UAE - Unix Amiga Emulator

di Marcello Mancini (n. matr: 033296)

14 febbraio 2002

Indice

1 Introduzione

Unix Amiga Emulator (UAE) è un emulatore software della piattaforma Amiga che funziona su macchine Unix. Il suo scopo è quello di riprodurre il funzionamento dell'hardware del sistema originale, in modo che il software scritto per la macchina emulata "non si accorga" della differenza tra la macchina reale e quella virtuale creata dall'emulatore, e quindi "giri" esattamente come sul sistema originale.

In questa relazione descrivo le principali caratteristiche, sia dal punto di vista dell'hardware che del sistema operativo, dei sistemi Amiga, il funzionamento e le modalità di impiego dell'emulatore UAE, e confronto il funzionamento di una selezione di software Amiga su una macchina originale e sotto emulazione.

2 Amiga

Amiga è ricordata dall'appassionato medio di informatica come un computer dedicato ai videogiochi che ebbe uno straordinario successo di pubblico nel periodo tra il 1985 ed il 1994. Fu prodotta da *Commodore*, già celeberrima nella prima metà degli anni '80 per un altro prodotto di straordinario successo, il *Commodore 64*.

Ricordare Amiga solo come piattaforma per videogiochi è però limitante. Sin dall'uscita ufficiale, nel 1985, fu chiaro agli addetti ai lavori che Amiga offriva numerose soluzioni tecniche sia dal lato hardware che da quello software davvero interessanti e innovative (47 furono i brevetti originali legati ad Amiga), che la rendevano adatta non solo come macchina da gioco ma anche e soprattutto come piattaforma per l'utilizzo professionale, in particolare nel campo della grafica. L'importanza di Amiga nella storia dell'informatica può essere riassunta da un solo semplice fatto: la parola "**multimediale**" fu coniata proprio per questa piattaforma.

2.1 Breve storia

La storia del progetto Amiga vista nel dettaglio è complicata quanto la trama di una soap-opera, ed è un caso emblematico di quanta importanza abbiano le politiche di marketing e la pianificazione aziendale anche in un settore basato sul progresso e l'avanzamento tecnologico come l'informatica. Cercherò di riassumere in questo capitolo, a grandi linee, la storia di Amiga, tralasciando i dettagli tecnici riguardo all'hardware e al sistema operativo, i quali verranno esposti più avanti.

2.1.1 Nascita del progetto

Il progetto Amiga nasce nel 1980 da un'idea di **Jay Miner**, all'epoca dipendente della nota azienda giapponese *Atari*, in quel momento leader nel settore dei videogiochi grazie a console a 8 bit come il modello "*2600*". Miner propone ad Atari l'idea di un prodotto basato sul nuovo processore della *Motorola*, il **68000** (68k). Atari rifiuta la proposta e Miner decide di lasciare l'azienda per sviluppare il progetto in proprio.

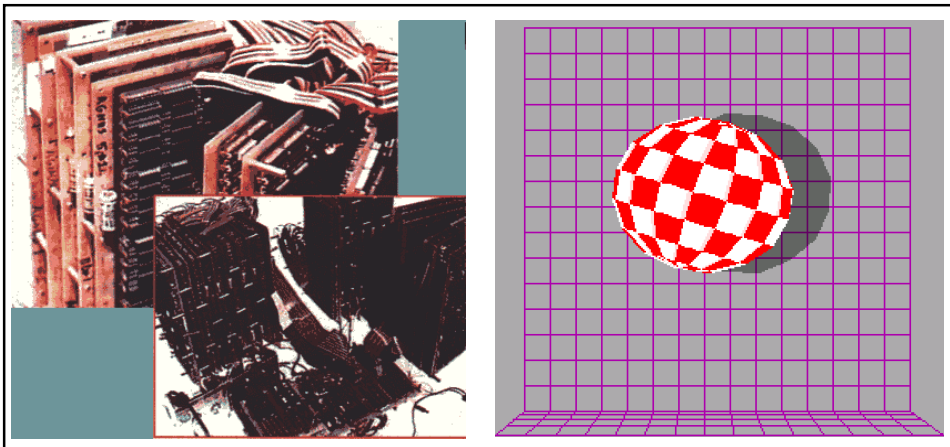
Miner fonda la propria società insieme con un ex-collega dell'Atari, **Larry Kaplan**. Chiamata da principio "*Hi-Toro*", la società viene ribattezzata "*Amiga*" ("amica" in spagnolo) per pure considerazioni di marketing: un nome curioso, simpatico, originale, e soprattutto che in ordine alfabetico viene prima di "*Apple*" e "Atari". **Dave Morse** è assunto come project-manager di quella che in principio vuole diventare una rivoluzionaria console per videogiochi. Il team di sviluppo, contro il parere degli stessi finanziatori della società (i primi furono dei dentisti con alcuni milioni di dollari

da investire nel settore dei videogame), si orienta però verso la realizzazione di un vero e proprio computer. Sviluppano perciò, quasi in segreto, vari componenti come tastiera, lettore di floppy-disk e porte di comunicazione.

L'idea di base del progetto è quella di affiancare al processore principale una serie di **chip custom** che si occupino della grafica, dell'*I/O* e della gestione dei *canali DMA*. Il progetto è all'avanguardia e i tecnici che ci lavorano, consapevoli della debole posizione della loro società nei confronti delle grandi aziende informatiche, lavorano nella più stretta segretezza: addirittura, ai chip custom vengono dati nomi di donne, per sviare eventuali tentativi di spionaggio.

2.1.2 Prototipo

Il primo prototipo del nuovo sistema, denominato "*Lorraine*", compare nel 1984 al *CES show* di Chicago, una mostra del settore informatico. I chip custom non sono ancora completati ed il prototipo è, dal punto di vista estetico, una "mostruosità" composta dalla motherboard e da quattro "*breadboards*", da 25 microchip ciascuna.



A sinistra, due foto di Lorraine, il primo prototipo di Amiga presentato al pubblico. A destra, un fotogramma della demo "Boing!": la palla a scacchi bianchi e rossi è ancora oggi il simbolo dell'architettura Amiga.

Nonostante l'aspetto, Lorraine suscita subito grande stupore ed interesse, soprattutto per un programma dimostrativo (*demo*), scritto da **Dale Luck** e **R.J. Mical** e chiamato "*Boing!*", che mostra l'animazione di una sfera a scacchi bianchi e rossi che rimbalza, renderizzata in *ray-tracing* sulla stessa piattaforma. La sfera a scacchi diventerà da quel momento il simbolo "non ufficiale" di Amiga. Il successo ottenuto spinge i progettisti ad accelerare lo sviluppo: le basi del sistema operativo vengono completate da Mical in tre settimane.

L'azienda si trova però pesantemente indebitata; per poter completare e produrre il progetto necessita di grandi finanziamenti e deve quindi rivolgersi ai colossi del settore. Dopo complicate trattative con varie aziende, tra cui anche l'Atari, "Amiga" viene infine acquisita da **Commodore**.

2.1.3 Il primo Amiga: A1000

Nel 1985 viene presentato il primo modello di Amiga prodotto, **A1000**.

Un Amiga A1000

L'aspetto è quello di un classico PC, con un cabinet di tipo desktop, monta 256 KB di RAM, ed è equipaggiato con un lettore di floppy-disk da 3.5 pollici, monitor a colori ed un mouse. I componenti principali del sistema operativo (denominati "*Kickstart*") vengono pre-caricati in memoria da un dischetto ad ogni accensione della macchina.

Il nuovo prodotto della Commodore è senz'altro interessante, ma poco competitivo sul versante del prezzo se confrontato con la concorrenza, in particolare con il diretto avversario, l'*Atari ST*, che anch'esso monta un processore 68000 ma che non utilizza chip custom e risulta quindi più economico da produrre. Inoltre, la dotazione di RAM, scelta dalla Commodore andando contro le indicazioni del team di progettisti, è troppo scarsa per permettere ad una qualunque applicazione seria di funzionare, e come se non bastasse il sistema operativo, completato in fretta e furia sotto la pressione della Commodore, è instabile e provoca spesso blocchi della macchina (che i programmatori del sistema avevano battezzato "**Guru meditation**"!).

L'atteggiamento della casa-madre irrita i "padri" del progetto Amiga e li spinge ad andarsene: entro pochi mesi dall'uscita sul mercato dell'A1000, nessuno dei progettisti originali di Amiga lavora più per la Commodore.

2.1.4 I "Classic-Amiga": A2000 e A500

L'anno successivo, il 1986, vede l'uscita di due nuovi modelli: il primo è l'**A2000**, evoluzione del precedente A1000, che monta di serie 512 KB di RAM e ha il Kickstart, giunto alla versione 1.3, su una memoria ROM; l'altro è l'**A500**, di caratteristiche analoghe al modello superiore ma di diverso aspetto: nello chassis infatti integra la circuiteria, la tastiera ed il lettore di floppy-disk da 3,5 pollici, ricordando esteticamente l'ultima versione del precedente successo di Commodore, il *C64*, e si presenta perciò al grande pubblico come successore del fortunato microcomputer.

Un A2000 (a sinistra) ed un A500 (a destra). Notare il mouse a due pulsanti fornito di serie per entrambi i prodotti. Il tasto "Help", posto sopra il tasto cursore destro, non troverà un'applicazione nel sistema operativo fino all'introduzione, nel 1992, dell'AmigaGuide.

La scelta di produrre l'A500 rivela l'intenzione di Commodore di trascurare le potenzialità della piattaforma per dedicarsi al mercato *consumer*, interessato ad una macchina da gioco che offrisse anche alcune funzionalità di un computer "serio", ma senza esserlo veramente. Eppure, l'architettura Amiga è tale da poter senz'altro fornire la base per applicazioni serie: il sistema operativo, ad esempio, ha un'interfaccia grafica a finestre, comandata dal mouse, che ricorda nell'aspetto quello dell'*Apple Macintosh*, e lavora in **multitasking** reale; le capacità grafiche sono eccezionali per l'epoca, visto che Amiga è in grado di visualizzare schermi a 64 colori da una palette di 4096, o fino a 4096 colori contemporanei sullo schermo, sfruttando un "trucco" hardware chiamato "*HAM*"; un'ulteriore caratteristica interessante è la compatibilità nativa del chip grafico con gli standard televisivi *PAL* ed *NTSC*, funzione che ancora oggi è disponibile solo a prezzo elevato e come optional sugli attuali PC.

Come detto, Commodore spinge sul mercato consumer, ed effettivamente le vendite del modello A500 decollano nel 1989, nel momento in cui le software-house cominciano a sfornare videogiochi che su altre piattaforme semplicemente non potevano essere realizzati. Il "fratello maggiore" dell'A500, l'A2000, trova estimatori soprattutto nel settore della grafica professionale.

2.1.5 Nuovi prodotti

Nel 1990 esce l'**A3000**, che monta un processore **68030**, hard-disk su bus *SCSI* e la nuova versione del Kickstart, la 2.0, la prima veramente affidabile e adatta all'uso professionale. Gli appassionati non sono però pienamente soddisfatti di questo prodotto. La nuova versione del custom chipset, denominata **ECS** (*Enhanced ChipSet*, mentre la precedente viene ribattezzata **OCS**, *Original ChipSet*) viene vista come "*il più piccolo avanzamento tecnologico nella storia dell'informatica*"; ci si aspettava da Commodore un maggiore sforzo per aumentare o almeno mantenere il vantaggio tecnologico rispetto alle architetture Macintosh e IBM-compatibili. Contestualmente all'uscita dell'A3000, esce anche l'**A500 Plus**, un A500 dotato di chipset ECS, Kickstart 2.0 e una maggiore dotazione di memoria.

Sempre nel '90 viene presentata la scheda "**Video Toaster**", prodotta non dalla Commodore, che aveva perso l'occasione di supportare il progetto, ma dall'azienda *NewTek*. Al contrario di quanto dichiarato scherzosamente in un'intervista da uno dei progettisti, alcuni mesi prima dell'uscita ufficiale dell'hardware, "Video Toaster" non è una rivoluzionaria stampante a "getto di marmellata" per personalizzare le fette di pane nei ristoranti, ma una potente scheda grafica acceleratrice pro-

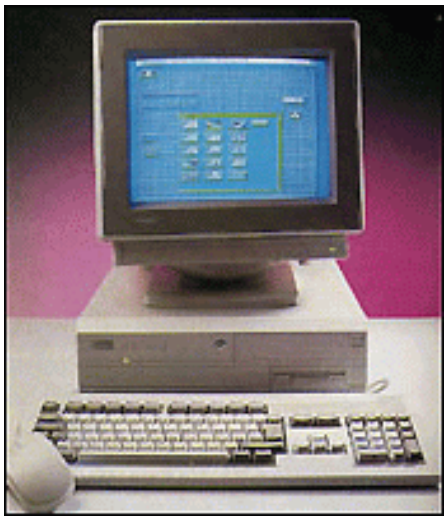
fessionale che, affiancata al software "*Lightwave 3D*", permetteva l'uso di Amiga come piattaforma specializzata negli effetti speciali visivi tridimensionali.

2.1.6 Sviluppi

Il progetto Amiga rimane praticamente fermo fino al 1992, quando esce sul mercato l'**A600**, un ibrido tra un normale A500 ed un computer portatile: di dimensioni inferiori all'A500, con una tastiera priva del tastierino numerico, incorpora un hard-disk **IDE** da 3,5 pollici collocato sotto alla tastiera; la capacità di essere collegato ad un qualunque televisore casalingo, ereditata dai precedenti modelli Amiga, rende possibile, con un po' di spirito di adattamento, il trasporto dell'A600, almeno da una casa all'altra. Caratteristica senz'altro più interessante è il fatto che l'A600 fosse il primo computer dotato di uno slot per le schede **PCMCIA**, standard che oggi si è effettivamente affermato nel mondo dei computer portatili.

Un Amiga 600. Lo slot PCMCIA non è visibile in figura, essendo collocato sul lato sinistro dell'unità.

Nello stesso anno esce l'**A4000**, erede dell'A3000, come facilmente intuibile. L'A4000 è il primo concreto passo in avanti nello sviluppo di Amiga dalla prima uscita ufficiale, nel 1985. Implementa un nuovo chipset, denominato **AGA**, che permette grafica a 256 colori contemporanei da una palette di 16 milioni, o fino a 256000 colori in una modalità simile all'HAM delle precedenti generazioni del chipset; il processore è un **68040**, montato su una scheda, separata dalla motherboard, che nelle intenzioni dovrebbe renderne facile la sostituzione con future evoluzioni. A4000 presenta l'ultima versione del Kickstart, la 3.1, che fornisce all'utente un sistema operativo veramente maturo e funzionale.



Un Amiga A4000

2.1.7 Progetti alternativi

Nel frattempo Commodore, sull'onda del successo dell'A500, si lancia nella realizzazione di progetti alternativi, basati sull'architettura Amiga. Il "**CDTV**", del 1990, è concepito come elettrodomestico per l'intrattenimento multimediale, ed è in pratica un A500 con un lettore di CD-Rom, dentro uno chassis simile a quello di un videoregistratore. Nelle intenzioni della Commodore, il CDTV deve attuare la *convergenza* tra televisione e computer, permettendo all'utilizzatore di giocare, guardare film ed ascoltare musica con lo stesso apparecchio; l'utente più evoluto può poi espandere il CDTV acquistando tastiera, floppy-disk driver e mouse e trasformare il prodotto in un vero e proprio computer.

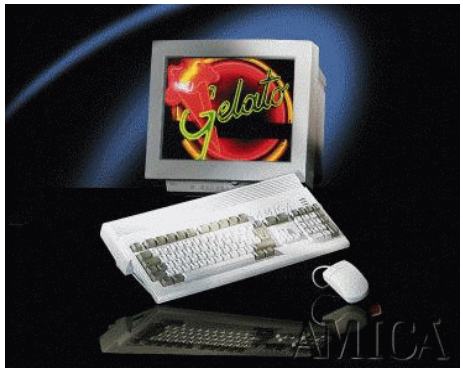
AmigaCD32, del 1993, è invece una console dedicata ai videogiochi, equipaggiata col processore **68020** con architettura a 32 bit e basata sull'uso di un lettore di CD-Rom integrato.

In alto, un CDTV. A destra, un Amiga CD32. Entrambi ebbero scarso successo.

Principalmente per l'assenza di titoli interessanti, queste macchine passarono pressochè inosservate nei negozi. Interessante a mio avviso notare che, con le differenze tecniche date dalla diversa età, due prodotti simili per filosofia hanno oggi straordinario successo: il *lettore di DVD* e "*Playstation*" della *Sony*.

2.1.8 Il canto del cigno

Questi progetti falliti contribuiscono a mettere la Commodore in difficoltà finanziarie. L'**A1200**, vero erede dell'A500, uscito alla fine del 1992, è l'ultimo computer della serie Amiga a essere venduto con il marchio Commodore. Viene equipaggiato con un processore 68020, 2 MB di RAM, Kickstart 3.1, hard disk IDE integrato sotto la tastiera e slot PCMCIA come per l'A600. Migliaia di Amiga A1200 vengono venduti "a scatola chiusa", ancora prima che il computer arrivi nei negozi. Le caratteristiche tecniche ed il successo di pubblico non salvano comunque Commodore dal **fallimento**, che avviene nel 1994.



Un Amiga A1200

Il grande pubblico abbandona Amiga per rivolgersi ai *PC MS-DOS compatibili*, che hanno effettuato con successo la rincorsa al sistema della Commodore, grazie alla potenza di calcolo dei processori *486* e *Pentium*, alle schede video *SVGA*, all'uscita di *Windows 3.1* e, bisogna dirlo, di videogiochi finalmente competitivi (un nome per tutti: "*Doom*" della *Id Software*).

2.1.9 Storia recente

Qui finisce la parte di storia di Amiga di interesse per la presente relazione. Bisogna però sottolineare che Amiga non è morta con Commodore, ma ha continuato a restare in vita ed evolvere, pur con alterne vicende, sostenuta da un gran numero di appassionati estimatori. E' proprio del 2001 una "rinascita" di Amiga, che vive un momento meno luminoso di quello a cavallo degli anni '90 ma comunque interessante. Molti utenti, come si può evincere seguendo su Usenet i newsgroup dedicati ad Amiga, riprendono i loro vecchi A1200, estraggono la circuiteria dallo chassis originale, la inseriscono in *tower-case*, e la modificano con kit custom, equipaggiando questi ibridi con processori *PowerPC* e slot *PCI* per utilizzare le schede prodotte per i PC, come le *3dfx Voodoo3* o le *Creative SoundBlaster*; sono in uscita nuovi videogiochi, e sono molti i siti Internet dedicati ad Amiga nati recentemente; l'azienda che attualmente detiene il copyright del software Amiga, **Amiga Inc.**, ha rilasciato nel 2001 una nuova versione del sistema operativo, giungendo alla 3.9, mentre negli anni si sono susseguite voci di nuovi sistemi operativi come **MorphOS** o **AmigaX**; quest'ultimo dovrebbe essere un sistema di tipo *Unix*.

E' probabile che una parte del merito del rinnovato interesse verso Amiga sia da attribuire proprio all'emulatore oggetto di questa relazione; giunto ad un livello di sviluppo tale da poter far funzionare senza problemi la stragrande maggioranza del software Amiga esistente, **UAE** ha riportato molti appassionati ai "vecchi tempi" e li ha indotti a riprendere in mano i propri Amiga.

2.1.10 Birra

Trovo divertente, ma anche molto efficace nel riassumere in poche righe la storia di Amiga, questo breve brano, scovato tra le varie citazioni ed i vari aneddoti contenuti nell'archivio del programma "*fortunes*" di *Linux*, nel quale si fa un curioso parallelo tra i sistemi operativi e la birra:

> *AmigaDOS Beer*: The company has gone out of business, but their recipe has been picked up by some weird German company, so now this beer will be an import. This beer never really sold very well because the original manufacturer didn't understand marketing. Like Unix Beer, AmigaDOS Beer fans are an extremely loyal and loud group. It originally came in a 16-oz. can, but now comes in 32-oz. cans too. When this can was originally introduced, it appeared flashy and colorful, but the design hasn't changed much over the years, so it appears dated now. Critics of this beer claim that it is only meant for watching TV anyway.
>
> (Birra AmigaDOS: L'azienda ha chiuso, ma la loro ricetta è stata acquistata da una curiosa compagnia tedesca, perciò ora questa birra sarà d'importazione. Non ha mai venduto molto bene perchè il produttore originale non capiva il marketing. Così come per la birra Unix, i fan della birra AmigaDOS sono un gruppo estremamente fedele e chiassoso. In origine veniva venduta in lattine da 16 once, ma ora esiste anche in lattine da 32 once. Quando fu introdotta, appariva brillante e colorata, ma il design non è cambiato molto negli anni, e oggi appare datato. I critici di questa birra sostengono che comunque era pensata solo per guardarci la TV.)

2.2 Amiga nel dettaglio

Nell'esaminare nel dettaglio le caratteristiche hardware e software, prenderò in considerazione principalmente quelle degli Amiga cosiddetti "**Classic**", ossia **A500** e **A2000**, e farò solo un breve accenno alle evoluzioni apparse nei modelli usciti successivamente.

