UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

INGENIERÍA EN COMPUTACIÓN

PROYECTO FINAL
COMPUTACIÓN GRÁFICA AVANZADA

DESARROLLO PROYECTO

PAC-MAN

MAZE MADNESS 3D

Elaborado por
Isabel Gómez Yareli Elizabeth
Ortiz Figueroa María Fernanda
Rivera Roldán Luis Ricardo

Grupo
1

Profesor
Reynaldo Martell Ávila

Ciudad Universitaria, Ciudad de México
18 de junio 2020

INTRODUCCIÓN

Actualmente, el mundo real representado en ambientes 3D por computadora con base a diferentes historias, planteamientos o dinámicas, permiten generar experiencias a los usuarios, por lo que su elaboración resulta todo un reto.

En la vida real nos encontramos con modelos humanoides que se mueven gracias a las articulaciones que poseen, lo que permite que estos puedan alzar la mano, brincar y caminar con cierto compás como lo hacen normalmente los brazos y piernas de una persona normal.

Ahora bien, nosotros como individuos nos encontramos inmersos en un espacio, el cual cuenta con paisaje, objetos/elementos naturales y artificiales, tales como rocas, agua, viento, fuego, neblina o bien diversa infraestructura construida por el ser humano.

Lo anterior, es parte con lo que cuenta nuestro mundo, lo cual sería imposible de visualizar sin una fuente de luz adecuada, dependiendo desde donde y quien lo observe, además de poder visualizar su sombra o alguna mezcla de colores, dependiendo del tipo objeto/material del que se trate, de tal forma que tenemos una mayor percepción y colores distintivos.

De esta manera, nuestros sentidos nos permiten apreciar el entorno de una forma más completa y realista, sin olvidar que lo visual no es lo único que tiene impacto, el sonido permite caracterizar ciertos objetos/elementos, desde lo que son, hasta sus distancia y dirección, que es percibida por nuestros oídos.

Naturalmente, estos son aspectos que para nosotros son normal en nuestro mundo, sin embargo, la manera en la que percibimos cada uno de estos aspectos, es lo que se busca generar a través de una computadora, por lo que recrear estos elementos implica la forma en la que se puede hacer y lo que se requiere para ello, por lo que el presente documento busca mostrar a partir del planteamiento de videojuego, los resultados que se pueden obtener para acercarnos a una realidad por computadora.

OBJETIVO

Desarrollar el videojuego de Pac-Man Maze Madness 3D, elegido por el equipo, para implementar los diversos temas vistos en el curso de Computación Gráfica Avanzada, para lograr crear características que percibimos del mundo real en un ambiente por computadora.

De tal forma, que un usuario pueda controlar al personaje de Pac-Man a través de un ***laberinto*** ubicado en un terreno, con el fin de recolectar todos los ***puntos amarillos*** distribuidos por el lugar en un determinado ***tiempo***, considerando que en el camino podrá encontrar obstáculos como ***fantasmas***, los cuales podrán restarle ***vida;*** algunas ***frutas***, las cuales proporcionarán puntos adicionales, y ***power pellets***, que le permitirán al personaje aumentar su velocidad y comer a cualquier enemigo del área por un tiempo limitado.

DESARROLLO.

- **Obtención y desarrollo de modelos.**

El modelo de Pac-Man, laberinto y fantasma, fueron desarrollados con ayuda de Maya y 3DS Max, mientras que los modelos de las frutas y antorcha se obtuvieron de internet.

Es importante señalar que en ambos casos se ajustó el pivote y escala, además de que se optimizaron, con ayuda de MeshLab, de tal manera que tuvieran el mínimo de polígonos sin deformar el modelo mismo.

