a parte quei pochi che ci riescono. Uno degli esempi più pertinenti è quello di Frank Gehry, che ha saputo continuare a lavorare secondo i suoi sistemi, ma che nel fare ciò ha introdotto nuove tecnologie. In altre parole, Frank Gehry ha l'abitudine di fare ampio uso di modellini: i suoi metodi risalgono a venti, trent'anni fa, ed è riuscito a seguirli nel suo lavoro, soltanto che oggi ricorre anche alle nuove tecnologie, scannerizza i suoi modellini a tre dimensioni, li elabora al computer per poi riprodurli con macchine a comandi numerici. Così ha progettato, per esempio, il Museo di Bilbao, un esempio illuminante e importantissimo per l'architettura di oggi e il suo legame con le nuove tecnologie.

Domanda 6

Come immagina una città del futuro?

Risposta

C'è un progetto a cui lavoro da due anni, che riguarda un quartiere di Tokyo. Credo che una delle prime cose che si potrebbero fare è cominciare a rimettere in discussione le nozioni dell'urbanistica classica, che prevedevano composizioni assiali o una sovrapposizione di elementi nella città. Oggi possiamo cominciare a lavorare su nuove basi. La città è una realtà estremamente dinamica, è fatta di flussi, di movimento, e anche se l'architettura è fondamentalmente statica la città è piena di vita, respira, e così si può cominciare a immaginarla, a lavorare alle città con tutto ciò che esse hanno di dinamico. Oggi è possibile simulare questi flussi, questi movimenti, per non dire che si procederà a realizzare simulazioni scientifiche di intere città, e anche se in certi casi la cosa è interessante di per sé, in effetti queste simulazioni possono contribuire enormemente all'immaginazione degli architetti in rapporto alle città, alla loro gestione, alla correlazione fra i vari programmi,

eccetera. Si possono ideare vari scenari nei quali le città non sono come le concepiamo oggi a causa dei nostri strumenti tradizionali, ma appaiono molto diverse da come le potevamo immaginare in passato.

Domanda 7

Come cambierà il rapporto fra uomini ed edifici in futuro nelle città digitali? Come pensa che cambierà il rapporto fra gli uomini e i nuovi stili architettonici?

Risposta

A quel livello si possono immaginare diverse cose, ad esempio che la tecnologia si applichi direttamente agli edifici: questo è un aspetto che già si comincia a sviluppare, ossia, si vedono già componenti interattive, facciate, elementi mobili e altre cose del genere. Ma soprattutto credo che la città cambi completamente perché le viene aggiunto uno spazio che non è il suo spazio reale ma quello virtuale, e questo mi sembra di grande importanza. Ciò modifica profondamente il rapporto fra lo spazio e gli individui. Oggi le persone entrano in gioco nel momento in cui si collegano in diretta via Internet o per altre vie a tre città contemporaneamente, e dialogano in tempo reale. Questo comporta una enorme trasformazione del nostro rapporto con lo spazio, ed è più questo rapporto a modificarsi che non lo spazio in se stesso. Si può immaginare o fantasticare di uno spazio più interattivo, uno spazio in movimento, ma in definitiva credo che la più radicale trasformazione in rapporto al nostro spazio sia questa compressione dello spazio che si viene a ottenere. Lo spazio è compresso nella dimensione virtuale, e questo modifica radicalmente il nostro rapporto con la città e il suo spazio.

Domanda 8

Lei ha studiato negli Stati Uniti. Per una persona come lei, nata in Europa, quali stimoli sono venuti da quell'esperienza in America? Aveva deciso di studiare in quel paese per qualche ragione particolare?

Risposta

Anzitutto sono nato sulle rive del Mediterraneo, e non in Europa, bensì in Libano. Ho studiato architettura a Parigi, e in quel periodo, erano gli anni Ottanta, mi accorsi che in Europa non si affrontavano problematiche veramente innovative in rapporto all'architettura. Per questo mi interessai molto di più a quanti esercitavano e insegnavano negli Stati Uniti, piuttosto che agli europei; in altre parole, l'architettura in Europa, e specialmente in Francia, mi annoiava terribilmente, e perciò avevo sempre il desiderio di partire, di entrare finalmente in contatto con persone interessanti. Negli Stati Uniti lavoravano molte persone che stavano elaborando teorie che mi apparivano molto più rilevanti che non le costruzioni semplici e le facili commissioni così frequenti in Francia e in Europa negli anni Ottanta.

