

## Smash - Blocks – Microsoft Arcade

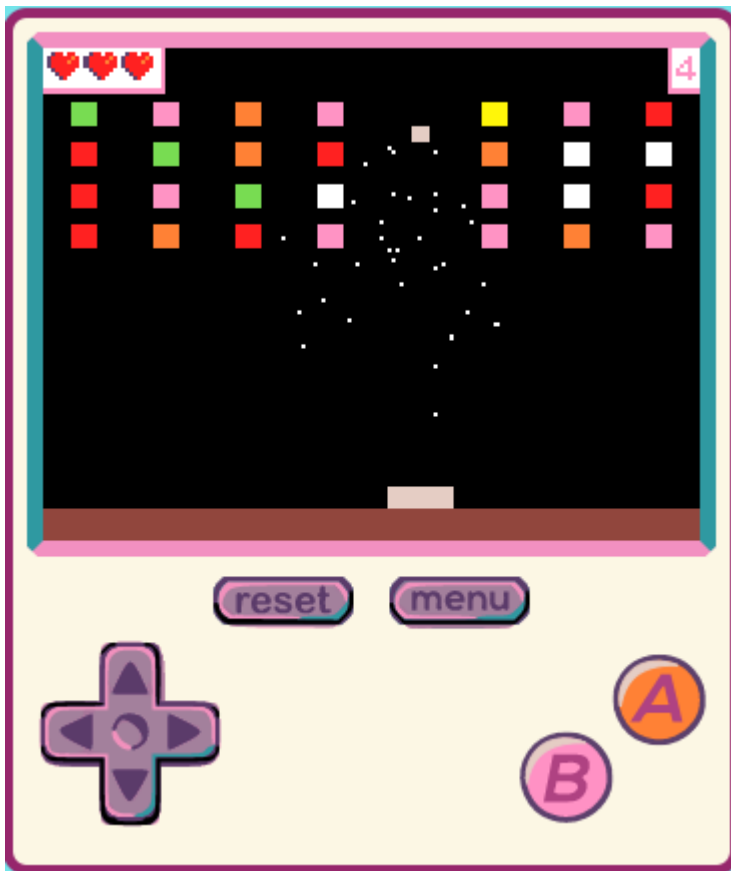
The following instructions will take you through the steps of creating a simple breakout clone where the aim of the game is to smash all the coloured blocks along the top of the screen by hitting them with a ball. The player controls a paddle that can be moved left and right along the bottom of the screen to “hit” the falling ball back up to the blocks.

A point is scored for each block that is smashed.

A life is lost each time the ball is “dropped”; i.e. not hit by the paddle.

The game ends when all the player lives are lost.

To start the game, serve the ball by pressing A.



### Controls

To control the paddle, use the left and right direction keys on the console. Alternatively, it may be easier to use the keyboard mappings for the keys as follows. Use the A button to serve the ball.

