Tilt Controlled Marble Maze

Created by Dano Wall

https://learn.adafruit.com/tilt-controlled-marble-maze

Last updated on 2023-08-29 04:11:04 PM EDT
# Table of Contents

## Overview
- Materials

## CircuitPython Code
- Getting familiar with CircuitPython
- Preparing your Board
- Uploading
- Changing the Code

## Build the Circuit

## Build the Maze
- Grid System
- Sketch your Maze
- Maze Walls
- Frame
- Base
Overview

This project uses the accelerometer on the Circuit Playground Express board to control the X and Y axes of a maze. The user can guide a marble through the maze by simply tilting the Circuit Playground Express. This project is constructed using only cardboard, hot glue, and a couple of screws.

Circuit Playground Express
Circuit Playground Express is the next step towards a perfect introduction to electronics and programming. We've taken the original Circuit Playground Classic and...
https://www.adafruit.com/product/3333
Micro servo
Tiny little servo can rotate approximately 180 degrees (90 in each direction) and works just like the standard kinds you're used to but smaller. You can use any servo...
https://www.adafruit.com/product/169

Small Alligator Clip to Male Jumper Wire Bundle - 6 Pieces
When working with unusual non-header-friendly surfaces, these handy cables will be your best friends! No longer will you have long, cumbersome strands of alligator clips. These...
https://www.adafruit.com/product/3448

USB cable - USB A to Micro-B
This here is your standard A to micro-B USB cable, for USB 1.1 or 2.0. Perfect for connecting a PC to your Metro, Feather, Raspberry Pi or other dev-board or...
https://www.adafruit.com/product/592

Materials

In addition to the electronics, for this project you will also need:

- Marble or steel ball
- Scrap cardboard
- Hot glue gun
- Scissors or hobby knife

©Adafruit Industries
CircuitPython Code

Getting familiar with CircuitPython

CircuitPython is a programming language based on Python, one of the fastest growing programming languages in the world. It is specifically designed to simplify experimenting and learning to code on low-cost microcontroller boards.

CircuitPython is easiest to use within the Mu Editor. If you haven't previously used Mu, this guide will get you started.

Preparing your Board

To get your Circuit Playground Express set up to run this code, follow these steps:

1) Install the latest CircuitPython for CPX from CircuitPython.org. For this guide, please use version 4.0.0 or greater. Version 3.1.2 is not as efficient on memory and may give an error.

2) Get the latest library pack, unzip it, and drag the library files adafruit_motor folder and simpleio.mpy over into the /lib folder on CIRCUITPY. If there is no lib directory, create one to put the file into. (More info on installing libraries, read information here.)
Uploading

Make sure you've connected the Circuit Playground Express to your computer (mac/PC/Linux) via a known good USB A to micro-B cable. Your board should show up as a flash disk drive named CIRCUITPY (If you see a disk name CPLAYBOOT, try pressing the reset button again. If the only drive name you get is named CPLAYBOOT, CircuitPython may not be loaded on the board. You can load CircuitPython as per this guide).)

Once your board is connected, copy code.py from the window below. You can select Download in the upper left tp save the code onto your computer.

```python
# SPDX-FileCopyrightText: 2019 Anne Barela for Adafruit Industries
#
# SPDX-License-Identifier: MIT

# CircuitPython code for the Gyroscopic Marble Maze
# Adafruit Industries, 2019. MIT License
import time
import board
import pwmio
from adafruit_motor import servo
import simpleio
from adafruit_circuitplayground.express import cpx

# create a PWMOut object on Pin A2.
pwm1 = pwmio.PWMOut(board.A1, duty_cycle=2 ** 15, frequency=50)
pwm2 = pwmio.PWMOut(board.A2, duty_cycle=2 ** 15, frequency=50)

# Create a servo object, my_servo.
my_servo1 = servo.Servo(pwm1)
my_servo2 = servo.Servo(pwm2)

NUM_READINGS = 8
roll_readings = [90] * NUM_READINGS
pitch_readings = [90] * NUM_READINGS

def Average(lst):
```
return sum(lst) / len(lst)

while True:
    x, y, z = cpx.acceleration  # x = green
    print((x, y, z))
    roll = simpleio.map_range(x, -9.8, 9.8, 0, 180)
    roll_readings = roll_readings[1:]
    roll_readings.append(roll)
    roll = Average(roll_readings)
    print(roll)
    my_servo1.angle = roll
    pitch = simpleio.map_range(y, -9.8, 9.8, 0, 180)
    pitch_readings = pitch_readings[1:]
    pitch_readings.append(pitch)
    pitch = Average(pitch_readings)
    print(pitch)
    my_servo2.angle = pitch
    time.sleep(0.05)

Open the code up in the Mu editor. Press the Save button and your code should automatically be saved to the CIRCUITPY disk drive (which appears when the Circuit Playground Express is plugged into your computer) as code.py.

Alternatively, you can use your operating system file explorer/finder to copy code.py to the CIRCUITPY drive.

The board will restart itself and start running the code.

Changing the Code

The default settings used in this code have been optimized for the particular servos used, but can be tweaked to suit your particular preferences. For example, you can increase NUM_READINGS from 8 up to 12 if you feel the maze is too jerky, or decrease it down to 2 (but not below 2) if you feel the maze is too slow to respond.

You can also update time.sleep from 0.05 to something like 0.20 to make the movement of the servos smoother.

Once your code is uploaded to Circuit Playground Express you can connect some motors to your board and see what this code does! Read on...
Build the Circuit

This project uses two micro servo motors, powered and controlled using Circuit Playground Express.

The Vout pad on Circuit Playground Express (CPX) is connected to each servos' red wire, the A1 and A2 pads to the yellow wires, and the two GND pads to the servos' brown wire.

The alligator clip-to-male jumper wire () connectors are great for connecting servo motors to CPX.
Once your motors are connected, you can connect your board to a power source and test how the servos respond to the board being tilted along its x-axis and y-axis.

If a servo does not move, check that you have form connections on the correct pads. Do not connect to pad A0, that is a sound output pad.

If you have issues running servos (especially two), use a better power source as a battery might be drained, etc. Servo motors use a fair amount of current, draining batteries over time.
Build the Maze

Grid System

Find a square-shaped piece of cardboard. Create a square grid (8"x8" / 20x20cm or 10"x10" / 25x25cm are good starting places).
Sketch your Maze

Using a dark marker, sketch out where the walls of your maze will go. Make it as simple or complex as you want! You can look at other examples of mazes online for ideas.
Maze Walls

Cut strips of cardboard to fill in the walls of the maze.
Use hot glue to start adding walls one at a time.
Be careful using hot glue as the dispenser and fresh glue can cause burns. Older Makers should help very young Makers with these steps.

A beautiful cardboard maze, ready to be robotically navigated.
Frame

Build a square frame to hold the maze using some strips of cardboard. Connect a servo to one edge of the frame, connect the maze to the opposite edge of the frame with a short bolt. This servo will control the x-axis.
Base

Create a base to hold the maze and frame.
Install a second servo motor in one side. This servo will control the y-axis.
Use hot glue to connect the servo arms to the edge of the maze frame. Once the glue has cooled and hardened, press the arm onto the servo horn.
Once the maze has been suspended in its frame, you can connect the servo motors up to Circuit Playground Express and test out its responsiveness.

Enjoy playing with your robotic labyrinth!