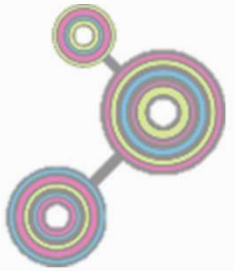
Creative Computing

Lives & Game Over!



THE PLAN

1) Give your player some lives.
2) Start a loop…
3) Check if the player has any lives left.
4) If no – game over!
5) If yes, run the game…
6) Check where the obstacle currently is.
7) Check where the player currently is.
8) Are they in the same place?
9) If yes, lose a life!
10) Return to #3

STEP 1: give your player some lives

Create a *variable* called lives. This goes above your loop, with the other variables for x, y and b!

```
b = 9
lives = 9

while True:
```

***Note**: I chose to give the player 9 lives, because **the obstacle only moves every 3 turns**, so the player loses lives more quickly!*

STEP 2: start a loop…

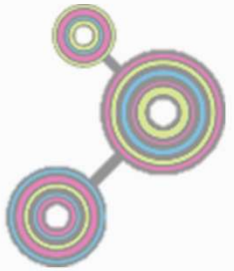
The rest of the code needs to go inside your `while True` loop.

You need to modify your existing code!

Turn over for instructions…

Creative Computing

Lives & Game Over!



STEP 3/4/5: check if player is still alive, if not then game over!
All of your current code happens forever.
You need to change it so that it only happens if the player has some lives left.

```
while True:
    if lives > 0:
        ...
    else:
        ...
```

indent your original code, to control the buttons

what do you want to happen if the player has no lives left?
try: `display.show(Image.NO)`

STEP 6/7: check locations
This code needs to run if your player has lives left! Add it at the bottom of this part of the code, underneath your button presses:

```
obstacle = get_road_pos()
player = x, y
```

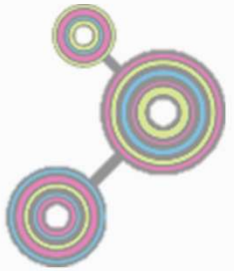
STEP 8/9: lose a life!
This code goes underneath the previous step's code:

```
if player == obstacle:
    lives -= 1
```

What now? Test it, debug it, expand it! Can you make a cool game over sequence?

Creative Computing

Extra Buttons & Pause



THE PLAN

1) Create a way of telling if the game is paused
2) Start a loop…
3) Check if the game is paused
4) If yes, wait for a button press to un-pause
5) If no, run the game…
6) Check for a button press
7) If the button is pressed, pause the game
8) Return to #3

STEP 1: is the game paused?

Create a *variable* called pause. This goes above your loop, with the other variables for x, y and b!

```
b = 9
pause = False

while True:
```

***Note**: I chose to set it to False at the start, so that the game starts straight away.*

STEP 2: start a loop…

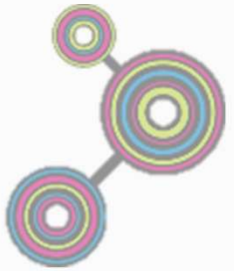
The rest of the code needs to go inside your `while True` loop.

You need to modify your existing code!

Turn over for instructions…

Creative Computing

Extra Buttons & Pause



STEP 3/5: check if the game is paused
All of your current code happens forever.
You need to change it so that it only happens if the game is not paused.

```
while True:
    if pause is False:
        ...
    else:
        pause = True
```

indent your original code, to control the buttons

STEP 4: wait for button to un-pause
After setting `pause = True`, you need to turn the game back on.
Underneath this code add:

```
button = button_press()
if button == 'E':
    pause = False
```

STEP 7: wait for button to pause
This code needs to run if your game is not paused! Add it at the bottom of this part of the code, underneath your other button presses:

```
if button == 'F':
    pause = True
```

What now?
Test it, debug it, expand it!
Can you change the display to show that the game is paused? Try:
`display.show(Image.DUCK)`

Can you think of any other uses for the buttons?

Creative Computing

Using the Joystick

THE PLAN

1) Calibrate your joystick.
2) Replace the buttons with the joystick!

Calibrate: Step A

Create a new program in Mu, by clicking:

Now visit: https://git.io/fjjVq

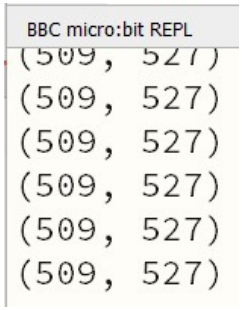
Copy and paste the code into your new program.

Calibrate: Step B

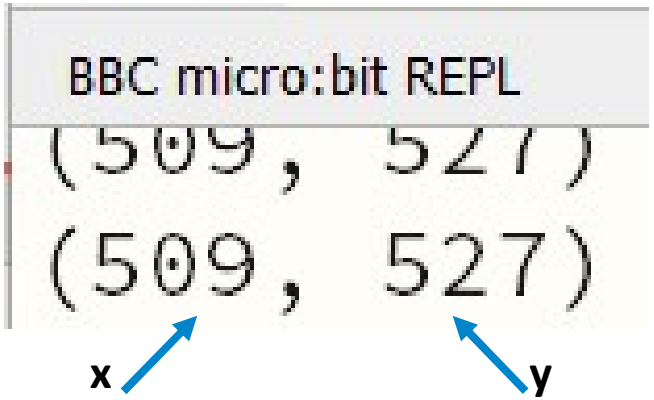
Flash the new code onto your micro:bit, then press the Repl button in Mu:

Now press the reset button **on the back of your micro:bit**.

You should see something like this on your screen:



Calibrate: Step C



Note down your two numbers!

Calibrate: Step D

```
if joystick[0] > 519:
```

Replace the first number with **your** x + 10

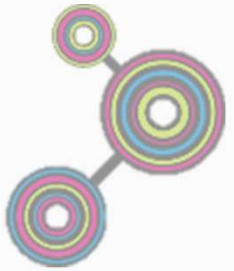
Repeat for all four options.

Test the joystick! Do the arrows match your actions?

1

Creative Computing

Using the Joystick



Replace the buttons: Step A
Go back to your original code.
Look through your code to find this line:

```
button = button_press()
```

This line checks for button presses, but you now want to check for joystick pushes as well!
Add this line underneath it:

```
joystick = joystick_push()
```

Replace the buttons: Step B
You have four new lines of code which interpret the joystick movement.

Match each one to the direction your player moves, and replace the button code:

```
if button == 'C' and y < 4:
```

becomes:

```
if joystick[1] < 516 and y < 4:
```

What now?
Test it, debug it, expand it!

You now have 6 free buttons!
What do you want them to do?

How could you use both the joystick and the buttons to make a two-player game?

Hint: you are currently controlling x and y - give a second player coordinates of x2 and y2!