Space Aliens - CircuitPython Game

Mr. Coxall

Jan 23, 2020

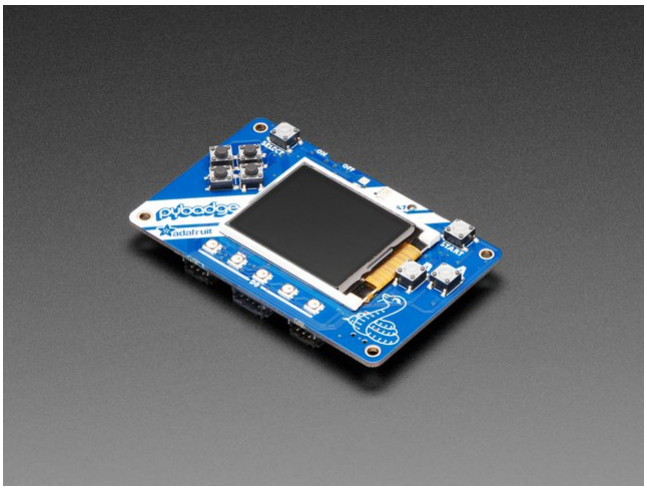
Contents

In this project we will be making an old school style video game for the Adafruit PyBadge. We will be using CircuitPython and the stage library to create a Thin Ice like game. The stage library makes it easy to make classic video games, with helper libraries for sound, sprites and collision detection. The game will also work on other variants of PyBadge hardware, like the PyGamer and the EdgeBadge. The full completed game code with all the assets can be found here.

The guide assumes that you have prior coding experience, hopefully in Python. It is designed to use just introductory concepts. No Object Oriented Programming (OOP) are used so that students in particular that have completed their first course in coding and know just variables, if statements, loops and functions will be able to follow along.

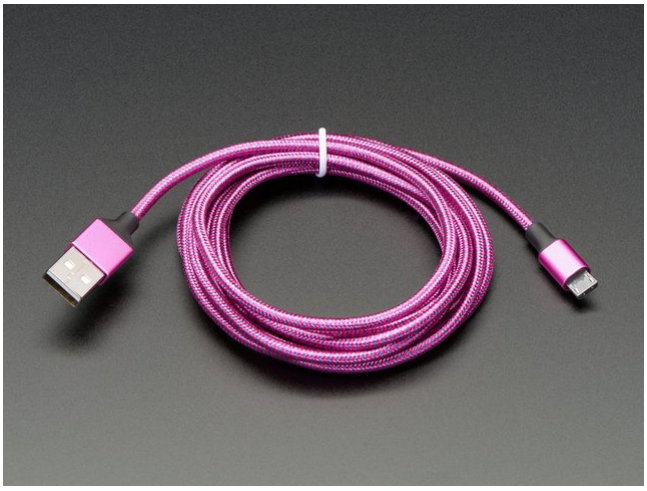
Parts

You will need the following items:



Adafruit PyBadge for MakeCode Arcade, CircuitPython or Arduino

PRODUCT ID: 4200

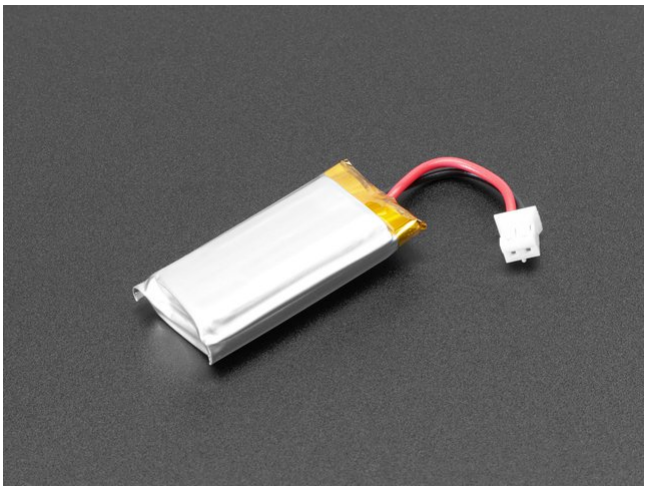


Pink and Purple Braided USB A to Micro B Cable - 2 meter long

PRODUCT ID: 4148

So you can move your CircuitPython code onto the PyBadge.

