BROCCOLICCHI Jérôme
PIERRON Raphaël

Université de Nice
Sophia-Antipolis,
Licence Informatique,
Année 2002-2003.

Travail d'Etude :
Étude sur l'émulation.

Enseignant responsable : BUFFA Michel

Remerciements

Il nous paraît indispensable d'exprimer par ces quelques mots notre gratitude à ceux qui ont concouru à l'élaboration de ce travail, notamment :

M. Michel BUFFA, enseignant au Département Informatique qui est à l'origine de ce travail d'étude, nous a mis sur la voie, et nous a guidé tout au long de la réalisation de ce dossier.

Nous remercions également toutes les personnes qui ont écrit les divers documents dans lesquels nous avons puisé nos informations, en particulier Mrs Victor Moya del Barrio et Marat Fayzullin; Cf. la bibliographie détaillée dans la section dédiée.

Sommaire

Remerciements.....2
Introduction.....4
1 Un petit tour sur l'émulation (informatique).....5
1.1 Qu'est ce que l'émulation (informatique). 5
1.2 Les difficultés..... 5
1.2.1 Se documenter sur le hardware.....5
1.2.2 Problèmes techniques. 6
1.2.3 Un système temps réel..... 6
1.3 Que dit la loi..... 6
2 L'histoire des émulateurs 7
2.1 Historique de l'émulation..... 7
2.2 Les Principaux émulateurs..... 8
2.2.1 Les émulateurs de Console/Arcade :..... 9
2.2.2 Systèmes d'exploitation..... 11
3 Programmer un émulateur.....14
3.1 Architecture générale..... 14
3.2 Émuler le CPU..... 16
3.2.1 L'émulation par interprétation..... 17
3.2.2 L'émulation par traduction binaire (ou recompilation)..... 18
3.2.3 L'émulation et le concept de HLE (High Level Emulation)..... 19
3.3 Émuler la mémoire et les entrées-sorties..... 21
3.4 Émuler le son..... 22
3.4.1 Exemple d'un Système PSG..... 23
4 M.A.M.E (Multi Arcade Machine Emulator).....25
4.1 Architecture de MAME..... 25
4.2 Les Roms de MAME..... 26
4.3 Extensions..... 27
4.4 Contribuer au projet..... 27
Conclusion.....28
Glossaire.....29
Bibliographie 31

Introduction.

Ce dossier est une étude sur l'émulation informatique. L'émulation informatique est le fait d'imiter le comportement d'une machine (ou plutôt d'un logiciel fonctionnant sur une machine) sur une autre machine.

Pour une lecture intuitive nous avons divisé ce rapport en quatre parties:

Tout d'abord nous définirons dans un première partie, ce qu'est l'émulation, et nous donnerons un aperçu des différentes difficultés que rencontrent les personnes impliquées dans la réalisation d'émulateurs.

La deuxième partie concerne l'histoire de l'émulation, ainsi que la description des principaux émulateurs (les premiers comme les plus récents).

La troisième partie porte sur la programmation d'un émulateur: l'architecture générale, l'émulation du CPU avec ses différentes techniques, et enfin l'émulation d'un composant sonore.

Nous conclurons avec une étude sur MAME, le projet d'émulation le plus complet jamais réalisé.

1 Un petit tour sur l'émulation (informatique).

1.1 Qu'est ce que l'émulation (informatique).

L'émulation est une manipulation logicielle et/ou matérielle visant à faire exécuter sur une machine des programmes conçus pour une autre machine d'architecture différente (la machine émulée, par exemple une console de jeux). On crée pour cela un programme, dit émulateur, qui va se charger d'exécuter le code (conçu pour la machine émulée) sur notre machine. Il existe différentes techniques pour réaliser une émulation, chacune avec des performances sensiblement différentes mais en général inférieures aux performances du programme original.

Il est ainsi possible (en théorie) d'émuler tous types de machines,évoluées ou non, comme par exemple les premiers ordinateurs ayant existés, un ordinateur Apple ou PC, ou encore des consoles de jeux; à condition bien sûr d'avoir une machine dotée de performances supérieures aux machines que l'on souhaite émuler...

1.2 Les difficultés

L'émulation, ce n'est pas copier sur un ordinateur le système concerné, mais fonctionner comme lui, ce qui engendre de nombreuses difficultés aux programmeurs d'émulateur.

1.2.1 Se documenter sur le hardware

Pour émuler un système, il faut connaître son architecture : processeur, puces, BIOS, etc. Pour les ordinateurs, des manuels existent; pour les consoles ou bornes d'arcades, les informations sont plus rares.
Les consoles ont souvent des processeurs multiples et spécialisés dans certaines fonctions (images, sons, mémoires...). Sans informations détaillées, le développeur ne peut faire reproduire à son émulateur les fonctions de tel ou tel processeur, d'où certains émulateurs incomplets (il existe par exemple un émulateur Neo Geo sans son !).

Certains programmeurs réussissent à obtenir les données techniques à priori confidentielles des machines, d'autres pratiquent le reverse engineering qui consiste à décrypter un programme à l'envers pour comprendre (en tâtonnant) comment il marche à l'endroit.

1.2.2 Problèmes techniques.

Un émulateur comprend un module de traduction des instructions du processeur de la machine émulée vers le processeur de la machine hôte. Il faut également simuler toutes les entrées-sorties : affichage, gestion des lecteurs de disquette, souris, son... Le programmeur doit donc connaître à la perfection le matériel (et la programmation du matériel) de ces deux machines!
Bien sûr, le programmeur doit aussi maîtriser les techniques d'émulation et choisir la plus adaptée en fonction de ses connaissances et des performances qu'il souhaite obtenir.

1.2.3 Un système temps réel

Une émulation est un système temps réel dans le sens où elle doit émuler le plus fidèlement possible le temps de la machine que l'on émule. Par exemple, si la machine émulée exécute 2 millions d'instructions et 60 images par secondes, l'émulateur devra émuler le même nombre d'instructions et d'images en une seconde. Cela signifie ralentir l'émulateur s'il s'exécute trop vite sur la machine hôte, ou accélérer en passant certains calculs (par exemple ne pas calculer toutes les images, ou réduire la qualité du son...) s'il est trop lent, pour synchroniser l'émulateur avec le temps prévu pour la machine réelle.

1.3 Que dit la loi

Selon la loi, le fait de créer et de distribuer des émulateurs est légal, par contre il est interdit de distribuer une rom (Cf.glossaire), ou de télécharger une rom dont vous n'avez pas l'original. De même, le dumping (Cf.glossaire) n'est pas en soi une entorse à la loi, mais c'est leur distribution et leur utilisation qui posent problème. Certains émulateurs ont besoin pour fonctionner du BIOS des machines originales, qui est bien sûr la propriété de l'entreprise qui a créé la machine, donc dans ce cas l'émulation est illégale. Pour télécharger la ROM d'un jeu en toute légalité vous devez posséder la cartouche originale (ou avoir l'accord du propriétaire des droits du jeu).

Il existe cependant des sociétés qui aident l'émulation en abandonnant les droits (ces jeux sont appelés « abandonware » mais en fait la définition est plus large, Cf. glossaire), en effet on émule généralement de vieux jeux qui ne sont plus commercialisés, et par conséquent il n'y a pas (ou très peu) de préjudice pour les créateurs du jeu. Certaines sociétés ont tenté des actions (Nintendo, Sega...), et beaucoup de sites qui distribuent les ROMs illégales des jeux ont dû fermer sous des menaces de poursuites judiciaires.

