Travail d'étude sur l'émulation

David BONFILS & Laurent TOSELLI

Licence d'informatique 2002-2003
Université de Nice–Sophia Antipolis

16 juin 2003

Table des matières

1 Un peu d'histoire

1.1 Introduction

Vous savez tous ce qu'est l'émulation, mais l'origine de cette technologie, ainsi que l'explication de ce mot à l'étymologie incertaine, et même inexacte, est floue. En effet, l'émulation représente systématiquement un retour en arrière pour un ordinateur, une habile manipulation logicielle et/ou matérielle visant à le transformer en un de ses ancêtres plus ou moins éloignés dans le temps et dans leur architecture. Alors pourquoi parler d'émulation ? On va voir qu'une fois de plus, une technologie informatique qui nous semble si moderne va s'avérer remonter à une époque bien lointaine.

1.2 Historique

De la fin de la deuxième guerre mondiale aux débuts des années 80, il est un fait qu'IBM[1] a régné sur le monde de l'informatique. Dans les années 50/60, des appareils tels que l'IBM 1401 font figure de référence absolue en matière d'ordinateurs, même si des compagnies comme Digital, Sperry-Rand, Burroughs ou Honeywell proposent d'autres produits qui sont parfois plus performants, mais ne peuvent se prévaloir du label « Big Blue ». Conscients que cette suprématie repose plus sur une réputation que sur les performances réelles de leurs machines, les dirigeants d'IBM initient alors une étude connue aujourd'hui sous le nom de SPREAD Report, qui a pour but d'aboutir à la conception d'un produit totalement nouveau, qui pourrait distancer durablement la concurrence. On vit alors une époque où les ordinateurs sont conçus comme des machines totalement indépendantes les unes des autres. Il n'est question ni de composants partagés, ni de compatibilité. Chaque système nécessite donc ses propres données logicielles, son propre hardware[2] adapté à l'usage que l'on veut en faire et les responsables de la recherche d'IBM voient là une piste à explorer.

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 est à l'époque

1. International Business Machines, compagnie fondée en 1911
2. Terme anglophone désignant le matériel informatique

où IBM s'apprête à lancer sa NPL[3], série d'ordinateurs révolutionnaires et fruit de son effort de développement technologique. On connaît aujourd'hui, notamment grâce aux systèmes d'exploitation Microsoft (tous basés, jusqu'à Windows NT et 2000 sur l'inévitable MS/DOS pour les rendre compatibles), les répercutions d'une telle politique sur les ventes d'ordinateurs 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). Il n'y a ainsi plus besoin d'accompagner la sortie d'un nouvel ordinateur d'une batterie de logiciels, puisqu'il peut se contenter de ceux de ses prédécesseurs en attendant d'avoir une logithèque conséquente.

Fin Septembre 1963, IBM confie à un des ses laboratoires situé à La Gaude, en France, la mission de développer une série de programmes de simulation ayant pour but d'imiter le comportement de ses sept ordinateurs les plus populaires, et d'établir une *compatibilité descendante* entre eux, le tout sur le plan purement logiciel. Les premiers résultats sont décourageants. La simulation a un fonctionnement au mieux deux fois moins rapide que les ordinateurs en question, et elle est 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 un an, les chercheurs d'IBM vont développer les ordinateurs de la NPL dans ce sens. Un ingénieur nommé Stuart Tucker est nommé responsable de la *compatibilité descendante*.

Pendant ce temps, les concurrents d'IBM ne se sont pas endormis. Honeywell sort le *H-200*, un ordinateur capable de faire fonctionner des programmes IBM, un évènement qui met un peu plus la pression sur Big Blue. Il n'est plus seulement question de *compatibilité descendante* au sein d'une ligne d'ordinateurs de même marque, mais carrément de compatibilité avec d'autres systèmes, d'autres marques. John Haanstra, un autre ingénieur chez IBM peu favorable aux projets de Tucker et à la NPL, propose alors une solution hardware, qui va aboutir à l'*IBM 1410*, 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, et ouverts sur les autres systèmes grâce à son adaptabilité logicielle.

L'équipe de Tucker, basée à Poughkeepsie, New Jersey, comprend un jeune talent très prometteur : Larry Moss. Sa théorie est que pour qu'un ordinateur puisse être compatible avec un autre, le hardware des deux machines doit être

3. New Product Line

FIG. 1 – *L'IBM 7070*

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 tel que celui sur lequel l'équipe Française de La Gaude à déjà travaillé. Il se lance donc dans l'étude d'une solution de compatibilité mixant les approches matérielles et logicielles qui ont opposé Tucker et Haanstra. 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 d'alors (en passe bien sûr d'être surpassé par les ordinateurs de la NPL).

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 » développée à La Gaude, de lui conférer l'appellation d'« émulation », expression indiquant une amélioration, une avancée. Tucker approuve le projet, qui aboutit le 7 avril 1964 à la sortie de la famille d'ordinateurs *IBM System/360*, et de l'émulateur Moss, connu aussi sous le nom de « 7070 Emulator », dont Joe Brown et Larry Moss sont les pères. L'appareil va connaître un franc succès, les possesseurs de System/360 étant ravis de continuer à utiliser leurs anciens programmes pour 7070, et sa commercialisation durera jusqu'au début des années 70.

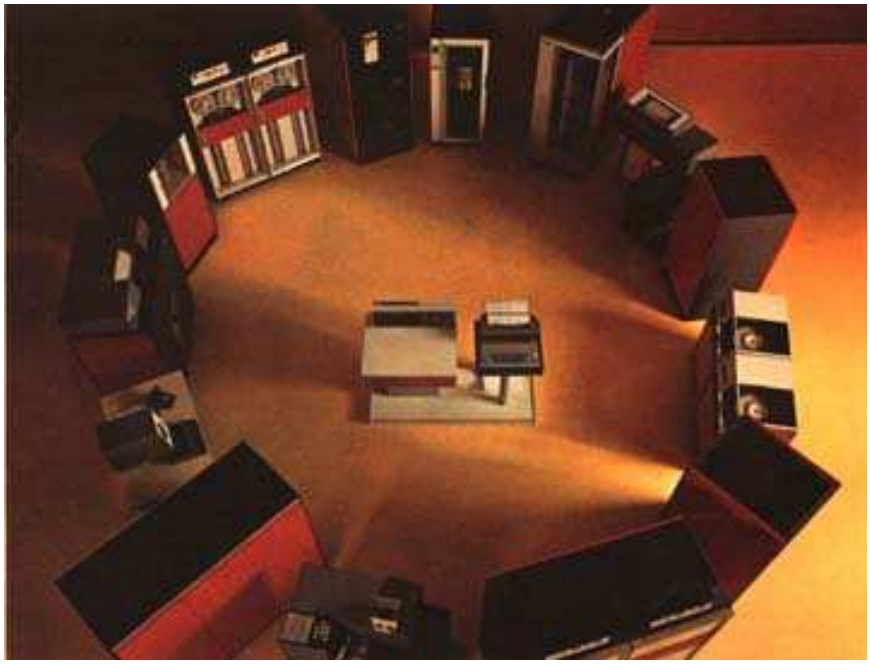

FIG. 2 – *L'IBM System/360*

Il faut savoir qu'il est indispensable d'émuler une machine à l'aide d'une autre plus puissante en terme de microprocesseur.

Ceci explique pourquoi la tentative fut peu renouvelée dans les années 80 (ordinateurs disposant de ressources processeurs limitées et équivalentes). À la fin des années 80 sort « spectre » qui permet à un Atari ST de restituer partiellement l'OS[4] du *Macintosh*... L'avènement des ordinateurs 16 bits marque un point clé pour l'émulation : Atari, Amiga et Mac profiteront d'émulateurs PC[5]...

C'est au cours du milieu des années 90 que les émulateurs tels qu'on les connait firent leurs apparitions. C'est à partir de là qu'il fut possible de restituer un OS complet de manière logicielle sans ajout de add-on[6].