Domanda 9

Oggi la figura dell'architetto, soprattutto in Europa, sembra essere in grave crisi: quali pensa dovrebbero essere le strategie di un giovane architetto nell'Europa di oggi?

Risposta

Sì, sono giovane, e mi sarebbe piaciuto fare molto di più di quanto mi è possibile al momento, ma purtroppo spesso manca il tempo, ventiquattr'ore al giorno sono un periodo alquanto limitato. Credo comunque che la cosa più importante, soprattutto nel momento presente, all'alba della nuova era digitale, sia fare attenzione. Al giorno d'oggi produrre belle immagini è un'impresa alla portata di tutti, non c'è bisogno di essere architetti a tal fine, e in un certo senso qui sta il pericolo di quest'era digitale, nella quale l'architettura potrebbe anche colare a picco. Ciò che reputo importante, e che cerco io stesso di fare dedicandovi una parte consistente del mio lavoro, è l'essere in grado, ancor prima di fare l'architetto e costruire palazzi, di costruire un pensiero. Oggi per diventare architetti occorre saper costruire il proprio pensiero, fare qualcosa che non sia solo l'immagine, o la produzione diretta. Gli strumenti attuali richiedono un'attenzione molto speciale per il modo in cui si procede nel nostro lavoro. C'è una parte teorica che occorre sviluppare e che è di fondamentale importanza: in Europa, però, negli ultimi anni, e mi riferisco soprattutto alla Francia, ci si è sostanzialmente disinteressati della teoria dell'architettura. Ora, non dico che bisogna diventare dei teorici puri, sebbene vi siano molti che guardano con interesse a una teoria dell'architettura, ma un architetto deve essere capace, a sua volta, di costruire il proprio pensiero, di esprimerlo con estrema chiarezza prima di passare alla fase della costruzione degli edifici che esistono nella realtà.

Domanda 10

Quali sono le difficoltà che incontri nel tuo mestiere?

Risposta

Credo che le maggiori difficoltà provengano dalla resistenza di certi ambienti. Si sente continuamente parlare di una crisi generazionale, che in sé è un fatto normale e governabile, ma che nella fattispecie viene esacerbata dall'introduzione delle nuove tecnologie, poiché in effetti fra le generazioni viene a crearsi una spaccatura profonda, e di natura diversa rispetto a quanto poteva avvenire in passato. Da qui potrebbero provenire le maggiori resistenze, ovvero da una mancata comprensione di quella che è la nostra ricerca, del nostro lavoro, che viene preso per l'appunto per una forma di architettura virtuale, ossia priva di qualsiasi legame con la realtà. Questo, a mio avviso, è l'errore più grave, al quale si deve sempre fare attenzione per evitare di sprofondare in ciò che, a torto, viene chiamata architettura virtuale, ossia qualcosa che sta al di fuori dell'architettura. Ma l'architettura virtuale è nella realtà, deve esistere nella realtà.

Domanda 11

In che modo questo sarà possibile?

Risposta

Come dicevo poc'anzi, l'architettura virtuale non è in opposizione con la realtà. La virtualità, lo spazio virtuale è un'estensione dello spazio reale, soltanto che cambia l'ambiente di lavoro dell'architettura, e anche lo spazio stesso si trasforma a causa di questa estensione. Per fare un esempio: quando si guarda si ha il cosiddetto spettro visivo che ci permette di vedere un campo limitato, fra raggi infrarossi e ultravioletti. In definitiva, mi piace considerare lo spazio virtuale come una protesi che, una volta applicata, ci permette di vedere ultravioletti, infrarossi, e lo spazio nel quale si vive. In fondo i computer di oggi non sono che delle protesi che ci consentono di penetrare in questo spazio virtuale, che viene ad aggiungersi ed è un'estensione dello spazio reale, e che certamente non si trova in opposizione o in contraddizione con esso.

Architettura sperimentale

Ma cosa si intende per architettura sperimentale? Secondo Betsky è l'architettura intesa come "rappresentazione, riutilizzo e riallocazione di immagini, materiali e persino idee". Non produzione di nuove forme o nuovi edifici, bensì "raccolta di ciò che già esiste e la sua trasformazioni in nuove strutture e nuove relazioni in grado di rivelare modi differenti di vita, uso o esperienza". È una architettura che ha superato le restrizioni della prassi edilizia e ha dato vita a controimmagini di ciò che l'architettura abitualmente produce.