2.2.1 Hardware

Come già accennato in precedenza, la caratteristica peculiare dell'hardware Amiga è il processore Motorola 68000 affiancato da tre chip custom-built che hanno lo scopo di alleviare il carico di lavoro del processore: **Agnus**, **Denise** e **Paula**.

68000 Il Motorola 68k è un processore di tipo CISC, con un bus dati a 16 bit, ma che internamente lavora a 32 bit; il bus degli indirizzi è invece a 24 bit, e ciò consente di indirizzare fino a 16 MB di RAM. Le sue caratteristiche di base sono:

- 8 registri indirizzi a 32 bit;
- 8 registri dati "general-purpose" a 32 bit;
- 56 istruzioni di base;
- 16 modalità di indirizzamento;
- Operazioni di I/O di tipo *memory-mapped*.

La frequenza di clock dei 68k che equipaggiano gli Amiga è di **7.1 MHz**. Il 68k è in grado di operare nelle *modalità utente* e *supervisore*, ed è perciò adatto all'utilizzo con sistemi operativi multitasking.

Agnus Agnus svolge tre funzioni specifiche:

- **controllo dei 25 canali *DMA***: attraverso i canali *DMA* i coprocessori e le periferiche di input, prime tra tutte le memorie di massa come floppy- e hard-disk, possono effettuare il trasferimento diretto dei dati in memoria, senza che occorra la supervisione del processore. In pratica, il 68000 si limita ad indicare l'indirizzo, la dimensione e la destinazione dei dati che gli occorrono, e Agnus si occupa del vero e proprio trasferimento;

- **sincronizzazione** con il pennello elettronico del monitor, attraverso il coprocessore **Copper**. La sincronizzazione con il monitor è indispensabile per avere animazioni fluide, senza scatti e senza il cosiddetto "*effetto neve*". La sincronia col segnale video è utilizzato anche per altre applicazioni nelle quali il timing è importante e deve essere indipendente dalla velocità di calcolo del processore, come ad esempio la riproduzione di musica;

- **gestione del Blitter**: il **Blitter** è un coprocessore il cui scopo è il *trasferimento rapido* di blocchi di memoria da una locazione all'altra. Dati l'indirizzo di partenza e di destinazione delle word da trasferire, il Blitter effettua l'operazione ad una velocità circa 10 volte maggiore di quanto sarebbe possibile utilizzando il 68000. Il Blitter ha accesso esclusivo a 4 canali DMA, e questo gli consente di accedere contemporaneamente a tre differenti sorgenti, combinarle tra loro con le operazioni logiche che è in grado di effettuare (*AND*, *OR*, *NOT*, *XOR*) e di utilizzare il restante canale per mettere il risultato nella destinazione finale. Utilizzando opportunamente le operazioni logiche e "maschere" di bit, il Blitter permette perciò di spostare elementi non necessariamente rettangolari. Altre funzioni del Blitter sono il riempimento rapido di aree di memoria con un valore arbitrario, ed il tracciamento di linee.
  L'utilità del Blitter è evidente nelle applicazioni che richiedono animazioni grafiche, come presentazioni multimediali, titolazioni professionali e videogiochi, ma anche nell'utilizzo dell'interfaccia grafica risulta determinante; basti pensare che la gestione delle finestre grafiche si limita in pratica al tracciamento di linee e rettangoli e allo spostamento di porzioni di schermo.

Denise Denise è il chip che si occupa della generazione dell'output video. Nella prima versione del chipset (denominato **OCS**, *Original ChipSet*) è in grado di gestire risoluzioni standard di 640x256 pixel non interallacciati, o di 640x512 pixel in modalità *interallacciata* (anche: *interlacciata*), che prevede la visualizzazione alternativamente delle righe pari e delle righe dispari dello schermo per ogni passaggio del pennello video sul monitor; nelle modalità cosiddette "*overscan*" la risoluzione può superare i 700 pixel in larghezza e i 300 in altezza, nelle modalità non interlacciate. Da notare che, al contrario di altre piattaforme come ad esempio i PC IBM-compatibili, non esiste una distinzione tra modalità grafiche e modalità solo-testo.

Le frequenze di aggiornamento sono compatibili con gli standard televisivi (furono prodotte due versioni dei chip Denise, una compatibile con lo standard NTSC e una compatibile con lo standard PAL, utilizzato in Europa ed in Italia in particolare); questo consentiva di collegare direttamente Amiga ai televisori casalinghi, anche se in questo caso l'utilizzo della modalità interallacciata, soprattutto con schermate dai colori molto contrastanti, era letteralmente un pugno negli occhi per l'utilizzatore. La compatibilità per gli standard televisivi rendeva anche Amiga utilizzabile senza grandi sforzi per generare titoli e animazioni professionali nel campo televisivo, ossia, utilizzando il gergo del settore, rendeva Amiga una macchina ideale per il **DTV** (*DeskTop Video*)[1].

[1] A tale proposito ricordo un aneddoto curioso, che risale al periodo di massima popolarità di Amiga, attorno all'anno 1990. Un giornalista di una delle testate specializzate nella macchina Commodore si sveglia una mattina ed accende il televisore: quello che vede comparire è la classica schermata lampeggiante che Amiga visualizza quando entra nelle sue

Denise visualizza fino a **64 colori** (al massimo 6 bit per pixel indicano il colore da utilizzare, tra quelli nei registri di colore), da una **palette di 4096** (12 bit, cioè 4 per ogni componente cromatica R, G e B, per ogni registro di colore). Attraverso la modalità detta **HAM** (acronimo di *Hold And Modify*), può mostrare *contemporaneamente* 4096 colori. L'HAM sfrutta un "trucco" hardware: infatti in questa modalità ogni pixel ha una profondità di 6 bit, e questo permetterebbe di utilizzare solo 64 colori; l'HAM, invece, usa per ogni pixel i 4 bit meno significativi per indicare "parzialmente" il colore che dovrà assumere, e gli altri due per indicare una variazione di colore rispetto al pixel immediatamente a sinistra. Per chiarire il concetto può essere utile una tabella con il significato dei 6 bit nella modalità HAM:

Bit 6 e 5	Bit da 4 a 1	Colore del pixel
00	XXXX	Colore del registro XXXX
01	XXXX	Stesso colore del pixel precedente, ma con intensità di blu pari a XXXX
10	XXXX	Stesso colore del pixel precedente, ma con intensità di rosso pari a XXXX
11	XXXX	Stesso colore del pixel precedente, ma con intensità di verde pari a XXXX

La storia della modalità HAM è curiosa: fu concepita da Miner dopo una visita agli stabilimenti della *Silicon Graphics*, all'epoca leader nella produzione di workstation grafiche, e subito implementata nei progetti del chip Denise. In seguito lo stesso Miner cambiò idea e decise di eliminare questa modalità da Denise, ma ciò avrebbe obbligato alla riprogettazione quasi totale del chip, così l'HAM rimase. Lo scetticismo di Miner fu probabilmente causato dal fatto che il funzionamento della modalità HAM rende le immagini sporche e sfocate; è comunque ancora utilizzabile soprattutto per la visualizzazione di fotografie digitalizzate. Guardando le cose in prospettiva, HAM superava di parecchio tutto quanto fosse disponibile a cavallo degli anni '90 sui personal computer, e rendeva quindi Amiga l'unica scelta possibile per chi volesse occuparsi di grafica pittorica e non potesse permettersi una workstation da centinaia di migliaia di dollari.

Un'immagine visualizzata in modalità HAM

Un'ulteriore funzione del chip Denise è la gestione degli **sprite**, oggetti grafici di dimensioni limitate (16 pixel di larghezza e fino a 255 di altezza), completamente indipendenti per risoluzione e

"*Guru meditation*", cioè quando il sistema operativo va in *crash*. L'uomo rimane perplesso per alcuni secondi, infatti il suo Amiga, collegato al televisore, è spento! Solo dopo capisce che l'Amiga in Guru non era il proprio, ma quello impiegato per i titoli elettronici dal canale televisivo sul quale si era sintonizzato!

numero di colori dallo schermo sottostante (detto "*playfield*"), e di cui è l'hardware stesso a implementare la visualizzazione e il movimento. Denise gestisce fino ad otto sprite a quattro colori; essi possono poi essere combinati per ottenere oggetti in movimento più grandi e con un maggior numero di colori; nello stesso fotogramma uno sprite può essere riutilizzato più volte, a condizione che tra due copie venga mantenuta la distanza di una riga video orizzontale. Far muovere un oggetto grafico sullo schermo usando gli sprite comporta per l'hardware Amiga un carico di lavoro molto inferiore a quello necessario per svolgere la stessa funzione con il *Blitter*; in effetti il movimento degli sprite risulta così veloce e fluido da dare l'impressione che essi non facciano parte della schermata grafica sottostante ma la "sorvolino"; un effetto che l'utilizzatore Amiga poteva provare quotidianamente: lo stesso *puntatore del mouse*, usato per interagire con la *GUI* del sistema, è uno sprite. Denise implementa anche una funzione che segnala quando avviene la collisione tra due sprite, oppure tra uno sprite e determinate aree del playfield.

Paula Paula è il chip sonoro, che gestisce le **quattro voci** digitali a 8 bit di Amiga. Amiga produce audio *stereo*: due voci audio sono associate al canale destro, e due al canale sinistro; tutte le voci sono indipendenti, quindi dai due canali provengono suoni completamente diversi, e questo può provocare un effetto fastidioso quando si ascolta l'output Amiga per mezzo di cuffie. Nonostante questo, l'audio offerto da Amiga era tra i migliori che si potessero avere nei primi anni '90 senza ricorrere ad attrezzature musicali professionali MIDI, che comunque Amiga poteva pilotare attraverso un apposito cavo; in questo campo solo l'*Atari STe* potè rivaleggiare con Amiga, mentre per quanto riguarda i PC, all'epoca il non-plus-ultra era rappresentato dalle schede *AdLib*...

Paula si occupa inoltre della gestione dei floppy-disk driver e delle porte di I/O: la porta parallela "*Centronics*" e la seriale *RS-232* a 25 poli.

Memoria Per le piattaforme Amiga esiste una distinzione tra diversi tipi di memoria RAM:

Chip RAM: la Chip RAM è una memoria a 16 bit montata direttamente sulla scheda madre. E' l'unica porzione di memoria, tra quelle installate su un sistema Amiga, a cui i chip custom possono accedere e che possono utilizzare. Questo impone perciò una **forte limitazione** all'architettura Amiga: tutti i dati grafici e audio che devono essere riprodotti dai chip dedicati devono obbligatoriamente risiedere nella memoria Chip. La Chip RAM è comunque **condivisa** col processore, questo significa che può essere utilizzata anche come normale memoria accessibile dal 68000 e allocabile per programmi e dati.
La massima quantità di memoria indirizzabile dalle diverse versioni del chipset fu di 512 KB per il l'OCS, e aumentò nelle successive evoluzioni.

Fast RAM: il gergo Amiga denomina Fast RAM tutta la memoria standard a 16 bit montata su una macchina, che non sia Chip RAM; è perciò la parte di RAM di un sistema Amiga alla quale la CPU ha accesso esclusivo. Essa veniva montata su appositi slot: slot denominati *Zorro* equipaggiavano i computer di forma "standard", ovvero A1000, A2000, A3000 e A4000, e servivano anche per le schede aggiuntive, implementando un dispositivo di autoconfigurazione precursore dell'odierno sistema "*Plug and Play*"; slot particolari, dedicati solo all'espansione della memoria e accedibili per mezzo di una feritoia chiusa da uno sportello in plastica posto sotto la tastiera erano disponibili sui mini-computer A500, A600 e A1200. La parola "Fast" non sottintende però una reale maggiore velocità di questo tipo di memoria rispetto a quella Chip; i due diversi tipi di memoria avevano un tempo di accesso paragonabile.

"Real" Fast RAM: è la memoria a 32 bit disponibile per A3000 e A4000. Al contrario della Fast RAM introdotta poco sopra, la "Real" Fast RAM era realmente più veloce della memoria Chip.

Evoluzioni Le caratteristiche elencate sopra sono valide come detto per gli Amiga "Classic", A500 e A2000. Modelli successivi dei sistemi Amiga introdussero alcuni miglioramenti.

Come processori furono utilizzati, oltre ai 68k, anche (rari) i 68010, **68020** a 14 MHz, **68030** fino a 50 MHz, **68040** fino a 40 MHz e **68060** a 60 MHz.

Rispetto alla serie OCS, i chip della serie **ECS** implementavano solo due nuove risoluzioni dette "*Productivity*", e la capacità di indirizzare fino ad 1 MB di Chip RAM; i chip grafici **AGA**, invece, portarono a 2 MB la memoria Chip indirizzabile, e introdussero le tanto attese modalità grafiche "**truecolor**", a **256 colori** da palette di 16 milioni.

Un fatto curioso fu che la "festa" degli appassionati utenti Amiga per l'avvento del chipset AGA fu "rovinata" dal fatto che le informazioni tecniche del nuovo chipset non furono pubblicate. Questo perchè i progettisti del nuovo hardware avevano incontrato grandi difficoltà durante lo sviluppo dell'AGA per mantenere la compatibilità con le vecchie applicazioni, difficoltà provocate dal fatto che molti programmatori, alla ricerca della massima prestazione ottenibile dalla piattaforma, avevano violato le indicazioni contenute nei manuali di programmazione dell'hardware Amiga e avevano abusivamente utilizzato registri e funzioni dichiarate riservate, invece di seguire gli inviti della documentazione ufficiale ad utilizzare quando possibile le routine del Kickstart. L'intenzione era perciò quella di costringere i programmatori ad usare l'hardware della macchina nel modo "giusto", precludendo la possibilità di scegliere il modo "sbagliato" e facilitandosi in questo modo il compito in vista di un ulteriore sviluppo del chipset. Il destino volle però che l'AGA fosse l'ultimo sviluppo del chipset Amiga!

2.3 Il sistema operativo

2.3.1 Introduzione

Il sistema operativo Amiga può essere visto come composto da tre blocchi principali:

Exec, che gestisce i processi, la memoria e tutte le strutture di controllo ad essi relativi;

AmigaDOS, che gestisce la parte di alto livello del filesystem e l'interfaccia a linea di comando (**CLI**, *Command Line Interface*);

Intuition, che fornisce i "mattoni" per la costruzione dell'interfaccia grafica a finestre.

Di questi tre, **Exec** è il *kernel* del sistema, la struttura portante che permette di ottenere multitasking reale su macchine con processori a 7 MHz e, contando anche la ROM del Kickstart, meno di un megabyte di memoria. Questo è possibile perchè Exec è stato progettato con l'intento di ottenere la *massima efficienza con la minima ridondanza*, e realizza questo scopo per mezzo di una struttura **orientata agli oggetti**[2].

Il motivo di questa scelta appare chiaro se si analizza un generico sistema operativo moderno: la maggior parte del lavoro del kernel si riduce alla gestione di strutture di dati come code, liste o tabelle, a seconda della scelta effettuata dai progettisti del sistema. Le funzioni alla base della gestione di questi elementi sono molto simili nel funzionamento, e si differenziano in generale soprattutto per le

[2]Curiosamente, Exec ha una struttura *object-oriented* ma **non è** stato scritto in un linguaggio OOP.

strutture dati trattate: ad esempio, le funzioni che permettono di inserire nuovi elementi nella ready queue saranno molto simili a quelle che svolgono lo stesso compito per la lista delle pagine di memoria attive di un processo: in entrambi i casi si deve allocare la memoria necessaria, e modificare nel modo opportuno dei puntatori agli altri elementi della lista; le funzioni hanno lo stesso principio di funzionamento ma sono necessariamente diverse perchè trattano strutture dati differenti.

Se si tiene conto del fatto che queste funzioni, essendo parte del sistema operativo, sono tenute nascoste ai programmatori di applicazioni utente, che quindi devono a loro volta implementare per ciascun programma funzioni che svolgono sempre gli stessi compiti ma si differenziano per i dati gestiti, risulta chiaro che in un sistema operativo tradizionale esiste, durante il funzionamento a regime, una certa quantità di funzioni "simili ma non identiche" che risultano *ridondanti*.

Exec sceglie una filosofia di funzionamento opposta: vengono implementate delle funzioni di utilizzo generale (*general-purpose*), che vadano altrettanto bene per ogni esigenza del sistema operativo; inoltre, queste funzioni vengono messe a disposizione del programmatore e dei programmi utente. I vantaggi di questa soluzione rispetto alla situazione precedente sono evidenti: un cospicuo *risparmio di memoria*; una *maggiore semplicità* del sistema, perchè esiste un numero molto minore di funzioni uniche le quali vengono utilizzate in contesti diversi nello stesso modo; la possibilità per il programmatore di usare *funzioni già ampiamente sperimentate* e perfettamente efficienti.

La **struttura orientata agli oggetti** è il mezzo ideale per implementare questa idea, grazie al concetto di **ereditarietà** (*inheritance*) di funzioni e strutture dati. Data una **classe** (*class*), un'entità autocontenuta comprendente delle strutture dati e le funzioni per gestirle, è sempre possibile creare un nuovo oggetto derivandolo dal primo: la nuova entità erediterà tutte le strutture dati e le funzioni che le gestiscono dall'oggetto "genitore", e potrà includerne di nuove, adatte per un utilizzo più specializzato. I vari oggetti di un sistema potranno essere ordinati gerarchicamente in un albero in cui i nodi più vicini alla radice sono le classi più generiche, e quelli più lontani sono le classi più specializzate, che però comprendono tutte le funzionalità degli oggetti da cui derivano.

In Exec la struttura scelta dai progettisti per le strutture di dati è la **lista a puntatori doppi** (*doubly linked list*). Questa scelta presenta un ovvio vantaggio ed un ovvio svantaggio. Il vantaggio sta nell'efficienza dell'utilizzo della memoria: una lista a puntatori può essere allocata dinamicamente con facilità, perciò in ogni momento solo la memoria effettivamente utilizzata dal sistema operativo è allocata; lo svantaggio sta nella velocità di accesso ad un singolo elemento della lista: per raggiungere un determinato elemento occorre scorrere tutti gli elementi che lo precedono.

L'elemento alla base di Exec è perciò il singolo elemento componente una lista a puntatori doppi, quindi un oggetto che possiede un'intestazione e due puntatori, i quali servono per collegare ogni elemento rispettivamente al precedente e al successivo della lista. Questo oggetto si chiama **MinNode** e la classe che gestisce le liste si chiama **MinList**; essa è concepita, come già accennato poco sopra, per essere il più possibile di utilizzo generale; contiene infatti un **set completo di funzioni** per aggiungere nuovi MinNode ad una lista, a partire dalla testa, dalla coda o da un punto qualsiasi della catena, permettendo in questo modo di implementare liste *FIFO* (**queue**), *LIFO* (**stack**) o liste generiche. Il numero di casi particolari da considerare e trattare con varianti delle funzioni di base viene ridotto utilizzando un espediente: una lista vuota contiene in realtà due MinNodes, chiamati *dummy-nodes*, e tutti i nodi aggiunti alla lista vengono posizionati tra di essi; i dummy-nodes sono perciò i marcatori di inizio e fine lista, e gli unici elementi della lista ad avere puntatori a *NULL*, mentre i nodi "significativi" hanno sempre un nodo precedente ed un nodo successivo.

Tutte le strutture dati di Exec estendono le funzioni della classe MinList, aggiungendo ai MinNode le funzionalità via via necessarie.

2.3.2 Gestione dei processi

L'unità di esecuzione fondamentale sui sistemi Amiga è detto **task**. Le varie funzioni a basso livello del sistema operativo, come il filesystem e l'accesso alle periferiche di I/O, sono implementate come task sempre in esecuzione. I programmi utente sono in genere task estesi, chiamati **processi**, i quali oltre ad avere le caratteristiche dei normali task hanno anche la possibilità di accedere alla libreria di funzioni *dos.library* (vd. par. 2.3.5) per tutte le operazioni riguardanti il filesystem, la gestione dell'I/O e la creazione di nuovi processi.

Ogni task può trovarsi in uno dei seguenti stati:

running: il task è attualmente in esecuzione;

ready: il task si trova nella *ready-queue* ed è pronto per essere eseguito;

waiting: il task attende un evento esterno, e non è pronto per essere eseguito;

added: il task è stato appena creato, e non è ancora pronto per l'esecuzione;

removed: il task è stato terminato e la sua struttura sta per essere rimossa dalla memoria;

exception: il task è pronto per l'esecuzione come *exception*. Una exception viene eseguita appena invocata e interrompe ogni altro processo in esecuzione. Si può pensare ad una exception come ad un *task real-time*.

Exec mantiene una coda di ready tasks (**ready queue**) ed una di waiting tasks (**waiting queue**). Ad ogni task è associato un numero intero detto **priorità**, che va da -127 a 128, e la ready queue è ordinata secondo la priorità dei task in coda: task a priorità maggiore verranno collocati in testa alla lista. La waiting queue è invece una lista di tipo *FIFO*, e nuovi elementi in arrivo verranno collocati in fondo alla lista.

Lo scheduling è piuttosto semplice, essendo in pratica un **priority round-robin** con **preemption**. Un task rimane attivo finchè non si verifica una delle seguenti condizioni:

- Un task con priorità maggiore diventa ready, oppure viene invocata una exception;
- Il task attualmente running deve attendere un evento;
- Il task attualmente running rimane funzionante per un'intero **quanto di tempo**;

In ognuno di questi casi il task che diventa running è quello ready a priorità maggiore; in caso di esistenza di più task alla stessa priorità nella ready queue, essi verranno eseguiti uno dopo l'altro in sequenza, secondo il loro ordine nella queue.

