Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Puebla

Construcción de Software y Toma de Decisiones

TC2005B.401

Tarea 1 en equipo: Ciclo 1. VideoJuegos

Profesora: Dr. Iván Olmos Pineda

Estudiantes:

Daniel Francisco Acosta Vázquez / A01736279

Diego García de los Salmones Ajuria / A01736106

Raúl Díaz Romero / A01735839

Rogelio Hernández Cortés / A01735819

Periodo Febrero - Junio 2023

20 / Marzo / 2023

Ciclo 1. VideoJuegos: Pacman en 2D

1. Texturas

En el proyecto se determinó que se utilizarán 5 diferentes texturas para representar el videojuego ‘Pac-Man’, las 5 texturas pertenecen a los siguientes objetos:

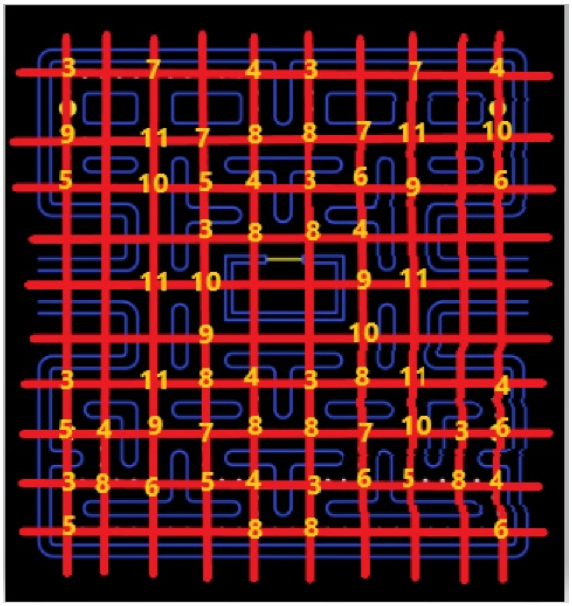
- **"maze.bmp",** textura que representa el tablero del videojuego.
- **"pacman.bmp",** textura que representa al ‘Pac-Man’ cuando su diseño tiene la boca abierta.
- **"pacman_closed.bmp",** textura que representa al ‘Pac-Man’ cuando su diseño tiene la boca cerrada.
- **"fantasma_rojo.bmp",** textura que representa al fantasma rojo utilizado en el videojuego.
- **"fantasma_azul.bmp",** textura que representa al fantasma azul utilizado en el videojuego.

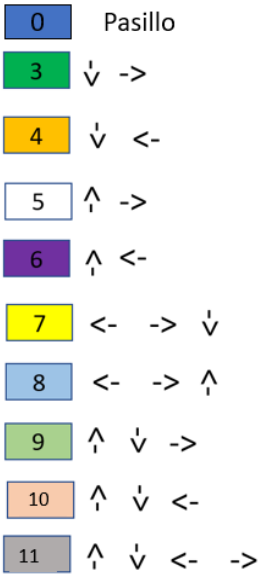
Cabe resaltar que se tienen 2 texturas para representar al ‘Pac-Man’ debido a que utilizamos un temporizador para cambiar entre texturas y simular que el ‘Pac-Man’ abre y cierra su boca durante el videojuego.

2. Matriz de Control

En el proyecto manejamos las intersecciones con una matriz de control de 10x10, en dicha matriz se le asigna un número a cada tipo de intersección y pasillo, esto debido a que dependiendo del tipo de intersección, los objetos establecidos se moverán de diferente manera. Esta matriz fue declarada dentro del archivo ‘main.cpp’, esto para poder ser accedida por todos los objetos del sistema.

A continuación se adjunta 2 imágenes representan la matriz de intersecciones y el número que representa a cada tipo de intersección.





Posteriormente al creado de la matriz de intersecciones, se crearon 2 arreglos para definir las coordenadas exactas en píxeles de cada intersección, por ello se generaron los arreglos 'X' y 'Y'. Estos arreglos fueron declarados dentro del archivo 'main.cpp', esto para poder ser accedidos por todos los objetos del sistema.

Por último, el propósito de los arreglos 'X' y 'Y' es determinar en todo momento en que intersección están los fantasmas y el 'Pac-Man', en caso del 'Pac-Man' se utiliza para definir qué movimientos puede tomar el jugador y en caso de los fantasmas se utiliza para definir en qué dirección se pueden mover.

3. Clase 'Pac-Man'

Para el control del objeto 'Pac-Man' se decidió generar una clase que contenga las diversas variables de control junto a las funciones necesarias para dibujar y actualizar la posición del objeto.

