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 9

NOMBRE COMPLETO: HERNANDEZ SOLIS BRANDON

Nº de Cuenta: 318263113

GRUPO DE LABORATORIO: 2

GRUPO DE TEORÍA: 6

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 15/10/2024

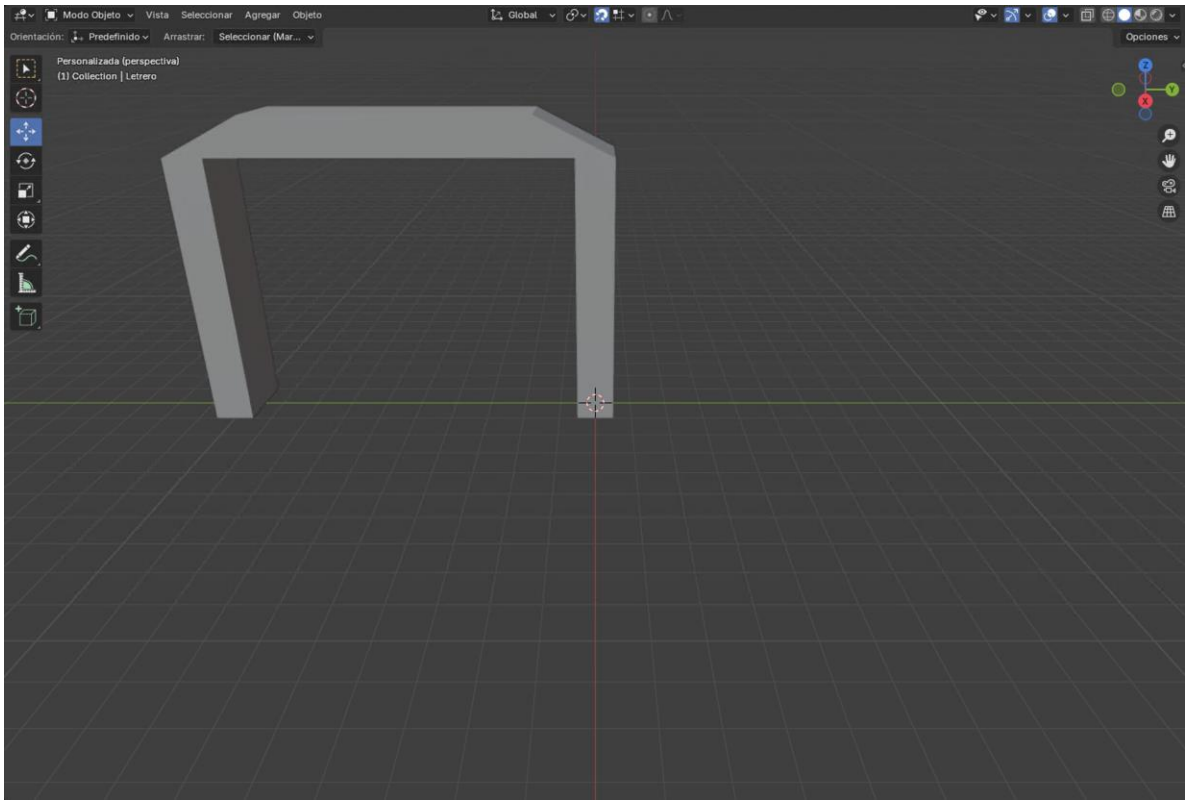
CALIFICACIÓN: _____

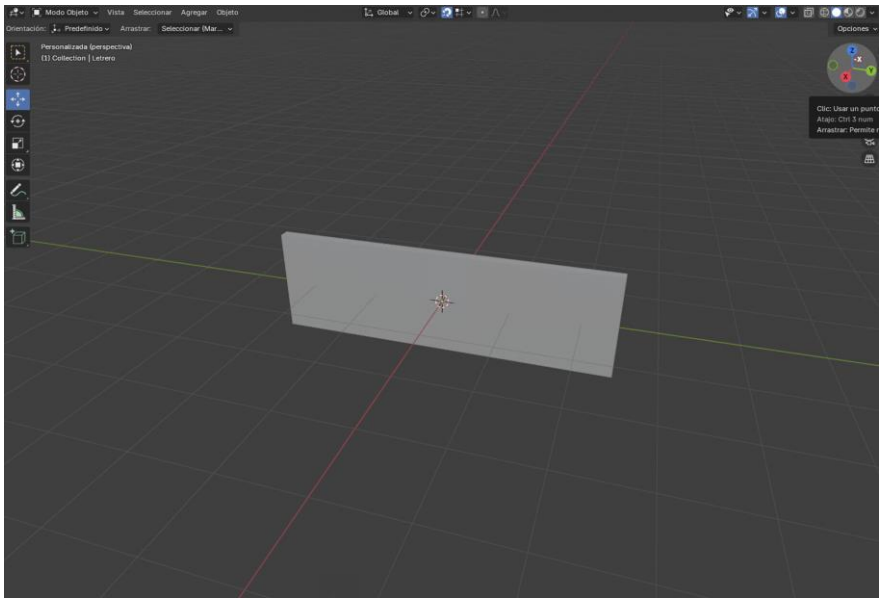
EJERCICIOS DE SESIÓN:

1. Actividades realizadas. Una descripción de los ejercicios y capturas de pantalla de bloques de código generados y de ejecución del programa

   1.- Separar del arco la parte del letrero

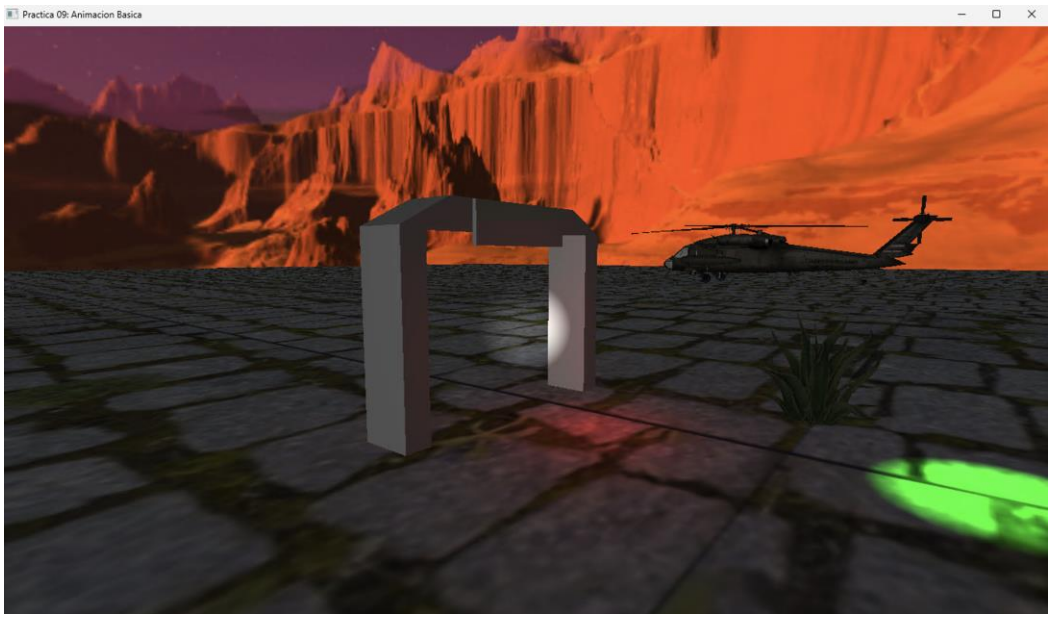
   Esta actividad se realzia desde blender con el modelo de nuestro letrero, para ello es necesario separar el letrero en dos objetos diferentes, asi podemos exportarlos por separado para poder trabajar con ellos en nuestro proyecto.