2 L'histoire des émulateurs

2.1 Historique de l'émulation.

De la fin de la deuxième guerre mondiale aux débuts des années 80, IBM (International Business Machines, compagnie fondée en 1911) a régné sur le monde de l'informatique. C'est pendant cette période qu'IBM va inventer l'émulation et créer les premiers émulateurs.

À cette époque les ordinateurs sont conçus comme des machines totalement indépendantes les unes des autres. Aucun composant ne peut être partagé et aucune compatibilité n'est possible. Chaque système nécessite donc ses propres données logicielles et son propre hardware adapté à l'usage que l'on veut en faire.

En 1962, un rapport circule chez IBM selon lequel un « utilisateur anonyme » est parvenu à modifier le hardware d'un IBM 705 afin qu'il puisse exécuter des programmes écrits pour IBM 1401, modèle plus ancien. Ce rapport va définitivement indiquer aux chercheurs de la compagnie la voie à suivre : La compatibilité descendante, à savoir la compatibilité de tout nouveau modèle IBM avec les précédents modèles sortis.

On connaît aujourd'hui, notamment grâce aux systèmes d'exploitation Microsoft (basés sur l'inévitable MS/DOS pour les rendre compatibles), les répercutions d'une telle politique sur les ventes d'une même marque, et la fidélisation de la clientèle qu'elle permet (même si le principe a aussi des inconvénients, voir toujours l'exemple de Microsoft et du DOS). En effet, Il n'y a ainsi plus besoin de se doter d'une batterie de logiciels lors de l'acquisition d'un nouvel ordinateur, puisqu'il peut se contenter de ceux de ses prédécesseurs.

En 1963, IBM confie à un des ses laboratoires la mission de développer une série de programmes de simulation ayant pour but d'imiter le comportement de ses 7 ordinateurs les plus populaires, et d'établir une compatibilité descendante entre eux, sur le plan logiciel. Les premiers résultats sont décourageants. La simulation a au mieux un fonctionnement deux fois moins rapide que les ordinateurs en question, donc inutilisable dans un cadre professionnel.

La compatibilité descendante ne peut donc pas être provoquée par un intermédiaire logiciel, mais doit faire partie intégrante du fonctionnement des ordinateurs.

Pendant ce temps, Honeywell (concurrent d'IBM) sort le H-200, un

ordinateur capable de faire fonctionner des programmes IBM. Il n'est plus seulement question de compatibilité descendante au sein d'une ligne d'ordinateurs de même marque, mais de compatibilité avec d'autres systèmes, d'autres marques.

IBM se penche alors vers une solution hardware, qui va aboutir à l'IBM 1410S, une machine permettant à IBM de rattraper temporairement son retard technologique sur Honeywell, mais qui ne préfigure aucune ligne d'ordinateurs tous compatibles entre eux.

Un chercheur d'IBM, Larry Moss affirme que pour qu'un ordinateur puisse être compatible avec un autre, le hardware des deux machines doit être le plus similaire possible, tout en permettant à la fois l'exécution des programmes pour lesquels ils sont conçus et le lancement d'un programme de simulation.

Il se lance donc dans l'étude d'une solution de compatibilité mixant les approches matérielles et logicielles.Il parvient à mettre au point une extension pour les ordinateurs NPL les rendant capables d'exécuter des programmes prévus pour IBM 7070, le plus sophistiqué des ordinateurs IBM.

Les travaux de Moss aboutissent à un fonctionnement parfait de ces programmes, ne montrant aucune perte de performances. Il décide, pour montrer la supériorité de son projet sur la « simulation», de lui conférer l'appellation d'« émulation », expression indiquant une amélioration, une avancée. On peut donc dire que Larry Moss est le « père » de l'émulation

La suite de l'histoire n'est qu'une longue série de projets d'émulateurs utilisant ou non des extensions matérielles, et rendant potentiellement compatibles des machines plus ou moins différentes les unes des autres.

Aujourd'hui, la course à l'émulation continue, en effet, de nouveaux émulateurs sortent chaque mois, et certaines machines font l'objet d'une multitude de projets d'émulation donnant pratiquement les mêmes résultats.

Faute d'anciennes machines à émuler, ou de machines sur lesquelles émuler, les spécialistes se tournent alors vers les ordinateurs, jeux d'arcades ou consoles qui sont encore d'actualité, et l'émulation devient (le plus souvent) du piratage.

2.2 Les Principaux émulateurs.

Actuellement, les émulateurs sont très utilisés, que ce soit dans le domaine du jeu ou dans des applications permettant de reproduire des systèmes d'exploitation. On peut pratiquement émuler toutes les architectures (ordinateurs, consoles, Palm, Pocket PC ...).

2.2.1 Les émulateurs de Console/Arcade :

Voici une liste de quelques émulateurs de console vidéo attention cette liste n'est pas exhaustive.

- Multi jeux (borne d'arcade) :

Nebula : C'est un émulateur arcade multi-système, Capcom CPS1-CPS2, Neogeo, PGM, Konami. C'est le seul émulateur PGM avec le son. C'est également le meilleur émulateur CPS2 en ce qui concerne le rendu, ainsi que le meilleur émulateur Neogeo CD (console).

Nebula model2 : Version uniquement Model2 de nebula, qui émule à présent 50% des différents model2, avec le son.

M.A.M.E (cf. chapitre 4) : Multiple Arcade Machine Emulator. Le plus gros émulateur multi-système. Il a pour but d'émuler toutes les bornes d'arcades.

Callus : Callus est un des meilleurs (si ce n'est le meilleur) émulateur Capcom CPS1, très rapide sur un P133. Ce système a connu la gloire avec Street Fighter II, mais aussi Final Fight (...). Créé par Sardu (Bloodlust sofware) qui est l'auteur de Genesyst et de Nesticle (développé entièrement en assembleur).

EmuDX : EmuDX est un émulateur d'arcade supportant de vieux jeux (tel que Pac Man) mais en change les graphismes pour les remettre au goût du jour.

- Atari ST :

Saint: Écrit (par Arnaud Carre) à l'origine pour faire tourner les démos, cet émulateur est maintenant un des meilleurs.. De plus, il a l'option géniale d'émuler le bruit du lecteur de l'atari.

Gemulator 2000 : Un émulateur Atari ST tournant sous Windows. Il est très complet étant donné qu'il supporte des éléments tels qu'un Joystick, un Modem, une Imprimante.

- Gameboy Advance : (1er console portable 32 Bits)

Visualboy Advance : Que dire si ce n'est qu'il s'agit du meilleur émulateur Game Boy Advance du moment. L'auteur fait beaucoup pour assurer une compatibilité maximale.

Boycott Advance : Un des meilleurs émulateurs Game Boy Advance,

une très bonne interface graphique et une bonne compatibilité.

FoolsGBA : Voici un émulateur GBA qui utilise la recompilation dynamique.

iGBA beta : Un émulateur Game Boy Advance plus que prometteur, resté la référence quelques semaines. Mais son auteur (Iki) a décidé de stopper son projet.