Gandolfi vede nella sperimentazione una preziosa opportunità per individuare strumenti in grado di aprire lo spettro di azione dell'architettura: "In un momento di forte

emergenza urbana. Basti pensare ai dati di UN Habitat per il quali entro il 2030 un abitante su tre nel mondo vivrà in una baraccopoli e al progressivo emergere di casi di inequità sociale. L'architetto sembra perdere progressivamente importanza, schiacciato tra lo *star system* delle architetture-spettacolo e le richieste sempre più imperiose di un capitalismo globale che disegna la crescita urbana esclusivamente in base a calcoli economici. È in questo clima che la sperimentazione perde interesse per la definizione di forme sempre più sofisticate e si confronta apertamente con l'osservazione dello stato attuale delle nostre realtà urbane".

L'architettura radicale e la critica
Carlotta Darò, ARCH'IT

È Germano Celant, all'inizio degli anni '70, a introdurre in Italia il fortunato termine di "architettura radicale". I protagonisti di quello che a posteriori diventerà un vero e proprio movimento sono alcuni studenti della Facoltà di Architettura di Firenze, in particolare del corso di Leonardo Savioli dedicato ai "Pipers". Nel novembre del 1966 i giovani architetti fiorentini organizzano la prima mostra di "Superarchitettura" in una piccola cantina della città di Pistoia.

Tra la prima e la seconda mostra di "Superarchitettura" si formano i due rinomati gruppi, *Archizoom* e *Superstudio*. Le riviste *Domus*, *Casabella* e in seguito *In* svolgono il ruolo di portavoce della neoavanguardia, mentre altri sporadici articoli escono su diverse pubblicazioni come *Marcatré*, *Controspazio*, *Interni*, *Modo*, *Progettare in più*, *Abitare*, *Pianeta Fresco*, *Flash art*, *Nac* e su riviste non di settore e di larga diffusione come *Panorama* e *l'Espresso*.

A parlare di questo nuovo e prezioso fenomeno che tenta una fuga disperata dall'accademismo imperante, sono gli stessi architetti o alcuni importanti critici del mondo dell'arte. Grazie al metodico lavoro teorico di Andrea Branzi, quello editoriale di Alessandro Mendini, Franco Raggi e altri, possediamo oggi una chiara e appassionante lettura dei protagonisti di questo singolare fenomeno. D'altro canto, all'esterno del loro stesso circuito, Achille Bonito Oliva, Gillo Dorfles e il già citato Germano Celant contribuiscono ad un riconoscimento esplicito a livello concettuale del movimento radicale. Nel campo della critica architettonica, l'esempio di seguito riportato mette in chiaro il valore conferito a questo movimento da due critici come Manfredo Tafuri e Francesco Dal Co: "La liberazione nell'ironia ripercorre le utopie delle avanguardie storiche: i progetti di deserti occupati da superoggetti metafisici –come esercitazioni autopropagandistiche dei gruppi italiani Archizoom e Superstudio– consumano fino alla nausea gli aneliti tardoromantici della tautiana *Aufloesung der staedte*". Il livello di apprezzamento dalla parte della critica italiana è di questa portata nei casi in cui, fortuna vuole, il movimento radicale viene per lo meno preso in considerazione.

Questo movimento non è, all'epoca, studiato e valutato in una prospettiva internazionale più ampia. Il fenomeno radicale italiano assume il suo vero valore, infatti, nell'ambito di una più complessa storia di ribellione disciplinare che avviene in contemporanea in diverse parti del mondo. Forse a causa dei suoi contenuti politici e ideologici, o forse a causa della sua pericolosa posizione antiaccademica, i radicali italiani non riescono, in casa, ad abbattere i pregiudizi di una critica ostile. In una ricostruzione prospettica fatta per ipotesi ci sarebbe da chiedersi come sarebbe andato il seguito di questa vicenda se qualche critico contemporaneo di prestigio avesse posto, allora, l'attenzione sul fenomeno. Chissà se oggi il ritorno all'ordine, ovvero all'accademia, dei radicali italiani avrebbe avuto un'evoluzione diversa o più coerente?

