UNIVERSIDAD NACIONAL

AUTÓNOMA DE MÉXICO

Facultad de Ingeniería

Computación Gráfica e Interacción Humano-Computadora

Profesor: Ing. Jose Roque Roman Guadarrama

Grupo 3

Proyecto Final

Manual técnico

Castelan Hernandez Mario

Fecha: 12 de agosto del 2021

Semestre 2021-2

Objetivos

- El alumno deberá aplicar y demostrar los conocimientos adquiridos durante todo el curso.
- El alumno deberá elaborar un proyecto en el que se demuestren los conceptos aprendidos a lo largo del curso de Computación Gráfica e Interacción Humano-Computadora.
- El alumno deberá seleccionar una fachada y un espacio que pueden ser reales o ficticios y recrearlo en 3D en OpenGL. Entrega final laboratorio

Enlace al repositorio en GitHub

https://github.com/MarioCastelan/ProyectoFinal_computacionGrafica

Alcance del proyecto

Recrear un ambiente virtual con la temática de Bob Esponja, en 3D en OpenGl, el cual constara de la habitación del personaje de Bob Esponja y la fachada, el cual es una piña.

Los 7 elementos a recrear son:

1. La cama
2. El barril
3. El reloj
4. Gary
5. Bob Esponja
6. El plato de comer de Gary con el periódico
7. El trampolín

Elementos extras y de ambientación son:

- El layout de la habitación con las paredes texturizadas
- La pelota
- Las ventanas
- Los cuadros de las puertas
- El piso
- La fachada el cual es una piña
- Las estrellas de la fachada
- Las rocas de la fachada
- Las medusas
- Las burbujas
- Camino de la fachada
- Cofre
- Arenero
- Audio (requisito para teoría)

A continuación se muestra las imágenes de referencia

Figura 1. Imágenes de referencia de la habitación y los 7 objetos

Figura 2. Imagen de referencia de la fachada

Implementar 5 animaciones, dos complejas y 3 básicas:

- Animación compleja: Movimiento del personaje Bob Esponja, el personaje corre hasta la posición de la pelota, la pelota es pateada por el personaje y esta rebota contra la pared y regresa a la posición original, por medio de la técnica de animación de KeyFrames y, rotaciones y traslaciones de la pelota.

- Animación compleja: Movimiento del personaje Gary, el personaje hace un recorrido en el escenario, sube a la puerta y la baja, avanza hacia el personaje Bob Esponja y se cola en su cabeza.
- Animación básica: Apertura de las puertas, las puertas se abren y se cierran.
- Animación básica: Movimiento del juguete de medusas, el anclaje del juguete rota hacia una dirección y las medusas de juguete rotan hacia otra dirección.
- Animación básica: Chimenea humeando burbujas, salen burbujas de la chimenea constantemente hasta que la animación sea parada, esta animación produce sonido.
- Animación extra: Activar alarma, el reloj se mueve y empieza a sacar burbujas, esta animación produce sonido.
- Animación extra: Apertura de las ventanas, las ventanas se abren y se cierran.
- Animación extra: movimiento de las medusas de la fachada, las medusas se desplazan ascendentemente, moviendo sus patitas.

Desarrollar el entorno utilizando OpenGL, mediante Visual Studio. Utilizar de base el código proporcionado por el profesor a lo largo de las prácticas de laboratorio, y modificarlo según sea necesario.

Modelar los objetos con el software de modelado Maya.

Posicionar los elementos dentro del ambiente virtual, los elementos que no son animados son posicionados con ayuda del software de modelado Maya

Carga de modelos

La carga se realiza a través de la creación de un objeto de tipo Model.

```
Model trampolin((char*)"Models/trampolin/trampolin.obj");
Model cama((char*)"Models/cama/cama.obj");
Model pelota((char*)"Models/pelota/pelota.obj");
Model reloj((char*)"Models/relojBarril/reloj.obj");
Model barril((char*)"Models/relojBarril/barril.obj");
Model plato((char*)"Models/platoGary/plato.obj");
```

Los objetos que no son animados son posicionados desde el software de modelado Maya, los objetos que son animados se posicionan manualmente.

Objetos no animados:

```
/*____________________Habitacion____________________*/
habitacion.Draw(lightingShader);
ventanaCuarto.Draw(lightingShader);
trampolin.Draw(lightingShader);
cama.Draw(lightingShader);
barril.Draw(lightingShader);
plato.Draw(lightingShader);
```

```
        marcoPuertaCuarto1.Draw(lightingShader);
        marcoPuertaCuarto2.Draw(lightingShader);
        arenero.Draw(lightingShader);
```

Objeto animado:

```
//PuertaCuarto1
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(1.7254f, 1.7624f, -6.3046));
animarPuerta(&model, &abrir_puerta1, &rotacionPuerta1);
glUniformMatrix4fv(glGetUniformLocation(lightingShader.Program, "model"), 1, GL_FALSE,
glm::value_ptr(model));
puertaCuarto1.Draw(lightingShader);
```

