Arcade Documentation

Release 1.0.0

Paul Vincent Craven

Aug 02, 2017

Contents

CHAPTER 1

Introduction

- My goal with this webinar is to share what I've learned teaching students to program using Python to create video games, along with a new Arcade library for 2D graphics.
- So if you are an educator, or if you work with interns and new employees, hopefully you'll find some ideas to try.
- If you are more of a visual learner, or if you want to follow up on anything I'm talking about later, you can see my notes for this webinar at: http://2017-craven-webinar.readthedocs.io

CHAPTER 2

Background/Overview

- I'm a computer science professor at Simpson College in Iowa.
- I've taught an *Intro to Programming* class using Python and Pygame for many years.
  - I created Program Arcade Games, a website, book, and video series to learn programming.
  - Students learn by creating their own video games.
  - About 1,800 people use it to learn each weekday during the school year.
- Pygame got to be a limiting factor to make the curriculum better.
  - Buggy, inconsistent, didn't use new Python 3 features.
  - Students spent too much time learning the 'oddities' of Pygame when they could have been learning more programming.
- Create the Arcade Library
  - Easy library for 2D video games
  - Fix issues with Pygame
  - Faster, OpenGL based
  - Take advantage of Python 3 features like type hinting.
- I have a new version of my on-line book that I'm working on that uses the Arcade Library instead of Pygame: Learn Python with Arcade Academy

CHAPTER 3

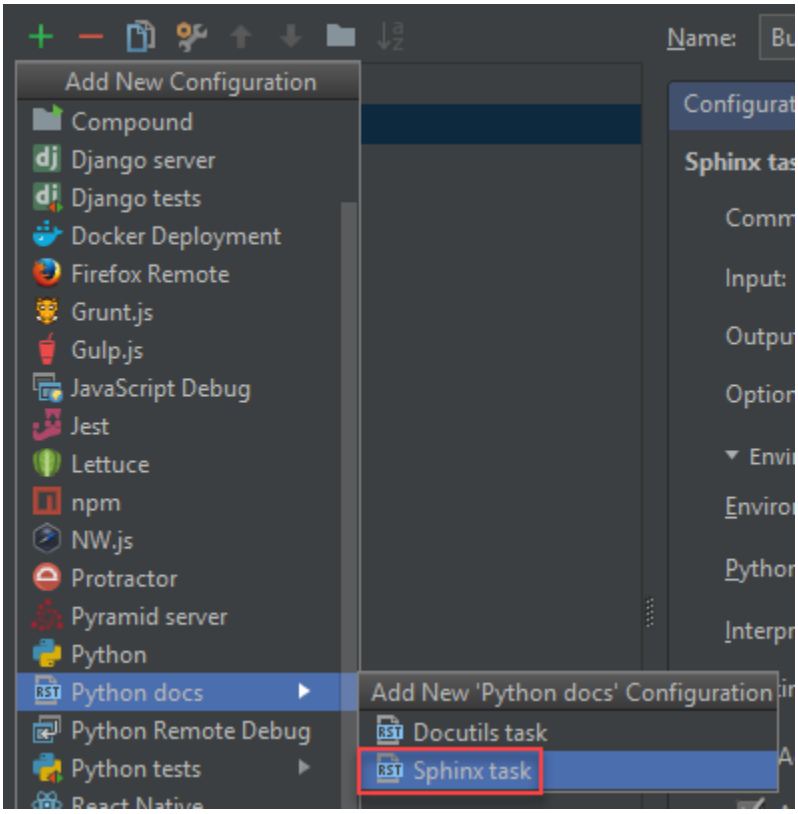
Tip 1: Revise Your Pedagogy

- Pedagogy (how you teach) should be treated like software.
- We keep releasing "learn to program" books, as one-and-done books. This is terrible.
- This applies even if you aren't writing a "how to program" book. For example, if you are writing docs on your in-house software, your docs on your company's development procedures.
- Most books probably haven't even had one set of people go through them to learn how to program before being released to market.
- Realize the first release of learning material, just like software, will be terrible.
- Revise your notes. Teach from the materials, find student questions, update it, repeat.
- Even the second version will likely be bad. When I create materials, it isn't until at least the third version I start being happy with the materials.
- Don't restart from scratch.
- After 8 years of revisions, 16 semesters of teaching the class, you can fine tune a class to a thing of beauty. For example, imagine updating a question where you add the text "If you think the answer is zero, go back and reread Section 12.3." can save you time, the student time, and you end up with a student who understands the material better than last semester's student. It is engineering.
- Remember: Any documentation you use watch people use it, take notes on how to make it better, and then engineer it. Make science out of learning.

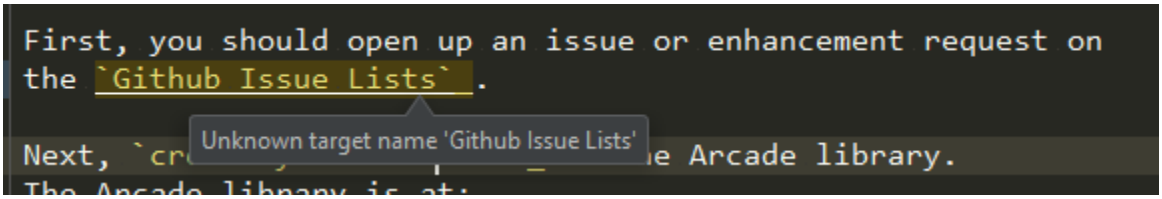
CHAPTER 4

Tip 2: Use a Good Tool-Chain for Teaching Materials

- How do you make this constant revision easy?
- I suggest using Sphinx. Sphinx is the software used to create the official Python documentation. It is a "Static Content Generator."
  - Note, there are a lot of other Static Content Generators out there you can use for your materials. I've also tried using a Wiki. I prefer Sphinx for its Python integration.
- Sphinx uses Restructured Text, which is much better than editing HTML by hand. You can also make e-books out of it.
- See this example of a Networking class I did: https://github.com/pvcraven/networking-class/commits/master
- You can easily build `.rst` files using Sphinx in PyCharm with the Sphinx task:



- PyCharm is a great editor to type up your restructured text files, as it can spell check, auto-complete rst directives, and even check and let you know if you have undefined references. You can look up references.



- It is easy to include syntax-highlighted code directly in your materials. You can include code snippets, or have the code in separate files.

```
print("Simple code snippet")
```

Listing 4.1: Including code from a different file

```
.. literalinclude:: ../../examples/bouncing_ball.py
    :caption: bouncing_ball.py
    :linenos:
```

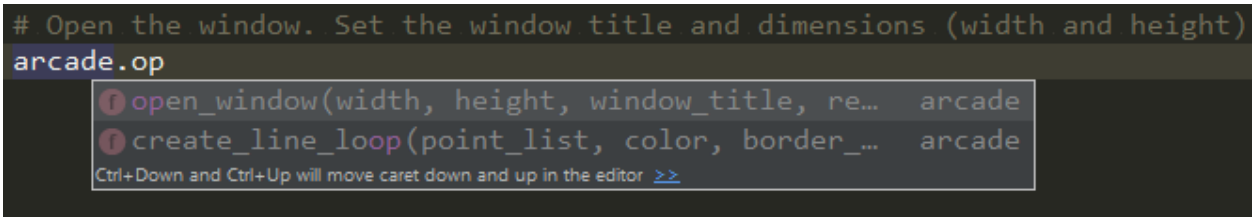
- Store revisions to your materials using Git and a site like GitHub. (Note, BitBucket offers free private repositories if you don't want to share the source to your materials.)
- Use Read the Docs to automatically build and host your materials, or use cloud hosting.
- The end result:
- Use any computer to update your teaching materials
- Work with other developers in your team to keep docs up to date using your same in-house VCS.

