|  |
| --- |
| Bishop’s University |
| CS321 |
| Final Project |

|  |
| --- |
| Name: TANG Chun Wai Wilson(002349732)  Due Date: 2023/4/26  No of Pages: 6 (Not including this page) |

Content

Topic selection

Program implementation

Accomplished features

Reasons of shortcoming

Implementation log

**Topic selection**

The original thought on this final project is a lightweight SVG font editor, which includes loading characters from font files (True Type Font, TTF or Open Type Font, OTF) as SVG paths, applying some modifications such as color changes, transformation and filters and writing back the results in its original format (TTF or OTF).

After substantial research on the topic, it was found out that the last part of the above description is the most difficult task to perform. In order to write back a font file, a update of the font cmap table is required and, practically, such mechanism is difficult to access and manipulate. Moreover, even though svg paths can be used in a font cmap table to describe the ‘glyph’ of a character, some of the svg filters cannot be applied to the paths[[1]](#footnote-1) and it reduces the practicality of the program and unfortunately the ‘write back to font file’ feature must be discarded.

However, based on the research on the first two tasks, it is believed that, by changing the output file format to svg, a SVG Banner Editor is achievable.

The specification of the program becomes a SVG Banner Editor. User can choose one set of fonts (3 font types from open source is provided in developing phrase) and a color, then type some characters (a slogan) as input. SVG Banner Editor will transform the input into svg paths in xml format. User can apply the following to these svg objects:

1. Basic attributes: fill and stroke
2. Transformation: translation, rotation, skewing and scaling.
3. Filters: blur, shadow, gradient

In the meantime, the parameters of the effect can be adjusted in the application user interface (UI). The application will also follow the specification from the guideline, which includes:

1. Tool bars
2. File handling: Open, Save…
3. Undo/Redo

**Program implementation**

SVG Banner Editor is implemented in C#, using winforms, and with the reference of a demo program from the professor. The program makes use of two publicly available libraries in GitHub: watertrans/Glyphloader[[2]](#footnote-2) and svg-net/SVG[[3]](#footnote-3). Glyphloader allows characters turn into glyphs and svg’s and SVG provides all other functionalities to modify svg paths.

The data model of the program is a set of svg paths, in which each path describes one character, in extensible markup language (XML) format (i.e. tags). Xml can be regarded as a superset of svg and is chosen because it is well supported by Visual Studio and internet resources (it is found out that, however, the learning curve of mastering xml is very steep at the very late stage of implementation.) And contrary to the demo program, which depends on user’s mouse actions over a pictureBox, SVG Banner Editor takes ‘text’ as user input and depends more heavily on xml manipulations. Such choice can also avoid any data conversion in saving user’s progress.

SVG Banner Editor makes use of the SVGViewer from SVG library. The viewer renders svg images in the PictureBox automatically when svg text is provided on the designated TextBox. With this layout, the implementation of the program would involve:

1. Data model:
   1. Design structure
   2. Implement corresponding properties and methods
   3. Implement features mentioned in last section
2. Modify UI:
   1. Design layout
   2. Arrange event handling

**Accomplished Features**

1. User can choose font file and color.
2. Characters will be transformed into svg and rendered.
3. Each character contains an individual <id> attribute for further reference.
4. A Clear button to clean up pictureBox and textBox.
5. The characters will become hollow with a button.
6. Blur effect can be applied with a button.
7. Mouse position can be tracked within the pictureBox.
8. Girds can be turned on and switched off.
9. Some operations on XML that should be useful in further implementation.

**Reasons of shortcoming**

Currently, SVG Banner Editor is far from its release state. The program, unfortunately, is not implemented to its full potential and missing several required features. Such delays are due to:

1. Poor time management

Due to other academic commitments, 5 full days of coding is thought to be sufficient for this program.

1. Unfamiliarity with winform

The interactions among different forms are difficult to achieve. The original plan is to separate SVGViewer into two forms and add another form for control. However, passing data, which may involve static/non-static object or delegates, becomes an issue and was found out it is difficult to implement.

1. Unfamiliarity with XML and Linq

At first glance, xml is easy to manipulate due to its text-base nature. It turned out, however, reading xml string as XDocument and using Linq are more complicated than it seems. Microsoft documentation[[4]](#footnote-4) is over-detailed and is not very applicable to this case. Majority of the coding time became figuring out the difference between XDocument[[5]](#footnote-5) and XElement[[6]](#footnote-6) and little was accomplished.

