UNIVERSIDAD NACIONAL DE INGENIERÍA

DACTIC

INGENIERÍA EN COMPUTACIÓN

Programación Grafica

Integrante

Conde Vasconcelos Sleyter Antonio // Carnet 2022-0308U

Castañeda Zambrano Saúl Efrain // Carnet 2022-0481U

Montenegro Munguía Angel Amaru // Carnet 2022-0391U

Profesor	**Grupo**
Danny Oswaldo Chávez Miranda	3T1-CO

Fecha de entrega

Viernes 05 de julio del 2024

Objetivo general

Desarrollar un videojuego en un entorno 3D que simule la mecánica y la jugabilidad del clásico juego Bricks Breaker, proporcionando una experiencia interactiva y atractiva para los usuarios mediante el uso de gráficos tridimensionales, física realista y controles intuitivos.

Objetivos específicos

1. Implementar un motor de gráficos con Python y OpenGl que permita simular los comportamientos de la pelota y los bloques en un entorno 3D

2. Utilizar mecánicas simples para que se vuelva más fácil al usuario conectar con el programa

Introducción

El siguiente proyecto fue creado con el fin de desarrollar un videojuego atractivo para todo aquel usuario que esté interesado en probarlo. Esta idea nace como un desafío para el grupo de desarrollo, con el objetivo de ofrecer una experiencia audiovisual inmersiva a los usuarios de este programa. Utilizando un entorno 3D y herramientas de recursos bajos, buscamos hacer que este proyecto sea accesible para una mayor cantidad de usuarios.

Este proyecto implementa las mecánicas y jugabilidad del juego clásico Bricks Breaker, utilizando físicas para mantener la esencia de este antiguo juego. Hemos trabajado para que la dinámica del juego sea lo más fiel posible al original, tratando de que cada aspecto refleje la nostalgia y el desafío que caracterizan a Bricks Breaker.

En el desarrollo del diseño y la programación, hemos integrado gráficos 3D con la ayuda de Python, OpenGL y librerías como PyGame. Este enfoque no solo representa un avance significativo en nuestro desarrollo profesional, sino que también nos ha permitido crear un juego visualmente atractivo y técnicamente sólido.

Desarrollo

Durante la creación de este proyecto se utilizan distintos archivos .py junto con algunas carpetas de los cuales se explicarán brevemente su funcionalidad para el correcto funcionamiento del programa.

En la carpeta **asset** se encuentran los archivos png necesarios para la pantalla de menú junto con la tipografia que este utiliza.

En **models** guardamos los archivos .obj y . mtl los cuales son el modelo y sus materiales.

Dentro de la carpeta **Objects** tenemos 4 archivos los cuales son **Base3DObjects**, **Environment**, **Game3DObjects** y **GameBricks** cada uno siendo fundamental para el funcionamiento de este programa.

En **Base3DObjects** se establecen los buffers para cada elemento drentro de este archivo para luego proceder a crear los bloques o “Cubes” (como se les llama en programa) al igual que la esfera o “Sphere” donde por medio de slices y stacks se dibujará con ayuda de procesos matemáticos que realizara el programa.

Luego le sigue **Environment** el cual establecera la esfera principal esta es la esfera que contendrá las demás figuras asi funcionando como una figura a la cual le podremos aplicar los "Skysphere" que es el equivalente al SkyBox pero aplicado a una esfera.

En **Game3DObjects** tenemos los objetos del juego los cuales heredan los atributos de las clases definidas en **Base3DObjects** entre los objetos tenemos bricks que son los ladrillos que tendremos que romper estos cuentan con los atributos de posición, ancho, alto y color. También tenemos Ball que será la pelota la cual destruirá los ladrillos para poder ganar tiene los mismos atributos que Bricks con la diferencia que en lugar de color esta tiene texturas, la plataforma o "Platform" que también cuenta con atributos como posición, ancho, alto este no toma una textura o color si no que tomara un modelo 3D el cual se utilizara para evitar que la pelota caiga al vacio. Estos elementos también cuentan con esquinas o "corner" esto para poder facilitar la detección de colisiones, en este también tenemos las clases "LineObstacle"," Wall", "Frame" que representan las líneas para detectar las colisiones, la pared que evita que la pelota salga del espacio de juego y el marco donde la pelota revotara.

Por último, dentro de esta carpeta tenemos **GameBricks** donde se establece los distintos tipos de ladrillos diferenciados por color y resistencia (Cuantos golpes soportan antes de desaparecer) además de establecer una animación para cuando un bloque se destruya al igual que cambiar de cambiar de textura para representar los golpes que este allá recibido.

