



Departamento de Ciencia  
de la Computación

**PROGRAMA PROFESIONAL:**  
**CIENCIA DE LA COMPUTACIÓN**

**TÍTULO DEL TRABAJO:**  
**PRESENTACIÓN FINAL DEL CUBO DE RUBIK**

**CURSO:**  
**COMPUTACIÓN GRÁFICA**

**ESTUDIANTES:**  
Becerra Sipiran, Cledy Elizabeth  
(cledy.becerra@ucsp.edu.pe)  
Oviedo Sivincha, Massiel Oviedo  
(massiel.oviedo@ucsp.edu.pe)  
Villanueva Borda, Harold Alejandro  
(harold.villanueva@ucsp.edu.pe)

**SEMESTRE: VII**

**AÑO: 2023**

**“El alumno declara haber realizado el presente trabajo de acuerdo a las normas  
de la Universidad Católica San Pablo.”**

## **Índice**

Introducción-----	3
Descripción de animación-----	3
Funcionalidades del programa-----	4
Problemas encontrados en la implementación-----	5
Conclusiones-----	6
Referencias-----	6

## **Introducción**

El trabajo final consiste en el modelado, renderizado y animación de un cubo de Rubik en OpenGL. Se implementaron los conceptos básicos para el desarrollo de aplicativos gráficos tridimensionales en OpenGL, tales como:

### 1. Modelado del cubo de Rubik

Primera etapa donde se creó la geometría del cubo de Rubik utilizando primitivas geométricas como cubos o cuadrados, definiendo las posiciones y dimensiones de cada una de sus partes.

### 2. Renderizado

Segunda etapa donde se implementó el algoritmo de renderizado para mostrar el cubo de Rubik en la pantalla. Esto involucró la configuración de la cámara y la proyección del objeto en la pantalla.

### 3. Texturizado

Tercera etapa se aplicaron texturas a cada una de las caras del cubo de Rubik para simular los colores de los diferentes stickers. Esto permitió representar visualmente el estado del cubo y el correcto uso de texturas en el modelo.

### 4. Interacción con el usuario

Se implementaron controles que permiten al usuario interactuar con el cubo de Rubik, cómo girar las capas del cubo en diferentes direcciones o mezclar las posiciones de las piezas.

### 5. Solver

En la etapa del solver, se realizó una cuidadosa evaluación de diferentes métodos y algoritmos para resolver el cubo de Rubik de manera eficiente. Entre las diversas opciones disponibles, se decidió utilizar el método de Kociemba debido a sus características y resultados comprobados.

### 6. Animación

Se desarrollaron algoritmos para animar el cubo de Rubik, permitiendo realizar movimientos suaves y fluidos entre los diferentes estados del cubo.

En resumen, el proyecto consistió en aplicar los conceptos básicos de OpenGL para crear un aplicativo gráfico tridimensional que permitiera modelar, renderizar y animar un cubo de Rubik.

## **Descripción de animación**

Las animaciones escogidas para el proyecto fueron dos. La primera se llama "Respiración Continua", en la cual los pequeños cubos que componen el cubo de Rubik son repelidos y atraídos, simulando un efecto de respiración. Esta animación puede ser controlada, y se explicará más adelante cómo hacerlo.

La segunda animación se denomina "Snake Animation". En esta animación, se usan 20 cubos rubik para la creación de la palabra CS. Durante la ejecución del programa, esta palabra dará el efecto de

“Respiración continua”, creando un efecto usando ambas animaciones. Agregando que la animacion simula snakes saliendo de el punto central con movimientos left y right.

Por ultimo se gallego una tercera animación que se denomina “Expansion Animation”. En esta animación, se usan 17 cubos rubik para la creación de la palabra CS. Durante la ejecución del programa, esta palabra dara el efecto de “Respiración continua”, creando un efecto usando ambas animaciones. Agregando que esta animación podria simular una explosion, los movimientos son diagonales.