- Push the changes and have them automatically built
- Even use bug/feature tracking to keep track of to-do lists
- If you teach people over-and-over, you'll end up with high-quality materials.
- Web Development
- Networking
- Intro to Programming
- 3D Graphics with Blender

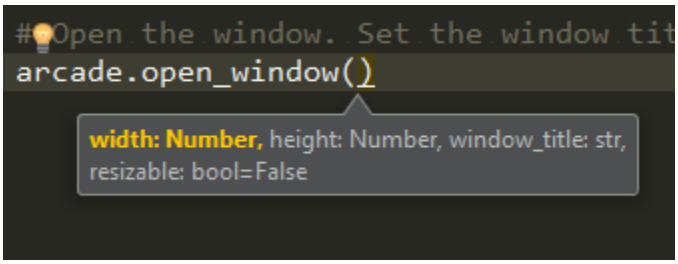
CHAPTER 5

Tip 3: Parameter and Type Hinting

The beauty of modern editors, is that for most students, you won't even have to point out that this exists. As soon as you type `arcade.` when using an IDE like PyCharm, you get a pop-up with the Arcade Library functions. This helps the students quite a bit.



Once you are start typing the function parameters, you also get a pop-up with the parameters and types:

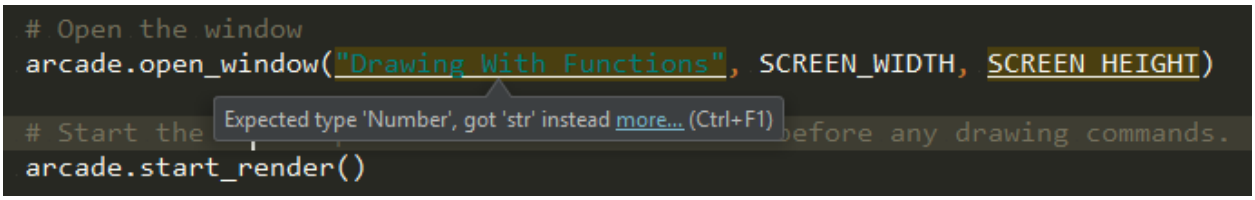


Note: Parameter Names are Important

This means the parameter names have to be good. What might make sense to the library programmer does not always make sense to the new programmer trying to figure use the library.

I've spent time time tuning the parameter names for the Arcade Library based on my in-person observations of what names seem to work best with students figuring out the library.

If you type the parameters incorrectly, you'll get an warning before you get any further:



The Arcade Library specifies parameter types using the new type hinting conventions introduced in Python 3.5 via PEP 484 and PEP 526.

The numbers and typing modules in Python help support type hinting. The code to implement type hinting in the Arcade Library looks like this:

```
def open_window(width: Number, height: Number, window_title: str, resizable: bool =␣
↪False):
```

In short, I've found these PyCharm features to be very effective with new programmers:

- Pop-up lists of functions.
- When typing a function, having the IDE pop up a list of good parameter variable names. IDE choice aside, it is important that the parameter names proven understandable as shown by actual experience observing programmers new to the library.
- Type hinting. Get some of the advantages of strong typing without the hassle of requiring new programmers to declare variables before using them. (If you've forgotten the difference between strong, weak, dynamic, and static typing, I suggest a quick review.)

Note: I hope someday we will also see docs for functions as you type them. Or a link to the HTML docs.

Students often treat the type hinting warnings as errors rather than warnings. This helps a lot with new programmers because they figure out their errors as they do the initial coding, rather than trying to figure out why a 500 line program doesn't work after they've coded it.

CHAPTER 6

Tip 4: Show Students the Linter

Stereotypically, programmers are shown with terrible clothing style choices. In reality, we are all about style. Style of our code. For Python, this is all defined in PEP-8.

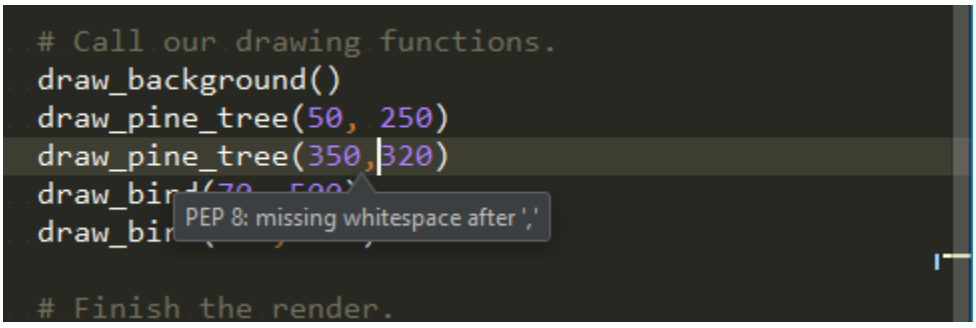
This might catch new programmers off-guard. A person might celebrate that feeling of accomplishment when their algorithm works. But get discouraged with the teacher nit-picks their program's style. New programmers often have issues with:

- Inconsistent indentation
- Forgetting to put spaces after commas
- Huge sections of blank lines in the middle of their program
- No blank lines in their program

In my experience:

- Feedback after-the-fact on style issues is mostly ignored, and if it isn't ignored it just serves to discourage students.
- Asking students to run a linter before turning in their program is just an extra step that they often skip.

I like Python because it is not so rigid as to discourage new programmers. But I do want to encourage new programmers to learn and adhere to programming standards. With pop-up warnings in the code, we get the best of both worlds. Here is what PyCharm does:



Since it comes from the editor, it seems more "official" than coming from the instructor. It also feels less like a personal insult to their intelligence. Poor students who don't care about programming can still ignore it. Average to good students can pay attention and write code that conforms to style guidelines.

In my experience, once students got PEP-8 warnings in their editor, the percent of student code that adhered to style guidelines improved.