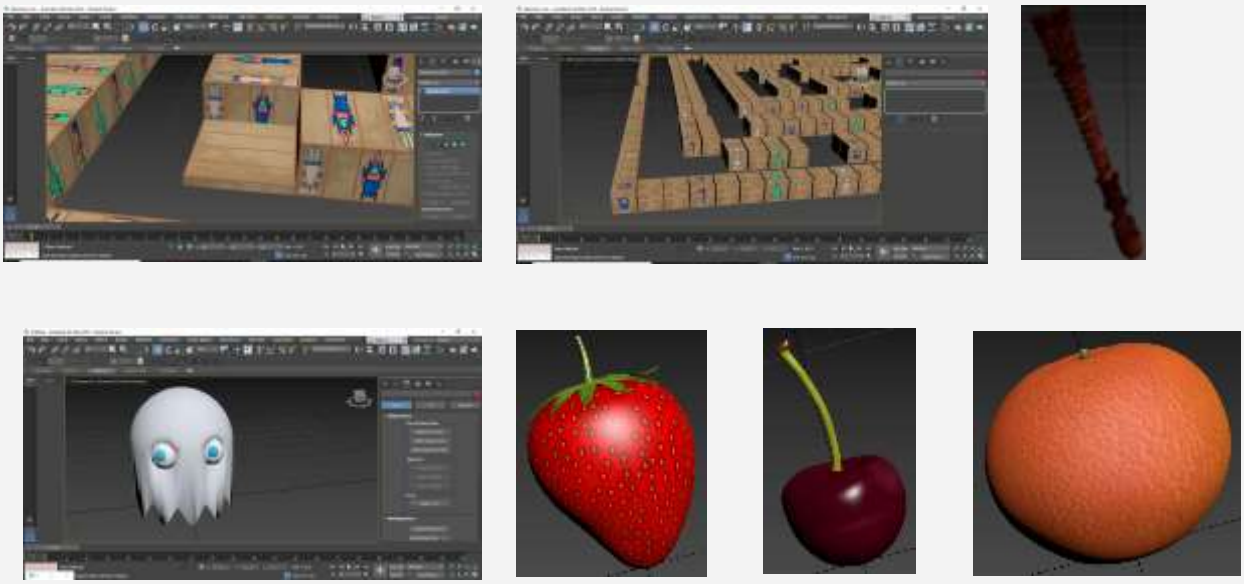

Imagen 1.Obtención y desarrollo de modelos.

-Animación modelo principal.

El modelo desarrollado de Pac-Man se realizó en T-pose, para poder generar las animaciones de este con a través de Mixamo, y con ayuda de Blender se colocaron en un solo modelo dichas animaciones (reposo, caminar, alzar la mano y morir).

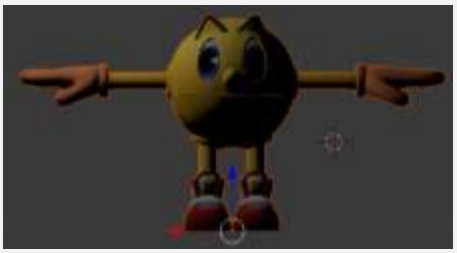

Imagen 2. Animación modelo principal.

- **Carga de modelos.**

Una vez que se tuvieron los modelos necesarios, se fueron incorporando al proyecto conforme se desarrollaba la funcionalidad del videojuego en el proyecto de VisualStudio, así como la incorporación de sombras para dar una mejor apariencia al ambiente 3D.

Imagen 3. Carga de modelos.

- **Implementación de cajas de colisión.**

En este caso, por el tipo de objetos que se manejaron, las cajas de colisión empleadas fueron la esfera (Pac-Man, puntos amarillo y frutas) y la OBB (muros del laberinto y fantasmas).

Cabe destacar, que el modelo del laberinto fue exportado por partes, para poder agregar a cada una de ellas una caja de colisión, de tal forma que el Pac-Man se pudiera desplazar por las zonas adecuadas.

