Proyecto 01

Galería de arte

Computación Gráfica

UNAM 2022-2

Gibran Zazueta Cruz

09/junio/2022

1. Introducción

Se utiliza OpenGL para crear una galeria virtual de arte compuesta de objetos 3D y shaders.

La exposición consiste de 8 objetos 3D. Los objetos se cargan de un formato wavefront obj y fueron en su mayoría obtenidos de internet. Se mecionan a continuación de acuerdo a sus posiciones

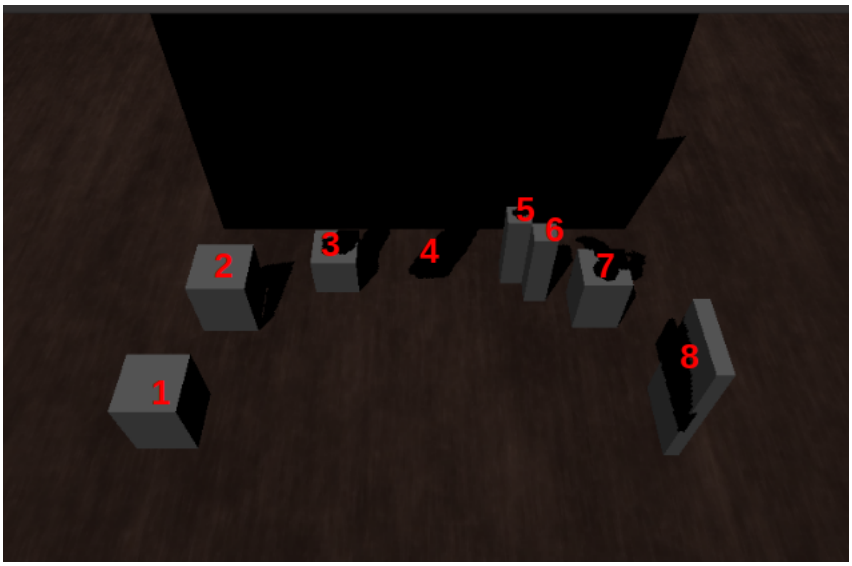

Figura 1: Posición de figuras en la escena

1. Conejo de stanford.
2. 'Vivi' (por 'Therealyton' en sketchfab)
3. Venus de milo (por la National Gallery of Denmark)
4. Estela maya (por Antoine Dresen)
5. 'Scholar'
6. Resting nude

7. Colorful rooster

8. Animación pizza

1.1. Texturizado

Se utilizaron texturas para las paredes y el suelo. Estas fueron obtenidas de 3dtexture.com.

Por otro lado, se mapea un 'skybox' con una textura cubemap alrededor de toda la escena

El cubemap es un tipo de dato de extura que se compone de 6 texturas distintas, que representan los lados de un cubo. Una ventaja de ellas es que sus cuyas coordenadas las define un vector de posición, que puede ser la posición dle fragmento. De esta manera, no es necesario especificar las coordenadas uv sobre la superficie. para realizar el mape de textura

1.2. Shaders implementados

Se implementaron los siguientes shaders:

Phong (default) y phong con textura Se programó el modelo de iluminación de Phong con sus componenetes ambiental, difusa y especular. La implementación incluye la posibilidad de hacer sombreado con variane cantidad de luces y materiales.

Mapa ambiental reflectivo Con ayuda de la textura cubemap se hace un mapeo ambiente sobre el objeto 1. Más específicamente el mapeo corresponde a un mapeo reflectivo.

Textura con animación Se crea una pequeña animación donde una textura es desplazada de acuero a una unidad de tiempo. La animación es actualizada cada frame

Toon shader Se realiza un toon shader básico forzando a la componente difusa a evitar la interpolación.

El modelo 1 utiliza el mapeo ambiental, el 8 la textura con animación y el 2 el toon shader. EL resto de los objetos utilizan el shader de phong con o sin textura.

1.3. Luces

Las luces utilizadas son una luz de sol (direccional para toda la escena) y luces seguidoras. Estas se implementaron con un ángulo de corte. Para las luces seguidoras se calculó una atenuación.

