Space Aliens - CircuitPython Game

Mr. Coxall

Jan 16, 2020

Contents

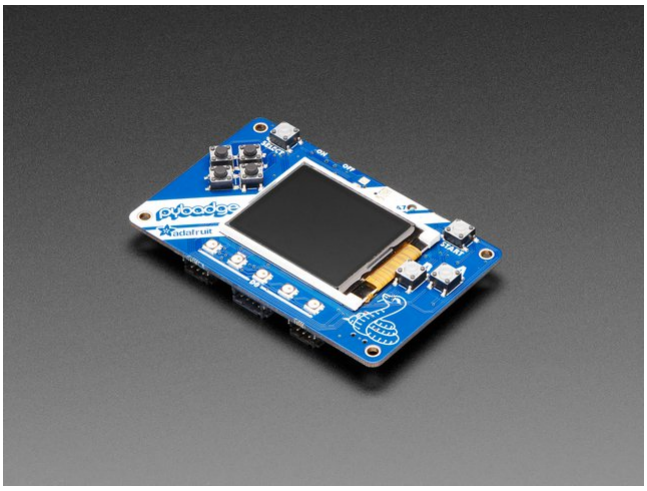
In this project we will be making an old school style video game for the Adafruit PyBadge. We will be using CircuitPython and the stage library to create a Guitar Hero like game. The stage library makes it easy to make classic video games, with helper libraries for sound, sprites and collision detection. 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 full game can be found at this GitHub link. You can download the repository and just copy the code over to your PyBadge. **Please remember NOT to copy over the docs folder!**

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.

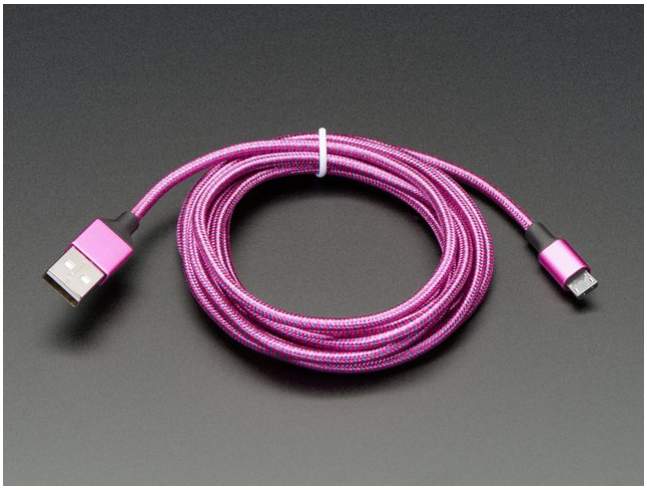
Parts

You will need the following items:



Adafruit PyBadge for MakeCode Arcade, CircuitPython or Arduino

PRODUCT ID: 4200

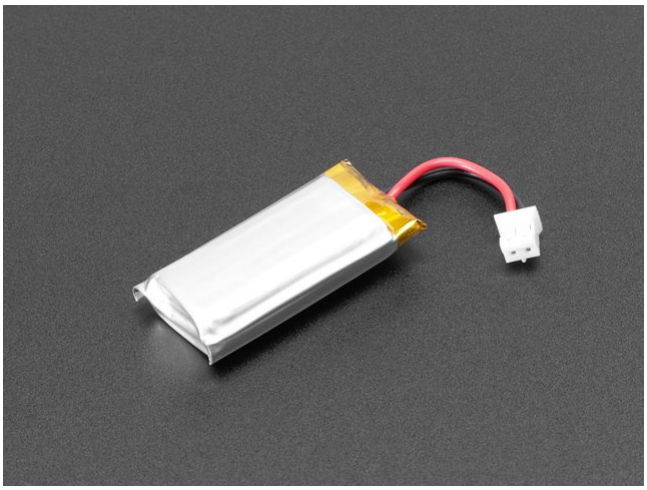


Pink and Purple Braided USB A to Micro B Cable - 2 meter long

PRODUCT ID: 4148

So you can move your CircuitPython code onto the PyBadge.

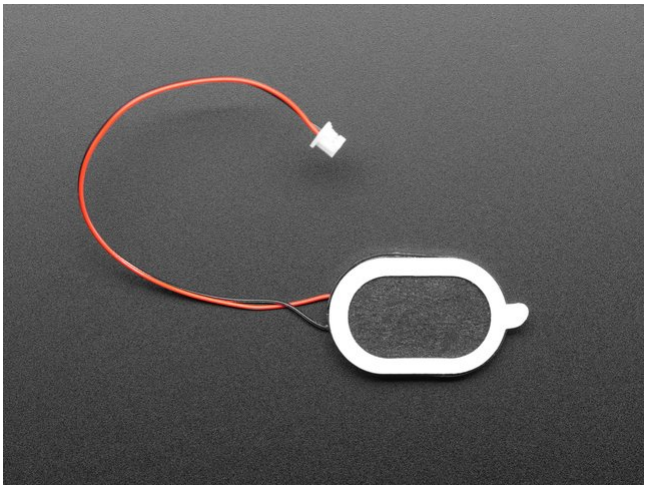
You might also want:



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

PRODUCT ID: 3898

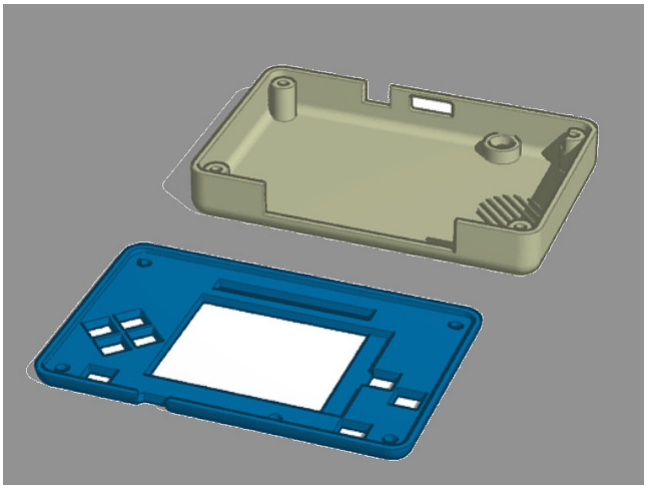
So that you can play the game without having it attached to a computer with a USB cable.



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.

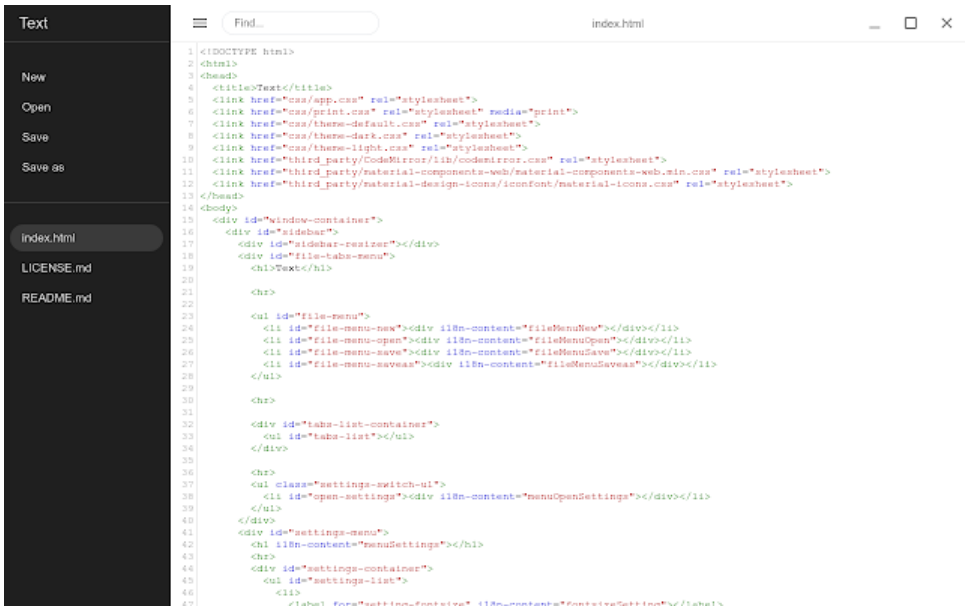


Fig. 1: Chromebook Text app

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.

