**UNIVERSIDAD**

**REY JUAN CARLOS**

GRÁFICOS 3D

MÁSTER DE INFORMÁTICA GRÁFICA

JUEGOS Y REALIDAD VIRTUAL

PRÁCTICA Nº1

|  |  |
| --- | --- |
| Integrantes: | Rizzo Carlos  Joan |
|  | Cristian Adrian Penizzotto |
|  |  |
|  | Periodo 2019-2020 |

Contenido.

[1. Objetivo. 3](#_Toc30164741)

[2. Desarrollo. 3](#_Toc30164742)

[3. Desarrollo de Parte Obligatoria 3](#_Toc30164743)

[3.1. Matriz de proyección con aspect ratio. 3](#_Toc30164744)

[3.2. Segundo cubo en escena 4](#_Toc30164745)

[3.3. Control de cámara por teclado 5](#_Toc30164746)

[3.4. Descartando fragmentos 7](#_Toc30164747)

[3.5. Conclusiones. 8](#_Toc30164748)

# Objetivo.

El objetivo de la práctica es entender los fundamentos de programación de “shaders”, explicados en clase.

# Desarrollo.

A continuación, se describe de forma abreviada los resultados obtenidos para las distintas instancias del práctico.

# Desarrollo de Parte Obligatoria.

# Matriz de proyección con aspect ratio.

En Figura 3.1‑1 se observa la definición del aspecto ratio por código, para luego incluirlo en los recortes con la matriz de proyección.

En Figura 3.1‑2 se comparan dos ventanas de distinto tamaño, en donde se observan los objetos manteniendo su relación de aspecto.