Che la gloria dovrà giungere dall'estero si intuisce già nel 1972, quando il MoMA di New York chiama numerosi radicali ad esporre alla mostra di controdesign *Italy: the New domestic Landscape*. Nel 1974 e 1977, escono in Italia due importanti pubblicazioni che rimettono ordine alla successione degli eventi del movimento ormai finito. La prima è la pubblicazione della tesi di Bruno Orlandoni e Paola Navone (con l'introduzione di Andrea Branzi) *Architettura radicale* e la seconda è *Dalla città al cucchiaio*, ancora di Bruno Orlandoni con Giorgio Vallino.

Il tempo passa e i vari protagonisti radicali intraprendono strade diverse che li portano a superare le loro posizioni nichiliste e ad affrontare le scelte o i compromessi della realtà professionale. All'inizio degli anni '90 si apre, in Francia, un nuovo orizzonte di ricerca storica e critica riguardante gli albori dei giovani studenti fiorentini. Dominique Rouillard intraprende un lavoro di ricostruzione e, finalmente, d'inquadramento storico dell'avanguardia italiana. Mettendo in evidenza l'influenza di *Superstudio* e *Archizoom*, rispettivamente sul lavoro dei giovani Rem Koolhaas e Bernarnd Tschumi, Rouillard introduce in Francia un dibattito critico, fino ad allora assente nella patria italiana, che mette in luce il vero valore concettuale e progettuale, e non ideologico, dei radicali italiani.

Questa operazione permette di ricostruire il fermento internazionale dell'epoca e le influenze reciproche in un'ottica che esce dal piccolo e combattuto contesto italiano. Inizia qui la consacrazione del valore storico dei radicali italiani con il recupero da parte del Centro Georges Pompidou, nella persona del conservatore Alain Guiheux, di numerose opere salvate dalle cantine polverose degli architetti. Il lavoro dei radicali italiani diventa progressivamente oggetto di mostre e manifestazioni di varia natura fatte in territorio francese, tra le quali va citata l'attività del FRAC Centre di Orléans iniziata dall'allora direttore Frédéric Migayrou.

Alla Biennale di Venezia del 1996, diretta da Hans Hollein, si assiste finalmente, in Italia, ad una retrospettiva internazionale sul fenomeno radicale operata da Gianni Pettena per il padiglione italiano. Segue, nel 1999, curata anch'essa da Pettena, la mostra al Palazzo Fabroni di Pistoia *Archipelago. Architettura sperimentale 1959-99*, che tende a voler attualizzare il movimento radicale avvicinandolo ad alcuni esempi di architettura contemporanea formalmente audaci o sperimentali. Luigi Prestinenza Puglisi dichiarerà, nel libro *This is tomorrow* che l'architettura radicale ha vinto e che la prova di tale vittoria risiede in un'opera come il Guggenheim di Bilbao di Gehry che stravolge i canoni tradizionali della progettazione. Ma il valore dell'architettura radicale è un valore negativo le cui tracce non si possono ritrovare in una "maniera" costruttiva. Gli architetti radicali si sono battuti, e forse sacrificati, in nome di una rivolta che mirava alla ridefinizione della disciplina intera e di un'ingenua battaglia contro il potere accademico. Purtroppo l'architettura radicale ha perso perché, al di là di alcune tracce significative nell'evoluzione storica di questa singolare avanguardia, non è riuscita a raggiungere l'utopica volontà di rottura e di rigenerazione della disciplina. Colpa del vuoto critico che ha accompagnato all'epoca questo movimento, o colpa piuttosto di una posizione che si poneva sin dal principio come eroicamente perdente, l'architettura radicale oggi lascia un importante patrimonio storico, giustamente messo in valore dall'intelligente recupero francese.

Esempio, per una volta italiano, di una posizione coraggiosa e "radicale", questa avanguardia ci lascia il merito di aver tentato o suggerito l'evasione verso altri orizzonti disciplinari, lontani dalla "regola" architettonica. L'architettura radicale, in quanto avanguardia destabilizzatrice, non poteva e non doveva vincere. "Fine ultimo dell'architettura è l'eliminazione dell'architettura stessa".

Realtà amplificata

La realtà amplificata (*augmented reality*) consiste nell'utilizzo di strumentazioni e tecnologie volte ad aumentare il nostro modo di percepire la vita reale, al contrario della realtà virtuale finalizzata a ricreare in tutto e per tutto un ambiente immaginario.

Se fino ad oggi la realtà amplificata è stata oggetto di molti studi e ricerche ma sempre in ambito accademico/sperimentale, ora le cose possono cambiare radicalmente, grazie all'avvento e alla diffusione di nuove strumentazioni necessarie all'utilizzo di programmi in grado di arricchire la nostra percezione della realtà.

