UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

REPORTE DE PRÁCTICA Nº 04

NOMBRE COMPLETO: Medrano Miranda Daniel Ulises

Nº de Cuenta: 318045351

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 07/Septiembre/2024

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

1.- Ejecución de los ejercicios que se dejaron, comentar cada uno y capturas de pantalla de bloques de código generados y de ejecución del programa.

1.- Terminar la Grúa con:

- Cuerpo (prisma rectangular)
- base (pirámide cuadrangular)
- 4 llantas (4 cilindros) con teclado se pueden girar las 4 llantas por separado

Modificaciones en el archivo Windows.h:

```
6   class Window
7   {
8   public:
9       Window();
10      Window(GLint windowWidth, GLint windowHeight);
11      int Initialise();
12      GLfloat getBufferWidth() { return bufferWidth; }
13      GLfloat getBufferHeight() { return bufferHeight; }
14      bool getShouldClose() {
15          return  glfwWindowShouldClose(mainWindow);}
16      bool* getsKeys() { return keys; }
17      GLfloat getXChange();
18      GLfloat getYChange();
19      void swapBuffers() { return glfwSwapBuffers(mainWindow); }
20      GLfloat getrotay() { return rotay; }
21      GLfloat getrotax() { return rotax; }
22      GLfloat getrotaz() { return rotaz; }
23      GLfloat getarticulacion1()  { return articulacion1; }
24      GLfloat getarticulacion2()  { return articulacion2; }
25      GLfloat getarticulacion3()  { return articulacion3; }
26      GLfloat getarticulacion4()  { return articulacion4; }
27      GLfloat getarticulacion5()  { return articulacion5; }
28      GLfloat getarticulacion6()  { return articulacion6; }
29      GLfloat getarticulacion7()  { return articulacion7; }
30      GLfloat getarticulacion8()  { return articulacion8; }
31      GLfloat getarticulacion9()  { return articulacion9; }
32      GLfloat getarticulacion10() { return articulacion10; }
33      GLfloat getarticulacion11() { return articulacion11; }
34      GLfloat getarticulacion12() { return articulacion12; }
35      GLfloat getarticulacion13() { return articulacion13; }
36
37      ~Window();
```

Modificaciones en el archivo Windows.cpp:

```cpp
Window::Window(GLint windowWidth, GLint windowHeight)
{
    width = windowWidth;
    height = windowHeight;
    rotax = 0.0f;
    rotay = 0.0f;
    rotaz = 0.0f;
    articulacion1 = 0.0f;
    articulacion2 = 0.0f;
    articulacion3 = 0.0f;
    articulacion4 = 0.0f;
    articulacion5 = 0.0f;
    articulacion6 = 0.0f;
    articulacion7 = 0.0f;
    articulacion8 = 0.0f;
    articulacion9 = 0.0f;
    articulacion10 = 0.0f;
    articulacion11 = 0.0f;
    articulacion12 = 0.0f;
    articulacion13 = 0.0f;

    for (size_t i = 0; i < 1024; i++)
    {
        keys[i] = 0;
    }
}
void Window::ManejaTeclado(GLFWwindow* window, int key, int code, int action, int mode)
{
    Window* theWindow = static_cast<Window*>(glfwGetWindowUserPointer(window));

    if (key == GLFW_KEY_ESCAPE && action == GLFW_PRESS)
    {
        glfwSetWindowShouldClose(window, GL_TRUE);
    }


    if (key == GLFW_KEY_E)
    {
        theWindow->rotax += 10.0;
    }
    if (key == GLFW_KEY_R)
    {
        theWindow->rotay += 10.0; //rotar sobre el eje y 10 grados
    }
    if (key == GLFW_KEY_T)
    {
        theWindow->rotaz += 10.0;
    }
    if (key == GLFW_KEY_F)
    {
        theWindow->articulacion1 += 10.0;
    }

    if (key == GLFW_KEY_G)
    {
        theWindow->articulacion2 += 10.0;
    }
```

```
144     if (key == GLFW_KEY_H)
145     {
146         theWindow->articulacion3 += 10.0;
147     }
148     if (key == GLFW_KEY_J)
149     {
150         theWindow->articulacion4 += 10.0;
151     }
152     if (key == GLFW_KEY_K)
153     {
154         theWindow->articulacion5 += 10.0;
155     }
156     if (key == GLFW_KEY_L)
157     {
158         theWindow->articulacion6 += 10.0;
159     }
160     if (key == GLFW_KEY_Z)
161     {
162         theWindow->articulacion7 += 10.0;
163     }
164
165     if (key == GLFW_KEY_X)
166     {
167         theWindow->articulacion8 += 10.0;
168     }
169     if (key == GLFW_KEY_C)
170     {
171         theWindow->articulacion9 += 10.0;
172     }
173     if (key == GLFW_KEY_V)
174     {
175         theWindow->articulacion10 += 10.0;
176     }
177     if (key == GLFW_KEY_B)
178     {
179         theWindow->articulacion11 += 10.0;
180     }
181     if (key == GLFW_KEY_N)
182     {
183         theWindow->articulacion12 += 10.0;
184     }
185     if (key == GLFW_KEY_M)
186     {
187         theWindow->articulacion13 += 10.0;
188     }
189
190     if (key == GLFW_KEY_D && action == GLFW_PRESS)
191     {
192         const char* key_name = glfwGetKeyName(GLFW_KEY_D, 0);
193         //printf("se presiono la tecla: %s\n",key_name);
194     }
```

