Ciencia de la Computación

Computación Gráfica

Docente Manuel Eduardo Loaiza Fernández

Ovni Abduce una Vaca en OpenGL

Entregado el 01/07/2024

Cristian Ramos Medina
Camila Andrea Luque Juárez

Semestre VII

2024-1

"Los alumnos declaran haber realizado el presente trabajo de acuerdo a las normas de la Universidad Católica San Pablo"

_____ _____

Computación Gráfica
Abducción de vaca

Camila Luque Juárez
Universidad Católica San Pablo
camila.luque.juarez@ucsp.edu.pe

Cristian Ramos Medina
Universidad Católica San Pablo
cristian.ramos@ucsp.edu.pe

1 Introducción

Este proyecto presenta la simulación de una escena en la que un objeto volador no identificado (OVNI) abduce una vaca. Utilizando OpenGL y la librería GLFW, se han aplicado conceptos avanzados de gráficos por computadora, incluyendo animación, manejo de shaders, y técnicas de iluminación. La motivación detrás de este proyecto es explorar y demostrar las capacidades de las herramientas gráficas modernas para crear escenas interactivas y visualmente atractivas, proporcionando una comprensión más profunda de los gráficos por computadora en aplicaciones realistas y ficticias.

2 Descripción del proyecto

Este proyecto simula la abducción de una vaca por un OVNI, empleando técnicas avanzadas de computación gráfica para crear una experiencia visual inmersiva. La implementación abarca desde la carga y renderización de modelos 3D hasta la aplicación de shaders para efectos de iluminación y animación. Utilizamos diversas bibliotecas y herramientas especializadas para manejar la gestión de ventanas, la carga de texturas y modelos, y los cálculos matemáticos necesarios para las transformaciones en el espacio 3D. Esta combinación permite simular el movimiento y la interacción de los objetos en la escena, además de aplicar técnicas de iluminación que añaden realismo. El resultado es una escena en la que un OVNI se desplaza y abduce una vaca, con movimientos fluidos y efectos visuales detallados y realistas. A continuación, se detallan los componentes y técnicas clave utilizados, explicando su contribución a la creación de esta simulación.

2.1 Herramientas

Para llevar a cabo el proyecto, se utilizaron las siguientes herramientas:

- **GLFW**: Esta biblioteca permite la creación de ventanas y la gestión de eventos de entrada, como el teclado y el ratón. GLFW es fundamental para inicializar el contexto de OpenGL y manejar las interacciones del usuario.
- **GLAD**: Un cargador de funciones de OpenGL que facilita la utilización de las extensiones de OpenGL necesarias para el proyecto.
- **GLM**: Esta biblioteca proporciona una serie de funciones y tipos para operaciones matemáticas, incluyendo matrices y vectores. Es esencial para realizar las transformaciones geométricas en el espacio 3D.

- **STB Image**: Biblioteca utilizada para cargar imágenes desde disco. En este proyecto, se utiliza para cargar texturas que se aplican a los modelos 3D.
- **TinyOBJLoader**: Un cargador de modelos en formato OBJ. Permite cargar modelos 3D complejos desde archivos externos, facilitando la inclusión de diversos elementos en la escena.

2.2 Animación

Este proyecto cuenta con cinco elementos fundamentales para lograr una abducción más realista de una vaca. La correcta implementación y sincronización de estos componentes permite una simulación realista y envolvente de la abducción A continuación se explica detalladamente cómo se maneja el movimiento del OVNI, la vaca, el cono de luz, el láser rojo y el movimiento de cámara.

2.2.1 Movimiento del OVNI

El movimiento del OVNI se controla mediante varias variables y transformaciones. Las variables clave incluyen `ufoRotationAngle`, `ufoPositionY`, `ufoPositionX` y `ufoScale`.

