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 8

NOMBRE COMPLETO: BRANDON HERNANDEZ SOLIS

Nº de Cuenta: 318263113

GRUPO DE LABORATORIO: 2

GRUPO DE TEORÍA: 6

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 19/10/2024

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. Su dado de 10 caras cae sobre el piso, gira y cae en un número "random", se repite la tirada al presionar una tecla.

Para realizar esta actividad tuve que crear una estructura parecida a un menú, donde el control de ellos dependía de la variable que elegiría la animación a ejecutar, adiciona a esto tuve que crear una animación 0 que es cuando se calcula el siguiente valor de la animación.

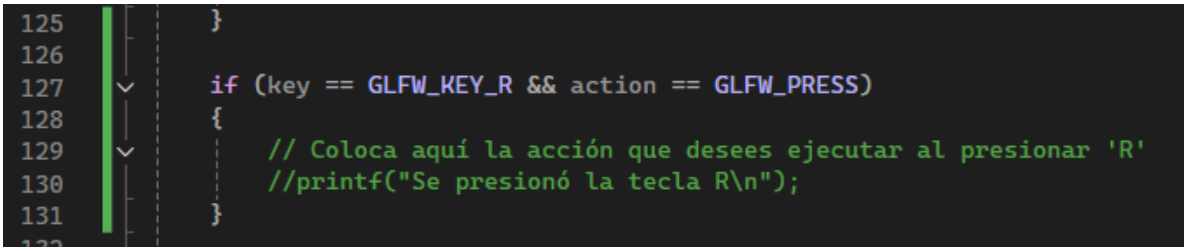
Para el cálculo "random" de la animación a ejecutar se usó la función ya incluida dentro de c++, de allí solo se sacaba el módulo para elegir la animación.

Dentro de cada animación teníamos una máquina de estados la cual controla los movimientos de los dados, dentro del control de los movimientos tenemos las condicionales las cuales son los movimientos que realizara antes de quedarse en el estado de espera para la siguiente ejecución.

Para reiniciar los valores y ejecutar una nueva animación, en el último estado asignamos los valores de movimiento nuevamente a 0 después de haber recibido la orden por el teclado para realizar una nueva tirada.

También se agregó una luz desde la parte superior hacia la zona donde estaría el dado.

Window.cpp



Main.cpp:

```
67  //  Practica
68
69  Model Dado;
70
71  // Variables declaradas
72  float movDadoX;
73  float movDadoY;
74  float movDadoZ;
75  float offsetMovDado;
76
77  float rotDadoX;
78  float rotDadoY;
79  float rotDadoZ;
80  float offsetRotDado;
81
82  int estado;
83  int animacion;
84
85  bool lanzar;
86  bool rKeyPressed;
87
```

```
252     BlackhawR_M.LoadModel("Models/uh60.obj");
253
254     //  Practica
255
256     Dado = Model();
257     Dado.LoadModel("Models/Dado10Caras.obj");
258
259
```

```
290
291       //luz fija
292       spotLights[1] = SpotLight(1.0f, 1.0f, 1.0f,
293           1.0f, 2.0f,
294           5.0f, 40.0f, 0.0f,
295           0.0f, -5.0f, 0.0f,
296           1.0f, 0.0f, 0.001f,
297           45.0f);
298       spotLightCount++;
```

```
316
317         //  Practica
318         movDadoX = 0.0f;
319         movDadoY = 0.0f;
320         movDadoZ = 0.0f;
321         offsetMovDado = 0.08f;      //Velocidad Movimiento
322
323         rotDadoX = 0.0f;
324         rotDadoY = 0.0f;
325         rotDadoZ = 0.0f;
326         offsetRotDado = 1.0f;   //Velocidad Rotacion
327
328         estado = 0;
329         animacion = 0;
330
331         lanzar = false;
332         rKeyPressed = false;
333
```

```
469
470         //  Practica    //////////////
471
472         //  Modelo
473
474         model = glm::mat4(1.0);
475         model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
476
477
478         model = glm::translate(model, glm::vec3(movDadoX, movDadoY + 12.0f, movDadoZ));
479
480         model = glm::rotate(model, rotDadoX * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
481         model = glm::rotate(model, rotDadoY * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
482         model = glm::rotate(model, rotDadoZ * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
483
484         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
485         Dado.RenderModel();
486
```