Listing 6.1: bouncing_ball.py

```
"""
Tip 3:
    Function completion
    PEP-8
    Spell checking

Tip 4:
    Use type hinting
"""

# Function completion when you type-to
# Pull up documentation with Ctrl-Q
# If you try setting the background color with only 2 numbers, you'll get an type hint warning
# (Show how to open declaration, and how type hinting is done)
# Forget to put a space after a comma, and get a PEP-8 warning
# This is a program a student wrote his second week of class
# He wrote it and, like many students, also made sure PyCharm showed no issues.
# See that you don't have to declare or create classes to do stuff in arcade!

import arcade

arcade.open_window(800, 600, "Flying V")

arcade.set_background_color((53, 242, 252))

arcade.start_render()

"""Guitar Body"""
arcade.draw_triangle_filled(450, 300, 150, 300, 20, 100, arcade.color.CADMIUM_RED)
arcade.draw_triangle_filled(450, 300, 150, 300, 20, 500, arcade.color.CADMIUM_RED)

"""Guitar Neck"""
arcade.draw_rectangle_filled(450, 300, 350, 50, arcade.color.DEEP_CHAMPAGNE)

"""Guitar Head"""
arcade.draw_triangle_filled(625, 260, 625, 340, 751, 300, arcade.color.BLACK)

"""Guitar Pickups"""
arcade.draw_rectangle_filled(270, 300, 50, 80, arcade.color.SILVER)

"""Fret Separators"""
arcade.draw_line(625, 325, 625, 275, arcade.color.BROWN, border_width=2)
arcade.draw_line(604, 325, 604, 275, arcade.color.BROWN, border_width=2)
arcade.draw_line(583, 325, 583, 275, arcade.color.BROWN, border_width=2)
arcade.draw_line(562, 325, 562, 275, arcade.color.BROWN, border_width=2)
arcade.draw_line(541, 325, 541, 275, arcade.color.BROWN, border_width=2)
arcade.draw_line(520, 325, 520, 275, arcade.color.BROWN, border_width=2)
arcade.draw_line(499, 325, 499, 275, arcade.color.BROWN, border_width=2)
arcade.draw_line(478, 325, 478, 275, arcade.color.BROWN, border_width=2)
```

```
arcade.draw_line(457, 325, 457, 275, arcade.color.BROWN, border_width=2)
arcade.draw_line(436, 325, 436, 275, arcade.color.BROWN, border_width=2)
arcade.draw_line(415, 325, 415, 275, arcade.color.BROWN, border_width=2)
arcade.draw_line(394, 325, 394, 275, arcade.color.BROWN, border_width=2)
arcade.draw_line(373, 325, 373, 275, arcade.color.BROWN, border_width=2)
arcade.draw_line(352, 325, 352, 275, arcade.color.BROWN, border_width=2)
arcade.draw_line(331, 325, 331, 275, arcade.color.BROWN, border_width=2)
arcade.draw_line(310, 325, 310, 275, arcade.color.BROWN, border_width=2)

"""Fret Indicators"""
arcade.draw_circle_filled(572.5, 300, 5, arcade.color.LIGHT_BROWN)
arcade.draw_circle_filled(530.5, 300, 5, arcade.color.LIGHT_BROWN)
arcade.draw_circle_filled(383.5, 312, 5, arcade.color.LIGHT_BROWN)
arcade.draw_circle_filled(383.5, 288, 5, arcade.color.LIGHT_BROWN)

"""Strings"""
arcade.draw_line(260, 280, 640, 280, arcade.color.GRAY, border_width=1)
arcade.draw_line(260, 288, 670, 288, arcade.color.GRAY, border_width=1)
arcade.draw_line(260, 296, 700, 296, arcade.color.GRAY, border_width=1)
arcade.draw_line(260, 304, 700, 304, arcade.color.GRAY, border_width=1)
arcade.draw_line(260, 312, 670, 312, arcade.color.GRAY, border_width=1)
arcade.draw_line(260, 320, 640, 320, arcade.color.GRAY, border_width=1)

"""String Exits"""
arcade.draw_circle_filled(640, 280, 2, arcade.color.SILVER)
arcade.draw_circle_filled(670, 288, 2, arcade.color.SILVER)
arcade.draw_circle_filled(700, 296, 2, arcade.color.SILVER)
arcade.draw_circle_filled(700, 304, 2, arcade.color.SILVER)
arcade.draw_circle_filled(670, 312, 2, arcade.color.SILVER)
arcade.draw_circle_filled(640, 320, 2, arcade.color.SILVER)

"""String Entrances"""
arcade.draw_circle_filled(260, 280, 2, arcade.color.BLACK)
arcade.draw_circle_filled(260, 288, 2, arcade.color.BLACK)
arcade.draw_circle_filled(260, 296, 2, arcade.color.BLACK)
arcade.draw_circle_filled(260, 304, 2, arcade.color.BLACK)
arcade.draw_circle_filled(260, 312, 2, arcade.color.BLACK)
arcade.draw_circle_filled(260, 320, 2, arcade.color.BLACK)

"""Volume and Tone Knobs"""
arcade.draw_circle_filled(160, 190, 10, arcade.color.BLACK)
arcade.draw_circle_filled(210, 215, 10, arcade.color.BLACK)
arcade.draw_circle_filled(260, 240, 10, arcade.color.BLACK)
arcade.draw_circle_outline(160, 190, 11, arcade.color.ANTI_FLASH_WHITE, border_
↪width=2)
arcade.draw_circle_outline(210, 215, 11, arcade.color.ANTI_FLASH_WHITE, border_
↪width=2)
arcade.draw_circle_outline(260, 240, 11, arcade.color.ANTI_FLASH_WHITE, border_
↪width=2)

arcade.finish_render()

arcade.run()
```

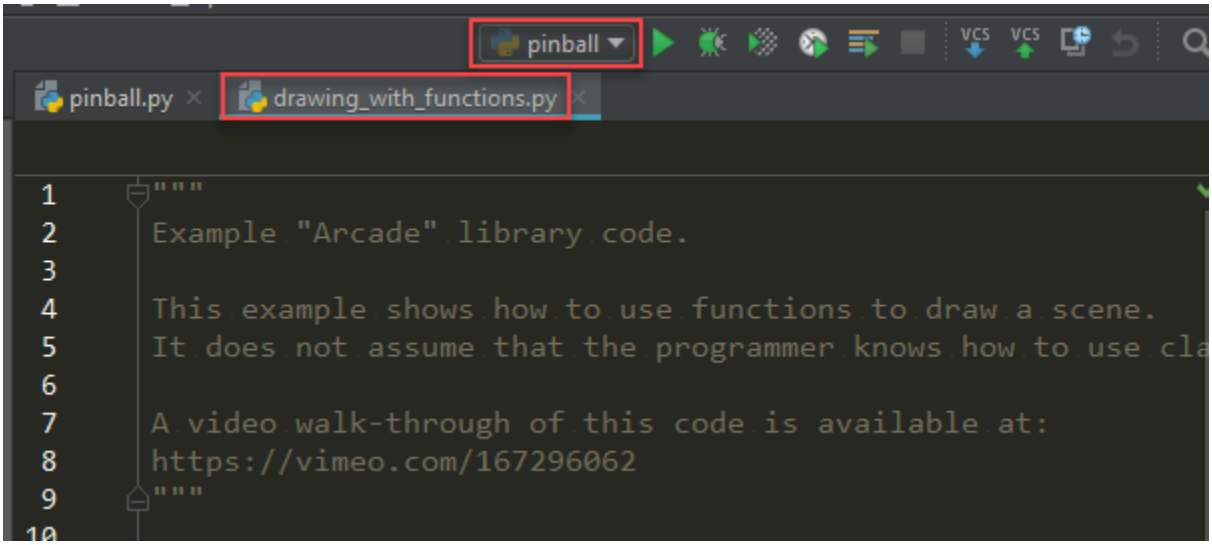
CHAPTER 7

Tip 5: Making IDEs Easier

Text editors are simple. Students write a file, and then run the file.