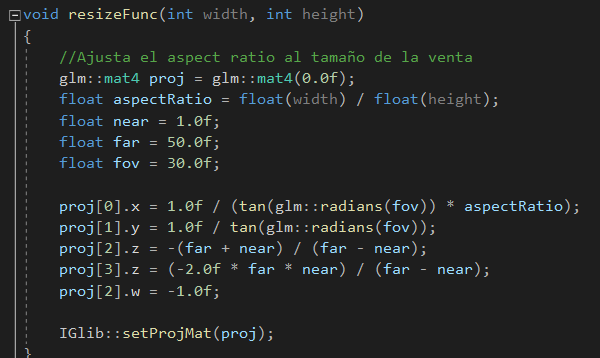


Figura 3.1‑1 Aspect ratio

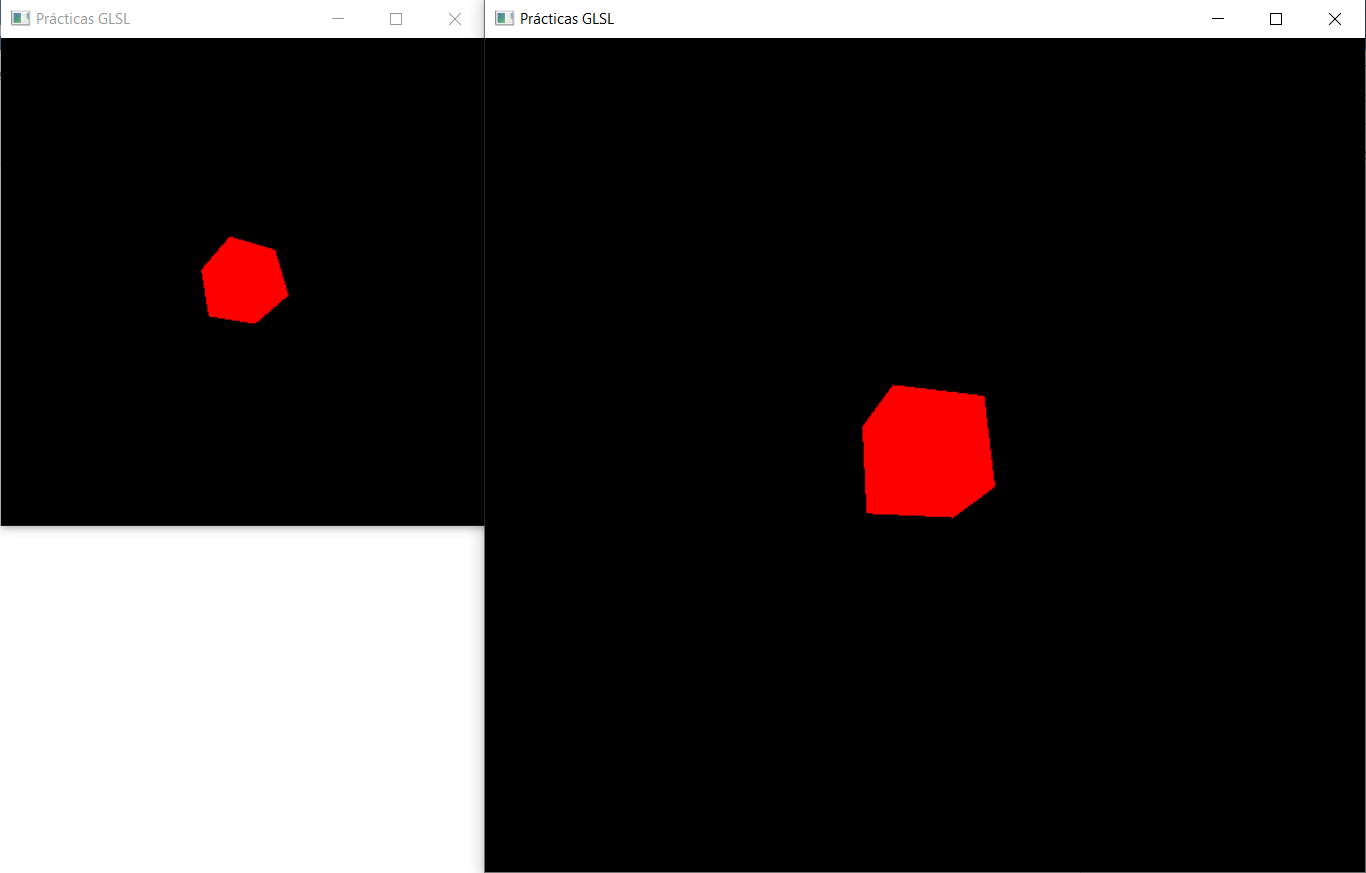
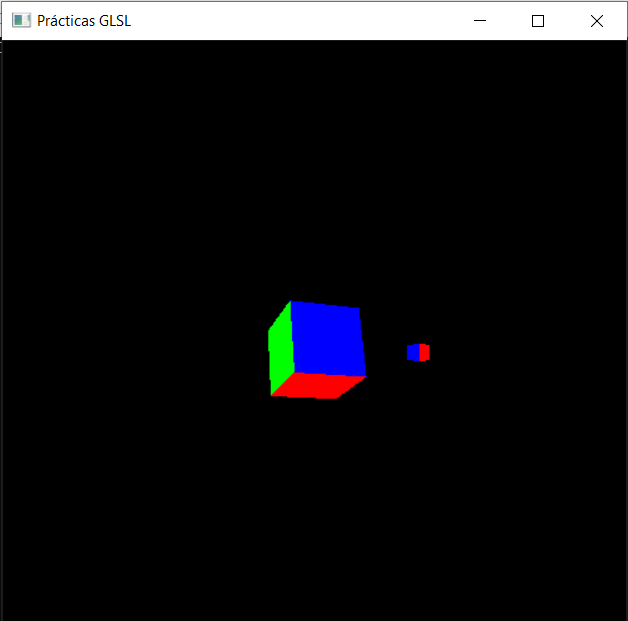


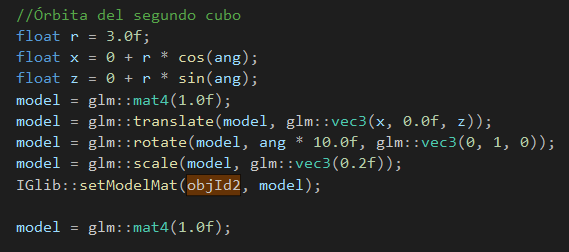
Figura 3.1‑2 Visualización del aspect ratio

# Segundo cubo en escena.

En Figura 3.2‑1 se observa el cubo agregado a la escena orbitando alrededor del primer cubo, sobre una trayectoria circular. Ver en Figura 3.2‑2 el código agregado para la órbita del cubo.

