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: CARBAJAL REYES IRVIN JAIR

Nº de Cuenta: 422042084

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 09 DE MARZO DE 2025

CALIFICACIÓN: _____

Actividades

1. Terminar de Construir la grúa con:

    -Cuerpo de la grúa(prisma rectangular).

    -brazo de 3 partes, 4 articulaciones, 1 canasta.

Para esta actividad, se implementaron 4 articulaciones, 3 brazos, una cabina y una base usando jerarquías ayudándonos de matrices auxiliares.

Base

```
//para reiniciar la matriz de modelo con valor de la matriz identidad
model = glm::mat4(1.0);
model = glm::scale(model, glm::vec3(8.0f, 4.0f, 1.0f));
model = glm::translate(model, glm::vec3(0.0f, 1.0f, -4.0f));
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(0.5f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
```

Para la base, se reinicia la matriz con identidad, de este modo no tendrá una rotación con los otros elementos de la grúa.

Para implementar la primera articulación, se vuelve a reiniciar la matríz, y se rotan lo ejes

```
373         model = glm::mat4(1.0);
374         //
375
376         //Articulación 1
377         model = glm::translate(model, glm::vec3(0.0f, 6.0f, -4.0f));
378         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3
379
380         //Primer brazo, conecta con cabina
381         //Traslación inicial para posicionar en -Z a los objetos
382         //model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));
383         //otras transformaciones para el objeto
384         model = glm::translate(model, glm::vec3(-1.0f, 2.0f, 0.0f));
385
386         model = glm::rotate(model, glm::radians(135.0f), glm::vec3(0.0f, 0.0f, 1.0f));
387         modelaux=model;
```

Los elementos de traslación y rotación del modelo se guardan en una matriz auxiliar, esto para poder usar la rotación de los ejes más adelante.

```
388         model = glm::scale(model, glm::vec3(5.0f, 1.0f, 1.0f));
389
390         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
391         //la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
392         //se programe cambio entre proyección ortogonal y perspectiva
393         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
394         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
395         color = glm::vec3(1.0f, 0.0f, 1.0f);
396         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
397         meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
398         //meshList[3]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono, pirám
399         //sp.render(); //dibuja esfera
400
401         model = modelaux;
```

Se implementa el prisma, modificando el modelo sin reiniciar la matriz. Una vez implementado, le asignamos los valores de la matriz auxiliar a la matriz modelo.

Para agregar la articulación al final del prisma, nos trasladamos las unidades necesarias con los ejes rotados y volvemos a guardar los valores dentro de la matriz auxiliar para usarlos más adelante.

```
402         //Articulación 2
403         model = glm::translate(model, glm::vec3(2.5f, 0.0f, 0.0f));
404         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec
405         modelaux = model;
406         //dibujar esfera
407         model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
408         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
409         sp.render();
410
411         model = modelaux;
```

Esto nos permitirá dibujar la esfera, una vez hecho eso, devolvemos los valores a la matriz modelo y podemos implementar otro brazo con el procedimiento anterior.

```
411         model = modelaux;
412         //segundo brazo
413         model = glm::translate(model, glm::vec3(0.0f, -2.5f, 0.0f));
```

```
427         modelaux = model;
428         model = glm::scale(model, glm::vec3(1.0f, 5.0f, 1.0f));
429
430         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
431         //la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
432         //se programe cambio entre proyección ortogonal y perspectiva
433         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
434         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
435         color = glm::vec3(0.0f, 1.0f, 0.0f);
436         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
437         meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
438
439         model = modelaux;
```