IDEs are more complex. The most common issues for new students:

- Students must create a project, then add the files to the project. Some students have difficulty with this concept, and even basic file/folder management. If a student just wants to run a two-line `for` loop example, forcing her to create a project to run it is crazy.
- Configurable IDEs have a lot of tool windows and other things that can be messed up, making it difficult for a student to get back to the state where they match the instructor's setup.
- Students often run the wrong program:



However, the trade-off for this complexity:

- Code-completion
- PEP-8 Linting

- Type hinting
- Spell checking
- If an error occurs, the IDE points right at the error without needing to decode a stack trace.

For many years I was in the "text editor" camp. This last year, after seeing the improved code quality with the PEP-8 linting, I'm in the IDE camp.

To mitigate the issues with forcing a project on the student, we create *one* project to use for the *whole* class. Students start with a pre-defined set of folders. There is one folder for each lab, and a folder for "scratch" code that students can store quick examples in.

CHAPTER 8

Tip 6: Teach Students to Search for Code Examples

Real-life programming rarely involves writing code from scratch. In fact, feedback we get from recent alumni is that they had not enough experience programming with large pre-existing project.

To help teach this, I provide a lot of code samples:

http://arcade.academy/examples/index.html

Don't just give students one example and step through it. That's a snooze-fest. Give lots of example code that form building blocks. Make it easy to navigate. Then give tasks that requires students to go through those code samples and build their own program.

CHAPTER 9

Tip 7: Show Students API Docs

Don't just show students how to program from your materials. Make it a point to show them the API docs. Have them look up functions out of the API docs and use them:

http://arcade.academy/quick_index.html

CHAPTER 10

Tip 8: Explain IDE Features As You Use Them

- Moving lines up and down
- Indent/unindent blocks of code
- Multi-cursor
- Go to definition

CHAPTER 11

Results

Playlist with some of the games:

https://www.youtube.com/playlist?list=PLUjR0nhln8ub1tPayFjz7w-LCTQ_gYs7V

Note that these aren't paint-by-number programs. There is a lot of creativity in what the students have put together.

Here is a breakout program that a student did:

The code mostly passes the linter check, and for a first-semester programming student is a rather impressive feat:

Listing 11.1: example.py

```
"""
Lab 12 - Final Project
Audio from http://opengameart.org
"""
import arcade
import random

SPRITE_SCALING = 0.5
SCREEN_WIDTH = 600
SCREEN_HEIGHT = 600
MOVEMENT_SPEED = 8
BALL_SPEED = 4

INSTRUCTION_PAGE = 0
GAME_RUNNING = 1
LEVEL_COMPLETE_PAGE = 2
GAME_OVER = 3
GAME_END = 4


def create_power_up(x, y, power_up_list, all_sprites_list):
    power_up = None
    power_up_type = random.randrange(3)
    if power_up_type == 0:
```

```
        # Image from Game Freezer: http://www.gamesfreezer.eu/2015_06_01_archive.html
        power_up = arcade.Sprite("images/extra_life.png", 0.4 * SPRITE_SCALING)
    if power_up_type == 1:
        # Image from Prek-8.com:
        power_up = arcade.Sprite("images/add_balls.png", 0.4 * SPRITE_SCALING)
    if power_up_type == 2:
        # image from ClipartFest: https://clipartfest.com/categories/view/
        power_up = arcade.Sprite("images/speed_up.png", 0.4 * SPRITE_SCALING)
    power_up.power_up_value = power_up_type
    power_up.change_y = -2
    power_up.center_x = x
    power_up.center_y = y
    power_up_list.append(power_up)
    all_sprites_list.append(power_up)


class Ball(arcade.Sprite):

    def __init__(self, all_sprites_list, power_up_list, wall_list,
                 brick_list, blue_brick_list, orange_brick_list, wall_brick_list,
                 player_sprite, application):
        # Image from Kenney.nl
        super().__init__("images/ball.png", 2 * SPRITE_SCALING)
        self.all_sprites_list = all_sprites_list
        self.wall_list = wall_list
        self.brick_list = brick_list
        self.blue_brick_list = blue_brick_list
        self.orange_brick_list = orange_brick_list
        self.wall_brick_list = wall_brick_list
        self.power_up_list = power_up_list
        self.player_sprite = player_sprite
        self.brick_sound = arcade.load_sound("sounds/brick_hit.wav")
        self.center_x = self.player_sprite.center_x
        self.center_y = 0.05 * SCREEN_HEIGHT + SCREEN_HEIGHT/15
        self.change_x = random.randrange(-3, 4)
        self.change_y = BALL_SPEED
        self.application = application

    def update(self):

        self.center_x += self.change_x

        orange_brick_hit_list = arcade.check_for_collision_with_list(self, self.
↪orange_brick_list)
        for brick in orange_brick_hit_list:
            # Image from Kenney.nl
            blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_SCALING)
            blue_brick.center_x = brick.center_x
            blue_brick.center_y = brick.center_y
            self.all_sprites_list.append(blue_brick)
            self.brick_list.append(blue_brick)
            self.blue_brick_list.append(blue_brick)
            brick.kill()
            arcade.play_sound(self.brick_sound)
            self.application.score += 1
            if self.change_x > 0:
                self.right = brick.left
            else:
```

```
                self.left = brick.right
        if len(orange_brick_hit_list) > 0:
            self.change_x *= -1

        blue_brick_hit_list = arcade.check_for_collision_with_list(self, self.blue_
↪brick_list)
        for brick in blue_brick_hit_list:
            brick.kill()
            arcade.play_sound(self.brick_sound)
            self.application.score += 1
            if random.randrange(10) == 0:
                create_power_up(self.center_x, self.center_y, self.power_up_list,␣
↪self.all_sprites_list)
            if self.change_x > 0:
                self.right = brick.left
            else:
                self.left = brick.right

        if len(blue_brick_hit_list) > 0:
            self.change_x *= -1

        wall_hit_list = arcade.check_for_collision_with_list(self, self.wall_list)
        for wall in wall_hit_list:
            if self.change_x > 0:
                self.right = wall.left
            else:
                self.left = wall.right
        if len(wall_hit_list) > 0:
            self.change_x *= -1

        wall_brick_hit_list = arcade.check_for_collision_with_list(self, self.wall_
↪brick_list)
        for wall_brick in wall_brick_hit_list:
            if self.change_x > 0:
                self.right = wall_brick.left
            else:
                self.left = wall_brick.right
        if len(wall_brick_hit_list) > 0:
            self.change_x *= -1

        self.center_y += self.change_y

        orange_brick_hit_list = arcade.check_for_collision_with_list(self, self.
↪orange_brick_list)
        for brick in orange_brick_hit_list:
            # Image from Kenney.nl
            blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_SCALING)
            blue_brick.center_x = brick.center_x
            blue_brick.center_y = brick.center_y
            self.all_sprites_list.append(blue_brick)
            self.brick_list.append(blue_brick)
            self.blue_brick_list.append(blue_brick)
            brick.kill()
            arcade.play_sound(self.brick_sound)
            if self.change_y > 0:
                self.top = brick.bottom
            else:
                self.bottom = brick.top
```

```
136             self.application.score += 1
137         if len(orange_brick_hit_list) > 0:
138             self.change_y *= -1
139
140         blue_brick_hit_list = arcade.check_for_collision_with_list(self, self.blue_
    ↪brick_list)
141         for brick in blue_brick_hit_list:
142             brick.kill()
143             arcade.play_sound(self.brick_sound)
144             if random.randrange(10) == 0:
145                 create_power_up(self.center_x, self.center_y, self.power_up_list,␣
    ↪self.all_sprites_list)
146             if self.change_y > 0:
147                 self.top = brick.bottom
148             else:
149                 self.bottom = brick.top
150             self.application.score += 1
151         if len(blue_brick_hit_list) > 0:
152             self.change_y *= -1
153
154         wall_hit_list = arcade.check_for_collision_with_list(self, self.wall_list)
155         for wall in wall_hit_list:
156             if self.change_y > 0:
157                 self.top = wall.bottom
158             else:
159                 self.bottom = wall.top
160         if len(wall_hit_list) > 0:
161             self.change_y *= -1
162
163         wall_brick_hit_list = arcade.check_for_collision_with_list(self, self.wall_
    ↪brick_list)
164         for wall_brick in wall_brick_hit_list:
165             if self.change_y > 0:
166                 self.top = wall_brick.bottom
167             else:
168                 self.bottom = wall_brick.top
169         if len(wall_brick_hit_list) > 0:
170             self.change_y *= -1
171
172     def get_score(self):
173         arcade.draw_text("Score: " + str(self.score), 40, 10, arcade.color.WHITE, 24)
174
175
176 class MyApplication(arcade.Window):
177     """ Main application class. """
178     def __init__(self, width, height):
179
180         super().__init__(width, height)
181         # Sprite lists
182         self.all_sprites_list = None
183
184         # Set up the player.
185
186         self.score = 0
187         self.player_sprite = None
188         self.player_sound = arcade.load_sound("sounds/player_hit.wav")
189         self.wall_list = None
190         self.brick_list = None
```

```
        self.wall_brick_list = None
        self.blue_brick_list = None
        self.orange_brick_list = None
        self.power_up_value = None
        self.power_up_list = None
        self.ball = None
        self.ball_list = None
        self.lives = 3
        self.next_level = False
        self.level = None
        self.current_state = INSTRUCTION_PAGE

    def setup(self):

        # Sprite lists
        self.score = 0
        self.all_sprites_list = arcade.SpriteList()
        self.wall_list = arcade.SpriteList()
        self.brick_list = arcade.SpriteList()
        self.blue_brick_list = arcade.SpriteList()
        self.orange_brick_list = arcade.SpriteList()
        self.wall_brick_list = arcade.SpriteList()
        self.power_up_list = arcade.SpriteList()
        self.ball_list = arcade.SpriteList()
        self.level = 0
        self.next_level = False
        # Image from Kenney.nl
        self.player_sprite = arcade.Sprite("images/player.png",
                                           0.25 * SPRITE_SCALING)
        self.player_sprite.center_x = 0.5 * SCREEN_WIDTH
        self.player_sprite.center_y = 0.075 * SCREEN_HEIGHT
        self.all_sprites_list.append(self.player_sprite)

        arcade.set_background_color(arcade.color.AMAZON)

        # -- Set up the boundary walls.
        for y in range(18, 600, 32):
            # Image from Kenney.nl
            wall = arcade.Sprite("images/wall.png", 0.5 * SPRITE_SCALING)
            wall.center_x = 18
            wall.center_y = y
            self.all_sprites_list.append(wall)
            self.wall_list.append(wall)

        for y in range(18, 600, 32):
            # Image from Kenney.nl
            wall = arcade.Sprite("images/wall.png", 0.5 * SPRITE_SCALING)
            wall.center_x = 582
            wall.center_y = y
            self.all_sprites_list.append(wall)
