Instituto Tecnológico y de Estudios Superiores de Monterrey

Gráficos Computacionales

Luis Palomino Ramírez

Laberinto 3D Interactivo

Proyecto Final de Gráficas Computacionales

Autor:
Alvaro Laguna
Luis Eduardo

Matrícula:
A01133709
A01225387

April 27, 2016

Tabla de Contenido

1 Introducción

A través de este proyecto se integran herramientas gráficas tales como GLUT y OpenGL para implementar una aplicación gráfica 3D en donde se pueda simular un laberinto, este se dibuja por medio de la interpretación de una matriz, se utilizan shaders e iluminación para poder darle una buena perspectiva de el punto de vista del jugador, el cual se presenta a través de una vista de tipo FPS.

1.1 Objetivo del Proyecto

El proyecto tiene como objetivo crear un escenario o laberinto, donde por medio del teclado el jugador puede explorar éste y llegar a su salida. Existen tres vistas: una vista GUI que modifica el laberinto, una aérea para mostrar la posición actual del jugador y otra en primera persona.

1.2 Descripción del Proyecto

El proyecto consta básicamente de 3 clases:

1. Controlador principal.
2. Laberinto.
3. Jugador.

El controlador principal es aquel el cual hace referencia a dos objetos, uno de tipo Player y otro de tipo Maze, estos interactuan a través del controlador para sincronizarse y así el controlador sepa donde debe pintar cada uno de los objetos del escenario. Al principio se cuenta con un grupo de vectores vacíos. Estos vectores

contienen el id de la figura a dibujar, sus posiciones, rotaciones y escalas.

La selección de colores y niveles son en base a variables predefinidas, las cuales enlazan su valor por medio de su selección a través de menús. El pintado de las figuras es a través de un ciclo for que recorre la matriz indicada por el nivel y ejecuta las operaciones matriciales corresponientes al maze y cada uno de sus componentes. Los valores de transformación de cada figura se van modificando cada vez que se presionan estos botones en el menú.

- Avanzar: Flecha superior
- Giro Derecho: Flecha derecha
- Giro Izquierdo: Flecha izquierda

