8 de mayo de 2024

1er Obligatorio de Introducción a la Computación Gráfica

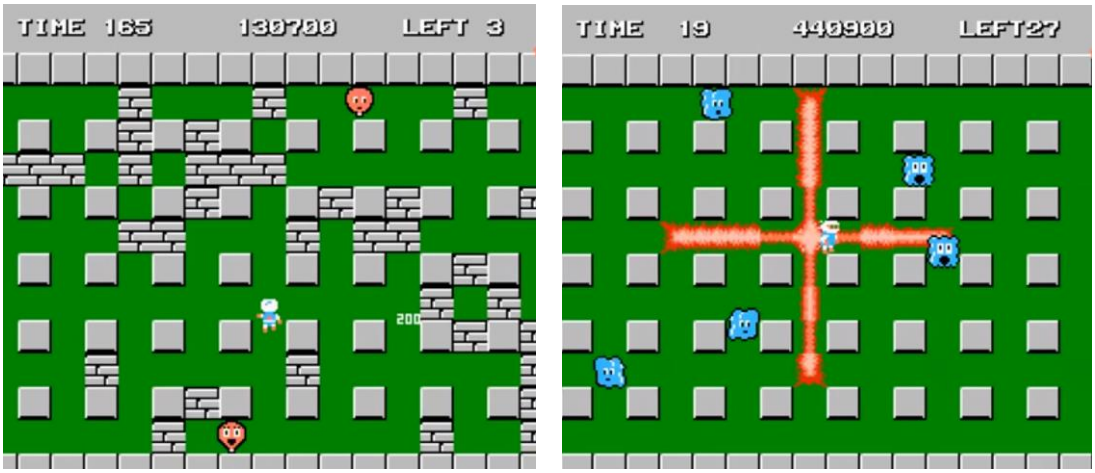
Videojuego sobre OpenGL

Introducción

El obligatorio se centra en conceptos vistos en el curso, así como en las bibliotecas OpenGL y GLU (para efectuar el render), y SDL (para manejo de dispositivos de entrada y creación de ventanas, entre otros conceptos).

Este es un obligatorio de Computación Gráfica, no de diseño de juegos. Se le dará mayor prioridad a todos aquellos aspectos que tengan que ver con el logro de efectos gráficos interesantes y asuntos relativos al "tiempo real", por encima de los aspectos relacionados con la estrategia, las reglas del juego o conceptos no afines a la computación gráfica.

Ejercicio (Bomberman)



Este ejercicio consiste en desarrollar el juego llamado Bomberman, que de origen pertenece a una franquicia de videojuegos estratégico-laberínticos desarrollada por Hudson Soft y actualmente por Konami. El juego original fue lanzado en 1983 para PC y posteriormente adaptado para la consola Family/NES en 1985. Es un juego 2D en el que el personaje principal es un hombre—un robot en la versión para NES—que debe atravesar un laberinto al tiempo que evita a diversos enemigos. Las puertas que conducen a otras salas del laberinto se encuentran bajo rocas que Bomberman debe destruir con bombas. Hay objetos que pueden ayudar a mejorar las bombas, como la habilidad de fuego, que mejora el alcance de las explosiones de las bombas.
Videos demostrativos del juego y links de interés:

a) Video: https://www.youtube.com/watch?v=CZ9Pu9Usk5o
b) Link: https://www.konami.com/games/bomberman/r/us/es/
c) Descrip. detallada (no es obligatorio seguirla): https://es.wikipedia.org/wiki/Bomberman_(videojuego)

No se evaluará el grado de reproducción o copia de los detalles estéticos del juego original. Se evaluará que la aplicación cumpla con los objetivos y características básicas del juego original, así como la innovación resultante del uso de elementos de computación gráfica. **La creatividad, especialmente en los efectos gráficos, influirá positivamente en el puntaje.**

Versiones posibles del juego en el pasaje a 3D

Existen diversas posibles variantes del juego 3D. En primer lugar, se podría realizar una implementación similar a la original, donde la jugabilidad se mantiene en un plano bidimensional pero los objetos tienen forma tridimensional. Esta variante permite modificar el ángulo de vista de la cámara e incluso una vista del personaje en primera persona.

Aquí se puede observar un ejemplo de esta variante: https://www.youtube.com/watch?v=5Xe0aPNz39Q. A esta variante podrían agregársele más de un mapa en diferentes niveles, de forma que el objeto pueda moverse libremente entre diferentes alturas, cada uno con sus laberintos, enemigos y obstáculos.