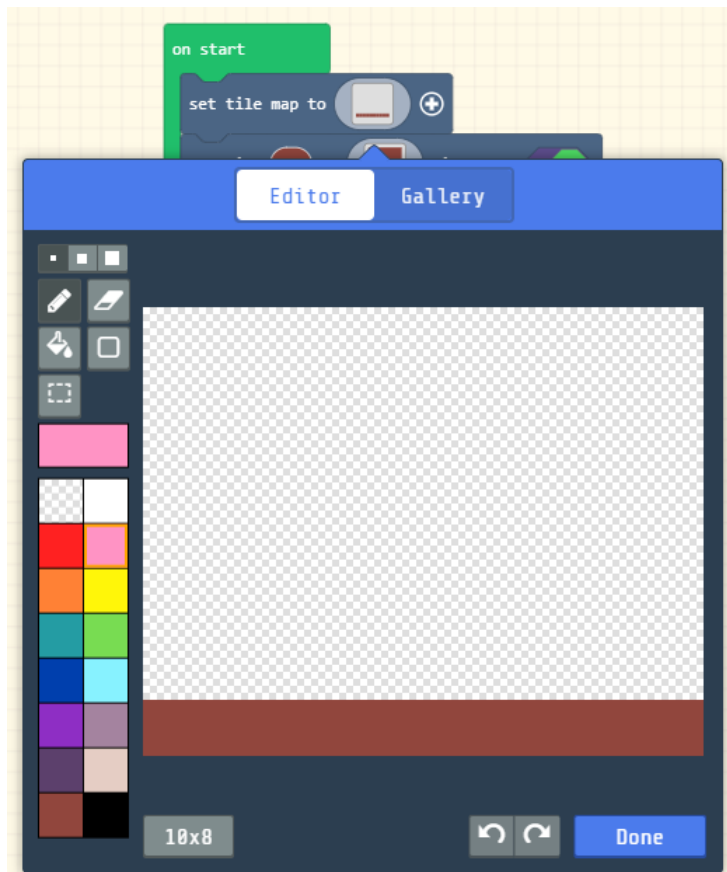
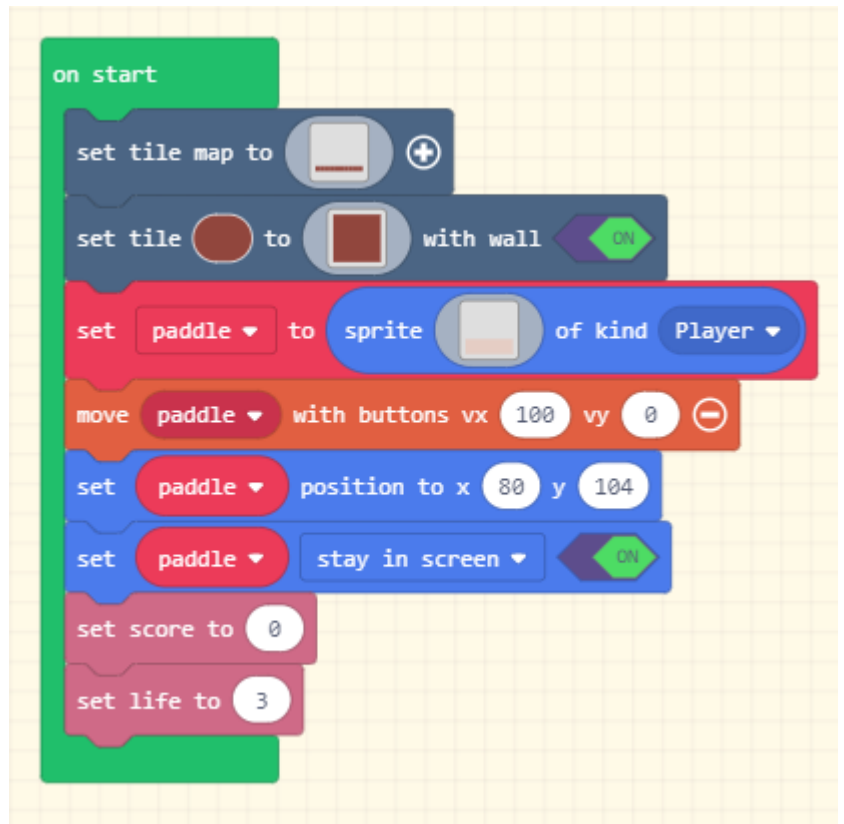
	<b>Up</b> (↑ or W)			
<b>Left</b> (← or A)		<b>Right</b> (→ or D)	<b>B</b> (E, X or Enter)	<b>A</b> (Q, Z or Space)
	<b>Down</b> (↓ or S)			

## Smash - Blocks – Microsoft Arcade

### Step 1 – Create the paddle and move

The first step in creating Smash is to setup the background and the paddle that the user will control. The tile map consists of two colours; the default background colour that we will leave black and a line along the bottom that we will use brown for. We set the brown tiles to be “walls” and use this fact later in the code to detect when the ball hits a wall (and the player loses a life). The code to write is given below and further screenshots follow that show the tile map and paddle sprite in more detail. The tile map should be a 10 x 8 tile map and the paddle should be a 16 x 16 sprite.

