Final Project

The final project for this course is to create a program using the C# Windows Forms library that can create SVG files.

In its simplest form this project could just be a simple version of MS Paint that exports SVG files. If you are feeling more ambitious or creative, you can add different features or change the general purpose of the application.

For inspiration, and to find solutions to some common coding problems take a look at: https://github.com/cdiggins/svg-editor.

Additional ideas for projects:

- A font editor
- A text editor
- A raster image editor that uses SVG operations
- A web-page layout tool
- A flip-book animation tool
- Creates and exports animations created by editing SVG
- An AI assistant that converts freehand SVG drawings into art work (or vice versa)

Required Functionality:

- A main menu with entries that support shortcut keys
- A picture control that displays the image being created
- A feature that allows export file to SVG
- Support mouse and keyboard interaction
- A method to save and recover work that the user is performing
- Some kind of “selection” capability - either to change properties, or as least to delete visual elements
- Ability to undo the last actions

Submission

Required deliverable:

- Public GitHub repository.
- Exe file can be downloaded from the “releases” section.
- Readme
    - describes what the program does
    - what its features are

    - what technologies or libraries it uses (acknowledgements are very important)
    - screen shots
    - List of bugs and known issues
    - To-do
- Summary report – 2-3 page word document.
    - Describe what changed from the initial plan, what turned out to be easier or harder than expected, what features had to be changed or dropped. Anything that you learned

Grading

Your project graded based on the quality of the code and documentation, and the complexity, usability, and robustness of the features. The grade will be divided into two parts (code and application/report). I will be using the following criteria when considering the grade.

- **50% - Code**
    - The code is well structured.
    - Follows standard C# coding conventions.
    - Code has tests.
    - No redundant or useless code
    - Code is easy to read and understand.
    - Documentation of major classes.
    - Code is organized logically into files, folders, and libraries.
    - Names of files, classes, and functions are descriptive.
    - Code is maintainable and can be easily modified.
- **50% - Application / Report**
    - Readme is complete, accurate, and well-written.
    - Application is documented adequately in read-me.
    - Is self-explanatory and easy to use.
    - Functionality - how many features does it have? Are they complex or simple?
    - Robustness – are there few bugs, and does it not crash easily?
    - Does it behave as one would expect, and as documented?
    - Report is well-written and complete.

Optional Feature Ideas

The following is a list of ideas for optional features.

- Tool bar
- Print dialog
- Export - Save to other format (e.g., PNG/BMP)
- Resizing of elements
- Flpping orientation of elements

- Drawing elements with the mouse
- Load and embed a rasterized image (e.g., as background) from different formats
- Draw shapes
- Draw text
- Create a grid in the background to help with alignment
- Zoom in/out
- Customization / Configuration / Settings / Personalization - remembered through file
- Show in browser - launch a browser
- Lighting feature - allow user to defined a light source and properties (e.g., intensity, color, direction) that affects entire picture of part of it
- Help system - explain menu items and/or list attributes of elements that can be edited
- Apply built-in filters (blur, distort, noise, etc.)
- Choose from sample or pre-defined shapes
- Text based editing of properties
- Shape drawing tool using mouse
- Cut / Copy / Paste - of elements from within
- Cut / Paste - text or images from other applications
- Drag / Drop - of text of images from other applications
- Layer editor - the ability to create and switch between layers
- Multiple tabs - ability to edit in
- Hex-code support for colors
- Change selected shape properties after drawing was completed
- Rotation of elements
- Grouping / ungrouping of elements (aka connecting of elements)
- Changing the “z-order” of elements to control what is drawn on top of what
- Eraser feature - delete specific elements by clicking
- Undo/redo
- Visualize the SVG code created
- Predefined shape library (assumedly as SVG)
- Freehand shape drawing
- Multiple art-boards in workspace: each one edited and exported as a standalone SVG
- Predefined size and proportion for graphic design (e..g, 8.5 x 11)
- Eyedrop feature
- Drawing bezier curves
- Customizable brush (dotted lines, crayon, etc.)
- Randomized shape drawing, or randomizing properties
- Changing brush while drawing
- Drawing shapes using the keyboard arrows
- Rounded shapes
- 3D
- Control dimensions via keyboard
- Gradients for fills
- Recent files list
- File logger

- Action recorder

Advanced features

The following are examples of features that could be considered to be of higher complexity and would require more effort to implement. Any feature could be made in such a way so as to required additional work. It will judged on a case per case basis.

- Load SVG
- Edit the SVG text manually
- Collaborate - allow other people to connect to your system
- Add-in system (extend with special shapes defined externally)
- Snap to grid
- Spell checker
- Open and edit existing SVG
- Ruler / guidelines
- Lasso select tool
- Search for and group or select items by property (color / shape)
- Properties dialog: click on any object and see customizable properties
- Customized
- Use shaders
- Shape fixing/normalization/auto-completion
- Shape to AI
- Duplicate shapes while moving mouse (trail of shapes)
- Auto-designer
- 3D shapes