

Introducción

En esta previo trabajaremos con el concepto de máquina de estados e implementaremos animaciones en open GL.

Donde una máquina de estados nos permite controlar y estructurar el comportamiento de las animaciones según distintas condiciones, en este caso vamos a animar un objeto 3D, usaremos el perrito que hemos estado utilizando en los ejemplos anteriores.

Implementaremos varios estados como: caminar, y dentro del estado de caminar estará moviendo cada una de sus extremidades, la cabeza, la cola y las patas.

Posteriormente vamos a ver como se puede alternar entre cada uno de estos estados.

¿Qué es una máquina de estado?

Esto es el modelo del comportamiento de un objeto y esto nos indica cuando el objeto esta en algún estado predefinido y los cambios que vamos a tener entre cada uno de ellos según las condiciones o eventos que sucedan.

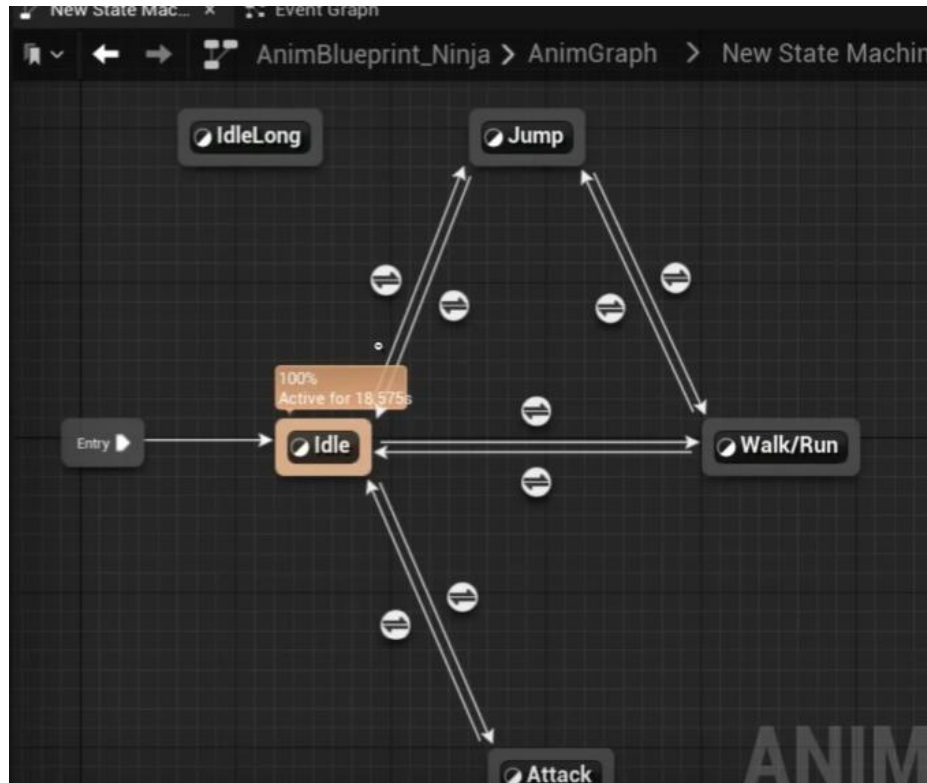
Ejemplo Video- Juegos.



Previo 11 Animación por máquina de estados
Fecha de entrega 10/28/2025

Alumna: Plata Alvarez Dayana
No. De cuenta: 318300764

Se utilizan las máquinas de estados para tener visibilidad de cuando esta quieto o se esta moviendo y cuando se esta intercambiando entre estos estados, se observa de la siguiente manera:



Puede cambiar al estado de saltar, o el de caminar, correr o atacar entre otros.

Y en este ejercicio, lo vamos a ver representado en como vamos a controlar cada parte del cuerpo del perrito.



Previo 11 Animación por máquina de estados
Fecha de entrega 10/28/2025

Alumna: Plata Alvarez Dayana
No. De cuenta: 318300764

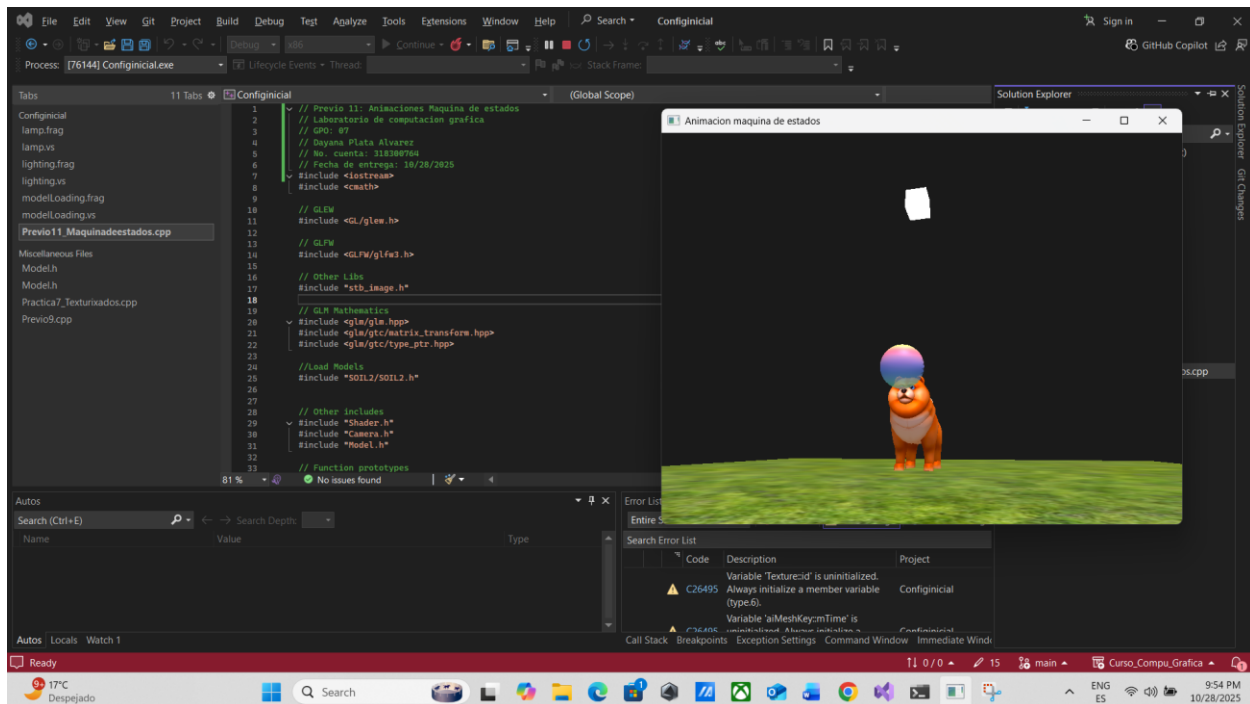
Lo que se tuvo que hacer, es adecuar el modelo para poderlo manipular por partes, en el ejercicio anterior teníamos el modelo como un solo objeto y aquí lo que se realizó fue el poderlo separar para que se pueda manipular cada uno de sus elementos por separado, y también se modificaron los pivotes para que los podamos utilizar de manera correspondiente, para que pueda caminar simulando que mueve un poco las patas, cabeza, y cola.

