Cuarto Avance proyecto de programación Cubo de Rubik

Mauricio Carazas Segovia
Computación Gráfica - CCOMP7-1
Escuela Profesional de Ciencia de la Computación
Universidad Católica San Pablo
mauricio.carazas@ucsp.edu.pe

El Cubo de Rubik es un famoso rompecabezas mecánico que ha capturado la atención global gracias a sus características distintivas. Conocido como un juguete clásico de desarrollo mental, ha sido utilizado por muchos investigadores y científicos para la investigación y el avance tecnológico. Inventado en 1974 por el escultor y profesor de arquitectura húngaro Ernő Rubik, este rompecabezas tridimensional originalmente se llamó Cubo Mágico.

El proyecto asignado implica la programación de un Cubo de Rubik de tamaño 3x3. Esta versión del cubo permite mover las capas del rompecabezas, girando alrededor del eje central, así como desplazar la cámara alrededor del cubo de Rubik. Además, el cubo se mostrará con texturas en cada una de sus caras, donde cada cara tendrá su propia textura.

Animación:

La animación implementada es un lluvia de estrellas, Cada objeto estrella tiene un estado de animación que puede ser "M" (movimiento), "G" (giro) o "B" (movimiento inverso).

El método animacion_estrella se encarga de realizar la animación de la estrella en función de su estado actual y los parámetros recibidos. La animación se divide en tres fases: movimiento, giro y movimiento inverso. Dependiendo del estado actual de la estrella, se realizan diferentes operaciones de traslación y rotación en el objeto Polygon asociado.

El método draw se utiliza para dibujar la estrella en la ventana GLFW.

En el bucle principal del programa, se llama al método animacion_estrella de cada objeto estrella para actualizar su estado y realizar la animación. Luego se llama al método draw para dibujar las estrellas en la ventana. Esto se repite en cada iteración del bucle, lo que crea la ilusión de animación.

Funcionalidades de su programa (Keyboard, movimiento de cámara):
El código proporcionado muestra una función llamada processInput que procesa la entrada del teclado a través de la biblioteca GLFW. Aquí está la descripción de las funcionalidades principales de esta función:

Cierre de la ventana: Si se presiona la tecla Escape (GLFW_KEY_ESCAPE), la función glfwSetWindowShouldClose se llama para indicar que la ventana debe cerrarse.

Control de movimiento de cámara: Si se presionan las teclas de flecha hacia arriba (GLFW_KEY_UP), hacia abajo (GLFW_KEY_DOWN), hacia la izquierda (GLFW_KEY_LEFT) o hacia la derecha (GLFW_KEY_RIGHT), los valores de pitch y yaw se ajustan según la velocidad de la cámara (cameraSpeed). Estas variables probablemente se utilizan para controlar la orientación de la cámara y se actualizan para lograr el movimiento deseado.

Animaciones: Si se presionan ciertas teclas (R, L, U, D, F, B) y la variable condition_input tiene un valor específico ("N"), se agrega una cadena correspondiente ("R", "L", "U", etc.) al vector reg_mov y se imprime un mensaje indicando el inicio de una animación particular. Además, se actualiza condition_input con el valor de la tecla presionada para indicar que se ha iniciado una animación.
Estas teclas indican el movimiento de ciertas camadas y están organizadas según la inicial de su Posición
R(right): camada derecha

L(left): camada izquierda
U(up): camada superior
D(down): camada inferior
F(front): camada frontal
B(back): camada trasera

Obtención de solución: Existe un bloque de código comentado que muestra una sección para la tecla K (GLFW_KEY_K). Esta sección parece estar relacionada con la obtención de una solución para un cubo y contiene algunas llamadas a funciones no mostradas en el código proporcionado.

Problemas encontrados en la implementación:

1.Control simultáneo de animaciones: Cree la variable condition_input para controlar el inicio de las animaciones y evitar que se inicien múltiples animaciones simultáneamente. Sin embargo, si el usuario presiona varias teclas de animación al mismo tiempo, es posible que se registren múltiples animaciones en el vector reg_mov y se cambie el valor de condition_input más de una vez antes de que se procese la animación actual. Esto podría llevar a resultados inesperados o a conflictos en la lógica de animación.

2.Falta de manejo de teclas de liberación: El código actual solo verifica si se presiona una tecla en un momento dado. No maneja las teclas de liberación, lo que significa que no hay una lógica implementada para detener o finalizar las animaciones cuando se deja de presionar una tecla. Esto podría llevar a problemas de control y comportamiento inesperado.

3.Problemas con la cámara: Inicialmente se tenía control de la cámara con el mouse y esto resultaba mucho más interactivo para controlarla; sin embargo las librerías utilizadas generaban conflicto con el cmake preparado del proyecto, es por esto que se cambió a un control de

cámara por teclado que si bien es mucho más básico, cumple con su función y no genera errores.

Próximos pasos:
Lo siguiente en la implementación es mejorar la animación o reemplazarla por una mejor y añadirle iluminación al proyecto.

Referencias:

[1]Utilizamos el algoritmo de dos fases de Kociemba modificado para obtener la solución en 20 movimientos o menos, que se encuentra aquí: https://github.com/muodov/kociemba/tree/master/kociemba/ckociemba

[2]https://learnopengl.com/Getting-started/Camera

[3]https://rosettacode.org/wiki/Solve_a_Rubik%27s_cube