INF-1400: Object-Oriented Programing

Assignment 1 report

Amund H. Strøm

February 12, 2022

Introduction

In this assignment we made a clone of a game called breakout, with the focus on using object-oriented programing with an emphasis on using classes and methods.

Technical Background

The focus of this assignment was to use object-oriented programing (OOP), which is a method for structuring a program. The main concept of OOP is to bundle together related properties and behaviors into individual objects. Lest say for example that you would want to make a contact list. If we use the concepts of OOP, we would for example make a class that represented the contact list, and inside this class would be different properties like; name, phone number, email, address, etc. there would also be different behavior, also called methods which describes what we could do with this class; add new contact, delete contact, sort contacts, etc.
There is also a possibility to share properties or behaviors between different classes. To do so we simply create a class with the properties or behaviors we want to share, and then hand them out to wherever they are needed. This means that the things we share are all the same, until we change them otherwise inside the class they are shared to.

Design

I have created 4 different classes

1. A player class where I define how the player-platform looks and moves, and how it respawns and draws on the screen.
2. A ball class where I define how the ball will move around and how it will react to hitting the screen boundaries, and how it respawns and draws on the screen
3. A text printing class where I bundle together all the different text that will be printed on the screen in different situations
4. An enemy block class where I define how the blocks will look and how it will be draw on the screen. I also use this class to create a list of enemy blocks.

I did not find any use for sharing different properties or behaviors between classes, since they have nothing in common. Of course, there are similar names in the different classes, but they all have different values, so there would be no need to share between classes.

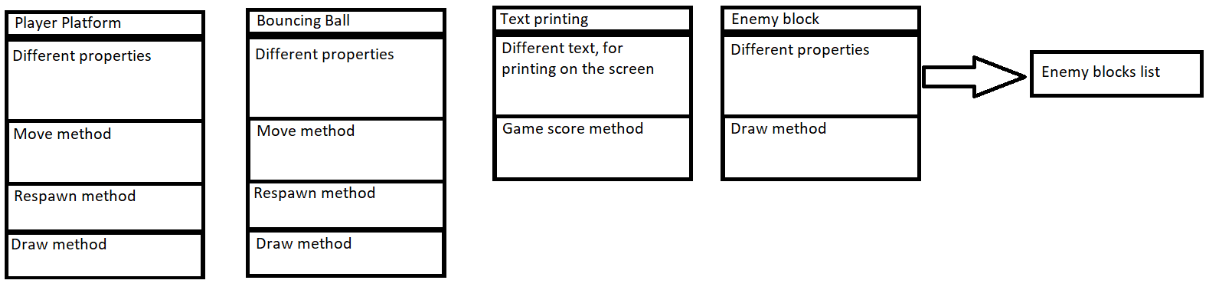

Figure 1: illustration of class implementation

Implementation

For this part I have chosen to talk about some parts that I struggled to figure out how to do and may not be obvious choices.

The first problem I encountered was how to make the ball bounce off with an angle when hitting the player platform. I worked out two different solutions, one is better than the other in my opinion. First, I realized that I had to use the pre-code to make my life that much easier. It provided me with two different methods, one for detecting if the ball collides with a rectangle and the other if the ball collides with another circle. After some trial and error, I figured out how to use these methods and implement them in my program. The first solution I figured out was to draw the player platform as a circle, and just hide most of the circle behind a rectangle, that way we visually got a semicircle. I then used the pre-code method to check if two circles collided, if they did, change the speed of the ball with the vector the method returned, multiply that with the speed of the ball, and then multiply that with negative 1 to make it change direction. To find the speed of the ball we use the Pythagoras theorem, since the ball has an X and Y speed, these will be the short sides of a right tringle, and the actual speed will be the hypotenuse. We do this to figure out the correct speed of the ball, so it will not slow down or gain speed every time it collides with something.

This was my first solution to make the ball bounce off with an angle, but I quickly realized that it would be difficult to make the semicircle functional, since right now it was only visually a semicircle. So, I figured out a different solution. That was to draw the platform as a rectangle, and then have a circle behind the platform, then we check if the ball collides with a rectangle (the player rectangle) and change its speed with the vector returned by the "circle collides with circle" method. This way, we only see the rectangle player platform, and it will bounce off with an angle when hitting different spots on the platform. This solution looks like this in game.

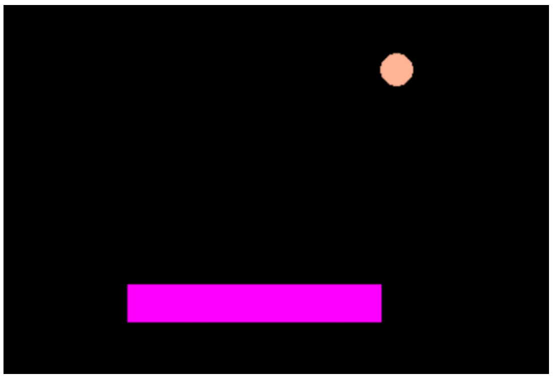
Figure 2: platform without the circle drawn

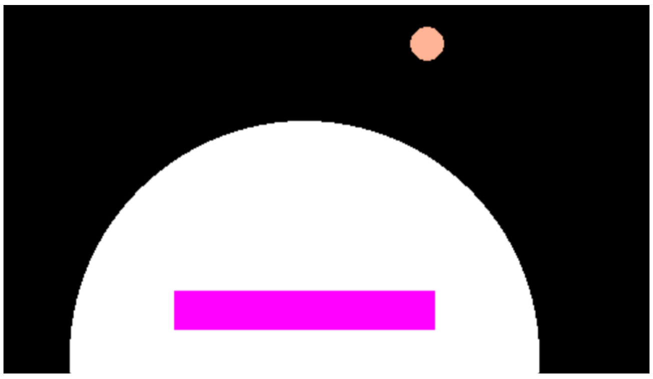
Figure 3: platform with the circle drawn

In my program I decided that I wanted the enemy blocks to have facial expressions, the way I have done it may not be that obvious and there is most likely a better solution to this. There are 4 different faces they can make, and two different "expressions", one is the idle state, and the other is the laughing state.

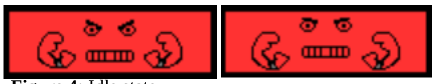
Figure 4: Idle state

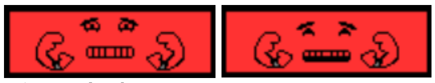
Figure 5: laughing state

The first thing I decided was that I didn't want every block to have a face, so the screen didn't look that cluttered. I did this by creating a random variable between 1 to 10, that rolled for every time a block was added to the "enemy block list". If this variable was higher than 6, that block would have a face. As I said earlier there are 2 different "expression" that the block can be in, the idle and laughing state. The idle state is the default expression, and it changes between two PNGs, to do so I have a counter that increases by one for every time the draw method is called, when this is equal to 60, change the blocks face, and when the counter is equal to 120, change the blocks face and reset counter. This way the block will change between these two different faces. To make it laugh I created another random variable between 1 to 100, that rolled every time the counter was equal to 120. If the random variable was higher than 90, it would instead change faces between the laughing faces, using the same logic as before. By using this method, we achieve all the blocks change faces at the same time, and randomly some blocks will start laughing.

I also wanted to stop the game when either the user missed the ball and lost, or when the user destroyed every block and won the game. To do so I have the move methods for the ball and platform inside 2 different if-statements. The outer statements checks if a variable is true, this variable is true if the ball has not hit the bottom of the screen, when it hits the bottom, it will change to false. The inner if-statements checks if the "enemy block list" is empty. There is also two different if-statements that uses the same variables and will print the correct text on the screen if one of these is true.

Evaluation

This submission fulfills all requirements:

1. Object-oriented programming
2. Platform is controlled by keyboard
3. Ball bounces with different angles on the player platform based on where it hits
4. A brick disappears when the ball hits it
5. The ball bounces of the wall and ceiling, angle in = angle out
6. The game is won when all bricks are destroyed, and the game is lost when the ball hits the bottom of the screen
7. Well-structured and commented code.

My code may fall a little short on well-structured, but I tried to comment everything, so it is understandable. I have also noticed that sometimes the ball will be sucked inside the wall, then suddenly pop out. I have no idea why this is happening and see no pattern to when and why this happens.

Discussion

There is probably a lot of things that could be improved, but I lack the skill and knowledge to do so. For example: I have not shared any properties or behaviors between classes, I did not find any use to do this, but there is probably a lot of places where I could have done this. The respawn method for the ball and platform just set all the necessary values back to the original value, this can probably be done better. The facial expressions for the enemy blocks can probably be changed with the use of the in-game time, but I didn't understand how to do that. And the list goes on. The thing I struggled the most with was to make the ball bounce with an angle depending on where it hit the player platform. As I said earlier, I found to different solutions to this, but I think the final solution was much better and easier. It was also a lot of help to use the pre-code, and not make my own function for calculating vectors for the ball. But overall, I learned a lot of things in this assignment, I thought that this was a nice way to introduce people to python and pygame. And hope to get some feedback on my code.

Conclusion

In this assignment we made a clone of the classic breakout game. I made a program that fulfills all the necessary requirements, but there are different things that can be improved, but I simply lack the skill and knowledge to do so. For me, this was a fun way to be introduced to python and pygame.