Simulación con Herramientas Open Source

Leonardo E. Fields M., Victor Sánchez U.
Fac. de Ingeniería Mecánica, Universidad Tecnológica de Panamá
leonardo.fields@utp.ac.pa, victor.sanchez@utp.ac.pa

Resumen- *Este artículo brinda una visión general en cuanto a la simulación de entornos tridimensionales por computadora utilizando herramientas de código libre. La tecnología para simulación ha experimentado un crecimiento acelerado en los últimos años gracias a las exigencias del mundo del entretenimiento, entre otras cosas. La existencia de herramientas libres para crear este tipo de entornos resulta útil a la comunidad científica, investigadores, estudiantes y aficionados.*

Palabras Claves- Computación, Simulación, Videojuegos, Código Abierto.

1. Introducción

La simulación de entornos 3D es un área de la computación que ha experimentado un crecimiento acelerado en los últimos años, debido entre otras cosas a las necesidades de la industria del entretenimiento. Esto ha ocasionado la aparición de múltiples herramientas comerciales y gratuitas, restringidas y de código libre que brindan soluciones de simulación a distintos niveles. Esta tecnología ha captado la atención de un grupo muy variado de individuos y entidades entre los que hay desde científicos hasta aficionados.

Existen muchos tipos de simuladores según su objetivo y el nivel de detalle de los mismos; pero a grandes rasgos, un simulador necesita:

- Selección del lenguaje de programación.
- Plataforma de desarrollo.
- Motor gráfico.
- Interacción con dispositivos de entrada y salida.

Otros componentes adicionales pueden incluir: motor físico, acceso a multimedia y acceso a redes.

2. Lenguaje

Existen tres lenguajes principales, ampliamente usados para la creación de simuladores: C/C++, Java y C#.

Java es un lenguaje orientado a objetos que utiliza una máquina virtual para compilar el código y luego hacerlo ejecutable. Es ampliamente usado en aplicaciones web, pero es muy criticado por la lentitud con que se inicializan las aplicaciones debido a su metodología de trabajo, problema que las últimas versiones de Java Virtual Machine han tratado de resolver.

C# es un lenguaje orientado a objetos derivado de C/C++, desarrollado por Microsoft para su plataforma NET. Es utilizado ampliamente en el desarrollo de videojuegos en el set de herramientas XNA de Microsoft.

C inició como un lenguaje estructurado, pero su predecesor C++ puede ser utilizado en aplicaciones orientadas a objetos. Es un lenguaje universal presente en casi cualquier medio. Muchas aplicaciones de todos los niveles son escritas en C/C++, incluyendo simuladores, videojuegos y sistemas operativos.

3. Código Abierto

A diferencia del software libre, el código abierto no se concentra en las implicaciones filosóficas de compartir el código fuente, más bien en los beneficios prácticos de hacerlo. Aún así podemos encontrar muchas aplicaciones de software libre y de código abierto que comparten las mismas licencias.

Entre los beneficios prácticos de usar este tipo de herramientas tenemos:

- Posibilidad de estudiar y modificar el código fuente.
- La mayoría de las aplicaciones y código fuente se distribuyen gratuitamente.
- Amplias comunidades de usuarios activos que brindan respuestas a consultas.

4. Herramientas

Existen muchas formas distintas de crear un simulador, y muchas herramientas disponibles para hacerlo. Por su naturaleza, podemos clasificar las formas de hacer un simulador en dos grandes grupos: usar un motor de simulación o integrar librerías.

4.1 Motor de simulación

Es una sola aplicación que contiene todas las herramientas necesarias para crear simulaciones. Muy similares a los Motores de Videojuegos (Game Engine) que están orientados principalmente a la creación de videojuegos.

Un motor de simulación puede integrar varias herramientas diferentes como motor gráfico, motor físico y acceso a dispositivos de entrada y salida en una plataforma de alto nivel que facilite el desarrollo de proyectos en esta línea. Estos motores pueden contar además con una interfaz gráfica de usuario (para hacer el desarrollo menos orientado a la programación), y con un modelador 3D.

Cry Engine 3 y Unreal Engine son algunos motores de código cerrado para videojuegos que han sido la base de muchos videojuegos comerciales populares.

