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: CARBAJAL REYES IRVIN JAIR

Nº de Cuenta: 422042084

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 13 DE MARZO DE 2025

CALIFICACIÓN: _____

Actividades

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

Para esta actividad, se implementó otra matríz auxiliar

```
327     glm::mat4 model(1.0);//Inicializar matriz de Modelo 4x4
328     glm::mat4 modelaux(1.0);//Inicializar matriz de Modelo 4x4
329     glm::mat4 modelaux2(1.0);//Inicializar matriz de Modelo 4x4
```

Luego, en el origen instanciamos un cubo que represente el cuerpo de la grúa.

```
367         model = glm::mat4(1.0);
368         model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));
369         modelaux = model;
370         modelaux2 = model;
371
372         model = glm::scale(model, glm::vec3(5.0f, 3.0f, 5.0f));
373
374         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
375         //la línea de proyección solo se manda una vez a menos que en tiempo de ejecuci
376         //se programe cambio entre proyección ortogonal y perspectiva
377         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
378         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateVie
379         color = glm::vec3(0.5f, 0.0f, 1.0f);
380         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color d
381         meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
```

Luego, guardamos el modelo en ambas matrices auxiliares, una nos ayudará para el resto del cuerpo de la grúa y la otra para los brazos de la misma.

Para la base triangular, usamos la matriz auxiliar 2.

```
385         model = modelaux2;
386         model = glm::translate(model, glm::vec3(0.0f, -3.0f, 0.0f));
387         modelaux2 = model;
388         model = glm::scale(model, glm::vec3(5.0f, 3.0f, 5.0f));
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 ejecució
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.calculateView
395         color = glm::vec3(1.0f, 0.0f, 0.0f);
396         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color de
397         meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
```

Seguimos usando la matriz auxiliar 2, esta vez la usaremos para las cuatro llantas.

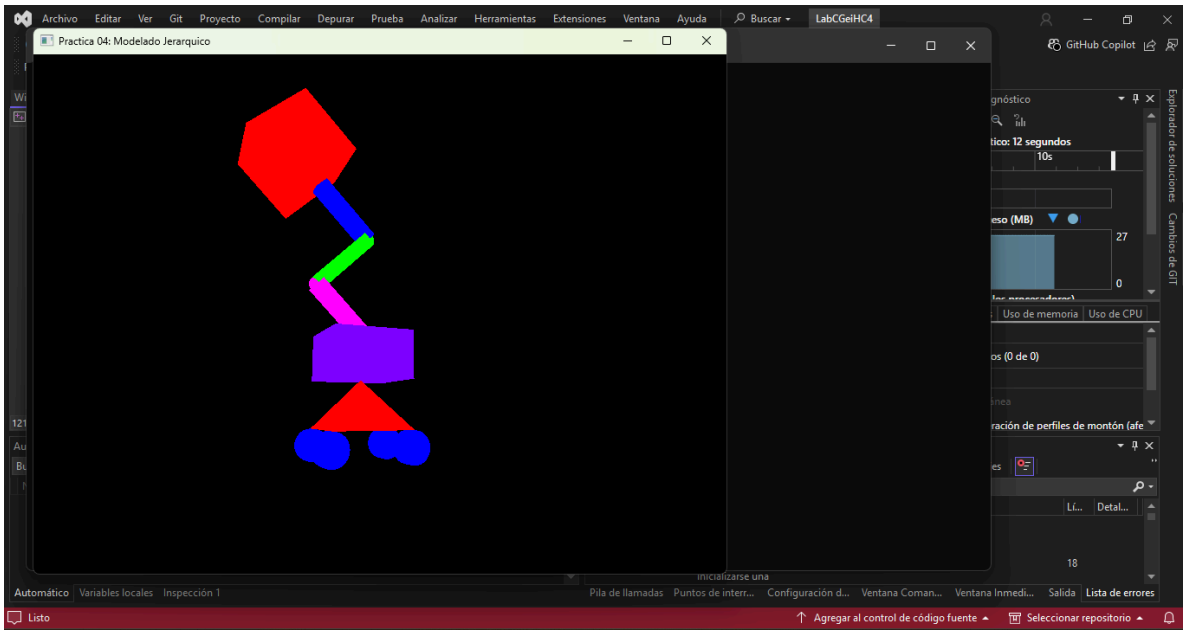
```
399         //Llantas
400         model = modelaux2;
401         model = glm::translate(model, glm::vec3(-2.5f, -2.5f, 2.0f));
402         model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
403         modelaux2 = model;
404         //model = glm::scale(model, glm::vec3(5.0f, 3.0f, 5.0f));
405
406         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
407         //la línea de proyección solo se manda una vez a menos que en tiempo de ejecució
408         //se programe cambio entre proyección ortogonal y perspectiva
409         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
410         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateView
411         color = glm::vec3(0.0f, 0.0f, 1.0f);
412         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color de
413         meshList[2]->RenderMeshGeometry(); //dibuja cubo y pirámide triangular
```