L'émulation constitue à l'époque un travail minutieux et de nombreuses documentations très difficiles à extirper aux constructeurs. S'ensuit ensuite

4. Operating System, système d'exploitation
5. Personal Computer
6. Sorte de patch, servant à améliorer le fonctionnement d'un système, en lui ajoutant une ou plusieurs fonctions

le processus de « *reverse engineering* », c'est-à-dire la décompilation des données afin de suivre pas à pas son exécution, méthode très efficace mais digne d'un travail de fourmi (*Cf. chapitre 2*). Parallèlement, les consoles ne furent pas en reste, et l'on vit des programmes tels que *MAME* (*Cf. Chapitre 6*)[7] pointer le bout de leur nez...

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, le terme a retrouvé son sens premier, puisque c'est une véritable course que se livrent les petits génies de la programmation et du décorticage électronique. De nouveaux émulateurs sortent chaque mois, et certaines machines font l'objet d'une multitude de projets d'émulation donnant tous presque 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 du piratage.

7. Multiple Arcade Machine Emulator

2 L'émulation en théorie

2.1 Quelques définitions

Voici quelques définitions et les distinctions entre ces différentes notions :

2.1.1 Émulation

L'émulation (à ne pas confondre avec la simulation), est en fait un procédé très simple (en théorie), qui consiste à reproduire sur un ordinateur le système d'une autre machine. En terme plus technique, l'émulation est la possibilité d'émuler du hardware sur un autre hardware via du software[8]. Par exemple, émuler une *PlayStation* sur un PC consiste à transformer un PC en *PlayStation* afin de pouvoir utiliser ses jeux. On va pour cela recréer le hardware PSX sur un PC uniquement au moyen d'un petit programme (software) que l'on appelle émulateur.

2.1.2 Simulation

La simulation quant à elle, consiste à imiter les fonctions d'un système. Par exemple, un programme imitant le hardware du Pacman d'arcade et utilisant la ROM[9] réel de Pacman est un émulateur. Un jeu Pacman écrit pour votre ordinateur mais ayant des graphismes similaires au jeu d'arcade est un simulateur.

2.1.3 ROMs

Copie d'un logiciel, souvent une cartouche de jeu provenant d'une console de jeux vidéo. En émulation, on parle de ROM pour les supports non réinscriptibles à l'origine, comme les cartouches de consoles. Dans l'émulation des ordinateurs (8 ou 16 bits), la ROM désigne bien souvent le système en mémoire. Les jeux, eux, sont des images-disques (pour les disquettes) ou images-cassettes (pour... Les jeux sur K7 !), on parle donc en général, et pour faire court, d'images.

8. Terme anglophone désignant un logiciel informatique
9. Read Only Memory, mémoire morte

2.1.4 Bootleg

Dans les émulateurs de bornes d'arcade (comme MAME par exemple), quelques ROMs portent la définition de "bootleg". Cela veut dire que le programme trouvé dans les ROMs n'est pas l'original : quelqu'un a construit une machine avec un hardware similaire, mis son nom sur l'écran de présentation du jeu et gagné de l'argent sans la permission du possesseur du "*copyright*"[10]. À noter que ces versions ont souvent leur protection "crackée" et peuvent fonctionner sur des émulateurs quand les ROMs originales, elles, ne marchent pas.

2.1.5 Sample

La plupart des sons dans les jeux (musiques, explosions, ...) sont émulés. Mais quelques jeux (peu nombreux, pour la plupart des émulateurs d'arcade) requièrent des fichiers samples, qui contiennent des enregistrements digitalisés des sons originaux provenant de la machine. Ils ne sont pas forcément essentiels, mais cela ne pourra qu'améliorer certains jeux.

2.1.6 Abandonware

Le terme s'applique aux jeux anciens qui ne sont plus commercialisés mais trouvent une seconde jeunesse en téléchargement gratuit sur Internet. La pratique est illégale mais globalement tolérée, voir encouragée par certains éditeurs qui abandonnent volontiers leur copyright (d'où le mot "abandonware") sur leurs jeux désormais invendables. À distinguer du **piratage** (ou **warez**, dans le jargon informatique) qui s'attaque quant à lui aux jeux encore en vente. http://www.abandonware-france.org

2.2 Que peut-on émuler ?

Ainsi, nous pouvons nous poser la question suivante, à savoir : qu'est-ce qui peut être émulé ? En théorie, tout ce qui contient un microprocesseur : une console, une calculatrice, une borne d'arcade, un ordinateur... L'émulation, ce n'est pas copier sur un ordinateur le système concerné, mais le reproduire fidèlement, fonctionner comme lui. Pour émuler un système, il faut connaître son architecture : processeur, puces, BIOS[11], etc. Pour les ordinateurs, des

10. Terme anglophone désignant la protection des droits d'auteurs
11. Basic Input/Output System

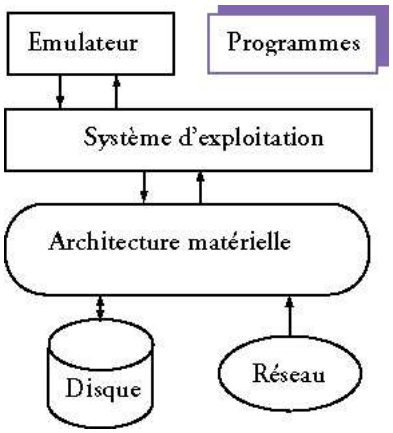

FIG. 3 – *Schéma de l'émulation logicielle*

manuels existent. Pour les consoles ou bornes d'arcades, les informations sont plus rares. Les consoles ont souvent des processeurs multiples, servant uniquement à certains types d'informations (images, sons, mémoires...). Sans informations détaillées, le développeur ne peut faire reproduire à son émulateur les fonctions de tel processeur, d'où, certains émulateurs incomplets (émulateur `Neo-Geo` sans le son). D'autres, émulateurs comme `LazyNES` (émulateur NES sous Windows) utilise DirectX pour l'émulation graphique, sonore, et propose même un mode multijoueurs sur internet. Certains programmeurs ont accès aux données confidentielles des machines, mais il faut pour cela en faire la demande au constructeur et signer une décharge qui vous interdit de livrer quoi que ce soit au public. D'autres pratiquent le « *reverse engineering* » méthode qui consiste à décrypter un programme à l'envers pour comprendre en tâtonnant comment il fonctionne à l'endroit. Ainsi, un émulateur peut fonctionner de trois façons différentes (*Cf. chapitre 4*) :

1. Par *interprétation*
2. Par *recompilation statique*
3. Par *recompilation dynamique*

Nous allons alors montrer (*Cf. chapitre 4*) que ces trois méthodes ne sont plus les seules. En outre, il est à noter qu'il existe plusieurs styles d'émulations :

- L' *émulation logicielle.*
- L' *émulation matérielle.*

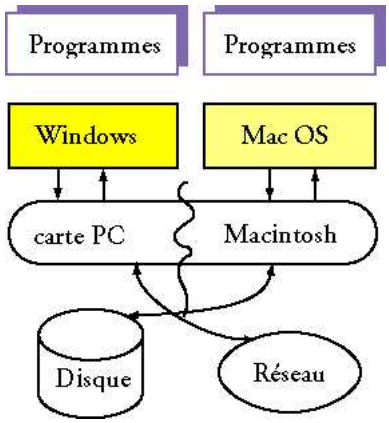

FIG. 4 – *Schéma de l'émulation matérielle et virtuelle*

– L' *émulation virtuelle*.