Posteriormente descargamos el material compartido por el profesor,

Maquina de estados.zip 2 elementos

Nombre	Última modificación	Tamaño del arc...
Models	-	-
Maquina de estados.cpp	-	-

y ejecutamos el código y obtenemos lo siguiente:



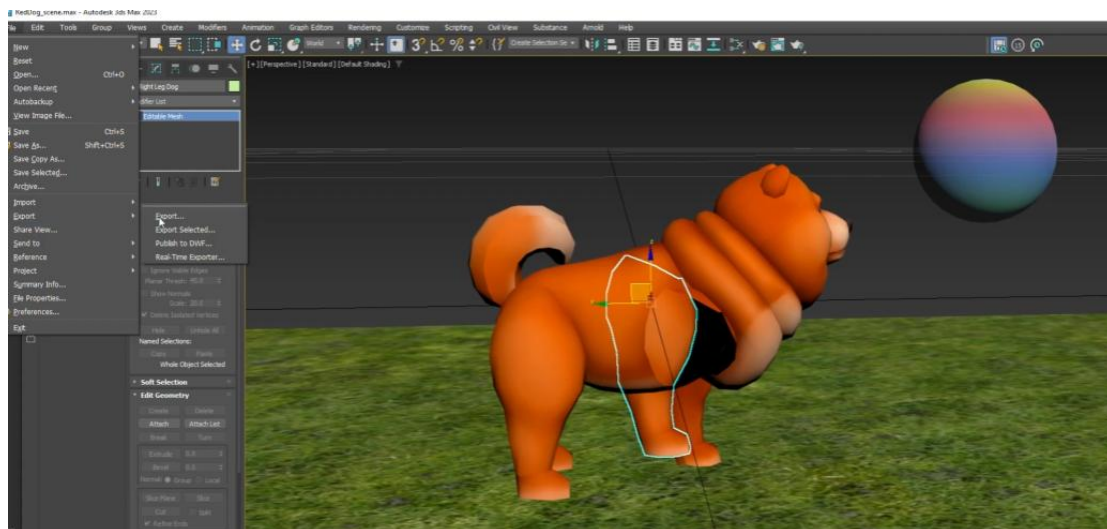
En un inicio no vamos a tener un cambio visible con respecto al ejercicio anterior de la animación básica, sin embargo, como ya tenemos el modelo por separado ya lo tenemos listo para poder ajustar y manipular con este tipo de animación.

Lo que se agregó fueron variables como se vio en el ejercicio anterior, estas variables nos van a permitir integrar y manipular cada uno de los elementos que nosotros vamos a integrar en el escenario 3D, en este caso tenemos de tipo boleano para saber las animaciones, tenemos de tipo flotante para controlar las rotaciones y las posiciones, en este caso para las posiciones del modelo por completo tenemos un elemento de tipo vector 3, para poder controlar la posición en x, y, z.

```
glm::vec3 Light1 = glm::vec3(0);  
//Anim  
float rotBall = 0.0f;  
bool AnimBall = false;  
bool AnimDog = false;  
float rotDog = 0.0f;  
int dogAnim = 0;  
float FLegs = 0.0f;  
float RLegs = 0.0f;  
float head = 0.0f;  
float tail = 0.0f;  
glm::vec3 dogPos (0.0f,0.0f,0.0f);  
float dogRot = 0.0f;  
bool step = false;
```

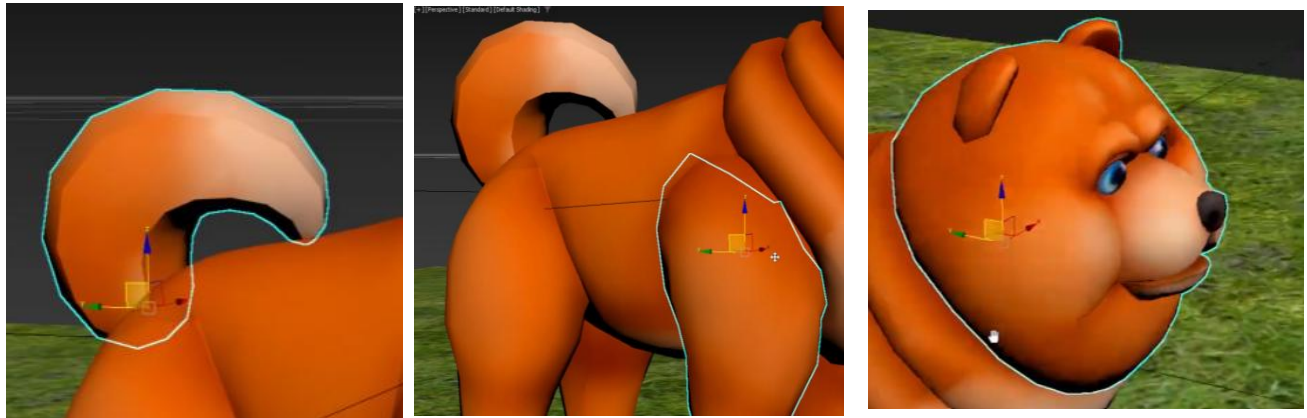
A su vez en la sección de carga de modelos lo que tenemos es cada una de las piezas por separado, desde el cuerpo, la cabeza, cada una de las patas, para poderla implementar.

Nota: Es importante recalcar que para que el modelo se pueda implementar correctamente, lo que se tiene que hacer previamente a la exportación es colocar cada uno de nuestros elementos que en este caso son las extremidades en el origen, es decir, yo cuando tengo aquí mi modelo lo ponemos en 0,0,0 y de esta manera lo exportamos.



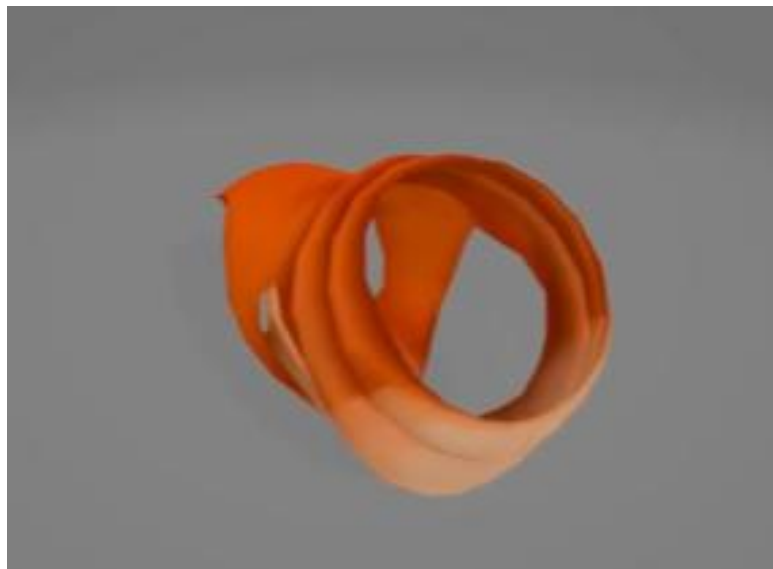
Posteriormente en el proyecto de visual studio lo tenemos que posicionar con esos parámetros, para que las coloquen nuevamente en la posición correcta y los pivotes funcionen correctamente

Aquí podemos observar que tenemos un pivote por cada sección del perrito, tanto para las patas, cola, y cabeza.



Si no funciona de esta manera, será muy difícil que podamos controlarlo con respecto a las animaciones, igual con la cabeza, **antes de exportarla se mandó al origen se exporta, y una vez estando en visual estudio se le colocan los valores correspondientes** que tenemos en el código.

Esto quiere decir que cuando vemos el modelo los tenemos por separado, pero todos están referenciados en el origen.