2 Arquitectura

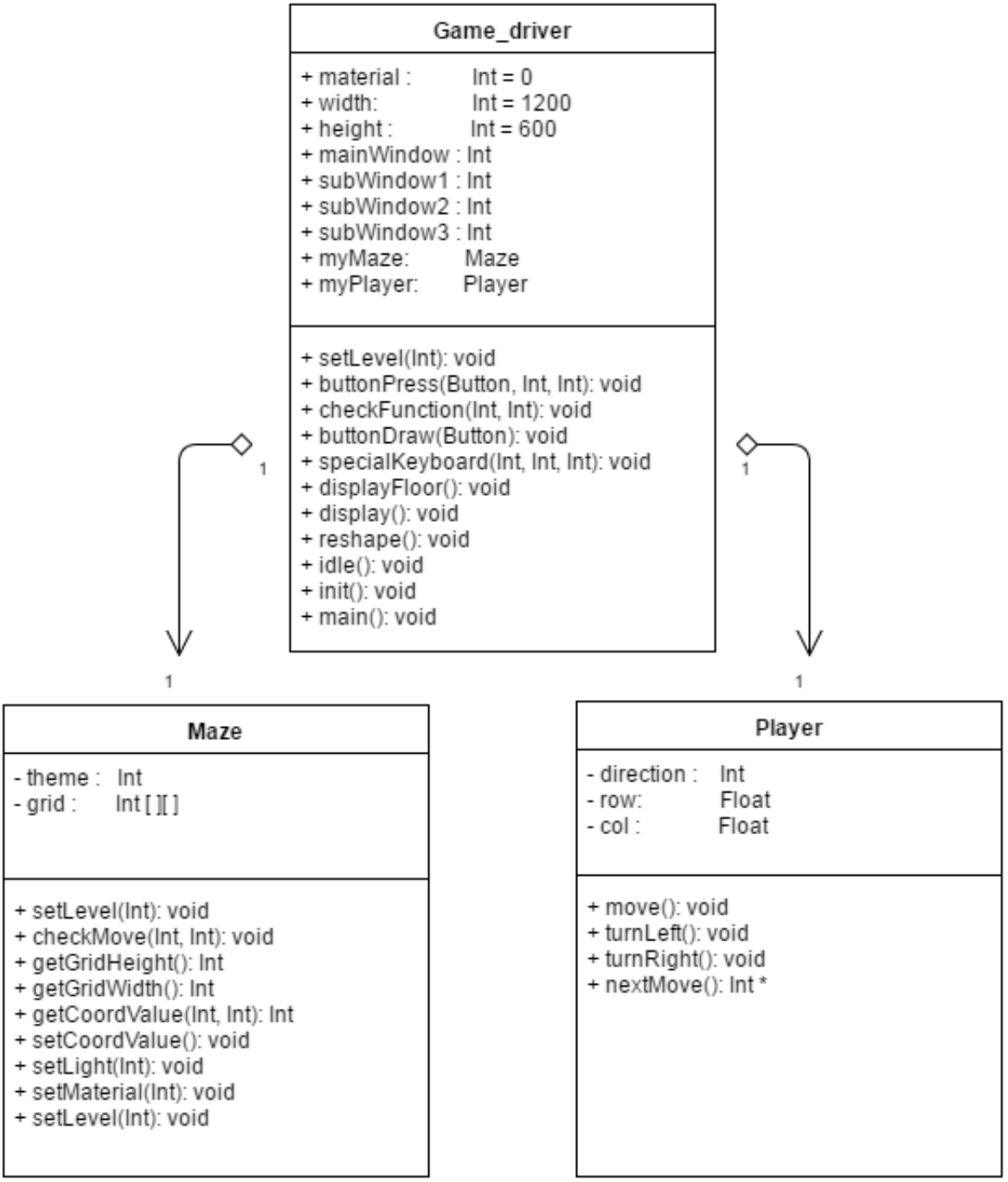

Figure 1: Diagrama UML de clases.

2.1 Game_driver

En sí éste es el punto central del programa, cuenta con diferentes variables para el ángulo de la cámara y sus vistas, pero nombraremos lo más importante, ésta se compone de una ventana principal con tres sub-ventanas, la primera para el menú interactivo donde se deciden los temas del laberinto y su nivel de dificultad, la segunda para una vista aérea de éste y la tercera para una vista en primera persona.

A continuacón se explican algunos de los métodos que no son descriptivos por sí mismos:

- void checkFunction() : Esta función se encarga de checar cual fue el botón presionado en el menú para proceder con el método correspondiente.
- void buttonDraw() : Función encargada de dibujar los botones, incluido el highlight de uno si hay hover.
- void displayFloor() : Dibuja en el suelo una cuadricula, para dar una mejor sensación de dirección.
- void specialKeyboard(int key, int xx, int yy) : Encargado de escuchar por las teclas presionadas y ejecutar los metodos correspondientes en el jugador para su movimiento.

2.2 Maze

En esta clase se define el laberinto, sus matrices en la cual se definen espacios vacios o paredes por medio de enteros y los temas de este. Además éste dependiendo del tema, que predefinido o escogido por el usuario, aplica la tonalidad de luz y colores correspondientes.

A continuacón se explican algunos de los métodos que no son descriptivos por si mismos:

- bool checkMove(int x, int y) : Sin duda uno de los métodos más importantes del maze, el cual ayuda para comunicarle al jugador si la coordenada a la cual desea moverse esta libre o es una pared.

2.3 Player

La clase representa al jugador, sirve para guardar en sus variables la dirección en la que se dirige y la posición en la que se encuentra, cuenta con métodos para girar a izquierda y/o derecha y uno para moverse al frente, que según la dirección en la que se encuentra sabe si debe aplicar operaciones de suma o resta a la columna o fila en la que se encuentra.

3 Validación

3.1 Capturas de Pantalla

Se realizaron diferentes pruebas del programa en las cuales se tomó captura de pantalla. El proyecto muestra una interfaz muy limpia y simple, además de tener una gran facilidad de uso.

