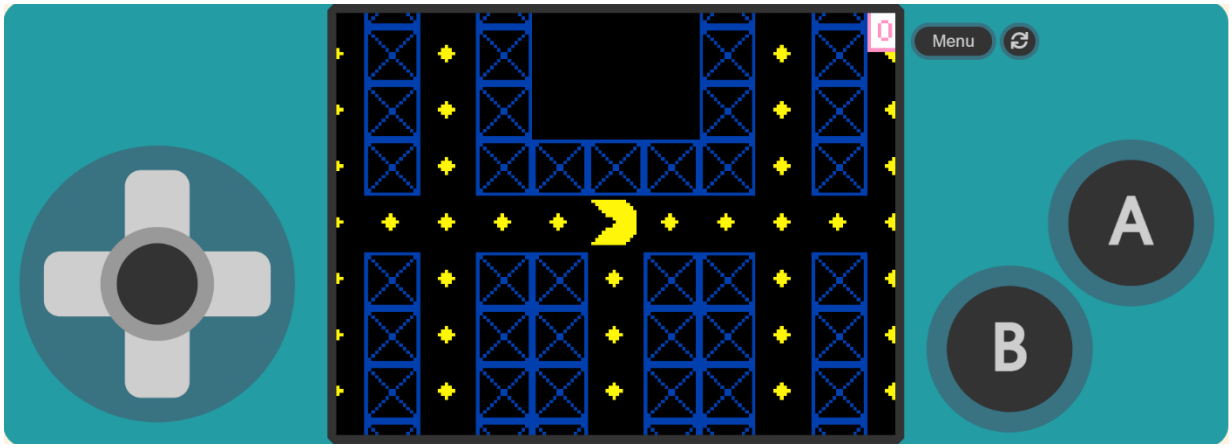
FIRST TILEMAPS TASK



Description

In this project, we will be creating the classic game of Pac-Man or Come Cocos. The main objective of the game is to eat as many coconuts as possible while avoiding getting caught by the ghosts.

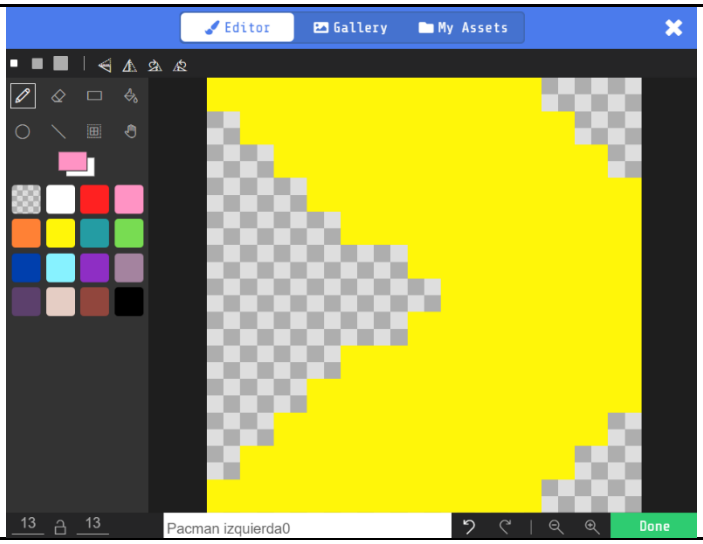
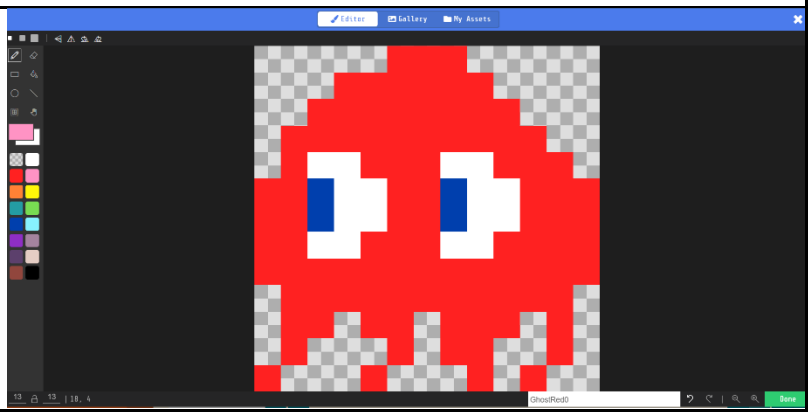
The concepts related to the essential aspects of a video game are:
Sprite design, the use of functions, or the use of colored tilemaps and replacing these colors with different sprites. We will also work on sprite animations, interactions with various elements of the game, and additional instructions that add quality to the video game.

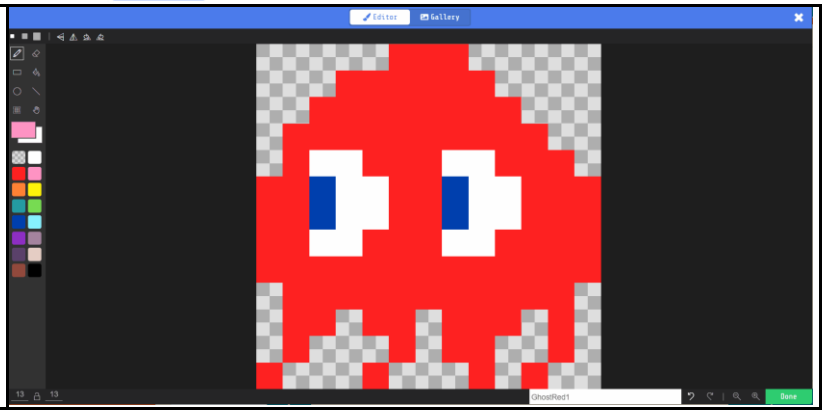
To do this, we will access MakeCode Arcade and perform the necessary operations.

