


| | |
|---|--|
|  | Manual de Prácticas |
| Secretaría/División: División de Ingeniería Eléctrica | Área/Departamento: Ingeniería en Computación |


Laboratorio de Computación Gráfica e Interacción Humano Computadora

Animación

N° de practica: 09

| | | |
|---------------------------------------|---|------------------|
| Nombre completo de los alumnos | | Firma |
| Arroyo Llanes Miguel Alejandro | | |
| N° de brigada: 2 | Fecha de ejecución: 26/04/2025 | Grupo: 03 |
| Calificación: | Profesor: Ing. Jose Roque Roman Guadarrama | |

| | | | |
|-----------------------|----------------------|------------------------|-----------------------|
| Elaborado por: | Revisado por: | Autorizado por: | Vigente desde: |
|-----------------------|----------------------|------------------------|-----------------------|

| | |
|---|--|
|  | Manual de Prácticas |
| Secretaría/División: División de Ingeniería Eléctrica | Área/Departamento: Ingeniería en Computación |

Comentario:

En esta práctica se reforzaron los conocimientos sobre modelado y texturizado, aplicándolos en la creación de animaciones, enfocándome específicamente en la animación de letras utilizando la tipografía proporcionada.

Para animar las letras, reutilicé una idea descartada de ejercicios anteriores, adaptándola para simular un letrero de entrada con movimiento deslizable. Además, edité el modelo del dragón, agregando texturas y separando las cabezas del cuerpo. También eliminé la base que tenía originalmente para lograr una apariencia más acorde a un dragón en vuelo.

Posteriormente, investigué distintos tipos de movimiento como el senoidal (onda simple y doble), espiral de Arquímedes, lemniscata y movimiento circular. Con esto, programé los desplazamientos de cada cabeza asegurándome de que no se alejaran demasiado de su posición base, manteniendo así coherencia y naturalidad en la animación.

Finalmente, buscando añadir funcionalidades extra, implementé una animación de alientos para cada cabeza al presionar la tecla 'E'. Aunque no logré vincular directamente las texturas de aliento al modelo del dragón, resolví este problema sincronizando manualmente el movimiento de los alientos con el cuerpo del dragón. La lógica programada garantiza que los alientos sólo sean visibles mientras se mantiene presionada la tecla 'E'.


Bloque de código:

Controles de activar el aliento en el window.h.

```

20  // ...
21
22  GLfloat getaliento() { return aliento; }
23
24  ~Window();
25  private:
26  GLFWwindow *mainWindow;
27  GLint width, height;
28
29  GLfloat aliento;

```

| | |
|---|--|
|  | Manual de Prácticas |
| Secretaría/División: División de Ingeniería Eléctrica | Área/Departamento: Ingeniería en Computación |

Controles de activar el aliento en el window.cpp.

```

11     }
12     Window::Window(GLint windowHeight, GLint windowHeight)
13     {
14         width = windowHeight;
15         height = windowHeight;
16
17         aliento = 0.0f;
18
119
120         if (key == GLFW_KEY_E && action == GLFW_PRESS)
121         {
122             theWindow->aliento = 1;
123         }
124         else if (key == GLFW_KEY_E && action == GLFW_RELEASE)
125         {
126             theWindow->aliento = 0;
127         }
128

```

Importar dragón y texturas.

```


88
89     //Tipografia
90     Texture HoraDeAventuraTipografia;
91
92     //Aliento dragones
93     Texture FireBreathTexture;
94     Texture LightningBreathTexture;
95     Texture PoisonBreathTexture;
96     Texture IceBreathTexture;
97     Texture SandBreathTexture;
98

```

```

103
104     //Dragon
105     Model BODY_DRAGON;
106     Model WING_L;
107     Model WING_R;
108     Model HEAD1;
109     Model HEAD2;
110     Model HEAD3;
111     Model HEAD4;
112     Model HEAD5;
113
114     //Arco
115     Model ARCO;
116

