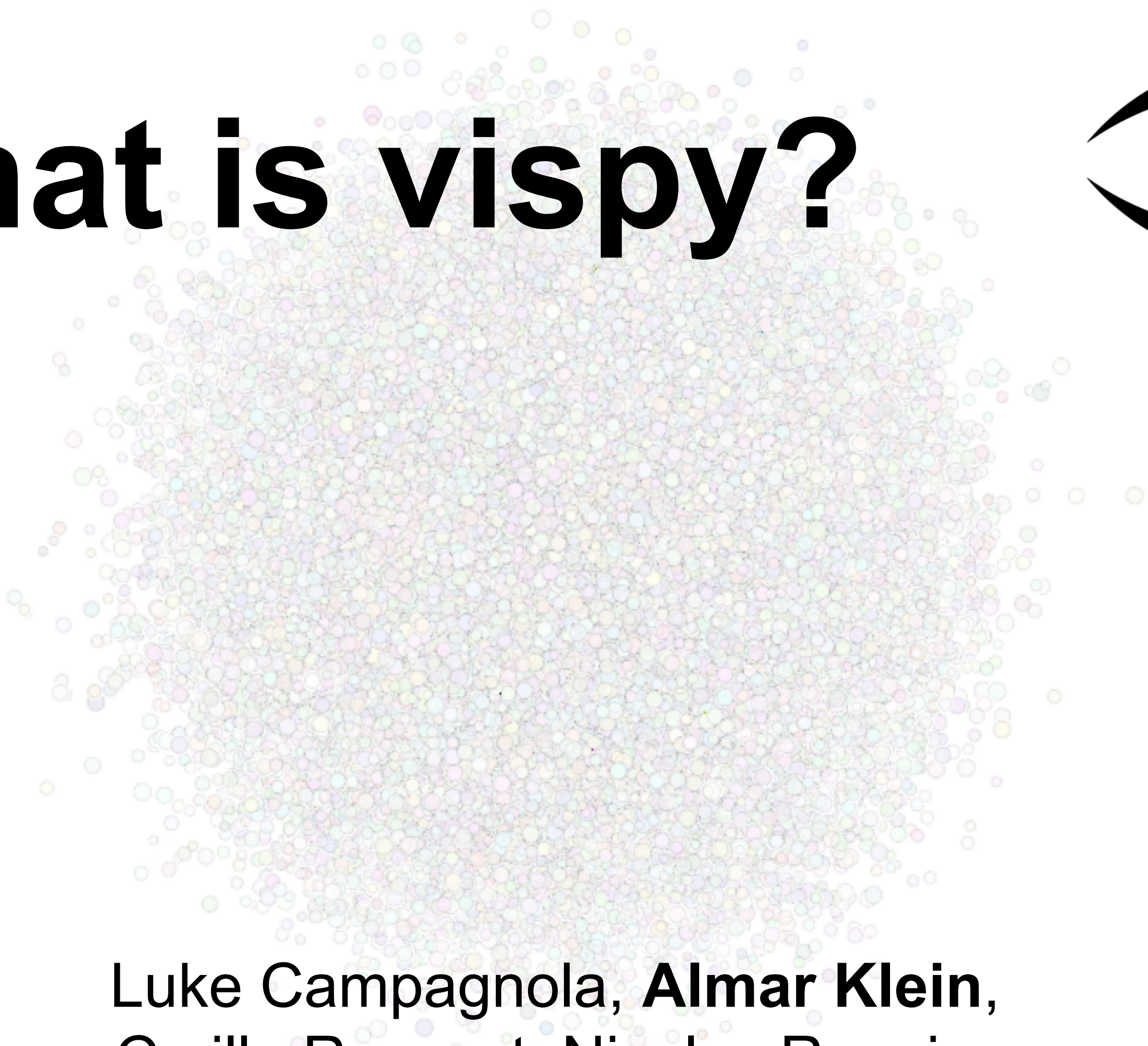
What is vispy?



Luke Campagnola, **Almar Klein**,
Cyrille Rossant, Nicolas Rougier

Plotting goes interactive

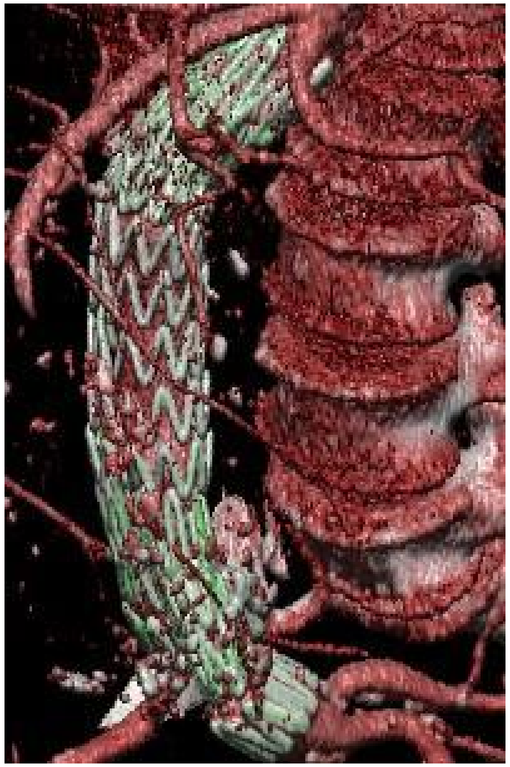
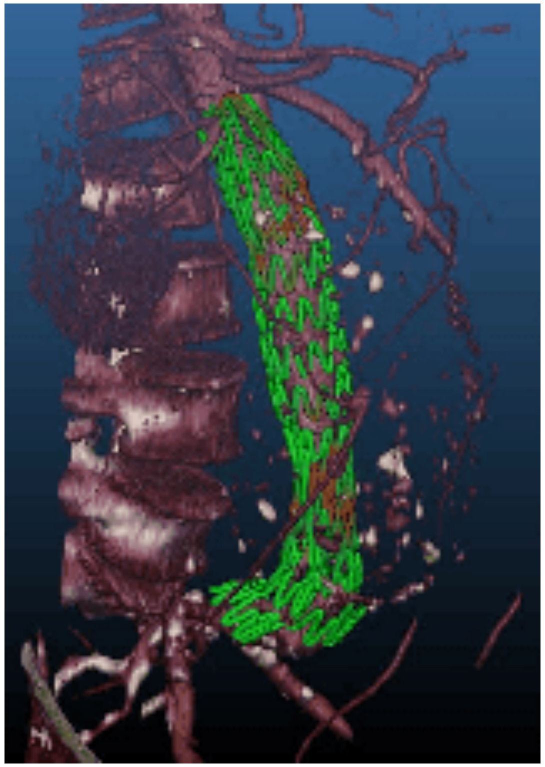
- Data is growing fast
- Explore rather than “look”

Problem:

- Deal with large datasets
- Need interaction (+speed)