Imagen 4. Caja de colisión Pac-Man

- **Implementación de mapa de alturas y texturas.**

A pesar de que en este caso este videojuego no requería la implementación de un mapa de alturas para tener un terreno con relieve, se implementaron, de tal forma que éste proporcionó un ligero relieve dentro del laberinto y a las afueras pequeñas elevaciones para simular dunas y depresiones para simular un pequeño río, por otro lado se empleó el mapa de texturas (arena, piso ladrillo café, agua y pasto), para poder marcar el camino dentro del laberinto, así como la apariencia al resto del terreno.

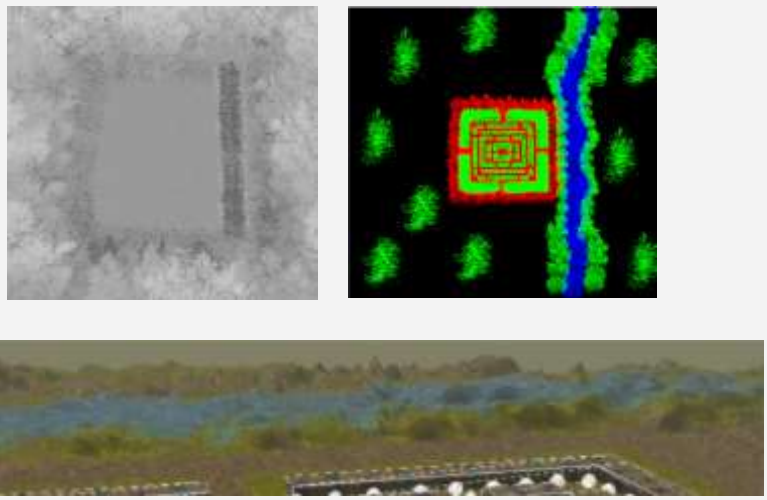

Imagen 5. Mapa de alturas y texturas.

- **Implementación de blending.**

El laberinto, al simular la apariencia de un desierto, se decidió colocar un poco de ***vegetación (agaves)*** para dar más ambientación en el terreno, de tal forma que auxiliándonos de la técnica de blending para implementarla.

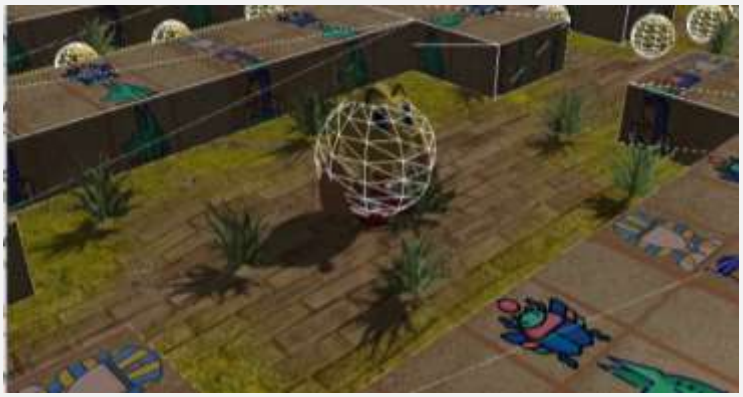

Imagen 6. Blending

Por otro lado, al requerir diversos fantasmas moviéndose por el laberinto, se tendría que cargar el mismo modelo pero en diversos colores, no obstante, para evitar esto se decidió manejar un modelo de fantasma de color blanco, cargarlo una solo vez, pero renderizarlo varias veces y con ayuda de la técnica de blending darle a cada uno un color diferente, y al usar alguna combinación de funciones, darle cierta transparencia y mejor apariencia.


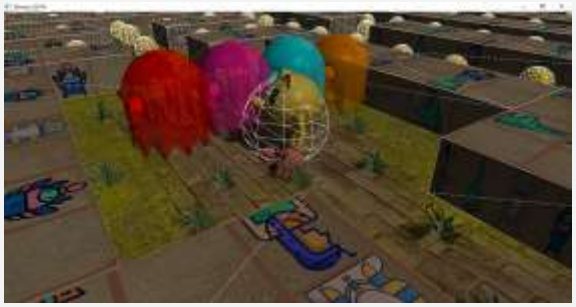

Imagen 7. Blending fantasmas.