1.4. Shadow Mapping

Se implementa renderizado de sombras con la técnica de shadow mapping.- Al iniciar el programa se construye el mapa de profundidad en un framebuffer y se almacena en una textura de 2048x2048. Debido a que hay 15 luces se subdivide en mapara en 16 por lo que el mapa de sombras de cada luz es de 512x512.

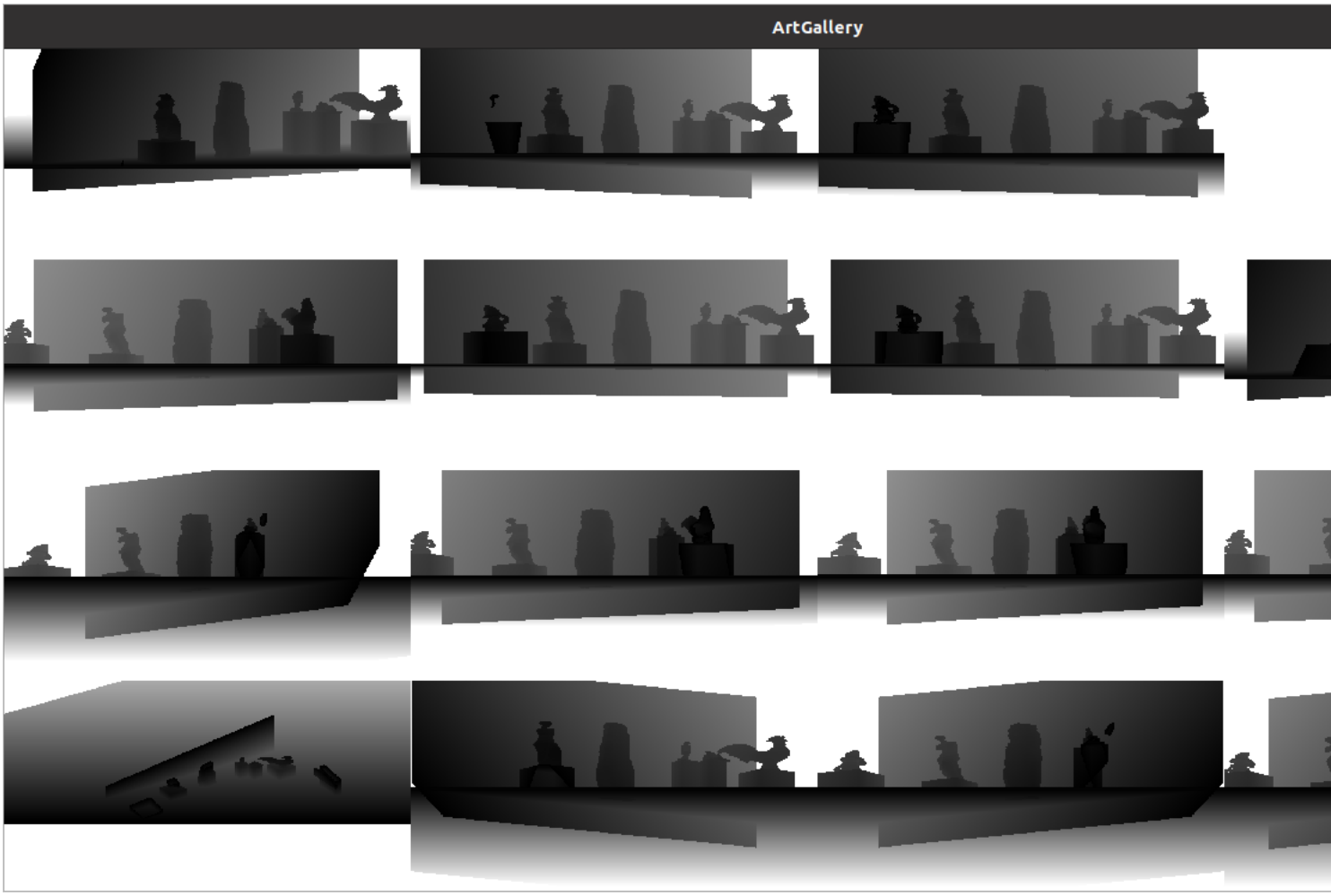

Figura 2: Mapa de sombras generado

2. Estructura del código

2.1. Object

Almacena la informaciòn del objeto a renderizar. Esto incluye coordenadas de los vertices, definición de las caras del poligono (por medio de indices a los vertices), normales de las caras, normales de los vértices, coordenadas UV e informaciòn sobre el material (coeficientes ka,kd,ke)

En el constructor de la clase de define el material a utilizar. En los materiales también se puede cargar una textura, correspondiente a ese material.

