Proyecto Game Engine - Hito 2

Introducción

El propósito de este hito es concretar una estructura inicial del game engine para obtener una aplicación interactiva en términos de gráficos y control por parte del usuario jugador. Siempre se deben tomar decisiones de diseño en dirección al tipo de juegos que se desea implementar con el game engine. Pueden presentarse dificultades técnicas en términos de gráficos y dispositivos de interacción humana, usted deberá elegir si utilizar/continuar con OpenGL/GLFW, utilizar un sistema de bajo nivel distinto, o utilizar una biblioteca dedicada como Ogre3D u otro similar.

Importante

- La nota final se calcula sumando los puntajes obtenidos a un punto base.
- El puntaje de cada ítem de la rúbrica será evaluado observando el código fuente y analizando un ÚNICO demo funcional. Toda funcionalidad implementada DEBE ser ejercitada en el mismo demo o no recibirá puntaje.
- Para acceder a una columna de puntaje, se debe cumplir con todo lo pedido en dicha celda, y en todas las celdas anteriores que otorgan menor puntaje.
- En caso de que alguna categoría no alcance un puntaje mínimo de 0.5, no se hace efectivo ningún puntaje de la columna bonus.
- Si supera el umbral de nota 7, el puntaje restante se guarda como puntaje extra para el hito 3, con un máximo de 1.0 punto de nota adicional.
- Se debe mantener la estructura de código lograda en el hito 1, continuando con la misma dinámica de mantener el engine en una biblioteca separada (estática o dinámica), y generar el release en una rama o tag especifico de nombre “hito 2”. Esto es un requerimiento mínimo forzado que no otorga puntaje. Si no hay un correcto uso de control de versiones, implica nota 1 en el hito.
- Idealmente el trabajo se debe realizar en un repositorio público con licencia open source (MIT es la más utilizada), si desea que su proyecto no sea open source (y a la vez sea privado), envíe un mail al profesor, y deberá otorgar permisos al equipo docente para poder revisar su repositorio.
- Puede estudiar código disponible en internet y de los repositorios públicos de sus compañeras/os, sin embargo se espera un trabajo original de cada estudiante orientado a cada tipo de juego en específico. Como siempre, copias serán sancionadas con el protocolo de la escuela.
- Si no se incluye alguno de los entregables, o requerimientos básicos indicados en el hito (ejemplo: no adjuntar la autoevaluación o video), se aplicará un descuento según criterio del profesor.

Video

- Debe adjuntar a su entrega, un video donde se ilustre el proceso de descarga de los archivos del repositorio, su compilación y la ejecución de demo funcional.
- Este ítem es especialmente importante si utiliza un sistema operativo distinto de Windows o Linux, y un lenguaje de programación distinto de C++.
- Edite el video para que en ningún caso tarde más de 5 minutos, pero mientras más breve, mejor.
- Adjunte un link a Youtube o suba el video a u-cursos en su entrega (u-cursos tiene restricción de tamaño de archivo).

Autoevaluación

- Con el fin de facilitar la evaluación, y a la vez dejar en claro lo que usted ha implementado y lo que no, se solicita que complete la rúbrica realizando una autoevaluación, indicando en cada celda lo que implementó para obtener dicho puntaje, y el archivo/función/clase donde habría que buscarlo. En material docente encontrará un archivo .odt con la rúbrica para completar su autoevaluación.

Entregables

- u-cursos:
  - Autoevaluación en pdf.
  - Comentario de entrega:
    - commit de su repositorio remoto git que será evaluado.
    - Link de Youtube a video de descarga/compilación/ejecución, puede ser privado.
  - Alternativamente al link de Youtube, puede adjuntar el video a su entrega si es que u-cursos lo permite dada su restricción para archivos muy grandes.
- Repositorio Git:
  - Archivo Readme.md con instrucciones claras y breves de compilación y ejecución (esta información debe estar actualizada al hito actual).
  - Licencia open source (comunicar vía e-mail al profesor de cátedra si es que desea que su código no sea público o no sea open source).
  - Código fuente del game engine y del demo.
  - Se revisará la rama o tag de nombre “hito 2”.