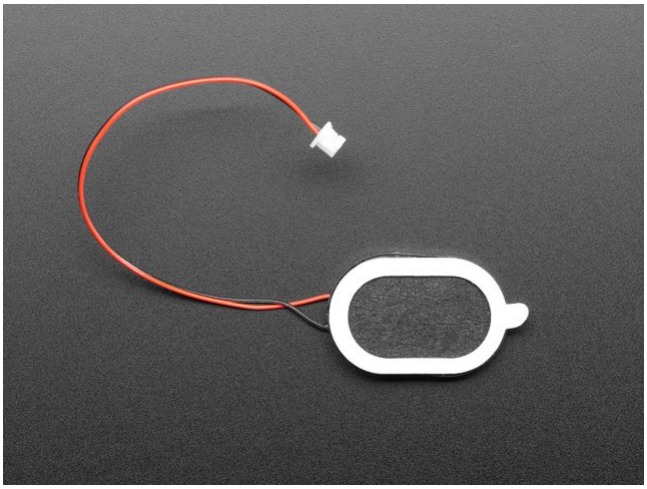
You might also want:



Lithium Ion Polymer Battery Ideal For Feathers - 3.7V 400mAh

PRODUCT ID: 3898

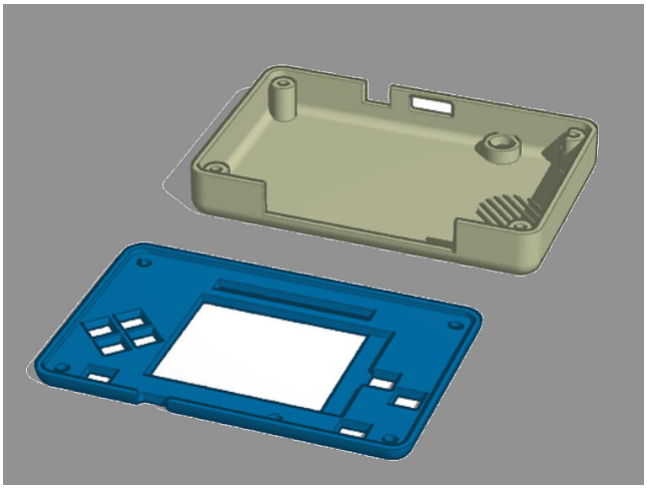
So that you can play the game without having it attached to a computer with a USB cable.



Mini Oval Speaker - 8 Ohm 1 Watt

PRODUCT ID: 3923

If you want lots of sound. Be warned, the built in speaker is actually pretty loud.



3D Printed Case

I did not create this case. I altered Adafruit's design. One of the screw posts was hitting the built in speaker and the

case was not closing properly. I also added a piece of plastic over the display ribbon cable, to keep it better protected. You will need 4 x 3M screws to hold the case together.

CHAPTER 1

Install CircuitPython

Fig. 1: Clearing the PyBadge and loading the CircuitPython UF2 file

Before doing anything else, you should delete everything already on your PyBadge and install the latest version of CircuitPython onto it. This ensures you have a clean build with all the latest updates and no leftover files floating around. Adafruit has an excellent quick start guide here to step you through the process of getting the latest build of CircuitPython onto your PyBadge. Adafruit also has a more detailed comprehensive version of all the steps with complete explanations here you can use, if this is your first time loading CircuitPython onto your PyBadge.

Just a reminder, if you are having any problems loading CircuitPython onto your PyBadge, ensure that you are using a USB cable that not only provides power, but also provides a data link. Many USB cables you buy are only for charging, not transfering data as well. Once the CircuitPython is all loaded, come on back to continue the tutorial.

CHAPTER 2

Your IDE

One of the great things about CircuitPython hardware is that it just automatically shows up as a USB drive when you attach it to your computer. This means that you can access and save your code using any text editor. This is particularly helpful in schools, where computers are likely to be locked down so students can not load anything. Also students might be using Chromebooks, where only "authorized" Chrome extensions can be loaded.

If you are working on a Chromebook, the easiest way to start coding is to just use the built in Text app. As soon as you open or save a file with a `*.py` extension, it will know it is Python code and automatically start syntax highlighting.

