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 9-2

NOMBRE COMPLETO: LOPEZ BETANCOURT MICHELLE

Nº de Cuenta: 318309028

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 06

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 27 DE OCTUBRE DE 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.

- Los ejercicios por realizar para esta práctica fueron primero mediante el modelo creado anteriormente del arco se debía mostrar la palabra PROYECTO CGEIHC MONOPOLY de forma animada y desplazándose las letras de izquierda a derecha como si fuera un letrero LCD/LED de forma cíclica. Para el caso del ejercicio dos se debían separar cada una de las cinco cabezas del dragón y agregar cada una de las siguientes animaciones:

  → Movimiento del cuerpo de ida y vuelta
  → Aleteo de las alas
  → Cada una de las 5 cabezas del dragón se debían mover de forma diferente de acuerdo con alguna función o algoritmo diferente (espiral de Arquímedes, movimiento senoidal, lemniscata, etc.)
  → Y cada una de las 5 cabezas debía tener un color distinto entre rojo, azul, verde, blanco y café.

Código generado

```
57
58      Window mainWindow;
59      std::vector<Mesh*> meshList;
60      std::vector<Shader> shaderList;
61
62      Camera camera;
63
64      Texture brickTexture;
65      Texture dirtTexture;
66      Texture plainTexture;
67      Texture pisoTexture;
68      Texture AgaveTexture;
69      Texture FlechaTexture;
70      Texture NumerosTexture;
71      Texture Numero1Texture;
72      Texture Numero2Texture;
73      Texture blancoTexture;
74      Texture letrasTexture;
75
76
77
78      Model Kitt_M;
79      Model Llanta_M;
80      Skybox skybox;
81      Model cartel_M;
82      Model arco_M;
83      Model cuerpoDragon_M;
84      Model alaDragonDerecha_M;
85      Model alaDragonIzquierda_M;
```

```
76
77
78      Model Kitt_M;
79      Model Llanta_M;
80      Skybox skybox;
81      Model cartel_M;
82      Model arco_M;
83      Model cuerpoDragon_M;
84      Model alaDragonDerecha_M;
85      Model alaDragonIzquierda_M;
86      Model cabeza1_M;
87      Model cabeza2_M;
88      Model cabeza3_M;
89      Model cabeza4_M;
90      Model cabeza5_M;
91
92      //materiales
93      Material Material_brillante;
94      Material Material_opaco;
95
```

```
265
266
267   void CreateShaders()
268   {
269       Shader *shader1 = new Shader();
270       shader1->CreateFromFiles(vShader, fShader);
271       shaderList.push_back(*shader1);
272   }
273
274   //funcion senoidal para movimiento de cabezas
275   float RotacionSenoidal(float time, float amplitud, float frecuencia, float fase) {
276       return amplitud * sin(frecuencia * time + fase);
277   }
278
279
280
```

```
291     camera = Camera(glm::vec3(0.0f, 0.0f, 0.0f), glm::vec3(0.0f,
292
293     brickTexture = Texture("Textures/brick.png");
294     brickTexture.LoadTextureA();
295     dirtTexture = Texture("Textures/dirt.png");
296     dirtTexture.LoadTextureA();
297     plainTexture = Texture("Textures/plain.png");
298     plainTexture.LoadTextureA();
299     pisoTexture = Texture("Textures/piso.tga");
300     pisoTexture.LoadTextureA();
301     AgaveTexture = Texture("Textures/Agave.tga");
302     AgaveTexture.LoadTextureA();
303     FlechaTexture = Texture("Textures/flechas.tga");
304     FlechaTexture.LoadTextureA();
305     NumerosTexture = Texture("Textures/numerosbase.tga");
306     NumerosTexture.LoadTextureA();
307     Numero1Texture = Texture("Textures/numero1.tga");
308     Numero1Texture.LoadTextureA();
309     Numero2Texture = Texture("Textures/numero2.tga");
310     Numero2Texture.LoadTextureA();
311     blancoTexture = Texture("Textures/blanco.png");
312     blancoTexture.LoadTextureA();
313     letrasTexture = Texture("Textures/letrasCarte.tga");
