GRADO EN DISEÑO

Y

DESARROLLO DE VIDEOJUEGOS (MÓSTOLES) (2020-21)

Informática Gráfica

Práctica 1: Introducción a GLSL

Participantes del grupo:
Pablo Burgaleta de la Peña
Raquel Mijarra Benítez
Adrián Mira García
Iván Pérez Ciruelos

Parte obligatoria

1. Define una matriz de proyección que conserve el aspect ratio cuando cambiamos el tamaño de la ventana.

Para definir una matriz de proyección que conserve el aspect ratio tenemos que definir una variable a_ratio, a la cual hay que forzar que sea float ya que la división entre width y height da un número entero, añadiendo el valor de esta variable a las variables right y left.
Además de las variables a_ratio, right y left; definimos las variables near, far, top y bottom con los valores {1.0, 15.0, 1.0, -1.0} respectivamente.

$$\begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Para la resolución de este apartado hemos consultado el código de esta página y adaptado el mismo para nuestra práctica.
https://es.stackoverflow.com/questions/206323/mantener-aspect-ratio-al-cambiar-tama%c3%b1o-de-ventana

2. Añade un nuevo cubo a la escena. El segundo cubo deberá orbitar alrededor del primero describiendo una circunferencia a la vez que rota sobre su eje Y.

Para crear un nuevo cubo a la escena hemos seguido los pasos marcados en clase para la creación de un cubo, definiendo sus vértices y la posición de los mismos.
Para su rotación alrededor del primer cubo y sobre su eje Y nos apoyamos en la función *idleFunc()* ya definida en el programa y la opción glm::rotate.

3. Control de la cámara con el teclado (First Person Shooter). Controles mínimos que deberán incluirse: movimiento hacia adelante, retroceso, movimientos laterales (izquierda y derecha) y giros (izquierda y derecha).

Para obtener el control de la cámara con el teclado tenemos que rehacer la matriz view, para ello definimos como variables globales los vectores que la componen: UP, LookAt y Cop (Pos).

Para el control de la cámara hacia adelante y hacia atrás tenemos que variar la posición de la cámara teniendo en cuenta el vector LookAt y el incremento deseado en la posición.
Para el control de la cámara hacia la derecha y la izquierda tenemos que variar la posición de la cámara teniendo en cuenta el vector Right (definido previamente como el producto vectorial de LookAt y Up) y el incremento deseado en la posición.
Para el control de la cámara en los giros hemos redefinido el vector LookAt respecto al coseno y seno de un ángulo determinado.

Para la resolución de este apartado hemos consultado el código de estas páginas y adaptado el mismo para nuestra práctica.
http://acodigo.blogspot.com/2016/04/tutorial-opengl-camara.html
https://learnopengl.com/Getting-started/Camera
http://www.opengl-tutorial.org/es/beginners-tutorials/tutorial-6-keyboard-and-mouse/

4. Crear un shader de vértices y otro de fragmentos de forma que:
a. Cuando el índice de la primitiva (gl_PrimitiveID) sea impar, el color del fragmento este determinado por su normal en coordenadas de la cámara.
b. Cuando el índice de la primitiva (gl_PrimitiveID) sea par, se pinten las coordenadas de textura.

Para este apartado han sido creados el shader de vértices “shader.A4.vert” y el shader de fragmentos “shader.A4.frag”.
En el shader de fragmentos obtenemos el Id de cada primitiva para así poder pintar cada una de una manera diferente.
En el shader de vértices definimos las variables y vectores que necesitamos para pintar cada primitiva de una forma diferente.

5. Crear un shader de vértices y otro de fragmentos de forma que se descarten algunos de los fragmentos de las caras del cubo. Nota: debes utilizar la sentencia discard.
a. Opción 1: Crea una textura en blanco y negro. Asigna dicha textura al cubo que se ha usado en las prácticas. Descarta los fragmentos que tengan asociado el color negro.

Para este apartado han sido creados el shader de vértices “shader.A5.vert” y el shader de fragmentos “shader.A5.frag”.
En el shader de fragmentos pintamos el cubo con una textura en blanco y negro de la forma aprendida en clase para después, con la sentencia discard poder eliminar las partes en negro habiéndolas identificado previamente.
En el shader de vértices definimos las variables y vectores que necesitamos para pintar cada primitiva de una forma diferente.