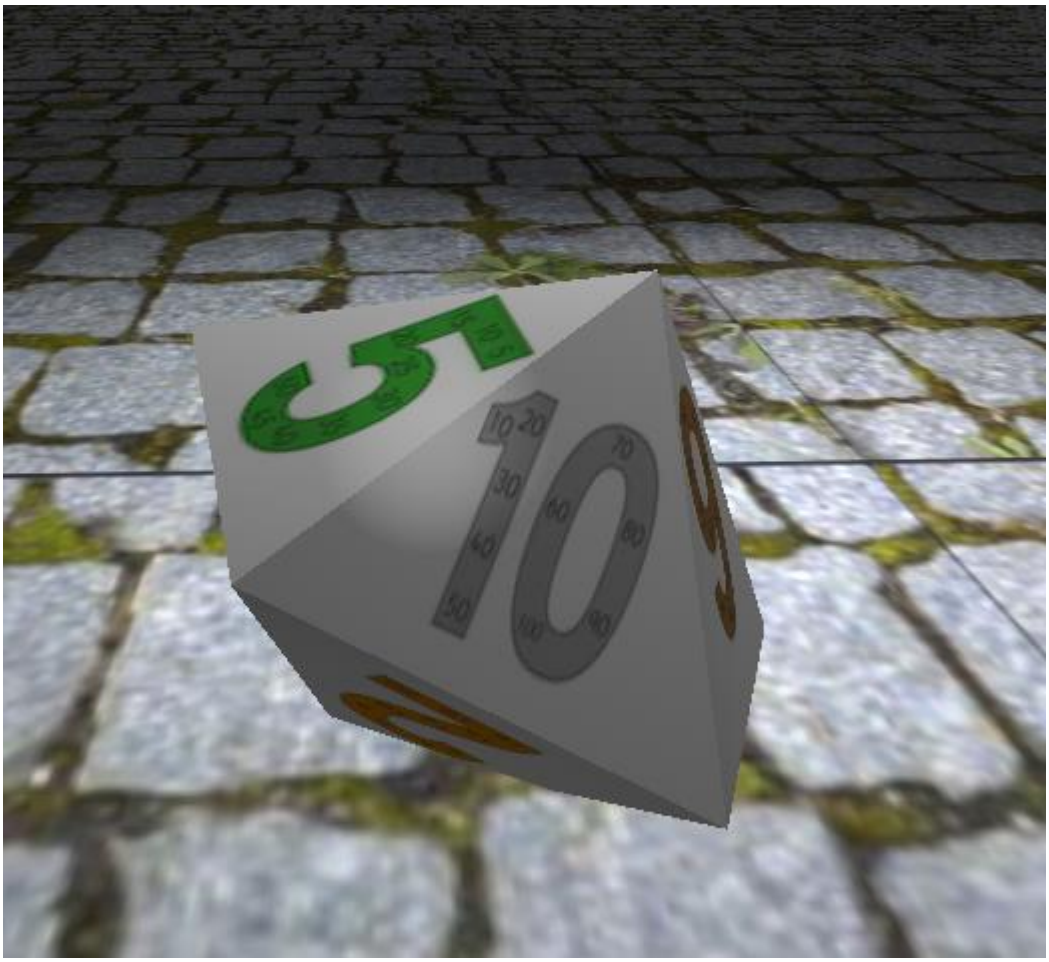
```
493             // Animaciones
494
495             if (glfwGetTime() > 5) {
496
497                 //  Animacion #0
498                 if (animacion == 0) {
499
500                     if (lanzar == false) {
501                         animacion = rand() % 4 + 1;
502                         printf("Lanzando dado ... \n");
503                         printf("Animacion # %d \n", animacion);
504                         printf("------------------- \n\n");
505                     }
506
507                 }
```

```
509             // Animacion #1
510             if (animacion == 1) {
511
512                 if (estado == 0) {
513                     if (movDadoY > -10.0f) {
514                         movDadoY -= offsetMovDado * deltaTime;
515                     }
516                     else {
517                         estado = 1;
518                     }
519                 }
520
521                 if (estado == 1) {
522                     if (movDadoX <= 20.0f && rotDadoY <= 45.0f && rotDadoZ <= 150.0f) {
523                         movDadoX -= offsetMovDado * deltaTime;
524                         rotDadoY += offsetRotDado * deltaTime;
525                         rotDadoZ += offsetRotDado * deltaTime;
526                     }
527                     else {
528                         estado = 2;
529                     }
530                 }
```

```
531
532                     // Espera la tecla
533                     if (estado == 2) {
534                         if (lanzar == true) {
535                             animacion = 0;
536                             estado = 0;
537
538                             movDadoX = 0.0f;
539                             movDadoY = 0.0f;
540                             movDadoZ = 0.0f;
541
542                             rotDadoX = 0.0f;
543                             rotDadoY = 0.0f;
544                             rotDadoZ = 0.0f;
545                         }
546                     }
547                 }
548
```

```
677
678         if (glfwGetKey(mainWindow.getMainWindow(), GLFW_KEY_R) == GLFW_PRESS && !rKeyPressed) {
679             // Alternar el booleano de la luz 2
680             lanzar = true;
681             rKeyPressed = true; // Marcar que la tecla se presionó
682         }
683
684         if (glfwGetKey(mainWindow.getMainWindow(), GLFW_KEY_R) == GLFW_RELEASE) {
685             rKeyPressed = false; // Permitir que la tecla pueda alternar nuevamente al soltarse
686             lanzar = false;
687         }
688
689
690
```

