Practica N°1

Ejercicio 1.1 Programación OpenGL en C

Intercambie, en el programa C que puede ser encontrado en el sitio de internet del curso (*Google Classroom*), el bloque con comandos `glVertex` con un bloque que genere las iniciales de su nombre con Quadtrips. Utilice además una fuente de luz direccional, así como características de material ambiental y difusa, para iluminar sus iniciales. Recuerde especificar las normales e inicializar los estados de OpenGL correctamente.
Además, piense en el efecto que tendría intercambiar los comandos `glRotate` y `glTranslate` en la función `drawgraphix`.

Procedimiento:

Es importante definir cada una de las letras en este caso la de nuestros nombres:

```
GLfloat LETRA_R[NUMVTR][3] = {{0.0f,0.0f,0.0f},{1.0f,0.0f,0.0f},
                                                {0.0f,6.0f,0.0f},{1.0f,5.0f,0.0f},
                                                {2.0f,5.0f,0.0f},{2.0f,6.0f,0.0f},
                                                {3.0f,4.0f,0.0f},{4.0f,4.0f,0.0f},
                                                {2.0f,3.0f,0.0f},{3.0f,3.0f,0.0f},
                                                {2.0f,3.0f,0.0f},{3.0f,2.0f,0.0f},
                                                {2.0f,3.0f,0.0f},{4.0f,0.0f,0.0f},
                                                {2.0f,3.0f,0.0f},{3.0f,0.0f,0.0f},
                                                {2.0f,3.0f,0.0f},{2.0f,2.0f,0.0f},
                                                {1.0f,3.0f,0.0f},{1.0f,2.0f,0.0f},
                                    };

GLfloat LETRA_V[NUMVTV][3] = { {0.0f,6.0f,0.0f},{1.0f,6.0f,0.0f},
                                                {3.0f,0.0f,0.0f},{3.0f,2.0f,0.0f},
                                                {4.0f,0.0f,0.0f},{4.0f,2.0f,0.0f},
                                                {6.0f,6.0f,0.0f},{7.0f,6.0f,0.0f},

                                               };

GLfloat COLOR[NUMVTR][3] = { {1,0,0},{1,0,0},
                                                {1,0,0},{1,0,0},
                                                {1,0,0},{1,0,0},
                                                {1,0,0},{1,0,0},
                                                {1,0,0},{1,0,0},
```

```
                                        {1,0,0},{1,0,0},
                                        {1,0,0},{1,0,0},
                                        {1,0,0},{1,0,0},
                                        {1,0,0},{1,0,0},
                                        {1,0,0},{1,0,0},
                                };
```