314     letrasTexture.LoadTextureA();
315
316
317     Kitt_M = Model();
318     Kitt_M.LoadModel("Models/kitt_optimizado.obj");
        Llanta_M = Model();
```

```
Kitt_M = Model();
Kitt_M.LoadModel("Models/kitt_optimizado.obj");
Llanta_M = Model();
Llanta_M.LoadModel("Models/llanta_optimizada.obj");
cuerpoDragon_M = Model();
cuerpoDragon_M.LoadModel("Models/cuerpoDragon.obj");
alaDragonDerecha_M = Model();
alaDragonDerecha_M.LoadModel("Models/alaDragonDerecha.obj");
alaDragonIzquierda_M = Model();
alaDragonIzquierda_M.LoadModel("Models/alaDragonIzquierda.obj");
arco_M = Model();
arco_M.LoadModel("Models/arco.obj");
cartel_M = Model();
cartel_M.LoadModel("Models/cartel.obj");
cabeza1_M = Model();
cabeza1_M.LoadModel("Models/cabeza1.obj");
cabeza2_M = Model();
cabeza2_M.LoadModel("Models/cabeza2.obj");
cabeza3_M = Model();
cabeza3_M.LoadModel("Models/cabeza3.obj");
cabeza4_M = Model();
cabeza4_M.LoadModel("Models/cabeza4.obj");
cabeza5_M = Model();
cabeza5_M.LoadModel("Models/cabeza5.obj");
```

```
rotllantaOffset = 10.0f;

//para mov cabezas dragon con movimiento senoidal
float amplitud = 20.0f; // Amplitud de la rotación en grados
float frecuencia = 0.009f;  // Frecuencia de la rotación
float fase = 0.0f;      // Desfase de la rotación
float time = 0.0f;       // Tiempo inicial
float deltaTime = 0.01f;  // Incremento del tiempo
float angulo1;
float angulo2;
float angulo3;
float angulo4;
float angulo5;
float angIzq;
float angDer;
//cuerpo dragon movimiento adelante y atras
float amplitudCuerpo = 130.0f; // Amplitud del movimiento del dragon (cuerpo)
float frecCuerpo = 0.0002f; // Frecuencia del movimiento
float movX;
```

```
493     //Practica 9-2: Animacion Avanzada
494
495     //1.- .Hacer que en el arco que crearon se muestre la palabra: PROYECTO CGEIHC MONOPOLY. animado desplazándose
496     // las letras de izquierda a derecha como si fuera letrero LCD/LED de forma cíclica
497
498     //arco
499     model = glm::mat4(1.0);
500     model = glm::translate(model, glm::vec3(-40.0f, -0.7f, -5.0f));
501     modelaux = model;
502     model = glm::scale(model, glm::vec3(0.03f, 0.03f, 0.03f));
503     model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
504     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
505     blancoTexture.UseTexture();
506     arco_M.RenderModel();
507
508     //cartel
509     model = modelaux;
510     model = glm::translate(model, glm::vec3(-0.5f, 6.0f, 0.0f));
511     model = glm::scale(model, glm::vec3(0.03f, 0.03f, 0.03f));
512     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
513     blancoTexture.UseTexture();
514     cartel_M.RenderModel();
515
516     //letras
517
518     toffsetflechau += 0.001;
519     toffsetflechav = 0.000;
520     //para que no se desborde la variable
521     if (toffsetflechau > 1.0)
522         toffsetflechau = 0.0;
```

```
523     //if (toffsetv > 1.0)
524     //  toffsetv = 0;
525     //printf("\ntfosset %f \n", toffsetu);
526     //pasar a la variable uniform el valor actualizado
527     toffset = glm::vec2(toffsetflechau, toffsetflechav);
528
529
530     model = glm::mat4(1.0);
531     model = glm::translate(model, glm::vec3(-40.2f, 6.3f, -3.3f));
532     model = glm::rotate(model, 90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
533     model = glm::scale(model, glm::vec3(7.0f, 1.0f, 1.0f));
534     glUniform2fv(uniformTextureOffset, 1, glm::value_ptr(toffset));
535     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
536     letrasTexture.UseTexture();
537     /*Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);*/
538     meshList[4]->RenderMesh();
539
540
541     //2.- Separar las cabezas del Dragón y agregar las siguientes animaciones:
542     // °  Movimiento del cuerpo ida y vuelta.
543     // ° Aleteo
544     // ° Cada cabeza se mueve de forma diferente de acuerdo a una función/algoritmo
545     // diferente (ejemplos: espiral de Arquímedes, movimiento senoidal, lemniscata,  etc..)
546     // ° Cada cabeza debe de verse de un color diferente: roja, azul, verde,, blanco,café.
547
548     //cuerpo dragon
549     model = glm::mat4(1.0);
550     //model = glm::translate(model, glm::vec3(0.0 - angulovaria / 100, 6.0f + (5 * sin(glm::radians(angulovaria))),
551     movX = amplitudCuerpo * sin(frecCuerpo * time);
552     model = glm::translate(model, glm::vec3(0.0f + movX, 6.0f, 6.0f));
```

```
553         modelaux = model;
554         model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.4f));
555         /*model = glm::rotate(model, -180 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));*/
556         Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
557         /*color = glm::vec3(0.0f, 1.0f, 0.0f);
558         glUniform3fv(uniformColor, 1, glm::value_ptr(color));*/
559         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
560         cuerpoDragon_M.RenderModel();
561         time += deltaTime;
562
563
564         //ala izquierda
565         model = modelaux;
566         model = glm::translate(model, glm::vec3(0.0, 1.3f, 0.0));
567         angIzq = RotacionSenoidal(time, amplitud, frecuencia, fase);
568         model = glm::rotate(model, glm::radians(-angIzq), glm::vec3(1.0f, 0.0f, 0.0f));
569         model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.4f));
570         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
571         alaDragonIzquierda_M.RenderModel();
572         time += deltaTime;
573
574         //ala derecha
575         model = modelaux;
576         model = glm::translate(model, glm::vec3(0.0, 1.3f, 0.0));
577         angDer = RotacionSenoidal(time, amplitud, frecuencia, fase);
578         model = glm::rotate(model, glm::radians(angDer), glm::vec3(1.0f, 0.0f, 0.0f));
579         model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.4f));
580         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
581         alaDragonDerecha_M.RenderModel();
582         time += deltaTime;
```

```
583
584         //cabeza 1 azul (de derecha a izquierda)
585         model = modelaux;
586         model = glm::translate(model, glm::vec3(-1.0, 0.86f, 0.14));
587         angulo1 = RotacionSenoidal(time, amplitud, frecuencia, fase);
588         model = glm::rotate(model, glm::radians(angulo1), glm::vec3(0.0f, 1.0f, 0.0f));
589         model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.4f));
590         color = glm::vec3(0.60f, 0.93f, 0.87f);
591         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
592         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
593         cabeza1_M.RenderModel();
594         time += deltaTime;
595
596
597         //cabeza 2 blanca
598         model = modelaux;
599         model = glm::translate(model, glm::vec3(-1.0, 0.87f, -0.15));
600         angulo2 = RotacionSenoidal(time, amplitud, frecuencia, fase);
601         model = glm::rotate(model, glm::radians(angulo2), glm::vec3(1.0f, 0.0f, 0.0f));
602         model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.4f));
603         color = glm::vec3(1.0f, 1.0f, 1.0f);
604         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
605         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
606         cabeza2_M.RenderModel();
607         time += deltaTime;
608
609
610         //cabeza 3 cafe
611         model = modelaux;
612         model = glm::translate(model, glm::vec3(-0.96, 1.48f, 0.279));
613         angulo3 = RotacionSenoidal(time, amplitud, frecuencia, fase);
```