3D data more common

- Interaction
- Speed
- Flexibility

Solution

- Leverage power of GPU
- Shaders for high quality results

About Vispy

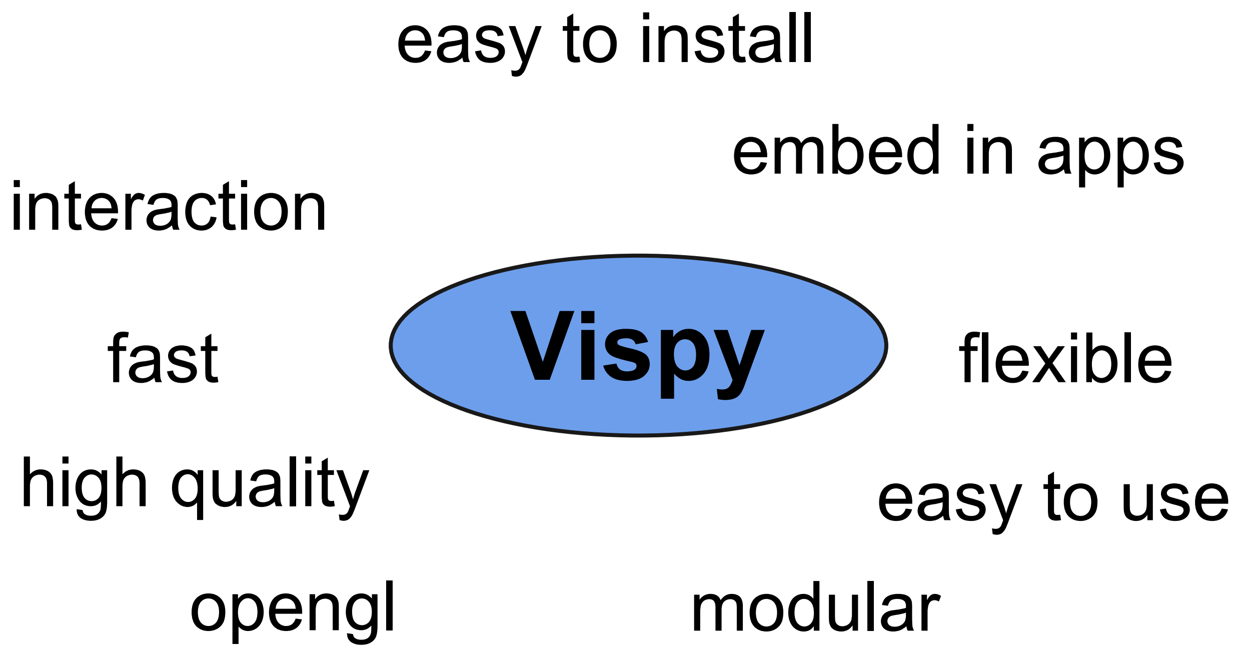
Who are we

- Luke Campagnola - Pyqtgraph
- Almar Klein - Visvis
- Cyrille Rossant - Galry
- Nicolas Rougier - Glumpy

Vispy: start from scratch:

best of all toolkits (and better)

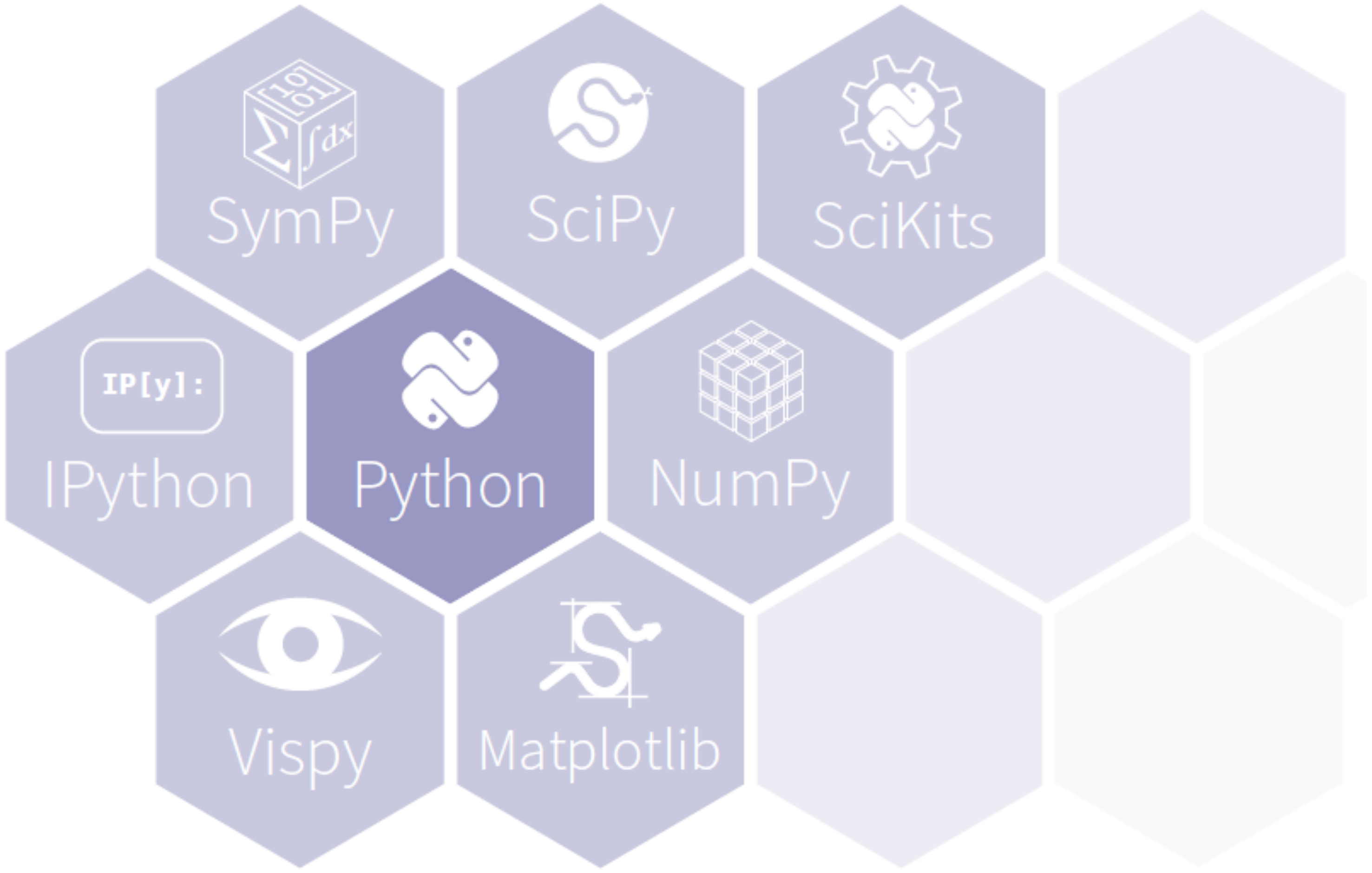
Vispy goals


easy to install
embed in apps
interaction
fast
Vispy
flexible
high quality
easy to use
opengl
modular

OpenGL ES 2.0

- Modern OpenGL
- Clean (just 150 functions)
- Good availability
- WebGL
- Mobile devices

