MultiWingSpan

BBC micro:bit
Charlie's Python Game

Introduction

This project is a conversion of Charlie's Game into Python. The block editor design is a little flawed and not easy to make better. This version is a little more flexible.

Programming

There's a fair bit of code here. Two classes are designed to allow us to create sprites and help with the moving dot work. This version also keeps score.

```
from microbit import *
import random

class Asteroid():
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def update(self):
        self.y = self.y + 1
        if self.y==5:
            self.y=0
            self.x=random.randint(0,4)
class Ship():
    def __init__(self, x, y):
        self.x = x
        self.y = y

def DrawGame(t):
    img = Image('00000:'*5)
    img.set_pixel(ship.x, ship.y, (t % 2)*9)
    img.set_pixel(asteroid1.x, asteroid1.y, 9)
    img.set_pixel(asteroid2.x, asteroid2.y, 9)
    return img

while True:
        display.show(Image.ARROW_W)
        if button_a.was_pressed():
            display.clear()
            score = 0
            asteroid1 = Asteroid(random.randint(0,4),0)
            asteroid2 = Asteroid(random.randint(0,4),2)
            ship = Ship(2,4)
            tick = 0
            paws = 75
            sleep(1000)
            playing = True
            while playing:
                if button_a.was_pressed() and ship.x!=0:
                    ship.x = ship.x - 1
                elif button_b.was_pressed() and ship.x!=4:
                    ship.x = ship.x + 1
                i = DrawGame(tick)
                # update ticks
                tick += 1
                if tick == 3:
                    tick = 0
                    asteroid1.update()
                    asteroid2.update()
                    score += 1
                    i = DrawGame(tick)
                if asteroid1.y==4:
                    if ship.x==asteroid1.x:
                        playing = False
                if asteroid2.y==4:
                    if ship.x==asteroid2.x:
                        playing = False
                display.show(i)
                sleep(paws)
            display.scroll(str(score))
            sleep(2000)
        sleep(50)
```

Challenges

1. Sam and Olly used Play-Doh buttons on the Concentration Game with a small change to the code. They will probably want to do the same with this game and you might too. Having to press two buttons for each movement makes things a little more interesting.
2. Charlie worked out a nice improvement to the game. Once you get over the start, it is quite easy if the speed doesn't change. The sleep(paws) statement is the key one to work with. Charlie had it so the length of this pause decreased as the score increased. The best way to do this is to find an arithmetic relationship between the score and the length of the delay. The **paws** variable is the one that says how long a delay is. If, in the while loop, you recalculated the length of this based on the score, you could make the game get quicker. Working out this formula is a bit tricky but a really cool thing to be able to do. Start by making a table of scores and delay lengths. The delay lengths should get shorter as the score gets higher. Make the changes at even intervals, every 50 or so.

BBC Microbit

Collapse All | Expand All

+ Block Editor - The Basics
+ Block Editor - Components
+ Kodu - micro:bit Worlds
+ JavaScript Blocks
+ JavaScript Blocks - Exercises
+ Blocks - Bit:Bot
+ Blocks - Bit:Commander
+ MicroPython - Starting Off
- MicroPython - Examples
  - Dice Rolling
  - Shut The Matrix
  - Encoding Morse Code
  - Encoding Ciphers
  - Drawing A Maze
  - Scrolling Race Track
  - Vertical Scroller
  - Concentration Game
  - Text Entry - Accelerometer
  - Charlie's Python Game
  - Lights Out Game
  - Sam's French Number Game
  - Bryn's Concentrated Clocks
  - Lattice Paths
  - Knight Moves
+ MicroPython - Components
+ MicroPython - Breakout Boards
+ MicroPython - Exercises
+ MicroPython - Pi Accessories
+ MicroPython - Bit:Bot
+ MicroPython - Bit:Commander
+ MicroPython - Projects
+ MicroPython - Visual Basic
+ Other - Odds & Ends

Then write another column showing what score // 50 is equal to. In Python, // is the operator for floor division. This is division with the remainder ignored. You can replace the 50 with a different number if you want the game to get quicker after a different amount.

3. The game needs a nicer start and end, something a bit more interesting and more helpful to the user.
4. Attaching a buzzer or some headphones gives you the opportunity to add sound effects to the game. I'd go for a noise every time the asteroids move down a row.
5. Add more asteroids.
6. Reuse the sprite classes for a different, better game idea.