Assignment 3 report - INF-1400

Jørgen Kristensen Ivan Moen

April 15, 2021

1 Introduction

In this assignment we were to make a *Space mayhem* clone using OOP principles, with a specific requirement being inheriting from the Sprite class. This assignment also emphasizes good coding practices with well documented classes and methods.

2 Background

For this assignment, Python 3.9.2 was used. For server-to-client communication, we used the library named *Asyncio*. For client frontend we are using the library Pygame to draw the game object on the screen of the player.

3 Design

Give an overarching view of the structure of your solution.

For this assignment we have one major class hierarchy of gameobjects, with a general layout of a gameobject (ie. a ship, planet etc.) that has two subclasses: One subclass is based on how the object is behaving on the server side, while the other is based on local representation of the object on the class, which means the local class naturally will inherit from the *Sprite* class in pygame as well.

We also used a object-oriented approach for server and client handling, thus one can create a new server by calling it as an object, and add new players as objects as well.

Describe how your objects fit together, a figure like figure **??** must be included, and you should refer to it in the design section.

4 Implementation

Describe implementation details, particularly those that are not obvious choices

Since the game is played online, most of the information is processed serverside. The players send certain information to the server:

- player id
- initial position, velocity and direction
- player input(combination of keyboard buttons left, right, up and space-bar)

To illustrate this client-server relation, we can see how the player input handling is working: the player clicks a button(lets say up arrow), the client will do an OR comparison with the default 0: $0000 \vee 0100 = 0100$. The server will then confirm this using an AND operation: $0100 \wedge 0100 = 0100$ and do the following on the remote object: increase velocity vector and reduce fuel. This new information will then be sent back to the client which will update the local object with the new information, and draw the new changes on the screen for the player to see. The information is coded and decoded into JSON.

For the implementation of the paddle, the visual representation on the screen is different from the internal representation used for collision detection, by representing the paddle in this way we acheive...

5 Evaluation

Examine if your submission fulfils the requirements and what shortcomings exist.

In this solution, all requirements are fulfilled, but collision detection between the ball and paddle is inaccurate, due to differences between the visual representation and the implementation...

6 Discussion

Discuss what could be done better, problems you had, experiences etc. (we also appreciate feedback on the assignment or group sessions)

The implementation of the paddle-ball collision could be done *some other way*, but due to *some reason*, the current implemetation is better.

After spending two days trying to write the report in LaTeX, I gave up, and wrote it in Word instead.

7 Conclusion

Sum up the previous sections.

I have implemented a solution that fulfills the requirements, the implementation is moderately buggy, but does not crash too much.