

Introducción a OGRE 3D

Tirado Granados, Gonzalo

gonzalo_tirado@alu.uma.es

Resumen

OGRE 3D es un motor de gráficos 3D de código abierto. Es flexible y orientado a objetos, además de ser portable a múltiples entornos como Windows, MacOS X o Linux. Ha sido diseñado con la idea de hacer más sencillo el desarrollo de juegos 3D. Explora al máximo las posibilidades de las tarjetas gráficas, ya que brinda soporte para vertex y píxel shaders, como HLSL (DirectX), GLSL (OpenGL) y Cg (DirectX/OpenGL). El motor está muy extendido y es muy valorado por la comunidad de código abierto. Con él se han desarrollado muchos juegos, algunos de ellos con carácter comercial.

Palabras clave

Motor Gráficos Renderizado OGRE OpenSource

Introducción

Un motor hace referencia a una serie de rutinas de programación que permiten el diseño, la creación y la representación del juego. La analogía con el motor de un automóvil es bastante ilustrativa: el motor debajo del capó no es visible, pero le da la funcionalidad al automóvil que es la de transportar. La misma analogía permite explicar algunos de los aspectos que, generalmente, maneja un motor de juegos, en el que las texturas y los modelos 3D serían la carrocería, pintura e interiores.

Del mismo modo en que carrocería, pintura y exteriores no andan sin un motor, el arte y los guiones del juego no funcionan sin un motor de juegos. Un motor de juegos es el núcleo software de un videojuego o de una aplicación interactiva con gráficos en tiempo real. Éste provee las tecnologías necesarias, simplifica el desarrollo y, en algunos casos, permite el desarrollo para distintas plataformas (Mac/Linux/Windows/Consolas).

Existen muchas librerías que proporcionan las funciones básicas de un motor de juegos, como DirectX, OpenGL, Java3D, SDL, Newton, entre otros. Las librerías gráficas como OpenGL y

DirectX usan los drivers de las tarjetas gráficas, de forma que constituyen la interfaz entre la aplicación software (juego) y el adaptador gráfico hardware. Aúnan funciones para dibujar objetos 2D/3D y los motores gráficos se suelen sustentar en ellas. Una librería gráfica, a partir de modelos en 3D, consiguen obtener un espacio bidimensional, pero un motor de juegos hace más cosas, como iluminación, anti-aliasing o mezcla de texturas. Existen varias razones por la que el uso de motores de juego es importante. Algunas de ellas son:

- Facilita el desarrollo
- Abre nuevas oportunidades de negocio, como es la venta de las licencias de los motores para la realización de otros juegos
- Abstracción de la plataforma. Si un Motor corre en varias plataformas, nuestro juego también
- Separación de Motor/Contenidos, lo cual permite tener varios grupos de trabajo en paralelo
- Beneficios a terceros, ya que los avances en el motor pueden beneficiar a varios juegos
- Permite enfatizar en la parte artística, ya que, al tener mayor grado de abstracción, se tiene menor carga de trabajo en programación
- Mayor modularidad y reaprovechamiento

Los motores de juego surgen en la década de los 90 tras el éxito de Doom y Quake. Juegos como Quake 3 y Unreal ya se diseñaron separando motor y contenidos ¿Qué ilustra el éxito de estos juegos? Por un lado, una mayor rapidez del mundo de los videojuegos. Desarrollar motores agiliza el desarrollo de juegos y permite sacar secuelas más fácilmente para mantener el mercado “caliente”. Además, el licenciar posteriormente los motores permite obtener unos beneficios extras muy suculentos.

La arquitectura de un motor de juegos debe proporcionar mucha abstracción. Con que las librerías utilizadas estén portadas a todos los SO, permitimos que nuestro motor o juego funcione en todas las arquitecturas necesarias. Un parámetro que se debe tener muy en cuenta al elegir un motor de juegos es su portabilidad y en qué librerías se basa. Los motores de juego suelen tener una estructura como la que sigue:

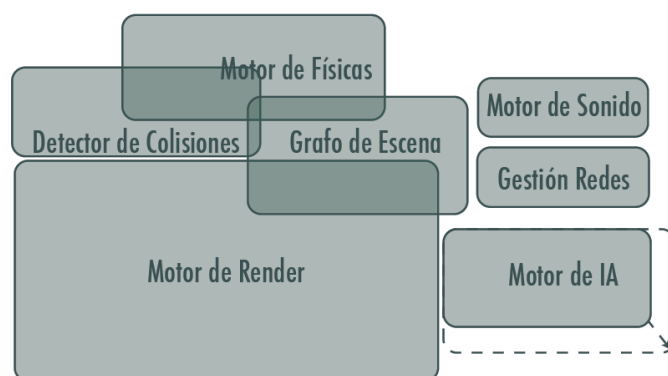


Figura 1: Esquema típico de un motor de juegos

El **motor gráfico** o de render es el que proporciona funciones de renderizado 2D, 3D y de sprites. Se suele apoyar en librerías/APIs gráficas como OpenGL o DirectX. Define una capa de abstracción entre el HW/SO y el juego en sí. Además, se encarga de la visibilidad (BSP y buffers), el mapeado, el antialiasing, la gestión de mallas 3D, entre otras cosas.