            self.wall_list.append(wall)

        for x in range(12, 600, 32):
            # Image from Kenney.nl
            wall = arcade.Sprite("images/wall.png", 0.5 * SPRITE_SCALING)
            wall.center_x = x
            wall.center_y = 596
            self.all_sprites_list.append(wall)
```

```
249             self.wall_list.append(wall)
250
251     def level_1(self):
252         self.next_level = False
253         ball = Ball(self.all_sprites_list, self.power_up_list,
254                     self.wall_list, self.brick_list, self.blue_brick_list,
255                     self.orange_brick_list, self.wall_brick_list, self.player_sprite,␣
↪self)
256         self.ball_list.append(ball)
257         self.all_sprites_list.append(ball)
258
259         for row in range(4):
260             for column in range(-(8 - 2 * row), 0):
261                 # Image from Kenney.nl
262                 blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_
↪SCALING)
263                 blue_brick.center_x = SCREEN_WIDTH / 10 * row + SCREEN_WIDTH / 2
264                 blue_brick.center_y = 30 * column + 590
265                 self.all_sprites_list.append(blue_brick)
266                 self.brick_list.append(blue_brick)
267                 self.blue_brick_list.append(blue_brick)
268
269         for column in range(4):
270             for row in range(-2 * column, 0):
271                 # Image from Kenney.nl
272                 blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_
↪SCALING)
273                 blue_brick.center_x = 60 * column + 60
274                 blue_brick.center_y = 30 * row + 590
275                 self.all_sprites_list.append(blue_brick)
276                 self.brick_list.append(blue_brick)
277                 self.blue_brick_list.append(blue_brick)
278
279     def level_2(self):
280         self.next_level = False
281         ball = Ball(self.all_sprites_list, self.power_up_list,
282                     self.wall_list, self.brick_list, self.blue_brick_list,
283                     self.orange_brick_list, self.wall_brick_list, self.player_sprite,␣
↪self)
284         self.ball_list.append(ball)
285         self.all_sprites_list.append(ball)
286
287         for row in range(4):
288             for column in range(0, (4 - row)):
289                 # Image from Kenney.nl
290                 blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_
↪SCALING)
291                 blue_brick.center_x = SCREEN_WIDTH / 10 * row + 70
292                 blue_brick.center_y = 30 * column + 300
293                 self.all_sprites_list.append(blue_brick)
294                 self.brick_list.append(blue_brick)
295                 self.blue_brick_list.append(blue_brick)
296
297         for row in range(4):
298             for column in range(-(4 - row), 0):
299                 # Image from Kenney.nl
300                 blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_
↪SCALING)
```

```
                blue_brick.center_x = SCREEN_WIDTH / 10 * row + 70
                blue_brick.center_y = 30 * column + 590
                self.all_sprites_list.append(blue_brick)
                self.brick_list.append(blue_brick)
                self.blue_brick_list.append(blue_brick)

        for column in range(5):
            for row in range(0, column):
                # Image from Kenney.nl
                blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_
↪SCALING)
                blue_brick.center_x = 60 * column + 290
                blue_brick.center_y = 30 * row + 300
                self.all_sprites_list.append(blue_brick)
                self.brick_list.append(blue_brick)
                self.blue_brick_list.append(blue_brick)

        for column in range(5):
            for row in range(-column, 0):
                # Image from Kenney.nl
                blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_
↪SCALING)
                blue_brick.center_x = 60 * column + 290
                blue_brick.center_y = 30 * row + 590
                self.all_sprites_list.append(blue_brick)
                self.brick_list.append(blue_brick)
                self.blue_brick_list.append(blue_brick)

        for column in range(5):
            # Image from Kenney.nl
            wall_brick = arcade.Sprite("images/black_brick.png", 0.6 * SPRITE_SCALING)
            wall_brick.center_x = 60 * column + 180
            wall_brick.center_y = 430
            self.all_sprites_list.append(wall_brick)
            self.wall_brick_list.append(wall_brick)

    def level_3(self):
        self.next_level = False
        ball = Ball(self.all_sprites_list, self.power_up_list,
                    self.wall_list, self.brick_list, self.blue_brick_list,
                    self.orange_brick_list, self.wall_brick_list, self.player_sprite,␣
↪self)
        self.ball_list.append(ball)
        self.all_sprites_list.append(ball)
        # Image from Kenney.nl
        orange_brick = arcade.Sprite("images/orange_brick.png", 0.6 * SPRITE_SCALING)
        orange_brick.center_x = SCREEN_WIDTH / 2
        orange_brick.center_y = 3 * SCREEN_HEIGHT / 4
        self.all_sprites_list.append(orange_brick)
        self.brick_list.append(orange_brick)
        self.orange_brick_list.append(orange_brick)

        for row in range(3):
            # Image from Kenney.nl
            blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_SCALING)
            blue_brick.center_x = 240
            blue_brick.center_y = 30 * row + 420
            self.all_sprites_list.append(blue_brick)
```

```
356             self.brick_list.append(blue_brick)
357             self.blue_brick_list.append(blue_brick)
358
359         for row in range(3):
360             # Image from Kenney.nl
361             blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_SCALING)
362             blue_brick.center_x = 360
363             blue_brick.center_y = 30 * row + 420
364             self.all_sprites_list.append(blue_brick)
365             self.brick_list.append(blue_brick)
366             self.blue_brick_list.append(blue_brick)
367         # Image from Kenney.nl
368         blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_SCALING)
369         blue_brick.center_x = 300
370         blue_brick.center_y = 420
371         self.all_sprites_list.append(blue_brick)
372         self.brick_list.append(blue_brick)
373         self.blue_brick_list.append(blue_brick)
374         # Image from Kenney.nl
375         blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_SCALING)
376         blue_brick.center_x = 300
377         blue_brick.center_y = 480
378         self.all_sprites_list.append(blue_brick)
379         self.brick_list.append(blue_brick)
380         self.blue_brick_list.append(blue_brick)
381
382         for row in range(5):
383             # Image from Kenney.nl
384             orange_brick = arcade.Sprite("images/orange_brick.png", 0.6 * SPRITE_
    ↪SCALING)
385             orange_brick.center_x = 180
386             orange_brick.center_y = 30 * row + 390
387             self.all_sprites_list.append(orange_brick)
388             self.brick_list.append(orange_brick)
389             self.orange_brick_list.append(orange_brick)
390
391         for row in range(5):
392             # Image from Kenney.nl
393             orange_brick = arcade.Sprite("images/orange_brick.png", 0.6 * SPRITE_
    ↪SCALING)
394             orange_brick.center_x = 420
395             orange_brick.center_y = 30 * row + 390
396             self.all_sprites_list.append(orange_brick)
397             self.brick_list.append(orange_brick)
398             self.orange_brick_list.append(orange_brick)
399
400         for column in range(3):
401             # Image from Kenney.nl
402             orange_brick = arcade.Sprite("images/orange_brick.png", 0.6 * SPRITE_
    ↪SCALING)
403             orange_brick.center_x = 60 * column + 240
404             orange_brick.center_y = 390
405             self.all_sprites_list.append(orange_brick)
406             self.brick_list.append(orange_brick)
407             self.orange_brick_list.append(orange_brick)
408
409         for column in range(3):
410             # Image from Kenney.nl
```

```
411             orange_brick = arcade.Sprite("images/orange_brick.png", 0.6 * SPRITE_
↪SCALING)
412             orange_brick.center_x = 60 * column + 240
413             orange_brick.center_y = 510
414             self.all_sprites_list.append(orange_brick)
415             self.brick_list.append(orange_brick)
416             self.orange_brick_list.append(orange_brick)
417
418         for row in range(7):
419             # Image from Kenney.nl
420             blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_SCALING)
421             blue_brick.center_x = 120
422             blue_brick.center_y = 30 * row + 360
423             self.all_sprites_list.append(blue_brick)
424             self.brick_list.append(blue_brick)
425             self.blue_brick_list.append(blue_brick)
426
427         for row in range(7):
428             # Image from Kenney.nl
429             blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_SCALING)
