Dr. Asieh Parsania
Institut für Mathematik
Universität Zürich

MAT101: Programming

Group Project: Flappy Bird Game

Loreno Heer

Deadline: 14th December 2018

1 Objectives

In this project you will program a clone of the game "Flappy Bird". We will use the python module pygame to help us implement the game. The main tasks will consist of implementing graphics, collision detection, game physics a highscore system and input handling. A template will be provided in which the code can be implemented. If you have completed the main tasks you can come up with some exta functionality of your own choosing.

2 Prerequisites

Every member of the group needs to know the basic of the python pygame library and should read the game design tutorial: .

Every member should document what they programm, also indicate which part of the code has been written by whom. Also the code itself should be well documented using proper docstrings.

To use the pygame module you will have to install it by executing the following command from the terminal:

```
python3 -m pip install -U pygame --user
```

You can test that it is working by executing the following command:

```
python3 -m pygame.examples.aliens
```

You can look at the source code of the example programs. They should be in the following directory:

```
~/.local/lib/python3.5/site-packages/pygame/examples
```

You can change to this directory by typing the command:

```
cd ~/.local/lib/python3.5/site-packages/pygame/examples
```

3 Guidelines

The following tasks are to be implemented in the file called template.py:

Task 1 (Graphics)
In this task one handles the loading of graphics and the drawing of graphics on the screen. Whenever the `update_graphics()` function is called one should re-draw everything on the screen.

There are some sample graphics provided. You can change them if you want. Either you can draw your own graphics or use some graphics from the websites provided in the links below this document.

Make sure you calculate the correct positions, for where to put the graphics. For example the pipes should always overlap either above or below on the screen. They should not start out of nowhere (float in the air).

You can change the background after a certain amount of collected points to the night background.

Task 2 (Game physics)
In this task one handles the positioning of the bird, the change of position in each timeframe as well as the collision detection. We use the following variables to store the information of the bird: `bird_pos = (x,y)` and `bird_vel`. In each timeframe the birds vertical position should change according to the formula: $y_{t+1} = y_t + bird\_vel$ and $bird\_vel_{t+1} = bird\_vel - g$ where g is a reasonably (for the game) choosen gravitational constant.

Make sure that the bird can not fly higher than the top of the window/screen.

Task 3 (Collision Detection)
You should write an algorithm that calculates if the bird touches a pipe or if he touches the ground. This should cause the game to end. Also you should detect if the bird successfully flies through a pipe and then count a point.

Task 4 (Input handling)
The game will run in a loop that consists of handling input, checking game physics and then updating the screen. Here one has to check which(if any) key(s) have been pressed on the keyboard and then update the bird data accordingly. For example if one wants to use the space key as the jump/fly key, then upon pressing the key, the birds upward velocity should be set to a reasonable value.

Task 5 (Highscore system)
In this task one should load and save the highscores and usernames. Before playing a player should enter his username which will then be stored with the score. The score should only be updated in the file if it is greater than the previous score or if the user did not play before. The 10 best scores should be stored in the file. Other scores should be discarded. The highscore table should be displayed whenever one player quits or looses a game.

Task 6 (Extra Task)
Think about some extra(s) to implement in your game. For example food could periodically appear on the screen giving extra points if the bird eats it.

Task 7 (Optional Task)
You can use the game code to create a different game of your own design. For example a space shooter or a jump and run kind of game. (Depending on what you want to do, this can be easy or hard).

Feel free to ask me any questions you have while writing the project.

4 References

Make sure you read the documentation of the pygame module. (you dont need to understand everything but you should understand the basics and especially all the parts that are used in the code). You can also read some of the tutorials to understand how games are programmed.

- Pygame documentation: `https://www.pygame.org/docs/`
- Pygame tutorials: `https://www.pygame.org/wiki/tutorials`
- Free game assets (graphics and sounds): `https://itch.io/game-assets/free`
- More free game assets: `https://www.garagegames.com/community/resources/view/23092`
- The Spyder Python IDE: `https://docs.spyder-ide.org/installation.html`

- The PyCharm Python IDE: `https://www.jetbrains.com/pycharm/`
- The Eric Pytho IDE: `https://eric-ide.python-projects.org/`
- Emacs, the best editor (but difficult to use)[1]: `https://www.gnu.org/software/emacs/`
- Vim, the second best editor (also difficult): `https://www.vim.org/`

5 General Notes

- The goal of this project is to experience programming in a group. Discuss the project as a group and then divide the tasks among yourselves.
- You should discuss your progress with the supervisor of the project. Whenever you have questions about your project, feel free to ask them during the exercise class or post them in the forum.
- Once you have written your code you should briefly describe your results. You should include interesting examples and illustrations (if they are part of your project).
- To hand in your project, just send an email to the supervisor of your project. Make sure that it is clear who is responsible for which task.
- It is important that you understand the entire code of your group, not just the part that you have written yourself. In particular, you should be familiar with the prerequisites.
- During the last week of the semester, every group will have a 20-minute presentation of their project. Each member of the group should prepare a 4-minute presentation of their own code and be ready to answer questions about the entire project.
- For the project you will be graded as a group, but for the presentation you will be graded individually. Together the project and the presentation account for 30 percent of your final grade.
- The exams will take place during the exercise classes, i.e. on December 18th.

[1]Emacs and Vim heavily rely on keyboard use and do not have much mouse functionality