Dream GBA : Un très bon émulateur Game Boy Advance.

- Playstation :

ePSXe : Un très bon émulateur PSX, très proche de l'original, hélas un peu dur à configurer pour les débutants. Il s'accommode de plugin PSEmuPro.

AdriPSX : Emulateur PSX qui utilise un moteur de recompilation dynamique complet et rapide.

VGS (Virtual Game Station) : Un très bon émulateur, un des rares avec Bleem! a ne pas demander de plugins (payant).

- NES :

Nesticle : superbe émulateur NES.

- Super Nintendo :

ZSNES : Sûrement le plus développé des émulateurs SNES. Il supporte des chips tel que le Super FX ou le DSP de Pilotwings.

Snes 9x : Tout simplement un excellent émulateur SNES.

NLKE : Esnes fût un très bon émulateur SNES, sortit mi-1997. Il a depuis fusionné avec NLKSNES pour devenir NLKE.

- Nintendo 64 :

1964 : Un émulateur N64 passé Open Source, vous pouvez donc participer à son développement. Notez qu'il émule déjà plus de 100 jeux commerciaux.

Ultra HLE : Cet émulateur a fait sensation à sa sortie. En fait il émulait les principaux jeux sur un PC puissant (PII300) munit d'une carte supportant le

glide (3dfx).

Project64K : Project64K est une version non officielle incorporant le module Kaillera afin de pouvoir jouer à plusieurs en réseau.

Corn : Corn est un émulateur N64 rapide, il gère le son et les sauvegardes.

- Neo-Geo :

NeoRage X : Fonctionne aussi bien sous Win9x, Win2000 que sous Win XP.

KBMAME : KBMAME est une version de MAME qui ne supporte que les jeux Neo Geo, mais elle ajoute aussi plusieurs modifications dont : - Support des Combos programmables - Ajustement de l'autofire (...) Cette version a été compilée en C.

- Playstation 2 :

PCSX2 : Le tout premier émulateur PS2 disponible, comme le nom le suggère, il a été créé par Shadow & Linuzappz, la team de l'émulateur PSX PCSX. Bien entendu il ne lance aucun jeux pour le moment, mais déjà quelques démos techniques.

Neutrino : Un émulateur PS2 open source, qui commence à montrer les premiers écrans de jeux commerciaux émulés.

- X-Box :

"CXBX is designed to be an XBox Emulator for the PC". Vous vous en doutez pour le moment ce projet n'est qu'à l'état de développement peu avancé bien qu'il progresse.

2.2.2 Systèmes d'exploitation

Certaines applications ne sont pas des émulateurs à proprement parler car le système cible a le même hardware (le système émulé tourne comme une extension du système émulant).

- Les « vrais» émulateurs :

- VmWare : VmWare s'installe comme n'importe quelle application standard sur la machine hôte. Une fois installé, vous pouvez faire tourner un autre système d'exploitation et ses applications de la même façon que sur une machine physique. Le nouvel environnement tourne dans une machine virtuelle parfaitement isolée des autres machines virtuelles comme de la machine hôte. La défaillance de l'une des applications ou de l'environnement virtuel n'aura aucune incidence, ni sur la machine physique, ni même sur les autres machines virtuelles qu'elle héberge.

Utiliser plusieurs machines virtuelles est aussi facile que de passer d'une application à une autre sur votre poste de travail. Les machines virtuelles peuvent tourner dans des fenêtres ou bien en plein écran. Des fonctions avancées incluent la création de réseaux virtuels, l'arrêt, la mise en veille et le redémarrage de la machine virtuelle, la gestion des disques " non persistants ", la portabilité des machines virtuelles.

Avec les disques " non persistants " vous pouvez essayer des scénarios du type " et si " sans avoir à craindre de détruire votre environnement de test ou sans avoir à le reconstruire. En effet, une machine virtuelle peut être restaurée d'un simple clic de souris.

Avec VmWare, vous pouvez construire sur un simple PC des environnements réseaux complexes et permettre aux machines virtuelles de communiquer entres elles, avec la machine hôte ou bien encore avec d'autres réseaux. VmWare peut être considéré comme la référence des émulateurs, son équivalent libre est plex86 (en cours de développement).

- VirtualPC : Initialement conçu pour les plates-formes Macintosh, Virtual PC, de Connectix , permet d'utiliser avec Mac OS des logiciels conçus pour Windows.

Connectix s'intéresse aujourd'hui aux utilisateurs Windows en lançant Virtual PC for Windows, une nouvelle version de son logiciel d'émulation. Elle permet d'émuler des systèmes d'exploitation PC par-dessus une version de Windows.

Les systèmes d'exploitation émulés par Virtual PC sont : Windows 3x, Windows 95, Windows 98, Windows Millennium Edition, Windows NT, Windows 2000, MS-DOS, Red Hat Linux, TurboLinux, Free BSD, OS/2 Warp et Novell NetWare.

- Bochs : Le programme Bochs est un émulateur pour PC x86 facilement importable sur d'autres systèmes. Il est écrit en C++ et il s'exécute sur la plupart des plates-formes. Il inclut l'émulation de l'unité centrale Intel x86, des périphériques communs IO et d'un BIOS sur mesure. Généralement, Bochs se

compile pour émuler une unité centrale Pentium, un 386 ou un 486. Bochs est capable de faire fonctionner la plupart des systèmes d'exploitation à l'intérieur de l'émulation.

- Windows -> Macintosh :

Basilisk II : C'est un émulateur de Macintosh à base de 680x0, placé en Open Source et développé par Christian Bauer. L'objectif de Basilisk II/JIT est d'y apporter un noyau d'émulation utilisant la méthode de recompilation dynamique. Basilisk II/JIT n'est disponible que pour Linux/i386 et FreeBSD/i386. Une ancienne version expérimentale pour Windows est toujours disponible mais elle n'est plus maintenue.

Les « faux » émulateurs (même hardware):

- Windows -> Unix/Linux :

Cygwin : Environnement Unix pour Windows, repose sur deux éléments principaux. D'une part, il propose une bibliothèque (cygwin1.dll) qui implémente les API d'Unix sous Windows. D'autre part, on y trouve un lot de programmes en provenance d'unix, fondé sur cette bibliothèque. Il a pour but de permettre le fonctionnement sous l'OS de Microsoft des applications Unix, tout en effectuant très peu de modifications dans le code source original.

L'environnement propose de nombreuses catégories de programmes usuels sous Linux (bash, cat, cp, mv, tar, gzip, libncurses, zlib...). Cygwin est le seul environnement libre de ce type. Pour conclure, Cygwin se montre satisfaisant pour utiliser des applications Unix légères mais est déconseillé pour l'usage soutenu des programmes en mode graphique.

Windows Services for Unix (SFU) : SFU fournit de nombreux composants servant à rendre la cohabitation entre Windows et Unix plus facile. Il s'agit d'un projet plus vaste que Cygwin, étant donné qu'il sert non seulement à faire fonctionner des applications Unix sous Windows mais également à faciliter l'administration d'un réseau. Il s'agit d'un environnement similaire à Cygwin.

Il propose de nombreuses commandes Unix de base (cat, cp, rm, tar, vi...) ainsi que des outils de développement (gcc, le débogueur gdb...) mais l'on regrette l'absence de quelques standards, comme bash et zsh. Services for Unix est stable mais il n'y a pas d'environnement graphique Xfree.

