UNIVERSIDAD NACIONAL DE INGENIERÍA
DACTIC

INGENIERÍA EN COMPUTACIÓN

Programación Grafica

Integrantes:

Martínez González Ian Eduardo 2020-0730I

Acuña Jirón Génesis Alexandra 2023-0738U

Docente: Ing. Danny Oswaldo Chávez Miranda

Grupo: 3T3-Co

Fecha de entrega: viernes 20 de junio del 2025

Contenido

Objetivos

Objetivo general

Desarrollar un videojuego en un entorno tridimensional que replique la mecánica y jugabilidad del clásico **Bricks Breaker**, brindando una experiencia interactiva y atractiva para el usuario, mediante la implementación de gráficos 3D, iluminación, física realista y controles intuitivos.

Objetivos específicos

1. Implementar un motor gráfico utilizando Python y OpenGL, capaz de simular el comportamiento de la pelota y los bloques en un entorno tridimensional.
2. Incorporar mecánicas sencillas que faciliten la interacción del usuario con la aplicación, promoviendo una experiencia accesible y amigable.

Introducción

El presente proyecto tiene como finalidad el desarrollo de un videojuego dirigido a usuarios interesados en experiencias interactivas de tipo arcade. Surge como un desafío académico y técnico para el equipo de desarrollo, con el objetivo de ofrecer una experiencia audiovisual inmersiva dentro de un entorno tridimensional.

A través del uso de herramientas accesibles y de bajo requerimiento computacional, se pretende que el producto final sea funcional en una amplia gama de equipos, promoviendo así su alcance y accesibilidad.

El juego implementa las principales características del clásico **Bricks Breaker**, manteniendo su esencia mediante el uso de sistemas físicos que replican de forma precisa el comportamiento esperado en cada interacción. Se ha procurado conservar tanto el carácter nostálgico como el nivel de desafío que caracterizan al título original.

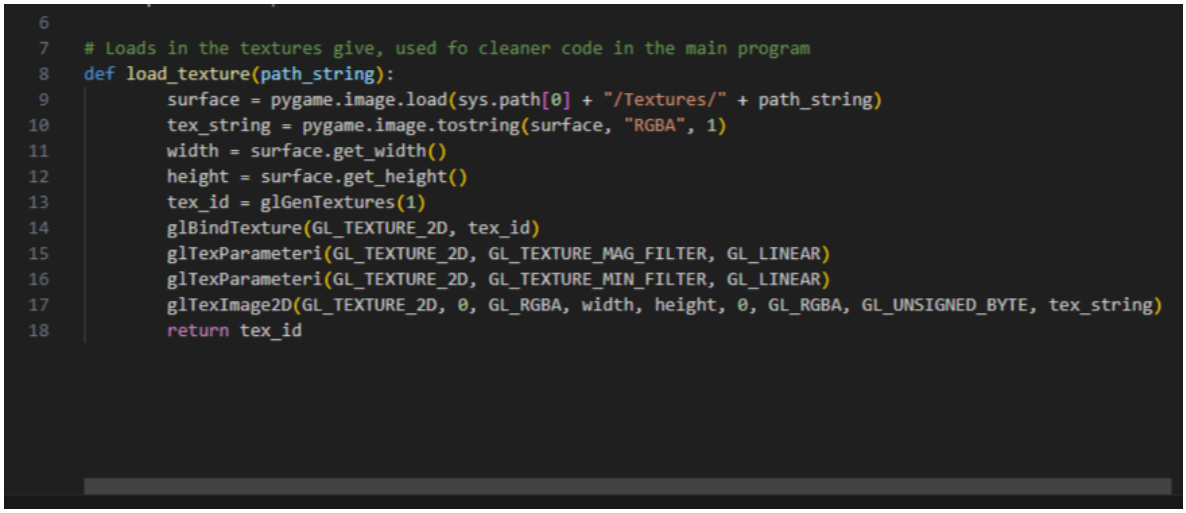
Durante el proceso de desarrollo, se integraron elementos gráficos tridimensionales mediante el uso de **Python**, **OpenGL** y bibliotecas complementarias como **PyGame**. Este enfoque ha representado una oportunidad para fortalecer nuestras habilidades técnicas, al mismo tiempo que se logra un producto visualmente atractivo y estructuralmente robusto.

Desarrollo

Para la construcción de este proyecto se ha hecho uso de diversos archivos con extensión `.py`, organizados en conjunto con carpetas específicas. A continuación, se detallan brevemente sus funcionalidades principales, con el fin de facilitar la comprensión de la arquitectura general del programa y asegurar su correcto funcionamiento.