Los atributos contenidos en esta clase son los siguientes:

- **int estado,** esta variable representa el estado o dirección actual del objeto, este valor se cambia dentro de la función 'SpecialInput', dicha función sirve para detectar los inputs del teclado hechos por el jugador. En este caso manejamos 4 tipos de estado para el objeto 'Pac-Man':
  - 1, dirección hacia arriba.
  - 2, dirección hacia abajo.
  - 3, dirección hacia derecha.
  - 4, dirección hacia izquierda.
- **int tempo**, esta variable representa un temporizador que determina el ritmo en que se intercambian las texturas que se dibujaran sobre el objeto 'Pac-Man', pues como se mencionó anteriormente, se manejan 2 texturas para el 'Pac-Man', una textura donde su boca está abierta y otra donde está cerrada.
- **int txtAct**, esta variable funciona para manejar qué textura se va a utilizar sobre el objeto 'Pac-Man', esta variable es modificada por a través de un condicional relacionado a la variable 'tempo'. A continuación se adjunta una imagen de dicha función.

```
tempo += 1;
if(tempo > 1000){
    txtAct = (txtAct == 0) ? 2:0;
    tempo = 0;
}
```

- **int bufferkey**, esta variable tiene el propósito de guardar el valor de de la variable estado en caso en que el estado que se haya detectado con la función 'SpecialInput' no se haya podido ejecutar inmediatamente. En otras palabras, la variable sirve para almacenar un input del teclado del jugador, esto en caso de que se encuentre en una

intersección que no le permite moverse a donde desea hacerlo, en ese caso la variable almacena el valor del input del teclado y lo aplica en la siguiente intersección.
- **float Position[2]**, esta variable almacena las coordenadas en el eje X y Y del objeto 'Pac-Man'.
- **float velocidad**, esta variable almacena la velocidad del objeto 'Pac-Man'.

Las funciones contenida en esta clase son las siguientes:

- **pacman()**, esta función sirve para construir un objeto de esta clase.
- **void draw()**, esta función sirve para dibujar el objeto 'Pac-Man', para ello se utiliza el arreglo Position[2] y las diversas funciones de OpenGl para dibujar correctamente al objeto junto a su textura.
- **void update(int)**, esta función recibe un entero que representa el tipo de intersección en que se encuentra y sirve para actualizar las variables de control del objeto 'Pac-Man', por ejemplo, se actualiza la nueva posición que tendrá el 'Pac-Man' conforma al estado e intersección en que se encuentra, igualmente, se actualiza la bufferkey, el tempo y el txtAct.

4. Clase 'Ghost'

Para el control de los objetos fantasma se decidió generar una clase que contenga las diversas variables de control junto a las funciones necesarias para dibujar y actualizar la posición del objeto.

Los atributos contenidos en esta clase son los siguientes:

- **int direccion,** esta variable representa la dirección actual del objeto, este valor se calcula automáticamente y varía de valor dependiendo en qué tipo de intersección esté. En este caso manejamos 4 tipos de dirección para el objeto:
    - 1, dirección hacia arriba.
    - 2, dirección hacia abajo.
    - 3, dirección hacia derecha.
    - 4, dirección hacia izquierda.

- **int txt**, esta variable funciona para manejar qué textura se va a utilizar sobre el objeto.
- **float Position[2]**, esta variable almacena las coordenadas en el eje X y Y del objeto.
- **float velocidad**, esta variable almacena la velocidad del objeto.

Las funciones contenida en esta clase son las siguientes:

- **ghost(int, float, float)**, esta función sirve para construir un objeto de esta clase, dicha función recibe el número de la textura del objeto, su posición inicial en el eje X y su posición inicial en el eje Y.

- **void draw()**, esta función sirve para dibujar el objeto fantasma, para ello utiliza el arreglo Position[2] y las diversas funciones de OpenGl para dibujar correctamente al objeto junto a su textura.
- **void update(int)**, esta función recibe un entero que representa el tipo de intersección en que se encuentra y sirve para actualizar las variables de control del objeto fantasma, por ejemplo, se actualiza la nueva posición y dirección que tendrá el fantasma conforma a la dirección e intersección en que se encuentra, esto se realiza para que el fantasma no entre en ciclos infinitos entre intersecciones

5. Archivo 'Main.cpp'

En el archivo 'Main.cpp' se manejan las funciones y variables principales del proyecto y se llaman a las funciones de las clases 'Pac-Man' y 'Ghost' para generar los objetos del sistema.

