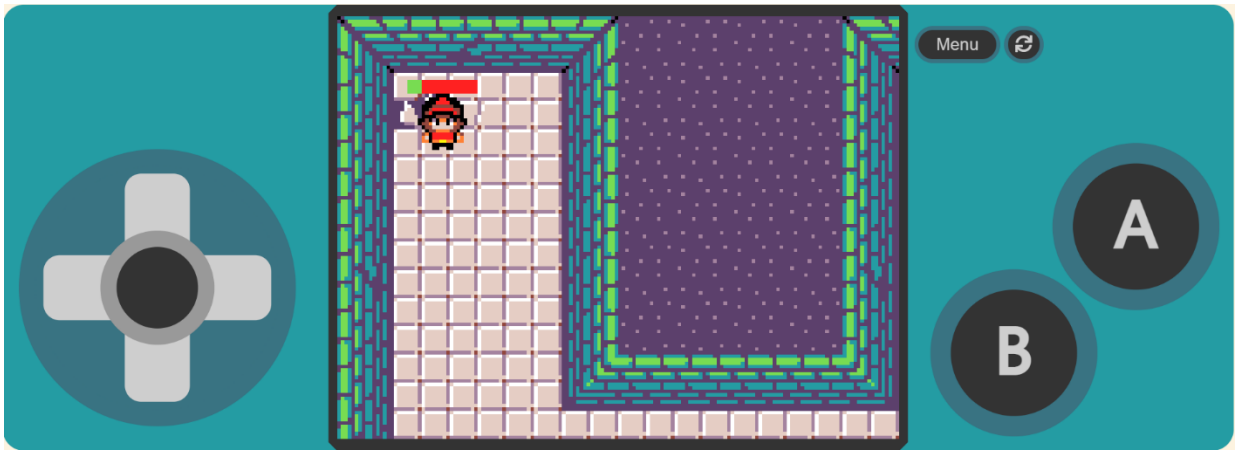
Primer ejercicio Arrays



Descripción

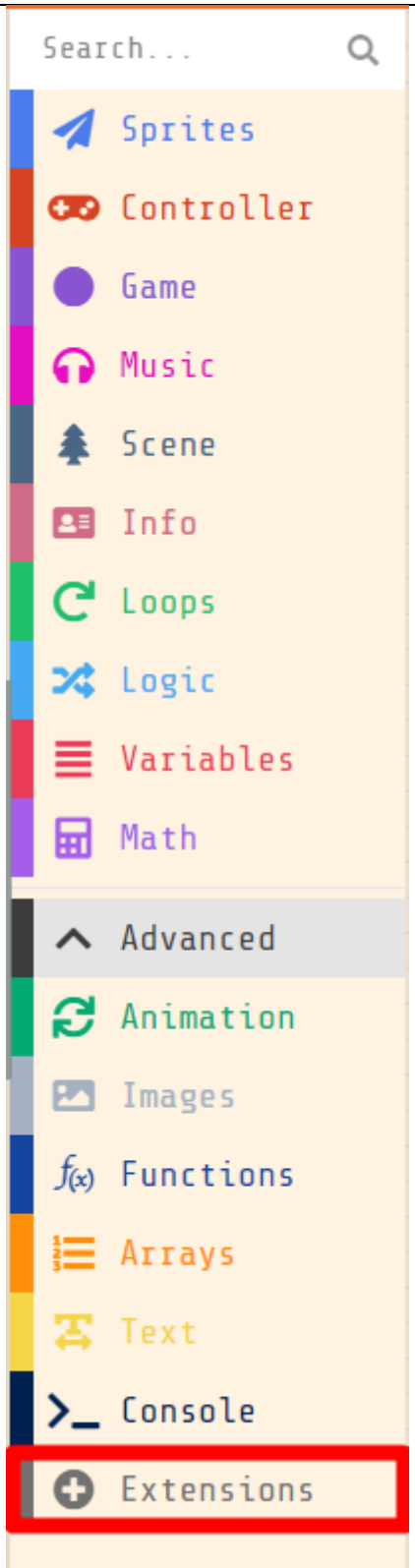
En este ejercicio crearemos un videojuego Zelda, donde nuestro protagonista tiene que explorar una mazmorra para encontrar la estrella de poder, pero no le será el camino fácil, ya que hay lava, muros infranqueables y un guardián de la estrella. ¿Serás capaz de llegar a la cámara secreta, derrotar al guardián y hacerte con la estrella?

