SEMANA 1 PRACTICA 1

Ejercicio 1:

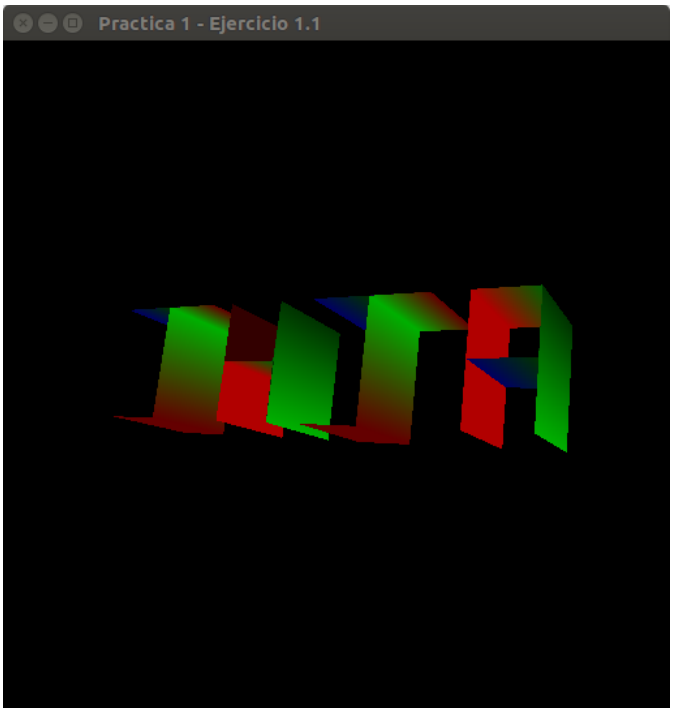
Para el primer ejercicio usamos la primitiva QUAD_STRIP para formar los cuadrados.
Primero declaramos los vertices de cada una de las letras, luego hacemos una translacion dependiendo del orden en que van las letras, indicamos el color del vertice, y finalmente declaramos su normal.

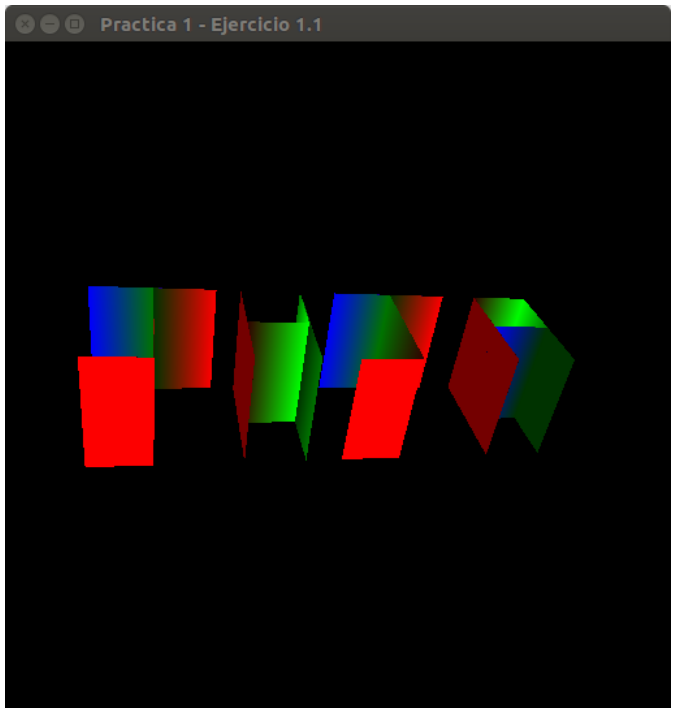
En la imagen se mueve el objeto(las letras), la camara es estatica y esta ubicada en (0,0,3).
La fuente de luz es direccional por lo que solo necesitamos indicar la direccion.
Para el shading en el material estamos incluyendo la componente difusa y ambiental.

Aqui mostramos como se muestra en la inicializacion.

Podemos rotar la imagen con el mouse, atravez del eje-x y el eje-y.
Como la camara es estatica entonces al girar los objetos deberian de iluminarse las caras que estan expuestas a la luz (en nuestro caso atravez del eje z), surge un problema por que por cada vertice se puede setear solo una normal , cuando el angulo entre la luz y la normal sea cercano a 0 la iluminacion sera maxima, pero cuando el angulo es 180, la iluminacion es minima , ya que el cos(theta)<0.0 y ahi tambien deberia iluminarse , para solucionar esto y dar el efecto de que se alumbra en las dos direcciones, por cada poligono superponemos otro con la normal opuesta.

