University of Tromsø

INF-1400-Object Oriented Programming

Mandatory assignment 3

Helge Hoff & Øystein Tveito

Department of Computer Science

April 7, 2014

Contents

1 Introduction 2
1.1 Technical Background 2
1.1.1 Mayhem 2

2 Design 2

3 Implementation 2

4 Discussion 2

5 Conclusion 4

References 4

1 Introduction

For this project a clone of the arcade game Mayhem will be implemented. There is two authors on this project, so the workload will be shared between.

1.1 Technical Background

1.1.1 Mayhem

2 Design

This game is a two player game sharing one screen and keyboard. Each player have a set of keys associated with him, making him able to navigate and shoot his spaceship.

3 Implementation

World

4 Discussion

This section starts with a list of requirements and an explanation on whether or not this implementation has successfully implemented this:

- Two spaceships with four controls: rotate left, rotate right, thrust, fire.
  - All the controls are implemented with the addition of backwards thrust. The backwards thrust is added to make it easier to avoid the black hole, and in the same time be able to shoot against your opponent.
- Minimum one obstacle in the game world. This can be as simple as a single rectangle in the middle of the screen.
  - A black hole has been added as an obstacle in the middle of the world.
- Spaceship can crash with walls/obstacles/other spaceship.
  - The spaceships will be absorbed by the black hole. In the event of a crash with an other spaceship, the spaceship with the least amount of health will be destroyed.
- Gravity acts on spaceships (the original has no gravity acting on the bullets, but you can choose what works best).
  - There is no gravity pulling downwards because this is in space. The black hole stands for the gravity, pulling everything towards it and in a spiral.
- Each player has a score that is displayed on the screen. A player's score increases when he shoots down the opponent. A player's score decreases if he crashes.
  - A score is implemented and shown on screen. The score increases when a shot from that ship hits the opponent. On destruction the score will be reset to zero.

- Each spaceship has a limited amount of fuel. To refuel, it must land on one of two landing pads. Alternatively, you can put a "fuel barrel" at a random position that is collected by the first spaceship reaching it.
  - Each spaceship has its own fuelling pad that recharges its fuel and bullets. The pad is unique to the spaceship, so a player can only land on his own pad.
- Scrolling window, as seen on the video, is NOT a requirement.
  - Not implemented.
- The implementation must consist of a minimum of two files. One of these shall be a config.py file containing global configuration constants, such as screen size, amount of gravity, amount of starting fuel.
  - The config file is implemented and holds some variables that govern screen size and some initial values.
- The main loop must have timing so that the game is playable on different computers.
  - The game loop is controlled by the pygame clock. This is set to 60 ticks each second, so as long as the computer can handle 60 FPS (any fairly modern computer can) the game will feel the same. If the computer is to slow for this, it will be slower. This is done to not make the spaceship jump long distances for each frame.
- The game shall be started using Python's if __name__ == '__main__': idiom. Inside the if test, a single line shall be instantiate the game object. All other code, except the game configuration constants, shall be inside the classes. This will simplify profiling and documentation generation.
  - The first exception from this rule is that the main function is in a file of its own. To make the game run, pygame need to be initiated in every file expecting to have anything with pygame. For this reason, the first line in main is initialization of pygame. The second exception is that the game class do not start the game by it self. This is done on purpose, so the caller can initiate the object, and chose when to start it. Giving more control to the user is a informed choice, and the authors stand by it.
- All visible objects shall subclass the pygame.sprite.Sprite class. The sprites shall be put into groups using pygame.sprite.Group. Then updating and drawing shall be performed using Group.update and Group.draw.
  - Every visible object is a subclass of sprite, and every object is updated and drawn using the Sprite functionality.
- All modules (files), classes and methods shall contain docstrings. If you are working in a team, the module docstring at the top of the file shall contain the name of both authors. When you are done programming, html documentation shall be generated using pydoc -w command.
  - This requirement is met.
- The last task is to profile the code using cProfiler. Take a screenshot of the result and include it in the report. Giva a short summary of the sesult and discuss where you would focus to imporve the performance of the implementation.
  -

5 Conclusion

References

[1] wikipedia.org, *Boids*: http://en.wikipedia.org/w/index.php?title=Boids&oldid=580365466 [Online; accessed 26-02-2013]

[2] wikipedia.org, *Leap*: http://en.wikipedia.org/w/index.php?title=Leap_Motion&oldid=592637003 [Online; accessed 03-03-2013]