Final Project: You choose

Proposal due 9:00am Wednesday, October 23rd, 2013
Design Docs due 9:00am Wednesday, October 30th, 2013
Basic Demo due 9:00am Wednesday, November 13th, 2013
Presentations in class on November 20th & December 4th, 2013
Final Submission due 9:00am Wednesday, December 4th, 2013

Goal

For your term project, you will be implementing a GUI of your own design. By doing this, you will prove that you have mastered the core GUI programming techniques covered in class and that you can design a GUI on your own. You will also have the opportunity to bring your own interests into the class, to see how programming can be used to supplement things you already do.

I **strongly** encourage you to develop a project idea related to something connected to your own life- a hobby, a task you often do, or something related to your work. You are also encouraged to meet with me before the proposal due date in order to brainstorm or discuss ideas you already have. If you truly have no preference on what type of project you want to do, you can choose one of the project alternatives (listed at the bottom of this document). Each of these describes an B- project idea; you can then add on new features to that project in order to increase the potential grade it would get.

Your project may vary greatly from others based on what your end goals are; however, every project will need to demonstrate mastery of the following concepts from class:

- MXML component coding
- Actionscript coding
- Events and user input

You will also need to demonstrate mastery of at least 2 of the concepts listed below:

- Drawing and graphics
- Skinning
- States and transitions
- Animation

Proposal (5 points)

The project proposal is due via email on **October 23rd**. You will need to write up a 1+ page document which *briefly* describes:

- The main idea of the GUI program you plan to build (think elevator pitch)
- A schedule describing how you will break the program up into *iterative steps* that can be developed and implemented over time (including what will be done by the 'basic demo' step)
- How you will demonstrate mastery of each of your selected topics

- Any external libraries/code that you plan to use
- The main challenges you will need to face
- 'Bonus features' that you'll add if you have time

By **October 26th** you will receive feedback from Kelly regarding your project idea and what your personalized rubric will look like.

Design Documents (10 points)

Design documents are due via email (or in person for paper products) on **October 30th.** You will need to prototype your system with a method that you feel comfortable using; this could be a paper prototype, sketches, a set of storyboards, a class diagram, or something completely different! Your prototype should demonstrate most of the functionality that will exist in the basic demo of your project. This will give you a chance to thoroughly map out what your project will look like and how interacting with it will work before you move into the coding stage.

By **November 2nd** you will receive feedback on the design of your system, along with recommendations for any changes that might need to be made as you move on to the coding stage.

Basic Demo (15 points)

On **November 13th** you will need to submit a working demo of your project (as an fxp/zip file via email) which has all the basic features implemented. It can be clunky, and there can be small bugs, but the majority of the coding (as specified by your project proposal) will need to be finished at this point. Also, the code should have **some basic level of documentation** at this point so that Kelly stands a chance at understanding it. This step is in place to save you from the horrors of trying to implement everything at the last minute when you have to present the next day. Start coding early!

By **November 17th** you will receive feedback on your current progress with coding. If you want the feedback to be targeted at any specific part of the program- a feature that you're concerned isn't clear, or a bug you can't fix- specify it in a comment at the top of your main MXML file.

Presentation (15 points)

Presentations will last at most 8 minutes. Your presentation should include at least:

- A very quick (half-minute) description of your project goal
- An explanation of your design process
    - Show early design steps from your design documents
- A live demonstration of your system that clearly shows off its features
    - Be prepared for demo gremlins! Also, you may want to work shortcuts into your code to assist in presenting functionality

- A high level overview of how you structured your code. The code structuring should show good software engineering principles such as:
    - Clear commenting
    - Clean style
    - Modularization – using classes, components, etc to make the code more organized and modular
    - Use of MVC paradigm (this is optional but may get bonus points)

Think of this presentation as a chance to sell your project- you want to convince the audience that the goal is interesting, show them how well the system works, and demonstrate your complete understanding of the implementation. Your final submission will mostly be graded based on this presentation, so make sure that you cover all the interesting features that you manage to implement!

After presenting, the audience will have two minutes to ask questions while the next speaker sets up. Questions about implementation or future work are especially encouraged!

Finally, there will be two presentation days- **November 20th** and **December 4th.** Eight slots will be open for each day. Sign-ups for the slots will be done at the end of class on October 30th. Students who present on November 20th will be allowed to present features that they have not yet implemented by demonstrating with paper prototypes or storyboards; however, they must at least show their basic working demos to the class. Students who present on that day who have already implemented their extra features may be eligible for bonus points!

Final Submission

Your final submission is due via email on **December 4th** before class. This should include all of your code and design documents, and everything needed to run the program. These final submissions will be consulted when determining final grades, but the majority of the grading will be done during presentation of the program. Students who present on November 20th may, of course, have their grades bumped up due to features implemented between presentation and final submission.

Project Alternatives

Alternative #1: A To-Do list (Base Score: 80)

Create a to-do list application. There are many free to-do list applications online that you can look for inspirations in UI design (Wunderlist, Remember the Milk, TeuxDeux, Google Tasks, etc.) Your application should include these basic features:

- Allow users to enter new tasks
- Allow users to "check off" a particular task to signify that it's done. The task should become dim or have strikethrough, etc.
- Allow users to delete a task
- Allow users to edit a task

Ideas for more points:

- Group similar tasks using colors, folders or tags, etc.
- Drag and drop tasks into a trashcan
- Save/restore functionality – being able to save the tasks and open them again (involves learning how to save to and read from file).

Alternative #2: Photo browser (Base score: 80)

This project alternative is inspired by applications like Picasa, iPhoto and websites like Flicker. Your application should have at least two views: a "grid" view that lays all the photos out in a grid, and a "navigation" view that shows a particular photo and allows users to switch photos using next/previous controls. Users should be able to change the size of photos in the grid view.

Ideas for more points:

- Add animations to your user interface
- Add more views to the photo browser
- Allow users to zoom in and out of photos
- Allow users to group photos into folders

Alternative #3: Connect Four (Base Score: 80)

This project involves creating the familiar classic game: Connect Four. If you're not familiar with the rules, you can play it here: http://www.gamesgames.com/game/Connect-Four.html

Because implementing a computer player is outside of the scope of this class, you may implement a game for two human players. Your project should keep track of game variables like whose turn it is, whether the game is over (i.e. someone made connect-4 and won), and should allow users to start a new game after the game is over.

Ideas for more points:

- Add sound effects for chips falling and some one winning
- Add animation (the pieces fall down smoothly)