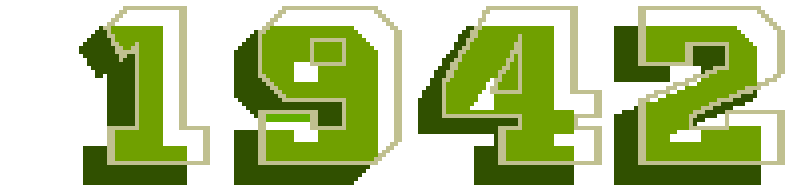
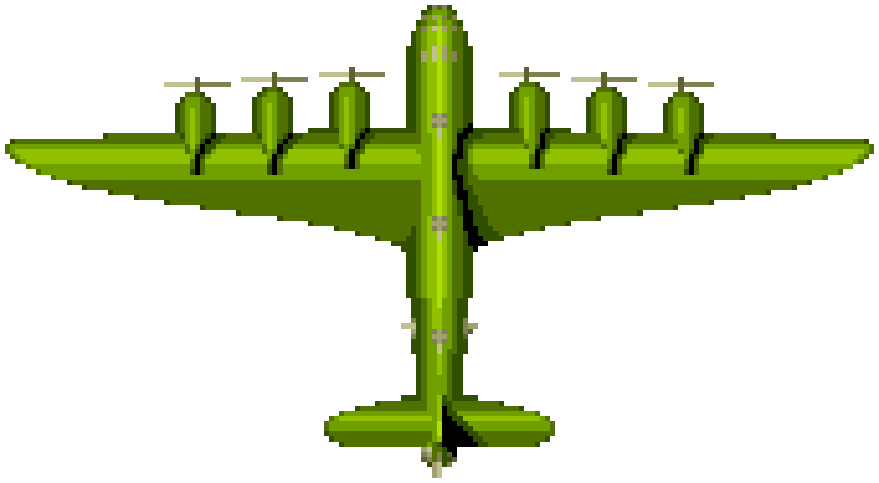
Memoria “1942”



Ripped by me, Magma Dragoon
Copyright Capcom - Credits not necessary



Autores:

Miguel Jimeno

Fernando López de Olmedo

Índice

Introducción

El juego “1942” es un juego basado en los aviones de la Segunda Guerra Mundial. Este es bastante antiguo y parecido al conocido juego Space Invaders. En el debemos conseguir esquivar tanto balas como enemigos (Enemigo Regular, Enemigo Rojo, Bombardero y Superbombardero) y conseguir la mayor puntuación posible destruyendo las aeronaves enemigas antes de que se nos agoten nuestras 4 vidas.

Es un juego simple pero entretenido, consigue engancharte siempre para intentar superar tu record personal.

A la hora de programarlo tienes mil maneras de hacerlo. Formarás la tuya con tus propios métodos, basados en tu conocimiento y manejo de la programación. Una parte interesante es la soltura que vas consiguiendo para hacer tus diferentes métodos a medida que avanzas haciendo este juego.

En este documento, se van a explicar tanto las diferentes clases de las que consta el programa como los métodos más confusos de cada una. También comentaremos el desarrollo de creación juego y por último daremos una opinión sobre este.

Instrucciones

Para que el juego funcione debemos seguir varias instrucciones. Estas son clave ya que sin ellas no servirá nada nuestro esfuerzo.

- Para ejecutar el juego debemos tener claro la ubicación de nuestro programa de Python en el ordenador
- Cuidar el lenguaje utilizado para nombrar los diferentes archivos ya que estos nos servirán para enlazar las rutas necesarias para el programa
- Asegurarnos de las actualizaciones de los programas como pyxel, que utilizamos en nuestro proyecto

MECANICAS

Los comandos y mecánicas del juego son sumamente sencillos:

- **Movimiento:**

  El programa está desarrollado para que el movimiento se pueda ejecutar tanto con las teclas (UP, DOWN, RIGH, LEFT) como con el (WASD)
- **Loop:**

