Dr. Asieh Parsania
Institut für Mathematik
Universität Zürich

MAT101: Programming

Group Project: Creating a board game.

Jon Eugster

Deadline: December 4th, 2017

1 Objectives

The goal of this project is to learn how to design a bigger Application with Python. In this project we will create a board game called Go, known for it simple rules but very high strategical complexity. This game is nice to implement because the graphics and concepts are not too complicated but still you learn about many important parts of creating more complex applications.

This project has more instructions since you will learn a bunch of new concepts (especially how to create the graphics) but those instructions guide you step-by-step through the project and contain example code for all new concepts.

Therefore, the project is also accessible for students that do not yet feel very comfortable about their coding skills. The finished application comprises about 60-80 lines of code per person, where a big part of it is given in form of examples in the instructions.

2 Prerequisites

What you know from the Python lecture so far. In particular you should be familiar with Classes and how to import from another module. Furthermore, you need to install the module Pyglet:

```
pip install pyglet
```

3 Guidelines

Task 1
(Controller, Easy) The controller is the main program that communicates between the Model, which does all the calculations and knows the game logic, and the View (the user interface). The goal of this task is to get data from the Model and pass it on to the View and vice versa get commands from the View and call the corresponding functions of the Model.

Task 2
(View 1, Medium) The View is the user interface. It displays the current state of the game and it passes on user inputs to the controller, but it does not do any computations. Those are done by the Model. The goal of this first task is to draw the background and the game itself, which comprises the grid layout and the placed stones. This is mainly displaying images in the right places.

Task 3
(View 2, Easy-Medium) In this second task all the information around the actual game will be displayed (E.g. labels displaying the score). Additionally, this task includes setting up the communication with the controller and getting some user inputs from clicked buttons.

Task 4
(Model 1, Medium) The Model is the part of the application that does all the calculations and knows the game logic. It is completely separated from the View (user interface). This task contains setting up the Model, preparing the information the View needs to know, and writing smaller functions that implement some parts of the game logic, e.g. a function to remove stones from the board.

Task 5
(Model 2, Medium-Challenging) This task comprises implementing the actual game logic. In particular the implementation of the function that gets called when a player wants to place a stone. First, this function needs to check if the move is valid. If it is, the stone can be placed, maybe some of the opponents stones need to be removed and also all allied adjacent stones need an update.

Task 6
(Additional) The instructions contain several small additional tasks and one big one ('mark territory' in the Model). The small ones are optional, for the big one there is a solution available (read the task, you can just import the solution). You can also extend the application with your own ideas.

See the additional sheets for detailed instructions for the single tasks.

4 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). This description may be very short; it should certainly not be more than one page of text.
- hand in your project, just send an email to the supervisor of your project. Make sure that it is clear who was 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 15-minute oral exam on their project. The main examiner will be the supervisor of your project, but another assistant will also be in attendance. Each member of the group should prepare a 2-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 oral exam you will be graded individually. Together the project and the oral exam account for 40 percent of your final grade: 30 percent for the project and 10 percent for the oral exam.
- The exams will take place during the exercise classes, i.e. on December 12$^{\text{th}}$ and December 15$^{\text{th}}$. On December 4$^{\text{th}}$ you will receive an e-mail asking you to choose a time slot for the oral exam.