Escuela Superior Politécnica del Litoral.

Facultad de Ingeniería Eléctrica y Computación

"CONJUNTO DE PRÁCTICAS PARA EL MANEJO DE DISPOSITIVOS Y HERRAMIENTAS PARA ENTORNOS VIRTUALES"

INFORME DE MATERIA DE GRADUACIÓN

Previa a la obtención del Título de:

INGENIERO EN COMPUTACIÓN ESPECIALIZACIÓN SISTEMAS MULTIMEDIA
INGENIERO EN CIENCIAS COMPUTACIONALES ESPECIALIZACIÓN SISTEMAS MULTIMEDIA
INGENIERO EN CIENCIAS COMPUTACIONALES ESPECIALIZACIÓN SISTEMAS MULTIMEDIA

Presentado por:

LISSETTE LIGIA BRIONES PEÑALOZA
NORMA MERCEDES MENDOZA ZAMORA
JENIFFER ELIZABETH VINUEZA BUSTOS

Guayaquil – Ecuador

2010

AGRADECIMIENTO

A Dios,
por darnos la vida y las oportunidades
para seguir con nuestras metas.

A nuestra Familia,
que es el pilar que impulsa nuestra vida y que
siempre han estado apoyándonos.

A Sixto García,
quien nos ayudó a culminar nuestra meta.

DEDICATORIA

Con mucho cariño,
a nuestra familia y amigos,
por estar siempre con nosotros.

TRIBUNAL DE SUSTENTACIÓN

Phd. Sixto García A.
PROFESOR DE LA MATERIA DE GRADUACIÓN

Ing. Federico Raue R.
PROFESOR DELEGADO DEL DECANO

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Proyecto de Grado, nos corresponde exclusivamente; y el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL"

(Reglamento de Graduación de la ESPOL)

Lissette Ligia Briones Peñaloza

Norma Mercedes Mendoza Zamora

Jeniffer Elizabeth Vinueza Bustos

RESUMEN

Este proyecto tiene como finalidad facilitar la incursión de la tecnología de realidad virtual a los estudiantes de esta institución.

Esta idea surge con la necesidad de complementar el aprendizaje de esta área que recién se esta introduciendo en la universidad.

Se proporcionan un conjunto de prácticas que ayudan al manejo de dispositivos y herramientas para entorno virtuales.

Para la elaboración de un escenario virtual inmersivo se requiere la creación de un ambiente tridimensional, efectos que ayuden a mejorar el realismo tales como texturas, luces y sonido, por último la interacción con la escena mediante los dispositivos de RV. Por tal motivo, basándose en este esquema se diseñaron 12 prácticas.

Estas prácticas consisten en brindar documentación donde se especifica el procedimiento a seguir para cumplir con el objetivo de éstas. Además se incluye una serie de tareas que harán que el estudiante adquiera habilidad del tema revisado.

El trabajo aquí mostrado sienta las bases para incentivar al estudiante en el aporte de futuras prácticas que mejoren el aprendizaje de esta área.

ÍNDICE GENERAL

ABREVIATURAS

Las abreviaturas que se utilizan en este trabajo son las siguientes:

RV: Realidad Virtual

OSG: OpenSceneGraph

3D: Tres dimensiones

OpenGL: Open Graphics Library

ÍNDICE DE FIGURAS

CAPÍTULO I

1 Introducción y análisis de la solución

En el presente capítulo, se describe el problema, el análisis de la solución, los objetivos y el alcance, así como la justificación para llevar a cabo este tema de proyecto.

1.1 Introducción

"Que la creatividad no se vea limitada por el desconocimiento"

Partiendo de esa frase nació la idea que la limitante para el estudiante politécnico, no será la falta de información para desarrollarse en este tema como es la realidad virtual.

Mediante estas prácticas que se proporcionan al estudiante se dan las bases para incursionar en esta área, éstas dejan las pautas para que él desarrolle su creatividad hasta donde el límite de su imaginación se lo permita.

Además el presente trabajo trata de facilitar el proceso de aprendizaje de los estudiantes, mejorando la comprensión mediante los documentos y las tareas que se brindan.

Los estudiantes, antes de motivarse a participar en las tareas, tendrán información clara sobre algunos conocimientos básicos descritos en los documentos, donde obtendrán la explicación detallada para familiarizarse con el tema tratado, además se le explica la importancia de las prácticas revisadas.

1.2 Definición del Problema

Al querer comenzar a trabajar con esta tecnología surgen muchas incógnitas, una de estas es por dónde comenzar. ¿Cómo lograr que el hardware pueda alterar la dimensión de una escena? ¿Cuál es el cuidado que se debe tener al momento de utilizar estos dispositivos? ¿Cómo recorrer un ambiente mediante los dispositivos? Y otras dudas que pudieran tener los estudiantes debido a que la realidad virtual recién se está introduciendo en la universidad.

Ciertamente en la materia de Gráficos por Computadora dictada en la Facultad de Ingeniería Eléctrica y Computación da un indicio en la creación de escenas tridimensionales, pero la utilización de este conocimiento conjunto con los dispositivos de Realidad Virtual es algo que no se ha profundizado.

Al ser recursos recién adquiridos por la ESPOL, el conocimiento en este tema es limitado al no existir las bases para su utilización y cuidado.

El objetivo de este proyecto es proveer al estudiante una base para que pueda desarrollarse en este tema.

1.3 Análisis de la Solución

En vista a las muchas dudas que resultan por falta de material, se elaboró una serie de prácticas que responden algunas de las interrogantes que surgen al trabajar con este tema poco desarrollado en la universidad.

Estas prácticas muestran al estudiante cómo comenzar, que temas debe revisar y cuales son los pre-requisitos necesarios para la realización de éstas.

En dichas prácticas se explica el uso correcto y el manejo de los dispositivos hapticos, algunas de éstas también brindan videos ilustrando de una forma más interactiva la instalación y la configuración de la librería usada para la creación de ambientes virtuales. Se muestra cómo utilizar los dispositivos de RV existentes en el laboratorio, los cuales se detallan en el siguiente capítulo.

Se dan pequeños ejemplos de aplicaciones para comprender mejor el uso de estos dispositivos, éstos podrían dar a los estudiantes una guía para realizar proyectos mucho más grandes a futuro.

1.4 Objetivos

En esta sección se plantean el alcance y los objetivos tanto general como específico que tiene el proyecto.

1.4.1 Objetivo General

La idea principal de este proyecto es ofrecer a los estudiantes una herramienta de aprendizaje para la utilización del hardware y software de realidad virtual y así introducirlos en esta área de aprendizaje.

1.4.2 Objetivos Específicos

- Enseñar al estudiante la instalación y el uso de la herramienta para entornos virtuales.
- Aprender el manejo y el cuidado de los dispositivos RV.
- Elaborar ambientes 3D que sirvan para la interacción con los dispositivos.
- Formular tareas a los estudiantes para que desarrollen habilidades en los temas tratados.

1.5 Estructura del documento

En el capítulo 2 se hace referencia tanto a los requerimientos funcionales como a los de hardware y software para la implementación de las prácticas.

El capítulo 3 analizará que herramientas se disponen para este problema. Se describen en detalle los dispositivos que están disponibles en el laboratorio y cuál será la herramienta que se utilizará para la generación de los ambientes.

En el capítulo 4 se describen el diseño del trabajo. Se exponen cuales son las secciones en las cuales se han dividió estas prácticas. Estas prácticas serán planteadas tomando en consideración cual es la solución que se va a mostrar. Algunos constarán de videos junto con la documentación escrita para su revisión. Otros solo constarán de la documentación escrita y el código fuente de la solución.

En el capítulo 5 se describen los detalles de la implementación, tales como la plataforma utilizada y los dispositivos de hardware.

Finalmente, se especifican las conclusiones y recomendaciones de este trabajo.

CAPÍTULO II

2 Análisis de Requerimientos

En esta sección se indican los requerimientos necesarios para la implementación de las prácticas.

2.1 Requisitos del Estudiante

- Haber cursado la materia de Gráficos por Computadora.
- Conocimiento básico de modelamiento de ambientes virtuales.
- Conocimientos de programación en lenguaje C.

2.2 Requerimientos de Software

- El software que se va a utilizar es el siguiente:
- Sistema Operativo XP
- Visual Studio .NET 2008
- Librerías OSG
- Modelador 3D (3D Studio Max, Maya, entre otros)

2.3 Requerimientos de Hardware

Los dispositivos que se van a utilizar en estas prácticas son:

5DT Data Glove Ultra

Polhemus Liberty Trackers

Proyector 3D y Gafas 3D Pasivas

5DT Head Mounted Display

Cada uno de estos dispositivos serán especificados en el siguiente capitulo.

Los requerimientos mínimos que se recomiendan, en cuanto a la PC, son los siguientes:

- Se recomienda un procesador Core 2 Duo 2.0GHz
- Para un buen desempeño de las aplicaciones se recomienda una memoria RAM 2 GB o superior.
- Se recomienda una tarjeta Gráfica AGP o PCI-Express.

2.4 Requerimientos Funcionales

Los requerimientos funcionales que se necesitan para las prácticas son las siguientes:

- Creación de videos donde se explican en detalle la instalación de la librería de OSG, la configuración de variables de entorno en el sistema y la configuración de las variables de OSG en Visual Studio.
- Generar un rectángulo con ayuda de la librería OSG.
- Modelar un objeto 3D para cargarlo en la escena mediante la librería OSG.
- Crear un rectángulo en 2D y añadirle textura con formato .jpg
- Crear esferas con la misma dimensión y añadir luz en diferentes posiciones para cada esfera.
- Utilizar la librería SDL en conjunto con OSG para reproducir un archivo de audio.
- Definir los pasos a seguir para la instalación y el uso del 5DT Data Glove.
- Crear un objeto que simule un guante 3D en OPENGL para la demostración de los gestos que son generados por el 5DT Data Glove.
- Crear un escenario 3D con 3 objetos para demostrar cómo elegir un objeto utilizando el mouse o el guante y hacer interacción con él como por ejemplo rotar, mover o escalar.
- Crear una escena más elaborada que contenga un objeto, el cuál será manipulado por el 5DT Data Glove, cada movimiento del objeto depende de un gesto.
- Mostrar la instalación, configuración y el uso adecuado del Liberty Tracker.

- Cargar un objeto 3D y moverlo dependiendo de los valores de posición del tracker.
- Crear un escenario 3D más elaborado para la interacción con los dispositivos, se utilizará el guante y el tracker para mover algún objeto en la escena o toda la escena. Los dispositivos para visualización en 3D también deberán ser incluidos.

CAPITULO III

3 Análisis de herramientas y dispositivos disponibles

En esta sección indicaremos la herramienta que vamos a utilizar y conocer un poco de cada dispositivo disponible en el laboratorio.

3.1 Herramientas para Entornos Virtuales

Se definen las herramientas utilizadas para la creación de entornos virtuales y su comparación.

OpenSceneGraph

Figura 3.1. Herramienta para Entornos Virtuales – OpenSceneGraph

Descripción

Imagen de un videojuego usando OpenSceneGraph

OpenSeneGraph (OSG) es un conjunto de herramientas gráfico de alto nivel y portable para el desarrollo de aplicaciones gráficas de alto rendimiento tales como simuladores de vuelo, juegos, realidad virtual visualización científica. Está orientado a objetos y construido a partir de la librería OpenGL, esto libera al desarrollador de implementar y optimizar llamadas gráficas de bajo nivel, y provee muchas utilidades adicionales para un rápido desarrollo de aplicaciones gráficas. **[1]**

Esta librería utiliza como lenguaje de programación C++, presenta independencia de la plataforma y además es un desarrollo de código abierto.

OpenSceneGraph emplea técnicas de grafos de escena para contener toda la información relativa a la escena generada. Un grafo de escena es una estructura de datos que permite crear una estructura jerárquica de la escena, de tal forma que se mantengan una serie de relaciones padre-hijo entre los distintos elementos. Por ejemplo, variaciones de posición y orientación en el nodo padre afectan a los nodos hijos.

Sus principales características son **[2]**:

- Uso de nodos de tipo (LOD). Gestionan la carga de polígonos dependiendo de la distancia de visualización de los objetos.

- Soporte para multiples formatos de archivos gracias al uso de *plug-ins* (mòdulos o programas anexados que aumentan las funcionalidades del programa principal) específicos. Existen *plug-ins* para archivos de tipo:

  LightWave (.lwo), VRML 1.0 (.wrl), Alias Wavefront (.obj), Carbon Graphics GEO (.geo), 3D Studio Max (3ds), Quake Character Models (.md2), Direct X (.x), Inventor Ascii 2.0 (.iv), Designer Workshop (.dw), AC3D (.ac).

- Soporte para los formatos de imagen:

  SGI Silicon Graphics (.rgb), Tagged Image Format (.tiff), Graphics Interchange Format (.gif), Independent JPEG Group (.jpg), Portable Network Graphics (.png), pic, bmp, DirectDraw Surface (.dds), Truevision Targa Files (.tga).

- Uso de *nodekits*. Éstos contienen los elementos necesarios para falcilitar el desarrollo de aplicaciones. Los principales tipos son los siguientes:

  OSGParticle. Para sistemas de partículas,

  OSGShadow. Para el manejo de sombras

  OSGManipulator. Proporciona contrala 3D interactivos

OSGTerrain. Facilita la renderización de terrenos.

OSGVolume. Renderización de volumen de alta calidad para aplicaciones médicas.

OSGAnimation. Para la creación y simulación de elementos del cuerpo humano.

OpenGL

OpenGL (Open Graphics Library) es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. La interfaz consiste en más de 250 funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas simples, tales como puntos, líneas y triángulos **[3].**

OpenGL tiene dos propósitos esenciales:

- Ocultar la complejidad de la interfaz con las diferentes tarjetas gráficas, presentando al programador una API única y uniforme.
- Ocultar las diferentes capacidades de las diversas plataformas hardware, requiriendo que todas las implementaciones soporten la funcionalidad completa de OpenGL (utilizando emulación software si fuese necesario).

OpenGL es una API basada en procedimientos de bajo nivel que requiere que el programador dicte los pasos exactos necesarios para renderizar una escena. Esto contrasta con las APIs descriptivas, donde un programador sólo debe describir la escena y puede dejar que la biblioteca controle los detalles para representarla.

OpenSceneGraph Vs OpenGL

OpenGL El diseño de bajo nivel de OpenGL requiere que los programadores conozcan en profundidad la pipeline gráfica, a cambio de darles libertad para implementar algoritmos gráficos novedosos. En cambio OSG es una librería de alto nivel o descriptiva que genera un código mas sencillo y comprensible.

La elección de OpenSceneGraph se ha basado fundamentalmente en su código abierto, su gratuidad, su independencia de la plataforma y, sobretodo, sus posibilidades de expansión. El principal inconveniente es la falta de documentación específica. Pero este problema es minimizado mediante una serie de ejemplos que aportan los conocimientos básicos de las distintas capacidades de la librería **[1].**

3.2 Dispositivos de Realidad Virtual

El hardware consiste de dispositivos físicos que forman parte de un sistema de RV y son los que estimulan al usuario en distintas maneras. Estos estímulos son los que le permiten alimentar los sentidos del usuario, y así, inducirlo a un mundo creado para él **[4]**.

Existen diferentes tipos de dispositivos como los de entrada, los cuales son dispositivos visuales, de sonido, la máquina de realidad (reality engine) y de salida como los dispositivos hápticos. Ambos tipos de dispositivos se describen brevemente a continuación.

Máquina de realidad virtual

La máquina de realidad virtual se refiere al hardware que nos permite generar modelos virtuales, este hardware puede ser desde una simple PC, hasta estaciones de trabajo diseñadas especialmente para tratar con este tipo de tareas. Estas máquinas se encargan de realizar todo el proceso de trazado de las imágenes **[4]**.

Dispositivos visuales

Los dispositivos visuales son una de las herramientas más importantes de retroalimentación para el usuario, en la mayoría de los casos es la entrada primaria que este recibe del sistema de RV.

Uno de las principales consideraciones en este tipo de dispositivos es el detalle de las imágenes contra la rapidez en la formación de las imágenes que forman las escenas, además de una visión monoscópica contra una estereoscópica. La formación de escenas en tiempo real le da un sentido de realidad al usuario al eliminar la discontinuidad **[4]**.

Dispositivos sonoros

La utilización de sonido provee un canal de comunicación muy importante dentro de los sistemas de realidad virtual puesto que el sistema auditivo es uno de nuestros componentes perceptuales más importantes, y ha sido demostrado que usar sonido para proporcionar información alternativa o suplementaria a un usuario de computadora puede grandemente aumentar la cantidad de información que ellos pueden ingerir. El sonido estéreo convencional fácilmente puede poner un sonido en cualquier lugar entre el lado izquierdo y el derecho. Sin embargo, con el sonido 3D, puede ser colocado en cualquier lugar, ya sea en el lado izquierdo, derecho, arriba, abajo, cerca o lejos.

Dispositivos Hápticos

Según Ivan Sutherland "el sentido humano kinestésico es como otro canal independiente al cerebro, un canal cuya información es asimilada de una manera bastante subconciente". Háptica es el estudio de cómo utilizar el sentido del tacto en un mundo generado por computadora. El estimular el sentido del tacto, como permitir al usuario "tocar" objetos de manera que pueda sentir la forma, textura, temperatura, firmeza y fuerza de éstos, puede agregar un buen nivel de realismo al ambiente virtual **[4]**.

Dispositivos a utilizar en el laboratorio

5DT Data Glove Ultra

Figura 3.4. 5DT Data Glove Ultra

Descripción:

El guante de datos 5DT ultra está diseñado para satisfacer los rigurosos requisitos de los modernos profesionales de la animación y captura del movimiento. Ofrece comodidad y facilidad de empleo. La alta calidad de los datos, la baja correlación cruzada y la alta tasa de transmisión de datos lo hacen ideal para la animación en tiempo real. Fabricado en licra se puede utilizar con manos de cualquier tamaño.

El guante de datos 5 5DT ultra mide la flexión del dedo (1 sensor por cada dedo) de la mano del usuario. Se comunica con el ordenador vía USB. Dispone de conexión con el puerto serie (RS 232 - independiente de la plataforma) disponible a través del kit de la interfaz ultra en serie del guante de datos 5DT. Ofrece resolución de 8-bit en la flexión, comodidad extrema, deriva baja y una arquitectura abierta.

Liberty Trackers

Figura 3.6 3D Tracking System

Descripción:

Liberty es el tracker electromagnético más rápido y más preciso disponible. Con soporte escalable para hasta 16 sensores, que es el precursor indiscutible en tecnología de seguimiento. State-of-the-art procesador digital de señal (DSP), la electrónica, una intuitiva interfaz de usuario gráfica (GUI) y un host de características técnicas más avanzadas.