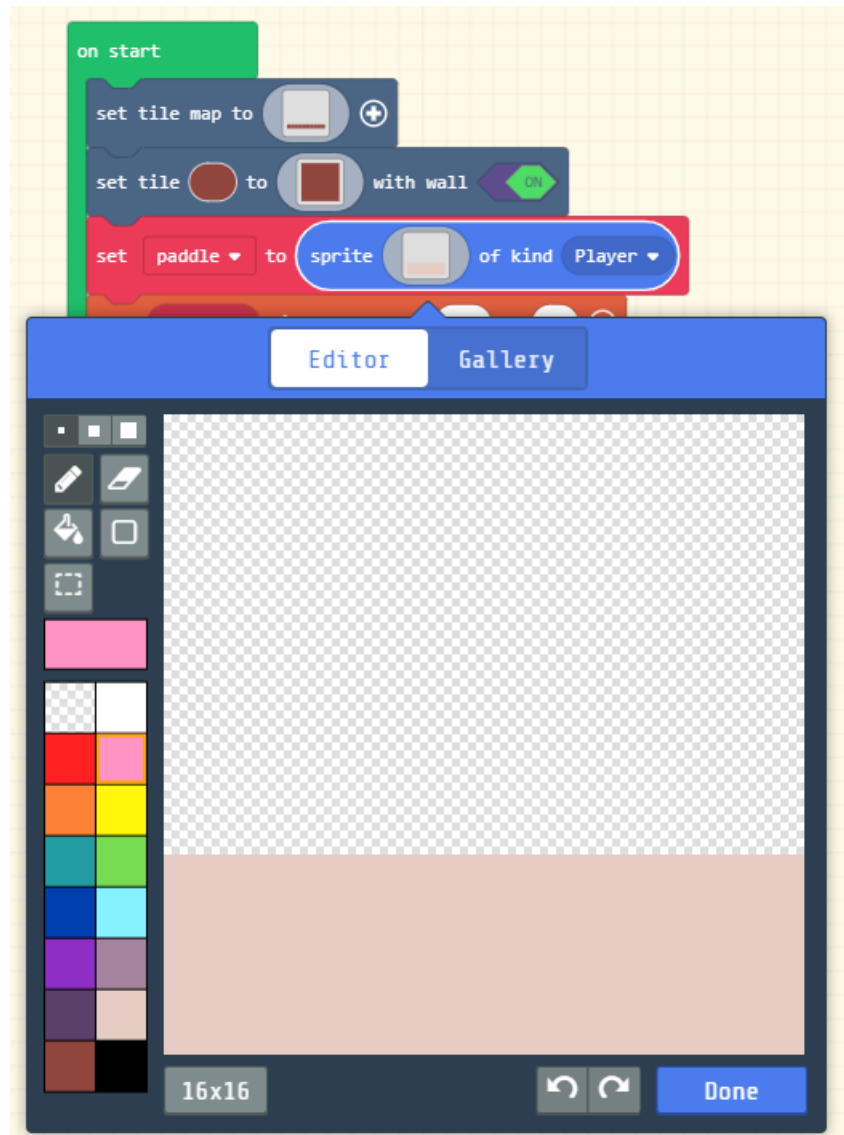
Don't forget to rename your sprite to paddle.



The screen shot to the left shows the detail of the tile map. Note that it is a 10 x 8 tile map and only the bottom row is brown.

## Smash - Blocks – Microsoft Arcade

The following screen shot shows the detail for the paddle sprite. Note that the sprite is 16 x 16 pixels and the bottom 5 pixels are coloured in. You can pick any colour you choose.