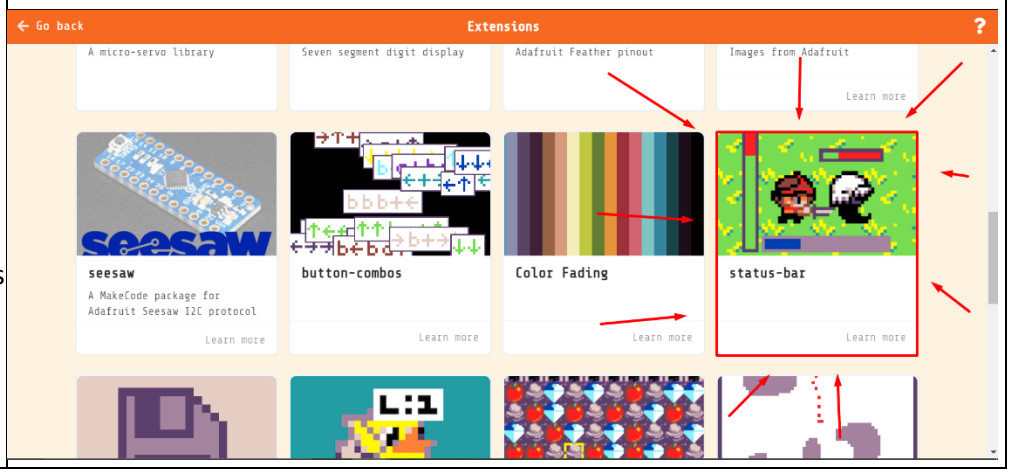
Para ello accederemos a **MakeCode Arcade** y realizaremos las operaciones necesarias.

Objetivos de programación y diseño

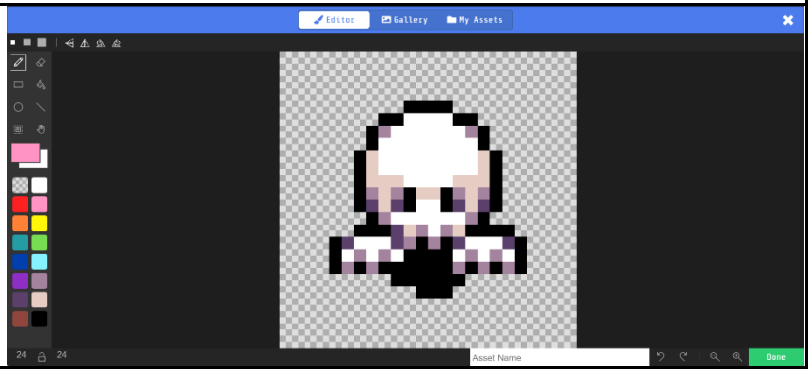
- Trabajar arrays mediante el desarrollo de un juego en MakeCode Arcade.
- Trabajar y perfeccionar las variables en MakeCode Arcade.
- Asignar estados del jugador.
- Descubrir más extensiones del MakeCode Arcade.
- Introducir al diseño de escenarios.

Antes de empezar

INSTALACIÓN DE EXTENSIONES NECESARIAS	
Para instalar una extensión, hacemos clic abajo del todo en **Extensions**.	