L'algoritmo per implementare uno scheduler così concepito è senz'altro molto semplice e richiede perciò una minima quantità di tempo di CPU per calcolare i task da mandare in esecuzione. Inoltre, si avvicina all'ideale scheduler real-time, visto che task a priorità più alta verranno eseguiti appena divengono ready. Un ovvio difetto è che per i task a priorità più bassa c'è il rischio di **starvation**, se task a priorità maggiore sono costantemente in ready queue, anche perchè Exec non implementa alcuna politica per il ricalcolo dinamico delle priorità. Questo è a prima vista un grosso difetto dell'architettura Amiga; bisogna però prendere in considerazione alcuni fatti:

- i task di sistema hanno in genere priorità compresa tra +20 e -20; di default le applicazioni utente hanno invece tutte priorità zero, perciò si spartiranno equamente il tempo di CPU;

- Amiga è un sistema *monoutente*, perciò è lecito aspettarsi che in ogni momento ci sia una sola applicazione attiva con la quale l'utente interagisce, più una serie di task di servizio e di processi che girano in background;
- esistono comandi di sistema con i quali l'utilizzatore può cambiare la priorità dei processi, in caso di esigenze particolari.

Tenuto conto del contesto, ci si rende conto che questa politica di scheduling rappresenta un buon compromesso tra leggerezza, semplicità di implementazione ed efficienza.

2.3.3 Comunicazione tra i task

I task comunicano tra loro per mezzo di **segnali** (*signals*) e **messaggi** (*messages*).

Un segnale è costituito da una word di 32 bit che un task spedisce ad un altro. Ogni bit del segnale è un flag: i primi 16 sono riservati al sistema operativo mentre gli altri 16 sono utilizzabili arbitrariamente dal programmatore. Un segnale è un mezzo molto comodo per "risvegliare" un task in waiting state, avvertendolo dell'avvenimento di un evento. Ogni task che si trova in stato waiting possiede un pattern di segnali dei quali è in attesa; al ricevimento di un segnale, la word del signal viene confrontata con il pattern dei messaggi attesi, ed un risultato non-zero provoca l'inserimento del task nella ready-queue.

Un messaggio è un'estensione del concetto di segnale, ed è in pratica una stringa che un task invia ad un'apposita **porta** (*message port*), dalla quale gli altri task in "ascolto" possono prelevare e leggere il messaggio. Ogni task può creare un numero arbitrario di porte e messaggi, essendo il solo limite la quantità di memoria disponibile; le porte possono essere private e di uso esclusivo di alcuni task, oppure un task può aggiungere una porta alla lista delle message port di sistema. I messaggi sono, ad esempio, la base delle richieste di I/O agli appositi task, e la struttura utilizzata per comunicare ai processi gli eventi attivati dall'utilizzo del mouse sulle GUI.

2.3.4 Gestione della memoria

Su Amiga la memoria RAM non viene divisa in pagine ma trattata come **risorsa unica**. Una porzione di memoria continua è detta **blocco**; Exec mantiene una **lista di blocchi di memoria liberi**: per ciascun blocco l'elemento della lista contiene l'indirizzo e la dimensione. Quando un processo necessita di allocare memoria effettua una richiesta ad Exec, il quale seleziona un blocco di memoria libero, lo assegna al processo e modifica di conseguenza la lista dei blocchi liberi.

Non esiste alcuna funzione di gestione della memoria virtuale: il funzionamento del sistema è legato alla presenza di una quantità sufficiente di memoria fisica. Comunque, l'unico limite alla memoria allocabile da un singolo programma è la memoria totale disponibile; questo significa che un processo non incorre in un errore di "*out-of-memory*" finchè non è l'intero sistema ad esaurire la RAM.

Exec non implementa alcun meccanismo di *protezione* della memoria; un programma mal scritto o "malizioso" può invadere la memoria di altri processi e modificarla; il risultato in questo caso è il danneggiamento delle altre applicazioni e una probabile *Guru Meditation*. Inoltre, Exec non mantiene una lista dei blocchi di memoria effettivamente utilizzati, perciò se un processo in fase di conclusione non provvede a liberare la memoria che ha allocato, essa non sarà più disponibile fino al riavvio della macchina. Questo è certamente il più grande limite dell'architettura Amiga. C'è da dire che funzioni di *resource-tracking* erano previste nel progetto iniziale del sistema operativo di Amiga, ma esse non furono implementate per la fretta di uscire sul mercato.

2.3.5 Librerie

Una *libreria* (**library**[3]) è un gruppo di funzioni relative ad una determinata funzionalità del sistema; le funzioni fornite sono scritte in codice rientrante e sono perciò utilizzabili da più task contemporaneamente. Su Amiga tutte le funzioni di servizio, come le primitive per la gestione delle liste, della memoria, delle message ports, sono contenute nella principale libreria di sistema, *exec.library*.

In memoria le librerie hanno una struttura molto semplice: il codice vero e proprio delle diverse funzioni di libreria è preceduto da una *jump table* in cui ogni campo contiene un'istruzione di salto (jump) all'indirizzo dell'inizio del codice di ciascuna funzione. Conoscendo quale posizione occupa una determinata funzione nell'elenco delle funzioni di una libreria (tale informazione è ovviamente contenuta nella documentazione ufficiale e nei file include forniti con il *Software Development Kit*), basterà moltiplicare la posizione per la dimensione di ogni campo della jump table (6 byte), sommare l'indirizzo dell'inizio della jump table stessa, per ottenere l'indirizzo dell'istruzione di jump al codice della funzione desiderata. Ad esempio se la jump table di una generica libreria è collocata all'indirizzo `0x0000F000`, e si vuole invocare la quarta funzione della libreria, si dovrà effettuare il calcolo:

```
destinazione = 0x0000F0000 + 0x06 * 0x04 = 0x0000F000 + 0x18 = 0x0000F018
```

A questo punto basta effettuare un jump all'indirizzo `destinazione`, che a sua volta contiene l'istruzione di jump all'inizio del codice della funzione che si voleva utilizzare.

La jump-table della exec.library è fissa all'indirizzo `0x00000004`; tutte le altre librerie, sia quelle di sistema che quelle prodotte da terzi, sono caricate dinamicamente in memoria quando occorrono, perciò occuperanno verosimilmente una posizione in memoria diversa ad ogni boot; il loro indirizzo è comunque ottenibile attraverso un'apposita funzione della exec.library.

Il vantaggio di questa soluzione è dato dal fatto che la chiamata al sistema non è molto diversa dal jump del codice ad una qualsiasi funzione interna al task chiamante, perciò non c'è la necessità di effettuare il cambio di contesto, ossia di interrompere il programma in esecuzione, salvarne lo stato e i registri, eseguire la funzione richiesta, e poi ripristinare il programma chiamante. Un altro vantaggio è la relativa facilità con cui si può espandere il sistema, aggiungendo librerie le quali al momento dell'esecuzione sono indistinguibili dalle librerie di sistema, e persino sostituendo alle librerie già presenti delle versioni modificate.

2.3.6 Device

L'utilizzo delle diverse periferiche da parte dei task avviene, come già accennato, per mezzo di appositi task che rimangono in esecuzione in background. Questi task vengono chiamati **device**; le richieste di accesso ad una periferica avvengono in genere attraverso messaggi spediti alla relativa porta. Le richieste sono sempre **asincrone**, ma per garantire la sincronia tra le richieste dei dati ed il loro utilizzo esse possono essere effettuate attraverso apposite funzioni della libreria *dos.library*, le quali mettono il task chiamante in stato waiting finchè le richieste non vengono soddisfatte, rendendo perciò l'accesso a periferica sincrono dal punto di vista del singolo task; il passaggio attraverso la dos.library non è comunque obbligato.

[3]La traduzione corretta di library è "*biblioteca*", ma nel gergo Amiga viene preferita la parola "*libreria*". Quando nella stesura della versione italiana del manuale della versione 2.0 del sistema operativo la parola "library" venne tradotta letteralmente, le riviste specializzate e gli addetti ai lavori protestarono e criticarono aspramente la scelta di affidare a traduttori estranei al mondo Amiga la traduzione della documentazione ufficiale del sistema.

2.3.7 Accesso concorrente a memoria condivisa

Spesso in un sistema multitasking c'è la necessità che diversi processi possano accedere agli stessi dati e all'occorrenza modificarli; i sistemi operativi devono perciò implementare dei metodi per la gestione dell'accesso concorrente alla memoria, in modo che un processo non vada a modificare i dati che un altro processo sta accedendo.

Amiga mette a disposizione tre tecniche per la gestione degli accessi concorrenti a memoria. Le prime due sono piuttosto primitive e poco efficienti, e consistono nel disabilitare il multitasking per il tempo necessario ad un task per teminare l'accesso ai dati che gli occorrono. In particolare:

Forbidding: le due funzioni `Forbid()` e `Permit()` rispettivamente disabilitano e riabilitano il multitasking. Disabilitare il multitasking significa impedire il dispatching dei task, in pratica imporre al sistema di mantenere in esecuzione il task che ha invocato la funzione `Forbid()`, finchè il task stesso non chiamerà la funzione `Permit()`; questo indipendentemente dalla priorità del task. Rimangono però abilitati gli interrupt;

Disabling: il disabling, che si effettua con la coppia di funzioni `Disable()` ed `Enable()`, ha gli stessi effetti del forbidding ma in più disabilita anche gli interrupt. Un task che effettua il disabling ha perciò pieno ed esclusivo controllo del sistema;

La terza tecnica, più raffinata e sicuramente più corretta dal punto di vista formale, è la regolazione dell'accesso alla memoria condivisa attraverso i semafori, messi a disposizione dal sistema. L'uso dei semafori ha il grande vantaggio di bloccare solo i task che tentano di accedere ai dati regolati dal semaforo, mentre i task "estranei" non vengono influenzati.

Una volta inizializzato un semaforo, esso viene utilizzato come un classico semaforo binario, attraverso le funzioni `ObtainSemaphore()` e `ReleaseSemaphore()`. In dettaglio, un task che vuole accedere ad una memoria condivisa invoca la funzione `ObtainSemaphore()` utilizzando come parametro il semaforo che vuole utilizzare: se il semaforo è libero, il task chiamante ne prende possesso ed è libero di accedere alla memoria; se invece il semaforo è già stato ottenuto da un altro processo, il task viene posto in waiting state e la sua richiesta accodata alle altre eventualmente già effettuate da altri task. Quando il task che possiede il semaforo lo rilascia utilizzando la funzione `ReleaseSemaphore()`, esso viene "dato" al primo task in attesa. I task possono poi semplicemente informarsi sullo stato del semaforo utilizzando la funzione `AttemptSemaphore()`.

Un utilizzo alternativo dei semafori viene dalle funzioni `Procure()` e `Vacate()`. Il task che invoca il `Procure()` "prenota" il semaforo indicato come parametro alla funzione, ma non viene bloccato se il semaforo richiesto è già in possesso di un altro task. Il task continua a funzionare normalmente, svolgendo le operazioni che non richiedono l'accesso alla memoria regolata dal semaforo, fin quando il semaforo stesso viene liberato; a quel punto il sistema invia un message al task per informarlo che il semaforo è libero e che è il suo turno per prenderlo. Il `Vacate()` svolge la stessa funzione del `ReleaseSemaphore()`, ma può anche servire per annullare la "prenotazione" nel caso al task non occorra più.

2.3.8 AmigaDOS e filesystem

L'AmigaDOS è la parte del sistema operativo Amiga che fornisce l'accesso ai dati contenuti nelle memorie di massa, mediato dal paradigma del file.

Il filesystem Amiga è diviso in **volumi** (*volumes*): ogni floppy-disk inserito nei lettori e ogni partizione degli hard-disk installati su un sistema costituisce un volume. Una particolarità è data dal fatto che AmigaDOS rileva automaticamente un floppy-disk quando viene inserito nel lettore e lo

aggiunge all'elenco dei volumi montati nel sistema. Per rilevare quando un dischetto è inserito, il sistema operativo aziona ad intervalli di tempo regolari la testina di ogni driver, tentando di spostarla di un passo in una direzione in cui non è possibile muoverla se un disco è presente: perciò se il tentativo non ha successo, effettivamente c'è un disco nel lettore, e AmigaDOS può procedere andando a leggerne il RootBlock. Questo movimento della testina provoca il "click" tipico dei driver Amiga. Anche l'operazione inversa è automatica; se un disco viene estratto dal lettore, AmigaDOS può reagire in due diversi modi: se alcune risorse presenti sul disco sono in uso dal sistema, AmigaDOS lascia il volume nell'elenco dei volumi montati, e se dovesse aver bisogno di leggere altri dati su quel disco, utilizzerà la GUI (vd. par. 2.3.12) per chiedere all'utente di reinserire il disco in un driver; altrimenti, il disco è rimosso dall'elenco dei volumi montati.

I volumi si distinguono per mezzo del nome, il quale può essere una qualunque stringa alfanumerica, contenente qualunque carattere ad esclusione di "`/`" e "`:`", fino ad un massimo di 127. Ogni volume contiene file e directory: il limite di file e directory contenuti in un volume è dato solo dalla capienza del volume stesso; i loro nomi seguono le stesse regole dei nomi di volume, mentre la dimensione è limitata a 30 caratteri. Il filesystem Amiga è **case-insensitive**, come *FAT* e al contrario di *ext*. In un path, il carattere "`:`" indica la root di un volume, mentre "`/`" è il separatore delle directory.

Un volume è fisicamente diviso in unità di allocazione dette **blocchi** (*blocks*), la cui dimensione è tipicamente di 512 byte. Un floppy-disk di tipo DD, ovvero a bassa densità, contiene per ciascuno dei due lati 80 **tracce** (*cylinders*) da 11 blocchi ciascuna, per un totale di 1760 blocchi e 880 KB di capienza; floppy-disk HD, montati in genere su Amiga di fascia alta, hanno un numero doppio di blocchi per traccia e perciò contengono un totale di 3520 blocchi e 1760 KB. Per un hard-disk i valori dipendono ovviamente dalla geometria del supporto.

I dati di volume sono contenuti nel *RootBlock*, il blocco situato nella posizione intermedia del supporto (880 per i dischi DD, 1760 per i dischi HD). Esso contiene il nome del volume, un puntatore ai dati della root directory, la posizione delle bitmap che indicano i blocchi usati e quelli liberi, la data di creazione e ultima modifica del filesystem e un checksum. Una data è costituita da tre `ulong` (1 ulong = 4 byte) che indicano rispettivamente il numero di giorni trascorsi dall'1/1/1978, di minuti trascorsi dalla mezzanotte e di tick (1 tick = 1/50 di secondo) trascorso dall'ultimo minuto; da notare perciò che Amiga non risente di nessuno dei problemi noti come "*Millenium bug*": tali problemi si manifesteranno non prima che siano trascorsi 11 milioni di anni!

Ogni directory è memorizzata come lista concatenata di blocchi, e contiene le informazioni su file e directory contenute, in particolare il nome, la data di creazione e modifica, i bit di protezione, la posizione del primo blocco che ne contiene le informazioni, ed un checksum. I bit di protezione sono costituiti da una ulong in cui ogni bit è un flag per un determinato attributo.

Flag	Significato (flag = 1)
0	non cancellabile
1	non eseguibile
2	non scrivibile
3	non leggibile
4	archivio
5	puro
6	script
7	hold
8-31	riservati

Un file è *puro* quando è un eseguibile scritto in codice rientrante, che è possibile rendere residente (vd. par. 2.3.10). L'attributo *hold* sta ad indicare un file eseguibile che può essere reso residente e rimanere in memoria anche dopo il soft-reset del sistema.

Un file viene memorizzato in blocchi che sono concatenati tra loro come una lista. I blocchi costituenti un file vengono memorizzati dove possibile, perciò dischi in cui vengono effettuate frequenti aggiunte e cancellazioni di file diventano frammentati molto velocemente. La frammentazione è causa di rallentamenti nell'accesso ai dati su disco; il fenomeno è particolarmente evidente quando il disco di lavoro è un floppy-disk, ed è sottolineato sia dai tempi di attesa che si prolungano in modo sensibile, sia dal tipico rumore della testina del floppy-disk driver Amiga che effettua il seek delle tracce da una parte all'altra del disco. Tra i tool di sistema non esiste un programma per deframmentare i volumi: tali applicazioni sono però prodotte da terzi e disponibili nel panorama dei programmi free e shareware.

2.3.9 Nomi di periferica, device virtuali e assign

Ad ogni periferica installata su un sistema Amiga, sia essa un'unità di memorizzazione o una porta di comunicazione, corrisponde un volume: i floppy disk hanno nome `DFx:`, dove "`x`" è un numero da 0 a 3; gli hard-disk si chiamano `HDx:`, dove "`x`" è un numero intero da 0 a 19. Per riferirsi ad un floppy disk si può perciò sia usare il nome del volume, che il nome del driver in cui è inserito. Le porte seriale e parallela si chiamano rispettivamente `SER:` e `PAR:`; il fatto di poter riferirsi alle porte di comunicazione come volumi consente ad esempio di mandare un file da un computer all'altro attraverso la porta seriale semplicemente copiando il file sul volume `SER:`. Anche alla stampante corrisponde un volume, il cui nome è `PRT:`; ad essa può corrispondere la porta seriale o quella parallela, a seconda di come sono state impostate le preferenze di sistema.

Esistono inoltre dei device che non hanno una corrispondenza diretta con una periferica fisica installata nel sistema. Alcuni di essi vengono trattati come veri e propri filesystem virtuali; alcuni altri sono dei "canali" a cui vanno rediretti i dati, come succede per `SER:` e `PAR:`; altri infine rappresentano una "scorciatoia" per riferirsi a directory residenti sui filesystem fisici, e vengono chiamati assign.

Non c'è limite ai device che possono essere aggiunti ad un sistema Amiga attraverso gli appositi file binari e file di configurazione, che vengono poi attivati da command line (vd. par. 2.3.10) attraverso il comando "`mount`". Alcuni di questi vengono forniti con il Kickstart e vengono utilizzati per alcune operazioni del sistema operativo: vediamo i più importanti.

Filesystem virtuali I filesystem virtuali con i quali anche l'utente meno interessato alle funzionalità avanzate del sistema Amiga entra in contatto sono quelli che rendono possibile l'accesso alla memoria RAM come se fosse un normale volume, in cui è possibile copiare file e creare directory. Il loro scopo è quello di diminuire quanto possibile lo scambio di dischetti nel lettore: infatti sui sistemi Amiga dotati di uno o due floppy-disk driver e nessun hard-disk, configurazione tipica per le installazioni casalinghe più economiche, lo "swapping" dei dischetti poteva a volte diventare una delle principali attività dell'utente e impiegare una quantità enorme del suo tempo!

Il `RAM:` è un disco virtuale di dimensione dinamica, ossia le sue dimensioni sono sempre uguali a quelle delle informazioni che contiene, ed è sempre possibile aggiungere file e directory fintantochè rimane memoria fisica disponibile. Il contenuto del `RAM:` viene ovviamente perduto ad ogni reboot del sistema. Il `RAM:` viene utilizzato dal sistema stesso, ad esempio come contenitore della clipboard per le operazioni di copia-e-incolla delle applicazioni, ed è perciò sempre presente e disponibile.

Il `RAD:` è invece un disco virtuale di dimensioni prefissate, scelte a piacere al momento della sua inizializzazione, e resistente ai soft-reset: il suo contenuto viene cioè preservato al reset del sistema, ma ovviamente viene perduto quando la macchina viene spenta. Il device del `RAD:` viene fornito col sistema operativo a partire dalla versione 2, ma di default non viene attivato automaticamente; può però essere attivato dall'utente in qualsiasi momento.

Un device prodotto da terze parti, perciò non disponibile “di serie” su Amiga, e chiamato `RAS:`, unisce le proprietà del RAM e del RAD, è cioè un disco virtuale dinamico e resistente al reset.

Canali Il canale più importante in Amiga è `CON:`; i dati inviati a `CON:` compaiono in una console, una finestra che si apre nella GUI (vd. par. 2.3.12) e che serve a visualizzare testi. La funzione principale di `CON:` è quella di fornire la finestra per la CLI (vd. par. 2.3.10), l’interfaccia a linea di comando. `CON:` permette la gestione di alcuni parametri come la dimensione, la posizione e le caratteristiche della finestra di console.

Una possibilità senz’altro interessante offerta da Amiga è quella della sostituzione dei device forniti col sistema con altri scritti da terze parti. Un esempio notevole è il device *KingCON* (`KCON:`), pensato come sostituto di `CON:` che aggiunge una serie di funzionalità alla console, come una barra di scorrimento per rivedere l’output della finestra, la possibilità di salvare l’output in un file e l’autocompletamento del nome dei file attraverso il tasto TAB, come accade in Bash.

Un altro canale di interesse è `PIPE:`, l’implementazione in Amiga del *pipe*: si tratta di un buffer del tipo *FIFO* che può essere utilizzato per il passaggio di dati da un processo ad un altro. L’utilizzo è paragonabile a quello dei pipe di Unix, con la differenza che il pipe Amiga è uno solo, perciò solo due processi alla volta, uno che invia i dati e uno che li riceve, possono usare `PIPE:` per comunicare.