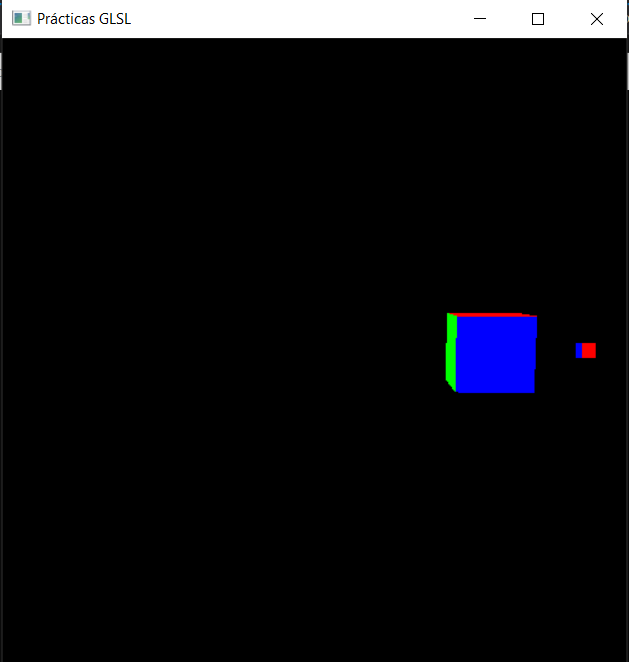


**Figura 3.2‑1** Segundo cubo en escena

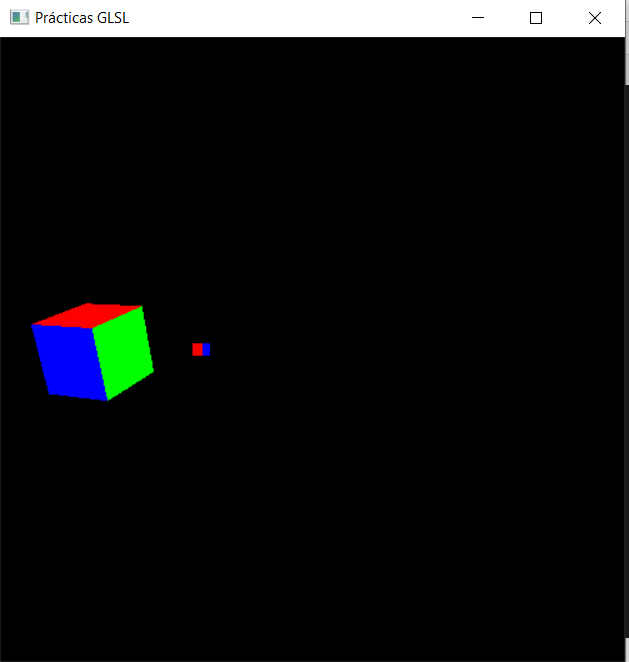


**Figura 3.2‑2** Código de órbita de segundo cubo

# Control de cámara por teclado.



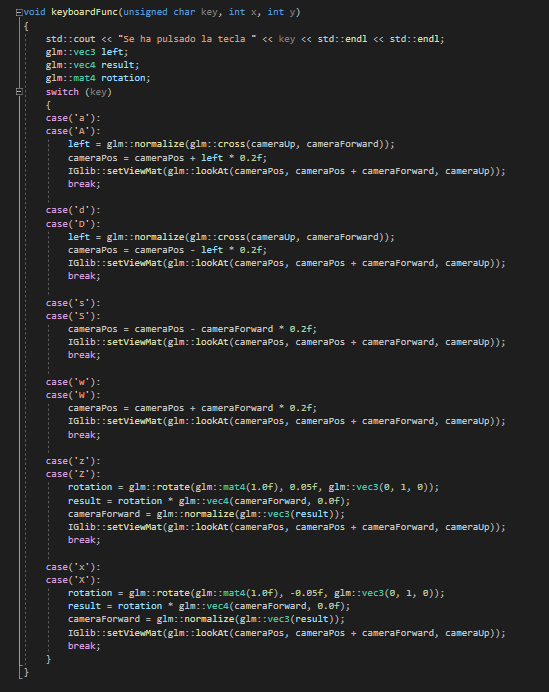
**Figura 3.3‑1** Cámara hacia la izquierda



**Figura 3.3‑2** Cámara hacia la derecha

En la Figura 3.3‑1 y Figura 3.3‑2 se observan a los cubos en dos posiciones distintas de la pantalla. El movimiento de la cámara respecto del mundo se hace, en este caso, por teclado. En el caso de la Figura 3.3‑1 ,movimiento de cámara hacia la izquierda, se logra con la tecla “A” y para el movimiento de cámara hacia la derecha; con la tecla “D”, ver Figura 3.3‑2.