UWin (Unix for Windows) : L'environnement commercial Uwin incarne une autre alternative à Cygwin. Créé par David Korn l'auteur du KornShell, il fonctionne avec toutes les versions de Windows. Uwin se base sur plusieurs bibliothèques DLL qui fournissent les appels systèmes Unix. Il fournit des commandes basiques, et pour les versions les plus évoluées il propose en

plus des outils pour l'administration.

- WinaXe Plus : WinaXe est un environnement X Windows. Il apporte des applications Unix (se connecte à la plupart des hôtes Unix) à distance sur votre PC, chacune avec une fenêtre séparée. Dans cette environnement tout se déroule de manière concurrente et est totalement interactive. WinaXe est un logiciel 32 bits qui fournit un moyen de transformer votre PC en poste de travail X Windows.

- Linux -> Windows :

- Wine : Wine est un projet open source qui peut être caractérisé par deux choses. D'une part, Wine Is Not Emulator, Wine est une librairie qui peut aider à recompiler un programme destiné à Windows pour qu'il fonctionne sous Linux (et maintenant BeOS), et d'autre part, WINdows Emulator, Wine est un programme qui émule Windows, pour permettre d'utiliser des programmes que l'on ne peut pas recompiler sous Linux en transformant les appels aux fonctions de Windows par des appels aux fonctions de X ou de Linux.
  Les programmeurs (plus de 300 personnes) ont réécrit les DLL de Windows pour que Wine puisse les utiliser lorsqu'un programme en a besoin (Wine rend les applications Windows native Linux).
  (Cf. chap. 3), Callus, System 16, Raine et Rage...

3 Programmer un émulateur

3.1 Architecture générale

Dans un ordinateur, le CPU est le « centre de contrôle » de la machine, le CPU émulé sera donc le centre de contrôle de l'émulation, c'est lui qui commandera les autres périphériques émulés.
La partie principale de l'émulation du CPU est une boucle: récupérer les instructions à exécuter, les décoder, les exécuter et continuer la boucle.
Il faut savoir que les périphériques travaillent en parallèle alors qu'ils ne sont pas émulés en parallèle (Cf. figure 1).

Les modèles parallèles d'émulateurs sont difficiles à implémenter à cause de la synchronisation nécessaire entre les différents périphériques émulés. Un modèle comportant des threads serait encore plus difficile à implémenter pour les mêmes raisons; aussi nous ne considérerons pas le problème de l'implémentation d'un émulateur en parallèle ou comportant des threads.

La boucle principale d'un émulateur suit ce schéma:

```
while(!stopper_l_emulation)
{
     executeCPU(cycles_à_exécuter);        (1)
     générerLesInterruptions();            (2)
     émulerLeGraphisme();                  (3)
     émulerLeSon();                        (4)
     émulerCeQuilResteÀÉmuler();
     Synchronisation();                    (5)
}
```

Figure 1: Algorithme général d'un émulateur

Explications:

(1)Étant donné que nous connaissons le CPU que l'on émule, nous connaissons sa vitesse d'horloge, et le nombre de cycles qu'il prend pour chaque instruction. À partir de là, nous pouvons demander au CPU d'exécuter un certain nombre de cycles (cycles_à_exécuter), et nous savons combien de temps le système réel aurait pris pour faire ces instructions, ce qui nous permettra en (5) de synchroniser notre émulation avec la machine émulée (jeter un oeil à « 1.2.3 L'émulation, un système temps-réel »). Le nombre de cycles à exécuter est dépendant de la fréquence des interruptions qu'il faut capturer (le clavier par exemple): en effet, plus on a d'interruptions à capturer, plus il faudra s'arrêter souvent et donc plus le nombre de cycles sera faible.

(2)Dans la boucle générale, on teste si les conditions nécessaires sont réunies pour générer des interruptions, si oui on les signale à la boucle du CPU (au CPU émulé), et elles seront traitées la prochaine fois qu'on lancera le CPU.

(3)Ceci est présenté dans la partie «2.2.2 Émuler le son »

(4)Ceci n'est pas détaillé dans ce dossier, mais l'approche est la même que pour émuler le son.

(5)Selon la vitesse du système, la fonction « synchronisation » fait une pause ou notifie aux fonctions de graphisme et de son qu'il faut accélérer (en agissant sur les paramètres de qualité par exemple).

Une partie des machines émulées est prévue pour le jeu, et contient des composants spécialisés pour gérer le son, le graphisme... qu'il faudra prendre soin d'implémenter de façon efficace: pour réaliser ces tâches, notre machine hôte ne devra « à priori » compter que sur son CPU; dans certains cas le pourcentage de CPU nécessaire pour émuler ces tâches est très supérieur à celui pour émuler

le CPU de la machine émulée...

On peut diviser les composants à émuler en deux parties: l'émulation du CPU et l'émulation des autres composants (son, graphisme ...), qui sont bien distinctes. Le rôle du CPU est d'exécuter les instructions, alors que, par exemple, le rôle d'une carte son est (quand elle reçoit l'ordre d'émettre) d'envoyer un signal dans les haut-parleurs pour produire un son, et le rôle d'une carte graphique est d'afficher une image sur un écran. Les algorithmes servant à émuler le graphisme ou le son n'ont donc rien à voir avec les algorithmes servant à émuler le CPU.

3.2 Émuler le CPU

Il est possible de devoir émuler des systèmes multiprocesseur, nous nous limiterons (pour simplifier) à des systèmes monoprocesseurs. Ce que nous voulons émuler du CPU, c'est la manière dont il opère. Les seuls aspects du CPU qui nous intéressent sont ceux qui ont un impact sur le fonctionnement de la machine.

Il y a deux manières principales d'émuler un CPU.

- Par interprétation :

On récupère le code source, et dans une boucle récuperer_les_instructions-décoder-exécuter, on décode quelles instructions CPU impliquent les octets récupérés puis on exécute les fonctions associées aux instructions décodées. C'est la méthode la plus simple à programmer, mais aussi la plus gourmande en temps CPU.

- Par traduction (ou recompilation) :

On récupère le code source, puis on le traduit pour le CPU de la machine hôte. Ce procédé est appelé « traduction binaire » (Binary Translation) dynamique (DBT) ou statique (SBT), ou encore (de manière abusive) recompilation dynamique (ou statique).

3.2.1 L'émulation par interprétation

L'interprétation est la façon la plus simple d'émuler un CPU.

Il s'agit de reproduire la manière dont un processeur basique fonctionne. Il lit des octets à l'adresse pointée par un registre spécial (PC ou Program Counter). Ces octets contiennent l'instruction que doit exécuter le processeur. Le CPU doit alors décoder ces octets et décider de ce qu'il doit faire.

Ensuite il réalise l'action demandée, actualise le PC et lit un autre, ou d'autres octet(s) (le nombre d'octets dépend de la taille d'une instruction pour le CPU émulé). Une des façons les plus simples est un énoncé switch/case pour chaque différent opcode (groupe d'octets définissant une instruction).

```
switch(memory[PC++])  /* en fait on incrémente PC de la
                         taille d'une instruction */
{
     case OPCODE1:
         opcode1();
         break;
     case OPCODE2:
         opcode2()
         break;

     case OPCODEn
         opcodeN();
         break;
}
```

Figure 2: Algorithme général d'un émulateur par interprétation

Une émulation par interprétation est lente comparée aux autres formes d'émulations CPU. Le problème vient du décodage et de l'ordonnancement des instructions.

