**Universidad Nacional de Ingeniería**
**Facultad de Ciencias**
**Escuela Profesional de Ciencias de la Computación**

**Ciclo 2020-2**

**Examen Parcial de Computación Gráfica CC431**

***Walter Jesús Felipe Tolentino - 20172714F***

# 1. Preparando el juego

## 1.1. Variables y modelos

Variables de posicionamiento de los objetos, ángulos de rotación, pasos de traslación, incrementos de rotación y traslación, identificadores de texturas, vao, vbo, identificador de objetos, booleanos de estado, matrices *glm* (pMat, vMat, mMat y mvMat) y una pila para la dependencia de movimientos de las partes del cuerpo que tiene el personaje y por último enteros para las dimensiones de la pantalla y para la entrada de datos por teclado.

Además se importa los modelos *obj* para las rocas, la casa y una escena de montaña que se añadirá para la siguiente presentación.

```
float cameraX, cameraY, cameraZ, RockLocX, RockLocY, RockLocZ, HouseLocX, HouseLocY, HouseLocZ
    , FlatLocX, FlatLocY, FlatLocZ, SkyLocX, SkyLocY, SkyLocZ, aspect, mountainLocX,
    mountainLocY, mountainLocZ, humanPosZ, humanPosY, humanPosX, closeto, angleCamera,
    angleCameraInc, step, incRotLeg1, incRotLeg2, rotLeg1, rotLeg2, rotLeg3, incRotLeg3,
    rotArm, incRotArm, rotSky, incRotSky, rotBodyHuman, incRotBodyHuman, incJumping;

GLuint renderingProgram, vao[numVAOs], vbo[numVBOs], mvLoc, projLoc, obj, rockTexture,
    houseTexture, flatTexture,houseNavTexture, mountainTexture, skyTexture;

int width, height, keyboard, actionKeyboard;

bool walkingBool,jumpingBool;

glm::mat4 pMat, vMat, mMat, mvMat;
glm::vec4 posLeftLeg, posRightLeg, posTrunk, posHead, posLeftArm, posRightArm;

stack<glm::mat4> mvStack;

// models and instance
ImportedModel Rock("../models/Rock_big_single_b_LOD0.obj");
ImportedModel House("../models/house.obj");
ImportedModel Mountain("../models/mountain/Mountain.obj");
Sphere mySphere = Sphere(48);
```

Listing 1: main.cpp :: *Variables y modelos*

## 1.2. *init(.)*

Configuración de las condiciones iniciales del juego, de los vertices y la carga de texturas para el plano, las rocas, la casa y el cielo esférico.

```
void init(GLFWwindow* window) {
    renderingProgram = createShaderProgram("shaders/vs.glsl", "shaders/fs.glsl");
    cameraX = 0.0f; cameraY = 3.0f; cameraZ = 100.0f;
    angleCamera = 0.5f; angleCameraInc=0.001f;

    RockLocX = -300.0f; RockLocY = 0.5f; RockLocZ = 0.0f;
    HouseLocX = 0.0f; HouseLocY = 33.0f; HouseLocZ = -200.0f;
    FlatLocX = 0.0f; FlatLocY = 0.0f; FlatLocZ = 0.0f;
    mountainLocX = 0.0f; mountainLocY = 0.0f; mountainLocZ = 0.0f;
    SkyLocX = 0.0f;SkyLocY = 0.0f;SkyLocZ = 0.0f;
    humanPosZ = 300.0;humanPosX = 0.0;humanPosY = 40.0;

    step = 0.4f; closeto = 3.0f;rotLeg1=0.0;rotLeg2=0.0;
    incRotLeg1=0.05f/2.0f;incRotLeg2=2.0*(0.05f/2.0f);
    incRotArm=0.05f/2.0f; incRotLeg3 = 0.0; rotLeg3=0.0;
    rotBodyHuman = 0.0f; incRotBodyHuman = 2.0*(0.05f/2.0f);
    rotSky = 0.0f; incRotSky = (float)M_PI/5000.0f;incJumping = 0.5f;