Proyector 3D y Gafas 3D Pasivas

Figura 3.7 3D Proyector 3D

Figura 3.8 3D Gafas 3D Pasivas

Descripción:

Proyector 3D es el primer WXGA estereoscópico portátil, que ofrece una resolución de 1280x720. Esta impresionante capacidad de 120 cuadros por segundo es aproximadamente equivalente a la mayor difusión 2D estándar hasta la fecha: 1080 60p. Además, estos proyectores pueden fácilmente mostrar imágenes de alta definición.

Las gafas 3D pasivas utilizan un sistema y tecnología muy similar al que todos hemos escuchado o visto, es decir, las gafas típicas que tienen lentes anaglíficas. Las gafas anaglíficas usan dos lentes de colores diferentes (generalmente rojo y azul) para filtrar las imágenes de una película 3D.

5DT Head Mounted Display

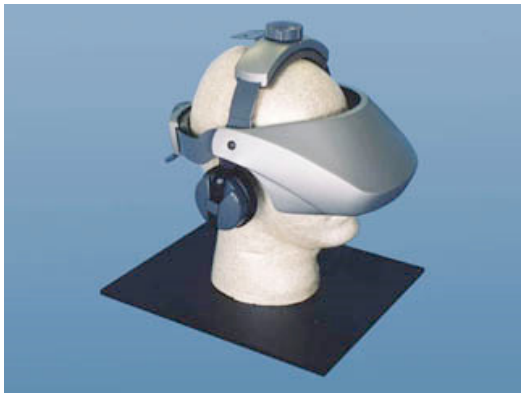


Figura 3.9 5DT Head Mounted Display

Descripción:

El 5DT mounted display (HMD) ofrece la calidad de usuario asequible, de alta resolución (SVGA), una imagen nítida y calidad de sonido superior en un diseño headmount elegante, cómodo y extremadamente ligero. El HMD proporciona para el usuario niveles de inmersión configurables. Otras características tienen trinquetes ajustables arriba / back , una base de montaje para los seguidores de cabeza y una tapa de modo para los controles de la realidad.

CAPÍTULO IV

4 Diseño

4.1 Descripción del diseño

Como ya hemos mencionado estas prácticas de realidad virtual están conformadas por tareas a seguir.

Las 12 prácticas han sido elaboradas de modo que el estudiante avance paulatinamente en su conocimiento. Además, las prácticas se han pensado de tal forma que tenga como objetivo elaborar una escena tridimensional lo más real posible y para ello deberá aprender a colocar textura al objeto, sonido, luces a la escena y luego la interacción con cada uno de los dispositivos para lograr la realidad.

Las prácticas constarán de la siguiente estructura:

Requerimientos de las Prácticas: son los elementos necesarios para lograr con éxito el laboratorio.

Procedimiento a seguir: Esta sección explicará paso a paso lo que necesita saber el estudiante para entender el taller ya sea funciones proporcionadas por las librerías, etc.

Recurso proporcionado: Estos son códigos documentados para que el estudiante tenga una guía para que pueda realizar la tarea asignada.

Tareas: Son ejercicios que se le van a asignar al estudiante para que adquiera las habilidades necesarias de la practica vista.

En el siguiente gráfico se describe la estructura de las prácticas

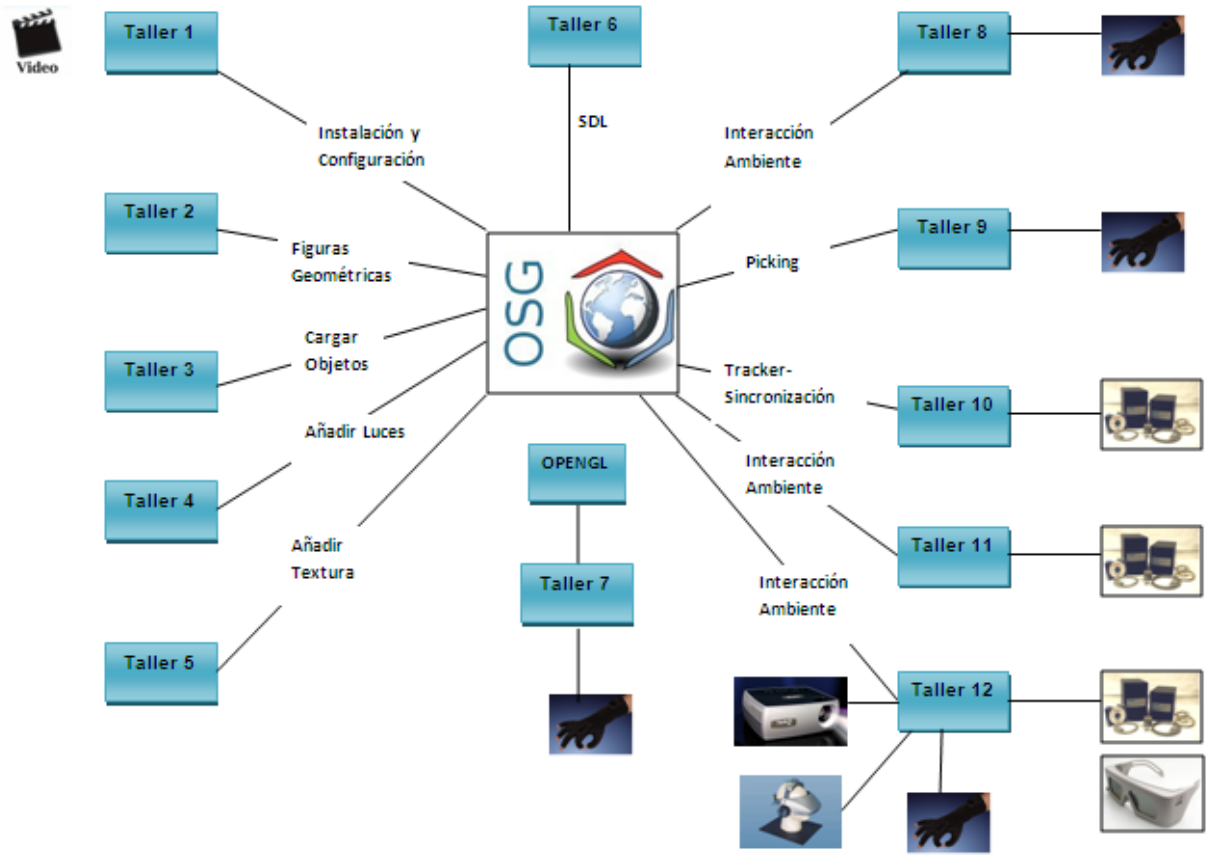


Figura 4.1 Estructura de las prácticas

4.2 Descripción de Módulos

A continuación se van a detallar cada una de las prácticas.

1. Osg - Instalación

Aquí se muestran los pasos que se deben seguir para poder instalar las librerías de OpenSceneGraph y configurar el sistema para que este las puedas reconocer.

También se proporciona un video con la explicación paso a paso de la instalación y configuración de OSG.

Al finalizar esta práctica el estudiante tendrá listo el ambiente para comenzar la programación con la librería OSG.

Ver Anexo1: Configuración de OpenSceneGraph

2. Osg - Figuras Geométricas

En este taller se enseña cómo crear figuras geométricas y generarlas para que sean mostradas en la ventana que OSG dispone.

Al concluir esta práctica el estudiante aprenderá a utilizar las funciones geométricas que OSG dispone tales como cono, cilindro y otras.

Ver Anexo2: Uso de funciones básicas en OpenSceneGraph.

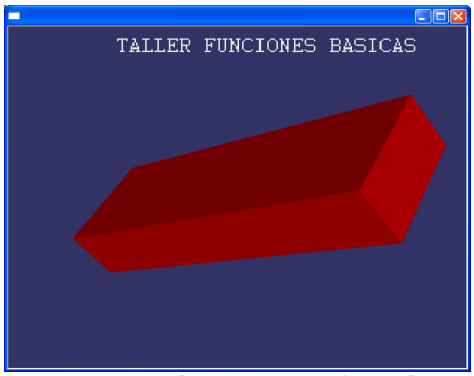


Figura 4.2.Práctica Figuras Geométricas

3. Osg-Cargar objetos

Ya conociendo las figuras básicas que se pueden generar en OSG ahora es momento de cargar un objeto 3D y que este pueda ser generado.

El estudiante aprenderá cómo cargar modelos de objetos, identificará que tipos de formatos OpenSceneGraph soporta al momento de cargar un objeto. Aprenderá a colocar los objetos en una escena usando transformaciones.

Ver Anexo3: Cargar Objetos 3D en OpenSceneGraph.

Figura 4.3.Práctica Cargar Objeto 3D

4. Osg-Añadiendo luces

En este taller aprenderá a crear luces con diferentes características que luego serán puestas en una escena.

Al tener claro de cómo crear luces, el estudiante no solo se limitará a poner una luz en una escena, sino las luces que se deseen dependiendo de las necesidades en el momento de aplicarlas.

Ver Anexo4: Añadir luces a una escena usando OpenSceneGraph.

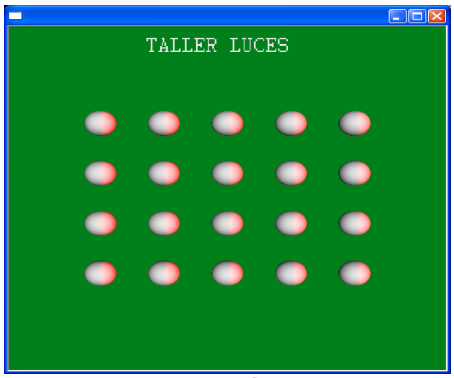


Figura 4.4.Práctica Luces

5. Osg-Añadiendo texturas

Añadir una textura a un ambiente virtual proporciona un mayor realismo. Este taller guía en los pasos para añadir una textura y observar que característica se dispone al momento de hacerlo. El estudiante pueda comprender el correcto uso de texturas y así poder desarrollar escenarios que luzcan vistosos y reales.

Ver Anexo5: Añadir textura a un Objeto usando OpenSceneGraph.

Figura 4.5.PrácticaTextura

6. Osg - SDL-Sonido

En un ambiente virtual no solo es necesario lo que se ve sino también lo que se escucha. En esta práctica se aprenderá a añadir los sonidos que se desee en un escenario.

El estudiante será capaz de combinar esta librería SDL con OSG para agregar audio en sus aplicaciones.

Ver Anexo6: Aplicando SDL

Figura 4.6.Práctica SDL

7. Guante – Configuración del 5DT Data Glove

Es hora de hacer uso de los dispositivos existentes y poder interactuar con ellos. Se guía al estudiante en la instalación y en el uso del 5DT Data Glove. Este guante está equipado con diferentes sensores que podrán detectar el flexionamiento de los dedos.

Al finalizar la tarea el estudiante podrá instalar el guante e interactuar con el ambiente.

Ver Anexo7: Sincronización con Dispositivos – Data Glove

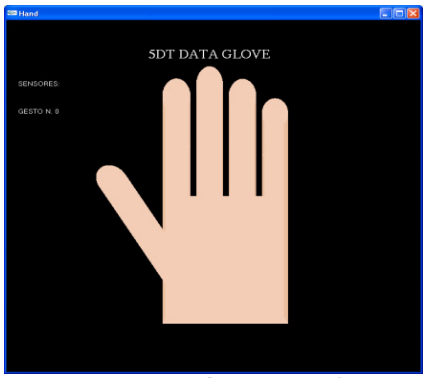


Figura 4.7Práctica Data Glove

8. Guante - Interacción con entorno virtual

En este taller el estudiante puede manipular el ambiente por medio de los gestos que genera el data glove.

Se dará cuenta cómo puede interpretar los gestos que el guante puede generar, para poder realizar cualquier tipo de interacción en un escenario cualquiera, por ejemplo podrá realizar movimientos de un objeto, escalarlo o mover toda la escena.

Ver Anexo8: Interacción con Dispositivos – Data Glove. Interactuando con un escenario 3D.

Figura 4.8Práctica Interacción Entorno - Data Glove

9. Guante - Picking

En esta practica el estudiante podrá seleccionar un objeto por medio de un gesto que se realice con el guante, además de poder rotarlo, escarlo o moverlo con diferentes gestos programas para estas acciones.

Al estudiar este taller se aprenderá como seleccionar un objeto y realizar ciertos cambios al mismo.

Ver Anexo9: Picking con el Data Glove.

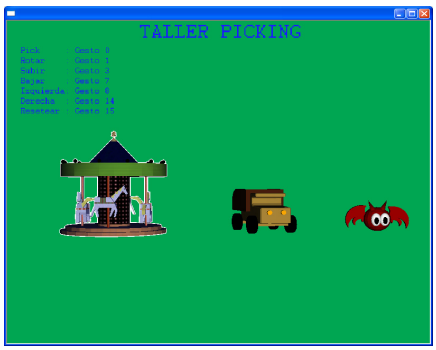


Figura 4.9 Práctica Picking

10. Tracker – Instalación del Tracker

En esta guía se explica otro de los dispositivos del laboratorio, los trackers, se muestra la configuración para que el sistema los reconozca y se pueda utilizar en las aplicaciones. También se ilustra el cuidado que se debe tener al momento de usarlo.

Al concluir este taller el estudiante podrá hacer uso del tracker y poder captar por medio de los sensores las posiciones y llevar estas a la escena.

Ver Anexo10: Sincronización con Dispositivos – Polhemus Tracker

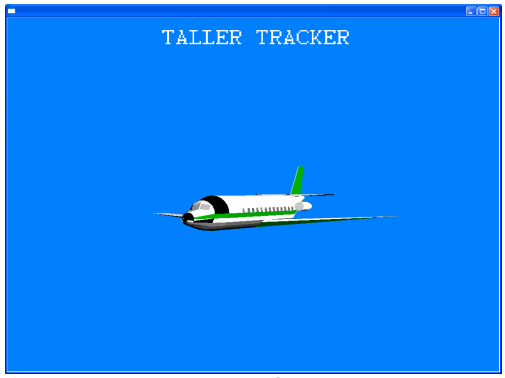


Figura 4.10 Práctica Tracker

11. Tracker - Interacción con entorno virtual

Una vez conociendo como trabajan los trackers, se los puede añadir a un escenario para su interacción con el mismo. Se crea un escenario con un objeto

y según las posiciones que se obtengan del sensor el objeto será movido en diferentes direcciones.

Ver Anexo11: Interacción con Dispositivos – Polhemus Tracker. Interactuando con un escenario 3D.

Figura 4.11 Práctica Interacción Entorno - Tracker

12. Dispositivos de RV - Interacción con entorno virtual

Es hora de trabajar con más de un dispositivo. Este taller muestra como añadir guantes y trackers a la vez e interactuar con un ambiente virtual. El tracker hace que toda la escena se mueva a la izquierda o derecha y dependiendo del gesto generado por el guante se moverá un objeto en la escena. También se utilizarán los dispositivos de visión tales como gafas, casco y monitor.

Al finalizar este taller el estudiante conocerá como combinar varios dispositivos en la escena.

Ver Anexo12: Interacción con Dispositivos – Polhemus Tracker y Guante.

Figura 4.12 Práctica Interacción Entorno - Data Glove y Tracker

CAPÍTULO V

5 Implementación

En este capítulo se describirán los detalles de la implementación tanto de hardware como de software así como las pruebas que se realizaron para determinar el mejor resultado del uso de los dispositivos.

5.1 Plataforma utilizada

La implementación de cada uno de los talleres fue realizado en el ambiente de desarrollo Microsoft Visual Studio 2008. Las interfaces utilizadas fueron OpenGL y OpenSceneGraph en el lenguaje C++. Aunque estas interfaces son multiplataforma, este trabajo fue realizado en Windows XP.

El modelado de los diseños fue realizado en 3D Studio Max 2009 y Google Sketchup Pro 7.

5.2 Dispositivos de Hardware

La implementación se llevo a cabo en las instalaciones del laboratorio de sistemas multimedia de la universidad, donde cuentan con computadores con las siguientes características:

Sistema:

- Microsoft Windows XP Profesional Versión 2002
- Service Pack 3

Equipo:

- Procesador Intel Xeon Quad-Core E5440 @ 2.83GHZ.
- 4GB de Memoria RAM
- Tarjeta de video NVIDIA Quadro FX 3700.

También se utilizaron PCs normales pero que tuvieran como mínimo 2GB de memoria RAM.

5.3 Detalles al usar los dispositivos RV

Para la utilización de los equipos se tomaron en cuenta ciertas consideraciones:

- Las librerías que cada dispositivo necesita para su utilización.
- Los cuidados necesarios al momento de su uso.

- Los datos que cada dispositivo genera, datos del 5DT Data Glove o del Sistema Tracker.

5DT Data Glove

Es necesario que al momento de correr una aplicación, el guante genere el Gesto 0, este es el gesto que se utiliza como inicialización.

Gesto 0

Al momento de utilizar el guante algunos gestos tornaban a confundirse, por motivo que algunos de éstos son un poco complicados de realizar por ello se recomienda hacer una prueba de todos los gestos, esto ayuda a determinar cuáles son más sencillos de realizar.

Liberty Trackers

La referencia utilizada en las aplicaciones fue Y positivo hacia la derecha y X positivos hacia arriba como se muestra en la figura en la siguiente figura.

Referencia Tracker

Los datos que se generan son muy sensibles a cualquier movimiento, por tal motivo en este trabajo se utilizó una expresión matemática (Escala de reducción) **[5]** utilizada para representar un plano menor que el real, si en nuestro caso movíamos el sensor del tracker 10 cm, en la escena solo se percibía una distancia de 0.1 cm, esto ayudó a que estos valores vayan de acorde con lo que se quería realizar en las diferentes aplicaciones.

CONCLUSIONES y RECOMENDACIONES

Una vez concluido este proyecto, tenemos como resultado un Conjunto De Prácticas para el Manejo de Dispositivos y Herramientas para Entornos Virtuales y podemos concluir lo siguiente:

1. Estas prácticas proveen un complemento para el aprendizaje de la tecnología de realidad virtual que esta incursionando en la universidad.

2. Al utilizar los dispositivos RV se observó que su manejo implica cierto cuidado y éste debe ser considerado muy importante para evitar que se dañen por el alto costo que estos representan.

3. Al obtener los datos del sensor del sistema Tracker, estos datos al principio no reflejaban posiciones adecuadas para visualizar el objeto en la escena virtual, para ello se considero realizar un escalamiento de datos que ayudó a resolver este problema.

4. Se pudo observar que al momento de querer generar un gesto con el guante, en ciertas ocasiones los gestos se confundían, esto se debía a que algunos gestos eran difíciles de realizar, por esto fue necesario escoger que gestos se podían generar de una manera más sencilla.

5. Al momento de realizar las prácticas se vio la necesidad de recordar el aprendizaje obtenido en materias importantes como son Gráfico por Computadora, Procesamiento de Audio y Video y Sistemas Multimedia.