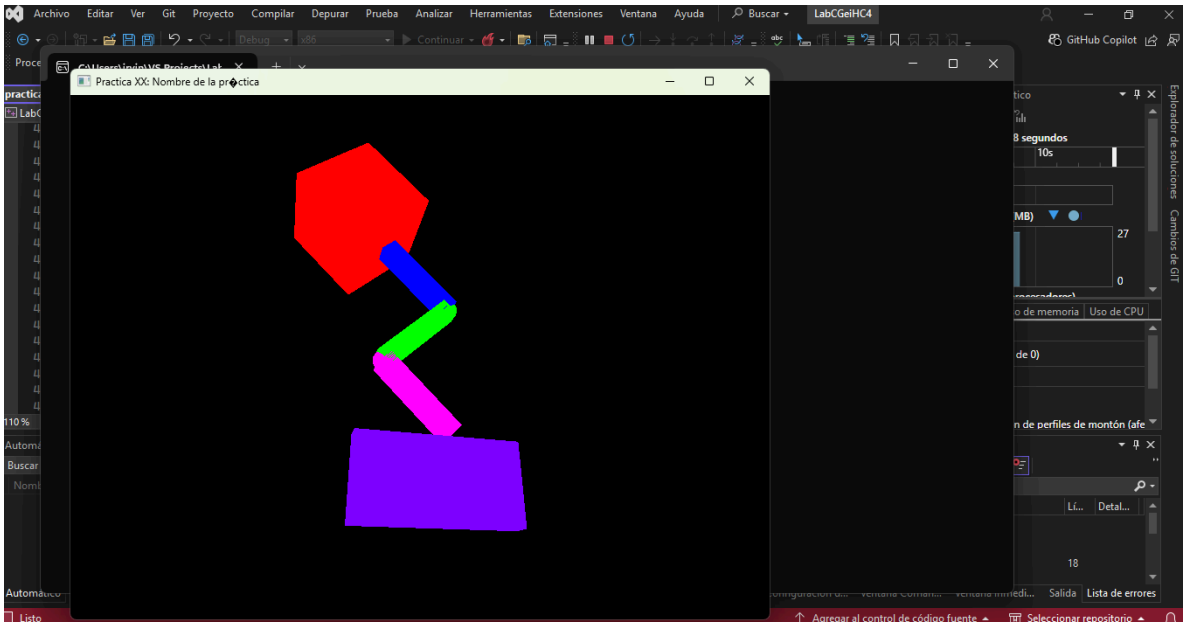
Siempre al recorrer la posición de nuestros ejes guardaremos esos valores en la matriz auxiliar para poder agregar figuras y se modifiquen los valores del modelo sin perderse.

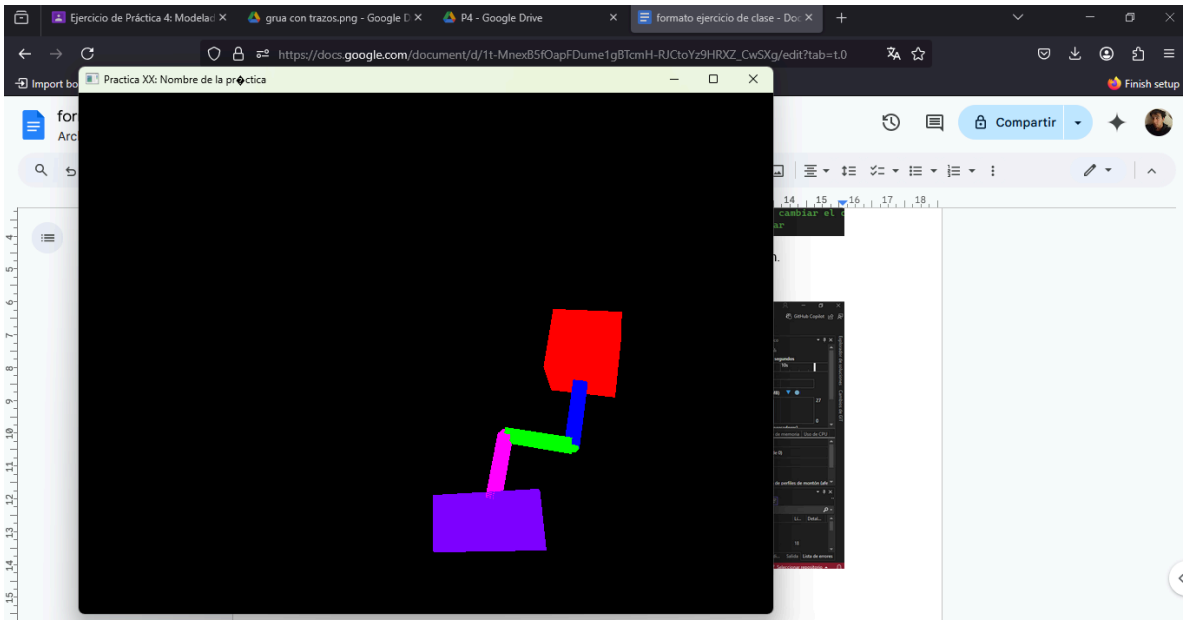
La cabina se agregó de modo similar a los prismas anteriores.

