Les API 3D pour Java

par

Date de publication :

Dernière mise à jour :

Actuellement il n'existe aucune API pour faire de la 3D dans le JDK officiel. Cependant il existe plusieurs projets qui comblent ce manque. Certains projets sont développés par Sun eux-mêmes.

1 - Introduction

Java est un langage puissant, possèdant une API très riche. Cependant il n'existe actuellement aucune API permettant de faire de la 3D dans le JDK.

Cependant nous allons voir que plusieurs librairies viennent combler ce manque, et certaines le font même très bien. Mais avant, voyons quelles sont les API permettant d'obtenir une accéleration graphique matérielle.

1-1 - Direct3D

Il s'agit de l'API de Microsoft, qui de ce fait est donc propriétaire. Cette API est disponible uniquement sur la plateforme Windows. Comme cela est contraire aux principes de Java (portabilité), il y a très peu de moyen d'accéder à cette API en Java. Il y eut certains projets pour accéder à cette API en Java, mais ils n'ont suscité que très peu d'intêrets auprès des développeurs et ont donc vite été abandonnés.

1-2 - OpenGL

Il s'agit d'une API dévéloppée à la base par Silicon Graphics. Cette API dispose de nombreuses implémentations, dont certaines étant libres. OpenGL est disponible sur de nombreux systèmes. Pour citer quelques exemples de jeux écrits en OpenGL, prenons les fameux développeurs d'ID Software : Quake 3, Doom3 et encore Quake 4 sont réalisés en OpenGL, d'où leur disponibilité sur Windows, Linux et Mac OS X.

Il existe plusieurs implémentations de cette API, comme celle de Silicon Graphics eux-mêmes, mais elle est à l'abandon, celle de Microsoft, mais qui n'est pas à jour, l'implémentation libre Mesa et les implémentations des constructeurs de cartes graphiques.

Pour faire une petite parenthèse sur ces derniers, sachez que NVidia soutient très fortement l'OpenGL et développe des drivers de très bonnes qualité pour l'OpenGL et propose de nombreuses documentations sur cette API.

Du côté de ATI, l'OpenGL est moins présent, leurs drivers OpenGL (surtout celui pour Linux) sont de plus mauvaise qualité que ceux de NVidia. Cependant, suite à une pétition, ils se sont engagés à les améliorer (et même les réécrire). Depuis les choses ont un peu changées, les drivers ATI pour Linux ont été améliorés et permettent de jouer à des jeux 3D nécessitant une accéleration 3D. Ces implémentations sont normalement toutes écrites en C, mais nous verront que nous pouvons quand même très bien y accéder en Java.

2 - Les différents types d'API

Avant de les passer en revu, il faut comprendre la différence entre deux grands types de librairies pour faire de la 3D en Java.

Nous avons d'une part les "bindings OpenGL" et d'autre part les "scénographes" et moteurs 3D.

2-1 - Les bindings OpenGL

Comme je le disais avant, la plupart des implémentations OpenGL sont écrites en C. Donc nous ne pouvons pas y accéder tel quel. Il existe un projet d'implémentation de l'OpenGL en Java, mais pour plusieurs raisons (activité du projet et problème de performance par exemple), nous allons le laisser de côté. Si vous connaissez un peu le Java, vous devez sûrement avoir déjà entendu parler de JNI (Java Native Interface). Il s'agit d'une API du langage Java très puissante. JNI permet d'utiliser du code natif écrit en C/C++ dans du Java. Donc voilà en gros l'idée de ces bindings OpenGL. Il s'agit d'API utilisant JNI pour accéder à l'implémentation en C d'OpenGL. Donc toute cette gestion d'accès au code natif, etc... est fait à l'intérieur de ces API. Nous au final, nous avons des API permettant d'utiliser l'OpenGL (presque) comme si l'on développait en C. Grâce à ça, vous pouvez par exemple utiliser des tutos d'OpenGL écrits en C ou en C++ sans trop de problèmes, car il vous suffit juste d'observer la partie de l'OpenGL et de la réutiliser dans vos codes en Java quasi tel quel. Pour ce qui est des performances, elles sont quasiment identiques que celles en C (en ajoutant les appels JNI et diverses petites choses).

