Práctica 0

PONG

Daniel Finestrat Martínez – 48719584H

Tabla de contenido

1.- Preguntas 3
¿En qué ruta se almacena el resultado de la compilación? 3
¿Qué rutas son accesibles desde el proyecto? 3
¿Dónde has situado las imágenes de la plantilla para poder acceder a ellas? 3
¿Qué pasa si un *sprite* sale de la pantalla visible? ¿Por qué crees que esto ocurre así? 3
¿Se ejecuta tu juego igual independientemente de la potencia del ordenador donde se ejecute? Explica brevemente un ciclo de juego. 3
2.- Implementación del Proyecto 4
Clases 4
IA 4
Movimiento de la Bola 5
Colisiones: 5
Velocidad: 5
Ángulo: 5

1.- Preguntas

¿En qué ruta se almacena el resultado de la compilación?

Los resultados de la compilación se almacenan dentro de la carpeta del proyecto en la ruta "./dist/release/GNU-Linux/" la ruta puede variar dependiendo de si estás compilando el proyecto como "*Debug*" o como "*Release*" y para qué plataforma lo estés compilando.

¿Qué rutas son accesibles desde el proyecto?

Desde el proyecto son accesibles todas las rutas a las que el usuario tenga acceso. Por ejemplo, yo podría acceder a elementos situados en el escritorio poniendo la ruta relativa a la carpeta del proyecto (en mi caso, para acceder al fichero de *sprites* tendría que introducir la ruta "../../Escritorio/sprites.png")

¿Dónde has situado las imágenes de la plantilla para poder acceder a ellas?

Por el bien de la simplicidad, he creado la carpeta de recursos en la propia carpeta del proyecto. De esta manera, desde NetBeans podía compilar y acceder directamente (ya que aunque la carpeta no esté en la misma ruta del ejecutable NetBeans actúa como si lo estuviese), pero para distribuir mi juego tendría que empaquetar el ejecutable y la carpeta de recursos en la misma ruta.

¿Qué pasa si un *sprite* sale de la pantalla visible? ¿Por qué crees que esto ocurre así?

He estado testeando y cuando un sprite deja la zona visible de la pantalla no pasa absolutamente nada, sigue ahí, sigue existiendo en la lógica del programa y, si podías moverlo antes, sigues pudiendo moverlo. El sprite hará todas las funciones para las que estaba diseñado, simplemente no se visualizará.

¿Se ejecuta tu juego igual independientemente de la potencia del ordenador donde se ejecute? Explica brevemente un ciclo de juego.

Sí, excepto un pequeño detalle. Para empezar, la velocidad de las pala enemiga está calculada teniendo en cuenta un dt (*delta time*) por lo cual depende del tiempo que transcurre y no de los *frames*. Una vez dicho esto, sí que es verdad que cuando la bola coge una cierta velocidad, puede llegar a atravesar la pala si el ordenador que está ejecutando el juego no tiene suficiente potencia ya que en un *frame* estará delante de la pala y en el siguiente ya la habrá sobrepasado. Un ciclo de juego transcurre de la siguiente manera:

- Se reinicia el dt.
- Capturamos eventos del teclado y ratón.
    - Si se ha pulsado la tecla escape, terminamos el bucle de juego.
    - Si se ha movido el ratón, actualizamos la posición de nuestra pala.
- Movemos la CPU respecto a la posición de la bola (se explicará más adelante el algoritmo que consigue esto).
- Movemos la bola.

- Actualizamos los *strings* de texto (puntos, vidas y veces que hemos golpeado la pelota seguidas) con los nuevos valores, si es que los hay.
- Renderizamos todo.
    - Pintamos todo de negro para eliminar lo que había antes.
    - Pintamos el *background*.
    - Pintamos las palas y la bola.
    - Pintamos los mensajes de texto.

