Final Project
CS111: Intro to Computer Science, Winter 2016
Anna Rafferty

You have three options for a final project: creating a computer game that incorporates some level of computer intelligence, creating a simulation environment, or creating a data analysis and visualization. Regardless of which option you choose, your implementation should include at least two different classes that you create.

1. **Choice 1:** A computer game that incorporates some level of computer intelligence.
   
   - Your game can be entirely text based, or you can use Turtle graphics, graphics.py, or cImage.py if you’d like to make it more visually appealing. No graphics are required, though; I’m much more interested in seeing how much you’ve learned about how to implement the algorithms that you want to use. Regardless of whether your game uses text or graphics, it should include at least one class and likely at least two.
   
   - Some possible games to implement: Scrabble, Bagels, Dots and Boxes, Battleship, Go Fish, Bejeweled, Connect Four, Mastermind, etc.
   
   - Some games to be wary of: a text-adventure game (these can be done well, but they usually devolve into a morass of nested if statements that aren’t interesting to write or read); Blackjack (the intelligence either ends up being too simple with following rules you found elsewhere, or learning how to play well becomes too complex); Hangman, Tic-Tac-Toe, Pig, etc. (too simple/we already did that); Go, your favorite German boardgame (too complex).
   
   - The requirement of “some level of computer intelligence” means this: the computer should play against you or manage your game in some kind of intelligent way. You don’t need the computer to be a world-class player at your game (or even close!) - that’s what CS 321 (Artificial Intelligence) is for - but you should make an effort on this front. This is an opportunity to be creative and to think about how to turn an idea for a smart strategy into an algorithm.

2. **Choice 2:** A simulation environment.
   
   - One of the things computers are good at is simulation of systems that evolve over time. Consider, for example, a computer model of the solar system like we talked about in class. You can give the computer the initial positions, masses, and velocities of the largest objects in the solar system, plus a computational mechanism for simulating the effects of Newton’s law of gravitation. Then you can set the system running, and your solar system will go through its motions. You could then try adding a comet with a mouse click or two, and watch how the comet’s orbit is perturbed by close encounters with planets or asteroids. If the simulation takes relevant physical laws into account, the evolution of the system will be a good representation of the workings of the real solar system.
   
   - If you choose this project, you will simulate some time-dependent system that can be represented effectively in a rectangular display. For example, you could show a side view of an aquarium or a landscape with rain, clouds, plants, and birds. Your system will need to be animated, and there will need to be a diverse enough collection of elements to your system that the objects will have interactions of some kind with one another. For instance, big fish eat little fish, and raindrops cause plants to grow. The objects interact based on their proximity and their type. You should not choose a solar system, since the book already gives an example of that.
   
   - Your simulation may include pretty much any features you can cook up, but its minimal features must include: (1) More than one type of moving object (including multiple classes), (2) “Interactions” between objects (i.e., one object causes another object to react in a certain way), and (3) Some feature that allows either mouse-clicks or keystrokes or some other form of user input to affect the simulation in some way (click to create a plant, set the animation speed, etc.)
3. **Choice 3:** Data investigation/visualization.

- If you choose this option, you should select the dataset of your choice and come up with a well-formed, testable hypothesis. For example, maybe you want to scrape the web for census data and public-health statistics, then compare all that to some theoretical epidemiological model. Or maybe you’d like to process a bunch of ancient texts written in Latin, looking for allusions among them. Or perhaps you’ve got a giant database of protein interactions in human cells, and you want to try to infer something about the network of interactions as a whole.

- The software you write should analyze the data, collect it autonomously (if that makes sense for your data source), and present the results in a useful and non-trivial way (e.g., a visualization).

- Your analysis should incorporate a fairly large dataset, and your final project should also include a short writeup (3-4 pages, single-spaced) that discusses the background of your problem, your hypothesis, your data sources, how your program addresses your hypothesis, the results, and the conclusions you draw from these results.

- Your visualization should also be influenced by user input in some way: for instance, the user could indicate how many years of climate data to include, or determine how many graphs to display, etc. Your project should include at least one class and likely at least two.

