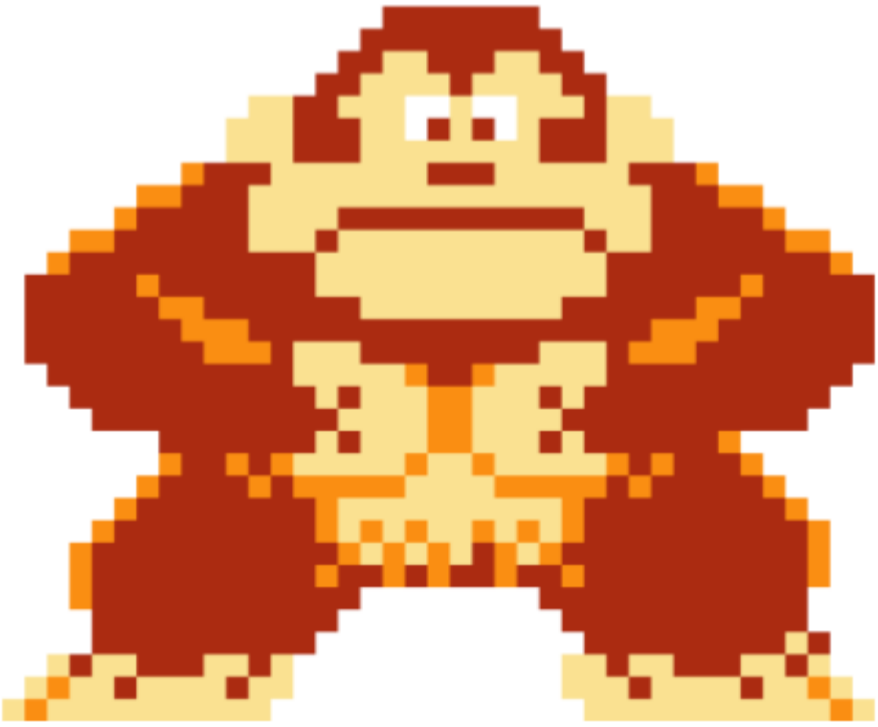


Proyecto Final Programación

Juego Donkey Kong Arcade

Miguel Martín e Iker Tejero

Índice

Observaciones

Uno de los grandes problemas que nos hemos encontrado a la hora de implementar el juego es la falta de documentación y escaso uso de la herramienta utilizada. Por lo que hemos podido ver, Pyxel no es demasiado popular o por lo menos no mucha gente lo utiliza. Habría sido útil poder haber encontrado más ejemplos de cómo usar los métodos propios de esta librería. Otro de los problemas que nos hemos encontrado ha sido la falta de organización, planificación y pensar el código que íbamos a escribir. El 30% del tiempo lo hemos pasado escribiendo código y el 70% restante intentando debuggear el código anteriormente escrito. Por lo demás, ha sido relativamente fácil adaptarnos. La complicación

En cuanto a comentarios personales, el proyecto ha sido, cuanto menos, un gran reto. Muchas partes del juego eran bastante entretenidas, aunque algunas podría llegar a ser bastante frustrantes. Consideramos que lo mejor del trabajo ha sido la gratificación y satisfacción sentida al conseguir tener un programa que funcione.

