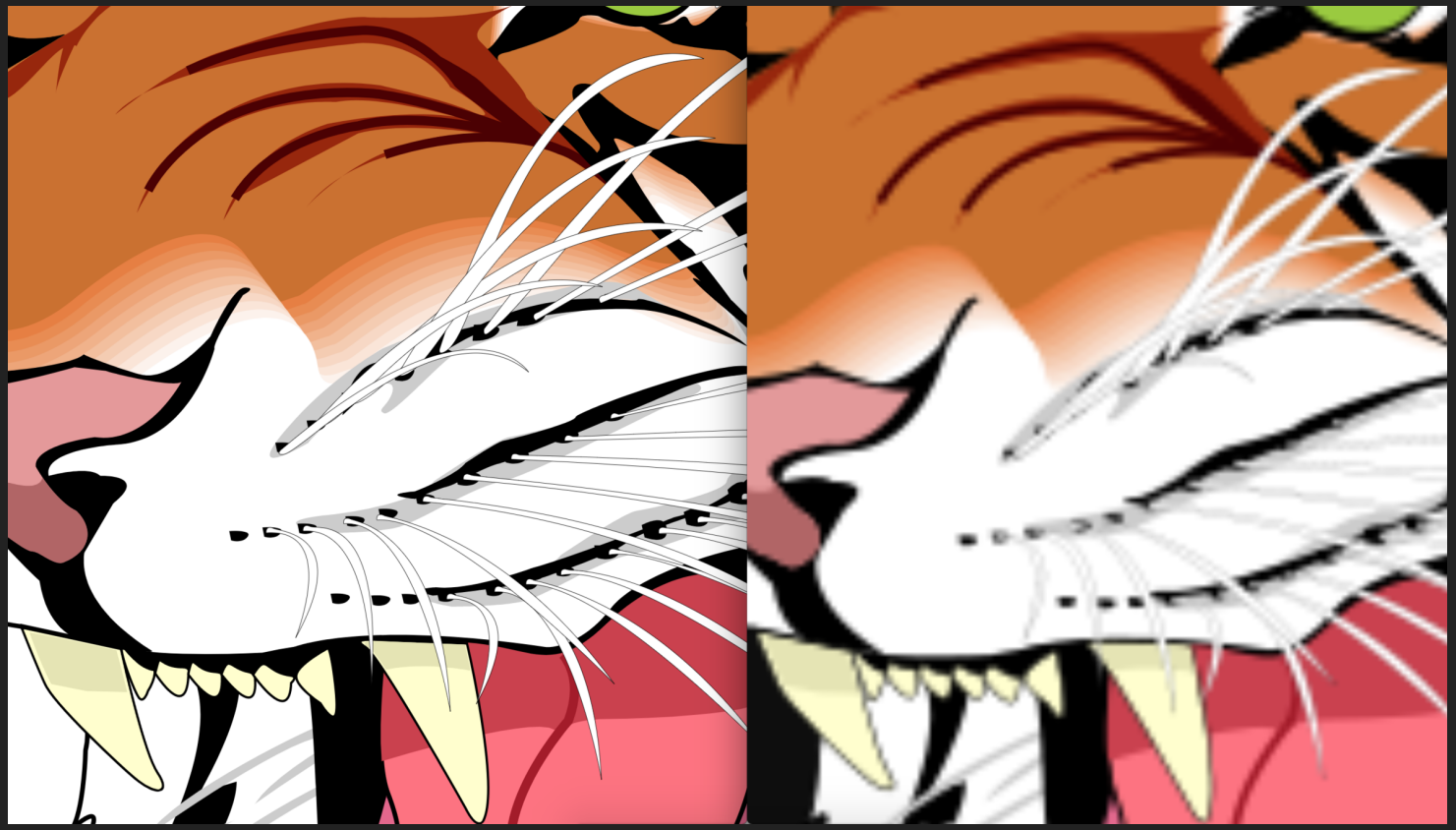
Animate++

Wode "Nimo" Ni
Xuanyuan Zhang

A vector graphics animation library written in *modern* C++.

