Proyecto 01

Motor Grafica

Computación Gráfico

UNAM 2022-2

Gibran Zazueta Cruz

24/marzo/2022

1. Introducción

El programa que se presenta a continuación recibe una malla 3d desde un archivo OBJ y lo renderiza dentro de una escena. Los objetos se definen en 3 archivos diferentes con geometría para un cubo de 8 vértices y 6 caras cuadradas, un cubo de 8 vértices y 12 caras triangulares y una esfera con caras rectangulares.

Se implementa

el pipeline grafico 2d coordinates to rester coordinates Lectura de vertices y normales calculo de normales

Sombreado de Phong y gouroud

texturizado sobre caras especifica

2. pipeline

se proyecta sobre una vetnana de 400x400

2.1. Proyecciòn con la càmara

Para lograr la proyección se implementaron las matrices de transformación necesarias. Primero la cámara recibe su posición desde la matriz de transformación mundo-cámara.

$$\begin{bmatrix} M_c^T & M_c^T o_w \\ O^T & 1 \end{bmatrix} \tag{1}$$

Donde M_c^T son los vectores unitarios del marco de la cámara y o_w es el desplazamaiento con respecto al mundo. Para las dos vistas de las cámaras utilizadas en la escena resultan las siguientes matrices:

$$M_{camara1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -25 \\ 0 & 0 & 0 & 1 \end{bmatrix} M_{camara1} = \begin{bmatrix} 0 & 0 & 1 & -10 \\ 0 & 1 & 0 & 5 \\ -1 & 0 & 0 & -10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ARREGLO DE OBJETO CON CAMARAS

3. scan convertsion interpolacion

A continuaciòn se explica el método de scan conversion
Para lograr rellenar un polígono primero se almacena la información de sus aristas en un *Buffer*.

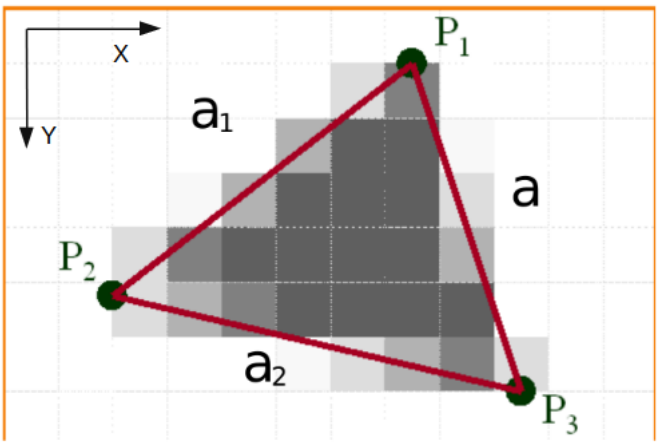

Figura 1: Algortimo rellenado de poligonos con Scan Conversion (imagen obtenida de [**?**])

Se siguen las aristas de la figura, una por una. Se calcula la pendiente de la recta y se va avanzando una unidad (de pixel) por y, calculando el valor correspondiente para x. Este proceso se debe realizar en orden, es decir, las aristas se siguen en orden horario o antihorario.
Tomando como ejemplo la figura 2, un procesamiento antihorario sería: $a_1 \rightarrow a_2 \rightarrow a_3$.

La idea es dividir las aristas del polígono entre un 'lado izquierdo' y un 'lado derecho' que permita colorear los pixeles desde un borde hasta el otro, por cada coordenada y. Con un procesamiento de este tipo y para un polígono convexo el buffer almacenará 2 coordendas de x por cada coordenada de y.

La manera en que se clasifican los bordes, entra parte derecha e izquierda, es tomando en cuenta el valor menor entre los dos vértices que forman la arista (por ejemplo P_{1_y} y P_{2_y} para la arista a_1).
Para un recorrido antihorario, mientras el valor del primer punto de la arista sea menor al del segundo se considera que esta pertenece al lado izquierdo. Cuando el valor del segundo punto de la arista sea menor al primero, la recta pertenece al lado derecho.
La información de las coordendas x (derecha e izquierda) correspondientes a cada coordenada y se almacena en el buffer.

