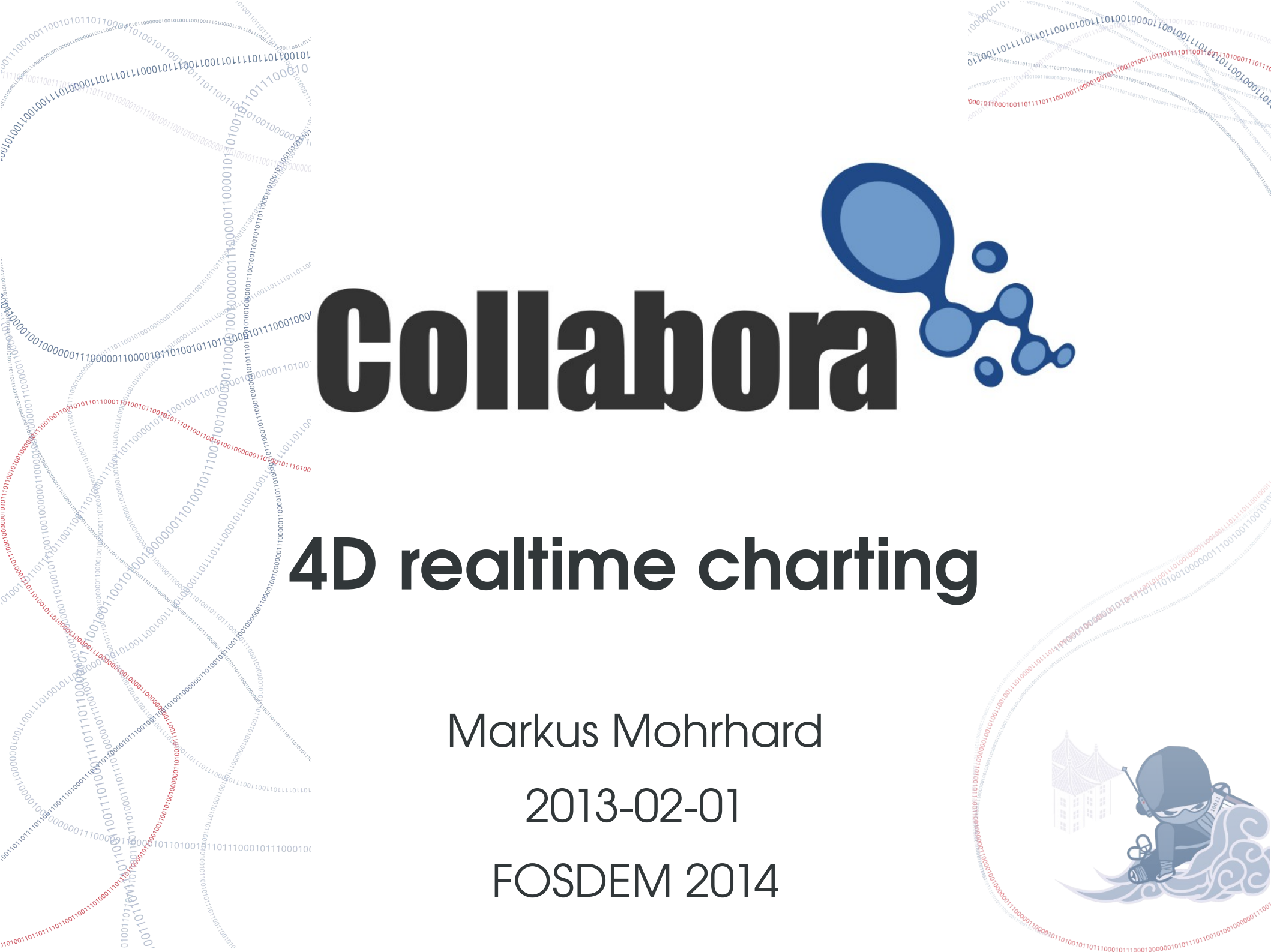


4D realtime charting

Markus Mohrhard

2013-02-01

FOSDEM 2014

Overview

What is “4D realtime charting”