Bionic Eye è la prima applicazione che offre una dimostrazione pratica di amplificazione della realtà: consente infatti di visualizzare a 360 gradi nell'ambiente reale in cui ci si sta muovendo, in 3D, tutti i punti di interesse presenti, come ristoranti, locali, bar o luoghi turistici. Più precisamente l'applicazione mostra i *POI* in sovrapposizione all'area che si sta riprendendo in tempo reale con la videocamera.

Realtà amplificata
Antonio Leonardi, MediaMente

Alla Boeing di Seattle, la fabbrica dove si costruiscono i Jumbo Jet, c'è un gruppetto di operai che da qualche mese lavora in modo un po' particolare.

All'inizio del loro turno, oltre alla vecchia tuta, questi uomini indossano un casco con una microtelecamera, uno speciale visore e tutta una serie di circuiti elettronici; si agganciano alla cintura un mini-computer portatile; e addobbati come chi gioca con la realtà virtuale cominciano ad armeggiare attorno agli aeroplani.

Ma gli operai non stanno affatto giocando. Anzi, presto anche molti dei loro colleghi potrebbero indossare caschi simili. Infatti sono, diciamo così, le cavie di un esperimento che potrebbe permettere di risparmiare molto tempo, e molti dollari, nella costruzione di ogni aereo. Vediamo come.

La parola magica è augmented reality, cioè realtà amplificata. La realtà amplificata è una parente stretta della realtà virtuale. Ma in questo caso non si tratta di costruire con il computer un ambiente immaginario o di ricrearne uno diverso da quello in cui siamo.

Si tratta piuttosto di amplificare la realtà reale. O meglio, di amplificare i nostri mezzi di percepirla. Insomma, i nostri occhi, le nostre orecchie, le nostre mani e il nostro cervello, potrebbero essere aiutati da alcuni dispositivi elettronici a svolgere meglio il loro compito.

Ma torniamo alla Boeing. Ogni aereo è percorso da decine di chilometri di fili elettrici. Una ragnatela di cavi che corre in ogni angolo del velivolo e che ne costituisce il "sistema nervoso".

E ogni cavo deve essere connesso esattamente alla sua presa perché anche un piccolo errore potrebbe essere molto grave. Inoltre ciascun aereo è leggermente diverso dall'altro, a seconda delle esigenze del cliente.

Insomma, cablare un aereo è un lavoro certosino: migliaia di fili devono trovare le loro connessioni e gli operai sono costretti a interrompere il lavoro molto spesso per osservare gli schemi di costruzione. E' un lavoro lungo e costoso.

Ed ecco dove interviene la realtà amplificata. La telecamera sul casco inquadra dei punti di riferimento colorati dipinti sui pannelli da cablare e li trasmette via radio a un computer centrale.

In base ai punti di riferimento, il computer capisce in quale punto dell'aereo sta lavorando l'operaio e invia tutte le informazioni registrate nella sua memoria.

Gli schemi elettrici e le istruzioni dettagliate su come connettere le prese appaiono così sul visore del lavoratore, e sono continuamente aggiornati in base alla zona che egli sta osservando e inquadrando con la microcamera.

Per ora quello della Boeing è un esperimento limitato. Ma sembra che grazie alla realtà amplificata completare l'impianto elettrico di un Jumbo sia dal 20 al 50% più veloce. E quindi più economico.

Se il casco per la realtà amplificata è arrivato agli operai, in campo aeronautico i primi a usarlo sono stati però i piloti. Infatti sono stati sperimentati dispositivi che proiettano informazioni e parametri di volo direttamente sulla visiera del casco, senza che il pilota debba distrarsi per osservare gli strumenti.

Ma un giorno o l'altro, chiunque di noi potrebbe portarsi in giro un dispositivo simile a quello della Boeing. Ci stanno lavorando al Massachusetts Institute of Technology. A cosa dovrebbe servire? Vediamo un esempio.

A molti di noi capita di incontrare qualcuno il cui viso ci è noto. Ma, nonostante gli sforzi, non riusciamo a ricordare il suo nome o dove e quando ci siamo già visti.

A parte l'imbarazzo, la memoria annaspa per cercare il ricordo giusto. Con un piccolo aiuto dalla realtà amplificata la situazione sarebbe risolta.