```

| | |
|---|--|
|  | Manual de Prácticas |
| Secretaría/División: División de Ingeniería Eléctrica | Área/Departamento: Ingeniería en Computación |

```


330      //Tipografia
331      HoraDeAventuraTipografia = Texture("Textures/HoraDeAventura.png");
332      HoraDeAventuraTipografia.LoadTextureA();
333
334      //Aliento dragones
335      FireBreathTexture = Texture("Textures/FireBreathTexture.png");
336      FireBreathTexture.LoadTextureA();
337      LightningBreathTexture = Texture("Textures/LightningBreathTexture.png");
338      LightningBreathTexture.LoadTextureA();
339      PoisonBreathTexture = Texture("Textures/PoisonBreathTexture.png");
340      PoisonBreathTexture.LoadTextureA();
341      IceBreathTexture = Texture("Textures/IceBreathTexture.png");
342      IceBreathTexture.LoadTextureA();
343      SandBreathTexture = Texture("Textures/SandBreathTexture.png");
344      SandBreathTexture.LoadTextureA();
345

```

```

350      //Dragon
351      Dragon_M = Model();
352      Dragon_M.LoadModel("Models/BODY_DRAGON.obj");
353      WING_L = Model();
354      WING_L.LoadModel("Models/WING_L.obj");
355      WING_R = Model();
356      WING_R.LoadModel("Models/WING_R.obj");
357      HEAD1 = Model();
358      HEAD1.LoadModel("Models/HEAD1.obj");
359      HEAD2 = Model();
360      HEAD2.LoadModel("Models/HEAD2.obj");
361      HEAD3 = Model();
362      HEAD3.LoadModel("Models/HEAD3.obj");
363      HEAD4 = Model();
364      HEAD4.LoadModel("Models/HEAD4.obj");
365      HEAD5 = Model();
366      HEAD5.LoadModel("Models/HEAD5.obj");
367
368      ARCO = Model();
369      ARCO.LoadModel("Models/ARCO.obj");
370

```

| | |
|---|--|
|  | Manual de Prácticas |
| Secretaría/División: División de Ingeniería Eléctrica | Área/Departamento: Ingeniería en Computación |

Creación del modelo del dragón, arco y las texturas animadas.

```


560 model = glm::mat4(1.0);
561 model = glm::translate(model, glm::vec3(0.0f, 5.0f+sin(glm::radians(anguloVaria)), 6.0));
562 model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
563 model = glm::rotate(model, -90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
564 //Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
565 //color = glm::vec3(0.0f, 1.0f, 0.0f);
566 glUniform3fv(uniformColor, 1, glm::value_ptr(color));*/
567 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
568 Dragon_M.RenderModel();
569
570
571 // Animación de aleteo
572 float anguloAla = cos(glm::radians(glm::getTime() * 5.0f)) * 20.0f; // de -20 a 20 grados
573
574 // Ala izquierda (X negativo)
575 glm::mat4 modelAlaL = model;
576 modelAlaL = glm::translate(modelAlaL, glm::vec3(-0.8f, -0.5f, 3.3f)); // mover al "hombro"
577 modelAlaL = glm::rotate(modelAlaL, glm::radians(anguloAla), glm::vec3(1.0f, 0.0f, 0.0f));
578 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelAlaL));
579 WING_L.RenderModel();
580
581 // Ala derecha (X positivo)
582 glm::mat4 modelAlaR = model;
583 modelAlaR = glm::translate(modelAlaR, glm::vec3(0.8f, -0.5f, 3.3f));
584 modelAlaR = glm::rotate(modelAlaR, glm::radians(-anguloAla), glm::vec3(1.0f, 0.0f, 0.0f));
585 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelAlaR));
586 WING_R.RenderModel();
587
588 float tiempo = glm::getTime();
589