Why vector graphics?

Small, Scale-invariant



How to animate?

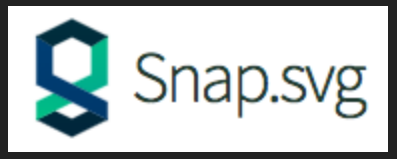
```
<?xml version="1.0"?>
<svg version="1.1" xmlns="http://www.w3.org/2000/svg">
        <rect x="20" y="20" rx="3" ry="3"
                  width="100" height="100"
                  fill="red" stroke="black" stroke-width="5"
                  id="rect_8">
                <animateTransform
                        attributeName="transform"
                        type="rotate"
                        dur="10s"
                        from="0 100 100"
                        repeatCount="indefinite"
                        to="360 100 100" />
        </rect>
</svg>
```

SVG + JS, SVG + CSS, **SVG + SMIL** (self-contained!)

How people create SVG

Javascript is the first choice

- It is born for front-end
- easy to hand on



But Why we stick with C++ ?

Advantage of C++

Too much...

For this specific task:

1. Restrictions on rampant SVG styles.
2. Utilizing OOP features for better design purpose.
3. For those who love C++ and SVG.

Existing projects in C++ ?

No.

There has not been any library in C++ focusing on general purpose SVG manipulation and animation

SVG++: SVG parser mainly for import and export.

Goal

Provide intuitive abstractions for reading, modifying, composing, and **Animating** SVG files!

Library structure

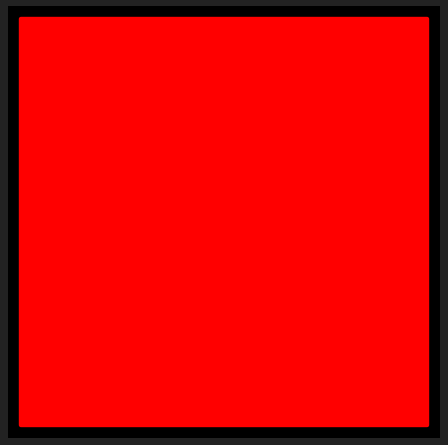
- <animate.hpp>
  - <anipp/parser.hpp>
  - <anipp/shapes.hpp>
  - <anipp/utils.hpp>

(Depends on pugixml 1.9 to parse raw XML files)

Basic Shapes - Rectangle

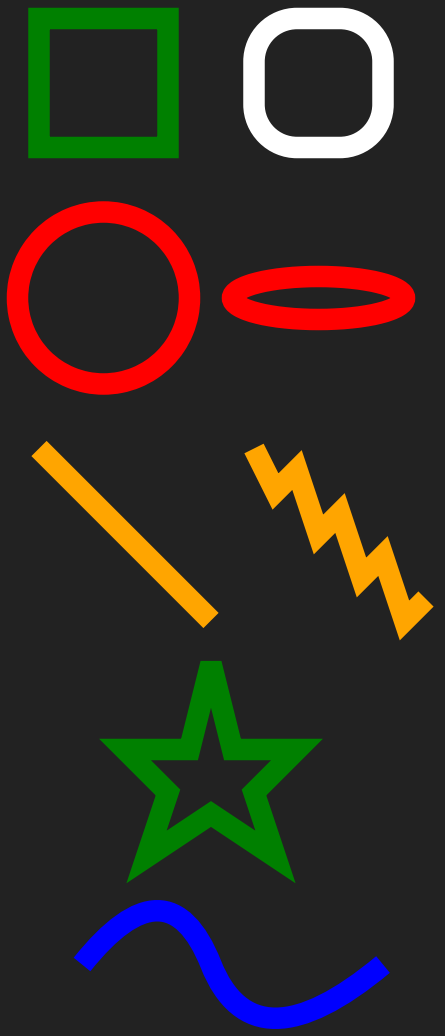
- Standard SVG API:
  - (x, y, width, length, *rx, xy*)
  - *rx, ry* for rounded corners
- Free attribute specification
  - nested list initialization
  - `std::map`

```
Rect r(20, 20, 100, 100, 3, 3);
r.attr({
    {"fill", "red"},
    {"stroke", "black"},
    {"stroke-width", "5"}
});
```