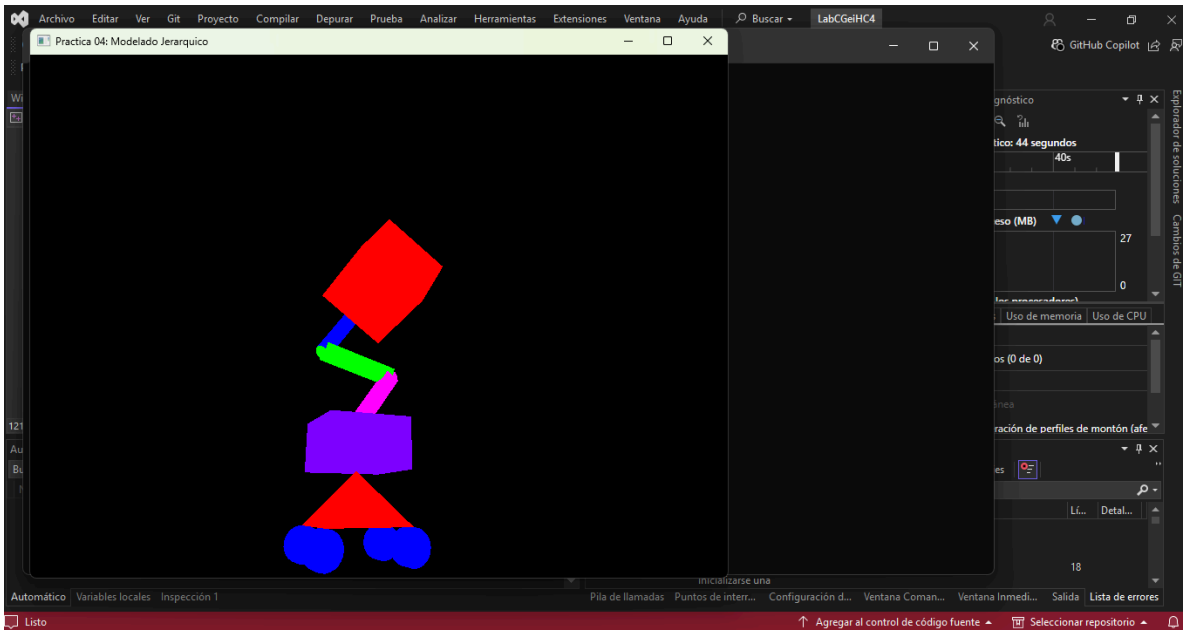
```
441         model = modelaux2;
442         model = glm::translate(model, glm::vec3(-5.0f, 0.0f, 0.0f));
443         modelaux2 = model;
444
445         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
446         //la línea de proyección solo se manda una vez a menos que en tiempo de ejecució
447         //se programe cambio entre proyección ortogonal y perspectiva
448         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
449         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateView
450         color = glm::vec3(0.0f, 0.0f, 1.0f);
451         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color de
452         meshList[2]->RenderMeshGeometry(); //dibuja cubo y pirámide triangular
```

Para conectar el brazo hecho en el ejercicio práctico, usamos la matriz auxiliar 1.

```
454         //
455         model = modelaux;
456         //Articulación 1
457         model = glm::translate(model, glm::vec3(0.0f, 1.5f, -1.0f));
458         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3
459
460         //Primer brazo, conecta con cabina
461         //Traslación inicial para posicionar en -Z a los objetos
462         //otras transformaciones para el objeto
463         model = glm::translate(model, glm::vec3(-1.0f, 1.5f, 0.0f));
464
465         model = glm::rotate(model, glm::radians(135.0f), glm::vec3(0.0f, 0.0f, 1.0f));
466         modelaux=model;
467         model = glm::scale(model, glm::vec3(5.0f, 1.0f, 1.0f));
468
469         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
470         //la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
471         //se programe cambio entre proyección ortogonal y perspectiva
472         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
473         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewM
474         color = glm::vec3(1.0f, 0.0f, 1.0f);
475         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del
```

La ejecución se ve de la siguiente manera.