MTL
Estructura
Estructura
MTL
Letrero
Letrero

Resultados:


Practica 09: Animacion Basica

2.- Hacer que el letrero baje y suba recorriendo toda la altura del arco, esto de acuerdo con un contador de tiempo que durará 2 segundos (2 segundos en bajar, 2 segundos a nivel del piso, 2 segundos subiendo, 2 segundos en la altura máxima y así sucesivamente.

3.- El letrero rotara 360° sobre su propio centro estando en la parte superior y en la parte inferior alrededor del eje X (suponiendo que vemos al arco en dirección de Z negativo).

Los dos ejercicios los juntare ya que están relacionados estrechamente en la sección de código.

Para hacer esto de manera sencilla se puede implementar de manera similar a una máquina de estados, así el control de las acciones sera determinada por el estado anterior y será difícil tener errores en la secuencia.

Para calcular los tiempos he optado por usar banderas que me indicarían en la salida de la consola el tiempo transcurrido en cada estado, asi ajuste mis velocidades de movimiento y rotación para conseguir lo más cercano a los 2 segundos, aunque esto es relativo porque depende de la velocidad de cada procesador.

Código:

```
66
67      //  Ejercicio
68
69      Model Letrero;
70      Model Arco;
71
72      float movLetrero;
73      float offsetLetrero;
74      float rotLetrero;
75      float offsetRotLetrero;
76
77      int estado;
```

```
244
245     //  Ejercicio
246
247     Letrero = Model();
248     Letrero.LoadModel("Models/Letrero.obj");
249     Arco = Model();
250     Arco.LoadModel("Models/Estructura.obj");
251
```

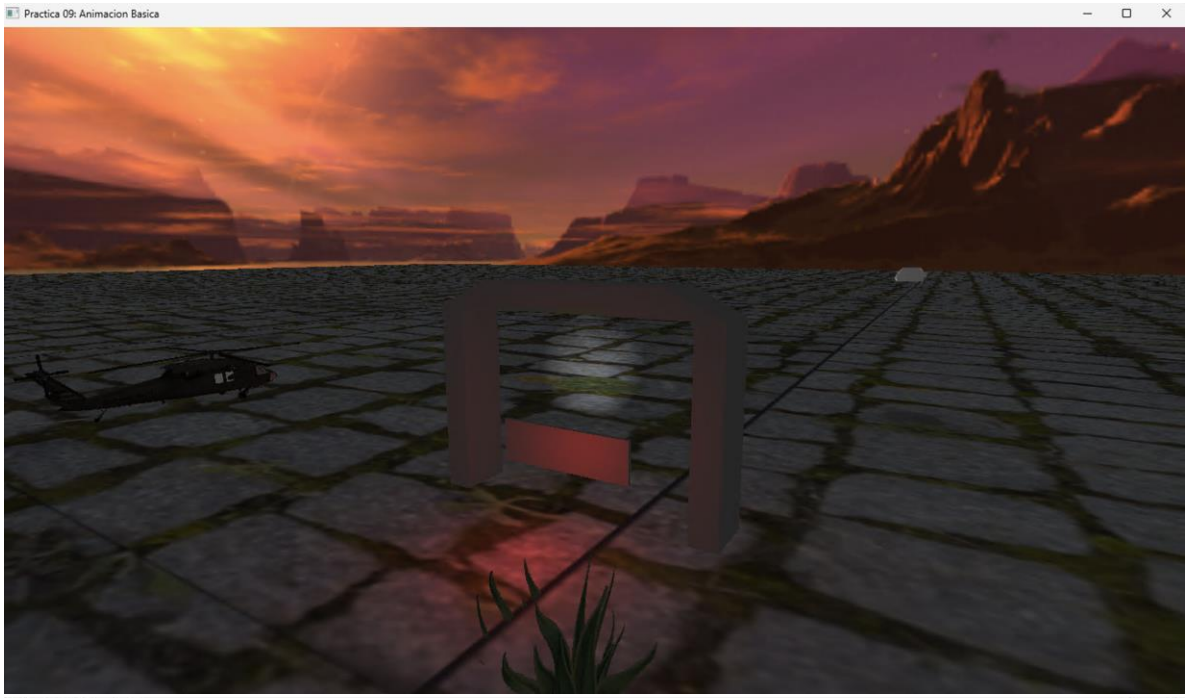
```
315
316     //  Ejercicio
317     movLetrero = 0.0f;
318     offsetLetrero = 0.058;      //Velocidad Movimiento
319     rotLetrero = 0.0f;
320     offsetRotLetrero = 2.9f;    //Velocidad Rotacion
321
322     estado = 0;
323
```

