INF-1400

Assignment 1

Magnus Dahl-Hansen

February 2020

# Introduction

In this report I describe my implementation of the game “Breakout” where I try to have a object-oriented design and make use of objects and classes. My program is written in Python with the help of the Pygame library.

## Requirements

The game should support the basic mechanics of the original game:

* First requirement: create a display and the objects used in the original game
* Second requirement: control a platform with the keyboard or the mouse
* Third requirement: make a ball bounce off the platform in different angles according to where the ball hits the platform
* Fourth requirement: make a brick disappear when the ball hits one
* Fifth requirement: make the ball bounce of the walls and ceiling in the same angle as it hit the walls/ceiling.

# Technical Background

When starting on this assignment one should be familiar with how classes, if-statements, loops, different Pygame library functions, blitting and lists work.

Classes: an object contains a collection of variables that gives it a form and a behavior. A class is a container where all these variables are stored.

If-statements: decides whether an algorithm should continue or stop its task depending on if the given statement is true or false.

Loops: executes an algorithm if a stated condition is true. When the given statement becomes false the loop stops or breaks.

Pygame library functions: the Pygame library contains many different functions. Some of the functions used in this implementation are:

* pygame.draw, which draws a specific figure
* pygame.image.load: loads a picture to the program
* pygame.transform.scale: scales a picture to a desired size
* pygame.display.update: updates the display
* pygame.event.get: checks for user events

Blitting: is copying the pixels that belongs to an object or an image onto a desired destination desired, for example the screen.

List: is a structure that contains a sequence of values or elements. The elements or values inside the list are often referred to as items.

# Design/Implementation

I created three classes for the objects in the game, a paddle, a ball and a block. These classes contain the variables for the objects and a couple of different functions that draws the figures and makes them interact with each other.

The first two classes contain properties for a paddle and the blocks in the game. Both contains a function which draws rectangles, but in addition the paddle class also has a function that moves it. The paddle is moved in a desired direction by using a function that detects if a key is pressed. If a key is pressed, the paddle’s position variables are changed, and it can be moved to a different place along the x-axis.

The class for the ball contains the properties of a circle and four functions that moves it and makes it interact with the walls, paddle and the blocks. The ball moves similarly to the paddle by changing its x and y-coordinates continuously. When the ball hits a rectangle or a wall its speed in the x and y-axis is changed depending on where it hits a rectangle or a wall.

Instead of using the pre-code for the interaction between the ball and the paddle or block, I implemented these functions myself. Each of the functions checks if the ball hits one of the rectangles by testing if its center position plus its radius are bigger or less than the top, bottom or side edges of the rectangle. If the ball hits on of the sides, its speed changes to the opposite direction.

The paddle is divided into three pieces and each of these parts changes the ball’s speed in the x and y-axis depending on where it hits the paddle. If it hits the paddle on the left side the ball will bounce to the left and its speed in the x-direction is increased so it will bounce off with a bigger angle. If it hits the right side, it would bounce to the right side and with the same angle as to the left side, but only it’s y-direction is affected if it hits the middle of the paddle.

To create several blocks, without making a class for each block, I made a loop that produces three rows and eight columns each containing a block. Inside the loop the coordinates for each block are changed so that they do not appear on top of each other. After being created they are then stored inside a list. Inside the main program loop they are once again looped over and drawn on the screen one by one. In addition, there is one more loop that checks if the ball hits a block, and if this is the case the block will be removed from the list and erased from the screen.

If the ball touches the floor of the screen, all the blocks are then cleared from the list and a “game over” picture is blitted to the screen.

# Discussion

I know the design of my program is not optimal and it has some flaws. A minor flaw is that when the ball hits the floor and the game is over, the program still runs in the background, until the user closes it.

Another problem occurs when the ball interacts with the middle part of the paddle. The paddle is divided into three parts, left side, middle and right side. When the ball hits one of the sides its speed increases, and when the ball hits the middle the speed in the x-direction should decrease, but It does not. I have tested and tried to figure out what is going on by printing out a statement if the ball hits the middle part of the paddle, but even though it hits the middle part and prints the statement, its speed is still unaffected.

The interaction between the ball and the blocks is not perfect either. Sometimes when hit in the right angle, the ball sort of glides through two blocks at the same time, or it could bounce upwards even if it hit the block from underneath. I think the reason for this problem could be an error in how implemented the interaction between the circle and the rectangle or it could be a problem with the draw circle function which has an optional width-variable that I did not use. While discussing this optional width variable it says on pygames website that the draw circle function returns “*a rectangle bounding the changed pixels, if nothing is drawn the bounding rect’s position will be the center parameter value (float values will be truncated) and its width and height will be 0”.* I am not quite certain on what this means, but I believe that when the ball hits the corner of a rectangle something happens with its values, causing them to become floats or truncated resulting in the ball behaving in a strange way. A reasonable solution to this problem could be to draw the ball as a small rectangle instead of a circle, getting rid of the edges and make it easier for it to interact with other rectangles.

I have not had the time to fix this problem since most of my functions are based on a circle interacting with the rectangles and changing these would take some time.

# Conclusion

This assignments task was to recreate a game called *breakout* which should support the basic mechanics of the original game. I have explained my implementation and design and tested my program. It runs smoothly but have some minor errors when the ball interacts with the paddle or blocks.

From this assignment I have learned how to make objects interact with each other and what typical errors could occur. For the next assignment I will implement a more structured code and hopefully come up with better solutions to solve problems like the ones in this assignment

# References

<https://www.yorkcs.com/2019/10/04/pygame-basics-keyboard-input/>

<https://www.pygame.org/docs/ref/draw.html#pygame.draw.rect>

<https://www.pygame.org/docs/ref/draw.html#pygame.draw.circle>