430             blue_brick.center_x = 480
431             blue_brick.center_y = 30 * row + 360
432             self.all_sprites_list.append(blue_brick)
433             self.brick_list.append(blue_brick)
434             self.blue_brick_list.append(blue_brick)
435
436         for column in range(5):
437             # Image from Kenney.nl
438             blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_SCALING)
439             blue_brick.center_x = 60 * column + 180
440             blue_brick.center_y = 360
441             self.all_sprites_list.append(blue_brick)
442             self.brick_list.append(blue_brick)
443             self.blue_brick_list.append(blue_brick)
444
445         for column in range(5):
446             # Image from Kenney.nl
447             blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_SCALING)
448             blue_brick.center_x = 60 * column + 180
449             blue_brick.center_y = 540
450             self.all_sprites_list.append(blue_brick)
451             self.brick_list.append(blue_brick)
452             self.blue_brick_list.append(blue_brick)
453
454     def level_4(self):
455         self.next_level = False
456         ball = Ball(self.all_sprites_list, self.power_up_list,
457                     self.wall_list, self.brick_list, self.blue_brick_list,
458                     self.orange_brick_list, self.wall_brick_list, self.player_sprite,␣
↪self)
459         self.ball_list.append(ball)
460         self.all_sprites_list.append(ball)
461
462         for row in range(5):
463             for column in range(8):
464                 # Image from Kenney.nl
465                 blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_
↪SCALING)
```

```
466                 blue_brick.center_x = SCREEN_WIDTH / 10 * column + 90
467                 blue_brick.center_y = 30 * row + 430
468                 self.all_sprites_list.append(blue_brick)
469                 self.brick_list.append(blue_brick)
470                 self.blue_brick_list.append(blue_brick)
471
472         for column in range(4):
473             # Image from Kenney.nl
474             wall_brick = arcade.Sprite("images/black_brick.png", 0.6 * SPRITE_SCALING)
475             wall_brick.center_x = 120 * column + 110
476             wall_brick.center_y = 220
477             wall_brick.change_x = -2
478             self.all_sprites_list.append(wall_brick)
479             self.wall_brick_list.append(wall_brick)
480
481         for column in range(4):
482             # Image from Kenney.nl
483             wall_brick = arcade.Sprite("images/black_brick.png", 0.6 * SPRITE_SCALING)
484             wall_brick.center_x = 120 * column + 130
485             wall_brick.center_y = 280
486             wall_brick.change_x = 2
487             self.all_sprites_list.append(wall_brick)
488             self.wall_brick_list.append(wall_brick)
489
490     def level_5(self):
491         self.next_level = False
492         ball = Ball(self.all_sprites_list,  self.power_up_list,
493                     self.wall_list, self.brick_list, self.blue_brick_list,
494                     self.orange_brick_list, self.wall_brick_list, self.player_sprite,␣
    ↪self)
495         self.ball_list.append(ball)
496         self.all_sprites_list.append(ball)
497         # Image from Kenney.nl
498         orange_brick = arcade.Sprite("images/orange_brick.png", 0.6 * SPRITE_SCALING)
499         orange_brick.center_x = 300
500         orange_brick.center_y = 350
501         self.all_sprites_list.append(orange_brick)
502         self.brick_list.append(orange_brick)
503         self.orange_brick_list.append(orange_brick)
504
505         for row in range(3):
506             # Image from Kenney.nl
507             blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_SCALING)
508             blue_brick.center_x = 240
509             blue_brick.center_y = 30 * row + 320
510             self.all_sprites_list.append(blue_brick)
511             self.brick_list.append(blue_brick)
512             self.blue_brick_list.append(blue_brick)
513
514         for row in range(3):
515             # Image from Kenney.nl
516             blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_SCALING)
517             blue_brick.center_x = 360
518             blue_brick.center_y = 30 * row + 320
519             self.all_sprites_list.append(blue_brick)
520             self.brick_list.append(blue_brick)
521             self.blue_brick_list.append(blue_brick)
522         # Image from Kenney.nl
```

```
523         blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_SCALING)
524         blue_brick.center_x = 300
525         blue_brick.center_y = 320
526         self.all_sprites_list.append(blue_brick)
527         self.brick_list.append(blue_brick)
528         self.blue_brick_list.append(blue_brick)
529         # Image from Kenney.nl
530         blue_brick = arcade.Sprite("images/blue_brick.png", 0.6 * SPRITE_SCALING)
531         blue_brick.center_x = 300
532         blue_brick.center_y = 380
533         self.all_sprites_list.append(blue_brick)
534         self.brick_list.append(blue_brick)
535         self.blue_brick_list.append(blue_brick)
536
537         for row in range(5):
538             # Image from Kenney.nl
539             orange_brick = arcade.Sprite("images/orange_brick.png", 0.6 * SPRITE_
    ↪SCALING)
540             orange_brick.center_x = 180
541             orange_brick.center_y = 30 * row + 290
542             self.all_sprites_list.append(orange_brick)
543             self.brick_list.append(orange_brick)
544             self.orange_brick_list.append(orange_brick)
545
546         for row in range(5):
547             # Image from Kenney.nl
548             orange_brick = arcade.Sprite("images/orange_brick.png", 0.6 * SPRITE_
    ↪SCALING)
549             orange_brick.center_x = 420
550             orange_brick.center_y = 30 * row + 290
551             self.all_sprites_list.append(orange_brick)
552             self.brick_list.append(orange_brick)
553             self.orange_brick_list.append(orange_brick)
554
555         for column in range(3):
556             # Image from Kenney.nl
557             orange_brick = arcade.Sprite("images/orange_brick.png", 0.6 * SPRITE_
    ↪SCALING)
558             orange_brick.center_x = 60 * column + 240
559             orange_brick.center_y = 290
560             self.all_sprites_list.append(orange_brick)
561             self.brick_list.append(orange_brick)
562             self.orange_brick_list.append(orange_brick)
563
564         for column in range(3):
565             # Image from Kenney.nl
566             orange_brick = arcade.Sprite("images/orange_brick.png", 0.6 * SPRITE_
    ↪SCALING)
567             orange_brick.center_x = 60 * column + 240
568             orange_brick.center_y = 410
569             self.all_sprites_list.append(orange_brick)
570             self.brick_list.append(orange_brick)
571             self.orange_brick_list.append(orange_brick)
572
573         for column in range(4):
574             # Image from Kenney.nl
575             wall_brick = arcade.Sprite("images/black_brick.png", 0.6 * SPRITE_SCALING)
576             wall_brick.center_x = 140 * column + 90
```

```
577             wall_brick.center_y = 220
578             wall_brick.change_x = 2
579             wall_brick.change_y = 0
580             self.all_sprites_list.append(wall_brick)
581             self.wall_brick_list.append(wall_brick)
582 
583         for column in range(4):
584             # Image from Kenney.nl
585             wall_brick = arcade.Sprite("images/black_brick.png", 0.6 * SPRITE_SCALING)
586             wall_brick.center_x = 140 * column + 90
587             wall_brick.center_y = 480
588             wall_brick.change_x = -2
589             wall_brick.change_y = 0
590             self.all_sprites_list.append(wall_brick)
591             self.wall_brick_list.append(wall_brick)
592         # Image from Kenney.nl
593         wall_brick = arcade.Sprite("images/black_brick.png", 0.6 * SPRITE_SCALING)
594         wall_brick.center_x = 90
595         wall_brick.center_y = 350
596         wall_brick.change_x = 0
597         wall_brick.change_y = -2
598         self.all_sprites_list.append(wall_brick)
599         self.wall_brick_list.append(wall_brick)
600         # Image from Kenney.nl
601         wall_brick = arcade.Sprite("images/black_brick.png", 0.6 * SPRITE_SCALING)
602         wall_brick.center_x = 510
603         wall_brick.center_y = 350
604         wall_brick.change_x = 0
605         wall_brick.change_y = 2
606         self.all_sprites_list.append(wall_brick)
607         self.wall_brick_list.append(wall_brick)
608 
609     def add_life(self):
610         self.lives += 1
611 
612     def add_balls(self):
613         for i in range(3):
614             ball = Ball(self.all_sprites_list, self.power_up_list,
615                         self.wall_list, self.brick_list, self.blue_brick_list,
616                         self.orange_brick_list, self.wall_brick_list, self.player_
↪sprite, self)
617             self.ball_list.append(ball)
618             self.all_sprites_list.append(ball)