---

### ***Experiment***

*Run your game and move the paddle left and right. Make sure that the paddle cannot go outside the bounds of the screen and cannot go up or down.*

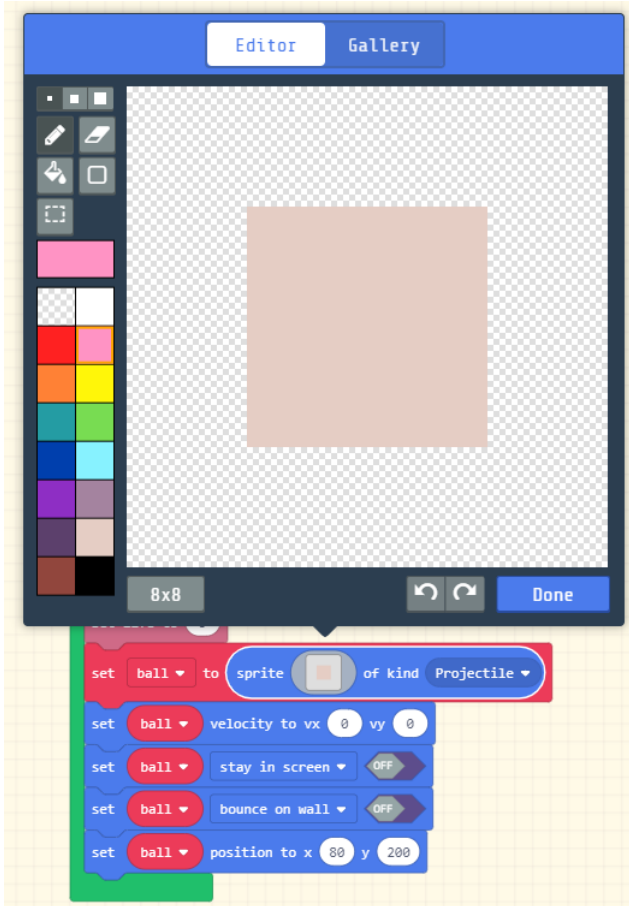
---

## Smash - Blocks – Microsoft Arcade

### Step 2 – Adding the ball

We are now going to add the ball sprite; it won't do anything yet. Add the following code (highlighted in the red box in the diagram to the right) to your existing code in the "on start" loop just below the code you wrote in step 1.

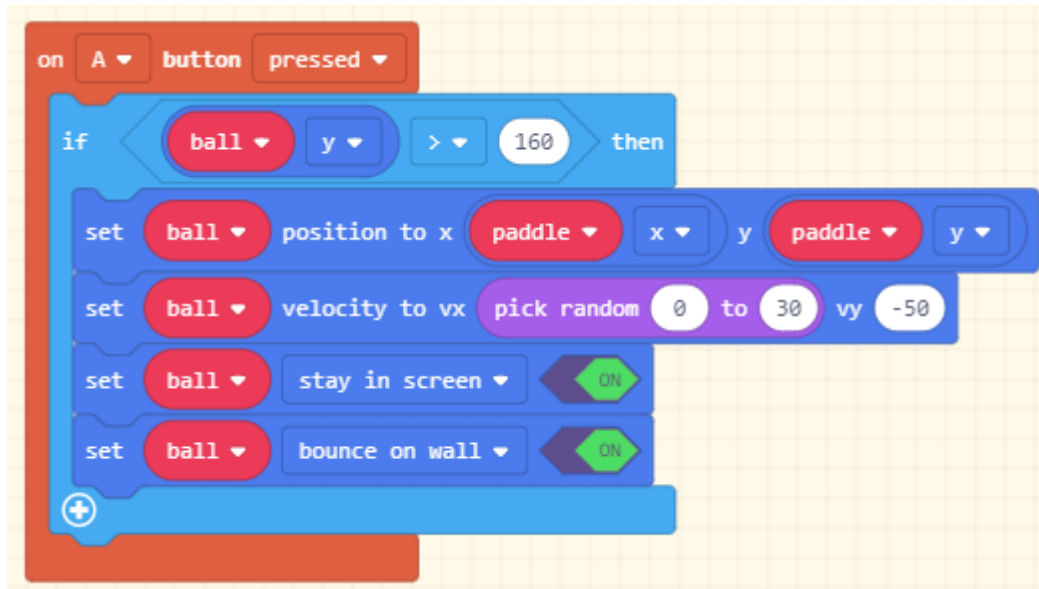
As before, don't forget to rename your sprite; in the example code we renamed it to "ball". The ball sprite is 8 x 8 pixels with the middle 4 x 4 pixels coloured in as shown in the image below. You can choose any suitable colour for the ball.