Scipy ecosystem



Target use cases

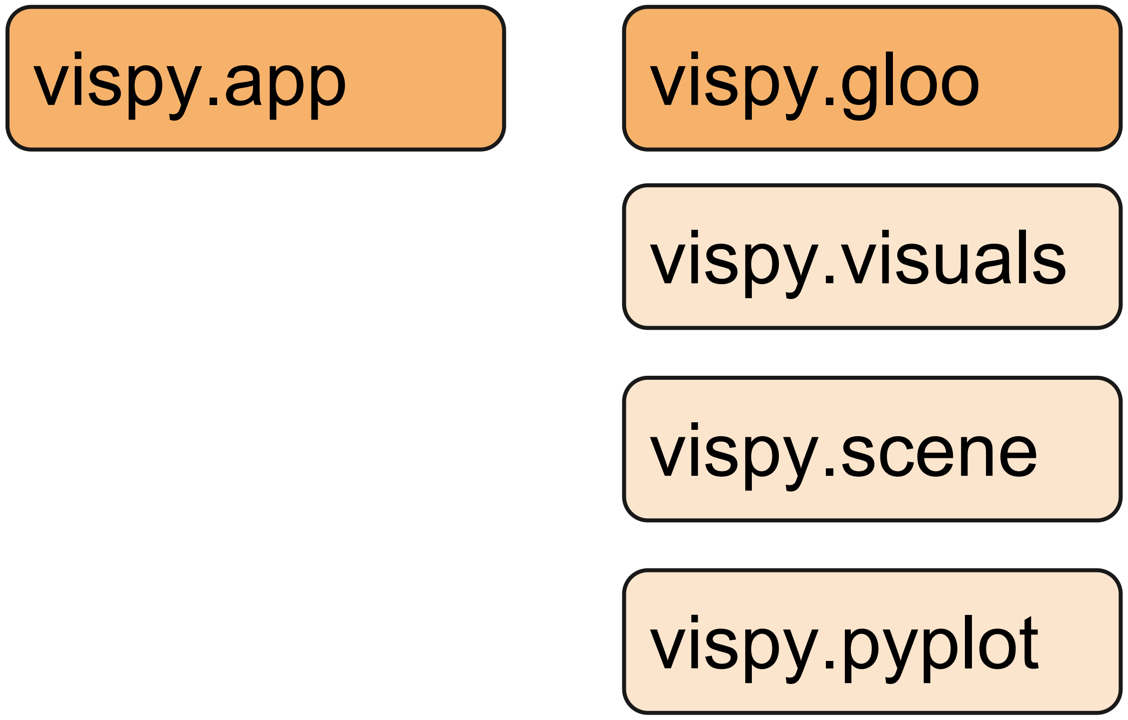
Anywhere you need visualization …

- Plotting, also big-ish data
- Figures for publications
- 3D (e.g. volume rendering)
- Simulations
- Specific user interfaces
- Games
- VR / AR
- Art

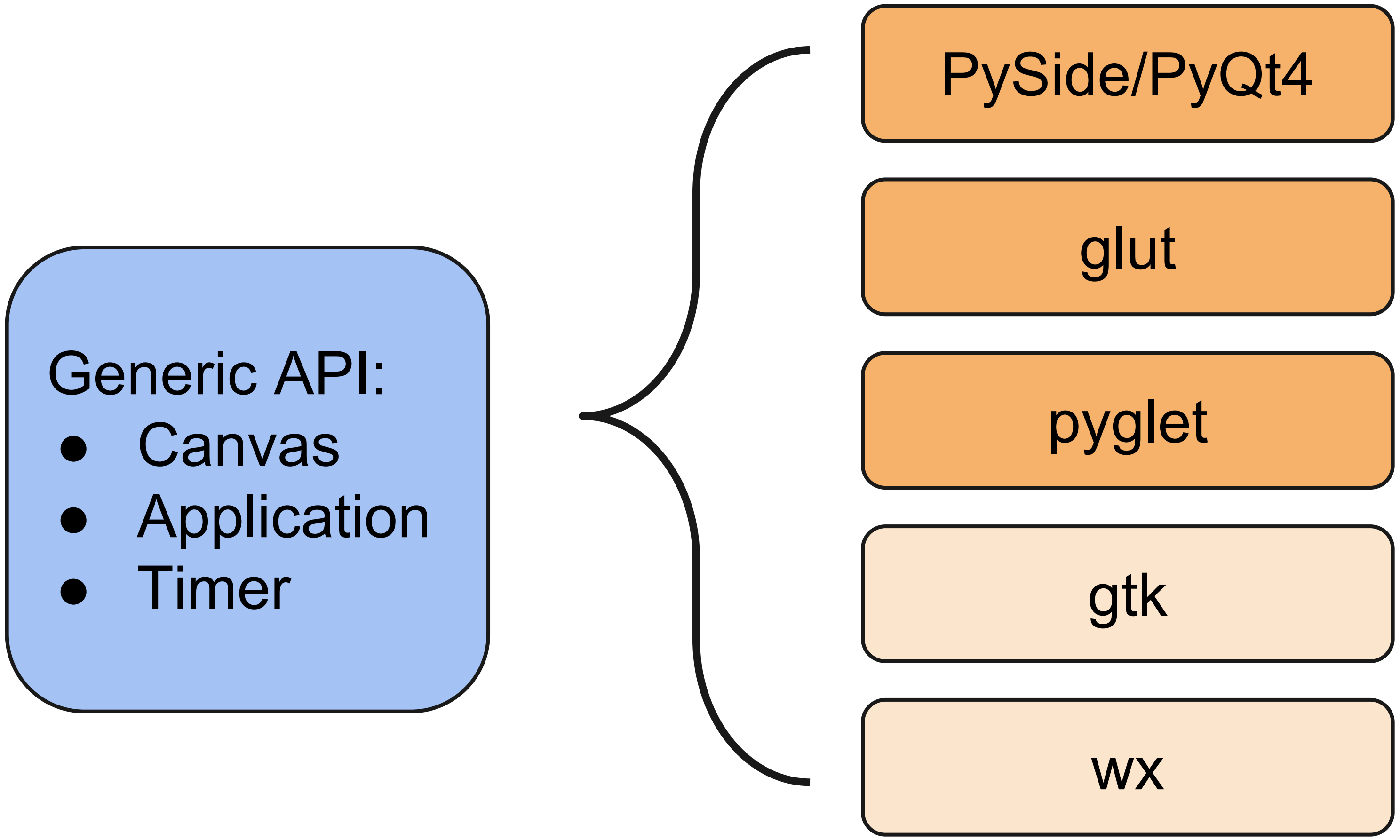
Inside Vispy

Package layout

Vispy structure



vispy.app



vispy.app.use()

vispy.gloo

API fits on one slide!

