1
Présentation et histoire

Dans ce chapitre, je vais vous expliquer la nature d'OpenGL et le situer parmi les logiciels permettant de faire de la synthèse d'images. Ensuite, je raconte son histoire et je finis par une présentation de son état actuel.

1.1. La synthèse d'images et OpenGL

La synthèse d'images est une discipline à cheval entre les sciences et les arts graphiques. Elle consiste à produire une image de toutes pièces à partir de données abstraites. Une telle image n'est pas un collage de photographies réalisé avec un outil de dessin d'images comme GIMP ou Photoshop, mais la représentation d'une scène entièrement modélisée par des concepts mathématiques. Par exemple, une fractale de Mandelbrot est définie par un algorithme mathématique. Selon les méthodes et les intentions de l'auteur, les images se veulent réalistes ou au contraire totalement abstraites.

OpenGL vise plus particulièrement le dessin d'images reproduisant le plus rapidement possible une certaine réalité. Le problème de fond est que la réalité fait intervenir des phénomènes physiques – les interactions entre la lumière et la matière qui sont extrêmement complexes en termes de quantité, de vitesse et de modélisation. Dans une scène réelle, il est extrêmement difficile de prendre en compte tous les photons, de savoir ce qui se passe à chaque instant (par exemple les équilibres qui se créent entre les surfaces qui émettent de la lumière et celles qui en absorbent) et ce qui arrive au niveau le plus fin à n'importe lequel de ces photons lorsqu'il interagit avec une surface. Donc, ce qu'on peut faire en synthèse d'images est une simplification, et encore plus avec OpenGL à cause de la nécessité d'aller vite.

OpenGL est une *API*[1] pour dessiner des scènes tridimensionnelles. Il existe des API pour dessiner en deux dimensions, par exemple Cairo, SDL_gfx ou GDI. Ces bibliothèques offrent des fonctions pour dessiner toutes sortes de formes : lignes, rectangles, ellipses, etc. Ces formes sont définies par des points qui ont deux coordonnées (x et y), donc sont seulement dessinées dans un plan. Certaines de ces bibliothèques permettent de superposer plusieurs tracés à l'aide de calques (*layers*), mais cela ne constitue pas pour

[1] Consulter le glossaire pour une définition.

autant une scène tridimensionnelle. Dans OpenGL, il y a également des fonctions pour dessiner des formes simples : lignes, triangles, etc. mais les points qui les définissent ont trois coordonnées x, y et z totalement libres. Nous verrons plus loin comment elles sont définies, mais cela signifie qu'il y a un axe de plus pour positionner les tracés et construire une scène ayant du volume, une profondeur. D'autre part, OpenGL permet de simuler très facilement une sorte de caméra, un point de vue dont la position est entièrement libre dans la scène. À chaque position de cette caméra correspond un dessin différent sur l'écran. Si on se rapproche d'un élément, celui-ci devient plus grand, et inversement comme dans la réalité.

Évidemment, un écran ordinaire n'affiche qu'une image plate, c'est-à-dire en deux dimensions, car c'est une surface sur laquelle la scène 3D est projetée et non pas une zone d'espace visible comme avec un hologramme. Un écran 3D permet de percevoir la profondeur, en général à l'aide d'une paire d'images spécifiquement destinées à chaque œil. L'avenir nous offrira peut-être des écrans holographiques qui nous donneront la pleine perception d'un espace à trois dimensions.

Plusieurs techniques pour dessiner une image 3D existent. Vous avez peut-être entendu parler du lancer de rayons ou du calcul des radiosités. Elles ne sont pas encore temps réel, il faut plusieurs minutes à plusieurs heures pour dessiner une seule image. En contrepartie, la qualité est très importante : ce type d'images paraissent très réalistes. Inversement, OpenGL dessine des images très rapidement, mais au prix d'une qualité moindre, car certains effets visuels ne peuvent pas être réalisés. Ces effets sont, entre autres et pour simplifier, tout ce qui concerne le passage de la lumière à l'intérieur d'objets transparents : réfractions sur des surfaces courbes et reflets caustiques par exemples. J'en reparlerai dans la suite de ce livre.