Otra variante posible consiste en un mapa con terreno 3D en el que el objeto principal puede moverse libremente entre montañas, valles, y otros tipos de relieve. Un ejemplo similar puede observarse en https://www.youtube.com/watch?v=h6q4jmTK_50. Otra variante aún con mayor libertad de movimiento podría ser un bomberman montado en un avión y con movimiento libre en todo el espacio 3D, siendo atacado por enemigos voladores y pudiendo disparar bombas para eliminarlos.

Tanto el personaje principal como los demás objetos pueden tener formas variadas. Podría ocurrir que otros objetos oculten la ubicación del bomberman, obligando al jugador a cambiar de punto de vista. Se está en libertad de implementar alguna de estas variantes u otra que ustedes crean conveniente, siempre manteniendo el uso de objetos tridimensionales.

Sobre la física del juego y explosiones

El movimiento del personaje principal no requiere de una física compleja, por lo que debería ser sencillo implementarla. De todas formas, de creerlo conveniente, se podría utilizar algún motor de física, como por ejemplo Bullet, Box2D, Chipmunk, etc. pero su uso deberá estar justificado por un resultado de calidad superior.

Respecto a la explosión de las bombas, idealmente debería hacerse con un sistema de partículas, típicamente implementados mediante cientos de pequeños cuadriláteros con texturas transparentes que aparecen en la posición inicial de la explosión, se mueven en direcciones aleatorias y desaparecen rápidamente. De no ser posible, podría simplemente iluminarse el terreno utilizando luces de OpenGL o nuevas texturas que representen una explosión en la zona afectada.

Requerimientos del videojuego

El conjunto de requerimientos que se detallan a continuación representa los mínimos necesarios para aprobar este trabajo obligatorio. La ausencia de alguno de los puntos especificados sin la correcta justificación significará la no aprobación del obligatorio (esto es, la pérdida del curso).

- El programa entregado debe ejecutar sobre el sistema operativo Windows, no debiendo requerir la instalación de ningún driver particular.
- El juego debe estar gobernado por el ratón (para la rotación de la cámara) y las flechas del teclado.
- Es posible rotar la cámara alrededor de la escena o del personaje.
- La tecla V debe permitir el cambio del modo de vista. Modos de vista de ejemplo serían: 1) Vista similar a la vista de la versión original, 2) vista centrada en el personaje (o detrás de él), 3) vista en perspectiva donde se pueda ajustar la posición de la cámara.
- Se puede detener el juego (tecla P), y salir (tecla Q).
- **Ajustes (settings)**: Se debe disponer de una interfaz para ajustar los siguientes parámetros:
  - Velocidad del juego: La velocidad de visualización del juego (jugar a cámara lenta o rápida), debe poder variarse manteniendo aproximadamente constante la cantidad de imágenes generadas por segundo. Para esto se debe multiplicar al tiempo transcurrido entre frames por un valor acorde a la velocidad del juego.
  - Estados posibles: wireframe (on/off); texturas (on/off); facetado/interpolado. Cada estado tiene dos valores y se los puede ajustar de forma independiente. Se debe poder cambiar de estado durante la ejecución.
  - Dirección y color de por lo menos una luz. Las direcciones y colores pueden tener valores predeterminados o se puede implementar una interfaz para poder ajustar sus valores.
- Utilización de al menos una textura (para los personajes, el fondo, objetos, etc.).

- Cargado y renderizado de al menos un modelo 3D (personajes, objetos). Para esto se pueden utilizar bibliotecas externas, siempre y cuando sean portables.
- Game HUD básico (puntaje, tiempo, etc.) dibujados con una proyección ortogonal.
- Además de entregar todos los códigos fuentes, se deberá entregar una carpeta que contenga únicamente: el ejecutable, texturas y modelos, y bibliotecas utilizadas. El ejecutable deberá obligatoriamente funcionar haciendo doble-clic en una máquina Windows. Dicha carpeta no debe exceder los 20 MBytes. Se recomienda almacenar las imágenes en formatos con compresión (JPG, png, gif, etc.). Notar que no se considera los dlls para calcular el peso total.

Requerimientos opcionales

Los requerimientos que se detallan a continuación representan elementos que tienen un valor agregado, pero no influyen en la aprobación del trabajo obligatorio. La realización de requerimientos opcionales no exime la realización de algún punto especificado como obligatorio.