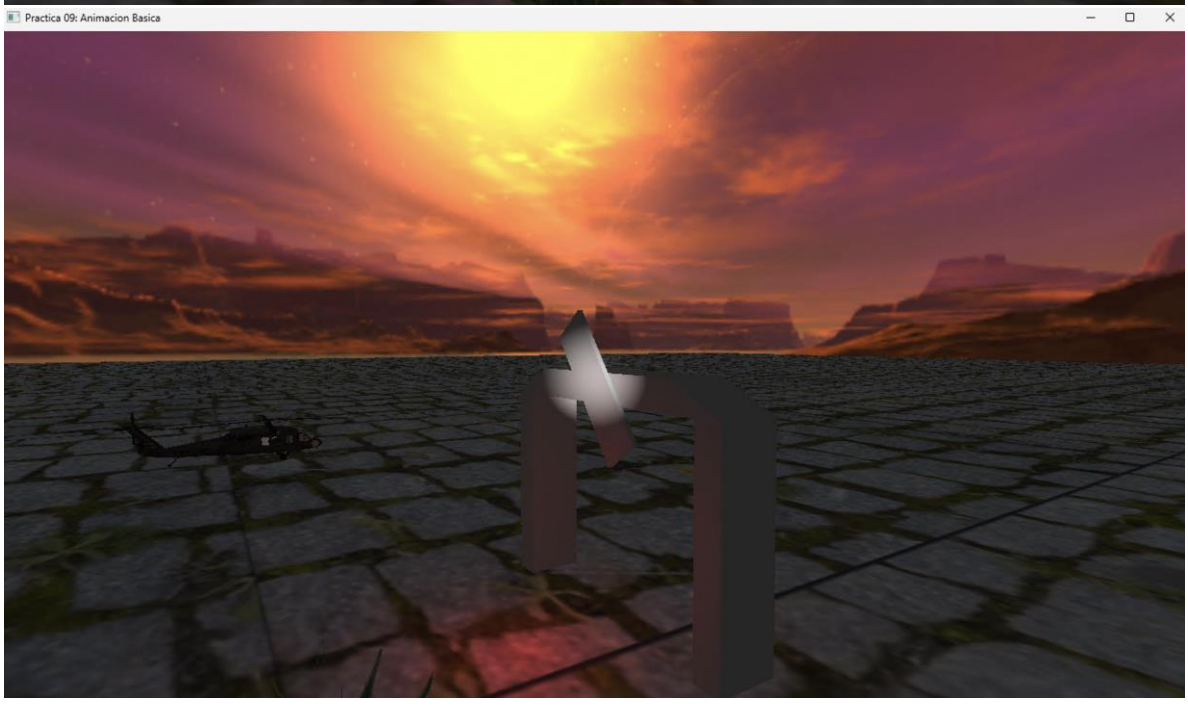
```
481
482     //  Ejercicio
483
484     model = glm::mat4(1.0);
485     model = glm::translate(model, glm::vec3(-10.0f, -1.0f, -3.0f));
486     model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
487     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
488     Arco.RenderModel();
489
490     model = glm::mat4(1.0);
491     model = glm::translate(model, glm::vec3(-8.9f, movLetrero + 6.5f, 2.0f));          //Movimiento en Y
492     model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
493     model = glm::rotate(model, rotLetrero * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));    //Rotacion sobre eje X
494     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
495     Letrero.RenderModel();
496
```