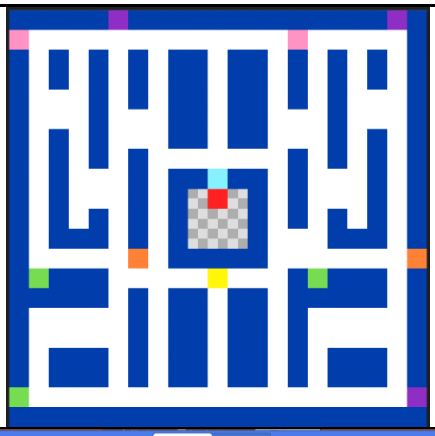
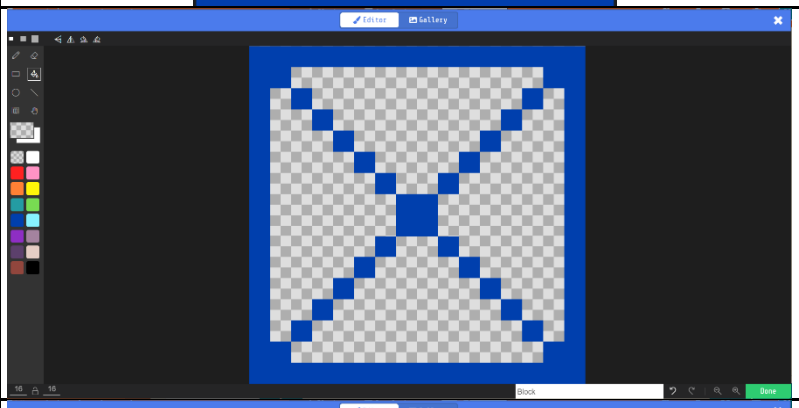
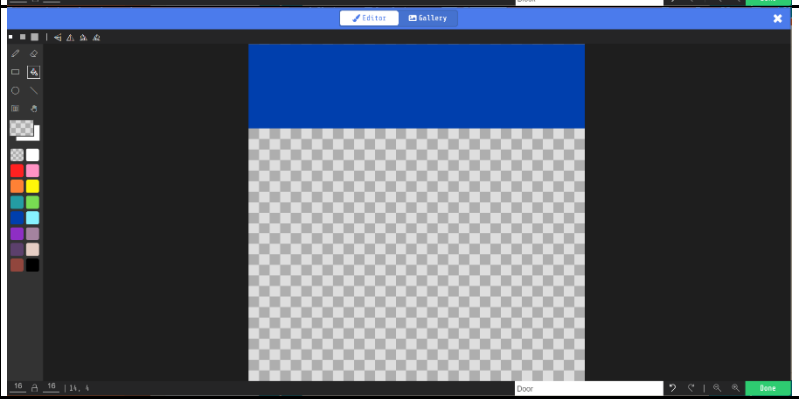
Goals

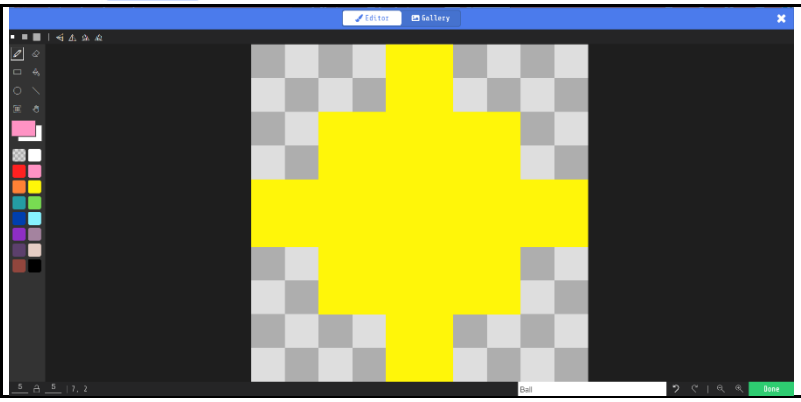
- Draw a map with colours and replace each colour with different elements for the game.
- Create a Sprite for our main character: "Pac Man" that we can control its movement.
- Create an animation for our character.
- Create Sprites for our opponents.
- Place opponents that patrol.
- Place an opponent that follows "Pac Man."
-