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:

- jungle joe image bank
- backgrounds image bank
- elemental studios image bank
- coin sound
- boom 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 scene is where you will build this game, there are many parts you must include but the most important will be the collison between the buttons coming from the top of the screen and the buttons which indicate when to press the designated button. The animation on the left of the screen will not be shown how to create here but in the jungle joe animation section. After setting the background you must first we must make the button locations on the screen show up to know when to press the designated button which is just putting a sprite at a specific location and is shown in the code below. After the sprites are in place and your backgroud is set now we will get sprites to come from top of screen. Since there are six unique sprites and they need to have the same X location as there stationary counterparts that are on screen.

The most important part of this game is its collision, as any game of this genre has this as there main attraction. The collision is like any other when the two sprites are touching and the designated button is pressed, the moving sprite should be moved to the off screen location. The best part about this game is that there will never be two of the same games as there is a random factor which determines how many more sprites come down and which sprites come down. The code for the collision detection is apart of the game code which will be shown below:

Next we must make the lives and score system, In this game the user has 5 lives (unless playing on endless mode) and the score they can reach is unlimited. Every time the user presses the designated button at the correct time, the user will recieve a point. The code for this is your standard text code that is shown apart of the code below and the score palette is a palette you can use. The lives system is very simple, if a moving sprite reaches the max Y screen size + its sprite size the user will lose a life. The lives will be shown through the neopixels. Instead of putting lives on the screen and making the screen feel cramped, we will be moving the lives count to the neopixals that are at the bottom of the pybadge. The lights on the pybadge will be set to gren at the start of the game and when the user loses a life, one of the lights will turn to red until all are red and user loses the game. The code for the lives system is also shown in the finshed code for the game scene which is below.

Now we will be adding the finshing touches to the game scene, to get everything to load we must render all the sprites, and call the opther functions while taking the final hieght, which is shown in the jungle joe animation section, and the users final score. This is shown in the final code below.

After you have finished with the game scene you may add sounds if you desire, the sounds will mostly relate to jungle joe animation and when you lose the game. .. toctree:

```
def game_scene(game_mode):
 # this function is the game scene
```

(continues on next page)

(continued from previous page)

```
sprites = []
number_of_lives = 5
score = 0
button_speed = 1
height = 0
image_bank_5 = stage.Bank.from_bmp16("backgrounds.bmp")
image_bank_3 = stage.Bank.from_bmp16("jungle_joe.bmp")

a_button = constants.button_state["button_up"]
b_button = constants.button_state["button_up"]
up_button = constants.button_state["button_up"]
down_button = constants.button_state["button_up"]
left_button = constants.button_state["button_up"]
right_button = constants.button_state["button_up"]

background = stage.Grid(image_bank_5, constants.SCREEN_GRID_X, constants.SCREEN_GRID_
↪Y)
for x_location in range(constants.SCREEN_GRID_2_X):
    for y_location in range(constants.SCREEN_GRID_Y):
        tile_picked = random.randint(2,3)
        background.tile(x_location, y_location, tile_picked)
for x_location in range(constants.SCREEN_GRID_2_X, constants.SCREEN_GRID_X):
    for y_location in range(constants.SCREEN_GRID_Y):
        background.tile(x_location, y_location, 5)

# Displays key sprites.
a_button_sprite = stage.Sprite(image_bank_3, 12, constants.A_BUTTON, constants.
↪BUTTON_HEIGHT)
sprites.append(a_button_sprite)
b_button_sprite = stage.Sprite(image_bank_3, 11, constants.B_BUTTON, constants.
↪BUTTON_HEIGHT)
sprites.append(b_button_sprite)
left_arrow = stage.Sprite(image_bank_3, 8, constants.LEFT_BUTTON, constants.BUTTON_
↪HEIGHT)
sprites.append(left_arrow)
right_arrow = stage.Sprite(image_bank_3, 7, constants.RIGHT_BUTTON, constants.BUTTON_
↪HEIGHT)
sprites.append(right_arrow)
up_arrow = stage.Sprite(image_bank_3, 10, constants.UP_BUTTON, constants.BUTTON_
↪HEIGHT)
sprites.append(up_arrow)
down_arrow = stage.Sprite(image_bank_3, 9, constants.DOWN_BUTTON, constants.BUTTON_
↪HEIGHT)
sprites.append(down_arrow)

text = []
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(1, 1)
score_text.text("Score:{0}".format(score))
text.append(score_text)

pixels = neopixel.NeoPixel(board.NEOPIXEL, 5, auto_write=False)
for pixel_number in range(0, 5):
    pixels[pixel_number] = (0, 10, 0)
```

(continues on next page)

(continued from previous page)

```
pixels.show()

def score_update():
    # I know this is a function that is using variables outside of itself!
    # BUT this code is going to be used in multiple places
    # update the score when you correctly hit a button or when you hit a milestone
    score = score + 1
    # Refreshes score text
    score_text.clear()
    score_text.cursor(0, 0)
    score_text.move(1, 1)
    score_text.text("Score:{0}".format(score))
    game.render_block()

def show_abutton():
    # I know this is a function that is using variables outside of itself!
    #   BUT this code is going to be used in 2 places :)
    # make a button show up on screen in the x-axis
    for a_button_number in range(len(abutton)):
        if abutton[a_button_number].x < 0: # meaning it is off the screen, so␣
↪available to move on the screen
            abutton[a_button_number].move(constants.A_BUTTON, random.
↪randint(constants.OFF_SCREEN_Y, 0 - constants.SPRITE_SIZE))
            break

# create buttons
abutton = []
for a_button_number in range(constants.TOTAL_NUMBER_OF_A_BUTTON):
    a_single_abutton = stage.Sprite(image_bank_3, 6, constants.OFF_SCREEN_X,␣
↪constants.OFF_SCREEN_Y)
    abutton.append(a_single_abutton)

# current number of buttons that should be moving down screen, start with just 1
abutton_count = 1
show_abutton()

def show_bbutton():
    # I know this is a function that is using variables outside of itself!
    #   BUT this code is going to be used in 2 places :)
    # make an button show up on screen in the x-axis
    for b_button_number in range(len(bbutton)):
        if bbutton[b_button_number].x < 0: # meaning it is off the screen, so␣
↪available to move on the screen
            bbutton[b_button_number].move(constants.B_BUTTON, random.
↪randint(constants.OFF_SCREEN_Y, 0 - constants.SPRITE_SIZE))
            break

# create buttons
bbutton = []
for b_button_number in range(constants.TOTAL_NUMBER_OF_B_BUTTON):
    a_single_bbutton = stage.Sprite(image_bank_3, 5, constants.OFF_SCREEN_X,␣
↪constants.OFF_SCREEN_Y)
    bbutton.append(a_single_bbutton)

# current number of buttons that should be moving down screen, start with just 1
bbutton_count = 0
show_bbutton()
```

(continues on next page)

(continued from previous page)

```
def show_upbutton():
    # I know this is a function that is using variables outside of itself!
    #   BUT this code is going to be used in 2 places :)
    # make an button show up on screen in the x-axis
    for up_button_number in range(len(upbutton)):
        if upbutton[up_button_number].x < 0: # meaning it is off the screen, so␣
↪available to move on the screen
            upbutton[up_button_number].move(constants.UP_BUTTON, random.
↪randint(constants.OFF_SCREEN_Y, 0 - constants.SPRITE_SIZE))
            break

# create buttons
upbutton = []
for up_button_number in range(constants.TOTAL_NUMBER_OF_UP_BUTTON):
    a_single_upbutton = stage.Sprite(image_bank_3, 4, constants.OFF_SCREEN_X,␣
↪constants.OFF_SCREEN_Y)
    upbutton.append(a_single_upbutton)

# current number of buttons that should be moving down screen, start with just 1
upbutton_count = 0
show_upbutton()

def show_downbutton():
    # I know this is a function that is using variables outside of itself!
    #   BUT this code is going to be used in 2 places :)
    # make an button show up on screen in the x-axis
    for down_button_number in range(len(downbutton)):
        if downbutton[down_button_number].x < 0: # meaning it is off the screen, so␣
↪available to move on the screen
            downbutton[down_button_number].move(constants.DOWN_BUTTON, random.
↪randint(constants.OFF_SCREEN_Y, 0 - constants.SPRITE_SIZE))
            break

# create buttons
downbutton = []
for down_button_number in range(constants.TOTAL_NUMBER_OF_DOWN_BUTTON):
    a_single_downbutton = stage.Sprite(image_bank_3, 3, constants.OFF_SCREEN_X,␣
↪constants.OFF_SCREEN_Y)
    downbutton.append(a_single_downbutton)

# current number of buttons that should be moving down screen, start with just 1
downbutton_count = 0
show_downbutton()

def show_leftbutton():
    # I know this is a function that is using variables outside of itself!
    #   BUT this code is going to be used in 2 places :)
    # make an button show up on screen in the x-axis
    for left_button_number in range(len(leftbutton)):
        if leftbutton[left_button_number].x < 0: # meaning it is off the screen, so␣
↪available to move on the screen
            leftbutton[left_button_number].move(constants.LEFT_BUTTON, random.
↪randint(constants.OFF_SCREEN_Y, 0 - constants.SPRITE_SIZE))
            break

# create buttons
```

