Técnicas de gráficos por computadora

Trabajo Práctico

Grupo: Twisted Chano

Integrantes:

- Francisco Enzo Di Giorgio
  - Legajo: 149.744-3
  - Mail: f.e.digiorgio@gmail.com
- Santiago Martinez García
  - Legajo: 152.664-9
  - Mail: santymartinez96@gmail.com
- Emmanuel Sánchez
  - Legajo: 114.802-3
  - Mail: esanchez.utn@gmail.com

Curso: K3051

Año: 1er Cuatrimestre, 2017

Días: Jueves

Turno: Noche

Profesor: Ing. Rodrigo García

Tutor: Ing. René Juan Rico Mendoza

Objetivo:

El objetivo del trabajo práctico es desarrollar un ejemplo creativo basado en el juego Twisted Metal, utilizando la plataforma de diseño provista por la catedra (TGC Viewer) en conjunto con las nociones teóricas aprendidas durante la cursada.

Idea básica del juego:

Twisted Chano es un videojuego en el cual un jugador y su auto se enfrentan a otros 4 autos manejados por la computadora en una pelea que se desarrolla tanto en un ambiente urbano, con pocos espacios y muchos obstáculos, como en un ambiente más abierto con obstáculos ocasionales y mucha mayor libertad de movimiento. Para lograr ser el ganador, el jugador deberá chocar a sus rivales hasta eliminarlos mientras que evita ser chocado por ellos y así poder llegar con vida al final del juego.

Funcionalidades obligatorias:

- Construir un escenario en donde hay un auto manejado en tercera persona por el usuario dentro de un estadio grande y cerrado. Los autos no pueden salir de este estadio.
- El auto debe poseer los siguientes movimientos:
  - Acelerar y Frenar.
  - Doblar.
  - Saltar
- Para cada uno de los movimientos anteriores, las ruedas del auto deben rotar acorde a ellos.
- El auto debe tener detección de colisiones contra los objetos del escenario (paredes, otros autos, etc). La colisión deberá tener en cuenta los siguientes aspectos:
  - Ser lo más precisa posible.
  - Adaptarse de la mejor manera posible al volumen de la malla del auto.
  - Tener en cuenta la rotación de la malla.
  - No podrá ser utilizada una solución de detección de colisiones simple con AxisAlignedBoundingBox o BoundingSphere (los obstáculos sí pueden ser implementados con AxisAlignedBoundingBox)
  - Al chocar contra un obstáculo, se debe dar una respuesta de colisión apropiada, según la velocidad y ángulo de choque del auto.
- La cámara debe seguir al coche principal (los laterales no pueden afectar la visión)
- Se debe mostrar el tiempo y el marcador por pantalla.
- El auto debe tener efectos gráficos.
  - Reflejos (Shader de Environment Map).
  - Partículas al chocar.
  - Humo del motor o escape.
  - Deformar la malla del auto al chocar.
- Poder moverse marcha atrás con el auto y doblar marcha atrás.
- El estadio debe tener mínimo una luz dinámica de tipo Point Light y generar sombras.
- Incorporar otro auto con Inteligencia Artificial que persigue al primero e intenta chocarlo.

Funcionalidades opcionales:

- Agregar power ups (items) que le den al jugador habilidades temporales:
- Que los autos tengan dos luces dinámicas en sus faroles delanteros y traseros.
- Agregar más competidores con Inteligencia Artificial y que estos intenten evitar colisiones con los objetos del escenario (exceptuando a los otros autos)
- Agregar un velocímetro en pantalla que refleje los cambios en la velocidad del auto del usuario.
- Agregar sonidos a los distintos eventos que se producen durante el juego (Inicio, auto andando, choques, uso de power ups, etc)

¿Cuando finaliza el juego?

El juego termina cuando uno de dos eventos ha sucedido, o bien el jugador perdió toda su salud y quedo eliminado (derrota), o el jugador logro eliminar a los 4 autos manejados por la IA y mantenerse vivo (victoria).

Herramientas utilizadas:

- Microsoft Visual C# 2017
- DirectX SDK
- Framework de la catedra (TGC Viewer)
- 3D Max

Diseño del escenario:

Se creó una ciudad compuesta por edificios y calles junto con sus alrededores utilizando el TGC Tools y en base a ese mapa, luego, se agregaron una variedad de objetos (postes, distintos tipos de árboles, rocas, etc). Para poder mejorar el rendimiento del juego, cada uno de estos tipos de objetos tenía un mesh especifico que luego se utilizaba para crear cada instancia de cada objeto. Para poblar el mapa se utilizó una distribución aleatoria, asegurando que el mapa pueda cambiar de juego en juego, siempre chequeando antes de la colocación de un objeto para ver que no colisionara con otro ya existente.

Detección de colisiones:

Para lograr poder detectar colisiones entre autos y objetos, o de autos entre sí, se utilizaron Axis Aligned Bounding Box en el caso de los objetos del escenario (ya que su rotación no va a cambiar durante la ejecución del juego), y Oriented Bounding Box para los autos (cuya rotación si nos interesa que se vea reflejada en el Bounding Box), estos a su vez fueron divididos en partes para poder detectar de forma más precisa cuales partes del auto se encontraban colisionando y cuáles no.

En cuanto a la respuesta a las colisiones, esta varía dependiendo tanto de la dirección del choque como de la velocidad que tenía uno u ambos autos en el momento anterior a que se produzca este choque.

Optimización:

Para mejorar el rendimiento del juego y aumentar la cantidad de FPS, se utilizó un Quadtree, el cual nos permite dividir nuestro mapa de juego en zonas, y luego al comparar estas zonas con el Frustum, podemos evitar hacer cálculos de colisión y renderizar todos aquellos objetos que no van a ser visibles en la pantalla del jugador.

Inteligencia Artificial:

Toda conducta realizada por cualquiera de los 4 coches rivales es responsabilidad de la computadora, es decir, los coches no tienen ninguna conducta ni ruta predeterminada. Para lograr esto, durante cada ciclo del Update, los autos rivales realizan una comparación entre su dirección de movimiento actual y su vector de distancia con respecto al auto del jugador (este último se normaliza). Mediante esta comparación la maquina puede determinar si está en curso de colisión con el jugador o no, si es así, el auto comienza a acelerar con el objetivo de llegar a chocarlo. En caso de que su dirección actual no le permita alcanzar al jugador, el auto cambia su dirección rotando en círculos alrededor del lugar donde se encontraba hasta encontrar al jugador y ahora si dirigirse hacia él.

Por otro lado, los autos de la IA poseen una forma básica de prevenir colisiones con los objetos del escenario (plantas, rocas, postes, etc.) que consiste en utilizar un rayo que es lanzado desde la posición actual del auto y con la misma dirección que este, y en caso de encontrar que el rayo colisiono con un objeto que se encuentra a una corta distancia, la IA altera mínimamente la dirección de su auto y continúa avanzando.

Partículas:

Se utilizó la clase ParticleEmitter para generar el humo del caño de escape y de las colisiones del auto con otros objetos o autos.

Environment Map:

Se aplicó la técnica de Environment Map para poder generar un efecto de reflejo en los autos. Se dibujó para cada lado del cubo la escena y luego se aplicó sobre cada uno de los autos de acuerdo a la posición de la cámara. Para poder optimizar un poco la velocidad se renderiza solo en FPS que son módulos de 3.

Shadow Map:

Se aplicó la técnica de Shadow Map para poder generar un efectos de sombra en los objetos del escenario y autos. Se posicionó la luz que genera las sombras en la misma posición del auto, para que cuando este se moviera, se generaran sombras dinámicas.

Para hacer esto, se hizo una primera pasada para generar las sombras de los objetos teniendo en cuenta la posición de la cámara y de la luz, luego se hizo otra pasada para renderizar los objetos. También se incorporó otro foco de luz arriba del auto para poder generar la sombra al saltar el auto.

Diffuse Lights:

Se aplicó la técnica de Diffuse Lights para poder generar un brillo en la luna que se encuentra en el medio del escenario, tocando el techo (por un tema de cómo armamos el escenario cerrado solo se ve cuando se utiliza la cámara libre).

Controles:

W o Flecha Arriba: Acelerar el auto.

S o Flecha Abajo: Frenar el auto.

A o Flecha Izquierda: Girar el auto en sentido anti horario.

D o Flecha Derecha: Girar el auto en sentido horario.

Barra Espaciadora: Saltar.

F: Muestra los bounding box para todos los objetos del escenario.

F1: Cambio de cámara entre una cámara fija detrás del auto del jugador y una cámara libre.