A ogni incontro la telecamera e i microfoni del casco registrano il volto dell'interlocutore, la sua voce, gli argomenti della nostra conversazione e così via. Grazie al posizionamento satellitare Gps il sistema registra automaticamente anche data, ora e luogo dell'incontro.

Tutto finisce nella memoria del mini-computer che gestisce il dispositivo. A ogni incontro successivo, l'apparecchio restituisce le informazioni del suo archivio: con chi stiamo parlando, dove e quando lo abbiamo già incontrato, di cosa avevamo discusso e così via.

Ma non è tutto. Se, per esempio, nella nostra chiacchierata si iniziasse a parlare dei Beatles, il sistema potrebbe trasmettere tutte le informazioni sul complesso: la loro storia, i loro dischi, i loro successi. Potrebbe insomma fornirci tutti gli elementi per sostenere una discussione brillante e approfondita.

La città digitale fuori e cemento sulla pelle

Il titolo fa il verso allo slogan di una vecchia pubblicità *lana fuori e cotone sulla pelle*. Concettualmente le cose non cambiano. Due tecnologie sovrapposte a determinare la nuova condizione.

Con un sistema esteso di realtà amplificata è possibile ipotizzare che la materialità degli oggetti architettonici si possa fermare a struttura e soluzioni tecnologiche sostenibili delegando l'apparenza totalmente, o quasi, al digitale. Si rende necessario un *medium* atto allo scopo. Un esempio, anche se banale come delle note di testo poste di volta in volta sul frigorifero o sulla cucina o altro, può rendere meglio l'idea.

Mi viene in mente un classico esercizio, di modellazione e resa, per sovrapporre un disco volante ad un qualsivoglia paesaggio, con tanto di ombre.

Occhiali prima e *brain chip* dopo et voilà il gioco è fatto. È un percorso, è una possibilità che è venuta a palesarsi con l'era digitale, una fra un'infinità.

Ma a questo punto ognuno potrebbe scegliersi la città dove vivere, una città su misura, una città personalizzata.

La città personalizzata

Brain chip e oltre. Oltre l'abbigliamento la mattina potremmo scegliere dove vivere o quando vivere. Come? Semplicemente *upgradando* il nostro cervello con un bel *brain chip* nuovo di zecca. Ultimo modello, manco a dirlo.

“Oggi mi va di vivere in Cina nel '500 a. C.” Tutto georeferenziato, tutto sotto controllo, tutto gestito dal *sistemone* centrale e me ne vado dove mi pare.

Un'altra eventualità, sicuramente estrema ma tipi di cose con cui il pensare o il fare architettura potrebbe, in futuro, trattare.

Fantasticando per questa strada mi vengono alla mente i disegni di Piranesi e alla possibilità di immaginare un software che valga una prigione. Opprimente ma efficace. Ho la possibilità di immaginare, in tanti lo si è fatto in questi anni, tutte le eventuali applicazioni ludiche. Si riesce ad intravedere la fusione fra cinema e videogiochi in immersione totale.

Tante, troppe cose. Vengono le vertigini. Dalle dimore celesti agli incubi più angosciosi. Senza perdersi oltre e ritornando sul pianeta terra proviamo ad affrontare un concreto esempio di realtà amplificata dato dalle ricostruzioni in tempo reale.

Ricostruzioni in tempo reale

Concettualmente l'idea è molto semplice. Si utilizza un sistema di realtà amplificata per sovrapporre la ricostruzione di un reperto archeologico al reperto stesso.

Il tutto avverrebbe, ovviamente, in tempo reale sfruttando georeferenziazione, occhiali virtuali e resa avanzata.

Immaginiamo, per esempio, la città di Pompei e di fornire all'ingresso, al turista o altro, due apparentemente semplici occhiali con cui visualizzare di volta in volta il com'era e il com'è. È evidente la discrezionalità come è evidente la possibilità di uso e sviluppo della faccenda.

Con la tecnologia attuale abbiamo la possibilità di realizzare un sistema fruibile per movimenti limitati, causa le interferenze con altre persone, ma è ipotizzabile, in un immediato futuro, la possibilità di integrare, nel sistema di gestione, la presenza degli altri, addirittura sostituendoli con personaggi dell'epoca. A dire la verità riesco a pensare che sia più semplice, per il momento, utilizzare dei proxies con personaggi d'epoca piuttosto che le persone reali.

In un paese come il nostro, ricco di testimonianze storiche, la cosa potrebbe avere anche interessanti risvolti economici, in Italia come nel resto del mondo.