Las recomendaciones que podemos valorar al finalizar este Proyecto son:

6. Puede ser mejorado en muchos aspectos tales como creación de nuevos ambientes virtuales, creando funciones más elaboradas, aumentando la complejidad y mejorando la interacción entre el hardware y las aplicaciones.

7. El estudiante podría proponer nuevas ideas útiles e innovadoras que mejoren en dimensión este conjunto de prácticas.

8. Se propone la creación de una aplicación que tenga la capacidad de calibrar los dispositivos sin necesidad de hacerlo en cada aplicación en la cual utilice estos dispositivos.

ANEXOS

PRÁCTICA Nº. 1

CONFIGURACIÓN DE OPENSCENEGRAPH

RESUMEN

Al finalizar esta práctica el estudiante tendrá listo el ambiente para comenzar la programación con la librería OSG.

OBJETIVOS:

- Configurar OSG dentro del sistema operativo Windows XP creando variables de entorno.
- Crear un proyecto en Visual Studio .NET para correr aplicaciones hechas en OSG.

PRE-REQUISITOS:

Tener instalado Visual Studio .NET Versión 9.0 con lenguaje de programación c++.

DESARROLLO PARA LA PRÁCTICA:

Instalación[1]

El primer paso para la instalación es descargar estos 3 archivos.

- openscenegraph-all-2.8.2-win32-x86-vc90-Debug
- openscenegraph-all-2.8.2-win32-x86-vc90-Release

Todos estos pasos se encuentra detallado en un video para una mayor comprensión del estudiante.

[1] **Ver Anexo:** instalacion_osg.avi

- OpenSceneGraph-Data-2.8.0

Los cuales dos de los mencionados los encontraras en la siguiente url:

http://www.openscenegraph.org/downloads/stable_releases/OpenSceneGraph-2.8/binaries/Windows/VisualStudio9/

El tercero archivo en la siguiente url:

http://www.openscenegraph.org/projects/osg/wiki/Downloads/SampleDatasets

Cabe mencionar que los archivos que estamos bajando son exclusivos para Microsoft Visual Studio 9.0. Si usted desea trabajar con otra versión de Visual Studio tendrá que bajarse los archivos específicos para esa versión. Usted podrá darse cuenta de esto observando el nombre de los archivos. Observe el nombre de los archivos que nos bajamos en el nombre dice VC90, la versión del visual que estamos usando.

Una vez que nos hemos bajado todos los archivos los descomprimimos.

Ahora creamos la ruta donde vamos a instalar OSG:

C:\Archivos de Programa\OpenSceneGraph-2.8.0

Dentro de esta ruta se ubicaran las siguientes carpetas: bin, doc, include, lib y share. Estas carpetas son las que se encuentran dentro de los archivos que se descomprimieron anteriormente.

Una vez que tenemos creada la ruta para nuestra instalación, el siguiente paso será copiar los archivos. Vamos donde tenemos descomprimidos los archivos y elegimos el siguiente openscenegraph-all-2.8.2-win32-x86-vc90-Debug y copiamos todas las carpetas que se encuentran dentro a la ruta de la instalación C:\Archivos de Programa\OpenSceneGraph-2.8.0\.

Hacemos lo mismo con el archivo openscenegraph-all-2.8.2-win32-x86-vc90-Release, cuando este nos pida reemplazar los archivos, elegimos **No reemplazar**.

Para el último archivo OpenSceneGraph-Data-2.8.0 tenemos que crear la carpeta "samples" en el directorio de nuestra instalación y copiamos la carpeta OpenSceneGraph-Data-2.8.0 que descomprimimos.

Configuración de las variables de entorno[2]

En estos momentos los archivos están instalados pero Windows no sabe cómo usar estos archivos, para ello tenemos que modificar las variables de entorno.

Para modificar las variables de entorno tenemos que ir Mi PC damos click derecho y elegimos Propiedades. En la ventana de propiedades del sistema, nos vamos al tab de Opciones avanzadas ahí elegimos el botón variables de entorno. Luego en Variables del Sistema damos click en el botón Nueva. Y comenzamos a crear las nuevas variables, dándoles un nombre y un valor. Un consejo cuando se crea una variable de entorno, es no dejar ningún espacio en blanco ni a la izquierda ni a la derecha de la línea. Así no tendrán ningún problema al momento de que las variables sean reconocidas por el sistema.

La primera variable va a ser el root, donde se encuentra instalado OSG. Si decidiste poner otra ruta para la instalación entonces tendrás que especificar esa ruta.

Nombre: OSG_ROOT **Valor**: C:\Archivos de programa\OpenSceneGraph-2.8.0

Nombre: OSG_BIN_PATH **Valor**: %OSG_ROOT%\bin

---

[2] **Ver Anexo:** Variables_Entorno.avi

Nombre: OSG_FILE_PATH **Valor**: %OSG_ROOT%\samples\OpenSceneGraph-Data-2.8.0

Nombre: OSG_INCLUDE_PATH **Valor**: %OSG_ROOT%\include

Nombre: OSG_LIB_PATH **Valor**: %OSG_ROOT%\lib

Nombre OSG_SAMPLES_PATH **Valor**: %OSG_ROOT%\share\OpenSceneGraph\bin

Y por ultimo modificamos la variable Path y añadimos al final lo siguiente: %OSG_BIN_PATH%;%OSG_SAMPLES_PATH%;

Una vez que hemos creados las nuevas variables de entorno tenemos que verificar si estas variables son reconocidas por el sistema

Para ello ejecutamos una ventana de consola y tipiamos echo %osg_root%, con este comando verificamos la ruta de la instalación.

Luego escribimos osgversiond, este nos mostrara la versión del OpenSceneGraph que estamos utilizando.

Si no ha existido ningún problema con los comandos anteriores, ahora mostraremos un ejemplo que viene incluido en la instalación. Para ello usaremos el comando osgviewerd y el objeto cow.osg. Si el objeto es mostrado, toda la instalación fue correcta.

Configuración de visual studio 9.0[3]

El siguiente paso para trabajar con OSG es configurar el Visual Studio 2008

[3] **Ver Anexo:** configuracion_visual_v9.0.avi

Para ello abrimos Visual Studio y creamos un Nuevo Proyecto. Nos vamos a Archivo->Nuevo->Proyecto, elegimos proyecto vacio, le damos un nombre y aceptamos.

Ahora añadimos un nuevo elemento. Damos click derecho en el nombre del proyecto y elegimos Agregar Nuevo Elemento, luego escogemos Archivo C++ con el nombre de main.

A continuación vamos a configurar el proyecto para que reconozca las librerías de openscenegraph. Damos click derecho en el nombre del proyecto, nos vamos a Propiedades. Ahora elegimos Propiedades de configuración->C++->General y en Directorios de inclusión adicionales añadimos lo siguiente: $(OSG_INCLUDE_PATH). Ahora en Preprocesador en Definiciones del preprocesador añadimos WIN32;_DEBUG;_WIN32. Luego nos vamos a Vincular-General y en Directorios de bibliotecas adicionales añadimos $(OSG_LIB_PATH). Y en Entrada->Dependencias Adicionales, añadimos las librerías que vamos a utilizar en nuestro proyecto, por ejemplo osgd.lib osgViewerd.lib osgDBd.lib. Nótese que todos los nombres de las librerías terminan con d, esto pasa porque estamos configurando Debug. En Release no debemos poner al final del nombre de las librerías la letra d.

Ya configuramos Debug, si queremos hacerlo en Release hacemos lo mismo y lo único que tenemos que cambiar es que en Preprocesador->Definiciones del preprocesador en vez de poner _DEBUG, tenemos que poner NDEBUG. Y como mencionamos anteriormente añadir las librerías sin poner al final del nombre la letra d.

Una vez que hemos configurado nuestro proyecto ya podemos correr un pequeño ejemplo para demostrar que las configuraciones fueron correctas.

Existen muchos ejemplos en la página de OSG, aquí la url para que te los puedas descargar.

http://www.openscenegraph.org/projects/osg/wiki/Downloads

TAREAS

1. Revisar la documentación proporcionada y seguir cada uno de los pasos que se muestran para la instalación y configuración de OSG.
2. Correr un ejemplo de la url antes mencionada, en Visual Studio .NET.

REFERENCIAS:

- Installing Openscenegraph 2.8.0

  http://dwightdesign.com/2009/05/installing-openscenegraph-280/

PRÁCTICA Nº. 2

USO DE FUNCIONES PARA GENERAR FIGURAS GEOMÉTRICAS EN OPENSCENEGRAPH

RESUMEN

En esta práctica se enseña al estudiante cómo crear figuras geométricas, modificar ciertas características tanto como color, posición, entre otros. Y generarlas para que sean mostradas en la ventana que OSG dispone.

OBJETIVO:

Relacionarse con las funciones que contiene OSG para la generación de figuras geométricas.

PRE-REQUISITOS

Cumplir con la Práctica 1.

DESARROLLO DE LA PRÁCTICA

Se da una breve descripción de los conceptos básicos de OSG.

Grafos de escena en OSG

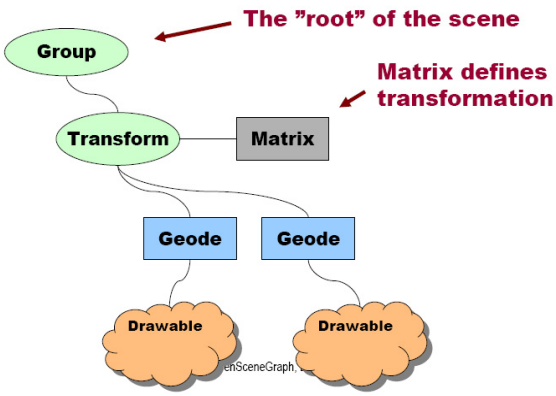
OSG emplea técnicas de grafos de escena para contener toda la información relativa a la escena generada. Un grafo de escena es una estructura de datos que permite crear una estructura jerárquica de la escena, de tal forma que se mantengan una serie de relaciones padre-hijo entre los distintos elementos. Por ejemplo, variaciones de posición y orientación en el nodo padre afectan a los nodos hijos. De esta forma se puede crear un brazo robot con varios eslabones

cada uno dependiente del anterior y con sólo aplicar un movimiento al eslabón inicial, el resto de los eslabones dependientes de él se moverán de acuerdo a la estructura definida.

Una visión general de algunas clases básicas OSG

Nodos en el árbol de la escena gráfica

- ”**Node**”, clase base
  - ”Group”, contiene un conjunto de nodos hijos
    - ”Transform”, transforma todos los nodos hijos por una matriz de 4x4
    - ”LightSource”, nodo hoja que define una luza en la escena
    - ”Switch”, cambia entre nodos hijos, por ejemplo. traffic lights
    - ”LOD”, nivel de detalle, cambia según la distancia al viewer
  - ”Geode”, nodo hoja para agrupar Drawables
    - ”Billboard”, orienta Drawables para mostrar siempre el viewer
- **Drawable**”, clase base abstracta para gráficos dibujables.
  - ”Geometry”, tiene vértices, normales, caras, coordenadas de textura,….
  - ”Text”, para elaboración de texto
  - ”DrawPixels”, encapsula dibujo de imágenes utilizando glDrawPixels
- ”**StateSet**”, encapsula los estados y atributos de OpenGL
- ”**Texture**”, encapsula la funcionalidad de textura de OpenGL



OSG viene con un número de figuras primitivas tales como Box, Sphere, Cone, Cylinder, Capsule. Más algunas figuras especiales, por ejemplo: InfinitePlane

Descripción de Clases Esenciales

Esta clase es en realidad una de las más importantes en las clases OSG porque aquí es donde los usuarios deben organizar el grafo de la escena.

- La clase **ShapDrawable** es una clase que deriva de osg::Drawable. Su propósito es permitir la creación de figuras comunes predefinidas. Una vez creadas, estas figuras pueden ser añadidas a la escena grafica usando osg::Geode.
- La clase **Geode** se deriva de la clase osg::Node y está diseñado para contener osg::Drawable y sus derivados. De hecho, Geode representa Geometry-Node. Tenga en cuenta que la geometría de la escena (datos de la escena) no puede ser añadida directamente a la escena gráfica. Es decir, no puede añadir osg::Geometry a osg::SceneView. En su lugar, hay que añadir la geometría a uno de los contenedores intermedios tales como osg::Geode.

- La clase **Viewer** maneja múltiples sincronización de cámaras para renderizar una simple vista que abarca varios monitores. Viewer crear su propia ventana (s) y contexto gráfico (s) basado en las capacidades de su sistema de gráficos, por lo que una sola aplicación basada en el Viewer se ejecutan en uno o varios sistemas de visualización.

Posición de Objetos en la escena

Existen principalmente dos formas de especificar la posición de los objetos en OSG:

- Usando la clase **PositionAttitudeTransform**. Este hace uso de un vector 3D. Se pueden aplicar transformaciones (traslación, escalamiento y rotación) a los objetos.
- Utilizando un objeto **MatrixTransform**. A diferencia de la clase anterior, esta utiliza una Matriz para aplicar la transformación a sus objetos hijos.

Este es el algoritmo:

- Crear un objeto PositionAttitudeTransform (resp. MatrixTransform).

```
osg::PositionAttitudeTransform *pat = new osg::PositionAttitudeTransform;
```

- Colóquelo en el nodo raíz de la escena.

```
root->addChild(pat);
```

- Añada su objeto a PositionAttitudeTransform (resp. MatrixTransform).

```
pat->addChild(objeto);
```

- Cualquier transformación que se especifique a este nodo será aplicado a su objeto.

```
pat->setScale(osg::Vec3d(0.2,0.2,0.2));
```

Creando formas geométricas básicas en OSG

- Cree un proyecto en Visual Studio y configúrelo para hacer uso de las librerías de OSG tal como se muestra en la práctica 1.
- Añadir los siguientes archivos de cabecera, estos archivos contienen las funciones geométricas.

```
#include <osg/Geode>

#include <osg/ShapeDrawable>
```

- Añade el siguiente código en el main de la aplicación.

```
int main( int argc, char **argv )
{
  //Creando el Viewer
  osgViewer::Viewer* viewer=new osgViewer::Viewer();
  //Creando el nodo root
  osg::ref_ptr<osg::Group> root = new osg::Group;

  //Código para la creación de un cilindro
  //El objeto Geode contiene la figura
  osg::ref_ptr<osg::Geode> cylGeode = new osg::Geode;
  //Objeto Shape Drawable
  osg::ShapeDrawable *cylShapeDrawable = new
osg::ShapeDrawable(new
osg::Cylinder(osg::Vec3(0.0f,0.0f,0.0),1,4));
  cylShapeDrawable->setColor(osg::Vec4f(1.0f, 0.0f, 1.0f,
1.0f));// magenta
  cylGeode->addDrawable(cylShapeDrawable);
  root->addChild(cylGeode);

  viewer->setUpViewInWindow(500,100,600,500);
  viewer->setSceneData(root);
  viewer->run();
}
```

- Finalmente genera el proyecto y visualice los resultados.

TAREAS

1. Crear nuevas figuras geométricas (box, sphere, cone, capsule) en base al código que fue proporcionado. Cada figura tendrá diferentes colores.

Box: rojo
Sphere: amarilla
Cone: verde

Capsule: cyan

2. Tratar que las posiciones de las figuras anteriores generadas queden tal como se muestra en el siguiente gráfico.

3. Usted puede haber notado que la mayoría de los objetos fueron creados con punteros inteligentes (ref ptr). Sin embargo, el objeto de visualización se ha creado sin un puntero inteligente conectado a éste, ¿por qué cree que el objeto Viewer no debe ser creado con punteros inteligentes?

REFERENCIAS

- Openscenegraph – Examples
  http://www.openscenegraph.org/projects/osg/wiki/Support/UserGuides/Examples
- Openscenegraph – Tutorials
  http://www.openscenegraph.org/projects/osg/wiki/Support/Tutorials
- Loading Models from Files and Positioning Them in a Scene
  http://www.openscenegraph.org/projects/osg/wiki/Support/Tutorials/FileLoadingAndTransforms
- Motor gráfico interactivo para la visualización en tiempo real de gran volumen de información vectorial y texturizada
  http://www.ingegraf.es/pdf/titulos/COMUNICACIONES%20ACEPTADAS/RV2.pdf

PRÁCTICA Nº. 3

CARGAR OBJETOS 3D EN OPENSCENEGRAPH

RESUMEN

En esta práctica el estudiante aprenderá a cargar modelos de objetos 3D, identificará que tipos de formatos OSG soporta. Aprenderá a colocar los objetos en una escena usando transformaciones.

OBJETIVO:

- Aprender cómo cargar modelos de objetos.
- Identificar que tipos de formatos OSG soporta.
- Colocar los objetos 3D en una escena usando transformaciones.

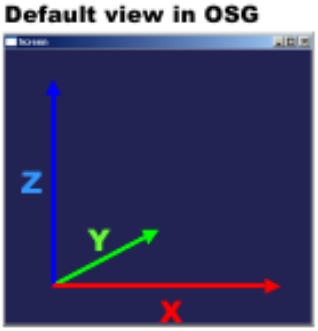
PRE-REQUISITOS

Cumplir con la práctica 2 para que el estudiante tenga el conocimiento de cómo utilizar la librería de OSG.

DESARROLLO DE LA PRÁCTICA

Sistema de coordenadas en OSG

Lo primero que se debe tener en cuenta es el sistema de coordenadas que OSG usa. Como podemos observar en la figura OSG por defecto utiliza el sentido positivo del eje Z apuntando hacia arriba, el sentido positivo del eje X apunta hacia la derecha y el sentido positivo del eje Y apunta hacia el interior de la pantalla.



Cargar un objeto en OSG

En la versión actual de OSG, usted será capaz de cargar cualquier tipo de formato de archivo para el que se dispone de un plugin. Están incluidos los siguientes archivos con formatos: 3dc, 3ds, flt, geo, iv, ive, lwo, md2, obj, osg. Y los siguientes formatos de archivos de imágenes: bmp, gif, jpeg, rgb, tga, tif.

1. Modelos geométricos son representados como nodos en la escena gráfica. Para cargar o manipular un archivo geométrico necesitaremos declarar punteros a un tipo de instancia osg::Node (algunos archivos de cabecera son necesarios).

```
#include <osg/Node>
#include <osgDB/ReadFile>
```

```
osg::Node * objetoNodo = NULL;
objetoNodo = osgDB::readNodeFile( "imagenes/objeto.osg" );
```

ó

```
objeto = osgDB::readNodeFile( "imagenes/objeto.3ds" );
```

Eso es todo lo requerido para cargar una base de datos.