2.- Implementación del Proyecto

Clases

Mi proyecto se divide entre dos clases y el programa principal. De haber sido un juego más complicado, las palas, la bola, y cualquier otro elemento parte del juego (como los marcadores de puntuación) hubiesen tenido su propia clase. Mis clases, que principalmente se encargan de que los recursos de mi programa no se eliminen de la memoria, son:

- **"resources":** Esta clase maneja las fuentes, texturas e imágenes de mi juego, cargándolas en memoria nada mas ejecutar el programa y liberando la memoria antes de cerrarlo. Esta clase cuenta con unos mapas de fuentes, texturas e imágenes para poder encontrar una referencia a los recursos cargados con sólo pasar su ruta (también guardada en la clase, en una variable de tipo string).
- **"musicPlayer":** Esta clase maneja los efectos de sonido del juego y, si hubiese creado alguna música para el mismo, también la manejaría. La clase tiene unas variables de mapa y funciones de cargado y liberación de memoria parecidas a las de la clase "resources", sólo que esta vez solo nos interesan los *Sound* y los *SoundBuffer*. A parte de cargar y liberar los audios en memoria, la clase tiene funciones para reproducirlos, haciendo así fácil el uso de audios por parte de las demás clases si las hubiera y, en mi caso, desde el programa principal especialmente.

IA

La inteligencia artificial de la pala enemiga es muy simple: La pala se mueve hacia arriba o hacia abajo (dependiendo de su velocidad) durante un **tiempo indicado**. Tenemos que determinar esta velocidad, en concreto su signo o, si la posición de la bola relativa a la posición de nuestra pala cumple con unos requisitos, 0. Para calcular la velocidad, entonces, tendremos que calcular la posición relativa en el eje de las ordenadas (eje Y) entre la bola y la pala, si la primera se encuentra dentro del rango que abarca la pala (desde el principio al final), la velocidad será 0, si está por debajo, será una velocidad positiva y si está por encima, negativa. Para ajustar la dificultad de la IA fácilmente, he puesto también un **tiempo de *idle*** en el cual no se recalculará la posición de la bola respecto a la pala.

Movimiento de la Bola

Colisiones:

- Con las paredes: Son las colisiones más sencillas, simplemente detectar si la bola ha tocado la posición 0 o máxima en el eje de ordenadas y, si así es, multiplicar el ángulo por -1. Si por el contrario estamos detectando la colisión con las paredes verticales, lo haremos de la misma forma solo que al detectar colisión devolveremos la bola al centro de la pantalla.
- Con los jugadores: Colisiones algo más complicadas. Básicamente compruebo si la bola está dentro de la primera mitad vertical de cada una de las palas y, si se encuentra en esta zona, calculo la nueva velocidad de la bola, el nuevo ángulo, y la devuelvo.

Velocidad:

Uno de los dos factores que influye en el movimiento de la bola es la velocidad. La bola tiene una velocidad máxima, una mínima y, por supuesto, la velocidad actual. La velocidad cambiará cuando la bola colisiones contra una de las palas y se cambiará en un factor de 0% a 50% absoluto, o lo que es decir, se cambiará en un factor de -½ a +½ dependiendo de si la bola choca con el centro (-½ de la velocidad actual) o con los extremos (+½ de la velocidad actual). De este modo, la bola irá más rápido cuanto más al borde de la pala colisione.

Ángulo:

El segundo de los factores que influyen en el movimiento de la bola. Este ángulo se calcula en dos momentos distintos de la partida:

- Después de cada punto el ángulo se vuelve a calcular de manera automática y aleatoria, asegurándonos siempre de que este ángulo es casi horizontal, y dejando al azar si la bola saldrá hacia la izquierda o hacia la derecha.
- Cuando la bola choca con una pala, en cuyo caso la pelota tomará un ángulo diferente en relación a como de alejada del centro de la pala colisione. Si colisiona en el centro absoluto, la bola saldrá con un ángulo de 0º y si colisiona con el límite absoluto, saldrá con un ángulo máximo definido de 70.