Un logiciel basé sur OpenGL fonctionne ainsi. Votre programme, utilisateur de fonctions OpenGL, crée des structures de données représentant des objets 3D, par exemple des cubes, des sphères, etc. pour définir une scène. Ensuite, votre programme positionne une sorte de caméra. Le travail d'OpenGL consiste à dessiner la scène telle qu'elle paraît vue de cette caméra. Tout cela est fait par des appels de fonctions dans le cadre d'un programme. Dans ce sens, ce n'est pas interactif. Si vous voulez changer la scène, vous devez changer les instructions écrites dans le programme. Les logiciels de création 3D appelés *modeleurs 3D* tels que Blender, 3ds Max, Maya, etc. font la même chose, mais avec une interface utilisateur interactive. Seulement, ils ne sont pas du tout faits pour être intégrés à un programme. Au contraire, c'est à votre programme de s'intégrer à eux, par exemple sous la forme d'un module (*plugin*). L'API OpenGL, à la différence de ces logiciels, ne fournit aucune interface, aucune vue, aucune boîte à outils. C'est seulement une bibliothèque de fonctions de dessin pour écrire des programmes. Pour disposer d'une interface, il faut intégrer la partie OpenGL dans une fenêtre construite à

l'aide d'une autre API, par exemple, Windows, GLFW, Gtk, Qt, wxWidgets, etc. dans le cas d'un programme binaire exécutable. D'autres API existent pour des architectures spécifiques, par exemple iOS et Android pour les *smartphones*, ou HTML5 en intégration dans un navigateur internet.

Il existe des alternatives à OpenGL. Au même niveau de fonctionnalités, il y a DirectX sur Windows et XBox. Cette API est assez proche d'OpenGL. Elle évolue plus vite à cause du marché des jeux vidéo. À un niveau plus élevé mais moins finement personnalisable, il y a Ogre, Unity, CryEngine, pour n'en citer que trois. Ces derniers sont appelés *moteurs 3D*. Ils s'appuient généralement sur OpenGL ou DirectX. Leur intérêt est qu'ils ont une notion très complète d'une scène 3D : il reste seulement à ajouter des objets. Les calculs d'éclairages, d'effets visuels et d'animations sont déjà en place. Ils masquent totalement les mécanismes d'affichage sous-jacents. Cela facilite la création de logiciels de type jeu vidéo en vue subjective, mais rendent difficile la réalisation d'un logiciel différent de ce qu'ils prévoient. Inversement, OpenGL ne propose rien d'autre que des primitives de dessin très simples. Tout est à construire.

Note *› OpenGL fait un usage intense de la carte graphique de l'ordinateur. Cette carte graphique est presque un ordinateur à elle seule. En effet, elle contient un processeur de calcul, appelé GPU [Graphics Processing Unit en anglais]. C'est un dispositif comptant maintenant davantage de transistors*[2] *que le processeur principal. Ce dernier est appelé CPU [Central Processing Unit en anglais]. Le CPU Intel i7/8 à six cœurs compte 2.27 milliards de transistors, tandis que le GPU nVidia Kepler G110, sorti à peu près en même temps, en compte 7 milliards. Toutefois, il ne faut pas comparer ces populations sur leur seul effectif ; tous ces transistors ne sont pas employés de la même manière. La carte graphique possède aussi de la mémoire vive, plusieurs Go de mémoire dynamique extrêmement rapide à la fois par le temps de réponse et par la largeur du bus de communication avec le GPU. Pour achever la comparaison avec un ordinateur, la carte graphique dispose de deux interfaces : une sortie vers un écran aux normes VGA, DVI ou HDMI et une interface de communication avec le CPU appelée bus PCI express (ou AGP sur les anciens ordinateurs).*

La carte graphique est gérée par ce qui s'appelle un pilote d'affichage [*graphic driver* en anglais]. Le pilote *implémente*, c'est-à-dire met en œuvre, toutes les fonctions d'OpenGL, et celles de DirectX sur un PC Windows. Quand vous demandez de dessiner un triangle, c'est le pilote qui traduit cette demande en instructions et configurations internes, spécifiques de votre modèle de GPU.