2. Siguiente, debemos declarar un nodo que servirá como el nodo raíz para el escenario gráfico. Dado que vamos a añadir “childrens” de estos nodos que tenemos que hacer una instancia “group”. La clase “Node” representa la más genérica versión de nodos. Estos incluyen nodos que no tienen hijos (nodos hojas). La clase “group” es una versión especializada de clases de nodo. Este añade funciones asociadas con la adición y la manipulación de los hijos.

```
osg::Group* root = new osg::Group();
root->addChild(objetoNodo);
```

3. Declarar la transformación, inicializada con valores defaults

```
osg::PositionAttitudeTransform* objetoXform = new
osg::PositionAttitudeTransform();
```

Declare e inicialice una instancia de Vec3 para cambiar la posición del objeto en la escena.

```
osg::Vec3 objetoPosit(5,0,0);
```

```
objetoXform ->setPosition(objetoPosit);
```

Use el método “addChild” de la clase osg::Group para añadir la transformada como un hijo del nodo root y el nodo objetoNodo como un hijo de la transformación.

```
root->addChild(objetoXform);
objetoXform ->addChild(objetoNodo);
```

4. Ahora tenemos un escenario gráfico compuesto de un nodo raíz con dos hijos, uno es un modelo geométrico de un objeto cualquiera. El otro hijo es un sub-árbol que consta de un nodo transformación con un modelo geométrico de un objeto como su único hijo. Para ver la escena, tendremos que establecer un Viewer. He aquí como crearemos un Viewer.

```
osgViewer::Viewer* viewer=new osgViewer::Viewer();
```

Luego se necesitara asignar a la escena gráfica que nosotros hemos creado a este viewer:

```
viewer->setSceneData( root );
```

Podemos también setear el tamaño de la ventana que queremos mostrar en nuestra aplicación.

```
viewer->setUpViewInWindow(100,100,1000,600);
```

5. Y por último mandaremos a ejecutar nuestro viewer

```
viewer->run();
```

TAREAS

1. Cree un proyecto en Visual Studio .NET versión 9.0 y configúrelo para hacer uso de las librerías de OSG.
2. Crear un objeto 3D o bajarlo de Internet, el objeto debe ser con extensión obj ó 3ds.
3. Cargar el objeto anterior creado en la aplicación y visualizar el resultado.

4. Añadir dos objetos más a la escena, estos deben ser añadidos por separados y cargados como dos nodos diferentes.
5. Escoger uno de los objetos y rotarlo con respecto al eje Z.
6. Cambiar el tamaño del otro objeto que no fue seleccionado en el punto anterior.

REFERENCIAS

- Openscenegraph – Examples http://www.openscenegraph.org/projects/osg/wiki/Support/UserGuides/Examples
- Openscenegraph – Tutorials http://www.openscenegraph.org/projects/osg/wiki/Support/Tutorials
- Loading Models from Files and Positioning Them in a Scene http://www.openscenegraph.org/projects/osg/wiki/Support/Tutorials/FileLoadingAndTransforms
- Motor gráfico interactivo para la visualización en tiempo real de gran volumen de información vectorial y texturizada http://www.ingegraf.es/pdf/titulos/COMUNICACIONES%20ACEPTADAS/RV2.pdf.

PRÁCTICA Nº. 4

AÑADIR LUCES A UNA ESCENA USANDO OPENSCENEGRAPH

RESUMEN

En esta práctica se aprenderá como añadir luces a una escena, que puede estar conformada por uno o varios objetos. Se aplican varios tipos de luces y colores así como también las ubicaciones de cada una.

OBJETIVO

- Aprender a crear luces en una escena.
- Cambiar las características de la luz, tanto color como posición.

PRE-REQUISITOS

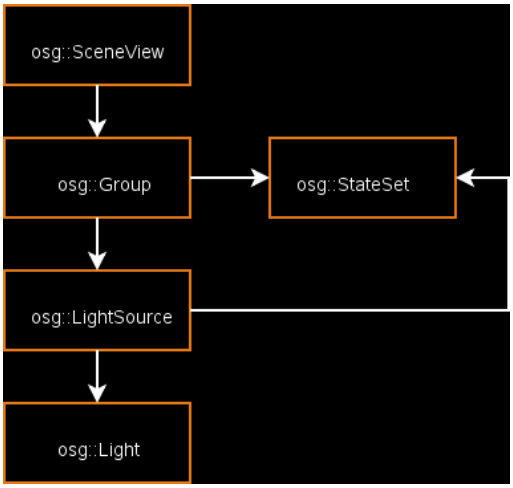
Haber revisado las práctica 2 y 3, para que el estudiante tenga el conocimiento de la creación de figuras geométricas y además como cargar un objeto 3D en la escena.

DESARROLLO DE LA PRÁCTICA

Para añadir luz en una escena se necesita de 2 clases osg::LightSource y osg::Light. Para añadir osg::Light a la escena, primero debe ser añadido a osg::LightSource. A continuación, cada osg::LightSource debe ser añadido a osg::Group que es directamente adjuntado a osg::SceneView. El osg::StateSet almacena una colección de valores de estado (llamado modos y atributos) en la

clase StateSet. Cualquier osg::Node en el grafo de la escena puede tener un StateSet asociado a ella.

StateSets contienen listas de atributo / valor de OpenGL. Estos StateSets se pueden asociar a los nodos de la escena gráfica. StateSets se acumulan desde la raíz a los nodos hoja. Atributos de State que no se cambian en un nodo son simplemente heredados desde arriba.



Iluminación

1. Cree un proyecto en Visual Studio .NET versión 9.0 y configúrelo para hacer uso de las librerías de OSG.
2. Añadir los siguientes archivos de cabecera.

```
#include <osgViewer/Viewer>
#include <osg/ShapeDrawable>
#include <osg/StateSet>
#include <osg/Light>
#include <osg/LightSource>
```

3. Crear una figura geométrica básica y añadirla al nodo raíz de la aplicación.

4. Crear el nodo raíz.

```
osg::Group * root = new osg::Group;
```

5. Crear un objeto LightSource.

```
osg::LightSource* pLightSource = new osg::LightSource;
```

6. Crear un objeto Light.

```
osg::Light * plight = new osg::Light();
```

7. Establecer las propiedades de los objetos de luz. Esto incluyen posición de la luz, el color de la luz, tipo de luz y otros.

```
pLight->setAmbient(osg::Vec4d(0.0, 0.0, 0.0, 0.0) );
pLight->setDiffuse(osg::Vec4(1.0f,   0.0f,   0.0f,   1.0f));
//Consta de los componentes RGB y el valor alpha que
siempre debe ser 1.0
pLight->setPosition(osg::Vec4(1,0,0,w));    //w    =    0.0
direccion de la luz //w = 1.0 luz puntual (posicion)
```

8. Setear a LightSource el nodo luz

```
pLightSource->setLight( pLight );
```

9. Agregar LightSource al nodo raíz de la escena.

```
root->addChild( pLightSource );
```

10. Creando el Viewer

```
osgViewer::Viewer* viewer=new osgViewer::Viewer();
viewer->setUpViewInWindow(500,100,600,500);
viewer->setSceneData(root);
viewer->run();
```

Estado de la Iluminación

Para obtener los efectos de iluminación en OSG, se debe de habilitar la iluminación y al menos una fuente de luz. Osgviewer hace esto por defecto mediante el establecimiento de las formas apropiadas de un nodo raíz StateSet.

El siguiente fragmento de código permite la iluminación con dos fuentes de luz (GL_LIGHT0 y GL_LIGHT1) en StateSet.

```
osg::StateSet* state = root->getOrCreateStateSet();
state->setMode( GL_LIGHTING, osg::StateAttribute::ON );
state->setMode( GL_LIGHT0, osg::StateAttribute::ON );
state->setMode( GL_LIGHT1, osg::StateAttribute::ON );
```

TAREA

- Crear tres objetos (estos pueden ser los mismos) y aplicarle a cada uno una luz de diferente color (roja, verde y azul), tal como se muestra a continuación.

REFERENCIAS

- Tutorial 9: Lighting in OSG by Franclin Foping – Mayo 11, 2008
- Source Lighting: http://www.cs.clemson.edu/~malloy/courses/3dgames-2007/tutor/web/light/light.html
- Source Lighting: http://www.osghelp.com/readarticle.php?article_id=20
- http://www.lulu.com/items/volume_51/767000/767629/3/print/OSGQSG.pdf

PRÁCTICA Nº. 5

AÑADIR TEXTURA A UN OBJETO USANDO OPENSCENEGRAPH

RESUMEN

Esta práctica guía los pasos para añadir una textura a un objeto en la escena y así poder desarrollar escenarios que luzcan reales.

OBJETIVO

- Añadir una textura a un objeto en la escena.
- Conocer los tipos de formato de la textura que OSG reconoce.

PRE-REQUISITOS

Revisar la tarea de la práctica 2 para que el estudiante tenga el conocimiento de la creación de figuras geométricas.

DESARROLLO DE LA PRÁCTICA

Mapeando Textura

Las texturas dan al escenario un mayor realismo, por eso es muy importante saber cómo utilizarlas.

Los siguientes formatos de archivos de imágenes son los permitidos: bmp, gif, jpeg, rgb, tga, tif.

1. Cree un proyecto en Visual Studio .NET versión 9.0 y configúrelo para hacer uso de las librerías de OSG.

2. Añadir los siguientes archivos de cabecera.

```
#include <osg/Geometry>
#include <osgDB/ReadFile>
```

3. Crear una figura primitiva (un cuadrado) al cual se le va a añadir la textura. Siga el siguiente código.

// Crear un objeto Geometry

```
 osg::Geometry* pGeo = new osg::Geometry;
```

//añadir 4 vértices para crear un cuadrado

```
 osg::Vec3Array* pVerts = new osg::Vec3Array;
 pVerts->push_back( osg::Vec3( 0, 0, 0 ) );
 pVerts->push_back( osg::Vec3( 1, 0, 0 ) );
 pVerts->push_back( osg::Vec3( 1, 0, 1 ) );
 pVerts->push_back( osg::Vec3( 0, 0, 1 ) );
 pGeo->setVertexArray( pVerts );
```

// Crear un objeto PrimitiveSet

```
 osg::DrawElementsUInt* pPrimitiveSet = new
osg::DrawElementsUInt( osg::PrimitiveSet::QUADS, 0 );
 pPrimitiveSet->push_back( 3 );
 pPrimitiveSet->push_back( 2 );
 pPrimitiveSet->push_back( 1 );
 pPrimitiveSet->push_back( 0 );
 pGeo->addPrimitiveSet( pPrimitiveSet );
```

// Crear un arreglo para las coordenadas de las texturas

```
 osg::Vec2Array* pTexCoords = new osg::Vec2Array( 4 );
 (*pTexCoords)[0].set( 0.0f, 0.0f );
 (*pTexCoords)[1].set( 1.0f, 0.0f );
 (*pTexCoords)[2].set( 1.0f, 1.0f );
 (*pTexCoords)[3].set( 0.0f, 1.0f );
 pGeo->setTexCoordArray( 0, pTexCoords );
```

4. Cargar una textura, para agregarla vamos a declarar una instancia de la textura y establecer su variación en los datos como "dinámica". (Si no se declara la textura como dinámico, algunas de las rutinas de optimización de la OSG podrían quitarla.). El código siguiente muestra cómo leer una imagen de un archivo y asociar ésta con una textura.

```
osg::Texture2D* texture = new osg::Texture2D;
texture->setDataVariance(osg::Object::DYNAMIC);
//cargar una imagen de una archivo:
osg::Image* img = osgDB::readImageFile("imagen.jpg");
texture->setImage(img);
pStateSet->setTextureAttributeAndModes( 0, texture,
osg::StateAttribute::ON );
```

Las texturas pueden ser asociadas con StateSets.

5. El siguiente paso es crear un StateSet,

```
// Crear un nuevo StateSet con configuraciones default:
 osg::Geode* pGeode = new osg::Geode;
 osg::StateSet* pStateSet = pGeode->getOrCreateStateSet();
 pStateSet->setMode( GL_LIGHTING, osg::StateAttribute::OFF );
 pGeode->addDrawable( pGeo );
```

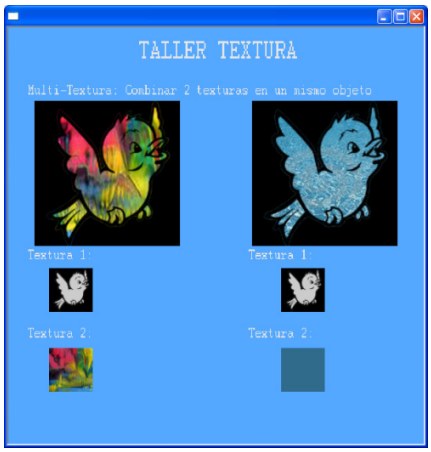
6. Finalmente añadir al grupo de la escena,

```
root->addChild(pGeode);
```

TAREAS

- Crear otro proyecto siguiendo los principios del anterior
- Conseguir dos texturas, una de las dos debe ser una imagen alpha (blanco y negro).

- Mezclar las dos texturas con el objetivo de que el área blanca de la textura alpha se combine con la otra textura, para conseguir el efecto mostrado en la siguiente imagen.



REFERENCIAS

- Creating textured geometry using statesets http://www.openscenegraph.org/projects/osg/wiki/Support/Tutorials/Textures
- Código texturas: http://www.osghelp.com/readarticle.php?article_id=26
- Código texturas: http://www.osghelp.com/readarticle.php?article_id=5
- http://www.lulu.com/items/volume_51/767000/767629/3/print/OSGQSG.pdf

PRÁCTICA Nº. 6

APLICANDO SDL

RESUMEN

En esta práctica, se describen las técnicas para incorporar audio en la escena. Esta técnica explota las capacidades de audio de SDL a través de una biblioteca de extensión, SDL_mixer.

OBJETIVO

- Conocer la librería SDL y las posibilidades que nos ofrece.
- Conocer que formatos de audio SDL reconoce.

PRE-REQUISITOS

Tener instalado Visual Studio .NET Versión 9.0 con lenguaje de programación c++ e incluir las librerías de OSG.

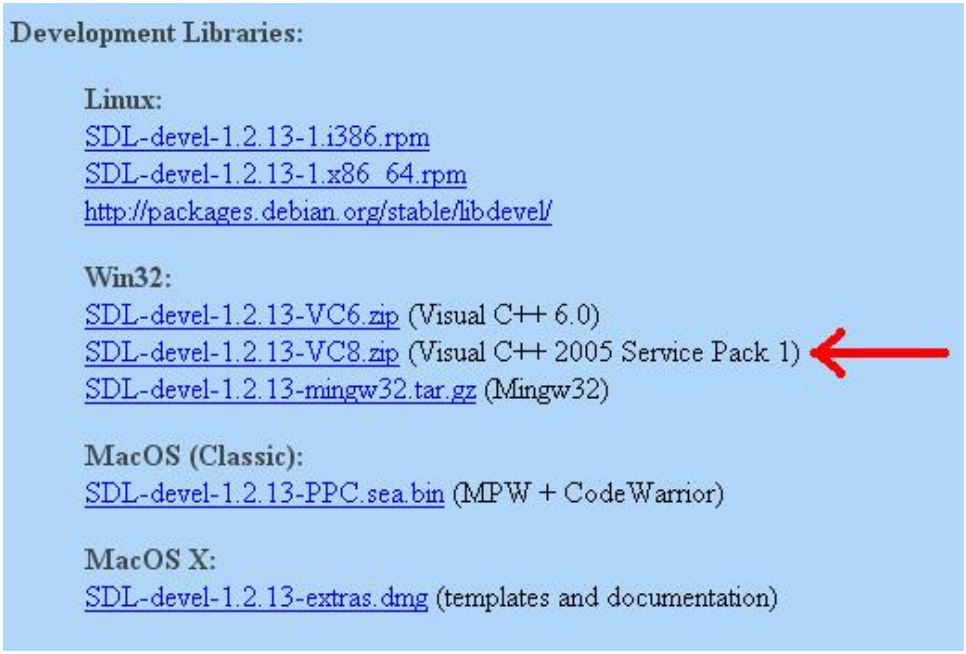
DESARROLLO DE LA PRÁCTICA

Instalación y configuración de la librería SDL

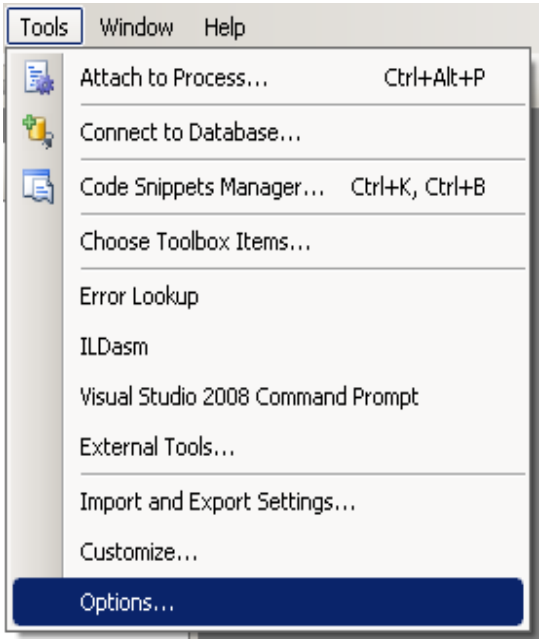
1. Descargar las cabeceras de SDL y los binarios, los cuáles los encontrará en la página web de SDL, específicamente en el siguiente link:

   http://www.libsdl.org/download-1.2.php

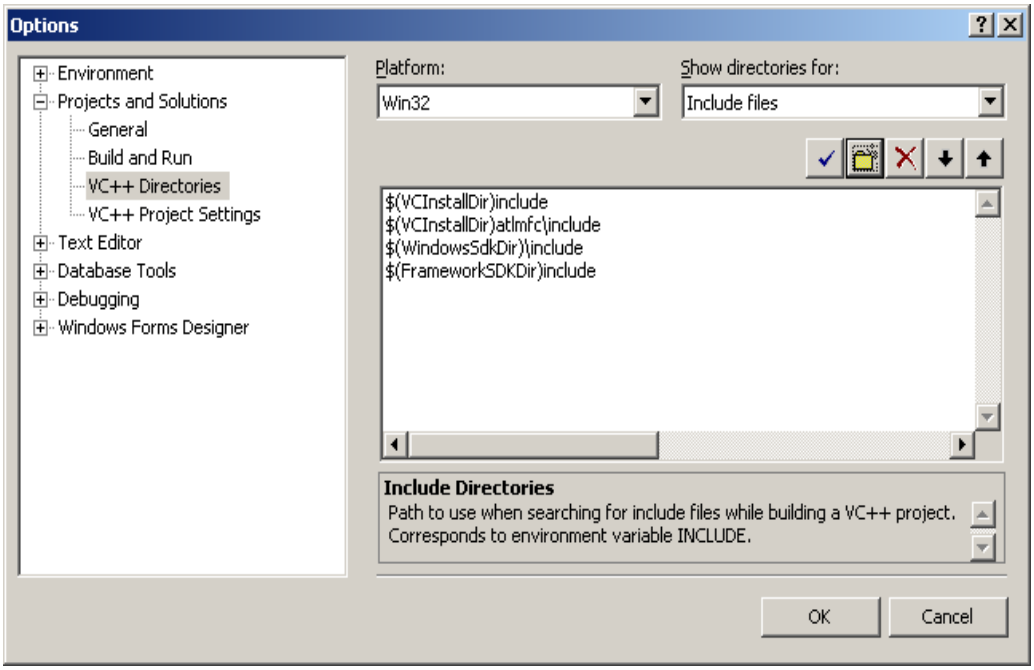
   Vaya a la sección Bibliotecas de desarrollo y descargue la librería de desarrollo para Windows