- **Creación de dinámica para comer puntos amarillos.**

En el videojuego de Pac-Man, parte de su dinámica es que este vaya comiendo los puntos amarillos distribuidos por el laberinto en un determinado tiempo, para poder manejarlos se creó una estructura del tipo mapa (llave/valor), la llave en este caso fue un identificador para cada punto, mientras que su valor sería la posición y una variable para saber si el punto se encontraba habilitado (0, no ha sido comido) o deshabilitado (1, ha sido comido), de esta manera solo se requirió cargar un solo modelo de una esfera sencilla, la cual sería renderizada con base a las posiciones de esta estructura, recordando que en cada ciclo se vuelve a renderizar la escena que vemos en el ambiente 3D y en el caso de que algún punto no estuviera ya habilitado, este no sería renderizado ni considerado dentro del de la estructura encargada de manejar el modelo de colisión, lo cual indica que se ha detectado la colisión entre dicho punto y el Pac-Man, por lo que ya no se requiere visualizar en la escena, lo que permite generar la apariencia de que Pac-Man ha comido el punto amarillo, además se lleva la cuenta de cada punto que come.

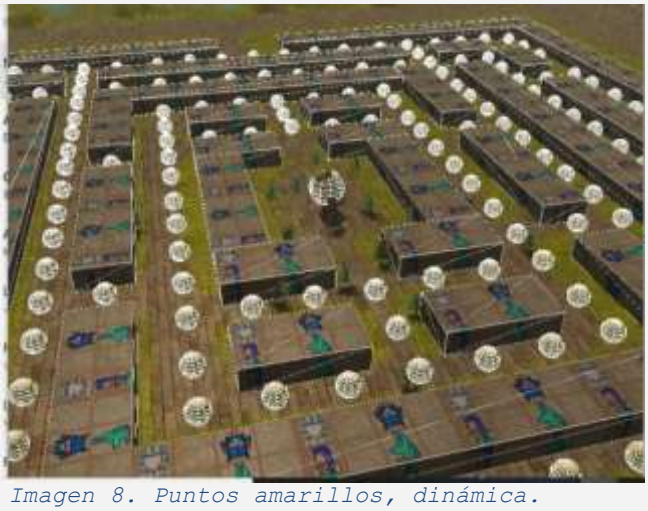

Imagen 8. Puntos amarillos, dinámica.

- **Creación de dinámica para movimiento de Pac-Man.**

Considerando que para manejar un modelo se hace uso de una matriz, y en caso de querer simular algún desplazamiento se aplica algún tipo de transformación a dicha matriz, por lo que al mover alguna de las flechas (arriba, izquierda, abajo o

derecha), se aplicaba una transformación de translate para que este se moviera en dicha dirección, sin embargo, el modelo conservaba su posición de vista, es decir, si se desplazaba hacia atrás el modelo seguía mirando al frente, sin embargo, al aplicar la transformación de rotate según se desplazará, el sentido de los movimiento se cambiaba y se perdía en sentido del movimiento con las flechas, por lo que analizando dicho comportamiento, el Pac-Man siempre hace un translate hacia adelante y considerando si su movimiento anterior es el mismo o a es otro, tomar la decisión de qué transformación de rotate aplicar, permitiendo tener un movimiento más natural del Pac-Man dentro de la escena.

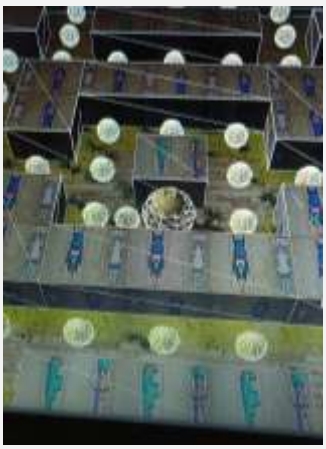

Imagen 9. Movimientos Pac-Man.