(continues on next page)

(continued from previous page)

```
leftbutton = []
for left_button_number in range(constants.TOTAL_NUMBER_OF_LEFT_BUTTON):
    a_single_leftbutton = stage.Sprite(image_bank_3, 2, constants.OFF_SCREEN_X,␣
↪constants.OFF_SCREEN_Y)
    leftbutton.append(a_single_leftbutton)

# current number of buttons that should be moving down screen, start with just 1
leftbutton_count = 0
show_leftbutton()

def show_rightbutton():
    # I know this is a function that is using variables outside of itself!
    #   BUT this code is going to be used in 2 places :)
    # make an button show up on screen in the x-axis
    for right_button_number in range(len(rightbutton)):
        if rightbutton[right_button_number].x < 0: # meaning it is off the screen,␣
↪so available to move on the screen
            rightbutton[right_button_number].move(constants.RIGHT_BUTTON, random.
↪randint(constants.OFF_SCREEN_Y, 0 - constants.SPRITE_SIZE))
            break

# create buttons
rightbutton = []
for right_button_number in range(constants.TOTAL_NUMBER_OF_RIGHT_BUTTON):
    a_single_rightbutton = stage.Sprite(image_bank_3, 1, constants.OFF_SCREEN_X,␣
↪constants.OFF_SCREEN_Y)
    rightbutton.append(a_single_rightbutton)

# current number of button that should be moving down screen, start with just 1
rightbutton_count = 0
show_rightbutton()

game = stage.Stage(ugame.display, constants.FPS)
game.layers = text + jungle_joe + logs + border + abutton + bbutton + upbutton +␣
↪downbutton + leftbutton + rightbutton + sprites + [background]

game.render_block()

# repeat forever, game loop
while True:
    # get user input
    keys = ugame.buttons.get_pressed()
    # update game logic
    if keys & ugame.K_X != 0:
        if a_button == constants.button_state["button_up"]:
            a_button = constants.button_state["button_just_pressed"]
        elif a_button == constants.button_state["button_just_pressed"]:
            a_button = constants.button_state["button_still_pressed"]
    else:
        if a_button == constants.button_state["button_still_pressed"]:
            a_button = constants.button_state["button_released"]
        else:
            a_button = constants.button_state["button_up"]

    if keys & ugame.K_O != 0:
        if b_button == constants.button_state["button_up"]:
            b_button = constants.button_state["button_just_pressed"]
```

(continues on next page)

(continued from previous page)

```
        elif b_button == constants.button_state["button_just_pressed"]:
            b_button = constants.button_state["button_still_pressed"]
    else:
        if b_button == constants.button_state["button_still_pressed"]:
            b_button = constants.button_state["button_released"]
        else:
            b_button = constants.button_state["button_up"]

    if keys & ugame.K_UP != 0:
        if up_button == constants.button_state["button_up"]:
            up_button = constants.button_state["button_just_pressed"]
        elif up_button == constants.button_state["button_just_pressed"]:
            up_button = constants.button_state["button_still_pressed"]
    else:
        if up_button == constants.button_state["button_still_pressed"]:
            up_button = constants.button_state["button_released"]
        else:
            up_button = constants.button_state["button_up"]

    if keys & ugame.K_DOWN != 0:
        if down_button == constants.button_state["button_up"]:
            down_button = constants.button_state["button_just_pressed"]
        elif down_button == constants.button_state["button_just_pressed"]:
            down_button = constants.button_state["button_still_pressed"]
    else:
        if down_button == constants.button_state["button_still_pressed"]:
            down_button = constants.button_state["button_released"]
        else:
            down_button = constants.button_state["button_up"]

    if keys & ugame.K_LEFT != 0:
        if left_button == constants.button_state["button_up"]:
            left_button = constants.button_state["button_just_pressed"]
        elif left_button == constants.button_state["button_just_pressed"]:
            left_button = constants.button_state["button_still_pressed"]
    else:
        if left_button == constants.button_state["button_still_pressed"]:
            left_button = constants.button_state["button_released"]
        else:
            left_button = constants.button_state["button_up"]

    if keys & ugame.K_RIGHT != 0:
        if right_button == constants.button_state["button_up"]:
            right_button = constants.button_state["button_just_pressed"]
        elif right_button == constants.button_state["button_just_pressed"]:
            right_button = constants.button_state["button_still_pressed"]
    else:
        if right_button == constants.button_state["button_still_pressed"]:
            right_button = constants.button_state["button_released"]
        else:
            right_button = constants.button_state["button_up"]

    for a_button_number in range(len(abutton)):
        if abutton_count > 0:
            if abutton[a_button_number].x > 0: # meaning it is on the screen
                abutton[a_button_number].move(abutton[a_button_number].x, abutton[a_
↪button_number].y + button_speed)
```

(continues on next page)

(continued from previous page)

```
                if abutton[a_button_number].y > constants.SCREEN_Y:
                    abutton[a_button_number].move(constants.OFF_SCREEN_X, constants.
↪OFF_SCREEN_Y)
                    show_abutton() # make it randomly show up at top again
                    if game_mode == 0:
                        number_of_lives = number_of_lives - 1

    for b_button_number in range(len(bbutton)):
        if bbutton_count > 0:
            if bbutton[b_button_number].x > 0: # meaning it is on the screen
                bbutton[b_button_number].move(bbutton[b_button_number].x, bbutton[b_
↪button_number].y + button_speed)
                if bbutton[b_button_number].y > constants.SCREEN_Y:
                    bbutton[b_button_number].move(constants.OFF_SCREEN_X, constants.
↪OFF_SCREEN_Y)
                    show_bbutton() # make it randomly show up at top again
                    if game_mode == 0:
                        number_of_lives = number_of_lives - 1

    for up_button_number in range(len(upbutton)):
        if upbutton_count > 0:
            if upbutton[up_button_number].x > 0: # meaning it is on the screen
                upbutton[up_button_number].move(upbutton[up_button_number].x,␣
↪upbutton[up_button_number].y + button_speed)
                if upbutton[up_button_number].y > constants.SCREEN_Y:
                    upbutton[up_button_number].move(constants.OFF_SCREEN_X,␣
↪constants.OFF_SCREEN_Y)
                    show_upbutton() # make it randomly show up at top again
                    if game_mode == 0:
                        number_of_lives = number_of_lives - 1

    for down_button_number in range(len(downbutton)):
        if downbutton_count > 0:
            if downbutton[down_button_number].x > 0: # meaning it is on the screen
                downbutton[down_button_number].move(downbutton[down_button_number].x,
↪ downbutton[down_button_number].y + button_speed)
                if downbutton[down_button_number].y > constants.SCREEN_Y:
                    downbutton[down_button_number].move(constants.OFF_SCREEN_X,␣
↪constants.OFF_SCREEN_Y)
                    show_downbutton() # make it randomly show up at top again
                    if game_mode == 0:
                        number_of_lives = number_of_lives - 1

    for left_button_number in range(len(leftbutton)):
        if leftbutton_count > 0:
            if leftbutton[left_button_number].x > 0: # meaning it is on the screen
                leftbutton[left_button_number].move(leftbutton[left_button_number].x,
↪ leftbutton[left_button_number].y + button_speed)
                if leftbutton[left_button_number].y > constants.SCREEN_Y:
                    leftbutton[left_button_number].move(constants.OFF_SCREEN_X,␣
↪constants.OFF_SCREEN_Y)
                    show_leftbutton() # make it randomly show up at top again
                    if game_mode == 0:
                        number_of_lives = number_of_lives - 1

    for right_button_number in range(len(rightbutton)):
        if rightbutton_count > 0:
```

(continues on next page)