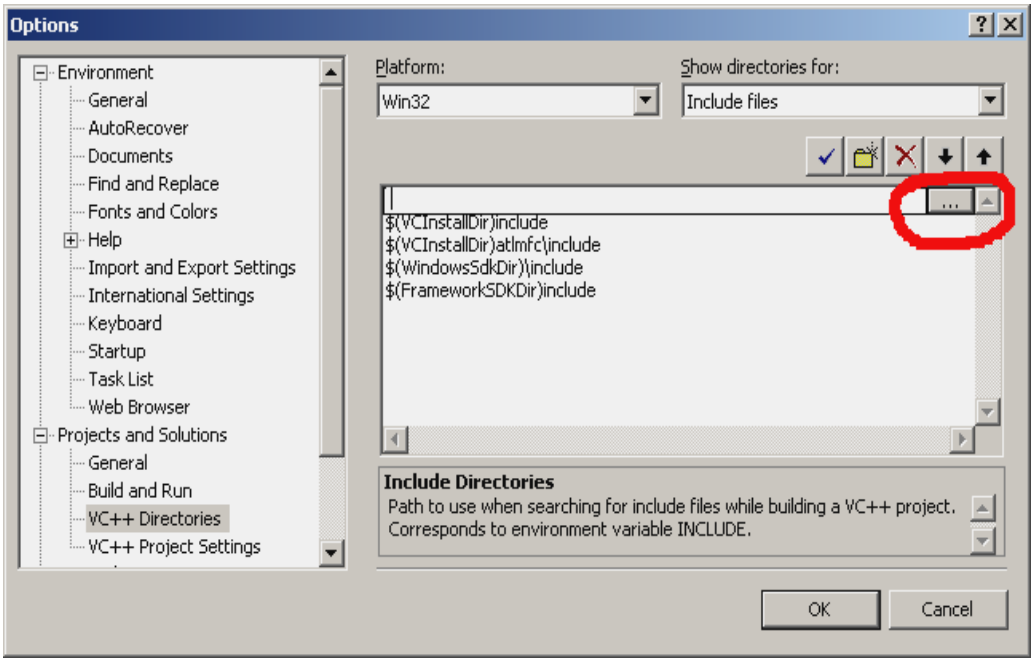
2. Descomprima el .zip y copie esta carpeta en C: \.
3. Abrir el programa Visual Studio .NET y vaya a Herramientas -> Opciones:

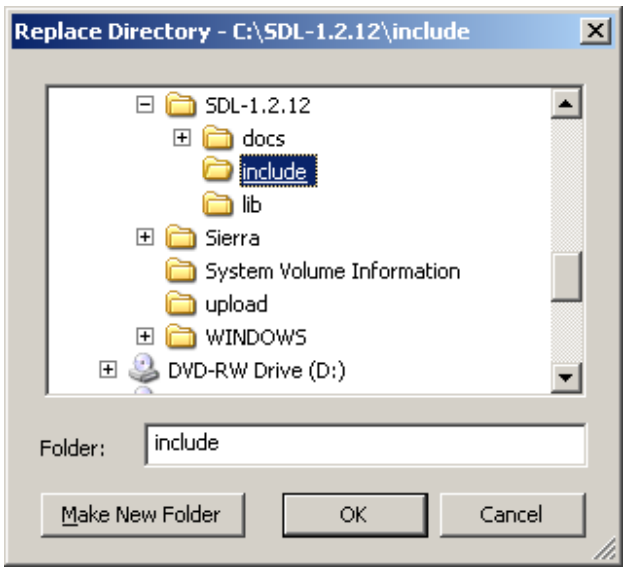


4. Ir luego a "Directorios de VC + +" en "Projects and Solutions". Elegir en "Show Directories for:" la opción "Include files". Haga clic en el icono de carpeta.

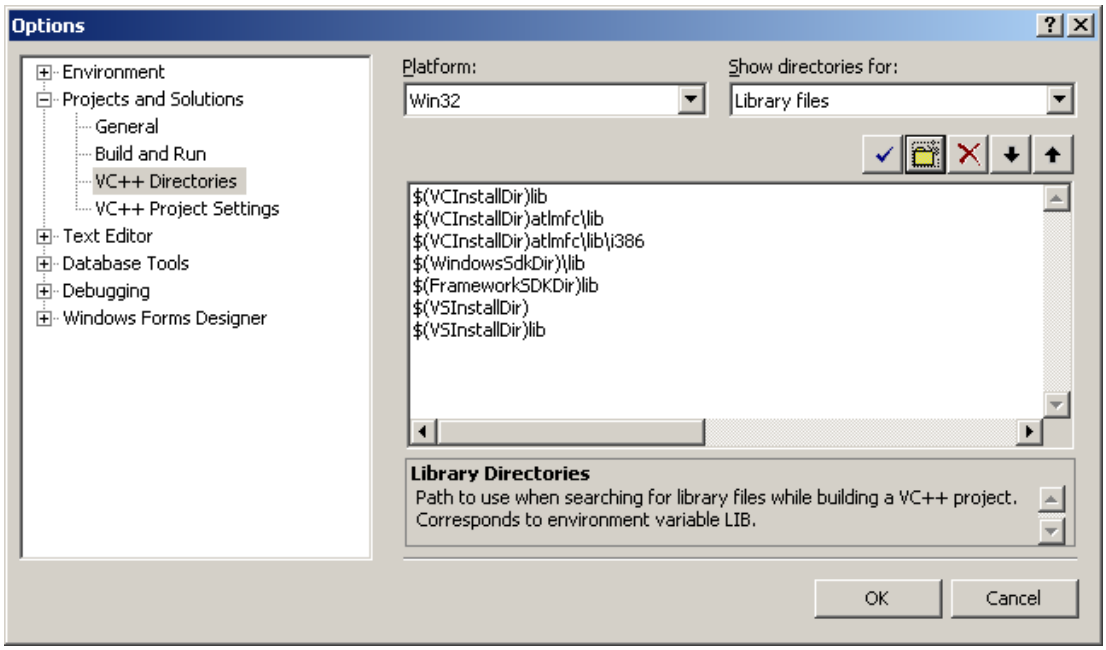


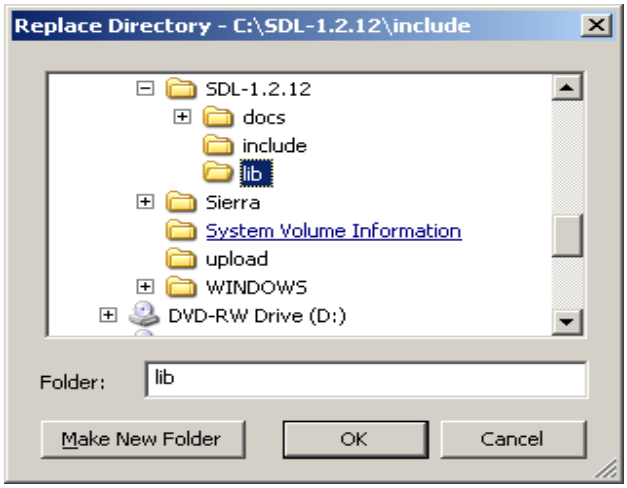
5. Agregue el directorio include de la carpeta que extrajo SDL.





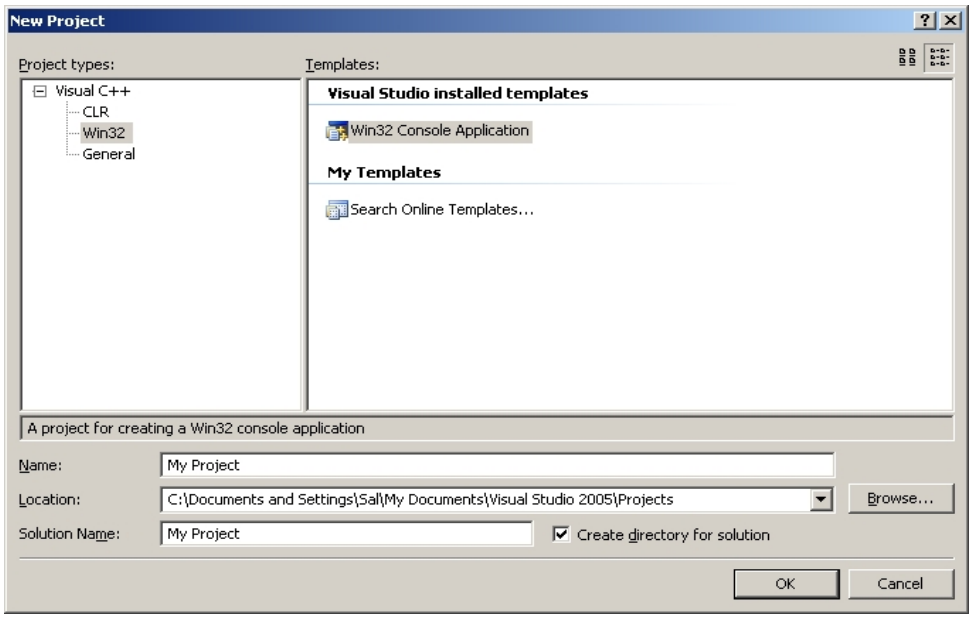
6. Elegir en "Show Directories For:" la opción "Library files", haga clic en el ícono de carpeta y agregue el directorio lib de la carpeta que extrajo SDL.





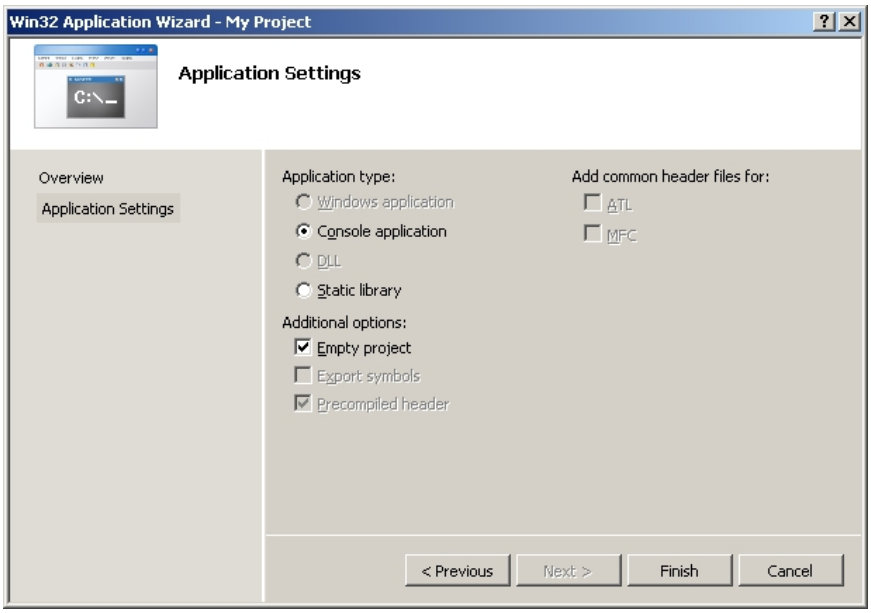
Ahora tome la sdl.dll (ésta debe estar dentro de la subcarpeta lib) y póngalo en el mismo directorio donde se genera el ejecutable.

7. Ahora cree un nuevo proyecto de consola Win32:



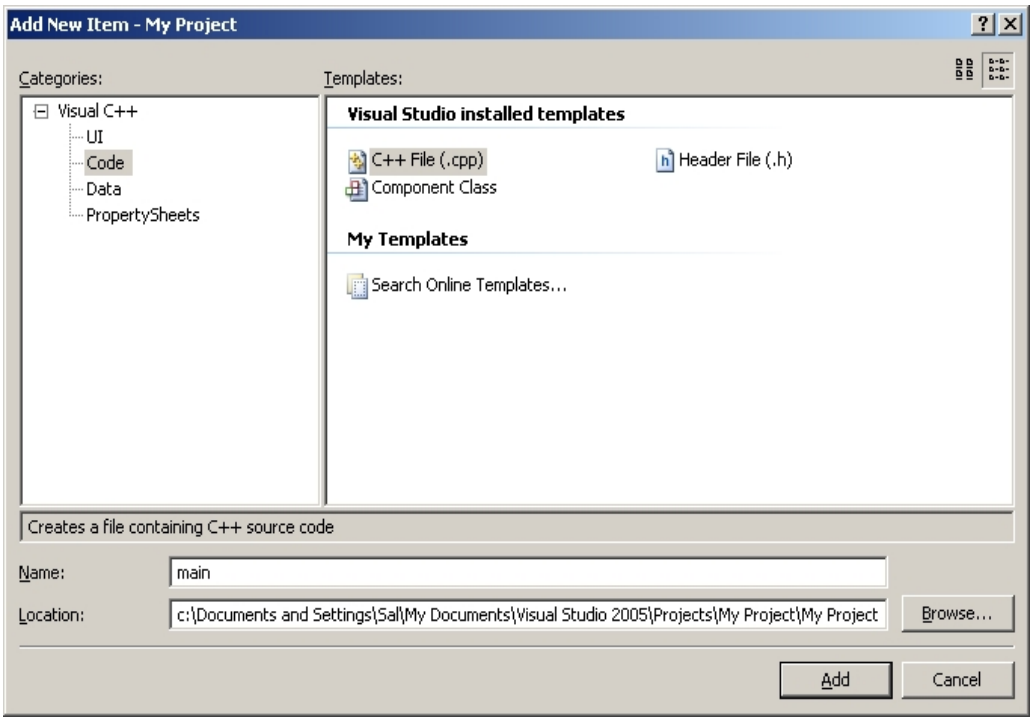
Y haga clic en Aceptar.

8. Ir a la configuración de la aplicación y asegúrese de que sea un proyecto vacío:

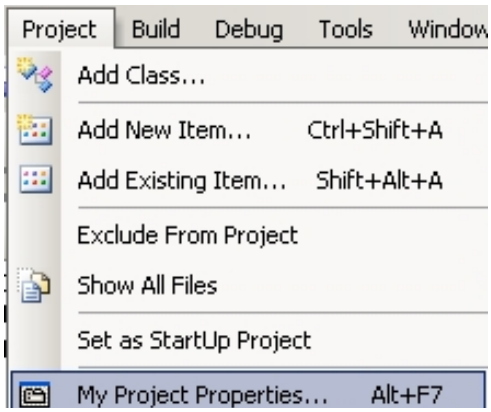


y haga clic en Aceptar.

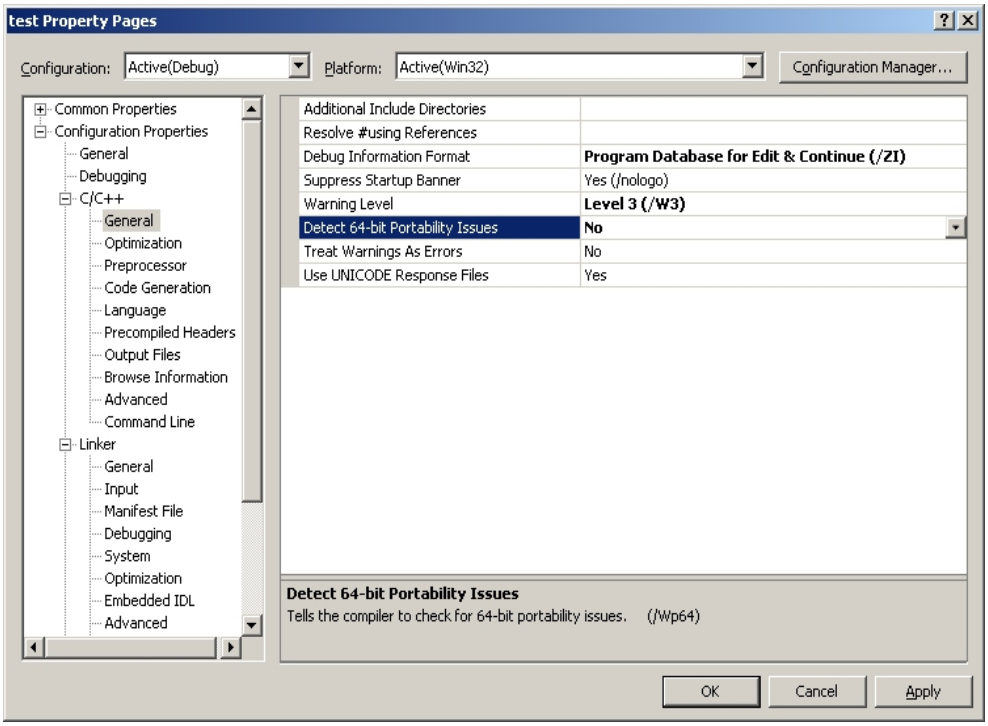
9. A continuación, agregue un nuevo fichero fuente al proyecto:



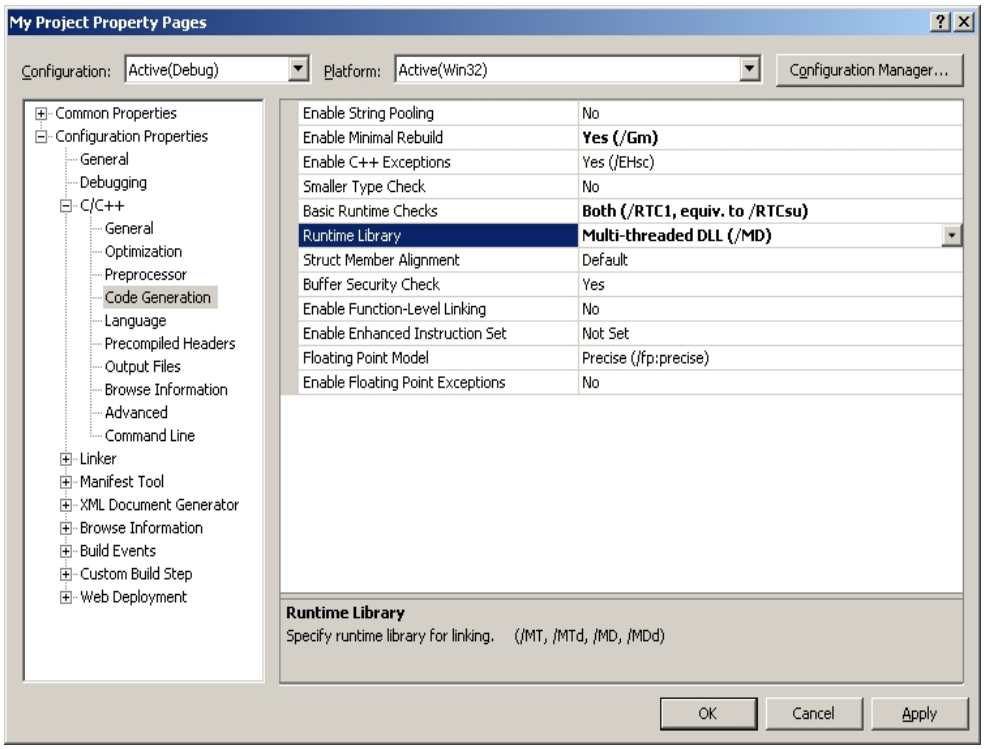
10. Ir a la configuración del proyecto.