2. Crear un animal robot 3d

   -Intanciando 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

Para el desarrollo de esta actividad iniciamos declarando dos matrices auxiliares, de este modo nos será más fácil regresar a los puntos del torso donde inician las extremidades articuladas.

```
327         glm::mat4 model(1.0);//Inicializar matriz de Modelo 4x4
328         glm::mat4 modelaux(1.0);//Inicializar matriz de Modelo 4x4
329         glm::mat4 modelaux2(1.0);//Inicializar matriz de Modelo 4x4
```

Instanciamos un cubo para que represente el torso de nuestro león.

```
560         //Creación del torso
561         model = glm::mat4(1.0);
562         model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));
563         modelaux = model;
564         modelaux2 = model;
565
566         model = glm::scale(model, glm::vec3(6.0f, 4.0f, 4.0f));
567
568         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
569         //la línea de proyección solo se manda una vez a menos que en tiempo c
570         //se programe cambio entre proyección ortogonal y perspectiva
571         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(proj
572         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.cal
573         color = glm::vec3(0.85f, 0.65f, 0.25f);
574         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar e
575         meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
576
```

Notamos que en las matrices auxiliares se almacena el modelo inicial, esto es para guardar este punto como de retorno.

Una vez instanciado el cubo, modificando su escala y su color para que represente el torso de nuestro león, usamos la primera matriz auxiliar para comenzar el primer brazo.

```
577         //Brazo 1: Articulación 1
578         model = modelaux;
579         model = glm::translate(model, glm::vec3(-3.0f, -2.0f, 2.0f));
580         modelaux2 = model;
581         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()),
582         modelaux = model;
583
584         //dibujar esfera
585         model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
586         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
587         sp.render();
588
589         model = modelaux;
```

Una vez que nos movemos al punto donde colocaremos la primer articulación, almacenaremos este punto dentro de la segunda matríz auxiliar, después colocamos la rotación, y guardamos el modelo en la primer matríz auxiliar, ya que a partir de esta se generarán las articulaciones que le siguen al brazo. Dibujamos

nuestra esfera y regresamos al punto de la articulación. Nos trasladamos al punto donde queremos dibujar el primer brazo y guardamos en la matriz auxiliar 1 antes de dibujar.

```
590         //Brazo 1A
591         model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
592
593         modelaux = model;
594         model = glm::scale(model, glm::vec3(1.0f, 3.0f, 1.0f));
595
596         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
597         color = glm::vec3(0.85f, 0.65f, 0.25f);
598         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el
599         meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
600
601         model = modelaux;
```

Una vez dibujado, regresamos al punto guardado. Para la articulación 2 y la segunda parte del brazo se sigue el mismo procedimiento.

```
603         //Brazo 1: Articulación 2
604         model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
605         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()),
606         modelaux = model;
607         //dibujar esfera
608         model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
609         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
610         sp.render();
611
612         model = modelaux;
613         //Brazo 1B
614         model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
615
616         modelaux = model;
617         model = glm::scale(model, glm::vec3(1.0f, 3.0f, 1.0f));
618
619         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
620         color = glm::vec3(0.85f, 0.65f, 0.25f);
621         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el
622         meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
623
624         model = modelaux;
```

Después colocamos el pie.

```
625         //Brazo 1: Pie
626         model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
627         modelaux = model;
628
629         model = glm::scale(model, glm::vec3(1.5f, 1.0f, 1.0f));
630         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
631         color = glm::vec3(0.45f, 0.25f, 0.1f);
632         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar e
633         meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
```

