UNIVERSITY OF CALIFORNIA, MERCED

BERKELEY · DAVIS · IRVINE · LOS ANGELES · MERCED · RIVERSIDE · SAN DIEGO · SAN FRANCISCO

SANTA BARBARA · SANTA CRUZ

CSE 165/ENGR 140: Introduction to Object-Oriented Programming Final Projects

Angelo Kyrilov

Spring, 2018

1 Introduction

The final project in this course is designed to assess your mastery of concepts covered in class as well as your ability to work in a team. You will be asked to get into groups of up to 4 members. Individual projects will be permitted but you are encouraged to try to form a team.

There are three project topics, described in detail below. Your group is to implement a project related to one of the topics. You will be required to write a report, describing your project, and to present your project to the class. Your project will be graded based on the quality of the implementation, as well as the effectiveness of your presentation.

Please refer to the subsequent sections for a list of deliverables and their associated deadlines, as well as detailed descriptions of each project topic and the required features of each.

2 Deliverables

2.1 Project Proposal

Deadline: Friday, April 6, 2018 at 11:59 pm

The project proposal should contain the following information:

- Team name
- List of team members
- Project title and brief description
- Proposed division of labor and time plan

Only one proposal per team is required. One of the team members should upload the document to the "Assignments" section of CatCourses.

2.2 Presentation

Time and Place: Tuesday, May 8, 2018 in COB 120

Each team will have 5 minutes to present their project. The goal of your presentation should be to demonstrate that all the required features have been implemented. Each presentation should include a live demo of the code and each team member should speak. Please also highlight additional features that you have implemented in order to impress your peers and instructors.

2.3 Final Report

Deadline: Thursday, May 10, 2018 at 11:59 pm

Unlike the proposal, each team member will be required to submit a final report. Some of the sections of the report can be shared between team members while others will have to be written individually.

These are the common sections, where team members can work together to write.

- Team name, project title and team members
- A more detailed description of the project. This should include not only what the app does but also how certain features were implemented. Code snippets should be included, particularly to highlight Object-Oriented Programming features that make the code more readable, maintainable, and extendable.

Individual sections, that each team member needs to write independently.

- Time Plan and Division of Labor. Please describe if the original time plan was adhered to, and if not, what changes were necessary. Also comment on how well the team worked together, and whether the division of labor was fair.
- Lessons Learned: In this section, please write about any interesting lessons you learned while working on the project. These do not need to necessarily be about C++ or Object-Oriented Programming, they can be on teamwork, time management skills, or any other aspect related to the project.
- Additional Comments. This section is optional. Feel free to write any comment/suggestion/complaint about the project that has not been covered in the above sections.

The report should be well-written and presented. The sections should be clearly labelled and should contain full paragraphs. Please do not write everything in bullet point form and do not use one-liners such as "The team worked well together". Instead, write a story that demonstrates this.

2.4 Source Code

Deadline: Thursday, May 10, 2018 at 11:59 pm

Upload your entire project folder as a ZIP archive. Please include a text file with compilation instructions.

3 Project Topics

3.1 Video Game

Implement a 2D video game. It can be a well-known game, such as Space Invaders or Pac-man, or a game of your own invention. You are required to use the GLUT template provided in class as a starting point. If you do not wish to use this, you may use a GLUT equivalent library, not a game engine such as Unity.

Your game needs to have at least one object that moves autonomously and at least one object controlled by the player. You are also required to use textures on your objects and/or backgrounds to make your game look better. Your game should be as close to a finished product as possible. You should have things like power-ups, keeping high scores, ability to pause/reset game, ability to save game, etc ...

3.2 Algorithm Animation

Implement an application for animating algorithms. You are required to use the GLUT template provided in class, or an equivalent library. The data structures that the algorithms are operating on must be represented graphically in OpenGL. All operations performed by the algorithm, should be animated. The number of algorithms in your application should not be less than the number of members of your group.

Sorting algorithms are particularly suited to this kind of application, as the input can be represented as a histogram, where the height of each bar represents the value. Comparing two input elements to each other can be represented by highlighting the corresponding bars, and swap operations can be animated by having two bars exchange positions. Graph algorithms are also well suited for this project. Textures should be used to make the objects on screen look better.

Your application should have additional usability features. Think about a user who is trying to learn about a particular algorithm in your application. What kind of features would that user find helpful? Possible examples include the ability to pause/resume the animation, step the animation back or forward one operation at a time, the ability to provide input values, the ability to run two algorithms side-by-side for comparison purposes, etc ...

3.3 Graphics Editor

Implement a graphical editor similar to Microsoft Paint or MacPaint. Your user should have the ability to draw and/or place different shapes on a canvas. Some of the shapes may be drawn free-hand, while others are placed. The user should be able to select between different shapes by clicking on buttons. The interface should make it clear if a shape tool is currently selected, by highlighting its button. Similarly, there should be a color palette where the user can select the color of the shape to be drawn. The current color should also be highlighted. The button faces should all have textures. You can also include the ability for users to select and resize shapes that have already been drawn, an eraser tool, and anything else you find interesting.

3.4 Independant topic

If you have a different topic in mind for your project, you will have to get approval from the instructor. Please make sure your proposal is detailed enough to give us a good idea of what you intend to do.

4 Grading Policy

In addition to meeting the requirements described above, your project should demonstrate your ability to apply some of the techniques covered in class, such as object hierarchies, in order to make your code more readable, understandable, maintainable, and extendable. We will also be assessing your ability to work in a team. Different pieces of the project, that were implemented by different people, should look like they belong together. We do not want to see a project that looks like 4 separate mini-projects mashed together. There are Object-Oriented Programming techniques that we have covered in class (abstract classes for example), that make such tasks easier, so please make use of them.

You have to have running code in order to receive any credit for your project. Your code should also compile on computers other than your own. When we grade your reports (after your presentations), we will be looking at your code and compiling it. Please ensure that we are able to compile on a Mac with Xcode, on Windows with Visual Studio, or on Linux in the command line. If you make use of any outside libraries, they should be included with your submission, along with instructions of how to get your project to compile and run.

The project grade will be made up as follows:

50%	Minimum viable product
20%	Mastery of Object-Oriented Programming concepts
15%	Quality of presentation
15%	Quality of report

If you are unsure of anything related to the project, please contact the instructor.