Una vez se tiene la información del buffer, se recorre la figura de arriba hacia abajo (tomando en cuenta el eje de coordenadas del dispostivo, esto corresponde a un incremento en y), donde por cada valor de y se colorean los pixeles desde la coordenada x derecha hasta la x izquierda.
Esta implementación´ está basada en la información presentada en [1]

4. shading

se utiliza gouroud y phong
Componente ambental, difusa y especular

Ecuaciones
Ambiental
Se calcula como

$$I_A = \kappa_A \Lambda_A \tag{2}$$

La luz difusa se calcula como

$$I_D = \kappa_D \Lambda_D (\vec{n} \cdot \vec{l}) \tag{3}$$

especular El calcula es el siguiente

$$I_E = \kappa_E \Lambda_E (\vec{o} \cdot \vec{l})^\rho \tag{4}$$

donde $\vec{o}$ es el vector de dirección del observador. La ecuación (3) también se puede escribir

$$I_E = \kappa_E \Lambda_E (\vec{o} \cdot \vec{h})^\rho \tag{5}$$

donde $h = 2(\vec{o} \cdot \vec{l})\vec{n} - \vec{l}$
SE INTERPOLAN NORMALES Y LUZ COOM SE CALCULAN?

4.1. materiales y escena

Los objetos cuentan con 2 posibles materiales a seleccionar, estos se definen dentro del código como:

Material 1

- Ambiental = 0.0, 0.0, 0.0, 1.0,
- Difusa = 0.50, 0.50, 0.50, 1.0,
- Especular 0.70, 0.70, 0.70, 1.0
- $\rho = 32.0$.

Material 2

- Ambiental = 0.23125, 0.23125, 0.23125, 1.0,
- Difusa = 0.2775, 0.2775, 0.2775, 1.0,
- Especular 0.773911, 0.773911, 0.773911, 1.0
- $\rho = 89.6$.

Finalmente, la escena a renderizar cuenta con 2 luces (blanca y azul) y 3 cámaras.
Se disponen de la siguiente manera:

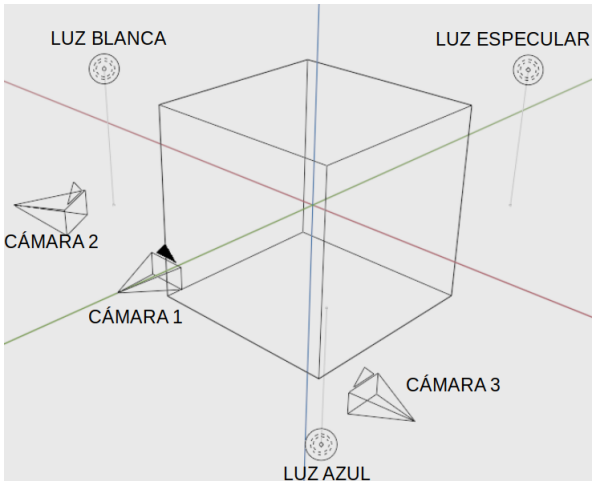

Figura 2: Posición de las componentes de la escena

5. Archivo obj

se toma informaciòn de los vèrtices y de las normales, estos datos fueron proporcionados LA NORMAL SE PUEDE CALCULAR

5.0.1. Librería *assimp*

Para leer los archivos se utiliza la libreria de assimp. *Assimp* es una librería open source que soporta multiples formatos de geometria 3d, entre llos .OBJ. Además funciona con diversos sistemas operativos y provee una interfaz de C++.

También soporta una jerarquía de nodos para mallas, materiales, texturas y animaciones de *bones*. Para el programa de este reporte se hace uso principalmente de la función *import* clase Assimp::Importer que almacena datos es una estructura llamada aiScene.

6. Estructura del código

en mainwindow se define la gui y es **cubeobject** Almacena la infromaciòn 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 los vértices, normales de las caras, coordenadas uv e informaciòn sobre el material, esto es coeficientes ka,kd,ke

En los materiales tambièn se cargan las texturas

en el constructor de la clase de definen los 2 materiales a utilizar ESta clase tambièn tiene mètodos para realizar rotaciòn de euler en x y y y para calcular las normles de las caras y de os vértices. la información de normales tambièn se puede almancenar del archivo obj

7. raster

raster es la clase mas importante ya que en su mètodo pipeline se realiza la proyecciòn, scan conversiòn y sombreado

Para la proyecciòn se utliza la clase camProjection entrega las coordenadas de los vertices en el dispositivo e informaciòn de profundidad

después se calculan los primeros valores para los vectores N,L,O, con ellos se pueden calular el valor de color en los vèrtices, esto es ùtil para realizar la interpolaciòn de sombreado de gouroud

con el método *fillCubeFace* se utiliza el aloritmo de scan conversion descrito anteriormente para rellenar los pixeles del polígono y a su vez para interpolar N,L,O COlor y coordenadas de yextura

la primera interpolaciòn sobre los bordes se realiza en scanline. se busca aplicar scanline entre todos los vèrtices del polìgonos. posteriormente se realiza la interpolaciòn horizontal. PAra ella se utilizan las funciones *scanConversion* y *horInterpolation*

tambièn aqui se limita el dibujar texturas a ciertas caras especificadas

lights LA clase lights almacena la informaciòn de las luces de la escena. Se define su intensidad, color y posiciòn