- **Creación de dinámica para movimiento de fantasmas.**

Observando el videojuego original de Pac-Man Maze Madness, nos percatamos de que al moverse los fantasmas estos colisionaban con las paredes del laberinto lo que hacía que cambiara la dirección de su movimiento, independiente si estaba cerca el personaje o no.

Partiendo de la idea anterior, se realizó un pequeño algoritmo, para que se movieran los fantasmas, para poder avanzar primero se tiene que mover hacia la izquierda y a la derecha un poco, teniendo los siguientes casos:

- Si existe colisión a la izquierda y colisión a la derecha, avanzar adelante.
- Si existe solo colisión a la izquierda, girar a la derecha y avanzar.
- Si existe solo colisión a la derecha, girar a la izquierda y avanzar.

- Si no existe colisión en ningún caso, girar de manera aleatoria a la izquierda o derecha y avanzar.

Lo anterior permitió el movimiento libre de los fantasmas por el laberinto, sin embargo, se tuvo que considerar la distancia que había entre ellos, si esta era muy próxima, después de realizar alguno de los casos anteriores se indicaba un giro adicional para que cambie de dirección.

- **Creación de dinámica para comer fantasmas.**

Ahora bien, en caso de que el Pac-Man haya comido un punto del tipo Power Pellets, ubicados en la esquina del laberinto, se comienza una cuenta en el que todos los fantasmas por ese tiempo pasan a ser de color azul, al Pac-Man se le aumenta la velocidad de desplazamiento, y en caso de colisionar el Pac-Man con algún fantasma en ese tiempo, este será comido, dará cierto puntaje adicional al usuario y permanecerá deshabilitado, una vez transcurrido el tiempo para comer fantasma y el tiempo de muerto del fantasma, este vuelve a su posición inicial y comienza de nuevo su desplazamiento por el laberinto.

Imagen 10. Comer fantasmas.

- **Manejo de cámara.**

La cámara para implementar para este tipo de videojuego fue en **tercera persona**, de tal forma que este sigue al personaje de Pac-Man conforme este se desplaza en el laberinto, lo que permite tener cierta visualización de este hasta cierto punto.

Imagen 11. Manejo de cámara.

Por el tipo de cámara implementada, la visualización de todo el laberinto no es posible y por la dinámica del juego no siempre se sabe dónde se encuentran los puntos amarillos faltante para terminar y ganar, por lo que se agregó la **vista área**, de tal forma que se puede tener la vista de todo el laberinto, sin embargo, se decidió que el Pac-Man no

se pueda mover en esta visualización y el tiempo siguiera transcurriendo.

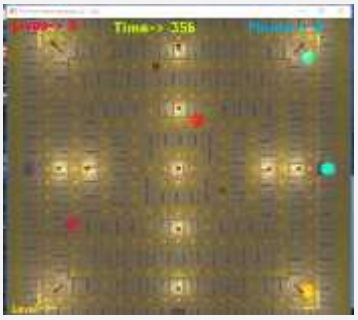

Imagen 12.Vista aérea.

- **Creación de dinámicas generales.**

En general se lleva la cuenta del tiempo transcurrido, ya que solo se tienen 8 minutos para terminar el laberinto.

Así mismo, se lleva la cuenta de los puntos obtenidos por comer un punto amarillo, una fruta encontrada en el camino o comer un fantasma.
Por otro lado, se cuenta con cierto número de vidas, por lo que al colisionar con un fantasma, cuando no es tiempo de comer, se restará una vida, y en caso haber perdido todas las vidas, Pac-Man morirá y por ende será un caso de perder en este juego.

Así mismo, en el laberinto han sido colocadas 3 diferentes frutas, lo que le darán puntos adicionales al usuario.