- **Rotación del OVNI**: La variable `ufoRotationAngle` se incrementa en cada iteración del bucle de renderizado, lo que hace que el OVNI gire continuamente. Esta rotación se aplica usando `glm::rotate` en la matriz de modelo del OVNI.
- **Descenso del OVNI**: Inicialmente, `ufoDescending` es verdadero, lo que causa que `ufoPositionY` se reduzca en cada frame hasta alcanzar una altura mínima (`30.0f`), momento en el cual `ufoDescending` se establece en falso y `ufoMovingRight` en verdadero.
- **Movimiento Horizontal del OVNI**: Cuando `ufoMovingRight` es verdadero, `ufoPositionX` se incrementa hasta que el OVNI alcanza la posición central (`0.0f`), activando `cowAscending`.
- **Retirada del OVNI**: Una vez que la vaca ha sido completamente abducida (`cowAbducted` es verdadero), `ufoRetreating` se establece en verdadero y `ufoPositionY` comienza a incrementarse para simular la retirada del OVNI. Si `ufoPositionY` alcanza 70.0f, el OVNI empieza a salir de la escena horizontalmente, reduciendo su tamaño (`ufoScale`).

Estas transformaciones se aplican en cada frame para actualizar la matriz de modelo del OVNI `ufoModelMatrix`.

2.2.2 Movimiento de la Vaca

El movimiento de la vaca también se controla mediante varias variables y transformaciones clave, incluyendo `cowPositionOffsetY`, `cowScale`, y `cowRotationAngle`.

- **Ascenso de la Vaca**: Cuando `cowAscending` es verdadero, `cowPositionOffsetY` se incrementa, elevando la posición de la vaca. Simultáneamente, `cowScale` se reduce para dar la ilusión de que la vaca se aleja.
- **Rotación de la Vaca**: La variable `cowRotating` se activa durante el ascenso, incrementando `cowRotationAngle` para hacer que la vaca gire mientras asciende.

- **Completación de la Abducción**: Si `cowScale` llega a `0.0f`, `cowAscending` se establece en falso y `cowAbducted` en verdadero, indicando que la vaca ha sido completamente abducida.

La matriz de modelo de la vaca (`cowModelMatrix`) se actualiza en cada frame utilizando estas transformaciones, contribuyendo a una animación convincente de la abducción.

2.2.3 Cono de Luz

El cono de luz se renderiza utilizando el shader `coneFragmentShaderSource` con cierta opacidad en el color y se activa bajo ciertas condiciones.

- **Activación del Cono**: El cono de luz se dibuja cuando el OVNI ha terminado de descender (`ufoDescending` es falso) y antes de que la vaca sea completamente abducida (`cowAbducted` es falso).
- **Transformación del Cono**: La posición del cono se ajusta en función de la posición del OVNI (`ufoPositionX` y `ufoPositionY - coneHeight / 2.0f`), y se rota de acuerdo con `ufoRotationAngle`.

Estas transformaciones se aplican a la matriz de modelo del cono (`coneModelMatrix`), añadiendo un efecto visual crucial para la escena.

2.2.4 Láser Rojo

El láser rojo se simula mediante el uso de shaders específicos y se anima usando una lógica de movimiento circular.

- **Posición del Láser**: El láser se origina desde la posición del OVNI y termina en un punto en movimiento circular en el suelo, calculado usando `cos(time_laser)` y `sin(time_laser)`.
- **Shader del Láser**: El shader `laserFragmentShaderSource` define el color rojo del láser con cierta opacidad (`0.35`).

El láser se dibuja ajustando el ancho de la línea (`glLineWidth(5.0f)`) y actualizando los vértices del láser en cada frame, contribuyendo a la sensación de acción en la escena.

2.2.5 Movimiento de la Cámara

El movimiento de la cámara es esencial para capturar la acción desde un ángulo dinámico y atractivo. La cámara se controla utilizando la clase `Camera`, con variables y métodos que actualizan su posición y orientación.

- **Posición Inicial**: La cámara se inicializa en la posición `(120.0f, 20.0f, 120.0f)` con un centro de atención en `(-50.0f, 10.0f, 0.0f)`.
- **Movimiento durante la Ascensión de la Vaca**: Mientras la vaca asciende (`cowAscending` es verdadero) y no ha sido completamente abducida (`cowAbducted` es falso), la cámara se mueve hacia arriba (`move(glm::vec3(0.0f, 0.08f, 0.0f))`) y gira (`turn(glm::vec3(0.0f, 0.08f, 0.0f))`) para seguir la acción.