Cámaras

Se implementan 4 cámaras estáticas y una cámara libre. La posición de las cámaras estáticas se encuentra en la siguiente estructura:

```
glm::vec3 camarasEstaticas[] = {
     glm::vec3(3.77304f, 3.34218f, 7.98635f),
     glm::vec3(-6.32789f, 3.084f, 7.57777f),
     glm::vec3(-1.0281f, 3.15635f, 9.12915f),
     glm::vec3(-0.856808f, 2.16123f, 86.2266f)
};
```

Utilizo una variable de índice para elegir qué cámara será seleccionada

```
//camara estatica indice
int camaraEstaticaIndice = 0;
```

Para mantener estática la cámara, se realiza mediante una variable booleana

```
//Bloqueo de camara libre
bool bloquearCamara = false;
```

Con el uso de la función que procesa la entra por teclado, se realiza el cambio de cámara

```
//Cambio de camara estatica
 if (keys[GLFW_KEY_C]) {
     bloquearCamara = true;
     camera.position = camarasEstaticas[camaraEstaticaIndice];
     camaraEstaticaIndice += 1;
     if (camaraEstaticaIndice > 3)
         camaraEstaticaIndice = 0;
 }
 //Camara libre
 if (keys[GLFW_KEY_F]) {
     bloquearCamara = false;
 }
```

Funciones y su uso

```
void KeyCallback(GLFWwindow* window, int key, int scancode, int action, int mode);
```

Se encarga de procesar la entrada por teclado, las teclas que se presionan una vez para realizar una única acción. Aquí se encuentran las teclas relacionadas con la interacción y animaciones del ambiente virtual. Además es la función que procesa la tecla ESC para cerrar la ventana del ambiente virtual.

```
void MouseCallback(GLFWwindow* window, double xPos, double yPos);
```

Se encarga de procesar la entrada por ratón. Actualiza la dirección de la cámara según el movimiento del ratón.

```
void DoMovement();
```

Se encarga de procesar la entrada continuamente por teclado, las teclas que se mantienen presionadas varían continuamente un valor. En esta función sólo se encuentran las teclas relacionadas al movimiento de la cámara WASDQE y flechas del teclado.

```
void animarPuerta(glm::mat4* model, bool* puerta, float* rotacionPuerta);
```

Esta función hace que la Puerta rote 90°, toma como parámetros un apuntador de la matriz modelo, esta para aplicarle una rotación y en el código que es llamada se aplica la transformación, además recibe otros dos apuntadores Puerta y rotacionPuerta, estos son para saber que Puerta se va a animar, debido a que hay tres puertas y su estado de la rotación.

```
void animar();
```

Esta función activa o desactiva las variables de control de las animaciones de las puertas en base a la distancia entre la posición de la cámara y la posición de la puerta a animar, de esta manera solo se abrirá o cerrará la puerta en la que la cámara se encuentre cercas.

```
void animarJugueteMedusas(glm::mat4* model, int parte);
```

Esta funcion realiza la animación del juguete de medusas, recibe un apuntador a la matriz modelo y una variable parte, esta variable tiene el propósito de saber que parte del juguete de medusas se animara debido a que este objeto se compone de tres objetos, la animación la realiza a través de la rotación de los tres objetos en diferentes direcciones.

```
void animarVentana(glm::mat4* model);
```

Esta función realiza la animación de las ventanas de la fachada, recibe un apuntador a la matriz modelo.

```
void animarReloj(glm::mat4* model);
```

Esta función realiza la animación de la alarma del reloj, en la caricatura el reloj se mueve en diferentes direcciones y cada vez que suena avienta burbujas, esto es lo que se recreó, para la animación se utiliza una variable global relojAnimacionEstado, esta ayuda a llevar un control del estado de la animación y a través de traslaciones y rotaciones, además se tiene una variable de control para las burbujas activar_burbujas, esta ayuda en saber en que estado de la animación del reloj, empieza la animación de las burbujas.

```
void animarBurbujaReloj(glm::mat4* model, int burbuja);
```

Esta función realiza la animación de las burbujas que se utilizan en la animación del reloj, toma como parámetros un apuntador de la matriz modelo y una variable burbuja, esta sirve como control para saber que burbuja animar ya que se cuenta con 14 burbujas para la animación del reloj.

```
void animarBurbujaChimenea(glm::mat4* model, int burbuja);
```

Esta función realiza la animación de las burbujas de la chimenea de la fachada, toma como parámetros un apuntador de la matriz modelo y la variable burbuja, esta sirve como

control para saber que burbuja animar debido a que se cuenta con 5 burbujas para la animación.

```
void animarMedusa(glm::mat4* model, float* traslacionMedusa, float* rotacionPataMedusa, int parte, bool* estadoPataMedusa);
```