## Smash - Blocks – Microsoft Arcade

### Step 3 – Serving the ball

The player will start a new game or continue an existing game after they have dropped the ball by serving the ball upwards towards the top of the screen. Add the following code to your program.



---

### **Experiment**

*If you run the game now, what happens? Why do you think it happens?*

---

## Smash - Blocks – Microsoft Arcade

### Step 4 – Hitting the ball with the paddle

Now we need to add the code that allows the player to “hit” the ball with the paddle to send it back up towards the blocks. This is a little more complicated than it at first seems. The following explains what the code to the right is doing.

The code is executed if the ball (which is a projectile) touches the paddle (which is the player)

The vertical direction of the ball is changed so that it travels upwards.

If the player is pressing either the left or right buttons, the horizontal direction of the ball is adjusted in the corresponding direction. This allows the player some element of control of the ball.

If the horizontal velocity (either left or right) is less than 10 (left is a negative value velocity and right is a positive value velocity) then the horizontal velocity is set to 10 (either left 10 or right 10).

Similarly, the horizontal velocity is checked to make sure it is less than 70 and is also constrained to those values.

A sound is played signifying the ball was hit.

```
on sprite of kind Projectile overlaps otherSprite of kind Player
  set sprite vy (velocity y) to -50
  if is left button pressed then
    change sprite vx (velocity x) by -25
  if is right button pressed then
    change sprite vx (velocity x) by 25
  if absolute of sprite vx (velocity x) ≤ 10 then
    if sprite vx (velocity x) ≤ 0 then
      set sprite vx (velocity x) to -10
    else if sprite vx (velocity x) ≥ 0 then
      set sprite vx (velocity x) to 10
  if absolute of sprite vx (velocity x) ≥ 70 then
    if sprite vx (velocity x) ≤ 0 then
      set sprite vx (velocity x) to -70
    else if sprite vx (velocity x) ≥ 0 then
      set sprite vx (velocity x) to 70
  play sound ba ding
```

---

### Challenge

*There is a block in the Math section that can be used to simplify the code inside the if statements that constrains the horizontal velocity. Can you find it and change the code to use it?*

---

## Smash - Blocks – Microsoft Arcade

### Step 5 – Dropping the ball

Because the ball bounces off the edges of the screen, we need a way to detect it is about to hit the bottom of the screen. When this happens, the player has “dropped” the ball and loses a life. The method that we use is to detect when the ball (which is a Projectile) hits a brown tile. The code to do this is shown on the right. Try it out to make sure it works and detects the ball is “dropped”