```

```
    jumpingBool = false;walkingBool=false;

    glfwGetFramebufferSize(window, &width, &height);
    aspect = (float)width / (float)height;
    pMat = glm::perspective(1.3472f, aspect, 0.1f, 1000000.0f);
    pMat = glm::rotate(pMat, angleCamera ,glm::vec3(1.0,0.0,0.0));

    setupVertices();

    loadTexture("../textures/Rock_big_single_b_diffuse_desert.jpg", rockTexture);
    loadTexture("../textures/house/house_diffuse.jpg", houseTexture);
    loadTexture("../textures/Rock_big_single_b_sandstone_flat.jpg", flatTexture);
    loadTexture("../textures/mountain/Color.png", mountainTexture);
    loadTexture("../textures/sky/sky4.jpg", skyTexture);
}
```

Listing 2: main.cpp :: *init(.)*

### 1.3. *display(.)*

La cámara acompaña al personaje a una cierta distancia, esto se logra usando la matriz *lookAt* de *glm*.

```
void display(GLFWwindow* window, double currentTime) {
    glClear(GL_DEPTH_BUFFER_BIT);
    glClearColor(0.0,0.0,0.0,0.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glUseProgram(renderingProgram);

    mvLoc = glGetUniformLocation(renderingProgram, "mv_matrix");
    projLoc = glGetUniformLocation(renderingProgram, "proj_matrix");
    obj = glGetUniformLocation(renderingProgram, "obj");


    vMat = glm::lookAt(glm::vec3(cameraX, cameraY, cameraZ)+glm::vec3(humanPosX, humanPosY,
    humanPosZ),
                        glm::vec3(humanPosX, humanPosY,humanPosZ),
                        glm::vec3(0.0f, 10.0f, 0.0f));
    glUniformMatrix4fv(projLoc, 1, GL_FALSE, glm::value_ptr(pMat));

    sceneDessert();
    humanAdvancedAnimation();
}
```

Listing 3: main.cpp :: *display(.)*

## 2. Construcción de la escena

La escena principal *sceneDessert(.)* esta conformado por un plano con un textura de tierra, rocas distribuidas por todo el plano con *gl_InstanceID* y una casa para la almacén del tesoro (que se va distribuir aleatoriamente) que encuentre el personaje interactuando por toda la escena.

Toda esta escena está dentro de una esfera (*Sphere.cpp*) rotando un pequeño ángulo por *frame*, con un textura de cielo nocturno con estrellas, que le brinda un detalle más realista.

```
void sceneDessert(void){
    //rocks
    mMat = glm::translate(glm::mat4(1.0f), glm::vec3(RockLocX, RockLocY, RockLocZ));
    mvMat = vMat * mMat;
    glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvMat));
    glUniform1i(obj,1);
    glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
    glEnableVertexAttribArray(0);
    glBindBuffer(GL_ARRAY_BUFFER, vbo[1]);
    glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 0, 0);
    glEnableVertexAttribArray(1);
    glActiveTexture(GL_TEXTURE0);
    glBindTexture(GL_TEXTURE_2D, rockTexture);
    glEnable(GL_DEPTH_TEST);
    glDepthFunc(GL_LEQUAL);
    glDrawArraysInstanced(GL_TRIANGLES, 0, Rock.getNumVertices(), 4);

    //House
    mMat = glm::translate(glm::mat4(1.0f), glm::vec3(HouseLocX, HouseLocY+30.0, HouseLocZ));
```

```
    mMat = glm::rotate(mMat, 2.5f, glm::vec3(0.0,1.0,0.0));
    mMat = glm::scale(mMat, glm::vec3(2.0,2.0,2.0));
    mvMat = vMat * mMat;
    glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvMat));
    glUniform1i(obj,2);
    glBindBuffer(GL_ARRAY_BUFFER, vbo[3]);
    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
    glEnableVertexAttribArray(0);
    glBindBuffer(GL_ARRAY_BUFFER, vbo[4]);
    glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 0, 0);
    glEnableVertexAttribArray(1);
    glActiveTexture(GL_TEXTURE0);
    glBindTexture(GL_TEXTURE_2D, houseTexture);
    glDrawArrays(GL_TRIANGLES, 0, House.getNumVertices());

    //Flat
    mMat = glm::translate(glm::mat4(1.0f), glm::vec3(FlatLocX, FlatLocY, FlatLocZ));
    mvMat = vMat * mMat;
    glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvMat));
    glUniform1i(obj,3);
    glBindBuffer(GL_ARRAY_BUFFER, vbo[6]);
    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
    glEnableVertexAttribArray(0);
    glBindBuffer(GL_ARRAY_BUFFER, vbo[7]);
    glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 0, 0);
    glEnableVertexAttribArray(1);
    glActiveTexture(GL_TEXTURE0);
    glBindTexture(GL_TEXTURE_2D, flatTexture);
    glDrawArrays(GL_TRIANGLES, 0, 6);

    //Sky
    rotSky += incRotSky;
    mMat = glm::translate(glm::mat4(1.0f), glm::vec3(SkyLocX, SkyLocY, SkyLocZ));
    mMat = glm::rotate(mMat, rotSky, glm::vec3(0.0,1.0,0.0));
    mMat = glm::scale(mMat, glm::vec3(10000.0,10000.0,10000.0));
    mvMat = vMat * mMat;
    glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvMat));
    glBindBuffer(GL_ARRAY_BUFFER, vbo[15]);
    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
    glEnableVertexAttribArray(0);
    glBindBuffer(GL_ARRAY_BUFFER, vbo[16]);
    glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 0, 0);
    glEnableVertexAttribArray(1);
    glActiveTexture(GL_TEXTURE0);
    glBindTexture(GL_TEXTURE_2D,skyTexture);
    glDrawArrays(GL_TRIANGLES, 0, mySphere.getNumIndices());
}
```

