Ciencia de la Computación

Computación Gráfica

Informe: Cubo Rubik

Docente Manuel Eduardo Loaiza Fernández

Julio 2023

Autores:
Sebastian Andre Paz Ballon
Joey Patrick Flores Davila

Semestre VII
2023-01

Informe: Cubo Rubik

Sebastian Andre Paz Ballon
sebastian.paz.ballon@ucsp.edu.pe

Joey Patrick Flores Davila
joey.flores@ucsp.edu.pe

Julio 2023

1 Introducción

El trabajo final consiste en realizar el modelado, renderizado y animación de un cubo Rubik en OpenGL **Figure 1**. Para lograr el objetivo, se utilizaron conceptos básicos para el desarrollo de aplicativos gráficos tridimensionales en OpenGL como:

- El modelado utilizando Vertex Buffer Objects (VBOs) y Vertex Array Objects (VAOs), el cual es una técnica esencial en la computación gráfica. Consiste en almacenar y administrar eficientemente los datos de los vértices de un modelo 3D. Los VBOs se encargan de guardar los datos de los vértices en la memoria de la tarjeta gráfica, mientras que los VAOs registran la configuración de los VBOs. Esta estrategia optimiza la transferencia de datos hacia la GPU y simplifica el proceso de renderizado de modelos complejos. Al utilizar VBOs y VAOs, se logra un mejor rendimiento y una mayor calidad visual en las aplicaciones gráficas. En resumen, el uso de VBOs y VAOs es fundamental para el modelado eficiente en la computación gráfica y contribuye al éxito de nuestras creaciones visuales.[1]

- Programación de Shaders tales como el vertex shader y fragment shader. Ambos son componentes clave en la programación gráfica. El vertex shader se encarga de transformar y posicionar los vértices de un objeto en el espacio de pantalla, aplicando operaciones como rotaciones, escalados y traslaciones. Por otro lado, el fragment shader se encarga de determinar el color final de cada píxel o fragmento en la pantalla, aplicando efectos visuales, sombreado y texturas. En conjunto, el vertex shader y el fragment shader permiten crear efectos visuales personalizados y realistas en las aplicaciones gráficas 3D.Se utilizaron tipos de dato in, out y uniform.[2]

- Animación del Cubo Rubik utilizando matrices de transformación para poder aplicar rotaciones, traslaciones y escala. Para lograrlo, se utilizó la librería GLM(OpenGL Mathematics), la cual es una biblioteca de matemáticas para gráficos en 3D que proporciona funciones y estructuras de datos útiles para trabajar con vectores, matrices y transformaciones. Está diseñada para ser compatible con OpenGL y se utiliza ampliamente en computación gráfica para realizar operaciones matemáticas comunes, como transformaciones de coordenadas, interpolaciones, cálculos de normalización y proyecciones. GLM ofrece una interfaz intuitiva y eficiente, lo que facilita el desarrollo de aplicaciones gráficas utilizando OpenGL.[3]

- Programación de la matriz de proyección ortogonal y en perspectiva que nos permite visualizar de distintos modos nuestro Cubo Rubik.[4]

- Programación de una cámara lookAt que permite la visualización de la escena. Dicha cámara se basa en el concepto de que la cámara siempre está mirando hacia un objetivo específico. Al utilizar la función "lookat", se especifica la posición de la cámara, la posición del objetivo al que la cámara debe apuntar y un vector de "arriba" que define la orientación vertical de la cámara. Esta técnica calcula automáticamente la matriz de transformación necesaria para posicionar y orientar la cámara de manera adecuada. [5]

