Techniques d'optimisation des temps de chargements dans les jeux à monde ouvert
Ou comment réduire les temps de chargement

CHRISTIAN NGO & JONATHAN MULLER
ESGI - 5ème année IJV
12 janvier 2014

Résumé

Dans ce mémoire nous verrons comment réduire les temps de chargement des jeux vidéos en se basant à la fois sur les composants de la plate-forme cible, sur les techniques de compression, sur l'algorithme et sur le level design.

Mots clefs : chargement , optimisation , ressources , streaming , jeu

1 Introduction

Les jeux vidéos comportent année après année un nombre grandissant de ressources nécessaires pour leur fonctionnement. Des textures aux modèles 3D, le poids de ces ressources augmente tout comme leur nombre, l'augmentation de la taille mémoire des machines permettant aux créateurs de jeux de charger plus de ressources à la fois. Cependant, augmenter la taille et le nombre des ressources a un prix, celui du temps de chargement. Dans ce mémoire nous verrons comment réduire ces temps de chargement en s'adaptant aux médias qui permettent de stocker les ressources du jeu, en tirant partie des capacités des processeurs, de l'espace disponible et de la mémoire RAM, des différents formats de compression, mais aussi des adaptations de gameplay qui peuvent permettre un chargement transparent pour l'utilisateur.

Table des matières

1.1 Pourquoi avons nous besoin d'améliorer les temps de chargement

La quantité de mémoire disponible augmente régulièrement sur les postes des utilisateurs, ce qui permet aux concepteurs de jeux de proposer des jeux de plus en plus riches et chargés en éléments graphiques, ce qui demande beaucoup de ressources à créer, stocker, et donc charger. En améliorant les temps de chargement des programmes, nous pouvons faire passer cette quantité de données en mémoire plus rapidement, et donc avoir des jeux qui permettent de jouer rapidement, sans passer de longs instants devant un écran de chargement à plusieurs reprises.

1.2 Comment influencer les temps de chargement

Les temps de chargement peuvent être influencés de différentes manières.

Nous pouvons améliorer les temps de chargement en utilisant des techniques de stockage qui varient en fonction du média sur lequel le jeu est enregistré. Par exemple, les temps d'accès étant plutôt long sur un cd-rom, il faut veiller à limiter le nombre d'accès aux ressources et penser à charger des éléments qui ne seront pas forcément affichés tout de suite, lors des premiers chargements si la mémoire le permet.

Au contraire sur un disque SSD, les temps d'accès étant très bas et la capacité de transfert élevée, on pourra séparer les ressources dans des fichiers différents afin de les charger quasiment à la volée.

Les temps de chargement peuvent également être réduits en utilisant le processeur pour décompresser des ressources : ainsi le temps nécessaire au transfert de la donnée depuis le disque à la mémoire est réduit puisque la donnée est compressée sur le dit disque. Il faut cependant que le mode de compression soit adapté à la puissance du processeur et à la vitesse de transmission du support de stockage.

En modifiant l'algorithme du jeu, nous pouvons aussi influer sur le chargement des données, en évitant par exemple de stocker plusieurs fois des ressources très similaires. Si nous prennons l'exemple des sprites 2D, nous pouvons stocker l'image de base une fois, puis effectuer toutes les déclinaisons de couleurs de manière algorithmique en modifiant la palette de couleurs. De cette façon, nous n'avons qu'à stocker une texture, puis uniquement des palettes de couleurs.

Enfin, une méthode pour cacher les chargements aux joueurs consiste à construire son level design pour permettre certaines zones "tampons" comme un pont ou un couloir dont le passage est obligatoire pour passer d'une zone à l'autre, et qui va permettre de charger/décharger les zones concernées en tout transparence pour l'utilisateur. Cependant, cette technique est déconseillée puisqu'elle limite la liberté des level-designers en leur imposant des contraintes "techniques".

2 Optimisations liées au matériel

2.1 Benchmark des différents supports de stockage

Pour mieux comprendre comment organiser nos ressources et la façon de les charger en fonction du support de stockage destiné au jeu, nous devons d'abord étudier et comprendre les deux principaux facteurs des temps de chargement liés au matériel : la vitesse de transfert et le temps d'accès. Le temps d'accès représente le temps écoulé entre le moment où la demande d'accès aux données va être effectuée et le début du transfert de cette donnée. Par exemple sur un disque dur, le temps que celui ci va prendre pour positionner sa tête de lecture, faire tourner le disque à la bonne position et commencer à lire la donnée. Le taux de transfert est le nombre de données qu'il est capable de faire transiter du support de stockage à la mémoire, par seconde.

2.1.1 Les CD-ROM

Les CD-ROM ne sont plus beaucoup utilisés pour les jeux vidéos, qui dépassent maintenant souvent la capacité de stockage de ceux ci (entre 600 et 900Mo). Ils sont remplacés par des supports comme les DVD-ROM ou les Bluray. Cependant, il est intéressant d'étudier ce support de stockage pour une meilleur compréhension de l'évolution des périphériques.