Listing 4: main.cpp :: *sceneDessert(.)*

## 3. Construcción y animación del personaje

El personaje está construido con paralelepipedos y presenta las animaciones para caminar, girar sobre su eje $Y$ y saltar.

```
// prototypes
void humanAdvancedAnimation(void);
void walking(int direction);
void turning(int direction)
void jumping(void);
```

Listing 5: main.cpp :: *Prototipos para la animación*

La implementación de cada uno de los métodos de animación toma como referencia el vídeo OpenGL Walking Animation y nociones intuitivas del movimiento de una persona.

```
void walking(int direction){
    walkingBool=true;

    if(direction == 1){//hacia adelante
        humanPosZ -= step*cos(rotBodyHuman);
        humanPosX -= step*sin(rotBodyHuman);
    }else if(direction == 2){//hacia atras
        humanPosZ += step*cos(rotBodyHuman);
        humanPosX += step*sin(rotBodyHuman);
```

```
    }

    if (rotArm > M_PI/8.0){
        incRotArm = -0.05/2.0;
    }
    else if (rotArm < -M_PI/8.0){
        incRotArm = 0.05/2.0;
    }
    rotArm += incRotArm;


    if (rotLeg1 > M_PI/8.0){
        incRotLeg1 = -0.05/2.0;
    }
    else if (rotLeg1 < -M_PI/8.0){
        incRotLeg1 = 0.05/2.0;
    }
    rotLeg1 += incRotLeg1;


    if(rotLeg2 > M_PI/4.0f){ //Rotacion de canilla derecha inicialmente rotado PI/4
        incRotLeg2 = 0.0;
    }
    if(rotLeg1 < -M_PI/8.0){ //comienza cuando el muslo esta en -M_PI/8
        incRotLeg2 = -2.0*(0.05/2.0);
    }
    if(rotLeg2 < 0.0){
        incRotLeg2 = 2.0*(0.05/2.0);
    }
    rotLeg2 += incRotLeg2;


    if(rotLeg3 > M_PI/4.0f){//Rotacion de canilla izquierda
        incRotLeg3 = -2.0*(0.05/2.0);
    }
    if(rotLeg1 > M_PI/8.0){ //Rotacion de canilla izquierda comienza cuando el muslo esta en -
    M_PI/8
        incRotLeg3 = 2.0*(0.05/2.0);
    }
    if(rotLeg1 < -M_PI/8.0){
        incRotLeg3 = 0.0;
    }
    rotLeg3 += incRotLeg3;
}

void turning(int direction){
    if(direction == 1){
        rotBodyHuman -= incRotBodyHuman;
    }else if(direction == 2){
        rotBodyHuman += incRotBodyHuman;
    }
}

void jumping(void){
    humanPosY += incJumping;
    if (humanPosY > 48.0){ // 8.0f de salto
        incJumping = -0.5;
    }
    else if (humanPosY == 40.0){
        jumpingBool = false;
        incJumping = 0.5;
    }
}
```