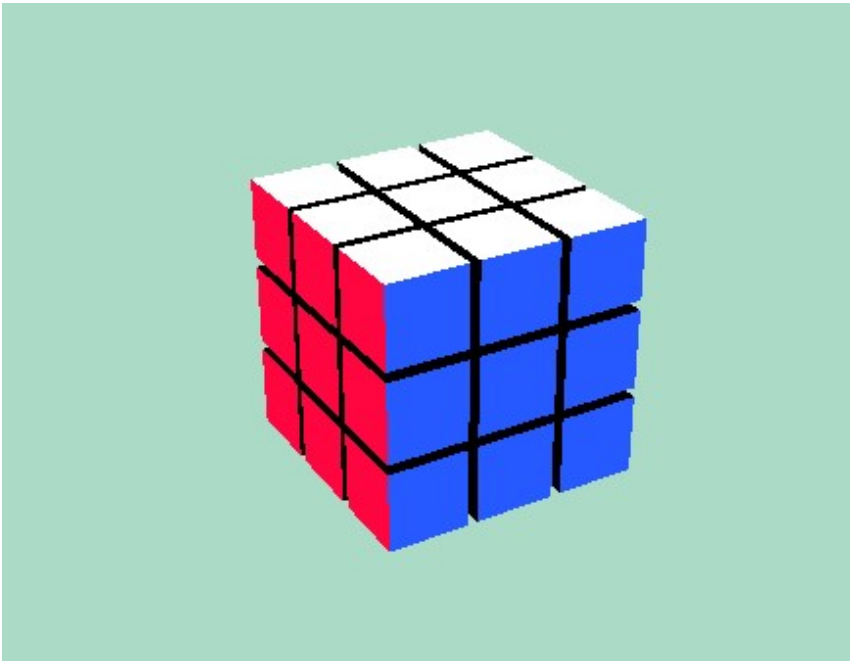

Figure 1: Imagen del cubo de rubik.

2 Animación implementada

La animación consiste en las siguientes seis etapas:

- **Elevación de cubitos:** En esta etapa el cubo se va a dividir en tres niveles, el nivel superior, el nivel del medio y el nivel inferior. Cada cubito de cada nivel va a ser trasladado en el eje Y hacia arriba, manteniendo una forma ordenada. La elevación va a empezar por el nivel superior.
- **Caída de cubitos:** En esta etapa cada cubito de cada nivel va a ser trasladado en el eje Y hacia abajo, excepto los centros de cada cara. Al final va a quedar el cubo rubik con huecos en sus caras.
- **Unificación del cubo:** En esta etapa los centros de las caras del cubo se van a trasladar de tal forma que el cubo se complete. Las caras se trasladarán una a una.
- **Rotación:** En esta etapa el cubo va a empezar a girar en sentido horario.
- **Explosión:** En esta etapa cada cubito va a ser trasladado en los ejes X, Y o Z. De tal forma que haga la ilusión que hubo una explosión en el cubo.
- **Implosión:** En esta etapa cada cubito hará una traslación inversa a la que fue realizada en la explosión. Esto va a generar que el cubo rubik se rearme y vuelva a su estado inicial.

3 Funcionalidades

El proyecto cuenta con las siguientes funcionalidades:

3.1 Movimientos de las caras

En **Figure 2** se muestra qué cara es rotada según el tipo de movimiento. Las teclas que se deben presionar para mover las caras del cubo rubik y para cambiar la orientación del giro entre horario y antihorario son las siguientes:

- Flecha de arriba: Cara superior (U).
- Flecha de abajo: Cara inferior (D).
- Tecla F: Cara frontal (F).
- Tecla B: Cara trasera (B).
- Flecha derecha: Cara derecha (R).

- Flecha izquierda: Cara izquierda (L).
- Tecla Shift del lado derecho: Cambiar orientación del giro (horario/antihorario).