Une façon d'améliorer sensiblement les performances d'un émulateur par interprétation est de programmer ce type d'émulateur en code assembleur, en effet la programmation en C (ou en un autre langage de haut niveau) des instructions CPU de base est coûteuse; on améliorera donc l'exécution mais pas

la traduction. Le problème de la programmation en assembleur est qu'il n'est pas portable sans réécrire le code pour chaque CPU.

Il faut donc comparer les performances de la machine hôte et de la machine émulée pour voir si une programmation en assembleur se justifie. Une autre amélioration possible concerne le décodage des instructions, lorsque le même code est utilisé plusieurs fois; on stocke l'instruction traduite, et on la ressort lorsque l'on rencontre à nouveau le même code. Ce type d'interpréteur est appelé « émulateur avec threads » (threaded emulators).

3.2.2 L'émulation par traduction binaire (ou recompilation)

Il est plus rapide de traduire les opcodes de la machine émulée en opcodes pour la machine cible que de récupérer chaque opcode, de décoder l'instruction, et enfin d'exécuter une fonction qui implémente cette instruction. C'est ce constat qui fait de l'émulation par traduction la méthode d'émulation la plus efficace.

Le procédé de traduction binaire doit d'abord récupérer le code natif pour le CPU émulé, puis le traduire, en utilisant des techniques de compilation par exemple, en code pour le CPU cible. Le code traduit est alors stocké et exécuté chaque fois que l'émulation du CPU est appelée.

La traduction se fait par blocs de code. Les raisons de traduire par blocs plutôt que de traduire tout le programme en une seule fois sont diverses: on ne connaît pas forcément tout le code à traduire dès le début, et il doit y avoir des points d'arrêts de l'émulation pour faciliter le processus de traduction.

Il y a deux possibilités de traduire le code: de façon statique ou de façon dynamique. Faire la traduction de façon statique implique qu'à la fin de la traduction, le code est entièrement traduit pour le CPU cible (puis on peu l'exécuter directement). Faire la traduction de façon dynamique signifie que le code pour le CPU émulé est traduit au fur et à mesure de l'exécution.

La traduction statique ne peut pas être utilisée dans beaucoup de cas: par exemple on ne pourrait pas traduire de façon statique les milliers de programmes qu'un ordinateur pourrait exécuter, ou alors traduire du code qui s'auto-modifie. Pour certaines machines, comme les machines d'arcade, on peut très bien utiliser cette technique.

La traduction dynamique, est une technique très utile pour émuler un CPU. Cette technique est parfois appelée traduction à la demande. Le code n'est traduit que lorsque l'on en a vraiment besoin. Quand du code est traduit pour la première fois, on stocke sa traduction pour le ressortir quand on en a à nouveau besoin.

Ce procédé de traduction dynamique peut paraître long, mais il faut bien voir que chaque instruction n'est traduite qu'une fois (exactement comme avec la traduction statique!), cette méthode est donc meilleure que la traduction statique parce qu'elle est plus générale, et elle est meilleure que l'interprétation parce qu'elle est plus performante (l'interprétation doit décoder toutes les instructions, et toutes les exécuter sauf si elle est threadée).

La traduction, qu'elle soit dynamique ou statique, pose beaucoup de problèmes comme la différence entre l'ISA (Instruction Set Architecture) de la machine cible et de celle que l'on émule, l'allocation des registres, la manière d'utiliser les blocs de code, ou encore les optimisations possibles.

Un autre problème est que ce type d'émulation est difficile à réaliser pour du code qui s'auto-modifie, ou de la génération dynamique de code. C'est même impossible pour un traducteur statique, étant donné que tout le code doit être généré avant l'exécution alors qu'un traducteur dynamique peut détecter les zones de mémoire qui contiennent du code traduit modifié. A contrario, le simple interpréteur n'est pas affecté par ce problème. Il faut donc bien réfléchir avant de choisir une des techniques d'émulation.

3.2.3 L'émulation et le concept de HLE (High Level Emulation)

Le premier émulateur de Nintendo 64, UltraHLE, a introduit le concept de "High Level Emulation" (émulation haut niveau). Ce concept n'est pas nouveau, mais il n'était pas utilisé dans les émulateurs le précédant, ou pas de la même manière.

Le principe de HLE est d'arrêter l'émulation le plus tôt possible et convertir alors les instructions directement en langage de haut niveau. Bien sûr, cela n'a un intérêt que si ce qu'on émule est programmé dans un langage haut niveau (il serait idiot de convertir du code qui a été écrit à la main, en assembleur, dans un langage de haut niveau). Cette méthode est généralement utilisée en la combinant avec une des deux techniques d'émulations présentées ci-dessus.

Prenons l'exemple de UltraHLE. UltraHLE détecte les points du code émulé où on utilise une fonction graphique ou sonore. Il y pose des points d'arrêts, et quand ces fonctions sont appelées, l'émulateur récupère les paramètres de la fonction et émule son comportement (grâce à une fonction de l'émulateur spécialement prévue pour émuler le graphisme ou le son). Dans les premières versions de UltraHLE, l'émulation était programmée en C, mais elle est à présent émulée directement en code machine (ix86), l'émulation du CPU est donc ici une traduction binaire.

Sur cette plateforme (la N64), le graphisme étant très puissant, il est nécessaire d'avoir une carte spécialisée gérant la 3D et différents effets. Il s'agit alors de convertir l'API graphique de la N64 vers l'API de la carte graphique (OpenGL ou Glide). C'est un gain important de performances puisque l'on émule pas le hardware de la machine émulée, mais son comportement en utilisant le hardware de la machine cible (au lieu de n'utiliser que son CPU).

En fait cette idée est utilisée depuis longtemps dans des traducteurs binaires, conçus pour porter des programmes d'un système vers un autre (et donc, éventuellement, d'une machine vers une autre). Ces logiciels, en plus d'émuler le reste de l'OS, convertissent les appels systèmes du système émulé vers le deuxième système. Ceci est facile pour des systèmes qui ont des appels systèmes voisins, comme les systèmes UNIX (au sens large) ou les systèmes Windows.

En plus d'être utilisé pour les appels systèmes, ce concept peut aussi être utilisé dès lors qu'il existe un "standard" (une API standard) (sur l'OS émulé). Par exemple FX!32 est un traducteur binaire (et une émulation) d'applications win32 sous x86 vers les plateformes Alpha/Windows NT. FX!32 contient des traducteurs pour beaucoup de DLL (les bibliothèques partagées du monde Windows), ce qui permet d'utiliser des DLL natives pour la machine cible plutôt que d'émuler de façon "généraliste" les librairies originales.

Dans le domaine des consoles de jeux (hormis les consoles récentes), des arcades et des ordinateurs, cette approche est limitée. Dans les premiers systèmes, il n'y avait pas de système d'exploitation, de librairie ou d'API, et la plupart du temps les développeurs programmaient eux-mêmes les librairies en relation avec le hardware. Donc dans ce cas l'émulation du hardware est nécessaire.