No se encontraron problemas

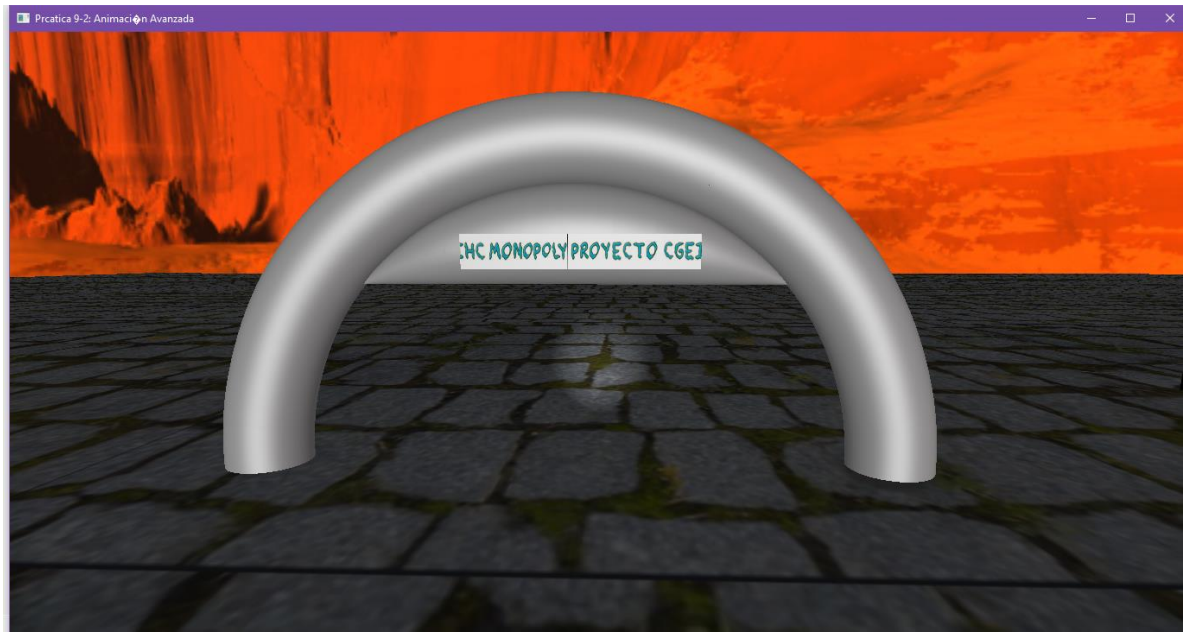
Línea: 531

```
613         angulo3 = RotacionSenoidal(time, amplitud, frecuencia, fase);
614         model = glm::rotate(model, glm::radians(angulo3), glm::vec3(0.0f, 0.0f, 1.0f));
615         model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.4f));
616         color = glm::vec3(0.59f, 0.37f, 0.10f);
617         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
618         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
619         cabeza3_M.RenderModel();
620         time += deltaTime;
621
622         // cabeza 4 verde
623         model = modelaux;
624         model = glm::translate(model, glm::vec3(-0.97, 1.47f, -0.44));
625         angulo4 = RotacionSenoidal(time, amplitud, frecuencia, fase);
626         model = glm::rotate(model, glm::radians(angulo4), glm::vec3(0.0f, 1.0f, 0.0f));
627         model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.4f));
628         color = glm::vec3(0.56f, 0.95f, 0.30f);
629         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
630         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
631         cabeza4_M.RenderModel();
632         time += deltaTime;
633
634         //cabeza 5 roja
635         model = modelaux;
636         model = glm::translate(model, glm::vec3(-1.12f, 0.85f, -0.4));
637         angulo5 = RotacionSenoidal(time, amplitud, frecuencia, fase);
638         model = glm::rotate(model, glm::radians(angulo5), glm::vec3(0.0f, 1.0f, 0.0f));
639         model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.4f));
640         color = glm::vec3(0.97f, 0.28f, 0.22f);
641         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
642         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
```

```
632         time += deltaTime;
633
634         //cabeza 5 roja
635         model = modelaux;
636         model = glm::translate(model, glm::vec3(-1.12f, 0.85f, -0.4));
637         angulo5 = RotacionSenoidal(time, amplitud, frecuencia, fase);
638         model = glm::rotate(model, glm::radians(angulo5), glm::vec3(0.0f, 1.0f, 0.0f));
639         model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.4f));
640         color = glm::vec3(0.97f, 0.28f, 0.22f);
641         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
642         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
643         cabeza5_M.RenderModel();
644         time += deltaTime;
645
646
```

Ejecución del programa


Prcatica 9-2: Animaci?n Avanzada


Prcatica 9-2: Animaci?n Avanzada
HC MONOPOLY PROYECTO CGEJ

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.

- Uno de los primero problemas que presente fue al momento de texturizar la imagen donde venían las letras correspondientes para poder hacer el cartel, ya que tuve algunos problemas con la forma en como se exporto el archivo pero logre realizarlo.

3.- Conclusión:

a. Los ejercicios del reporte: Complejidad, Explicación.
   Después de haber realizado los ejercicios propuestos para esta práctica puedo concluir que logré realizarlos todos con éxito, ya que comprendí la manera de implementar la animación avanzada a los objetos mediante la respectiva implementación del código y funciones, lo que hicimos durante estos ejercicios fue primero para el caso del cártel se debía texturizar la imagen con las letras del universo que elegimos y hacer que se viera de forma cíclica leyéndose de manera de izquierda a derecha la palabra respectiva en el cartel, después con el segundo ejercicio lo primero que se hizo fue separar las 5 cabezas del dragón y acomodarlas en su respectivo lugar mediante el código en OpenGL, después se implementó la función senoidal para lograr darle movimiento tanto a la cabeza como a las alas y el movimiento de ida y vuelta del dragón, por último se agregaron los colores respectivos a cada una de las cabezas siendo para la primera el color azul, la segunda blanca, la tercera café la cuarta verde y la quinta roja.

La complejidad de los ejercicios para el movimiento del dragón si tuvo algo de dificultad, pero logré realizarlo, pero para el caso del cartel con las letras en movimiento considero que así fue más complicado ya que no supe cómo hacerlo mediante la forma correcta qué mencionó el profesor.

b. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica
   Por mi parte puedo concluir que en el caso de esta práctica me quedaron varias dudas, quizá falto comentar un poco más acerca de la parte para el cartel y como se debía exportar correctamente la imagen.

c. Conclusión
   Después de haber concluido con la práctica puedo mencionar que se cumplió al 100% los objetivos propuestos, debido a que, comprendí el uso correcto de la animación avanzada y el uso de funciones en nuestros modelos y cómo es que se generan, mediante las actividades propuestas pude poner en práctica los conocimientos que he aprendido durante el desarrollo del laboratorio y así cumplir con la mayoría de los ejercicios planteados.