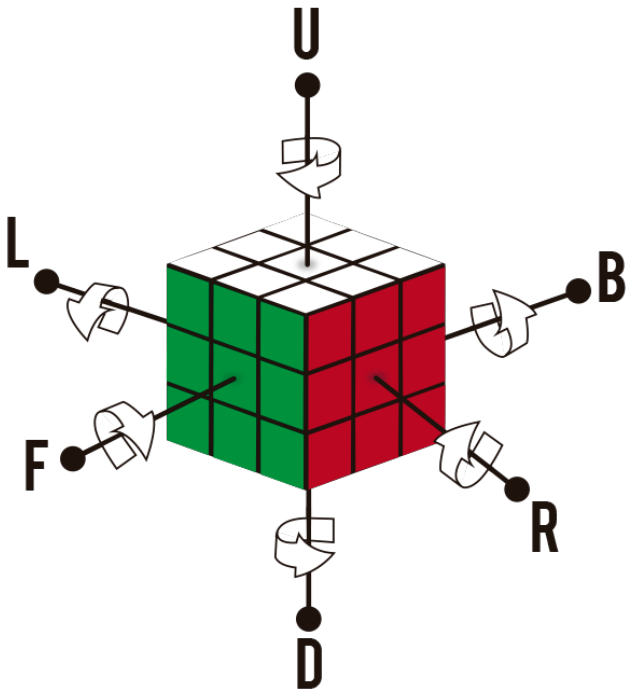

Figure 2: Movimientos del cubo rubik.

3.2 Resolución del Cubo Rubik

Al presionar la tecla Q, se aplica el algoritmo de Kociemba[6] para mostrar su solución. Como entrada, el algoritmo recibe el estado actual del cubo con todas sus rotaciones hasta el momento y el algoritmo retorna los movimientos necesarios para su solución.

3.3 Movimiento de la cámara

La cámara se puede controlar con el mouse, y con el teclado. Las teclas para controlar la cámara son las siguientes:

- Tecla W: Mover hacia arriba.
- Tecla A: Mover hacia la izquierda.
- Tecla S: Mover hacia abajo.
- Tecla D: Mover hacia la derecha.

3.4 Inicio de la animación

Para iniciar la animación se debe presionar la tecla Z.

4 Problemas encontrados en la implementación

Durante la implementación del proyecto se presentaron los siguientes inconvenientes:

- **Poder mover la cámara mientras se aplica la animación:** Al momento de realizar movimientos en las caras del cubo, sí se puede mover la cámara para visualizar el cubo, sin embargo, en las animaciones existen bucles for que limitan esta funcionalidad. Se intentó actualizar la cámara

dentro de la misma animación pero no dió resultado, es por esto que al momento de iniciar la animación, la cámara se queda estática hasta que se termine la animación.

- **Modificar el Solver:** El solver encontrado estaba escrito en python, por lo que se tuvo que aplicar las modificaciones necesarias para trabajar con C++.
- **Aplicar iluminación al cubo:** Aplicar iluminación al cubo significa una modificación en el programa y por temas de tiempo faltó aplicar esta modificación.

5 Próximos pasos

Los siguientes pasos que se planea realizar es la implementación de la iluminación en el aplicativo del cubo rubik, utilizando los tutoriales de la página OpenGL como guía. También se va a implementar el movimiento de la cámara mientras se realice la animación del cubo.

6 Conclusión

Durante la realización de este trabajo hemos aprendido cómo utilizar los conceptos básicos para desarrollar un aplicativo de computación gráfica. Tales como aplicar la matemática en las tranformaciones de los objetos creados. Se ha cumplido con el objetivo de culminar con el desarrollo el cubo rubik en OpenGL utilizando C++.

References

[1] J. Vries, "LearnOpenGL - Hello Triangle", Learnopengl.com, 2023. [Online]. Available: https://learnopengl.com/Getting-started/Hello-Triangle. [Accessed: 02 Jul 2023].

[2] J. Vries, "LearnOpenGL - Shaders", Learnopengl.com, 2023. [Online]. Available: https://learnopengl.com/Getting-started/Shaders. [Accessed: 02 Jul 2023].

[3] J. Vries, "LearnOpenGL - Transformations", Learnopengl.com, 2023. [Online]. Available: https://learnopengl.com/Getting-started/Transformations. [Accesed: 02 Jul 2023].

[4] J. Vries, "LearnOpenGL - Coordinate Systems", Learnopengl.com, 2023. [Online]. Available: https://learnopengl.com/Getting-started/Coordinate-Systems. [Accesed: 02 Jul 2023].

[5] J. Vries, "LearnOpenGL - Camera", Learnopengl.com, 2021. [Online]. Available: https://learnopengl.com/Getting-started/Camera. [Accessed: 02 Jul 2023].

[6] Kociemba, H. (s/f). RubiksCube-TwophaseSolverFiveFaces: Solve Rubiks cube without turning the Back face in less than 22 moves on average. This is a modified version of the RubiksCube-TwophaseSolver repository.