(continued from previous page)

```
            if rightbutton[right_button_number].x > 0: # meaning it is on the screen
                rightbutton[right_button_number].move(rightbutton[right_button_
↪number].x, rightbutton[right_button_number].y + button_speed)
                if rightbutton[right_button_number].y > constants.SCREEN_Y:
                    rightbutton[right_button_number].move(constants.OFF_SCREEN_X,␣
↪constants.OFF_SCREEN_Y)
                    show_rightbutton() # make it randomly show up at top again
                    if game_mode == 0:
                        number_of_lives = number_of_lives - 1

    for a_button_number in range(len(abutton)):
        if abutton[a_button_number].x > 0 and a_button == constants.button_state[
↪"button_just_pressed"]:
            if stage.collide(abutton[a_button_number].x, abutton[a_button_number].y,
                             abutton[a_button_number].x, abutton[a_button_
↪number].y + 7,
                             a_button_sprite.x, a_button_sprite.y,
                             a_button_sprite.x, a_button_sprite.y + 7):
                # when you press designated button when it is on top of sprite
                abutton[a_button_number].move(constants.OFF_SCREEN_X, constants.OFF_
↪SCREEN_Y)
                score_update()
                abutton_count = 0
                rand_amount_number = random.randint(1, 2)
                for loop_counter in range(rand_amount_number):
                    random_selection = random.randint(1, 6)
                    if random_selection == 1:
                        abutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 2:
                        bbutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 3:
                        upbutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 4:
                        downbutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 5:
                        leftbutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 6:
                        rightbutton_count = 1
                        loop_counter = loop_counter + 1
                show_abutton()

    for b_button_number in range(len(bbutton)):
        if bbutton[b_button_number].x > 0 and b_button == constants.button_state[
↪"button_just_pressed"]:
            if stage.collide(bbutton[b_button_number].x, bbutton[b_button_number].y,
                             bbutton[b_button_number].x, bbutton[b_button_
↪number].y + 7,
                             b_button_sprite.x, b_button_sprite.y,
                             b_button_sprite.x, b_button_sprite.y + 7):
                # when you press designated button when it is on top of sprite
                bbutton[b_button_number].move(constants.OFF_SCREEN_X, constants.OFF_
↪SCREEN_Y)
```

(continues on next page)

(continued from previous page)

```
                    score_update()
                    bbutton_count = 0
                    rand_amount_number = random.randint(1, 2)
                    for loop_counter in range(rand_amount_number):
                        random_selection = random.randint(1, 6)
                        if random_selection == 1:
                            abutton_count = 1
                            loop_counter = loop_counter + 1
                        elif random_selection == 2:
                            bbutton_count = 1
                            loop_counter = loop_counter + 1
                        elif random_selection == 3:
                            upbutton_count = 1
                            loop_counter = loop_counter + 1
                        elif random_selection == 4:
                            downbutton_count = 1
                            loop_counter = loop_counter + 1
                        elif random_selection == 5:
                            leftbutton_count = 1
                            loop_counter = loop_counter + 1
                        elif random_selection == 6:
                            rightbutton_count = 1
                            loop_counter = loop_counter + 1
                    show_bbutton()

        for up_button_number in range(len(upbutton)):
            if upbutton[up_button_number].x > 0 and up_button == constants.button_state[
↪"button_just_pressed"]:
                if stage.collide(upbutton[up_button_number].x, upbutton[up_button_
↪number].y,
                                 upbutton[up_button_number].x, upbutton[up_button_
↪number].y + 7,
                                 up_arrow.x, up_arrow.y,
                                 up_arrow.x, up_arrow.y + 7):
                    # when you press designated button when it is on top of sprite
                    upbutton[up_button_number].move(constants.OFF_SCREEN_X, constants.
↪OFF_SCREEN_Y)
                    score_update()
                    upbutton_count = 0
                    rand_amount_number = random.randint(1, 2)
                    for loop_counter in range(rand_amount_number):
                        random_selection = random.randint(1, 6)
                        if random_selection == 1:
                            abutton_count = 1
                            loop_counter = loop_counter + 1
                        elif random_selection == 2:
                            bbutton_count = 1
                            loop_counter = loop_counter + 1
                        elif random_selection == 3:
                            upbutton_count = 1
                            loop_counter = loop_counter + 1
                        elif random_selection == 4:
                            downbutton_count = 1
                            loop_counter = loop_counter + 1
                        elif random_selection == 5:
                            leftbutton_count = 1
                            loop_counter = loop_counter + 1
```

(continues on next page)

(continued from previous page)

```
                    elif random_selection == 6:
                        rightbutton_count = 1
                        loop_counter = loop_counter + 1
                show_upbutton()

    for down_button_number in range(len(downbutton)):
        if downbutton[down_button_number].x > 0 and down_button == constants.button_
↪state["button_just_pressed"]:
            if stage.collide(downbutton[down_button_number].x, downbutton[down_
↪button_number].y,
                             downbutton[down_button_number].x, downbutton[down_
↪button_number].y + 7,
                             down_arrow.x, down_arrow.y,
                             down_arrow.x, down_arrow.y + 7):
                # when you press designated button when it is on top of sprite
                downbutton[down_button_number].move(constants.OFF_SCREEN_X,␣
↪constants.OFF_SCREEN_Y)
                score_update()
                downbutton_count = 0
                rand_amount_number = random.randint(1, 2)
                for loop_counter in range(rand_amount_number):
                    random_selection = random.randint(1, 6)
                    if random_selection == 1:
                        abutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 2:
                        bbutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 3:
                        upbutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 4:
                        downbutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 5:
                        leftbutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 6:
                        rightbutton_count = 1
                        loop_counter = loop_counter + 1
                show_downbutton()

    for left_button_number in range(len(leftbutton)):
        if leftbutton[left_button_number].x > 0 and left_button == constants.button_
↪state["button_just_pressed"]:
            if stage.collide(leftbutton[left_button_number].x, leftbutton[left_
↪button_number].y,
                             leftbutton[left_button_number].x, leftbutton[left_
↪button_number].y + 7,
                             left_arrow.x, left_arrow.y,
                             left_arrow.x, left_arrow.y + 7):
                # when you press designated button when it is on top of sprite
                leftbutton[left_button_number].move(constants.OFF_SCREEN_X,␣
↪constants.OFF_SCREEN_Y)
                score_update()
                leftbutton_count = 0
                rand_amount_number = random.randint(1, 2)
```

(continues on next page)

(continued from previous page)

```
                for loop_counter in range(rand_amount_number):
                    random_selection = random.randint(1, 6)
                    if random_selection == 1:
                        abutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 2:
                        bbutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 3:
                        upbutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 4:
                        downbutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 5:
                        leftbutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 6:
                        rightbutton_count = 1
                        loop_counter = loop_counter + 1
                show_leftbutton()

    for right_button_number in range(len(rightbutton)):
        if rightbutton[right_button_number].x > 0 and right_button == constants.
↪button_state["button_just_pressed"]:
            if stage.collide(rightbutton[right_button_number].x, rightbutton[right_
↪button_number].y,
                             rightbutton[right_button_number].x, rightbutton[right_
↪button_number].y + 7,
                             right_arrow.x, right_arrow.y,
                             right_arrow.x, right_arrow.y + 7):
                # when you press designated button when it is on top of sprite
                rightbutton[right_button_number].move(constants.OFF_SCREEN_X,␣
↪constants.OFF_SCREEN_Y)
                score_update()
                rightbutton_count = 0
                rand_amount_number = random.randint(1, 2)
                for loop_counter in range(rand_amount_number):
                    random_selection = random.randint(1, 6)
                    if random_selection == 1:
                        abutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 2:
                        bbutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 3:
                        upbutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 4:
                        downbutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 5:
                        leftbutton_count = 1
                        loop_counter = loop_counter + 1
                    elif random_selection == 6:
                        rightbutton_count = 1
                        loop_counter = loop_counter + 1
```

(continues on next page)

(continued from previous page)

```
                show_rightbutton()

    if number_of_lives == 5:
        for pixel_number in range(0, 5):
                pixels[pixel_number] = (0, 10, 0)
                pixels.show()
    if number_of_lives == 4:
        for pixel_number in range(1):
                pixels[pixel_number] = (25, 0, 0)
                pixels.show()
    if number_of_lives == 3:
        for pixel_number in range(2):
                pixels[pixel_number] = (25, 0, 0)
                pixels.show()
    if number_of_lives == 2:
        for pixel_number in range(3):
                pixels[pixel_number] = (25, 0, 0)
                pixels.show()
    if number_of_lives == 1:
        for pixel_number in range(4):
                pixels[pixel_number] = (25, 0, 0)
                pixels.show()
    if number_of_lives == 0:
        for pixel_number in range(5):
                pixels[pixel_number] = (25, 0, 0)
                pixels.show()
                game_over_scene(score, height)

    # redraw sprite list
    game.render_sprites(logs + sprites + jungle_joe + abutton + bbutton + upbutton +␣
↪downbutton + leftbutton + rightbutton)
    game.tick()  # wait until refresh rate finishes
```

