Final Project

The final project for this course is to create a turn-based game in Python with a graphical user interface. The choice of game and details of its rules are up to you. The end product should be a finished game that is fun to play.

Turn-based Game

The game you choose should involve one, two, or more players taking turns to be active. In a turn-based game, the computer waits for input, doing nothing until the player makes a move. Battleship, Monopoly, and minesweeper are examples of turn based games. For contrast, in an arcade game, the player is continuously providing input while at the same time the computer takes other actions.

There is a large list of turn-based games on our course website. You can select one of these, choose another game you know and enjoy, or even create your own completely new game. However, it is probably to your advantage to imitate an existing game, where the end goal is clearly defined.

Graphical User Interface

Your game should have a graphical user interface written with cs1graphics. During the first steps programming the game, you will probably want to implement a text interface as well, using print and `raw_input`. The finished product should be completely playable without having to provide command line input. If your code is well designed, changing the user interface back and forth between text and graphics should be straightforward.

Computer Opponent

The game must have the option of playing with human or computer players. A two player game should provide the option of human vs. human, human vs. computer, or computer vs. computer play. Multiplayer games should be equally flexible, and even a single player game (such as minesweeper) should have the option of a computer player that makes moves automatically.

The computer opponent should always make legal moves, but does not need to play well at all. Having it select a (legal) move randomly is perfectly acceptable.

Software Design

Good software design is an important component of the project. Separate your code into modules in separate files. Classes and most methods should have docstrings. Create unit tests as you go, and leave them in the final product. Choose names carefully and follow a consistent naming convention.

In addition, the program should provide rules and instructions to the player, preferrably as a help screen or at least with printed output at startup.

Deliverables

There are four checkpoints in this project where you need to hand in something. The proposal, where you describe the project you plan to create, an alpha version, a beta version which is playable, and a finished version.

Project Proposal

The proposal should describe the game you plan to create. It should include, at least:

- The name of the game and the rules as you plan to implement them. How is the game set up, how does a turn work, and how does the game end?
- A plan for the user interface: What will the player see, what actions will the player need to take to play the game. Give the sequence of user actions and game responses that happen during the course of a turn. A sketch of the screen layout would also be appropriate here. You should also consider how you will implement a simple text interface for development and testing purposes.
- Additional "fantasy" features you might include if time permits.
- The major components (classes) your program will need, and some of the methods that they will support. A component diagram such as Figure 7.4 of the textbook would be helpful.
- A discussion of steps you can take towards writing the program. What is the first thing you will write? What do you expect the alpha and beta versions to look like?
- An idea for how the computer might play the game. You can be sophisticated, but also try to think of the simplest possible way to generate legal moves.

The proposal documents the specifications and high-level design of your game. A thoughtful and detailed proposal will be helpful to you as a reference when actually writing code. It also gives you a chance to get feedback on major design issues before committing to them. The proposal serves as an agreement about what your project will deliver. Though this is allowed to change, you should check with the instructor before making a major change to the proposed game.

Alpha Version

The alpha version should include basic classes and data structures your game will need, the beginnings of a text-based interface to the game, and any other components of the game that can be isolated and built. For example, a Battleship game would include a Board class at this point, with the ability to store ships, hits, and misses. It would be appropriate to have a text display of the board and methods to place ships and take shots.

The alpha version does not have to do anything, although each file should have unit tests and it will serve you well to keep them working at all times.

Beta Version

The beta version should be a playable version of the game, which may be missing important features or still have many bugs. The core component of game play should be working well, possibly only through a text-based interface. The graphical interface should be well underway. The basic classes your program requires should be written, stable, and documented with working unit tests. The computer opponent should be working.

For example, a checkers game might have a graphic display of the board but still require the user to input moves by typing them. Checking for legal moves should work, but maybe advanced game rules such as double jumps and kings would be missing at this stage. The end of the game (declaring a winner) might not be finished yet. The computer player works by trying random moves until it happens to find a legal one.

Assignment Details

Due Dates

Proposal Friday, November 18, in class or by email (`bryan@slu.edu`) before 2pm.

Alpha Tuesday, November 29, midnight

Beta Tuesday, December 5, midnight

Final Tuesday, December 13, midnight.

The proposal, alpha, and beta versions should be turned in on time. The final version will have a late policy with a loss of 5 percent per day until Sunday, December 18 at midnight, after which no projects will be accepted under any circumstances. Our final exam is Monday, December 19 from 8-9:50am.

The alpha, beta, and final version should be handed in electronically by their due dates. Hand in all files necessary to run your program. You may also want to hand in a plain text file called `README` if you have any special requirements or instructions to the user.

Grading

The proposal, alpha, and beta versions each count 10% towards the final project score, and you can expect to receive full credit for these if they are handed in on time. The difficulty and complexity of the project does count towards the final grade. For example, a playable Monopoly game which is missing some game features would be a better project than a perfectly working tic-tac-toe program. The proposal and the feedback you receive on the proposal should give a good indication of what features you will need to include in your game.

This project will count as one third (1/3) of your programming score for the course, which amounts to about 17% of the total course grade.