Esta función realiza la animación de las medusas de la fachada, recibe un apuntador a la matriz modelo y recibe otros tres apuntadores, estos tienen la finalidad de controlar que medusa y que parte de la medusa se está animando y trasformando su matriz modelo.

```
void keyFrameAnimacion();
```

Función que se encarga de reproducir las animaciones del personaje de Bob Esponja y Gary con Keyframes.

```
void resetElements(void);
```

Función que restablecer las posiciones y las rotaciones a los valores iniciales los personajes de Bob Esponja y Gary.

```
void interpolation(void);
```

Se encarga de realizar la interpolación necesaria para calcular los incrementos entre las posiciones de cada Frame y obtener un movimiento acumulado y no inmediato. El número de incrementos define que tan rápido y lento incrementa el movimiento hasta cierto Frame.

```
void animarPelota(glm::mat4* model);
```

Esta función realiza la animación de la pelota, recibe un apuntador a la matriz modelo, la animación es activado por una variable de control, controlada por la función keyFrameAnimacion(). Esta animación es activada cuando el personaje Bob Esponja patea la pelota.

Animaciones

Todas las animaciones que utilizan un apuntador a la matriz de modelo son llamadas de la siguiente manera:

```
//Puerta casa
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(-4.6333, 0.0000, 18.0085));
animarPuerta(&model, &abrir_puertaCasa, &rotacionPuertaCasa);
glUniformMatrix4fv(glGetUniformLocation(lightingShader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
puertaCasa.Draw(lightingShader);
```

Esto permite cambiar la matriz modelo que se esté trabajando en cada llamada de un modelo, por lo que así no es necesario establecer explícitamente las transformaciones, si no que las transformaciones son generadas en la función de la animación correspondiente, además de generar un código más limpio.

La carga de los Frames para las animaciones de Bob Esponja y Gary es cargada antes del game loop, de la siguiente manera:

```
KeyFrame.push_back(FRAME(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0));
KeyFrame.push_back(FRAME(0, 55, -110, 0, -55, 0, 0, 0, 0, 0, 0, 0, 0));
KeyFrame.push_back(FRAME(0, -110, 55, -55, 110, 0.5, 0, 0, 0, 0, 0, 0, 0));
```

Por lo que se ve en el código se modificó la estructura que guarda los Frames por un vector para facilitar la carga de los Frames.

Skybox

Para la implementación del Skybox se encuentra en una clase para no tener que hacer la declaración de los vértices en el código principal y para realizar la generación del skybox más simple como se muestra a continuación:

Codigo para caragar las imágenes que forman el Skybox

```
SkyBox fondo((char*)"Models/SkyBox/right.tga", (char*)"Models/SkyBox/left.tga", (char*)"Models/SkyBox/top.tga",
    (char*)"Models/SkyBox/bottom.tga", (char*)"Models/SkyBox/back.tga", (char*)"Models/SkyBox/front.tga",
    skyboxVertices_1, sizeof(skyboxVertices_1));
```

Codigo para dibujar el Skybox

```
fondo.Draw();
```

Sonido

Para la implementación del sonido se utilizó la biblioteca IrrKlang, el uso de esta librería es libre para usos no comerciales, pero para uso comercial se tiene que pagar una licencia.
Para reproducir un audio primero se tiene que declarar el motor de audio:

```
//Motor de sonido
irrklang::ISoundEngine* SoundEngine = irrklang::createIrrKlangDevice();
```

Una vez declarado el motor, se necesitan declarar algunas interfaces para el control del audio como pausar, reanudar o parar:

```
irrklang::ISound* sndBubujas; //Variable de control del sonido de las burbujas
irrklang::ISound* sndJuguete; //Variable de control del sonido del juguete de medusas
irrklang::ISound* sndSoundTrack; //Variable de control del soundtrack
```

Una vez hecho las declaraciones pertinentes, el audio se puede reproducir indicando la dirección de este, además se debe de usar alguna interfaz declarada anteriormente para guardar una referencia del audio que se esta reproduciendo:

```
//Inicia el soundtrack para el ambiente virtual
sndSoundTrack = SoundEngine->play2D("audio/SpongeBobClosingTheme.mp3", true, false,
true);
```

Para pausar o parar el audio se realiza de la siguiente manera:

```
sndSoundTrack ->stop(); //Detiene el sonido
sndSoundTrack ->setIsPaused(true); //Pausa el sonido
```

Referencias

Algunos modelos utilizados en el Proyecto fueron obtenidos de la siguiente página web https://sketchfab.com/
Todos los modelos obtenidos en aquella página fueron modificados tanto algunas primitivas como el texturizado debido a que los modelos que se encontró no eran fieles a la caricatura.
Otros modelos fueron desarrollados por mí mismo en Maya, al igual que algunas texturas.

Las imágenes del Skybox fueron descargas de la siguiente página web https://gamebanana.com/mods/8363

El audio para el soundtrack y para los efecto de sonido fueron descargados de la siguiente pagina Web https://www.zedge.net/ringtones

Biblioteca de audio https://www.ambiera.com/irrklang/downloads.html