Universidad Nacional de Ingeniería

Recinto Universitario Simón Bolívar

Proyecto Final : Solar System Simulator

Elaborado por:

- ✔ Mendoza Herrera Alejandro Antonio
- ✔ Gonzalez Salinas Ethan de Jesús
- ✔ Rayo Mejía Anthony Alexander
- ✔ Alemán García Jorge Alexander

Número de Carnet:

- ✔ 2022-0265U
- ✔ 2022-0260U
- ✔ 2022-0388U
- ✔ 2022-0334U

Grupo:

3T1-CO

Docente: Danny Chavez

Fecha: Viernes 5 de julio de 2024

I. Introducción

Este proyecto de simulación interactiva del sistema solar está diseñado para todas aquellas personas que están interesadas en la exploración del sistema solar, proporcionando una plataforma educativa y visualmente atractiva que permite a los usuarios navegar y aprender sobre los diversos cuerpos celestes que componen nuestro sistema solar. Utilizando tecnologías avanzadas de programación gráfica como OpenGL y Pygame, esta simulación ofrece una experiencia interactiva y educativa junto de una interfaz amigable y accesible.

En este proyecto, los usuarios podrán explorar libremente el sistema solar, observando de cerca los planetas haciendo uso del teclado y mouse. Además, de tener una sección extra en la cual se desplegará una ventana informativa con modelos 3D de cada planeta y datos relevantes, como su tamaño, composición, atmósfera, etc.

Además de esto, la creación de esta simulación del sistema solar también sirve para afianzar conocimientos en programación gráfica. A lo largo del desarrollo del proyecto, se aplicaron conceptos avanzados de OpenGL y Pygame, desde la renderización de modelos 3D y la implementación de texturas realistas, hasta la gestión de interacciones de usuario mediante teclado y mouse.

El resultado final se puede apreciar en el siguiente video: https://drive.google.com/drive/u/1/folders/16IBn-4lkfgrN5-uYVBhCWZQMV3WAcZ17 ,junto al repositorio de github Rayo070305/Proyecto_final_pg (github.com) .

II. Objetivos

Objetivo General:

Desarrollar una simulación interactiva del sistema solar que permita a los usuarios explorar y aprender sobre los planetas y sus características a través de una experiencia visual, utilizando tecnologías como python y OpenGL.

Objetivos específicos:

1. Proporcionar un entorno de navegación interactivo que permita a los usuarios moverse libremente por el sistema solar, haciendo uso del teclado y mouse.
2. Implementar una sección informativa para cada planeta, mostrando modelos 3D detallados sus características básicas, con el fin de ofrecer una experiencia educativa completa y accesible.
3. Utilizar tecnologías como Pygame y OpenGL con el fin de mostrar modelos 3D de los planetas y satélites artificiales.

III. Desarrollo

Para este proyecto se han utilizado distintas librerías de las cuales están incluidas: pygame(y sus derivados), sys, subprocess, numpy, GLM(y sus derivaciones), Math, os, python y OpenGL(y sus derivados).

1. Solar system

En el siguiente paso se creó una función que creará una esfera que en este caso haría un sol y que su alrededor se agregara otras esferas que a su vez rotan alrededor de la esfera principal, primero se comenzó con 1 esfera principal(el sol) y los primeros 3 planetas . A cada una de estas esferas se le anexó una textura que fue sacada de una página web llamada “solar system scope”.

```
def draw_sphere(radius, slices, stacks, texture_id):
    quad = gluNewQuadric()
    gluQuadricNormals(quad, GLU_SMOOTH)
    gluQuadricTexture(quad, GL_TRUE)

    glBindTexture(GL_TEXTURE_2D, texture_id)
    gluSphere(quad, radius, slices, stacks)
```

```
sun_texture_id = load_texture("opengl project/image/suns.jpg")
planet1_texture_id = load_texture("opengl project/image/mercu.jpg")
planet2_texture_id = load_texture("opengl project/image/venus.jpg")
planet3_texture_id = load_texture("opengl project/image/earth.jpg")
planet4_texture_id = load_texture("opengl project/image/mars.jpg")
planet5_texture_id = load_texture("opengl project/image/jupiter.jpg")
planet6_texture_id = load_texture("opengl project/image/saturn.jpg")
planet7_texture_id = load_texture("opengl project/image/uranus.jpg")
planet8_texture_id = load_texture("opengl project/image/neptune.jpg")
```