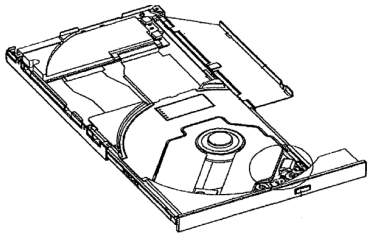

FIGURE 1 – Lecteur CD-ROM

Le CD-ROM a beaucoup été utilisé dans les années 90 et 2000, car le coût de fabrication de ce support est faible et ses capacité de stockage suffisantes, quitte à fournir régulièrement 4 CD-ROM dans les boites de jeu.

La vitesse de lecture de ce support est, par contre assez faible et rapidement les jeux ont utilisé ce média pour simplement installer les données sur le disque dur et servir de DRM.

Vitesse de transfert des lecteurs CD-ROM : [1]

Vitesse du lecteur	Taux de transfert (BPS)	Latence (ms)
Single-speed (1x)	153,600	400
Double-speed (2x)	307,200	300
Triple-speed (3x)	460,800	200
Quad-speed (4x)	614,400	150
Six-speed (6x)	921,600	150
Eight-speed (8x)	1,228,800	100
Ten speed (10x)	1,536,000	100
Twelve speed (12x)	1,843,200	100
Sixteen speed (16x)	2,457,600	90
Eighteen speed (18x)	2,764,800	90
Twenty four speed (24x)	3,686,400	90
Thirty two speed (32x)	4,915,200	85
One hundred speed (100x)	15,360,000	80
CAV drives (12x - 24x)	1,843,200 - 3,686,400	150-90

Comme nous pouvons le constater, les temps d'accès du lecteur CD-ROM sont assez élevé. Si nous devons charger 50 petits fichiers sur un lecteur 16x, nous ajoutons inutilement 4 secondes et demi de temps de chargement. Il convient donc d'encapsuler ces 50 fichiers dans un seul, de le charger d'une traite puis de faire la séparation des fichiers en mémoire. C'est pour cette raison que sur les CD-ROMs de jeux nous trouvons souvent que peu de fichiers, mais qui prennent une grande partie de l'espace.

2.1.2 Lecteur DVD-ROM

Le DVD-ROM a une vitesse 9 fois supérieure à celle d'un CD-ROM,[2] pour une capacité allant de 4.7Go à 17.08Go (ce dernier étant très rare). Le temps d'accès aux données sur un DVD-ROM est d'environ 100ms pour les plus rapides, ce temps variant en fonction des lecteurs. Nous retrouvons donc avec les DVD-ROM le même problème qu'avec les CD-ROM : le temps d'accès au données élevé. Le taux de transfert et la quantité de données qu'il est possible de stocker sur celui ci en fait cependant un successeur de choix au CD-ROM, améliorant au passage les temps de chargement grâce à sa vitesse de lecture améliorée.

2.1.3 Lecteur BD-ROM

Le BD-ROM est capable de transférer les données de 4.5Mo par seconde (x1) à 72Mo par seconde (x16)[3] et peut stocker entre 25Go et 50Go de données. Cela représente 50 fois la vitesse d'un CD-ROM. Bien que ses temps d'accès soient similaires voir plus grands que pour les DVD-ROMs, sa vitesse de transfert lui permet cependant de proposer des temps de chargement plus rapides, les données étant stockées sous forme de flux dans l'ordre d'utilisation comme pour les autres supports sous forme de disque, sa capacité de stockage permet en plus d'alléger le travail du processeur en proposant des données moins compressées.

2.1.4 Le Disque Dur

Le disque dur est le support de stockage le plus utilisé sur PC. Présent depuis les années 60 dans les ordinateurs de bureau, il a depuis continué à s'améliorer et à augmenter ses performances de lecture/écriture et ses temps d'accès. Le premier disque dur avait un temps d'accès de 600ms. Dès les années 80, ce temps a été réduit à 20ms et a continué de diminuer depuis.

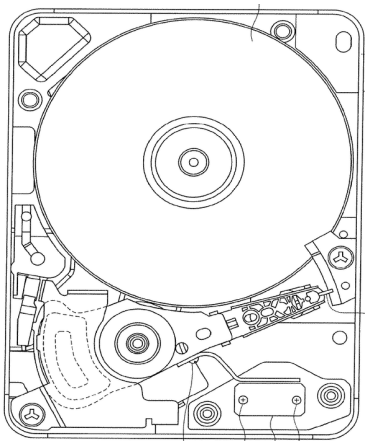

FIGURE 2 – Disque dur

Il existe 5 vitesses pour les disques durs, qui sont le nombre de rotation effectuées par le disque, par minute. Les vitesses de disque les plus répandues sont celles à 5400 tours/minute et 7200 tours/minute (respectivement 5.55ms et 4.16ms de temps d'accès moyen).[4]

Les taux de transfert des disques à 7200 tours par minute en 2010 sont en moyenne de 125Mo par seconde. Comparé aux supports de stockage au format optique, le disque dur les surpasse tous. Cependant des problèmes peuvent intervenir sur les disques durs comme la corruption de données, ou la perte de vitesse à cause de la fragmentation qui peut intervenir sur certains systèmes de fichiers. Séparer les fichiers sur ce type de disque pour ne pas avoir de chargement sous forme de flux peut donc s'avérer pénalisant sur un disque fortement fragmenté.