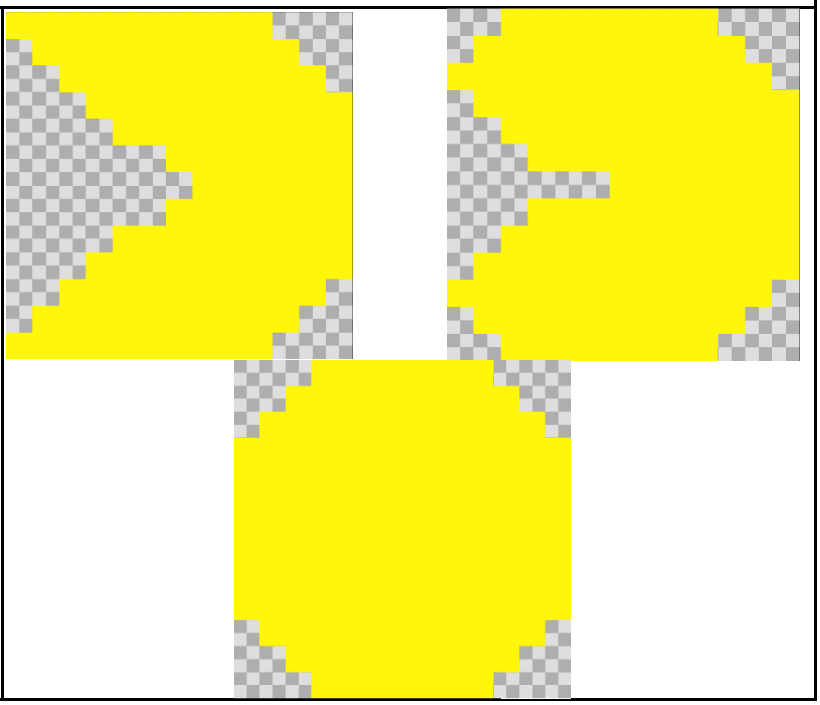
Programming guide.

Here you have part of the Assets and programming https://makecode.com/_D1hapJLR02hx
ASSETS CREATION
MAIN SPRITE CREATION
We recommend using a 13x13 pixel matrix for the " Pac Man " Sprite. 
ENEMIES SPRITE CREATION
We recommend using a 13x13 px for the ''GhostRed'' Sprite 

To create the rest of them, simply duplicate the previous sprite and change the base color to green, orange, and pink, respectively.	

TILEMAP CREATION	
Then we will go to the " set tile map to" block and create the scenario with different colours, which will be later replaced with different elements. The dimensions of the tile map are 21x21 px.	
Now we will create the Sprite for the , which will be 16x16 px and will replace the tile map.	
We create the Sprite for the door of our ghost zone	

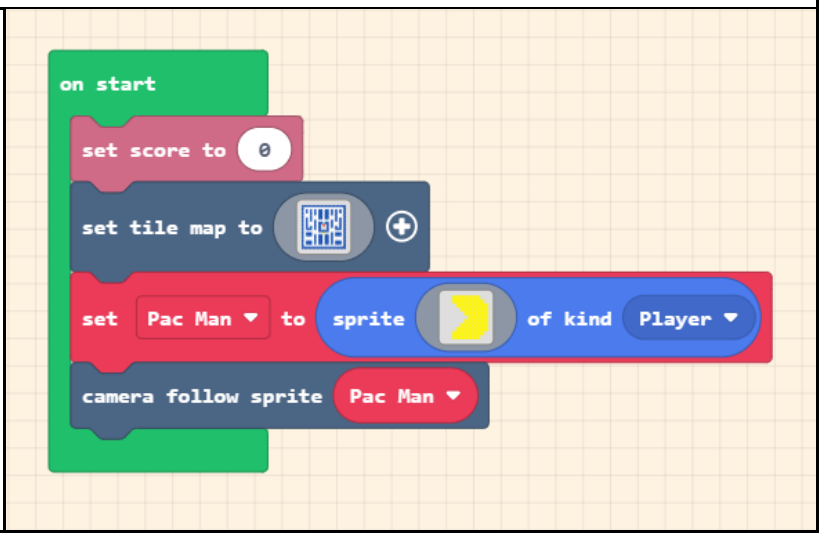
We create the Sprite for the sphere that Pac-Man will collect.	

ANIMATION SPRITE CREATION	
With our Pac Man Sprite, we will create the different Sprites that will compose the animation .	

MAIN PROGRAMMING

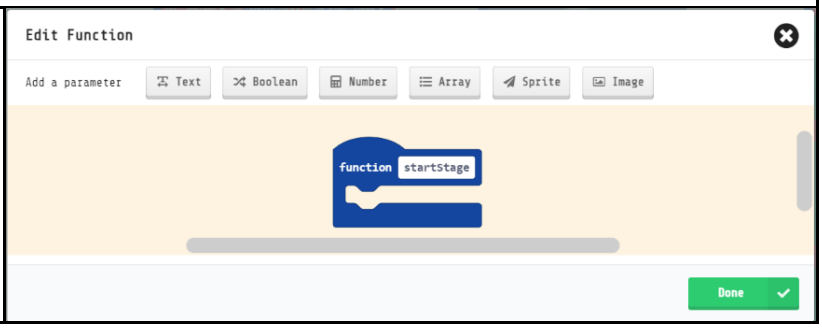
ON START GAME CREATION

To begin, we will create an " on start" block and place the following inside it: "set score to" and set it to 0, " set tile map to" (which we will create later), the " Pac Man" Sprite, and finally, camera follow sprite.



START STAGE CREATION

We create a “startStage” function.



First, we will set the background colour and then we will replace different colours with elements of the scene. We will create a 16x16 px block-shaped Sprite. We will also activate walls in the colours blue, green, purple, orange, and pink, so that the " Pac Man" Sprite cannot pass through them. The light blue and red colours will be left without a Sprite because we will replace them latLoer in a different way.