In the end, a new attribute can be created and removed from a xml string, existing attribute can be modified. But further studies are required to compare attribute values.

1. Poor organization

Too much attention was focused on the ‘winform’ aspect of this project and the data model design was ignored until latter stage. As mentioned, passing data among forms became problematic and it is believed that if the data model is thorough considered and well designed, such issue can be avoided.

Conclusion

This SVG Banner Editor demonstrates the ability to create svg from characters and such svg can be manipulated by adding, removing or modifying corresponding tags in xml. When completed, it will be able to become a handy tool to experiment and generate text in svg with a range of filters provided by svg standard.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Log for implementing SVGbanner | | |  |  |  |  |
| No. | Task | Details | Start date | Hours spent | Note/Outcome | Reference used |
| 1 | Feasibility Studies | definiton of fonts | 2023-04-11 | 6 |  | Please see technical spike |
|  |  | rasterization/glyph |  |  |  |  |
|  |  | possibility of glyph to SVG |  |  | library(Watertrans) found to perform such task |  |
|  |  | update ttf/otf files |  |  | almost impossible to write back ttf/otf |  |
| 2 | C# winform | setup menuStrip + statusStrip | 2023-04-15 | 2 |  |  |
|  |  | keep track of mouse position |  |  | require to update event handler | prof's demo program |
|  |  | add grids to picturebox |  | 2 |  | https://stackoverflow.com/questions/12710390/drawing-a-grid-on-a-picturebox |
|  |  | load font to combo box menu | 2023-04-17 | 3 | access to winform object property from main() | https://stackoverflow.com/questions/41084457/c-sharp-adding-txt-file-names-to-a-combo-box |
|  |  | add input text box |  | 1 | check key press | https://stackoverflow.com/questions/3558814/net-textbox-handling-the-enter-key |
|  |  |  |  |  |  |  |
| 3 | Data model | implement object of glyph | 2023-04-18 | 2 | better to inherit svg object from svg library |  |
|  |  | convert input text into individual glyph |  |  |  |  |
|  |  | NUnit test setup |  | 1 | need to change target framework |  |
|  |  |  |  |  |  |  |
| 4 | Presentation of data | modify picturebox to SVG viewer | 2023-04-19 | 1 | modify code such that picturebox can read svg text | SVGViewer from SVG-net sample code |
|  |  | transfer textbox to another form | 2023-04-20 | 4 | constant stack overflow: class inheritance issue |  |
|  |  | handle xmlwriter | 2023-04-22 | 10 | use xmlwriter to update svg objects | https://learn.microsoft.com/en-us/dotnet/api/system.xml.xmldocument?view=net-7.0 |
|  |  |  |  |  | - moving node and adding attributes |  |
|  |  |  |  |  | - linq XDocument/XElement/XAttribute provide easier coding | https://learn.microsoft.com/en-us/dotnet/api/system.xml.linq.xdocument?view=net-7.0 |
|  |  |  |  |  | - distinguish XElement and Xdocument |  |
|  |  | moving data between forms | 2023-04-24 | 6 | may require event handlers and may cause exceptions | https://stackoverflow.com/questions/7886544/passing-a-value-from-one-form-to-another-form |
|  |  |  |  |  |  |  |
| 5 | Functionalities | change 'fill' attribute value | 2023-04-25 | 4 | set attribute |  |
|  |  | translation |  |  |  |  |
|  |  | resize |  |  |  |  |
|  |  | rotation |  |  |  |  |
|  |  | apply svg filter | 2023-04-25 | 1 | add new xml element |  |
|  |  |  |  |  |  |  |
| 6 | File handling | record user action (Undo/Redo stack) |  |  |  |  |
|  |  |  |  |  |  |  |
| 7 | Export |  |  |  |  |  |

1. <https://learn.microsoft.com/en-us/typography/opentype/spec/svg> [↑](#footnote-ref-1)
2. <https://github.com/watertrans/GlyphLoader> [↑](#footnote-ref-2)
3. <https://github.com/svg-net/SVG> [↑](#footnote-ref-3)
4. <https://learn.microsoft.com/en-us/dotnet/api/system.xml.linq?view=net-7.0> [↑](#footnote-ref-4)
5. <https://learn.microsoft.com/en-us/dotnet/api/system.xml.linq.xdocument?view=net-7.0> [↑](#footnote-ref-5)
6. <https://learn.microsoft.com/en-us/dotnet/api/system.xml.linq.xelement?view=net-7.0> [↑](#footnote-ref-6)