8. renderwindow

se utiliza qt para pimtar sobre el canvas aqui se toma en cuenta la profundidad interpolad (o mas bien el inverso 1/z) el punto solo se dibujarà en el canvas si cumple la restricci´on de profundidad

mainWindow Desde *mainwindow.cpp* se llama a la función *importFile()* que está definida en *functions.cpp*. La función recibe el path del archivo (como una cadena std::string) y apuntadores al contenedor de vertices y caras del objeto cubeObject,

Después de esto, en mainwindow se crean los objetos de clase *light* para las luces blanca, azul y especular, se especifican intensidad, color y posicionamiento. También se crea el grid de botones que conforma la GUI de la aplicacion. Finalmente se llama a la funcion *drawObject()*

En la funcion *drawObject()* se llama a la función pipeline del objeto raster, que como ya se mencionó relaza todo lo referente a la proyección, rellenado y sombreado.

9. Ejecutar el programa

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

bash

```
./GraphicsEngine-Run
```

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

bash

```
qmake GraphicsEngine.pro
```

Después se construye el proyecto con *make*

10. Instrucciones de uso

Se presenta la interfaz del programa.

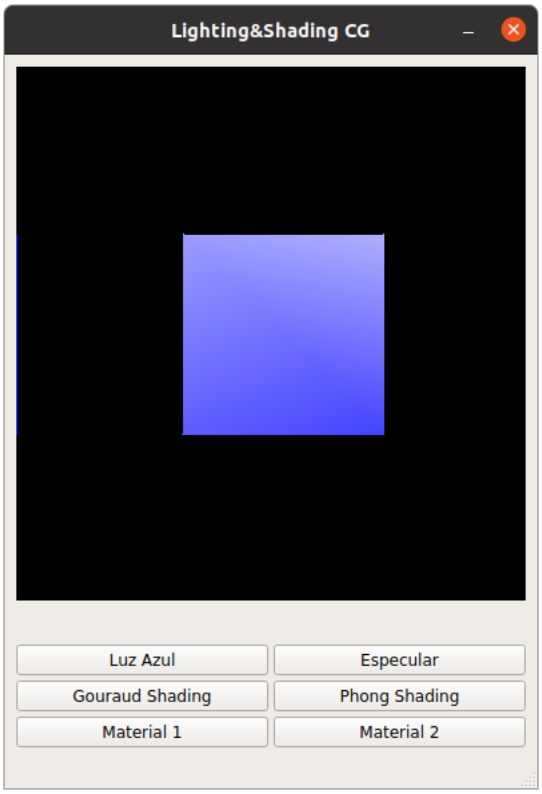

Figura 3: Interfaz gráfica del programa

Para cambiar entre las camaras se utilizan las teclas de los numeros

- "1". Cambia a la cámara 1
- "2". Cambia a la cámara 2
- "3". Cambia a la cámara 3

Para rotar el objeto sobre el eje X se presiona la tecla R. Se vuelve a apresionar para que deje de rotar.

Los botones *Luz Azul* y *Especular* encienden o apagan las luces correspondientes. El programa inicia con todas las luces activas.

Gouraud Shading activa el Sombreado por Gouraud y *Phong Shading* el sombreado de Phong.

Es posible cambiar entre archivos desde el código. En la variable path de mainwindow se cambia la extension por el archivo correspondiete. Por defecto es $Cube\_Triangle$, los otros dos son $Cube\_Quads$ y sphere.