  Para realizar el loop, el cual solo se podrá hacer 3 veces se deberá de presionar la tecla Z. Gracias a esta mecánica podremos esquivar tanto las balas enemigas como a estos mismos sin perder ninguna vida.
- **Disparo:**

  Para que el avión dispare a los enemigos se ha de pulsar la BARRA ESPACIADORA, con este disparó podremos eliminar a los enemigos. Si conseguimos alcanzar a los enemigos, sumaremos puntos (Enemigos Regulares y Rojos: 20 pnts, Bombardero:100 pnts, Superbombardero: 200 pnts), cada enemigo tiene diferentes vidas, por lo que no será tan fácil derrotar a algunos de ellos (Regulares y Rojos: 1 vida, Bombardero: 6 vidas, Superbombardero: 12 vidas)
- **Inicio y Fin:**

  Como bien indica el propio juego, para inicializarlo, se ha de presionar el TAB, y una vez pierdes tus 4 vidas correspondientes (3 más la vida 0), se ha de pulsar ESC para cerrar el programa.

Clases utilizadas

Tenemos varias clases en este programa, estas se resumen en:

- **Tablero:**

  Esta clase almacena todo lo que se va a ejecutar en el programa. Gracias a los métodos update y draw que contiene esta clase, se corren todos los demás métodos creados.

  A parte en esta clase nos encargamos de crear la lista de enemigos, gracias a la función de pyxel “frame count”, con esta hacemos que se vayan añadiendo los enemigos a la lista en función de los frames.

  También encontramos funciones que se encargan de detectar cuando se produce una colisión, tanto de la bala con los enemigos como de los enemigos (o sus respectivas balas) con el avión. Para ejecutar las colisiones con sus respectivos

sprites, hay funciones para el cambio de sprites de la colisión todos los enemigos, como para la explosión del avión.
En el tablero pintamos todo lo que tiene que ver con las vidas, las islas, las colisiones…
Definimos algunos métodos con los que borramos ciertos enemigos o balas.
Y por último definimos el cambio de pantalla. Empezamos con la de inicio y posteriormente inicializamos todo lo que tiene que ver con el juego. Cuando el juego termina salta la pantalla de FIN.

- **Avión:**

  En esta clase encontramos como es de suponer todo lo que tiene que ver con el avión.
  Definimos tanto su movimiento como su giro en dos diferentes métodos pero que luego son ejecutados en el "update del avión" (que solo se ejecuta al ser metido en el update del tablero). En este último método nos ayudamos del "btn", para que en función de las teclas que toques suceda una cosa o otra.
  En este update también metemos el loop del avión, cuyo cambio de sprites está definido en otro método.
  Las balas son añadidas a la lista balas también en esta clase.
  Por último, hay una serie de métodos que los utilizamos para pintar gracias al "blt"

- **Enemigos:**

  En esta clase creamos la definición del movimiento de todos los enemigos presentes en el juego. Como ya he mencionado la parte de incluirlos en una lista esta definida en el tablero.
  cada enemigo tiene su propio y complejo movimiento, como su propio sprites, por ello la clase tiene tantas variables.
  Dentro de este archivo está también la clase **BalaEnem,** en la que como su propio nombre indica se crean las balas que van a disparar los enemigos, ya que cada uno tiene su propio disparo implementado

- **Balas:**

  Se define el movimiento y Sprite de las balas del avión.

- **Mapa**:

  En esta clase encontramos las definiciones necesarias mover las islas creadas en el mapa. Cuando esta desaparece del tablero de juego, se vuelve a crear arriba del mapa.

  Encontramos también la clase **Portaaviones**, que sirve para crear el portaaviones que se dibuja al inicializar el programa.

- **Colisiones:**

En esta clase definimos las colisiones de los enemigos Regulares y los denominados como Rojos. Dentro de este archivo encontramos otras clases que siguen exactamente el mismo funcionamiento, pero con la **explosión del bombardero** y con la **explosión del superbombardero.**

- **Constantes:**

  No es una clase, pero creo que hay que hablar de este archivo, ya que gracias a él aclaramos mucho el resto del código ya que todos los datos se encuentran aquí.