Assign Attraverso il comando `assign` dato da CLI (vd. par. 2.3.10) è possibile creare un numero a piacere di assign in un sistema Amiga: un possibile utilizzo è quello con le applicazioni che fanno riferimento ai propri dati attraverso il nome del volume in cui essi devono risiedere; tali dati possono essere messi ovunque su un filesystem, basta che un assign faccia riferimento a tale directory col nome del volume che si aspetta. Se ad esempio un programma di word-processing si aspetta di trovare i file di definizione dell’aspetto dei caratteri nel volume `FONTS:`, e si è creata una directory `System:Fonts` in cui si desidera collocare questi file, basterà assegnare il nome `FONTS:` a quella directory perchè il programma riesca a trovarli senza problemi. E’ anche per questa caratteristica di Amiga che per installare un programma su hard-disk spesso basta copiare la directory in una qualsiasi directory sull’hard-disk e creare gli opportuni assign.

Alcuni assign hanno una funzione particolare per il sistema, e vengono creati automaticamente al boot. Essi sono:

Nome assign	Descrizione
`C:`	E’ la directory che contiene i comandi della CLI. Di default viene assegnata alla directory `:c` del volume da cui viene effettuato il boot.
`S:`	In `S:` vanno collocati gli script di sistema, ad esempio “`startup-sequence`“, che contiene i comandi da eseguire al boot. Viene assegnata a `:s`.
`DEVS:`	Il volume che contiene i file binari e di configurazione dei device. Viene assegnata a `:devs`
`LIBS:`	Il volume che contiene le librerie di sistema. Viene assegnata a `:libs`
`L:`	E’ il volume che le applicazioni usano per memorizzarvi le opzioni di configurazione. Viene assegnata a `:l`

Nome assign	Descrizione
ENV:	Il volume in cui vengono memorizzate temporaneamente le variabili ambientali, sotto forma di file di testo. Lo startup-sequence del Workbench lo assegna ad una directory :env che crea in RAM:.
ENVARC:	Il volume in cui le variabili ambientali contenute in ENV: vengono memorizzate quando le si rende permanenti. Viene assegnata a :Prefs/Env-Archive
T:	E' il volume usato come directory temporanea. Lo startup-sequence del Workbench lo assegna ad una directory :t che crea in RAM:
SYS:	E' la root del volume da cui è stato effettuato il boot.

Ad un assign possono fare riferimento più directory. Ad esempio se su un dischetto si hanno delle librerie che non si vogliono installare nel sistema ma servono una tantum per una certa applicazione, è possibile aggiungere il path del dischetto a quello della directory già assegnata all'assign LIBS:: il sistema cercherà le librerie richieste dall'applicazione nelle due diverse directory.

2.3.10 CLI

La CLI è l'interfaccia a linea di comando di Amiga, uno dei due mezzi forniti per interagire col sistema. Per impartire un comando bisogna digitarne il nome seguito dagli eventuali parametri. I nomi dei comandi, la loro sintassi e i nomi dei loro parametri sono simili a quelli degli altri sistemi operativi, ma non seguono con precisione nessuno degli standard stabiliti; ad esempio esiste il comando dir per la visualizzazione dell'elenco dei file contenuti in una directory, ma i parametri che ne cambiano il comportamento sono parole di senso compiuto ("ALL" per una lista ricorsiva dei file a partire da una determinata directory, "FILES" per avere l'elenco dei soli file, ecc...) che vanno collocate dopo il nome della directory di cui visualizzare il contenuto. E' possibile avere un elenco sintentico dei parametri di un comando mettendo il carattere "?" come parametro: dopo aver mostrato questo aiuto il comando non termina ma rimane in attesa che l'utente specifichi i parametri che intende utilizzare e prema il tasto Return. Un'altra caratteristica particolare è che, a partire dalla versione 2.0 del Kickstart, non c'è bisogno di usare il comando "cd" per il cambio della directory corrente, anche se il comando esiste: basta indicare il nome della directory in cui ci si vuole spostare, come se fosse il nome di un comando.

Ho detto che la directory in cui vanno collocati i comandi della CLI è quella assegnata a C:: tutti i comandi elementari del CLI, come ad esempio cd o quelli che hanno senso solo se usati in uno script (come if, else, lab; vd. par. 2.3.11) non sono fisicamente in C:, ma residenti nella memoria ROM del Kickstart. E' possibile rendere residenti in RAM altri comandi, attraverso il comando resident, a patto che siano scritti in codice rientrante e abbiano l'attributo pure: tutti i comandi di sistema hanno queste caratteristiche. Il comando resident ha senso in quei sistemi che non siano dotati di hard-disk; l'invocazione di un comando non residente richiederebbe la presenza del volume contentente C: in uno dei floppy driver, e questo può essere scomodo e comportare grandi perdite di tempo per lo scambio dei dischetti; rendendo residenti i comandi di uso più comune, come dir o copy, essi vengono automaticamente collocati in RAM (non nel volume RAM:, infatti dir non riporta

la presenza dei comandi resi residenti nel filesystem di `RAM:`) e la loro esecuzione non necessita della presenza del disco di sistema nel driver.

La CLI supporta la redirezione dell'output dei comandi su file e device, utilizzando il carattere ">". Ad esempio, scrivendo:

```
echo "Hello World" >PRT:
```

si ottiene la stampa della stringa tra gli apici. Redirigere l'output di un comando al device virtuale `NIL:` fa in modo che nessun output venga visualizzato.

2.3.11 Script

In Amiga esistono due tipi di script. Il primo tipo di script è fondamentalmente un file di testo ASCII, di cui è stato attivato il bit di protezione "script", contenente una serie di comandi AmigaDOS che una CLI esegue in sequenza; uno script può però essere reso più complesso con i comandi `if`, `else`, `lab`, `skip` e altri, che permettono l'inserimento di salti condizionati e loop. La complessità raggiungibile dagli script Amiga non è paragonabile a quella delle potentissime shell Unix come Bash o Csh, è comunque superiore a quella dei file batch di MS-DOS, ed è sufficiente per gli utilizzi più comuni.

L'altro tipo di script è **ARexx**, la versione Amiga di *REXX*, un linguaggio di scripting sviluppato da *IBM*. Uno script ARexx è in grado di pilotare le applicazioni Amiga predisposte, automatizzandone l'utilizzo per svolgere operazioni complesse e ripetitive. L'interprete ARexx, incluso nel Workbench, non viene caricato in memoria di default, ma può essere inizializzato in ogni momento.

2.3.12 Intuition

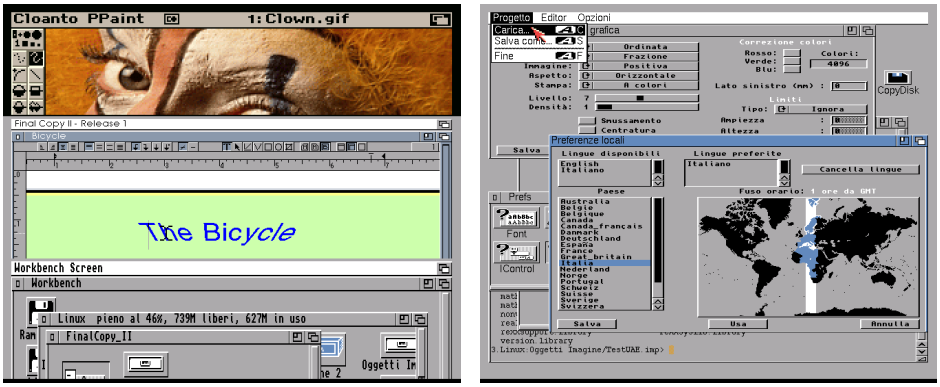
Intuition è la parte del sistema operativo Amiga che si occupa della gestione dell'interfaccia grafica.

Intuition è un classico sistema a finestre comandato da un puntatore mosso dal mouse; nel descriverne le funzionalità do per noti gli elementi tipici di ogni interfaccia grafica e pongo l'attenzione in particolare sui suoi elementi caratteristici.

Attraverso la libreria *intuition.library* vengono messe a disposizione del sistema e delle applicazioni una serie di oggetti prefabbricati con cui costruire un'interfaccia con l'utente uniforme e di facile utilizzo. Ogni programma ha una o più finestre nelle quali sono collocati gli elementi dell'interfaccia, come pulsanti, barre di scorrimento e checkbox, che nel gergo Amiga sono chiamati *gadget*. In ogni momento c'è una ed una sola finestra attiva, che è quella che riceve gli input da tastiera: la finestra attiva si distingue dalle altre per il colore del bordo e della barra del titolo; una finestra si attiva facendo clic al suo interno col mouse. E' da notare che, al contrario di quanto succede nelle più famose interfacce a finestre, come *Windows*, *MacOS* o *KDE*, la finestra attiva non è necessariamente in primo piano, può invece essere coperta da altre finestre, e cliccarne una parzialmente coperta la attiva ma non la porta in primo piano. Per portare una finestra in primo piano c'è un apposito gadget, che viene da alcuni chiamato "*multitasking gadget*" (il nome corretto è *gadget di profondità*), collocato sulla parte destra della barra del titolo accanto al pulsante di ridimensionamento, che alterna le dimensioni della finestra tra due impostazioni predefinite; il pulsante di chiusura delle finestre si trova invece sulla parte sinistra della barra del titolo.

L'area che contiene tutte le finestre è a sua volta una particolare finestra detta *schermo*, che ha le dimensioni dell'intera area visibile e non è ridimensionabile. La barra del titolo è utilizzata di volta in volta come barra di stato del programma che ha la finestra attiva. Una caratteristica peculiare di Intuition è che in ogni momento possono essere aperti più schermi, ciascuno con la propria risoluzione, profondità dei pixel e palette; è possibile passare da uno schermo all'altro usando il multitasking

gadget posto sulla barra del titolo, è inoltre possibile **trascinare** la barra del titolo di uno schermo per abbassarlo e *mostrare lo schermo sottostante*. Uno schermo può in teoria essere anche più grande dell'area effettivamente visibile ad una data risoluzione; ad esempio uno schermo può essere in risoluzione 640x256, ma avere dimensioni di 1000x1000 pixel, se si dispone di sufficiente memoria Chip: per raggiungere le aree non visibili basta spostare il puntatore del mouse sui bordi dell'area visualizzata in quel momento, per fare "scrollare" lo schermo.



Intuition: a sinistra, tre schermi di diverse risoluzioni sovrapposti;
a destra, una vista d'insieme delle possibilità di Intuition

Il mouse dei sistemi Amiga ha due pulsanti: quello sinistro si usa per "fare clic", quindi per attivare gli elementi dell'interfaccia, ridimensionare le finestre e trascinare le icone, con le stesse modalità di tutte le più famose interfacce grafiche. Premendo e tenendo premuto il tasto destro del mouse invece, la barra del titolo dello schermo in cui risiede la finestra attiva cambia contenuto e mostra le intestazioni dei menu a tendina del programma attivo; per scorrere il contenuto dei menu e selezionare un comando bisogna portare il puntatore sulle intestazioni, sempre tenendo premuto il tasto destro. Una volta posizionato il puntatore sul comando desiderato, rilasciando il tasto destro del mouse esso viene eseguito.

Oltre a quelli di finestre e gadget, Intuition dispone delle primitive di strutture più complesse, chiamate *Autorequest*: utilizzando una sola funzione dell'*intuition.library* è possibile invocare finestre di dialogo e per la selezione di un file, con un vantaggio sia per il programmatore, che trova queste strutture già pronte e non deve costruirsele da solo, e per l'utente, che ritrova in molti diversi programmi la stessa interfaccia alle operazioni più comuni.

2.3.13 Workbench

Il Workbench è il **file-manager** di Amiga, che fornisce l'accesso a tutte le sue funzioni, sfruttando appieno le possibilità di Intuition. Lo schermo del Workbench contiene le **icone** di tutti i volumi, fisici e virtuali, che sono montati nel sistema. Il menu contiene una serie di comandi per agire sulle finestre, sulle icone e sulle risorse da esse rappresentate.

Le icone si selezionano con un clic del mouse e si attivano con un doppio clic. Attivando l'icona di un volume si apre una finestra che ne mostra il contenuto. Al contrario di quanto succede con i file-manager più famosi, come Explorer, **non tutti** i file presenti in una directory vengono mostrati nella relativa finestra; di una directory o di un file viene visualizzata l'icona solo se nella stessa directory in cui si trova l'oggetto esiste un altro file, con lo stesso nome più l'estensione "`.info`" finale. I file `.info` sono file speciali che contengono le informazioni sulla forma dell'icona e sul modo di attivazione del file a cui sono associati. Per avere l'elenco completo dei file in una directory, e per poterli manipolare, bisogna usare la CLI o, in alternativa, selezionare un apposito comando dal

menu del Workbench, che lo forza a mostrare tutti i file, usando un'icona standard per quelli che non hanno un file `.info`. Non esiste un comando per creare un file `.info` per un file che non l'ha già: in genere le `.info` di file contenenti dati vengono create dalle applicazioni che producono quei dati; altrimenti, quello che si può fare è copiare un file `.info` già esistente, dandogli il nome della risorsa alla quale si vuole assegnare l'icona.

Un menu permette di agire sulle icone, copiandole, cancellandole, cambiandone il nome. Il comando "*Show informations*" (o "*Informazioni*" nella versione italiana) mostra appunto le informazioni su un file: il nome, le dimensioni, i flag di protezione. Le informazioni sulle icone di file non eseguibili comprendono anche un campo con il **programma associato**, ovvero il programma che viene eseguito quando l'icona viene attivata, e che serve per accedere alle informazioni contenute nel file. Quindi non è l'estensione a determinare quale programma utilizzare per leggere un determinato file, come accade in Windows, e nemmeno un meccanismo come il *Magic* di Linux; l'associazione avviene separatamente per ciascun file: diversi file dello stesso tipo possono essere associati a programmi diversi. Tra le informazioni c'è inoltre una lista di parametri che cambiano il comportamento che il sistema assume all'attivazione dell'icona: svolgono la stessa funzione dei parametri per i comandi impartiti da command-line.

Intuition supporta il *drag-and-drop* e quindi trascinare le icone in punti particolari dell'interfaccia ha un effetto: ad esempio trascinare le icone di file da una finestra ad un'altra del Workbench, o sopra le icone di volumi o directory provoca la loro copia nella destinazione, mentre portare l'icona di un file di dati sull'icona di un programma provoca l'apertura del file da parte del programma.

Non esiste nel Workbench un equivalente della *Startbar* di Windows 9x, o del *menu "Mela"* del MacOS; l'unico sistema per lanciare un'applicazione è quello di aprire il volume che la contiene e andarne a cercare l'icona tra le directory. In realtà sin dalle prime versioni del Workbench esiste un menu chiamato "*Tools*" (o "*Strumenti*" se il Workbench è localizzato in italiano) che avrebbe proprio la funzione della Startbar, ma con il sistema non è mai stato fornito alcun programma per aggiungere comandi a questo menu; bisogna utilizzare programmi prodotti da terzi per poter aggiungervi delle voci.

La configurazione di tutti gli aspetti del sistema avviene per via grafica attraverso i programmi "Preferences", contenuti nella directory SYS:Prefs; la loro funzione è molto simile a quella delle applicazioni del Pannello di controllo di Windows: ogni programma serve a configurare un particolare aspetto del sistema, dalla stampante alla porta seriale al comportamento di mouse e tastiera.

A partire dalla versione 2.1 il Workbench ingloba un sistema di aiuto in linea ipertestuale chiamato "**AmigaGuide**"; ogni file di guida è un unico file ASCII che contiene il testo con degli appositi tag che distinguono le varie sezioni; alcune parole del testo, indicate da tag, appaiono nel visualizzatore come pulsanti che cliccati portano alla corrispondente sezione nella guida; nel testo possono essere inglobati anche file di immagine, in modo simile a quanto previsto dall'HTML.

2.3.14 Datatype

Un'ulteriore interessante caratteristica del sistema operativo Amiga, introdotta col Kickstart 3.1, è costituita dai **Datatype**; essi possono essere paragonati ai "*plug-in*" dei browser più sofisticati, *Netscape* ed *Explorer*, con la differenza che le loro funzionalità vengono fornite non ad un singolo programma, ma potenzialmente a tutte le applicazioni Amiga: si tratta infatti di moduli che vanno collocati in un'apposita directory contenuta in `DEVS:` e che forniscono ai programmi compatibili con questa feature la possibilità di leggere e scrivere i più disparati formati di file. Un esempio applicativo è dato dal tool *Multiview*, compreso nel Workbench 3.1: si tratta di un visualizzatore universale, in grado di aprire e mostrare ogni file il cui formato sia descritto in uno dei Datatype installati nel sistema. Ad esempio, insieme col Workbench 3.1 sono forniti Datatype per il formato *AmigaGuide*

e per i file immagine *IFF ILBM*, ma su Internet (ad esempio nell'archivio *Aminet*) sono disponibili Datatype per *JPEG*, *GIF*, *PostScript*, *Rich Text* ecc... La sola aggiunta di questi file estende perciò le capacità di Multiview e di tutti i programmi scritti per utilizzare la *datatype.library*.

3 UAE

3.1 Il sistema "ospite"

Il sistema utilizzato durante le prove è un PC IBM compatibile che monta il seguente hardware:

- Scheda madre ATX con chipset SiS 735
- Processore AMD Duron 700
- 128 MB di RAM a 133 MHz
- Scheda video S3-Trio
- Scheda audio SoundBlaster PCI128

Il sistema operativo usato è Linux, installato dalla distribuzione Red-Hat 7.1, con kernel 2.4.2-2. Il sistema viene utilizzato mediante interfaccia grafica KDE 2, in uno schermo con risoluzione 1024 x 768 a 64000 colori. La partizione di swap è di 512 MB.

3.1.1 Il sistema di confronto

Per testare l'efficacia dell'emulazione, UAE è stato confrontato con un sistema reale, un A500 con Kickstart 2.0, 512 KB di Chip RAM e 1.5 MB di Fast RAM.

3.2 Installazione

La forma preferita per la distribuzione delle ultime versioni di UAE è quella di tar-ball contenente il codice sorgente. L'installazione consiste quindi nella compilazione dei sorgenti, che avviene in modo automatico grazie ai classici script di autoconfigurazione forniti nell'archivio.

Entrando nel dettaglio, i passi da seguire per avere sul proprio sistema un'installazione di UAE funzionante sono:

Scaricare l'archivio più recente dal sito di UAE, all'indirizzo http://www.freiburg.linux.de/~uae/;

"Scompattare" l'archivio in una directory opportuna (`/usr/src` è la scelta più logica);

Fare cd nella directory in cui è stato scompattato l'archivio;

Lanciare, con i privilegi di root, lo script di auto-configurazione del processo di compilazione, usando: `./configure`
Se nel sistema sono presenti tutte le librerie necessarie ed ovviamente il compilatore C adatto, il processo di autoconfigurazione viene portato a termine con successo in pochi secondi.

Lanciare lo script di compilazione dando il comando: `make`
Al termine della compilazione nella directory corrente si troverà il file eseguibile "uae".

Copiare o linkare il file in una directory opportuna (`/usr/X11R6/bin/uae` è la directory da me scelta).

Da questo momento in poi i privilegi di root non sono più necessari. In realtà è necessario essere root per poter sfruttare le librerie grafiche che consentono il funzionamento dell'emulatore a pieno schermo, ma ricordando il principio del minimo privilegio ed il fatto che stiamo usando un software in versione sperimentale e quindi potenzialmente instabile, è conveniente rinunciare a questa caratteristica ed eseguire il programma come normale utente.

3.2.1 Versioni del software

Ogni versione di UAE è caratterizzata da un numero di versione, composto da tre numeri separati da un punto. Il primo numero è 0, ad indicare che il programma, sebbene già funzionante ed utilizzabile, è ancora da considerarsi nella fase beta dello sviluppo, perciò non completamente affidabile e non adatto ad utilizzatori inesperti; il secondo numero indica che la versione è stabile (*stable*) se il numero è dispari, e sperimentale (*development*) se il numero è pari; l'ultimo numero viene incrementato ad ogni versione del software pubblicata.

Al momento in cui scrivo la versione stabile più recente di UAE è la 0.7.6, mentre quella sperimentale è la 0.8.20. Nella home-page del programma e in tutti i file di documentazione viene comunque specificato chiaramente che in realtà le ultime revision della versione 0.8 sono più stabili e meglio funzionanti della versione 0.7.6, e sono perciò in ogni caso preferibili a quest'ultima. La versione usata nelle prove è perciò la 0.8.20.

3.2.2 Modalità alternative di installazione

In alternativa all'archivio di codice sorgente, è possibile scaricare dalla home-page il software già compilato, in un normale archivio *gzip* o in pacchetto *RPM*. In questi casi però la versione disponibile per il download non è la più recente; inoltre non viene inclusa la documentazione dell'emulatore, indispensabile per l'utilizzo del programma.