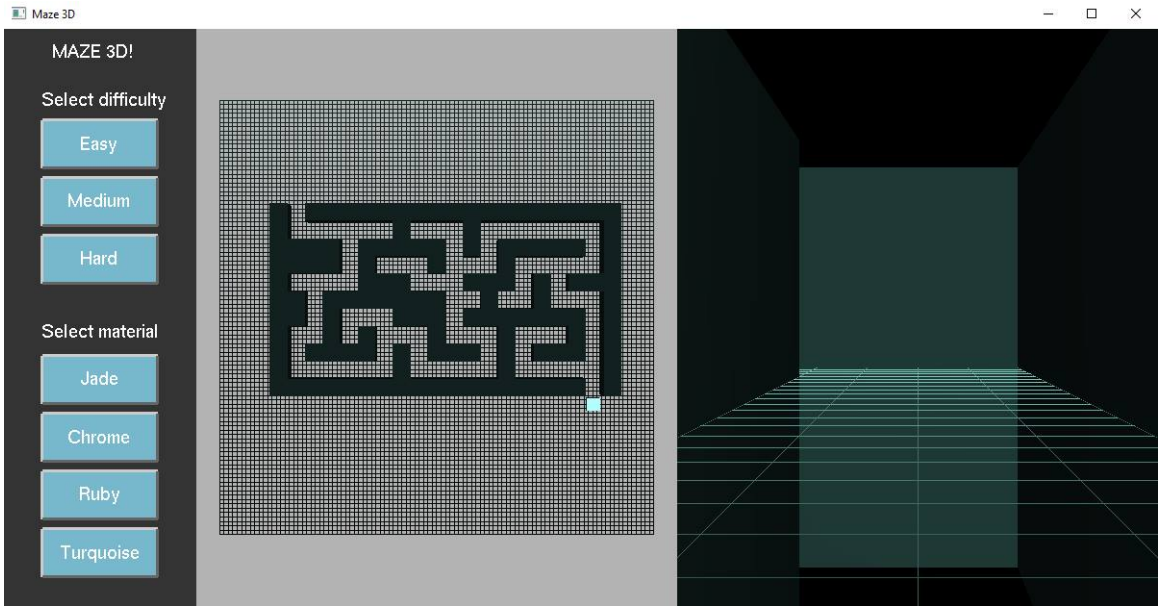

Figure 2: Dificultad difícil y material turquesa.

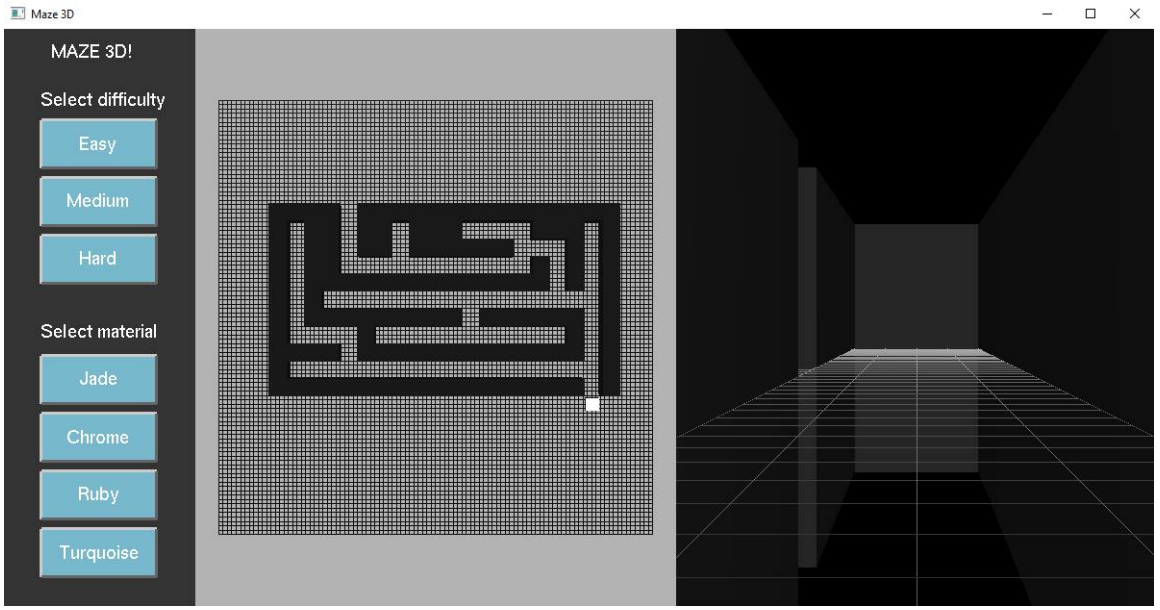

Figure 3: Dificultad intermedia y material cromo.