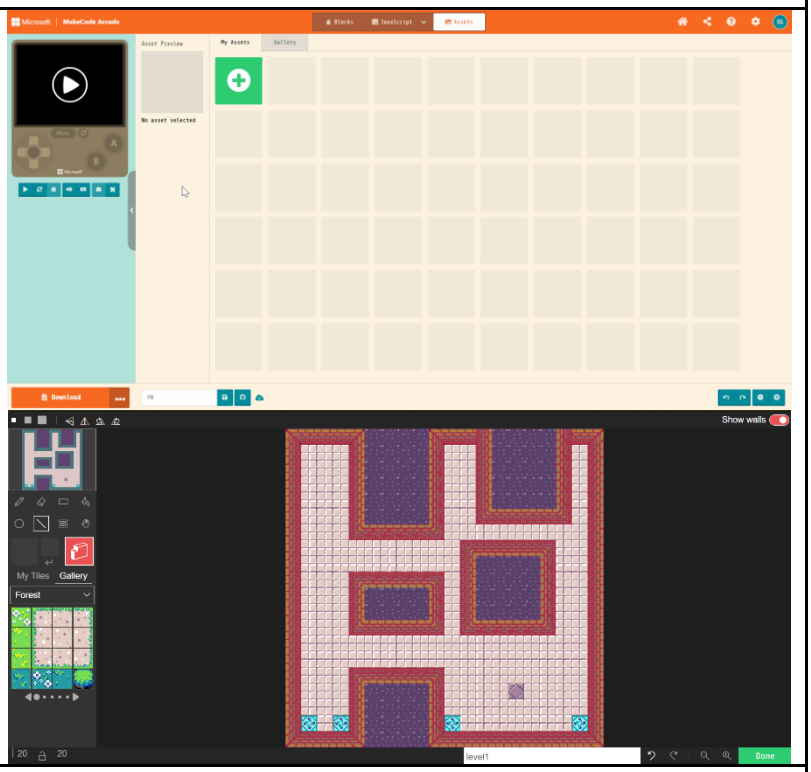
Después realizaremos la misma operación y seleccionaremos “status-bar”.	

Programación del juego

CREACIÓN DE ASSETS	
CREACIÓN SPRITE PROTAGONISTA	
Te recomendamos utilizar una matriz de 16x16 px para el Sprite de hero.	
CREACIÓN SPRITE ANTAGONISTA	
Te recomendamos utilizar una matriz de 24X24 px para el Sprite de guardian.	

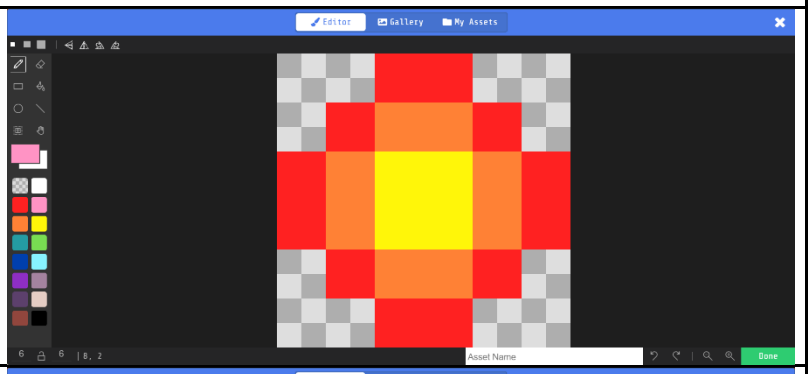
CREACIÓN TILEMAP

Te recomendamos utilizar una matriz de 20x20 px.