```

```

590 //Cabeza 1
591 // Cabeza 1 - Movimiento senoidal
592 glm::mat4 modelHEAD1 = model;
593 modelHEAD1 = glm::translate(modelHEAD1, glm::vec3(-2.5f, -0.3f + sin(tiempo) * 0.1f, 2.5f));
594 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelHEAD1));
595 HEAD1.RenderModel();
596
597 //Cabeza 2 - Espiral de Arquimedes
598 glm::mat4 modelHEAD2 = model;
599 modelHEAD2 = glm::translate(modelHEAD2, glm::vec3(-2.0f + (tiempo * 0.01f) * cos(tiempo), -0.3f + (tiempo * 0.01f) * sin(tiempo), 3.8f));
600 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelHEAD2));
601 HEAD2.RenderModel();
602
603 //Cabeza 3 - Movimiento lemniscata
604 float x3 = (0.2 * cos(tiempo)) / (1 + sin(tiempo) * sin(tiempo));
605 float y3 = (0.2 * cos(tiempo) * sin(tiempo)) / (1 + sin(tiempo) * sin(tiempo));
606 glm::mat4 modelHEAD3 = model;
607 modelHEAD3 = glm::translate(modelHEAD3, glm::vec3(-2.1f + x3, 0.85f + y3, 3.7f));
608 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelHEAD3));
609 HEAD3.RenderModel();
610
611 //Cabeza 4 - Movimiento senoidal (onda doble)
612 glm::mat4 modelHEAD4 = model;
613 modelHEAD4 = glm::translate(modelHEAD4, glm::vec3(-2.35f, 1.1f + sin(tiempo * 2.0f) * 0.1f, 2.6f));
614 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelHEAD4));
615 HEAD4.RenderModel();
616
617 //Cabeza 5 - Movimiento circular
618 glm::mat4 modelHEAD5 = model;
619 modelHEAD5 = glm::translate(modelHEAD5, glm::vec3(-2.6f + cos(tiempo) * 0.2f, 0.16f + sin(tiempo) * 0.2f, 2.9f));
620 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelHEAD5));
621 HEAD5.RenderModel();

```

| | |
|---|--|
|  | Manual de Prácticas |
| Secretaría/División: División de Ingeniería Eléctrica | Área/Departamento: Ingeniería en Computación |

```


624     model = glm::mat4(1.0);
625     modelaux = model;
626     model = glm::scale(model, glm::vec3(5.5f, 5.5f, 5.5f));
627     model = glm::translate(model, glm::vec3(0.0f, 0.64f, 6.0f));
628     model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
629     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
630     ARCO.RenderModel();

```

```

797     toffsetpalabracambiau += 0.0003;
798     if (toffsetpalabracambiau > 1.0)
799         toffsetpalabracambiau = 0.0;
800     toffsetpalabrav = 0.0;
801     toffset = glm::vec2(toffsetpalabracambiau, toffsetpalabrav);
802
803     model = glm::mat4(1.0);
804     model = glm::translate(model, glm::vec3(0.0f, 7.6f, 31.77f));
805     model = glm::scale(model, glm::vec3(4.0f, 0.7f, 2.0f));
806     model = glm::rotate(model, 90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
807     model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
808
809     glUniform2fv(uniformTextureOffset, 1, glm::value_ptr(toffset));
810     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
811     color = glm::vec3(1.0f, 1.0f, 1.0f);
812     glUniform3fv(uniformColor, 1, glm::value_ptr(color));
813
814     // Dibuja la tipografía
815     HoraDeAventuraTipografia.UseTexture();
816     meshList[5]->RenderMesh();

```

| | |
|---|--|
|  | Manual de Prácticas |
| Secretaría/División: División de Ingeniería Eléctrica | Área/Departamento: Ingeniería en Computación |

```


818 if (mainWindow.getaliento() == 1)
819 {
820     glm::vec2 toffsetFijo = glm::vec2(0.0f, 0.0f); //Para que no siga la animación de las letras
821     glUniform2fv(uniformTextureOffset, 1, glm::value_ptr(toffsetFijo));
822     float animacion = sin(glm::radians(angulovariable));
823
824     //Aliento de rayo
825     model = glm::mat4(1.0);
826     model = glm::translate(model, glm::vec3(-2.4f, 5.1f + animacion, 7.4));
827     model = glm::rotate(model, 90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
828     model = glm::rotate(model, 140 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
829     model = glm::rotate(model, -20 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
830     model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
831     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
832     LightningBreathTexture.UseTexture();
833     meshList[5]-->RenderMesh();
834
835     //Aliento de hielo
836     model = glm::mat4(1.0);
837     model = glm::translate(model, glm::vec3(-1.2f, 6.8f + animacion, 7.2));
838     model = glm::rotate(model, 90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
839     model = glm::rotate(model, 110 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
840     model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
841     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
842     IceBreathTexture.UseTexture();
843     meshList[5]-->RenderMesh();