Código en el archivo main:

```
345     // - - - - - - - - - - - - - - - - - - - - - - - - GRUA / CAMIÓN - - - - - - - - - - - - - - - - - - - - - - - - -
346     // Creando el brazo de una grúa
347     //articulacion1 hasta articulación5 sólo son puntos de rotación o articulación, en este caso no dibujaremos esfe
348
349     //para reiniciar la matriz de modelo con valor de la matriz identidad
350     model = glm::mat4(1.0);
351     //AQUÍ SE DIBUJA LA CABINA, LA BASE, LAS 4 LLANTAS
352
353     // SE EMPIEZA EL DIBUJO DEL BRAZO
354     //articulación 1
355     model = glm::translate(model, glm::vec3(0.0f, 6.0f, -4.0f));
356     //rotación alrededor de la articulación que une con la cabina
357     model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 0.0f, 1.0f));
358
359     //primer brazo que conecta con la cabina
360     // //Traslación inicial para posicionar en -Z a los objetos
361     //model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));
362     //otras transformaciones para el objeto
363     model = glm::translate(model, glm::vec3(-1.0f, 2.0f, 0.0f));
364     model = glm::rotate(model, glm::radians(135.0f), glm::vec3(0.0f, 0.0f, 1.0f));
365     modelaux = model;
366     model = glm::scale(model, glm::vec3(5.0f, 1.0f, 1.0f));
367
368
369     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
370     //la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
371     //se programe cambio entre proyección ortogonal y perspectiva
372     glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
373     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
374     color = glm::vec3(1.0f, 0.0f, 1.0f);
375     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
376     meshList[0]->RenderMesh(); //dibuja cubo, pirámide triangular, pirámide base cuadrangular
377     //meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono
378
379     //para descartar la escala que no quiero heredar se carga la información de la matrix auxiliar
380     model = modelaux;
381     //articulación 2
382     model = glm::translate(model, glm::vec3(2.5f, 0.0f, 0.0f));
383     model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 0.0f, 1.0f));
384     modelaux = model;
385     //dibujar una pequeña esfera
386     model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
387     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
388     sp.render();
389
390     model = modelaux;
391     //segundo brazo
392     model = glm::translate(model, glm::vec3(0.0f, -2.5f, 0.0f));
393
394     modelaux = model;
395     model = glm::scale(model, glm::vec3(1.0f, 5.0f, 1.0f));
396     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
397     color = glm::vec3(0.0f, 1.0f, 0.0f);
398     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
399     meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
400
401     model = modelaux;
402
403     //articulación 3 extremo derecho del segundo brazo
404     model = glm::translate(model, glm::vec3(0.0f, -2.5f, 0.0f));
405     model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(0.0f, 0.0f, 1.0f));
406     modelaux = model;
407
408
409     //dibujar una pequeña esfera
410     model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
411     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
```

```
412         sp.render();
413
414         // Crear instancias para completar el brazo y  la cabina. Imporante considerar que la cabina es el nodo padre.
415         //La cabina y el brazo deben de estar unidos a la cabina
416
417         model = modelaux;
418         //tercer brazo
419         model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
420         model = glm::translate(model, glm::vec3(0.0f, -2.5f, 0.0f));
421         modelaux = model;
422         model = glm::scale(model, glm::vec3(1.0f, 5.0f, 1.0f));
423         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
424         color = glm::vec3(1.0f, 0.0f, 0.0f);
425         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
426         meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
427
428         //articulación 4 extremo izquierdo del tercer brazo
429         model = modelaux;
430         model = glm::translate(model, glm::vec3(0.0f, -2.5f, 0.0f));
431         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion4()), glm::vec3(1.0f, 1.0f, 0.0f));
432         modelaux = model;
433         //dibujar una pequeña esfera
434         model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
