Memoria Donkey Kong:

* + Portada, tabla de contenidos y breve resumen del documento. Later
  + Descripción del diseño de las clases, incluidos los campos y métodos más relevantes.
  + Descripción general de los algoritmos más relevantes utilizados.
  + Descripción del trabajo realizado, funcionalidad incluida, partes no realizadas y/o funcionalidades adicionales proporcionadas.
  + Conclusiones:
    - Resumen final del trabajo realizado.
    - Principales problemas encontrados al implementar el juego.
    - Comentarios personales (retroalimentación) incluyendo críticas al proyecto final.

Descripción del diseño de las clases, incluidos los campos y métodos más relevantes:

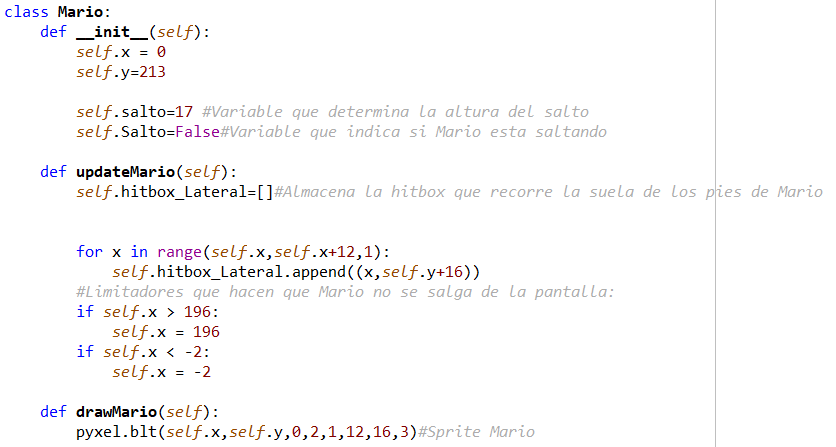
Dentro del trabajo encontraremos las siguientes clases: Tablero, Mario, Plataform, Stairs, Barril, Donkey Kong, Pauline, Vidas y Puntuación

Como resumen rápido, cada una de estas clases representa a un elemento que aparece dentro de juego, bastante explicativos por su nombre, excepto Tablero, lo que hace Tablero es generar objetos de las clases dentro de él con sus características y propiedades y que interactúen dentro de él para luego ejecutarse con la función “run”. A continuación vienen explicados cada uno de ellos acompañados del fragmento de código que usan:

Nota:

(Cabe mencionar que la mayoría de la metodología que usan estos objetos se halle en el tablero y no aparezca específicamente dentro del propio objeto, pero será explicado dentro de la clase Tablero)

Mario:



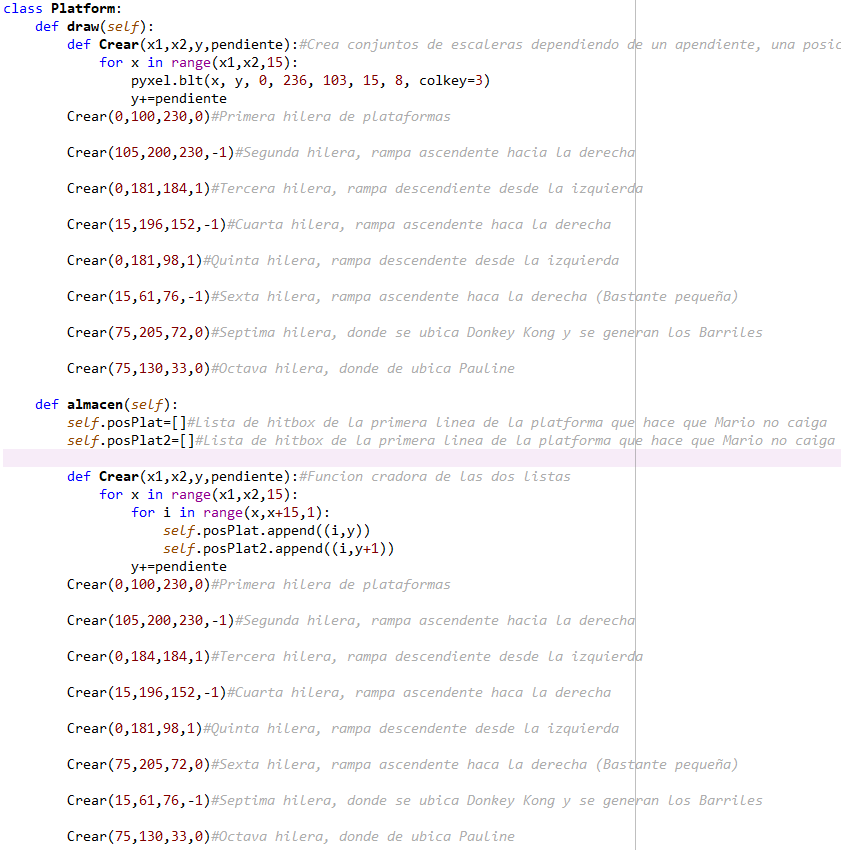
El objeto de clase Mario dentro del juego es el avatar del jugador y entre sus características está la de moverse hacia izquierda o derecha, saltar y subir escaleras.