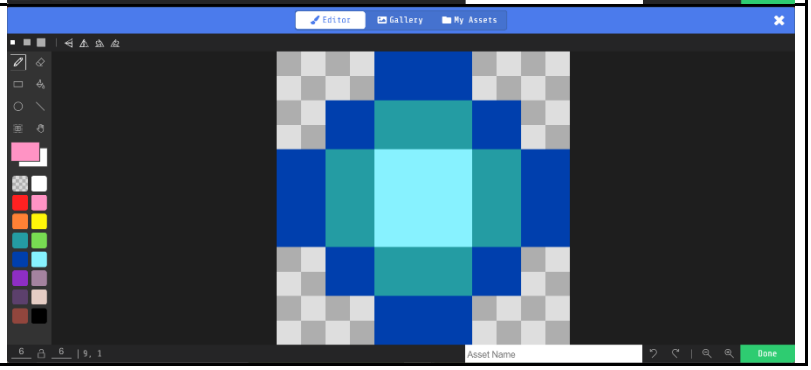


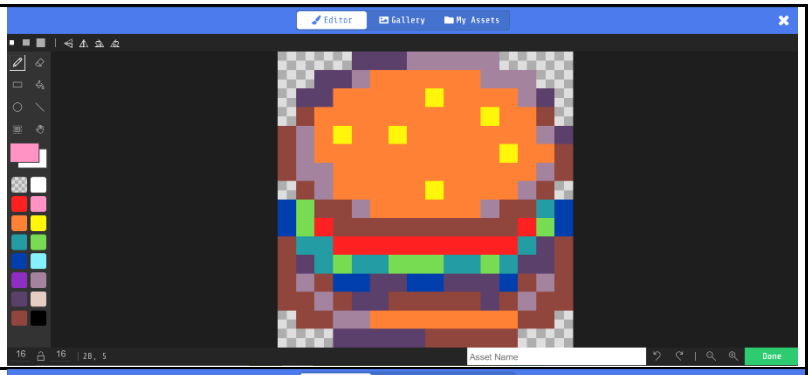
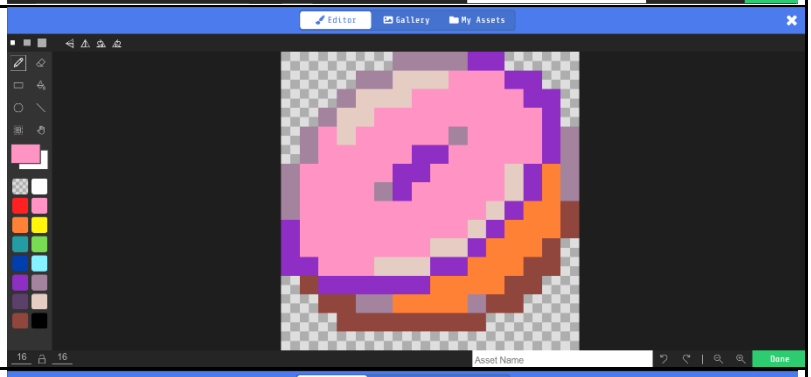
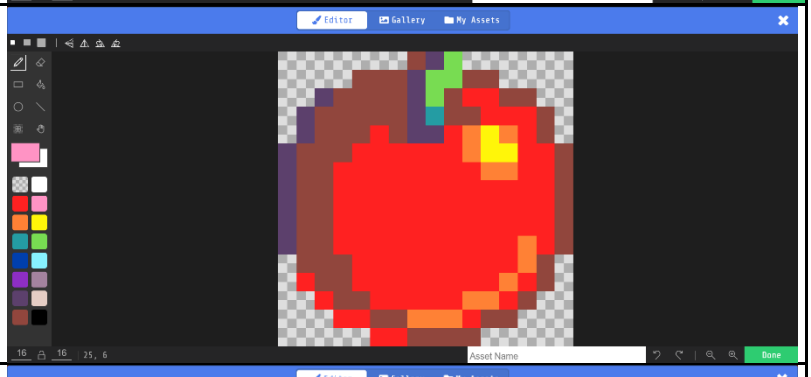
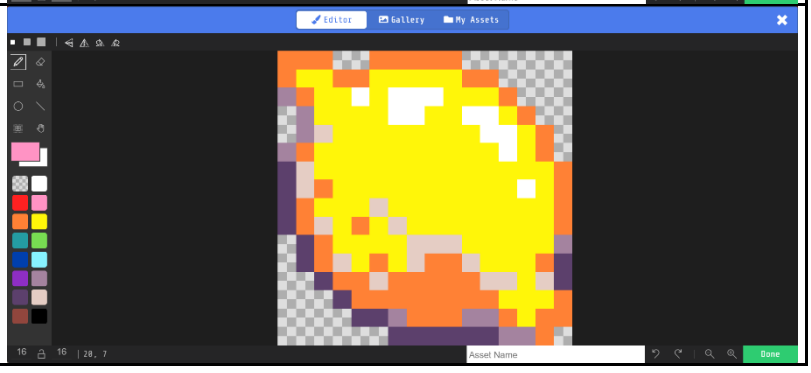
CREACIÓN SPRITES ADICIONALES

Te recomendamos utilizar una matriz de 6x6 px para el Sprite de fireEnemy.



Te recomendamos utilizar una matriz de 6x6 px para el Sprite de projectile.



Te recomendamos utilizar una matriz de 16x16 px para el Sprite de powerUp1.	
Te recomendamos utilizar una matriz de 16x16 px para el Sprite de powerUp2.	
Te recomendamos utilizar una matriz de 16x16 px para el Sprite de powerUp3.	
Te recomendamos utilizar una matriz de 16x16 px para el Sprite de powerUp4.	

PROGRAMACIÓN PRINCIPAL

CREACIÓN INICIO DEL JUEGO

El primer paso es crear un bloque on start y en él meteremos muchas cosas.