For the yellow colour, we will use the block " Place 'Pac Man' on top of random". This will make "Pac Man" start at that point.

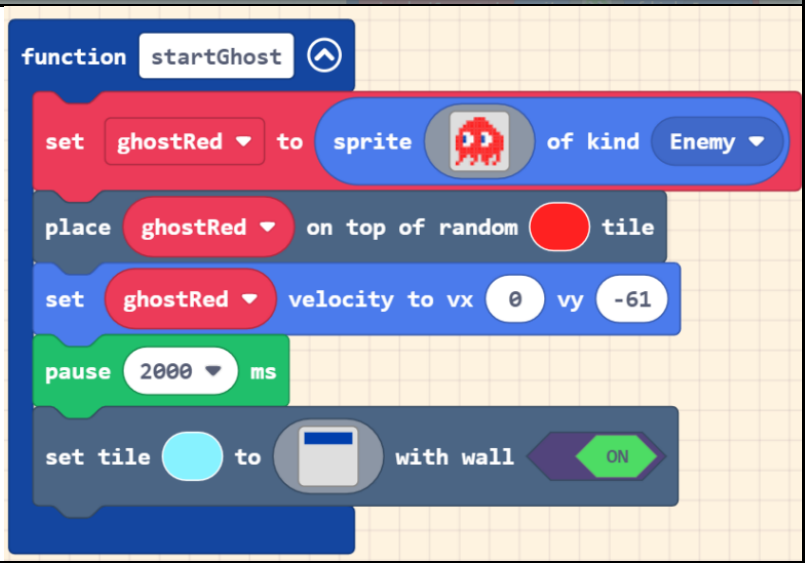
Next, in the " Loop " category, we will use the block "for element value of list ". Inside the "list" block, we will use the " array of all tiles" block, so that the programming inside the loop runs for each white area.

When the programming detects these white areas, it will create a 5x5 px sprite called " Ball" and place it in these areas.

GHOST SPAWN FUNCTIONS

We create the “startGhost” function.

We will place the " ghostRed" Sprite and right below it, use the block " place ‘ghostRed’ on top of random " to replace the red color. We will set a vertical speed of -61 to make it move upwards. In the " loops " category, we will add a pause of 2000ms to give the ghost enough time to exit its area. Finally, we will use the block " set tile ‘color 'light blue color' to a ‘sprite’ with wall ‘on ” to replace the light blue color with a sprite.



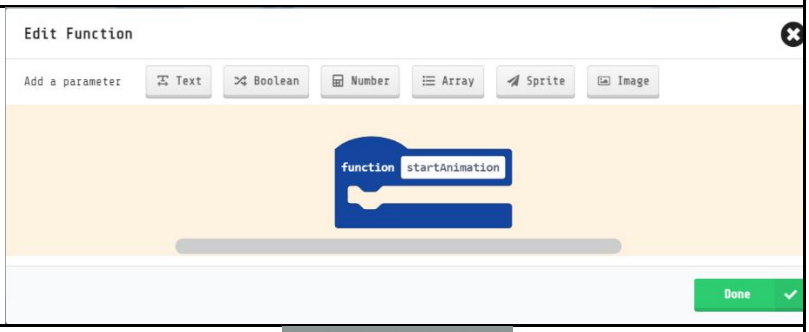
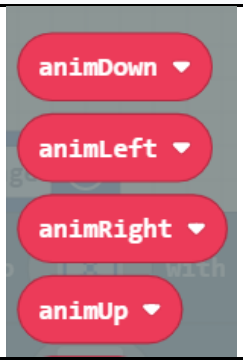
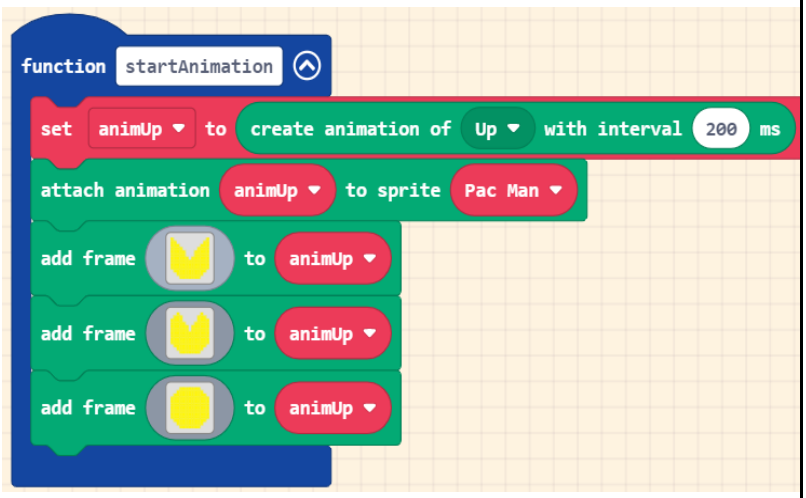
Next, we will place the rest of the opponents.

For " ghostGreen ", we will use the block " place ‘ghostGreen’ on top of ‘tilemap col 19 row 1'" and set a horizontal speed of -35.