GLObject

handle
activate()
deactivate()
delete()

FragmentShader
VertexShader

code
source
set_code()

Program

shaders
attributes
uniforms
activate_object()
attach()
detach()
draw()
set_vars()

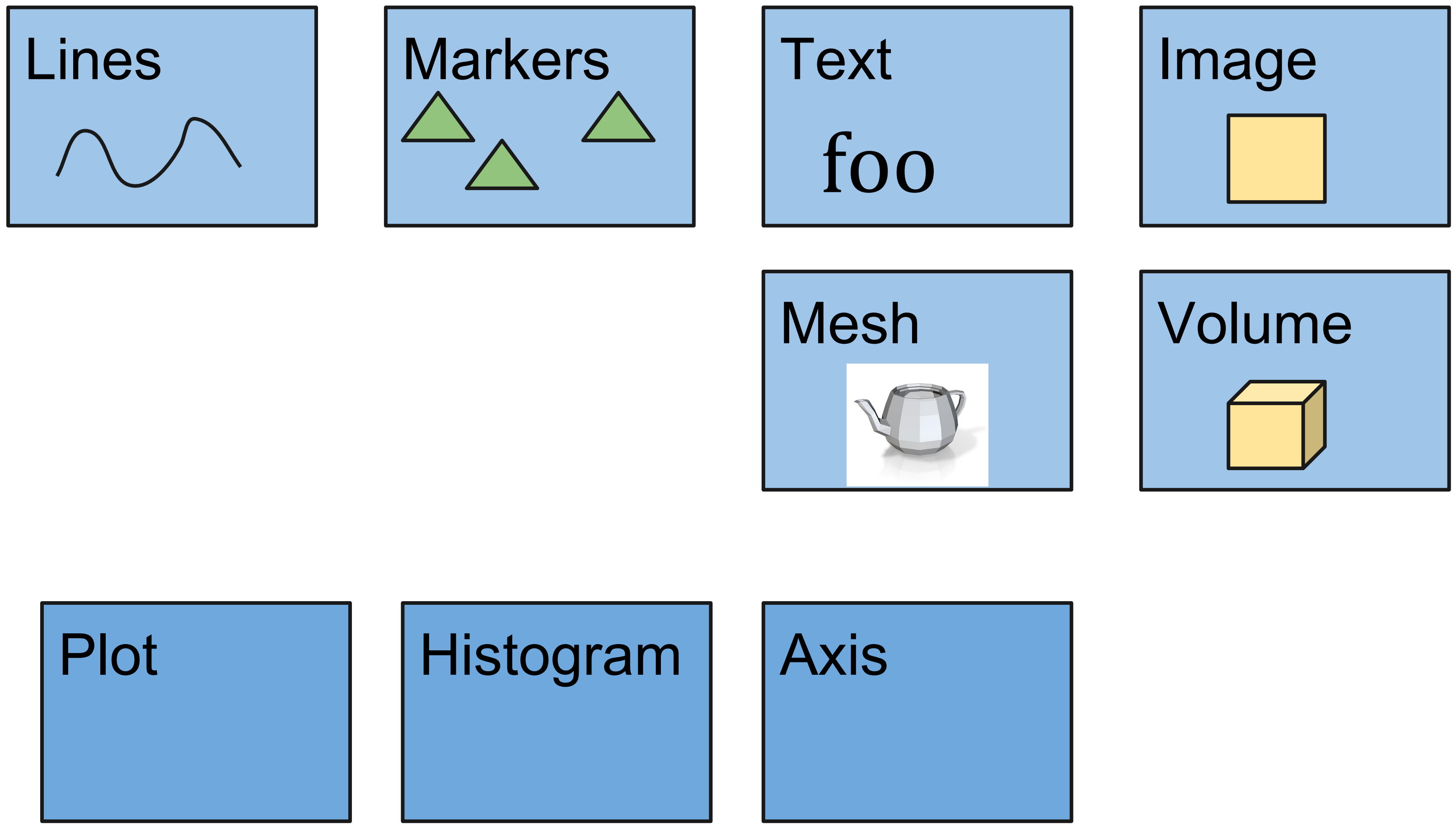
VertexBuffer
ElementBuffer

nbytes
count
dtype
offset
stride
vsize
set_data()
set_nbytes()
set_subdata()
set_count()

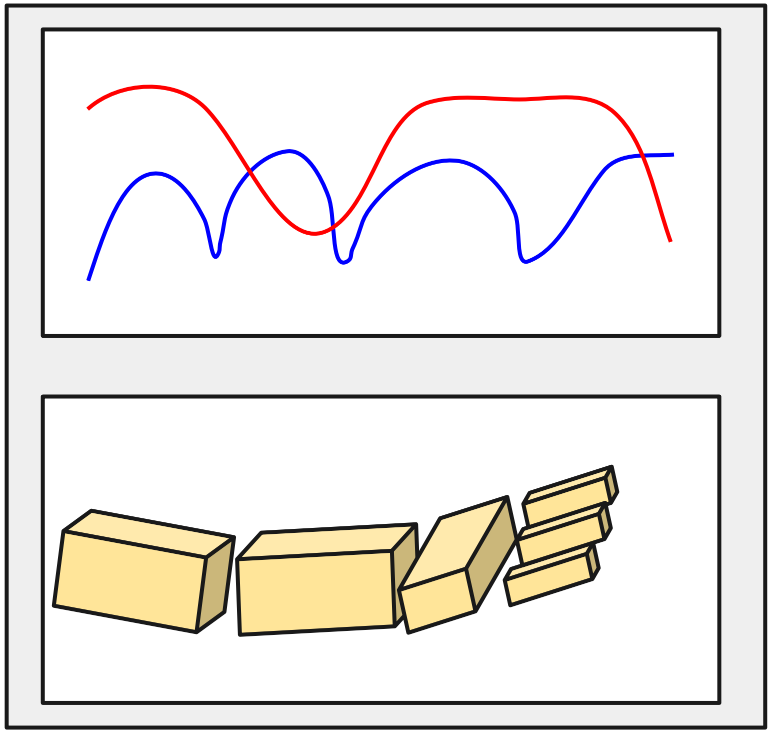
Texture2D
Texture3D
TextureCubema
p

set_data()
set_filter()
set_shape()
set_subdata()
set_wrapping()

vispy.visuals



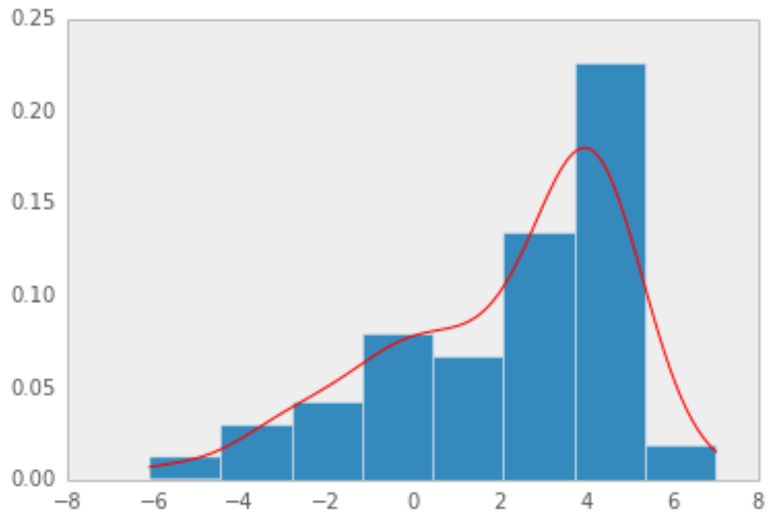
vispy.scene



- Viewbox (i.e. subplot)
- Object hierarchy
- Collections?