Imagen 13. Frutas.

- **Implementación de partículas.**

La implementación de partículas se decidió aplicar en dos casos:

- Antorcha

Simulación de fuego en las antorchas de tal forma que dieran la apariencia de estar encendidas.

Imagen 14. Partículas en antorcha (fuego).

- Portales

Los laberintos de este clásico juego, en los extremos cuenta con salidas que trasladan a Pac-Man a un extremo u otro del laberinto, por lo que para indicar esta zona que denominamos portales, se colocó cierto efecto de partículas.

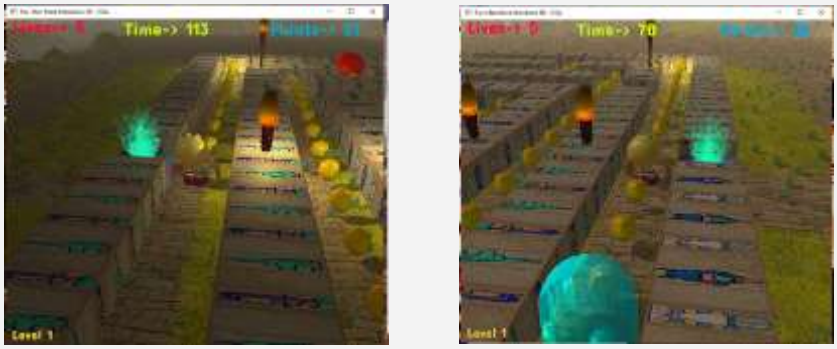

Imagen 15. Partículas portales.

- **Implementación de neblina.**

Al considerar la temática de desierto, se decidió emplear el efecto de neblina para dar la apariencia de arena movida por el viento, a las afueras del laberinto.

Imagen 16. Neblina.

- **Implementación de luces.**

Para dar una mejor ambientación se decidió manejar solo los siguientes tipos de luces:

Imagen 17. Luces.

- Luz direccional, la cual es empleada en conjunto con cierto conteo para tener la transición día y noche dentro del ambiente.

- Luz puntual, la cual es empleada para:

  Iluminación de las antorchas en caso de que sea de noche.

Imagen 18. Luz puntual.

Iluminación para resaltar Power Pellets, en el momento que estos puntos son comidos, la iluminación en este punto desaparece.

Imagen 19. Power pallets iluminados.

- **Implementación de texto.**

OpenGL no es una API con capacidades para manejar texto, por lo que depende de nosotros definir un sistema para representar el texto en pantalla, ahora bien no existen primitivas gráficas para los caracteres de texto, por lo que decidimos usar la biblioteca de **FreeType** en donde se representa a través de texturas caracteres en quads, de manera general se sigue el siguiente procedimiento:

- Cargar la fuente.
- Definir el tamaño de la fuente de pixel.
- Configurar el indicador de carga.
- Para poder manejar cada carácter se almacenan en una estructura tipo mapa.
- Enumerar los caracteres del conjunto a trabajar.
- Recuperar los caracteres correspondientes.
- Definir el vertex shader para poder renderizar los caracteres.
- Definir el fragment shader, el cual toma la imagen de mapa de bits del carácter y ajusta el color final del texto.
- Habilitar el blend y definir alguna función.

- A partir de una proyección ortográfica se específica la posición del texto en la pantalla.
- Crear un VBO y VAO para renderizar los quads.
- Crear una función para renderizar una cadena de texto específico.
- Ajustar dentro de la ejecución del proyecto el renderizado del texto.

Imagen 20. Implementación de texto.

- **Implementación de sonido.**

Para poder dar más realismo a nuestro videojuego, se implementó la API de OpenAL, la cual permite modelar una colección de fuentes de audio que se mueven en un espacio 3D que son escuchadas por solo un oyente en algún lugar de ese espacio, de esta manera se manejan 3 objetos básicos: oyente, origen y buffer.
Para este juego no fue necesario especificar el oyente ya que OpenAL nos brinda un oyente por default, sin embargo fue necesario agregar los buffer que asignamos a cada una de nuestras fuentes, para cada una de las fuentes fue necesario agregar su posición, ganancia, máxima distancia, entre otras características.