Al finalizar esta sección, se anexó una función que crearía y dibujaría el espacio del skybox, dándole una imagen de estrella que se sacó de la anterior página web.

Para crear todo este espacio se creó una función cámara que permita viajar a través el espacio del skybox dándole el énfasis necesario para que sea un trabajo de manera 3D que se merece. Para que esto se viera de manera de una manera realista se utilizó las teclas a(izquierda),s(atrás),w(delante),d(derecha), space(arriba), ctrl(abajo); y con la ayuda del mouse poder mover el ángulo de la cámara.

```
def draw_skybox(size, texture):
    glDisable(GL_LIGHTING)
    glEnable(GL_TEXTURE_2D)

    glPushMatrix()
    glTranslatef(0.0, 0.0, 0.0)
    glBindTexture(GL_TEXTURE_2D, texture)

    # Front
    glBegin(GL_QUADS)
    glTexCoord2f(0, 0); glVertex3f(-size, -size, -size)
    glTexCoord2f(1, 0); glVertex3f(size, -size, -size)
    glTexCoord2f(1, 1); glVertex3f(size, size, -size)
    glTexCoord2f(0, 1); glVertex3f(-size, size, -size)
    glEnd()

    # Left
    glBegin(GL_QUADS)
    glTexCoord2f(0, 0); glVertex3f(-size, -size, size)
    glTexCoord2f(1, 0); glVertex3f(-size, -size, -size)
    glTexCoord2f(1, 1); glVertex3f(-size, size, -size)
    glTexCoord2f(0, 1); glVertex3f(-size, size, size)
    glEnd()

    # Right
    glBegin(GL_QUADS)
    glTexCoord2f(0, 0); glVertex3f(size, -size, -size)
    glTexCoord2f(1, 0); glVertex3f(size, -size, size)
    glTexCoord2f(1, 1); glVertex3f(size, size, size)
    glTexCoord2f(0, 1); glVertex3f(size, size, -size)
    glEnd()
```

```
    # Top
    glBegin(GL_QUADS)
    glTexCoord2f(0, 0); glVertex3f(-size, size, -size)
    glTexCoord2f(1, 0); glVertex3f(size, size, -size)
    glTexCoord2f(1, 1); glVertex3f(size, size, size)
    glTexCoord2f(0, 1); glVertex3f(-size, size, size)
    glEnd()

    # Back
    glBegin(GL_QUADS)
    glTexCoord2f(0, 0); glVertex3f(-size, -size, size)
    glTexCoord2f(1, 0); glVertex3f(size, -size, size)
    glTexCoord2f(1, 1); glVertex3f(size, size, size)
    glTexCoord2f(0, 1); glVertex3f(-size, size, size)
    glEnd()

    # Bottom
    glBegin(GL_QUADS)
    glTexCoord2f(0, 0); glVertex3f(-size, -size, size)
    glTexCoord2f(1, 0); glVertex3f(size, -size, size)
    glTexCoord2f(1, 1); glVertex3f(size, -size, -size)
    glTexCoord2f(0, 1); glVertex3f(-size, -size, -size)
    glEnd()

    glPopMatrix()
    glDisable(GL_LIGHTING)
    glDepthMask(GL_TRUE)
```

```
skybox_texture = load_texture("opengl project/skybox/stars.jpg")   #textura del skybox
```

Por consiguiente se dedicó a ingresar un audio mp3 con un sonido relajante que haga tener en cuenta lo que está en el espacio.

```
def music():
    pygame.mixer.init()
    pygame.mixer.music.load('opengl project/sounds/relax.mp3')
    pygame.mixer.music.play()
    pygame.time.delay(2000)  # Espera 2 segundos antes de continuar con el bucle principal
```

De esta forma se hizo para crear la versión beta del proyecto la cual fue presentada el día que fue expuesto la idea del sistema solar.