11. En C / C + + en el menú general, ajuste "Detect 64-bit Portability Issues" a "No".

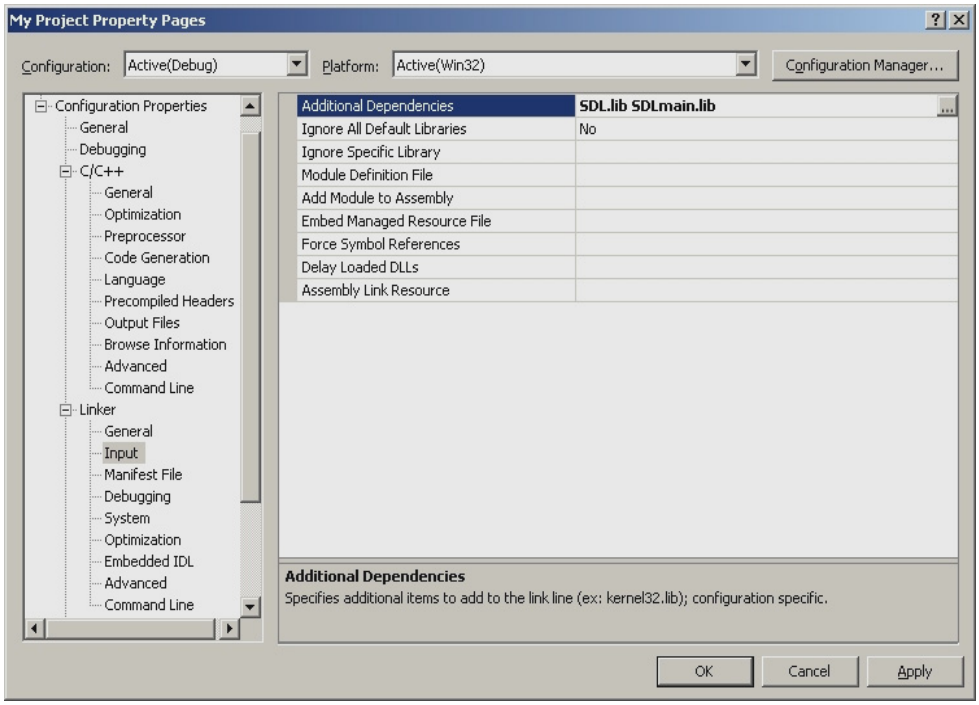


12. En C / C + + en el menú de Code Generation, Asigne en "Runtime a Library" Multi-threaded DLL(/MD).

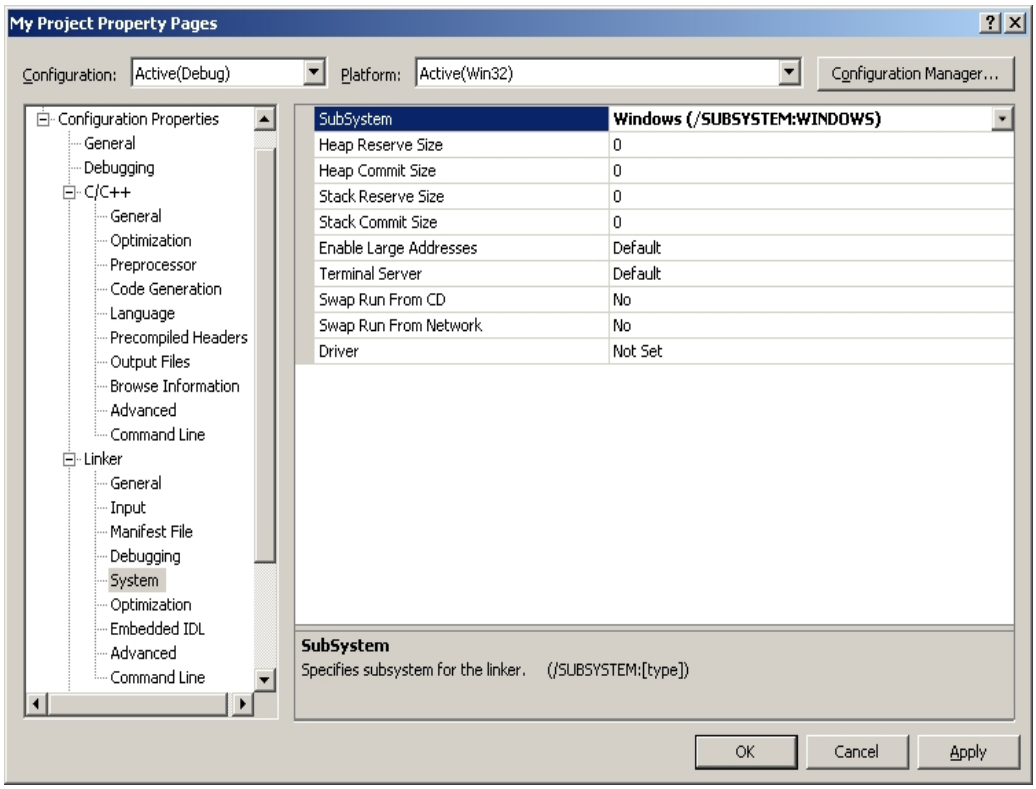


13. En el menú Linker -> Imput, en Additional Dependencies pegue:

SDL.lib SDLmain.lib



14. En el menú System, en Subsystem colocar Windows:



Ahora Generar. Asegúrese de que sdl.dll es en el mismo directorio que el proyecto / ejecutable.

PROGRAMAR CON LA LIBRERÍA SDL

SDL_mixer

Es una biblioteca externa que mejora la funcionalidad de audio proporcionando una manera fácil de cargar y reproducir muestras de audio.

SDL_mixer soporta los siguientes formatos:

- WAVE/RIFF (.wav)
- AIFF (.aiff)

- VOC (.voc)
- MOD (.mod .xm .s3m .669 .it .med y más) requiere libmikmod en su sistema.
- MIDI (.mid)
- OggVorbis (.ogg) require de la librería ogg/vorbis en su sistema
- MP3 (.mp3) requiere la librería SMPEG o MAD en su sistema.
- FLAC (.flac) requiere la librería FLAC en su sistema

Sin embargo, para esta demostración, estaremos usando .Wav para nuestras muestras de audio, y .Ogg para los archivos de música.

Para utilizar SDL_mixer, 3 sencillos pasos se deben seguir:

1. Iniciar SDL audio y SDL_mixer.
2. Cargar los archivos en forma de Mix_Chunk y Mix_Music
3. Lanzar eventos para iniciar las muestras

Inicialización

Antes de que cualquier archivo de audio puede escucharse, el SDL Audio subsystem y SDL_mixer debe ser inicializado en ese orden.

Para inicializar el SDL Audio subsystem:

```
/* Initialization */
if(SDL_InitSubSystem(SDL_INIT_AUDIO) == -1){
   // SDL Audio subsystem could not be started
}
/* Uninitialization */
SDL_QuitSubSystem(SDL_INIT_AUDIO);
```

Y para SDL_mixer:

```
/* Initialization */
if(Mix_OpenAudio(sampling_rate, format, playback_channels, buffer_size)
== -1){
    // SDL_Mixer could not be started
}

/* Uninitialization */
Mix_CloseAudio();
```

Muestras

Las muestras representan un archivo de audio cargado en memoria, en este caso bajo la forma de Mix_Chunk SDL_mixer. Para ello, podemos realizar lo siguiente:

```
Mix_Chunk * chunk = Mix_LoadWAV(path_and_name_of_audio_file);
if(!chunk){
    // Could not load file from disk
}
```

Y para liberar el bloque una vez que haya terminado con él:

```
Mix_FreeChunk(pointer_to_chunk);
```

El volumen de la muestra también puede modificarse mediante el siguiente código:

```
Mix_VolumeChunk(pointer_to_chunk, volume_level)
```

En lo anterior, el nivel de volumen debe ser un valor entre 0 (mínimo) y 128 (máximo).

Canales

Los canales permiten reproducir más de un sonido en un mismo momento. Cada canal puede reproducir un chunk diferente a la vez. Puedes decidir el número de canales que vayan a estar disponibles en el sistema con la funciones de inicialización y cambiarlo en un momento dado de la ejecución del programa. Hay opciones aplicables a cada canal que contenga un chunk, como por ejemplo el número de repeticiones, especificar cuánto tiempo se va a reproducir el sonido, o producir un efecto de fade en el canal de un determinado chunk. Podemos especificar en qué canal queremos reproducir un determinado sonido. Una vez que el sonido esté reproduciéndose podremos pausar, reanudar o parar uno de los canales, independientemente de los demás.

Ya tenemos el sonido cargado en un chunk. El siguiente paso que tenemos que seguir es el de reproducirlo en un canal de sonido. SDL_mixer tiene cuatro funciones dedicadas a la tarea de reproducir sonidos. La primera función que se encarga de esta tarea es:

```
Int Mix_PlayChannel(int channel, Mix_Chunk *chunk, int loops)
```

Como podrás observar esta función reproduce el sonido del parámetro chunk en el canal especificado en channel y si queremos reproducirlo sólo una vez deberemos de pasarle al parámetro loops el valor 0. Si queremos que el sonido de reproduzca una y otra vez, indefinidamente, pasaremos -1 en el parámetro loops.

SDL_mixer tiene la capacidad de poder seleccionar el canal de reproducción automáticamente. Para conseguir esto debemos de pasarle a la función precedente el valor -1 en el parámetro channel.

Además de esta función, como comentamos anteriormente, SDL_mixer proporciona otras tres funciones para reproducir sonidos. Estas funciones son:

```
int Mix_PlayChannelTimed(int channel, Mix_Chunk *chunk, int loops, int ticks);
```

Esta función es idéntica a Mix_PlayChannel pero el sonido se reproducirá durante los milisegundos indicados en el parámetro ticks. Si pasamos el valor -1 en el parámetro ticks el sonido se reproducirá indefinidamente, según la configuración propuesta en los otros parámetros. La configuración de los otros parámetros es exactamente igual que en Mix_PlayChannel.

TAREA

1. Estudiar la configuración de las librerías para hacer uso de SDL.
2. Añadir esta librería al proyecto y verificar si la instalación fue correcta.

REFERENCIAS

- Audio with SDL_Mixer http://www.cs.clemson.edu/~malloy/courses/3dgames-2007/tutor/web/audio/audio.html

PRÁCTICA Nº. 7

DISPOSITIVOS – DATA GLOVE

RESUMEN

En esta sección, usted habrá aprendido como instalar y usar el data glove.

OBJETIVO

Comprobar los diferentes gestos que el driver del guante puede distinguir.

Incluirlo en una aplicación con ambiente 3D.

Usar las funciones que dispone el SDK del guante .

PRE-REQUISITOS

- Obtener las librerías de 5DT.
- Crear una Mano 3D para utilizarla en la simulación de gestos, la mano tiene que ser diseñada utilizando figuras básicas de OpenGL como se muestra en la siguiente imagen. Esta imagen es solo un ejemplo, el diseño puede ser mejorado.

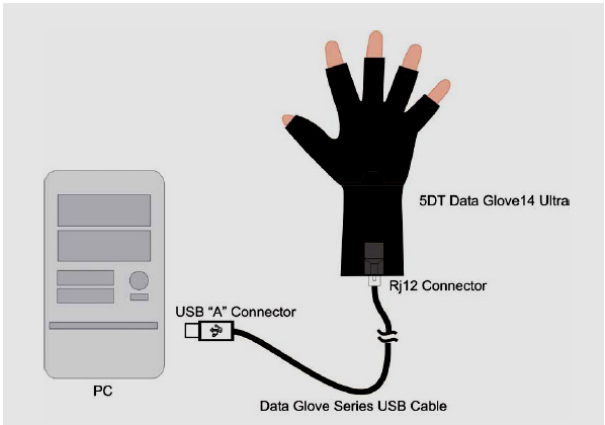
Mano diseñada utilizando OpenGL

DESARROLLO DE LA PRÁCTICA

INTEGRACIÓN DEL DRIVER DEL DATA GLOVE

En esta sección, el estudiante va aprender como configurar el data glove y usarlo para la interacción en OpenSceneGraph.

1. Estar seguro que el driver del data glove este instalado en la PC. Si el driver del guante está instalado usted debería encontrarlo normalmente en C:\Archivos de programa\5DT y en el menú Inicio de Windows sobre Todos los Programas/5DT.
2. Conectando el guante:



Conectar el guante a la PC como se muestra en la figura anterior. Tu puedes usar el Glove Manager (Inicio -> Todos los Programas -> 5DT -> Data Glove Ultra -> Glove Manager) para probar la operación del guante. Los detalles de la operación pueden ser encontrados en la Sección 4 del 5DT Data Glove Ultra Manual, también habilitado desde el menú inicio.

3. Usted también necesitarías añadir el directorio C:\Archivos de programa\5DT\Data Glove Ultra\SDK\dll a tu variable PATH (separando paths con una coma y sin espacios en blancos), Usted necesitaras aceptar todos los cambios realizados una vez para que los cambios tomen efecto.

4. Ahora que tu estas seguro que el guante está trabajando, abrimos Visual Studio. El primer paso que se tiene que hacer es nuevamente decirle a Visual Studio los paths del driver del guante. En Visual Studio, en el menú principal vamos a Herramientas/Opcions. En la siguiente ventana seleccionamos la categoría Proyectos y soluciones/Directorios de VC++. Ahora estar seguro que los siguientes directorios están incluidos y añadirlos si no lo están:

   a. En la categoría Archivos de inclusión: C:\Archivos de programa\5DT\Data Glove Ultra\SDK\inc

   b. En archivos ejecutables: C:\Archivos de programa\5DT\Data Glove Ultra\SDK\dll

   c. Y en archivos de bibliotecas: C:\Archivos de programa\5DT\Data Glove Ultra\SDK\lib

5. Ahora creamos un nuevo Proyecto en Visual Studio. Vamos a las propiedades del Proyecto (Click derecho sobre el proyecto y seleccionamos propiedades). En esta ventana de la categoría

(Propiedades de configuración/Vinculador/Entrada). En Dependencias adicionales añadimos fgloved.lib para la Configuración de Debug (debería ser standard) y fglove.lib para la configuración de Release (únicamente si su plan es compilar en la versión de release).

6. Ahora el siguiente fragmento de código tiene que ser incluido en la programa para poder trabajar con el data glove:

En la sección de include estar seguro de incluir el archivo de cabecera del guante:

```
#include <windows.h> //librería necesaria
#include <fglove.h>
```

En la sección de inicialización de tu programa, inicializa el data glove:

```
fdGlove *glove = fdOpen("USB0");
if (glove == NULL{
printf("Failed initializing the glove\n");
return -1;
}
```

Después, antes del final del programa, añadir la línea:

```
fdClose(glove);
```

Después de estos pasos, Usted puedes ahora usar la variable para consultar el estado del guante, para cada frame.

Consultar el estado actual del guante

Usted tienes diferentes opciones para usar el guante. La cual una es dependiente sobre como Usted quieres integrar el guante en tu aplicación. Las dos más usadas son:

Detección de gestos: El driver del guante puede distinguir entre 16 diferentes gestos discretos. El número del gesto actual es adquirido con esta función:

```
int gesture_no = fdGetGesture(glove);
```

Gesture Illustration

Usted puede encontrar el significado de los gestos en la página 30 y 31 del manual del Data Glove para una mejor ilustración.

Escalado del sensor de dato: El método anterior solo nos da datos discretos. Si usted quiere datos continuos, por ejemplo, si es importante para usted distinguir si un cierto dedo esta flexionado un 60% o 70%, entonces use la siguiente función:

float sens_value = fdGetSensorScaled(glove, sens_no);

El valor de retorno de esta función es un float entre 0 y 1 que expresa el grado en el cual un sensor específico es doblado. El segundo parámetro de la función (sens_no) es un int que indica cual sensor es usado. El valor para estos rangos entre cero y el número de sensores -1. El número de sensores debería ser 18 (aunque el ultimo -17- se refiere al sensor de inclinación que no está incluido en nuestro modelo del guante, este solo podría retornar 0 constantemente). Puedes también saber el número de sensores usando la siguiente función:

fdGetNumSensors();

Usted puede buscar más información sobre los sensores en la página 28 y 29 del manual de guante, para una ilustración de la locación del sensor a través de la mano.

Nota: Para mayor información sobre estas funciones y una completa documentación del API del guante, revisar el manual, el cual puede encontrarse como un PDF en el menú inicio en la carpeta del 5DT.

TAREA

Esta tarea será realizada en OpenGL, con esto se quiere demostrar la diferencia de trabajar entre las librerías de OSG y OpenGL.

1. Estudiar el documento para iniciar la instalación del Data Glove.
2. Crear un proyecto en Visual Studio .NET versión 9.0 y configúrelo para hacer uso de las librerías de OpenGL.
3. Integrar la mano virtual en el proyecto y haciendo uso de las funciones del Data Glove, generar los diferentes gestos por medio de la mano 3D. Es decir si en el ambiente real se genera el gesto 0, el objeto 3D también deberá generarlo en el ambiente virtual.
4. Crear otro tipo de gesto teniendo como base los gestos que ya dispone el Data Glove. Usted podrá por ejemplo unir dos o más gestos básicos o crear uno totalmente nuevo.

REFERENCIAS

- Paper: Part4 Data Glove Interaction www.macs.hw.ac.uk/.../Part%204%20Data%20Glove%20Interaction%20m.doc
- Paper: Interaction Devices – Data Glove http://www.macs.hw.ac.uk/modules/F24VS2/Labs/OSG-TutInteractionDevices.pdf

PRÁCTICA NO. 8

INTERACCIÓN CON DISPOSITIVOS – DATA GLOVE

Interactuando con un escenario 3D

RESUMEN

En esta práctica, usted se dispondrá a realizar la interacción entre el guante y una escena 3D, mediante gestos que se obtiene del dispositivo.