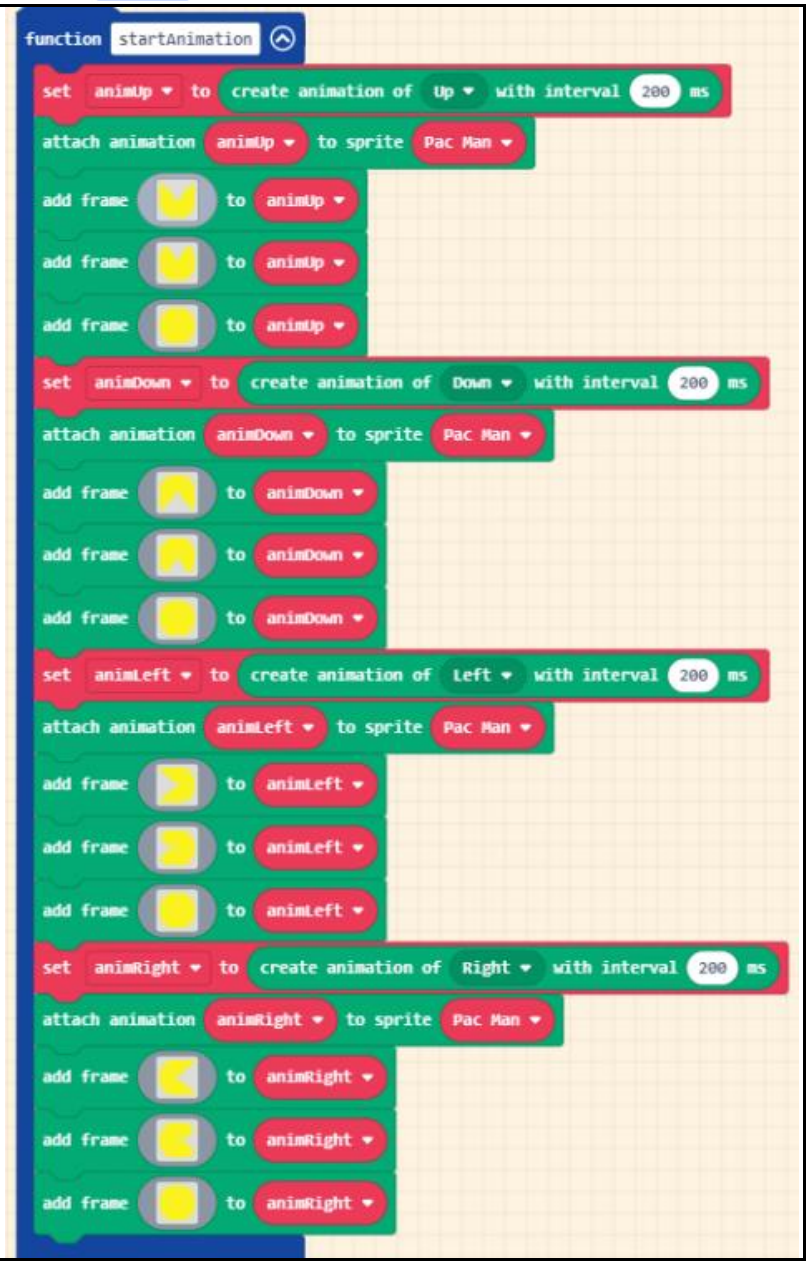
For " ghostOrange ", we will use the block " place ‘ghostOrange’ on top of ‘tilemap col 5 row 12 " and set a vertical speed of -35.

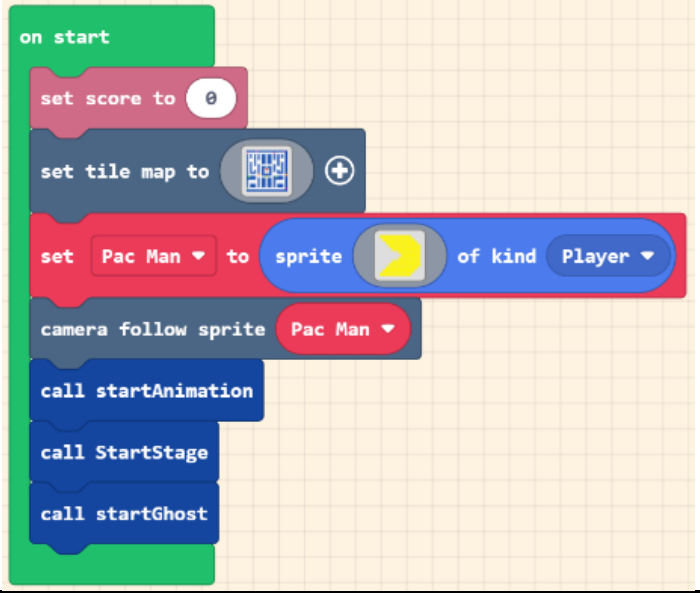
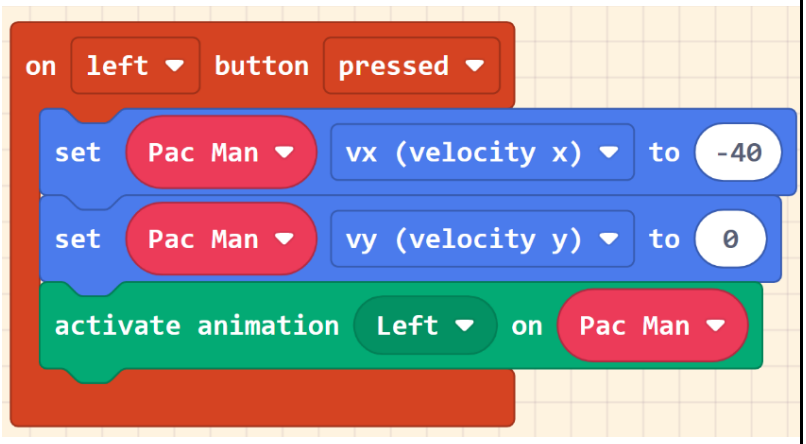
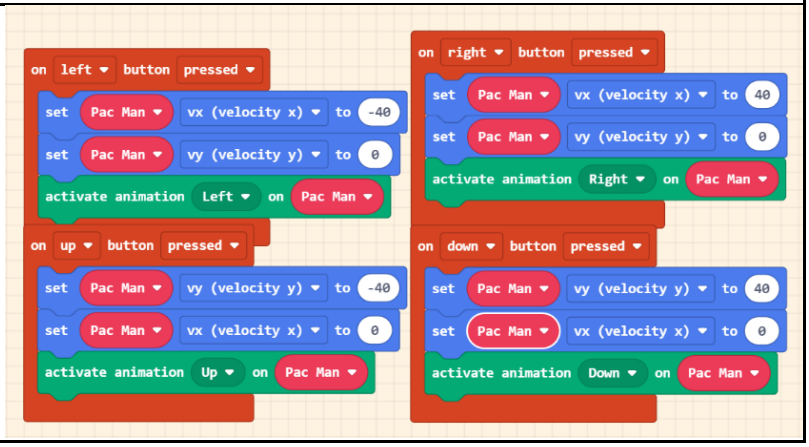
For "ghostPink", we will use the block "place 'ghostPink' on top of 'tilemap col 19 row 19'" and set a horizontal speed of -35.