Bibliografia

Uomini e macchine intelligenti, Jeremy Bernstein
Logic, Biology and Automata - Some Historical Reflections, Arthur W. Burks
Hackers, Steven Levy
Cybernetics, Norbert Wiener
Le leggi della semplicità, John Maeda
Digital Design Media, William J. Mitchell e Malcolm McCullough
Architettura dei Sistemi di Elaborazione, volume 1, F. Baiardi, A. Tomasi e M. Vanneschi
Architettura dei Sistemi di Elaborazione, volume 2, F. Baiardi, A. Tomasi e M. Vanneschi
Sistemi operativi. Concetti ed esempi, A. Silberschatz, P. Galvin, G. Gagne
Sistemi operativi. Architettura e Programmazione Concorrente, Giorgio Clemente, Federico Filira, Michele Moro
In the Beginning...was the Command Line, Neal Stephenson
The Mythical Man-Month: Essays on Software Engineering, Frederick P. Brooks
CAD Tutor 3D. Corso interattivo di Autocad 3D, Claudio Gasparini
LightWave 3D Guida completa, Dan Ablan
Computer Graphics Techniques: Theory and Practice, David Rogers, Rae Earnshaw
Visual Effects in a Digital World, Karen E. Goulekas
3D Studio MAX 2 Guida completa, Michael Todd Peterson
Maya 5 Guida completa, John Kundert-Gibb e Peter Lee
Maya Guida completa volume 1, Roberto Strippoli
Architettura e Computer, James Steele
Realtà virtuale, Ammar Eloueini
L'architettura radicale e la critica, Carlotta Darò
Realtà amplificata, Antonio Leonardi
Computer Grafica tecniche & applicazioni
3D professional
CG computer Gazette

Sitografia

INFORMAZIONI

Magazines, studios, artists..

Computer Graphics World
HighEnd 3D
3DLuVr
CG Channel
Cinefex
DV Live
Zoorender 3D
Menithings
3D World
3D Render
3D Artists
Computer Arts
Renderosity
Archvision
Industrial Light & Magic
Pixar
Digital Domain
Pacific Data Images
Sony Pictures Imageworks
Tippett Studio
MillFilm
BUF
BlackPool Studios
Attitude (Eve Solal)
Blur Studios
Daniel Robichaud
Grzegorz Jonkajtys
Johannes Schlörb's
Paul Sherstobitoff
Taron
Doug Chiang
Soanala
Pepeland
Igor Posavec
Istvan Pely
Willi Hammes
Boring3d
Feng Zhu
Michael Koch
Ttrinisica
Suurland
SpineFinger Design
WandBilt
Rendium
Skyraider's Aviation Art
Harald Belker
Hunsekigan
Scott Robertson & Neville
Eterea Estudios
Richard Bluff
Teknoel
TJ Frame
Syd Mead
Neil Blevins
Frank DeLise
Steve Burke
Erik Asorson
Brian Taylor´s Rustboy
Ruben Borup
Eklettica
Victor Navone

SOFTWARE

2d, 3d, hardware, tutorials..

3D Studio Max
Lightwave
Cinema 4D
Maya
Softimage
FormZ
EIAS
Houdini
MentalRay
RenderMan
Rhinoceros
Modo
VR Toolbox
PhotoModeler
RealFlow/NextLimit
Project:Messiah
Photoshop
Illustrator
AfterEffects
Premiere
flay.com
finalRender
Modelling a car using nurms
Box modeling a car

RISORSE

Objects, textures, free stuff..

Freetextures
Forrest Textures
People for People
Nemeng
Virtual Scenes
MacWeb3D
Panoguide
DigitalJuice
DV Garage
The Jackals Forge
Animax free textures
Dosch Design
Cultured Stone
3D plants seamless textures
Concept Cars
Bikepics
Airliners
Photovault
Weather Photography
Greatbuildings
3D Up
3D Cafe

MANIFESTAZIONI

Competitions, festivals, etc..

Siggraph
Milia-Cannes
Festival de Annecy
Viper
Festival Anim. Hiroshima
Festival Anim. Holanda
Festival de Stuttgart
ArtFutura
GFX Artist
CG Talk
Digital Art.org
Max Underground
Noir
3DLuvr
Max Forum
NASA
Slashdot
Ars Technica
infoSync
HobbyLink Japan
Car Design News