À présent, les constructeurs créent avec leurs consoles de gros kits de développement, et l'approche HLE devient intéressante. Les nouveaux systèmes commencent à avoir des systèmes d'exploitation (Windows CE pour la console dreamcast) et des API standard,mais ils ne sont pas toujours utilisés, les développeurs de jeu préférant souvent développer au plus près du matériel pour des questions de performances.

Dans le cas de la N64, cette approche est utile (en effet les partie graphiques et sonores de cette console sont programmées en un langage de haut niveau); mais cette implémentation est limitée par la difficulté d'implémenter toutes les fonctions graphiques et sonores de la console; c'est pourquoi actuellement seule une dizaine de jeux fonctionne correctement sur l'émulateur UltraHLE.

Étant donné que les parties de l'émulation programmées en HLE n'utilisent pas la

façon dont fonctionne le hardware mais plutôt ce qu'il fait, certaines personnes se refusent à parler d'émulation pour la technique HLE (par exemple le créateur de MAME).

3.3 Émuler la mémoire et les entrées-sorties

L'émulation de la mémoire et des entrées-sorties aussi est primordiale pour les performances de l'émulateur. En effet la mémoire est utilisée très fréquemment, il arrive même que de vieilles architectures avec peu de mémoire y accèdent à chaque instruction.

Les entrées-sorties sont moins utilisées, mais comme elles servent à communiquer avec les autres périphériques de la machine, leur implémentation peut ralentir considérablement les performances. L'émulation de la mémoire et des entrées-sorties est aussi l'émulation du bus de la machine, car elle émule la communication entre le CPU et les autres périphériques. Il n'est en général pas nécessaire de connaître le comportement réel du bus, mais il est parfois à prendre en compte pour reproduire fidèlement le système émulé.

L'émulation des entrées-sorties est implémentée avec l'émulation de la mémoire pour deux raisons: elles fonctionnent de manière similaire, et dans beaucoup d'ordinateurs les adresses des entrées-sorties sont déjà adressées dans l'espace mémoire. L'émulation CPU peut différencier les adresses mémoires de celles d'entrées-sorties, et en fonction du type de mémoire elle appelle une fonction spécialisée pour communiquer avec ce périphérique.

Un autre aspect à connaître est l'agencement de la mémoire (big endian ou little endian). Il faudra émuler cette caractéristique soit dans l'émulation du CPU, soit dans l'émulation de la mémoire. Bien sûr si le système que l'on émule et la machine hôte utilisent le même agencement de mémoire, le problème ne se pose pas et la mémoire peut être accédée directement. Dans le cas inverse cela peut avoir un impact important sur les performances de l'émulateur.

3.4 Émuler le son

Le son est surtout important lorsque l'on émule des plateformes de jeux (consoles et arcades). On peut distinguer deux types de sons : les effets tels que les explosions, les voix et la musique. Les premiers composants spécialisés pour la production de sons l'ont été, comme pour le graphisme, pour décharger le CPU de ces calculs; au départ ils ne pouvaient produire qu'un son fixe (un bip par exemple pour Space Invaders); puis ils ont évolués pour générer différentes tonalités et des sons synthétisés.

Le son ne nécessite que peu de données (comparé au graphisme) pour produire des effets et musiques de bonne qualité.

Sur de nombreux points, les composants graphiques et sonores se ressemblent. Bien sûr les algorithmes d'émulation sont différents (en effet, une image et un son ne sont pas codés de la même manière), mais les deux sont des composants dont le rôle est de décharger le CPU des calculs spécialisés et fonctionnent donc vis-à-vis du CPU de la même manière.

Les composants sonores les plus courants sont de type PSG et FM, MIDI et WAVE, ou basés sur un système d'échantillonnage et DSP.

Les systèmes PSG (Programmable Sound Generator) et FM (Frequency Modulation) produisent des sons plus ou moins complexes, mais la quantité d'informations qui leur est nécessaire est très faible. Un système PSG génère simplement des sons purs et la possibilité de changer le volume. Un système FM essaie d'émuler un instrument et comporte plus d'options que le système PSG. Il fonctionne par superposition d'ondes sinusoïdales. Le son ne paraît pas provenir d'un instrument, mais est tout de même de meilleure qualité que des sons PSG. Le système PSG code les sons sur 8 bits, tandis que le système FM les code sur 16 bits.

Les systèmes Midi et Wave fonctionnent sur le même principe que le FM, mais produisent des sons plus fidèles (à des instruments). Ils nécessitent également des composants plus puissants (et plus chers), et ne sont pas utilisés souvent sur les systèmes que l'on veut émuler.

Les composants basés sur l'échantillonnage utilisent un DAC(convertisseur de son digital vers analogique) pour produire du son. Ils ont besoin de beaucoup plus d'informations que les systèmes précédents (par exemple coder une seconde de son en qualité CD pèse 172 KO de données).

Les composants de ce type les plus simples envoient directement les données au DAC, mais il en existe des (un peu) plus évolués qui utilisent diverses formes de compression des échantillons, superposent différents sons et ajoutent des effets... Les plus avancés utilisent le système DSP(Digital Signal/Sound Processor) pour générer le son. Ce type de système est basé sur les interruptions et les timers. L'échantillon est placé dans un tampon, et quand il doit être joué, le composant commence la lecture du buffer pour le DAC; quand il arrive à la fin du tampon soit il génère une interruption soit il boucle et recommence la lecture du buffer.

S'il est évidemment primordial de connaître le système à émuler, il est également indispensable de connaître la machine qui devra exécuter l'émulation.

Dans les PC les composants de base sont basés sur des échantillonnages, mais la plupart des cartes sons peuvent aussi gérer le FM et le PSG (la Sound Blaster 16 standard utilise un processeur FM). Certaines peuvent également gérer le MIDI et le WAVE.

Sous les systèmes Microsoft Windows, le FM est difficile à utiliser et ne produit en général pas le son souhaité, donc la manière habituellement utilisée est de transformer le son en échantillons, de les superposer et de les envoyer directement à la carte son.

Le son est en général la dernière chose à programmer dans un émulateur; en effet c'est la partie la moins « vitale »: on peut jouer ou utiliser un programme sans son mais pas sans graphisme...

3.4.1 Exemple d'un Système PSG

Un système PSG est simplement composé d'un processeur de génération de sons qui possède plusieurs voies, et dont chacune d'elles peut être programmées pour rendre une fréquence et un volume particuliers. Certains de ces systèmes ont également d'autres options. L'onde résultante peut avoir n'importe quelle forme, il existe des systèmes PSG qui produisent des ondes sinusoïdales, carrées et même triangulaires. Le système PSG de la master system (console SEGA) est un processeur SN74489 (puce fabriquée par TI), avec une fréquence d'horloge de 3,57MHz. Elle possède quatre voies, trois pour produire des sons particuliers, et une pour les bruits de percussion et les explosions. Cela correspond à une configuration de console standard (pour l'époque). Cette puce est la plus basique de celles que l'on peut trouver dans une console de jeux. Elle produit des ondes (modulées en signal) carrées, qui peuvent être paramétrées par leur fréquence, de 122Hz à 125kHz, et par leur volume qui possède 16 niveaux.