START ANIMATION CREATION

Now we will create the final function " startAnimation", and with it, we will control the different animations that our Pac Man will have based on its movement.	
To start, we will set a series of variables to store the state of Pac Man, that is, to know if it is moving left, right, up, or down. " animLeft ", "animRight", "animUp" and "animDown " respectively.	
Let's start programming the upward animation. To do that, we'll begin with the Animation "set 'animUp' to create animation of 'Up' with Interval 200ms". This assigns a name 'Up' to the animation and assigns it to the variable 'animUp . Next, we'll place the following block: " ttach animation 'animUp' to sprite 'Pac Man ". This links the animation to our Pac Man character. Finally, we'll add the frames we have for the animation and assign them to our animation.	

After that, we replicate the entire block and make changes according to the different animations.



Once we have our 3 functions ready, we add them to our " on start" block using the " call startAnimation " block, the " all startStage " block, and the "call startGhost" block.	
PAC MAN MOVEMENT MECHANICS CREATION	
Now we will start with the controls for our 'Pac Man'. In the " Controller " section, we select the " on 'left' button 'pressed " block. To make Pac Man move to the left and only to the left, we set 'Pac Man' 'vx (velocity x)' to -40 and ' set 'Pac Man' 'vy (velocity y)' to 0. This ensures that when we change direction, Pac Man will move only in that direction. Ahora comenzaremos con los controles de nuestro 'Pac Man'.	
Now we repeat this block with different buttons and directions.	

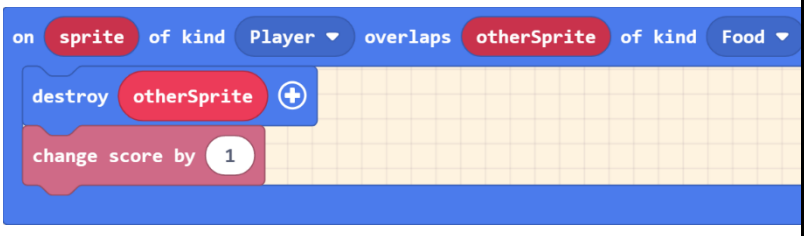
ENEMIES MOVEMENT MECHANICS

We're going to make our 'ghostRed' follow ' Pac Man '.
In the " Loops " section, we use the " forever " block and add " set 'ghostRed' follow' Pac Man' with speed 35". This will make the 'ghostRed' sprite follow 'Pac Man' at a speed of 35.