2-2 - Les scénographes

Il s'agit d'API de plus haut niveau. Ils permettent de représenter le monde 3D virtuel et ses objets de façon à ce qu'ils soient plus faciles à manipuler. Ils encapsulent généralement des API comme JOGL ou encore LWJGL. Ici vous ne ferez plus d'appels directs aux fonctions d'OpenGL (même si avec certains scénographes cela reste possible mais falcutatif) mais vous utilisez des fonctions du scénographes. Ces scénographes sont construits au dessus des API comme OpenGL et Direct3D. Le développement est plus facile car vous n'avez pas à vous soucier de ce qui se passe en dessous (bas niveau). Cependant, vous possibilités sont déjà réduites, et les performances sont en général moindres que si vous développiez directement avec les API bas niveau (fonctions OpenGL par exemple).

Nous avons vu la différence entre les deux grands types d'API 3D pour Java.

Je ne vais vous présenter que cinq API. Tout d'abord parce que les autres sont soit abandonnées, soit bien trop jeunes.

Ces APIs sont celles qui sont le plus utilisées et celles pour lesquelles vous avez le plus de chance de trouver un support, de l'aide et de la documentation.

Nous aurons deux bindings OpenGL et quatre scénographes.

3 - Les bindings OpenGL

3-1 - JOGL (Java OpenGL)

JOGL est dévéloppée par Sun et Silicon Graphics eux-même, elle est libre et est sous license BSD. C'est l'API de réference pour faire de l'OpenGL en Java.

JOGL est une API dans un style OO (Object Oriented), c'est à dire que l'API est orienté objet, alors que l'implémentation en C ne l'est pas. Ce qui signifie que les codes OpenGL que vous trouverez en C devront subir quelques petites modifications.

Cependant le JSR-231 va remplacer l'ancien JOGL et sera certainement incorporé directement dans le JDK. *J'ai donc écrit le texte ci-dessous pour qu'il fasse référence au JSR-231, encore en bétâ à ce jour.* Toutefois elle a quelques inconvénients : Pour le système de fenêtrage, elle utilise AWT (une des API graphiques officielles du JDK). Pour certains c'est un avantage, car il est bien plus facile d'utiliser cette API, donc faire de l'OpenGL, dans des applications AWT ou Swing (qui sont donc rarement des jeux, ou plutôt, des jeux 3D). Mais cela implique donc d'un peu moins bonnes performances, mais le plus génânt est le FSM (FullScreen Mode, mode plein écran). Surtout sous Linux, le FSM de AWT n'en est pas un vrai. Cependant ces problèmes n'ont plus lieu d'être. Dans Java 6.0 (Mustang), encore en développement, le support du FSM et du changement de display ont été résolus sous Linux entre autre. De plus il y a de nombreuses améliorations au niveau de AWT/Swing et JOGL.

Quelques démos de JOGL : https://jogl-demos.dev.java.net/ Pour l'installer, rendez-vous ici pour télécharger la dernière version (bloc Current nightly build à droite). Téléchargez les fichiers : jogl.jar et jogl-natives-votreOS.jar. Placez ensuite le fichier jogl.jar dans le dossier jre/lib/ext/ de votre dossier d'installation de Java. Pour l'autre, dézippez le et placez les fichiers dans le dossier jre/lib/i386 de votre dossier d'installation de Java (normallement sous Windows ce sont deux .dll, sous Linux deux .so et sous MacOS X un ou deux .jnilib).

3-2 - LWJGL

LWJGL est dévéloppée par plusieurs grands développeurs de jeux en Java (professionnels ou non), sur leur temps libre. Elle est sous license BSD comme JOGL et son développement est très actif.

Il s'agit en faite d'une librairie légère mais très complète pour le développement de jeux en Java. Elle propose des bindings pour OpenGL, OpenAL (son), DevIL (traitement d'images), FMOD3 (son mais propriétaire) et dispose de fonctions pour la gestions des périphériques utilisateurs (clavier,souris,joysticks,manettes). Pour l'instant nous nous occuperons uniquement de la partie OpenGL.

Tout comme JOGL, elle implémente la quasi totalité d'OpenGL (donc jusqu'à 2.0). Mais elle n'implémente que ce qui est nécessaire aux jeux vidéos, et si il existe deux ou plusieurs moyens de faire quelque chose, elle n'implémente que le plus performant. Elle dispose de son propre système de fenêtrage natif pour chaque plateformes (Windows, Linux et Mac), donc elle ne dépend pas de AWT ni de Swing. Elle permet donc de faire du vrai FSM même sous Linux et Mac OS X.

