THE UNIVERSITY OF WESTERN AUSTRALIA

Achieve International Excellence

Number Representations and Precision in Vector Graphics

Implementation of an Arbitrary Precision SVG Viewer

Sam Moore

Supervisors: Tim French, Rowan Davies

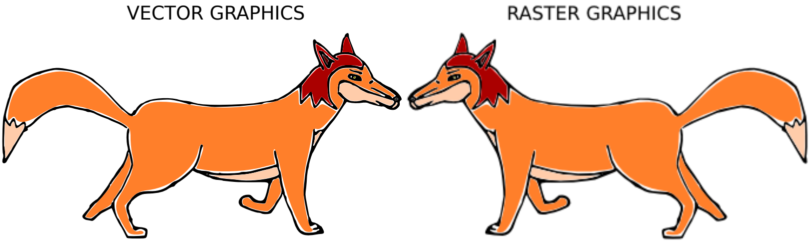
October 13, 2014

Summary

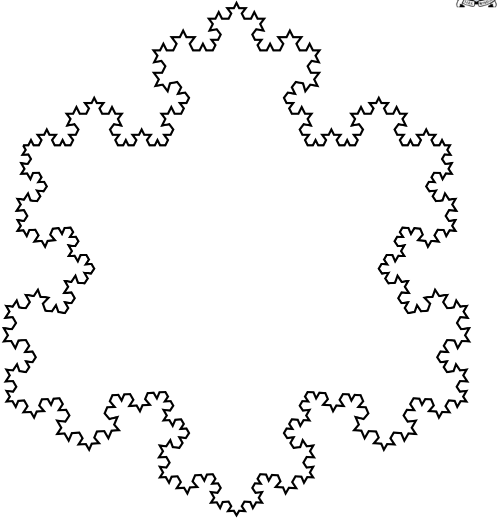
- Vector graphics allow detail to be scaled but not by an arbitrary amount
- We've implemented a vector graphics viewer which does allow arbitrary scaling

Graphics Formats

- Document formats (eg: PDF and SVG) are formats for vector graphics
- Vector graphics scale better than raster graphics



Is there a zoom limit?



Is there a zoom limit?

- SVG, PostScript, PDF specify IEEE-754 *single* floating point number representations
- Range of values: $\approx 3 \times 10^{-38} \rightarrow 3 \times 10^{+38}$
- Rough Floating Point Definition[1]:

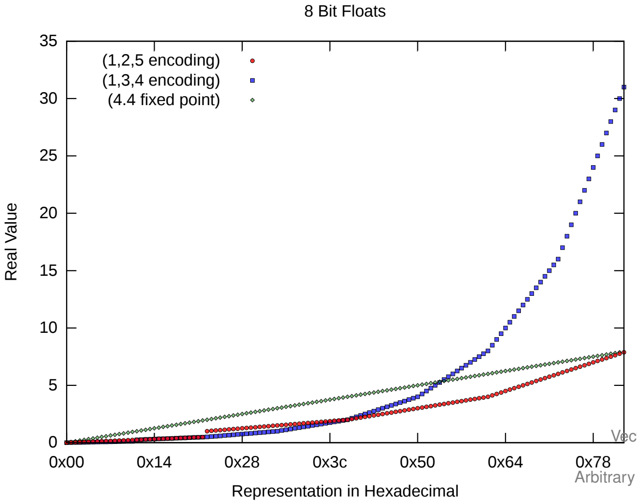
$$X = m \times 2^{E} \tag{1}$$

- *m* and *E* are encoded in a *fixed length* string of bits
- Floating Point ≈ Scientific Notation for computers