Bien entendu, il existe plusieurs logiciels d'émulations, à savoir:

– pour les ordinateurs, nous avons *VMWare*, *Plex86*, etc. qui sont associés à la création de machine virtuelle plutôt qu'à de l'émulation proprement parlé qu'on retrouve pour les PC. En effet, *VMWare* n'émule pas un systéme d'exploitation particulier mais une machine virtuelle avec des composants virtuels. *Bochs*, quant à lui, est considéré comme un véritable émulateur de PC (il existe à la fois pour PC et pour Mac). À noter que pour Mac, il existe également le très célèbre *Virtual PC*. Par principe, un émulateur de type *VMWare* s'exécutera plus rapidement qu'un émulateur de type *Virtual PC*, ce dernier occupant une charge au CPU[12] beaucoup plus importante dû à son architecture machine. En effet, l'architecture d'un Mac diffère d'une architecture PC. Notons également, l'existence de MESS[13] qui se trouve être un émulateur destiné aux anciens ordinateurs (`Amstrad 6128plus`, `Apple I`, `Atari 2600`, ...) qui est l'équivalent de MAME pour les bornes d'arcades.
– pour les consoles, la liste des émulateurs est longue : *ZSNES*, *Snes9X* pour la *Super Nintendo*, *PSEmu Pro* pour la *PlayStation*, *Project64*, *UltraHLE* pour la *Nintendo 64*, *NeoRage X* pour la *Neo-Geo*, etc.
– pour les calculatrices, nous retrouvons le génialissime *Virtual TI* qui permet l'émulation des calculatrices scientifiques de chez `TI`[14], mais

12. Central Processing Unit, microprocesseur
13. Multiple Emulator Super System
14. Texas Instrument

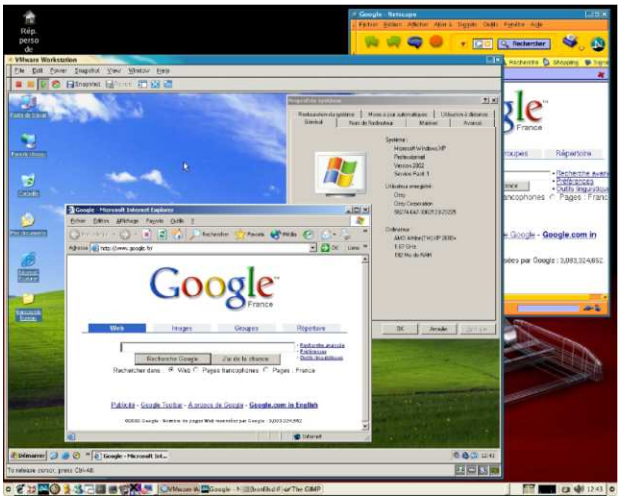

FIG. 5 – *WMWare 4.0 WorkStation sous Linux*

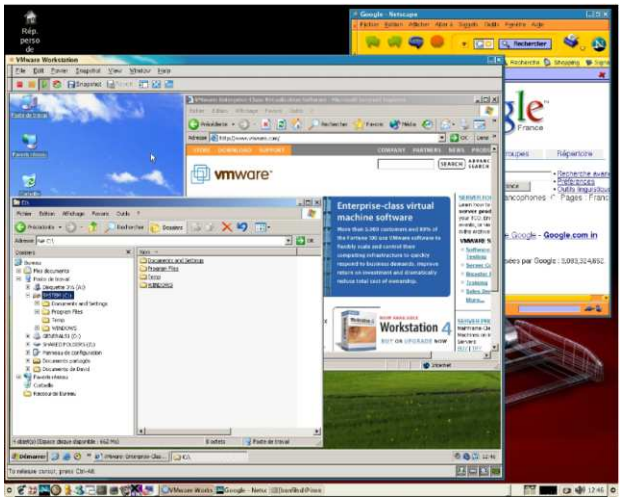

FIG. 6 – *VMWare 4.0 WorkStation sous Linux*

FIG. 7 – *L'émulateur Nintendo 64 - Project64*

aussi *Emulateur HP 49G* pour la calculatrice HP-49G de chez HP[15], ainsi que tant d'autres ...

– enfin parmi les émulateurs de bornes d'arcades, *MAME* est le plus abouti de tous. Développé par Nicola Salmoria et toute son équipe, *MAME* supporte des centaines de jeux (ROMs), des plus anciens aux plus récents. Différentes versions existent, optimisées pour un type de matériel bien précis. Ainsi vous trouverez une version Windows, une version DOS, une version optimisée Intel, une autre pour AMD, etc. Les paramétrages possibles pour votre matériel sont on ne peut plus complets et précis. Le gros avantage de *MAME*, mis à part le nombre de jeux qu'il émule, est qu'il peut très bien tourner sur un 486 dans sa version DOS. Pour tous les nostalgiques des machines d'arcade, *MAME* est la solution ultime. Pour y avoir un aperçu vous pouvez vous rendre à l'adresse http://perso.wanadoo.fr/arcaderevival/. Bien sûr, il ne s'agit pas du seul émulateur. D'autres noms plus ou moins connus à l'image de *Modeler*, *Raine*, *Zinc*, etc., peuvent également être cités. Pour plus d'informations sur les émulateurs cités (et même d'autres), cette adresse vous donnera entière satisfaction : http://www.emultoo.com/

15. Hewlet-Packard

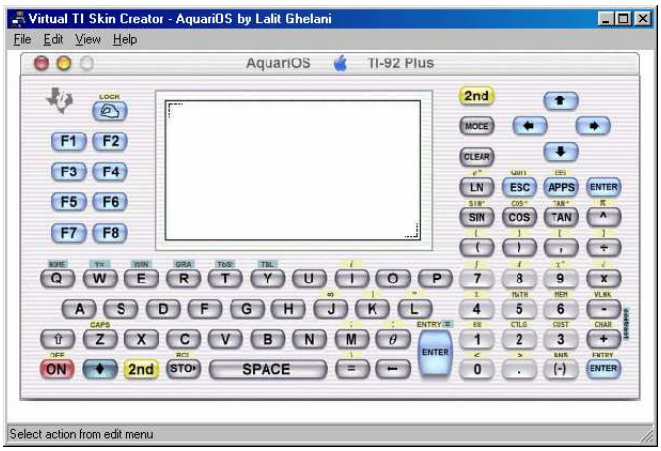

FIG. 8 – *Habillage Apple pour la Virtual TI*

FIG. 9 – *Borne d'arcade (avec logo MAME)*

3 Et la loi dans tout ça?

3.1 Introduction

Selon la loi, l'émulation n'est pas illégale, par contre certains émulateurs ont besoin pour fonctionner du BIOS des machines originales, et celà est illégal. De même, le « *dumping* », méthode consistant à relier des boîtiers au PC par un câble dans lequel on insère les cartouches pour en faire une image sur le disque, appelés ROMs, n'est pas en soi une entorse à la loi, mais c'est leur distribution en masse et leur utilisation qui peuvent poser des problèmes.

3.2 Les grosses rumeurs

La rumeur la plus courante, c'est que la loi accepterait le téléchargement de ROMs pendant vingt-quatre heures, soit une journée entière. Cette rumeur n'est fondée sur aucune loi. Il est, en France du moins, et dans la majorité des pays à politique moderne, strictement interdit de posséder un logiciel (un jeu est un logiciel, même sur cartouche) sans la LICENCE. Ce qui nous amène à la deuxième "rumeur".

On dit souvent qu'il est obligatoire de posséder la cartouche ou le CD original, c'est évidemment *faux*, AU CONTRAIRE. Ce que vous achetez, en magasin, c'est comme pour les ordinateurs : ce n'est pas le CD en lui-même (enfin si évidemment, mais ça n'est qu'une infime partie), mais la LICENCE, c'est-à-dire, en gros, le droit de possèder et d'utiliser le jeu. Par conséquent, ce qu'il vous est nécessaire, c'est la licence. La licence est soit sur un papier à part (en général c'est plutôt pour les logiciels PC), soit, le plus souvent, une preuve d'achat. Autrement dit, si vous prouvez que vous avez acheté le jeu à l'aide de n'importe quel papier (je ne pense pas que le ticket de caisse suffise, mais ça n'est pas impossible), vous êtes PLUS OU MOINS en droit de posséder une ROM.