4.1 Background

The game scene's background is split up into two different portions. These being the jungle side, which takes up roughly 2/5ths of the screen, and the game portion, which takes up roughlt 3/5ths of the screen. To do this, your going to need to paint the first 2/5ths of the screen (x grid = 0-4) with the tree sprite (sprites 2 and 3 of backgrounds) and the other 3/5ths (5-10) with the grey background sprite (sprite 5 of backgrounds) using the code below. Finally, these last steps are optional, but if you want you can randomize the tree sprites on the left hand of the screen using the lines of the code below. Another thing you can do is generate a border between the two backgrounds. You can make this by using the border sprite (sprite 6 of backgrounds) and the code below. .. toctree:

```
code.py:
image_bank_5 = stage.Bank.from_bmp16("backgrounds.bmp")

background = stage.Grid(image_bank_5, constants.SCREEN_GRID_X, constants.SCREEN_GRID_
↪Y)
    for x_location in range(constants.SCREEN_GRID_2_X):
        for y_location in range(constants.SCREEN_GRID_Y):
            tile_picked = random.randint(2,3)
            background.tile(x_location, y_location, tile_picked)
    for x_location in range(constants.SCREEN_GRID_2_X, constants.SCREEN_GRID_X):
        for y_location in range(constants.SCREEN_GRID_Y):
```

(continues on next page)

(continued from previous page)

```
            background.tile(x_location, y_location, 5)

# Displays the border.
border_1 = stage.Sprite(image_bank_5, 6, constants.BORDER_LOCATION, 0)
border.append(border_1)
border_2 = stage.Sprite(image_bank_5, 6, constants.BORDER_LOCATION, 16)
border.append(border_2)
border_3 = stage.Sprite(image_bank_5, 6, constants.BORDER_LOCATION, 32)
border.append(border_3)
border_4 = stage.Sprite(image_bank_5, 6, constants.BORDER_LOCATION, 48)
border.append(border_4)
border_5 = stage.Sprite(image_bank_5, 6, constants.BORDER_LOCATION, 64)
border.append(border_5)
border_6 = stage.Sprite(image_bank_5, 6, constants.BORDER_LOCATION, 80)
border.append(border_6)
border_7 = stage.Sprite(image_bank_5, 6, constants.BORDER_LOCATION, 96)
border.append(border_7)
border_8 = stage.Sprite(image_bank_5, 6, constants.BORDER_LOCATION, 112)
border.append(border_8)

constants.py:
SCREEN_GRID_X = 16
SCREEN_GRID_Y = 8
SCREEN_GRID_2_X = 4
```

4.2 Jungle Joe

Jungle Joe is completely optional to the actual game, but he adds alot to it with his animations. To start, you need to make two sprites using sprites 14 and 15 of the image bank, jungle_joe. To make his animations, you are going to want to make a new function called score_update to save space and so you wont need to write the code six times. A quick sumary of how the code works is that the game checks to see where jungle joe is, then moves him accordingly so that he is above the next log up. Finally, it moves everything downwards and spawns a new log. .. toctree:

```
code.py:
# Displays Jungle Joe and logs
jungle_joe_standing = stage.Sprite(image_bank_3, 15, constants.OFF_SCREEN_X,␣
↪constants.OFF_SCREEN_Y )
jungle_joe.append(jungle_joe_standing)
jungle_joe_jumping = stage.Sprite(image_bank_3, 14, constants.JUNGLE_JOE_START_X,␣
↪constants.OFF_TOP_SCREEN)
jungle_joe.append(jungle_joe_jumping)

 logs = []
for log_number in range(constants.TOTAL_NUMBER_OF_A_LOGS):
    a_single_log = stage.Sprite(image_bank_3, 13, constants.OFF_SCREEN_X, constants.
↪OFF_SCREEN_Y)
    logs.append(a_single_log)

logs[0].move(constants.LOG_1_START_X, constants.LOG_1_START_Y)
logs[1].move(constants.RIGHT_LOG, constants.LOG_2_START_Y)

def score_update():
    # I know this is a function that is using variables outside of itself!
    #   BUT this code is going to be used in multiple places
```

(continues on next page)

(continued from previous page)

```
    # update the score when you correctly hit a button or when you hit a milestone
    score = score + 1
    # Refreshes score text
    score_text.clear()
    score_text.cursor(0, 0)
    score_text.move(1, 1)
    score_text.text("Score:{0}".format(score))
    game.render_block()
    if score % 10 == 0:
        sound.play(coin_sound)
        height = height + 24
        button_speed += constants.SPEED_INCREASE
        jungle_joe[1].move(jungle_joe[0].x, jungle_joe[0].y)
        jungle_joe[0].move(constants.OFF_SCREEN_X, constants.OFF_SCREEN_Y)
        while True:
            if logs[0].y < 50:
                if jungle_joe[1].x > logs[0].x:
                    jungle_joe[1].move(jungle_joe[1].x - constants.JUNGLE_JOE_X_SPEED,
↪ jungle_joe[1].y)
                if jungle_joe[1].y > logs[0].y - constants.SPRITE_SIZE:
                    jungle_joe[1].move(jungle_joe[1].x, jungle_joe[1].y - constants.
↪JUNGLE_JOE_Y_SPEED)
            if jungle_joe[1].x == logs[0].x and jungle_joe[1].y == logs[0].y -␣
↪constants.SPRITE_SIZE and jungle_joe[1].y < 50:
                jungle_joe[0].move(jungle_joe[1].x, jungle_joe[1].y)
                jungle_joe[1].move(constants.OFF_SCREEN_X, constants.OFF_SCREEN_Y)
                break

constants.py
JUNGLE_JOE_START_X = 0
JUNGLE_JOE_NORMAL_Y = 97
TOTAL_NUMBER_OF_A_LOGS = 2
SCROLL_SPEED = 4
LOG_1_START_X = 0
LOG_1_START_Y = 112
RIGHT_LOG = 40
LOG_2_START_Y = 48
INCOMING_LOG_HEIGHT = 0
LEFT_LOG = 0
JUNGLE_JOE_X_SPEED = 0.5
JUNGLE_JOE_Y_SPEED = 1
SCROLL_SPEED = 1
SPEED_INCREASE = 0.5
```

CHAPTER 5

Menu System

This section splits off into the trhee menu scenes in the game, click the links to go to each one.

5.1 Start Scene

[IMPORTANT] Most of the menu scene here is COMPLETELY optional. You can easily make a good menu where you press start to go to gaem scene. as such, I will not explain how I turned it into a menu that you would find in any other modern game but I will give you a quick summary. Basically, this program only changes the text to give off the appearance that the option is selected, while in actuality I just have the buttons change the option number (to change the text/start game) and the game mode(to change game mode). Other than that, the rest is very simple. i simply painted the top 6/8ths of the screen with blue, the bottom 2/8ths of the screen black and placed the tree top sprites just over it. Also, I decided to spawn clouds by choosing a randon Y value to spawn at (off screen) and then having their x value increase by 1 each time the game goes through the while true loop. Fianlly, I placed jungle joe and the sun just like any other sprites and placed text on the screen where I wanted. I did all this using the code below. .. toctree:

```
code.py
# this function is the main menu scene
text = []
sprites = []
sun = []
game_mode_text = []
game_mode = 0
option = 1

image_bank_5 = stage.Bank.from_bmp16("Backgrounds.bmp")
image_bank_5 = stage.Bank.from_bmp16("backgrounds.bmp")
image_bank_3 = stage.Bank.from_bmp16("jungle_joe.bmp")

# sets the background to image 0 in the bank
background = stage.Grid(image_bank_5, constants.SCREEN_GRID_X, constants.SCREEN_GRID_
↪Y)
```

(continues on next page)