Las variables globales utilizadas en este archivo son las siguientes:

- **GLuint texture[NTextures]**, este arreglo almacena las diversas texturas utilizadas en el proyecto, en este caso se utilizan 5 texturas en total.
- **pacman manpac,** este objeto representa al 'Pac-man' controlado por el jugador.
- **ghost red(3, 7.0, 231.0),** este objeto representa al fantasma rojo utilizado en el videojuego, igualmente se puede notar cómo se inicializa con el número de su textura y sus coordenadas iniciales.
- **ghost blue(4, 207.0, 231.0),** este objeto representa al fantasma azul utilizado en el videojuego, igualmente se puede notar cómo se inicializa con el número de su textura y sus coordenadas iniciales.
- **int control_matrix[10][10],** esta matriz de 10x10 representa la matriz de control utilizada en el proyecto, su función es establecer los diversos tipos de intersecciones/celdas que se encuentran en el tablero.
- **int X[224], Y[248],** estos arreglos representan las coordenadas X y Y en píxeles exactos de las intersecciones/celdas del tablero.

Las funciones contenida en esta clase son las siguientes:

- **void loadTextureFromFile(char *filename, int id),** esta función sirve para cargar las diversas texturas a nuestro sistema, esto para poder utilizarlas correctamente al dibujarlas en los diversos objetos.
- **void init_MatAdy(),** esta función sirve para inicializar los valores de la matriz de control y los arreglos X y Y.
- **int obtenerCelda(float p[2]),** esta función recibe un arreglo de las coordenadas del objeto 'Pac-Man', estas coordenadas las utiliza para verificar que valor contienen en los arreglos X y Y. Posteriormente, si dichos valores son diferentes de -1, se verifica qué tipo de intersección/celda hay en dichas coordenadas a través de la matriz de control. A continuación se adjunta una imagen de la función.

```
int obtenerCelda(float p[2]){
    int x = int(p[0]);
    int y = int(p[1]);

    if(X[x] != -1 && Y[y] != -1){
        return control_matrix[Y[y]][X[x]];
    }
    return 0;
}
```

- **int obtenerCeldaG(float p[2]),** esta función es sumamente similar a la función obtenerCelda, la cual se utiliza para determinar en qué tipo de intersección se encuentra el 'Pac-Man', sin embargo existe una diferencia importante entre las funciones. La función obtenerCeldaG sirve verificar en qué tipo de intersección se encuentra el objeto en las coordenadas recibidas a través de la matriz de control, sin embargo, solo verifica en qué tipo de intersección está sí el valor absoluto de la resta de las coordenadas actuales con el valor truncado de dichas coordenadas es menor al valor de la velocidad del objeto. Esto se realizó para que no se verificará la intersección/celda en la que se encuentra el objeto múltiples veces, pues la velocidad a la que avanza el objeto 'Ghost' es de menos de 1 píxel, lo cual causa que se verifique múltiples veces la misma intersección. A continuación se adjunta una imagen de la función.

```
int obtenerCeldaG(float p[2]){
    int x = (abs(p[0] - floor(p[0])) < red.getVelocidad()) ? int(p[0]):0;
    int y = (abs(p[1] - floor(p[1])) < red.getVelocidad()) ? int(p[1]):0;

    if(X[x] != -1 && Y[y] != -1){
        return control_matrix[Y[y]][X[x]];
    }
    return 0;
}
```

- **void initPacman(),** esta función sirve para inicializar el objeto 'Pac-Man' y dibujarlo en su posición inicial (esquina superior izquierda).
- **void drawScene(void),** esta función sirve para actualizar y posteriormente dibujar cada uno de los objetos del sistema (Pacman, fantasmas y tablero).
- **void init(),** esta función sirve para inicializar las matrices de modelado y proyección del sistema, dimensiones del sistema, texturas del sistema y se llaman a las funciones init_MatAdy() e initPacman().
- **void SpecialInput(int key, int x, int y),** esta función sirve para detectar el input de teclado del jugador y actuar acorde a dicho input y la intersección en que se encuentre el objeto 'Pac-Man', por ello en esta función se llama a la función obtenerCelda(float p[2]), para poder actualizar el estado y bufferkey del 'Pac-Man' acorde a dicho valor.
- **int main(int argc, char** argv),** esta función sirve para inicializar el programa con las diversas funciones de OpenGl necesarias para poder inicializar valores importantes y generar el glutMainLoop() para que el programa cicle continuamente.

6. Reflexiones

- **Daniel Francisco Acosta Vázquez (A01736279)**: Este proyecto fue sin dudas bastante complicado. Videojuegos es un área que siempre me ha apasionado y es a lo que me quiero dedicar profesionalmente, por lo que este proyecto fue un buen acercamiento hacia lo complejo, aunque muy interesante, que puede ser el desarrollo de los mismos, y más cuando no se utilizan herramientas más avanzadas. La parte más complicada fue a lo que no estaba acostumbrado, que fue la graficación inicial al igual que acostumbrarse al uso de OpenGL. Una vez dominado esto, fue sencillo, aunque retador, el desarrollo del proyecto. Por momentos debíamos parar a pensar cómo programar algo, como por ejemplo las colisiones, desde cero. Este proyecto sin dudas tuvo un alto nivel de complejidad, pero fue un proyecto bastante interesante y que disfruté pues bien aun tengo ganas de ver que mas se puede realizar utilizando OpenGL en este mismo proyecto, y es posible que pruebe a intentar hacer mejoras en un futuro cercano. Espero entrar más al desarrollo de videojuegos también en un futuro no muy lejano.