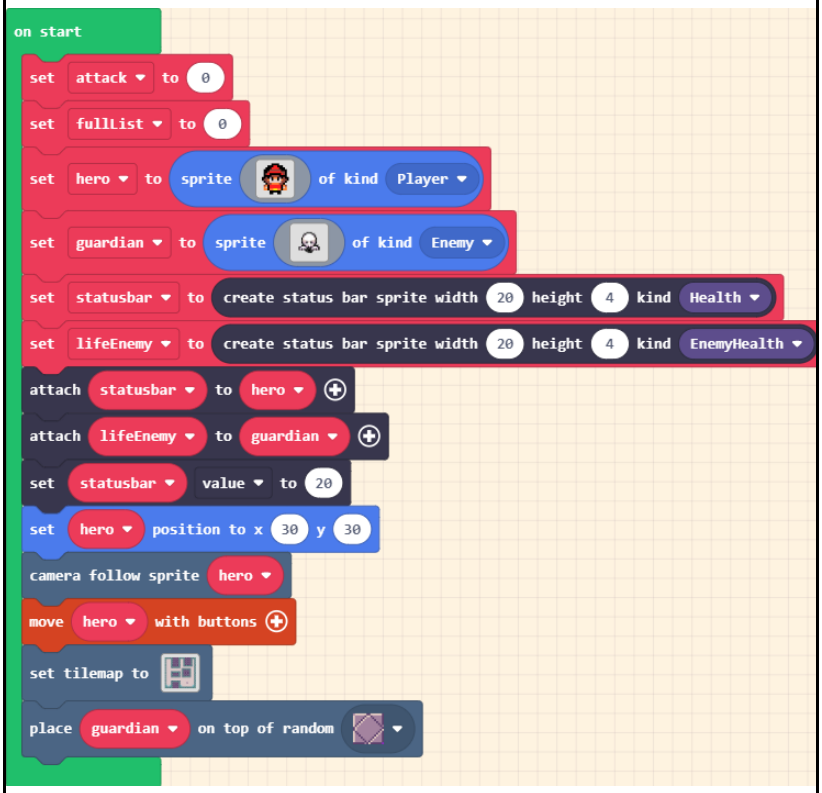
Empezaremos por crear las variables “attack, fullList, hero, guardian, statusbar y lifeEnemy”.

Luego estableceremos las variables “attack y fullList” a 0.

A continuación, colocaremos nuestros sprites “hero y guardian” y le daremos tipo Player y Enemy respectivamente.

Seguidamente, crearemos las barras de vida del protagonista y del enemigo, para ello, estableceremos las variables “statusbar y lifeEnemy” de tipo Health y EnemyHealth respectivamente. También colocaremos dichas barras en los sprites “hero y guardian”. Establecemos a la barra de vida del protagonista el valor 20, para reflejar su poca salud y así tener que recoger la comida.

Por último, establecemos la posición del protagonista, que la cámara le siga, que se mueva con la cruceta, que se cargue nuestro mapa y que nuestro enemigo se coloque en la baldosa indicada.

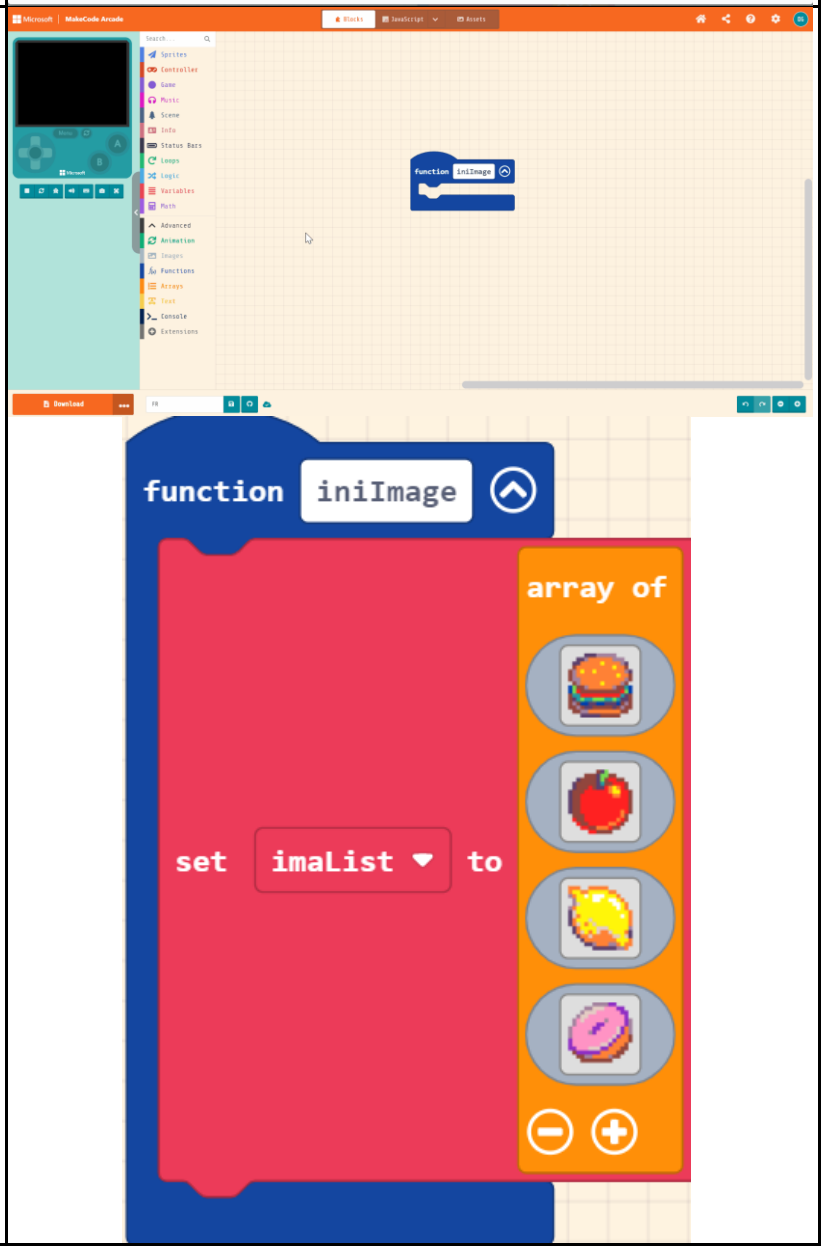


CREACIÓN DE LA MECÁNICA APARICIÓN DE LOS POWER UPS

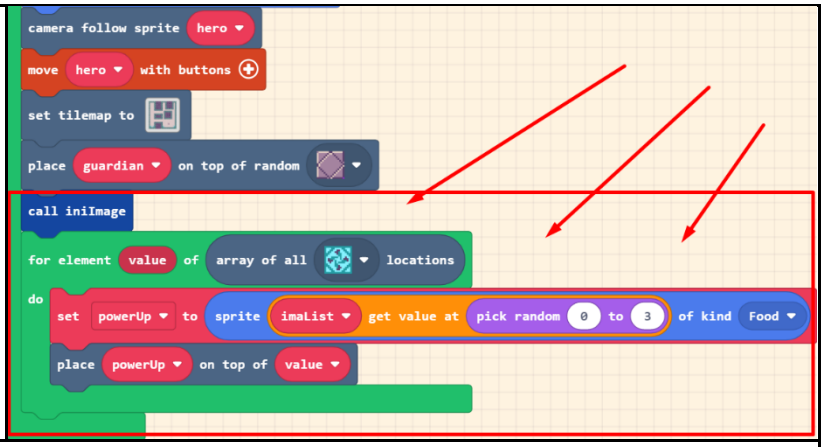
CREACIÓN DE LA FUNCIÓN iniImage

Vamos a crear una función donde guardaremos todas las imágenes que contendrán los power ups

Aquí colocaremos nuestra variable y como valor un array e introducimos todos los sprites de power ups.



Después de crear la función, la llamaremos en el bloque on start justo debajo del todo.
A continuación, colocamos un Loops en el que indicamos que para las losas que coincidan con la mostrada, colocas encima un power up de forma aleatoria, que hay dentro de nuestro array y lo pones de tipo Food.