- **Carpeta assets**: Contiene los archivos `.png` necesarios para la pantalla del menú, junto con la tipografía utilizada.
- **Carpeta models**: Se almacenan los archivos `.obj` y `.mtl`, que corresponden al modelo y sus materiales.
- **Carpeta Objects**: Contiene cuatro archivos fundamentales:
  - `Base3DObjects`: Establece los *buffers* para cada elemento. Aquí se crean los bloques o "Cubes" (como se les llama en el programa) y la esfera o "Sphere", dibujada mediante *slices* y *stacks* con ayuda de procesos matemáticos.
  - `Environment`: Establece la esfera principal que contendrá las demás figuras, funcionando como una figura a la cual se le puede aplicar el "Skysphere", equivalente al *SkyBox* pero aplicado a una esfera.
  - `Game3DObjects`: Contiene los objetos del juego que heredan atributos de las clases definidas en `Base3DObjects`. Incluye:
    - `Bricks`: Ladrillos que deben romperse, con atributos como posición, ancho, alto y color.
    - `Ball`: Pelota encargada de destruir los ladrillos. Tiene los mismos atributos que `Bricks`, pero usa texturas en lugar de color.
    - `Platform`: Plataforma para evitar que la pelota caiga al vacío. Usa un modelo 3D en lugar de textura o color.
    - También se incluyen elementos como `corner` para detección de colisiones y clases como `LineObstacle`, `Wall` y `Frame` que representan líneas de colisión, paredes del área de juego y el marco donde la pelota rebota.
  - `GameBricks`: Establece los distintos tipos de ladrillos, diferenciados por color y resistencia (cantidad de golpes que resisten antes de desaparecer). También incluye animaciones para la destrucción de bloques y cambio de texturas según el daño recibido.
- **Carpeta sounds**: Contiene archivos de audio `.mp3` usados para volver el juego más inmersivo.
- **Carpeta Textures**: Aquí se encuentran todas las texturas utilizadas, como el *skydome*, la textura de la pelota y la del modelo de la plataforma.
- **`BezierMotion.py`**: Carga una animación para el inicio del juego, recorriendo el escenario hasta posicionarse en el centro.

- **menu.py**: Contiene la interfaz de usuario, con botones interactivos que permiten:
  - `Play`: Iniciar el juego.
  - `How to play`: Mostrar controles y funciones de las teclas.
  - `Quit`: Cerrar el programa.
- **Carpeta Matrices**: Agrupa procesos relacionados con matrices, como movimiento y transformación de la cámara (aunque está desactivado por posibles mareos), así como transformaciones de rotación, escalado y traslación.
- **obj_3D_loading.py**: Carga modelos `.obj` y sus materiales.
- **Shaders.py**: Gestiona los *shaders* para renderizar objetos 3D y *sprites* 2D. Se encarga de crear, compilar y enlazar *shaders*.
- **texture_loading.py**: Carga texturas eficientemente con PyGame. Convierte imágenes en cadenas de bytes, obtiene alto y ancho, y enlaza la textura para su uso en el programa.



```
6
7   # Loads in the textures give, used fo cleaner code in the main program
8   def load_texture(path_string):
9           surface = pygame.image.load(sys.path[0] + "/Textures/" + path_string)
10          tex_string = pygame.image.tostring(surface, "RGBA", 1)
11          width = surface.get_width()
12          height = surface.get_height()
13          tex_id = glGenTextures(1)
14          glBindTexture(GL_TEXTURE_2D, tex_id)
15          glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR)
16          glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR)
17          glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0, GL_RGBA, GL_UNSIGNED_BYTE, tex_string)
18          return tex_id
```

- **Control3DProgram.py**: Archivo principal del juego. Aquí se integran todos los componentes y se definen las teclas y sus funciones. También se establecen condiciones para iniciar el juego, como esperar a que la animación inicial termine antes de lanzar la pelota o actualizar otros módulos.

```
if event.key == K_a:
    self.A_key_down = True
if event.key == K_d:
    self.D_key_down = True
if event.key == K_SPACE:
    self.SPACE_key_down = True
if event.key == K_p:
    if self.pause_game:
        self.pause_game = self.pause_game = False
        self.sound.play(-1)
        self.music_paused = False
```

- **simple3D.frag y simple3D.vert**: Realizan cálculos de iluminación combinando un foco principal y aplican texturas a los materiales.
- **sprite_shader.frag y sprite_shader.vert**: Cumplen la misma función que los anteriores, pero aplicados a *sprites* 2D dentro del entorno 3D.

Conclusión

Se logró desarrollar una versión tridimensional del reconocido juego **Bricks Breaker**, titulada **Bricks3D**, empleando herramientas como **Python**, **OpenGL** y la biblioteca **PyGame**. El proyecto permitió poner en práctica diversos conocimientos adquiridos a lo largo del semestre, tales como transformaciones espaciales (rotación, traslación y escalado), así como la implementación de sistemas físicos que garantizan el comportamiento adecuado de los elementos del juego. Esta experiencia contribuyó significativamente al fortalecimiento de nuestras habilidades en programación gráfica y desarrollo de entornos interactivos en 3D.

Video demostrativo**:**

https://www.youtube.com/watch?v=pBVwkcrplzg

Bibliografía

baraltech. (s.f.). *Game menú* [Video]. YouTube. https://www.youtube.com/watch?v=GMBqjxcKogA

ChatGPT. (2024). *ChatGPT* [Aplicación web]. OpenAI. https://chatgpt.com/

DeepWiki. (s.f.). *Blocks3D por Nymoz15*. https://deepwiki.com/Nymoz15/Blocks3D

NumPy. (s.f.). *NumPy Documentation*. https://numpy.org/doc/

PyGame. (s.f.). *PyGame Documentation*. https://www.pygame.org/docs/

PyOpenGL. (s.f.). *PyOpenGL Documentation*. https://pyopengl.sourceforge.net/documentation/

ModernGL. (s.f.). *ModernGL 5.8.2 Documentation*. https://moderngl.readthedocs.io/en/5.8.2/

Créditos

baraltech. (s.f.). *Repositorio GitHub* [Perfil GitHub]. https://github.com/baraltech