Y cargamos cada uno de los elementos por separado del modelo.

```
//models
Model DogBody((char*)"Models/DogBody.obj");
Model HeadDog((char*)"Models/HeadDog.obj");
Model DogTail((char*)"Models/TailDog.obj");
Model F_RightLeg((char*)"Models/F_RightLegDog.obj");
Model F_LeftLeg((char*)"Models/F_LeftLegDog.obj");
Model B_RightLeg((char*)"Models/B_RightLegDog.obj");
Model B_LeftLeg((char*)"Models/B_LeftLegDog.obj");
Model Piso((char*)"Models/piso.obj");
Model Ball((char*)"Models/ball.obj");
```

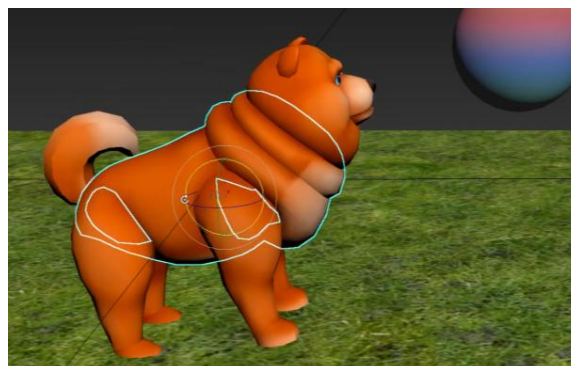
En la sección de dibujo vamos a implementar o utilizar nuevamente nuestra función de animación.

Aquí podemos observar que comienza la carga de modelos:

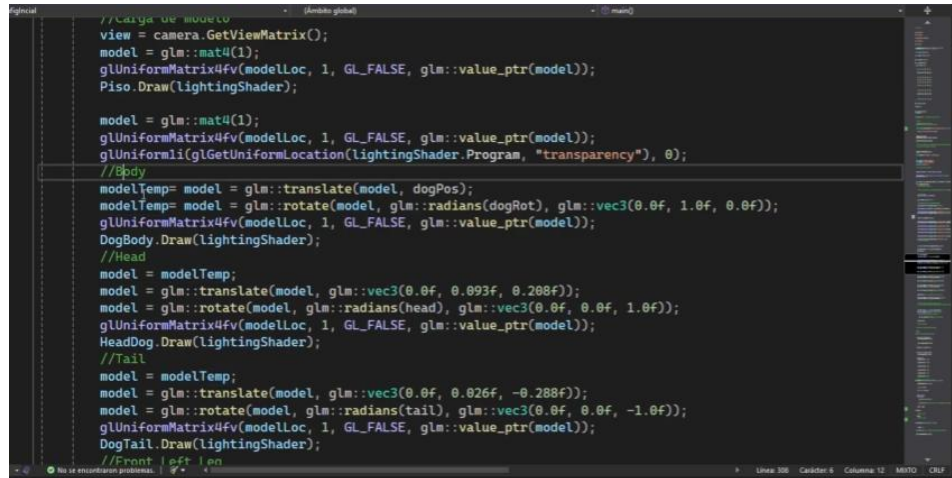
```
//Carga de modelo
view = camera.GetViewMatrix();
model = glm::mat4(1);
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
//Body
modelTemp= model = glm::translate(model, dogPos);
modelTemp= model = glm::rotate(model, glm::radians(dogRot), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
DogBody.Draw(LightingShader);
//Head
model = modelTemp;
model = glm::translate(model, glm::vec3(0.0f, 0.093f, 0.208f));
model = glm::rotate(model, glm::radians(head), glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
HeadDog.Draw(LightingShader);
```

Jerarquía del objeto

El elemento padre de todos será el cuerpo, ya que, si este lo manipulamos, todos los elementos se tienen que mover junto con él.



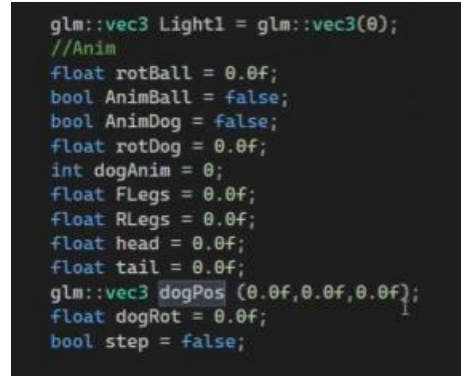
Y de ahí cada una de las extremidades serian los elementos hijos, respetando el mismo nivel de jerarquía. Para lograr eso dentro de visual studio establecemos el orden que en este caso primero va a ser el cuerpo. Aquí recordaremos lo que se hizo en la práctica de modelado jerárquico.



```
//Carga de modelo
view = camera.GetViewMatrix();
model = glm::mat4(1);
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Piso.Draw(LightingShader);

modelTemp = glm::mat4(1);
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1i(glGetUniformLocation(LightingShader.Program, "transparency"), 0);
//Body
modelTemp = glm::translate(model, dogPos);
modelTemp = glm::rotate(modelTemp, glm::radians(dogRot), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(modelTemp));
DogBody.Draw(LightingShader);
//Head
model = modelTemp;
model = glm::translate(model, glm::vec3(0.0f, 0.093f, 0.288f));
model = glm::rotate(model, glm::radians(head), glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
HeadDog.Draw(LightingShader);
//Tail
model = modelTemp;
model = glm::translate(model, glm::vec3(0.0f, 0.026f, -0.288f));
model = glm::rotate(model, glm::radians(tail), glm::vec3(0.0f, 0.0f, -1.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
DogTail.Draw(LightingShader);
//Front left leg
```

Para establecer este orden y vamos a utilizar un concepto que es la matriz temporal ModelTemp, donde aquí la traslación que le vamos a dar es justamente la variable de la posición que hemos creado en este caso es este vector:



```
glm::vec3 Light1 = glm::vec3(0);
//Anim
float rotBall = 0.0f;
bool AnimBall = false;
bool AnimDog = false;
float rotDog = 0.0f;
int dogAnim = 0;
float FLegs = 0.0f;
float RLegs = 0.0f;
float head = 0.0f;
float tail = 0.0f;
glm::vec3 dogPos (0.0f,0.0f,0.0f);
float dogRot = 0.0f;
bool step = false;
```

Que en este momento esta inicializado en 0,0,0, quiere decir que se encuentra en el origen.

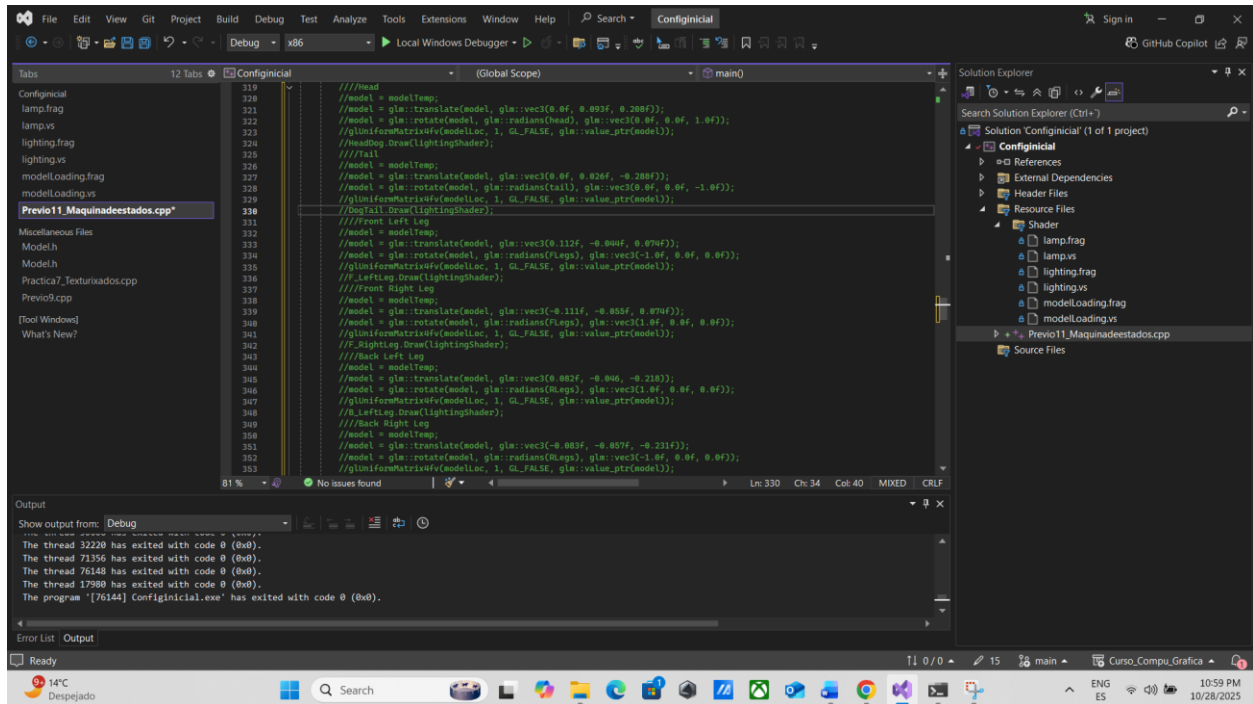
Posteriormente vamos a guardar la rotación en esta matriz temporal que esta denotada para la variable que aquí denotamos para el perro.