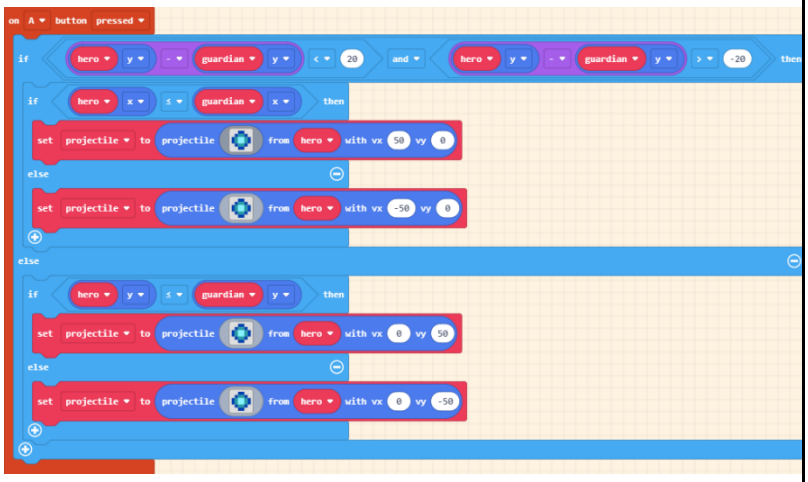


CREACIÓN MECÁNICA RECUPERAR SALUD

Ahora, para que nuestro protagonista recupere salud, le diremos al juego que cada vez que el sprite de tipo Player colisione con uno de tipo Food éste, en función de cual sea su sprite, obtendrá diferente salud. Para logra esto lo hacemos con if.

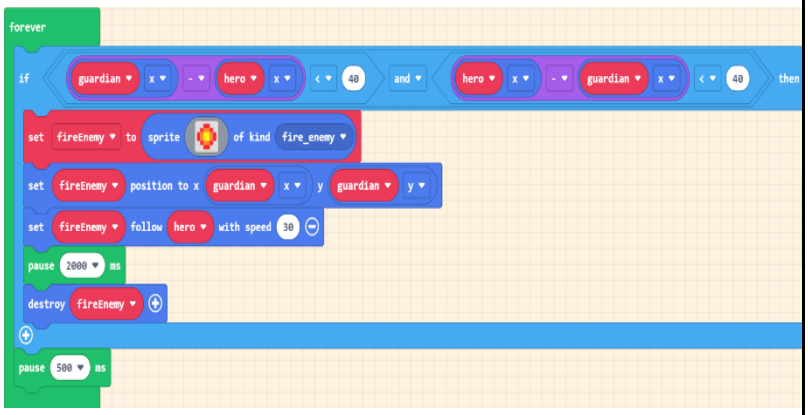
CREACIÓN MECÁNICA DISPARO DEL JUGADOR

Primero le diremos que cuando presione el botón A haga las siguientes comprobaciones antes de disparar para saber en que dirección:
Primero nos aseguraremos de estar en la misma línea horizontal (aproximadamente) de nuestro enemigo. Para ello, con un condicional, le decimos que compruebe que la diferencia entre la y de nuestro protagonista y enemigo sea menor que 20 y mayor que -20. Si es así, hará una segunda comprobación en la cual verificará si nuestro protagonista está a la izquierda o derecha de nuestro enemigo y en función a ello disparará hacia un lado u otro.
Si no está en línea con nuestro enemigo, comprobará si está arriba o debajo de él y disparará en función a ello.



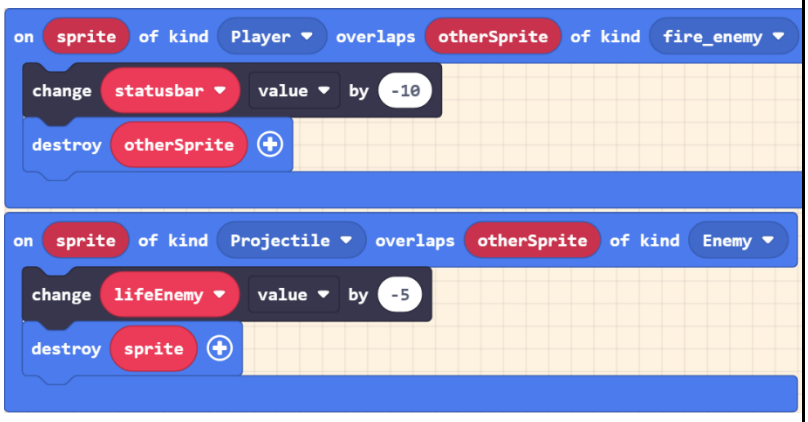
CREACIÓN MECÁNICA DISPARO DEL ENEMIGO

Nuestro enemigo es muy poderoso, con lo cual su disparo es peculiar ya que nos seguirá allá a donde vayamos.
Para ello, dentro de un forever, meteremos un condicional donde comprobará que nuestro protagonista esté cerca del enemigo y si es así, disparará y la bola de fuego seguirá a nuestro protagonista durante 2 segundo después se destruirá. Para que no dispare muy rápido, le daremos una pausa de medio segundo entre comprobación y comprobación.