Dans les milieux un peu plus sérieux, on parle du fait que les cartouches et les CD sont considérés comme un moyen de conservation fiable, et que par conséquent, nous n'avions pas le droit à une copie de sauvegarde. Aucune loi n'en fait état, en réalité. Il y'a certes eu un procés qui a été perdu à cause de cet argument et qui fait jurisprudence[16], mais il ne s'applique que dans le cas où vous possédez la cartouche ou le CD original EN PLUS de la copie de sauvegarde. Si, par exemple, vous vous êtes tout simplement fait voler votre

16. Ensemble de décisions de justice

CD, cette rumeur ne s'applique plus.

3.3 Licence et droit ?

En fait, c'est trés compliqué. Le code de la propriété intellectuelle stipule que tout possesseur d'un logiciel (sous-entendu la licence), vous avez le droit à une copie de sauvegarde, voici ce texte:

Article L122-5
Lorsque l'oeuvre a été divulguée, l'auteur ne peut interdire :
2°- Les copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective, à l'exception des copies des oeuvres d'art destinées à être utilisées pour des fins identiques à celles pour lesquelles l'oeuvre originale a été créée et des copies d'un logiciel autres que la copie de sauvegarde établie dans les conditions prévues au II de l'article L. 122-6-1 ainsi que des copies ou des reproductions d'une base de données électronique;

Donc, on n'y apprend que toutes les copies pour usage **privé**, c'est-à-dire UNIQUEMENT pour celui qui fait la copie ou fait copier le produit, sont légales, sauf quelques détails expliqués dans l'article L. 122-6-1. Que nous dit cet article ?

Article L122-6-1
I. Les actes prévus aux 1° et 2° de l'article L. 122-6 ne sont pas soumis à l'autorisation de l'auteur lorsqu'ils sont nécessaires pour permettre l'utilisation du logiciel, conformément à sa destination, par la personne ayant le droit de l'utiliser, y compris pour corriger des erreurs. Toutefois, l'auteur est habilité à se réserver par contrat le droit de corriger les erreurs et de déterminer les modalités particulières auxquelles seront soumis les actes prévus aux 1° et 2° de l'article L. 122-6, nécessaires pour permettre l'utilisation du logiciel, conformément à sa destination, par la personne ayant le droit de l'utiliser.
II. La personne ayant le droit d'utiliser le logiciel peut faire une copie de sauvegarde lorsque celle-ci est nécessaire pour préserver l'utilisation du logiciel.

Là, on y apprend que SI et seulement SI les auteurs ou éditeurs précisent que EUX s'opposent à la copie, et seulement À CONDITION qu'ils proposent UNE SOLUTION DE RECHANGE, ce qui n'est jamais le cas, à ma connaissance. TOUTEFOIS, si jamais l'éditeur s'engageait à vous fournir une copie du logiciel sur demande, vous ne pourriez pas utiliser de ROM. Si ce n'est pas le cas, ALORS vous auriez le droit de posséder UNE ROM du jeu... OUI MAIS :

3.4 Oui, mais ?

Des sociétés comme Sega considèrent que le CD que l'on vous fournit est déja la copie de sauvegarde. Par conséquent, nous n'avons le droit à la ROM ou à l' ISO[17] que si nous n'avons plus cette copie de sauvegarde. TOUTEFOIS, `SEGA` n'arrivera pas à prouver dans un tribunal que la preuve d'achat original inséré dans la *`Dreamcast`* suffit à jouer au jeu. Par conséquent le CD original est plus qu'une copie de sauvegarde. OUI MAIS : encore une fois. Mais on l'a vu plus haut, on parle de copies de sauvegarde lorsqu'elles sont nécessaires à faire tourner le logiciel. Ne s'agit-il pas de ça ? Probablement. Il y'a un certain vide juridique sur le sujet. Par conséquent, pour être à peu prés sûr d'être dans la légalité, n'ayez pas le CD original ou la cartouche originale en plus de la ROM ou de l'ISO. C'est l'un, ou l'autre !

3.5 Émulateurs

Les émulateurs sont TOUS légaux, sauf s'ils ont étés faits à partir d'informations obtenues de manière frauduleuse (vol de données déposées). La plupart des émulateurs sont faits à partir de ce que l'on appelle du « *reverse engeneering* ». Celui-ci, en France et dans de nombreux pays, est légal, comme l'atteste cet article de loi :

> *Les droits conférés par le brevet ne s'étendent pas :*
> ***a)*** *Aux actes accomplis dans un cadre privé et à des fins non commerciales ;*
> ***b)*** *Aux actes accomplis à titre expérimental qui portent sur l'objet de l'invention brevetée ;*

17. International Organization for Standardization, organisation internationale de normalisation. Désigne des copies de CD-ROM complets. On appelle ces copies téléchargeable, des "images de CD"

En revanche, ce qui est (souvent) illégal, ce sont les BIOS, parfois pourtant indispensables pour faire fonctionner l'émulateur. En effet, les BIOS sont protégés par les lois du "*copyright*" et, en réalité, fonctionnent de la même façon que les ROMs, dans la mesure où il s'agit là aussi d'un micro-programme informatique. Ainsi, pour savoir comment ça marche niveau légal, reportez-vous aux ROMs.

Dans tous les cas, mis à part pour le BIOS, les seuls qui peuvent avoir des problèmes vis-à-vis des émulateurs, sont les auteurs, en aucun cas les possesseurs.

3.6 Conclusion

En conclusion, la loi est parfois dûre, mais c'est la loi. Je ne suis pas là pour vous faire une leçon de morale, juste pour vous mettre au courant, afin que vous agissiez en connaissance de cause. L'émulation, la vraie, n'est que rarement légale, lorsque vous voyez un "*disclaimer*" [18], fuyez en courant, c'est un site dont le webmaster n'y connait rien. Toutefois, des sites comme Consolemul sont dans la totale légalité, mais mis à part quelques émulateurs, on ne trouve pas sur ces sites, grand chose qui intéresse le commun des visiteurs de sites d'émulation.

18. Terme anglophone désignant le « dégagement de responsabilité »

4 L'émulation en pratique

4.1 Explications

Comme nous l'avons énoncé précédemment, il existe trois façons de faire fonctionner un émulateur.

4.1.1 Par interprétation

L'émulation par « *interprétation* » (qui nécessite beaucoup de mémoire), lit le code émulé à partir de la mémoire octet par octet, le décode, et effectue les commandes approppriées sur les registres émulés, la mémoire, et les entrée/sortie (I/O)[19]. L'algorithme général pour un tel émulateur est le suivant:

```
while(CPUIsRunning)
{
  Fetch OpCode
  Interpret OpCode
}
```

Les qualités de ce modèle comprennent sa facilité à debugger, sa portabilité, et sa facilité de synchronisation (vous pouvez en effet facilement compter les cycles d'horloge et rattacher le reste de l'émulation à ce cycle). Son seul point faible - mais non des moindres - s'avère évidemment sa faible performance. L'interprétation occupe beaucoup de temps au CPU et nécessite un ordinateur assez rapide afin de faire tourner votre code à une vitesse décente.

4.1.2 Par recompilation statique

L'émulation par « *recompilation statique* » prend le code émulé et on le transforme en langage compréhensible par le PC. Le résultat est un fichier exécutable qui peut tourner sur votre ordinateur sans aucun outil spécial. Bien que la recompilation statique semble attirante, elle n'est pas toujours possible. Par exemple, vous ne pouvez pas recompiler statiquement un code se modifiant de lui-même car il n'y a pas de méthode pour savoir ce qu'il deviendra sans tourner. Afin d'éviter de telles situations, vous pouvez essayer de combiner un recompilateur statique avec un interpréteur ou un recompilateur dynamique. Cette méthode est tout de même jusqu'à présent utilisée

19. Input/Output, Entrée/Sortie

par la plupart des émulateurs.

4.1.3 Par recompilation dynamique

L'émulation par « *recompilation dynamique* » quant à elle, utilise la même méthode que celle de la recompilation statique, si ce n'est qu'au lieu de transformer le code en une fois, le code est transformé pendant l'exécution du programme. Cette méthode dépend essentiellement de la puissance du CPU. Pour information, on fait tourner '*Tekken 3*' ou '*Gran Turismo*' à peu près correctement sur un '*Pentium II 333*'...