Pour implémenter une onde carrée, on utilise un registre que l'on décrémente (ou incrémente), et on change le signal de 0 à 1 ou de 1 à 0 chaque fois que l'on arrive à 0. Le contenu initial (et à chaque fois que l'on remet le registre à jour) du registre est la fréquence de l'onde que l'on souhaite produire. Le signal de sortie de chaque voie est atténué par le volume choisi, puis les quatre voies sont mises ensemble.

La façon d'émuler ce système est de générer des échantillons pour chaque voie, les mettre ensemble puis les mettre dans un buffer pour faire jouer le son par la carte son. Les échantillons calculés pour chaque voie seront simplement des valeurs O ou 1 (ou -1 et 1 si on a choisi de travailler en signé) multipliées par le volume souhaité.

Étant donné que le volume est un facteur d'atténuation (un volume de

15 signifie aucun son et un volume de 0 le son maximal), la valeur d'un échantillon se calcule grâce à la formule VOLUME_MAXIMAL * (15 - volume_des_voie[i]) * valeur_des_voies[i]. Pour actualiser la valeur des voies, on a un compteur qui contient initialement le nombre de cycles avant le prochain changement (le diviseur de fréquence).

Ce compteur est décrémenté d'un nombre de cycles (passé à l'émulateur), chaque fois que l'émulateur du système PSG est appelé. Il me paraît vraisemblable que la valeur que l'on décrémente soit liée au nombre de cycles du CPU émulé entre chaque appel du son. Quand notre compteur devient inférieur ou égal à 0, on change la valeur de la voie en question, et on rajoute au compteur le diviseur de fréquence (il est important de prendre en compte la différence avec 0 pour rester synchronisé avec le CPU et obtenir un son de la bonne fréquence).

Une fois que l'on a créé les échantillons pour chaque voie, il ne reste plus qu'à les mixer. Si on a choisi de travailler en signé, il suffit de les additionner en prenant en compte les saturations possibles. Sinon c'est plus complexe, il faut prendre en compte une valeur centrale en plus de la saturation. La façon la plus simple est donc de travailler en signé, et si notre carte son travaille en non signé de convertir ensuite notre signal en non signé.

4 M.A.M.E (Multi Arcade Machine Emulator)

Mame est un projet visant à créer un émulateur permettant de faire tourner un maximum de ROMs (Cf.glossaire) d'arcade

Le 24 décembre 1996, Nicola Salmoria a commencé à travailler sur plusieurs émulateurs de jeux basés sur un seul et même hardware (ex : Multi-Pac, différents clones de Pac-Man), les projets ont été par la suite regroupés en un seul programme courant Janvier 1997. Il nomma le tout "Multiple Arcade Machine Emulator", ou "MAME".

La toute première version fut MAME v0.1. Utilisant une architecture portative avec des drivers modulaire, il ne manquait plus que le projet soit Open Source pour qu'il prenne très vite une ampleur considérable. La version en cours contient 3749 sets de ROMs, pour 2142 jeux (à comparer aux 42 jeux du projet sparcade, un projet d'émulaion de bornes d'arcade qui n'est pas open source).

Même si la plupart des utilisateurs de MAME y voient juste un gros émulateur permettant de jouer à de vieux jeux d'arcades, le but principal du projet est de documenter le hardware des machines d'arcade. Il y a beaucoup de bornes d'arcades qui n'existent plus physiquement et que MAME a fait revivre. Pouvoir jouer à des jeux est juste un agréable effet de bord...

4.1 Architecture de MAME

L'architecture de MAME est basée sur des drivers modulaires. C'est-à-dire qu'il existe (par exemple) un driver pour le Z80, un autre pour le 68000, un autre pour tel ou tel composant sonore... Lorsqu'une personne veut émuler un jeu, il faut donc qu'elle regarde quels sont les composants (y compris le/les CPU) pour lesquels il existe des drivers et quels sont ceux pour lesquels il faudra créer un nouveau module (par exemple s'il n'existe pas de module émulant le composant son de l'arcade), puis écrire le code de l'émulateur proprement dit (en utilisant les modules correspondant aux composants de l'arcade).

Tout ceci, que ce soit l'émulation CPU ou l'émulation sonore est écrit en C, ce qui confère à MAME une très grande portabilité (existe pour les systèmes Windows, compatibles POSIX, et il existe même une version pour certains appareils photos numeriques!!).

MAME est programmé dans un langage de haut niveau, ce qui peut porter à croire que ses performances sont mauvaises; mais MAME est un émulateur de machines d'arcade, qui sont des machines (en général) peu puissantes (par rapport aux machines actuelles), même en considérant qu'il faut émuler tous les processeurs de l'arcade (cpu principal, graphisme, son et autres...) avec le seul CPU de l'ordinateur.

D'après certains renseignements des newsgroups, mame est difficilement utilisable sur un pentium 90, mais fonctionne correctement sur un pentium 150 (pour les jeux qui n'ont pas besoin d'émuler de la 3d, c'est à dire la plupart des jeux).

4.2 Les Roms de MAME

Pour obtenir les ROMs des jeux, MAME fait appel à des donateurs. Si jamais vous remarquez que tel ou tel jeu d'arcade n'est pas disponible pour MAME, il vous faut trouver la PCB (Prined Card Board, la carte mère de l'arcade) de cette arcade et l'envoyer à « the guru », le principal dumpeur de MAME. En fait, c'est un peu plus compliqué, il faut que le jeu se trouve sur la liste des jeux souhaités pour MAME, et trouver la PCB peut être très difficile, certaines sont très rares et ont une grande valeur!

En plus de la difficulté à trouver les PCB, elles sont souvent dotées de protection(s) contre le dumping (Cf.glossaire).
Une des méthodes les plus vicieuses, incrackable avant mame, était une carte branchée sur la carte mère. Pendant l'exécution, le jeu (la ROM) vérifiait la présence de cette carte et éteignait la machine si elle n'était pas présente. C'est pourquoi pendant un certain temps, certains jeux dont la ROM était disponible

mais impossible à émuler.

Une autre manière de protéger les ROMs était de les crypter. La carte mère décodait la ROM pendant l'exécution, en général à l'aide d'un processeur spécialisé.

Certains systèmes sont allés jusqu'à mettre la clé du cryptage dans de la RAM, ainsi si un « dumpeur » enlevait la RAM où était stocké la clé, elle s'effaçait et la borne d'arcade devenait du même coup inutilisable.

Bien sûr la difficulté du dumping était encore plus grande quand les constructeurs utilisaient une combinaison de ces systèmes. Une puce, construite par Dallas semiconductor, combinait un système de cryptage et un système d'effaçage de la clé.

Le problème de ces puces spécialisées est qu'elles sont utilisées de la même manière sur différentes bornes d'arcade, il est donc plus facile d'inventer une méthode de crackage que lorsque la protection est différente pour chaque arcade. Plusieurs méthodes ont été inventées pour contourner les protections de cette puce, notamment une méthode inventée par Makus Kuhn, disponible sur son site (http://www.cl.cam.ac.uk/~mgk25/).

4.3 Extensions

La gestion des Rom pour MAME s'effectue par défaut par l'utilisateur mais il est possible d'installer un front-end. En effet un front-end permet de gérer vos ROM , il peut les trier, les renommer et effacer certains types de jeux.