CREACIÓN MECÁNICA PERDIDA DE SALUD DEL PROTAGONISTA Y ENEMIGO

Ahora haremos que cada vez que un proyectil (enemigo o nuestro) reste salud.
Para ello empezaremos restando salud a nuestro protagonista. Con un bloque de código overlaps, haremos que el sprite de tipo fire_enemy colisione con tipo Player le reste 10 de salud y se destruya el sprite.
Para realizar lo mismo con nuestro enemigo, replicamos el mismo bloque y cambiamos los tipos de sprites: Player por Projectile y fire_enemy por Enemy y le restamos 5 a la salud de nuestro enemigo.



CREACIÓN MECÁNICA DE FINAL DEL JUEGO

El último paso es hacer que acabemos la mazmorra. Para ello le indicaremos al juego que si se acaba la barra de salud de nuestro enemigo haga que ganemos el juego y si la barra de salud del protagonista se acaba, haga que perdamos la partida.

Con esta programación, nuestro jugador tendrá que derrotar al enemigo para ganar y salir de la mazmorra. Hemos aprendido a implementar barras de salud a los personajes, a utilizar las arrays para reducir variables al poder meter muchos valores en una misma variable. También hemos aprendido un poco a como diseñar un escenario de juego.

Ahora, es tu turno de personalizarlo y añadirle contenido. Aquí te dejamos el nuestro para que te inspires un poco: https://makecode.com/_WPzA6gasbRTg

Glosario

Bucles: Secuencia de código que se ejecuta repetidas veces.

Ejemplo: Forever, while, for.

Condicionales: Secuencia de instrucciones que se ejecuten en función del valor de una condición.

Ejemplo: If, If…Else

If: Sentencia condicional que, según el resultado de una operación lógica, ejecuta una secuencia de instrucciones o se omite.

If-Else: Sentencia condicional que si se cumple ejecuta una secuencia de instrucciones y otra que omite.

If…Else if: Secuencia de condicionales en la cual, iremos pasando, de manera ordenada, de una condición a otra hasta que se cumpla una de ella.

Operadores de comparación: Son operadores que comparan un valor a otro y se introducen dentro de una condición.

Variables: Es un espacio asociado a un identificador, en ese espacio hay un valor que puede ser modificado.

Funciones: Es un subprograma que recoge un conjunto de instrucciones y pueden ejecutarse desde el programa principal haciendo una llamada a él.

Sprite: Un sprite es un elemento gráfico dibujado en un mapa de bits. Este elemento se le aplicará distintos atributos cómo posición, velocidad, aceleración, etc.

También suelen llamarse objetos, ya que son componentes que suelen cumplir con una función dentro del videojuego.

Son diseñados con un mapa de bits, que es una matriz en la que cada espacio representa un pixel.

Mapa de bits: Es una rejilla de píxeles que se usa para dibujar Sprites.

Jugador: Un participante en un juego

Escenario: Espacio donde se desarrolla el videojuego.

Muros: Objetos o espacios donde los distintos elementos del juego no pueden atravesar.

Ciclo de Vida: Duración de un elemento de un programa desde su creación hasta su destrucción.

Cámara: Objeto dentro de un escenario que sirve como vista del jugador respecto al juego.

Aleatoriedad: La generación de números que tienen la misma probabilidad de generarse.

Vida: Recurso que tiene el jugador para poder seguir en la misma partida. Una vez que terminen las vidas, la partida se acaba.

Game Over: La partida se ha terminado. Se suele mostrar puntuaciones y te pregunta si quieres jugar otra partida.

Narrativa: Parte de un videojuego que sirve para construir una historia.

Género de juego: Una clasificación de los videojuegos en función de su jugabilidad.

Efecto: Algo que se aplica sobre el escenario, objeto, personaje y más elementos para transmitir realismo o una sensación dentro del juego.

Paleta de Colores: Panel donde se dispone de una variedad de colores que permite seleccionar un color para poder aplicarlo en algún elemento del videojuego.