PROYECTO FINAL: 1942

REALIZADO POR SIQI XU SUN Y XINBO CHEN CHEN

12/12/2022

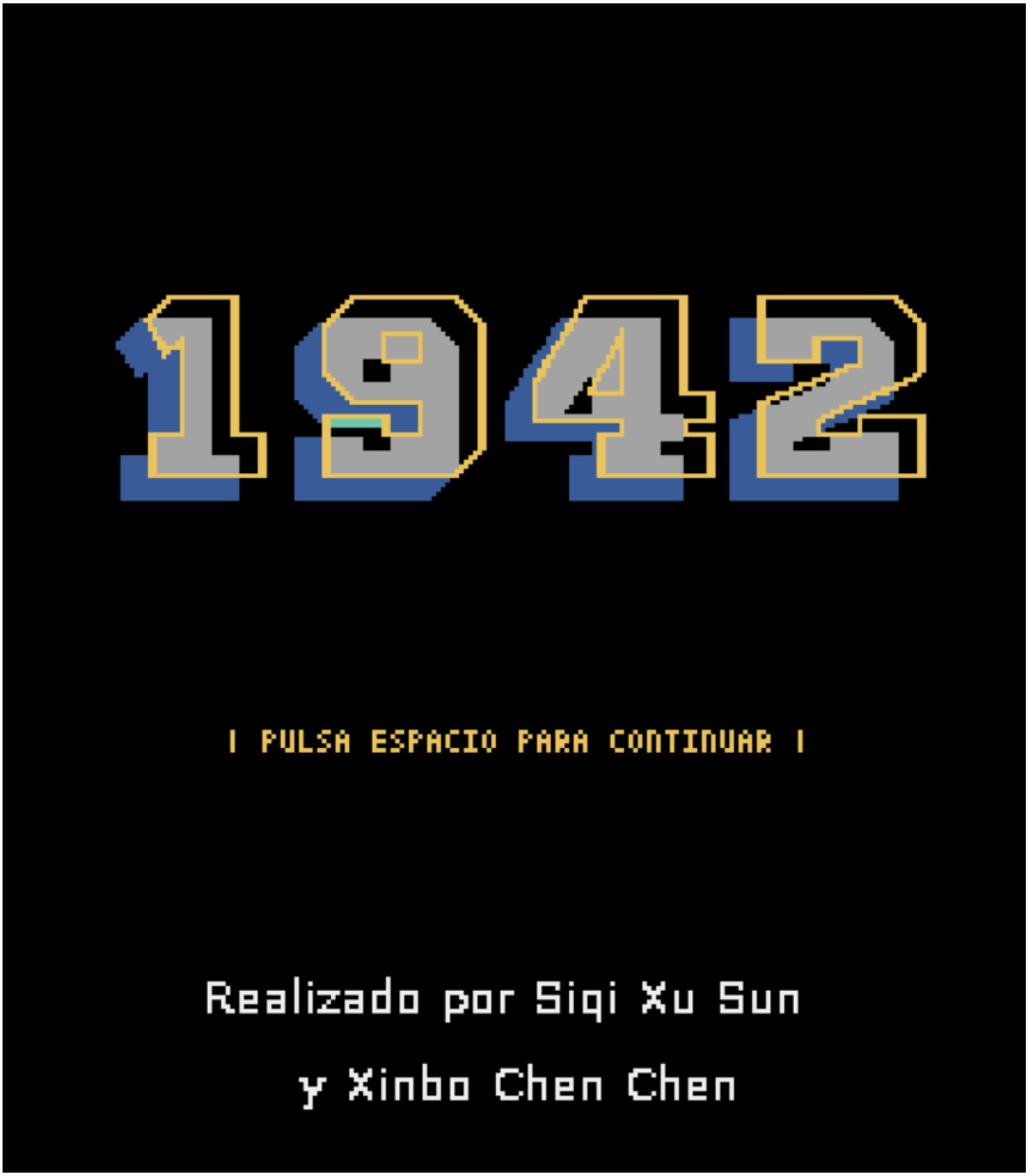

TABLA DE CONTENIDOS

INTRODUCCIÓN

En esta memoria recogemos una descripción sobre el videojuego arcade 1942 que programamos con Python y utilizando el motor de videojuegos retro Pyxel.
Para el desarrollo de este proyecto utilizamos tres diferentes imágenes para los correspondientes sprites de los elementos, como pueden ser los aviones enemigos, el jugador, las islas, etc. Dentro del programa nos encontramos con una carpeta *assets*, donde se almacenan las imágenes, y nueve archivos de Python, cada uno correspondiente a una clase diferente y el juego se ejecuta en *main.py*.

