11/17/2018 multiwingspan

MultiWingSpan

Home Programming Web Design Computer Science Twisting Puzzles Arduino BBC micro:bit

BBC micro:bit Bit:Commander Simon Game

Introduction

This is the first non-controller idea that I had for the Bit:Commander. It has the lights, the buzzer and the 4 different coloured buttons that you need.



The first time I tried this, I had the LEDs brighter than they are. When I gave it to Chris to try out in a lesson, I saw how brightly it lit up his whole head and how much he rubbed his eyes when he'd finished playing. I've probably left it a little too bright still. Tweak the numbers for the colours down a little until you are comfortable using this for a game.

```
from microbit import *
import music
import random
import neopixel
# red, blue, green, yellow
button_pins = [pin12, pin15, pin14, pin16]
# Initialise neopixels
npix = neopixel.NeoPixel(pin13, 6)
red = (64,0,0)
green = (0,64,0)
blue = (0,0,64)
yellow = (64, 24, 0)
off = (0,0,0)
ledcols = [red,blue,green,yellow]
# matrix helper
d = [Image.ARROW_N,Image.ARROW_E,Image.ARROW_S,Image.ARROW_W]
notes = [659, 880, 330, 554]
def LightAll(col):
    for pix in range(0, len(npix)):
        npix[pix] = col
    npix.show()
def GetRandomSequence():
    return [int(4*random.random()) for i in range(0,100)]
def PlaySequence(t, s)
    for i in range(0,t):
         LightAll(ledcols[s[i]])
         display.show(d[s[i]])
         music.pitch(notes[s[i]],500)
        display.clear()
LightAll(off)
```

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
- + MicroPython Components
- + MicroPython Breakout Boards
- + MicroPython Exercises
- + MicroPython Pi Accessories
- + MicroPython Bit:Bot
- MicroPython Bit:Commander
- * Bit:Commander
- * The Joystick
- * The Neopixels
- * The Potentiometer
- ★ The Pushbuttons
- * The Buzzer
- * Evasion Game
- ★ Light's Out Game
- ★ Simon Game
- ★ Bit:Bot/Robot Controller
- * Text Entry
- ★ Unicorn Commander
- + MicroPython Projects
- + MicroPython Visual Basic
- + Other Odds & Ends





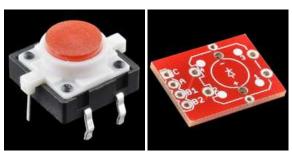


11/17/2018 multiwingspan

```
sleep(200)
def Win():
     for i in range(0,4):
          for j in range(0,4):
              LightAll(ledcols[j])
              music.pitch(notes[j],50)
LightAll(off)
     sleep(1000)
def Loss():
    music.pitch(131,500)
     sleep(1500)
def PlayGame():
     turn = 0
     sequence = GetRandomSequence()
    userSequence = [0]*100
seqlen = 0
    playing = True
played = False
     while playing==True:
if turn==0:
              # Just started
              Win()
              turn = turn + 1
         if seqlen==0 and played==False:
# Sequence needs playing
              PlaySequence(turn, sequence)
              played = True
         elif seqlen==turn:
              # User has entered the sequence
              Win()
              played = False
              turn = turn + 1
              seqlen = 0
         else:
             # User still entering pattern
             for i in range(0,4):
                 if button_pins[i].read_digital()==1:
    userSequence[seqlen] = i
                      if userSequence[seqlen]!=sequence[seqlen]:
                           Loss()
                           playing = False
                      else:
                           seqlen = seqlen + 1
LightAll(ledcols[i])
                           display.show(d[i])
                           music.pitch(notes[i],500)
                           display.clear()
LightAll(off)
def main():
     sleep(50)
     while True:
         display.show(Image.ARROW_W)
         if button_a.was_pressed():
              display.clear()
              PlayGame()
              sleep(1000)
         sleep(50)
main()
```

I've always enjoyed making this game with microcontrollers. Apart from circuits that swear or make rude noises, it's the one circuit guaranteed to please the muggles.

I normally use these lovely tactile buttons, that you can solder to these handy breakout boards.



That makes the cabling easier if you were trying to replicate this implementation with the separate components. On the Bit:Commander, you have a completely portable version that you can pass around and take to the muggles to play with.

Pages designed and coded by MHA since 2003 | Valid $\underline{\text{HTML 4.01}}(\text{Strict}) \mid \underline{\text{CSS}}$