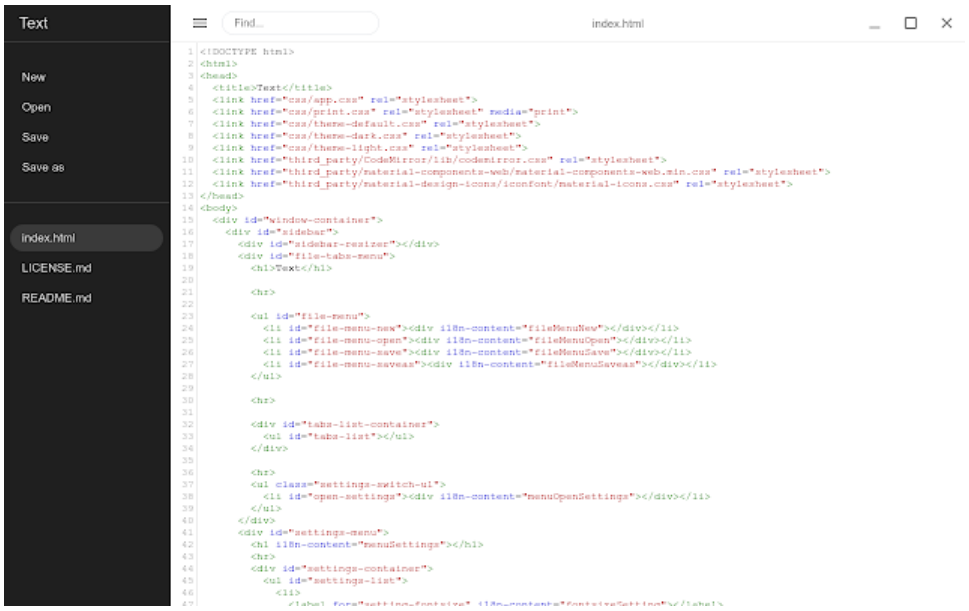


Fig. 1: Chromebook Text app

If you are using a non-Chromebook computer, your best beat for an IDE is Mu. You can get it for Windows, Mac, Raspberry Pi and Linux. It works seamlessly with CircuitPython and the serial console will give you much needed debugging information. You can download Mu here.

Fig. 2: Mu IDE

Since with CircuitPython devices you are just writing Python files to a USB drive, you are more than welcome to use any IDE that you are familiar using.

2.1 Hello, World!

Yes, you know that first program you should always run when starting a new coding adventure, just to ensure everything is running correctly! Once you have access to your IDE and you have CircuitPython loaded, you should make sure everything is working before you move on. To do this we will do the traditional "Hello, World!" program. By default CircuitPython looks for a file called `code.py` in the root directory of the PyBadge to start up. You will place the following code in the `code.py` file:

```
print("Hello, World!")
```

As soon as you save the file onto the PyBadge, the screen should flash and you should see something like:

Although this code does work just as is, it is always nice to ensure we are following proper coding conventions, including style and comments. Here is a better version of Hello, World! You will notice that I have a call to a `main()` function. This is common in Python code but not normally seen in CircuitPython. I am including it because by breaking the code into different functions to match different scenes, eventually will be really helpful.

```
#!/usr/bin/env python3

# Created by : Mr. Coxall
# Created on : January 2020
# This program prints out Hello, World! onto the PyBadge


```

(continues on next page)

Fig. 3: Hello, World! program on PyBadge

(continued from previous page)

```
8  def main():
9      # this function prints out Hello, World! onto the PyBadge
10     print("Hello, World!")
11
12
13 if __name__ == "__main__":
14     main()
```

Congratulations, we are ready to start.

CHAPTER 3

Image Banks

Before we can start coding a video game, we need to have the artwork and other assets. The stage library from CircuitPython we will be using is designed to import an "image bank". These image banks are 16 sprites staked on top of each other, each with a resolution of 16x16 pixels. This means the resulting image bank is 16x256 pixels in size. Also the image bank **must** be saved as a 16-color BMP file, with a pallet of 16 colors. To get a sprite image to show up on the screen, we will load an image bank into memory, select the image from the bank we want to use and then tell CircuitPython where we would like it placed on the screen.

Fig. 1: An Image Bank for Ice Ice Baby

For sound, the stage library can play back `*.wav` files in PCM 16-bit Mono Wave files at 22KHz sample rate. Adafruit has a great learning guide on how to save your sound files to the correct format here.

If you do not want to get into creating your own assets, other people have already made assets available to use. All the assets for this guide can be found in the GitHub repo here:

- space aliens image bank
- *mt game studios image bank <https://github.com/joseph-palermo/ICS3U-2019-Group3/blob/master/iib_game_files/mt_game_studio.bmp>*
- boot up sound
- key collecting sound
- splash screen sound

Please download the assets and place them on the PyBadge, in the root directory. Your previoud "Hello, World!" program should restart and run again each time you load a new file onto the PyBadge, hopefully with no errors once more.