2.1.5 Le SSD

Le SSD est le support de stockage qui remplace peu à peu les disques durs. Ses taux d'accès étant très faible (en dessous de 0.1ms) et sa capacité de transfert élevé (entre 100 et 550Mo par seconde)[5], il permet d'avoir peu voire pas de chargement visible pour l'utilisateur dans les jeux vidéos.

Les caractéristiques du SSD sont proches de celles des cassettes de jeu vidéo utilisées sur les machines avant l'ère du CD. Ce type de support permet d'avoir des jeux où les chargements sont transparents pour l'utilisateur. De plus, la fragmentation des fichiers n'a aucun effet sur un disque SSD, les données peuvent donc être séparées pour minimiser le chargement de ressources inutiles et permettre un chargement à la volée rapide.

2.2 Multiplier les supports

Si nous avons vu la supériorité des disques durs et SSD par rapport aux supports au format disque, nous pouvons quand même tirer partie du lecteur de disque pour améliorer encore les performances. Ainsi, nous pouvons nous servir du DVD/BD pour installer le jeu sur le disque dur, et garder une partie du disque qui servira lorsque le jeu sera lancé. Cela permettra d'accéder aux données à la fois du disque dur et sur le DVD/BD dans le jeu. C'est cette méthode qui est employée actuellement sur console, profitant du fait qu'elles soient équipées en disque dur pour réduire les temps de chargement.

3 Techniques d'optimisation

3.1 Généralités

Il existe 3 façons majeures de charger les données nécessaires à un jeu en mémoire.

La première est de charger les données au démarrage du niveau ou du jeu et de les garder en mémoire tout le long du jeu. C'est la meilleure solution pour les ressources qui seront nécessaires à toutes les étapes du jeu, ou pour les ressources critiques du jeu sans lesquelles le jeu ne pourra pas tourner (cela permet de faire une vérification par la même occasion). Puisque ces données seront chargées quoi qu'il arrive, nous pouvons optimiser le temps de chargement en les chargeant dès le menu ou dès l'apparition des sponsors au début du jeu.

La deuxième façon est de charger les données nécessaires en fonction de la position du joueur et de l'orientation de la caméra. Nous pouvons charger les données aux alentours du joueur et des données plus détaillées en fonction de la direction dans laquelle le joueur regarde. Ainsi, ce sont autant de données en moins à charger au lancement du niveau ou du jeu, et cela permet d'éviter des écrans de chargement qui détériorent l'expérience de jeu de l'utilisateur.

La troisième façon est basée sur les événements : par exemple commencer à charger une zone lorsque le joueur sélectionne un objet qui lui permet de s'y téléporter, charger une séquence d'animation quand le joueur parle à un personnage qui va la déclencher, etc.

3.2 Stockage des données

Pour améliorer les temps de chargement, les données sont organisées sur les disques en fonction de leurs secteurs et généralement dans un seul fichier d'archive (afin de minimiser les appels aux fonctions open et close du système d'exploitation, qui sont relativement lentes).

Ainsi notre fichier contenant les ressources reste toujours ouvert, et il ne nous reste plus qu'à lire les parties du fichier qui nous intéressent. Il convient donc d'optimiser la partie lecture pour que notre jeu charge le plus vite possible. Pour cela, il faut faire attention à la manière dont sont stockées les données, en l'occurrence leur position vis à vis des secteurs.

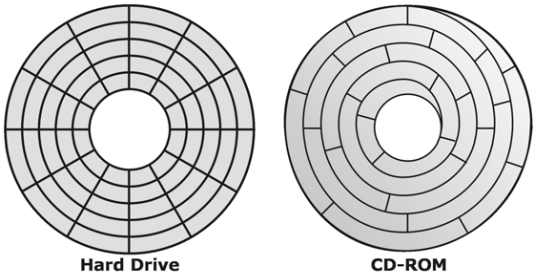

FIGURE 3 – Secteurs d'un disque dur et d'un CD-ROM

Un secteur est un fragment d'un disque qui va contenir les données stockées, un disque est composé de plusieurs secteurs mis les uns à la suite des autres. Lors d'une opération de lecture, la tête de lecture du disque ou le laser pour un disque optique va se positionner en face du secteur, et commencer à lire les données. Le temps de positionnement est le plus long, mais à chaque fois que la tête se déplacer pour parcourir un secteur, nous perdons du temps. Une technique pour réduire cette perte est d'aligner les fichiers sur les secteurs (ce qui entraine une perte d'espace, si le média à un espace limité, il ne sera peut être pas possible de réaliser cette opération) [6].

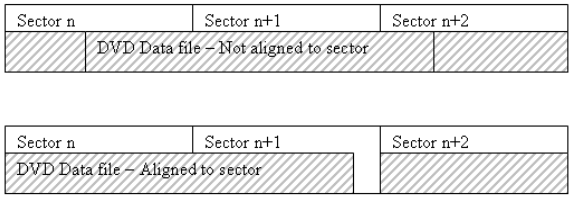

FIGURE 4 – Alignement des données sur les secteurs

3.3 Organisation et duplication des fichiers

La technique la plus utilisée dans les jeux à monde ouverts disponibles sur disque optique est la technique du streaming, ou chargement par flux. Les données sont organisées sur le disque de manière à être chargées les unes à la suite des autres pour que le lecteur de disque puisse lire les données sans effectuer un nouveau tour pour se repositionner.