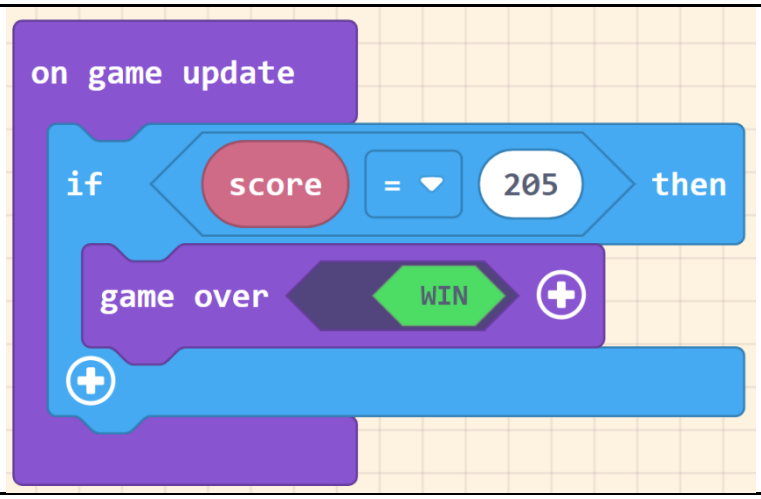
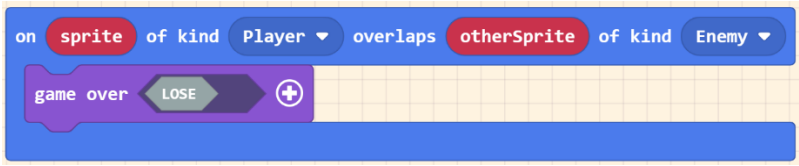
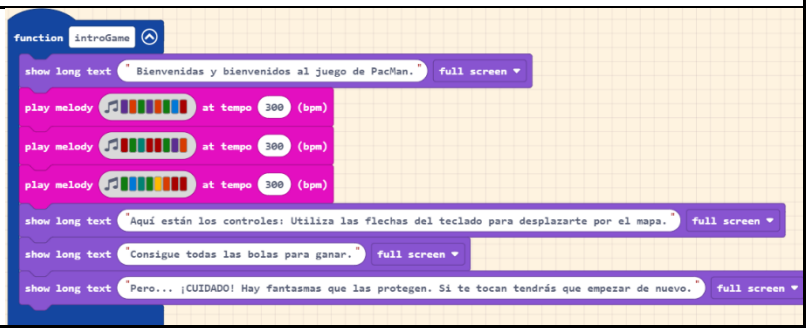
We're going to create the patrol behavior for the rest of the ghosts. We'll create a collision block for each color.
Let's start with "on 'sprite' of kind 'enemy hits wall 'pink colour '" and then add "set 'sprite' velocity to vx 0 y vy 35".
Next, we continue with the following colors with the appropriate changes.
" on 'sprite' of kind 'enemy hits wall 'purple colour '" and then add " set 'sprite' velocity to vx -35 y vy 0".
" on 'sprite' of kind 'enemy' hits wall 'orange colour" and then we add"set 'sprite' velocity to vx 0 y vy -35'"
" on 'sprite' of kind 'enemy' hits wall 'green colour '" and then add " set 'sprite' velocity to vx 0 y vy -35".

EATING MECHANIC INTERACTIONS

Now, we will make our Pac Man able to eat the orbs to score points.
" on 'sprite' of kind 'Player' overlaps 'otherSprite' of kind 'Food' ". This indicates that when the player overlaps with the food, do the following.
" destroy 'otherSprite'". This destroys the food orb.

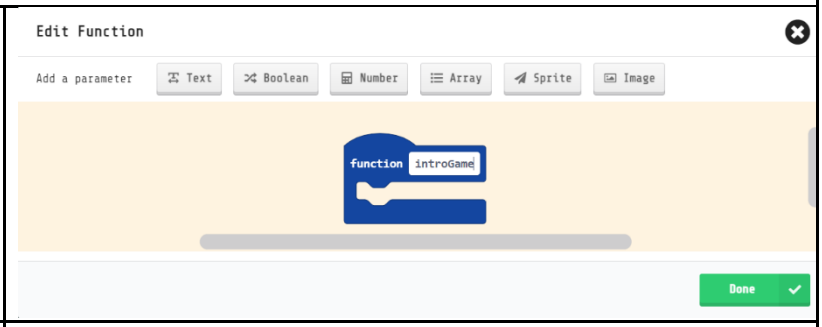


" change score by 1". This adds one point to our score counter.	

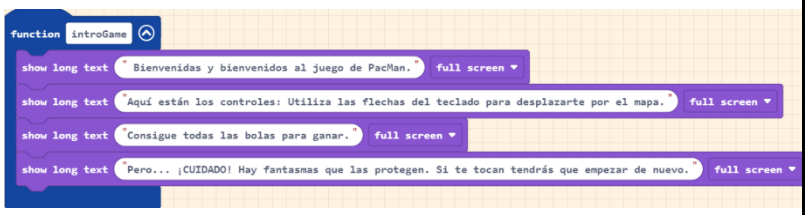
WIN AND LOSE CREATION MECHANICS	
When 'Pac Man' collects all the points, we need to end the game and declare victory. To do this, we will follow these steps: In the Game section, we will use the "on game update " block and inside it, we will add a Logic block with the condition "if 'score = 205' then". Inside this condition, we will add the block "game over 'win'" from the Game section.	
Finally, we will make our Player lose the game and have to start over when it collides with any Enemy. " on 'sprite' of kind 'Player' overlaps 'otherSprite' of kind 'Enemy ". And then add a " game over 'lose'" block to indicate that the game is lost.	
LAST DETAILS	
To provide context to the game and create a more immersive experience, we can add explanatory texts and a short introductory music.	

INTRO FUNCTION CREATION

First, we will create a function called introGame " where we will place our texts and the introductory melody.