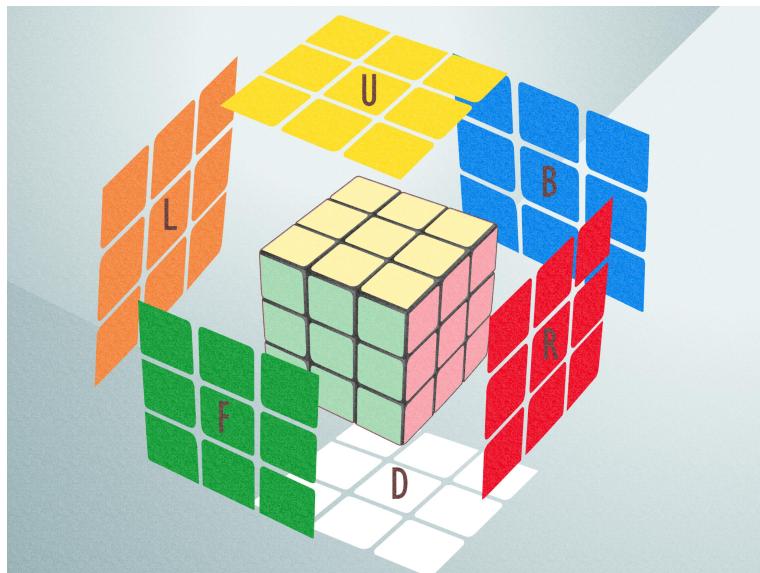
### **Funcionalidades del programa**

El programa ofrece diversas capacidades tanto a nivel del proyecto principal, que es el cubo de Rubik, como también en relación a los conceptos de la computación gráfica, como el uso de la cámara y la proyección de la forma. Asimismo, se han implementado herramientas tales para facilitar la resolución del cubo, como algoritmos de solución automática y la posibilidad de desordenar aleatoriamente para luego ser ordenados.

- Tenemos 3 animaciones, la primera animación “snake” consiste en mover algunos cubos del Rubik's Cube en una secuencia específica durante un período de tiempo determinado. En este caso, la secuencia de movimientos es predefinida y se ejecuta cada cierto tiempo. En el código, se itera a través de los cubos del Rubik's Cube (excepto el cubo principal) y se les aplica un movimiento en cada iteración.
- En la animación 2, el “breathe”, se crea un efecto de "respiración" en el Rubik's Cube, expandiéndolo y luego contrayéndolo repetidamente. Para lograr esto, se utiliza una variable de incremento (incremento2) que se incrementa continuamente. Dependiendo del valor de incremento2, se ajusta la posición de los cubos para lograr el efecto de expansión y contracción. Así como se activa, se puede desactivar en cualquier momento.
- La animación 3, la “expansión” implica una secuencia específica de movimientos en los cubos del Rubik's Cube. En este caso, algunos de los cubos se mueven hacia arriba y hacia la derecha, mientras que otros se mueven hacia abajo y hacia la izquierda. Esta secuencia de movimientos se repite continuamente mientras el programa se ejecuta. Los movimientos se realizan utilizando las funciones move\_north(), move\_south(), move\_east(), move\_west() que realizan traslación en los objetos Rubik correspondientes. Asimismo, se realiza intercalando transformaciones en los ejes.

### **Manejo de cubo**

Las siguientes instrucciones presentan una serie de movimientos y acciones relacionadas con el famoso Cubo de Rubik. Se describen las teclas correspondientes a cada movimiento, así como las funciones de resolver y desordenar el cubo.



- Tecla Q: Movimiento R en sentido horario
- Tecla W: Movimiento L en sentido horario
- Tecla E: Movimiento U en sentido horario
- Tecla R: Movimiento D en sentido horario
- Tecla T: Movimiento F en sentido horario
- Tecla Y: Movimiento B en sentido horario
- Tecla Z: Resolverá el cubo que se transformó con los movimientos clásicos del Cubo de Rubik
- Tecla X: Desordenará aleatoriamente el cubo de Rubik
- Tecla C: Resolverá el cubo de Rubik desordenado aleatoriamente.
- Tecla N: Realizará la animación de expandir el cubo.
- Tecla M: Realizará la animación de contraer el cubo.
- Tecla 1: Activará la animación Snake Animation
- Tecla 2: Activará la animación Breath Animation
- Tecla 3: Activará la animación Expansión Animation
- Tecla right: Movimiento de cámara hacia la derecha en el eje x.
- Tecla left: Movimiento de la cámara hacia la izquierda en el eje x.
- Tecla up: Acercamiento de la cámara.
- Tecla down: Alejamiento de la cámara.