Listing 6: main.cpp :: *Implementación de la animación del personaje*

La interacción con el usuario viene dado por las teclas T(adelante), G(atras), F(giro izquierda), H(giro derecha) y Space (salto), estás estan controladas dentro del método *humanAdvancedAnimation();*

```
void humanAdvancedAnimation(void){

    if (keyboard == 84 && (actionKeyboard==GLFW_PRESS || actionKeyboard == GLFW_REPEAT)){ //T
        walking(1);
    }else if (keyboard == 71 && (actionKeyboard==GLFW_PRESS || actionKeyboard == GLFW_REPEAT))
    { //G
        walking(2);
    }
```

```
8
9       if (keyboard == GLFW_KEY_SPACE && actionKeyboard==GLFW_PRESS ){ //space
10          jumpingBool = true;
11      }
12
13      if(jumpingBool == true){
14          jumping();
15      }
16
17      if (keyboard == 72 && (actionKeyboard==GLFW_PRESS || actionKeyboard == GLFW_REPEAT)){ //H
18          turning(1);
19      }else if (keyboard == 70 && (actionKeyboard==GLFW_PRESS || actionKeyboard == GLFW_REPEAT))
        { //F
20          turning(2);
21      }
22      ...
23  }
```

Listing 7: main.cpp :: *Control del movimiento por teclado*

Y como lo había mencionado, el personaje está construido usando paralelepípedos para la cabeza, cuello, tronco superior, tronco inferior, brazos, antebrazos, manos, muslos, canillas y pies. Esto escalando convenientemente un cubo posicionado en el $(0, 0, 0)$ y de lado unitario cuya Data está en el vbo[11].

```
1   void humanAdvancedAnimation(void){
2       ...
3
4       //Head
5       mvStack.push(vMat);
6       glUniform1i(obj,43);
7       mvStack.push(mvStack.top());
8       mvStack.top() *= glm::translate(glm::mat4(1.0f), glm::vec3(humanPosX,humanPosY,humanPosZ))
        ;
9       mvStack.top() *= glm::rotate(glm::mat4(1.0f), (float)M_PI,glm::vec3(0.0,1.0,0.0));//mirar
        hacia adelante
10      mvStack.top() *= glm::rotate(glm::mat4(1.0f), rotBodyHuman  ,glm::vec3(0.0,1.0,0.0));//
        rotacion
11      mvStack.push(mvStack.top());
12      mvStack.top() *= glm::scale(glm::mat4(1.0f), glm::vec3(2.0,2.25,2.0));
13      glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvStack.top()));
14      glBindBuffer(GL_ARRAY_BUFFER, vbo[11]);
15      glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
16      glEnableVertexAttribArray(0);
17      glDrawArrays(GL_TRIANGLES, 0, 36);
18      mvStack.pop();
19
20      //Neck
21      glUniform1i(obj,43);
22      mvStack.push(mvStack.top());
23      mvStack.top() *= glm::translate(glm::mat4(1.0f), glm::vec3(0.0,-(2.25 + 1.0),0.0));
24      mvStack.push(mvStack.top());
25      mvStack.top() *= glm::scale(glm::mat4(1.0f), glm::vec3(1.0,1.0,1.0));
26      glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvStack.top()));
27      glBindBuffer(GL_ARRAY_BUFFER, vbo[11]);
28      glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
29      glEnableVertexAttribArray(0);
30      glDrawArrays(GL_TRIANGLES, 0, 36);
31      mvStack.pop();
32
33      //Trunk top
34      glUniform1i(obj,42);
35      mvStack.push(mvStack.top());
36      mvStack.top() *= glm::translate(glm::mat4(1.0f), glm::vec3(0.0,-(1.00 + 5.0),0.0));
37      mvStack.push(mvStack.top());
38      mvStack.top() *= glm::scale(glm::mat4(1.0f), glm::vec3(4.0,5.0,2.0));
39      glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvStack.top()));
40      glBindBuffer(GL_ARRAY_BUFFER, vbo[11]);
41      glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
42      glEnableVertexAttribArray(0);
43      glDrawArrays(GL_TRIANGLES, 0, 36);
44      mvStack.pop();
45
46      //ARMS
47          // Right
48                  //Arm top
49                  glUniform1i(obj,42);
50                  mvStack.push(mvStack.top());
```

```
51                mvStack.top() *= glm::translate(glm::mat4(1.0f), glm::vec3(4.0 + 1.0
    ,5.0-3.5+4.5,0.0));
52                mvStack.top() *= glm::rotate(glm::mat4(1.0f), (float)M_PI/50.0f ,glm::vec3
    (0.0,0.0,1.0));
53                mvStack.top() *= glm::rotate(glm::mat4(1.0f), rotArm , glm::vec3(1.0,0.0,0.0))
    ;
54                mvStack.push(mvStack.top());
55                mvStack.top()  *= glm::translate(glm::mat4(1.0f), glm::vec3(0.0,-4.5,0.0));
56                mvStack.top() *= glm::scale(glm::mat4(1.0f), glm::vec3(1.0,3.5,1.0));
57                glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvStack.top()));
58                glBindBuffer(GL_ARRAY_BUFFER, vbo[11]);
59                glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
60                glEnableVertexAttribArray(0);
61                glDrawArrays(GL_TRIANGLES, 0, 36);
62                mvStack.pop();
63
64                //Forearm
65                glUniform1i(obj,43);
66                mvStack.push(mvStack.top());
67                mvStack.top() *= glm::translate(glm::mat4(1.0f), glm::vec3(0.0,-2.5-4.5,0.0));
68                mvStack.top() *= glm::rotate(glm::mat4(1.0f), -(float)M_PI/10.0f, glm::vec3
    (1.0,0.0,0.0));
