Paulo Sandoval Suazo

Nancy Hitschfeld K.

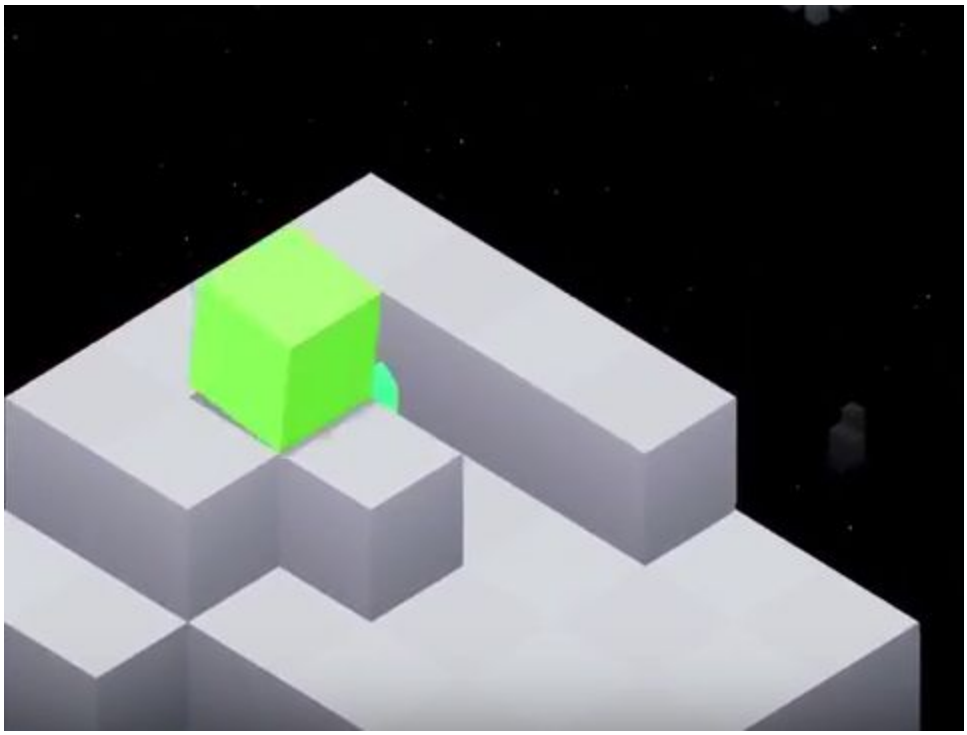
CC3501 - Computación Gráfica

28 de Junio de 2017

Tarea 3: EDGE

Breve Descripción del problema

El problema consiste en diseñar y desarrollar un videojuego similar a EDGE Extended pero simplificado.



El juego original consiste en mover un cubo por una plataforma tridimensional con distintas alturas, coleccionando prismas y superando obstáculos. Los mapas se vencen al coleccionar todos los prismas de la etapa y llegando al final de la etapa.

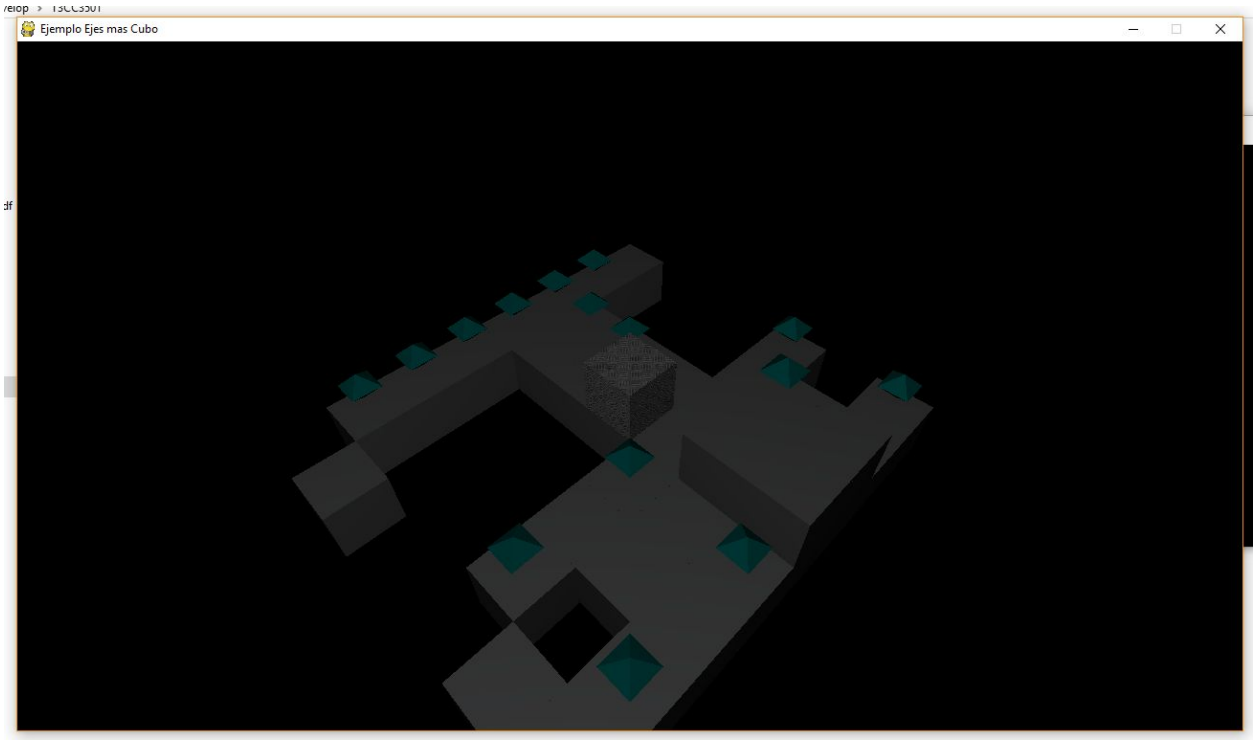
Descripción de la Solución

La solución pedida por la Tarea es un programa de Python que utilizando las librerías PyGame y PyOpenGL represente la escena que se vive en EDGE Extended.

