UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

EJERCICIOS DE CLASE 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: 03/Septiembre/2024

CALIFICACIÓN: _____

CAPTURAS

```
346         // Creando el brazo de una grúa
347         //articulacion1 hasta articulación5 sólo son puntos de rotación o articulación, en este caso no dibujaremos esfer
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             glUseProgram(0);
461             mainWindow.swapBuffers();
462         }
463         return 0;
464     }
465
466
467
```