Resultados:



2. Por integrante del equipo elegirán un tipo de vehículo: terrestre o aéreo. Cada integrante del equipo creará un recorrido en el cual el vehículo se moverá alrededor de su tablero de Monopoly. Cada vehículo iniciará su recorrido a partir de una esquina diferente. (el vehículo terrestre no puede ser un carro o vehículo similar motorizado de 4 ruedas, se debe de tener movimiento de llantas o de hélices en sus vehículos.)

Para esta actividad solo se importó el modelo del helicóptero de la temática de Halo, se requirió escalar y acomodar antes de renderizar. Para la animación se sigue un ciclo por el cual los if estaban anidados y había una variable que controlaba cada aspecto del movimiento del helicóptero.

Main.cpp:

```
 89
 90      float movHelX;
 91      float movHelY;
 92      float movHelZ;
 93      float offsetMovHel;
 94
 95      float rotHelX;
 96      float rotHelY;
 97      float rotHelZ;
 98      float offsetRotHel;
 99
100      int estado2;
101
```

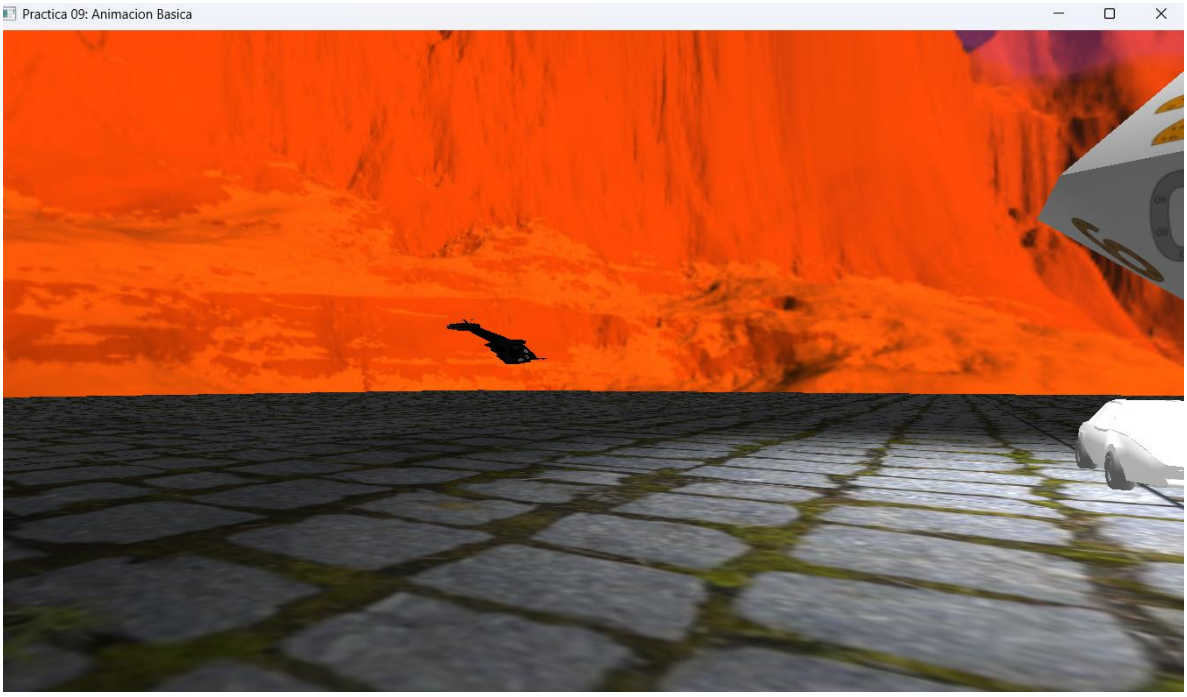
```
351          movHelX = 0.0f;
352          movHelY = 0.0f;
353          movHelZ = 0.0f;
354          offsetMovHel = 10.0f;
355
356          rotHelX = 0.0f;
357          rotHelY = 0.0f;
358          rotHelZ = 0.0f;
359          offsetRotHel = 0.5f;
360
361          estado2 = 0;
362
```

```
499         //  Practica    //////////////
500
501         //  Modelo
502
503         model = glm::mat4(1.0);
504         model = glm::translate(model, glm::vec3(-150.0f, 10.0f, -150.0f));
505         model = glm::scale(model, glm::vec3(0.1f, 0.1f, 0.1f));
506
507         model = glm::translate(model, glm::vec3(movHelX * 10, 0.0f, movHelZ * 10));
508
509         model = glm::rotate(model, rotHelY * toRadians, glm::vec3(0.0f, -1.0f, 0.0f));