Certain front-end permettent aussi de paramétrer MAME (version DOS : MAME classic) pour chaque jeu, afficher la liste de vos jeux, les sample, alternate rom et le fichier history.dat.

Le fichier history.dat permet à MAME de vous donner de nombreuses informations sur les jeux informations complémentaires à celles données par mameinfo (code secret, description des jeux...). Le fichier MameInfo.dat permet à MAME de vous donner la qualité d'émulation de chaque jeu, vous donne quelques infos comme l'histoire et possède des listes de jeux recommandés.

Il est également possible de rajouter (ou d'enlever pour optimiser les performances) des drivers à MAME lors la compilation. En effet, certains jeux nécessitent des drivers non compilés par défaut (et parfois en phase de test). Il faut pour cela ajouter/enlever les lignes des drivers dans le fichier driver.c, et remplacer TESTDRIVER par DRIVER pour les drivers de test à rajouter.

4.4 Contribuer au projet

Il existe plusieurs façons de contribuer au projet MAME:

-Écrire des modules qui n'existent pas encore dans MAME.

-Corriger des bugs dans les modules, bugs qui sont annoncés par les testeurs (sur le site www.mametesters.com).

-Acheter une PCB recherchée par les développeurs du projet et l'envoyer au dumpeur de MAME, the guru.

-Donner directement des fonds pour l'achat de PCB, ce qui permettra d'enrichir la bibliothèque des roms de MAME.

Conclusion

Ce dossier nous a permis de comprendre ce qu'est un émulateur, et surtout comment procède l'émulation; nous avons étudié ce qu'on peut émuler, les différentes techniques d'émulation, ainsi que leurs avantages et leurs inconvénients (performances, difficulté d'implémentation ...).

Nous avons aussi étudié les différents émulateurs existants, nous avons pu ainsi voir quels étaient les meilleurs et surtout pourquoi (technique d'émulation, projet OpenSource ou non...).

Ces recherches nous ont également permis de mieux comprendre dans l'histoire de l'informatique le développement de la compatibilité des machines (le premier émulateur ayant était créé dans un souci de compatibilité).

L'émulation est un procédé très intéressant et instructif. Cependant certaines réalisations des programmeurs ne plaisent pas à tout le monde et notamment aux géants de la console (Nintendo, Sony...), surtout depuis la sortie d'émulateurs de consoles récentes, qui peuvent avoir un impact économique important sur les constructeurs. On est alors en droit de se demander si le public habituel des consoles pourrait s'en détourner, sachant que l'émulation necessite un ordinateur, qui est moins intuitif qu'une console de jeu (demarrage, configuration, bugs...).

Glossaire

BIOS :

Basic Input Output System.

Dumping :
Le dumping est le fait de mettre une copie du code du programme que l'on émule sur le disque (variant entre 200Ko et 15 Mo), ces copies sont appelées ROMs (Read Only Memory).
L' importation se fait généralement à l' aide d' un dumper (machine prévue pour copier des jeux).

Émulateur :

Programme qui imite le comportement d'une machine ou d'un logiciel sur une autre machine, en se basant sur le hardware de la machine émulée.

Émulation :

"Nom féminin, Sentiment qui porte à égaler ou surpasser quelqu'un : encourager l'émulation chez les enfants. (Synonyme : Compétition, Concurrence, lutte)."

Désigne également le fait de créer les émulateurs: la théorie de l'émulation.

Frontend :

Interface Graphique crée pour les émulateurs qui n'en possède pas (Ex : MAME).

Larry Moss :

Le père de l'émulation.

Open-GL :

Librairie Graphique utilisé par les logiciels de 3D.

Rom :
Dans le domaine de l' émulation, une rom (Read Only Memory) est un fichier résultant de l' importation d' un jeu sur votre machine, contenant toutes les informations de ce jeu.

Comme vous le savez, vous ne pouvez pas mettre de cartouches dans votre PC d' où l' utilisation des roms pour jouer avec les émulateurs. Généralement pour les systèmes à CD (comme la Playstation) il n' existe pas de roms, vous devez donc utilisez les CD originaux.

2xSai (Mode) :

Signifie 2 fois Scale And Interpolation. Technique d'affichage utilisée par certains émulateurs. Pour rendre un jeu plus beau sur PC qu'il ne l'est dans la réalité (valable uniquement avec un moniteur).

Abandonware :

Terme désignant d'anciens logiciels souvent disponibles sur Internet. Le plus communément, on considère qu'un logiciel entre dans la catégorie des abandonwares s'il répond à un certain nombre de critères. Notamment : sa date de sortie est supérieure à cinq ans, le logiciel n'est plus disponible dans le commerce. Certains rajoutent des critères plus restrictifs comme l'indisponibilité auprès de l'éditeur lui-même ou encore l'abandon écrit par la personne morale ou réelle qui détient les droits. Dans la réalité, le terme abandonware n'existe pas au niveau juridique et la position des éditeurs à son sujet reste très variée, certains soutenant ce phénomène, d'autres le combattant. La sortie d'anciens titres sur Gameboy Advance (*Super Ghouls & Ghosts* ou *Zelda 3*) relance le débat, au détriment des supporters de l'abandonware.

Bibliographie

http://fms.komkon.org/EMUL8/HOWTO.html: Un document très instructif sur comment écrire un émulateur, écrit par Marat Fayzullin

http://www.worldofspectrum.org/EmuFAQ2000/ : Un document sur la légalité des émulateurs.

http://howtoemulation.emuhq.com/general.shtml : Un site contenant d'excellents dossiers dont un de Marat Fayzullin , et un de Victor Moya del Barrio : study of the techniques for emulation programming, une bible détaillée (152 pages) de la programmation d'un émulateur, mais pas toujours très claire.

http://www.liacs.nl/~atijms/bintrans.pdf : « Binary translation : Classification of emulators » par Arjan Tijms, un document pour bien cerner les concepts d'émulation et de traduction, et des détails intéressants, notamment sur la fidélité de l'émulateur par rapport à la machine émulée, et sur HLE.

http://creezvosjeux.online.fr/ : Ce site contient un petit guide pour développer sur Dreamcast.

http://www.mame.net/ ou **http://www.mameworld.net/ :** Des sites entièrement consacrés à l'émulateur MAME, dernière et ancienne version, faq…

http://www.emulfrance.free.fr/ : Des informations sur l'émulation et les jeux vidéo. PC, Consoles, Arcade, Ordinateur, sur les émulateurs, utilitaires, frontends, INI, Pug-ins… Ainsi que des articles et des liens vers des sites de roms.

http://www.emu-france.com/ : idem emulfrance mais avec un super dossier sur comment compiler MAME.

http://perso.wanadoo.fr/desby02/ : Un super article sur l'origine de l'émulation. L'historique de beaucoup de console (snes, megadrive, gameboy…).

http://www.zophar.net/index.phtml : Contient pratiquement tous les émulateurs , avec la possibilité de télécharger les sources.

http://www.emulation.fr/ : Un site internet dédié à l'émulation logicielle sur toutes les plates-formes. Ce site propose de répondre à vos questions.

http://site.ifrance.com/webemulation/liens/emu.html : Plein de liens sur l'émulation.

http://www.01net.com/ : Un site contenant toujours des informations et quelques émulateurs à télécharger.