Si nous prenons l'exemple de Just Cause, les données du terrain sont séparées sur le disque sous forme de patch, contenant la heightmap du terrain, les textures et les données du mesh. Compte tenu du fait qu'ils avaient encore de l'espace disponible sur le disque optique, ils ont stocké les données du terrain deux fois : une fois ordonné en x vers z et une fois en z vers x. Ainsi les données peuvent être chargées sous forme de flux en fonction de la direction du joueur [7].

Ainsi nous remarquons qu'une technique d'optimisation de chargement peut passer par la duplication des données pour s'adapter au support de stockage. A partir de ce constat nous pouvons imaginer de nombreuses possibilités pour optimiser le chargement de nos données : fichiers par flux sur le disque optique et fichiers organisés de manière séparée sur le disque dur, en chargeant les données du terrain sur le disque en flux en fonction des déplacements du joueur et celles du disque dur pour les événements aléatoires ou les chargements de ressources contenant beaucoup de données.

Une technique pour constituer les fichiers de flux est de placer des logs dans le code pour voir dans quel ordre les ressources sont chargées, ainsi il suffira de les placer dans le même ordre dans le fichier de flux.

3.4 Chargement en tâche de fond

Le chargement des fichiers nécessaires pour mettre en place un jeu de type Open World peut être immédiat comme prendre plusieurs secondes pour être accompli. Ce type de jeu ne présentant normalement pas d'écran de chargement, il ne faut pas que cette tâche soit bloquante pour l'utilisateur du jeu, il faut donc prévoir un chargement en tâche de fond, qui n'altère pas l'expérience de jeu sur la machine utilisée pour jouer.

Pour effectuer ce chargement de la manière la plus fluide possible, l'équipe de Dungeon Siege l'a découpé en 3 étapes distinctes[8] :

D'abord, les informations sur le mesh sont chargées (sans les textures et les animations), puis l'objet 3D est ajouté aux nœuds de la scène une fois chargé. Ensuite, si le mesh est dans le frustrum du joueur, la texture est chargée et appliquée à l'objet 3D. Ainsi ils économisent le chargement de la texture dans le cas ou l'objet 3D ne sera jamais rendu à l'écran.

Notons tout de même qu'en cas de déplacement brusque ou de nombreux objets à afficher, il se peut que le joueur aperçoive un court instant des objets 3D sans texture. Pour améliorer cela, nous pouvons charger les textures en fonction de l'espace mémoire disponible pour les objets proches du joueur, qu'ils soient dans le frustrum ou non.

Ce découpage permet d'annuler le chargement en cours de route si le joueur décide de changer de direction ou passe trop vite pour que l'objet soit dans son champ de vision.

Le chargement en tâche de fond prend aussi des ressources processeur, utilisée par le jeu en tâche principale et ce à cause notamment de la compression des fichiers (voir section Optimisation par compression). Il convient donc de limiter cet usage pour laisser au jeu le plus possible de capacité de calcul pour que le chargement ne fasse pas chuter les taux de rafraichissement.

3.5 Définir des zones

Pour permettre un jeu fluide et éviter les écrans de chargements, il convient d'anticiper les déplacements du joueur et de connaitre les zones qu'il peut potentiellement visiter. Ainsi, il est utile de connaitre la zone dans laquelle est placé le joueur ainsi que les zones qui l'entourent. Lorsqu'un joueur visitera une zone, les zones d'à côté seront chargées en tâche de fond et celles qui sont éloignées seront déchargées ou gardées uniquement sous forme simplifiée.

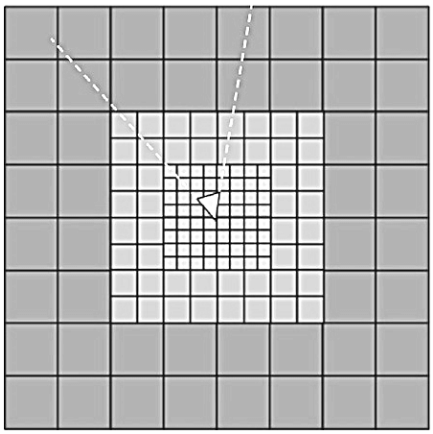

FIGURE 5 – Système de patch

La figure ci contre représente un système de patch sous forme d'arbre avec plusieurs niveaux de détail. Au centre, les patch sont détaillés et découpés en petites zones et sont chargées avec tout leur contenu en mémoire. Les cellules moyennes qui sont plus éloignées du joueur, sont elles chargées de manière incomplète (seul les données importantes au rendu à moyenne distance : données terrain, textures, habitations, arbres, etc.). Les patch plus gros sont eux chargés grossièrement, pour avoir un rendu simpliste nécessitant peu de ressources. Ainsi en fonction du déplacement du joueur, les zones autour de lui vont changer en suivant la même direction pour passer en mode détaillé, moyen ou léger. Un problème peut survenir dans le cas où le joueur se déplace de manière rapide. Dans ce cas, seul les zones contenues dans le frustrum pourront passer en détaillé, tandis que les celles derrière le véhicule et sur les côtés pourront passer plus rapidement en mode léger.