OBJETIVO

- Obtener los gestos del guante e Interactuar con la escena
- Realizar algunas acciones con la escena para obtener habilidad en la manipulación del dispositivo.

PRE-REQUISITOS

- Realizar la práctica 3, la cual explica cómo cargar objetos a la escena.
- Cumplir con la práctica 7 para conocer como se trabaja con el 5DT Data Glove.
- Crear un ambiente virtual, el cual debe constar de varios objetos. El estudiante podrá decidir si desea un escenario sencillo o un poco más elaborado.

DESARROLLO DE LA PRÁCTICA

Se muestra la forma de trabajar de la Clase Viewer y el uso de la Cámara en la aplicación.

La Clase Viewer

El código fuente demuestra el mínimo código requerido para renderizar una escena en OSG. Las instancias de Viewer en el objeto osgViewer::Viewer, añaden la escena grafica a ésta y permiten su renderización. Las siguientes líneas deberán serán añadidas.

```
#include <osgViewer/Viewer>
#include <osgDB/ReadFile>
int main( int, char ** )
{
osgViewer::Viewer viewer;
viewer.setSceneData( osgDB::readNodeFile( "cow.osg" ) );
return viewer.run();
}
```

Cambiando la Vista

Viewer crea un objeto osg::Camera para manejar la matriz de modelo-vista en OpenGl. Existen 2 maneras para controla la Cámara.

- Añadir un manipulador de cámara al Viewer. Si tu aplicación no hace esto, Viewer::run() crea un osgGA::TrackballManipulator para controlar la Cámara. La librería osgGA define algunos manipuladores que se pueden utilizar. Viewer::setCameraManipulator() para especificar tu propio manipulador.

- Establecer la proyección de la cámara y las matrices de perspectiva que se van a definir. Esto da a la aplicación un completo control sobre la vista.

Si se elige establecer la Cámara directamente, usando Viewer::run() no es la forma correcta porque esto no permite cambiar la vista por frame. En su lugar, tendrá que codificar un pequeño bucle que realice interactivamente las actualizaciones de la vista y renderizar el frame.

Se muestra cómo controlar el Viewer para cambiar la vista por cada frame.

```
osgViewer::Viewer viewer;
viewer.setSceneData( osgDB::readNodeFile( "cow.osg" ) );
viewer.getCamera()->setProjectionMatrixAsPerspective(
40., 1., 1., 100. );
// Create a matrix to specify a distance from the viewpoint.
osg::Matrix trans;
trans.makeTranslate( 0., 0., -12. );
// Rotation angle (in radians)
```

```
double angle( 0. );
while (!viewer.done())
{
// Create the rotation matrix.
osg::Matrix rot;
rot.makeRotate( angle, osg::Vec3( 1., 0., 0. ) );
angle += 0.01;
// Set the view matrix (the concatenation of the rotation and
// translation matrices).
viewer.getCamera()->setViewMatrix( rot * trans );
// Draw the next frame.
viewer.frame();
}
```

Geodes y Geometry describen brevemente las coordenadas por defecto del sistema de orientación de OSG. El sistema de coordenadas por defecto de la Cámara es X positivo a la derecha, Z positivo hacia arriba y Y positivo hacia dentro de la pantalla. Lo siguiente describe como cambiar la orientación por defecto de la Cámara. El siguiente texto describe cómo cambiar el default de la Cámara.

```
void setProjectionMatrix( const osg::Matrix& matrix );
void setProjectionMatrixAsOrtho( double left, double right,
double bottom, double top, double zNear, double zFar );
void setProjectionMatrixAsOrtho2D( double left, double right,
double bottom, double top );
void setProjectionMatrixAsFrustum( double left, double right,
double bottom, double top, double zNear, double zFar );
void setProjectionMatrixAsPerspective( double fovy,
double aspectRatio, double zNear, double zFar );
```

Camera::setProjectionMatrix() toma un osg::Matrix como un parámetro y es análogo a la siguiente secuencia de los comando de OpenGL:

```
glMatrixMode( GL_PROJECTION );
glLoadMatrixf( m );
```

Camera::setProjectionMatrixAsOrtho() crea un matriz de proyección usando un algoritmo idéntico a glOrtho() de OpenGL, mientras el método setProjectionMatrixAsOrtho2D() es mas análogo al punto de entrada de GLU, gluOrtho2D(). Cámara también provee métodos para establecer una perspectiva de proyección análoga a los comandos glFrustum() y gluPerspective().

Estableciendo Color

El objeto Cámara provee interfaces para diferentes operaciones además de establecer la vista. El siguiente código muestra como establecer el color a negro.

```
viewer.getCamera()->setClearColor( osg::Vec4( 0., 0., 0., 1. ) );
```

Archivos de Cabecera

```
#include <osgViewer/Viewer>
#include <osgGA/GUIEventHandler>
#include <osg/ShapeDrawable>
#include <osg/PositionAttitudeTransform>
```

En los archivos ***handlerCam.h*** y ***handlerCam.cpp*** se muestra un ejemplo para el manejo de la cámara utilizando la clase osgGA::MatrixManipulator.

TAREAS

1. Revisar el código proporcionado y entender cómo es el manejo de la cámara en la escena gráfica.
2. Crear un proyecto en Visual Studio .NET versión 9.0 y configúrelo para hacer uso de las librerías de OSG, así como las del guante.
3. Incluir en la aplicación los archivos proporcionados para hacer uso de la cámara en conjunto con el dispositivo. Incluya las líneas de código necesarias para utilizar el dispositivo.

4. Ya con el ambiente virtual creado y añadido a la aplicación, elegir un objeto dentro de éste y manipularlo con los gestos generados por el guante, por lo menos se deberá elegir 3 gestos.

5. Controlar el movimiento de la cámara a través de gestos elegidos por el estudiante, los movimientos a representar son: cámara adelante, cámara retroceder, izquierda y derecha.

6. Cambiar el color de un objeto mediante un gesto del dispositivo y con otro gesto moverlo en diferentes posiciones.

RECURSOS

Archivos: handlerCam.h, handlerCam.cpp y main.cpp.

REFERENCIAS

- Document: Part4 Data Glove Interaction
  www.macs.hw.ac.uk/.../Part%204%20Data%20Glove%20Interaction%20m.doc

- http://www.lulu.com/items/volume_51/767000/767629/3/print/OSGQSG.pdf

PRÁCTICA Nº. 9

PICKING (SELECCIÓN DE OBJETOS)

RESUMEN

Esta práctica muestra otra funcionalidad del cual dispone OSG, la cual es seleccionar un objeto de la escena por medio de un manejador, en este caso PickHandler. Este permite saber la posición del mouse y así seleccionar el objeto que se desea.

OBJETIVOS

- Mostrar cómo seleccionar un objeto dentro de la escena con los eventos del mouse.
- Utilizar los eventos del teclado para conseguir que los objetos seleccionados realicen ciertos movimientos al presionar específicas teclas.
- Conseguir la interacción entre el mouse y el guante para la selección de un objeto en la escena.

PRE-REQUISITOS

- Realizar la práctica 3, la cual explica cómo cargar objetos a la escena.
- Cumplir con la práctica 7 para conocer como se trabaja con el 5DT Data Glove.
- Crear un escenario con al menos 3 objetos 3D.

DESARROLLO DE LA PRÁCTICA

Se explican las clases necesarias para la realizar Picking.

Clase NodeVisitor

NodeVisitor recorre un grafo de la escena y llama a una función para cada nodo visitado. Esta simple técnica reside como una clase base para muchas operaciones de OSG, incluyendo osgUtil::Optimizer, las clases de procesamiento de geometría de la librería osgUtil y file output. OSG usa la clase osgUtil::UpdateVisitor (derivada de NodeVisitor) para ejecutar la actualización del recorrido.

NodeVisitor es una clase que la aplicación nunca instancia directamente. Se puede crear su propia clase derivada de NodeVisitor. Cuando un NodeVisitor recorre una escena gráfica, este llama apropiadamente el método para cada nodo que visita.

Seleccionar (Picking)

La mayoría de las aplicaciones 3D requieren de alguna forma la funcionalidad de seleccionar, para permitir que el usuario final seleccione una porción de la imagen mostrada. En su forma más simple, el usuario posiciona el mouse sobre la escena mostrada y hace clicks con el botón de mouse. Internamente, la aplicación realiza una operación para asignar la ubicación 2D XY del mouse a la correspondiente escena gráfica y almacena la dirección de los nodos para operaciones futuras.

En esencia, las aplicaciones basadas en OSG realizan dos operaciones necesarias para ejecutar la selección.

- Recibir los eventos del mouse. La librería osgGA provee clases de eventos que permiten que las aplicaciones reciban los eventos del mouse de forma independiente.

- Determinar que parte de la escena gráfica esta debajo del cursor del mouse. La librería osgUtil provee las clases de intersección que crean un volumen alrededor de la ubicación XY del mouse y permiten interceptar ese volumen con la escena gráfica. osgUtil retorna una lista de los nodos interceptados por el volumen, los cuales son clasificados en orden de adelante hacia atrás.

En esta sección se describe cómo implementar ambas operaciones.

Capturando Eventos del mouse

El método handle() tiene dos parámetros, un osgGA::GUIEventAdapter y un osgGA::GUIActionAdapter, como se muestra a continuación:

```
bool handle(const osgGA::GUIEventAdapter& ea, osgGA::GUIActionAdapter& aa)
{ }
```

En la implementación de handle recibe eventos GUI incluyendo eventos del mouse en GUIEventAdapter. El archivo de cabecera GUIEventAdapter declara un EventType que permite que la aplicación examine únicamente eventos de interés, tales como eventos del mouse.

GUIActionAdapter es la interfaz en la aplicación del sistema GUI. En el caso de mouse picking, agrega el picking GUIEventHandler a la clase viewer. Esto es para que se pueda realizar una intersección con la escena basada en la presente vista.

Antes de reproducir la escena, crea una instancia de la nueva clase GUIEventHandler y añade esta al viewer con el método Viewer::addEventHandler(). Como su nombre lo indica, el viewer puede tener muchos manejadores de eventos. El viewer llama al handle() por cada evento GUI hasta que uno de los métodos handle() retorna true.

Clase PickHandler

Para implementar picking en OSG, usa una subclase derivada de osgGA::GUIEventHandler. Se muestra la clase PickHandler. Esta clase define dos métodos, uno recibe los eventos del mouse y el otro la operación picking implementada sobre el mouse release.

Archivos de cabecera necesarios

```
#include <osg/io_utils>
#include <osg/observer_ptr>
#include <osg/MatrixTransform>
#include <osg/PositionAttitudeTransform>

#include <osgUtil/IntersectionVisitor>
#include <osgUtil/PolytopeIntersector>
#include <osgUtil/LineSegmentIntersector>

#include <osgDB/ReadFile>
#include <osgDB/WriteFile>

#include <osgGA/TrackballManipulator>
#include <osgGA/StateSetManipulator>
```

Clase PickHandler

```
#include <osgViewer/Viewer>

// PickHandler -- A GUIEventHandler that implements picking.
class PickHandler : public osgGA::GUIEventHandler
{
public:
        PickHandler() : _mX( 0. ),_mY( 0. ) {}
        ~PickHandler() {}
        bool handle(const osgGA::GUIEventAdapter& ea,osgGA::GUIActionAdapter& aa )
       {
          osgViewer::Viewer* viewer =
          dynamic_cast<osgViewer::Viewer*>( &aa );
          if (!viewer) return false;
           switch( ea.getEventType() )
           {
               case osgGA::GUIEventAdapter::PUSH:
               case osgGA::GUIEventAdapter::MOVE:
               {
                    // Record mouse location for the button press
```

```
                // and move events.
                _mX = ea.getX();
                _mY = ea.getY();
                return false;
            }
            case osgGA::GUIEventAdapter::RELEASE:
            {
                // If the mouse hasn't moved since the last
                // button press or move event, perform a
                // pick. (Otherwise, the trackball
                // manipulator will handle it.)
                if (_mX == ea.getX() && _mY == ea.getY())
                {
                    if (pick( ea.getXnormalized(), ea.getYnormalized(), viewer ))
                        return true;
                }
                return false;
            }
            default:
                return false;
        }
    }
protected:
    // Store mouse xy location for button press & move events.
    float _mX,_mY;
    // Perform a pick operation.
    bool pick( const double x, const double y, osgViewer::Viewer* viewer )
    {
        if (!viewer->getSceneData())
        // Nothing to pick.
            return false;
        double w( .05 ), h( .05 );
        osgUtil::PolytopeIntersector* picker =
                new osgUtil::PolytopeIntersector(osgUtil::Intersector::PROJECTION,
                                                 x-w, y-h, x+w, y+h );
                osgUtil::IntersectionVisitor iv( picker );
                viewer->getCamera()->accept( iv );
          if (picker->containsIntersections())
          {
                Const osg::NodePath& nodePath =
                                                 picker->getFirstIntersection().nodePath;
                unsigned int idx = nodePath.size();
                while (idx--)
                {
                    // Find the LAST MatrixTransform in the node
                    // path; this will be the MatrixTransform
                    // to attach our callback to.
                    osg::MatrixTransform* mt =
                    dynamic_cast<osg::MatrixTransform*>(nodePath[ idx ] );
```

```
                        if (mt == NULL)
                                continue;
                         // If we get here, we just found a
                         // MatrixTransform in the nodePath.
                         if (_selectedNode.valid())
                                // Clear the previous selected node's
                                // callback to make it stop spinning.
                                _selectedNode->setUpdateCallback( NULL );
                        _selectedNode = mt;
                        _selectedNode->setUpdateCallback( new RotateCB );
                        break;
                      }
                      if (!_selectedNode.valid())
                            osg::notify() << "Pick failed." << std::endl;
                    }
                    else if (_selectedNode.valid())
                    {
                        _selectedNode->setUpdateCallback( NULL );
                        _selectedNode = NULL;
                     }
                      return _selectedNode.valid();
                }
            };

int main( int argc, char **argv )
{
    // create the view of the scene.
      osgViewer::Viewer viewer;
      viewer.setSceneData( createScene().get() );
    // add the pick handler
      viewer.addEventHandler( new PickHandler );
      return viewer.run();
}
```

También se muestra la función main() del ejemplo de Picking para ilustrar el uso del método Viewer::addEventHandler() para añadir un evento handler al viewer.

Intercepciones

OSG emplea la propia naturaleza jerarquica de la escena gráfica para calcular eficientemente las intersecciones, evitando a menudo la lenta función de selección de OpenGL. La clase osgUtil::IntersectionVisitor deriva de NodeVisitor y prueba cada delimitación de volumen del Nodo contra el volumen de

intersección, lo que permite saltar recorridos de subgrafos que no tengan la posibilidad de una intersección.

IntersectionVisitor puede ser configurado para pruebas de intersección con diferentes construcciones geométricas incluyendo de planos y segmentos de líneas. Su constructor toma un osgUtil::Intersector como un parámetro, el cual define la geometría y selecciona la actual prueba de intersección. Intersector es una clase base virtual que la aplicación no puede instanciar. La librería osgUtil deriva de algunas clases de Intersector para representar diferentes construcciones geométricas, incluyendo osgUtil::PolytopeIntersector, el cual es ideal para hacer selecciones basadas en el mouse.

Algunas aplicaciones requieren recoger vértices individuales o polígonos. Otras aplicaciones simplemente necesitan saber el nodo que lo contiene o la geometría seleccionada. Conocer estos requerimientos, IntersectionVisitor retorna un osg::NodePath. NodePath es un std::vector<osg::Node> que representa la ruta a través de la jerarquía de nodos que va desde el nodo raíz hasta el nodo hoja. Si la aplicación requiere un grupo de nodo intermedio, busca el NodePath de atrás hacia adelante hasta encontrar el nodo que se adapte con los requerimientos de la aplicación.

En resumen, para realizar un picking con el mouse en OSG, su aplicación debe realizar los siguientes pasos:

- Crear y configurar un PolytopeIntersector usando la ubicación almacenada del mouse en el GUIEventAdapter.
- Crear un IntersectionVisitor y pasar el PolytopeIntersector como un parámetro en el constructor.
- Iniciar el IntersectionVisitor en el nodo raíz de la escena gráfica, usualmente a través del Viewer de la Cámara, como en el siguiente código:

```
// 'iv' is an IntersectionVisitor
viewer->getCamera()->accept( iv );
```

- Si el PolytopeIntersector contiene intercepciones, obtiene el NodePath y busca hasta encontrar el nodo de interés.

Eventos del teclado

```
case osgGA::GUIEventAdapter::KEYDOWN:
    {
        if(ea.getKey()=='r')
        {
            rotar = true;
        }
    }
case(osgGA::GUIEventAdapter::KEYUP):
    {
        if(ea.getKey()=='r')
        {
            rotar = false;
        }
    }
case(osgGA::GUIEventAdapter::FRAME):
    {
        If(rotar){}
    }
```

Función OsgFX::Scribe

Función que crea un highlight con OsgFX::Scribe para mostrar que ese nodo ha sido seleccionado, crea un borde alrededor del objeto seleccionado.

```
void toggleScribe(osg::Group* parent, osg::Node* node) {
if (!parent || !node) return;
    std::cout<<"  parent "<<parent->className()<<std::endl;
        osgFX::Outline* parentAsScribe =
dynamic_cast<osgFX::Outline*>(parent);
    if (!parentAsScribe)
    {
        osgFX::Outline *outline = new osgFX::Outline;
        outline->addChild(node);
        parent->replaceChild(node,outline);
    }
    else
    {
       // node already picked so we want to remove scribe to unpick it.
       osg::Node::ParentList parentList = parentAsScribe->getParents();
```

```
        for(osg::Node::ParentList::iterator itr=parentList.begin();
           itr!=parentList.end();
           ++itr)
         {
                              (*itr)->replaceChild(parentAsScribe,node);
         }
    }
}
```

TAREAS

1. Revisar el código anteriormente explicado y entender el manejo de intercepciones y la forma de seleccionar objetos específicos por medio de la clase PickHandler.
2. Crear un proyecto en Visual Studio .NET versión 9.0 y configúrelo para hacer uso de las librerías de OSG.
3. Utilizar el escenario creado para mostrar el uso de la clase PickHandler, conseguir seleccionar por medio del mouse los objetos contenidos en la escena.
4. Una vez que se haya conseguido la selección de los objetos, utilizar eventos del teclado para conseguir que al presionar una tecla el objeto rote o se traslade en diferentes direcciones.
5. Integrar el 5DT Data Glove a la aplicación, y conseguir que la selección del objeto sea con un gesto del guante. Cabe recalcar que el gesto solo servirá para seleccionar el objeto que se encuentra en una área de la pantalla que ha sido indicada por el puntero del mouse más no para determinar el área de selección, ésta se hará con el mouse; el guante solo retorna un entero que es asignado a un gesto, dependiendo de la posición de los dedos de la mano y no es utilizado para determinar posiciones (coordenadas X,Y, Z) en la

pantalla, esto lo realiza otro tipo de dispositivo como lo es el Tracker el cual será visto en otra práctica.