619 
620     def speed_up(self):
621         for ball in self.ball_list:
622             ball.change_y += 1
623 
624     def draw_instructions_page(self):
625         arcade.draw_text("Welcome to Brick Breaker!", 50, 500, arcade.color.WHITE, 32)
626         arcade.draw_text("Press SPACE to Continue", 55, 350, arcade.color.WHITE, 32)
627         self.lives = 3
628 
629     def draw_game(self):
630         """
631         Draw all the sprites, along with the score.
632         """
633         # Draw all the sprites.
```

```
634         self.all_sprites_list.draw()
635         # Draw the score in the bottom left.
636         Ball.get_score(self)
637         arcade.draw_text("Level " + str(self.level), 455, 10, arcade.color.WHITE, 24)
638         arcade.draw_text(str(self.lives) + " Lives left", 230, 10, arcade.color.WHITE,
    ↪ 24)
639 
640     def draw_level_complete(self):
641         # Draw at the end of every level
642         arcade.draw_text("Level  " + str(self.level), 150, 400, arcade.color.WHITE,␣
    ↪64)
643         arcade.draw_text("Complete!", 100, 290, arcade.color.WHITE, 64)
644         arcade.draw_text("Press Space to Continue", 65, 210, arcade.color.WHITE, 32)
645 
646     def draw_game_over(self):
647         arcade.draw_text("Game Over", 80, 330, arcade.color.WHITE, 64)
648         arcade.draw_text("Press R to Restart", 80, 220, arcade.color.WHITE, 40)
649         arcade.draw_text("Press Q to Quit", 120, 140, arcade.color.WHITE, 40)
650 
651     def draw_game_end(self):
652         arcade.draw_text("Congratulations!", 50, 520, arcade.color.WHITE, 52)
653         arcade.draw_text("You Won!", 90, 420, arcade.color.WHITE, 72)
654         arcade.draw_text("Your Final Score was ", 40, 300, arcade.color.WHITE, 40)
655         arcade.draw_text(str(self.score), 280, 240, arcade.color.WHITE, 40)
656         arcade.draw_text("Press R to Restart", 80, 140, arcade.color.WHITE, 40)
657         arcade.draw_text("Press Q to Quit", 120, 70, arcade.color.WHITE, 40)
658 
659     def on_draw(self):
660 
661         arcade.start_render()
662         # Draw all the sprites.
663         self.all_sprites_list.draw()
664         if self.current_state == INSTRUCTION_PAGE:
665             self.draw_instructions_page()
666 
667         elif self.current_state == GAME_RUNNING:
668             self.draw_game()
669 
670         elif self.current_state == LEVEL_COMPLETE_PAGE:
671             self.draw_level_complete()
672 
673         elif self.current_state == GAME_END:
674             self.draw_game_end()
675 
676         else:
677             self.draw_game_over()
678         Ball.get_score(self)
679 
680         if len(self.brick_list) == 0 and self.next_level:
681             if self.level < 5:
682                 self.level += 1
683                 if self.level == 1:
684                     self.level_1()
685                 elif self.level == 2:
686                     self.level_2()
687                 elif self.level == 3:
688                     self.level_3()
689                 elif self.level == 4:
```

```
690                     self.level_4()
691                 elif self.level == 5:
692                     self.level_5()
693
694             else:
695                 self.draw_game_end()
696
697     def on_key_press(self, key, modifiers):
698         """Called whenever a key is pressed. """
699
700         if key == arcade.key.Q:
701             MyApplication.close(self)
702         elif key == arcade.key.SPACE and len(self.brick_list) == 0:
703             if not self.next_level:
704                 if self.level < 6:
705                     self.current_state = GAME_RUNNING
706                     self.draw_game()
707                     self.next_level = True
708                 else:
709                     self.current_state = GAME_END
710         elif key == arcade.key.R:
711             self.current_state = INSTRUCTION_PAGE
712             self.draw_instructions_page()
713             MyApplication.setup(self)
714         elif key == arcade.key.LEFT or key == arcade.key.A:
715             self.player_sprite.change_x = -MOVEMENT_SPEED
716         elif key == arcade.key.RIGHT or key == arcade.key.D:
717             self.player_sprite.change_x = MOVEMENT_SPEED
718
719     def on_key_release(self, key, modifiers):
720         if key == arcade.key.LEFT or key == arcade.key.A \
721                 or key == arcade.key.RIGHT or key == arcade.key.D:
722             self.player_sprite.change_x = 0
723
724     def animate(self, delta_time):
725
726         for ball in self.ball_list:
727             if ball.center_y < 0:
728                 ball.kill()
729                 if self.lives > 0 and len(self.ball_list) == 0:
730                     self.lives -= 1
731                     ball = Ball(self.all_sprites_list, self.power_up_list,
732                                 self.wall_list, self.brick_list, self.blue_brick_list,
733                                 self.orange_brick_list, self.wall_brick_list, self.
↪player_sprite, self)
734                     self.ball_list.append(ball)
735                     self.all_sprites_list.append(ball)
736                 elif self.lives <= 0:
737                     for power_up in self.power_up_list:
738                         power_up.kill()
739                     self.draw_game_over()
740                     self.current_state = GAME_OVER
741
742         for power_up in self.power_up_list:
743             if power_up.top <= 0:
744                 power_up.kill()
745
746         if len(self.brick_list) == 0 and self.level > 0:
```

```
747             if not self.next_level:
748                 for wall_brick in self.wall_brick_list:
749                     wall_brick.kill()
750                 for ball in self.ball_list:
751                     ball.kill()
752                 for power_up in self.power_up_list:
753                     power_up.kill()
754                 self.current_state = LEVEL_COMPLETE_PAGE
755                 self.draw_level_complete()
756 
757         wall_player_hit_list = arcade.check_for_collision_with_list(self.player_
    ↪sprite, self.wall_list)
758         for wall in wall_player_hit_list:
759             if self.player_sprite.right >= wall.left and self.player_sprite.center_x >
    ↪ SCREEN_WIDTH / 2:
760                 self.player_sprite.right = wall.left
761 
762             elif self.player_sprite.left <= wall.right:
763                 self.player_sprite.left = wall.right
764 
765         for wall_brick in self.wall_brick_list:
766             if self.current_state != GAME_RUNNING:
767                 wall_brick.change_x = 0
768                 wall_brick.change_y = 0
769             elif self.level == 4:
770                 if wall_brick.left < 70:
771                     for brick in self.wall_brick_list:
772                         brick.change_x *= -1
773                 elif wall_brick.right > 550:
774                     for brick in self.wall_brick_list:
775                         brick.change_x *= -1
776             elif self.level == 5:
777                 if wall_brick.center_x < 90 and wall_brick.change_x < 0:
778                     wall_brick.change_x = 0
779                     wall_brick.change_y = -2
780                 elif wall_brick.center_y < 220 and wall_brick.change_y < 0:
781                     wall_brick.change_x = 2
782                     wall_brick.change_y = 0
783                 elif wall_brick.center_x > 510 and wall_brick.change_x > 0:
784                     wall_brick.change_x = 0
785                     wall_brick.change_y = 2
786                 elif wall_brick.center_y > 480 and wall_brick.change_y > 0:
787                     wall_brick.change_x = -2
788                     wall_brick.change_y = 0
789 
790         ball_hit_list = arcade.check_for_collision_with_list(self.player_sprite, self.
    ↪ball_list)
791         for ball in ball_hit_list:
792             arcade.play_sound(self.player_sound)
793             if ball.change_y < 0:
794                 ball.change_y *= -1
795             if ball.center_x < self.player_sprite.center_x:
796                 ball.change_x = random.randrange(-5, -2)
797             elif ball.center_x > self.player_sprite.center_x:
798                 ball.change_x = random.randrange(3, 6)
799             else:
800                 ball.change_x = 0
801 
```

```
802         power_up_hit_list = arcade.check_for_collision_with_list(self.player_sprite,␣
    ↪self.power_up_list)
803         for power_up in power_up_hit_list:
804             power_up.kill()
805             if power_up.power_up_value == 0:
806                 self.add_life()
807
808             elif power_up.power_up_value == 1:
809                 self.add_balls()
810
811             elif power_up.power_up_value == 2:
812                 self.speed_up()
813
814         self.all_sprites_list.update()
815         self.ball_list.update()
816         self.player_sprite.update()
817         self.wall_brick_list.update()
818
819 window = MyApplication(SCREEN_WIDTH, SCREEN_HEIGHT)
820 window.setup()
821
822 arcade.run()
```