Como la segunda matríz auxiliar es la que nos ayuda a regresar al torso del león, colocamos el siguiente brazo con el procedimiento anterior pero ahora usando la matríz auxiliar 2.

```
635         //Brazo 2: Articulación 1
636         model = modelaux2;
637         model = glm::translate(model, glm::vec3(6.0f, 0.0f, 0.0f));
638         modelaux = model;
639         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()),
640         modelaux2 = model;
641         //dibujar esfera
642         model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
643         color = glm::vec3(0.85f, 0.65f, 0.25f);
644         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
645         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
646         sp.render();
```

Notamos que ahora la matriz auxiliar 1 se reinicia antes de la rotación.

```
648         //Brazo 2A
649         model = modelaux2;
650         model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
651
652         modelaux2 = model;
653         model = glm::scale(model, glm::vec3(1.0f, 3.0f, 1.0f));
654
655         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
656         color = glm::vec3(0.85f, 0.65f, 0.25f);
657         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el
658         meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
659
660         model = modelaux2;
```

Una vez implementados los brazos, instanciamos un cilindro y un como para crear la cola del león.

Como para el brazo cuatro se usó la matriz auxiliar 2, la matríz auxiliar 1 es la que nos ayudó a iniciar la cola.

```
812         //Cola: Articulación 1
813         model = modelaux;
814         model = glm::translate(model, glm::vec3(6.0f, 2.0f, 2.0f));
815         modelaux2 = model;
816         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion9()),
817         modelaux = model;
818
819         //dibujar esfera
820         model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
821         color = glm::vec3(0.45f, 0.25f, 0.1f);
822         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
823         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
824         sp.render();
```

Luego, dibujamos el cilindro con la rotación y escala necesario para que luzca como cola.

```
827         //Cola A
828         model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
829
830         modelaux = model;
831         model = glm::scale(model, glm::vec3(5.0f, 1.0f, 1.0f));
832         model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
833         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
834         color = glm::vec3(0.45f, 0.25f, 0.1f);
835         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color
836         meshList[2]->RenderMeshGeometry(); //dibuja cubo y pirámide triangular
837
838         model = modelaux;
```

Para la segunda parte de la cola, se implementó otra articulación de la misma manera que con los brazos.

```
839         //Cola: Articulación 2
840         model = glm::translate(model, glm::vec3(2.5f, 0.0f, 0.0f));
841         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion10())
842         modelaux = model;
843         //dibujar esfera
844         model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
845         color = glm::vec3(0.45f, 0.25f, 0.1f);
846         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el
847         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
848         sp.render();
849
850         model = modelaux;
```

Luego, dibujamos el cono.

```
851         //Cola 2
852         model = glm::translate(model, glm::vec3(2.5f, 0.0f, 0.0f));
853
854         modelaux = model;
855         model = glm::scale(model, glm::vec3(5.0f, 1.0f, 1.0f));
856         model = glm::rotate(model, glm::radians(-90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
857         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
858         color = glm::vec3(0.45f, 0.25f, 0.1f);
859         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color d
860         meshList[3]->RenderMeshGeometry(); //dibuja cubo y pirámide triangular
```

Para la cabeza, se instanció un cubo.

```
862         //Cabeza
863         model = modelaux2;
864         model = glm::translate(model, glm::vec3(-6.0f, 2.5f, 0.0f));
865         //model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
866         modelaux = model;
867         modelaux2 = model;
868         model = glm::scale(model, glm::vec3(3.0f, 3.0f, 5.0f));
869
870         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
871         //la línea de proyección solo se manda una vez a menos que en tiempo de ejecució
872         //se programe cambio entre proyección ortogonal y perspectiva
873         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
874         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateView
875         color = glm::vec3(0.85f, 0.65f, 0.25f);
876         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color de
877         meshList[0]->RenderMeshGeometry(); //dibuja cubo y pirámide triangular
```

Para la melena se usaron 3 cubos

```
880         model = modelaux2;
881         model = glm::translate(model, glm::vec3(0.0f, 2.0f, 0.0f));
882         //model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
883         modelaux2 = model;
884         model = glm::scale(model, glm::vec3(3.0f, 2.0f, 7.0f));
885
886         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
887         //la línea de proyección solo se manda una vez a menos que en tiempo de ejecució
888         //se programe cambio entre proyección ortogonal y perspectiva
889         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
890         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateView
891         color = glm::vec3(0.45f, 0.25f, 0.1f);
892         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color de
893         meshList[0]->RenderMeshGeometry(); //dibuja cubo y pirámide triangular
```