Cependant au cours des dernières années, on assiste à une importante avancée technologique des composants électroniques (graphiques, sonores, ...) que possède notamment les consoles de jeux comme la *Nintendo 64*. On parle alors de HLE[20].

4.1.4 HLE : High Level Emulation

L'architecture de base qui se trouve derrière l'UltraHLE (émulateur de la *Nintendo 64*) diffère beaucoup des autres émulateurs.

Le H.L.E. constitue l'émulation de haut niveau. Ainsi, au lieu d'essayer d'émuler le matériel aussi étroitement que possible et de supporter des opérations de bas niveau, l'approche se révèle être justement le contraire : émulez le moins possible et essayez de détecter des opérations dès que possible, et de les émuler en appliquant le code C optimisé.

Par exemple, il n'y a aucun "BOOT ROM" et le "BOOT CODE" situé dans l'image de la ROM est ignoré. Des routines communes au logiciel d'exploitation comme des ajustements d'interruption sont arrêtées et ignorées. En traitant avec les composants graphiques et sonores, le niveau d'abstraction est élevé. L'émulation de CPU et de DMA[21] est à un bas niveau, mais utilise toujours quelques ruses pour détecter des opérations communes et pour les exécuter plus efficacement.

20. High Level Emulation, émulation de haut niveau
21. Direct Memory Access, accès direct à la mémoire. Méthode de transfert de données dans un ordinateur évitant d'avoir à utiliser le processeur et/ou une zone d'E/S standard

Cela ne s'adapterait probablement pas à une console plus ancienne, où la programmation de bas niveau et en assembleur sont de règle. Mais la *Nintendo64* est tout à fait différente. La majeure partie du code est écrite en C de haut niveau. Ainsi les choses comme les exceptions ou la mémoire virtuelle peuvent seulement être émulées approximativement. Les composants graphiques et sonores utilisant tous deux des listes de commandes, sont raisonnablement normalisés entre les différents jeux.

Aussi, il est à noter l'existence de nombreux "plug-ins"[22] (audios, vidéos, contrôles, ...) qui viennent compléter l'émulateur en question. En principe, les émulateurs comme l'UltraHLE ou encore Project64 disposent d'un répertoire *plugins* destiné aux ajouts d'éléments (encore non pris en charge lors de sa sortie) et/ou à d'éventuelles améliorations. Ce même procédé existe bien entendu dans les émulateurs nouvelles générations comme la PSX, PSEmu Pro, ePSXe, PSXeven, ... qui sont des émulateurs de la *PlayStation*. Voici un bref recensement des "plug-ins" existant concernant la *Nintendo 64*.

Pour la vidéo :

- "Open GL" pour la gestion de l'OpenGL
- "Glide 64" optimisé pour les cartes *3Dfx*
- "Jabo's nVidia OpenGL" optimisé pour les *GeForce* de chez `Nvidia`
- "TR64 Texture Dumping" permettant de dumper les textures de vos ROMs
- ...

Pour l'audio :

- Azimer Audio HLE
- Jabo's DirectSound
- TR64 HLE Audio
- Zilmar's Basic Audio
- ...

Pour les contrôles :

- DarkMan DInput
- Jabo's Direct Input
- N64 Virtual Pad

22. Logiciel greffé à un logiciel principal pour exécuter une tâche externe spécifique

– ...

En outre, on découvre l'existence des "Glides Wrappers" [23] puisque lorsque Ultra HLE a été développé, il était destiné a être utilisé par des cartes graphiques *3Dfx*. Ainsi, Ultra HLE ne pouvait être utilisé par tout le monde. C'est ainsi que les "Glides Wrappers" sont apparus. Ils permettent de traduire les codes *3Dfx* pour les convertir en d'autres langages tels que Direct3D ou OpenGL. Cela permet d'utiliser Ultra HLE avec d'autres cartes graphiques. Cependant, les "Glide Wrappers" ne produisent pas des résultats 100% parfaits, mais certains sont impressionnants et accélèrent terriblement la vitesse d'animation de l'émulation. Je pense que cette rubrique ravira tous les possesseurs de *GeForce* !!!

Pour les "glides wrappers" :

– Boost Wrapper : ce "Glide Wrapper" est très rapide et a une bonne qualité graphique. Il est destiné aux personnes qui ont des *Riva 128*, *Riva TNT*, *Riva TNT2*, et *GeForce*. Ceci est la version normale de ce "glide wrapper", mais ne supporte pas le mode plein écran.
– Clide : ce "Glide Wrapper" est un des meilleurs qui ont été testés. C'est celui qui est le plus rapide et qui affiche le plus d'images par seconde.
– Evoodoo : ce "Glide Wrapper" est tout simplement fabuleux ! Il est compatible avec presque toutes les cartes graphiques, et sa qualité est exceptionnelle. Il marche également très bien avec les *Geforce*.
– MGlide : ce "Glide Wrapper" est principalement utilisé pour les Cartes *Intel i740*. Il fonctionne avec de gros bugs sur d'autres cartes graphiques.
– MUGLide : ce "Glide Wrapper" a été conçu principalement pour des cartes *Matrox* et il a tendance à être relativement lent.
– ...

4.2 Comment émuler un CPU ?

Pour ceux qui désirent écrire leur propre noyau de l'émulation d'un CPU ou qui sont intéressés pour savoir comment celà marche, Marat Fayzullin [24] fournit un squelette d'un émulateur classique de CPU en C. Dans l'émulateur réel, vous pouvez en sauter quelques parties et en ajouter d'autres de votre

23. Protocole utilisé par un grand nombre de jeux pour les graphismes 3D et reconnu par la plupart des cartes vidéos

24. http://fms.komkon.org/EMUL8/HOWTO.html

FIG. 10 – *Tekken 3 sur Bleemcast*

FIG. 11 – *Gran Turismo 2 sur Bleemcast*

plein gré.

4.2.1 Pour débuter

Tout d'abord, nous assignons des valeurs initiales au CPU le compteur de cycle (Counter) et le compteur ordinal (PC).

```
Counter=InterruptPeriod;
PC=InitialPC;
```

Le compteur contient le nombre de cycles du CPU restant jusqu'à la prochaine interruption attendue. Notez que l'interruption ne devrait pas nécessairement avoir lieu lorsque ce compteur expire : vous pouvez l'utiliser pour d'autres sujets, tels que synchroniser les timers, ou mettre à jour les scanlines (ce sont les lignes horizontales des écrans de télévision. Les écrans d'ordinateurs n'en ayant pas, elles peuvent être ajoutées, simulées pour un meilleur confort visuel. En effet, les graphismes des jeux sur consoles sont étudiés en tenant compte de ces lignes) sur l'écran. Le PC contient l'adresse mémoire à partir de laquelle le CPU émulé lira son prochain « op code » [25]. Après avoir initialisé les valeurs initiales, on commence la boucle principale :

25. Code d'opération. En général, c'est le numérro identifiant une fonction dans une bibliothèqque

```
for(;;)
{
```

Notez que cette boucle peut aussi être implémentée de la sorte:

```
while(CPUIsRunning)
{
```

où CPUIsRunning est une variable booléenne. Ceci comporte certains avantages, puisque vous pouvez terminer la boucle à n'importe quel moment par CPUIsRunning=0. Malheureusement, tester cette variable à chaque passe prend beaucoup de temps au CPU et devrait être évité si possible. C'est pourquoi n'implémenter pas cette boucle de la façon suivante:

```
while(1)
{
```

car dans ce cas, certains compilateurs vont générer un code qui teste si la valeur est à 1 ou 0. Vous ne souhaitez certainement pas que le compilateur fasse ce travail inutile à chaque passe de la boucle.