Regardless of your project selection, it may be in a partnership of your choice. Unlike with pair-programming assignments from the course so far, you are not both required to work on all aspects of the project, but you are all expected to do your fair share. I will allow individual projects if you have a good reason that you wish to work alone. I very strongly encourage you to work in a partnership; it will let you accomplish more and, I hope, have more fun. If you want a partner but haven’t been able to find one, please try posting on Piazza (there’s a thread specifically for this) or come talk to me. We’ll make sure that everyone who wants a partner has one.

**Grading**

Your project will be graded based on the following criteria, in decreasing order of importance:

1. **Correctness:** How cleanly does the program run? Is it bug free? Does it crash? Do the functions actually do what the comments say they do? (i.e., does what you say you implemented match your actual implementation in terms of the logic?)

2. **Complexity:** How complicated is your game/simulation/data analysis? If programming a game, how complex is the computer intelligence?

3. **Style:** Is the code well organized, readable, and well documented? Are you making classes in cases where it might be helpful? Are you avoiding duplicated code? (Keep following the style guidelines on Moodle!)

**Correctness is the most important factor.** Let’s say you choose the game option. A simple text-based program that runs correctly, is error-free, and achieves the above specifications will get a better grade than an incredibly smart chess program that crashes and produces unreadable output. (Of course, if I’m comparing side by side two programs that run perfectly, the one that accomplishes more will receive the better grade. But that’s a secondary concern.)

*To reiterate: It is most important to submit something that works. Set modest goals and achieve them first, then enhance your program later. If you choose a game, only work on getting computer intelligence working after you have the basic game working with human player(s).*

**On academic honesty:** You must submit a program that you write yourself/yourselves. You are more than welcome to obtain help from other students in getting ideas and in debugging your code, but you must write your own program. No one else should type in any code that you submit. You may not use Python code for your game from the Internet or from other sources. (Ask me first if you’d like to use a module
that you find online.) You must acknowledge any sources of assistance. Remember, I must report suspected violations of academic integrity to the dean, and if you are found responsible, the penalty in this class is failing 111. It’s not worth it.

Submission: Project Proposal

Due date: Submitted on Moodle by Friday, February 26, 11:55PM. Only one of you and your partner needs to submit a proposal.
You must submit a proposal for your project. This proposal should include your partnership (or an explanation of your plan to work alone), what type of project you’re planning to do, and a list of pieces that you plan to implement, in order, to create your project. This should be a plan of attack for your project, so that you can think through what steps you will need to take to make your project work. Writing a good proposal will make it easier to actually carry out your project. Proposals should be roughly 1 page.

Let me know if you have questions about the rules or suitability of a particular idea. I recommend talking to me about your thoughts about the project early to make sure that you haven’t picked something too easy or too hard. The biggest trap on this assignment is picking something too complex and not having the time to finish it. Consult with me.

Submission: Final Project

Due date: Submitted on Moodle by 5PM on the last day of finals, Monday, March 14. No late projects may be accepted without an extension from the dean’s office.
Create a directory that has all of the files for your final project - make sure all of the files are in that directory. In this directory, include a file called readme.txt. Zip up this directory, and upload it to Moodle. Additionally, fill out the final project partner survey on Moodle if you worked in a partnership; this allows me to know if you both contributed or if someone did all of the work, and to set grades accordingly. If you work in a partnership and do not fill out the partner survey, you will not receive credit for the final project.

The readme.txt file should include the following information:

- A description of your program and its features. This should include a complete description of the strategy of your computer intelligence if you programmed a game, or the details of how your simulation works if you chose a simulation.
- A brief description/justification of how it is constructed (class organization, how data are stored, etc.)
- A discussion of the current status of your program - what works and what doesn’t, etc.
- Instructions for running your program.

This file may be brief, but it’s very important that it’s clear and complete: if I don’t know how to run your program or I’m not sure what your computer intelligence is supposed to do, it’s very hard to evaluate it!

If you chose a data analysis project, also upload your 3-4 page write up.

Final Project Demos

During our final exam time slot (8:30-11AM, Saturday, March 12), we will have a demo session. Your projects don’t have to be complete at this point, but you should make sure you have something to share with me and with your classmates. This is a great opportunity to show off your work, see what others have done, and get formative feedback to help you make your project as awesome as possible by the due date. We’ll likely spend about the first half of the exam period on demos, and in the second half, I’ll stick around for informal office hours in case you want help debugging/figuring out how to do something/etc. Everyone is required to attend the demo session.