Un second problème se pose dans le cas ou un déplacement instantané peut survenir (téléportation). Pour résoudre en partie ce problème, il convient de trouver les zones dans lesquelles le joueur va pouvoir se téléporter et de les charger en mode moyen à l'avance si l'espace mémoire le permet, même si elles ne sont pas voisines. Ainsi le joueur pourra être téléporté et la zone finira de charger du mode moyen au mode détaillé en cours de route. Ce processus peut intervenir à proximité d'un portail de téléportation, ou lorsque le joueur est dans son inventaire et possède des objets qui lui permettent de se téléporter.

3.6 Prioriser les chargements

Une technique pour rendre les chargements le plus discret possible serait de définir un ordre de priorité pour les ressources et objets à afficher.

Par exemple, nous pouvons retarder le chargement et l'apparition d'un objet qui n'a qu'une utilité esthétique dans notre jeu ou notre scène. Les objets à charger en premier lieux sont ceux avec lequel le joueur peut entrer en interaction (levier, bouton sur lequel le joueur doit appuyer, porte, etc.) pour qu'il puisse continuer à avancer dans avoir à attendre que les objets apparaissent ou qu'il ne se retrouve pas bloqué sans comprendre ce qu'il se passe. Les murs et le terrain, même s'ils ne sont pas des objets interactifs doivent être chargés également car sans eux le joueur se déplacerait dans le vide et n'aurait aucun moyen de se repérer dans l'espace.

Les objets à charger en suite sont les NPC, ou personnages non joueur : un animal dans le décors que le joueur peut tuer pour récolter des ressources, ou un personnage donneur de quête par exemple. Ceux ci ne sont pas critiques pour l'avancée du joueur et souvent nécessitent un certain nombre de ressources (modélisation, sons et animations) ainsi il convient de les charger en second lieux.

Viennent en dernier lieux les objets de décoration, la végétation, et les sons. Cela ne pose pas de problème de permettre au joueur d'avancer avant d'entendre ses bruits de pas par exemple. Avoir différentes textures de fleur pour un sol en plein air n'est pas important pour le joueur, tout comme des tuyaux qui viendraient joncher un couloir. Les charger en dernier permet d'avoir un affichage rapide des niveaux et de réduire au maximum les temps d'attente du joueur.

3.7 Préparer des billboards

Les billboards sont les textures qui remplacent les objets 3D au loin dans les jeux, pas exemple les arbres. Ils font toujours face à la caméra du joueur et sont généralement rafraichies que lorsque le joueur se déplace d'une certaine distance par rapport à sa position de base lorsque le billboard a été calculé la première fois.

FIGURE 6 – Billboards dans Battlefield 3 (arbres au loin)

Nous pouvons nous inspirer de cette technique pour améliorer les temps de chargement, en préparant au préalable des billboards pour certains objets 3D non prioritaires qui seront alors chargés en premier lieu sous la forme de billboards. Si nous prenons l'exemple d'un joueur qui entre dans une maison et se dirige à proximité de la cuisine, nous pouvons lui afficher les tasses et couverts qui sont posés sur la table sous la forme d'un billboard, et si le joueur s'attarde dans cette même cuisine ou si le CPU le permet et qu'il n'y a pas d'autres objets plus prioritaires à charger dans la queue, nous chargeons le vrai objet 3D.

4 Optimisation par compression

4.1 Généralités

La compression de données regroupe un ensemble de techniques permettant de réduire considérablement le volume de données avec ou sans perte d'information. On utilise généralement des algorithmes qui permettent de compresser les données. Si l'algorithme utilisé est sans perte, la totalité des données compressées est récupérée lors de la décompression tandis que si l'algorithme est avec perte, seule une partie des données originales est récupérée (on essaie alors de garder les informations pertinentes). La perte reste minime en fonction des algorithmes utilisés ce qui rend la reconstruction de l'information raisonnable et satisfaisante.