ASPECTOS TÉCNICOS

Los controles son los siguientes:
- Teclas de flechas: Movimiento del avión principal.
- S: Disparar.
- Z: Loop del avión principal (3 loops disponibles).
- Q: Salir del juego.

La dimensión del juego es de 224x256 píxeles. El juego se ejecutará a 25 fps.
Para completar el juego es necesario sobrevivir hasta el final del mapa (256x1536), que ocurre cerca del minuto de juego y es cuando aparece un último enemigo (superbombardero). Una vez que lo elimine el juego termina.

DESCRIPCIÓN DE LOS FICHEROS

Como se ha mencionado anteriormente, nuestro proyecto cuenta con nueve archivos distintos entre los cuales tenemos siete clases (la clase de Enemigo tiene cuatro clases hijas para cada enemigo, luego son once si los contamos dentro del cómputo de clases). En este apartado vamos a describir cada una de las clases y de cada archivo Python:

-avion.py

En la clase Avion tenemos un método *init* en el que está como atributos la posición x e y, el sprite, la hélice (para la animación), una lista de disparos, las vidas y loops restantes y por último un atributo self.pulsado booleano que es True cuando el usuario pulsa la tecla Z (loop). También se encuentra un método *mover* que sirve para mover el avión en las cuatro direcciones posibles. Dependiendo de la tecla de dirección pulsada el avión se dirigirá hacia el lugar si no está en el límite de donde puede ir. Que es un cuadrado encerrado en x(0,224) e y(81,228)

Por último nos encontramos con un método *disparar* en el que mientras el avión no esté haciendo el loop este pueda disparar. Dentro del método creamos la variable local disparo que se le asignará un proyectil de la clase Proyectil el cual va avanzando en línea recta hacia arriba con la posición del avión y este disparo se añade a la lista de disparos.

- constantes.py

Aquí almacenamos todos los valores que permanecerán constantes en el juego: el alto y el ancho de la pantalla, el color de fondo transparente, las dimensiones de los sprites de los enemigos, del mapa, del jugador, de los proyectiles y de las

explosiones. También permanecerán como constantes la velocidad del jugador y la de los enemigos.

- enemigo.py

En enemigo.py tenemos una clase madre Enemigo con un método init en el que se guardarán la posición x e y, una lista de disparos, un atributo booleano para saber si está vivo, otro para la vuelta del avión regular y un último atributo (self.pos) como contador para el cambio de sprites de la vuelta del avión regular.

Los diferentes tipos de enemigo se especificarán en la sección de las funcionalidades del juego.

- explosion.py

En él se encuentra la clase Explosión que sirve tanto para la eliminación de enemigos tanto como del avión principal, el cual recorre una lista de sprites que es la animación de la explosión. Para que pueda haber varias explosiones a la vez, en tablero hay una lista de explosiones a las que se añaden cuando se crean.

- main.py

En este archivo se encuentra la dimensión del tablero y es donde ejecutamos el programa.

- mapa.py

La clase mapa solo contiene un método *init* en el que está incluido el sprite del mapa.

- proyectil.py

En él encontramos la clase Proyectil, que servirá tanto para los proyectiles del jugador como el de los enemigos. En caso del jugador, el proyectil irá recto desde abajo hasta arriba tomando como posición inicial la posición actual del jugador. En caso de los enemigos, la posición inicial es la posición actual del enemigo y van de arriba hacia abajo pero sin ser recta la trayectoria debido a que también se mueve de forma random horizontalmente.