[1] IEEE-754 is more complicated

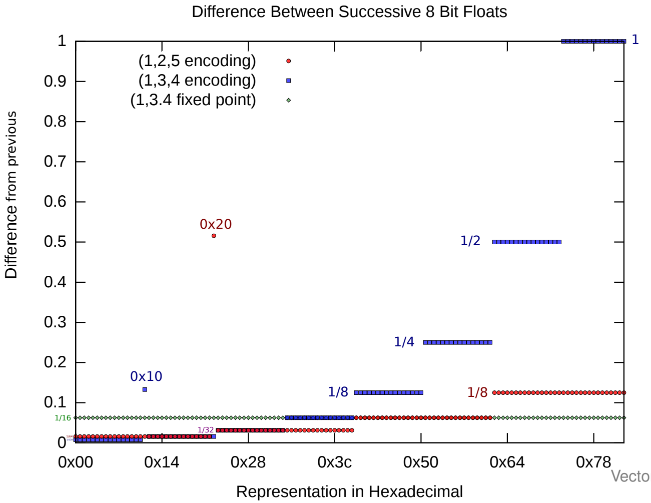
Visualisation of Floats

- With total length of *m* and *E* limited to 7 bits (1 sign bit)
- Operations are inexact (in general)

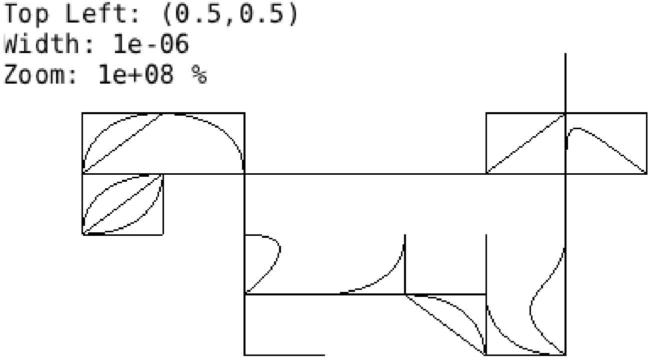


Visualisation of Floats II

- Difference between successive floats



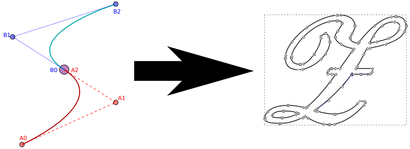
Precision is limited



```
Top Left: (0.5,0.5)
Width: 1e-06
Zoom: 1e+08 %
```

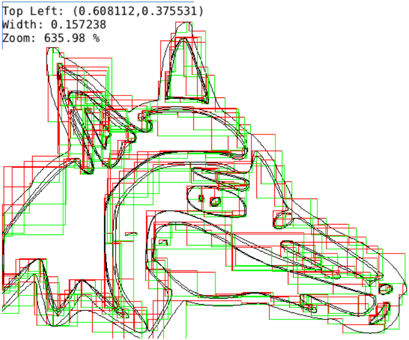
Structure of Vector Graphics

- Bézier Curve (Quadratic or Cubic Parametric Polynomial)
- Path of Bézier Curves → Shapes (with fill)
- Shapes include font glyphs, like this $\mathscr{X}$



Structure of Vector Graphics II

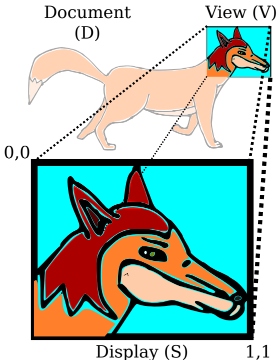
- Upload *bounding rectangles* of individual objects to renderer (OpenGL)
- Rectangles show individual Béziers forming outline of the Fox



```
Top Left: (0.608112,0.375531)
Width: 0.157238
Zoom: 635.98 %
```

Viewing Vector Graphics

- Tranform coordinates in document → display



Document via View to Display (S)

$$(X, Y) \longrightarrow (S_X, S_Y)$$

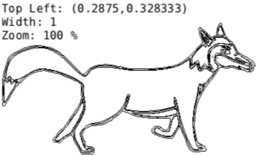
$$S_X = \frac{X - V_x}{V_w} \quad S_Y = \frac{Y - V_y}{V_h}$$

Floating point calculations go wrong

- Example: Insert objects at very small scale

$$S_X = \frac{X - V_x}{\enclose{circle}{V_w}} \quad S_Y = \frac{Y - V_y}{\enclose{circle}{V_h}}$$

