



# 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

<b>1. Introducción</b>	<b>2</b>
<b>2. Objetivo</b>	<b>2</b>
<b>3. Desarrollo</b>	<b>2</b>
3.1. Obtención de modelos . . . . .	2
3.2. Animaciones . . . . .	2
3.3. Terreno . . . . .	2
3.4. Generación del escenario . . . . .	2
3.5. Colisiones . . . . .	3
3.6. Cámara . . . . .	4
3.7. Sistema de partículas . . . . .	4
3.8. Neblina . . . . .	5
3.9. Luces . . . . .	5
3.10. Instanciamiento de objetos . . . . .	5
3.11. Comportamiento del enemigo . . . . .	5
3.12. Sonido . . . . .	6
<b>4. Costo y comercialización</b>	<b>6</b>
<b>5. Principales retos y dificultades</b>	<b>6</b>
<b>6. Mejoras y trabajo a futuro</b>	<b>7</b>
<b>7. Conclusiones</b>	<b>7</b>

## 1. Introducción

En el año 2014 se anunció que un nuevo juego de Silent Hill sería desarrollado por Hideo Kojima en colaboración con Guillermo del Toro. No mucho tiempo después, Konami rompió los sueños de todos aquellos que esperábamos este juego. Este proyecto nació inspirado por el cariño hacia esta franquicia de videojuegos.

## 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.

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.



