Rapport Oblig 1 INF-1400

By

Alexander Einshøj
INF-1400
John Markus Bjørndalen

1 Introduction

In this assignment, I've created a breakout clone using OOP in Python, focusing on the use of classes and methods.

2 Background

In this assignment, I have used Python 3.4[2]. Python is a programming language that supports the OOP principles used in this assignment. I have also used Pygame 3.4, which allows me to use the Pygame library for easier making games.

3 Design

None of the classes in my code have any particular relationship with each other, there's particularly no use of inheritance in the code, as I didn't find it to be needed. The game consists of four classes; *Ball*, *Paddle*, *Canvas* and *Brick.* The *Canvas*-class is in many ways the main class, as it initializes the pygame window, and is keeping the methods which execute most of the actions in-game. As we can see from **Figure 1**, Canvas has way more methods than any of the other classes. The most important method in this class will have to be the *update* method, which tells the interpreter what to do for every passed frame (or clock tick). *Handle_events* is responsible for handling keyboard input of any kind, such as moving the player-controlled paddle. As seen in **Figure 1**, *Canvas* also has a *construct_level* method and a *new_level* method, which respectively creates the first level and the levels past the first one. Further on, we can see from **Figure 1** that both *Ball* and *Paddle* has an update method that tells us about the movement of each of the objects in game.

4 Implementation

For this task, I have chosen to use the built-in sprite library which can be found within the pygame library. This allows the objects to easily communicate with each other, and there are some really useful functions, that make it possible to empty a group of sprites (for example, I can remove all the bricks from the screen with a single line of code). The sprite system is really practical when making a game like this. There's a built-in collision detection, and you can easily manipulate each group of sprites, or all of them at the same time. In the class *Canvas*, there is a method called *remove_sprites*. All it does is, as it also states, removing all visible objects from the screen, it simply empties the sprite groups. The sprites cover close to all the code, by being so easily implemented and manipulated to get the results I desire.

5 Evaluation

In my solution, all requirements are fulfilled, and I have also added extra levels, life and score. The collision tests between the ball and the paddle don't work as well as I could have wished. Sometimes, the ball will get squeezed between the paddle and either of the walls, or just on either vertical edge of the paddle, which makes the ball do some weird movements. Further on, as is, the ball is able to destroy two bricks at once, if you hit right between two blocks, and I'm not really sure why. The ball also doesn't start by the paddle so the player can choose when to start, the ball just starts dropping down. This is an issue I really haven't had the opportunity to get around to, and it wasn't a priority either

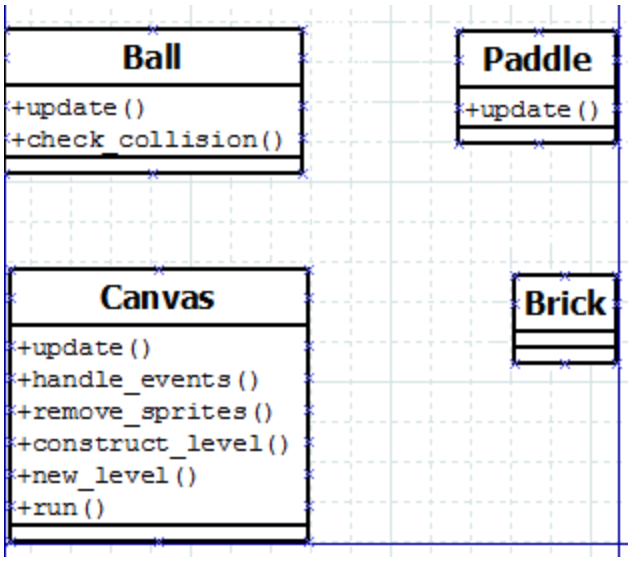
6 Discussion

The implementation of both the ball-brick collision and the ball-paddle collision could have been done better, as neither is fully optimal. The code is also not as well written as it could have been, as I find it somewhat hard to expand further at this point. One problem I came across at the later stages, which I haven't solved, was that I wanted there to be a delay after the objects are blit onto the screen, but before the ball starts dropping down. I think this is particularly hard because of the way my code is structured. I had a lot of problems early on, as I have no prior experience in creating games, and didn't know how to plan the code in a way that helped me at all. The implementation part was the hardest for me, I wrote several different "Breakout" files, trying to find a way to write it that best fit me. I ended up with using sprites, but it took me quite a while to learn how to implement it and write the actual code for it.

As for the group sessions, I've found them really helpful. It helps a lot being in a classroom with fellow students who all have different questions about the same topic. The assignment itself has been a very funny project, although it took me multiple weeks to complete the code, I feel it's really worth it, and I've definitely learned a whole lot!

7 Conclusion

To sum it up, I'm quite happy about the chosen implementation, although there are some bugs, two as far as I know. All the requirements are fulfilled, as well as some extra ones. This whole process has been an explosive learning period.


Ball
+update()
+check_collision()
Paddle
+update()
Canvas
+update()
+handle_events()
+remove_sprites()
+construct_level()
+new_level()
+run()
Brick