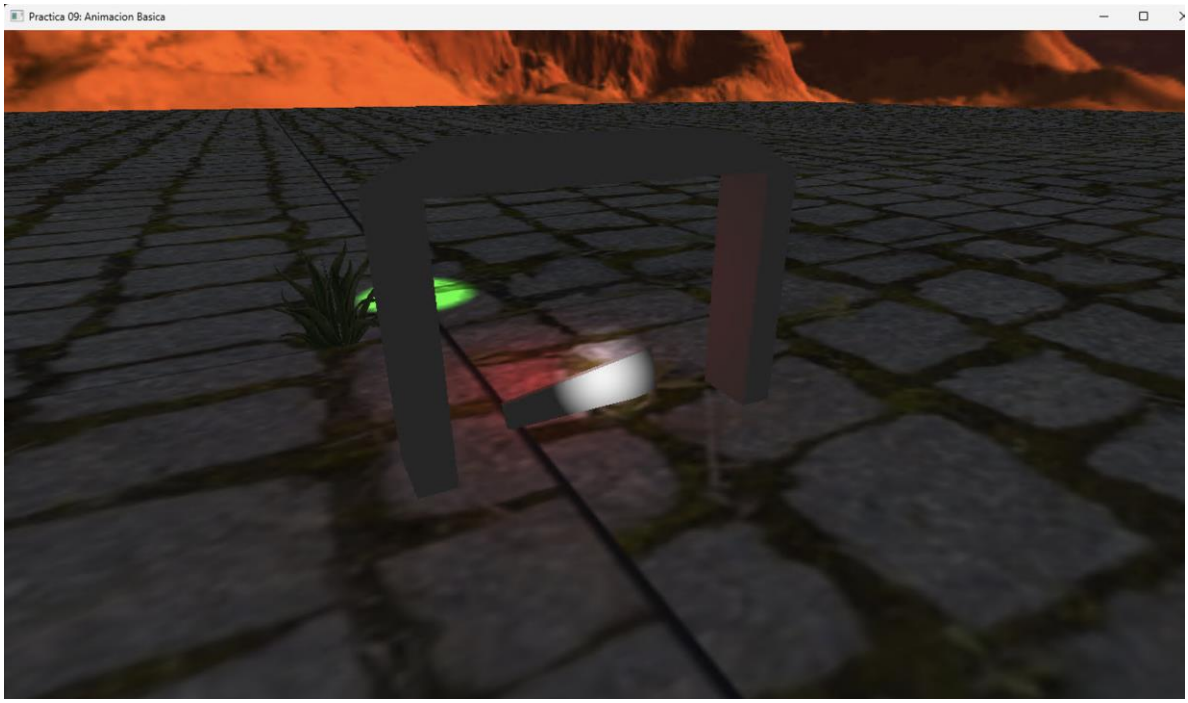
Animación:

```
496
497         //  Animacion
498
499         if (glfwGetTime() > 2) {
500
501             if (estado == 0) {
502                 if (movLetrero > -7.0f) {
503                     movLetrero -= offsetLetrero * deltaTime;
504                 }
505                 else {
506                     printf("Estado #0: %f \n", glfwGetTime());
507                     estado = 1;
508                 }
509             }
510
511             if (estado == 1) {
512                 if (rotLetrero <= 360.0f) {
513                     rotLetrero += offsetRotLetrero * deltaTime;
514                 }
515                 else {
516                     printf("Estado #1: %f \n", glfwGetTime());
517                     rotLetrero = 0;
518                     estado = 2;
519                 }
520             }
521
522             if (estado == 2) {
523                 if (movLetrero < 0.0f) {
524                     movLetrero += offsetLetrero * deltaTime;
525                 }
526                 else {
527                     printf("Estado #2: %f \n", glfwGetTime());
528                     estado = 3;
529                 }
530             }
531
532             if (estado == 3) {
533                 if (rotLetrero >= -360.0f) {
534                     rotLetrero -= offsetRotLetrero * deltaTime;
535                 }
536                 else {
537                     printf("Estado #3: %f \n", glfwGetTime());
538                     rotLetrero = 0;
539                     estado = 0;
540                 }
541             }
542
543         }
544
```