Donkey Kong - Miguel Martín e Iker Tejero

Ventana

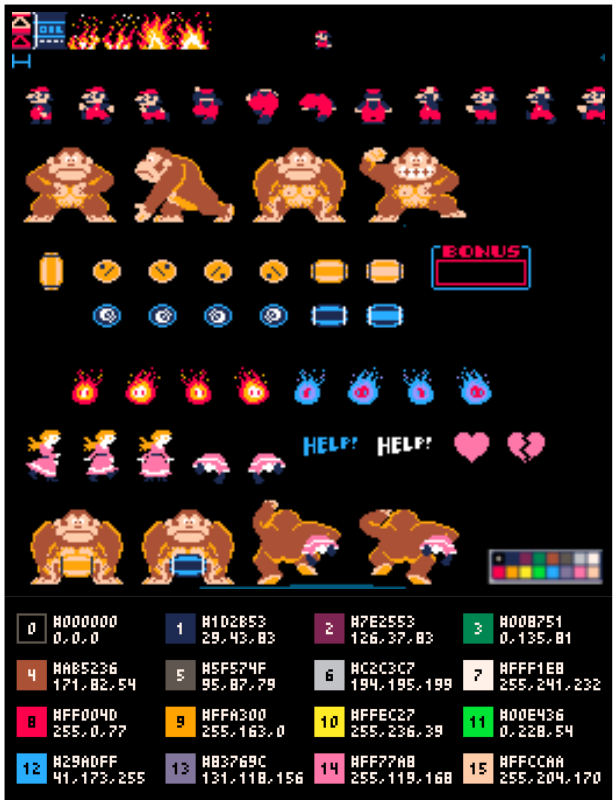
Para la creación de la ventana del programa emplearemos el comando de pyxel: **pyxel.init**
Este comando recoge los siguientes atributos:

```
init(width, height, [caption], [scale], [palette], [fps], [border_width], [border_color], [quit_key])
```

En este caso, únicamente hemos empleado: **width (anchura)** = 224, **height (altura)** = 256, **scale (escala)** = 3, y **fps (fotogramas por segundo)** = 30.

Banco de Imágenes

Para incluir las imágenes hemos creado un archivo **.pyxres** que aloja tres bancos distintos de imágenes, donde se encuentran todas las formas necesarias en el programa. Importamos el banco mediante el comando:
pyxel.load(ruta del archivo**)**
El banco de imágenes ha sido el siguiente:



Imágenes

Una vez ya importado el banco de imágenes, para incluirlas dentro del programa emplearemos el comando:

```
blt(x, y, img, u, v, w, h, [colkey])
```

x: posición x en la ventana
y: posición y en la ventana
img: banco de imágenes (0, 1, 2)
u: posición x en el banco de imágenes
v: posición y en el banco de imágenes
w: anchura de la imagen
h: altura de la imagen
[colkey]: eliminar color de la imagen

Para iniciar el programa, emplearemos el comando **pyxel.run**, `run(update, draw)` que ejecuta las funciones update y draw mientras el programa esté abierto.

Funciones Básicas del Programa

Update

La **función update** es la encargada de actualizar el programa cada fotograma o frame. En ella hemos incluido básicamente los métodos que necesitamos que sean ejecutados en todo momento. Destacan **mario.move**, con el cual podemos ver en todo momento a mario en su posición adecuada y cómo interacciona con el entorno; y **barriles.move**, con el cual controlamos el movimiento de los barriles. Además incluimos los comandos **pyxel.btnp** que controlan si se presiona una tecla durante la ejecución del programa, con el fin de activar atributos como **mario.saltar**, o ejecutar el comando **pyxel.quit** para salir del programa.

Draw

La **función draw** es la encargada de mostrar las imágenes del programa en todo momento. En ella es donde se incluyen principalmente todos los objetos o clases del programa, para así mostrar sus imágenes en pantalla. Además incluimos todos los elementos gráficos de la interfaz del programa, ya sean las pantallas de nivel superado, game over, o el escenario del nivel.

Clases

Mario

La **clase Mario** es una de las principales de nuestro programa, compuesto por una serie de los siguientes atributos: **x, y, marioAppear, saltar, salto, vidas, vivo, wins y puntos**. De los cuales, todos son atributos privados, menos **saltar, x,** e **y**.
Saltar y **salto** son atributos que hemos utilizado a la hora de controlar el salto de Mario. **X** e **y** son también fundamentales para el movimiento del personaje. **Vidas** y **puntos** como su nombre indica, contabilizan las vidas que le quedan a mario y los puntos que ha ido ganando. **Wins** es un atributo booleano del que nos servimos para saber cuándo Mario ha conseguido llegar a la princesa. De igual manera, **vivo**, a pesar de no destacar por la originalidad en el nombre, es otro booleano con el que sabemos cuándo el personaje ha perdido todas sus vidas.