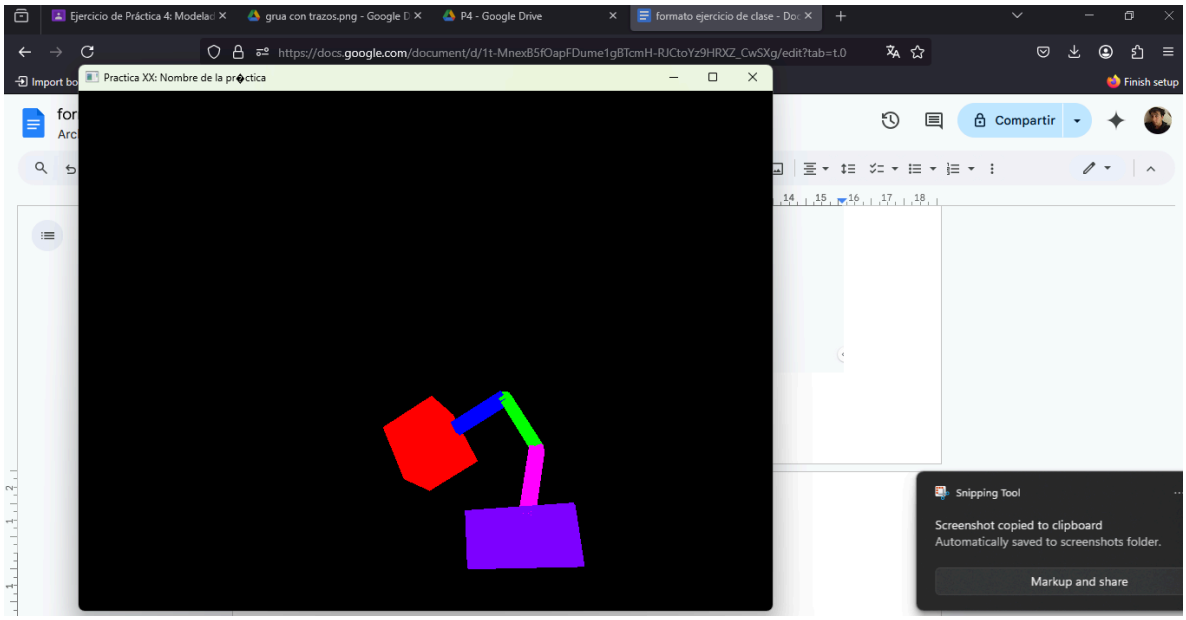
```
472         //Cabina
473         model = glm::translate(model, glm::vec3(2.5f, 0.0f, 0.0f));
474
475         modelaux = model;
476         model = glm::scale(model, glm::vec3(5.0f, 5.0f, 5.0f));
477
478         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
479         color = glm::vec3(1.0f, 0.0f, 0.0f);
480         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el c
481         meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
```