- Podría haber más de un escenario (o nivel) posible. El movimiento de los objetos de la escena (pinchos, fuego, etc.) podrá ser predefinido o en base a un cómputo aleatorio (que defina, por ejemplo, la velocidad y momento de aparición de cada objeto, o el número de objetos total).
- El nivel del juego puede ser definido en un archivo XML. Por ejemplo, definiendo la ubicación de los cubos, propiedades de estos, etc. Se recomienda utilizar alguna de las siguientes bibliotecas:
  - TinyXML (http://sourceforge.net/projects/tinyxml/)
  - PugiXml (http://pugixml.org/)
- Implementar un editor de niveles.
  - Como resultado se espera que el editor pueda guardar un archivo XML que luego será utilizado como uno de los niveles del juego.
- Implementar efectos gráficos, por ejemplo, utilizar un sistema de partículas para la explosión de una bomba, para ir marcando el recorrido del personaje, festejo cuando "muere" un monstruo, festejo por un récord de puntos obtenidos. También puede haber cambio de colores o movimientos especiales como duelo cada vez que se pierde una vida.
- Simular sombras de los personajes con una textura oscura debajo de los mismos.

Recursos a utilizar

Las salas 312 y 315 disponen de máquinas con Windows. A su vez, en cualquier PC o notebook razonablemente moderno van a poder desarrollar y correr un juego de estas características. Se podrá utilizar OpenGL 1.2 o 1.3, y GLU.

El paquete SDL de desarrollo puede ser descargado desde la página oficial de la librería. Se puede utilizar SDL 1.2 o 2.0 indistintamente. `http://www.libsdl.org/`

Se puede utilizar el IDE de Visual Studio. También se pueden utilizar otros compiladores/IDEs C++:

- Mingw dentro del IDE Code::Blocks
- Mingw dentro del IDE Bloodshed Dev-C++

Desarróllenlo en modo RELEASE para así acelerar su funcionamiento.
Si no disponen de ninguna máquina Windows, de forma excepcional (previo aviso) les autorizaríamos a utilizar Linux. Los drivers de OpenGL en Windows suelen tener mejor rendimiento que los equivalentes de Linux.

¿Trabajo individual o colectivo?

Se deben formar grupos de 3 estudiantes (en algún caso 2 estudiantes, pero con la debida justificación). La formación de los grupos queda por cuenta de los estudiantes. Si fuese necesario, pueden utilizar el foro para buscar compañeros y así poder armar los grupos.

Defensa del obligatorio (jueves 13 de mayo)

La defensa consta de 2 partes, una "interactiva" y otra en base a la documentación presentada.

- **Defensa interactiva:** Se realizará en la sala 315. La lógica de la defensa es que se presenta cada grupo (con todos sus integrantes) y entregan el ejecutable y la documentación (máximo 8 páginas sin contar la carátula ni el índice). Los grupos realizan una breve demostración de sus aplicaciones, que deben ejecutar preferentemente sobre el sistema operativo Windows. Es importante que prueben sus aplicaciones antes de la defensa en diversas máquinas para evitar contratiempos innecesarios. También podrán comentar algunos aspectos de la aplicación y de la documentación. Los docentes harán preguntas sobre el trabajo realizado.
- **Documentación:** Se hará entrega al docente la documentación que consiste en:
  - Una documentación en la que se explica:
    1. Arquitectura de la aplicación.
    2. Explicar cómo resolvieron el modelado de objetos, los temas de iluminación, textura, y otros aspectos que hacen a los temas gráficos de la aplicación.
    3. Agregar un capítulo de referencias donde se mencione la documentación utilizada. Si se utilizaron partes de código fuente extraído de otras aplicaciones, explicar cuáles fueron y de dónde se extrajeron. Lo mismo para ideas extraídas de otras aplicaciones.
    4. Versión utilizada de SDL.

- **Entrega**: Cada grupo entrega en la defensa un zip con: código fuente, ejecutable, archivo de la documentación. Recuerden (lean más arriba) las características de la carpeta con el ejecutable, modelos y textura.

En base a la evaluación de ambas instancias se definirá la nota final del obligatorio.

Parece obvio, pero no está permitido utilizar códigos que resuelvan el obligatorio de forma trivial.

¡Concurso!

Como en años anteriores, este año se realizará un concurso entre los trabajos presentados. **El concurso no influye en la nota y no es de participación obligatoria**. La mecánica del concurso consiste en que se pondrán los ejecutables en el sitio web de la asignatura, con un código de forma que nadie sepa quiénes son los autores (a menos que Uds. hagan aparecer sus nombres en la salida gráfica). Luego, al final del curso se habilitará una encuesta, donde todos los estudiantes de la asignatura podrán votar de forma anónima e individual a los tres mejores juegos. En esa votación, al mejor juego votado se le asignan tres votos, al segundo dos votos, y al tercer juego se le asigna un voto.

Sugerencias

- Utilizar la documentación provista en `www.opengl.org`
- Examinar videos del juego para extraer ideas que aporten a lo gráfico.
- Utilizar el EVA para intercambiar opiniones sobre problemas, dudas, en lo que respecta a la letra, a la programación en OpenGL, etc.