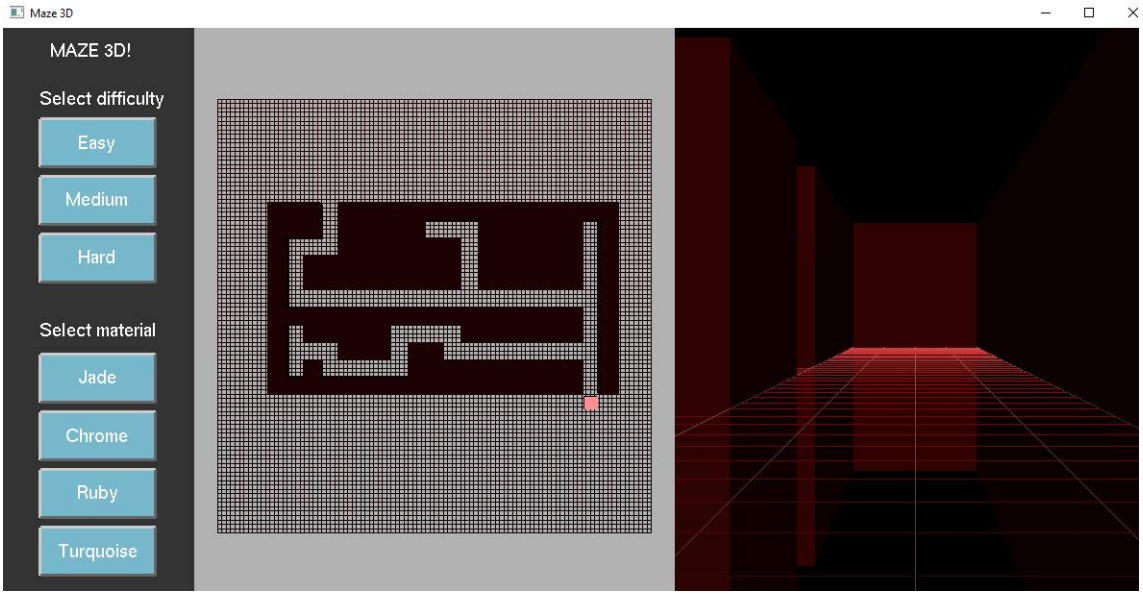

Figure 4: Dificultad fácil y material ruby.

4 Resultados

La matriz de laberinto actual se tiene como propiedad de la clase Maze y se reconfigura copiando algún laberinto almacenado cada vez que se da un click en los botones de dificultad. El material del laberinto se reconfigura antes de pintar según la selección del botón que se haya hecho. El jugador sólo es capaz de moverse por caminos del laberinto, lo cual impide que atraviese paredes o se mueva libremente después de terminar el laberinto.

5 Conclusiones

Nos agradó bastante el poder integrar diferentes conceptos y mismas prácticas del curso. De hecho nos complicamos un poco al pensar cómo integrar las partes de opengl para la comunicación entre clases, lo cual resolvimos posteriormente. Consideramos que el proyecto quedó simple pero eficiente. A lo largo del curso aprendimos el core de como funcionan las gráficas computacionales, las diferentes operaciones que se aplican y como se modelan matemáticamente en matrices, fue un curso interesante y una buena base para quienes decidan incursionar en esta área de la profesión.

6 REFERENCIAS

Palomino, L. (n.d.). Presentaciones y Programas del M. Luis Palomino. Retrieved April 22, 2016, from http://miscursos.itesm.mx/webapps/portal/frameset.jsp?url=/webapps/blackboard/execute/launcher?type=Courseid=3406801url=

The Industry's Foundation for High Performance Graphics. (n.d.). Retrieved April 27, 2016, from https://www.opengl.org/documentation/