(continued from previous page)

```
for x_location in range(constants.SCREEN_GRID_X):
    for y_location in range(constants.TREE_TOP_GRID_Y, constants.TREE_TOP_GRID_2_Y):
        background.tile(x_location, y_location, 1)
for x_location in range(constants.SCREEN_GRID_X):
    for y_location in range(constants.BLACK_BACK_GRID_Y, constants.BLACK_BACK_GRID_2_
↪Y):
        background.tile(x_location, y_location, 7)

coin_sound = open("coin.wav", 'rb')
sound = ugame.audio
sound.stop()
sound.mute(False)

text_1 = stage.Text(width=29, height=14, font=None, palette=constants.SCORE_PALETTE,␣
↪buffer=None)
text_1.move(40, 20)
text_1.text("JUNGLE JOE")
text.append(text_1)

text_2 = stage.Text(width=29, height=14, font=None, palette=constants.SCORE_PALETTE,␣
↪buffer=None)
text_2.move(40, 30)
text_2.text("& SNAKOB'S")
text.append(text_2)

text_3 = stage.Text(width=29, height=14, font=None, palette=constants.SCORE_PALETTE,␣
↪buffer=None)
text_3.move(25, 40)
text_3.text("BONGO BANANZA!")
text.append(text_3)

text_3 = stage.Text(width=29, height=14, font=None, palette=constants.SCORE_PALETTE,␣
↪buffer=None)
text_3.move(0, 0)
text_3.text("Version:{0}".format(constants.VERSION_NUMBER))
text.append(text_3)

start_text = stage.Text(width=29, height=14, font=None, palette=constants.SCORE_
↪PALETTE, buffer=None)
start_text.clear()
start_text.cursor(0, 0)
start_text.move(constants.START_X, constants.START_Y)
start_text.text("   START   ")
text.append(start_text)

game_mode_text = stage.Text(width=29, height=14, font=None, palette=constants.SCORE_
↪PALETTE, buffer=None)
game_mode_text.clear()
game_mode_text.cursor(0, 0)
game_mode_text.move(constants.GAME_MODE_1_X, constants.GAME_MODE_Y)
game_mode_text.text("<< NORMAL MODE >>")
text.append(game_mode_text)

a_button = constants.button_state["button_up"]
b_button = constants.button_state["button_up"]
up_button = constants.button_state["button_up"]
down_button = constants.button_state["button_up"]
```

(continues on next page)

(continued from previous page)

```
start_button = constants.button_state["button_up"]
select_button = constants.button_state["button_up"]

# Displays the sun
sun_top_left = stage.Sprite(image_bank_5, 11, 128, 0)
sun.append(sun_top_left)
sun_top_right = stage.Sprite(image_bank_5, 10, 144, 0)
sun.append(sun_top_right)
sun_bottom_left = stage.Sprite(image_bank_5, 8, 128, 16)
sun.append(sun_bottom_left)
sun_bottom_right = stage.Sprite(image_bank_5, 9, 144, 16)
sun.append(sun_bottom_right)
# Displays Jungle Joe
jungle_joe = stage.Sprite(image_bank_3, 15, 71, 66)
sprites.append(jungle_joe)
jungle_joe_jumping = stage.Sprite(image_bank_3, 14, constants.OFF_SCREEN_X, constants.
↪OFF_SCREEN_Y)
sprites.append(jungle_joe_jumping)

clouds = []
for cloud_number in range(constants.TOTAL_CLOUDS):
    a_single_cloud = stage.Sprite(image_bank_5, 4, constants.OFF_SCREEN_X, constants.
↪OFF_SCREEN_Y)
    clouds.append(a_single_cloud)

def Show_clouds():
    for cloud_number in range(len(clouds)):
        if clouds[cloud_number].y < 0:
            clouds[cloud_number].move(constants.OFF_LEFT_SCREEN, random.randint(0 -␣
↪constants.SPRITE_SIZE, constants.CLOUD_SPAWN_Y - constants.SPRITE_SIZE))
            break

cloud_count = 6
Show_clouds()
Show_clouds()

game = stage.Stage(ugame.display, constants.FPS)
# set the layers, items show up in order
game.layers = sprites + text + clouds + sun + [background]
# render the background and inital location of sprite list
# most likely you will only render background once per scene
# wait until refresh rate finishes
game.render_block()
# repeat forever, game loop

while True:
    # get user input
    keys = ugame.buttons.get_pressed()

    #print(keys)
    if keys & ugame.K_UP != 0:
        if up_button == constants.button_state["button_up"]:
            up_button = constants.button_state["button_just_pressed"]
        elif up_button == constants.button_state["button_just_pressed"]:
            up_button = constants.button_state["button_still_pressed"]
    else:
        if up_button == constants.button_state["button_still_pressed"]:
```

(continues on next page)

(continued from previous page)

```
            up_button = constants.button_state["button_released"]
        else:
            up_button = constants.button_state["button_up"]

    if keys & ugame.K_DOWN != 0:
        if down_button == constants.button_state["button_up"]:
            down_button = constants.button_state["button_just_pressed"]
        elif down_button == constants.button_state["button_just_pressed"]:
            down_button = constants.button_state["button_still_pressed"]
    else:
        if down_button == constants.button_state["button_still_pressed"]:
            down_button = constants.button_state["button_released"]
        else:
            down_button = constants.button_state["button_up"]

    if keys & ugame.K_X != 0:
        if a_button == constants.button_state["button_up"]:
            a_button = constants.button_state["button_just_pressed"]
        elif a_button == constants.button_state["button_just_pressed"]:
            a_button = constants.button_state["button_still_pressed"]
    else:
        if a_button == constants.button_state["button_still_pressed"]:
            a_button = constants.button_state["button_released"]
        else:
            a_button = constants.button_state["button_up"]

    if keys & ugame.K_O != 0:
        if b_button == constants.button_state["button_up"]:
            b_button = constants.button_state["button_just_pressed"]
        elif b_button == constants.button_state["button_just_pressed"]:
            b_button = constants.button_state["button_still_pressed"]
    else:
        if b_button == constants.button_state["button_still_pressed"]:
            b_button = constants.button_state["button_released"]
        else:
            b_button = constants.button_state["button_up"]

    if keys & ugame.K_SELECT != 0:
        if select_button == constants.button_state["button_up"]:
            select_button = constants.button_state["button_just_pressed"]
        elif select_button == constants.button_state["button_just_pressed"]:
            select_button = constants.button_state["button_still_pressed"]
    else:
        if select_button == constants.button_state["button_still_pressed"]:
            select_button = constants.button_state["button_released"]
        else:
            select_button = constants.button_state["button_up"]

    if keys & ugame.K_START != 0:
        if start_button == constants.button_state["button_up"]:
            start_button = constants.button_state["button_just_pressed"]
        elif start_button == constants.button_state["button_just_pressed"]:
            start_button = constants.button_state["button_still_pressed"]
    else:
        if start_button == constants.button_state["button_still_pressed"]:
            start_button = constants.button_state["button_released"]
        else:
```

(continues on next page)

(continued from previous page)

```
            start_button = constants.button_state["button_up"]

    if down_button == constants.button_state["button_just_pressed"] or up_button ==␣
↪constants.button_state["button_just_pressed"]:
        if option == 0:
            option = 1
            start_text.clear()
            start_text.cursor(0, 0)
            start_text.move(constants.START_X, constants.START_Y)
            start_text.text("   START   ")
            game.render_block()
            if game_mode == 0:
                game_mode_text.clear()
                game_mode_text.cursor(0, 0)
                game_mode_text.move(constants.GAME_MODE_1_X, constants.GAME_MODE_Y)
                game_mode_text.text("<< NORMAL MODE >>")
                game.render_block()
            elif game_mode == 1:
                game_mode_text.clear()
                game_mode_text.cursor(0, 0)
                game_mode_text.move(constants.GAME_MODE_2_X, constants.GAME_MODE_Y)
                game_mode_text.text("<< ENDLESS MODE >>")
                game.render_block()
        elif option == 1:
            option = 0
            start_text.clear()
            start_text.cursor(0, 0)
            start_text.move(constants.START_X, constants.START_Y)
            start_text.text("<< START >>")
            game.render_block()
            if game_mode == 0:
                game_mode_text.clear()
                game_mode_text.cursor(0, 0)
                game_mode_text.move(constants.GAME_MODE_1_X, constants.GAME_MODE_Y)
                game_mode_text.text("   NORMAL MODE   ")
                game.render_block()
            elif game_mode == 1:
                game_mode_text.clear()
                game_mode_text.cursor(0, 0)
                game_mode_text.move(constants.GAME_MODE_2_X, constants.GAME_MODE_Y)
                game_mode_text.text("   ENDLESS MODE   ")
                game.render_block()

    if (start_button == constants.button_state["button_just_pressed"] or select_
↪button == constants.button_state["button_just_pressed"]
        or a_button == constants.button_state["button_just_pressed"] or b_button ==␣
↪constants.button_state["button_just_pressed"]):
        if option == 0:
            sound.play(coin_sound)
            jungle_joe_jumping.move(jungle_joe.x, jungle_joe.y)
            jungle_joe.move(constants.OFF_SCREEN_X, constants.OFF_SCREEN_Y)
            while True:
                if jungle_joe_jumping.y > 50:
                    jungle_joe_jumping.move(jungle_joe_jumping.x, jungle_joe_jumping.
↪y - constants.JUNGLE_JOE_Y_SPEED)
                    game.render_sprites(sprites)
                    game.tick()
```