Para el siguiente paso comenzamos realizando todos los planetas restantes en el main. En el caso de Saturno se sabe que su característica principal es tener sus anillos, para este paso se creó una función que le realice por medio de una imagen de textura los anillos que igualmente fue sacada de dicha página.

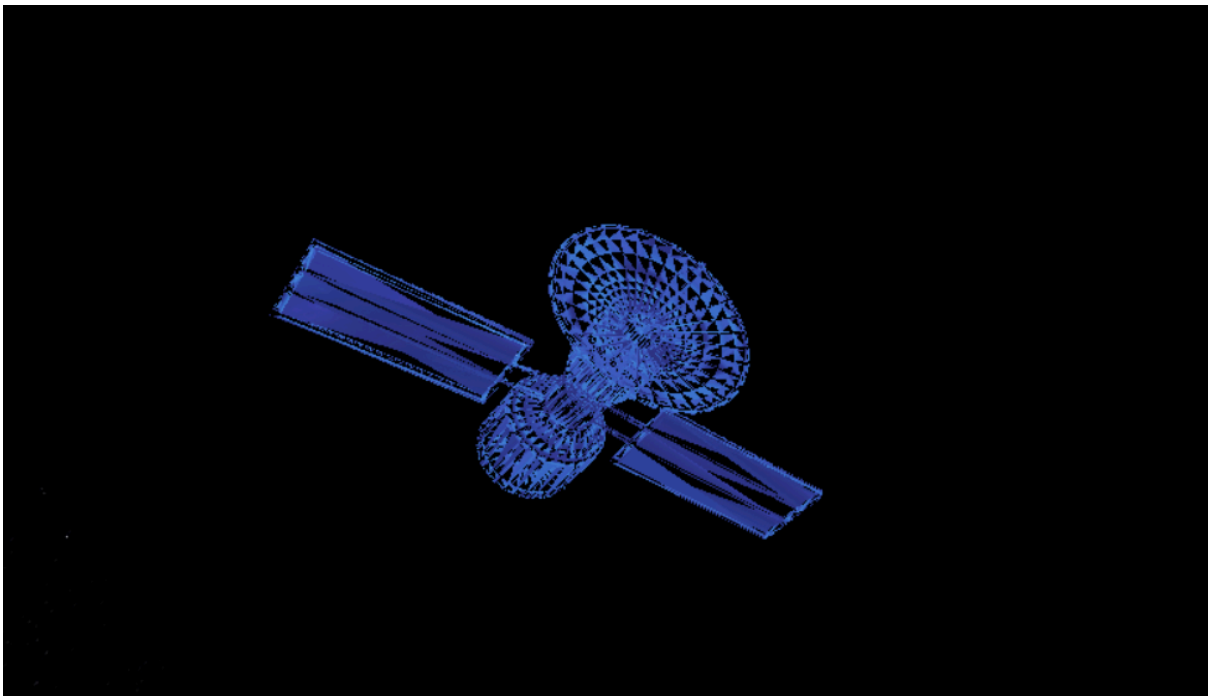
```
def draw_rings(inner_radius, outer_radius, segments=100):
    glBegin(GL_QUAD_STRIP)
    for i in range(segments + 1):
        theta = 2.0 * math.pi * i / segments
        x = math.cos(theta)
        z = math.sin(theta)
        glTexCoord2f(i / segments, 1)
        glVertex3f(outer_radius * x, 0.0, outer_radius * z)
        glTexCoord2f(i / segments, 0)
        glVertex3f(inner_radius * x, 0.0, inner_radius * z)
    glEnd()
```

```
ring_texture_id = load_texture("opengl project/image/rings.jpg")  # los anillos de Saturno
```