Blender es un modelador 3D gratuito de código abierto que puede utilizarse como motor para simulación y videojuegos. Tiene una interfaz poco intuitiva, pero sus múltiples características lo hacen muy utilizado.

4.2 Librerías

Algunos desarrolladores evitarán el uso de motores de simulación y videojuegos debido a que desean usar herramientas específicas, no necesitan hacer uso de todas las características que estos proveen, prefieren desarrollar sobre una aplicación más ligera o desean tener más control sobre el código fuente de su proyecto.

Existen muchas librerías de código libre que controlan partes específicas de la simulación, o varias de ellas, haciendo referencias a las mencionadas en la introducción de este documento. Estas variantes pueden incluir librerías que controlen:

- Plataforma de desarrollo.
- Video, audio.
- Video, audio, dispositivos E/S.

- Video, dispositivos E/S.
- Leyes físicas.

Algunas de estas librerías y sus funciones principales son:

- Plataformas de desarrollo: Qt, GTK+, wxWidgets, FOXTOOLKIT.
- Motores gráficos: Ogre, Irrlicht, OpenSceneGraph.
- Motores físicos: Open Dynamics Engine, Bullet Physics.
- Programación de videojuegos: Allegro, SDL, SFML.

Si el desarrollador no desea construir los modelos 3D de la escena usando el motor gráfico directamente, puede hacer uso de modeladores 3D como Blender, Art of Illusion, K-3D.

5. Integración

Para poder utilizar distintas librerías en un solo simulador es necesario integrarlas. El grado de conocimiento necesario para realizar una integración depende de factores como si las herramientas están escritas en el mismo lenguaje, y la forma en que están construidas. No se logra de igual forma para todos los proyectos pues depende de las necesidades de los mismos.

La integración es posible debido a que las distintas herramientas tienen elementos en común que les permiten comunicarse. Por ejemplo los tipos de datos (números enteros, números flotantes y caracteres) y los punteros (si los permite el lenguaje). Incluso si las librerías estuviesen escritas en lenguajes distintos, muchas de ellas pueden trabajar con otros lenguajes mediante el uso de "bindings".

Delta3D es un ejemplo de un motor de simulación que integra distintas librerías, las cuales son: OpenSceneGraph, OpenDynamicsEngine, Character Animation Library, y OpenAL.

Otro ejemplo de integración es la de las librerías Qt, OpenSceneGraph, SDL y Bullet Physics; el cual se explicará en más detalle en los siguientes puntos.

5.1 Las herramientas

Qt es una librería escrita en C/C++ para el desarrollo de aplicaciones con o sin interfaz gráfica de usuario. Cuenta además con un completo entorno de desarrollo con interfaz gráfica de usuario, y complementa su lenguaje nativo C/C++ con métodos, clases y tipos de datos de alto nivel propios de QT.

OpenSceneGraph (OSG) es un motor 3D orientado a objetos que trabaja usando grafos de escena, lo cual evita al programador tener que utilizar rutinas de bajo nivel para construir el entorno. Soporta además varios formatos de modelos 3D y texturas.

SDL es una herramienta para desarrollo de videojuegos. No es un Game Engine, pero provee de varias funcionalidades necesarias para una simulación tales como: video 2D y 3D, audio, acceso a dispositivos E/S, acceso a imágenes, entre otras.

Bullet Physics es un motor físico. Permite simular el comportamiento de los objetos al ser sometidos a las leyes físicas. Permite simular colisiones entre objetos, restricciones, motores y soporta tanto objetos rígidos como blandos.

5.2 La plataforma y el motor gráfico

La ventaja de integrar un motor físico dentro de una plataforma de desarrollo es que se tiene acceso a todas las características de dicha plataforma y pueden utilizarse para robustecer el simulador. Qt posee un módulo de compatibilidad con OpenGL, que le permite integrar en una ventana hecha en Qt, un escenario tridmensional.

OpenSceneGraph es un derivado de alto nivel de OpenGL y cuenta con clases que le permiten integrar escenarios 3D en ventanas de otras herramientas. Existe un ejemplo de cómo integrar ambas herramientas en el sitio Web de OSG - véase la Figura 1.

Figura 1. Integración de Qt y OSG.

5.3 Dispositivos e/s y audio

Aunque tanto Qt como OSG tienen acceso a lecturas desde mouse y teclado, algunos simuladores pueden requerir acceso a otros dispositivos tales como los *joysticks*. Es por eso que para la comunicación con los dispositivos E/S se utiliza SDL.