Para que pueda haber varios proyectiles a la vez, hay una lista de proyectiles, la cual siempre que algún avión dispare, este disparo se añadirá a la lista

- puntuacion.py

La clase puntuación es simplemente atributo que se pone a cero el cual va aumentando cuando los enemigos son eliminados

- tablero.py

En la clase Tablero es donde se realizará en esencia el programa. En él se importan todos los archivos necesarios. Primero empezamos declarando el método *init* en el que guardaremos el ancho y alto del tablero, iniciamos la pantalla con pyxel.init y cargamos las imágenes de los sprites (la del mapa no) en los dos primeros bancos de imágenes. A continuación creamos todos los elementos que interactúan en el juego (el avión y su proyectil, el mapa, la puntuación, etc.). También habrá un atributo (*self.pos*) que usaremos como contador para el cambio de sprites del loop del avión. Por último ejecutamos el juego.

- Principales métodos

Update: En este método que se ejecuta cada frame invocamos aquellos métodos que actualizan los diferentes objetos. Dentro del método tenemos la detección de

las teclas para salir del juego o iniciarlo. Una vez empezado el juego, importamos el mapa, guardamos una posición inicial de los enemigos (lo colocamos aquí porque algunas x deben variar). Para el avión añadimos su movimiento, el loop y su disparo. Para los enemigos, los declaramos aquí (si lo declaramos en el método init de tablero surge la complicación de que el avión enemigo será siempre el mismo), añadimos sus movimientos y disparos. A continuación realizamos el área de colisión/muerte para cada tipo de enemigo y el avión. Nosotros realizamos un área de un cuadrado/rectángulo para cada enemigo.

Pintar inicio: Pinta la portada con todo su texto al iniciar el programa.

Pintar avión: En la aparición del avión se ha hecho que aparezca siempre en el mismo lugar. Si no se pulsa la tecla z ejecutará la función que hace que se mueven las hélices del avión. Si se pulsa la z y hay giros disponibles el avión realizará un giro recorriendo la lista y restando un giro ya que el máximo es tres y al finalizarlo el valor se pondrá a 0 y pulsado a false para que al pulsarlo otra vez pueda volver a dar la vuelta. Si el avión es alcanzado por un proyectil o un enemigo explotará, se le restará una vida hasta que llegue a cero y el atributo vida sea Falso.

Si el avión no está vivo entonces explotará y será el fin del juego.

Pintar enemigo: Para la aparición de los enemigos, hay un atributo que es una tupla con tuplas que declara la posición inicial de cada uno de los enemigos dependiendo del valor de la i. Para que así más abajo, dependiendo del tipo que sea el enemigo, tomará su valor correspondiente en la tuplas para así aparecer donde debe.

Pintar explosiones: Para cada enemigo hay una explosión, como los disparos, por eso el bucle for, y si el enemigo no está vivo, es decir ha sido alcanzado por un proyectil y se creará la explosión en donde ha muerto recorriendo la lista de sprites de explosión sumando valores y al final se reiniciará a 0 para que pueda repetirse y eliminando al enemigo de la lista de enemigos.

Pintar la barra de estado (hud): La puntuación se encuentra arriba a la izquierda, los giros disponibles abajo a la derecha y las vidas restantes abajo a la izquierda.

Pintar final: Cuando el avión es eliminado salta acaba el juego y empieza un contador que suma frames para evitar así el parpadeo constante de los frames pares

Draw: Pinta todo lo anterior mencionado. Ejecuta cada frame cargando primero a pantalla de inicio y una vez empezado pintará todo excepto el final que se ejecutará cuando el avión muera o se gane el juego.

FUNCIONALIDADES DEL JUEGO

Primeramente, el movimiento de los elementos. El **avión principal** se mueve de las cuatro direcciones con las teclas de las flechas y con una velocidad de 5, es decir, avanza 5 píxeles en la dirección seguida. Su **proyectil** solo se mueve hacia arriba, es decir, en el sentido positivo del eje y (negativo en el caso de pyxel) con una velocidad de 7.

En cuanto al **movimiento de los enemigos** tenemos diferentes.