Cette API utilise un style C-Like, c'est à dire non orienté objet, et quasi identique à l'implémentation en C. Donc par exemple si l'on fait un import static de l'API (il s'agit d'une nouvelle fonctionnalité en Java depuis la 1.5, faire un import static revient à ne pas ajouter le nom de la classe pour les membres et fonctions static du package ou classes importées de la sorte, par exemple au lieu d'écrire GL11.glBegin(), on écrit directement glBegin()), on peut prend en général la partie OpenGL des tutoriaux et codes sources en C/C++ tel quel et la mettre directement dans notre code Java.

Quelques démos de LWJGL : http://lwjgl.org/demos.php Pour installer LWJGL, allez ici et téléchargez la version

pour votre système d'exploitation (et actuellement, la version 0.98, mais si il y en a une plus récente prenez là). Nous allons l'installer de la même manière que JOGL même si le README de LWJGL le déconseille. Téléchargez donc le zip pour votre système d'exploitation et décompressez le. Mettez ensuite tous les fichiers du dossier jar/ (pas le dossier lui même, mais uniquement ce qu'il contient) dans le dossier jre/lib/ext/ de votre dossier d'installation de Java. Ensuite mettez tout le contenu du dossier native/ dans le dossier jre/lib/i386/ de votre dossier d'installation de Java.5

4 - Les scénographes

4-1 - Java3D

Java3D est développé par Sun mais est devenu libre. Il n'est plus aussi actif pour priviligier JOGL, mais depuis peu il a retrouvé un certain engouement auprès des développeurs. Le projet LG3D de Sun (VM en 3D) l'utilise. Il est construit directement au dessus d'OpenGL et de Direct3D pour Windows (même si cette dernière API va être abandonnée par Java3D). Il est encore pas mal utilisé mais pour les jeux et autres applications où les performances sont primordiales, il ne s'agit pas toujours de la meilleure solution. Son architecture est très bonne (en gros il s'agit d'une réprésentation du monde 3D, orientée objet, sous forme d'un arbre sans cycle. C'est un graphe formé de noeuds et d'arcs).

4-2 - Xith3D

Xith3D est le concurrent direct de Java3D. Très fortement inspiré de ce dernier, vous n'aurez presque aucun mal à porter vos codes Java3D votre Xith3D. Les performances jusqu'à présent semblaient meilleures que celle de Java3D mais depuis peu, Java3D semble être repassé devant du point de vue performance. Cependant vous pourrez sans problème développer des jeux 3D avec. Il est construit au dessus de JOGL et de LWJGL (au choix). A la différence de Java3D, vous pouvez continuer à utiliser directement des fonctions OpenGL si vous le souhaitez.

Pour l'installer, suivez les instructions disponibles sur cette page : http://xith.org/tiki-index.php?page=Installing. Pour les petites démos officielles, c'est ici que cela se passe : http://xith.org/demo/com.xith3d.test.php.

4-3 - jMonkey Engine

jME est un scénographe puissant et libre, sous license BSD. Il est actuellement construit au dessus de LWJGL et le sera bientôt aussi au dessus de JOGL. Pour téléchargez la dernière version de jME, c'est ici. Pour l'installation, tout est expliqué http://jmonkeyengine.com/index.php?option=com_content&task=view&id=24&Itemid=47. Pour voir les nombreuses démos de jME, c'est ici : http://jmonkeyengine.com/index.php?option=com_content&task=view&id=25&Itemid=48.

4-4 - Aviatrix3D

Aviatrix3D est un scénographe construit au dessus de JOGL (l'ancien, mais le port vers le JSR-231 est imminent).

Ce scénographe supporte aussi le son (OpenAL avec JOAL -binding Java pour OpenAL-) spatial.

Vous pouvez le télécharger à cette adresse http://aviatrix3d.j3d.org/download.html. De plus, vous trouverez des exemples ici.

5 - Conclusion

Nous avons donc vu qu'il existe finalement de nombreuses API pour faire de la 3D en Java.

Avec l'incorporation du JSR-231 (JOGL) dans le JDK, l'utilisation de la 3D avec OpenGL, au sein des applications graphiques Swing ou AWT, ou non, sera rendue possible sans utiliser de librairies externes.

Il existe aussi de nombreux moteurs graphiques et plateformes de développements de jeux, construits généralement au dessus des API que nous avons vu tout au long de cet article.

Ces moteurs apportent une couche d'abstraction supplémentaire, facilitant donc le développement et nous permettant de nous concentrer sur les parties principales du développement du jeu (gameplay, graphisme...).

Vous trouverez plusieurs très bons articles sur l'intégration de la 3D dans des applications Swing sur les sites et blogs de Gfx : http://www.jroller.com/page/gfx et http://gfx.developpez.com/.