Los métodos más importantes de esta clase son: **move(), chocaPlataformas(), teclas(), drawVidas(), drawPuntos(),** y **muerteMario().**
El fundamental, es **move()** pues es en el que se incluyen la gran mayoría de los métodos. Esta función se ejecuta cada frame y, con la ayuda de los demás métodos, chequea si Mario se sale de la pantalla, si está encima de una plataforma, si se choca contra un barril o si está en el rango para subir por las escaleras.

Dentro de **move(),** encontramos los métodos **chocaPlataformas(), teclas() y muerteMario().** El primero es bastante auto explicativo: es el método que detecta cuándo Mario está en las mismas coordenadas que una plataforma y hace que no las atraviese. **Teclas()** se encarga de mirar cuándo el usuario pulsa la flecha de arriba, la de los laterales y el espacio, con eso controla los movimientos del personaje incluyendo el salto. Y **muerteMario(),** se encarga de, cuando Mario sea golpeado por un barril, comprobar que le quedan vidas, en cuyo caso le resta uno y hace que reaparezca al principio y, en caso de no tener más, poner el atributo **vivo** en False.

Donkey Kong (MonoFurioso)

La clase de Donkey Kong, a la cual le hemos llamado de una manera jocosa **MonoFurioso**, tiene únicamente los atributos monoAppear , **x,** e **y**. **X** e **y**, sirven para saber las coordenadas en las que dibujamos a Donkey Kong. **MonoAppear** es integer que nos sirve para poder animar a este personaje. Cuando el atributo es igual a 1, implica que los frames son múltiplos de 60 y Donkey Kong se dibuja hacia la derecha; cuando es 2, se dibuja hacia la izquierda; y cuando es 3, se dibuja agarrando el barril y mirando al frente.
Los métodos más importantes son, precisamente, esos: **drawMono(), drawMonoCenter(), drawMonoRight() y drawMonoLeft().** Y sirven para lo que hemos descrito anteriormente.

Pauline

La clase para la princesa Pauline es muy, muy básica pues sólo tiene dos atributos: **x** e **y**, que nos ayudan a colocar al personaje en sus coordenadas. Su único método se reduce a **drawPauline()**

Barriles (BarrilesRodando)

Esta clase únicamente tiene dos atributos y son **x** e **y**, los cuales están prefijados a unas coordenadas en concreto que son donde aparecen los barriles. Entre sus métodos destaca **moveBarril()** el cual es parecido al método que hemos descrito para la clase **Mario**, pero con algunas peculiaridades. El método se encarga de ver cuándo choca con las plataformas y, dependiendo de en cuál esté, se mueve de una forma u otra. En las plataformas crecientes, su **x** se mueve hacia la derecha, mientras que en las decrecientes se mueve hacia la izquierda. También nos encontramos **drawBarrilesQuietos()** que es meramente para dibujar los 4 barriles a la izquierda de Donkey Kong que ni se mueven ni están animados.

Escalera

La clase **Escalera** se compone de los atributos de posición **x** e **y**, más un extra respecto a las otras clases: el atributo **h**, el cual nos ayuda a conocer la altura que tiene la escalera en concreto. Su único método es **drawEscaleras()**, el cual hace lo que su nombre indica y va dibujando las escaleras en las posiciones específicas. También tiene una lista llamada **escaleras[]** en la cual guardamos las coordenadas en las que se encuentran las imágenes que hemos dibujado. Esta lista nos ayuda a la hora de saber cuándo **Mario** puede subir o no por las susodichas.

Plataformas

La clase **Plataforma** es cuanto menos curiosa, pues no posee ningún atributo. Los métodos más importantes son **platInferior(), platDecreciente(), platCreciente(), platDecreciente1(), platCreciente1(), platSuperior(), platPauline(), y drawPlataformas().** Todos estos métodos reciben como argumento una coordenada **x** y una coordenada **y** para saber dónde empezar a dibujar las plataformas. Cada uno de los métodos dibuja la Plataforma indicada en su nombre y el último método llama a todos los demás para dibujar las plataformas (este método era para darle claridad y comodidad al código).

Descripción general de los algoritmos más relevantes utilizados.
o Descripción del trabajo realizado, funcionalidad incluida, partes no realizadas y
/ o funcionalidades adicionales proporcionadas..