UNIVERSITY OF CALIFORNIA, MERCED

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

SANTA BARBARA · SANTA CRUZ

CSE-165/ENGR-140: Object-Oriented Programming
Class Project

Spring 2023

Preliminary Notes

- **You are allowed to work on the project individually or as a group of up to 3 students.** Your group member(s) can be in any section. You can discuss your ideas with others outside of the group, but your submission must be yours and your group's only. Do not copy code from the internet, unless it is outside of the scope of your significant features. (If in doubt, ask us.) Be reminded of the CSE academic honesty policy!
- Your solution must be exclusively submitted via CatCourses. Email submission will not be accepted. Pay attention to the posted deadline because **the system automatically stops accepting submissions when the deadline passes.**
- **To receive credit, after you make your submission you must also demo your project to your TA in your lab section.** You must show your working program, be prepared to explain your code, and clarify what parts each member worked on (if working in a group). You can demo at any point after you submit, but you must demo before the end of your last lab of the semester. You must demo the same program you submitted on CatCourses. **You will get a 0 if you do not demo, even if you made a submission.**
- The expectation is that you work on the project outside of lab time, but you may certainly use any extra time in lab if you have already finished your other programming assignments.

Contents

1 Overview

In this project, you will be programming a C++ app of your choice, making use of the various objected-oriented programming concepts we have seen over the course of the semester.

The project has 3 main parts:

1.1 Part 1: Checkpoint

You have until the **Checkpoint deadline on CatCourses** to tell us the following: a) define your chosen topic, b) define your group, and c) tell us what each member will do (if a group project). More information follows below:

- **Choose topic**: Choose one of the suggested topics listed later. For custom app ideas, your TA will have to agree that your proposal is reasonable before you move forward. Whatever topic you choose, you must also satisfy the globally required features.
- **Define group**: You may work alone or in groups of up to 3. Members of the group may be in different lab sections. Each member of a group must be responsible for at least one "significant feature" of the project to be implemented completely by the member. External support code can be used but each member is expected to implement a significant feature of the project by himself/herself.
- **Changes**: If for any reason the plans for your project topic or group membership change, please communicate immediately to your TA, so that we know what to expect from each project.

1.2 Part 2: Submission

You have until the **main project deadline on CatCourses** to make your final submission, which should be a zip file containing: a) a project report, and b) your implementation code. Only 1 member of a group needs to submit.

a) **Report**: a short written report of at least 3 paragraphs containing an explanation of the chosen app and what it does, the significant feature(s) each member of the group was responsible for, how you used objected-oriented programming concepts in your implementation, and any additional comments you would like to make.

b) **Implementation**: all your implemented code has to be zipped and uploaded to CatCourses in a way that we can recompile your project ourselves if needed, so be sure no files are missing. Before submitting, test to unzip your package in a different directory in order to make sure it can still be compiled with no errors. If the project is too big please notify us and you may upload just the source files.

1.3 Part 3: Presentation

You have until **your last lab of the semester** to demo your project to your TA. You may, of course, demo earlier if you finish early. Do not forget to do this, as we will NOT be accepting late demo's or submissions for the project.

At the demonstration you will run and explain your project to the TA. You can use your own laptop or any other computer in the room, just be sure your application will run fine on presentation day. The entire group should be present in order for each member to explain the parts they implemented.

2 Topics

The following subsections describe the three possibilities of topics for the project.

2.1 Video Game

Use OpenGL to develop a video game. You are not expected to know or learn advanced graphics concepts, as that is beyond the scope of this class; you may simply work with basic 2D primitives. An introduction to OpenGL and its simpler immediate mode will be given in lecture, as well as support code to act as a starting point.

Examples of 2D games you can implement: Super Mario, Space Invaders, Pac-Man, Brick Out, Tetris, Bomberman, etc.

As a group project, one member can have the role of implementing the main game logic while other members can be divided to work on many other tasks: to write classes for drawing the various elements of the game, for drawing the environments of the levels, designing classes that will compute the behavior of the needed game entities, etc.

Note that because the OpenGL API itself is not object-oriented, you will be expected to make heavier use of objected-oriented concepts to compensate.

Minimum requirements:

- The game must have at least one object controllable by the user (but it is also fine to have multiple players).
- At least one object must move autonomously (independent of player input) according to game logic.
- The user must be able to interact with the autonomous object(s) in order to play the game in some way.
- You must make use of polymorphism. You should have at least one abstract class from which different types of games objects are derived, and use upcasting at some point in the game logic. (For example, in the Super Mario context: you may have an abstract Block class, from which are derived CoinBlock, BreakableBlock, UnbreakableBlock, PowerUpBlock, ... classes, which are upcasted and stored in a vector of Blocks to be used by the game logic.)

2.2 Useful App

Use Qt (or some other equivalent object-oriented framework like MFC) to develop some sort of useful app, such as a text editor/word processor (Notepad++, Word), an image editor (Paint, GIMP), a to-do list organizer (Todoist), a graphing calculator (Desmos), etc.

For Qt, refer to the tutorial: Qt for Beginners

Minimum requirements:

- User must be able to input data via a graphical user interface.
- Data must be processed with one or more algorithms.
- Data must be output in a useful format (e.g. tables, graph, charts, etc).

2.3 Custom Topic

If you have an idea for a different program that does not fit into either of the previous categories, you may propose the idea to your TA. You must define what the minimum requirements for your custom app are, similar in vein to the significant features that each member will have to implement and along the lines of the two previous subsections. Your TA will decide whether your proposal is reasonable, and only then should you move forward with this.

3 Globally Required Features

No matter the choice of topic, all projects must meet the following minimum requirements:

- You must make use of classes for the different types of objects in your program, and make proper use of constructors, destructors, accessors, and mutators.
- You must make use of inheritance in some way.

4 Extra Credit Features

4.1 Version Control Integration

Integrate version control software into your project (Git, SVN, etc.). For example, you could host your project on GitHub, and your group works together on the repository. This is extremely common practice in software projects involving teams of programmers, and so is valuable practice for you.

To receive credit for this feature, version control must have been integrated into the project for at least two weeks prior to the last submission, and contain commits from every group member.

4.2 Qt AND OpenGL

For this feature, you would make use of both Qt (or other equivalent framework) AND OpenGL in your app in some **significant way**. To receive full credit for this feature, it would not be enough to, for example, simply use Qt's OpenGL widget but not use any of the other Qt functionality.

5 Grade Breakdown

Item	**Points**
Checkpoint (group and topic information)	10
Minimum required features of the chosen app	40
Globally required features	20
Overall quality of the results	20
Written project report	10
Total	**100**

Extra Credit	Points added to final grade
GitHub (or equivalent version control) integration	+20
Use Qt AND OpenGL together in a significant way	+20