- **Diego García de los Salmones Ajuria (A01736106)**: Este proyecto representó mi primer acercamiento con el desarrollo de videojuegos y fue un verdadero reto, pues tuvimos la tarea de crear un videojuego de Pacman y para ello fue necesario aplicar diversos conocimientos de programación con C++ y OpenGL y matemáticas en un alto nivel, lo que hizo que el desarrollar este videojuego fuese un proceso largo y complejo, a pesar de esto, este proyecto me permitió identificar varias cosas que se deben considerar para el proceso de desarrollo de un videojuego, lo que me pareció bastante interesante, tomando en cuenta que los videojuegos casi siempre han estado presentes en mi vida y son uno de mis hobbies favoritos. Considero que para ser nuestro primer proyecto de este tipo, realizamos un muy buen trabajo, creando un videojuego funcional de Pacman en 2D; por supuesto que es un proyecto al que se le podrían aplicar mejoras en caso de continuar trabajando en él, que lo hagan un videojuego más retador y más cercano al juego de Pacman original, pero tal y como dije antes, el resultado obtenido es muy bueno siendo nuestro primer acercamiento al desarrollo de videojuegos, pero también me deja con ganas de seguir aprendiendo más cosas sobre esta área.

- **Raúl Díaz Romero (A01735839)**: Este proyecto fue sumamente interesante y revelador, pues toda mi vida he jugado y admirado diversos videojuegos, sin embargo, nunca indagué lo suficiente para comprender todo el desarrollo que hay detrás de la creación de uno.

  A través de este proyecto pude entender y visualizar la cantidad de esfuerzo y tiempo que requiere crear un videojuego, ya que se necesitan muchísimos conocimientos y talentos para crear un videojuego divertido y funcional, tal como el videojuego ‘Pac-Man’.

  Los principales aprendizajes que obtuve durante este proyecto se desglosan en aprender a generar/manipular texturas, Dibujar objetos, trasladar objetos, escalar objetos, manejar inputs del teclado durante la ejecución de un programa y generar matrices de modelado y proyección a través de OpenGL y C++.

Además, aprendí a cómo manejar eficientemente los diversos objetos generados en nuestro proyecto (Pacman, Fantasmas y tablero), pues cada uno de los objetos utilizados en este proyecto se relacionan entre sí de una u otra manera, un ejemplo de ello es como el movimiento de los fantasmas y el Pacman dependen completamente de sus posiciones en el tablero. Esta parte fue bastante retadora, pues manejar la posición exacta en píxeles y la intersección en que se encuentran todos los objetos fue complicado, pero una vez se logró, todo se hizo más claro y fue satisfactorio ver los frutos de nuestro esfuerzo.
Por último, las mejoras que aplicaría al proyecto serían más que nada agregar más detalles, tales como: agregar más fantasmas, que dichos fantasmas utilicen algoritmos de búsqueda para acercarse al 'Pac-Man', mantener un puntaje durante el juego, la capacidad de perder durante el juego, la capacidad de que el 'Pac-Man' pueda ingerir pastillas de poder y comer a los fantasmas, introducir efectos de sonido en el videojuego y generar múltiples niveles en el videojuego.
En conclusión, este proyecto fue retador, pero igualmente satisfactorio y grato, pues ver todos los aprendizajes adquiridos a través del producto final, completamente funcional dentro de los parámetros del reto, resulta en un sentimiento bastante agradable y pleno.

- **Rogelio Hernández Cortés (A01735819)**: Toda mi vida estuve atraído a los videojuegos, sin embargo, nunca me había imaginado el gran grado de complejidad que conlleva el desarrollo de uno. Gracias a este proyecto fui capaz de comprender algunas de las fases de desarrollo detrás de estos proyectos.
  Considero que este proyecto fue bastante retador, desde mi punto de vista, fue la demostración de la aplicación directa de el conjunto de conocimientos que hemos ido adquiriendo a lo largo de la carrera, desde que fue un refuerzo en el lenguaje C++, en estructura de datos, lógica, hasta que fue la adquisición de nuevos conocimientos como fueron las bases del OpenGL.
  En conclusión, fue un buen proyecto para introducirnos en el desarrollo de videojuegos y reforzar conocimientos elementales de nuestra área de estudio.