435         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
436         sp.render();
437
438         model = modelaux;
439         //cabina
440         model = glm::rotate(model, glm::radians(-45.0f), glm::vec3(0.0f, 0.0f, 1.0f));
441         model = glm::translate(model, glm::vec3(0.0f, -2.0f, 0.0f));
442         modelaux = model;
443         model = glm::scale(model, glm::vec3(2.0f, 3.0f, 2.0f));
444         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
445         color = glm::vec3(0.0f, 1.0f, 1.0f);
446         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
447         meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
448
449         //Base
450         model = glm::mat4(1.0); //model = modelaux;
451         //model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
452         model = glm::translate(model, glm::vec3(0.0f, 6.0f, -4.0f));
453         modelaux = model;
454         model = glm::scale(model, glm::vec3(6.0f, 3.0f, 4.0f));
455         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
456         color = glm::vec3(0.0f, 1.0f, 1.0f);
457         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
458         meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
459
460         //Soporte Llantas
461         model = modelaux;
462         //model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
463         model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
464         modelaux = model;
465         model = glm::scale(model, glm::vec3(6.0f, 5.0f, 4.0f));
466         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
467         color = glm::vec3(1.0f, 0.0f, 0.0f);
468         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
469         meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
470
471         //Llanta 1
472         model = modelaux;
473         model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
474         model = glm::translate(model, glm::vec3(2.5f, 2.2f, 2.5f));
475         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0.0f, -1.0f, 0.0f));
476         //modelaux = model;
477         model = glm::scale(model, glm::vec3(1.5f, 1.0f, 1.5f));
```

```cpp
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
//meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono

//Llanta 2
model = modelaux;
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::translate(model, glm::vec3(2.5f, -2.2f, 2.5f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion6()), glm::vec3(0.0f, -1.0f, 0.0f));
//modelaux = model;
model = glm::scale(model, glm::vec3(1.5f, 1.0f, 1.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
//meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono

//Llanta 3
model = modelaux;
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::translate(model, glm::vec3(-2.5f, -2.2f, 2.5f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion7()), glm::vec3(0.0f, -1.0f, 0.0f));
//modelaux = model;
model = glm::scale(model, glm::vec3(1.5f, 1.0f, 1.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
//meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono

//Llanta 4
model = modelaux;
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::translate(model, glm::vec3(-2.5f, 2.2f, 2.5f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion8()), glm::vec3(0.0f, -1.0f, 0.0f));
//modelaux = model;
model = glm::scale(model, glm::vec3(1.5f, 1.0f, 1.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
//meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Consideraciones:

- Letra ‘F’ mueve articulación brazo magenta
- Letra ‘G’ mueve articulación brazo verde
- Letra ‘H mueve articulación brazo rojo
- Letra ‘J’ mueve cabina cyan
- Letra ‘K’ mueve Llanta Derecha En Frente (imagen 1)
- Letra ‘L’ mueve Llanta Derecha Atrás (imagen 1)
- Letra ‘Z’ mueve Llanta Izquierda Atrás (imagen 1)
- Letra ‘X’ mueve Llanta Izquierda En Frente (imagen 1)
  ** *El frente de la grúa está de lado izquierdo*

Imágenes de la grúa:

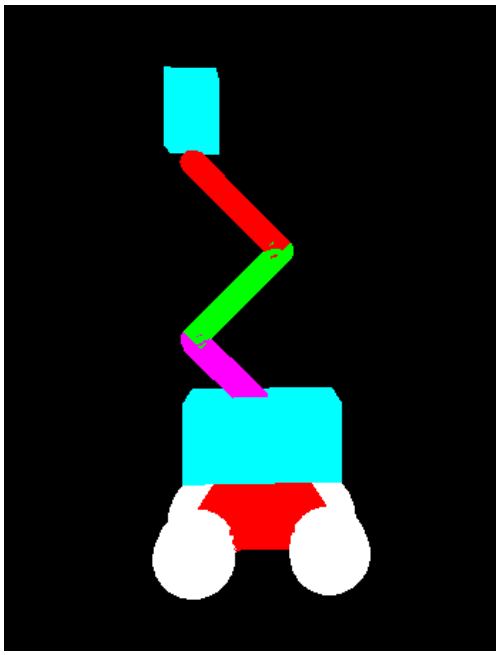


Imagen 1

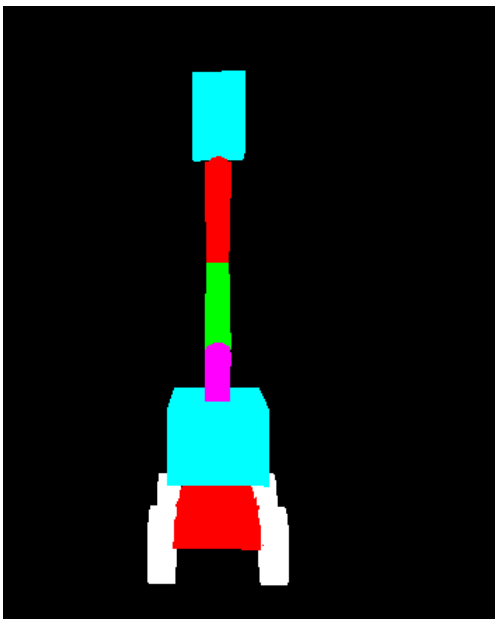


Imagen 2

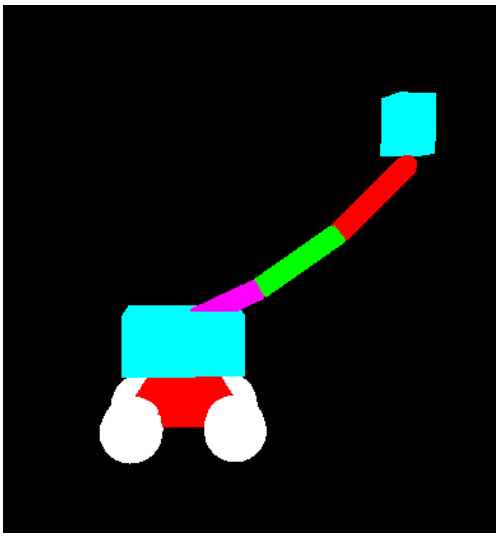


Imagen 3

2.- Crear un animal robot 3d

- Instanciando cubos, pirámides, cilindros, conos, esferas:
- 4 patas articuladas en 2 partes (con teclado se puede mover las dos articulaciones de cada pata)
- cola articulada o 2 orejas articuladas. (con teclado se puede mover la cola o cada oreja independiente

Imagen de referencia:

Código en el archivo main:

```
// - - - - - - - - - - - - - - - - - - - - - - - - ROBOT - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

//  < < CREACIÓN DEL CUERPO > >
// Esfera Frente del Cuerpo
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-4.0f, 5.0f, -4.0f));
modelaux = model;
modelaux4 = model;
modelaux5 = model;
model = glm::scale(model, glm::vec3(2.5f, 2.5f, 2.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
//la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
//se programe cambio entre proyección ortogonal y perspectiva
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();

//Esfera Atras del Cuerpo
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(4.0f, 5.0f, -4.0f));
modelaux1 = model;
modelaux6 = model;
modelaux7 = model;
//dibujar una pequeña esfera
model = glm::scale(model, glm::vec3(2.5f, 2.5f, 2.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();

//Cilindro En Medio del Cuerpo
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 5.0f, -4.0f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(2.5f, 8.0f, 2.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
//meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono

//  < < CREACIÓN DEL CUELLO > >
model = modelaux;
model = glm::translate(model, glm::vec3(-0.5f, 3.5f, 0.0f));
model = glm::rotate(model, glm::radians(15.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, -1.0f, 0.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(1.5f, 4.0f, 1.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.7f, 0.7f, 0.7f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
//meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono

//  < < CREACIÓN DE LA CABEZA > >
model = modelaux;
model = glm::translate(model, glm::vec3(-2.0f, 2.0f, 0.0f));
model = glm::rotate(model, glm::radians(-15.0f), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
//dibujar una pequeña esfera
model = glm::scale(model, glm::vec3(4.0f, 2.0f, 2.25f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.0f, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();

//  < < CREACIÓN DE LAS OREJAS > >
//Oreja Izquierda
```

```
597         model = modelaux;
598         model = glm::translate(model, glm::vec3(2.8f, 2.0f, 1.0f));
599         model = glm::rotate(model, glm::radians(-25.0f), glm::vec3(0.0f, 0.0f, 1.0f));
600         model = glm::scale(model, glm::vec3(1.5f, 2.0f, 1.0f));
601         modelaux2 = model;
602         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
603         color = glm::vec3(0.0f, 0.8f, 0.8f);
604         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
605         meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
606         //meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono
607
608         model = modelaux2;
609         model = glm::translate(model, glm::vec3(0.0f, 0.0f, -0.05f));
610         //modelaux2 = model;
611         model = glm::scale(model, glm::vec3(0.5f, 0.6f, 0.5f));
612         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
613         color = glm::vec3(0.7f, 0.7f, 0.7f);
614         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
615         meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
616         //meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono
617
618
619         //Oreja Derecha
620         model = modelaux;
621         model = glm::translate(model, glm::vec3(2.8f, 2.0f, -1.0f));
622         model = glm::rotate(model, glm::radians(-25.0f), glm::vec3(0.0f, 0.0f, 1.0f));
623         model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
624         model = glm::scale(model, glm::vec3(1.5f, 2.0f, 1.0f));
625         modelaux2 = model;
626         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
627         color = glm::vec3(0.0f, 0.8f, 0.8f);
628         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
629         meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
630         //meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono
631
632         model = modelaux2;
633         model = glm::translate(model, glm::vec3(0.0f, 0.0f, -0.05f));
634         //modelaux2 = model;
635         model = glm::scale(model, glm::vec3(0.5f, 0.6f, 0.5f));
636         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
637         color = glm::vec3(0.7f, 0.7f, 0.7f);
638         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
639         meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