510
511
512         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
513         Helicopero.RenderModel();
```

```
728
729             if (glfwGetTime() > 3) {
730
731                 if (estado2 == 0) {
732
733                     rotHelX = 0.0f;
734                     rotHelY = -90.0f;
735                     rotHelZ = 0.0f;
736
737                     estado2 += 1;
738                 }
739
740                 if (estado2 == 1) {
741                     if (movHelX <= 300.0f) {
742                         movHelX += offsetMovHel * deltaTime;
743                     }
744                     else {
745                         estado2 += 1;
746                     }
747                 }
```

```
793
794             if (estado2 == 7) {
795                 if (movHelZ >= -300.0f) {
796                     movHelZ -= offsetMovHel * deltaTime;
797                 }
798                 else {
799                     estado2 += 1;
800                 }
801             }
802
803             if (estado2 == 8) {
804                 if (rotHelY <= 270.0f) {
805                     rotHelY += offsetRotHel * deltaTime;
806                 }
807                 else {
808                     estado2 = 0;
809                 }
810             }
811
812         }
813
814
```

Resultados:



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.

No hubo problemas

3.- Conclusión:

a. Los ejercicios del reporte: Complejidad, Explicación.
b. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica
c. Conclusión

La práctica sirvió para aprender a animar básicamente cualquier modelo de una forma sencilla a través del código simple que ya conocíamos, solo se requirió a diseñar bien la forma de controlar todas las variables.