En la función **\_\_init\_\_** (función que se ejecuta al crear un objeto de esta clase y se usa generalmente para establecer los valores iniciales del objeto), se establecen una posición “x” e “y” iniciales, “salto”, que será utilizado más adelante y determina la altura máxima del salto de Mario, y “Salto”, un estado que representa si Mario está saltando o no.

La función **updateMario** se utiliza para definir en cada frame lo que es la “hitbox” de Mario, con un bucle for que almacena el conjunto de posiciones “x” e “y” referido a los pies de Mario, como se ve en la imagen, el segundo bloque se utiliza fara que al intentar salir del mapa, Mario sea rechazado de vuelta a la pantalla.

Por último la función **drawMario** imprime en la pantalla la imagen de Mario desde el archivo de imágenes (Explicado en el Tablero), en una la posición actúan de Mario, es decir en su “x” y su “y”.

Platform:

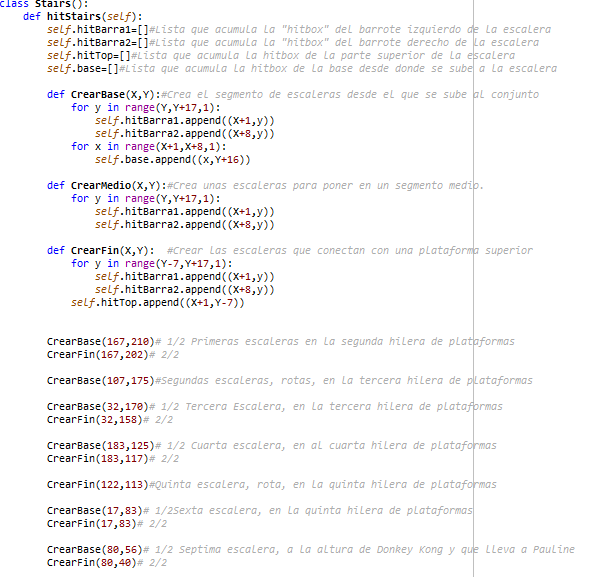


El objeto de clase Platform dentro del juego representa las plataformas sobre las que Mario puedes saltar y moverse, y donde los barriles caen y ruedan.

La función **draw** dibuja series de plataformas dentro del mapa gracias a su sub-función **Crear**, que sabiendo la pendiente (negativa o positiva siendo su valor lo que determina la distancia entre escalones), la y (posición y sobre la que se genera la primera plataforma), y su principio y final x1, x2). Con estos datos se usa un bucle for que cada cierta distancia y altura dibuja una plataforma, creando así hileras de plataformas que pueden crear rampas

La función **almacen** crea una dos listas en las que se almacena la posición “x” e “y” de cada uno de los puntos que componen la “hitbox” de las plataformas, siendo la “posPlat2” una lista que se utiliza para subir los escalones y la primera una que se usa para que ni Mario ni los barriles caigan al vacío. Para crear estas listas de forma óptima, se utiliza la función propia **Crear**, que utilizando el mismo método que en “draw”, pero en vez de pintarse una plataforma en un tablero, se añaden posiciones a las listas ya mencionadas.

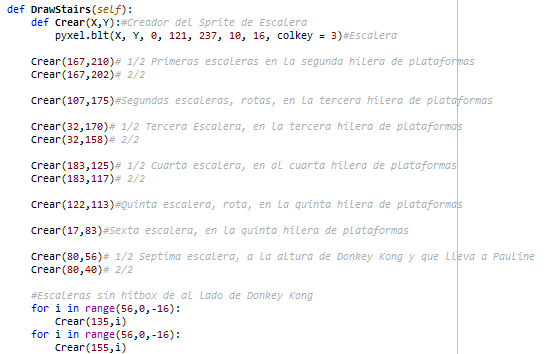
Stairs:



El objeto de clase Stairs dentro del juego representa a las escaleras que utiliza Mario para subir y bajar de un nivel de plataforma a otro y que utilizan los barriles para, de forma aleatoria, bajar de plataforma.

La función **hitStairs** crea unas listas que almacenan todo lo referente a “hitbox” que tienen las escaleras: “hitBarra 1 y 2” almacenan las posiciones “x” e “y” de los barrotes verticales de las escalera, que se usaran para que Mario las suba. “hitTop” almacena las posiciones “x” e “y” del punto final de las escaleras, los usan los barriles para detectar que hay unas escaleras y de forma aleatoria dejarse caer, y Mario para subirse a la escalera desde arriba y bajar. “base” almacena las posiciones “x” e “y” del comienzo de las escaleras, utilizadas para que cuando Mario la toque a la vez que los dos barrotes sea capaz de subirse a las escaleras.

Para rellenar estas listas de forma eficiente se usan las funciones **CrearBase, CrearMedio y CrearFin**, que utilizan un método casi idéntico al de la generación de hitbox en las plataformas solo que en vertical, la verdadera diferencia entre las distintas funciones radica en que CrarBase genera los barrotes y añade elementos a la lista base, CrearFin añade elementos a la lista hitTop y CrearMedio solo genera los barrotes para poder introducir secciones intermedias entre el principio y el fin (Realmente esta función nunca se usa pero en cuanto a un concepto de diseño de niveles más amplio me ha parecido correcto introducir esta variación de escalera.



La función **DrawStairs** dibuja en la pantalla todas las escaleras, para ello recurre a la función propia **Crear**, que simplemente dibuja en una posición “x” e “y” establecidas dibuja la imagen del banco de imágenes que corresponde a las escaleras.