640         //meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono
641
642         //  < < CREACIÓN DE OJOS > >
643         model = modelaux;
644         model = glm::translate(model, glm::vec3(0.0f, 0.2f, 0.0f));
645         model = glm::scale(model, glm::vec3(1.0f, 2.0f, 2.25f));
646         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
647         color = glm::vec3(0.3f, 0.3f, 0.3f);
648         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
649         sp.render();
650
651         //  < < CREACIÓN DE NARIZ > >
652         model = modelaux;
653         model = glm::translate(model, glm::vec3(-4.0f, 0.2f, 0.0f));
654         model = glm::scale(model, glm::vec3(0.7f, 0.7f, 0.7f));
655         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
656         color = glm::vec3(0.3f, 0.3f, 0.3f);
657         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
658         sp.render();
659
660         //  < < CREACIÓN DE PATAS > >
661         // - - - Pata Frontal Derecha - - -
662         //Articulación Superior Pata Frontal Derecha
```

```
663         model = modelaux4;
664         model = glm::translate(model, glm::vec3(0.0f, 0.0f, 2.5f));
665         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 0.0f, -1.0f));
666         modelaux4 = model;
667         model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.5f));
668         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
669         color = glm::vec3(0.7f, 0.7f, 0.7f);
670         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
671         sp.render();
672
673         //Parte Superior Pata Frontal Derecha
674         model = modelaux4;
675         model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.8f));
676         modelaux4 = model;
677         model = glm::scale(model, glm::vec3(0.8f, 3.0f, 0.8f));
678         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
679         color = glm::vec3(0.0f, 0.9f, 0.9f);
680         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
681         //meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
682         meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono
683
684         //Articulación Inferior Pata Frontal Derecha
685         model = modelaux4;
686         model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
687         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(0.0f, 0.0f, -1.0f));
688         modelaux4 = model;
689         //model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
690         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
691         color = glm::vec3(0.7f, 0.7f, 0.7f);
692         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
693         sp.render();
694
695         //Parte Inferior Pata Frontal Derecha
696         model = modelaux4;
697         model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
698         modelaux4 = model;
699         model = glm::scale(model, glm::vec3(0.8f, 3.0f, 0.8f));
700         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
701         color = glm::vec3(0.0f, 0.9f, 0.9f);
702         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
703         //meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
704         meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono
705
706         //Pie Pata Frontal Derecha
707         model = modelaux4;
708         model = glm::translate(model, glm::vec3(-0.5f, -1.5f, 0.0f));
709         model = glm::scale(model, glm::vec3(1.3f, 0.7f, 1.0f));
710         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
711         color = glm::vec3(0.7f, 0.7f, 0.7f);
712         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
713         sp.render();
714
715         // - - - Pata Frontal Izquierda - - -
716         //Articulación Superior Pata Frontal Izquierda
717         model = modelaux5;
718         model = glm::translate(model, glm::vec3(0.0f, 0.0f, -2.5f));
719         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion4()), glm::vec3(0.0f, 0.0f, -1.0f));
720         modelaux5 = model;
721         model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.5f));
722         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
723         color = glm::vec3(0.7f, 0.7f, 0.7f);
724         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
725         sp.render();
726
727         //Parte Superior Pata Frontal Izquierda
728         model = modelaux5;
```