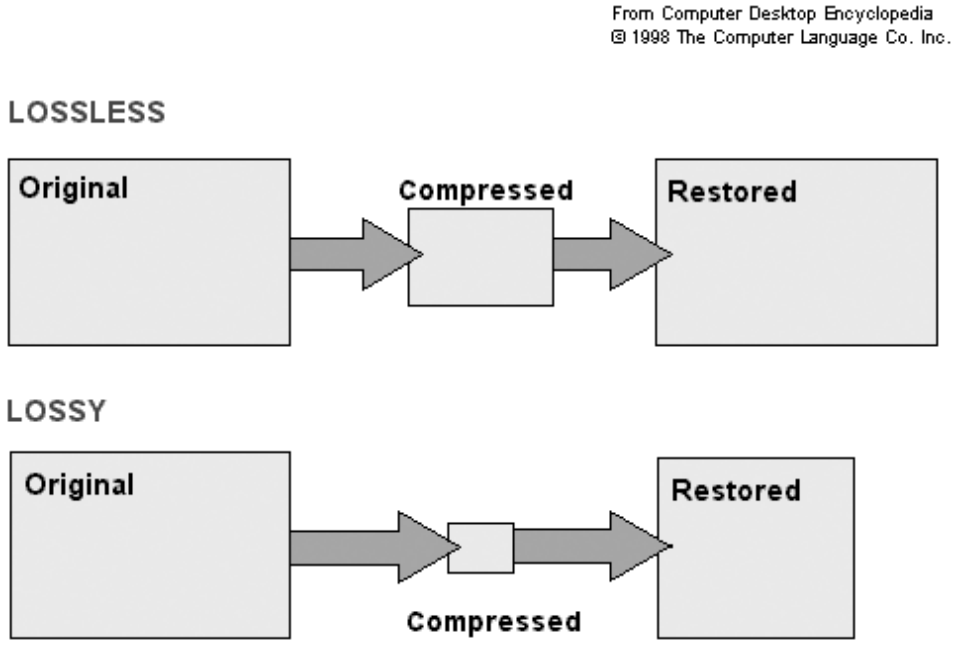

FIGURE 7 – Compression des données avec et sans perte

Un fichier compressé prend moins de place sur un support de stockage qu'un fichier non compressé, cela permet de réduire le temps de passage d'un fichier du support de stockage à la mémoire de l'ordinateur ou de la carte graphique. Autrement dit le volume de la donnée compressée est plus petit, de même le temps des transferts est réduit. Puisque la taille des données contenues dans les jeux est parfois très grande, il serait impensable de les stocker sur le support de stockage sans les compresser au préalable. De plus en plus, la compression devient quasiment indispensable pour

stocker la totalité d'un jeu. En effet, même si la capacité et les performances des différents composants des ordinateurs ou des consoles évoluent d'année en année, le volume des données est tellement grand que nos machines ne parviennent pas toujours à gérer une quantité aussi grande d'information.

Dans les jeux vidéo de nombreux éléments sont utilisés et doivent être stockés (images, sons, objets 3D, etc.). De ne jours, la totalité des données d'un jeu peut être contenue sur un disque et il s'agit très souvent de données compressées qui doivent être décompressées lors de leur chargement en cours de jeu.

Étant donnée la diversité des éléments qui composent un jeu, plusieurs techniques spécifiques sont parfois nécessaires pour compresser et décompresser tel ou tel type de fichier. Ainsi les images, les vidéos et les sons ne seront pas forcément traités de la même manière. Ces éléments sont souvent très volumineux et le but est de parvenir à les compresser au maximum pour atteindre une vitesse de transfert satisfaisante tout en conservant un niveau de qualité le plus élevé possible.

Les données des images et des vidéos sont mesurés par un nombre d'unité binaire (bits), c'est pourquoi on parle de vitesse de transfert de bits (bit rate) comme étant le paramètre principal lors de la compression de données.

4.2 Compression des images

Les images sont très utilisées et en grande quantité dans les jeux vidéo. C'est pourquoi la compression joue un rôle très important. Ils existent plusieurs algorithmes de compression qui permettent de comprimer les images. Bien entendu, certaines méthodes seront plus adaptées pour un type d'image particulier que pour un autre.

Voici les différents types d'images et leurs caractéristiques :

Format	Nom	Caractéristique
BMP	Windows bitmap	Format non compressé
TIFF	Tagged Image File Format	Sans perte
PNG	Portable Network Graphics	Sans perte, amélioration du format GIF. Basé sur l'algorithme de compression Deflate
JPEG	Joint Photographic Experts Group	Avec perte, taux de compression élevé
JPEG 2000	Joint Photographic Experts Group 2000	Avec perte, une amélioration du format JPEG

Quelques remarques concernant ces différents formats d'image :

- Le format JPEG a un taux de compression élevé, réduisant ainsi la qualité de l'image. Il est idéal pour les grandes images et les photos.
- Le format PNG est un algorithme de compression sans perte. Il est très efficace pour des images contenant peu de couleurs différentes ou possédant des grandes zones contenant une couleur unie.
- Il est préférable d'utiliser le format PNG pour des images contenant du texte.
- Le format JPEG 2000 a un taux de compression supérieur à celui du format JPEG.

Voici les résultats récupérés dans le document "Comparison of different image compression formats"[9] présentant les tests de Paula Aguilera concernant les différentes type de compressions d'image :

4.2.1 Compression sans perte

Pour une image de 24 bit de profondeur de couleur pesant 696KB dans un format BMP contenant de nombreuses zones de couleur homogène.

Format	Poids (en KB)	Taux de compression
BMP	696	1 :1
TIFF-LZW	378	2 :1
PNG	258	2.7 :1
JPEG	43.3	16 :1

Pour une image en niveau de gris pesant 257KB dans un format BMP contenant beaucoup de détails.

Format	Poids (en KB)	Taux de compression
BMP	257	1 :1
TIFF-LZW	251	1 :1
PNG	173	1.5 :1
JPEG	79	3.2 :1

4.2.2 Compression avec perte

Pour une image de 24 bit de profondeur de couleur pesant 768KB dans un format BMP contenant des couleurs vives et des textures. Pour une compression au format JPEG, on peut choisir le degré de qualité que l'on souhaite conserver lors de la compression. Il s'agit d'un nombre entre 1 et 100, plus le chiffre est grand plus le taux de compression sera faible et donc plus la qualité d'image sera élevée. Voici les résultats des tests avec une compression au format JPEG avec des niveaux définis à 100, 50, 10 et 1.

Format	Poids (en KB)	Taux de compression
BMP	768	1 :1
TIFF	768	1 :1
PNG	768	1 :1
JPEG - 100	334	2.3 :1
JPEG - 50	49.5	15.5 :1
JPEG - 10	16.3	47 :1
JPEG - 1	6.3	121 :1

D'après ces tests, les compressions en format PNG et TIFF ne sont pas efficaces à cause des couleurs vives et des textures.