Mandamos la información al shader y cargamos nuestro primer elemento que es la parte del cuerpo.

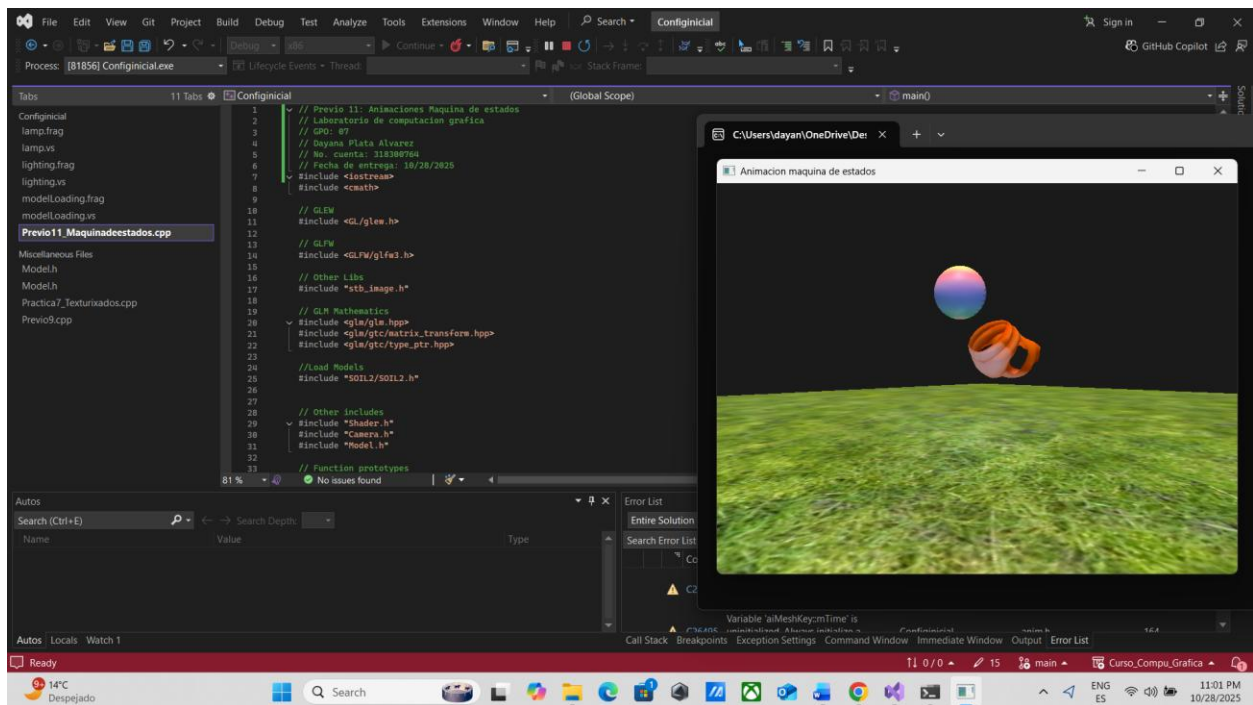
Para comenzar con el proceso comentamos la sección.

Previo 11 Animación por máquina de estados
Fecha de entrega 10/28/2025

Alumna: Plata Alvarez Dayana
No. De cuenta: 318300764



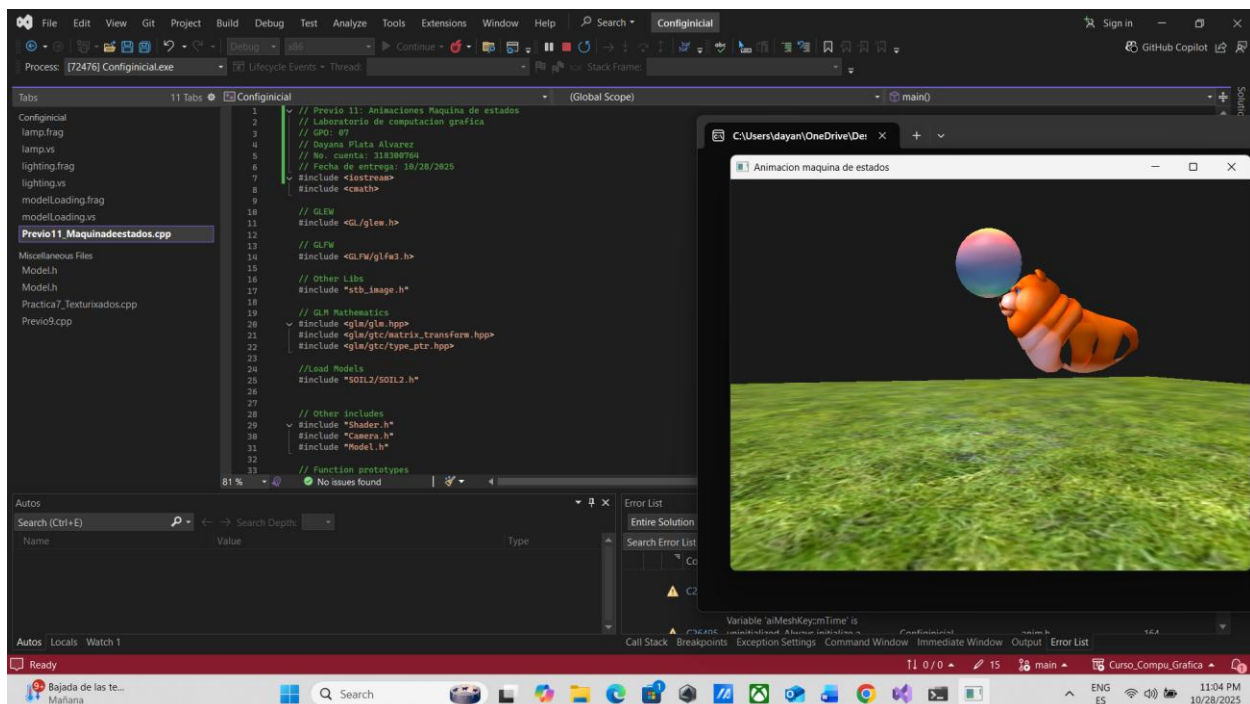
Y si ejecutamos el código podemos observar que solo el cuerpo se logra observar en el origen, donde este es el elemento principal de la jerarquía.



Previo 11 Animación por máquina de estados
Fecha de entrega 10/28/2025

Alumna: Plata Alvarez Dayana
No. De cuenta: 318300764

Ahora si descomentamos la sección de la cabeza obtenemos el siguiente elemento de la jerarquía, donde regresamos nuestro matriz de modelo a nuestro modelo temporal. Esto lo hacemos ya que si hacemos cambios en estas variables, también se verán afectados por los modelos hijos que en este caso vamos a comenzar por la cabeza, y aplicamos el valor correspondiente en este caso en tridi-max ya que este software ya nos da los valores, el sistema de referencia cambia en visual el elemento vertical es la componente z y la componente de profundidad es la componente y a diferencia de lo que hemos trabajado en visual studio, la componente de la altura o el eje vertical es y la profundidad es en z. por lo tanto los valores que tenemos en y en tridimax en visual studio lo colocamos en z y el elemento que es en z lo colocamos en y.