vispy.pyplot

- Functional interface
- compatible with Matplotlib.pyplot (and Matlab)

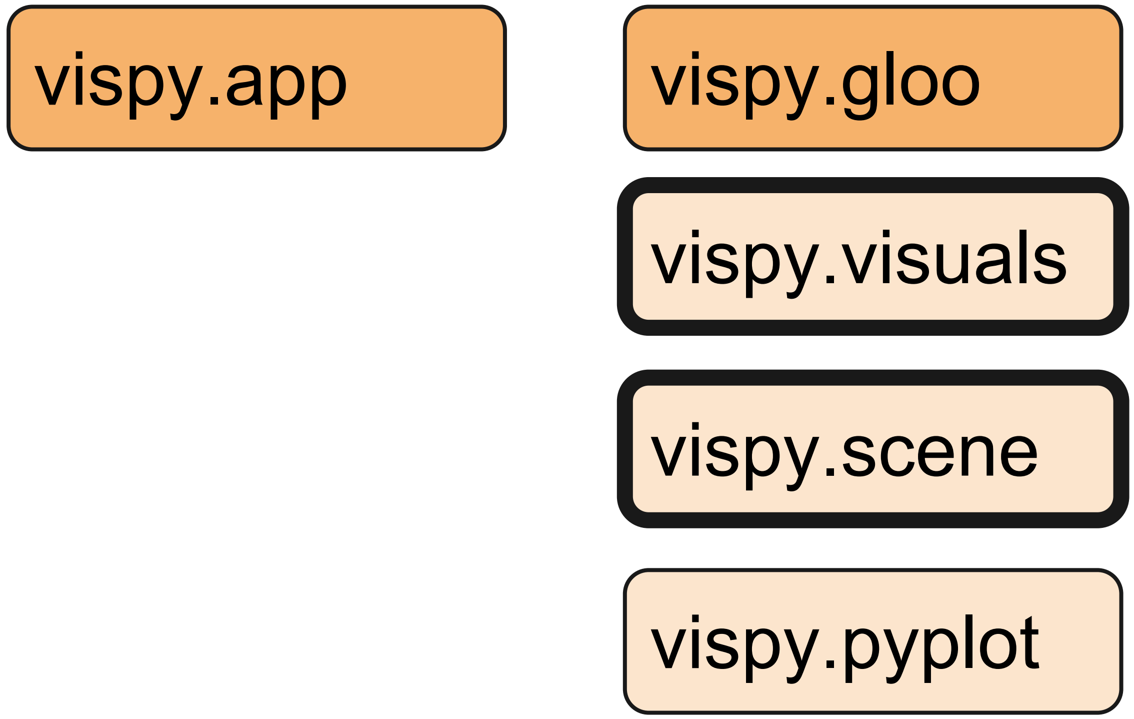


Current work

What we want to work on this week

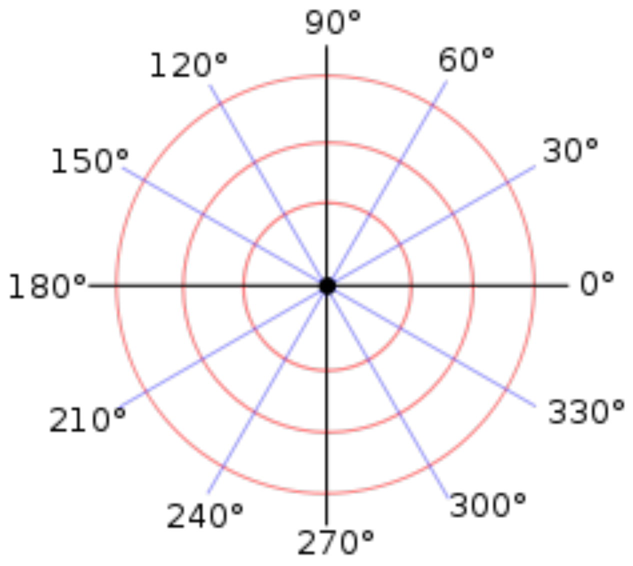
Visuals and Scene layer

When we have these, Vispy becomes much easier to use (no OpenGL required).

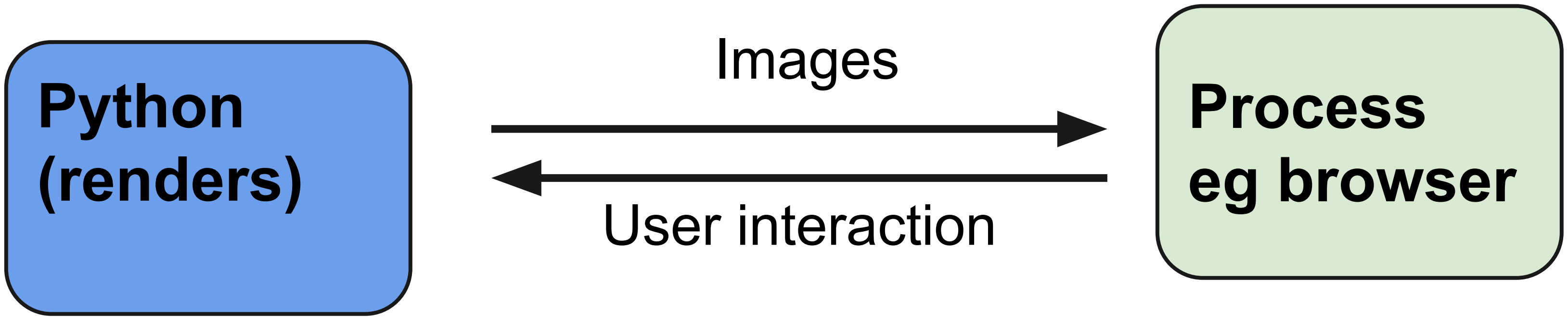


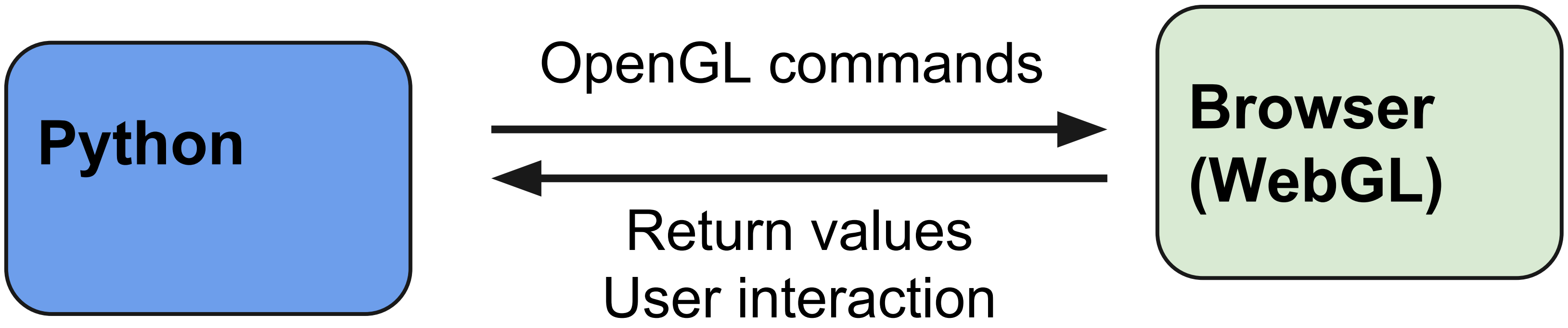
Transformations

- Object hierarchy
- Complex transformations
  - log
  - polar
  - maps?