En la carpeta **sounds** encontramos los archivos tipo audio .mp3 los cuales se utilizan en múltiples partes del juego para volverlo inmersivo.

En la carpeta **Textures** se encuentran todas las texturas que se utilizan en el programa como lo son el skydom o las texturas de la pelota y la del modelo que usamos como plataforma.

BezierMotion es un archivo .py donde se cargará una pequeña animación para el inicio del juego la cual recorrerá el escenario para luego posicionarse en las coordenadas centrales para iniciar el juego.

Luego tenemos **menu.py** donde se ubicará la interfaz de usuario que funciona con **button** donde se maneja la creación de botone interactivos para dar vida a este menú con 3 opciones distintas **Play** el cual nos permite empezar el juego, **how to play** en este se almacenan las funciones de las teclas habilitadas para este juego, y por último **QUIT** donde se cerrara el programa.

En **Matrices** encontramos todos los procesos que ameriten el uso de matrices como el movimiento de la cámara (Desactivado ya que podría causar mareos) los movimientos del modelo al igual que sus transformaciones rotación, escalación y traslación.

ojb_3D_loading es responsable de cargar el modelo a partir de un archivo .obj y sus materiales necesarios.

Shaders.py se encarga de gestionar los sheders para el renderizado de objetos 3D y sprites 2D además de crear, compilar y enlazar los sheders necesarios para el renderizado.

texture_loading es el responsable de cargar las texturas de todo el proyecto de una manera eficiente utilizandon PyGame para esto primero se carga la imagen en pygame luego se trasforma en una cadena de bytes, se optiene el alto y ancho de la imagen y para luego identificar y enlazar la textura y poder ser utilizada en el código principal.

```
def load_texture(path_string):
        surface = pygame.image.load(sys.path[0] + "/Textures/" + path_string)
        tex_string = pygame.image.tostring(surface, "RGBA", 1)
        width = surface.get_width()
        height = surface.get_height()
        tex_id = glGenTextures(1)
        glBindTexture(GL_TEXTURE_2D, tex_id)
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR)
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR)
        glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0, GL_RGBA, GL_UNSIGNED_BYTE, tex_string)
        return tex_id
```

Control3DProgram este es el archivo principal donde se utilizan todos los archivos anteriores para crear el juego con sus mecánicas y los recursos necesarios para esto, además que en este archivo se establecen las teclas y funcionamientos de estas misma.

```
if event.key == K_LEFT:
    self.LEFT_key_down = True
if event.key == K_RIGHT:
    self.RIGHT_key_down = True
if event.key == K_SPACE:
    self.SPACE_key_down = True
if event.key == K_p:
    if self.pause_game:
```

Además de encontrar las condiciones para que el juego inicie, así como lo es que la animación inicial allá terminado primero antes de permitir lanzar la pelota o de llamar a los demás archivos para que se actualicen y no provoquen errores en la lógica del juego.

simple3D.frag y **simple3D.vert** realiza cálculos de iluminación combinado con la iluminación principal de un foco(spot) y agregar el uso de texturas para el material.

sprite_shader.frag y **sprite_shader.vert** hacen lo mismo que los simple3D pero aplicado a sprites 2D que están dentro de un entorno 3D.

Conclusión

Se logró recrear el conocido juego Bricks Breaker en un entorno 3D, conocido como Bricks3D, utilizando herramientas como Python, OpenGL y la librería Pygame. Además, se aplicaron los conocimientos adquiridos durante el semestre, como la rotación, traslación y escalado de los elementos en pantalla, y se desarrollaron las físicas necesarias para que el juego funcione correctamente.

Video Demostrativo

https://www.youtube.com/watch?v=PRjFw_t8N-g&ab_channel=saul

Bibliografia

ModelPaddle:https://free3d.com/es/modelo-3d/pingpong-paddle-v1--337871.html

ChatGTP - https://chatgpt.com/

3Denvironmentfirstversión:

https://www.youtube.com/watch?v=eJDIsFJN4OQ&ab_channel=CoderSpace

Game menú:

https://www.youtube.com/watch?v=GMBqjxcKogA&t=222s&ab_channel=baraltech

PyGame: https://www.pygame.org/docs/

Pyopengl: https://pyopengl.sourceforge.net/documentation/

Numpy: https://numpy.org/doc/

Moderngl: https://moderngl.readthedocs.io/en/5.8.2/

Creditos:

baraltech

https://github.com/baraltech

StanislavPetrovV

https://github.com/StanislavPetrovV