Esta escena debe tener un cubo que se mueva respecto a los comandos del teclado (4 direcciones cardinales), atrapar prismas, etc.

Las interacciones del cubo con el ambiente deben incluir subir y bajar de altura de cubos. Y en caso de no estar pisando un cubo (pisando el aire/vacío?) el juego se termina.

El juego se termina una vez que el jugador logra atrapar todos los prismas de una etapa.



Resultados Obtenidos

Mi implementación está en https://github.com/ez2torta/T3CC3501

Utilicé la librería desarrollada por Pablo Pizarro https://github.com/ppizarror/pyopengl-toolbox y me basé en un par de ejemplos de él para comenzar a crear cubos.

Si bien pude empezar a desarrollar muy rápido con la librería pyopengl-toolbox, para funcionalidades más avanzadas no es tan útil. Creo que en parte es problema de python y OpenGL, ya que si no se utiliza una librería como la desarrollada por Pablo, el código hubiese quedado demasiado repetitivo y al estilo C.

Nota: No es que programar en C sea malo, pero es menos pseudocódigo entendible que Python en si.

Las clases que utilicé fueron:

- Cubo: Jugador Principal del videojuego, se encarga de mover el Cubo que representa el jugador
- Matriz: Crea el mapa, utilizando una matriz con números 0, 1 y 2 se crean cubos para que el jugador pueda moverse
- Prisma: Crea prismas, utilizando una matriz similar a la anterior para darle locaciones a los prismas en la etapa.

La clase principal (prueba.py) es la utilizada para correr el videojuego y en estas se encuentran las funciones principales del programa, dentro de ellas destacan:

- Movimiento de la cámara
- Mandos del Teclado (Arriba, Abajo, Izquierda, Derecha)
- Interacciones entre Objetos del Sistema
- Texto de Salida Estándar
- Terminar el Juego

La música utilizada es:

Vangelis - Chariots of Fire -> La utilicé por la velocidad del juego, no lo hice muy rápido y la quest se vuelve más emocionante.

Los efectos de sonido:

Al recolectar un prisma -> Metal Slug 3 Coin (sonido de insertar ficha en Metal Slug 3)
Al mover el cubo -> Moving Robot (un sonido con licencia creative commons que encontré en la internet)

El videojuego muestra su estado por la salida estandar, y se termina una vez recolectado todos los prismas o si el jugador mueve su cubo hacia una zona que no tiene cubos.

No hay selección de Mapa ni textos en el videojuego.

Discusión sobre las dificultades encontradas

Me enfoque en tratar de sacar la mayor cantidad de funcionalidades *core* del videojuego.

En particular el primer problema encontrado fue el que más me frustró:

No pude hacer rotar el cubo por su arista, y esto es por un tema del uso de pyopengl-toolbox. Como partí basando mi código en esto, las rotaciones y relacionados estaban referenciadas al centro de los ejes euclidianos.

No pude lograr una rotación que no fuese en relación a los ejes centrales del cubo o una rotación respecto a los ejes euclidianos. Bajo esto, hice las rotaciones bajo los propios ejes del cubo mientras se mueve de manera dispar.

Tampoco logré mover la cámara en relación constante con el cubo.. Pretendía que estuviese siempre al centro de la cámara pero no pude hacerlo.

Lo último que más me frustró fue incrustar pygame con pyopengl.. No pude utilizar ninguna clase de pygame que incluyera texto en la pantalla, al parecer debía pasar todo el texto por OpenGL puesto que me lanzaba errores pygame del tipo “Esta ventana es de GLUT” o similares.

Por lo último, los mensajes del juego los estaba tirando por Salida estandar (Consola).

Decidí no seguir implementando el resto de features (cubos que se caen y cubos que cambian la dirección) por temas de tiempo. Aunque dada la implementación actual no es complicada la realización de estos features (crear clases nuevas para los nuevos tipos de cubo y ubicarlos en el mapa utilizando una matriz similar a las clases anteriores y agregando la inteligencia de las interacciones en la clase principal).

Conclusiones

Para trabajar con OpenGL es necesario no utilizar abstracciones o extenderlas lo suficiente para poder darle toda la funcionalidad necesaria a la librería.

Si bien Python es un lenguaje muy ágil y que corre en muchas plataformas creo que el lenguaje indicado para trabajar bajo OpenGL es C++ debido a cómo deben ser programadas las funciones de OpenGL llamadas en sucesión.

El desarrollo en 3D me pareció más divertido que en 2D pero al utilizar la librería de Pablo resultó frustrante para el desarrollo íntegro de la Tarea, si bien no era obligación utilizar esa librería quería partir de un suelo estable y me fui por esto.