Informe de la práctica 1

Ejercicio 1.1

Este ejercicio tenía como propósito generar las iniciales “LLPA” usando Quadtrips usando el código proporcionado como base, que se muestra a continuación:

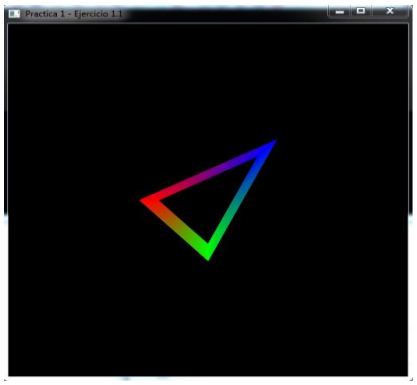


Imagen del triángulo formado por 3 planos

```
static int winwidth,winheight;
static int mx,my;
static int flag=0;
static float rotx=0.0f, roty=0.0f;

GLfloat v[NUMVTX][3]={{-1,0,1},{-1,0,-1},
                      {0,-1,1},{0,-1,-1},
                      {1,1,1},{1,1,-1},
                      {-1,0,1},{-1,0,-1}};

GLfloat c[NUMVTX][3]={{1,0,0},{1,0,0},
                      {0,1,0},{0,1,0},
                      {0,0,1},{0,0,1},
                      {1,0,0},{1,0,0}};
```

Imagen de las variables y los vértices

Que al realizar los cambios en los vértices queda de la siguiente forma:

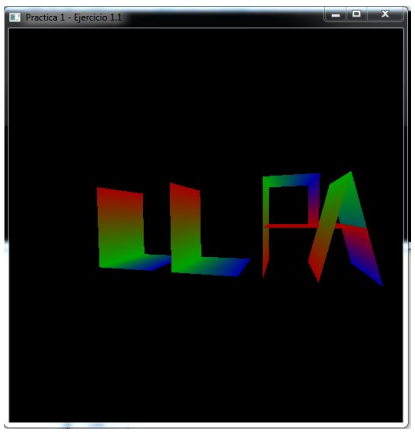


Imagen de las iniciales LLPA

```
static int winwidth,winheight;
static int mx,my;
static int flag=0;
static float rotx=0.0f, roty=0.0f;


//Leticia Leonor

GLfloat l[24][3]={  {0,1,1},{0,1,-1},
                    {0,-1,1},{0,-1,-1},
                    {1.5,-1,1},{1.5,-1,-1}

                    };
// P en 2D
GLfloat p[10][3]={  {0,-1,1},{0,-1,-1},
                    {0,1,1},{0,1,-1},
                    {1,1,1},{1,1,-1},
                    {1,0,1},{1,0,-1},
                    {0,0,1},{0,0,-1}
                    };
//Alva
GLfloat a[10][3]={  {-0.5,-1,1},{-0.5,-1,-1},
                    {0,1,1},{0,1,-1},
                    {0.5,-1,1},{0.5,-1,-1},
                    {0.25,0,1},{0.25,0,-1},
                    {-0.25,0,1},{-0.25,0,-1}
                    };
```

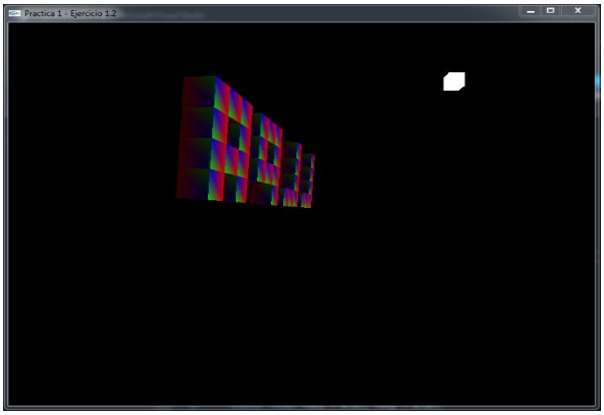
Vértices para formar las iniciales LLPA

Para escribir las iniciales se tuvo que entender que los Quadtrips van generando cuadriláteros que serán nuestros planos en el espacio 3D, para poder posicionar las variables juntas y no sobrepuestas se procedió a desplazar sobre el eje x, usando la función drawVertex (-2.5f, 6, l, c) que me permite saber que mi nueva posición inicial será en -2,5f con 6 vértices la matriz l (inicial L), la matriz c (matriz del color). Cabe resaltar que la función glTranslatef (0.0f, 0.0f, -3.0f); nos ayuda a generar la traslación sobre el eje Z negativo, rotando x sobre el eje Y glRotatef (rotx,0.0f,1.0f,0.0f) y rotando sobre el eje X glRotatef(roty,1.0f,0.0f,0.0f); .

```
glEnable(GL_LIGHTING);

const GLfloat light_ambient[] = { 0.0f,0.0f,0.0f,1.0f};
const GLfloat light_diffuse[] = { 1.0f,1.0f,1.0f,1.0f}; //color 1 0 0
const GLfloat light_specular[] = { 1.0f,1.0f,1.0f,1.0f};
const GLfloat light_position[] = { 5.0f,0.0f,5.0f,0.0f};

//
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
//glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);

glEnable(GL_LIGHT0);
glEnable(GL_COLOR_MATERIAL);

const GLfloat mat_ambient[] = { 0.0f,0.0f,0.0f,1.0f};
const GLfloat mat_diffuse[] = { 0.0f,1.0f,0.0f,1.0f};
const GLfloat mat_specular[] = { 1.0f,1.0f,1.0f,1.0f};
const GLfloat mat_shininess[] = { 10.0f};

glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, mat_ambient);
glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, mat_diffuse);
//glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS, mat_shininess);
```