3.3 Hardware reale e controparti emulate

3.3.1 Kickstart e floppy-disk

I sistemi Amiga originali leggono il Kickstart da una ROM collocata sulla scheda madre, ed il software da dischetti in formato 3,5 pollici attraverso il lettore di dischetti. I dischetti sono normalissimi floppy-disk da 3.5 pollici del tipo DS-DD (Double Sided - Double Density), ma il file system usato li rende assolutamente illeggibili dalle unità a disco con cui sono equipaggiati gli odierni PC, a meno di utilizzare hardware particolare. Questo perchè i dischetti in formato MS-DOS sono formattati in una suddivisione a tracce o cilindri, a loro volta divisi in settori, e tra i settori viene lasciato un gap non utilizzato per la memorizzazione; i dischi in formato AmigaDOS hanno 80 cilindri ma i settori in cui sono suddivisi sono contigui, e questo li rende illeggibili perchè i controller delle unità a disco dei PC non prevedono un simile formato. E' invece possibile leggere i dischetti in formato MS-DOS dall'Amiga, utilizzando l'apposito device "*CrossDOS*" presente dalla versione 2.1 del Workbench, oppure software prodotto da terze parti per le versioni precedenti.

Non potendo leggere il software Amiga direttamente dai dischetti originali, l'emulatore legge delle immagini dei dischetti, riprodotte in file in formato **ADF**. Un apposito programma AmigaDOS incluso nell'archivio di UAE (*transdisk*) permette la produzione di questi file a partire dai dischetti originali. I file ADF così prodotti possono essere poi trasferiti su PC o per mezzo di dischetti in

formato MS-DOS, oppure attraverso un collegamento null-modem seriale o con cavo Lap-Link; nel primo caso bisognerà "splittare" ogni file ADF in due parti, visto che ogni dischetto Amiga ha una capienza di 880 KB e i file ADF, essendo delle immagini, hanno esattamente queste dimensioni, perciò non entrano sui dischetti MS-DOS da 720 KB, gli unici che Amiga riesca a leggere; nel secondo caso occorre procurarsi del software apposito, e trasferirlo su Amiga.

I file ADF sono però perfettamente inutili se non si dispone del contenuto delle ROM del Kickstart. Per ottenere questi dati viene fornito il programma *transdos*; i file prodotti potranno essere trasferiti sul PC con le stesse modalità dei file ADF, con il vantaggio che le ROM Amiga hanno una capienza di 512 KB, e perciò le loro immagini trovano posto in un solo dischetto MS-DOS a bassa densità. Un'alternativa è l'acquisto di un CD-Rom chiamato "*Amiga Forever*", prodotto da *Cloanto*, che include appunto i file immagine delle ROM, oltre ad alcuni emulatori, UAE compreso, e una cospicua dotazione di software. Procurarsi le immagini delle ROM in qualsiasi altro modo, ad esempio scaricandole da Internet, è un'azione illegale, perchè il software contenuto nelle ROM è ancora sotto copyright e, per scelta della Amiga Inc., detentrice attuale dei diritti, solo Cloanto ha diritto di distribuirlo. Ovviamente non è impossibile trovare comunque questi file in Rete!

Lo stesso discorso varrebbe naturalmente per il software vero e proprio, ma in questo caso è possibile trovare un gran numero di programmi, in particolare videogiochi, che gli autori hanno consentito di distribuire liberamente. Emblematico è il caso della software house "*Factor 5*", che esordì nel 1987 proprio con una serie di giochi per Amiga, e che oggi è ancora attiva con prodotti per PC e per le console più diffuse. Factor 5 nella propria home-page rende disponibili per lo scaricamento alcuni dei propri giochi classici, aggiungendo un esplicito ringraziamento agli autori di UAE, che con questo emulatore mantengono vive le radici di questa storica software-house.

3.3.2 Hard-disk

UAE emula gli hard-disk di Amiga in due diversi modi.

Il primo modo è l'hardfile, un file immagine di un filesystem originale Amiga; esso viene letto per mezzo di un device appositamente scritto, chiamato uaehf.device. E' concepito per permettere agli utilizzatori di sistemi Amiga con hard-disk di "trapiantare" il proprio sistema su un PC con UAE semplicemente creandone un'immagine fisica e trasferendola sull'hard-disk del PC. In questo modo si conserva la struttura della tabella delle partizioni e del filesystem con gli attributi dei singoli file. E' comunque possibile creare da zero un file di dimensioni opportune da usare come hardfile, ad esempio con il comando:

```
$ dd if=/dev/zero of=hardfile bs=512 count=64k
```

In questo modo si crea un file di 32 MB (65536 blocchi da 512 bytes) di tutti zero, di nome appunto "hardfile". Bisogna poi impostare UAE per riconoscere il file come hard-disk, indicandone opportunamente la geometria, specificandone il numero di tracce, di testine e la dimensione dei singoli blocchi. Nel caso di un hardfile creato come immagine di un hard-disk Amiga, scegliendo correttamente i parametri della geometria esso potrà lessere letto da UAE. Nel caso di un hardfile creato ad-hoc, qualunque geometria andrà bene, ma scegliere una dimensione dei blocchi troppo grande porta ad un accesso più lento al disco virtuale e un maggior spreco di spazio. Nel mio caso ho utilizzato la seguente impostazione:

```
hardfile=rw,32,1,2,1024,path/dell/hardfile
```

che significa che il file hardfile viene visto come un hard-disk non protetto dalla scrittura (rw), con 32 tracce, una testina, due blocchi riservati e una dimensione dei blocchi di 1024 bytes.

A questo punto all'interno dell'emulatore l'hardfile si gestisce come un normale hard-disk; utilizzando le utility di sistema dell'AmigaDOS si procede alla formattazione del disco, dopodichè diventa utilizzabile per copiarvi dati ed installarvi applicazioni. Da Linux, se il kernel, versione almeno 2.0.5, è stato compilato in modo opportuno e se è presente il relativo modulo, è possibile montare l'hardfile come filesystem Amiga (sotto Linux viene indicato col nome "*AFFS*", *Amiga Fast File System*) in modalità loop, e in questo modo accedere ai file contenuti nell'hard-disk virtuale anche senza lanciare l'emulatore. Il comando da dare può essere:

```
# mount path/dell/hardfile /mnt/mountpoint -t affs -o loop
```

Il secondo modo per emulare un hard-disk in UAE è quello di indicare all'emulatore una directory del proprio filesystem, da utilizzare come hard-disk virtuale. In pratica la directory viene condivisa tra il sistema operativo ospite e quello emulato. Questo secondo modo offre alcuni vantaggi su quello che usa l'hardfile: non viene sprecato spazio sul disco, mentre con la modalità hardfile lo spazio visto come libero da Amiga è comunque occupato sul filesystem ospite. Inoltre, entrambi i sistemi possono accedervi direttamente, e questo è un vantaggio nel caso si abbiano file sul proprio disco che si vuole far vedere all'emulatore: basta copiare i file nella directory prescelta. Infine, la gestione dell'hardfile un UAE presenta alcuni bug noti: nella documentazione del programma si consiglia esplicitamente di utilizzare l'emulazione dell'hard-disk da filesystem in ogni caso. Uno di questi bug è stato incontrato durante le prove ed ha portato a risultati spiacevoli.

Perchè UAE possa vedere una directory del proprio sistema come volume ed accedervi, bisogna indicargli il path della directory ed il nome che esso assumerà per l'Amiga. L'impostazione da me usata è:

```
filesystem=rw,Linux:/home/macman/amiga/amigafs/
```

L'interfaccia di UAE offre la possibilità di aggiungere nuove directory all'elenco dei volumi visti dall'Amiga emulato, senza dover riavviare l'emulatore; un nuovo filesystem non verrà però visto dal sistema se non dopo un reset.

3.3.3 Monitor

Una finestra sullo schermo di X fa le veci del monitor dell'Amiga emulato. Facendo partire il programma coi privilegi di root e impostando un'apposita opzione sarebbe possibile visualizzare la finestra di UAE a pieno schermo, ma a parte i rischi che l'utilizzo di un'applicazione nella fase beta dello sviluppo con i privilegi di root comportano per l'integrità del proprio sistema, un tentativo di utilizzare questa possibilità effettuato durante le prove ha dato esito negativo. In realtà del pieno schermo non si avverte il bisogno; ad una risoluzione dello schermo di 1024x768 pixel, la finestra di UAE è abbastanza grande da permettere una visione chiara dell'output grafico dell'Amiga emulato, ma anche abbastanza piccolo da lasciare un discreto spazio "di manovra" per operare con la GUI del programma, ma anche con le altre applicazioni di Linux.

La finestra di UAE comprende nella parte inferiore una serie di cinque "led" virtuali, ovvero piccoli rettangoli che durante l'uso del programma possono assumere due sfumature di colore, una più scura ed una più intensa, "simulando" il comportamento di led luminosi: quattro, di colore verde, mostrano in ogni istante la posizione delle testine dei floppy-disk driver virtuali, mediante un numero da 0 a 79, e si "illuminano" quando il loro motore è in funzione; l'ultimo, di colore rosso, riproduce il comportamento del led che sull'Amiga reale indica che il sistema è alimentato.

3.3.4 Mouse e joystick

Gli Amiga reali hanno due porte di comunicazione a 9 pin denominate rispettivamente “joyport 0” e “joyport 1”, e dedicate alla connessione del mouse o di joystick.

Il mouse è una periferica fondamentale per l’utilizzo di Amiga. Come funzionamento esso è identico ad un qualunque mouse a due tasti per PC: UAE utilizza perciò il mouse del PC per emulare il mouse Amiga. UAE per Linux funziona, come visto nel paragrafo 3.3.3, mostrando l’output grafico in finestra, perciò durante il funzionamento dell’emulatore i puntatori del mouse visualizzati sono due: uno è quello del server grafico di Linux, X, l’altro è quello emulato dell’Amiga. Questa situazione può portare a disorientamento e difficoltà di utilizzo del sistema emulato: il movimento del mouse non viene rilevato da UAE se il puntatore di X è fuori dalla sua finestra; d’altro canto i due puntatori del mouse possono avere velocità diverse e quindi percorrere distanze in corrispondenza dello stesso movimento del mouse. Perciò UAE offre la possibilità di bloccare il puntatore del mouse all’interno della finestra dell’emulatore e di nascondere il puntatore di X lasciando in vista solo quello emulato; tale funzione è accessibile con la combinazione di tasti `F12+'g'`. Si tratta di una combinazione volutamente “originale”: l’autore di UAE ha voluto esser certo che essa non si sovrapponesse ad alcuna combinazione di tasti valida nei programmi emulati, con risultati imprevedibili. In questo modo non si corre alcun rischio, visto che le tastiere Amiga hanno solo dieci tasti funzione. La stessa combinazione di tasti sblocca il puntatore, restituendo quello di X, ad esempio per agire sulla GUI dell’emulatore.

Il joystick non è altrettanto indispensabile, ma ovviamente è utile nel caso si voglia utilizzare UAE per giocare; la stragrande maggioranza dei videogiochi richiede infatti un joystick digitale a 4 direzioni e un pulsante di sparo, connesso alla joyport 1. UAE offre la possibilità di utilizzare un joystick reale o di emularlo utilizzando alcuni tasti della tastiera. Allo scopo sono molto comodi i tasti cursore, ma in questo caso la loro funzione si sovrapporrebbe a quella dei tasti cursore della tastiera Amiga emulata; perciò UAE offre due alternative. Comunque sia in ogni momento attraverso la GUI è possibile modificare le impostazioni riguardanti i joystick in ogni momento durante l’emulazione. Come extra UAE offre una funzione di “autofire”, che quando attivato preme e rilascia velocemente il pulsante di sparo del joystick virtuale: è una funzione che trova la sua utilità in alcuni videogiochi.

3.3.5 Sonoro

Il sonoro dell’Amiga emulato viene naturalmente riprodotto dalla scheda audio del sistema, passando attraverso il server audio e il device `/dev/dsp`. Anche le più umili tra le schede sonore oggi disponibili (la SoundBlaster che equipaggia il mio sistema può essere inserita in questa categoria) sono in grado di riprodurre senza alcun problema il sonoro a 8 bit stereo degli Amiga originali, a patto che il server audio di Linux sia funzionante e correttamente configurato.

3.4 Struttura e funzionamento del programma

UAE è un programma *Open Source* e, come detto nel paragrafo 3.2, la forma preferita per la sua distribuzione è il codice sorgente. Questo però purtroppo non rende più facile capire a fondo come funziona internamente questo programma: come se non bastasse la sua intrinseca complessità e l’intricatezza dei riferimenti incrociati tra i diversi moduli, inevitabili dovendo emulare un’architettura così sofisticata, il codice è anche assai poco commentato e perciò di difficile lettura per chi non sia un ottimo conoscitore dell’hardware Amiga. E’ comunque possibile dedurre quali sono i moduli più importanti ed estrapolarne i principi di funzionamento.

Bisogna innanzitutto dire che il codice è concepito per favorirne la portabilità sulle più diverse piattaforma hardware, ed infatti esistono versioni di UAE per Windows e MacOS, ma anche per Alpha e persino per Amiga; quest'ultima versione è pensata per i modelli più recenti di questi computer, che non montano il chipset originale e non possono perciò far girare i vecchi videogiochi nativamente. La ricerca della portabilità porta a volte a costruzioni inconsuete, un esempio può essere il seguente:

```
void start_program (void)
{
    do_start_program ();
}

void leave_program (void)
{
    do_leave_program ();
}
```

Il tentativo è quello di mantenere quanto possibile una struttura coerente del codice tra i vari porting: le funzioni `do_start_program()` e `do_leave_program()` esisteranno e avranno la stessa forma in tutte le versioni, mentre `start_program()` e `leave_program()` potranno includere all'occorrenza altre righe di codice.

L'intento della portabilità porta anche allo sforzo di scrivere codice il più possibile indipendente dal sistema operativo sottostante il programma. La versione per Linux fa esplicito riferimento alle librerie del server grafico *X11*, che forniscono le primitive per la scrittura della grafica a video ma anche per l'intercettazione della pressione dei tasti e dei movimenti del mouse, e a due *include* ed un modulo C, per la gestione dei joystick, scritte appositamente per questo sistema operativo; per il resto utilizza librerie standard dell'*ANSI C* e altre librerie diffuse (ad esempio per la GUI e per la gestione dei file compressi con *gzip*).

Il modulo principale è `main.c`: le sue funzioni sono:

- dichiarare una serie di costanti che verranno usate in tutto il resto del codice;
- effettuare il parsing del file di configurazione e della command-line per impostare le diverse opzioni;
- attivare gli altri moduli del programma (grafica, sonoro, emulazione del processore, dei floppy-disk, del chipset);
- infine lanciare l'emulazione.

Al lancio il programma esegue 10000 cicli a vuoto e misura il tempo impiegato dal sistema per completarli, usando questo parametro come misura approssimativa della velocità della CPU. In seguito vengono attivati il sonoro e l'emulazione dei joystick, e costruite in memoria le strutture del 68000 e dei chip custom. A questo punto viene avviata l'emulazione vera e propria.

Il cuore dell'emulatore è l'interprete del codice macchina del 68000. Emulando il comportamento del processore reale, l'interprete effettua il fetch del codice che risiede alla locazione di memoria puntata dal registro *PC* (*Program Counter*); il 68000 è un processore *CISC* e le sue istruzioni hanno lunghezze diverse: è perciò spesso necessario effettuare più di un fetch per avere il comando completo da eseguire. L'istruzione prelevata viene disassemblata, ossia il codice macchina viene tradotto nel relativo opcode mnemonico: per far questo il codice viene confrontato con una *look-up table*, generata all'inizializzazione del programma, che contiene tutte le istruzioni del 68k, in questa forma:

```
0001 DDDd ddss sSSS:00:-NZ00:-----:12: MOVE.B s,d[!Areg]
0010 DDDd ddss sSSS:00:-----:-----:12: MOVEA.L s,d[Areg]
```

Una volta ottenuto l'opcode dell'istruzione da eseguire, UAE esegue la relativa function tra quelle contenute nel modulo `cpuemu.c`; questa sequenza va eseguita per ogni istruzione del 68k, con un overhead stimato di ben 35 cicli di clock per istruzione effettivamente eseguita. Se si considera che su Amiga gran parte del lavoro è effettivamente svolto dai chip custom, questa informazione può far prevedere prestazioni dell'emulatore non entusiasmanti, come in effetti risulterà dalle prove.

Dopo ogni esecuzione il programma controlla se sono stati generati eventi, che vengono posizionati in una coda, e a quale ciclo di clock del processore virtuale debbano essere eseguiti; aggiorna il contatore dei cicli di clock aggiungendo il numero di cicli necessari per l'ultima esecuzione, e confronta il risultato col tempo a cui devono avvenire gli eventi, decidendo se gestirli o lasciarli in coda per eseguire il successivo fetch. In questo modo avviene la sincronizzazione tra 68000 e chip custom.

Tutte le operazioni hardware avvengono quando il processore accede alle locazioni di memoria che corrispondono all'hardware stesso (*I/O memory mapped*). La memoria, gestita attraverso le funzioni del modulo `memory.c`, viene allocata da UAE in banchi di 64 KB, ciascuno dei quali è caratterizzato da puntatori a funzioni per l'utilizzo del banco stesso: Per l'accesso ad un determinato banco il programma utilizza queste funzioni: i banchi che virtualizzano la memoria RAM potranno essere acceduti direttamente in lettura e scrittura: quelli che invece risiedono agli indirizzi dei chip custom o delle periferiche di I/O genereranno degli eventi che l'emulatore raccoglierà nella propria coda e gestirà in sequenza.

Le funzioni che gestiscono questi eventi, ovvero che eseguono i compiti richiesti dal processore con l'accesso alla relativa locazione in memoria, risiedono nei diversi moduli di UAE: `custom.c` è il colossale modulo che riproduce le operazioni svolte dai chip custom; esse vengono eseguite usando le funzioni contenute in `disk.c` per l'accesso ai dischi floppy, e `xwin.c` per quanto riguarda le scritture a video, oltre naturalmente ad altri moduli per la gestione dei restanti aspetti dell'emulazione (hardfile, filesystem, suono, ecc...). `xwin.c` si occupa anche della traduzione della pressione dei tasti della tastiera del PC nei relativi tasti della tastiera Amiga e nei movimenti dei joystick.

UAE fa un utilizzo limitato dei *thread* in spazio kernel di Linux: il principale scopo è impedire il blocco dell'intera applicazione alla richiesta al sistema operativo di I/O di periferiche, come i lettori di CD-Rom. Al momento questa feature è ancora in fase sperimentale e più che apportare benefici rallenta il sistema, come indicato anche dall'autore nella documentazione.

3.4.1 JIT

Il fatto di essere *Open Source* porta grandi benefici al progetto UAE: chiunque sia in grado di comprendere approfonditamente il funzionamento del codice, e conosca perfettamente l'hardware Amiga, è libero di apportare miglioramenti al programma, a patto che spedisca le patch all'autore perchè le renda disponibili per tutti. Nelle intestazioni dei vari moduli sono indicati numerosi collaboratori per i diversi aspetti del programma.

Uno dei miglioramenti più interessanti sta nel **Just In Time compiler** (*JIT*), che intende ovviare ai problemi di performance dell'emulatore. L'emulatore standard esegue in sequenza e "ciecamente" il codice macchina: per il *principio di località* esso si troverà, con molta probabilità, a ripetere le stesse brevi porzioni di codice molte volte, e per ogni passaggio dovrà ripetere interamente tutte le operazioni di fetch ed interpretazione prima di eseguire effettivamente i comandi.

Il principio del JIT è invece quello di scansire il codice macchina e riconoscere le porzioni (chiamate **blocchi**) che verranno eseguite a ripetizione. Il riconoscimento avviene identificando i

comandi di salto, condizionato o incondizionato, all'indietro nella sequenza di opcodes. Ad esempio si può osservare la seguente porzione:

```
          mov.l 0x42000000,A0
loop:     mov.l (A1+),(A0+)
          cmp.l 0x42010000,A0
          bne   loop
```

In essa si trova un salto condizionato, comandato dall'istruzione `bne`, all'indietro, alla riga identificata dalla label `loop`. E' perciò piuttosto facile riconoscere, anche in modo automatico, nelle ultime tre righe di codice un blocco, che verrà ripetuto diverse volte.

Con l'emulazione standard ogni loop del blocco comporterebbe approssimativamente 105 cicli di clock del processore ospite; di questi una buona percentuale è utilizzata per il riconoscimento dell'istruzione, non per l'esecuzione. Il JIT invece riconosce il blocco, e interpreta in un solo passaggio i tre comandi che lo compongono, mettendo il risultante codice per la macchina ospite in una parte della memoria che funge da cache. A questo punto è questo codice in forma nativa per il processore ospite che viene eseguito; l'overhead causato dal riconoscimento del blocco è ampiamente assorbito dal considerevole risparmio sui cicli utilizzati per eseguire il blocco stesso. Con un'opportuna gestione della cache del codice tradotto è possibile conservare i blocchi più recentemente utilizzati, facendo affidamento ancora sul principio di località, che garantisce che con una certa probabilità tali blocchi verranno riutilizzati.