```
		model = glm::translate(model, glm::vec3(0.0f, -1.5f, -0.8f));
		modelaux5 = model;
		model = glm::scale(model, glm::vec3(0.8f, 3.0f, 0.8f));
		glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
		color = glm::vec3(0.0f, 0.9f, 0.9f);
		glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
		//meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
		meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono

		//Articulación Inferior Pata Frontal Izquierda
		model = modelaux5;
		model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
		model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0.0f, 0.0f, -1.0f));
		modelaux5 = model;
		//model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
		glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
		color = glm::vec3(0.7f, 0.7f, 0.7f);
		glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
		sp.render();

		//Parte Inferior Pata Frontal Izquierda
		model = modelaux5;
		model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
		modelaux5 = model;
		model = glm::scale(model, glm::vec3(0.8f, 3.0f, 0.8f));
		glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
		color = glm::vec3(0.0f, 0.9f, 0.9f);
		glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
		//meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
		meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono

		//Pie Pata Frontal Izquierda
		model = modelaux5;
		model = glm::translate(model, glm::vec3(-0.5f, -1.5f, 0.0f));
		model = glm::scale(model, glm::vec3(1.3f, 0.7f, 1.0f));
		glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
		color = glm::vec3(0.7f, 0.7f, 0.7f);
		glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
		sp.render();

		// - - - Pata Trasera Derecha - - -
		//Articulación Superior Pata Trasera Derecha
		model = modelaux1;
		model = glm::translate(model, glm::vec3(0.0f, 0.0f, 2.5f));
		model = glm::rotate(model, glm::radians(mainWindow.getarticulacion6()), glm::vec3(0.0f, 0.0f, -1.0f));
		modelaux1 = model;
		model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.5f));
		glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
		color = glm::vec3(0.7f, 0.7f, 0.7f);
		glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
		sp.render();

		//Parte Superior Pata Frontal Derecha
		model = modelaux1;
		model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.8f));
		modelaux1 = model;
		model = glm::scale(model, glm::vec3(0.8f, 3.0f, 0.8f));
		glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
		color = glm::vec3(0.0f, 0.9f, 0.9f);
		glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
		//meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
		meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono

		//Articulación Inferior Pata Frontal Derecha
		model = modelaux1;
		model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
```

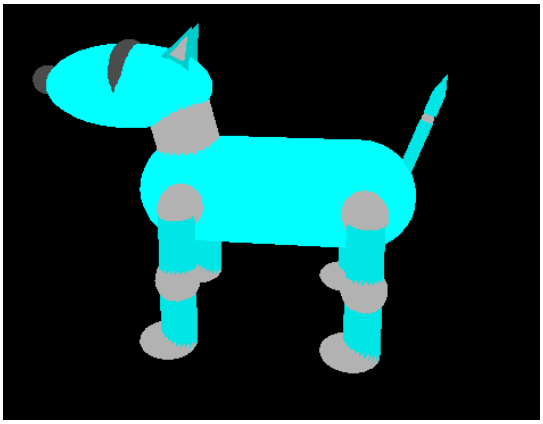
```
795         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion7()), glm::vec3(0.0f, 0.0f, -1.0f));
796         modelaux1 = model;
797         //model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
798         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
799         color = glm::vec3(0.7f, 0.7f, 0.7f);
800         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
801         sp.render();
802
803         //Parte Inferior Pata Frontal Derecha
804         model = modelaux1;
805         model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
806         modelaux1 = model;
807         model = glm::scale(model, glm::vec3(0.8f, 3.0f, 0.8f));
808         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
809         color = glm::vec3(0.0f, 0.9f, 0.9f);
810         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
811         //meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
812         meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono
813
814         //Pie Pata Frontal Derecha
815         model = modelaux1;
816         model = glm::translate(model, glm::vec3(-0.5f, -1.5f, 0.0f));
817         model = glm::scale(model, glm::vec3(1.3f, 0.7f, 1.0f));
818         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
819         color = glm::vec3(0.7f, 0.7f, 0.7f);
820         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
821         sp.render();
822
823         // - - - Pata Trasera Izquierda - - -
824         //Articulación Superior Pata Trasera Izquierda
825         model = modelaux6;
826         model = glm::translate(model, glm::vec3(0.0f, 0.0f, -2.5f));
827         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion8()), glm::vec3(0.0f, 0.0f, -1.0f));
828         modelaux6 = model;
829         model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.5f));
830         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
831         color = glm::vec3(0.7f, 0.7f, 0.7f);
832         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
833         sp.render();
834
835         //Parte Superior Pata Frontal Izquierda
836         model = modelaux6;
837         model = glm::translate(model, glm::vec3(0.0f, -1.5f, -0.8f));
838         modelaux6 = model;
839         model = glm::scale(model, glm::vec3(0.8f, 3.0f, 0.8f));
840         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
841         color = glm::vec3(0.0f, 0.9f, 0.9f);
842         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
843         //meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
844         meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono
845
846         //Articulación Inferior Pata Frontal Izquierda
847         model = modelaux6;
848         model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
849         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion9()), glm::vec3(0.0f, 0.0f, -1.0f));
850         modelaux6 = model;
851         //model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
852         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
853         color = glm::vec3(0.7f, 0.7f, 0.7f);
854         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
855         sp.render();
856
857         //Parte Inferior Pata Frontal Izquierda
858         model = modelaux6;
859         model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
860         modelaux6 = model;
```

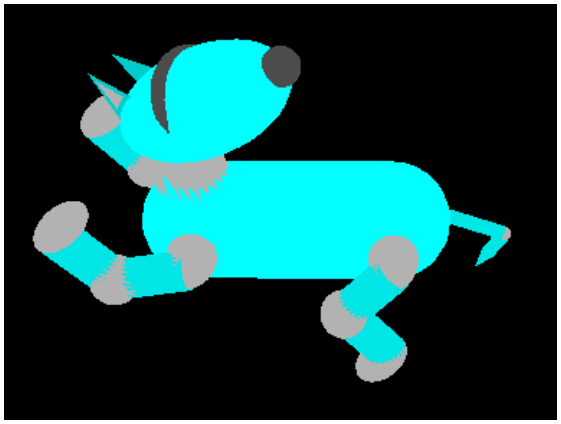
```
861             model = glm::scale(model, glm::vec3(0.8f, 3.0f, 0.8f));
862             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
863             color = glm::vec3(0.0f, 0.9f, 0.9f);
864             glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
865             //meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
866             meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono
867
868             //Pie Pata Frontal Izquierda
869             model = modelaux6;
870             model = glm::translate(model, glm::vec3(-0.5f, -1.5f, 0.0f));
871             model = glm::scale(model, glm::vec3(1.3f, 0.7f, 1.0f));
872             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
873             color = glm::vec3(0.7f, 0.7f, 0.7f);
874             glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
875             sp.render();
876
877             //  < < CREACIÓN DE LA COLA > >
878             //Articulación Base de la Cola
879             model = modelaux7;
880             model = glm::translate(model, glm::vec3(1.7f, 0.5f, 0.0f));
881             model = glm::rotate(model, glm::radians(-20.0f), glm::vec3(0.0f, 0.0f, 1.0f));
882             model = glm::rotate(model, glm::radians(mainWindow.getarticulacion10()), glm::vec3(1.0f, 1.0f, 0.0f));
883             modelaux7 = model;