Considerando lo anterior, así como la funcionalidad misma del videojuego elaborado, se agregaron los audios adecuados dentro del mismo.

- **Implementación de menú principal, menú pause y muerte en caso de perder.**

Finalmente, para poder tener un videojuego más interactivo en cuanto apariencia con el usuario se configuró un menú principal de manejar la cámara, el texto a visualizar y controlar que es lo que se ven en la escena, de tal forma que al momento de iniciar el juego se deje de visualizar las opciones de menú principal, y ahora se observe el laberinto y demás elementos.

Por otro lado, se configuró un pequeño menú para pausar el juego, de tal forma que esta es la única manera de detener el tiempo del juego, así como los movimientos de los fantasmas y del mismo

Pac-Man, por lo que dependerá del usuario decidir continuar, volver al menú principal o salir del juego mismo.

Así mismo, se configuró un pequeño lapso para mostrar la animación de que el Pac-Man ha muerto en caso de terminar el tiempo y no haya terminado de comer todos los puntos amarillo del laberinto, o en caso de perder todas sus vidas, lo que permite dar al usuario cierta retroalimentación al tener alguna de estas situaciones, antes de volver al menú principal para volver a iniciar el juego.

Imagen 21. Menú principal.

RESULTADOS Y TRABAJO FUTURO

Al final del desarrollo explicado anteriormente, podemos decir que se logró crear y desarrollar de manera exitosa una versión demo del juego de Pac-Man Maze Madness en 3D, con dos niveles con la misma temática y condiciones, considerando diversas funcionalidades del juego original, de tal forma que la apariencia y jugabilidad son aceptables.

Es importante mencionar, que la funcionalidad de este videojuego depende en gran medida de las capacidades del equipo en donde sea ejecutado, puesto que el único problema al final fue al momento de agregar luces del tipo puntuales, ya que el rendimiento en nuestros equipo se veía realmente afectado, provocando que se pudiera jugar a una velocidad muy lenta.

Ahora bien, con lo realizado consideramos que es la base del juego original de Pac-Man, por lo que el trabajo a futuro sería:

- Mejorar el algoritmo de movimiento de los fantasmas, de tal forma que ahora se implementen las dinámicas originales de cada uno de ellos.
- Desarrollar y agregar más niveles, con diferentes temáticas, así como agregar algún otro tipo de dinámicas u obstáculos.
- Implementación de un control.
- Optimización del funcionamiento en general, para mejorar aún más el rendimiento del juego.
- Mejorar las condiciones de colisión del Pac-Man para evitar que se pierda el sentido de su movimiento con las flechas.

CONCLUSIONES

Con las actividades realizadas en este proyecto podemos decir que la implementación de los conceptos vistos a lo largo del curso de Computación Gráfica Avanzada, se lograron implementar de manera exitosa, de tal forma que nos permitió una mejor comprensión y aplicación de estos al resolver otro tipo de problemáticas al momento de crear y desarrollar el videojuego propuesto.

Por otro lado, logramos mejorar nuestras habilidades y conocimientos en cuanto al uso de OpenGL, C++ y Shaders, así

como la implementación de otro tipo de bibliotecas para obtener un proyecto más robusto.

Finalmente, podemos decir que realizar este tipo de proyecto ha sido un gran reto para nosotros, ya que el hecho de construir un videojuego con herramientas y conceptos básicos es más laborioso, sin embargo, al adquirir una base sólida nos permite trasladar diversos conceptos y funcionalidades a otro tipo de herramientas más robustas, además de que en la actualidad aún se sigue empleado en diversas aplicaciones las herramientas y conceptos que hemos empleado.