69                mvStack.push(mvStack.top());
70                mvStack.top()  *= glm::translate(glm::mat4(1.0f), glm::vec3(0.0,-4.0,0.0));
71                mvStack.top()  *= glm::scale(glm::mat4(1.0f), glm::vec3(1.0,3.0,1.0));
72                glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvStack.top()));
73                glBindBuffer(GL_ARRAY_BUFFER, vbo[11]);
74                glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
75                glEnableVertexAttribArray(0);
76                glDrawArrays(GL_TRIANGLES, 0, 36);
77                mvStack.pop();
78
79                //hand
80                glUniform1i(obj,42);
81                mvStack.push(mvStack.top());
82                mvStack.top() *= glm::translate(glm::mat4(1.0f), glm::vec3(0.0,-(7.0 + 1.5)
    ,0.0));
83                mvStack.top() *= glm::scale(glm::mat4(1.0f), glm::vec3(1.0,1.5,1.0));
84                glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvStack.top()));
85                glBindBuffer(GL_ARRAY_BUFFER, vbo[11]);
86                glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
87                glEnableVertexAttribArray(0);
88                glDrawArrays(GL_TRIANGLES, 0, 36);
89                mvStack.pop();
90                mvStack.pop();
91                mvStack.pop();
92
93            // Left
94                //Arm top
95                glUniform1i(obj,42);
96                mvStack.push(mvStack.top());
97                mvStack.top() *= glm::translate(glm::mat4(1.0f), glm::vec3(-4.0 -
    1.0,5.0-3.5+4.5,0.0));
98                mvStack.top() *= glm::rotate(glm::mat4(1.0f), -(float)M_PI/50.0f ,glm::vec3
    (0.0,0.0,1.0));
99                mvStack.top() *= glm::rotate(glm::mat4(1.0f), -rotArm , glm::vec3(1.0,0.0,0.0)
    );
100               mvStack.push(mvStack.top());
101               mvStack.top()  *= glm::translate(glm::mat4(1.0f), glm::vec3(0.0,-4.5,0.0));
102               mvStack.top() *= glm::scale(glm::mat4(1.0f), glm::vec3(1.0,3.5,1.0));
103               glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvStack.top()));
104               glBindBuffer(GL_ARRAY_BUFFER, vbo[11]);
105               glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
106               glEnableVertexAttribArray(0);
107               glDrawArrays(GL_TRIANGLES, 0, 36);
108               mvStack.pop();
109
110               //Forearm
111               glUniform1i(obj,43);
112               mvStack.push(mvStack.top());
113               mvStack.top() *= glm::translate(glm::mat4(1.0f), glm::vec3(0.0,-2.5-4.5,0.0));
114               mvStack.top() *= glm::rotate(glm::mat4(1.0f), -(float)M_PI/10.0f, glm::vec3
    (1.0,0.0,0.0));
115               mvStack.push(mvStack.top());
116               mvStack.top()  *= glm::translate(glm::mat4(1.0f), glm::vec3(0.0,-4.0,0.0));
117               mvStack.top()  *= glm::scale(glm::mat4(1.0f), glm::vec3(1.0,3.0,1.0));
```

```
118                 glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvStack.top()));
119                 glBindBuffer(GL_ARRAY_BUFFER, vbo[11]);
120                 glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
121                 glEnableVertexAttribArray(0);
122                 glDrawArrays(GL_TRIANGLES, 0, 36);
123                 mvStack.pop();
124 
125                 //hand
126                 glUniform1i(obj,42);
127                 mvStack.push(mvStack.top());
128                 mvStack.top() *= glm::translate(glm::mat4(1.0f), glm::vec3(0.0,-(7.0 + 1.5)
    ,0.0));
129                 mvStack.top() *= glm::scale(glm::mat4(1.0f), glm::vec3(1.0,1.5,1.0));
130                 glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvStack.top()));
131                 glBindBuffer(GL_ARRAY_BUFFER, vbo[11]);
132                 glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
133                 glEnableVertexAttribArray(0);
134                 glDrawArrays(GL_TRIANGLES, 0, 36);
135                 mvStack.pop();
136                 mvStack.pop();
137                 mvStack.pop();
138 
139     //Trunk down
140     glUniform1i(obj,44);
141     mvStack.push(mvStack.top());
142     mvStack.top() *= glm::translate(glm::mat4(1.0f), glm::vec3(0.0,-(5.0 + 1.5),0.0));
143     mvStack.push(mvStack.top());
144     mvStack.top() *= glm::scale(glm::mat4(1.0f), glm::vec3(4.0,1.5,2.0));
145     glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvStack.top()));
146     glBindBuffer(GL_ARRAY_BUFFER, vbo[11]);
147     glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
148     glEnableVertexAttribArray(0);
149     glDrawArrays(GL_TRIANGLES, 0, 36);
150     mvStack.pop();
151 
152     //LEGS
153             //RIGHT
154                 glUniform1i(obj,44);
155                 mvStack.push(mvStack.top());
156                 mvStack.top() *= glm::translate(glm::mat4(1.0f), glm::vec3(2.5,-(1.5 + 4.5) +
    5.5,0.0));
