Animate++ Project Status Document

Wode "Nimo" Ni - wn2155@columbia.edu
Xuanyuan Zhang - xz2580@columbia.edu

Goals

`0.8` Version

- A simple SVG loader and exporter
  - Testing criteria: the system can load an SVG into our own data structure, and then exported as another SVG file that is restored to have the same effect as the initial SVG.
  - Challenges: Interfacing with the parser/generator library, design of the Shape classes
- Basic animation: translation, rotation, and scaling
  - Testing criteria: Generate an SVG file with valid syntax and animated output
  - Challenges: there are many ways to animate SVG files. For example, we could either embed tags, or generate a separate CSS file that binds elements in the SVG canvas and apply transformations on them.

`1.0` Version

- More detailed Styling of objects: gradients, fonts, patterns, strokes and fills
- Support for higher-level composition of SVG animation. For example, provide grouping of geometries and animating objects as a group
- Complex animations: object traveling along Bezier curves

`1.2` version

- Extend the library to allow import and export of `<canvas>` elements
- A fast renderer for preview
- Support some form of direct manipulation editing and update the modification in the UI back to the code

Progress

`0.8` Version: all clear

- XML parsing under the support of pugi library
- SVG loader supports all common shapes individually including
  - Rectangle
  - Circle
  - Ellipse
  - Line
  - Polyline
  - Path
  - Bezier curve
- Compound shape with any number of individual objects can also be loaded
  (a code snippet of how we load any type of shape is shown in below).

```
// load SVG
ShapePtr g = load(in_path);
// output the object to an SVG file
g->save(out_path);
```

- Valid SVG file can be generated.

```
<?xml version="1.0"?>
<svg version="1.1" xmlns="http://www.w3.org/2000/svg">
        <g>
                <rect x="10" y="10" rx="0" ry="0" width="30" height="30" fill="transparent" stroke="black" stroke-width="5" />
                <rect x="60" y="10" rx="10" ry="10" width="30" height="30" fill="transparent" stroke="black" stroke-width="5"
                <circle cx="25" cy="75" r="20" fill="transparent" stroke="red" stroke-width="5" />
                <ellipse cx="75" cy="75" rx="20" ry="5" fill="transparent" stroke="red" stroke-width="5" />
                <line x1="10" y1="110" x2="50" y2="150" stroke="orange" stroke-width="5" />
                <polyline points="60 110 65 120 70 115 75 130 80 125 85 140 90 135 95 150 100 145" fill="transparent" stroke="
                <polygon points="50 160 55 180 70 180 60 190 65 205 50 195 35 205 40 190 30 180 45 180" fill="transparent" str
                <path d="M 20 230 Q 40 205 50 230 T 90 230 " fill="none" stroke="blue" stroke-width="5" />
        </g>
</svg>
```

- Basic animations including rotation, scaling and translation of all basic objects have been accomplished. Users are free to manipulate the existing objects in their ways of favor. (An example of how we edit our circle animation is shown by the c++ code snippet in below)

Here is the SVG before editing

```
<circle cx="100" cy="100" r="100" fill="yellow" opacity="0.5">
</circle>
```

Our code to modify animation

```
c.animate.translate(Point(100, 100), Point(0, 200), true)
         .duration("2.5s")
         .loop(true);
```

The SVG after editing

```
<circle cx="100" cy="100" r="100" fill="yellow" opacity="0.5">
    <animateTransform attributeName="transform" type="translate" dur="2.5s" from="100 100" repeatCount="indefinite" to="0 200"
</circle>
```

`1.0` Version: all clear

- Fonts, fills and strokes are all completed(shown in the code snippet in below)

```
Circle c(100, 100, 100);
c.attr({
    {"fill", "yellow"},
    {"opacity", "0.5"}
});
```

- Gradients and patterns are special entities in SVG standards, which require building and referring to xml references. We have utilized it in our logo, but up to now there is no general support for these features.

- Complicated animation has also been accomplished. In the example given below, we load a tiger SVG as a combination of more than 200 beizer curves with over 700 lines of code. The whole tiger head can rotate and translate.

- Path class completed. Can now modify all different kinds of paths including line, quadratic curve, bezier curve and arc.

1.2 Version Something done.

- Motions along all arbitrary paths are achieved. Also, any object, simple as `Circle` or complex as large `Group`s, can all be animated in the same manner
- We attempted generating `<canvas>` code. However, there seemed not be be any accessible JavaScript generation library written in C++, which is quite the contrary from the SVG case. Therefore, maintaining syntactical correctness is significantly more challenging than SVG. Therefore, we did not release our attempt.