Para los ojos se usaron cilindros y esferas

```
925         //Ojos
926         model = modelaux2;
927         model = glm::translate(model, glm::vec3(-2.0f, -0.5f, 2.0f));
928         modelaux2 = model;
929         model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
930
931         model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
932
933         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
934         //la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
935         //se programe cambio entre proyección ortogonal y perspectiva
936         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
937         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMa
938         color = glm::vec3(1.0f, 1.0f, 1.0f);
939         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del
940         meshList[2]->RenderMeshGeometry(); //dibuja cubo y pirámide triangular
941
942         model = modelaux2;
```

```
958         model = modelaux2;
959         model = glm::translate(model, glm::vec3(-0.5f, 0.0f, 0.0f));
960         modelaux2 = model;
961         model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.2f));
962         color = glm::vec3(0.0f, 0.0f, 0.0f);
963         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el
964         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
965         sp.render();
```

Para la nariz una pirámide cuadrangular.

```
976         //Nariz
977         model = modelaux2;
978         model = glm::translate(model, glm::vec3(0.2f, -1.5f, 1.0f));
979         modelaux2 = model;
980         model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
981
982         model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
983
984         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
985         //la línea de proyección solo se manda una vez a menos que en tiempo de ejecuci
986         //se programe cambio entre proyección ortogonal y perspectiva
987         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
988         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateVie
989         color = glm::vec3(0.0f, 0.0f, 0.0f);
990         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color d
991         meshList[4]->RenderMeshGeometry(); //dibuja cubo y pirámide triangular
992
```

Ejecución:

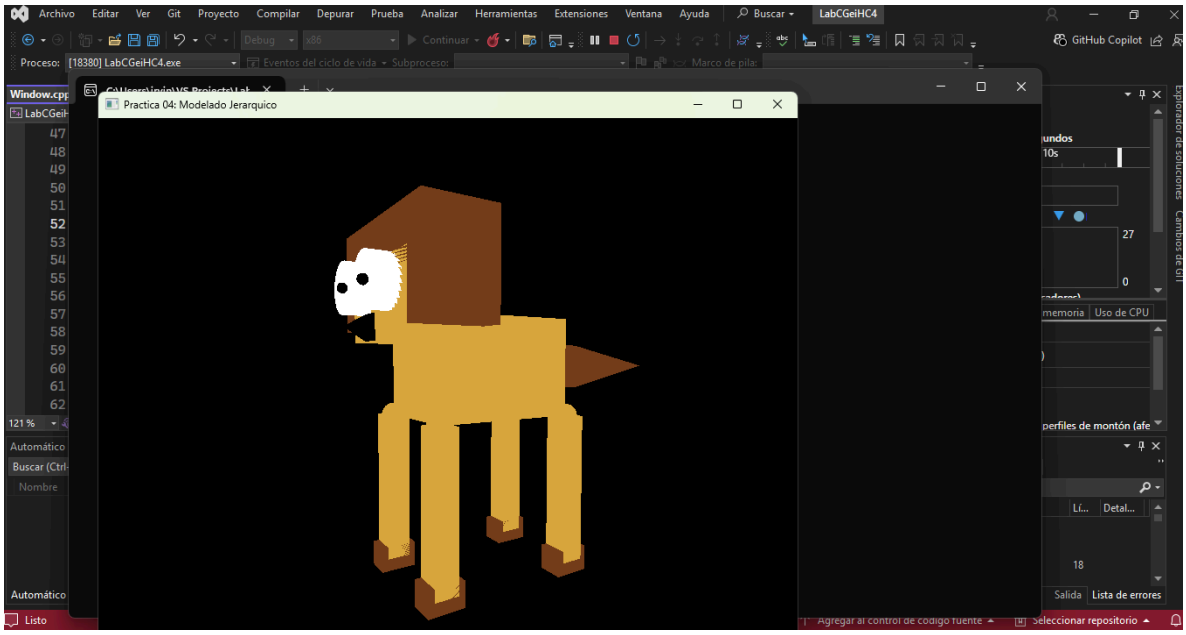
Pata 1: Teclas F y G

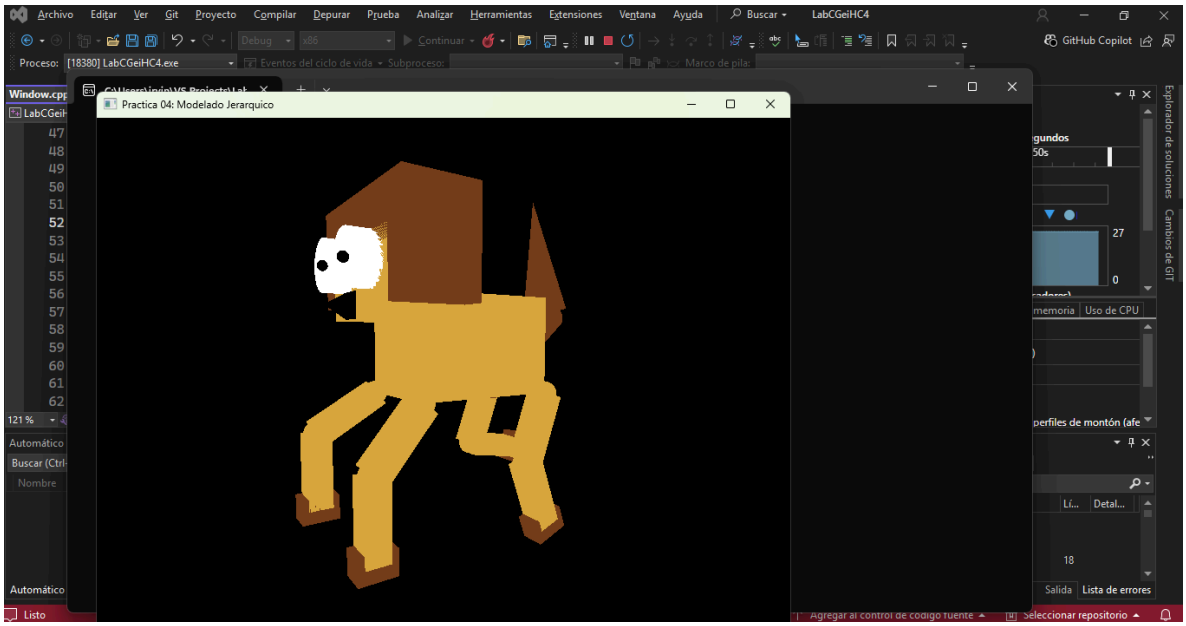
Pata 2: Teclas H y J

Pata 3: Teclas K y L

Pata 4: Teclas U y I

Cola Teclas O y P





Problemáticas

La principal problemática presentada durante el desarrollo de esta práctica fue entender por completo lo que implica aplicar jerarquía a los modelos, ya que en un principio no se obtenían los resultados esperados al aplicar escala después de la jerarquía. Sin embargo, una vez comprendido cómo funciona el modelado jerárquico fue más sencillo instanciar las figuras y crear las articulaciones.

Conclusiones

Durante el desarrollo de esta práctica, se realizaron los ejercicios para comprender por completo el modelado jerárquico, esto nos permitió no tener que reiniciar la matriz modelo cada que instanciamos figuras y poder reutilizar modelos anteriores para poder implementar articulaciones, esto con ayuda de matrices auxiliares, las cuales almacenaban todo lo que colocamos en nuestros modelos y utilizarlos más adelante si es que teníamos que modificar el modelo en ese instante. La explicación del tema fue muy clara, y el video fue de gran ayuda ya que en la explicación dentro del laboratorio no me fue posible seguirla por completo, gracias a eso pude comprender por completo el tema y desarrollar los ejercicios de forma satisfactoria. La complejidad de las actividades fueron adecuadas y nos permitieron entender las ventajas del modelado jerárquico.

Bibliografía

- Tutorial 3: Matrices. (n.d.). https://www-opengl--tutorial-org.translate.goog/beginners-tutorials/tutorial-3-matrices/?_x_tr_sch=http&_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc