Assignment 1 report - INF-1400

Sera Madeleine Elstad

February 17, 2022

1 Introduction

The first assignment in INF-1400 is to create a variation of the classic game breakout. The game will be implemented using object-oriented programming principles, which means it will make use of classes and methods. Python and the Pygame library will be used to create the game. A player-controlled platform, a wall with removable bricks, and a bouncing ball are created for the game. The game's objective is to remove all the bricks from the wall. This is accomplished by having the ball strike the bricks, after which each brick is removed. The player wins if all the bricks are removed and lose if the ball hits the bottom of the screen, and there are no lives left.

2 Technical Background

2.1 Pygame

Pygame is a set of Python modules that can be used to create video games. Pygame adds functionality to the SDL library. This allows the programmer to create full-featured games and multimedia programs using Python. To be able to use Pygame in Python, the Pygame library must be imported. This is done by writing *import Pygame* in the beginning. [1]

2.2 Class and objects

Almost everything in Python is an object. Objects are nothing more than a grouping of variables and Python functions. An object is defined by a class. A Python class is a user-defined blueprint for an object that defines the various attributes of the object. The class includes methods for changing the state of an object. They also specify the characteristics that an object may have. A Python class variable is shared by all object instances of a class. Variables are defined when a class is created. They are not defined in any of the methods of a class. Variables and functions are defined within the class and accessed via objects. [2]

2.3 Sprites

Sprite is a module that includes some higher-level classes to manage any objects on the screen that can move around. While playing a 2D game, all the objects that are displayed on the screen are sprites. Sprites objects can be animated, controlled by the player, they can interact with each other, and they can be added to Sprite groups.

A Sprite group is a container class that holds and manages multiple Sprite objects at once. By using a Sprite group, the programmer can easily add, remove, draw, and update one or more sprites with a single command. This is beneficial because it shortens the code and thus makes it easier to read.

The built-in sprite function makes it simple to find sprites that intersect with one another. The sprite-collide function, for example, finds all the sprites in a group that intersects with each other and returns a list containing all these sprites. This can be done for the bounding box of each sprite, but it can also be done for an image that does not have a rectangular shape, for example, a ball. To get this shape you can use the built-in sprite function "mask".

The mask function creates a mask object from the given surface by setting all the opaque pixels and not setting the transparent pixels (https://www.pygame.org/docs/ref/mask.html). By doing this the program can better detect collisions between not rectangular shapes. The downside with using this is that the program needs more time to run the code, but for small programs like this, with few collisions this function is good. [3]

2.4 Vector 2

Vector 2 is a module for two-dimensional vectors that support several numerical operations. Using vectors is beneficial to have control of both the x and y value of a moving object. [4]

3 Design and implementation

3.1 Implementation of the objects

3.1.1 The ball

The ball object is found in the ball.py file and is composed of a class called Ball, which includes the functions update and bounce wall. To draw the ball, an image is imported and scaled to the appropriate size. Self.rect is used to get the ball's position, and self.rect.center is a vector that is set to be the ball's center at the start of the game. Within the object, a mask is defined, which is important for collision detection.

Update is a ball object function that updates the position of the ball based on the given ball speed. The detection with the screen edges is also detected within the update. If one of the screen edges is detected, the x or y speed value is multiplied by -1 to cause it to turn rather than disappear outside the screen.

Bounce wall is another function found within the ball class. When a collision with a wall block is detected, this function is called. When this happens, the speed in both the x and y directions is multiplied by -1, causing it to change direction.

3.1.2 Wall

Inside the wall.py file, a class is created for the wall. The class only has one function that holds all the attributes needed for the program to draw the rectangles, which is the image, the x and y positions of each rectangle, and the rectangle height and width. The function for drawing the wall is placed outside the rectangle class. It's here because if it were inside the class, all the rectangles would be drawn with the same coordinates.

To draw the wall, first, a spirit group that contains all the rectangles is made. Then a for-loop that contains another for-loop is created. The first for-loop iterates through all the rows, while the second iterates through all the columns. All rectangles are drawn and added to the sprite group in the second for-loop.

To make the wall appear on the screen, the rectangle group is added to the group that contains all of the other sprites, which can be found in the main file. The sprite draw function is then used to draw the wall and other objects on the screen.

3.1.3 Paddle

The paddle, like all other objects, is created using a sprite class, and the code can be found in paddle.py. To draw the paddle some build-in functions are used, and the rect function is used to keep track of paddle position, which is needed to calculate the angle of the ball when it hits the platform. The update function checks whether the right or the left key has been pressed and moves the paddle in the correct direction at a given speed during the update.

3.2 Implementation of collision between ball and paddle

The sprite function collide is used to detect a collision between the ball and the paddle. To control the ball, the paddle was divided into two sections, left and right (see figure 1). If the ball comes from the right side and collides with the right side of the platform, it should change direction (see the green lines). If the ball, on the other hand, collides with the left side, it should bounce back in the same direction it came from (see the blue lines). If the ball collides with the center of the paddle, it will bounce up 90 degrees from the platform. The closer to the center the ball collide, the greater is the angle the ball is returned with. The only difference if the ball comes from the left side is that everything will be reflected.

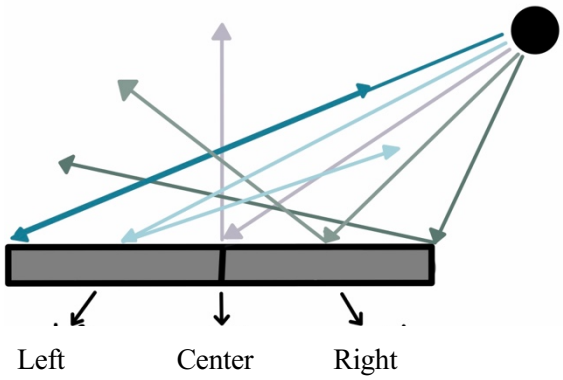

Figure 1

4 Evaluation and discussion

The implementation of the game has been tested by playing it with different variables to see how it behaves. With this implementation, all requirements are fulfilled, but there are some minor issues. First, if the ball collides with one of the paddle sides it will get stuck inside the paddle. This could probably be fixed by changing the variables if the ball collides with the sides.

Second, if the ball collides with two blocks, both will vanish and the ball will not change direction. This is most likely due to the fact that the y-speed is multiplied by -1 each time a block is removed. When two blocks are removed simultaneously, the y-speed is multiplied by $(-1)^2$, which equals 1, and the ball continues to remove blocks.

5 Conclusion

The task for this assignment was to create a variation of the classic game *breakout* using the principles of object-oriented programming. A player-controlled platform, a wall with removable bricks, and a bouncing ball are all objects in this game, and they all work as expected. Some things could have been done differently for this implementation, and the program could have been even better as a result, but this solution meets the requirements.

6 References

[1] Pygame, "Pygame," [Online]. Available: https://www.pygame.org/wiki/about. [Accessed February 2022].

[2] Geeksforgeeks, "geeksforgeeks," 10 June 2021. [Online]. Available: https://www.geeksforgeeks.org/python-classes-and-objects/. [Accessed February 2022].

[3] P. Shinners, "pygame documentation," [Online]. Available: https://www.pygame.org/docs/tut/SpriteIntro.html. [Accessed February 2022].

[4] Pygame, "pygame documentation," [Online]. Available: https://www.pygame.org/docs/ref/math.html. [Accessed February 2022].

[5] Pygame, "pygame.org," [Online]. Available: https://www.pygame.org/wiki/about?parent=. [Accessed 7 February 2022].