Ya que si lo lo dejamos en el origen se sobreponen los elementos de la siguiente manera:



Realizamos lo mismo para los demás elementos del perrito.

Ahora lo que se coloca posterior a la traslación de cada extremidad es una rotación, esta rotación nos va a permitir poder manipular los elementos con respecto a la animación que queramos generar, y aquí es donde utilizamos las variables que se han creado en esta sección una me va a permitir controlar las extremidades de las patas traseras y delanteras de la cabeza y de la cola y es lo que yo tengo aquí dentro de cada una de las rotaciones dependiendo de como quiero generar el movimiento en este caso para la cabeza va a afectar la rotación en el eje z para la cola en el eje z pero en -1 es decir en sentido contrario. Para las piernas o patas delanteras afectara en el eje x en sentido contrario -1, y para las patas traseras igual en el eje x en sentido opuestos.

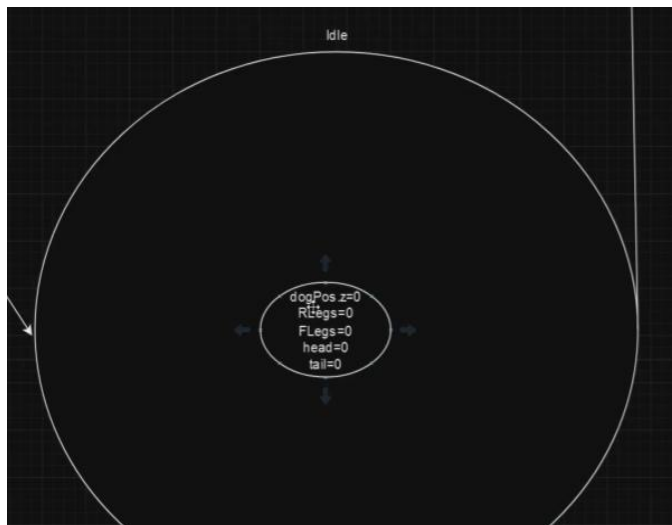
Esto nos permitirá hacer el apartado de movimiento y lo primero que vamos a generar es como si el modelo estuviera caminando.

Modelo caminando

Tenemos que hacer una traslación y aplicamos rotaciones de cada una de las extremidades, y la cola y la cabeza se van a mover de forma lateral y se terminara sincronizando en el momento en que activemos la animación.

Diagrama

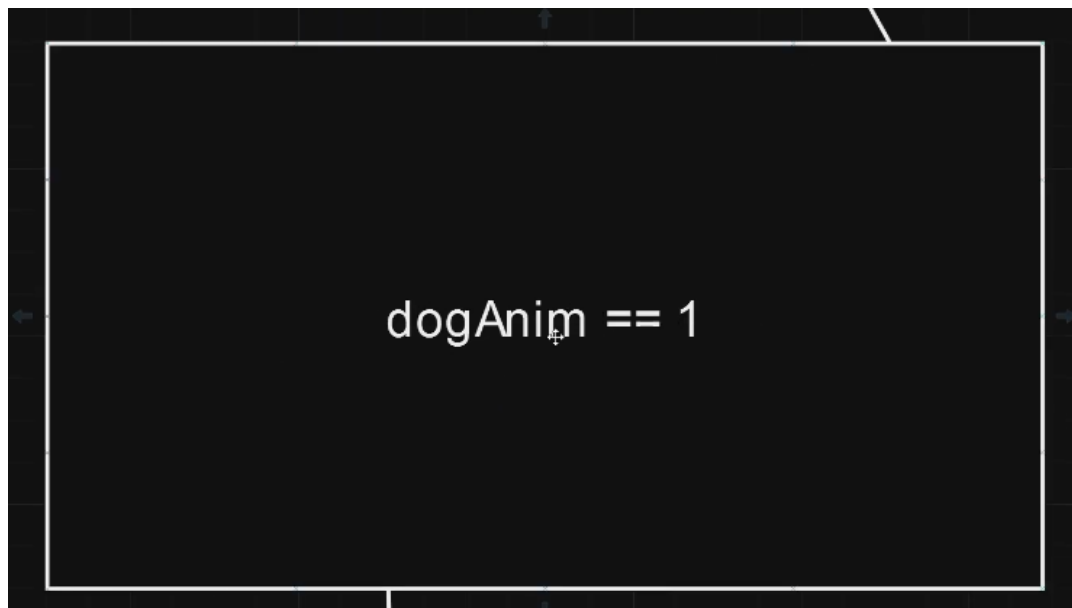
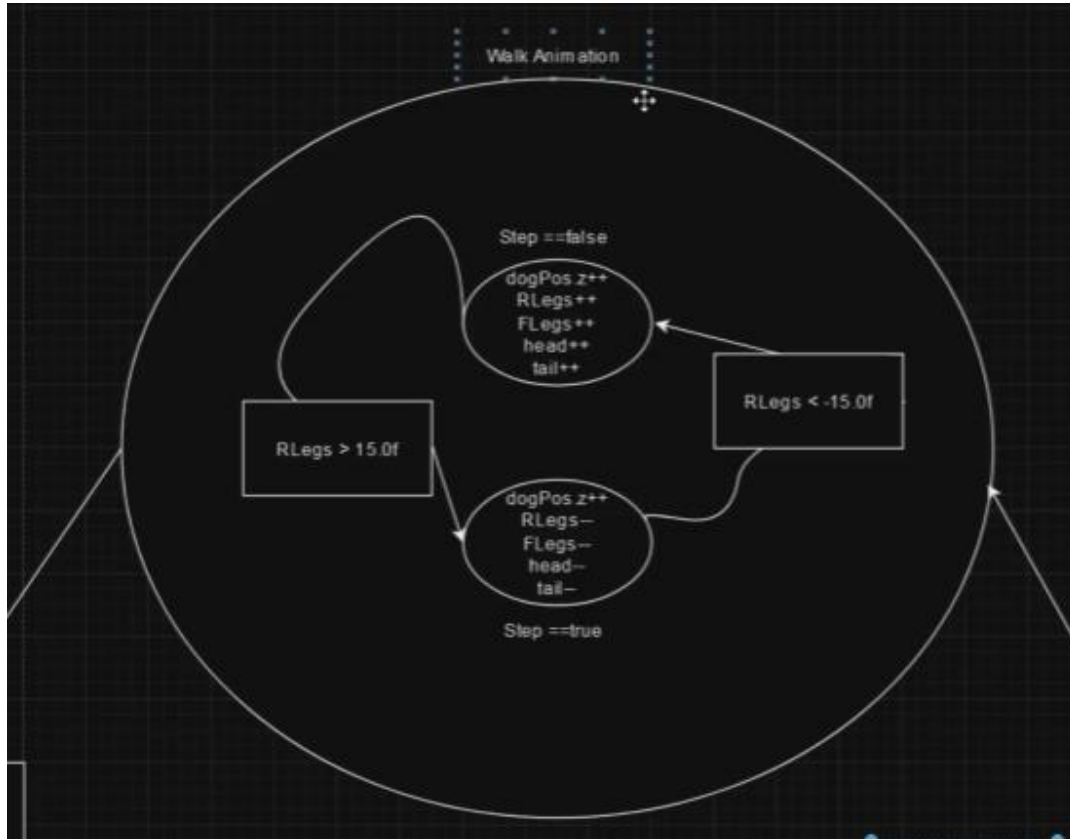
Estado 1 – todas las variables están inicializadas en cero