(continues on next page)

(continued from previous page)

```
                else:
                    break
            while True:
                jungle_joe_jumping.move(jungle_joe_jumping.x, jungle_joe_jumping.y +␣
↪constants.JUNGLE_JOE_Y_SPEED)
                game.render_sprites(sprites)
                game.tick()
                if jungle_joe_jumping.y > constants.SCREEN_Y:
                    game_scene(game_mode)
        elif option == 1:
            if game_mode == 1:
                game_mode = 0
                game_mode_text.clear()
                game_mode_text.cursor(0, 0)
                game_mode_text.move(constants.GAME_MODE_1_X, constants.GAME_MODE_Y)
                game_mode_text.text("<< NORMAL MODE >>")
                game.render_block()
            elif game_mode == 0:
                game_mode = 1
                game_mode_text.clear()
                game_mode_text.cursor(0, 0)
                game_mode_text.move(constants.GAME_MODE_2_X, constants.GAME_MODE_Y)
                game_mode_text.text("<< ENDLESS MODE >>")
                game.render_block()

    # update game logic
    for cloud_number in range (len(clouds)):
        if clouds[cloud_number].y > 0:
            clouds[cloud_number].move(clouds[cloud_number].x
                                      + constants.CLOUD_SPEED,
                                      clouds[cloud_number].y)
            if clouds[cloud_number].x > constants.SCREEN_X + constants.SPRITE_SIZE:
                clouds[cloud_number].move(constants.OFF_SCREEN_X,
                                          constants.OFF_SCREEN_Y)
                Show_clouds()
            if clouds[cloud_number].x > constants.SCREEN_X / 2:
                Show_clouds()
    # redraw sprite list
    pass # just a placeholder until you write the code

    game.render_sprites(clouds)
    game.tick()

constants.py
VERSION_NUMBER = "1.0.1"
TREE_TOP_GRID_Y = 5
TREE_TOP_GRID_2_Y = 6
BLACK_BACK_GRID_Y = 6
BLACK_BACK_GRID_2_Y = 8
SPRITE_SIZE = 16
TOTAL_CLOUDS = 5
CLOUD_SPEED = 0.25
CLOUD_SPAWN_Y = 80
GAME_MODE_1_X = 10
GAME_MODE_2_X = 8
GAME_MODE_Y = 100
START_X = 35
```

(continues on next page)

(continued from previous page)

```
START_Y = 118
```

5.2 Splash Scene

in our version of the game, we have a splash screen with our company logo, Elemental Studios. However, this step will most likely vary from person to person as you may want to use your own company logo. To do this, we used two seperate image banks so we can have more sprites on the screen for the elements in the corners. This, hwoever, meant that we needed to render each part of the elements in the corners as seperate sprites. You can do something similar with the following code. .. toctree:

```
code.py
def game_splash_scene():
# this function is the game scene
text = []
sprites = []
image_bank_3 = stage.Bank.from_bmp16("jungle_joe.bmp")
image_bank_4 = stage.Bank.from_bmp16("elemental_studios.bmp")
# sets the background to image 0 in the bank
background = stage.Grid(image_bank_3, constants.SCREEN_GRID_X, constants.SCREEN_GRID_
↪Y)
text_1 = stage.Text(width=29, height=14, font=None, palette=constants.SCORE_PALETTE,␣
↪buffer=None)
text_1.move(13, 60)
text_1.text("ELEMENTAL STUDIOS")
text.append(text_1)

text_2 = stage.Text(width=29, height=14, font=None, palette=constants.SCORE_PALETTE,␣
↪buffer=None)
text_2.move(40, 80)
text_2.text("PRESENTS...")
text.append(text_2)

fire_upper_right = stage.Sprite(image_bank_4, 0, 16, 0)
sprites.append(fire_upper_right)
fire_bottom_right = stage.Sprite(image_bank_4, 1, 16, 16)
sprites.append(fire_bottom_right)
fire_upper_left = stage.Sprite(image_bank_4, 2, 0, 0)
sprites.append(fire_upper_left)
fire_bottom_left = stage.Sprite(image_bank_4, 3, 0, 16)
sprites.append(fire_bottom_left)
water_upper_right = stage.Sprite(image_bank_4, 6, 144, 0)
sprites.append(water_upper_right)
water_bottom_right = stage.Sprite(image_bank_4, 7, 144, 16)
sprites.append(water_bottom_right)
water_upper_left = stage.Sprite(image_bank_4, 4, 128, 0)
sprites.append(water_upper_left)
water_bottom_left = stage.Sprite(image_bank_4, 5, 128, 16)
sprites.append(water_bottom_left)
earth_upper_right = stage.Sprite(image_bank_4, 10, 16, 98)
sprites.append(earth_upper_right)
earth_bottom_right = stage.Sprite(image_bank_4, 11, 16, 112)
sprites.append(earth_bottom_right)
earth_upper_left = stage.Sprite(image_bank_4, 8, 0, 98)
sprites.append(earth_upper_left)
```

(continues on next page)

(continued from previous page)

```
earth_bottom_left = stage.Sprite(image_bank_4, 9, 0, 112)
sprites.append(earth_bottom_left)
wind_upper_right = stage.Sprite(image_bank_4, 14, 144, 98)
sprites.append(wind_upper_right)
wind_bottom_right = stage.Sprite(image_bank_4, 15, 144, 112)
sprites.append(wind_bottom_right)
wind_upper_left = stage.Sprite(image_bank_4, 12, 128, 98)
sprites.append(wind_upper_left)
wind_bottom_left = stage.Sprite(image_bank_4, 13, 128, 112)
sprites.append(wind_bottom_left)
game = stage.Stage(ugame.display, 60)
# set the layers, items show up in order
game.layers = sprites + text + [background]
# render the background and inital location of sprite list
# most likely you will only render background once per scene
# wait until refresh rate finishes
game.render_block()
# repeat forever, game loop
while True:
    # get user input
    # update game logic
    time.sleep(1.0)
    main_menu_scene()
    # redraw sprite list
    pass # just a placeholder until you write the code

constants.py
SCREEN_GRID_X = 16
SCREEN_GRID_Y = 8
```

5.3 Game Over Scene

The game over scene does not take a lot of work, firstly make sure you set score and height as parameters when you call this function as you will need to use them here.

The only image bank you will need is the jungle joe image bank as it will act as the background for this scene. You will need both score and mt game studio palette to make this scene.

Now set the background like we have done in all the other scenes, after the background is set we must create the text for this scene. The text will display your final score, your final height, GAME OVER, and if you would like to retry or return to menu scene.