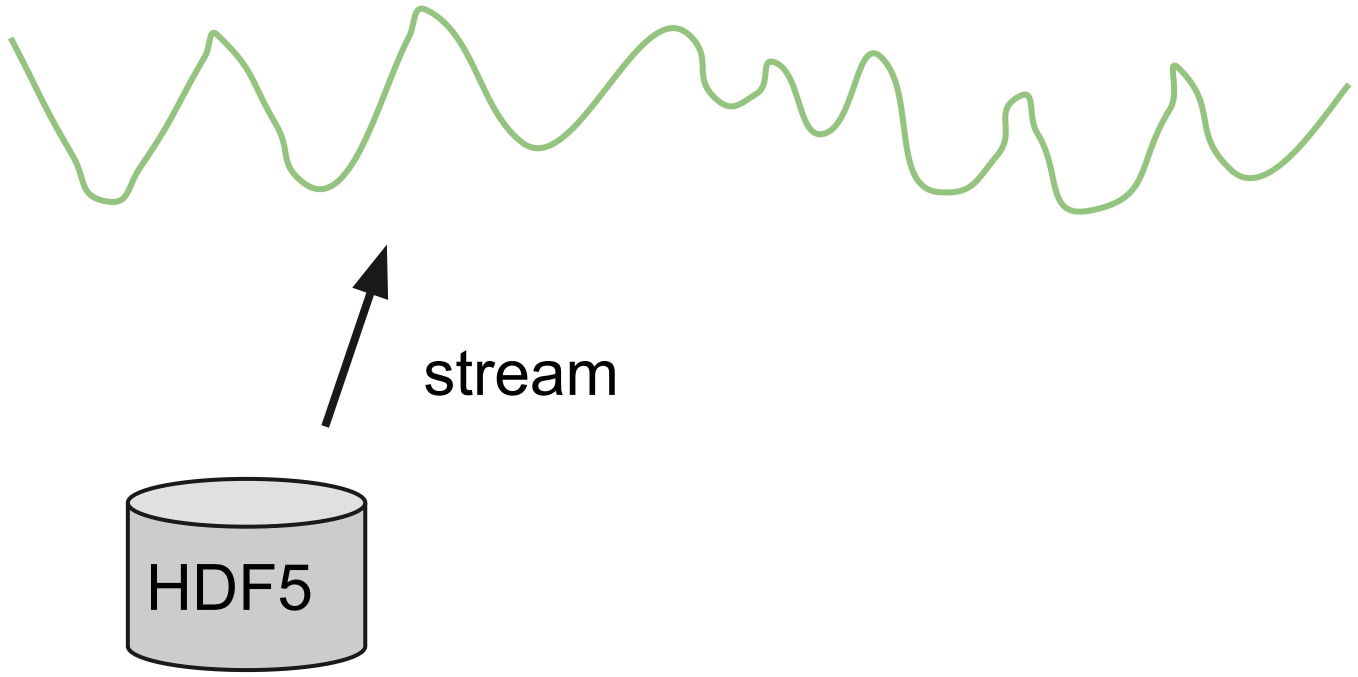
Remote rendering





Out of core plotting

When you have more data than you can draw.



Fast code

- Isosurface extraction
- mesh simplification
- Convex hull
- …

We need to think about:

- Include in Vispy?
- Separate package?

Other tasks

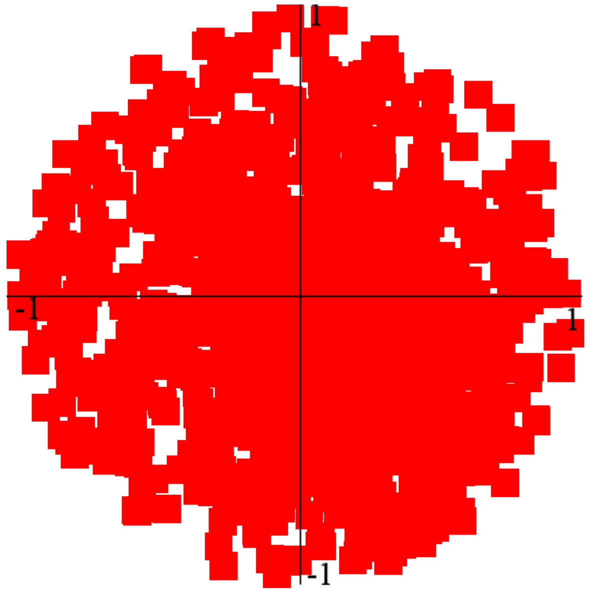
- More GUI backends
- Test coverage
- Test whether rendered image is correct
- Add visuals
- Add examples
- OpenCL interop
- Gloo standalone
- WebGL
- … etc.