11. Programa en ejecución

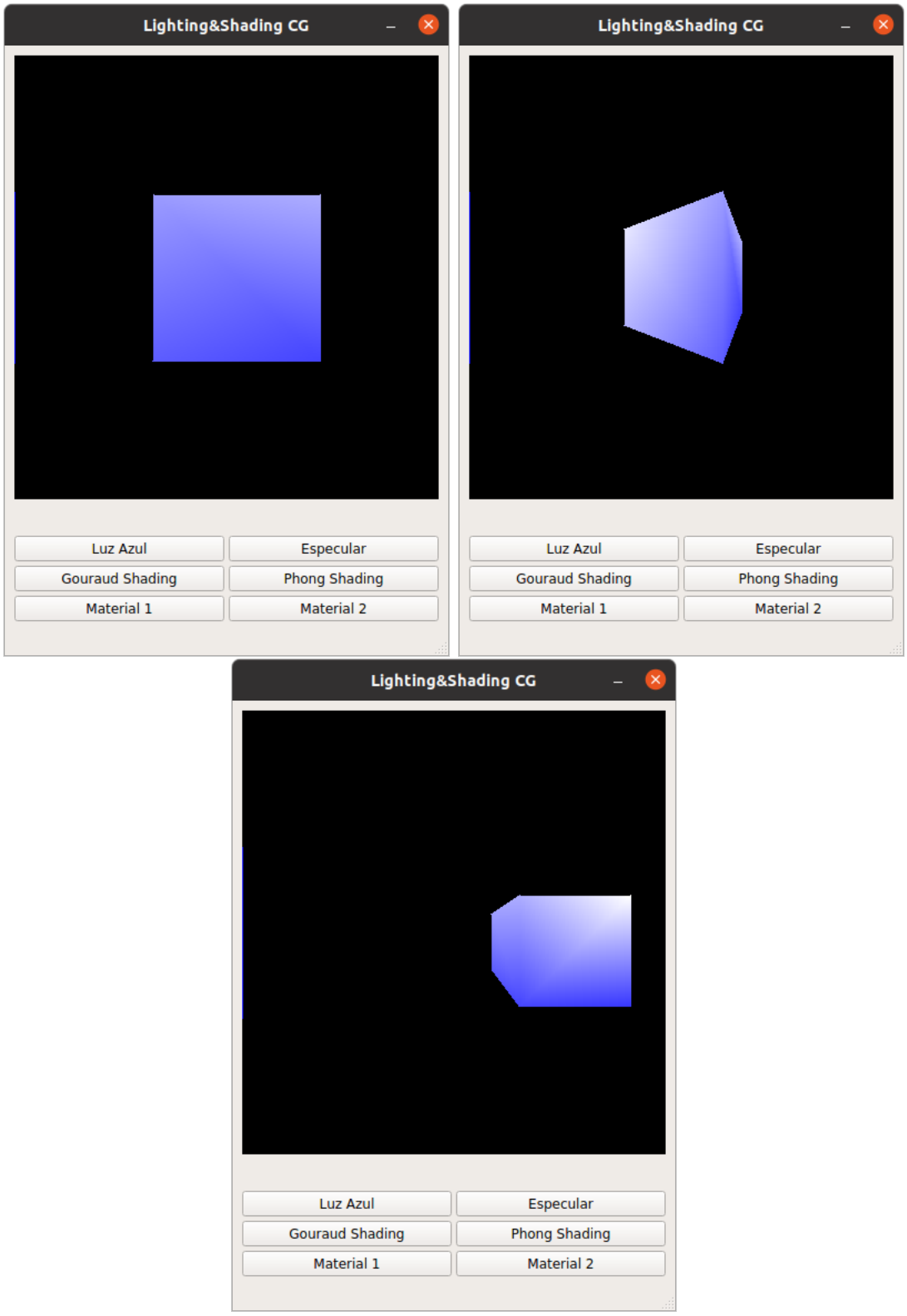

Figura 4: Gouraud Shading. Cámara 1, 2 y 3. Material 1

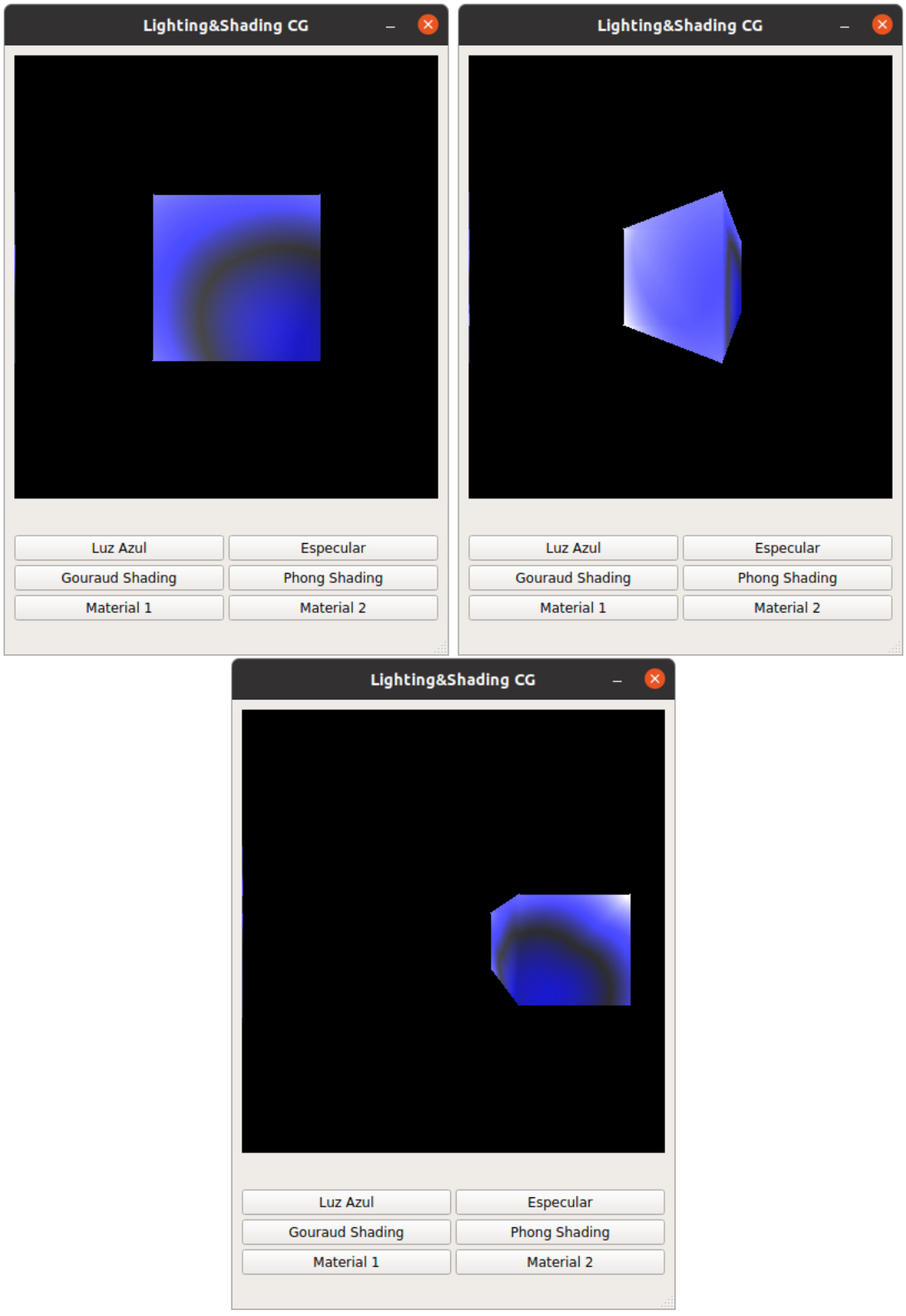

Figura 5: Phong Shading. Cámara 1, 2 y 3. Material 2