Algoritmos

- **Movimiento:**

  Tanto como para el avión, islas, balas, enemigos, con este método en el que se utilizan listas y otras variables definiremos sus movimientos, y en el caso de los enemigos también añadiremos las balas a la lista según su posición y tipo. Cuenta con coreografías diferentes para cada objeto. Para que el objeto se mueva utilizaremos principalmente operaciones con la variable x e y, que van acompañados de contadores y demás.

- **Creador de listas:**

  Gracias a este método añadimos los diferentes objetos a las listas para que se pueda definir un movimiento o sprite determinado.

- **Draw:**

  Con un pyxel.blt y en algunos casos utilizando el for para las listas, conseguimos que aparezca el sprite en el sitio y momento adecuado.

- **Update**:

  Desde aquí actualizaremos todos los demás métodos cada frame para que pueda ser jugable el videojuego.

Desarrollo y otros aspectos

Como hemos comentado anteriormente, el comienzo se hizo un poco duro ya que somos novatos en esto, pero a medida que hemos ido avanzando hemos superado todos los obstáculos que han podido interponerse a la hora de realizar el juego. Hemos llegado a desarrollar el juego con todo lo que se pedía, incluso con extras como lo es el giro del avión a la hora de moverse. Como aspectos a mejorar nos hubiese gustado

una mejor apariencia del mapa, pero debido a las limitaciones de pyxel no podíamos permitirnos hacer algo ostentoso. Queremos añadir que se podrían haber incluido sonidos a la hora de moverse, disparar, una música de fondo…, que hubieran hecho de este juego una experiencia mucho más disfrutable audiovisualmente hablando. Pero debido a la complejidad del proyecto y el escaso tiempo del que disponíamos no se ha podido implementar.

Resumen

Al ver el código de nuestro programa te puedes dar cuenta de que hemos utilizado en todo momento unos métodos y algoritmos propios, ya que a la hora de hacer algo así no te sirve coger aspectos de programas ajenos. Una vez empezado el tuyo, a tu manera, es prácticamente imposible acoplar otro tipo de métodos diferentes, todo está enlazado, y con un solo problema no conseguirás que corra el juego.

Pensamos que se podría hacer de una manera mas resumida, pero hemos querido que el juego no sea muy monótono y hemos tenido que innovar en algunos aspectos.

Para los pocos métodos e información con los que cuenta pyxel comparado con otros programas de Python y la escasa experiencia, consideramos que hemos realizado un gran trabajo y estamos bastante satisfechos con él.

Problemas

El principal problema que encontramos era a la hora de crear las listas y usarlas para el movimiento y draw, pero una vez entendimos el funcionamiento de estas era solamente un método más, igual que los demás.

Recalcar que las colisiones fue algo a lo que le estuvimos dando vueltas bastantes horas, pero gracias a programas (que descubrimos más tarde) que teníamos como ejemplos en github, pudimos desarrollar bastantes buenos métodos para llevarlas acabo en el juego.

Debemos de reconocer que nos ha resultado un poco pesado es usar el pyxel edit ya que no contábamos con todos los sprites deseados y hemos tenido que ir creando y actualizándolos para poder dar una mejor imagen al juego. A parte de que, por poner un ejemplo, coger los 15 sprites con los que contaba el loop del avión era una inversión de tiempo bastante grande.

Comentarios personales

En nuestra opinión, ha sido un trabajo entretenido de hacer y que te hace pensar mucho. No se ha hecho nada pesado comparado con las expectativas que teníamos de él y aparte nos ha servido bastante para desarrollar habilidades en la programación y tener una visión más amplia. Esto será algo que en un futuro examen o trabajos le podremos sacar mucho provecho. En conclusión, estamos muy satisfechos con nuestro proyecto y su funcionalidad.