En Figura 3.3‑3, se muestra el código para esta funcionalidad.



**Figura 3.3‑3** Código para movimiento de cámara por teclado

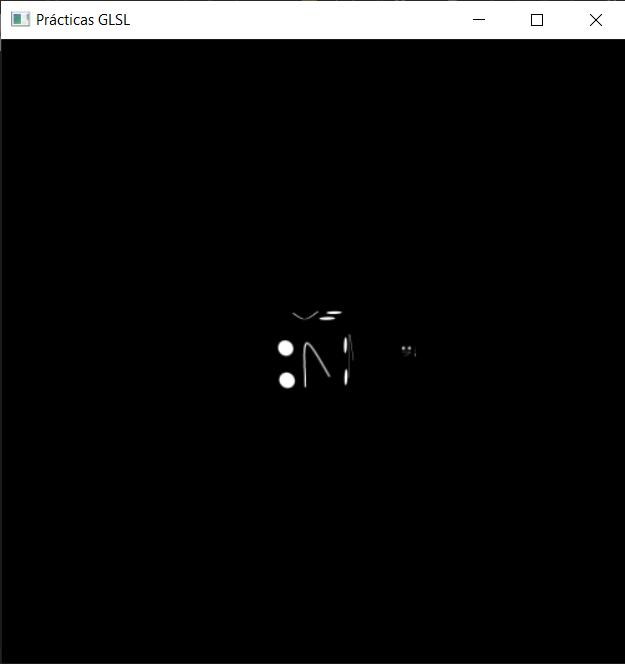
# Descartando fragmentos.

En este apartado se agrego una textura en los cubos y se configuraron los shaders para eliminar fragmentos asociados al color negro.

En Figura 3.4‑1 se muestra la textura utilizada y en Figura 3.4‑2 se observan a los cubos orbitando son esta textura, en donde en sus caras se ha pintado solo el color blanco.

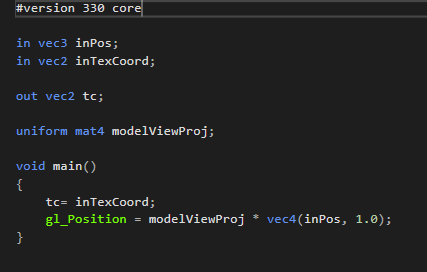


**Figura 3.4‑1** Imagen de textura

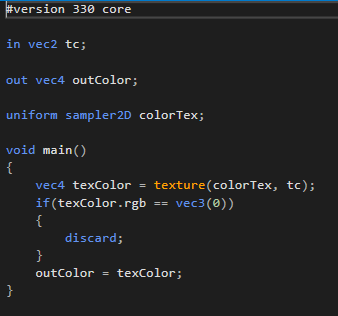


**Figura 3.4‑2** Cubos con texturas

En la Figura 3.4‑3 y Figura 3.4‑4 se muestra el código programado en los shaders.