Luego se vió que en el programa no se lograba apreciar bien el orden en el que giraban los planetas por lo tanto se realizó una función la cual crearía unas orbitas visibles para que se viera con más detalle el curso de su rotación

```
def draw_rings(inner_radius, outer_radius, segments=100):
    glBegin(GL_QUAD_STRIP)
    for i in range(segments + 1):
        theta = 2.0 * math.pi * i / segments
        x = math.cos(theta)
        z = math.sin(theta)
        glTexCoord2f(i / segments, 1)
        glVertex3f(outer_radius * x, 0.0, outer_radius * z)
        glTexCoord2f(i / segments, 0)
        glVertex3f(inner_radius * x, 0.0, inner_radius * z)
    glEnd()
```

Para que el programa fuese más creativo se le anexó un modelo obj del cual fue sacado de la página web ”free 3D ” que representa un satélite hecho que sigue a la tierra.



hubo un problema el cual era que al momento de alejarnos del skybox este mismo ocultaba el interior y era casi imposible de ver, así que se estableció por medio de una función “is_whitin_bound” que permitiera que hubiese un límite del rango total -1

```
def is_within_bounds(self, position):
    limit = self.skybox_size - 1  # Limitar a un poco menos que el tamaño del skybox
    return -limit <= position.x <= limit and -limit <= position.y <= limit and -limit <= position.z <= limit
```

Ya que es un programa educativo se necesitaba crear algo que cada planeta se pudiera especificar de cual era cual y se sugirió hacer un cubo con los nombres de cada uno de los planetas

```
def draw_cube(size, texture_id):
    glBindTexture(GL_TEXTURE_2D, texture_id)
    glBegin(GL_QUADS)
    # Front face
    glTexCoord2f(0, 0); glVertex3f(-size, -size, size)
    glTexCoord2f(1, 0); glVertex3f(size, -size, size)
    glTexCoord2f(1, 1); glVertex3f(size, size, size)
    glTexCoord2f(0, 1); glVertex3f(-size, size, size)
    # Back face
    glTexCoord2f(0, 0); glVertex3f(-size, -size, -size)
    glTexCoord2f(1, 0); glVertex3f(size, -size, -size)
    glTexCoord2f(1, 1); glVertex3f(size, size, -size)
    glTexCoord2f(0, 1); glVertex3f(-size, size, -size)
    # Left face
    glTexCoord2f(0, 0); glVertex3f(-size, -size, -size)
    glTexCoord2f(1, 0); glVertex3f(-size, -size, size)
    glTexCoord2f(1, 1); glVertex3f(-size, size, size)
    glTexCoord2f(0, 1); glVertex3f(-size, size, -size)
    # Right face
    glTexCoord2f(0, 0); glVertex3f(size, -size, -size)
    glTexCoord2f(1, 0); glVertex3f(size, -size, size)
    glTexCoord2f(1, 1); glVertex3f(size, size, size)
    glTexCoord2f(0, 1); glVertex3f(size, size, -size)
    # Top face
    glTexCoord2f(0, 0); glVertex3f(-size, size, -size)
    glTexCoord2f(1, 0); glVertex3f(size, size, -size)
    glTexCoord2f(1, 1); glVertex3f(size, size, size)
    glTexCoord2f(0, 1); glVertex3f(-size, size, size)
    # Bottom face
    glTexCoord2f(0, 0); glVertex3f(-size, -size, -size)
    glTexCoord2f(1, 0); glVertex3f(size, -size, -size)
    glTexCoord2f(1, 1); glVertex3f(size, -size, size)
    glTexCoord2f(0, 1); glVertex3f(-size, -size, size)
    glEnd()
```