Future work

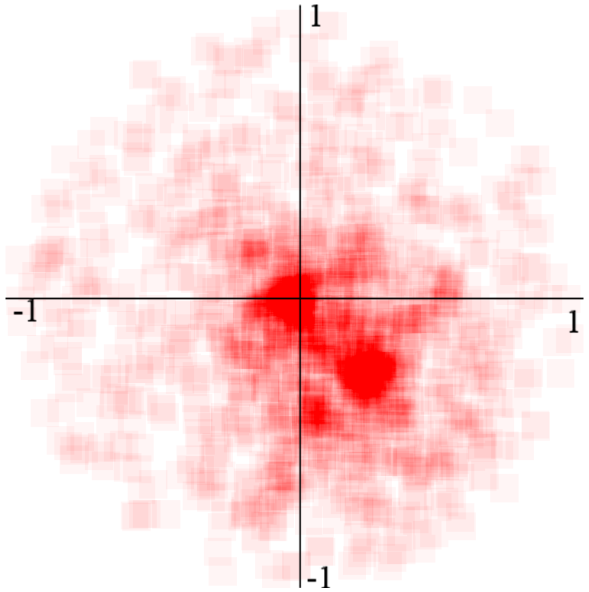
Ideas and experiments

Abstract rendering

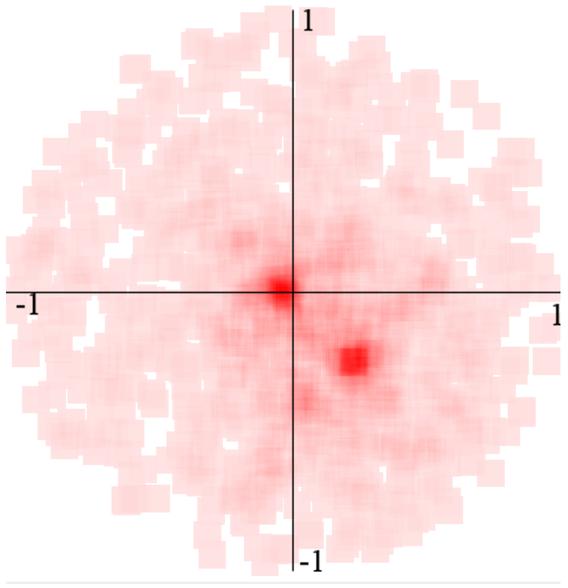
Ideas of Peter Wang



over-plot



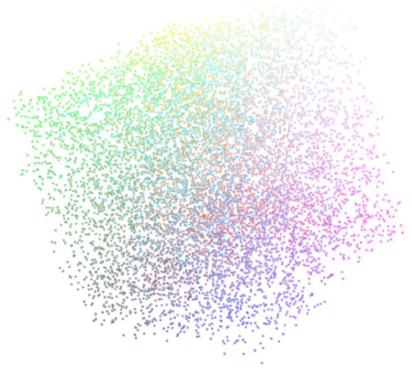
standard alpha

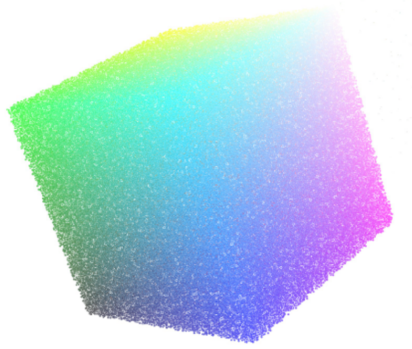


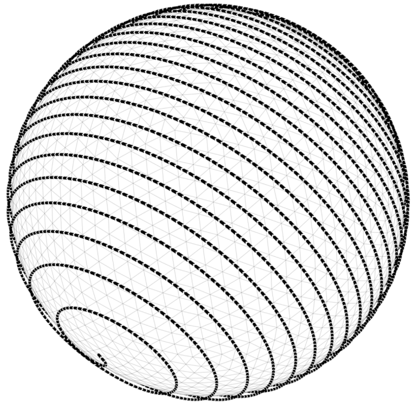
high-dev alpha

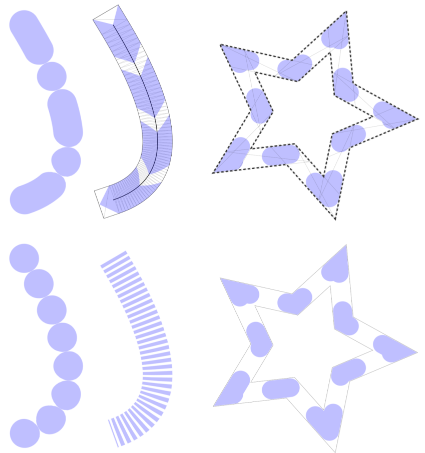
Beautiful lines / markers

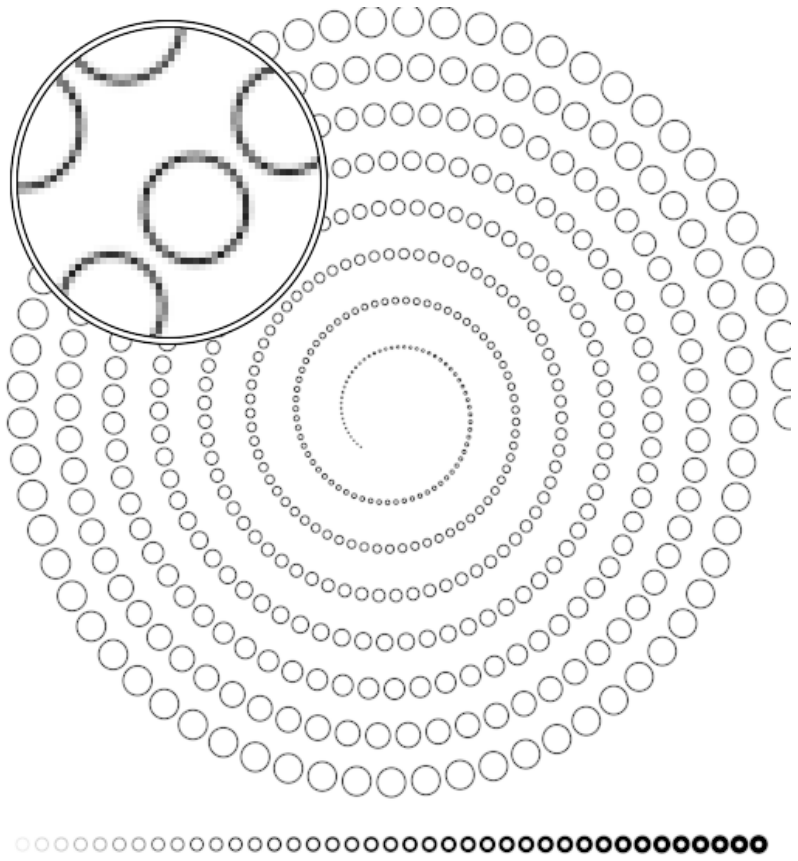
We do not (always) have to trade quality for speed


10,000 pts - 403 FPS


1,000,000 pts - 40 FPS



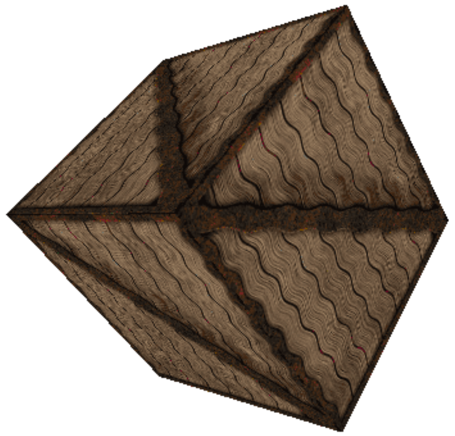




Dealing with transparency

- Doing it well is hard
- Depth sorting costs CP
- Two 'smart' techniques
  - (Dual) depth peeling
  - Weighted average