6. Lo que se realizó al presionar una tecla ahora se deberá realizar con un gesto del guante. Cada movimiento que se haga al objeto, tiene que ser realizado con diferentes tipos de gestos.

REFERENCIAS

- Ejemplo de picking http://www.openscenegraph.org/projects/osg/browser/OpenSceneGraph/trunk/examples/osgkeyboardmouse/osgkeyboardmouse.cpp
- Ejemplo para enmarcar objeto seleccionado http://www.sandbox.de/osg/src/OutlineFX_20091113.tar.gz
- Funciones de OSG http://www.lulu.com/items/volume_51/767000/767629/3/print/OSGQSG.pdf

PRÁCTICA Nº. 10

DISPOSITIVOS – POLHEMUS TRACKER

RESUMEN

En esta práctica se explica otro de los dispositivos del laboratorio, los Trackers, se muestra la configuración para que el sistema los reconozca y se pueda utilizar en las aplicaciones. También se ilustra el cuidado que se debe tener al momento de usarlo.

OBJETIVO

- Configurar una aplicación para incluir los Trackers y hacer uso de sus funciones para la interacción con el mismo.
- Reconocer las librerías que se necesitan para hacer uso del Polhemus Tracker.
- Capturar por medio de los sensores las posiciones y utilizar estos valores en la escena.

PRE-REQUISITOS

- Conseguir las librerías para el uso del Polhemus.
- Desarrollar la práctica 3 la cual muestra como cargar objetos 3D en la escena.
- Desarrollar la práctica 8 en la que se enseña el manejo de la cámara en la escena.
- Crear un escenario virtual con un objeto 3D, se puede elegir cualquier objeto.

DESARROLLO DE LA PRÁCTICA

En esta sección, el estudiante va aprender como configurar los trackers y utilizarlo para la interacción en OSG.

Figure 1-1 LIBERTY

Configuración del LIBERTY

1. Desempaque el sistema LIBERTY (System electronics unit, SEU), la fuente, sensores, USB y cables RS-232, CD, cable de alimentación y cables. Figura 1-1.
2. Configurar el sistema LIBERTY cercano a su computador y lejos de objetos de metal tales como armarios, escritorios de metal, etc, y lejos de piso y paredes.
3. Examine la parte trasera del LIBERTY y revisar la ubicación de la fuente, sensores, power, RS-323, USB y puertos de sincronización externa. Figura 1-2

Figure 1-2

4. Identificar la fuente (el dispositivo con conector DB-15) e insertar el conector en el receptáculo de la fuente de origen, teniendo cuidado de comprometerse firmemente a ella. Con los dedos apretar los dos tronillos de fijación para asegurar el conector. Figura 1-3

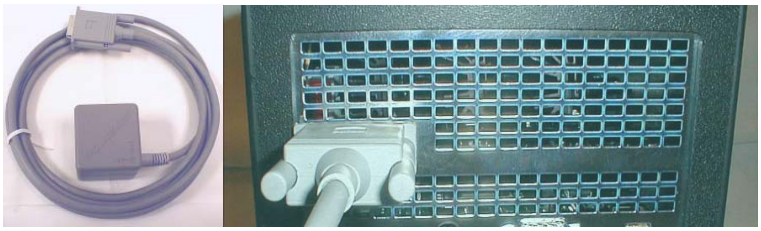


Figure 1-3 Source and Source Connection

5. Para empezar, utilice solamente un sensor. Identifique el sensor e inserte en cualquiera de los recipientes del sensor como se muestra a continuación. Comprometerse firmemente y bloquear el conector del sensor en su lugar en la misma manera que el conector de la fuente en el paso 3. Figura 1-4

Figure 1-4 Sensor and Sensor Connection

6. Para propósito de prueba, es conveniente montar la fuente y el sensor en un solo bloque de madera (2 X 4 o equivalente) alrededor de 16 pulgadas de distancia. Ubicación exacta de la fuente y el sensor no es importante para esta prueba, solo asegúrese de que los cables de ambos dispositivos no se encuentren muy juntos y que salgan de los extremos opuestos de la madera de montaje. Figura 1-5.

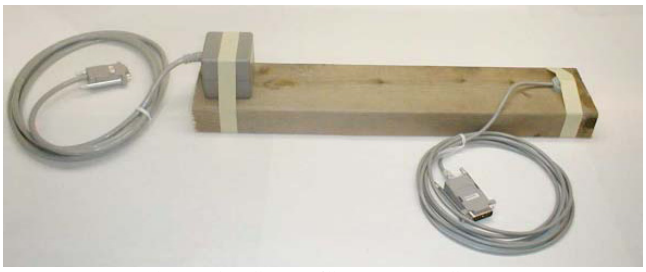


Figure 1-5 Sensor/Source Test Setup

7. Identificar el módulo de entrada de alimentación con el interruptor ON/OFF en el panel posterior. Asegúrese de que este interruptor está en la posición OFF (lógica “0” a la derecha) antes de insertar el cable de alimentación de CA y aplicando 110/220 VAC. Figura 1-6

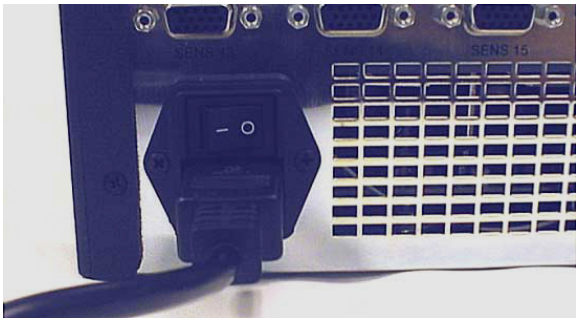


Figure 1-6 Power Connector

Comunicación USB o RS-232

Solo un puerto I/O (USB o RS-232) pueden estar activadas al mismo tiempo.

Para USB, continue con el paso 8

Para el RS-232, salte al paso 12

8. Comunicación USB:

Identificar el cable USB e introdúzcalo en el receptor como se muestra en la figura 1-7. Conecte el otro extremo del cable USB a la computadora.

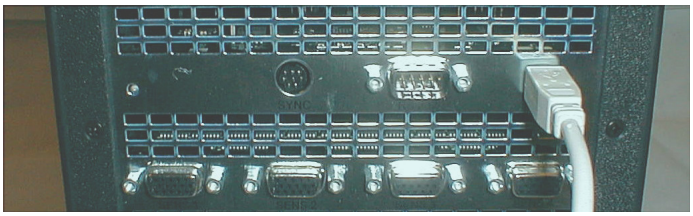


Figure 1-7 USB Cable Connection

9. En este punto, usted puede encender el sistema LIBERTY usando el interruptor de alimentación situado en el panel posterior de la SEU. Un indicador de estado del sistema se encuentra en la parte delantera y trasera de la ubicada electrónica debe parpadear rojo de 5 a 10 segundos indicando auto-prueba y puesta a punto. Cuando se completen estas rutinas, el indicador mostrara el estado del sistema de la siguiente manera:

Verde Fijo	Sistema operativo con una buena calidad de la señal magnética.	Sin distorsión
Rojo intermitente/verde	Sistema operativo con una señal magnética marginal.	Menor distorsión
Rojo solido	Sistema operativo con una señal magnética pobre.	Distorsiones importantes
Rojo intermitente	auto-prueba e instalación fallidos.	

10. El computador debe responder con un mensaje de “Nuevo hardware encontrado”. Siga el asistente de hardware para instalar los controladores necesarios del CD de LIBERTY. Para una guía paso a paso, consulte la instalación del controlador en la página 7 del manual.

NOTA: Una vez que el cable USB está conectado, no puede volver al modo RS-232 sin quitar la conexión USB y reiniciar (power OFF/ON).

11. Ahora puede utilizar la interfaz grafica de usuario PIMgr Polhemus. Si todavía usted no ha instalado el software, continúe con instalar el software del LIBERTY en la página 6 del manual. De lo contrario,

continúe con Uso de la Interfaz grafica de usuario PiMgr Polhemus de la página 7 y experimentar el LIBERTY de datos en la página 9 del manual.

Para comunicaciones RS-232

12. Busque el cable RS-232 e inserte en el receptor como se muestra en la figura 1-8.

La mayoría de ordenadores tiene una PC de 9 pines, macho “D” de tipo COM1. Si está utilizando COM1, conecte el otro extremo del cable en el puerto COM1 de la PC.

Si el equipo tiene uno de 25 pines “D” para el puerto RS-232, se necesita un adaptador de 9 a 25 pines “D”. Tenga en cuenta que este adaptador no debe comprometer el sentido del cable MODEM

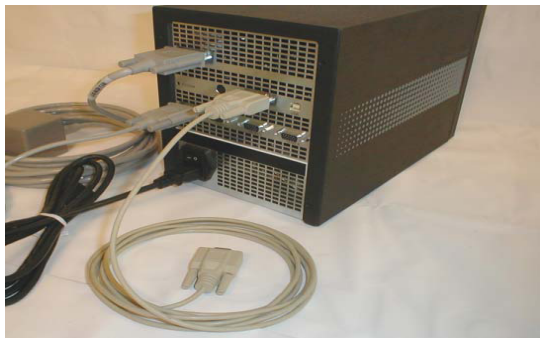


Figure 1-8 RS-232 Cable Connection

Instalando el Polhemus Tracker SDK

7. Estar seguro que el driver del tracker este instalado en la PC.

8. Seleccione cualquiera de las configuraciones disponibles, y asegúrese de que este seleccionada “Polhemus SDK Files”

Trabajando con Visual Studio.NET

1. En su solución vaya a Propiedades del proyecto -> C/C++ General y en Directorios de inclusión adicionales añada: C:\ Archivos de Programa\PDI_71\Inc

   Si usted instaló el PDI en otro directorio, entonces ubica ese que elegiste.

2. Ahora en Vinculador -> General y en Directorios de bibliotecas adicionales añada: C:\Archivos de Programa\PDI_71\Lib

3. En la misma pestaña de Vinculador -> Entrada y en Dependencias Adicionales añada: PDID.lib (para Debug) or PDIUD.lib (para UNICODE Debug) or PDI.lib (para release) or PDIU.lib (para UNICODE Release).

En tiempo de ejecución

Para ejecutar, su aplicación tendrá que ser capaz de localizar el PDI.dll que se está utilizando. Existen dos formas:

1. Copiar el archivo dll en el directorio de donde su aplicación está corriendo, dentro de la carpeta debug. Las dlls están localizadas en C:\Archivos de Programa\Polhemus\PDI_71\Lib\

2. Para depurar dentro del Visual Studio.NET, también puede agregar la ruta anterior a las Propiedades del Proyecto->Depuración->Directorio de trabajo.

Inicialización del Tracker

```
#include "Tracker.h"
Tracker *tracker = new Tracker;
tracker->Initialize();
```

Recogiendo puntos del Tracker

```
PFLOATpuntosTracker;

float           posXIniTracker = 0.0;
float           posYIniTracker = 0.0;
float           posZIniTracker = 0.0;
float           posAzIniTracker = 0.0;
float           posElevationIniTracker = 0.0;
float           posRollIniTracker = 0.0;

do{

        puntosTracker=this->tracker->ValoresTracker();

}while(puntosTracker==NULL);

        posXIniTracker=puntosTracker[0];

        posYIniTracker=puntosTracker[1];

        posZIniTracker=puntosTracker[2];

        posAlphaIniTracker=puntosTracker[3];
```

TAREA

1. Revisar la documentación proporcionada y seguir cada uno de los pasos que se muestran para la instalación y configuración de los trackers.
2. Crear un proyecto en Visual Studio .NET versión 9.0 y configúrelo para hacer uso de las librerías de OSG.
3. Utilizar los archivos *handlerCam.h* y *handlerCam.cpp* que se proporcionaron en la ***Práctica 8*** y cargar la escena creada anteriormente. Estos archivos se los utilizará para la inclusión del tracker, se trabajará de una forma similar a la realizada con el guante.
4. Probar los valores que se obtienen del sensor y hacer las pruebas necesarias para determinar la mejor referencia a utilizar. ¿Estos valores recogidos reflejan la realidad en la escena?¿Si te mueves 10 cm con el sensor, cuántos te mueves en la escena?¿Al hacer un movimiento con el

sensor, el objeto en la escena se mueve de acorde con el sensor?¿Qué expresión matemática utilizarías para arreglar este problema?

5. Con las coordenadas recogidas por el Tracker, mover el objeto que la escena posee, en las diferentes direcciones que el sensor provee (posición dinámicas (coordenadas cartesianas X, Y y Z), en tiempo real y orientación (azimuth, elevacion, y giro)).

REFERENCIAS

- Manual del Polhemus Tracker
- Sensores Polhemus

  http://www.siainteractive.com/sitio2/0212030303.htm

PRÁCTICA Nº. 11

INTERACCIÓN CON DISPOSITIVOS – POLHEMUS TRACKER

Interactuando con un escenario 3D

RESUMEN

En este taller, manipularemos un objeto 3D en la escena según las posiciones que se obtengan del sensor del Tracker.

OBJETIVO

- Capturar por medio de los sensores del tracker las posiciones y mover un objeto en la escena.
- Mover la Cámara de la escena con el dispositivo.

PRE-REQUISITOS

- Cumplir con la práctica 8 para conocer el manejo de la cámara en un escenario virtual.
- Realizar la práctica 10, la cual muestra la utilización del dispositivo Tracker.
- Crear un ambiente virtual. El estudiante podrá decidir si desea un escenario sencillo o un poco más elaborado.

TAREAS

1. Crear un proyecto en Visual Studio .NET versión 9.0 y configúrelo para hacer uso de las librerías de OSG, así como las del Tracker.

2. Modificar la ***práctica 8*** para hacer uso del dispositivo, incluya las líneas de código necesarias para utilizar el dispositivo. El escenario virtual puede ser el mismo o crear uno totalmente nuevo.

3. Controlar el movimiento de la cámara a través del sensor del tracker. Los movimientos a representar son: cámara adelante, cámara retroceder, cámara izquierda y derecha, cámara arriba y abajo.

4. Elegir un objeto en la escena y manipularlo.

Tareas Opcionales

1. Modificar la ***práctica 9*** de Picking para que la selección de las coordenadas del objeto se las realice con el tracker en lugar del puntero del mouse.

PRÁCTICA Nº. 12

INTERACCIÓN CON DISPOSITIVOS – POLHEMUS TRACKER y GUANTE

Interactuando con un escenario 3D

RESUMEN

Al finalizar este taller, usted habrá incluido en su aplicación más de un dispositivo los cuales interactuarán con la escena.

OBJETIVO

- Hacer uso de varios dispositivos en una sola aplicación. En esta práctica se utilizará el 5DT Data Glove, el Polhemus Tracker, Proyector 3D, Gafas 3D Pasivas y el 5DT Head Mounted Display.

PRE-REQUISITOS

- Cumplir con la práctica 8, la cual muestra la interacción del guante con un ambiente virtual.
- Realizar la práctica 11 que enseña el uso del tracker en un ambiente 3D.
- Tener creado un ambiente virtual un poco más elaborado, en el que consten varios objetos, que disponga de iluminación, texturas, un escenario que refleje algo se realidad. El escenario podría por ejemplo ser un paisaje completo, una casa que se puede ver en su interior o incluso un juego, todo depende de la imaginación del estudiante.

TAREAS

1. Crear un proyecto en Visual Studio .NET versión 9.0 y configúrelo para hacer uso de las librerías de OSG, las librerías del guante y las del tracker.

2. Integrar los dos dispositivos en la aplicación para realizar interacciones con el escenario.

3. Elegir un objeto del escenario virtual y hacerlo mover en diferentes direcciones, el objeto deberá moverse en conjunto con la cámara, para este efecto solo se utilizará el tracker. Por ejemplo si el objeto se desplaza hacia la derecha, la cámara también deberá moverse hacia esa dirección. Al estilo de los video juegos.

4. Elegir dos objetos de la escena, los cuales simularán una carrera. Un objeto será movido con el guante y el otro con el tracker.

5. Conectar el proyector 3D y utilizando las gafas pasivas, ejecutar la aplicación y visualizar los resultados al utilizar estos dos dispositivos. ¿El escenario virtual se observó igual que al utilizar el monitor de la PC? Explique la diferencia.

6. Conectar ahora el casco 3D y realizar los mismo que en el punto anterior. ¿Cuál fue la diferencia entre el proyector y el casco con respecto a la forma que se observo la escena?¿Cómo se vio la escena con los dos dispositivos?

Tareas Opcionales

1. Incluir la clase PickHandler en la aplicación, usando los dispositivos, el estudiante tendrá que elegir que dispositivo aplicar a cada acción que se vaya a realizar, la tarea será seleccionar (Picking) varios objetos dentro de la escena y cambiarlos de lugar (mover y colocar) para tratar de que el escenario se vea diferente.

2. Seleccionar (Picking) objetos y con un gesto cambiar el color y el tamaño de cada uno.

REFERENCIAS BIBLIOGRÁFICAS

[1] SANTAMARÍA PEÑA, MARTÍNEZ CÁMARA y SANZ ADÁN, "Motor gráfico interactivo para la visualización en tiempo real de gran volumen de información vectorial y texturizada", www.ingegraf.es/pdf/titulos/COMUNICACIONES%20ACEPTADAS/RV2.pdf, Último acceso: 29 de junio de 2010

[2] "Articulo Representación Tridimensional de Imágenes Termográficas", http://cde05.etse.urv.es/pub/pdf/1307pub.pdf, 29 junio de 2010

[3] Wikipedia, Definición de OpenGL, http://es.wikipedia.org/wiki/OpenGL, Último acceso:29 junio de 2010

[4] Grupo de Súper Cómputo, Reporte Técnico. "Tecnología de Realidad Virtual", http://telematica.cicese.mx/computo/super/cicese2000/realvirtual/Part3.html#III1, Último acceso: 29 de junio de 2010.

[5] Wikipedia, "Expresión matemática (Escala)", http://es.wikipedia.org/wiki/Escala_%28cartograf%C3%ADa%29, Último acceso: 29 de junio de 2010.

- Blog Tésis y Monografías, "Planteamiento del problema, justificación de la investigación, Objetivo General, Objetivo Específicos, Alcance y delimitación, Bases Teóricas, Bases Legales, Metodología de la Investigación", http://tesisymonografias.blogspot.com/, Último acceso: 15 de enero de 2010