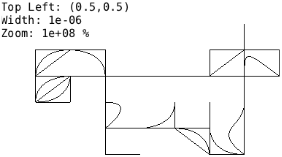
Division by small value $\Longrightarrow$ amplify rounding errors



```
Top Left: (0.2875,0.328333)
Width: 1
Zoom: 100 %
```

$V_w = 1$

✓



```
Top Left: (0.5,0.5)
Width: 1e-06
Zoom: 1e+08 %
```

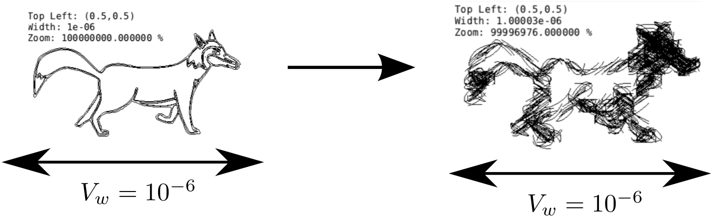
$V_w = 10^{-6}$

✗

Reducing error

- Don't apply view transformation directly
- Store object bounds relative to the display
- When modifying the view, modify object bounds
- Detail inserted into the view looks good, But...

Zoom out by a large amount then back in



```
Top Left: (0.5,0.5)
Width: 1e-06
Zoom: 100000000.000000 %
```

```
Top Left: (0.5,0.5)
Width: 1.00003e-06
Zoom: 99996976.000000 %
```

$V_w = 10^{-6}$

$V_w = 10^{-6}$

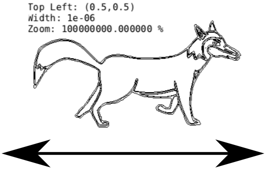
Inserted SVG

Accumulated Errors

Cumulated Errors with intermediate coord system

- Apply transformations to Paths not individual Béziers
- Paths render correctly, but drift appart

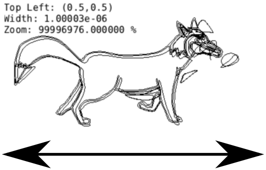
Zoom out by a large amount then back in



```
Top Left: (0.5,0.5)
Width: 1e-06
Zoom: 100000000.000000 %
```

$V_w = 10^{-6}$

Inserted SVG



```
Top Left: (0.5,0.5)
Width: 1.00003e-06
Zoom: 99996976.000000 %
```

$V_w = 10^{-6}$

Drifting of Paths

We only need to transform the Paths with increased precision

Arbitrary Precision Rationals

$$Q = \frac{N}{D} \tag{2}$$

- N and D are arbitrary precision integers

$$N = \sum_{i=0}^{S} d_i \beta^i \tag{3}$$

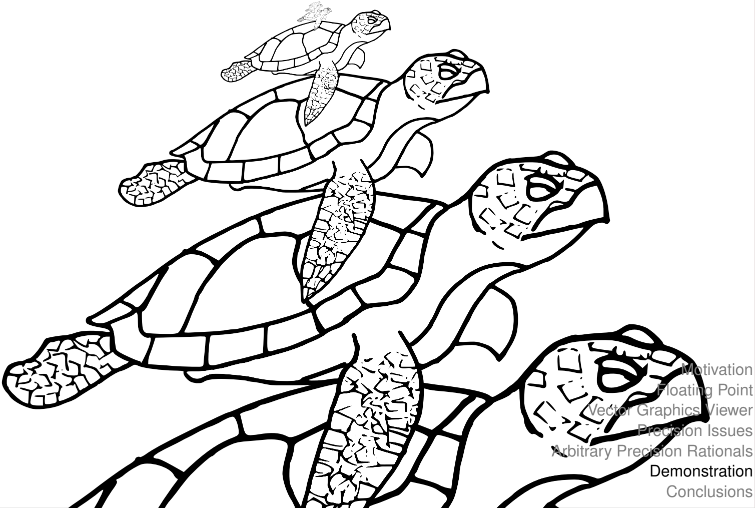
- d_i are fixed size integers, $\beta = 2^{64}$
- **Size S grows as needed**
- Operations are always exact
- Implemented by GNU Multiple Precision Library