Luego se procede a realizar el respectivo offset

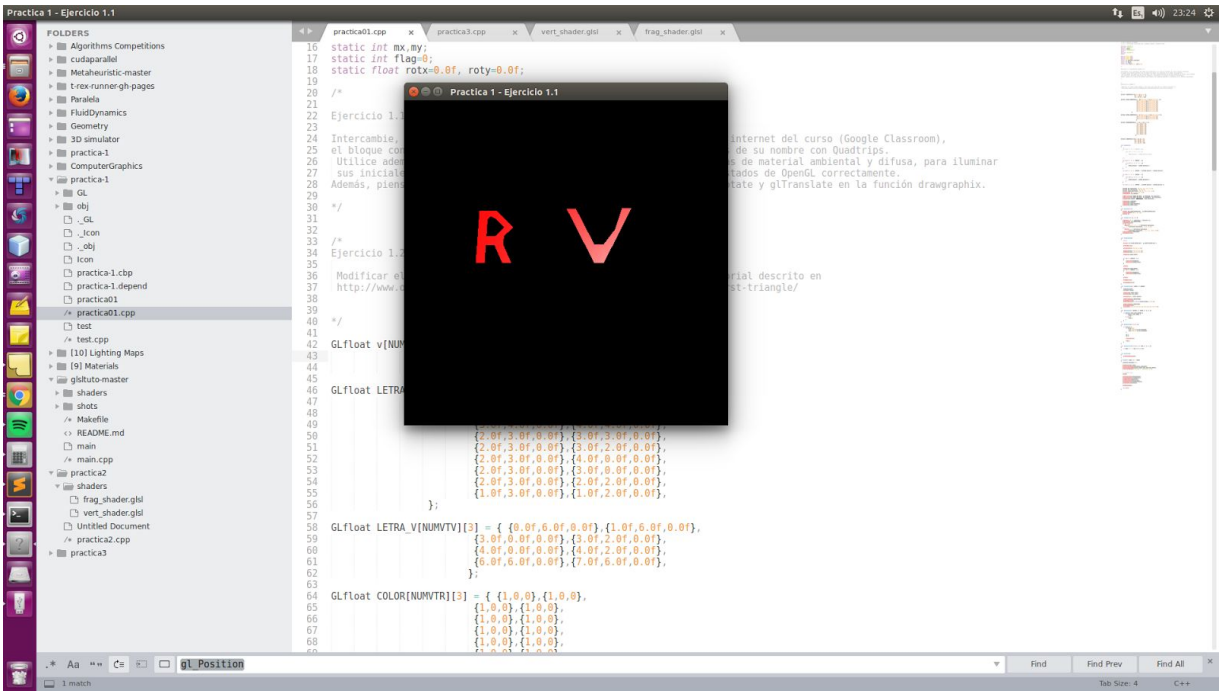
```
for (int i = 0; i < NUMVTV; ++i)
{
        for (int j = 0; j < 2; ++j)
        {
                LETRA_V[i][j] = (LETRA_V[i][j])/5;
        }
}

for (int i = 0; i < NUMVTV; ++i)LETRA_V[i][0] = (LETRA_V[i][0]);


for (int i = 0; i < NUMVTR; ++i)
{
        for (int j = 0; j < 2; ++j)
        {
                LETRA_R[i][j] = (LETRA_R[i][j])/5;
        }
}

for (int i = 0; i < NUMVTR; ++i)LETRA_R[i][0] = (LETRA_R[i][0]-2);
```

Resultado:



Ejercicio 1.2 OpenGL 3

Modificar el código usando OpenGL 3. Para esto usen como base el tutorial descrito en http://www.opengl-tutorial.org/beginners-tutorials/tutorial-2-the-first-triangle/

Luego de definir las letras se procede a hacer el respectivo offset

```
for(int i=0;i<=116;i++){
                firstLetter[i]=(firstLetter[i]-5)/10;
        }
        for(int i=0;i<=54;i++){
                SecondLetter[i]=(SecondLetter[i])/10;
        }
```

Se define los vertex buffer object:

```
glGenBuffers(2, VBOs);

	glGenVertexArrays(2, VAOs);

   glBindVertexArray(VAOs[0]);
	glBindBuffer(GL_ARRAY_BUFFER, VBOs[0]);
	glBufferData(GL_ARRAY_BUFFER, sizeof(firstLetter), firstLetter,
GL_STATIC_DRAW);
	glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(GLfloat), (GLvoid *)0);
	glEnableVertexAttribArray(0);

	glBindVertexArray(VAOs[1]);
	glBindBuffer(GL_ARRAY_BUFFER, VBOs[1]);
	glBufferData(GL_ARRAY_BUFFER, sizeof(SecondLetter), SecondLetter,
GL_STATIC_DRAW);
	glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(GLfloat), (GLvoid *)0);
	glEnableVertexAttribArray(0);
```

Se definen los shader:

Vertex_shader.glsl

```
#version 330 core

layout (location = 0) in vec3 aPos;

uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;

void main()

	//gl_Position = vec4(aPos.x, aPos.y, aPos.z, 1.0);
	gl_Position = projection * view * model * vec4(aPos, 1.0);
};
```

Frag_shader.glsl

```
#version 330 core
```

```
out vec4 FragColor;

void main()
{
	FragColor = vec4(1.0f, 0.5f, 0.2f, 1.0f);
};
```