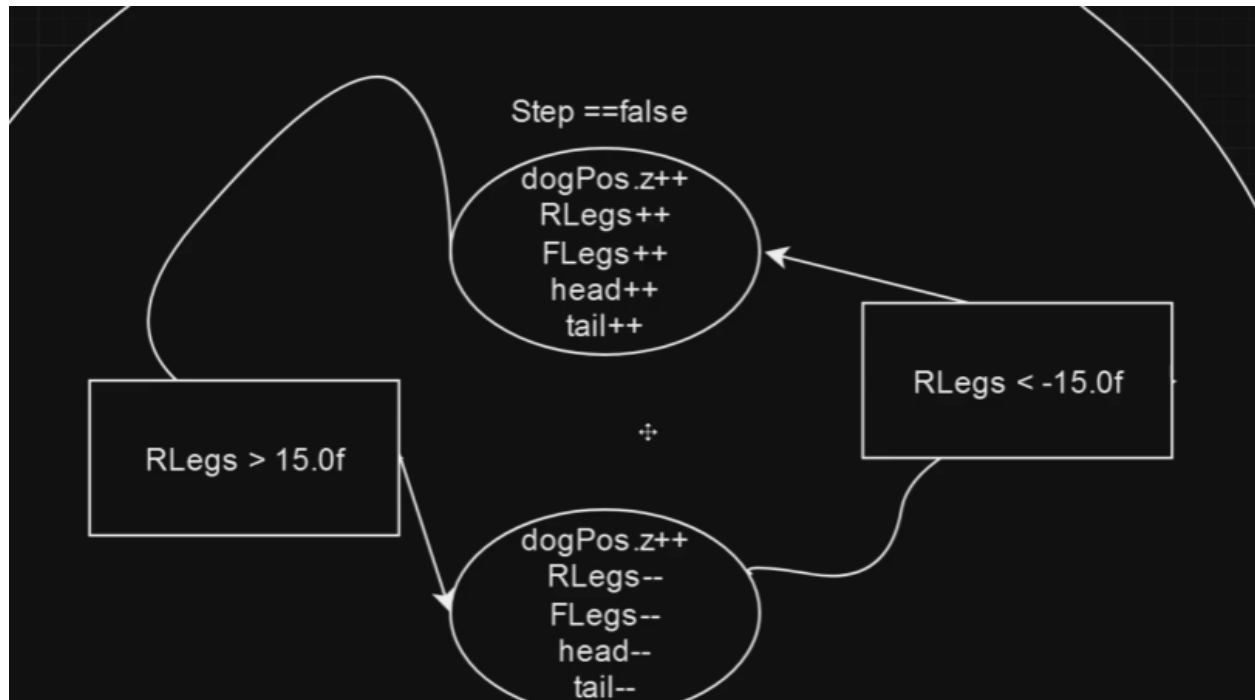
```
glm::vec3 Light1 = glm::vec3(0);  
//Anim  
float rotBall = 0.0f;  
bool AnimBall = false;  
bool AnimDog = false;  
float rotDog = 0.0f;  
int dogAnim = 0;  
float FLlegs = 0.0f;  
float RLlegs = 0.0f;  
float head = 0.0f;  
float tail = 0.0f;  
glm::vec3 dogPos (0.0f,0.0f,0.0f);  
float dogRot = 0.0f;  
bool step = false;
```

Estado 2- Animación Caminando

Se controla con una variable llamada dogAnim y si esta se encuentra en 1 quiere decir que el perrito esta caminando. Y se encuentra en estado de animación



Dentro del estado 2, se estará ejecutando el proceso o la rutina de animación



Donde se va a estar cambiando los valores de cada una de mis variables que manipulan la rotación y la posición de mis elementos que hemos cargado.

En este caso son las rotaciones para el movimiento de las extremidades. Y para el caso del cuerpo completo tiene que ser la traslación, en este caso como va a ser hacia en frente solo manipulamos la componente z, que es la que nos dará esa trayectoria. Y las demás variables solo se incrementan.

Ahora este proceso no todo el tiempo va a estar sucediendo ya que si las incrementamos constantemente lo que generaríamos es una rotación de 360° y no es lo que queremos, solo queremos que el perrito de un paso. Y cuando llegue a cierta posición haga el paso contrario.

A estos se les llamara como 2 estados, cuando el paso es falso y cuando el paso es verdadero, esto nos ayudara a saber si tenemos que mover hacia adelante la extremidad o hacia atrás. Y se controla con las condiciones donde nos vamos a basar en el angulo de una sola de las piernas llegue a cierto valor lo pasamos al estado falso RLeg mayor a $< -15.0f$ que de la primera parte del paso y lo contrario cuando sobre pase el valor contrario cambiamos al estado siguiente.

Por lo que podemos concluir que en el estado de animación dentro de esa maquina de estados tenemos una pequeña maquina de estados también, con un paso y posteriormente cambiamos a otro estado, que sería la otra parte del paso.

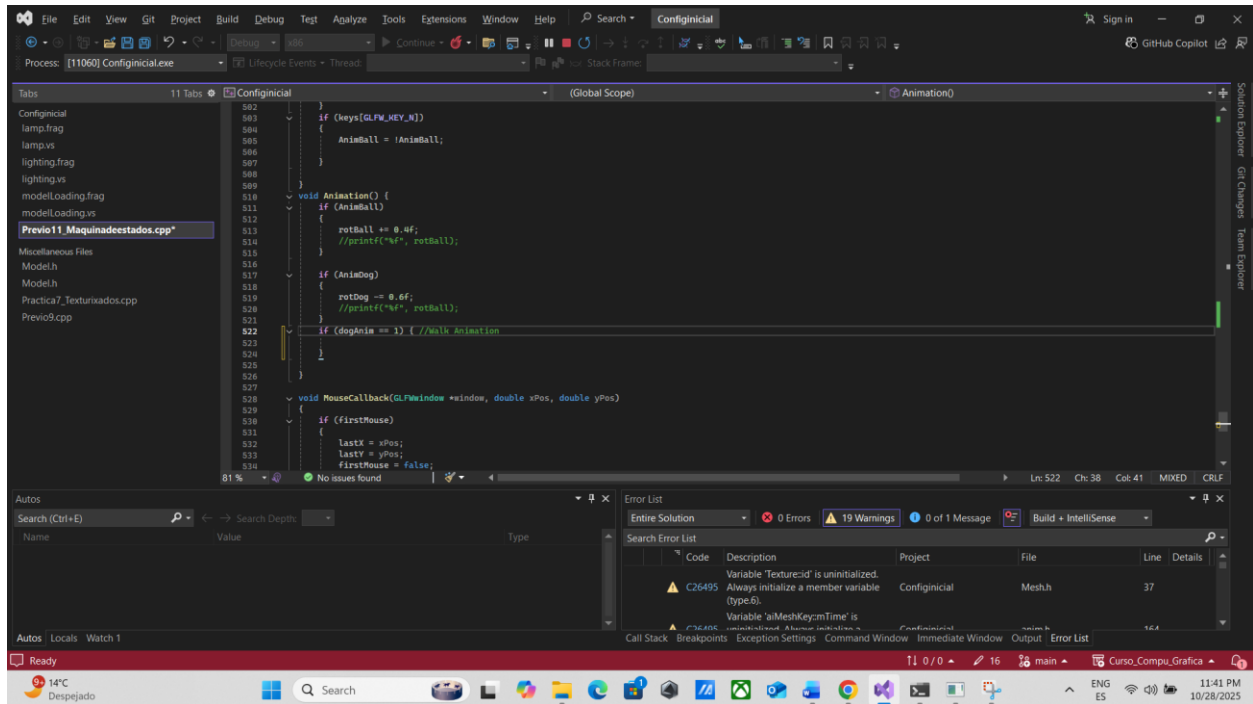
Previo 11 Animación por máquina de estados