Further information

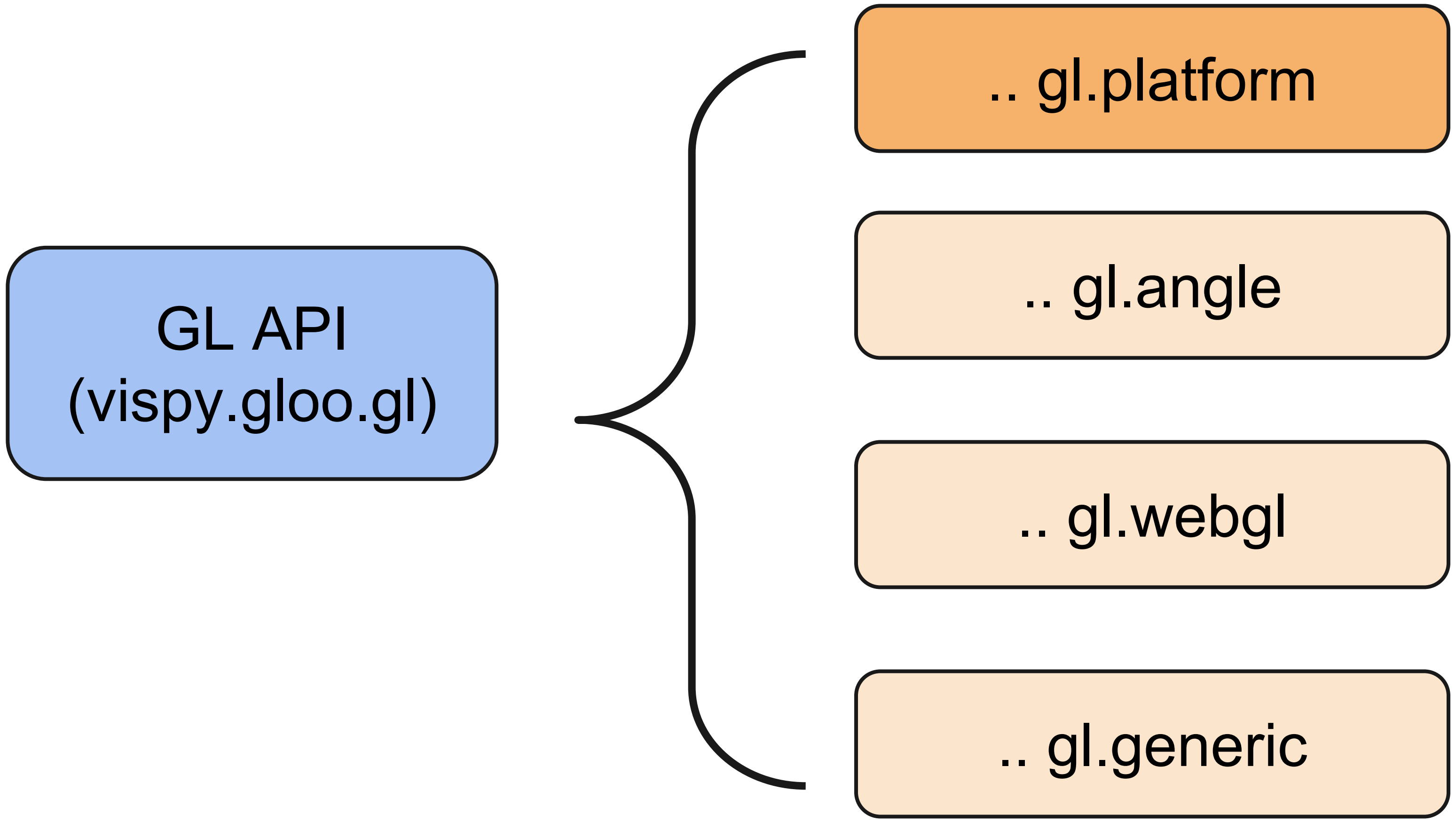


Website: vispy.org

Code repo: github.com/vispy/vispy

Extra slides

vispy.gloo.gl



Minimal example

```
from vispy import app, gl

c = app.Canvas(show=True)

@c.connect
def on_paint(event):
    gl.glClearColor(0,1,0,1)
    gl.glClear(gl.GL_COLOR_BUFFER_BIT)

app.run()
```

Minimal example (future)

```python
from vispy import pyplot as plt
import numpy as np

data = np.random.random((100000,3))

plt.scatter(data)
```

Shader example

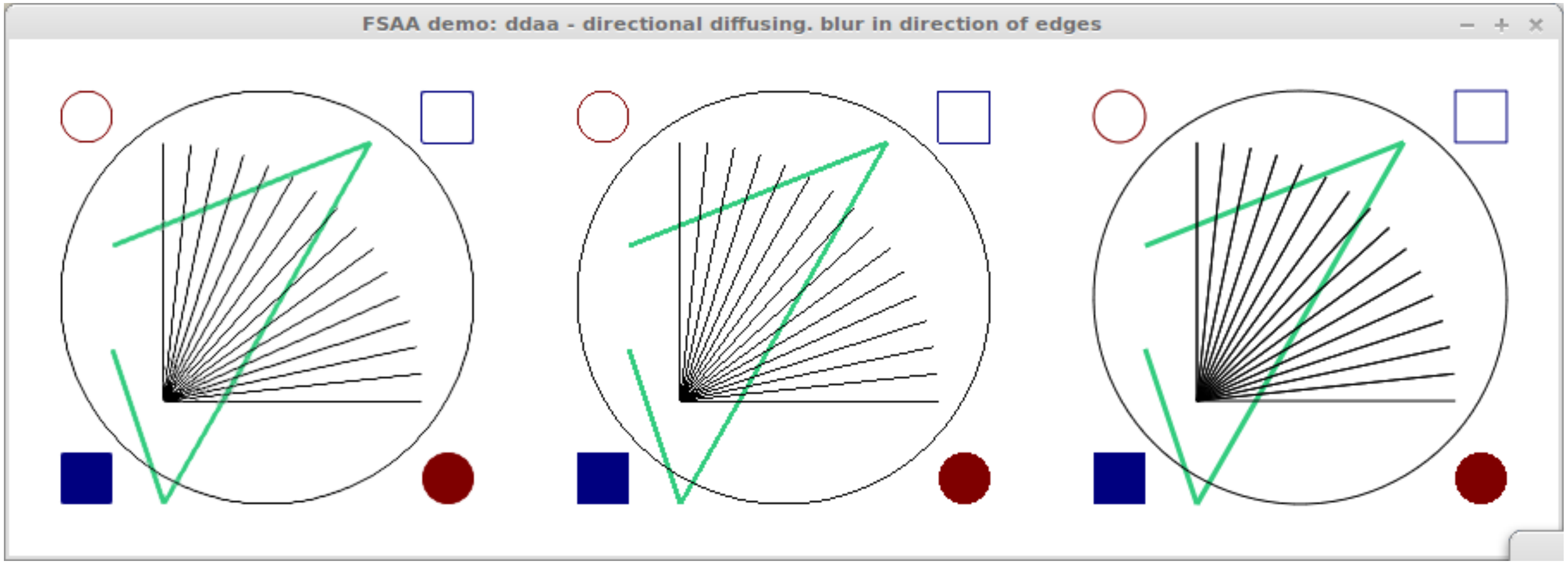
Vertex shader

```
attribute vec3 position;
void main()
{
    gl_Position = vec4(position,1.0);
}
```

Fragment shader

```
uniform vec4 color;
void main()
{
    gl_FragmentColor = color;
}
```

Full Screen AA (FSAA)


FSAA demo: ddaa - directional diffusing. blur in direction of edges