Project 3: **Universal App**
GUI Programming
100 points

Teams of two are to develop a Universal app that is non-trivial and implements the given requirements. The app can be a game, utility, or educational in nature. Some examples include:

- Tetris game that saves high scores
- Painting app that loads and saves pictures
- Door prize program that uses XAML animations
- Movie app that uses a web API to obtain movie data
- Super Hero database app that stores hero information in a SQLite database

Project Proposal

Each team must submit a typed project proposal by **Friday, Nov 10 at 4:00 pm**. Drop the proposal off at my office. You may propose more than one idea if you'd like. The instructor must accept your proposal before you begin developing. Your proposal should include the following:

1. Names of teammates
2. Description of what the app will do (one paragraph or more, bulleted list is fine)
3. A drawing or screenshot of the app's primary UI

The proposal counts as a homework score. Proposals that are well-written with a clear description and well-thought out UI will receive full credit.

Class Presentations

Each team will have 5 minutes to present their app to the class on the final week of class. I will clone your projects from GitHub on the podium computer so they can be quickly shown to the class.

Although the 5 min presentation time is very fast, do your best to impress! The impression you make in your short presentation will affect my evaluation of your app. Show how your app can restore its state during the presentation.

Requirements

All apps must implement the following:

1. Be composed of multiple pages, one of which describes what the app does and who programmed it
2. Has a custom icon and startup screen
3. Resizes appropriately so the UI is usable on various monitor sizes. Does not need to resize smaller than 600 pixel width.
4. Saves its state so it picks back up where the user left off even if it is terminated (Alt-F4)

Give special attention to usability issues. Make your application as intuitive as possible, and avoid making GUI bloopers. Make your app easy to figure out for a first-time user.

Teams

You may form your own two-person teams. If you are unsure about who to work with, you could post your idea for a project on Canvas and see if anyone would like to work with you.

Teams should create a private GitHub repo and regularly save their work to the repo. Add me as a contributor to your repo (username: fmccown). Your source code on GitHub should not be modified after the deadline to avoid late penalties. It should include all the images and sound files used to create the project. I will clone your repository on my machine to grade it. One teammate should submit the team's README file to Easel before the deadline; there is no need to submit anything else to Easel.

The repo's README file should contain the following information:

1. Link to the repo on GitHub
2. Brief summary of what the program does
3. List of any known bugs that still remain
4. List each teammate's contribution
5. The percentage of work performed by each teammate

Ideally the work performed by each teammate will be 50/50, but if one teammate does more than the other, the percentages should be adjusted accordingly. The percentages will be used to determine each teammate's final grade on the project. Note that the repo history on GitHub should be a fairly accurate indicator of how much work each teammate did.

Grading

Each team will meet with me for 20 minutes the final week of class. I will ask you questions about your project like why certain design decisions were made, what do you wish you had done differently, and how much effort was put into the project. The effort put into this project should at least match the effort put into previous projects.

Other criteria:

1. Does your app implement the goals set forth in the proposal?
2. Does your program make any GUI bloopers?
3. Is your program well written? Did you use proper C# naming conventions for variables, methods, and classes? Did you use constants and functions when appropriate?
4. Did you clearly cite any code that you found online?
5. Did you use git adequately when developing your solution? There should be at least 20 commits in your repo, and each person must have committed at least twice.