Fecha de entrega 10/28/2025

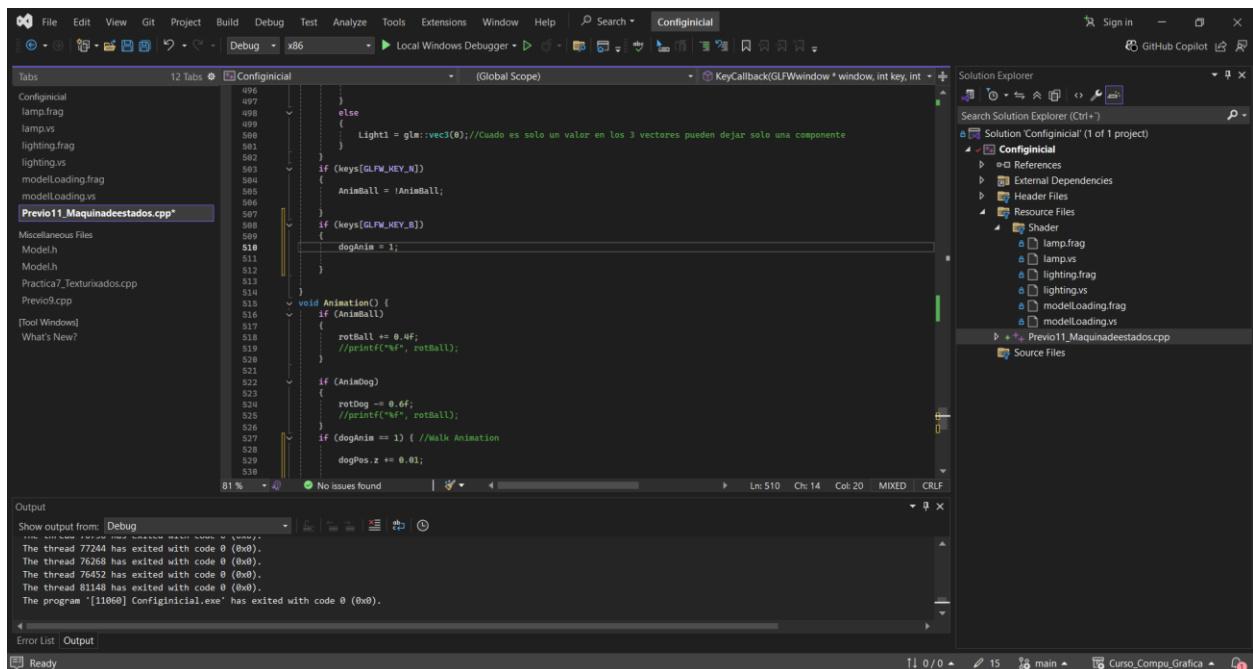
Alumna: Plata Alvarez Dayana

No. De cuenta: 318300764

Definimos las sentencias de control, que serian IF's

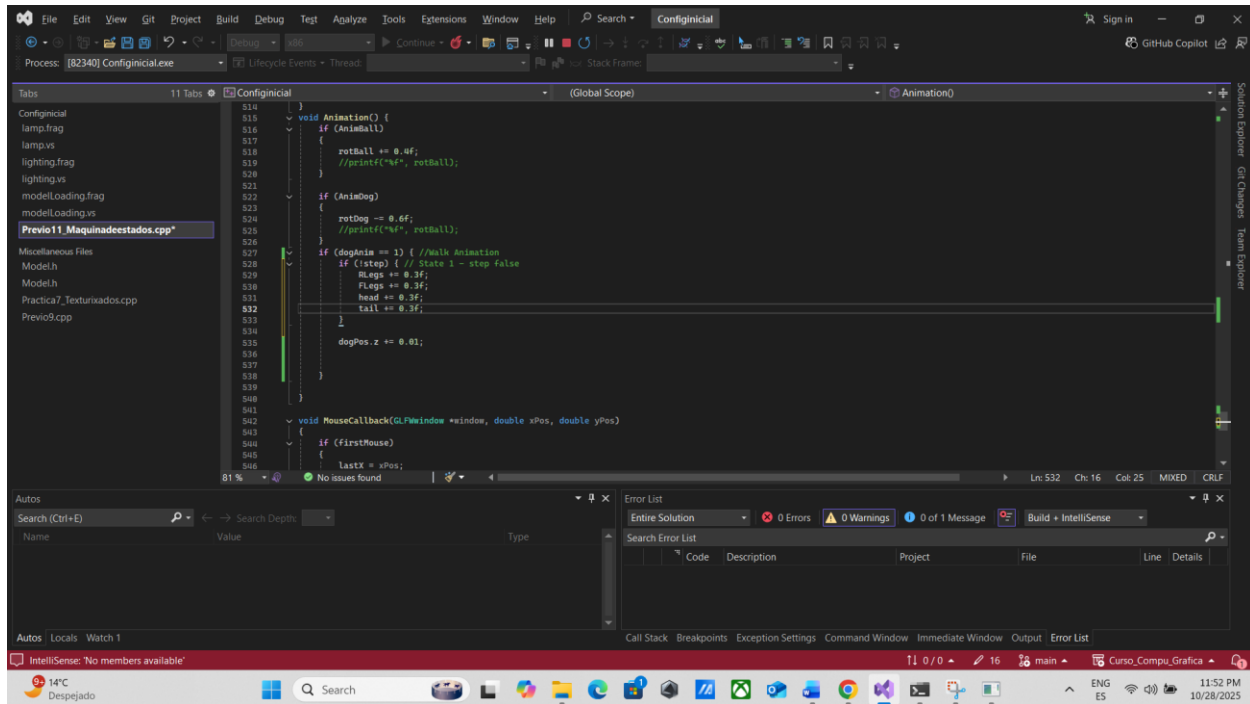


Con esto ya estamos dentro de este estado.