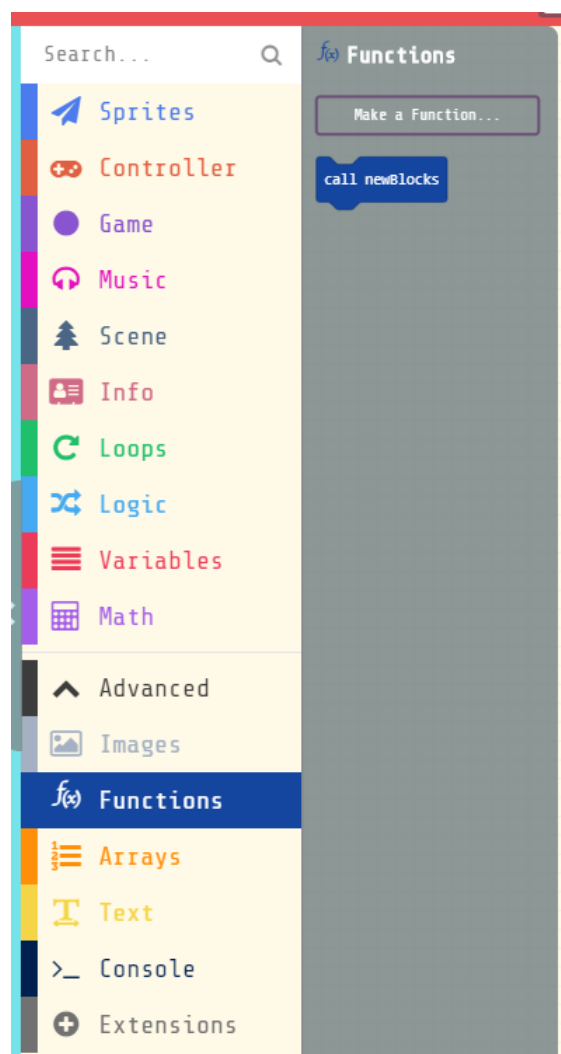


### Step 6 – Creating the blocks

The code to populate the screen with blocks is going to be used in two places in the game. We are therefore going to place this code in a new function that we will call “newBlocks”. A function can be thought of as a small program that does one thing (such as creating new blocks) and which is contained within a larger program. The larger program can then call that function to do its thing whenever it needs to.

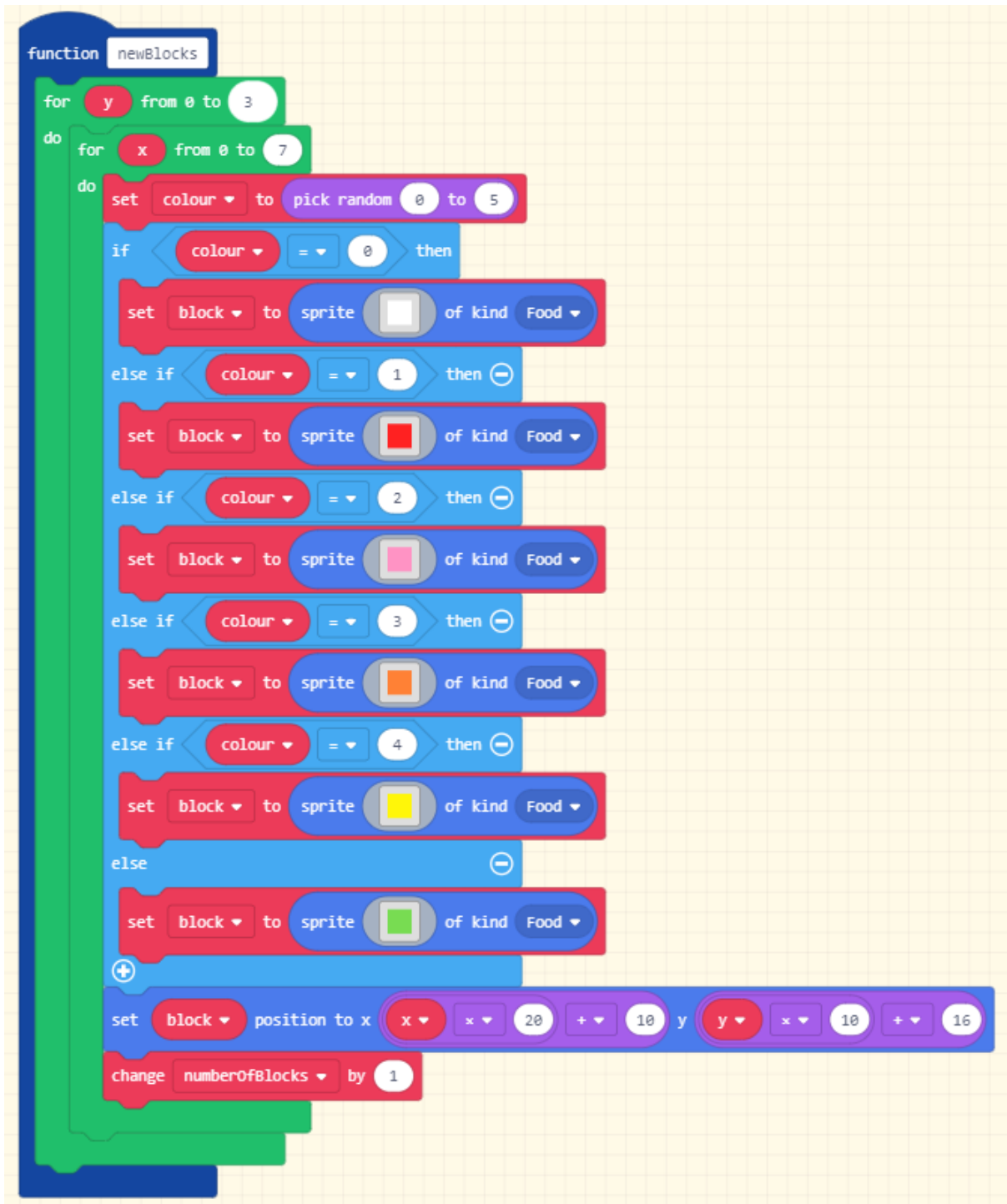
To create a function, click on the Functions section under the Advanced tab and then click the button “Make a Function...” (as show on the right).