Il JIT per UAE è in stato ancora primitivo, infatti è stato impossibile far funzionare una versione dell'emulatore (0.8.15) comprendente la patch che implementa il JIT. L'autore di UAE non ha ancora incluso il JIT nella distribuzione ufficiale, proprio per questo motivo. Comunque, la versione attualmente disponibile del porting per Windows di UAE è dotato di una versione riveduta e corretta del JIT che è possibile utilizzare. Alcune prove hanno rivelato l'immaturità del JIT, che risulta incompatibile con la maggior parte del software Amiga, correttamente eseguito dalla versione standard di UAE; le applicazioni *CPU-intensive* che funzionano hanno però mostrato un enorme incremento nella velocità di esecuzione. La differenza di prestazioni tra le due versioni di UAE, con e senza JIT, è invece molto meno evidente nei programmi che utilizzano intensamente i chip custom: l'emulazione delle funzioni da essi svolte non trae benefici dal JIT, per cui rimangono un collo di bottiglia.

3.5 File di configurazione

Un programma di tale complessità ha ovviamente un gran numero di opzioni e parametri da regolare per adattare il software al proprio sistema e farlo funzionare al meglio.

Come comportamento di default UAE legge i parametri di funzionamento dal file `$HOME/.uaerc`, il cui formato è molto semplice: per ogni riga è indicata un'impostazione, con la sintassi :

`parametro=valore`

Alcuni di questi parametri possono essere cambiati anche utilizzando la GUI del programma, ma altri sono impostabili solo attraverso il file di configurazione e vanno perciò specificati prima dell'avvio del programma. I più importanti sono riportati in questa tabella:

Parametro	Descrizione	Valore utilizzato
config_description	E' una semplice descrizione del file di configurazione	UAE default configuration

Parametro	Descrizione	Valore utilizzato
use_gui	Indica se si desidera usare la GUI. La GUI è molto utile ed suo utilizzo non ha alcuna controindicazione: perciò la attivo sempre.	yes
sound_output	Il sonoro può essere disabilitato (none), attivato (normal), attivato con la massima precisione (max) e infine disattivato ma calcolato. L'opzione normal è adatta alla stragrande maggioranza dei casi.	normal
sound_channels	Si può scegliere un suono mono, stereo o mixed, che rimedia al fastidioso effetto dei due canali audio completamente indipendenti mixando parzialmente i due canali, ma comporta un maggior peso per il processore.	stereo
sound_bits	Bit sonori utilizzati per l'audio (8 o 16)	16
sound_max_buff	Buffer dedicato al suono. Valori maggiori comportano una minor probabilità di avere un sonoro a "scatti", ma comportano un lieve ritardo del suono stesso rispetto al resto dell'emulazione, avvertibile in particolare nei giochi.	4096
sound_frequency	Frequenza dell'output sonoro.	44100
sound_interpol	Per una maggiore purezza del suono è possibile utilizzare una routine di interpolazione, che però comporta un maggior carico del processore. Il suono è comunque soddisfacente anche senza interpolazione.	none
joyport0	Indica quale periferica del PC verrà usata per simulare la periferica connessa alla porta 0 dell'Amiga. In genere alla porta 0 viene connesso il mouse.	mouse
joyport1	Indica quale periferica del PC verrà usata per simulare la periferica connessa alla porta 1 dell'Amiga. kbd1 indica che verranno usati i tasti del tastierino numerico del PC per emulare un joystick digitale.	kbd1
gfx_framerate	L'emulatore manterrà un framerate pari a 1 fratto il valore indicato, rispetto ad un Amiga reale. Un valore di 2 richiede perciò un framerate pari a 1/2 del framerate reale, che è di 50 frame/sec.	2

Parametro	Descrizione	Valore utilizzato
gfx_linemode	Gli Amiga reali alle risoluzioni non interlacciate tracciano una riga video alternata ad una riga nera. Se gfx_linemode è impostato a “scanline“, UAE riproduce questo effetto. Le alternative sono “double“, che raddoppia ogni riga orizzontale, con un output grafico migliore ma un carico di lavoro doppio per il server grafico, oppure “none“, che non lascia alcuno spazio tra una riga orizzontale e l’altra col risultato di avere una schermata schiacciata in senso verticale (molto brutta a vedersi).	scanline
gfx_colour_mode	Permette di impostare la profondità di colore desiderata per la finestra grafica. Il valore “8bit“ è in assoluto la più veloce ed è perciò consigliata. Altri valori possibili sono “16bit“, “32bit“, oppure “8bitdithered“ che effettua anche un dithering dell’output.	8bit
chipset	Permette di scegliere il chipset da emulare. Sono disponibili i tre chipset Amiga, ma l’AGA è implementato in uno stato ancora embrionale e risulta molto lento.	ocs
collision_level	Amiga implementa un meccanismo hardware che segnala quando avvengono collisioni tra due sprite o tra sprite e playfield. UAE implementa questo meccanismo, ma è piuttosto pesante e rallenta visibilmente il sistema. L’opzione “sprite“ limita la collision detection alle sole collisioni tra sprite, che è sufficiente per la maggior parte dei giochi.	sprite
fastmem_size	Dimensione in MB della Fast Memory che verrà “montata“ sull’Amiga emulato.	4
chipmem_size	Dimensione della Chip RAM dell’Amiga emulato. La memoria emulata sarà pari a 512 KB per il valore indicato. Perciò chipmem_size=2 richiede 1 MB di Chip RAM	2
cpu_speed	Indica che velocità dovrà avere il 68000 emulato, in rapporto col chipset emulato. “max“ impone la massima velocità ottenibile sul sistema ospite. Alcuni giochi però richiedono per funzionare che l’emulatore riproduca esattamente i timing dell’Amiga A500 o A2000. Questo parametro può comunque essere variato al “volo“, all’occorrenza, per mezzo della GUI durante l’emulazione, perciò lascio questo parametro su “max“	max

Parametro	Descrizione	Valore utilizzato
cpu_type	Indica quale processore dovrà essere emulato. Processori più potenti non saranno più veloci, perchè il limite è determinato dalla velocità del sistema "ospite"; i processori successivi al 68000 includono però nuovi opcodes.	68000
kbd_lang	Indica il tipo di tastiera che dovrà essere emulata. La tastiera degli Amiga è molto simile alla corrispondente tastiera per PC nella collocazione dei tasti tipici delle diverse nazionalità. Conviene perciò per comodità settare questo parametro indicando la nazionalità del layout della tastiera del proprio PC.	it
kickstart_rom_file	E' il path al file che contiene il kickstart. UAE implementa alcune basilari funzioni per il caricamento del bootblock per gli ADF di dischi non DOS, ma allo stato in cui si trovano esse sono sufficienti solo per pochissimi dischi demo. Il kickstart è poi indispensabile se si vuole utilizzare dischi DOS ed il sistema operativo.	<path al file>

4 Prove

Per testare l'efficacia e le prestazioni di UAE sono state utilizzate tre tipologie di software:

- **Software applicativo**: è stata testata una vasta gamma di software applicativo, con funzioni che vanno dal semplice mantenimento del sistema operativo, al word-processing, al desktop-publishing, alla composizione di musica, fino ad arrivare al settore in cui i sistemi Amiga sono stati per anni protagonisti, ossia la grafica, sia pittorica 2D che 3D in ray-tracing. In questo modo è stata collaudata l'emulazione del sistema operativo e l'implementazione delle periferiche di memorizzazione, ovvero i floppy-disk driver e gli hard-disk.

- **Giochi**: ho testato un certo numero di giochi non solo per "nostalgia" e divertimento personale, ma anche perchè i videogiochi impegnano l'hardware delle macchine più del software applicativo, essendo programmi che devono gestire in real-time e in perfetta coordinazione grafica, sonoro e i dispositivi di input per risultare giocabili. Bisogna poi ammettere che i giochi sono anche più facilmente reperibili in rete rispetto al software applicativo;

- **Demo**: si tratta di programmi scritti da programmatori Amiga al solo scopo di spingere al limite estremo le possibilità della macchina e provare il proprio valore come "coder". Sono programmi scritti per la maggior parte direttamente in assembler 68k e non hanno alcuna utilità concreta, ma producono a video animazioni ed effetti speciali, come scritte colorate e saltellanti, il classico "plasma" e figure animate pseudo-3D. Proprio perchè spingono al limite l'hardware su cui sono state scritte, queste demo sono i programmi ideali per testare l'efficacia nell'emulazione del chipset dei sistemi Amiga.

4.1 Software applicativo

In generale il software applicativo è stato eseguito correttamente da UAE. Solo un programma ha manifestato dei problemi, che però si risolvono da soli, senza modificare alcuna impostazione, dopo alcuni secondi di utilizzo.

Una nota interessante è data dal fatto che nelle risoluzioni dello schermo interlacciate la finestra di UAE imita uno schermo ad alta persistenza, eliminando il fastidioso flicker della grafica e consentendo perciò di rendere utilizzabili queste risoluzioni. Non viene invece risparmiato all'utente il tempo di attesa del caricamento dei programmi dai dischetti: per motivi di compatibilità UAE emula i movimenti della testina sulla superficie del dischetto, quindi i tempi di accesso ai file ADF si avvicinano a quelli dei floppy-disk driver reali, come dimostrato nel paragrafo 4.1.2. La serie di "led" posti nella parte inferiore della finestra di UAE, che mostrano in tempo reale lo stato dei floppy-disk driver virtuali (vd. par. 3.3.3), permettono di avere un riscontro immediato dei movimenti delle testine; ad esempio si può notare il movimento a vuoto attorno alle tracce più vicine alla 0, che serve per l'auto-mounting dei floppy, oppure il fatto che la prima traccia letta in un disco appena inserito è quella intermedia del supporto, dove risiede il RootBlock (vd. par. 2.3.8), o ancora i seek da una parte all'altra della superficie di un disco molto frammentato.

4.1.1 Software di sistema: Workbench 3.1

Il primo software testato è stato naturalmente il Workbench e le utilità con esso fornite.

La versione del Workbench scelta per la prova è la 3.1, che tra le varie funzioni ha un programma per l'installazione del sistema sull'hard-disk, e la localizzazione dei messaggi in italiano.

Il primo passo è stato quello di creare un file da utilizzare come hard-disk virtuale, per collaudare questa funzionalità. Utilizzando la sintassi indicata anche nel paragrafo 3.3.2 ho prodotto un file vuoto di 32 MB, e l'ho impostato come hard-disk principale dell'Amiga virtuale di UAE.

Ho poi effettuato il boot dalle immagini dei dischi di sistema, ottenendo dopo alcuni secondi di caricamento lo schermo del Workbench. L'hardfile a questo punto del processo viene visto dall'AmigaDOS e mostrato sul Workbench come volume non DOS, perchè non contiene un RootBlock. Per rendere l'hardfile un volume valido bisogna formattarlo, facendo clic sulla sua icona e scegliendo il comando "*Format*" del menu di sistema. Una volta scelti alcuni parametri del filesystem da creare, l'operazione di formattazione vera e propria richiede pochi secondi; non ho modo di confrontare il tempo impiegato con il tempo di formattazione di un hard-disk reale, ma penso di poter dire con un buon grado di sicurezza che è decisamente più breve: al contrario di quanto avviene con i floppy, non c'è motivo per cui il programma debba emulare i ritardi nell'accesso all'hard-disk.

A questo punto è bastato inserire il disco contenente il programma di installazione ed eseguirlo facendo doppio clic sulla relativa icona, effettuare il cambio dei dischetti nei floppy-disk driver quando richiesto, e resettare il sistema al termine dell'operazione, per avere in pochi minuti un Workbench Amiga perfettamente funzionante. Il boot del sistema è praticamente immediato. Le utilità di sistema funzionano tutte senza il minimo problema: solo le utilità per la gestione a basso livello dell'hard-disk, che serve per partizionare i dischi e ha funzioni di diagnostica, risultano inutilizzabili, perchè operano su dischi SCSI che utilizzino la *scsi.device*, mentre l'hardfile viene gestito con una device costruita ad-hoc per l'emulatore, *uaehf.device*.

Anche la CLI funziona perfettamente. E' stata anche installata ed utilizzata la device sostitutiva di `CON:`, *KingCON*, illustrata nel paragrafo 2.3.9. Anch'essa è stata correttamente eseguita dall'emulatore: anzi, le funzionalità aggiunte alla `CON:` standard sono risultate preziose, in particolare l'autocompletamento dei nomi di file e directory è stato indispensabile per operare velocemente sui file attraverso la CLI.

Per comodità UAE è stato configurato per utilizzare una directory del filesystem Linux come secondo hard-disk: utilizzando sia la CLI che il Workbench per operare sui file di questo volume non sono mai stati riscontrati comportamenti anomali che potessero portare a corruzione di file; non è stato possibile uscire dalla directory indicata a UAE come secondo hard-disk, la protezione del filesystem ospite da eventuali applicazioni difettose o virus che possano essere in funzione sull'Amiga emulato risulta perciò efficace. UAE sopporta senza problemi che si operi sul filesystem a lui assegnato contemporaneamente dall'Amiga emulato e dalla shell di Linux: è stato possibile ad esempio scompattare archivi *LHA*, scaricati da Internet e collocati in una directory "privata" della mia home, direttamente nella directory dedicata a UAE, con il Workbench in funzione, ed accedere immediatamente dall'Amiga ai nuovi dati caricati.

Il sistema è risultato stabile; a regime di normale funzionamento nè l'emulatore nè il sistema operativo emulato si sono mai bloccati, ed hanno sopportato senza problemi il mantenimento in esecuzione per diverse ore consecutive, in multitasking con altre applicazioni Linux. L'unico modo per mandare in crisi UAE è cambiare al volo un gran numero di impostazioni dalla GUI, come la quantità di memoria installata o il tipo e la modalità di funzionamento del processore; in particolare UAE è risultato sensibile all'attivazione dell'opzione "*Slow but more compatible mode*" che riguarda il 68000: in molti casi attivando questa funzione l'emulatore ha fatto crash per segment violation.

4.1.2 Operazioni sui dischi: DiskCopy 6.3 e DPU

DiskCopy è un potentissimo programma per la gestione dei floppy-disk; la sua funzione principale è, come chiaramente espresso dal nome, la copia dei dischetti, ma dispone di altre funzionalità come la formattazione rapida, il controllo degli errori presenti nel filesystem, la deframmentazione dei dischi. Lo scopo principale di DiskCopy rimane comunque la duplicazione di floppy-disk non DOS scritti con meccanismi di protezione dalla copia, in particolare videogiochi. Per fare questo non usa le funzioni del sistema operativo ma accede direttamente all'hardware della macchina. E' quindi il programma ideale per testare l'efficacia dell'emulazione dei floppy-disk driver da parte di UAE. Queste le prove effettuate:

- **Formattazione senza verifica**: con questa funzione il programma formatta rapidamente un dischetto. Sia l'A500 che UAE completano l'operazione in esattamente 36 secondi. In entrambi i casi il dischetto è stato poi provato per verificarne l'avvenuta corretta formattazione, copiandovi una directory per un totale di circa 600 KB, e la prova ha avuto successo.

- **Formattazione con verifica**: l'A500 completa la formattazione di un dischetto con verifica dei blocchi scritti in 1 minuto e 12 secondi; UAE invece non riesce ad effettuare l'operazione, riportando sin dal primo blocco gravi errori di scrittura alla verifica. Da notare che lo stesso file ADF è stato poi formattato con successo con l'opzione di formattazione senza verifica, e collaudato in modo approfondito per verificare che non ci fossero errori nel filesystem creato. Evidentemente DiskCopy utilizza in questo caso dei meccanismi di verifica che non sono correttamente implementati in UAE.

- **Controllo**: con questo comando il programma effettua uno scan rapido dei blocchi del dischetto, verificando che siano tutti leggibili. Sia l'A500 che UAE completano l'operazione nello stesso tempo, 37 secondi.

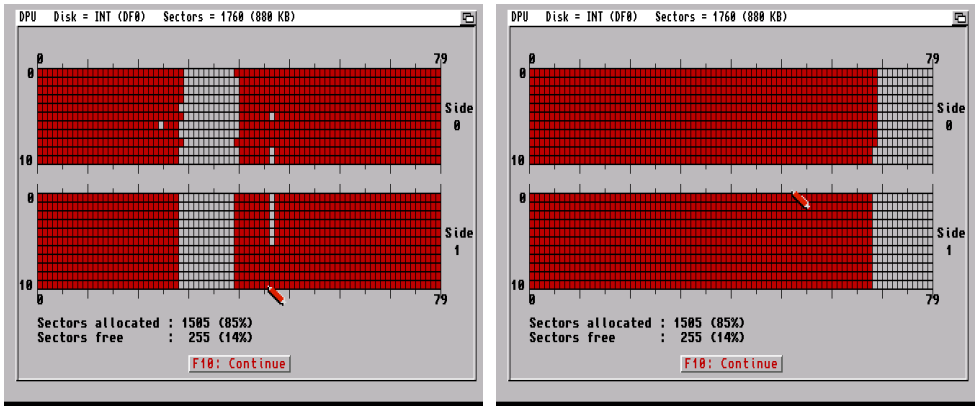
- **Ottimizzazione**: questo comando deframmenta il filesystem di un dischetto, riarrangiando i blocchi in modo che i file risiedano su blocchi consecutivi, diminuendo il tempo necessario alla loro lettura; inoltre effettua una verifica del dischetto per controllare che l'operazione sia

andata a buon fine. Ho effettuato l'ottimizzazione di un dischetto sotto Amiga A500, e della sua esatta immagine ADF sotto UAE. L'A500 completa l'operazione in 2 minuti e 21 secondi, mentre UAE impiega 2 minuti e 3 secondi.

- **Copia con verifica**: ho effettuato la copia di un dischetto AmigaDOS utilizzando sia come sorgente che come destinazione il drive `DF0:`; DiskCopy consente anche la copia diretta da un floppy-disk ad un altro, ma necessita ovviamente di due disk driver, mentre l'A500 a disposizione ne ha solo uno, quindi non sarebbe possibile in quel caso un confronto diretto. Il programma legge il contenuto del dischetto e lo memorizza in RAM, per poi scriverlo sul dischetto di destinazione, effettuando una verifica della scrittura. L'A500 originale ha impiegato per l'operazione 2 minuti e 25 secondi (il tempo impiegato per scambiare i dischetti nel drive non è conteggiato), mentre UAE la completa in 2 minuti e 8 secondi se viene configurato per usare un megabyte di Fast RAM, e 1 minuto e 55 secondi se si utilizza la Real Fast RAM.

Se si tiene conto che l'A500 utilizzato nelle prove monta un floppy-disk driver di circa 12 anni di età, che ha alle spalle svariate centinaia di ore di utilizzo, e che anche i dischetti utilizzati nelle prove hanno pressappoco la stessa età, con tutti i difetti di efficienza che il tempo e l'usura comportano, si può dire che i tempi di accesso dei floppy-disk driver emulati da UAE possono essere considerati paragonabili a quelli dell'hardware originale. Sia le operazioni che comportano la lettura o scrittura del disco sequenzialmente (copia, verifica, formattazione), sia quelle che richiedono seek della testina da una parte all'altra della superficie del disco (l'ottimizzazione, nonchè la normale lettura col Workbench) hanno richiesto tempi di poco inferiori a quelli necessari per le stesse operazioni sull'hardware originale; non è stato possibile per gli autori di UAE sfruttare il fatto che i dischetti sono emulati e vengono caricati in RAM, molti programmi altrimenti risulterebbero non funzionanti.

DPU, acronimo di *Disk Peek and Update*, è un programma freeware che permette di accedere direttamente ai dati memorizzati fisicamente sui dischi, senza passare attraverso il filesystem. Tra le funzioni più interessanti, la possibilità di visualizzare il *RootBlock* e la bitmap che indica i blocchi utilizzati e quelli liberi.



DPU mostra la bitmap dei blocchi liberi di un disco prima (a sinistra) e dopo (a destra) la "cura" di DiskCopy 6.3

Il programma funziona correttamente, e ha permesso di visualizzare le varie informazioni dei dischi; da notare che l'hardfile viene visto da DPU come un hard-disk gestito attraverso l'*uaehf.device*, i cui blocchi sono accessibili e visualizzabili direttamente, ma privo di RootBlock, per cui non è possibile mostrare la bitmap dei blocchi liberi; invece il filesystem virtualizzato come secondo hard-disk non viene rilevato e non è possibile selezionarlo come unità da analizzare.

4.1.3 Programmi musicali: Deluxe Music Construction Set e Protracker 2.1a

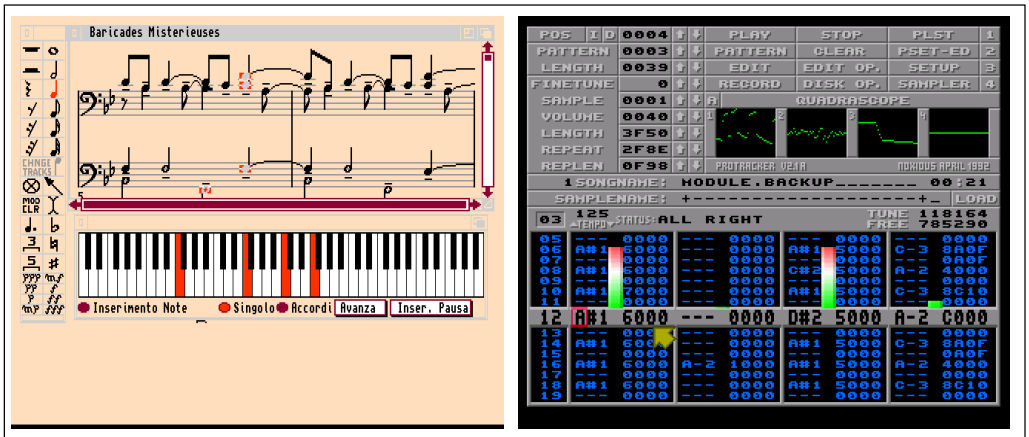
Deluxe Music Construction Set (*DMCS*) è un semplice programma per la composizione di musica; permette di selezionare uno strumento ed assegnargli la melodia da suonare semplicemente collocando le note su un pentagramma per mezzo del mouse, e di riprodurre la partitura con il chipset Amiga o, in alternativa, utilizzando il linguaggio MIDI per comunicare con strumenti musicali professionali come tastiere o *expander*. Tra le funzioni del programma c'è l'evidenziazione delle note suonate sul pentagramma, le cui "pagine" vengono sfogliate automaticamente, e su una tastiera che viene visualizzata sullo schermo.