A continuacion mostramos una imagenes en donde hemos rotado las letras.





Ejercicio 2:

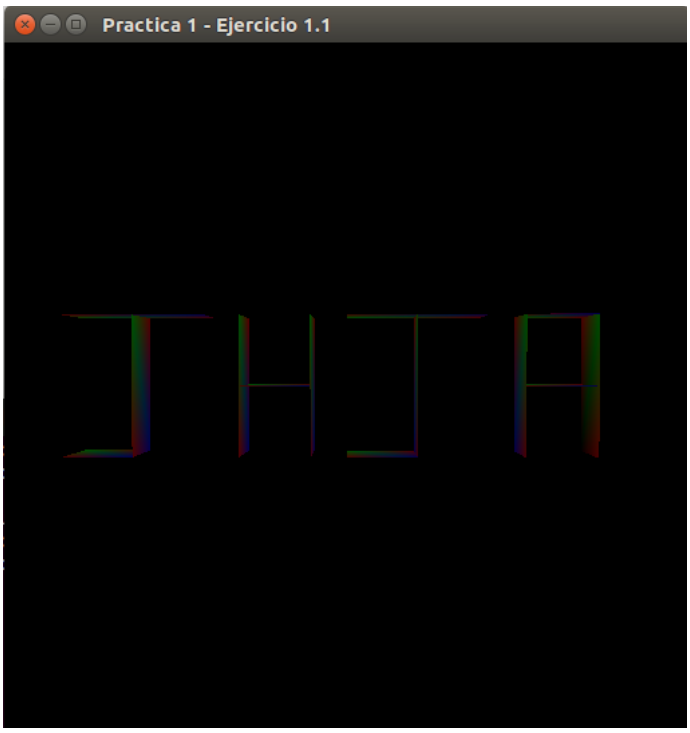
Nuestro vertex shader y fragment shader estan en nuestro ejercicio2.cpp, la configuracion como la camara y la luz son la mismas que la del ejercicio1.

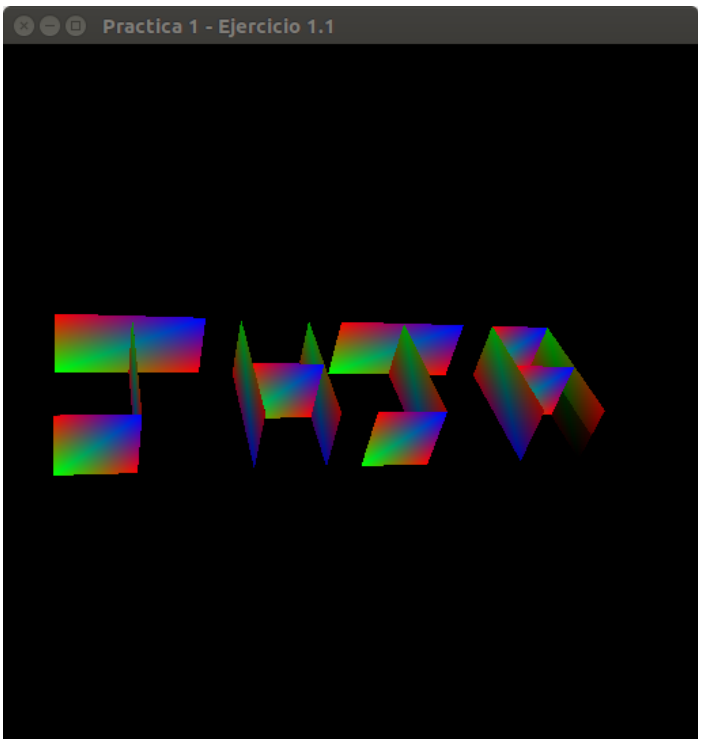
Declaramos un array para cada letra, cada array contiene los ubicacion, color y normal de cada vertice, luego declaramos 3 VBO (como tenemos 2 J solo necesitamos copiar una vez la J) para pasar los datos a la tarjeta grafica y el vertex shader trabaje sobre ellos.
Para renderizar cada letra usamos los mismo shaders, ya que para cada letra se tiene el mismo funcionamiento lo unico que cambia es su posicion en el model world, pero si se tuvo que declarar 3 VAO, ya que hay que indicarle al opengl sobre que VBO vamos a trabajar.

