Introduction to Computer Science

Project 6 - Final Project

Students will design, plan, and implement a medium- to large-scale final project of their own choosing.

Overview

During this course, you have learned a huge amount about computer science and programming in general, and Snap! in particular. In this project, you will put all of that knowledge, along with some new skills you will develop around design, planning, and project management, to build a relatively large and complex application that *you* choose. You can create almost anything you want and should ultimately produce a project that is interesting, useful, and challenging.

Details

Project Phases

This project will be significantly larger in scope than any of your previous assignments, so there will be more design and planning than before. More importantly, though, rather than be given a well-defined specification, *YOU* will be setting the requirements for your project by coming up with an idea, fleshing out the details, and defining the steps necessary to complete your program.

To help you through this process, there will be several steps to this project. You must complete **all** of the steps **in order** for your project to be successful. In fact, *half* of your grade will be based not on how well your program works, but on how well you completed the design and planning process.

The phases of the project will be:

- *Brainstorming* - coming up with as many possible project ideas as you can
- *Pitching* - choosing a few ideas and developing a short description of what the project will entail
- *Review* - getting feedback from your peers and instructors on your pitches and choosing one
- *Scenario Definition* - listing out the features the project will need and what they will look like
- *Wireframing* - drawing simple sketches of what the various "screens" in your program will look like
- *Specification* - fleshing out all the specifics of how the project will work

- *Scheduling* - listing the programming tasks necessary to complete your project and estimating how long each will take
- *Development* - writing the code for your project by following the spec and schedule created in the previous steps

Progress Tracking

In phase vi, you will complete a Final Project Plan Organizer and in phase vii you will complete a Final Project Development Plan. These documents will be your guides in the development phase and will help you stay on track and aware of your progress. Throughout the development phase of the project, you will be expected to keep your spec and plan up-to-date and make adjustments as you get ahead or behind, as requirements change, or as tasks or features get prioritized. At the end of each coding day, your spec and plan documents should be updated to reflect the current state of your project, and you will check in with an instructor at least once a week to make sure things are on track.

Implementation Requirements

Complexity and Creativity

Your final project should be sufficiently complex and large-scale to push your limits as a programmer, but not so sophisticated that you are not able to complete it in the time allotted. The complexity in your project should come from the *design* and the *algorithms* and not from the *code*. (That is, you cannot meet the complexity requirement simply by writing a lot of code. Your code must be challenging or interesting in some meaningful way.) In addition, you should not add complexity by introducing peripheral elements, such as graphics or sound effects. (Your program can certainly have these, but they will not be considered in determining the projects complexity.)
In addition, one of the main goals of this project is to allow you to unleash your creativity and allow you to create something of interest to you. To achieve this, your project must show some level of creativity or personalization that makes it your own. Simply creating your own version of some existing application will not fully meet this requirement.
For both the complexity and creativity requirements, you should talk to the instructors early and often to ensure your project is in line with our expectations.

Documentation and Style

As with all previous projects, your program must be well-written, well-documented, and readable. Writing code with good style is always a good idea, but in a project of this size and scope, following style guidelines will help you keep your thoughts organized and make it easier to keep track of your progress, pick up where you left off each day, and find and fix bugs. In particular, though this is certainly not a comprehensive list, pay attention to the following:

- organizing your scripts so that they can be read and comprehended easily
- giving your sprites, variables, lists, and custom blocks descriptive and meaningful names
- using the right type of variable (global, local, sprite) for each situation
- including comments to describe the structure of your program and track your progress
- avoiding redundancy with good use of loops, custom blocks, and/or lists
- practicing good procedural decomposition and abstraction

Required Snap! Elements

In order to show that you have fully mastered all the skills from the course, you project must include at least the following:

- a clear way to start the program, and clear prompts or instructions for any user interaction
- at least one loop, variable, custom block, and list, and more as necessary or appropriate
- each these must be used correctly and meaningfully– creating a list that contains a single element just to meet this requirement will not earn points
- at least one user interaction
- this can be prompting for information using ask, responding to key presses or mouse movements, or any other action that keeps the user involved

Required Checkpoints

At least three times during the project period, and at least once each week, you should check in with an instructor to ensure that your project is on track, that you are meeting the project requirements, and that you have the answers to any questions that might have arisen during your work. The course staff will work with you to set up a schedule for these checkpoints, but it is **your responsibility** to ensure that the meetings take place.

Introduction to Computer Science

Grading Scheme/Rubric

Design Phases	
Brainstorming	2 points
Project Pitches	6 points
Feature List	4 points
Wireframes/Sketches	4 points
Project Organizer (Specification)	8 points
Implementation Plan	8 points
Spec and plan are updated throughout project	8 points
Total	*40 points*
Implementation	
Project is appropriately complex and creative	8 points
Program is well-documented and shows good style	4 points
Program uses Snap! elements effectively, including all required elements	8 points
Final product meets all requirements and goals laid out in spec	8 points
Checkpoint 1	4 points
Checkpoint 2	4 points
Checkpoint 3	4 points
Total	*40 points*
Total	***80 points***