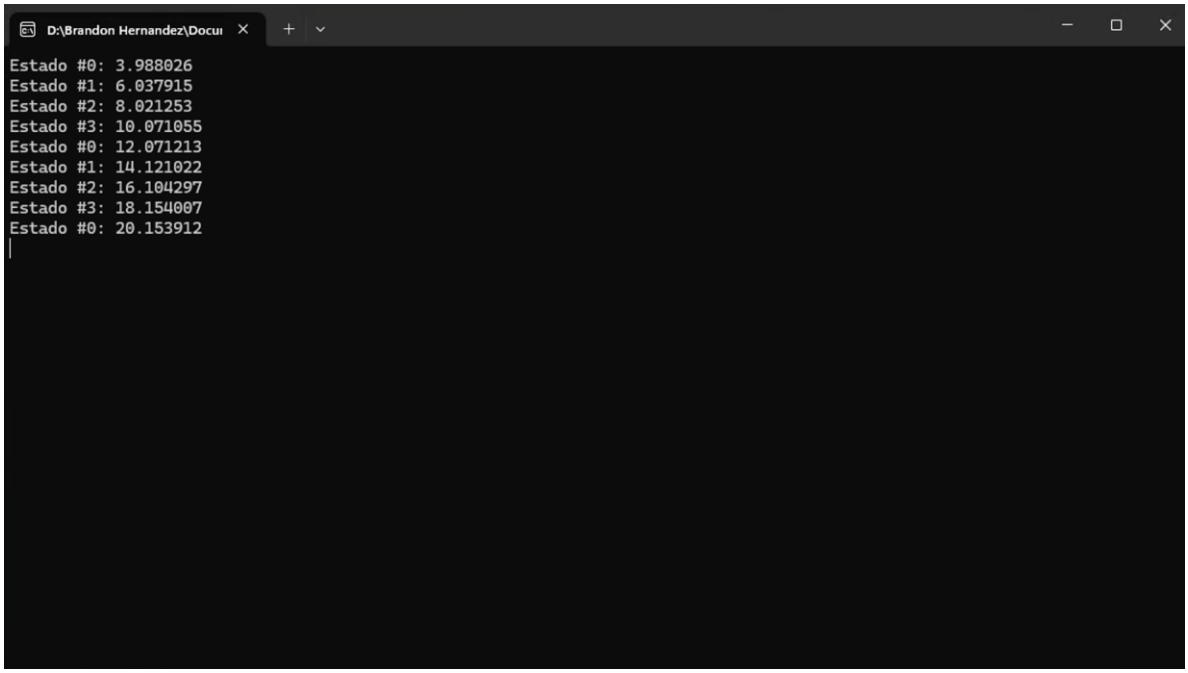
Resultados:


Practica 09: Animacion Basica


Practica 09: Animacion Basica



Salida de la terminal:



2. Problemas presentados. Listar si surgieron problemas a la hora de ejecutar el código

No hubo problemas

3. Conclusión:
    a. Los ejercicios de la clase: Complejidad, explicación
    b. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias.

Esta actividad de animación estuvo interesante porque pensé que sería más complejo de entender, pero realmente estamos usando nuestros conocimientos de programación en C++ para controlar las variables que realizan las traslaciones y rotaciones de nuestros modelos. Es interesante pensar que tanto podríamos animar si seguimos usando estructuras de control para modificar todas las variables que usamos normalmente, incluso las escalas de los objetos o los shaders de luz para dar efectos de que se funde una lampara o se atenúa la luz con el tiempo.