



Universidad Nacional Autónoma de México



Computación Gráfica Avanzada

Proyecto Final
Peaceful Hill
Manual Técnico

Profesor: Martell Ávila Reynaldo

Alumno: Galván Bazán Raúl

Grupo: 1

Semestre: 2021-1

Índice

1. Introducción	1
2. Objetivo	1
3. Desarrollo	1
3.1. Obtención de modelos	1
3.2. Animaciones	1
3.3. Terreno	1
3.4. Generación del escenario	2
3.5. Colisiones	2
3.6. Cámara	4
3.7. Sistema de partículas	4
3.8. Neblina	4
3.9. Luces	4
3.10. Instanciamiento de objetos	5
3.11. Comportamiento del enemigo	5

1. Introducción

There was an introduction here. It' gone now

2. Objetivo

El objetivo principal del proyecto fue la creación de un juego de terror/supervivencia utilizando el lenguaje de programación C++, la biblioteca openGL y el lenguaje de sombreado de openGl (OpenGL Shading Language o GLSL).

Dentro del juego, el jugador debe recorrer un escenario en búsqueda de artículos que ayuden a su supervivencia y evitar morir por el daño producido por los enemigos.

3. Desarrollo

3.1. Obtención de modelos

Todos los modelos fueron obtenidos de internet. En algunos casos los modelos se modificaron para cambiar sus texturas, su escala, el número de vértices y polígonos, o para agregar animaciones. Para modificar los modelos se utilizó el software Blender 2.9.

3.2. Animaciones

Para la animación de los modelos se hizo uso Mixamo (<https://www.mixamo.com>). Mixamo tiene una catálogo de animaciones predeterminadas que pueden ser aplicados a un modelo, la única condición es que dicho modelo sea antropomorfo.

Algunos modelos fueron animados usando el software Blender 2.9.

3.3. Terreno

Se utilizó el software GIMP 2.10 para generar una imagen que sirvió como mapa de alturas Fig. 1. Este mapa consiste de una imagen en escala de grises, en donde el valor negro representan la altura más baja, mientras que el valor blanco la altura más alta, los valores de grises representan alturas intermedias.

El mapa de alturas fue utilizado en el proyecto para generar un escenario que simulara un terreno montañoso con diversos pasajes y montículos.



Figura 1: Mapa de alturas utilizado en el proyecto

Se utilizó el software GIMP 2.10 para generar una imagen que sirvió como mapa de texturas Fig. 2. Se utilizó el mapa de texturas en conjunto con el mapa de alturas para texturizar el escenario creado con el mapa de alturas. Este mapa consta de una imagen en donde cada textura a utilizar se identifica con uno de los valores rojo (rgb(255, 0, 0)), verde (rgb(0, 255, 0)), azul (rgb(0, 0, 255)) o negro (rgb(0, 0, 0)).

Además se necesitó crear un shader para poder combinar el mapa de alturas y texturas de tal forma que se obtuviera un terreno completamente texturizado.

3.4. Generación del escenario

Para desarrollar el escenario fue necesario colocar diferentes modelos en el mapa del diseño. Para esto se utilizó el software Blender en el que se colocaron las imágenes del mapa de alturas y mapa de texturas Fig. 3. Estas imágenes se dimensionaron de tal forma que las coordenadas en Blender correspondieran a las coordenadas en openGL. De esta forma fue posible colocar los objetos en el escenario construido en Blender para obtener sus posiciones y así poder exportarlas a openGL.

3.5. Colisiones

Se utilizaron cajas de colisiones de tipo OBB (Oriented Bounding Box) para diversos motivos.

- En el personaje principal para interactuar con los diferentes elementos del juego.
- En los límites del escenario y sobre objetos para definir por dónde puede o no cruzar el personaje principal.

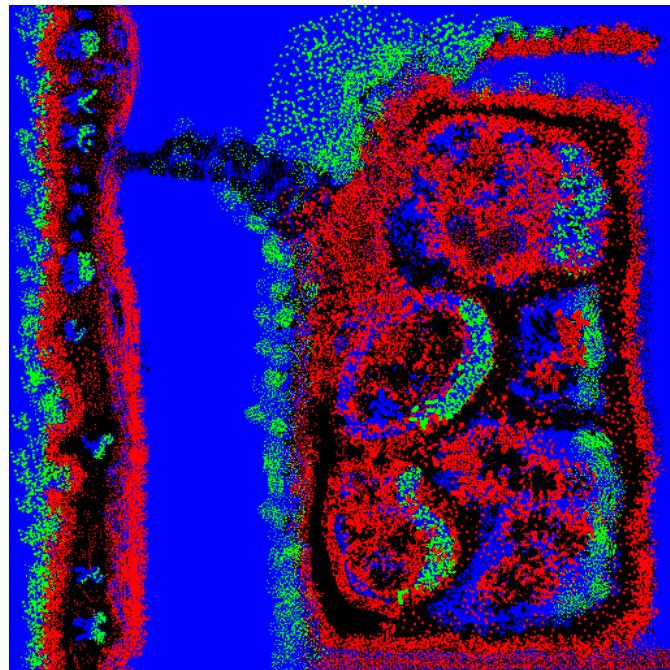


Figura 2: Mapa de alturas utilizado en el proyecto

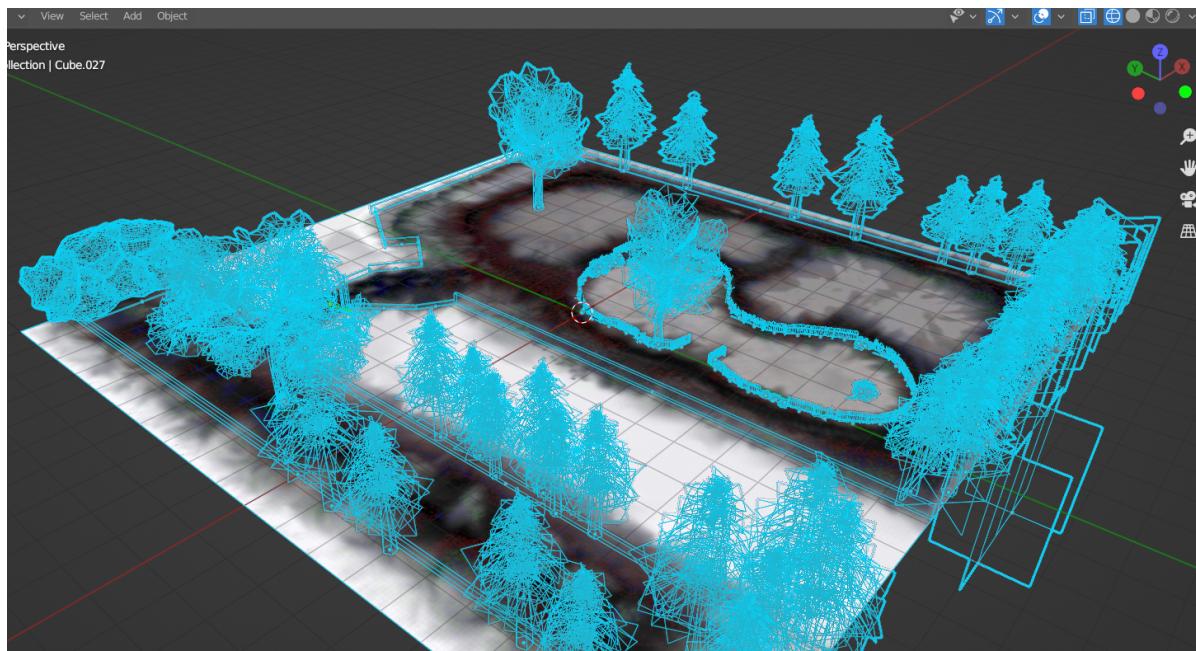


Figura 3: Mapa del escenario creado en Blender para obtener las posiciones de los objetos

- En el enemigo del juego, para determinar cuándo impacta en el personaje principal. Al impacto, el enemigo hace daño.
- En los artículos de curación para determinar cuando el personaje principal está junto a ellos y puede tomarlos.

Para manejar las colisiones, las cajas de colisiones, así como su posición, se colocaron en una estructura de datos conocida como mapa. De esta forma fue posible mantener un registro de las cajas de colisiones. Sabiendo la posición de las cajas es posible utilizar un algoritmo de detección de colisiones para saber qué

objetos colisionan con otros.

3.6. Cámara

La cámara principal del proyecto consiste en una cámara en tercera persona. Esta cámara tiene las siguientes características:

- Sigue al personaje principal.
- Puede girar alrededor del personaje principal. Está limitada en su rotación vertical.
- Puede hacer acercarse o alejarse del personaje principal. Este zoom está limitado para evitar alejarse o acercarse demasiado al personaje.
- Tiene un reset de cámara que devuelve a la cámara a una posición inicial.

Adicionalmente se implementó una cámara en primera persona, pero su utilización al momento es limitada.

Ambas cámaras son manipuladas utilizando el ratón.

3.7. Sistema de partículas

Se utilizó un sistema de partículas retroalimentado para generar un efecto de fuego que sale de la boca del enemigo del juego Fig. 4. Para la generación del sistema de partículas fue necesario crear un shader en donde se define su comportamiento y posicionamiento de las partículas (vertex shader) y el color de las partículas (fragment shader). Es sistema de partículas se encuentra posicionado en la boca del enemigo y sigue el movimiento del mismo. Se hizo uso de la técnica de blending para generar transparencia en las partículas.



Figura 4: Enemigo del juego con sistema de partículas en la boca

3.8. Neblina

Se implementó un efecto de neblina haciendo uso del buffer de profundidad. Mientras los objetos en el escenario se alejan de la cámara, van perdiendo visibilidad y van tornándose a un color grisáceo. De esta forma fue posible generar un efecto visual acorde a la temática del juego, además de que sirvió para reducir la distancia de renderizado sin que sea notado por el jugador, lo que ayudó al desempeño del programa.

3.9. Luces

Se utilizó el modelo de Phong para generar las luces. Se utilizaron dos tipos de luces: direccional (para iluminar todo el modelo) y spotlight (para simular una linterna). La luz spotlight sigue al personaje principal y puede ser manipulada hasta cierto punto.

Fue necesario especificar estas luces en todos los shaders utilizados.

3.10. Instanciamiento de objetos

Para poder poner una gran cantidad de objetos (del mismo tipo) en el escenario, sin impactar mucho el rendimiento, se hizo uso de la técnica de instanciamiento <https://learnopengl.com/Advanced-OpenGL/Instancing>. Esta técnica se utilizó con tres modelos: el pasto y dos modelos de pinos.

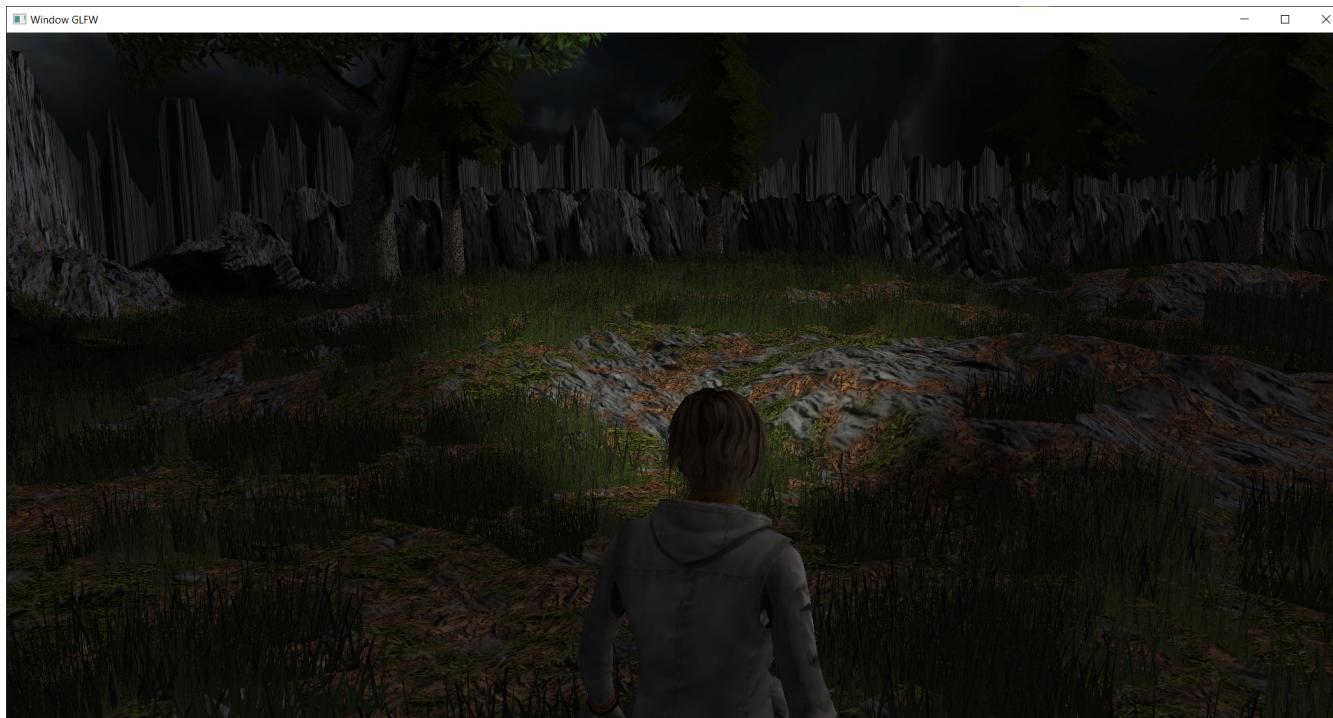


Figura 5: Instanciamiento en el modelo de pasto

Para realizar este procedimiento, fue necesario modificar el shader de modelos, así como la función para el renderizado de los modelos.

3.11. Comportamiento del enemigo

Para el comportamiento del enemigo se decidió realizar las siguientes condiciones:

- Si el Heather cruza un umbral, el enemigo rota y se desplaza hacia ella.
- En impacto, el enemigo deja de rotar hacia Heather y sigue desplazándose en la dirección que tenía.
- Al llegar al límite del escenario, el enemigo vuelve a activar la rotación para apuntar hacia Heather.
- El enemigo se detiene si se cumplen los objetivos del juego o la vida de Heather llega a cero.