Para lograr esta comunicación sólo en necesario saber el estado del dispositivo (hacia dónde se mueve o qué botón fue presionado). SDL puede retornar estos valores en forma de números o caracteres.

Reproducir audio también es posible desde SDL, y es recomendable hacerlo desde esta herramienta ya que es una de sus características principales.

5.4 Motor gráfico y motor físico

OSG y Bullet funcionan paralelamente en la ejecución de una aplicación, pero realizan tareas distintas. Uno dibuja el escenario, otro simula la física del mismo.

Para integrar ambas funciones primero debemos darle a todos los objetos del motor gráfico, un equivalente en el motor físico. Este equivalente no necesita ser idéntico en la forma, pero debe conservar un aspecto similar (Figura 2), tener la misma ubicación y orientación en el espacio.

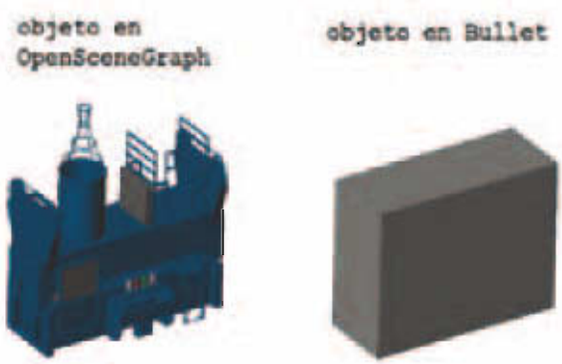

Figura 2. Objeto en OSG y su equivalente en Bullet.

Tanto los objetos de OSG como los de Bullet son colocados en el espacio mediante el uso de transformadas, las cuales son conjuntos de vectores que indican las coordenadas del objeto, su rotación, entre otras cosas.

Para igualar las transformadas de ambos objetos en tiempo de ejecución, puede utilizarse un CallBack de OSG que permite actualizar los valores de la transformada del objeto gráfico, según cambie la transformada del objeto físico. Un ejemplo de esta integración puede verse en el tutorial del sitio web de OSG, "Bullet demo" escrito por Jan Ciger.

6. Nivel de Detalle

Las simulaciones son aplicaciones que consumen gran cantidad de recursos de los computadores. Incluso hoy en día, las exigencias del mundo del entretenimiento ponen a prueba el desempeño de los equipos más modernos, para producir escenarios 3D de alto detalle y animaciones realistas.

Pero muchos desarrolladores y usuarios no tienen acceso a tecnología de última generación lo cual los obliga a usar de forma inteligente los recursos disponibles.

Los desarrolladores de simuladores pueden valerse de algunas técnicas sencillas para mejorar el desempeño de la aplicación en computadoras convencionales sin sacrificar mucho el nivel de detalle:

- Reducir el número de polígonos de los objetos 3D y compensar con texturas más detalladas.
- Reducir el detalle de los objetos en el motor físico y restringirse a usar sólo los necesarios.
- Reducir la frecuencia de actualización de la simulación.
- Asegurarse de optimizar el código mediante eliminar variables y punteros innecesarios; utilizar características de alto nivel de las herramientas y reutilizar.

7. El Hardware

Si se está por adquirir una computadora para desarrollar simuladores con herramientas open source, y se pretende desarrollar bajo un sistema operativo tipo Linux, se debe prestar atención al hardware que se utilizará.

Distribuciones de Linux como Ubuntu usan *drivers* de código abierto, y no los drivers que distribuye el fabricante del hardware, debido a que muchos de estos son exclusivos para sistemas Microsoft Windows. Esto puede causar problemas inesperados, especialmente cuando se trata de las tarjetas de video. Ati y Nvidia, algunos de los núcleos más populares para tarjetas de video tienen *drivers* tanto para Windows como Linux. Sin embargo, comentarios en múltiples foros en la web, parecen indicar que los núcleos Nvidia tienen por lo general un mejor desempeño en Linux.

Referencias

[1] F. Leonardo, "Simulador de Grúas STS con herramientas Open Source", Tesis de Ingeniería, Universidad Tecnológica de Panamá, Panamá, Panamá, Mar. 2011.