- **Movimiento durante la Retirada del OVNI**: Una vez que la vaca ha sido abducida (`cowAbducted` es verdadero), la cámara sigue al OVNI mientras este se retira (`ufoRetreating` es verdadero). La cámara se mueve horizontalmente (`move(glm::vec3(0.1f, 0.0f, 0.0f))`) para seguir el OVNI.
- **Detención de la Cámara**: La cámara se detiene (`cameraStopped` es verdadero) cuando el OVNI sale de la escena y su escala (`ufoScale`) llega a `0.0f`.

El método `updateCameraPosition` se llama en cada frame para ajustar la posición de la cámara basándose en las posiciones actuales del OVNI y la vaca, y las condiciones de la escena.

2.3 Secuencia de Animación

La animación presentada muestra de manera detallada y dinámica el proceso de abducción de una vaca por un objeto volador no identificado (OVNI). Esta secuencia no solo resalta los aspectos técnicos y visuales de la animación, sino que también busca sumergir al espectador en una experiencia narrativa envolvente con la inclusión de luces y diferentes elementos. A continuación, se describe cada fase de la animación, acompañada de imágenes que ilustran los momentos más significativos de esta intrigante historia.

Figure 1: OVNI utilizando rayo rastreador.

2.3.1 Descenso del OVNI:

El OVNI desciende desde el cielo nocturno, activando su rayo rastreador para buscar posibles objetivos en la superficie terrestre. Este paso es crucial ya que permite al OVNI localizar con precisión a su objetivo, utilizando una tecnología avanzada de rastreo que escanea el terreno en busca de vida. La luz del rayo rastreador ilumina el área debajo del OVNI, creando un efecto visual impactante que sugiere una tecnología alienígena avanzada (ver Figura 1).

2.3.2 Activación de las luces de abducción:

Figure 2: OVNI empieza a elevar a la vaca.

Al detectar una entidad cercana, el OVNI enciende sus luces de abducción y se desplaza hacia la vaca identificada como objetivo. Este paso no solo realza el dramatismo de la escena, sino que también muestra el poder y la precisión del OVNI. Las luces de abducción emiten un brillo verde que resalta en la oscuridad, proporcionando un contraste visual con el entorno nocturno.

2.3.3 Posicionamiento sobre la víctima:

El OVNI se posiciona precisamente sobre la vaca, asegurando una abducción eficiente. En esta fase, la nave maniobra con una precisión asombrosa, demostrando la avanzada tecnología de control y estabilización del OVNI. La vaca, sorprendida y paralizada por las luces, queda directamente debajo del rayo abductor, lista para ser levantada.

2.3.4 Inicio de la abducción:

El rayo abductor se activa, comenzando el proceso de elevar a la vaca hacia la nave. Este rayo es un haz de energía verde que envuelve a la vaca y la levanta suavemente en el aire. La tecnología del rayo abductor no solo levanta a la vaca, sino que también la mantiene inmovilizada, asegurando que no se lastime durante el proceso (ver Figura 2).

Figure 3: OVNI deforma a su víctima.

2.3.5 Reducción del tamaño de la vaca:

Durante la abducción, el OVNI reduce gradualmente el tamaño de la vaca para facilitar su transporte. Este proceso de miniaturización es una muestra de la avanzada tecnología alienígena, que permite compactar objetos sin dañarlos. La vaca, ahora más pequeña, es más fácil de manejar y almacenar dentro de la nave (ver Figura 3).

2.3.6 Apagado del rayo y preparación para el escape:

Figure 4: OVNI alistándose para escapar.

Una vez que la vaca está a bordo, el OVNI apaga su rayo abductor y se prepara para huir de la escena. El apagado del rayo es gradual, asegurando que la vaca esté completamente dentro de la nave antes de que el rayo se disipe. Este paso es crucial para evitar cualquier rastro de actividad alienígena (ver Figura 4).

2.3.7 Elevación de escape:

El OVNI inicia su elevación, alejándose rápidamente del área de abducción. La nave asciende con rapidez, dejando apenas un rastro de luz detrás de ella. Este movimiento rápido es vital para evitar la detección por parte de cualquier testigo ocular o dispositivo de vigilancia terrestre (ver Figura 5).