De manière générale, lors d'une compression le format PNG est un très bon choix car le taux de compression est raisonnable avec une qualité d'image satisfaisante, tandis que le format JPEG donnera un taux de compression élevé mais une qualité d'image moins intéressante. Même si le poids des images JPEG est plus faible que celles en PNG, leur qualité est appauvrie par la compression.

Il faut ensuite s'intéresser au temps de chargement nécessaire pour décompresser le fichier en mémoire pour pouvoir l'utiliser et cette étape du chargement sollicite le processeur.

4.3 Compression des meshes

Les jeux vidéo utilisent généralement des meshes (maillages) représentant toutes les formes possibles (personnages, bâtiment, véhicules, objets, etc.). Dans le domaine des jeux vidéo, on préfère avoir des maillages composés uniquement de triangles pour leur simplicité. Le nombre de polygones que les machines peuvent afficher est limité, c'est pourquoi on utilise les maillages. Ces formes sont plus ou moins complexes et doivent souvent être compressées pour des questions de volume et de transfert. Pour des jeux en 3D, une carte graphique peut afficher des milliers de polygones (en 2005, une carte graphique pouvait afficher 180 millions de triangles par secondes)[10].

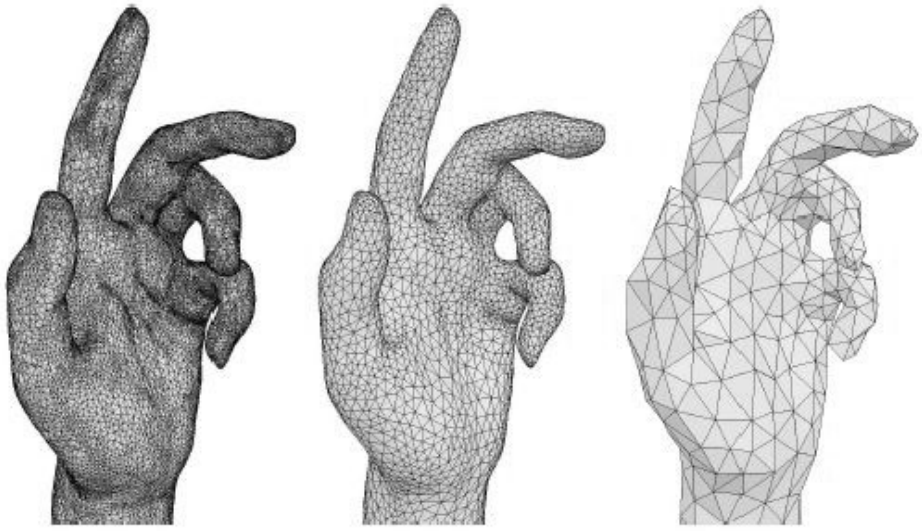

FIGURE 8 – Différents maillage de la surface d'une même forme

Une forme contenant de nombreux triangles sera plus complexe et donc plus détaillé qu'une forme simple qui en contient moins.

Plus le nombre de polygones que l'on souhaite afficher à l'écran est élevé, plus les capacités de la machine seront sollicitées. Dans les jeux 3D à monde ouvert, il faut charger et afficher les éléments de l'environnement les plus

proches du joueur mais aussi parfois les plus éloignés (paysages lointains par exemple). Le but est alors de réussir à charger ces différents objets le plus rapidement possible pour que le jeu soit fluide (le joueur préfère ne pas voir les objets apparaître devant son personnage). La vitesse de chargement des meshes dépend de leur taille, du nombre de polygones dont ils sont composés et des ressources de la machine. En effet il faudra plus de temps pour charger un objet très détaillé (définit par de nombreux polygones) qu'un objet simple. Une astuce consiste à réaliser des meshes complexes pour les éléments qui seront affichés près de la caméra, tandis que pour des éléments qui seront toujours éloigné du personnage tel que les montagnes ou les nuages, il s'agira plutôt de meshes simplifiés. Grâce à cette méthode, la vitesse de chargement de l'environnement est relativement plus rapide.

La compression des meshes permet aussi d'obtenir des fichiers moins volumineux. On distingue toujours la compression avec ou sans perte. La première insiste sur le fait que certains détails seront supprimés avant la compression, il s'agit en général d'éléments jugés non perceptibles par l'œil humain lors de la visualisation.

4.4 Compression des vidéos

5 Optimisation par le level design

5.1 Conception du monde

Une autre façon d'améliorer les temps de chargement est d'adapter le level design afin de satisfaire les exigences des plate formes visées. Par exemple, si nous prenons la carte d'un jeu à monde ouvert comme GTA 3, sorti sur Playstation 2 et PC nous pouvons voir que les différentes parties de la ville sont organisées de façon à laisser un 'couloir' disponible pour que les chargements liés aux déplacements du joueur puissent avoir lieu.