4.2.2 Pour poursuivre

Maintenant que nous sommes dans la boucle, la première chose consiste à lire le prochain « op code » et modifier le PC :

```
OpCode=Memory[PC++];
```

Notez que bien que ce soit la plus simple et la plus rapide des méthodes pour lire à partir de la mémoire émulée, ce n'est pas toujours possible. Une méthode plus universelle d'accès à la mémoire est décrite plus bas dans ce document.

Après le fetch[26] de l'« op code », le compteur de cycle du CPU décroît d'un nombre de cycles nécessaires pour ce « op code » :

```
Counter-=Cycles[OpCode];
```

Le tableau Cycles[] devrait contenir le nombre de cycles du CPU pour chaque « op code ». Il faut cependant faire attention que certains « op codes » (comme les sauts conditionnels ou les appels de sous-routines) prennent un

26. Terme technique signifiant rechercher

nombre différent de cycles suivant leur argument, ce qui peut toutefois être ajusté dans le code après.

Il est maintenant temps d'interpréter le « op code » et de l'exécuter:

```
switch(OpCode)
{
```

C'est une fausse idée courante de penser que la construction switch() est inefficace, pour la raison qu'elle est compilée en une chaîne de si() ... sinon () ... Bien que ce soit vrai que pour des constructions avec un petit nombre de cas, les constructions importantes (100-200 ou plus de cas), quant à elles, apparaissent toujours compilées en un tableau de saut, ce qui les rendent assez efficaces.

Il existe deux possibilités pour interpréter les « op codes ». La première consiste à construire un tableau de fonctions et à appeler la bonne. Cette méthode se révèle moins efficace qu'un switch() car vous payez les appels de fonction. La seconde méthode serait de créer un tableau d'étiquettes et d'utiliser la déclaration « goto ». Bien que cette méthode soit légèrement plus rapide qu'un switch(), elle ne marchera qu'avec des compilateurs qui supportent des "étiquettes précalculées". Les autres compilateurs ne vous permettent pas de créer un tableau d'adresses étiquetées.

Après avoir interprété et exécuté un « op code » avec succès, il est alors temps de vérifier si l'on a besoin d'une quelconque interruption. À ce moment, vous pouvez également remplir toutes les tâches qui ont besoin d'être synchronisées avec l'horloge système:

```
if(Counter<=0)
{
  /* Contrôle des interruptions et émulation hardware */
  ...
  Counter+=InterruptPeriod;
  if(ExitRequired) break;
}
```

Ces tâches cycliques sont traitées plus bas dans ce document. Notez que nous n'assignons pas simplement Counter=InterruptPeriod mais plutôt ainsi Counter+=InterruptPeriod : ceci afin de rendre le compte du cycle plus précis. En effet, il peut y avoir des nombres négatifs de cycle dans le compteur.

Jetez également un oeil sur :

```
if(ExitRequired) break;
```

Comme il est trop coûteux de vérifier une éventuelle sortie à chaque passe de la boucle, on ne le fait que lorsque le compteur expire : ceci permettra encore la sortie de l'émulation si vous fixez ExitRequired=1, mais nécessite moins de temps CPU.

4.3 Accès à la mémoire

La méthode la plus simple pour accéder à la mémoire émulée est de la gérer comme un tableau ordinaire d'octets (mots, etc). L'accès en est alors trivial :

```
Data=Memory[Address1]; /* lecture à partir d'Address1 */
Memory[Address2]=Data; /* écriture à Address2  */
```

4.3.1 Les différents types de mémoire

De tels accès mémoires ne sont pourtant pas toujours possibles pour les raisons suivantes:

- *Mémoire paginée* : l'espace d'adressage peut être fragmenté en pages échangeables (c'est-à-dire des blancs). C'est souvent le cas pour étendre la mémoire lorsque l'espace d'adressage est petit (64kB).
- *Mémoire réfléchie* : un endroit de la mémoire peut être accessible par plusieurs adresses différentes. Par exemple, les données que vous écrivez à l'adresse $4000 peuvent également apparaître à $6000 et $8000. Les ROMs peuvent aussi être réfléchies à cause d'un décodage d'adresses incomplètes.
- Protection des *ROMs* : certaines cartouches de jeux (comme les jeux MSX) essaient d'écrire dans leur propre ROM et refusent de marcher si l'écriture réussit. C'est souvent le cas de protection contre la copie. Pour faire marcher ces software sur votre émulateur, vous devez enlever la possibilité d'écrire dans la ROM.
- *Mémoire mappée* d'I/O : certains systèmes peuvent recourir à des I/O à mémoire mappée. Les accès vers de tels emplacements mémoires produisent des "effets spéciaux" et par conséquent doivent être recensés. Afin de faire face à ces problèmes, on introduit un couple de fonctions:

```
Data=ReadMemory(Address1);  /* lecture à partir d'Address1 */
```

```
WriteMemory(Address2,Data); /* écriture à l'Address2  */
```

Tous les processus spéciaux d'accès à la mémoire décrits ci-dessus sont réalisés à l'intérieur des fonctions *ReadMemory()* et *WriteMemory()* habituellement placées bien en avant dans l'émulation car elles sont appelées très souvent. Elles doivent ainsi être aussi efficaces que possibles. Voici un exemple pour ces fonctions écrites pour accéder à un espace d'adressage paginée :

```
static inline byte ReadMemory(register word Address)
{
  return(MemoryPage[Address>>13][Address&0x1FFF]);
}

static inline void WriteMemory(register word Address,
register byte Value)
{
  MemoryPage[Address>>13][Address&0x1FFF]=Value;
}
```

Remarquez le mot clé "inline". Il demande au compilateur d'inclure la fonction dans le code au lieu de faire des appels à elle. Si votre compilateur ne supporte pas "inline" ou "_inline", essayez la fonction "static", certains compilateurs (Watcom C par exemple) optimiseront de courtes fonctions statiques en les introduisant dans le code.

Gardez également bien à l'esprit, que, dans la plupart des cas, *ReadMemory()* est appelé plus fréquemment que *WriteMemory()*.

– Petite remarque en ce qui concerne la mémoire réfléchie : Comme il est préalablement précisé, beaucoup d'ordinateurs disposent de RAM[27] réfléchie dans laquelle une valeur peut être écrite à plusieurs emplacements mémoire. Bien que cette situation peut être réglée par la fonction *ReadMemory()*, ce n'est pas souhaitable, puisque *ReadMemory()* peut être appelé beaucoup plus fréquemment que *WriteMemory()*. Une méthode plus efficace consiste à implèmenter la mémoire réfléchie lors de l'écriture dans la fonction *WriteMemory()*.

27. Random Access Memory, mémoire vive

4.4 Tâches cycliques

4.4.1 Quelles sont-elles ?

Les tâches cycliques sont des processus qui devraient avoir lieu périodiquement dans une machine émulée, comme le rafraîchissement de l'écran, les interruptions « VBlank » et « HBlank », la mise à jour des timers, des paramètres sonores, de l'état du clavier, des joysticks...

Afin d'émuler de telles opérations, vous devez les relier à un nombre de cycles appropriés du CPU. Par exemple, si le CPU est supposé être cadencé à 2.5 MHz et l'affichage utilise un taux de rafraîchissement de 50 Hz (standard pour le PAL), l'interruption « VBlank » doit avoir lieu tous les $\frac{2500000}{50} = 50000$ cycles du CPU.

Maintenant, si vous optez pour l'écran de 256 scanlines de hauteur dont 212 actuellemnt affichés (soit 44 dans « VBlank »), vous obtiendrez une émulation qui rafraîchira un scanline sur $\frac{50000}{256} = 195$ cycles du CPU.

Après quoi, vous devriez générer une interruption « VBlank » puis ne rien faire jusqu'à ce que vous en ayez fini avec le « VBlank », c'est-à-dire pour $\frac{(256-212)*50000}{256} = 44 * 50000 = 8594$ cycles du CPU.