Other Basic Shapes

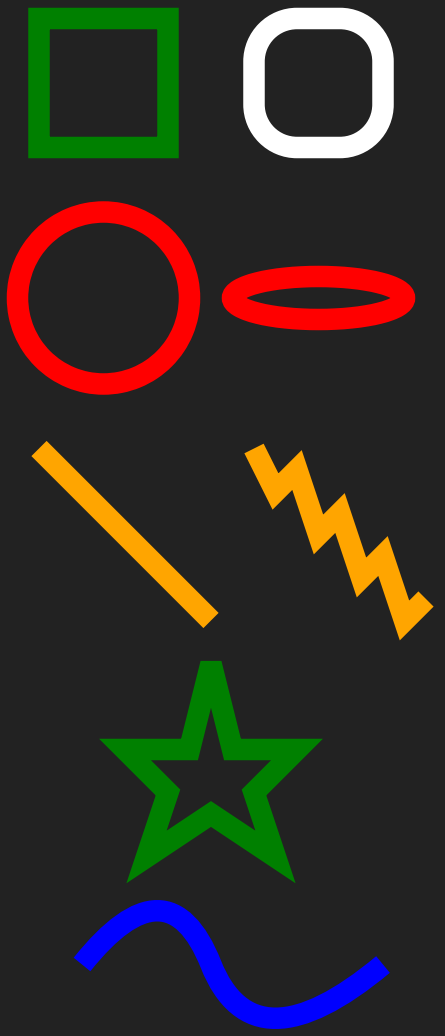
- API similar to Rectangle
- Other shapes supported:
  - Polygon: (x, y)+
  - Polyline: (x, y)+
  - Text: <any-string>
  - Line: (x1, y1), (x2, y2)



Other Basic Shapes

- API similar to Rectangle
- Other shapes supported:
  - Polygon: (x, y)+
  - Polyline: (x, y)+
  - Text: <any-string>
  - Line: (x1, y1), (x2, y2)

What's this?

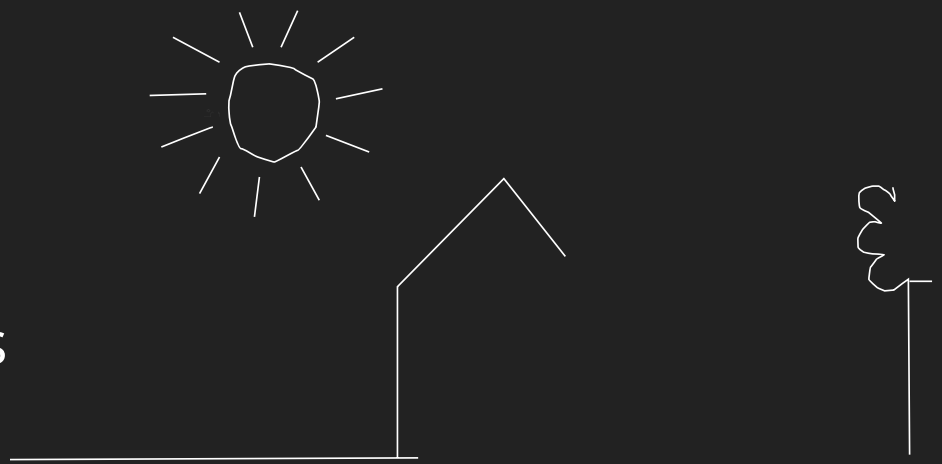


Basic Shapes - Paths

"Give me a good library and with Bézier curves I will draw the whole world"

-- vector graphics designers

- Full support to parse and compose:
  - Lines
  - Quadratic and Cubic Bézier curves
  - Elliptical curves
  - closePath