Figura 1: Mapa de alturas utilizado en el proyecto

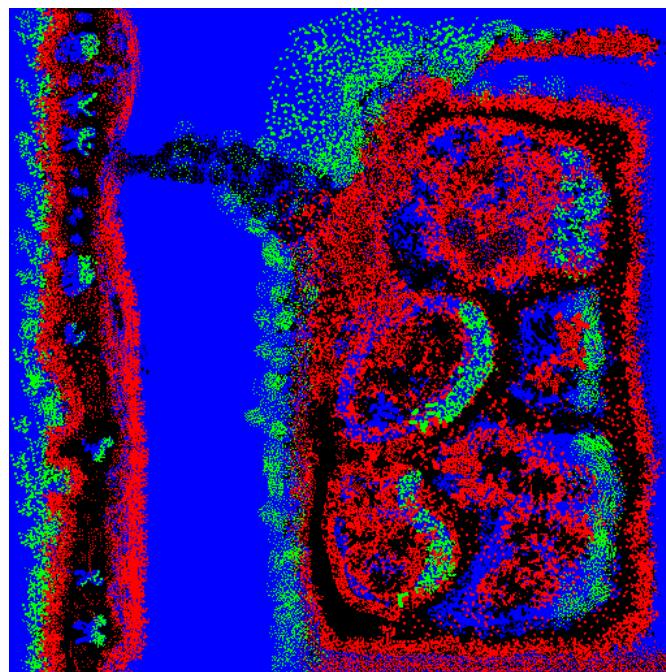


Figura 2: Mapa de alturas utilizado en el proyecto

### 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.

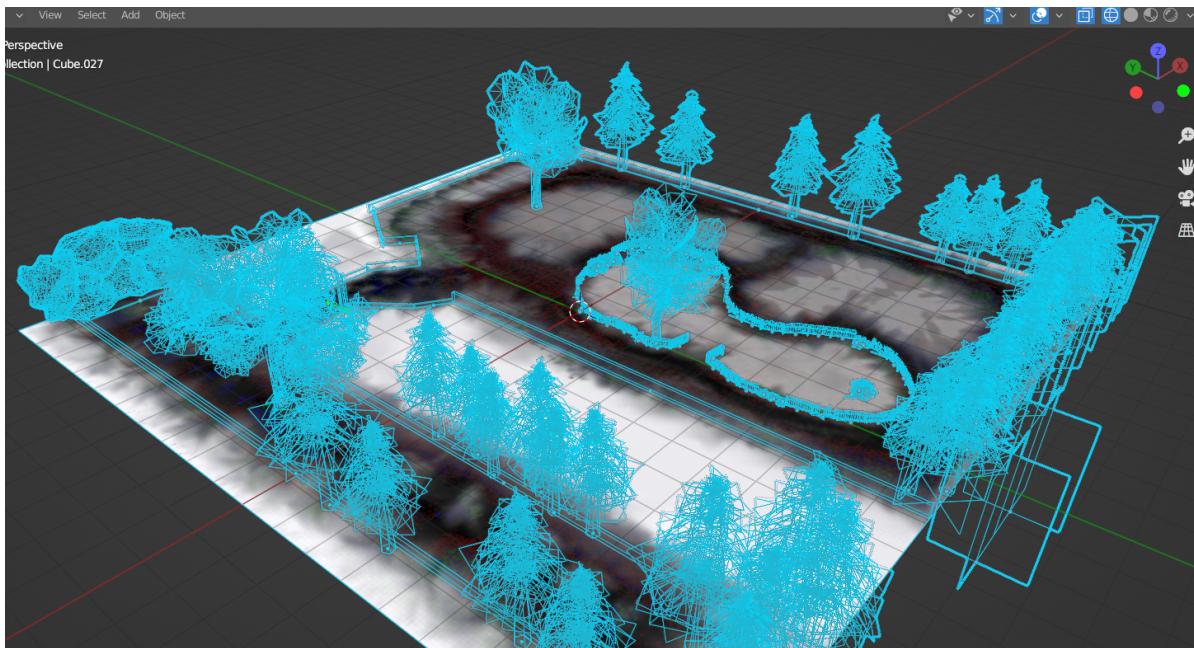


Figura 3: Mapa del escenario creado en Blender para obtener las posiciones de los objetos

- 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.

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.

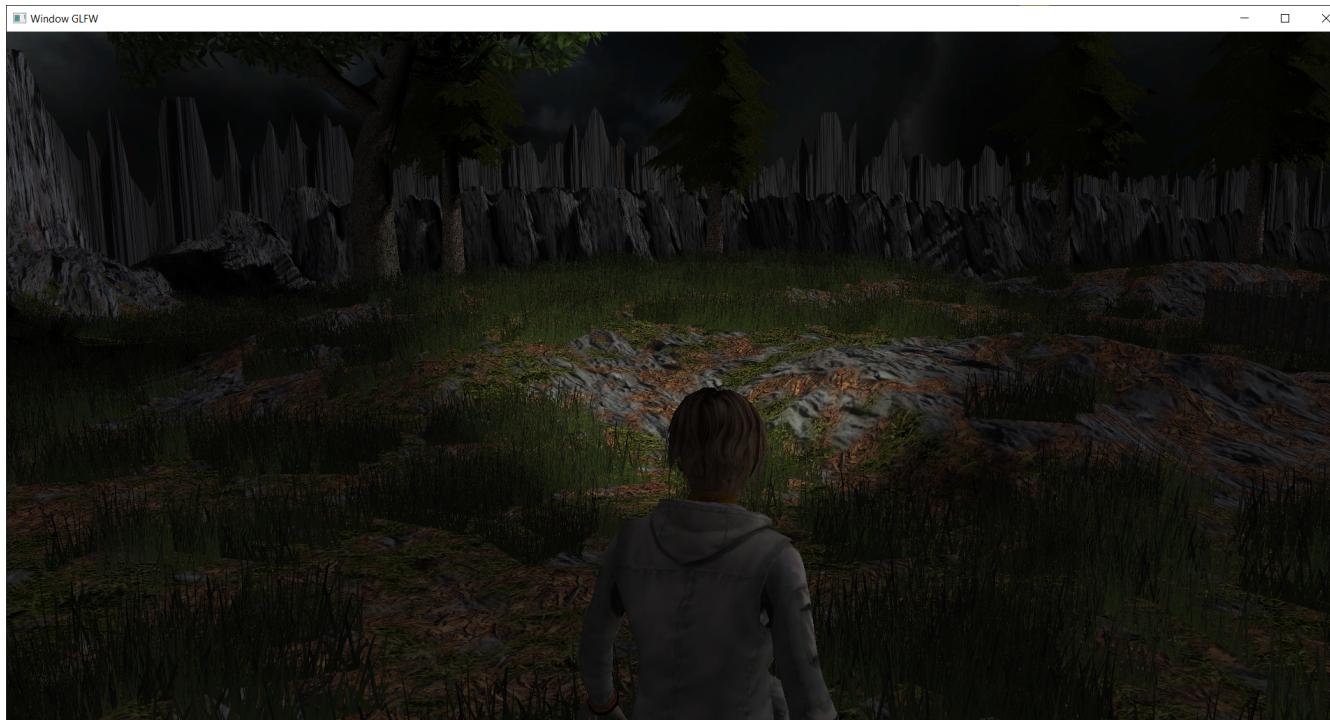


Figura 5: Instanciamiento en el modelo de pasto

### 3.12. Sonido

Para implementar el sonido se utilizó la API de OpenAL. De esta forma fue posible agregar distintas pistas de sonido, a las cuales se les dio una posición en el escenario. Algunos sonidos (los sonidos ambientales de grillos) se reproducen de manera constante en un loop. La pista de música se reproduce una sola vez. Otros sonidos se reproducen solo cuando se llevan a cabo diferentes acciones:

- Avanzar: Se activa el sonido de pasos cuando se da la orden de avanzar (tecla W), retroceder (tecla S) o correr (Control Derecho + W). La frecuencia de activación del sonido es distinta para cada una de estas acciones.
- Colisión con enemigo: Heather emite un sonido de dolor cuando el enemigo la golpea.
- Recolección de item: Al recojer un item se emite un sonido particular.