Calculez avec attention les nombres de cycle du CPU nécessaires à chaque tâche, puis prenez le plus grand commun diviseur pour l'« InterruptPeriod » et reliez-la à toutes les autres tâches (une exécution ne devrait pas être nécessaire à chaque fin de compteur).

5 L'émulation de composants

Pour commencer, prenons un exemple concret en ce qui concerne l'émulation des consoles.

5.1 Composants électroniques

Pour émuler les consoles, prenons par exemple une *PlayStation*, il faut connaître tout les composants électroniques qui composent cette console, ensuite il va falloir émuler composant par composant. Nous réalisons donc de l'émulation électronique ou encore de l'émulation de composants.

Attention à ne pas confondre avec la programmation des composants électroniques ou encore la simulation de circuits électronique qui est réalisée grâce à des langages comme VHDL[28].

Nous constatons alors que l'émulation d'une carte électronique est une succession d'émulations de composants et de circuits électronique qui composent cette carte. Ainsi de suite nous réalisons notre émulation circuit après circuit, carte après carte, jusqu'à obtenir l'émulation complète de la machine.

5.1.1 Prérequis

Pour réaliser l'émulation des composants, il est nécessaire d'avoir des documentations techniques correctes de ces composants. Les personnes réalisant l'émulation sont assez libre échange de code source (open-source) et n'hésitent pas à s'échanger sur certains sites électroniques différentes astuces de programmation sur différentes zones (circuits, composants) d'une machine.

Ainsi un composant déjà émulé dans un système ne sera peut être pas réécrit, car deux machines différentes peuvent avoir en commun des composants et cela peut réduire grandement le temps de travail sur l'émulation d'une nouvelle machine pour un programmeur.

28. Very High Development Language

5.1.2 Émulation du CPU

Des sites comme http://www.zophar.net/index.phtml propose des documentations de composants tels que le processeur *68000* de Motorola. Ce type de processeur est utilisé entre autre dans la console *Megadrive* également dans l'ancien *Atari ST*, et son émulation est dèjà réalisée et diffusée sur le net. Aussi on comprendra que quelqu'un qui veut réaliser un (nième) émulateur de cette console gagnerait du temps en évitant de réaliser un émulateur de micro-processeur *68000* de plus, puisque n'étant pas son objectif.

Aussi il pourrait passer plus de temps sur les autres composants tels graphiques ou sonores de la console.

5.1.3 Émulation de composants

Par exemple, l'*Atari ST* avait comme composant sonore le '*Yamaha 2194*' donc même système, même punition, il faut aller chercher la documentation du '*Yamaha 2194*' puis l'émuler lui tout seul avant de le combiner avec le processeur *68000* de Motorola. Il en va de même pour le composant graphique ainsi que pour tous les autres types de composants, à savoir :

- connaître le processeur utilisé
- chercher la documentation à son sujet
- émuler ce composant à part
- l'intégrer au processeur principal

Autre exemple, les capacités sonores du *Commodore 64* ne sont plus à présenter, celui-ci était certainement, pour son époque, l'ordinateur le plus capable musicalement. Des compositeurs célèbres comme Martin Galway ou Rob Hubbard ont réalisé des musiques magnifiques.

C'est sans doute pour cette raison que l'on retrouve actuellement des émulateurs sonores. Ceux-ci n'émulent que la puce sonore du *Commodore 64* : le SID [29] et éventuellement quelques autres composants pour reproduire fidèlement les musiques de l'époque. Ces émulateurs sont alimentés avec des fichiers sonores (extension .SID ou .RAW) que l'on retrouve un peu partout sur internet.

29. Sound Interface Device

FIG. 12 – *Logo MAME*

6 L'émulateur MAME

6.1 Les origines de MAME?

MAME a vu le jour le 24 décembre 1996. Ce fut Nicolas Salmoria qui a commencé à travailler dessus. Au départ, il ne devait émuler que le multi-Pac puis ce programme a pris une ampleur énorme et est devenu le numéro 1 mondial de l'émulation multi-arcade dominant et de loin les autres émulateurs. Aujourd'hui MAME (de son vrai nom anglais MAIM) supporte 3731 sets de roms pour 2131 jeux uniques et il est arrivé en version 0.70. À titre informatif, sachez que MAME est toujours aujourd'hui en version béta et continu sa montée fulgurante tous les jours.

6.2 Qu'est-ce que MAME?

M.A.M.E. signifie: "Multiple Arcade Machine Emulator". Utilisé correctement avec les fichiers datas de jeux (ROMs), MAME tentera de reproduire le plus fidèlement possible ce jeu sur un ordinateur (PC, Mac, ...). MAME "émule" aujourd'hui plus de 1500 jeux de bornes d'arcades des 70's aux 80's.

Les ROMs dont MAME a besoin sont directement "dumpées" (extraites) à partir des chipsets du circuit imprimé original de la borne d'arcade, MAME devient ainsi le "hardware" qui supporte la ROM, prenant la place des CPUs et des composants originaux de la borne d'arcade. Donc, ces jeux NE SONT PAS des simulations ou encore des adaptations, mais BEL ET BIEN les véritables jeux originaux qui apparurent dans les salles d'arcades et dans les

cafés...

L'objectif de MAME est de préserver les premières décennies de l'histoire du jeu vidéo. Alors que la technologie actuelle des jeux vidéos continue à avancer de plus en plus vite, MAME permet à tous ces jeux dits de la "bonne vieille époque" de ne pas tomber à jamais dans l'oubli et le néant ...

6.3 Quels matériels pour MAME?

Beaucoup d'ordinateurs, mais pas tous, feront tourner MAME sans problèmes. Ces performances dépendront fortement du CPU de votre PC et de sa carte graphique. Il est à noter également que les performances de certains jeux peuvent varier, selon la difficulté à émuler certains hardware plus puissants que d'autres.

Voici un PC de base qui pourra jouer (tout en assurant une émulation correcte) à peu près la moitié des jeux supportés par MAME (Neo-Geo exclus):

- Pentium 200MHz
- 16-32MB RAM
- une taille variable d'espace disque (la collection complète des ROMs MAME devrait prendre à peu près 5 Go, bien que beaucoup de ces ROMs sont très petites)
- Une carte graphique compatible VESA 2.0+
- Une carte-son Sound Blaster (ou compatible)

PERFORMANCES GRAPHIQUES: Ces jeux n'utilisant pas d'effets 3D ultra modernes, une carte accélératrice 3D ne vous sera donc d'AUCUNE utilité! Votre meilleur choix restera une bonne carte 2D supportant le VESA 2.0 et 2MB ou plus de mémoires vidéos (telles que les Matrox G200 or G400).

PERFORMANCES SONORES: Pour MAME32, la Sound Blaster SB-AWE32 reste la meilleure. Pour MAME DOS, les cartes SB64PCI, SB128PCI, et Ensoniq Soundscape PCI marcheront tout aussi bien, en utilisant l'option soundcard 7. (Attention!! La Sound Blaster Live NE MARCHERA PAS avec MAME DOS).

FIG. 13 – *Borne d'arcade (avec logo MAME)*

De maniére général, plus vous aurez de meilleurs composants dans votre PC, plus vous aurez de meilleurs résultats. Un "pauvre" PII/233 avec une bonne carte 2D sera capable de tourner 75% des jeux. Pour certains jeux cependant, même un PIII/500 ne sera pas assez rapide!!

Pour les `Macs`, toutes machines de base *`G3`* devrait faire tourner Mac-MAME correctement.

Rappelez-vous: Le projet MAME n'a pas pour but de rendre tous les jeux jouables, mais de les reproduire le plus fidélement. Acceptez le fait que certains jeux ne tourneront pas parfaitement sur votre systeme, vous vous épargnerez ainsi bien des décéptions. D'ailleurs, rappelez-vous également que MAME est GRATUIT !!.

6.4 Versions de MAME?

Il existe en effet plusieurs "portages" de MAME sur bien d'autres plate-formes, dont le Mac, Linux, et l'Amiga. (Se référrer à la page officielle de MAME pour une liste compléte). Il existerait même une version sur Nin-

tendo 64! - même si elle n'est pas facile à trouver (et totalement illégale, puisque les fichiers ROMS sont distribués sur le même support que MAME).