en este apartado se creó los cubos y sus caras correspondientes

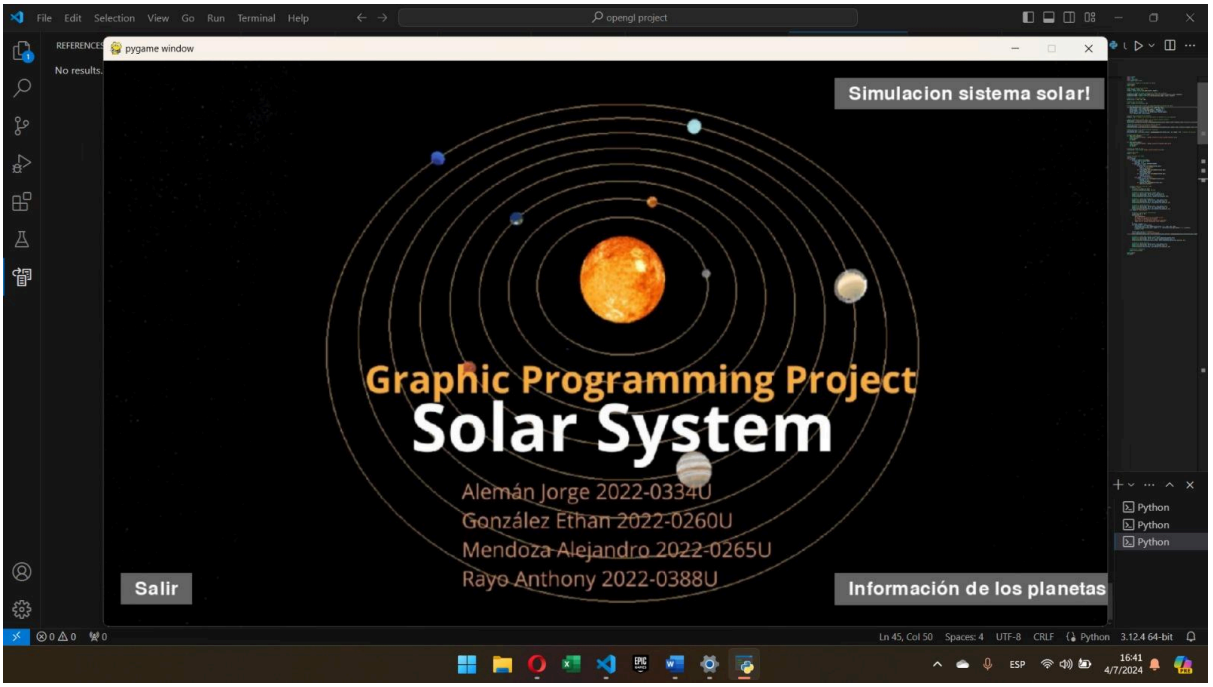
para finalizar le añadimos una luz que es la que permite que el sistema solar sea visible en el sol, con un código de función tal que:

```python
def setup_lighting():
    glEnable(GL_LIGHTING)
    glEnable(GL_LIGHT0)

    # Configurar la luz
    light_position = [0.0, 0.0, 0.0, 1.0]  # Posición del sol
    light_color = [1.0, 1.0, 1.0, 1.0]  # Color de la luz (blanco)
    ambient_light = [0.1, 0.1, 0.1, 1.0]  # Luz ambiental

    glLightfv(GL_LIGHT0, GL_POSITION, light_position)
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_color)
    glLightfv(GL_LIGHT0, GL_SPECULAR, light_color)
    glLightfv(GL_LIGHT0, GL_AMBIENT, ambient_light)
```