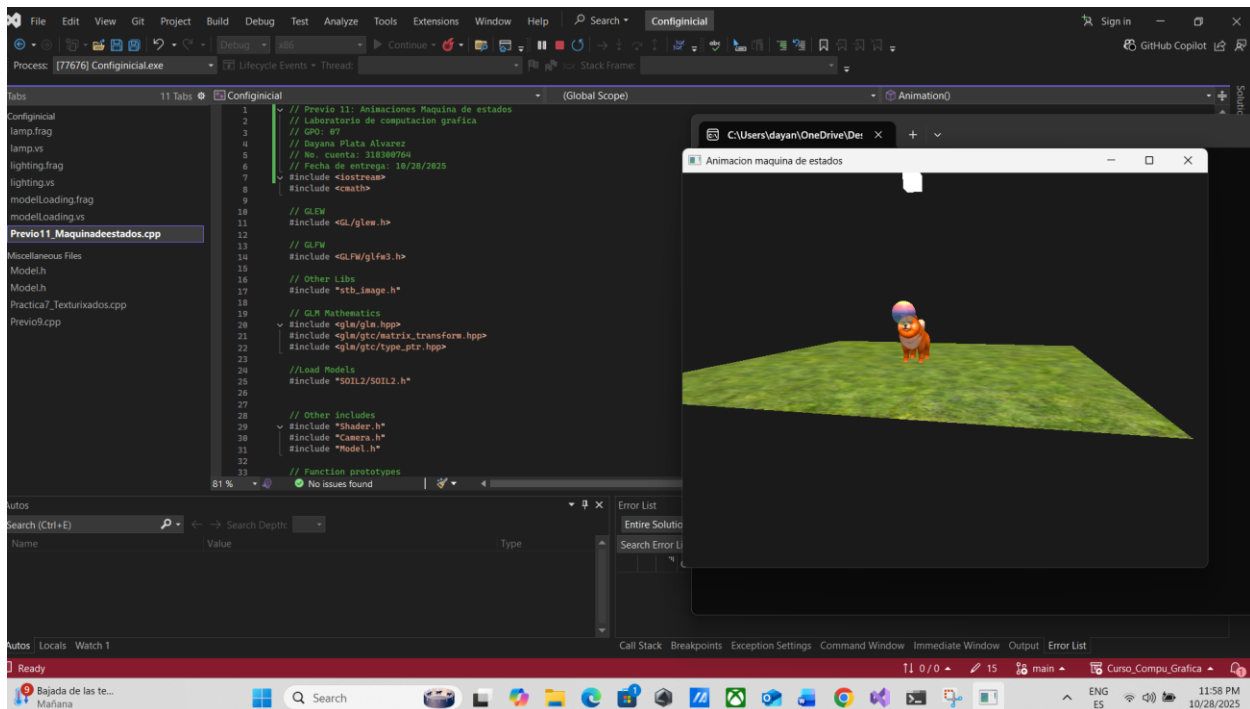


Previo 11 Animación por máquina de estados
Fecha de entrega 10/28/2025

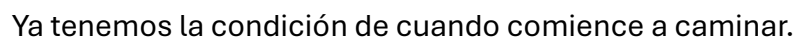
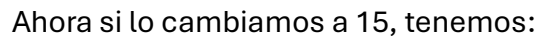
Alumna: Plata Alvarez Dayana
No. De cuenta: 318300764



Con esto ya cumplimos que elementos, ahora tenemos que aplicar la condición de hasta que momento dejaremos de aplicar ese estado.



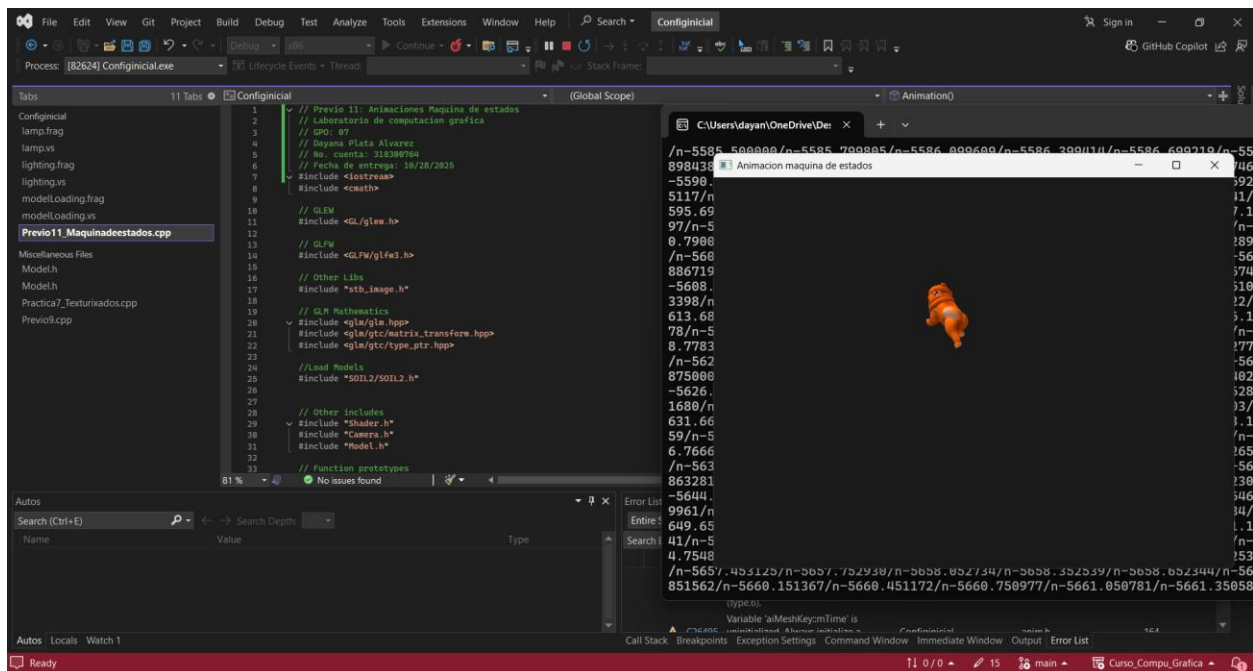
Alumna: Plata Alvarez Dayana
No. De cuenta: 318300764



Previo 11 Animación por máquina de estados
Fecha de entrega 10/28/2025

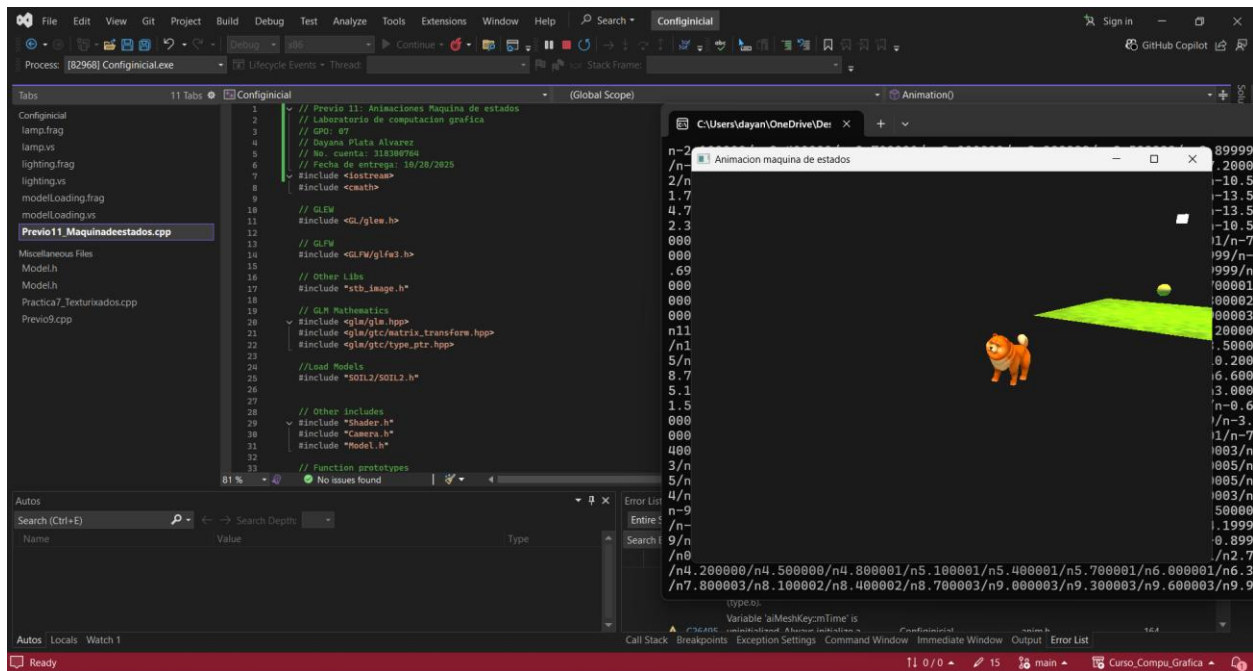
Alumna: Plata Alvarez Dayana
No. De cuenta: 318300764

Ahora agregamos el else, y obtenemos:



Ahora observamos que el perrito da una vuelta de 360°, lo cual requerimos de las condiciones para que se pueda evitar esta rotación y tengamos un parámetro establecido.

Ahora si tenemos los parámetros establecidos y tenemos que:



Por lo que podemos decir que cuando llega a un ángulo pasa a un estado y viceversa y nuevamente cuando pasa a otro ángulo pasa a otro estado.

Previo 11 Animación por máquina de estados
Fecha de entrega 10/28/2025

Alumna: Plata Alvarez Dayana
No. De cuenta: 318300764

Referencias:

<https://www.youtube.com/watch?v=B-mqD317HHM>

GIT:

https://github.com/dayanapa45/Curso_Compu_Grafica