The code to display the text is: .. toctree:

```
text = []

text0 = stage.Text(width=29, height=14, font=None, palette=constants.SCORE_PALETTE,␣
↪buffer=None)
text0.move(22, 20)
text0.text("Final Score: {:0>2d}".format(final_score))
text.append(text0)

text2 = stage.Text(width=29, height=14, font=None, palette=constants.SCORE_PALETTE,␣
↪buffer=None)
text2.move(37, 30)
```

(continues on next page)

(continued from previous page)

```
text2.text("Height: {:0>2d}ft".format(final_height))
text.append(text2)

text1 = stage.Text(width=29, height=14, font=None, palette=constants.MT_GAME_STUDIO_
↪PALETTE, buffer=None)
text1.move(43, 60)
text1.text("GAME OVER")
text.append(text1)

menu_text = stage.Text(width=29, height=14, font=None, palette=constants.SCORE_
↪PALETTE, buffer=None)
menu_text.clear()
menu_text.cursor(0, 0)
menu_text.move(constants.MENU_LOCATION_X, constants.MENU_LOCATION_Y)
menu_text.text("   MENU   ")
text.append(menu_text)

retry_text = stage.Text(width=29, height=14, font=None, palette=constants.SCORE_
↪PALETTE, buffer=None)
retry_text.clear()
retry_text.cursor(0, 0)
retry_text.move(constants.RETRY_LOCATION_X, constants.RETRY_LOCATION_Y)
retry_text.text("<< RETRY >>")
text.append(retry_text)
```

Next we need to set up the option to go back to menu or retry the game. To do this we have to set up the buttons to accept inputs and to change which screen you go to depending on which option the user presses. To do this you will need this code which codes for the user selecting the option they desire and them using their up and down arrow keys to go to the option. .. toctree:

```
while True:
    # get user input
    keys = ugame.buttons.get_pressed()

    if keys & ugame.K_UP != 0:
        if up_button == constants.button_state["button_up"]:
            up_button = constants.button_state["button_just_pressed"]
        elif up_button == constants.button_state["button_just_pressed"]:
            up_button = constants.button_state["button_still_pressed"]
    else:
        if up_button == constants.button_state["button_still_pressed"]:
            up_button = constants.button_state["button_released"]
        else:
            up_button = constants.button_state["button_up"]

    if keys & ugame.K_DOWN != 0:
        if down_button == constants.button_state["button_up"]:
            down_button = constants.button_state["button_just_pressed"]
        elif down_button == constants.button_state["button_just_pressed"]:
            down_button = constants.button_state["button_still_pressed"]
    else:
        if down_button == constants.button_state["button_still_pressed"]:
            down_button = constants.button_state["button_released"]
        else:
            down_button = constants.button_state["button_up"]
```

(continues on next page)

(continued from previous page)

```
    # update game logic

    #print(keys)
    if down_button == constants.button_state["button_just_pressed"] or up_button ==␣
↪constants.button_state["button_just_pressed"]:

        if option == 0:
            option = 1
            menu_text.clear()
            menu_text.cursor(0, 0)
            menu_text.move(constants.MENU_LOCATION_X, constants.MENU_LOCATION_Y)
            menu_text.text("<< MENU >>")
            game.render_block()
            retry_text.clear()
            retry_text.cursor(0, 0)
            retry_text.move(constants.RETRY_LOCATION_X, constants.RETRY_LOCATION_Y)
            retry_text.text("   RETRY   ")
            game.render_block()
        elif option == 1:
            option = 0
            retry_text.clear()
            retry_text.cursor(0, 0)
            retry_text.move(constants.RETRY_LOCATION_X, constants.RETRY_LOCATION_Y)
            retry_text.text("<< RETRY >>")
            game.render_block()
            menu_text.clear()
            menu_text.cursor(0, 0)
            menu_text.move(constants.MENU_LOCATION_X, constants.MENU_LOCATION_Y)
            menu_text.text("   MENU   ")
            game.render_block()

    if keys & ugame.K_X != 0 or keys & ugame.K_O != 0 or keys & ugame.K_START != 0 or␣
↪keys & ugame.K_SELECT != 0:  # A, B, start or select
        if option == 0:
            sound.play(coin_sound)
            # This is so they can hear the full sound
            time.sleep(1.0)
            game_scene(game_mode)
        elif option == 1:
            sound.play(coin_sound)
            # This is so they can hear the full sound
            time.sleep(1.0)
            main_menu_scene()
```

After you input all this code you must add the render text option and add in all variables you need. After all this is done your code should look like this: .. toctree:

```
def game_over_scene(final_score, final_height):
# this function is the game over scene
option = 0
# This is so it can get into the game scene. Only one option as you cant get here on␣
↪endless
game_mode = 0
# an image bank for CircuitPython
image_bank_3 = stage.Bank.from_bmp16("jungle_joe.bmp")
# sets the background to image 0 in the bank
```

(continues on next page)

(continued from previous page)

```
background = stage.Grid(image_bank_3, constants.SCREEN_GRID_X, constants.SCREEN_GRID_
↪Y)

coin_sound = open("coin.wav", 'rb')
sound = ugame.audio
sound.stop()
sound.mute(False)

text = []

text0 = stage.Text(width=29, height=14, font=None, palette=constants.SCORE_PALETTE,␣
↪buffer=None)
text0.move(22, 20)
text0.text("Final Score: {:0>2d}".format(final_score))
text.append(text0)

text2 = stage.Text(width=29, height=14, font=None, palette=constants.SCORE_PALETTE,␣
↪buffer=None)
text2.move(37, 30)
text2.text("Height: {:0>2d}ft".format(final_height))
text.append(text2)

text1 = stage.Text(width=29, height=14, font=None, palette=constants.MT_GAME_STUDIO_
↪PALETTE, buffer=None)
text1.move(43, 60)
text1.text("GAME OVER")
text.append(text1)

menu_text = stage.Text(width=29, height=14, font=None, palette=constants.SCORE_
↪PALETTE, buffer=None)
menu_text.clear()
menu_text.cursor(0, 0)
menu_text.move(constants.MENU_LOCATION_X, constants.MENU_LOCATION_Y)
menu_text.text("   MENU   ")
text.append(menu_text)

retry_text = stage.Text(width=29, height=14, font=None, palette=constants.SCORE_
↪PALETTE, buffer=None)
retry_text.clear()
retry_text.cursor(0, 0)
retry_text.move(constants.RETRY_LOCATION_X, constants.RETRY_LOCATION_Y)
retry_text.text("<< RETRY >>")
text.append(retry_text)

up_button = constants.button_state["button_up"]
down_button = constants.button_state["button_up"]

# create a stage for the background to show up on
#   and set the frame rate to 60fps
game = stage.Stage(ugame.display, 60)
# set the layers, items show up in order
game.layers = text + [background]
# render the background and inital location of sprite list
# most likely you will only render background once per scene
game.render_block()

# repeat forever, game loop
```

(continues on next page)

(continued from previous page)

```
while True:
    # get user input
    keys = ugame.buttons.get_pressed()

    if keys & ugame.K_UP != 0:
        if up_button == constants.button_state["button_up"]:
            up_button = constants.button_state["button_just_pressed"]
        elif up_button == constants.button_state["button_just_pressed"]:
            up_button = constants.button_state["button_still_pressed"]
    else:
        if up_button == constants.button_state["button_still_pressed"]:
            up_button = constants.button_state["button_released"]
        else:
            up_button = constants.button_state["button_up"]

    if keys & ugame.K_DOWN != 0:
        if down_button == constants.button_state["button_up"]:
            down_button = constants.button_state["button_just_pressed"]
        elif down_button == constants.button_state["button_just_pressed"]:
            down_button = constants.button_state["button_still_pressed"]
    else:
        if down_button == constants.button_state["button_still_pressed"]:
            down_button = constants.button_state["button_released"]
        else:
            down_button = constants.button_state["button_up"]

    # update game logic

    #print(keys)
    if down_button == constants.button_state["button_just_pressed"] or up_button ==␣
↪constants.button_state["button_just_pressed"]:

        if option == 0:
            option = 1
            menu_text.clear()
            menu_text.cursor(0, 0)
            menu_text.move(constants.MENU_LOCATION_X, constants.MENU_LOCATION_Y)
            menu_text.text("<< MENU >>")
            game.render_block()
            retry_text.clear()
            retry_text.cursor(0, 0)
            retry_text.move(constants.RETRY_LOCATION_X, constants.RETRY_LOCATION_Y)
            retry_text.text("   RETRY   ")
            game.render_block()
        elif option == 1:
            option = 0
            retry_text.clear()
            retry_text.cursor(0, 0)
            retry_text.move(constants.RETRY_LOCATION_X, constants.RETRY_LOCATION_Y)
            retry_text.text("<< RETRY >>")
            game.render_block()
            menu_text.clear()
            menu_text.cursor(0, 0)
            menu_text.move(constants.MENU_LOCATION_X, constants.MENU_LOCATION_Y)
            menu_text.text("   MENU   ")
            game.render_block()
```

(continues on next page)

(continued from previous page)

```
    if keys & ugame.K_X != 0 or keys & ugame.K_O != 0 or keys & ugame.K_START != 0 or␣
↪keys & ugame.K_SELECT != 0:  # A, B, start or select
        if option == 0:
            sound.play(coin_sound)
            # This is so they can hear the full sound
            time.sleep(1.0)
            game_scene(game_mode)
        elif option == 1:
            sound.play(coin_sound)
            # This is so they can hear the full sound
            time.sleep(1.0)
            main_menu_scene()
```