Les versions PC et Mac sortent la plupart du temps simultanément - les autres prennent souvent plus de temps à voir le jour. Si vous êtes sur l'une de ces autres plates-formes, attendez vous à un bon long délai entre ces sorties et la sortie DOS.

6.5 MAME est-il illégal?

Non. Émuler une autre plate-forme, en soi-même, n'est pas illégal. Il n'est donc pas illégal d'avoir MAME installé sur son ordinateur, sur sa page web, ou de le distribuer à des amis.

Pour les images des ROMs, c'est un autre probléme. Beaucoup de sites proposant des ROMs ont été poliment contactés par les propriétaires des « copyrights », et se sont vus demander de mettre ces images "offline". Pourtant au moment où j'écris cette phrase, aucun site n'a été LÉGALEMENT férmé, ou poursuivi en justice.

L'opinion de Nintendo à propos de la légalité des ROMs est clair : Ils pensent que faire des images des ROMs est illégal en toutes circonstances. (Bien sûr, cela voudrait dire que CAPCOM commet un crime en vendant leurs propres ROMs avec l'HotRod Joystick de Hanaho).

Posséder des images ROMs de circuits imprimés de bornes d'arcades que vous ne possédez pas peut constituer une violation du « copyright », tout comme copier le jeu d'un ami ou un CD Audio plutôt que de l'acheter. Mais celà reste dans une zone légale vague; posséder de telles ROMs ne mènera pas nécessairement à des poursuites judiciaires. De toutes manières, vous êtes responsable de vos propres actions. Personne n'ayant attrait au projet MAME ne pourrait être tenu pour responsable si vous avez des ennuis.

6.6 Contribuer au projet MAME?

Vous pouvez déjà commencer par tester les "drivers"[30], anciens comme nouveaux. Ainsi l'équipe MAME a un peu de "feedback"[31], et peut, si nécessaire, altérer ou améliorer certaines choses comme les sons et les graphiques. Vous trouverez plus d'informations à ce sujet sur le site http://www.mametesters.com.

30. Drivers (pilotes) - petit programme permettant de relier logiciellement l'ordinateur à un périphérique donné

31. Rétroaction, effet de retour. En informatique, il s'agit de retour d'information

7 Bornes d'arcade à domicile

Le chapitre précédent nous amène à parler des "hardcore nostaliques" [32].

7.1 "Hardcore nostalgiques"

7.1.1 De quoi s'agit-il?

Il s'agit tout simplement de personnes désireuses de réaliser en partie leur rêve de "gosse", et de restituer le décor et l'ambiance des jeux de café en y installant ou fabriquant leurs propres bornes d'arcade à la maison.

7.1.2 Les méthodes qu'ils utilisent?

Première étape : récupérer LA borne d'arcade bien évidemment. Pour cela, on vous conseille soit de rechercher sur internet notamment à l'adresse suivante : http://www.omnicade.com soit par la bonne vieille méthode qui consiste à feuilleter les pages jaunes. Cependant, celà pourrait vous coûter 150 €. Aussi, l'idéal serait d'en récupérer une complète (c'est-à-dire avec l'écran original). Bien sûr, mais vous allez certainement me demander pourquoi?

Et bien celà s'explique par le fait que les écrans de bornes d'arcade disposent d'un balayage horizontal de 15KHz et d'une résolution allant jusqu'à 640x288. Rigolo oui... Oui mais voilà ; ces moniteurs filtrent naturellement le signal et produisent un « anti-aliasing » [33] naturel inégalable sur une télé ou sur un moniteur de PC classique. La résolution native des jeux est aux alentours de 320x200 de toutes manières. Donc, pour avoir le meilleur rendu possible, il faut pouvoir utiliser le moniteur de la borne et obtenir un ration 1:1 entre la résolution interne des jeux et celle qui est rendue sur l'écran. Aussi, vous pouvez vous rendre à l'adresse suivante (http://www.ultimarc.com/monfaq.html) pour plus d'informations à ce sujet.

32. Incontestablement les plus "accros" des anciens jeux vidéos

33. Ou « Antialiasing ». Technique par laquelle on diminue l'effet d'escalier des images, en créant des dégradés de couleurs le long des contours, pour les lisser. L'anti-aliasing est utilisé par exemple en imagerie de synthèse, et coûte beaucoup de temps de calcul, car on calcule plusieurs fois chaque point avec de petites variations des paramètres

FIG. 14 – *Carte J-Pac (à gauche), avec les câbles reliés au PC*

Deuxième étape : Il s'agit de la partie technique dans l'acquisition de sa borne d'arcade. Elle reste cependant réalisable grâce aux explications que l'on peut trouver sur internet. En effet, il faut savoir que toutes les bornes assez récentes (<25 ans) suivent le standard *JAMMA*. On trouve dans la borne un connecteur plat relié au moniteur, aux manettes et boutons ainsi qu'aux monneyeurs. Il suffit de brancher un PC sur ce connecteur (clavier + VGA) et le tour est joué. On peut acheter chez http://www.ultimarc.com, une société anglaise pour 57 € un petit boitier (le J-PAC) qui prend d'un côté le VGA du PC et le clavier et de l'autre le connecteur JAMMA. Le J-PAC fait en outre office d'amplificateur vidéo pour passer le signal de 1 volt à 5 volts (la plupart des moniteurs attendent du 5v, le PC sort du 1v, mais certains moniteurs d'arcade récents acceptent les deux).

Également chez http://www.ultimarc.com, nous pouvons acheter une *ATI*

FIG. 15 – *Résultat obtenu à l'aide d'un PC et du logiciel AdvanceMAME*

Radeon spéciale (nommée "ArcadeVGA") qui gère toutes les basses résolutions et les affichages en 15Khz. Avec cette carte, on peut afficher windows et faire tourner les émulateurs PSX, SNES, sur le moniteur arcade, chose impossible autrement. La carte coûte tout de même 90 €.

Ainsi, si nous établissons un récapitulatif, nous pouvons obtenir le tout pour la coquette somme de 300 € environ ; mais quand on aime, on ne compte pas me diriez-vous et l'idée de réaliser un des rêves de gamins pour 300 € ne semble pas insurmontable.

8 Conclusion

On voit donc qu'un simple logiciel peut créer de véritables questions de fond, et nul doute que le débat est loin d'être clos. Car les points d'achoppement sont nombreux et les émulateurs multiplient les problématiques juridiques.

Ce qui reste sûr c'est que les émulateurs ont de beaux jours devant eux car ils permettent d'éviter de payer le matériel et facilitent grandement - avec la diffusion par internet - le piratage des jeux.

Toutefois, ces dérives ne sauraient occulter leur rôle primordial dans la sauvegarde de la "cyber culture", et un équilibre sera à trouver à l'avenir entre intérêts économiques et sauvegarde du patrimoine.

En outre, on a montré que l'émulation n'était pas uniquement destinée aux "gamers"[34] nostalgiques d'anciens jeux, mais pouvait aspirer - dans un avenir proche - à un rôle plus sérieux dans le domaine professionel.

Quoi qu'il en soit, l'émulation est de nos jours, associée à du piratage informatique, par le fait qu'elle favorise l'échange de jeux commerciaux entre utilisateurs, mais ceci est une autre histoire...

34. Terme anglophone désignant les joueurs (de jeux vidéos)

9 Remerciements

Nos remerciements s'adresseront d'une part aux adresses internet (citées ou non) pour leurs précieux renseignements. Bien entendu il a fallu filtrer leur contenu pour en tirer le meilleur.

D'autre part et pour finir, nous tenons à remercier tout particulièrement M. Michel BUFFA qui nous a encadrés et contribué à la réalisation de notre TE. Il nous a aussi indiqué la marche à suivre en nous proposant des adresses de sites qui ont constitué notre point de départ ainsi que notre plan.

Index