Assets from other people can be found here.

CHAPTER 4

Game

The game starts out with a little penguin, Vilheleme who has to travel his way to the star. The only way he can get through is if he waddles across the ice. But beware, the ice behind him melts everytime he steps off of it! You must complete two levels while making every tile water to finish with the highest score

4.1 Background

In the begininning of every function what uses a background, you must reference the image bank you wish to use. In this instance, its image_bank_1

```
image_bank_1 = stage.Bank.from_bmp16("iib_sprites.bmp")
```

To code the background of the splash screen, you must type this code after the reference code:

```
background = stage.Grid(image_bank_1, constants.SCREEN_GRID_X,
                        constants.SCREEN_GRID_Y)
for x_location in range(constants.SCREEN_GRID_X):
    for y_location in range(constants.SCREEN_GRID_Y):
        background.tile(x_location, y_location, 15)
```

In your menu sceen, after referencing which image bank you will use, you must type this:

```
background = stage.Grid(image_bank_2, constants.SCREEN_GRID_X,
                        constants.SCREEN_GRID_Y)
```

For level 1 of your game, you must type this code after referencing the image bank you choose to use:

```
background = stage.Grid(image_bank_1, constants.SCREEN_X,
                        constants.SCREEN_Y)
for x_location in range(constants.SCREEN_GRID_X):
    for y_location in range(constants.SCREEN_GRID_X):
        background.tile(x_location, y_location, 3)
```

4.2 Vilheleme

Vilheleme is the little penguin controlled by the D pad. When a player touches a key on the D pad, Vilheleme moves in the corrosponding direction. This is the code to make vilheleme show up:

```
def lvl_1():
    vilheleme_list = []

    image_bank_1 = stage.Bank.from_bmp16("iib_sprites.bmp")

    # create vilheleme
    vilheleme = stage.Sprite(image_bank_1, 2, 16, 64)
    vilheleme_list.append(vilheleme)  # insert at the top of sprite list
```

CHAPTER 5

Menu System

The menu system works like this, splash screen, menu screen. Then it transfers to the game. By combining the splasch screen and menu screen code, you can make the menu screen.

5.1 Start Scene

The menu screen offers the option for the player to press A to begin playing the game:

```
def menu_scene():

image_bank_1 = stage.Bank.from_bmp16("iib_sprites.bmp")
background = stage.Grid(image_bank_1, constants.SCREEN_GRID_X,
                        constants.SCREEN_GRID_Y)
for x_location in range(constants.SCREEN_GRID_X):
    for y_location in range(constants.SCREEN_GRID_Y):
        background.tile(x_location, y_location, 15)

sprites = []
text = []
text3 = []
text3_list = []
text3 = stage.Text(width=29, height=12, font=None,
                   palette=constants.ICE_ICE_BABY_PALETTE, buffer=None)
text3.move(30, 6)
text3.text("Ice Ice Baby")
text.append(text3)
text4 = []
text4_list = []
text4 = stage.Text(width=17, height=5, font=None,
                   palette=constants.ICE_ICE_BABY_PALETTE, buffer=None)
text4.move(16, 116)
text4.text("Press A To Begin")
text.append(text4)
```

(continues on next page)

(continued from previous page)

```
game = stage.Stage(ugame.display, constants.FPS)
game.layers = text + sprites + [background]
game.render_block()

# get sound ready
press_start_audio = open("press_start_audio.wav", 'rb')
sound = ugame.audio
sound.stop()
sound.mute(False)

lvl1 = None
final_score = None

while True:
    keys = ugame.buttons.get_pressed()
    if keys & ugame.K_X != 0:
        final_score = 0
        sound.play(press_start_audio)
        time.sleep(1)
        score = lvl_1()
        lvl1 = 1
    game.tick()
    if lvl1 == 1:
        final_score = lvl_2(score)
        lvl1 = 2
    game.tick()
    if lvl1 == 2:
        game_over(final_score)
    game.tick()
```

Then the game goes to level 1.