Le pilote graphique fait en sorte qu'OpenGL soit le plus rapide possible et réciproquement OpenGL a été défini pour être exécuté très rapidement sur les cartes graphiques actuelles. Le développement d'OpenGL est synchronisé avec les progrès des cartes graphiques. Les dernières versions d'OpenGL ont été définies pour correspondre à ce qui

[2] Un transistor est un composant élémentaire d'un circuit intégré logique, qui sert à construire des portes logiques ; celles-ci permettent de faire des calculs très simples avec des informations binaires : additions, comparaisons, logique, etc.

est maintenant possible avec les nouveaux GPU, et cela évolue en permanence. Pour être portable et dans le cadre de cette évolution permanente, OpenGL définit certaines fonctionnalités en option : certains pilotes de cartes graphiques très récentes les font faire par le GPU ; d'autres les implémentent seulement en mode logiciel, c'est-à-dire que les calculs sont faits par le CPU, et donc sont beaucoup plus lents ; et certains autres n'ont pas encore ces fonctionnalités : soit la carte graphique n'est pas suffisamment récente, soit le pilote n'est pas encore complet. Les différentes variantes d'OpenGL prennent d'ailleurs cela en compte.

Il est important qu'OpenGL soit très rapide car l'un de ses buts est de dessiner en temps réel des scènes animées, contenant de nombreux objets en mouvement. Ce mouvement est une illusion ; il est obtenu par des dessins successifs, à une cadence élevée, au moins une vingtaine par seconde. Les petits changements visibles d'une image à l'autre créent une illusion de mouvement continu. Ils sont obtenus en définissant les coordonnées des objets par des équations liées au temps, ainsi que le montrera la suite de ce livre.

Les mathématiques sont au cœur de la synthèse d'images, et plus particulièrement la géométrie. Les notions de vecteur, de point et de repère orthonormé seront considérées comme des prérequis pour ce livre, mais sans plus. Les autres concepts tels que les matrices de transformation, les produits scalaires et vectoriels seront expliqués avec une volonté de simplification et de bonne compréhension intuitive. L'un des attraits de la synthèse d'images avec OpenGL est de donner un sens à des concepts mathématiques relativement complexes. Des notions qui sont écrites à la manière formelle des mathématiques deviennent limpides lorsqu'elles apparaissent sous forme d'une belle image.

1.2. Brève histoire d'OpenGL

OpenGL trouve son origine dans la très forte compétence en synthèse d'images de l'entreprise Silicon Graphics Inc (SGI). Cette société a été créée en 1982 par des universitaires de Stanford. Elle a développé de nombreuses stations de travail spécialisées en synthèse d'images. La société a travaillé à la fois sur les aspects matériels : carte mère avec processeur graphique ; et sur les aspects logiciels avec un système d'exploitation dérivant d'Unix et une bibliothèque de fonctions 3D appelée IrisGL dont est issu OpenGL. Malheureusement, d'une certaine manière, elle a été victime de son succès : ayant ouvert la voie à OpenGL et aux cartes graphiques puissantes pour PC, elle a rendu caducs ses propres produits.

OpenGL est né d'IrisGL. À l'époque, au tout début des années 1990, un autre standard existait, PHIGS, mais il ne s'est pas imposé car IrisGL était plus souple, plus polyvalent. IrisGL demandait un peu plus de travail pour écrire un programme, mais il permettait davantage de choses au programmeur. L'un des avantages d'IrisGL sur PHIGS était