Presentación

- Debe asistir a la sesión de presentaciones en la fecha y hora indicada en la última versión de la planificación del curso.
- Si por razones de fuerza mayor no puede asistir, comuníquelo vía e-mail al profesor lo antes posible. Cuando sea la fecha, deberá enviar su presentación vía link a video privado de Youtube.
- La presentación se evalúa simplemente de manera Logrado/No logrado, lo importante es que explique y presente muy brevemente sus avances desde el hito anterior.

Rúbrica Hito 2 – CC5512

Criterio \ Puntaje	0	0.5	1.0	1.4	1.8	Bonus
Game Object Model	No se entrega o no se implementa o se presentan inconsistencias graves. Se implementa uso de mouse+teclado ó joystick en un mínimo funcional suficiente para una aplicación tipo.	Diseño de clases/estructuras claras para el manejo de Game Objects. Se maneja la misma estructura de Game Objects tanto para Tool-Time como en Run-Time. El demo muestra una cantidad limitada de Game Objects creados en la inicialización del Game Engine.	Modelo de game objects implementado permite añadir y eliminar game objects/componentes en run-time según se requiera.	Modelo de Game Objects en Run-time es distinto al de Tool-Time y se ejecuta el proceso de conversión/optimización de Tool-Time a Runtime.		
Dispositivos de Interacción Humana		Se implementa uso de mouse+teclado ó joystick en un mínimo funcional suficiente para una aplicación tipo.	Se implementa capa de abstracción de HID.	Se permite el manejo de entradas analógicas. Se manejan correctamente conexiones y desconexiones de dispositivos. Notificando al usuario del problema.	Se implementan al menos una de las siguientes funcionalidades: - Player remapping - Smooth control - Chords - Sequences	**(+0.5)** Se implementan al menos una de las siguientes funcionalidades (distinto del anterior, teniendo un total de 2): - Player remapping - Smooth control - Chords - Sequences **(+0.3)** Hay soporte para al menos 2 jugadores locales en distintos dispositivos. **(+0.8)** Manejo de al menos un HID distinto y de mayor complejidad como: acelerómetros, cámaras, micrófonos, etc.
Rendering Engine		Se permite la visualización de game objects en una ventana. Implementa al menos uno de los siguientes sistemas: - Sistema de manejo de sprites - Sistema de manejo de modelos 3D	Se implementa un command buffer con las instrucciones de rendering, de manera que se permitan operaciones de optimización antes de ejecutar las draw calls.	Implementa un sistema de optimización para rendering eficiente de una cantidad masiva de objetos en la escena. Ejemplos: - Clipping - Sistema de portales - Sistema de Antiportales - Grafo de escena para determinación de visibilidad.	Implementa un segundo sistema de optimización para rendering eficiente de una cantidad masiva de objetos en la escena. Misma lista enunciada en celda anterior. ó Implementa al menos uno de los siguientes sistemas (distinto del anterior): - Sistema de manejo de sprites - Sistema de manejo de modelos 3D * Modelos 3D deben estar texturizados.	**(+0.5)** Sistema eficiente en el intercambio de información CPU-GPU. Ejemplos: - Batch rendering - GPU instancing **(+0.5)** Sistema de partículas CPU **(+0.5)** Se permite que el usuario del engine pueda proporcionar sus propios shaders. Debe proporcionar un par de ejemplos. **(+0.5)** Sistema de Iluminación que se puede habilitar/deshabilitar y permite configurar hasta un máximo de luces. **(+0.3)** Sistema de renderizado de texto **(+0.7)** Implementación de material PBR.
Maps		Existe concepto de mapa o nivel al cual se agregan los Game Objects.				**(+0.5)** Capacidad de cargar y descargar de memoria distintos niveles en runtime.
Runtime Object Model		Engine se preocupa de actualizar los distintos game objects y sistemas implementados de manera interna, sin darle visibilidad al usuario del engine.				**(+0.4)** Actualización de los distintos sub-sistemas de manera ordenada y eficiente según se ve en cátedra.