BBC micro:bit
Scrolling Race Track

Introduction

This project is a simple game idea for a vertical scrolling race track. The track is drawn randomly. The player can move left and right using the A and B buttons. The player can only move forward if they are not going to bump into the track.

Programming

Comments are included in the code to explain what is going on.

```
from microbit import *
import random

# lines of track
track_bits = [
    "55550:",
    "55500:",
    "55000:",
    "50005:",
    "00055:",
    "00555:",
    "05555:"
]

# Game Variables
# **************

# current line of track
current = 3
# starting track
track = "50005:50005:50005:50005:50005:"
# used to allow track updates every third game loop
tick = 0
## co-ordinates of player character
x = 2
y = 4

# function creates an image from the track string
# the tick is used to vary the brightness of the
# character to make it blink
def draw(t):
    img = Image(track)
    img.set_pixel(x,y,tick * 4 + 1)
    return img

# Game Loop

while True:
    # make image of track
    tt = draw(tick)
    # check for button presses and
    # move character if no collision
    if button_a.was_pressed() and x!=0:
        if tt.get_pixel(x-1,y)==0:
            x = x - 1
    elif button_b.was_pressed() and x!=4:
        if tt.get_pixel(x+1,y)==0:
            x = x + 1
    # update ticks
    tick += 1
    # show track
    display.show(tt)
    # every third tick
    if tick == 3:
        # reset ticks
        tick =0
        # update track if possible
        if tt.get_pixel(x,y-1)==0:
            # delete bottom row of track
            track = track[:-6]
            # choose a new piece of track
            current = random.randint(current - 1, current + 1)
            # make sure the number is in range
            current = max(0, min(current, 6))
            # update track
            track = track_bits[current] + track
            # redraw screen
            tt = draw(tick)
            display.show(tt)
    # rest a while
    sleep(75)
```

Challenges

1. Improvement is needed. An easy first step is to use the **running_time()** method to decide when to end the game. Add a score variable and add 1 to it every time the track is successfully updated

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

(ie player not in the way). When the game is over, present the score to the player.
2. Add a buzzer and make it buzz every time the player's position prevents the track from being updated.
3. Allow the track to move off the matrix. When the player is at the edges of the matrix, you can move the whole image one place to the left or right. Work out how you should fill the edge that is created when you do this and how to choose your new piece of track and you will have a track that will appear to carry on off the screen.