## 4. Costo y comercialización

Es difícil estimar un costo del proyecto, ya que varios de los modelos utilizados no tienen una licencia de uso comercial y algunos son propiedad intelectual protegida. Por lo que sería necesario contactar a los creadores o generar modelos completamente nuevos.

Una vez sorteado este problema sería posible proponer la comercialización. En cuanto a este tema, se propone utilizar el sistema de *Early Access* en Steam <https://store.steampowered.com/genre/Early%20Access/>. Para esta primera etapa se propone un precio de entre tres y cuatro dólares, conforme se vaya desarrollando el proyecto se podría ir aumentando el costo.

## 5. Principales retos y dificultades

Los principales retos que tuve los listo a continuación:

- Posicionamiento de los objetos en el escenario: Debido a que se colocaron varios objetos en el escenario, fue tardado obtener la posición de todos ellos para poder ponerlos en el proyecto final.
- Cantidad de modelos: Una de mis intenciones principales fue la generar una ambientación adecuada para el juego. Debido a que el escenario era un bosque pensé que la mejor manera de lograr esto era utilizar varios modelos de vegetación. En un principio esto se llevó a cabo por medio de loops, pero pronto se llegó a un punto en que el desempeño del programa cayó bastante. Este problema se logró resolver la técnica de instanciamiento, pero esto generó un nuevo problema, que fue el de la modificación de los shaders y métodos de la clase de los modelos.
- Codificación del comportamiento del enemigo: En un principio se pensaba desarrollar un comportamiento más complejo para el enemigo. Sin embargo, se encontraron diferentes dificultades para esto, por lo que se decidió realizar un comportamiento más sencillo.
- Cámara en primera persona: la cámara en primera persona no fue particularmente difícil de implementar. Lo que se me complicó fue utilizar el ratón para que tanto la cámara, como el modelo de Heather, rotaran a la misma velocidad.
- Animaciones: Nuevamente, las animaciones por sí solas no fueron complicadas, pues la mayoría fueron hechas en Mixamo. Lo que fue complicado fue modificar los modelos ya animados para agregar modelos que siguiera la animación. Por ejemplo, se quiso agregar un arma a la animación de apuntar, dentro de Blender la animación se llevaba a cabo, pero en al importar la animación al proyecto, el modelo del arma quedaba en una posición diferente y sus dimensiones eran distintas a las observadas en el software de modelado.
- Optimización: Sin duda, el mayor reto que tuve fue el de la optimización. En varios puntos del desarrollo encontré que el desempeño del programa bajó por diferentes razones. Fue necesario realizar varias modificaciones para llegar a un punto en el que se tuviera un equilibrio entre rendimiento y calidad gráfica.

## 6. Mejoras y trabajo a futuro

- En un principio se pensó la generación de tres escenarios diferentes. Pero por los problemas listados anteriormente esto al final no fue posible. Por lo que una de las mejoras que haría sería desarrollar dichos escenarios o más.
- Otro punto que creo podría mejorar sería la optimización. Al momento, el juego corre entre 40-60 fps con una resolución de 720p. Creo que el rendimiento se puede mejorar de diferentes maneras como utilizando modelos más ligeros u otras técnicas como utilizar texturas con menor resolución para los objetos alejados.
- Otro aspecto en lo que creo que quedó corto el juego fue la jugabilidad. Algunas acciones del personaje fueron desechadas durante el desarrollo, como su habilidad para pelear con el enemigo.
- También sería bueno incrementar la variedad de enemigos.

## 7. Conclusiones

Con este proyecto fue posible llevar a la práctica diversas técnicas vistas durante la clase para poder llevar a cabo el desarrollo de un videojuego. Muchas de estas técnicas, aunque fueron relativamente fáciles de implementar durante las prácticas, fueron un poco más complicadas al integrarlas en un proyecto

más complejo, principalmente porque todas en conjunto pueden llevar a una pérdida de rendimiento del programa.

También es necesario tomar en cuenta que el juego desarrollado puede resultar muy pesado para algunas computadoras. En este caso se utilizó una computadora con una tarjeta gráfica dedicada y relativamente reciente. Por esta razón sería conveniente llevar a cabo una mayor optimización o generar diferentes perfiles para adaptarse a computadoras con diferentes características.

Creo que se logró cumplir con el objetivo del proyecto, sin embargo siento que me quedé algo corto con el alcance que quería que tuviera el juego, esto fue debido a diferentes problemas que fui encontrando en el camino. Sin embargo, ahora tengo un mejor entendimiento del funcionamiento de OpenGL, que creo me puede ayudar a encontrar nuevas soluciones a los problemas encontrados.