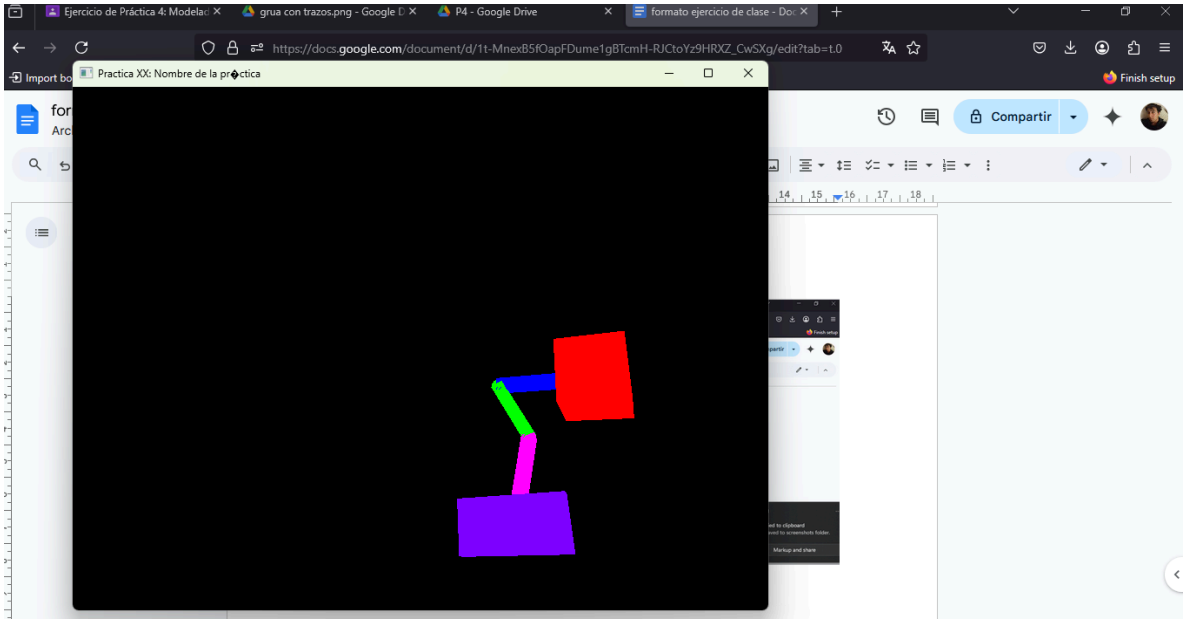
Ya no fue necesario guardar el modelo porque no habrá una quinta articulación.

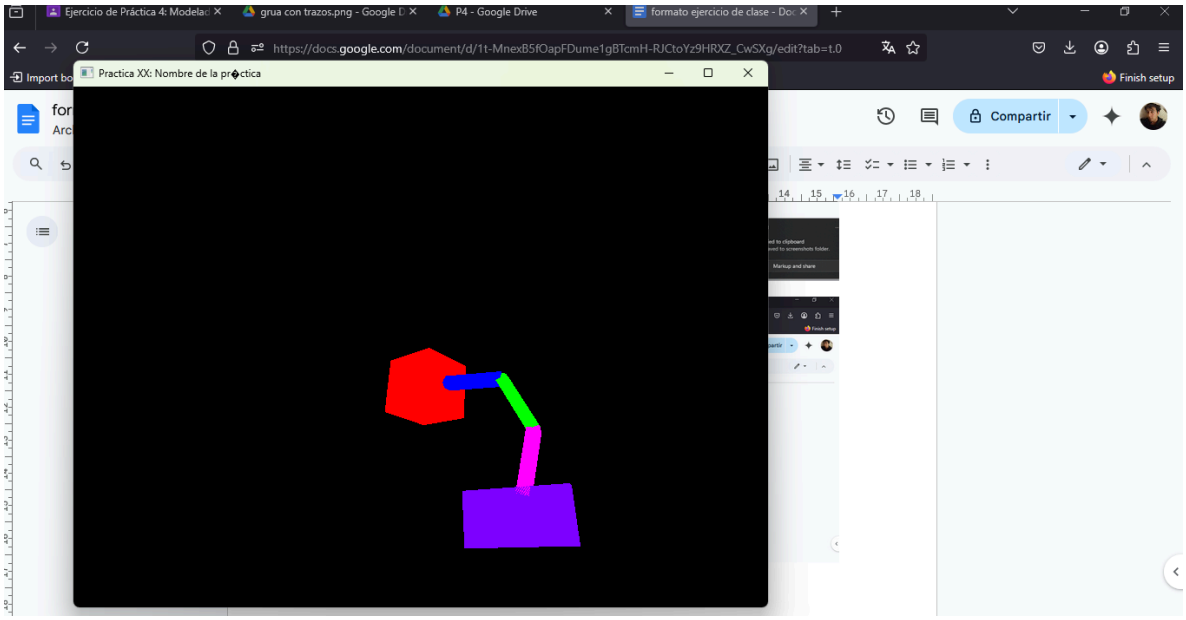
De este modo, la ejecución quedó de la siguiente forma:

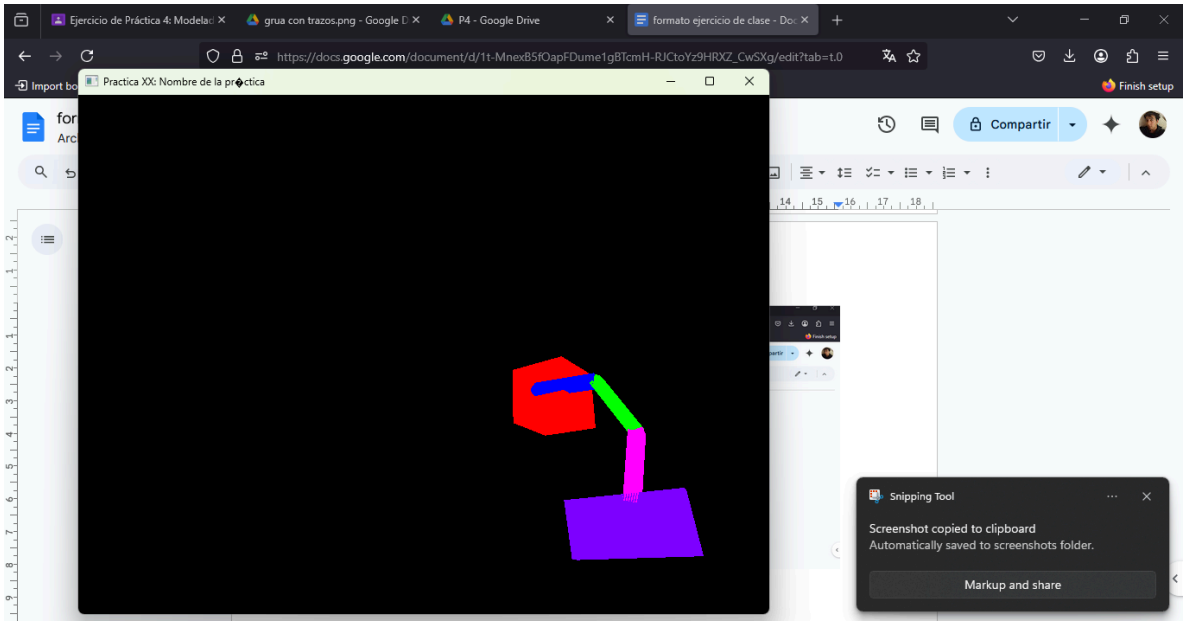





Practica XX: Nombre de la pr☐ctica
Compartir
Snipping Tool
Screenshot copied to clipboard
Automatically saved to screenshots folder.
Markup and share


Practica XX: Nombre de la pr☐ctica
Compartir





Problemas presentados

Durante la elaboración de este ejercicio, tuve que repasar cómo se emplea de manera correcta la jerarquización, ya que en un inicio no me quedó del todo claro, sin embargo el video de apoyo de la práctica me ayudó a entenderlo de manera completa facilitando la realización de mi ejercicio.

Otra problemática que se presentó es que no tenía muy claro en donde colocar el fragmento de código para la base, ya que al principio lo colocaba en zonas donde

generaba distorsión de las figuras, sin embargo, la solución fue reiniciar la matriz después de crear la cabina, y de ahí partir con la jerarquización.

Conclusiones

La dificultad del ejercicio me parece adecuado, siempre y cuando los conceptos queden claros, y si bien en la hora de clase me perdí un poco en la asimilación de los conceptos, el material de apoyo como lo fue el video de la práctica me fue de gran utilidad para comprender los fundamentos de manera completa y realizar el ejercicio de manera correcta aplicando jerarquización para poder generar las articulaciones en cada uno de los brazos y la cabina, y entender en qué situaciones no se debe implementar, como en la base, ya que no queremos que esta se mueva con el resto del brazo.