- time based charting

- 3+ dimensions without time

Implementation parts

- Property Mapping
- Time based approach in Calc/Chart2
- new rendering backend

Property Mapping

Adding more dimensions to our charts

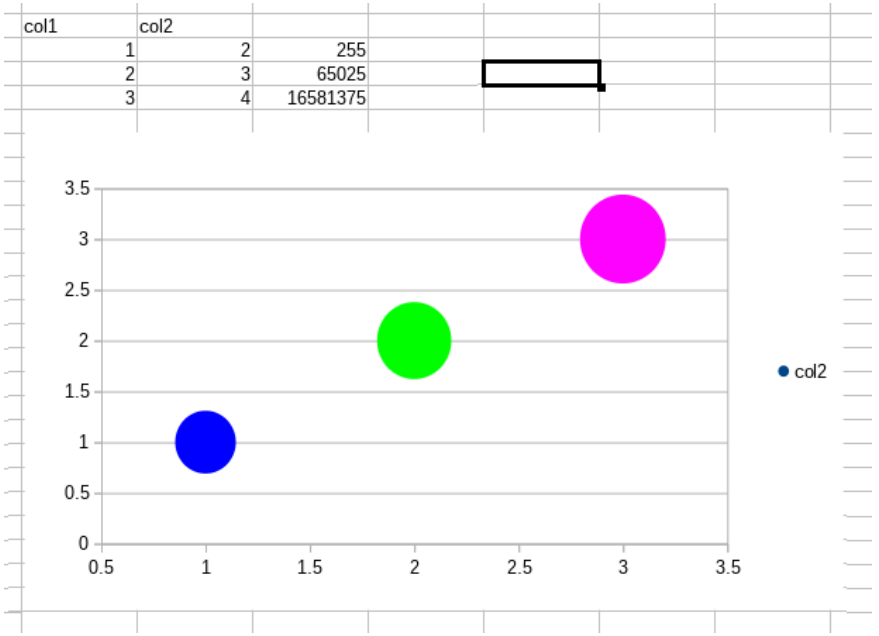
Idea

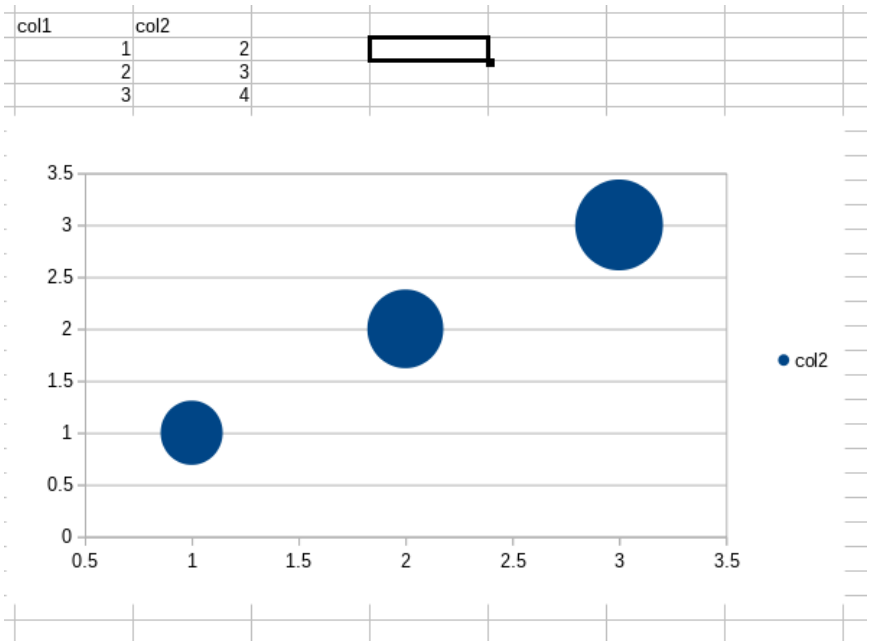
- Use properties and vary value
- Map value to property value
- Makes chart more flexible

Implementation

- New data series for properties
- Functor to map cell value(double) to property value
- Overwrites hard properties

Example


col1
col2
1
2
255
2
3
65025
3
4
16581375
3.5
3
2.5
2
1.5
1
0.5
0
0.5
1
1.5
2
2.5
3
3.5
col2


col1
col2
1
2
2
3
3
4
3.5
3
2.5
2
1.5
1
0.5
0
0.5
1
1.5
2
2.5
3
3.5
col2

Time based charting

Dynamically change your charts

Idea

- Animate charts through another dimension
- Each point in time is represented by a new data set
- Interpolate values between data sets

Implementation

- Sheets represent points in time
- Always keep two points cached in chart2
  - Interpolate between two points
  - Interpolate mapped properties
- Timer that triggers repaint of chart every few ms

Example

Problems

- Rendering too slow!!!
- Getting data through UNO is slow
- Destroy cache in calc with sheet switch