```

```


844
845     //Aliento de arena
846     model = glm::mat4(1.0);
847     model = glm::translate(model, glm::vec3(-1.6f, 6.8f + animacion, 5.0));
848     model = glm::rotate(model, 90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
849     model = glm::rotate(model, 210 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
850     model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
851     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
852     SandBreathTexture.UseTexture();
853     meshList[5]-->RenderMesh();
854
855     // Aliento de fuego
856     model = glm::mat4(1.0);
857     model = glm::translate(model, glm::vec3(-2.8f, 6.0f + animacion, 4.8));
858     model = glm::rotate(model, 90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
859     model = glm::rotate(model, 200 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
860     model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
861     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
862     FireBreathTexture.UseTexture();
863     meshList[5]-->RenderMesh();
864
865     //Aliento de veneno
866     model = glm::mat4(1.0);
867     model = glm::translate(model, glm::vec3(-2.0f, 6.4f + animacion, 5.7));
868     model = glm::rotate(model, 90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
869     model = glm::rotate(model, 200 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
870     model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
871     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
872     PoisonBreathTexture.UseTexture();
873     meshList[5]-->RenderMesh();
874 }

```


| | |
|---|--|
|  | Manual de Prácticas |
| Secretaría/División: División de Ingeniería Eléctrica | Área/Departamento: Ingeniería en Computación |

Ejecución:



| | |
|---|--|
|  | Manual de Prácticas |
| Secretaría/División: División de Ingeniería Eléctrica | Área/Departamento: Ingeniería en Computación |

Conclusión:

En esta práctica puse a prueba los conocimientos previamente adquiridos, notando un gran avance en la eficiencia al momento de texturizar modelos. Por otro lado, la animación sigue representando un reto para mí, principalmente debido a la falta de práctica, aunque reconozco que he logrado avances importantes en su comprensión. A pesar de las dificultades, me siento satisfecho con el progreso obtenido y considero que, con las habilidades desarrolladas hasta ahora, estoy preparado para afrontar mi proyecto final de manera satisfactoria.

Además, la versatilidad de soluciones prácticas ante problemas imprevistos, como fue el caso de la implementación de los alientos, me demuestra un crecimiento en mi formación como futuro ingeniero en software, al fortalecer mi capacidad de adaptación y resolución de problemas.

Bibliografía:

1. Tailor, P. (2022, 21 mayo). A Visual Guide to Archimedean Spirals and its Special Cases. Medium. <https://medium.com/math-simplified/a-visual-guide-to-archimedean-spirals-and-its-special-cases-5173966800bf>
2. Intuitive Understanding of Sine Waves – BetterExplained. (s. f.). https://betterexplained.com/articles/intuitive-understanding-of-sine-waves/?utm_source=chatgpt.com
3. Bourne, M. (s. f.). Lemniscate of Bernoulli. <https://www.intmath.com/historical-math-curves/lemniscate-bernoulli.php>
4. How can I move an object in an «infinity» or «figure 8» trajectory? (s. f.). Game Development Stack Exchange. https://gamedev.stackexchange.com/questions/43691/how-can-i-move-an-object-in-an-infinity-or-figure-8-trajectory?utm_source=chatgpt.com
5. Khan Academy. (s. f.). https://www.khanacademy.org/science/electrical-engineering/ee-circuit-analysis-topic/ee-ac-analysis/v/ee-sine-cosine-from-rotating-vector?utm_source=chatgpt.com