Path Commands

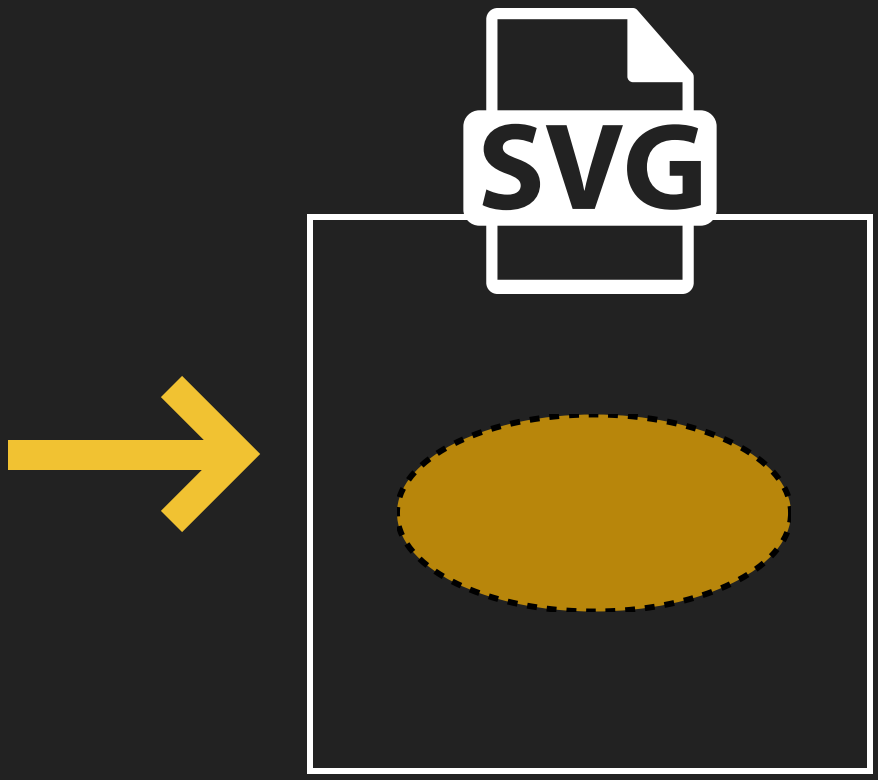
```
Path& moveTo(double x, double y, bool relative=false);
Path& lineTo(double x, double y, bool relative=false);
Path& quadraticCurveTo(double cpx, double cpy, double x,
    double y, bool relative=false);
Path& cubicCurveTo(double cp1x, double cp1y, double cp2x,
    double cp2y,  double x, double y, bool relative=false);
Path& arcTo(double rx, double ry, double x_axis_rotation,
    double large_arc_flag, double sweep_flag,
    double x, double y, bool relative=false);
Path& closePath();
```

How to draw a heart

```
Path p;
p.moveTo(121, 251)
 .cubicCurveTo(-25, -80, -50,-80, -100, -130, true)
 .arcTo(70, 70, -45, 0, 1, 100, -100, true)
 .arcTo(70, 70, 45, 0, 1, 100, 100, true)
 .cubicCurveTo(-50, 50, -75, 50, -100, 130, true);
p.animate.blink(2);
p.attr("fill", "DeepPink");
```

File I/O - Output

```
Ellipse e(100, 50, 100, 50);
e.attr({
    {"fill", "blue"},
    {"stroke", "black"},
    {"stroke-width", "5"},
    {"stroke-dasharray", "5, 10"}
});
e.save("sample");
```