OpenGL backend for charts

A new experimental renderer

Idea

- Use GPU for graphic processing
- Avoid our own slow rendering code
- Abstract rendering
-

Implementation

- moved all direct createInstance calls into factory
- created AbstractShapeFactory
  - normal ShapeFactory
  - new OpenglShapeFactory
- return Dummy objects in OpenGL case
  - store state for rendering

Implementation

- New OpenGL initialization code
  - Works currently on Linux and Windows
  - Do you have a Mac and some time?
- Vertex and Fragment shaders
  - OpenGL 3.0+ and GLSL 1.20+
- Offscreen rendering to FBO
- Get Image and put back into drawinglayer

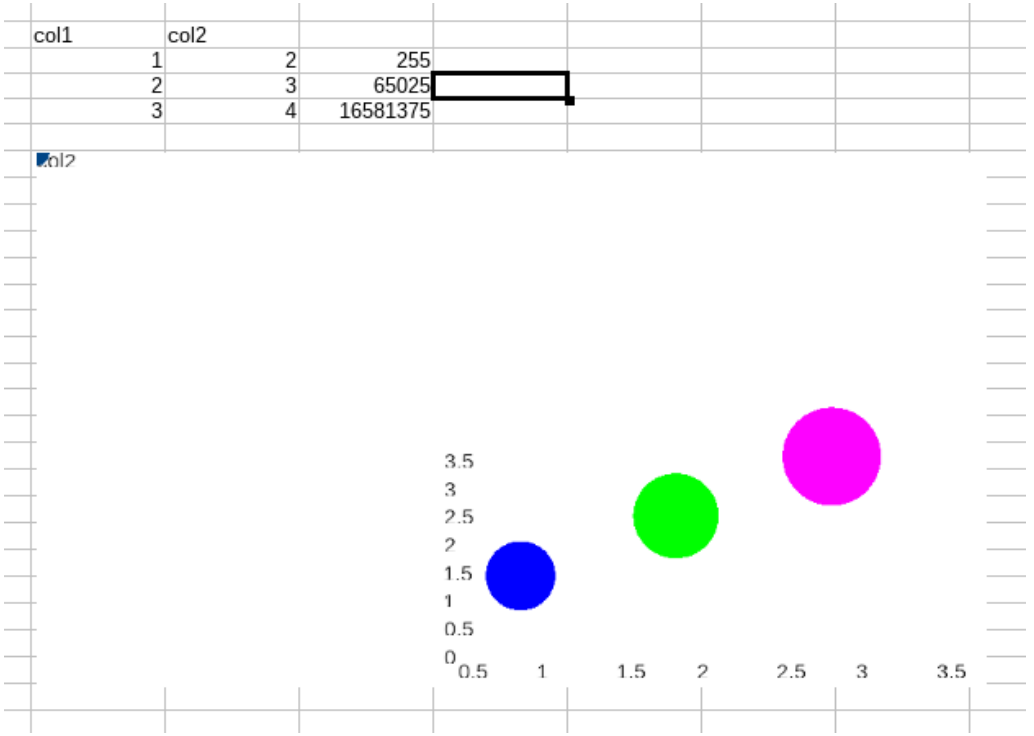
Typical shader

- Shader = program for the GPU

```
 8  */
 9
10 attribute vec3 vPosition;
11 uniform mat4 MVP;
12 uniform vec4 vColor;
13 varying vec4 fragmentColor;
14
15 void main()
16 {
17     gl_Position =  MVP * vec4(vPosition, 1);
18     fragmentColor = vColor;
19 }
20
```

```
 9
10 varying vec4 fragmentColor;
11
12 void main()
13 {
14     gl_FragColor = fragmentColor;
15 }
16
17 /* vim:set shiftwidth=4 softtabstop=4 expandtab: */
```


Example



Don’t look too closely. Still some work to do.

Problems

- Text rendering
- Positioning
- Selection of elements, …
- OpenGL on Linux
  - CentOS 5.0
-

Future

What we have planned and crazy ideas

Future chart

- File format work
- Improve OpenGL renderer
- Usability improvements
- Switch to OpenGL renderer by default (?)
- More performance work

Future rest of code

- Integrate OpenGL context creation code into VCL/(?)
- Replace existing OpenGL code with new code (programmable pipeline)
- Mac work
- OpenGL ES for Android/IOS (?)

Thanks

- AMD
- MCW
- TDF
- Collabora
- everyone who fixed my mess