2.3.8 Huida hacia el horizonte:

El OVNI se dirige hacia el horizonte, desplazándose a una velocidad que le permite evadir cualquier seguimiento terrestre. La nave, ahora casi invisible debido a su velocidad, se mezcla con el cielo oscuro, moviéndose de manera fluida y silenciosa.

Figure 5: OVNI se empieza a alejar.

2.3.9 Salida de la escena:

Apuntando hacia las montañas, el OVNI se aleja, disminuyendo su tamaño visual debido a la distancia, hasta que finalmente se pierde de vista. Este paso finaliza la abducción, con el OVNI desapareciendo en el horizonte montañoso, dejando el área en un silencio inquietante.

2.3.10 Desaparición entre las estrellas:

Figure 6: OVNI se pierde en las estrellas.

El OVNI se fusiona con el cielo estrellado, desapareciendo completamente de la vista como si nunca hubiera estado allí. La nave se disuelve en la vastedad del espacio, ocultándose entre las innumerables estrellas que brillan en la lejanía. Esta maniobra final demuestra la capacidad avanzada del OVNI para evitar ser detectado, incluso cuando parece haber sido avistado. La nave no solo desaparece visualmente, sino que también desvanece cualquier rastro de su presencia, como posibles señales electromagnéticas o térmicas que podrían ser detectadas por equipos de monitoreo. La misión del OVNI, completada con éxito, deja a la Tierra intacta, sin evidencia alguna de la abducción que ha tenido lugar. Este último acto de desaparición añade un aire de misterio y sofisticación tecnológica, resaltando la superioridad de la tecnología extraterrestre que permitió llevar a cabo la operación de manera impecable y sin dejar rastro alguno de su presencia (ver Figura 6).

2.4 Shaders Utilizados

En el proyecto, se emplean varios shaders programados en GLSL (OpenGL Shading Language) para manipular las propiedades visuales de los objetos en escena. Cada shader tiene un rol específico que contribuye a la interactividad y al realismo visual del entorno 3D.

2.4.1 Vertex Shader

El **Vertex Shader** procesa los vértices de los modelos 3D. Su función principal es transformar las coordenadas de los vértices desde el espacio del objeto al espacio de la pantalla, mientras pasa las coordenadas de textura y otras variables necesarias al **Fragment Shader**. Este shader también se encarga de calcular las posiciones transformadas y las normales para la iluminación.

Declaración del Vertex Shader:

- `const char* vertexShaderSource;`

2.4.2 Fragment Shader para Modelos

El **Fragment Shader** se utiliza para determinar los colores y otros atributos de los píxeles. En este proyecto, el shader está diseñado para aplicar texturas a los modelos 3D, contribuyendo a un acabado más detallado y visualmente atractivo de los objetos. Además, gestiona la iluminación básica de los modelos para asegurar que la interacción con fuentes de luz sea realista.

Declaración del Fragment Shader para Modelos:

- `const char* fragmentShaderSource;`

2.4.3 Fragment Shader para el Cono de Luz

Diseñado específicamente para el cono de luz del OVNI, este **Fragment Shader** simula el rayo abductor. Utiliza cálculos de iluminación como luz ambiental, difusa y especular para crear efectos visuales que imitan un foco de luz intensa. Este shader es crucial para resaltar

la acción de abducción, creando un efecto visual que indica la activación del rayo sobre la vaca objetivo.

Declaración del Fragment Shader para el Cono de Luz:

- `const char* coneFragmentShaderSource;`

2.4.4 Shaders del Laser

Para representar el rayo láser rojo del OVNI, se utilizan el **Vertex Shader del Láser** y el **Fragment Shader del Láser**. Estos shaders son responsables de la apariencia del láser y su integración en la escena. El láser es una parte integral de la narrativa visual, mostrando el momento en que el OVNI localiza y abduce a la vaca.

Declaración del Vertex Shader del Láser:

- `const char* laserVertexShaderSource;`

Declaración del Fragment Shader del Láser:

- `const char* laserFragmentShaderSource;`

Estos shaders son integrados en el programa principal a través de un proceso de compilación y enlace, asegurando que cada modelo en la escena se renderice con las propiedades ópticas adecuadas.