You should be presented with the dialogue box below. Enter the functions name as “newBlocks” and then press Done.



## Smash - Blocks – Microsoft Arcade

You should then populate the function with the following code which creates an 8 x 4 grid of randomly coloured blocks across the top of the screen. In order to complete this code, you will need to create a new variable called numberOfBlocks.



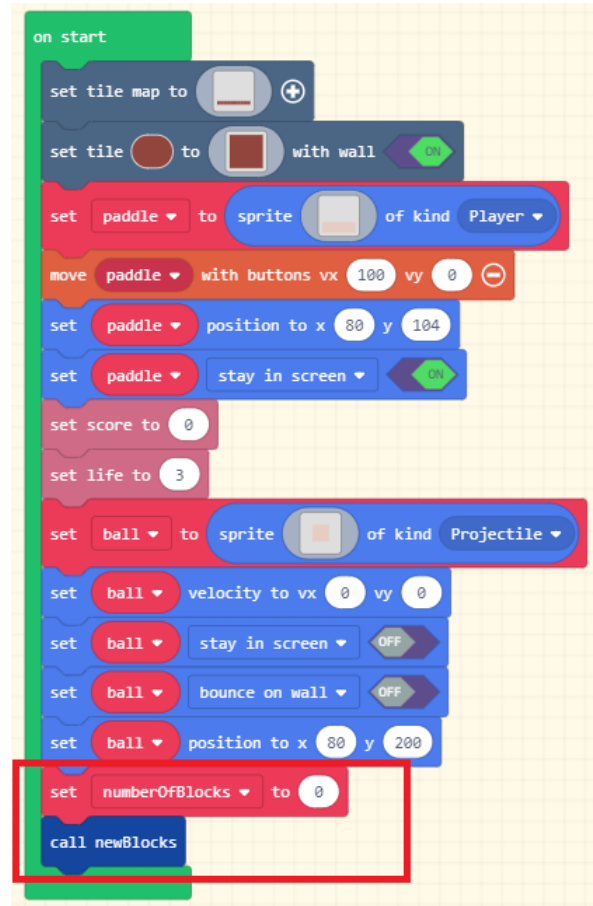
```
function newBlocks
  for y from 0 to 3
  do
    for x from 0 to 7
    do
      set colour to pick random 0 to 5
      if colour = 0 then
        set block to sprite of kind Food
      else if colour = 1 then
        set block to sprite of kind Food
      else if colour = 2 then
        set block to sprite of kind Food
      else if colour = 3 then
        set block to sprite of kind Food
      else if colour = 4 then
        set block to sprite of kind Food
      else
        set block to sprite of kind Food
      set block position to x * 20 + 10 y * 10 + 16
      change numberOfBlocks by 1
```

The image shows a Scratch code editor with a yellow grid background. A function block named 'newBlocks' is defined. It contains two nested 'for' loops. The outer loop iterates over 'y' from 0 to 3, and the inner loop iterates over 'x' from 0 to 7. Inside the inner loop, a 'set colour to pick random 0 to 5' block is used. This is followed by a series of 'if' and 'else if' blocks that check the value of 'colour' (0, 1, 2, 3, 4) and set the 'block' to a specific 'Food' sprite with a corresponding color (white, red, pink, orange, yellow). An 'else' block sets the 'block' to a green 'Food' sprite. After the 'if' blocks, a 'set block position to x \* 20 + 10 y \* 10 + 16' block is used to place the block at the correct coordinates. Finally, a 'change numberOfBlocks by 1' block is used to increment a variable named 'numberOfBlocks'.