**Figura 3.4‑3** Código shader.v10.vert



**Figura 3.4‑4** Código shader.v10.frag

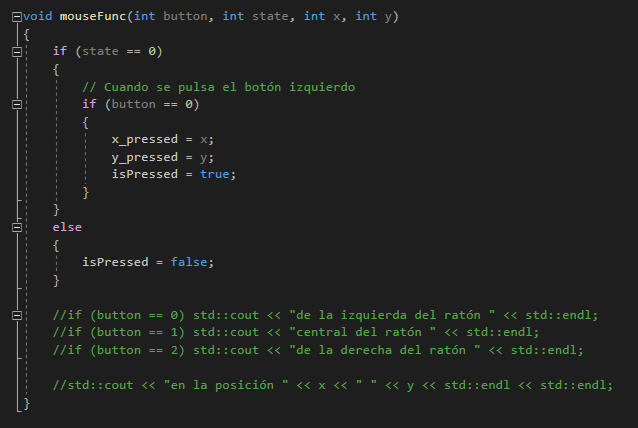
# Desarrollo de Parte Opcional.

# Control de giro de cámara con mouse.

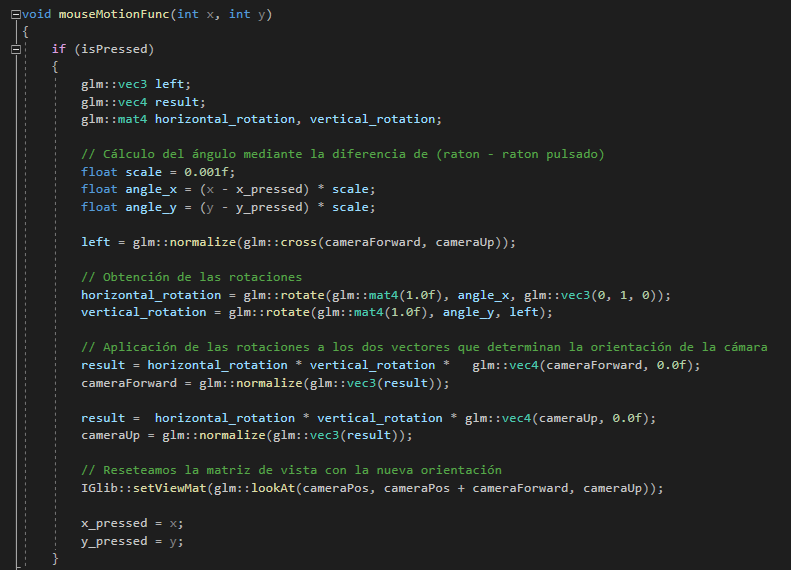
A continuación, se muestra lo realizado para el control de la cámara por medio del mouse.

En Figura 4.1‑1 se muestra el código programado de la función “mouseFunc” para el registro de las pulsaciones del mouse.

En Figura 4.1‑2 se muestra el código programado de la función “mouseMotionFunc”, con la cual se convierten las pulsaciones del mouse en movimientos de la cámara.