Use Rationals to represent Path Coordinates

- Can move view to arbitrary point
- Insert detail (ie: Test SVG image) in Display coordinates
- Move view to another arbitrary point
- Move view back
- Detail is unchanged

Demonstration

- We can import standard SVGs wherever we want
- If we are willing to wait long enough
- “... But, asks the scientist, what does that turtle stand on? To which the lady triumphantly answers: ’You’re very clever, young man, but it’s no use – it’s turtles all the way down!’.“



What was done

- Implemented a basic SVG viewer
- Demonstrated how precision affects rendering vector graphics
- Using GMP rationals, demonstrated the ability to render SVGs scaled to an arbitrary position in a document

Future work

- Implement more of the SVG standard
- Trial alternative number representations
- Allow for saving and loading SVGs with arbitrary precision

References & More information

- Work on SVG viewer collaborative with David Gow
  - See David Gow's presentation about Quadtrees
- Muller et al, *Handbook of Floating Point Arithmetic*,
- Hearn, Baker *Computer Graphics*
- Kahan et al, *IEEE-754* (1985 and 2008 revision)
- Dahlstóm et al, *SVG WC3 Recommendation 2011*
- Grunland et al, *GNU Multiple Precision Manual 6.0.0a*
- Kahan's website

  `http://http.cs.berkeley.edu/~wkahan`

Q: Why not just increase floating point precision?

- Any fixed precision format will still give inexact results
- Eg: Accuracy of rendering a grid

Q: Arbitrary precision floats?

$$X = m \times 2^{E} \qquad (4)$$

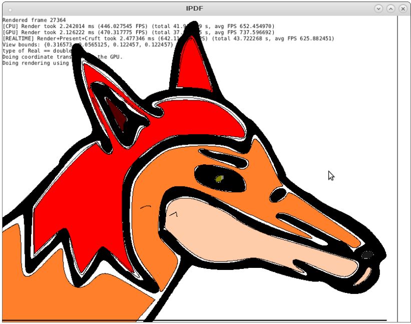
- *m* and *E* are of arbitrary size
- Implemented by MPFR or GMP
- Difficulties:
  - Need to manually set precision (size) of *m*
  - Some operations require infinite precision:

$$\frac{1}{3} = 0.3333333333333333333333\ \ldots \times 10^{0} \qquad (5)$$

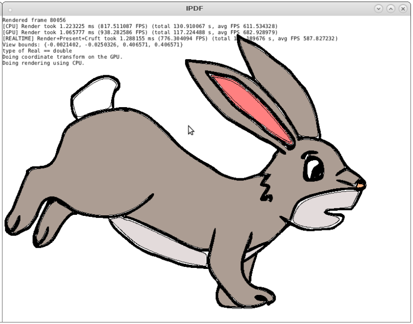
  - How do you choose when to increase precision?

Q: Why don't you have colour?

- We do![2]
- A complete implementation of SVG is "future work"



```
Rendered frame 27364
[CPU] Render took 2.242014 ms (446.027545 FPS) (total 41.9[illegible]9 s, avg FPS 652.454970)
[GPU] Render took 2.126222 ms (470.317775 FPS) (total 37.[illegible]5 s, avg FPS 737.596692)
[REALTIME] Render+Present+Cruft took 2.477346 ms (642.11[illegible]PS) (total 43.722268 s, avg FPS 625.882451)
View bounds: {0.316573, [illegible]0.0565125, 0.122457, 0.122457}
type of Real == double
Doing coordinate trans[illegible] the GPU.
Doing rendering using [illegible]
```



```
Rendered frame 80056
[CPU] Render took 1.223225 ms (817.511087 FPS) (total 130.910067 s, avg FPS 611.534328)
[GPU] Render took 1.065777 ms (938.282586 FPS) (total 117.224488 s, avg FPS 682.928979)
[REALTIME] Render+Present+Cruft took 1.288155 ms (776.304094 FPS) (total 1[illegible]189676 s, avg FPS 587.827232)
View bounds: {-0.0021402, -0.0250326, 0.406571, 0.406571}
type of Real == double
Doing coordinate transform on the GPU.
Doing rendering using CPU.
```

[2] If you are willing to wait long enough