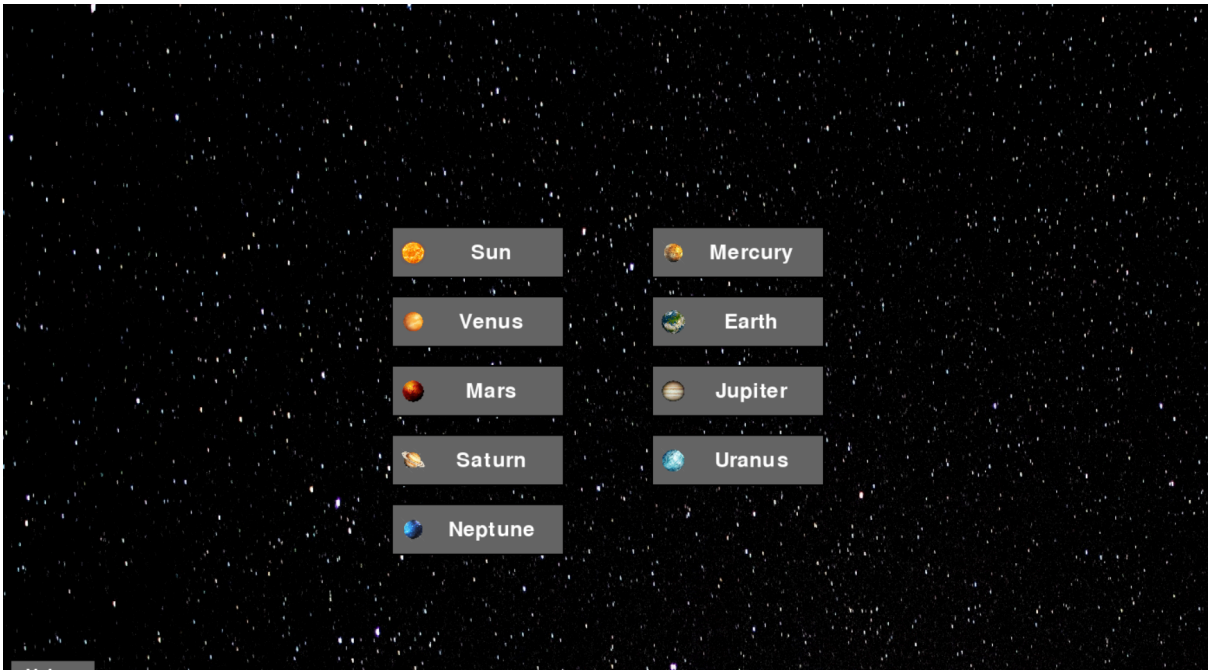
y así es como quedaría nuestro programa principal de simulación del sistema solar

1.2 Presentación junto con menu y información de los planetas

En esta parte se agregó el script que es de la presentación la cual contiene 3 botones, salir(cierra el programa) , iniciar simulación (invoca solar_system_simulator.py) e información(contiene un menú con todos los planetas, incluyendo el sol),



En el apartado de menú de información tenemos una ventana con botones para cada planeta que al darle click , invoca la respectiva información del planeta seleccionado



Al seleccionar se abre la ventana que contiene la esfera en 3D con la textura del planeta y su información en la parte izquierda , y en la esquina superior izquierda un botón para regresar a la ventana anterior.

IV. Conclusión

En conclusión, el proyecto de simulación interactiva del sistema solar no solo proporciona una plataforma educativa y accesible para los usuarios, sino que también sirve como un medio para explorar y comprender mejor los componentes y movimientos de los planetas dentro de nuestro sistema solar. A través de la combinación de navegación interactiva y contenido informativo, los usuarios pueden aprender sobre los planetas de una manera dinámica.

Además, el desarrollo de este proyecto ha permitido el fortalecimiento de los conocimientos aprendidos sobre programación gráfica, aplicando tecnologías como pygame, opengl y glm. Puesto que la creación del entorno requirió una mayor comprensión de los principios de renderización, iluminación y manejo de interacciones con el usuario.

V. Bibliografía

- Free 3D. (n.d.). Satélite [Modelo 3D]. https://free3d.com/es/modelos-3d/sat%C3%A9lite
- Solar System Scope. (n.d.). Texturas [Página web]. https://www.solarsystemscope.com/textures/
- Solar System Exploration - NASA Science. (n.d.). [Imágenes PNG]. (Descripciones de resolución y dimensiones variadas).