**Figura 4.1‑1** Código de función mouseFunc

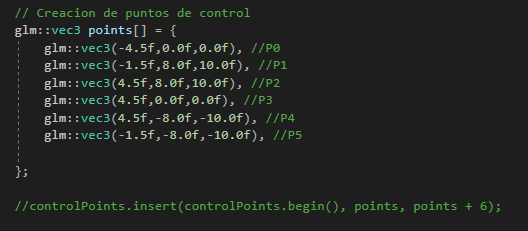


**Figura 4.1‑2** Código de función mouseMotionFunc

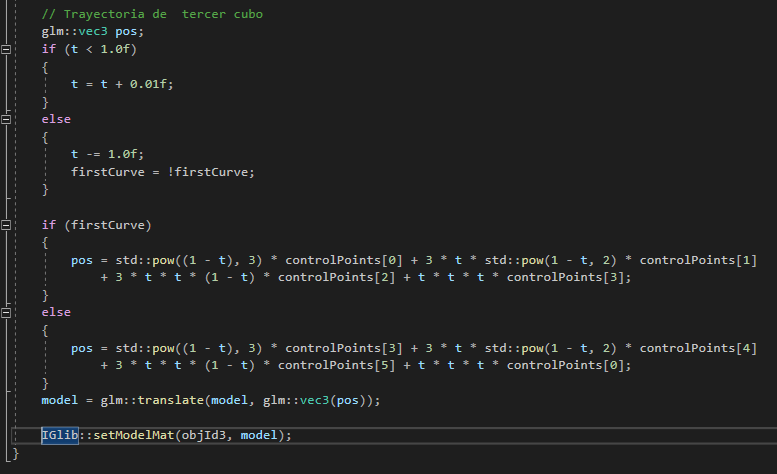
# Tercer cubo en escena.

A continuación, se agrega un tercer cubo en escena, orbitando alrededor de los otros dos, sobre una curva de bezier.

Se programaron 2 curvas de bezier para generar una órbita semejante a un oval. Las curvas se programaron con 4 puntos cada una siendo los puntos extremos los mismos en ambas para que estén unidas. Los puntos están posicionados en un plano inclinado, de tal modo que cuando el cubo está en la parte posterior de la pantalla, la cota en “y” es positiva; y cuando el cubo está en la posición mas adelantada, es decir, delante de los otros dos cubos, el tercer cubo está en cotas negativas.



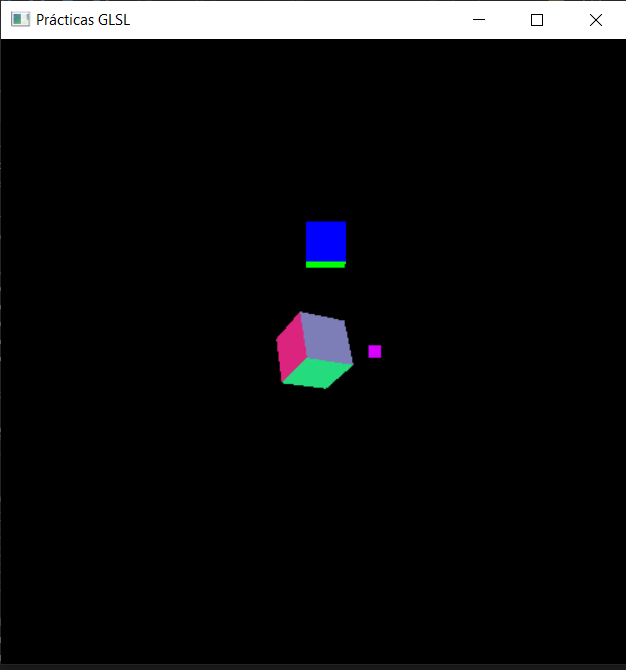
**Figura 4.2‑1** Puntos de control



**Figura 4.2‑2** Trayectoria del tercer cubo



**Figura 4.2‑3** Tercer cubo orbitando



**Figura 4.2‑4** Tercer cubo orbitando

En Figura 4.2‑3, se muestra al tercer cubo orbitando en la parte delantera, cerca del near en cotas “Y” negativas.

En Figura 4.2‑4, se muestra al tercer cubo orbitando en la parte posterior, más cercano del far en cotas “Y” positivas.

# Conclusiones.

Mediante la práctica se obtuvo manejo y entendimiento de la programación de shaders. Así como el control de la escena mediante la interacción cliente-servidor.