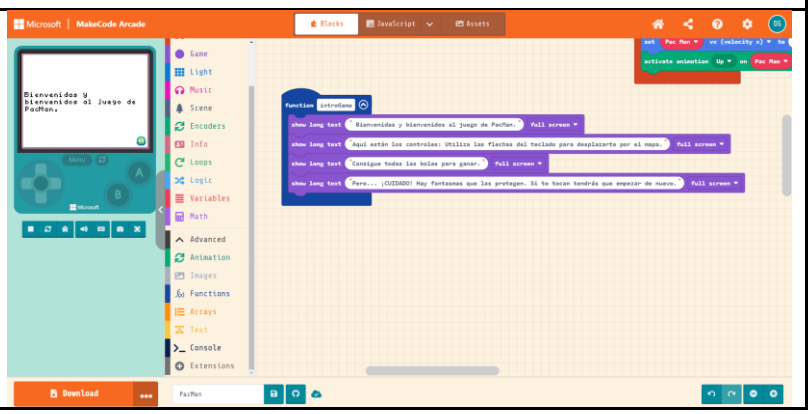


To start introducing the texts, we will go to " Game " and use the " show long text " block. Enter the desired text within the block and select the option to display it full screen. Repeat this step for each text you want to include.

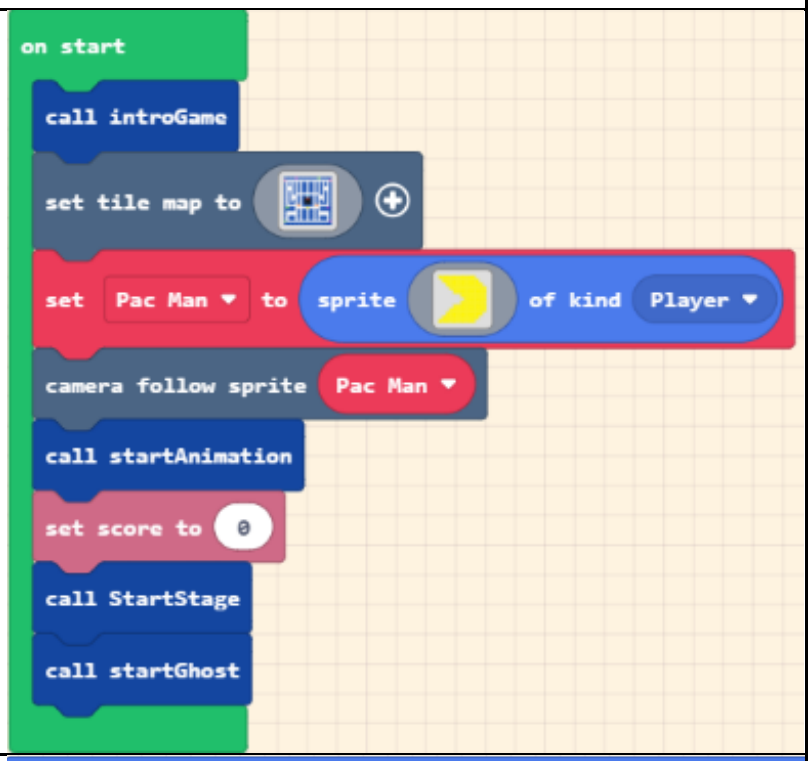
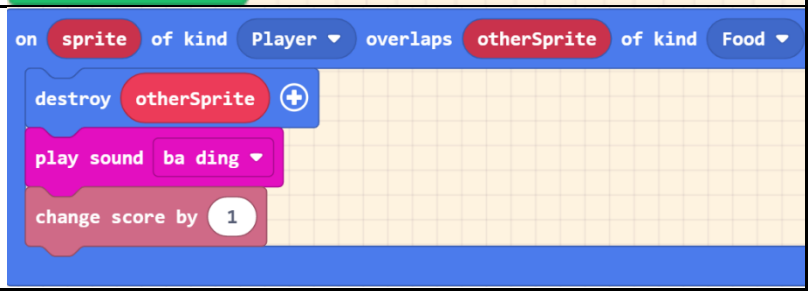


MELODY CREATION

To add melodies, go to " Music" and use the " Play Melody 'notes' at tempo 300 (bpm) " block. Enter the desired musical notes within the block. If you want to make the melody longer, simply add more melody blocks. With tempo we can change the speed of the .



LAST STEPS

Finally, we add the " call ‘introGame ‘" block to the " on start " block at the beginning of the code to start our introduction.	
The final detail is to add a sound every time Pac Man eats a sphere. To do this, go to " Music " and use the "play sound ‘ba ding ‘" block. Place this block within the code block that detects the collision between Pac Man and the sphere.	

Thanks to this programming, we have created a Pac-Man game in which we have learned to create tilemaps, animations, functions, and movements. Now it's time to enjoy our effort and, of course, give it our personal touch.

Glossary

Event: Executes a sequence of instructions when an external event occurs in the system.

Functions: A subroutine that contains a set of instructions and can be called from the main program.

Comparison Operators: Operators that compare one value to another and are used within conditions.

Conditionals: Sequence of instructions that are executed based on the value of a condition.

Acceleration: The rate of change of velocity per unit of time.

Velocity: A physical quantity that relates position to the rate of change of time.

Scene: The space where the game takes place.

Walls: Objects or areas where the different elements of the game cannot pass through.

Camera: An object within a scene that serves as the player's view of the game.

Game Over: The game has ended. It usually displays scores and asks if you want to play again.

Animation: The illusion of motion created by displaying a series of frames of a sprite.

Music: Combination of sounds and silences that compose a rhythm.

Narrative: Part of a video game that helps build a story.

Colour Palette: Panel that provides a variety of colours for selecting and applying them to game elements.

Score: Total points obtained by a player through certain interactions.