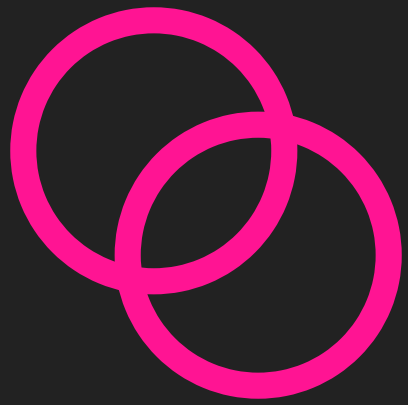
File I/O - Input

One liner!

```
ShapePtr tiger = load("tiger.svg");
```

Grouping

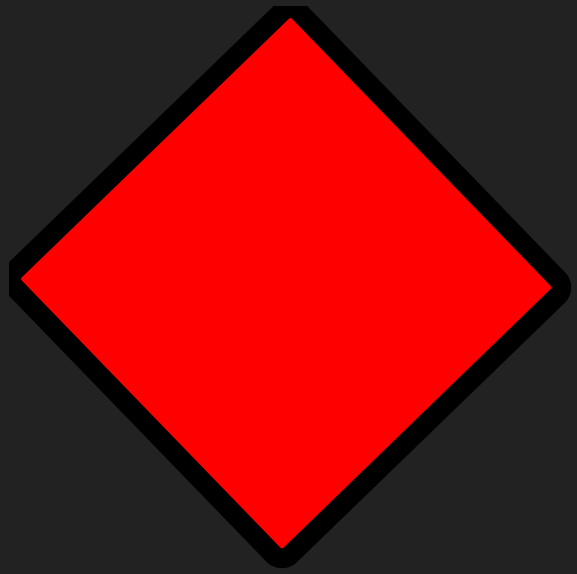
```
Circle c1(40, 40, 25);
Circle c2(60, 60, 25);
Group g(c1, c2);
g.attr({
    {"fill", "none"},
    {"stroke", "DeepPink"},
    {"stroke-width", "5"},
});
```



- Large SVGs contain multiple shapes and nested group structure
- Intuitive syntax to build groups
- memory management: ShapePtr

The Animator

```
Rect r(20, 20, 100, 100, 3, 3);
r.attr({
    {"fill", "red"},
    {"stroke", "black"},
    {"stroke-width", "5"}
});
Point center(70, 70);
r.animate.rotate(center, 0, center, 360)
         .duration("10s")
         .loop(true);
```



The animator object: obj.**animate**

Similar to D3.js, **method chaining** of basic functions

Animation

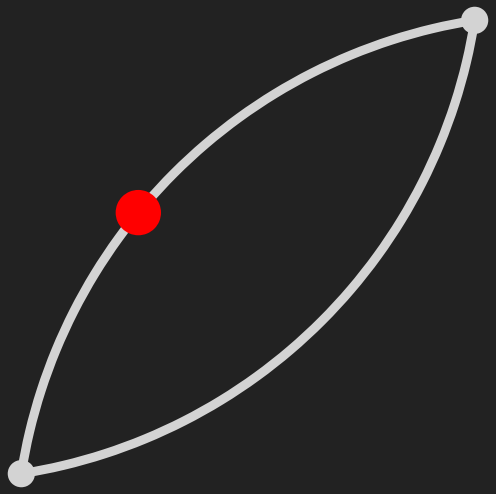
Animation class

Basic Animation

Complex movement

Composition of Animation

Possible: user defined animation



Library design: OOP

- Inheritance: shape classes
- export_SVG: example of OOP

```
xml_node Group::export_SVG(xml_document& doc) {
    auto group = doc.append_child("g");
    for(auto& shp : this->shapes) {
        auto node = shp->export_SVG(doc);
        group.append_move(node);
    }
    this->export_attributes(group);
    return this->add_animations(doc, group);
}
```

Testing

Visual test cases: I/O

Visual test cases: composition of SVGs

Modern C++ features

Fold expression!

```
template<typename... Args>
Group(Args&&... args)
{
    (this->shapes.push_back(args.clone()), ...);
}
```

- At some point in the project...
  - <optional>
  - std::any_of, std::none_of

The Wishlist

- Concept
- literal for units: *em, in, %, mm*
- More SVG features
  - <LinearGradient>
  - <pattern>
  - <use> and <set>

...

Demo

(you have ***already*** seen it)

Animate++