UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

TGC

Técnicas de Gráficos por Computadora

Trabajo Práctico
Snipers

Nombre del Grupo: CEGA

Integrantes
-Joaquin, Javier [120.383-6]
-Monsech Paez, Alexis [146.805-4]
-Manopella, Santiago [122.225-9]
-Suarez Lissi, Tomás [137.767-0]

Objetivo

El juego se trata de un escenario en primera persona en donde el jugador principal es perseguido por otros enemigos que intentan matarlo. Esto sucede cuando los enemigos lo logran interceptar.
El jugador dispone de un rifle con una mira telescópica que utiliza para matar a los enemigos y así evitar que lo intercepten. El escenario también posee barriles explosivos que al explotarlos le quitan vida a los enemigos que están cerca del mismo.

El objetivo del juego es tratar de sobrevivir la mayor cantidad de tiempo y sumar la mayor cantidad de puntos. La complejidad del mismo aparece en que a medida que pasa el tiempo, aparecen cada vez más enemigos.
Cada vez que el jugador muere, vuelve a aparecer en el lugar inicial y se le resta una vida. El juego termina cuando al jugador se le acaban las vidas (dispone de 5 vidas).

Controles

El juego cuenta con los siguientes controles:

+ W = Avanzar hacia adelante
+ D = Avanzar hacia la derecha
+ A = Avanzar hacia la izquierda
+ S = Avanzar hacia atrás
+ L = Bloquear/Desbloquear el mouse
+ SHIFT = Correr
+ Botón izquierdo mouse = Disparar
+ Botón derecho mouse = Habilitar/Deshabilitar mira telescópica
+ Movimiento mouse = Rotación de la cámara

Optimización

Como técnica de optimización para el rendering de las distintas mallas se eligió utilizar una grilla regular para la división del terreno. Se eligió usar una grilla por sobre otras técnicas de optimización debido a que los elementos del escenario están repartidos equitativamente en el mismo, y aplicando otros métodos como el QuadTree, no se encontraron diferencias en la performance ni en la división del escenario.
El tamaño de la grilla es fijo y el valor se decidió luego de intentar con valores más grandes y más chicos, haciendo foco en la performance.

La grilla también se utiliza para optimizar las colisiones entre los distintos objetos calculando cuáles son los nodos de la grilla que están cercanos y calculando las colisiones sólo en estos; esto se aplica para los enemigos y el jugador principal.

Colisiones

Hay varios tipos de colisión que se producen en el ejemplo:

Jugador-Objetos
Colisiones entre el jugador principal y los distintos objetos del escenario. En caso de que se produzca una colisión, el jugador no puede avanzar hasta que cambie su rumbo.
En el escenario hay tanto objetos colisionables (árboles, barriles, terreno) como no (pasto).

Jugador-Enemigo
Al producirse una colisión entre el jugador y un enemigo, el primero pierde una vida y vuelve a empezar.

Disparo-Enemigo
Las colisiones entre un disparo y enemigo le quitan vida. El enemigo puede ser impactado en la cabeza o en el cuerpo causándole distinto daño. Cuando el enemigo se queda sin vida, muere.

Disparo-Barril
Cuando un disparo colisiona un barril, el segundo explota causando daño a los enemigos que se encuentren en un radio determinado. Si están lo suficientemente cerca al barril, morirán.

Enemigo-Objetos
Cuando los enemigos colisionen con objetos, estos cambiaran su rumbo para esquivarlo.

Enemigo-Enemigo
Cuando un enemigo colisione con otro, uno de ellos cambiará su recorrido para esquivar al otro enemigo.

Clases

Se crearon varias clases, con distintos objetivos, para la resolución del trabajo práctico:

.\Snipers
Clase principal del ejemplo. Controla la creación, render y dispose de los elementos principales del ejemplo.

.\Interfaces\IRenderable
Interfaz para hacer un objeto “rendereable”. El ejemplo soporta una fase de rendering de elementos de UI, que es implementada en el método RenderUI de esta interfaz.

.\Interfaces\IUpdatable
Interfaz para hacer un objeto “updateable”

.\Scenes\PlayScene
Clase principal del escenario. Contiene la información y maneja la creación de los objetos presentes en el mismo. Aplica shader de post-procesamiento para efectos en los árboles y en el pasto

.\Scenes\SimpleTerrain
Clase utilizada para crear el Heightmap utilizado para los límites del escenario.

.\Scenes\VideoScene
Clase utilizada para reproducir el video inicial.

.\Units\ColisionesAdmin
Clase que administra todas los tipos de colisiones en el ejemplo. Esta clase fue implementada con un patrón “singleton”.

.\Units\Enemigo
Clase que contiene toda la información acerca de un enemigo e.g. vida, animación, etc.

.\Units\EnemigosAdmin
Clase que controla la creación, actualización y muerte de los enemigos en el escenario.

.\Units\FpsCamera
Clase que administra la cámara en primera persona utilizada.

.\Units\GrillaRegular
Clase utilizada para crear y recorrer la grilla utilizada para optimización.

.\Units\GrillaRegularNode
Clase que contiene la información de la estructura de los nodos de la grilla.

.\Units\Player
Clase principal que representa al jugador en primera persona. Contiene la información sobre el movimiento, el arma, los disparos y los efectos de la mira del arma.

Efectos

Se utilizaron shaders para generar distintos efectos en el trabajo práctico:

Efecto mira sniper

Para poder realizar el efecto en la mira del arma se utilizó un “Post-Processing” shader en el cual se incluyen los efectos:
+ Aberración cromática
+ Tinte azul
+ Distorsión del lente por un factor K & K cúbico

La combinación de estos efectos produce la sensación de estar mirando a través de un lente por la curvatura de la lente y la distorsión de los colores.
La distorsión del lente está basada en el algoritmo de SSontech, mientras que la aberración cromática fue implementada por el grupo.

Efecto viento

El efecto del viento tanto en los árboles como en el pasto está realizado con un “Vertex” shader utilizando una curva que utiliza senos y cosenos con un período de repetición espaciado y una forma en particular que se incrementa y decrementa por momentos. Las armónicas elegidas fueron calculadas en base a las fuerzas promedio que se encuentran en un árbol en la vida real, esto produce una oscilación suave. Cada árbol tiene aplicada una curva diferente, calculadas “corriendo el tiempo” de la misma, dando una sensación más realista.
Los valores de oscilación son aplicados radialmente a los vértices del árbol, usando un corrimiento porcentual para simular el tallo.
La curva utilizada por el shader está basada en el libro GPU Gems.

Efecto neblina

El efecto de neblina esta realizado con un “Pixel” shader para los árboles, y con el fog blending stage del pipeline de la placa de video para el resto de la escena. Ambos efectos son lineales, se aplican según la distancia con la posición de la cámara (cuanto más lejano, más pronunciado el efecto).