Para luego hacer los respectivos ploteos:

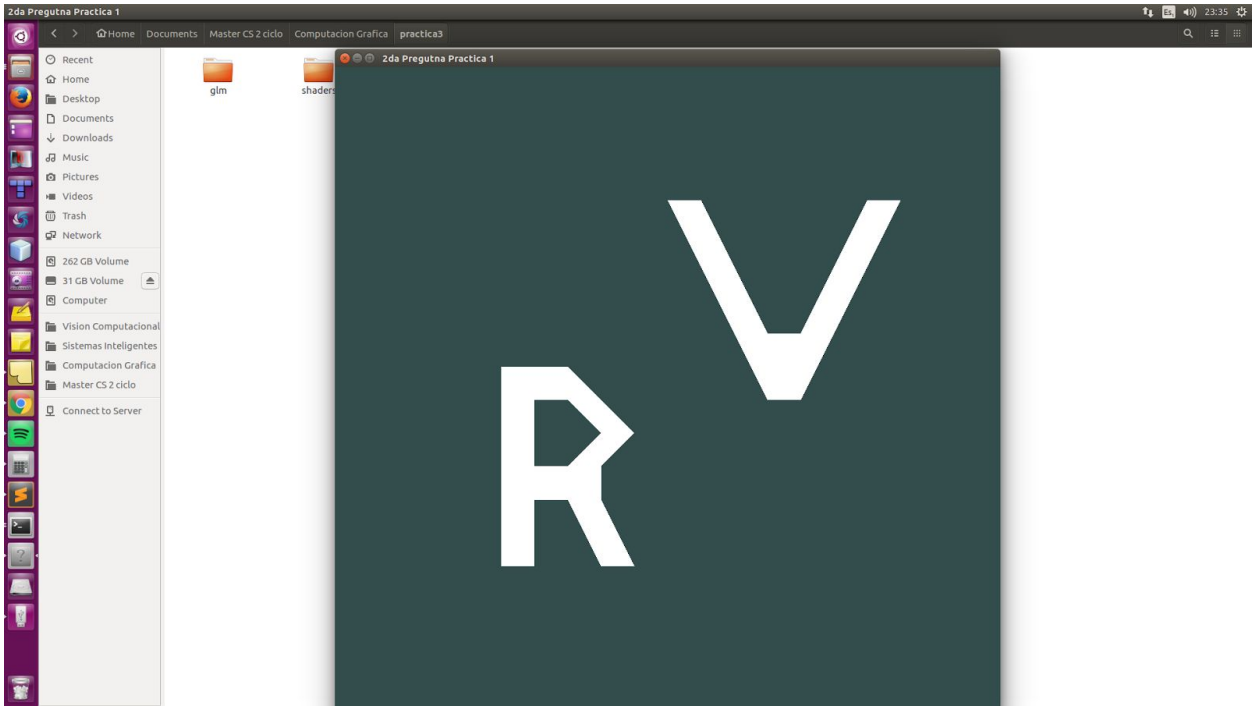
```
glUseProgram(prog_hdlr);

	glClearColor(0.2f, 0.3f, 0.3f, 1.0f);                                                    //
Set a specific color
	glClear(GL_COLOR_BUFFER_BIT);

	//glUseProgram(prog_hdlr);
	// draw first triangle using the data from the first VAO
	glBindVertexArray(VAOs[0]);
	glDrawArrays(GL_TRIANGLES, 0, 39);

	glBindVertexArray(VAOs[1]);
	glDrawArrays(GL_TRIANGLES, 0, 18);
```

Resultado:


2da Pregutna Practica 1
23:35
Home
Documents
Master CS 2 ciclo
Computacion Grafica
practica3
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Network
262 GB Volume
31 GB Volume
Computer
Vision Computacional
Sistemas Inteligentes
Computacion Grafica
Master CS 2 ciclo
Connect to Server
glm
shaders
2da Pregutna Practica 1