884             model = glm::scale(model, glm::vec3(0.6f, 0.6f, 0.6f));
885             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
886             color = glm::vec3(0.7f, 0.7f, 0.7f);
887             glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
888             sp.render();
889
890             //Base de la Cola
891             model = modelaux7;
892             model = glm::translate(model, glm::vec3(0.0f, 2.0f, 0.0f));
893             modelaux7 = model;
894             model = glm::scale(model, glm::vec3(0.3f, 2.5f, 0.3f));
895             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
896             color = glm::vec3(0.0f, 0.9f, 0.9f);
897             glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
898             //meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
899             meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono
900
901             //Articulación Punta de la Cola
902             model = modelaux7;
903             model = glm::translate(model, glm::vec3(0.0f, 1.4f, 0.0f));
904             model = glm::rotate(model, glm::radians(mainWindow.getarticulacion11()), glm::vec3(0.0f, 0.0f, 1.0f));
905             modelaux7 = model;
906             model = glm::scale(model, glm::vec3(0.28f, 0.28f, 0.28f));
907             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
908             color = glm::vec3(0.7f, 0.7f, 0.7f);
909             glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
910             sp.render();
911
912             //Punta de la Cola
913             model = modelaux7;
914             model = glm::translate(model, glm::vec3(0.0f, 0.7f, 0.0f));
915             modelaux7 = model;
916             model = glm::scale(model, glm::vec3(0.3f, 1.0f, 0.3f));
917             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
918             color = glm::vec3(0.0f, 0.9f, 0.9f);
919             glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
920             //meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
921             meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono
922
923             //Termino de la Cola
924             model = modelaux7;
925             model = glm::translate(model, glm::vec3(0.0f, 1.0f, 0.0f));
926             model = glm::scale(model, glm::vec3(0.152f, 1.0f, 0.152f));
927             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
928             color = glm::vec3(0.0f, 0.9f, 0.9f);
929             glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
930             //meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
931             meshList[3]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono
932
933             glUseProgram(0);
934             mainWindow.swapBuffers();
935         }
936         return 0;
937     }
```

Consideraciones:

- Letra ‘F’ mueve el cuello
- Letra ‘G’ mueve articulación superior de la pierda frontal izquierda
- Letra ‘H mueve articulación inferior de la pierda frontal izquierda
- Letra ‘J’ mueve articulación superior de la pierda frontal derecha
- Letra ‘K’ mueve articulación inferior de la pierda frontal derecha
- Letra ‘L’ mueve articulación superior de la pierda trasera izquierda
- Letra ‘Z mueve articulación inferior de la pierda trasera izquierda
- Letra ‘X’ mueve articulación superior de la pierda trasera derecha
- Letra ‘C’ mueve articulación inferior de la pierda trasera derecha
- Letra ‘V’ mueve articulación de la base de la cola
- Letra ‘B’ mueve articulación de la punta de la cola

Imágenes del Perro Robot:





2.- Liste los problemas que tuvo a la hora de hacer estos ejercicios y si los resolvió explicar cómo fue, en caso de error adjuntar captura de pantalla

A la hora de realizar los ejercicios no tuve ningún problema, esta vez sí entendí a la perfección lo que se debía hacer y cómo, el único inconveniente que tuve fue con los auxiliarmodels, ya que al principio traté de usar el mismo (caso grúa) pero después pensé en usar más matrices auxiliares y eso me ayudó mucho.

3.- Conclusión:

a. Los ejercicios del reporte: Complejidad, Explicación.

En esta ocasión me gustaron los ejercicios, me pareció divertido ver como pudimos hacer que nuestra figura se moviera de la forma que quisiéramos, me parece que este es un paso clave para poder mover nuestros personajes en nuestro futuro mundo 3D.

b. *Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica*

No sé si estuvo correcto o no usar varias matrices auxiliares, es algo que el profesor no comentó en clase, pero a mi parecer fueron de gran ayuda y me parece que las utilicé de manera correcta.

c. *Conclusión*

Para finalizar con este reporte, me gustaría volver a decir que esta práctica en particular me gustó mucho, al principio pensé que sería muy difícil, pero mientras realizaba la grúa gané experiencia para poder realizar mi perro robot, al final no fue tan difícil como pensaba, sólo que si necesité imaginación para saber cómo realizar el perro robot.