Esta clase también tiene mètodos para realizar rotaciòn de euler en x, y y z, y para calcular las normales de las caras y de los vértices. La información de las normales también se puede almancenar desde un archivo OBJ.

Por otro lado, por medio del método render se dibuja el objeto con glDrawElements. LA misma función se encarga de manejar el envio de datos en el vertex buffer.

2.2. lights

La clase *lights* almacena la informaciòn de las luces de la escena. Se define:

- Tipo (de sol o spotlight)
- Posición
- Dirección
- Color
- angulo de corte (para spotlights)
- intensidad

2.3. mainWindow

La clase de la ventana principal maneja todas las funciones para renderizar la escena. En *mainwindow.cpp* se crea el contextro de opengl, se importan los objetos, se inicializan los parámetros de las luces y la posición de la cámara. Después en *InitializeGL* se compilan los shaders con la función compileShaders. Posteriormente se crea el framebuffer y se liga su textura para dibujar profundidad FInalmente en paintGL se dibuja la escena sobre el framebuffer principal. Primero el ambiente con *renderEnviroment* y después la escena con *renderScene*

En general el proceso para renderizar un objeto es el siguiente. Si se utiliza textura se llama a setTextures, dando como argumento u string con el 'path' de la textura. Después se envían las variables necesaria al shader. Pra Phong se utliza setShaderValues. Finalmente se llama a renderizar al objeto con el método render de la clase Object.

maindown_scene.cpp

Contiene la función setScene() que importa losmodelos obj, inicializa su posición y transformaciones necesarias, modificando su matriz de modelo.

mainwindow_setShader.cpp

Contiene la función compileShaders y funciones específicas de cada shader para recibir las variables necesarias por el programa, por medio de glUniform.

mainwindow_shadowMap

Contiene la función genDepthMap()para generar el mapa de profundidad que permita realizar shadowmapping. D igual manera, la función renderShadowMapDebug() que sirve para debugiar el mencionado mapa de sombras.

3. Ejecutar el programa

En la carpeta de build se puede ejecutar el programa con el archivo ArtGallery-Run. Desde la consola de comandos de linux:

bash

```
./ArtGallery-Run
```

En la carpeta principal está el código fuente. Para generar el ejecutable primero se genera el Makefile con

bash

```
qmake ArtGallery.pro
```

Después se construye el proyecto con *make*

4. Instrucciones de uso

Se presenta la interfaz del programa.
Para navegar por el escenario se utilizan 2 juegos de teclas. Primero para cambiar la posición:

- "W". Moverse hacia enfrente
- .ª". Moverse la izquierda
- "S". Moverse hacia atrás
- "D". Moverse hacia la derecha

Para cambiar la dirección (hasta $90^{\underline{o}}$):

- Ï". Girar hacia arriba
- "J". Girar hacia la izquierda
- "K". Girar hacia abajo
- "L". Girar hacia la derecha

Para apagar las luces que iluminan los objetos se utilizan las teclas de número del 1 al 8. Cada tecla apaga la luz correspondiente a ese número de objeto. Debido a la manera en que están indexadas las luces apagar más de una a la vez provoca comportamiento erratico.
Para desactivar shaders

- *B* desactiva reflect shader
- *N* desactiva toon shader
- *M* desactiva movement shader (textura animada)

Volverlo a presionar hace que se activen de nuevo.

5. Resultados

El programa se corriò en un computadora con procesador AMD FX-8370, con tarjeta gráfica Nvidia GTX 970 y 12 gb de memoria ram.