5.2 Splash Scene

The splash screen displays the MT Studios logo and the creators of Ice Ice Baby, Liam and Joseph. It takes 2 seconds for the splash screen to transition to the menu screen.

```
def splash_scene():

image_bank_2 = stage.Bank.from_bmp16("mt_game_studio.bmp")
background = stage.Grid(image_bank_2, constants.SCREEN_GRID_X,
                        constants.SCREEN_GRID_Y)
background.tile(2, 2, 0)  # blank white
background.tile(3, 2, 1)
background.tile(4, 2, 2)
background.tile(5, 2, 3)
background.tile(6, 2, 4)
background.tile(7, 2, 0)  # blank white

background.tile(2, 3, 0)  # blank white
background.tile(3, 3, 5)
background.tile(4, 3, 6)
background.tile(5, 3, 7)
```

(continues on next page)

(continued from previous page)

```
background.tile(6, 3, 8)
background.tile(7, 3, 0)  # blank white

background.tile(2, 4, 0)  # blank white
background.tile(3, 4, 9)
background.tile(4, 4, 10)
background.tile(5, 4, 11)
background.tile(6, 4, 12)
background.tile(7, 4, 0)  # blank white

background.tile(2, 5, 0)  # blank white
background.tile(3, 5, 0)
background.tile(4, 5, 13)
background.tile(5, 5, 14)
background.tile(6, 5, 0)
background.tile(7, 5, 0)  # blank white

# get sound ready
boot_up = open("boot_up.wav", 'rb')
sound = ugame.audio
sound.stop()
sound.mute(False)

sprites = []
text = []
text2_list = []
text1 = stage.Text(width=29, height=12, font=None,
                   palette=constants.NEW_PALETTE, buffer=None)
text1.move(20, 10)
text1.text("MT Game Studios")
text.append(text1)
text2 = stage.Text(width=15, height=5, font=None,
                   palette=constants.NEW_PALETTE, buffer=None)
text2.move(30, 100)
text2.text("Made by        Liam & Joseph")
text.append(text2)

game = stage.Stage(ugame.display, constants.FPS)
game.layers = text + sprites + [background]
game.render_block()

image_bank_1 = stage.Bank.from_bmp16("iib_sprites.bmp")
background = stage.Grid(image_bank_1, constants.SCREEN_GRID_X,
                        constants.SCREEN_GRID_Y)
for x_location in range(constants.SCREEN_GRID_X):
    for y_location in range(constants.SCREEN_GRID_Y):
        background.tile(x_location, y_location, 15)
sound.play(boot_up)
time.sleep(2)
menu_scene()
game.tick()
```

Then it goes on to the menu scene.

5.3 Game Over Scene

The game moves to the game over scene when the player either wins the game or makes contact with the water. The game over scene displays the final score at the time of death. Press the start button to restart from the menu scene.

```
    def game_over(final_score):

    image_bank_1 = stage.Bank.from_bmp16("iib_sprites.bmp")
    background = stage.Grid(image_bank_1, constants.SCREEN_GRID_X,
                            constants.SCREEN_GRID_Y)
    for x_location in range(constants.SCREEN_GRID_X):
    for y_location in range(constants.SCREEN_GRID_X):
            background.tile(x_location, y_location, 3)

    sprites = []
    text = []
    text_game_over = []
    text_game_over_list = []
    text_game_over = stage.Text(width=29, height=12, font=None, palette=constants.ICE_
↪ICE_BABY_PALETTE, buffer=None)
    text_game_over.move(45, 35)
    text_game_over.text("Game Over")
    text.append(text_game_over)

    # V If game lags, change this V
    game = stage.Stage(ugame.display, constants.FPS)

    # add text at top of screen for score
    score_text = stage.Text(width=29, height=14, font=None, palette=constants.SCORE_
↪PALETTE, buffer=None)
    score_text.clear()
    score_text.cursor(0, 0)
    score_text.move(16, 100)
    score_text.text("Final Score: {0}".format(final_score))

    restart_text = stage.Text(width=11, height=14, font=None, palette=constants.SCORE_
↪PALETTE, buffer=None)
    restart_text.clear()
    restart_text.cursor(0, 0)
    restart_text.move(35, 70)
    restart_text.text("Press Startto restart".format(final_score))

    game.layers = [score_text] + [text_game_over] + [restart_text]+ [background]
    game.render_block()

    while True:
        keys = ugame.buttons.get_pressed()
        if keys & ugame.K_START != 0:
            return menu_scene()

if __name__ == "__main__":
    splash_scene()
```

The game then goes back to the menu screen