Space Aliens - CircuitPython Game

Mr. Coxall

Jan 22, 2020

Contents

In this project we will be making a smart clock for the Raspbbery pi. We will be using an app that we will build using Tkinter module to create a smart clock. The we will be using more than one library to make it easy to the clock and other widgets, with helper libraries for sound. The app can also work on other variants of devices, like windows and mac. The full completed app code 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, functions and dictionaries will be able to follow along.

Parts

You will need the following items:

Raspbbery pi 3 B+

3.5 inch LCD screen

push button

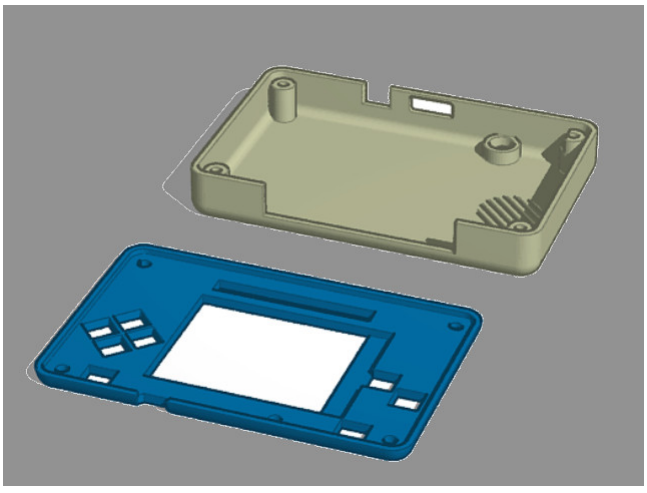
‘ Raspberry Pi 3B+ 3B Power Supply - 1.5 meter long <https://www.amazon.ca/Official-Raspberry-Pi-Supply-Version/dp/B01NCX6J2N>‘_

So you can power up the raspberry pi

breadboard to connect jumper wires and the push buttons

jumper wires to connect your products together

you might also want:



3D Printed Case

I created this case and it's based on the mesurments of the raspberrypi, the breadboard and the lcd screen I am using. You will need 3 x 4 screws to connect the parts together

CHAPTER 1

Install Raspian

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

Before doing anything else, you should format your sd card and install the latest version of Raspian onto it. This ensures you have a clean build with all the latest updates and no leftover files floating around. Rasberry has an excellent quick start guide here to step you through the process of getting the latest version of raspian onto your pi.

Just a reminder, if you are having any problems loading Raspian onto your Pi, ensure that you are using the right programs and a functional micro sd card. Once the Raspian is all loaded, come on back to continue the tutorial.

CHAPTER 2

Your IDE

One of the great things about raspbbery pi is that it is just a small sized a computer you simply need to install raspian on it then when you attach it to a monitor it will work as a computer. This means that you can access and save your code using any text editor installed at raspian. This is particularly helpful in schools, where computers are likely to be locked down so students can not load anything. To control the raspbbery pi from your own computer you will need an ssh connection

> your best beat for an IDE is Thonny Editor. it comes installed on the raspbbery pi. It works seamlessly and it will give you debugging information.

Since with CircuitPython devices you are just writing Python files, 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 Raspian loaded, you should make sure everything is working before you move on. To do this we will do the traditional "Hello, World!" program.

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

As soon as you save the file onto the pi, the terminal should come up 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 : Marwan Mashaly
# Created on : January 2020
# This program prints out Hello, World! onto the PyBadge
```

(continues on next page)

Fig. 1: Thonny IDE

Fig. 2: Hello, World! program

(continued from previous page)

```
6
7
8  def main():
9      # this function prints out Hello, World! onto the PyBadge
10
11     print("Hello, World!")
12
13
14 if __name__ == "__main__":
15     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: Image Bank for Space Aliens

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
- coin sound
- pew sound
- boom sound
- crash 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

Creating the gui

After writing your import statements you will go ahead and start in coding to show and create your Gui

```
root = Tk()
root.title('Smart Clock')
root.geometry("+200+200")

===========
```

The first line creates the Gui itself then the second one is to give it a title lastly the geometry is to resize your window or to open in a certain place in your desktop so since I am using a an lcd screen I need it to open in a certain place which is the middle of the screen

```
button1 = Button(26)
button2 = Button(19)
button3 = Button(6)
button4 = Button(5)
button5 = Button(16)
```

Those lines are to set the buttons for each gpio pin I used. you will need the gpio zero library to do that. you can't run the buttons code from your pc or laptop it has to be from the raspberry pi so make sure to comment it in your pc or laptop

```
news_frame = tk.Frame(root, bg='black')
news_frame.grid()

weather_frame = tk.Frame(root, bg='sky blue')
weather_frame.grid()

clock_frame = tk.Frame(root, bg='black')
clock_frame.grid()

calendar_frame = tk.Frame(root, bg='white')
```

(continues on next page)

(continued from previous page)

```
calendar_frame.grid()

time_frame = tk.Frame(root, bg='black')
time_frame.grid()
```

Lastly this is were we create the frames for each function so that we would be able to hide it if the other function runs. So for each of them the first line is to create it and set the background color and the second is to make it show up in the Gui screen if called.

CHAPTER 5

Menu System

X

5.1 Start Scene

X

5.2 Splash Scene

T

5.3 Game Over Scene

T