El **grafo de escena** es una parte muy importante de un motor de juegos. Ésta es una estructura de datos muy utilizada en editores gráficos basados en vectores y en videojuegos. Se encarga de ordenar la representación lógica y, también a veces, espacial de una escena gráfica. Cada nodo puede tener muchos hijos, pero, un nodo sólo suele tener un padre. El efecto de algo sobre un padre es visible por todos sus hijos. Así, una operación sobre un grupo se propaga hacia todos sus nodos hijo. Esto permite manejar (escalar/mover/rotar/etc.) grupos de objetos como si fueran uno solo. En los juegos, cada nodo suele representar un objeto en la escena.



Figura 2: Ejemplo de representación de varios objetos en grafo de escena

El **motor de físicas** es software que simula modelos de física utilizando variables del tipo velocidad, masa, etc. Su objetivo es simular y predecir los efectos bajo diversas situaciones de lo que ocurre en la vida real o en un mundo de fantasía. En los juegos se utilizan cada vez más, sobre todo en los de carreras donde factores como la fuerza centrífuga o derrapajes necesitan de ecuaciones físicas para aportar realismo.

El **detector de colisiones** es un algoritmo empleado para detectar cuándo dos o más objetos colisionan. Se suelen realizar mediante el cálculo de la intersección de volúmenes simples (Bounding Volumes). Se pueden utilizar distintos volúmenes para envolver el objeto y así detectar mejor las colisiones, como cubos o esferas. Se puede tomar como un componente independiente, aunque tiene mucho que ver con el grafo de escena y el motor de físicas.

El **motor de inteligencia artificial** es el encargado de dotar a ciertos elementos del juego un comportamiento pseudointeligente. Por regla general, lo que se aplica en los juegos son técnicas muy simples de inteligencia artificial como máquinas de estados o algoritmos de búsquedas. Sin embargo, cada vez más se utilizan nuevas técnicas para conseguir una inteligencia más “realista” como redes neuronales o algoritmos genéticos.

El **motor de sonido** es el encargado de reproducir la banda sonora del juego y los efectos de sonido, como disparos o explosiones, en sincronía con la acción del juego.

La **gestión de redes** está alcanzando una progresiva mayor importancia. Hoy por hoy, casi todos los juegos tienen una componente de red. Existen juegos exclusivamente online, de los cuales, la mayoría ofrece partidas multijugador a través de red. Es importante tener un componente que aglutine todas las utilidades de red, de forma que, el hecho de que se juegue de una forma u otra sea, en cierta medida, “transparente” para el desarrollador del juego.

Todas las partes que componen un motor de juegos tienen su importancia, pero lo más básico e imprescindible de un motor de juegos es el motor gráfico. Aunque el Grafo de Escena es también una pieza vital, el resto de componentes son más prescindibles y no aparecen en todos los motores, y cuando lo hacen, aparecen como plugins. La elección del motor de un motor u otro es algo fundamental, sobre todo si necesitamos alguno de estos componentes. Un ejemplo de motor de juegos básico es OGRE 3D, que, básicamente, aglutina únicamente funciones de tratamiento gráfico y deja a elección del usuario las otras componentes. La comunidad de OGRE es la que ofrece multitud de librerías para complementarlo y añadirle funcionalidad, las cuales se integran a la perfección con dicho motor. Pero OGRE 3D no es único, pues existen infinidad de motores. Éstos se suelen clasificar por su licencia de uso:

Los *motores comerciales* suelen tener la ventaja de proporcionar soporte y mayor número de herramientas de desarrollo adicionales. No obstante, coste de licencia puede suponer un gran inconveniente, puesto que suele ser mayor cuantas mejores prestaciones ofrezca el motor. Algunos de los distintos motores comerciales pueden ser:

- Torque Game Engine
- 3D Game Studio
- Blitz3D
- Reality Engine
- Deep Creator
- Cipher

Los aquí mostrados son los que se pueden llamar “asequibles”. Me refiero con ello a los que tienen licencias de bajo coste. Ninguno de estos se han utilizado para los últimos títulos AAA, pero son utilizados frecuentemente por pequeños estudios y grupos amateurs de desarrollo.

Los *motores de código libre* en cambio presentan la ventaja de que no tienen ningún coste, junto a que el usuario tiene la libertad de poder modificarlos si así lo necesita. Por el contrario, tienen la desventaja de que no hay soporte técnico “oficial”, aunque la comunidad suele ayudar, y puede sufrir en determinadas ocasiones falta de herramientas que ayuden al desarrollo. Además, hay que tener en cuenta que muchos de estos motores están regidos por unas licencias de uso limitadas y pueden surgir inconvenientes si el juego que se va desarrollar presenta fines comerciales. Algunos de los motores open source más populares son:

- OGRE
- Crystal Space
- Irrlicht
- jME
- Reality Factory
- RealmForge GDK

Como hemos visto OGRE 3D no es el único motor, e, igualmente, lo mismo no es el que más conviene, ya que cada uno de ellos ofrece diferentes funcionalidades y licencias de uso. Antes de aventurarse con un proyecto es conveniente analizar con profundidad cual es el que más se ajusta a nuestras necesidades, ya que una buena elección puede evitar numerosos quebraderos de cabeza.

Referencias bibliográficas

(Alternativas Libres, 2007) Alternativas libres: *Ogre3D*, web dedicada a la recomendación de aplicaciones Open Source. <http://alts.homelinux.net/libreapp.php?id=574> , 2007.

(Cortizo, 2007) J.C. Cortizo Pérez: *Teoría de motores*, Asignatura Motores del Master de la Universidad Europea de Madrid, <http://www.esi.uem.es/jccortizo/motores/teoria1.pdf> , 2007.