### **Problemas encontrados en la implementación**

- En la primera etapa de modelado se tuvo problemas con la creación de estructuras del cubo rubik. Inicialmente se acordó conjuntamente una clase Cubito con una función que animaría y juntaría los 26 cubos. Aunque no hubo problemas en la primera entrega debido a que cumplía con lo pedido, en las siguientes etapas se vio necesario aumentar una clase que maneja nuestras camadas y sus correspondientes animaciones.
- Otro problema encontrado en la primera etapa fue el cálculo de cantidad de nuestros cubitos; donde finalmente se decidió pasar de 26 cubitos a 27 cubitos . Debido a que cada uno de los 26 cubitos exteriores tiene una ubicación fija en relación con el cubo central y se mueve en conjunto con las capas exteriores durante las rotaciones. Además, el cubito central (cubito 27) no cambia de posición y permanece como una pieza fundamental para la estructura y

resolución del cubo de Rubik. Si omitimos el cubito central al crear el cubo de Rubik , se perdería la representación precisa de la estructura y funcionalidad del cubo de Rubik. Se tornó difícil realizar rotaciones y movimientos correctos sin la presencia del cubo central, lo que afecta las animaciones y la resolución del cubo.

- En la segunda etapa donde estábamos viendo las rotaciones, tuvimos complicaciones con la adecuación de las texturas a las animaciones realizadas. Principalmente recurrimos a una animación que constaba en el cambio de textura actualizadas en el cubo sin la animación de girar una camada. Cuando la camada hacia las animaciones correspondientes estas no se veían reflejadas en la animación pero si al final. Entonces lo que hicimos fue recorrer las texturas por coordenadas.
- El problema principal de la última fase es la implementación de la rotación automática de la cámara, además de agregarle iluminación. Esto se nos complicó debido a que usamos texturas y no aprendimos a combinar ambas cosas, sin embargo, sabemos que con un poco más de tiempo, esto habría sido posible.

## Conclusiones

Se ha creado una aplicación que emplea los fundamentos esenciales de la computación gráfica. Se ha logrado desarrollar la mayoría de los aspectos mencionados, tanto en la aplicación de los conceptos matemáticos como en los aspectos técnicos de la programación. Considero que este trabajo ha sido de gran utilidad, ya que me ha llevado a investigar los elementos clave para el desarrollo de programas gráficos.

## Referencias

- [1] J. Vries, "LearnOpenGL - Hello Triangle", Learnopengl.com, 2021. [Online]. Available: <https://learnopengl.com/Getting-started/Hello-Triangle>. [Accessed: 05-Dec- 2021].
- [2] J. Vries, "LearnOpenGL - Shaders", Learnopengl.com, 2021. [Online]. Available: <https://learnopengl.com/Getting-started/Shaders>. [Accessed: 05- Dec- 2021].
- [3] Vriez, J., 2021. LearnOpenGL - Textures. [online] Learnopengl.com. Available at: <<https://learnopengl.com/Getting-started/Textures>> [Accessed 5 December 2021].
- [4] J. Vries, "LearnOpenGL - Transformations", Learnopengl.com, 2021. [Online]. Available: <https://learnopengl.com/Getting-started/Transformations>. [Accessed: 05-Dec- 2021].
- [5] J. Vries, "LearnOpenGL - Coordinate Systems", Learnopengl.com, 2021. [Online]. Available: <https://learnopengl.com/Getting-started/Coordinate-Systems>. [Accessed: 05- Dec- 2021].
- [6] J. Vries, "LearnOpenGL - Camera", Learnopengl.com, 2021. [Online]. Available: <https://learnopengl.com/Getting-started/Camera>. [Accessed: 05- Dec- 2021].
- [7] S. Ahn, "Quaternion", Songho.ca, 2021. [Online]. Available: <http://www.songho.ca/math/quaternion/quaternion.html#reference>. [Accessed: 06- Dec- 2021].