157                 mvStack.top() *= glm::rotate(glm::mat4(1.0f), -rotLeg1 , glm::vec3
    (1.0,0.0,0.0));
158                 mvStack.push(mvStack.top());
159                 mvStack.top() *= glm::translate(glm::mat4(1.0f),glm::vec3(0.0,-5.5,0.0));
160                 mvStack.top() *= glm::scale(glm::mat4(1.0f), glm::vec3(1.5,4.5,1.75));
161                 glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvStack.top()));
162                 glBindBuffer(GL_ARRAY_BUFFER, vbo[11]);
163                 glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
164                 glEnableVertexAttribArray(0);
165                 glDrawArrays(GL_TRIANGLES, 0, 36);
166                 mvStack.pop();
167 
168                 glUniform1i(obj,43);
169                 mvStack.push(mvStack.top());
170                 mvStack.top() *= glm::translate(glm::mat4(1.0f), glm::vec3(0.0,-(4.5 + 4.5 +
    5.5)+5.5,0.0));
171                 mvStack.top() *= glm::rotate(glm::mat4(1.0f), (float)M_PI/4.0f, glm::vec3
    (1.0,0.0,0.0));
172                 mvStack.top() *= glm::rotate(glm::mat4(1.0f), -rotLeg2, glm::vec3(1.0,0.0,0.0)
    );
173                 mvStack.push(mvStack.top());
174                 mvStack.top() *= glm::translate(glm::mat4(1.0f),glm::vec3(0.0,-5.5,0.0));
175                 mvStack.top() *= glm::scale(glm::mat4(1.0f), glm::vec3(1.5,4.5,1.75));
176                 glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvStack.top()));
177                 glBindBuffer(GL_ARRAY_BUFFER, vbo[11]);
178                 glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
179                 glEnableVertexAttribArray(0);
180                 glDrawArrays(GL_TRIANGLES, 0, 36);
181                 mvStack.pop();
182                 //shoes
183                 glUniform1i(obj,42);
184                 mvStack.push(mvStack.top());
185                 mvStack.top() *= glm::translate(glm::mat4(1.0f), glm::vec3(0.0,-(4.5+1.0+5.5)
    ,1.0));
186                 mvStack.top() *= glm::scale(glm::mat4(1.0f), glm::vec3(1.5,1.0,3.0));
```

```
                glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvStack.top()));
                glBindBuffer(GL_ARRAY_BUFFER, vbo[11]);
                glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
                glEnableVertexAttribArray(0);
                glDrawArrays(GL_TRIANGLES, 0, 36);
                mvStack.pop();
                mvStack.pop();
                mvStack.pop();

            //LEFT
                glUniform1i(obj,44);
                mvStack.push(mvStack.top());
                mvStack.top() *= glm::translate(glm::mat4(1.0f), glm::vec3(-2.5,-(1.5 + 4.5) +
 5.5,0.0));
                mvStack.top() *= glm::rotate(glm::mat4(1.0f), rotLeg1 , glm::vec3(1.0,0.0,0.0)
);
                mvStack.push(mvStack.top());
                mvStack.top() *= glm::translate(glm::mat4(1.0f),glm::vec3(0.0,-5.5,0.0));
                mvStack.top() *= glm::scale(glm::mat4(1.0f), glm::vec3(1.5,4.5,1.75));
                glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvStack.top()));
                glBindBuffer(GL_ARRAY_BUFFER, vbo[11]);
                glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
                glEnableVertexAttribArray(0);
                glDrawArrays(GL_TRIANGLES, 0, 36);
                mvStack.pop();

                glUniform1i(obj,43);
                mvStack.push(mvStack.top());
                mvStack.top() *= glm::translate(glm::mat4(1.0f), glm::vec3(0.0,-(4.5 + 4.5 +
5.5)+5.5,0.0));
                mvStack.top() *= glm::rotate(glm::mat4(1.0f), rotLeg3, glm::vec3(1.0,0.0,0.0))
;

                mvStack.push(mvStack.top());
                mvStack.top() *= glm::translate(glm::mat4(1.0f),glm::vec3(0.0,-5.5,0.0));
                mvStack.top() *= glm::scale(glm::mat4(1.0f), glm::vec3(1.5,4.5,1.75));
                glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvStack.top()));
                glBindBuffer(GL_ARRAY_BUFFER, vbo[11]);
                glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
                glEnableVertexAttribArray(0);
                glDrawArrays(GL_TRIANGLES, 0, 36);
                mvStack.pop();
                //shoes
                glUniform1i(obj,42);
                mvStack.push(mvStack.top());
                mvStack.top() *= glm::translate(glm::mat4(1.0f), glm::vec3(0.0,-(4.5+1.0+5.5)
,1.0));
                mvStack.top() *= glm::scale(glm::mat4(1.0f), glm::vec3(1.5,1.0,3.0));
                glUniformMatrix4fv(mvLoc, 1, GL_FALSE, glm::value_ptr(mvStack.top()));
                glBindBuffer(GL_ARRAY_BUFFER, vbo[11]);
                glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
                glEnableVertexAttribArray(0);
                glDrawArrays(GL_TRIANGLES, 0, 36);
                mvStack.pop();

    mvStack.pop();
    mvStack.pop();
    mvStack.pop();
    mvStack.pop();
    mvStack.pop();
}
```