El avión regular, solo se mueve hacia abajo hasta que llega sobre la mitad de la pantalla, momento en el cual cambia al sentido contrario pero manteniendo la dirección y, con ello, sus sprites para crear así la animación de que se da una vuelta. Este avión dispara proyectiles desde la clase proyectiles desde donde está hacia abajo.
El avión rojo aparece por la izquierda de la pantalla, describe dos circunferencias mientras se mueve en el eje positivo de X y finalmente desaparece (si no es abatido antes) por la derecha. A diferencia de los otros aviones, este no disparará.
En cuanto al bombardero, éste aparecerá en la parte superior de la pantalla, y a diferencia de los regulares, no volverá por donde vino sino que dará una vuelta y seguirá recto hasta desaparecer en la parte inferior. Este enemigo tiene un atributo vidas, la cual disminuirá si las balas del avión principal impactan sobre él y cuando llegue a cero explotará el avión. También dispara proyectiles como los aviones regulares.
El enemigo final, el superbombardero aparecerá desde la parte inferior del mapa, subirá, dará una vuelta, y seguirá moviéndose de forma diagonal hasta quedarse en un punto. Este enemigo tiene un atributo vidas mayor que el bombardero por ser el jefe final, la cual disminuirá si las balas del avión principal impactan sobre él y cuando llegue a cero explotará el avión. También disparará varios proyectiles a la vez, los cuales se encuentran en la clase proyectil llamándose disparo superbombardero 1 y 2 los cuales se mueven de forma diagonal, y el disparo que tienen todos, que lo heredan de su clase madre Enemigo. Este enemigo tiene un atributo vidas, la cual disminuirá si las balas del avión principal impactan sobre él y cuando llegue a cero explotará el avión.

CONCLUSIONES

El videojuego arcade 1942 es complejo de programar, pero en esta versión reducida y adaptada hemos intentado crear un nivel del propio juego en el que el objetivo es destruir a un enemigo en concreto, el Superbombardero.
Este proyecto está conformado por diferentes ficheros para facilitar el trabajo con las clases y los valores que permanecen constantes en el juego.

PROBLEMAS ENCONTRADOS

Entre las diferentes dificultades que nos encontramos destacamos la limitación del motor de videojuegos Pyxel, el uso de los diversos sprites, la detección de muerte de los enemigos y del jugador, es decir, programar el área correspondiente y la detección para las explosiones; por último, la mala gestión en el inicio del proyecto que ocasionó en un trabajo mayor en los últimos días.

COMENTARIOS PERSONALES

SIQI XU SUN

El proyecto en general se me ha hecho muy pesado debido a la cantidad de métodos, atributos, clases que había. De manera que al hacer un método era muy

fácil equivocarse en el lenguaje por haber usado un atributo o una clase o un self, etc. donde no se debía. Sobre todo en las explosiones y la desaparición de enemigos, al incluir tantas listas y tener que borrar elementos si se cumplían ciertas condiciones se hizo muy lioso y complicado. También se hizo pesada la parte del movimiento, ya que era difícil realizar una curva mediante la posición, ya que si pasa por x posición lo haría otra vez sin así llegar al final, por lo que fue difícil llegar a la conclusión de usar frames para que se moviese. El apartado que más fácil ha sido la creación de clases y lo que se supone que hace cada cosa. La más difícil como ya he dicho, no confundir un atributo, clase, método con otro.

XINBO CHEN CHEN

Para mí el proyecto ha resultado bastante interesante. Me ha gustado aplicar la programación orientada a objetos a un caso real en el que tuvimos bastante libertad para programar el videojuego a nuestra manera. También me agradó la idea de que el proyecto no fuera desde cero y que los sprites pudieran ser colaborativos porque así no invertimos el tiempo en el diseño de las imágenes, sino que lo dedicamos al código. Sin embargo, también es cierto que debido a nuestra mala gestión del proyecto este resultara tedioso en los últimos días.

En resumen, este proyecto fue bastante gratificante, divertido y, en cierta medida, retador. Lo mejor de todo era cuando intentabas implementar una funcionalidad y ves que, efectivamente, funcionaba. También destacar que este proyecto ha sido llevado a cabo por parejas porque así facilita bastante el trabajo y se hace más llevadero.