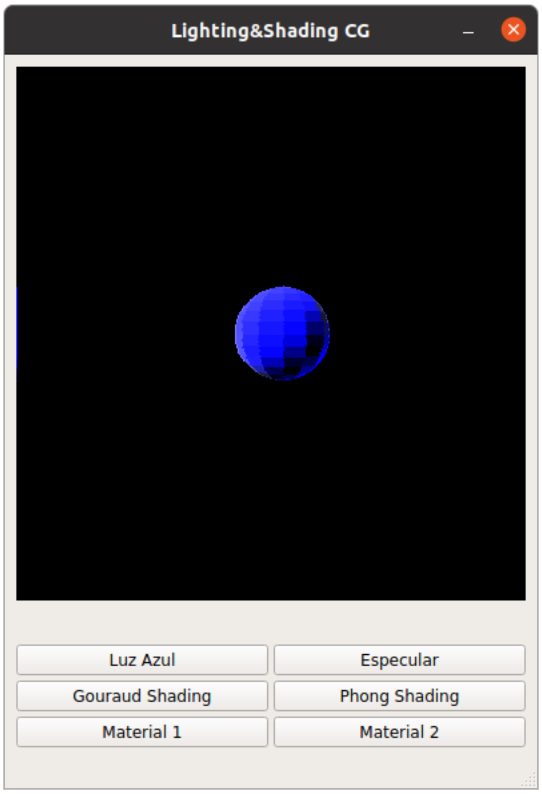

Figura 6: Renderizado esfera

Referencias

[1] Upssala Universit. Introduction to polygons. ($http://www.it.uu.se/edu/course/homepage/grafik1/ht06/Lectures/L02/LinePolygon/x_polyd.htm$)