## Smash - Blocks – Microsoft Arcade

The final step is to add the two lines of code to your existing “on start” loop.

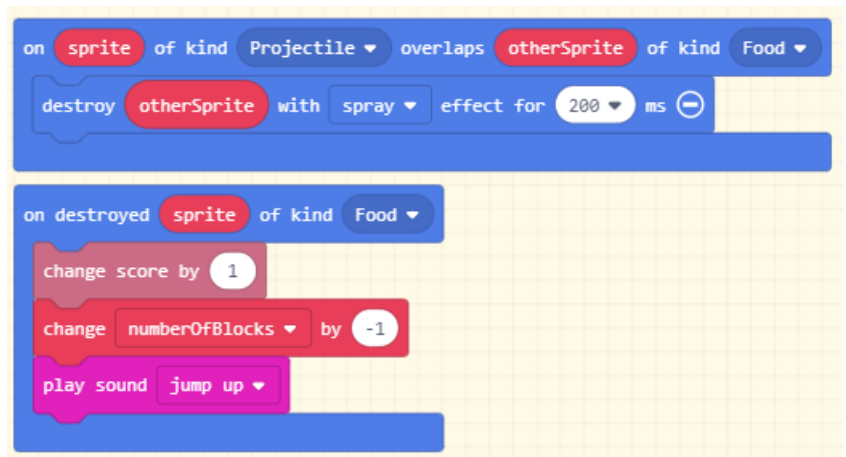


### Experiment

If you play the game now, what happens when the ball hits the blocks? Why do you think this is the case?

## Step 7 – Smashing the blocks

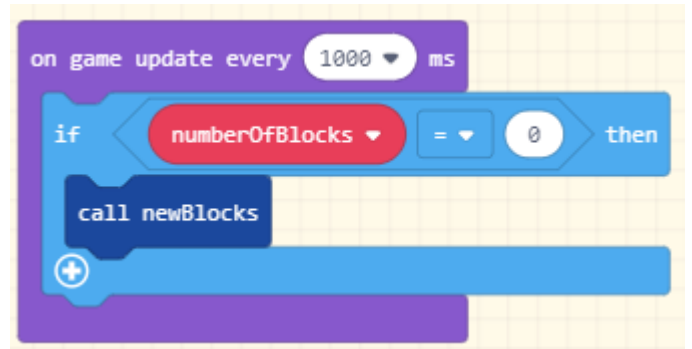
We now need to add the code that lets the ball smash the blocks. This requires two events to be added to the game. The first event is for the ball (which is a projectile) hitting a block (which is food) which destroys the block. The second event is what to do when a block (which is food) is destroyed; here we change the score, the count of the number of blocks on the screen and play a sound. Give the game a try.



## Smash - Blocks – Microsoft Arcade

### Step 8 – Repopulating the blocks

If you've played your game and managed to clear all the blocks, you will notice that it does not repopulate them. Because we track the number of blocks that are on the screen at any one time and because we have a function that will create new blocks for us, repopulating the blocks is very easy. We simply check every second if we've destroyed all the blocks and if we have then we create some new ones.



### Extending the game

There are many ways that this game can be extended. Just a few ideas are given below.

- Put in a custom background rather than just the plain black one.
- Animate the paddle and the ball like you have in previous games.
- Make the ball speed up as the players score increases.
- Award an extra life every time the player clears all of the blocks.

### Solution to challenge in step 4

The code below is one possible way to solve the challenge presented at the end of step 4.