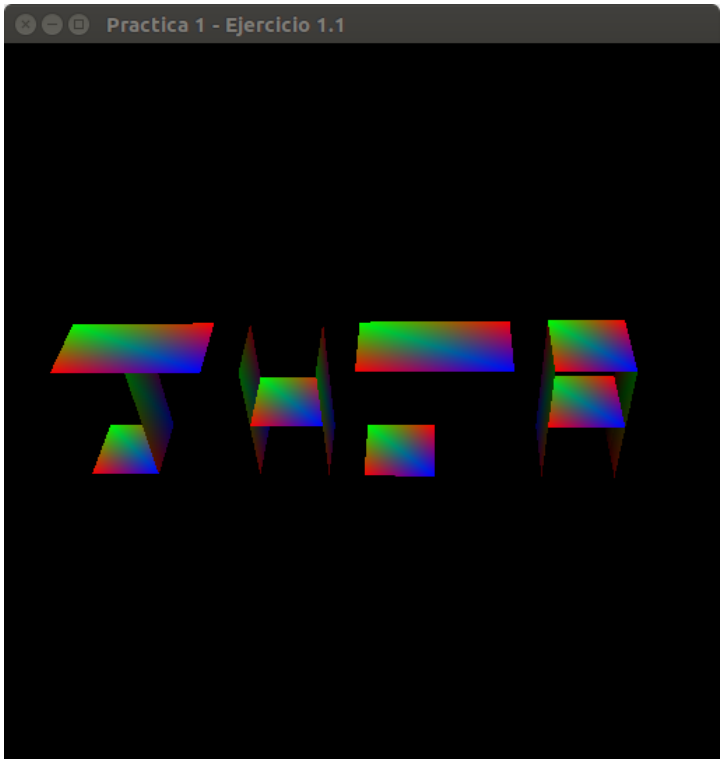
Para rotar la imagen se uso la misma funcion del ejercicio2 motionFunc para capturar la rotacion, luego simplemente se actualizaba la model matrix y normal matrix, para que se actualize su posicion en el espacio mundial, la actualizacion de la normal sirvepara la iluminacion, para eso usamos la transpuesta de la model matrix, todo eso se envia al vertex shader.
Al fragment shader se le envia el color de la luz y la direccion, para esto se uso las variable uniform , ya que con eso se puede enviar datos directamemente al fragment sin pasar por el vertex shader.

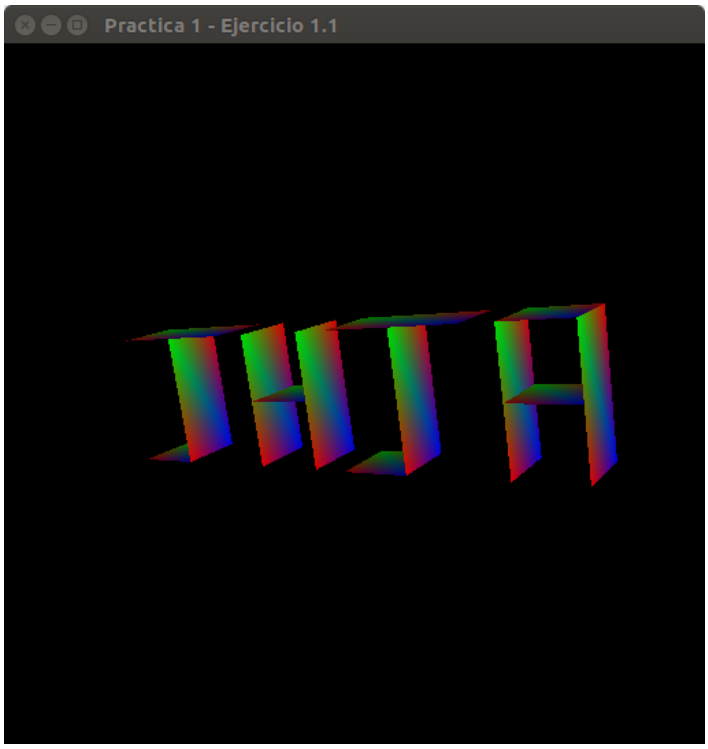
A diferencia del ejercion 1 , aqui lo que renderizamos son triangulos , por lo tanto para cada cuadrado necesitamos dos triangulos.

A continuacion mostramos imagenes usando el opengl 3.


Practica 1 - Ejercicio 1.1


Practica 1 - Ejercicio 1.1


Practica 1 - Ejercicio 1.1


Practica 1 - Ejercicio 1.1