Listing 8: main.cpp :: *Construcción del personaje*

# 4. Shaders

## 4.1. *Vertex Shader*

```
#version 430

layout (location = 0) in vec3 position;
layout (location = 1) in vec2 tex_coord;

out vec2 tc;

uniform mat4 mv_matrix;
uniform mat4 proj_matrix;
uniform int obj;

layout (binding=0) uniform sampler2D s;

void main(void)
{
    if(obj == 1){// la roca
        vec3 newposition;
        newposition = position + vec3(0.0,0.0,-180.0*(gl_InstanceID));
        gl_Position = proj_matrix * mv_matrix * vec4(newposition,1.0);
    }else{
        gl_Position = proj_matrix * mv_matrix * vec4(position,1.0);
    }
    tc = tex_coord;

}
```

Listing 9: vs.glsl :: Vertex Shader

## 4.2. *Fragment Shader*

```
#version 430

layout (location = 0) in vec3 position;
layout (location = 1) in vec2 tex_coord;

out vec2 tc;

uniform mat4 mv_matrix;
uniform mat4 proj_matrix;
uniform int obj;

layout (binding=0) uniform sampler2D s;

void main(void)
{
    if(obj == 1){// la roca
        vec3 newposition;
        newposition = position + vec3(0.0,0.0,-180.0*(gl_InstanceID));
        gl_Position = proj_matrix * mv_matrix * vec4(newposition,1.0);
    }else{
        gl_Position = proj_matrix * mv_matrix * vec4(position,1.0);
    }
    tc = tex_coord;

}
```

Listing 10: fs.glsl :: Fragment Shader

# Listings