DMCS è l'applicazione che ha dato i risultati meno soddisfacenti tra quelle testate: tentando di suonare subito dopo il caricamento uno dei brani forniti come esempio nel dischetto del programma, dalla scheda audio proviene solo rumore; tale rumore si stabilizza dopo alcuni secondi diventando infine il brano che si stava cercando di suonare. Se dopo si ferma la riproduzione della melodia e la si fa ripartire, il problema non si manifesta più. Rimangono però problemi nella riproduzione di brani particolarmente complessi: uno degli esempi, chiamato "*Baricades Misterieuses*", viene riprodotto in modo atroce, con delle pause e dei buchi nell'esecuzione che lo rendono irriconoscibile rispetto allo stesso brano suonato dall'Amiga reale.

Durante le prove, l'utilizzo di DMCS con particolari configurazioni (68020, 2 MB di Chip RAM, suono disattivato) ha portato alla corruzione dell'hardfile utilizzato come hard-disk di sistema, con la perdita di tutti i dati in esso contenuti. Si è trattato dell'unico caso in cui UAE ha provocato dei danni. L'incidente è stato in qualche modo "cercato", visto che l'autore di UAE sconsiglia esplicitamente di utilizzare l'hardfile; è stato comunque un fatto indisponente, perchè accaduto proprio alla fine delle prove del software.

Protracker è un *sequencer*, un programma per comporre musica completamente diverso per filosofia di funzionamento rispetto a DMCS. Innanzitutto i suoni degli strumenti non sono sintetizzati, vengono invece riprodotti dei campioni sonori: per produrre le diverse note, uno stesso campione viene riprodotto più o meno velocemente. In Protracker un brano musicale, detto **modulo** (*module*) è composto da un numero arbitrario di parti chiamate **pattern**, ciascuna delle quali è una tabella di 64 righe per 4 colonne, una colonna per ogni voce della sezione audio dell'hardware Amiga. Ogni casella della tabella contiene quattro byte, che a seconda del loro valore assumono il significato di suonare un determinato campione ad una data nota, oppure di cambiare la velocità di riproduzione, passare al pattern successivo o applicare determinati effetti ad un suono già in produzione. Il programma scorre in sequenza le righe del pattern, avanzando ad intervalli di tempo regolari, ed esegue i comandi riportati nelle caselle, agendo in modo indipendente per ciascuna delle quattro voci. Programmando Protracker perchè riproduca i diversi pattern in un determinato ordine, si ottiene un brano musicale completo. Protracker produce file che inglobano i pattern, l'ordine in cui suonarli e i campioni da suonare. Il formato di questi file è leggibile con una routine piuttosto semplice, pubblica e liberamente utilizzabile, ed il risultato ottenibile da un musicista abile è di tale qualità che il formato Protracker è stato il favorito per implementare le musiche nei videogiochi e nelle demo per Amiga.

Protracker 2.1a funziona regolarmente in emulazione: l'output sonoro prodotto dal PC è identico, per qualità audio e per temporizzazione, a quello prodotto dall'Amiga reale. L'esecuzione del programma presenta un solo difetto: alcune volte un campione sonoro non viene suonato affatto, anche se gli strumenti del programma, ad esempio l'analizzatore di spettro, mostrano che il comando di esecuzione del campione è stato interpretato. Il difetto si manifesta in modo evidente se si usa la tastiera del PC per far suonare i campioni come se si strimpellasse una tastiera musicale: alcune volte al tasto premuto non corrisponde alcun suono. Non sembra influire la velocità con cui si indicano le note da suonare.



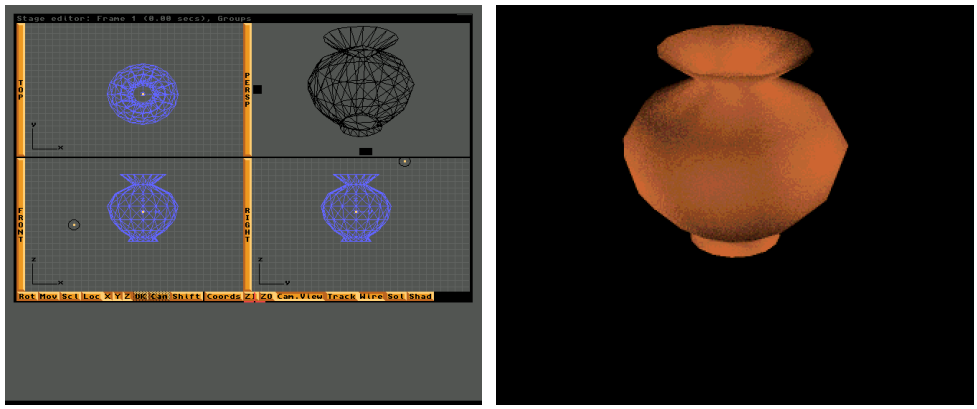
A sinistra DMCS; a destra Protracker

4.1.4 Imagine 2

Imagine 2 è uno dei più potenti programmi per la modellizzazione ed il rendering di scene ed animazioni tridimensionali mai prodotti. Permette di “scolpire”, a partire da una serie di forme base come sfere, cubi, cilindri e superfici spline, modelli tridimensionali di qualunque oggetto, di comporre questi oggetti in scene simili a set cinematografici virtuali, in cui posizionare la telecamere ed un numero arbitrario di luci, infine utilizza la tecnica del *ray-tracing* per produrre immagini ed animazioni in qualità fotorealistica delle scene composte, rendendo sulle superfici degli oggetti gli effetti della luce come rifrazione, riflessione e assorbimento, a partire da una lunga serie di regole che è possibile definire per ogni elemento della scena. Strumenti simili ad Imagine vengono utilizzati per la realizzazione degli effetti speciali digitali in film come Jurassic Park o Star Wars.

Imagine è, come è facile dedurre, un programma che fa un uso intensivo della CPU, ed è quindi un banco di prova efficace per verificare l’effettiva velocità del sistema Amiga emulato.

Imagine si comporta perfettamente bene in emulazione: sin dal caricamento si avverte un sensibile incremento della velocità rispetto all’utilizzo con l’A500 reale. Per effettuare una prova oggettiva ho realizzato una scena semplicissima, composta da una sfera deformata in modo da farle assumere la forma di un vaso, alla cui superficie ho assegnato gli attributi di colore, ruvidezza e durezza per imitare in modo approssimativo l’aspetto della terracotta; ho aggiunto due fonti di luce sferiche omnidirezionali e ho inquadrato il soggetto con la telecamera virtuale. La scena, composta per comodità nell’Imagine sotto UAE, è stata poi importata con successo nella stessa applicazione in funzione sull’A500 reale. Ho quindi lanciato il rendering sulle due piattaforme; Imagine mantiene un file con le statistiche sui frame renderizzati, per cui è possibile un confronto diretto e oggettivo dei tempi di elaborazione.



Imagine 2: a sinistra la fase di allestimento della scena; a destra l'immagine renderizzata della semplice scena di prova

Va detto che, come auspicabile, il risultato del rendering è identico sulle due piattaforme. L'A500 ha richiesto 34 minuti e 57 secondi per completare l'operazione, mentre UAE ha completato il rendering in 4 minuti e 11 secondi, in una configurazione che ricalca quella dell'A500 (512 KB di Chip RAM e 1 MB di Fast RAM), mentre è bastato un minuto e 27 secondi per rendere la stessa scena con la configurazione di default usata per tutte le altre prove (1 MB di Chip RAM e 4 MB di Real Fast RAM).

E' evidente che Imagine è in grado di sfruttare appieno la maggior potenza di calcolo puro del processore della macchina che ospita l'Amiga emulato in UAE. Bisogna aggiungere che la versione di Imagine utilizzata è in entrambi i casi quella compilata per processori 68000, e quindi non viene sfruttata in UAE alcuna funzione del processore che non sia disponibile sull'A500, come ad esempio il coprocessore matematico o l'accesso a memoria a 32 bit.

4.1.5 Altro software

Elenco qui altri programmi provati con UAE, che nelle prove sono stati eseguiti correttamente e non hanno manifestato alcun comportamento anomalo o comunque interessante da segnalare. Un'unica nota riguarda il fatto che non è stato possibile eseguire delle stampe direttamente dall'emulatore: a riguardo la documentazione riporta alcuni suggerimenti su come provare a effettuare la stampa dalle applicazioni Amiga, suggerimenti seguiti senza successo; la documentazione indica però esplicitamente che l'autore non ha mai effettuato alcuna prova a riguardo. Nemmeno dal forum ufficiale di UAE ho potuto ricavare informazioni utili: a quanto pare quella di stampare dall'emulatore non viene considerata come una possibilità interessante.

Altri programmi testati sono:

Final Copy II: word processor del tipo WYSIWYG (*What You See Is What You Get*), uno dei primi nel suo genere;

PageStream 2.2: programma per l'impaginazione dei testi che utilizza font vettoriali;

DICE: compilatore C;

AMOS: interprete e compilatore di un linguaggio simile al *BASIC*, orientato alla programmazione di videogiochi;

Deluxe Paint 4: programma di grafica pittorica;

Cloanto Personal Paint: programma di grafica pittorica in grado di sfruttare il chipset AGA;

Power Packer: "compattatore", ossia un software in grado di comprimere un file eseguibile, in modo che occupi meno spazio su disco, ma rimanga comunque direttamente eseguibile, come un normale programma.

4.2 Giochi

Impostare UAE per emulare i videogiochi Amiga richiede un minimo di pazienza e alcuni minuti di prove: spesso i videogiochi non partono o rimangono bloccati, se trovano che l'Amiga emulato differisce per qualche dettaglio dall'Amiga A500 per il quale sono stati scritti; altre volte i giochi partono ma risultano ingiocabili, perchè troppo veloci o troppo lenti. Ad ogni modo, la possibilità di variare "al volo" una gran parte delle impostazioni dell'emulatore hanno consentito di trovare il giusto "assetto" per quasi tutti i videogiochi provati; alcune volte però non è stato proprio possibile eliminare alcune imperfezioni, ed ho dovuto accontentarmi di compromessi.

L'impostazione a cui i videogiochi sono più sensibili è quella della velocità del processore in rapporto al resto del chipset: mentre le applicazioni AmigaDOS funzionano tranquillamente quando UAE è impostato ad usare la massima velocità ottenibile ("*Optimize for host CPU speed*"), la maggior parte dei giochi richiede che tale opzione sia regolata su "*Approximate 68000/7 MHz speed*".

4.2.1 Lotus Turbo Challenge 2

Lotus 2 è certamente uno dei videogiochi più famosi e più riusciti prodotti per Amiga. Prodotto nel 1991 dai *Magnetic Fields* per la *Gremlin*, con la licenza ufficiale della celeberrima casa automobilistica inglese Lotus, il gioco riprende il gameplay dei più classici arcade di guida, come ad esempio *OutRun* della *Sega*; il giocatore deve guidare la propria vettura su un percorso stradale, ovviamente tortuoso e pieno di ostacoli, e raggiungere i punti di controllo (*checkpoint*) prima che il tempo a disposizione si esaurisca, per poter proseguire la corsa. Lotus 2 vantava una grafica ed un sonoro accattivanti e caratteristiche tecniche di tutto rispetto: 25 frame al secondo, tracciati con cambi di pendenza e le più diverse condizioni atmosferiche, possibilità di giocare in due sullo stesso computer grazie alla funzione di *split-screen* (lo schermo si divide orizzontalmente in due), o addirittura in quattro collegando due Amiga attraverso un cavo seriale null-modem.

Due screenshot del gioco "Lotus 2"

Lotus 2 risulta uno dei giochi che meglio si comportano sotto emulazione. Avendo cura di regolare il timing del processore su "*Approximate 68000/7 MHz speed*", il gioco parte senza richiedere ulteriori modifiche della quantità di memoria installata o della versione del Kickstart. La grafica ed il sonoro appaiono corretti e in sincrono, il gioco funziona alla velocità giusta ed è perfettamente fluido e ottimamente giocabile.

4.2.2 Superfrog

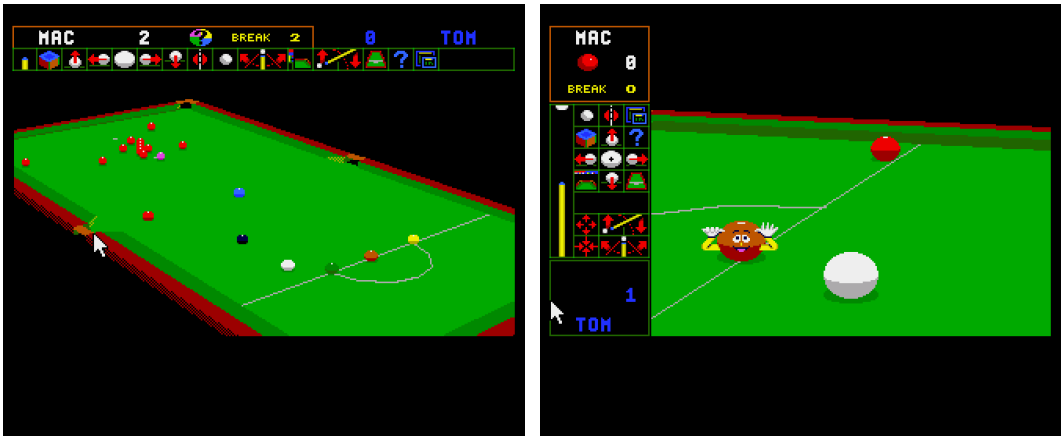
Superfrog, prodotto nel 1993 dal *Team 17*, fu uno dei rappresentanti della superiorità tecnica degli Amiga nei confronti degli altri personal computer dell'epoca. Il gameplay non rappresentava certo una novità, trattandosi del tipico "*platform-game*" sul genere di "Mario-Bros", in cui il giocatore deve guidare il personaggio attraverso i livelli del gioco, raccogliendo un determinato numero di monete per poter accedere al livello successivo, collezionando i bonus e i power-ups più vari e uccidendo i diversi nemici che trova sulla propria strada. Il gioco vantava però caratteristiche tecniche eccezionali: la grafica curatissima e divertente è disegnata in uno schermo overscan di oltre 700x300 pixel di dimensioni, e sfrutta a fondo l'hardware Amiga ottenendo 100 colori apparenti contemporanei sullo schermo attraverso l'uso di palette variabili, uno scrolling nelle quattro direzioni veloce e fluido, e oggetti animati a 50 fotogrammi al secondo. A questo si aggiunge l'ottima musica e l'enorme profondità (una quarantina di livelli che occupano un totale di quattro dischetti), per creare un gioco frenetico e divertente.

Purtroppo Superfrog va collocato nella categoria dei giochi che l'emulatore non riesce a far funzionare correttamente. Il difetto si manifesta in modo evidente sin dai primi istanti del gioco: nella parte inferiore dello schermo, una striscia di circa 50 pixel larga quanto tutto lo schermo, viene disegnata la parte sbagliata del paesaggio; il problema è ben visibile nello screenshot qui riprodotto. L'effetto è molto fastidioso, ed essendo Superfrog un gioco in cui è fondamentale la precisione dei movimenti, perchè si deve saltare su piccole piattaforme o sopra i nemici e schivare gli ostacoli, la sua giocabilità viene compromessa. Per il resto il gioco funziona bene: bisogna regolare UAE perchè tracci solo 25 fotogrammi al secondo, altrimenti il gioco risulta troppo lento, comunque sonoro e sprite dei personaggi del gioco sono quasi tutti corretti, così come gli elementi di contorno, in particolare la presentazione animata.

Due screenshot del gioco "Superfrog"
Si notano gli errori nel disegno di alcuni sprite (le api, a sinistra)
e della schermata del livello (a destra)

4.2.3 Jimmy White's Whirlwind Snooker

JWW Snooker è un simulatore di biliardo in grafica vettoriale tridimensionale scritto nel 1991 da *Archer McLean* per la *Virgin Games*. Viene riprodotto in ogni dettaglio lo *snooker*, gioco molto in voga in Inghilterra ma praticamente sconosciuto in Italia; il videogioco prende il nome da un famoso campione inglese di questa disciplina, Jimmy White, soprannominato "*Whirlwind*" ("Tromba d'aria") per la sua velocità nell'effettuare i colpi. All'epoca fu considerato il miglior simulatore di biliardo mai prodotto ed uno dei migliori videogiochi sportivi di sempre. Le caratteristiche peculiari del gioco sono: l'ottima grafica in 3D, molto bella a vedersi; il motore fisico, molto verosimile; l'intelligenza artificiale degli avversari, semplicemente micidiale ai livelli di difficoltà più elevati.



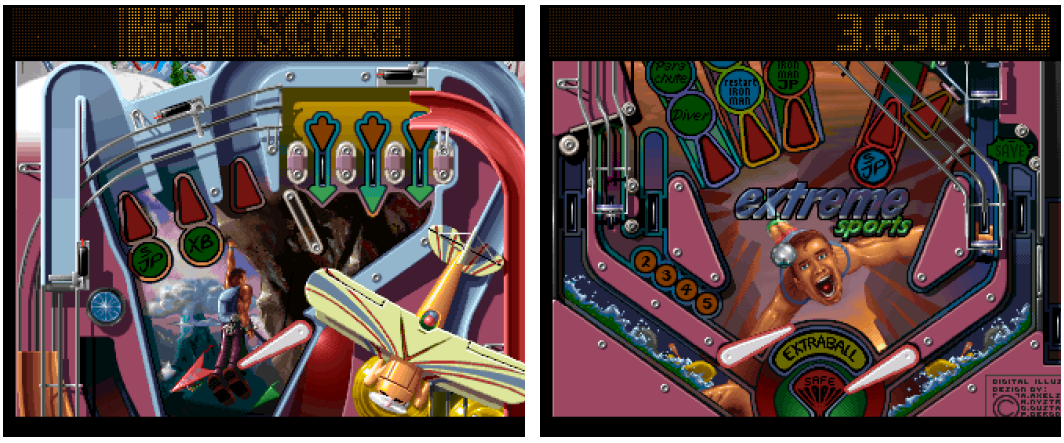
Due screenshot del gioco "JWW Snooker"
Come se il gioco non fosse abbastanza difficile, le palle da biliardo
(a destra) prendono in giro i giocatori!

Il gioco è tra i pochissimi ad avvantaggiarsi della potenza del processore che la macchina ospite mette a disposizione dell'emulatore, infatti appare sensibilmente più veloce sotto emulazione rispetto a quando è in funzione sulla macchina reale. La configurazione dell'emulatore va però accuratamente regolata, perchè alcuni settaggi non solo compromettono la qualità dell'esecuzione del gioco, ma spesso provocano o il reset della macchina virtuale o addirittura il crash di UAE per *segmentation fault*; in particolare la velocità del 68k va portata al massimo ("*Optimize for host CPU speed*"), e deve essere disattivata la rilevazione delle collisioni tra gli sprite. In questo modo il gioco funziona al massimo delle sue possibilità; un solo inconveniente si verifica quando la telecamera virtuale del motore vettoriale del gioco inquadra in primo piano le palla in movimento sul tappeto verde: in questi casi il suo movimento non è fluido, pare invece scattare velocemente avanti e indietro. Questo effetto è probabilmente da attribuirsi non tanto a problemi dell'emulatore, ma ad un effetto dovuto alla diversa sincronia del monitor usato, rispetto a quelli per cui il gioco è stato scritto. Approfondirò l'argomento più tardi, nel paragrafo 4.3.

4.2.4 Pinball Illusions

Nel 1992 i Digital Illusion (*DI*), un gruppo di ex-pirati informatici ed autori di demo, pubblica "*Pinball Dreams*", un simulatore di flipper per Amiga, reinventando un genere e creando un filone destinato a riscuotere un certo successo. Il gioco ha un concept è estremamente semplice: il tavolo del flipper viene visualizzato verticalmente e scrolla in alto e in basso seguendo i movimenti della pallina; il giocatore aziona i respingenti con la tastiera, cercando di spedire la pallina sui bersagli luminosi e di ottenere particolari combinazioni per guadagnare il maggior numero di punti. La realizzazione tecnica è però notevole per qualità di grafica, musica e programmazione, ed il gioco

risulta velocissimo e divertente. Al primo “episodio” segue, sempre nel 1992, “*Pinball Fantasies*”, e nel 1995 “*Pinball Illusions*”.