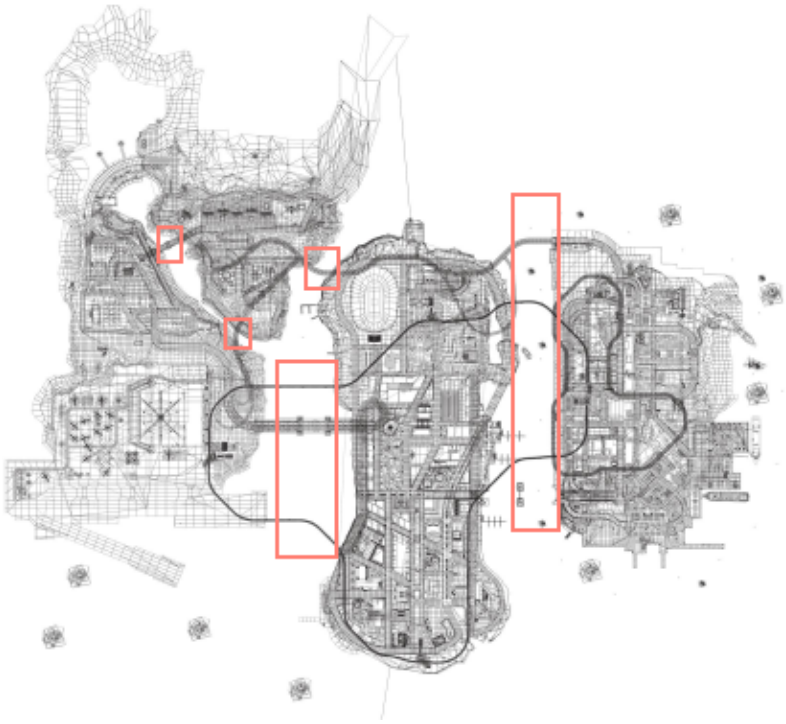

FIGURE 9 – Zones de chargement dans GTA 3

Cependant, les chargements sont appliqués de manière assez brutale, le jeu étant bloqué en attendant la réception des données. Rockstar a donc échoué à proposer un chargement fluide malgré l'adaptation du level design, cependant cette technique leur a permis d'imposer un chargement dans des zones sans intérêt pour que le joueur puisse profiter d'un jeu fluide en dehors de ces zones tampons.

5.2 Intégration au monde

Les chargements peuvent être intégrés au jeu et complètement transparents pour l'utilisateur. Si nous prenons l'exemple de Metroid Prime sur Gamecube, le jeu très détaillé nécessite de charger beaucoup de données. Cependant, il ne présente aucun écran de chargement. Les développeurs ont eu une idée astucieuse pour camoufler ces chargements et savoir dans quelle zone le joueur va se diriger pour charger uniquement le nécessaire.

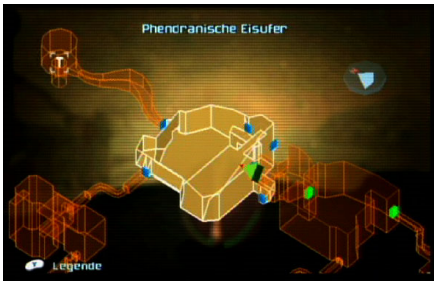

FIGURE 10 – Salle d'un niveau de Metroid Prime

Sur l'image ci-contre, nous pouvons apercevoir une salle d'un niveau du jeu, avec en bleu les portes qui permettent au joueur de quitter la salle pour en rejoindre une autre. A priori, impossible de savoir vers quelle porte le joueur va se diriger, et charger l'ensemble des zones connectées prendrait trop de mémoire et ne permettrait pas d'avoir un monde aussi détaillé. Pour résoudre ce problème, les développeurs du jeu ont trouvé une astuce : demander au joueur de réaliser une action vers la porte qu'il souhaite franchir, en tirant dessus. Une fois le tir envoyé sur la porte, le chargement de la zone connectée est lancé.

Visuellement, le joueur comprend la situation : une porte bleue est une porte qui peut être ouverte avec un tir, et une fois que le halo bleu a disparu, la porte se prépare à être ouverte au bout d'un certain délai, plus ou moins long. Cependant le fait que l'utilisateur soit actif pour ces zones de chargement donne l'impression que le jeu n'en présente pas.

FIGURE 11 – Porte avant activation

Ce mécanismes se retrouvent dans de nombreux autres jeux, où un obstacle nous fait face (souvent une porte ou un portail) et ne s'ouvre que lorsque la zone suivante est complètement chargée. Dans un monde ouvert, cela peut poser un problème puisque cette méthode semble plus adaptée aux jeux découpés en salles. Cependant nous pouvons appliquer ce mécanismes aux habitations qui bordent les routes, aux grottes bloquées par un rocher à exploser, etc.

6 Conclusion

Références

1. Computer Hope. Cd-rom transfer speeds, 2012.
2. Gabriel Torres. Measuring cd-rom and dvd-rom performance, 11 2004.
3. Wikipedia. Blu-ray disc, 2013.
4. Wikipedia. Hard disk drive, 2013.
5. Wikipedia. Solid-state drive, 2013.
6. Fredrik Lönn (Gamasutra). Streaming for loading in next generation games, 2000.
7. Gamasutra. About streaming in just cause, 2013.
8. Scott Bilas. About streaming in dungeon siege, 2003.
9. Paula Aguilera. Comparison of different image compression formats.
10. Pierre Alliez. Meshes compression, 2005.