Ejercicio 2.1

Al igual que en el ejercicio 1.1 se pide dibujar las iniciales del nombre (LLPA)



```
void AddCube(float x, float y)
{
    float s=0.2f;
    x = x * s * 2;
    y = y * s * 2;
    AddElements({
        -s + x, -s + y, -s,
        s + x, -s + y, -s,
        s + x,  s + y, -s,
        -s + x, s + y, -s,
    }, {}, new glm::vec3(0.0f, 0.0f, -1.0f));

    AddElements({
        -s + x, -s + y,  s,
        s + x, -s + y,  s,
        s + x,  s + y,  s,
        -s + x, s + y,  s,
    }, {}, new glm::vec3(0.0f, 0.0f, 1.0f));


    AddElements({
        -s + x,  s + y,  s,
        -s + x,  s + y, -s,
        -s + x, -s + y, -s,
        -s + x, -s + y,  s,
    }, {}, new glm::vec3(-1.0f, 0.0f, 0.0f));

    AddElements({
        s + x,  s + y,  s,
        s + x,  s + y, -s,
        s + x, -s + y, -s,
        s + x, -s + y,  s,
    }, {}, new glm::vec3(1.0f, 0.0f, 0.0f));

    AddElements({
        -s + x, -s + y, -s,
        s + x, -s + y, -s,
        s + x, -s + y,  s,
        -s + x, -s + y, s,
    }, {}, new glm::vec3(0.0f, -1.0f, 0.0f));

    AddElements({
        -s + x, s + y, -s,
        s + x,  s + y, -s,
        s + x,  s + y,  s,
        -s + x, s + y, s,
    }, {}, new glm::vec3(0.0f, 1.0f, 0.0f));

}
```

```
mName->AddCube(0.0f, 4.0f);
mName->AddCube(0.0f, 3.0f);
mName->AddCube(0.0f, 2.0f);
mName->AddCube(0.0f, 1.0f);
mName->AddCube(1.0f, 1.0f);
mName->AddCube(2.0f, 1.0f);

mName->AddCube(4.0f, 4.0f);
mName->AddCube(4.0f, 3.0f);
mName->AddCube(4.0f, 2.0f);
mName->AddCube(4.0f, 1.0f);
mName->AddCube(5.0f, 1.0f);
mName->AddCube(6.0f, 1.0f);

mName->AddCube(8.0f, 4.0f);
mName->AddCube(9.0f, 4.0f);
mName->AddCube(10.0f, 4.0f);
mName->AddCube(8.0f, 3.0f);
mName->AddCube(10.0f, 3.0f);
mName->AddCube(8.0f, 2.0f);
mName->AddCube(9.0f, 2.0f);
mName->AddCube(10.0f, 2.0f);
mName->AddCube(8.0f, 1.0f);

mName->AddCube(12.0f, 4.0f);
mName->AddCube(13.0f, 4.0f);
mName->AddCube(14.0f, 4.0f);
mName->AddCube(12.0f, 3.0f);
mName->AddCube(14.0f, 3.0f);
mName->AddCube(12.0f, 2.0f);
mName->AddCube(13.0f, 2.0f);
mName->AddCube(14.0f, 2.0f);
mName->AddCube(12.0f, 1.0f);
mName->AddCube(14.0f, 1.0f);
```

Imágenes que muestran los movimientos de rotación y traslación para una fuente de luz fija

Vértices para generar las iniciales usando cubos para poder armar las iniciales

Uno de los puntos más notorios de la diferencia entre opengl antiguo y el opengl moderno es el uso de shaders, para el caso de iluminación en opengl moderno hacemos uso de 2 shaders, uno simple para pintar nuestra lámpara (source light) que será representado por un cubo blanco y un shader denominado lighting para el proceso de iluminación que aplicará el lightcolor contra el objectcolor. A continuación, mostramos el LoadShader que se encarga de cargar los shaders.

```
void LoadShader()
{
    //
    viewposition= new glm::vec3(0.0f, 0.0f, -10.0f);
    view = glm::translate(view, *viewposition);
    projection = glm::perspective(glm::radians(60.0f), (float)swidth / (float)sheight, 0.1f, 100.0f);

    //Lampara de luz
    glm::mat4* modelLamp = new glm::mat4(1.0f);
    *modelLamp=glm::scale(*modelLamp, glm::vec3(0.2f, 0.2f, 0.2f));

    lightposition = new glm::vec3(20.0f, 20.0f, 0.0f);
    *modelLamp = glm::translate(*modelLamp, *lightposition);

    mLamp = new VertexPositions("res/shaders/Basic.shader",modelLamp,&view,&projection);
    cuboLampara();
    mLamp->ConstructBuffers();

    //objeto
    glm::vec3* lighColor = new glm::vec3(1.0f, 1.0f, 1.0f);
    mName = new VertexPositions("res/shaders/Lighting.shader", new glm::mat4(1.0f), &view, &projection, lighColor, lightposition, viewposition);
    cubo();
    mName->ConstructBuffers();

}
```