Due screenshot del gioco "Pinball Illusions"

La versione provata di Pinball Illusions è quella dedicata all’Amiga A1200, e sfrutta perciò la maggior potenza del processore e le migliorate capacità grafiche del chipset AGA, al contrario degli altri giochi provati, che utilizzano tutti l’OCS. Per questo motivo non è possibile comparare direttamente il funzionamento del gioco emulato con quello sulla macchina reale, visto che l’Amiga a disposizione non è equipaggiata con chipset AGA.

Il gioco viene caricato e funziona correttamente se si configura UAE in modo da emulare un Amiga A1200, se cioè si attivano il processore 68020 e il chipset AGA e si allocano 2 MB per la Chip RAM. La documentazione di UAE avverte del fatto che l’implementazione del chipset AGA è ancora in fase embrionale e non è ottimizzata; è però sufficiente a far funzionare Pinball Illusions in modo accettabile anche su una macchina poco potente come quella a disposizione: tutti gli effetti grafici e sonori appaiono corretti e la musica viene riprodotta in modo continuo e regolare, senza scatti nemmeno quando la macchina virtuale è maggiormente impegnata. Certamente lo scrolling dello schermo e i movimenti della pallina non sono perfettamente fluidi, e nei movimenti più veloci il “flickering” che ne deriva dà a volte fastidio; nonostante questo il gioco è utilizzabile. Gli unici momenti in cui l’emulazione AGA manifesta la sua immaturità è quando durante il gioco si entra in modalità “multiball”: quando sullo schermo di gioco compaiono più biglie, esse vengono disegnate in modo non corretto e appaiono lampeggianti.

Pinball Illusions è anche l’unico tra i giochi testati a risiedere su dischetti AmigaDOS, leggibili anche dal Workbench, e volendo può essere installato sull’Hard Disk: sull’ultimo dei quattro dischetti che compongono il gioco è presente uno script per l’Installer Commodore che si occupa dell’operazione in modo automatico; una volta installato il gioco viene lanciato facendo doppio clic sulla sua icona. Come lecito aspettarsi il comportamento del gioco non cambia quando esso viene avviato da hard-disk, rispetto a quando viene caricato dai dischetti, a parte ovviamente la maggiore velocità nell’accesso ai dati.

4.3 Demo

Le demo si sono dimostrate il banco di prova più duro e difficile per UAE, ed il software col quale l’emulatore si è comportato peggio. Mettendo fianco a fianco il PC con UAE in funzione ed un Amiga A500, la differenza della resa generale degli effetti delle demo è desolante, se si tiene conto che il PC è assemblato con componenti che hanno 15 anni meno di quelli di Amiga, ed il processore funziona ad una frequenza di clock cento volte superiore a quella del 68000 che equipaggia il micro-

computer; per quanto questo parametro non sia un termine di paragone affidabile per un confronto diretto tra le due macchine, è comunque un indice approssimativo delle forze in campo.

Le demo vengono riprodotte correttamente: sia la grafica che il sonoro non presentano imperfezioni, ma per il fatto che l'emulatore deve essere settato in modo da disegnare la metà dei frame delle animazioni originali, per riuscire a funzionare alla giusta velocità, il risultato è di qualità molto inferiore a quello che si ottiene con l'Amiga reale.

Difetti ineliminabili Una parte dei difetti di riproduzione delle demo da parte dell'emulatore non è però da attribuirsi al programma. Alcune sequenze dei programmi dimostrativi prevedono il movimento continuo, fluido e uniforme di scritte ed elementi grafici attraverso lo schermo. In tutti questi casi sullo schermo del PC quello che si ottiene è un movimento a scatti. Il problema in questo caso non è nell'efficacia di UAE, ma nel fatto che le animazioni sono progettate per essere riprodotte su un monitor la cui frequenza di refresh sia quella dei televisori casalinghi, ovvero 50 Hz per le televisioni PAL, 60 per quelle NTSC. I monitor odierni funzionano a frequenze maggiori, per limitare l'affaticamento degli occhi dell'utente di fronte allo schermo; tipicamente anche i monitor più economici non faticano a reggere frequenze di 70-80 Hz. Questa differenza costringe l'emulatore a non visualizzare alcuni fotogrammi dell'animazione, o mostrarne alcuni due volte, per mantenerne la sincronia; da qui l'effetto, ineliminabile, delle scritte che saltano e scattano sullo schermo.

Effetti Le demo mostrano tutta una serie di effetti ricorrenti. Gli autori delle demo (detti *coder*) erano infatti in costante competizione per vedere chi riusciva a ricreare lo stesso effetto con il codice più compatto e veloce; a partire da un'idea potevano poi venir sviluppate infinite varianti, ma il concetto di fondo rimaneva lo stesso.

Un elenco di questi effetti, da considerarsi introduttivo ed assolutamente incompleto, riportato ai soli fini di rendere più comprensibile l'analisi del funzionamento delle demo con UAE, può essere il seguente:

tunnel: degli elementi colorati vengono fatti muovere e ingranditi creando l'effetto di cadere in un tunnel multicolore. Alcune volte l'effetto è creato con grafica vettoriale, altre semplicemente con la riproduzione ed il movimento di forme colorate;

starfield: una miriade di puntini vengono fatti muovere tutti insieme a dare l'impressione di essere su un'astronave che si muove a velocità vertiginosa nello spazio, riproducendo un effetto simile a quello visto in molti film di fantascienza;

distorsore: un'immagine bitmap viene distorta e modificata in tempo reale; spesso viene prodotto un effetto prospettico, e si dà l'impressione che l'osservatore si muova in volo sopra l'immagine, che si allontani e si avvicini; altre volte vengono riprodotti effetti come quelli di riflessi sull'acqua o di lenti d'ingrandimento;

scroller: immancabile in ogni demo che si rispetti, si tratta di una scritta che scorre fluidamente sullo schermo, spesso con effetti speciali come cambi di colore o movimenti a forma di sinusoide. Il testo dello scroller riporta in genere saluti agli "alleati" e frasi di scherno per gli avversari del gruppo autore della demo.

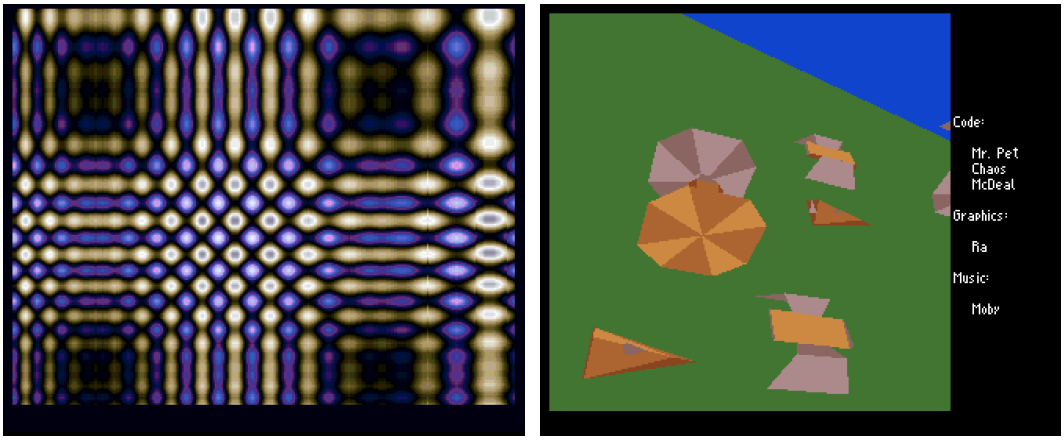
3d: animazioni in grafica 3d poligonale;

texture mapping: immagini bidimensionali che vengono modificate e distorte per dare l'illusione delle tre dimensioni; sono tipici cubi, parallelepipedi e sfere rotanti o saltellanti sullo schermo;

dissolvenze e tendine: usate per collegare diverse scene di una demo, a imitazione di quanto si vede nei film. Le dissolvenze sfruttano il cambio dei colori delle immagini a video; le tendine sovrappongono all'immagine di una scena quella della scena successiva.

4.3.1 Arte

Nell'archivio del sito "*Back 2 the Roots*", che raccoglie una enorme quantità di software per Amiga che gli autori hanno autorizzato a diffondere liberamente in rete, Arte figura al primo posto nella classifica delle demo per Amiga dotati di chipset OCS. Scritta nel 1993 dai *Sanity*, Arte è effettivamente un ottimo esempio di ciò che questi programmatori erano in grado di ottenere dagli Amiga. E' una carrellata di effetti che si susseguono senza soluzione di continuità, come se venisse seguito un unico filo conduttore; il tutto accompagnato da un'ottima musica e condito da una grafica disegnata con stile grottesco e divertente. Tra gli effetti più interessanti posso citare: il tunnel dell'inizio della demo, un corridoio pseudo-tridimensionale con curve e saliscendi percorso a velocità mozzafiato; una interessante sequenza in cui vengono animati dei puntini colorati, a formare scritte scorrevoli e figure in prospettiva; i "titoli" di coda che vedono affiancati uno scroller verticale con i saluti di rito e una scena 3d.



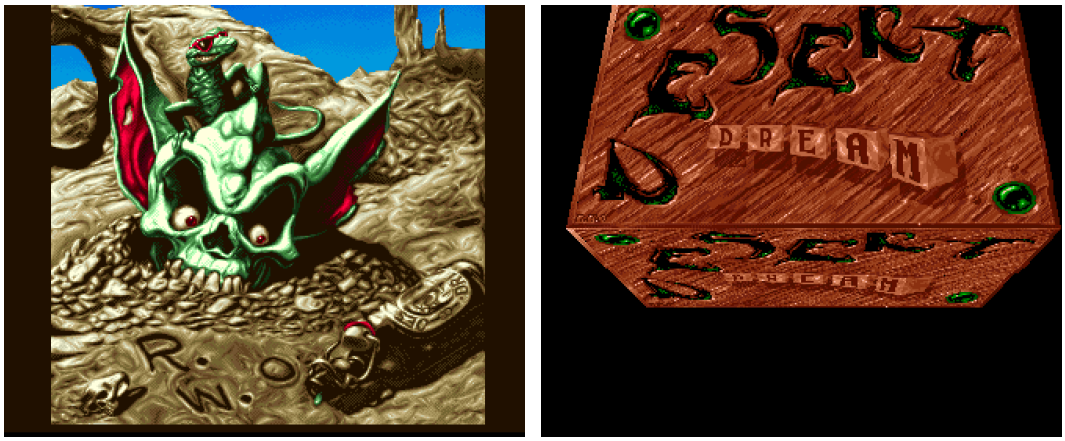
Due momenti della demo "Arte": a sinistra l'effetto plasma, a destra i titoli di coda

UAE esegue la demo dal principio alla fine senza il minimo problema, e senza dover toccare alcuna impostazione. Alcuni effetti mettono però in grossissima crisi la macchina ospite; il tunnel iniziale non viene riprodotto alla giusta velocità se non impostando UAE affinchè disegni solo un fotogramma ogni quattro, mentre con ogni impostazione provata, durante lo starfield della parte centrale della demo il PC accumula uno svantaggio di ben 30 secondi sul totale di circa due minuti di durata della sequenza. Il resto della demo può essere eseguita con il framerate impostato a 1/2 di quello reale. UAE si prende una parziale "rivincita" nella sequenza 3d finale, che appare più fluida in emulazione che sull'Amiga originale: merito dei programmatori della demo che hanno scritto codice compatibile con l'intera gamma Amiga, dall'A500 all'A4000; la demo funziona regolarmente anche se il timing del processore è impostato a "Optimize for host CPU speed" e perciò l'animazione vettoriale, che è una routine di pura potenza di calcolo, può avvantaggiarsi della maggiore velocità della macchina ospite, rispetto al 68000 a 7 MHz dell'Amiga originale.

4.3.2 Desert Dream

Desert Dream è una demo in due dischetti scritta nel 1993 dagli *ANKH*. Presenta un'animazione iniziale in 3d vettoriale, che racconta una storia la cui trama può far dubitare della sanità mentale

degli autori: un'astronave aliena scende sulla Terra e attacca le piramidi di Giza con una fetta gigante di anguria; le piramidi rispondono al fuoco e abbattono il velivolo... Segue una notevole animazione in cui un parallelepipedo texture-mapped ruota ad altissima velocità, ed una sequenza in cui un numero impressionante di punti colorati (da 10 a 12 mila, come viene specificato da scritte a video durante il rendering dell'effetto) viene animato a formare cubi, sfere e altre figure geometriche.



Due screenshot della demo "Desert Dream": a destra il parallelepipedo texture−mapped animato

UAE si comporta decisamente bene fino a questo punto della demo; anche in questo caso non c'è bisogno di modificare alcuna impostazione del programma. Qui però si interrompe e, nonostante i diversi tentativi, non sono riuscito a farlo proseguire. L'Amiga originale esegue invece regolarmente la demo fino alla fine, leggendola da dischetti che sono stati prodotti a partire dagli stessi file ADF utilizzati in UAE.

4.3.3 Red Sector Megademo

La RSM è una demo in due dischetti prodotta da *Red Sector* nel 1989. Più che un'unico programma è una raccolta di effetti che vengono caricati e mostrati in sequenza: un effetto rimane in rotazione finchè l'utente non preme il tasto sinistro del mouse. L'intera demo viene riprodotta senza grandi problemi da UAE, a patto di selezionare "*Approximate 68000/7 MHz speed*" prima del caricamento; altrimenti, la demo si avvia ma si comporta in modo buffo: i titoli iniziali vengono mostrati uno dopo l'altro troppo velocemente e la "voce narrante" non riesce ad annunciarli, e alla fine l'intero programma si inchioda.

Due screenshot del Red Sector Megademo: a destra un esempio di vectorball

La maggior parte delle sequenze sono varianti di scroller: scritte giganti e colorate in movimento che vengono fatte oscillare e saltare, con effetti che imitano la riflessione su uno specchio d'acqua, fuochi d'artificio e figure in movimento sullo sfondo. Il tutto usando schermi in overscan che riempiono la finestra di 800x600 pixel dell'emulatore. L'unico effetto notevole e interessante da guardare è quello chiamato "*vectorballs*": figure formate da sfere pseudo-tridimensionali vengono fatte volteggiare e animate sullo schermo. Altra particolarità è un test sulle conoscenze sulla programmazione Amiga a cui l'utente viene sottoposto alla fine della demo; per la cronaca sono risultato un "promettente principiante"...

5 Conclusioni

Secondo una stima empirica e approssimativa, un computer dispone di sufficiente potenza di calcolo per emulare correttamente un'architettura hardware risalente a non meno di 5 anni prima; è un'affermazione che si può condividere se si considera che oggi ci sono diversi software in grado di emulare perfettamente sui moderni PC videogiochi arcade progettati nel 1993. Amiga è però un'architettura di tale complessità da costituire un'eccezione importante a questa regola: computer in grado di far funzionare un emulatore Amiga in modo soddisfacente sono disponibili al pubblico solo oggi, a più di 15 anni dall'uscita del primissimo A1000. Questo può dare un'idea dell'enorme quantità di lavoro che è stata necessaria per lo sviluppo di UAE.

Durante i test è risultata una compatibilità molto elevata col software esistente: solo videogiochi e demo, che vanno ad effettuare operazioni particolari direttamente sull'hardware della macchina, hanno presentato problemi di funzionamento, ed alcuni di questi problemi possono essere risolti o aggirati selezionando dalla GUI dell'emulatore le opportune opzioni.

Il problema principale di UAE sta nella lentezza del programma: il computer utilizzato nelle prove non è di fascia alta, ma la sua potenza non è nemmeno paragonabile con quella della macchina emulata; eppure, UAE si trova in difficoltà se non gli viene indicato di rendere solo la metà dei fotogrammi prodotti dal software originale, e ciò causa un degrado della resa grafica globale dei programmi, che sui sistemi Amiga è una parte importante.

La lentezza non è causata dal fatto che in Linux non è permesso l'accesso diretto all'hardware della macchina ospite: un confronto con il funzionamento della versione per Windows di UAE, che sfrutta le routine DirectX, non ha mostrato differenze sensibili nella velocità di esecuzione del software. Il problema risiede nel motore alla base dell'emulatore; posso ipotizzare che la situazione non è destinata a mutare: con l'arrivo di processori e memorie sempre più veloci (è stata raggiunta la soglia dei 2 GHz per la frequenza di clock dei processori) i problemi di velocità di UAE vengono sempre più attenuati. Non sarà quindi UAE a cambiare per migliorare le proprie prestazioni: saranno invece i PC a raggiungere una potenza tale da far finalmente funzionare senza rallentamenti l'emulatore.

Non mancano poi alcuni bug, che causano in alcuni casi blocchi e segmentation fault. In un caso l'hardfile che veniva utilizzato come hard-disk di sistema è stato corrotto, ed i file che conteneva sono andati persi; d'altra parte questo non deve sorprendere: nella documentazione del software è chiaramente indicato che si tratta di un programma nella fase beta dello sviluppo.

Nonostante i difetti, UAE è un buon prodotto; si integra bene col sistema operativo ospite, funziona correttamente anche in multitasking con altre applicazioni, ed è sicuro da utilizzare anche per esperimenti con software potenzialmente dannoso (sempre se non si usa l'hardfile!). Certamente le sue prestazioni non lo rendono adatto come piattaforma per utilizzare il sistema operativo Amiga e le sue applicazioni in modo intensivo: è comunque, secondo la mia personale opinione, il

mezzo migliore per entrare (o rientrare) in contatto con una delle architetture più interessanti e più sottovalutate della storia dell'informatica.

Appendice A - Alternative a UAE

UAE è probabilmente la scelta migliore se si desidera utilizzare l'intera varietà del software Amiga esistente, videogiochi compresi. Esistono comunque alcune alternative, che possono avere alcuni vantaggi su UAE, ma anche alcune controindicazioni. Esse sono:

Fellow: è un altro emulatore dell'architettura hardware Amiga. Il fatto di essere stato scritto in buona parte direttamente in Assembler x86 gli dà un certo vantaggio in velocità pura rispetto a UAE; questa scelta ha però lo svantaggio della maggiore difficoltà di manutenzione e sviluppo. Questo fatto, unito alla minore "anzianità", è causa dell'immaturità del progetto, che si manifesta in una compatibilità col software esistente minore rispetto a UAE;

Amithlon, ARES, AmigaXL: si tratta di porting del sistema operativo Amiga sulle piattaforme Intel. Amithlon si appoggia su Linux, AmigaXL su un sistema operativo poco conosciuto ma interessante chiamato QNX, mentre ARES è indipendente; Amithlon e AmigaXL sono prodotti commerciali, il cui prezzo varia tra i 100 e i 150 euro, mentre ARES è distribuito con una licenza simile alla GPL e liberamente scaricabile dal sito. Facendo funzionare codice 68000 su processori Intel x86 compatibili sono anch'essi degli emulatori; al contrario di UAE e Fellow emulano il sistema operativo di Amiga e non l'hardware: questo significa che non possono far funzionare software che dipenda dall'hardware Amiga e che ne faccia un uso diretto (come la stragrande maggioranza di giochi e demo), ma anche che il processore non deve sostituirsi a gran parte del chipset, è meno impegnato e perciò può far girare il software ad una velocità molto maggiore. In effetti la velocità di PC di fascia media con uno di questi sistemi in funzione supera quella dei più veloci Amiga oggi esistenti.

Appendice B - "VIP" Amiga

Alcuni film e serie televisive in cui è stata usata Amiga per gli effetti speciali

- "Babylon 5" (film TV e serie)
- "Freejack": Amiga con VideoToaster utilizzata per la sequenza iniziale e gli effetti di "warping";
- "Jurassic Park": Amiga usata per le anteprime durante la realizzazione dei modelli animati dei dinosauri;
- "Max Headroom" (serie)
- "Robocop" e "Robocop 2"
- "SeaQuest DSV" (serie)
- "Star Trek VI" (film), "Star Trek - The Next Generation" e "Star Trek - Deep Space Nine" (serie)

- “Total Recall”
- “Terminator 2”: effetti di morphing 3d

Bibliografia

Per la stesura di questa relazione sono state utilizzate le seguenti risorse:

Documentazione:

- Amiga Reference Guide di Amiga Magazine, Gruppo Editoriale Jackson
- “The Object-Oriented Amiga Exec” di Tom Holloway, da Byte Magazine, Gennaio 1991
- Official Amiga Hardware Guide, Commodore
- Official Amiga Developer Autodocs, Commodore
- Documentazione allegata con il programma UAE, versioni 0.8.19 e 0.8.20 per Linux, 0.8.17r3 per Windows
- Documentazione allegata con UAE versione 0.8.15 con JIT
- Emulator Technical Centre (http://etc.home.dhs.org/)
- ADF FAQ (http://perso.club-internet.fr/lclevy/adflib/adf_info.html)
- The History of the Amiga (http://amiga.emugaming.com/Fahistory.html)

Forum e newsgroup:

- UAE Discussion Board (http://amiga.nvg.org/uae/index.html)
- alt.emulators.amiga
- it.comp.os.amiga

Archivi di software:

- Back to the Roots (http://www.back2roots.org/)
- Aminet - mirror italiano (http://it.aminet.net/~aminet/)
- World of Classics (http://www.marksplace.f9.co.uk/)

Testo composto con LyX.