3 Experimentos y resultados

Durante el desarrollo de nuestra animación, se realizaron una serie de experimentos para probar el dibujado de diferentes elementos en la escena, tales como árboles, montañas y la imagen de fondo. Cada uno de estos elementos desempeña un papel crucial en la creación de un entorno inmersivo y creíble.

3.1 Dibujado de arbustos

Para la representación de los arbustos, se utilizaron modelos 3D en formato OBJ, los cuales se cargaron y dibujaron en la escena utilizando shaders específicos. La disposición regular de los arbustos fue elegida para simular un entorno natural alrededor del área de abducción. Se realizaron múltiples pruebas para ajustar la escala y la posición de los arbustos, asegurando que se vean realistas y bien distribuidos en el paisaje.

3.2 Dibujado de montañas

La representación de montañas utilizó modelos de pasto distribuidos en patrones que simulaban la elevación del terreno. Dos montañas fueron diseñadas con características de altura y espaciado específicos, que añaden profundidad y variedad al escenario, ofreciendo un paisaje más dinámico y realista. La implementación involucró ajustar la función de seno para simular la elevación natural de una montaña, y asegurar que los modelos de pasto se dispusieran adecuadamente.

3.3 Imagen de Fondo

La imagen de fondo, representando el cielo nocturno ayuda a ambientar la escena de abducción. Se utilizó una imagen que se colocó y escaló adecuadamente para que cubriera el horizonte y diera una sensación de inmensidad.

3.4 Problemas al Implementar

Uno de los principales retos fue el manejo correcto de TinyOBJLoader para la carga y renderización de modelos OBJ. Los archivos MTL, asociados a los modelos OBJ, a menudo contenían varias texturas en formato PNG, y solo se leía el último PNG, el cual se aplicaba a todo el modelo, ignorando el resto de las texturas. Este desafío nos obligó a profundizar en el funcionamiento de archivos OBJ y aprendimos la importancia de gestionar adecuadamente los recursos gráficos y de optimizar los datos de los modelos para evitar problemas de rendimiento en la aplicación.

4 Conclusiones

Durante el desarrollo de este proyecto en OpenGL, enfrentamos diversos desafíos que contribuyeron a nuestro aprendizaje en manipulación y programación 3D. Trabajamos con bibliotecas clave como GLFW, GLM, STB Image y TinyOBJLoader, lo que nos proporcionó una comprensión profunda de la renderización y animación 3D.
La implementación de shaders fue fundamental para lograr efectos visuales optimos para el proyecto. Aprendimos a utilizar GLSL eficazmente para gestionar iluminación, texturas y transformaciones de objetos, mejorando la calidad visual de nuestra animación. El manejo de iluminación y texturas presentó desafíos específicos, especialmente en escenas complejas como la abducción de la vaca por el OVNI. Ajustar los parámetros de iluminación y trabajar con STB Image para la carga de texturas fueron aspectos cruciales para mejorar la apariencia de nuestros modelos.
Además, aprendimos a equilibrar la complejidad de los modelos con las capacidades de renderizado del hardware, garantizando así una animación fluida en diferentes configuraciones de computadoras.
Finalmente, este proyecto subrayó la importancia de la planificación y organización en proyectos gráficos. Desde la gestión de recursos hasta la implementación de técnicas avanzadas de renderizado, cada paso nos preparó para futuros proyectos más complejos y realistas en el campo de la animación 3D.

5 Repositorio

El código del proyecto, los objetos 3D, texturas y un video demostrativo pueden encontrarse en el siguiente link de GitHub: `https://github.com/Proyectos-CR-CL/Proyecto-Vaca-OpenGL`

References

[1] Learn OpenGL, `https://learnopengl.com/`

[2] TinyObjLoader, `https://github.com/syoyo/tinyobjloader`

[3] OpenGL Uniform Tutorial, `https://www.khronos.org/opengl/wiki/Uniform_(GLSL)`

[4] OpenGL with C++ 8: Obj Models, `https://www.youtube.com/watch?v=myEpn0nb0gA`

[5] Spot Lights / OpenGL Tutorial #23, `https://www.youtube.com/watch?v=MAJqiDll0a8&t=220s`