Figura 3: Escena que se muestra la iniciar el programa

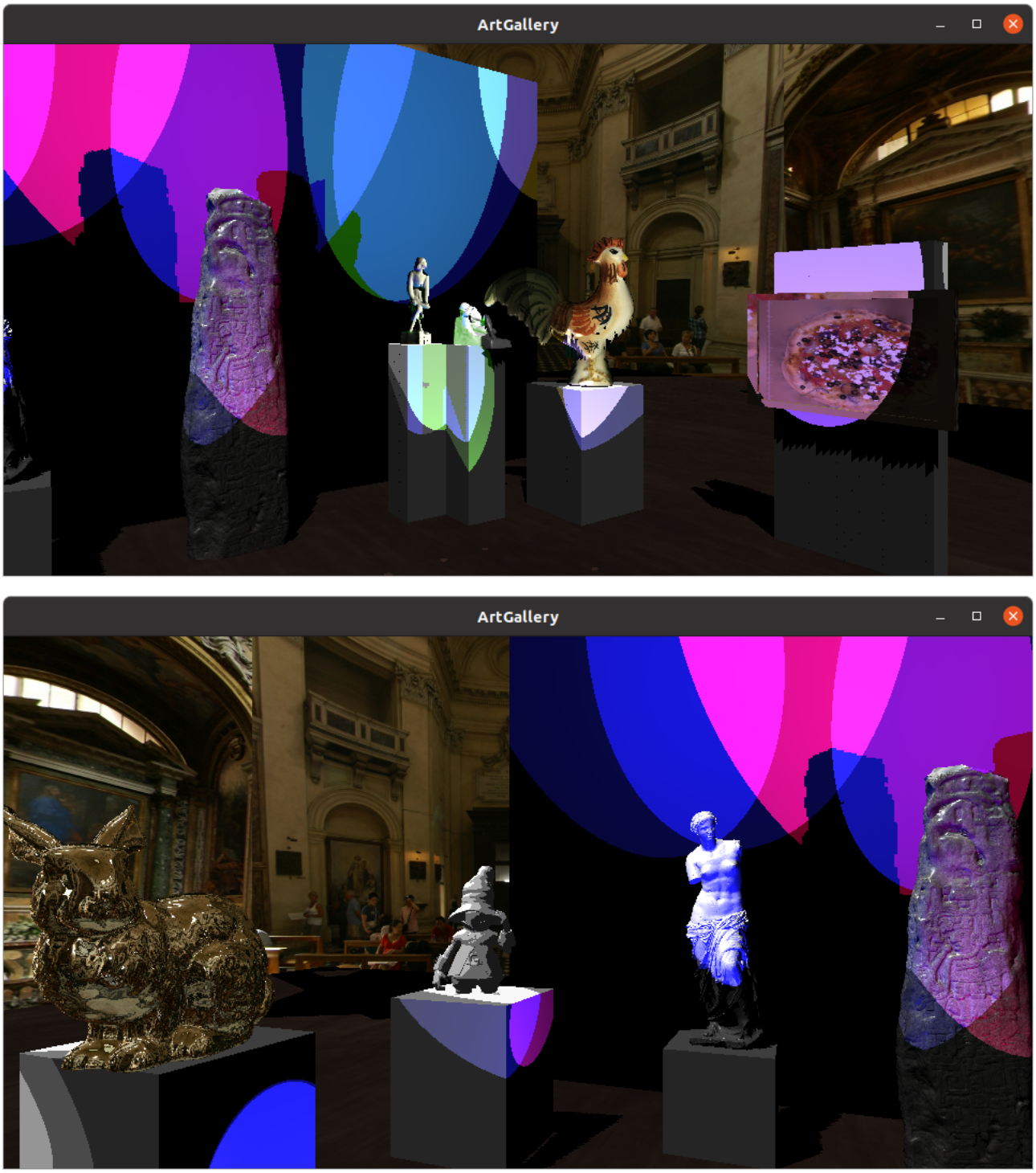

Figura 4: Galería de arte

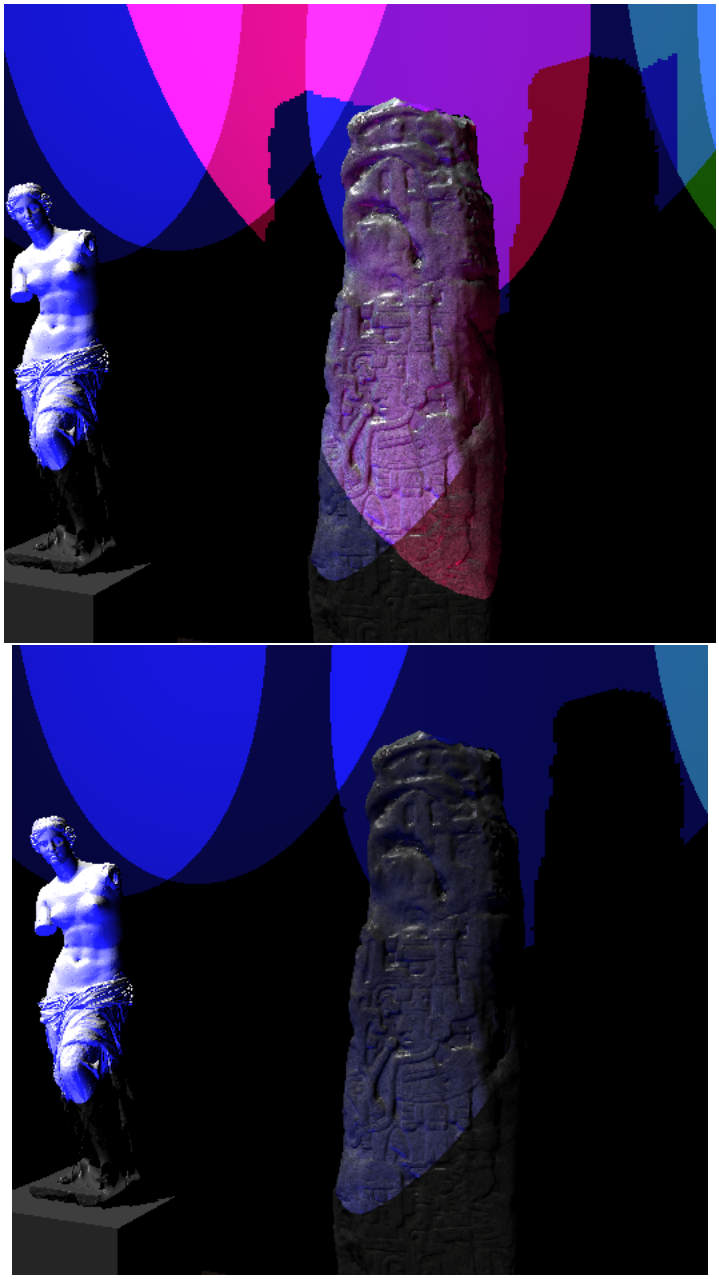

Figura 5: Desactivación de luces

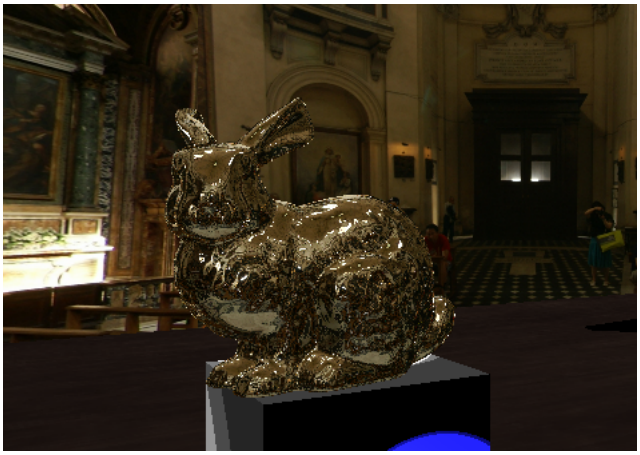

Figura 6: Mapeo ambiental reflectivo

Figura 7: Toon shader con y sin textura