

# Graphite for graphics

---

a short introduction...

# Outline

---

- 3D Rendering Libraries
  - Programming Model
  - Meshmaker
  - Graphite
  - Conclusions
-

# 3D Rendering Libraries

---

OpenGL	Direct X
<ul style="list-style-type: none"><li>• Hardware support</li><li>• Multiplatform</li><li>• C style API</li></ul>	<ul style="list-style-type: none"><li>• (Better) hardware support</li><li>• Windows ONLY</li><li>• Object oriented API</li></ul>

- We use mostly OpenGL...
-

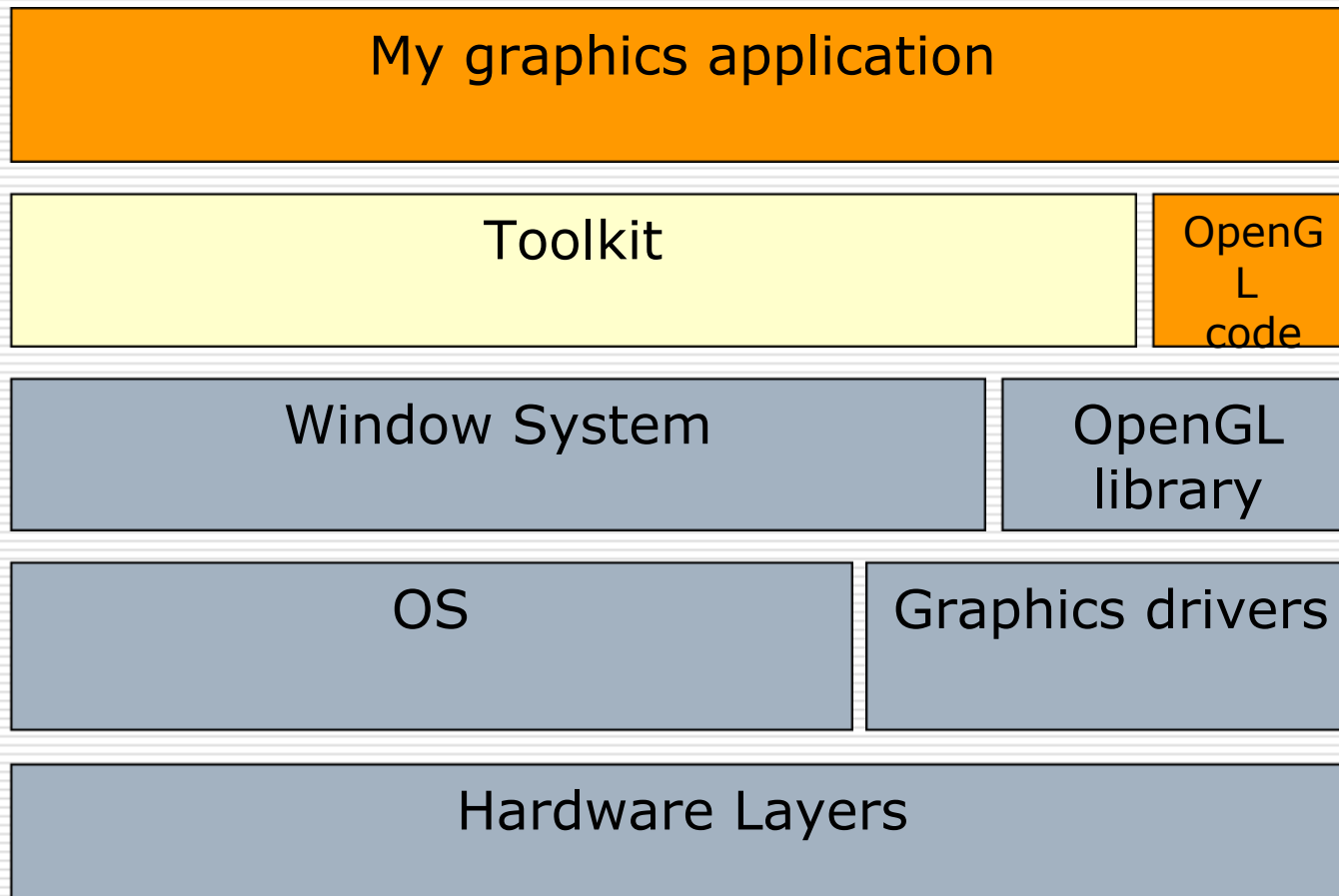
# Programming Model

---

1. Create window(s)
  2. Create an OpenGL context for the window (binding)
  3. Render and wait for events:
    - Mouse
    - Keyboard
    - Timers
    - Etc...
-

# Architecture

---



# Development kits

---

## OpenGL Utility Toolkit (GLUT)

### Pros:

- Very Portable
- Simple
- Designed for OpenGL

### Cons:

- Simple
-

# Development kits

---

.NET or older versions of Visual Studio

Pros:

- Uses the “standard” Microsoft Foundation Classes (MFC)
- Well supported

Cons:

- Windows only
  - OpenGL integration requires a bit of work
-

# Development kits:

---

Qt

Pros:

- Portable
- Complete framework

Cons:

- Non-trivial
-

# MeshMaker

---

- Uses MFC with OpenGL integrated
  - Includes:
    - Mesh Data Structure (Corner table)
    - VRML loading
    - Mesh rendering
-

# MeshMaker

---

- Pros
    - Good documentation
    - Simple
  - Cons
    - Windows only
    - Not modular
    - Little functionality
-

# Graphite

---

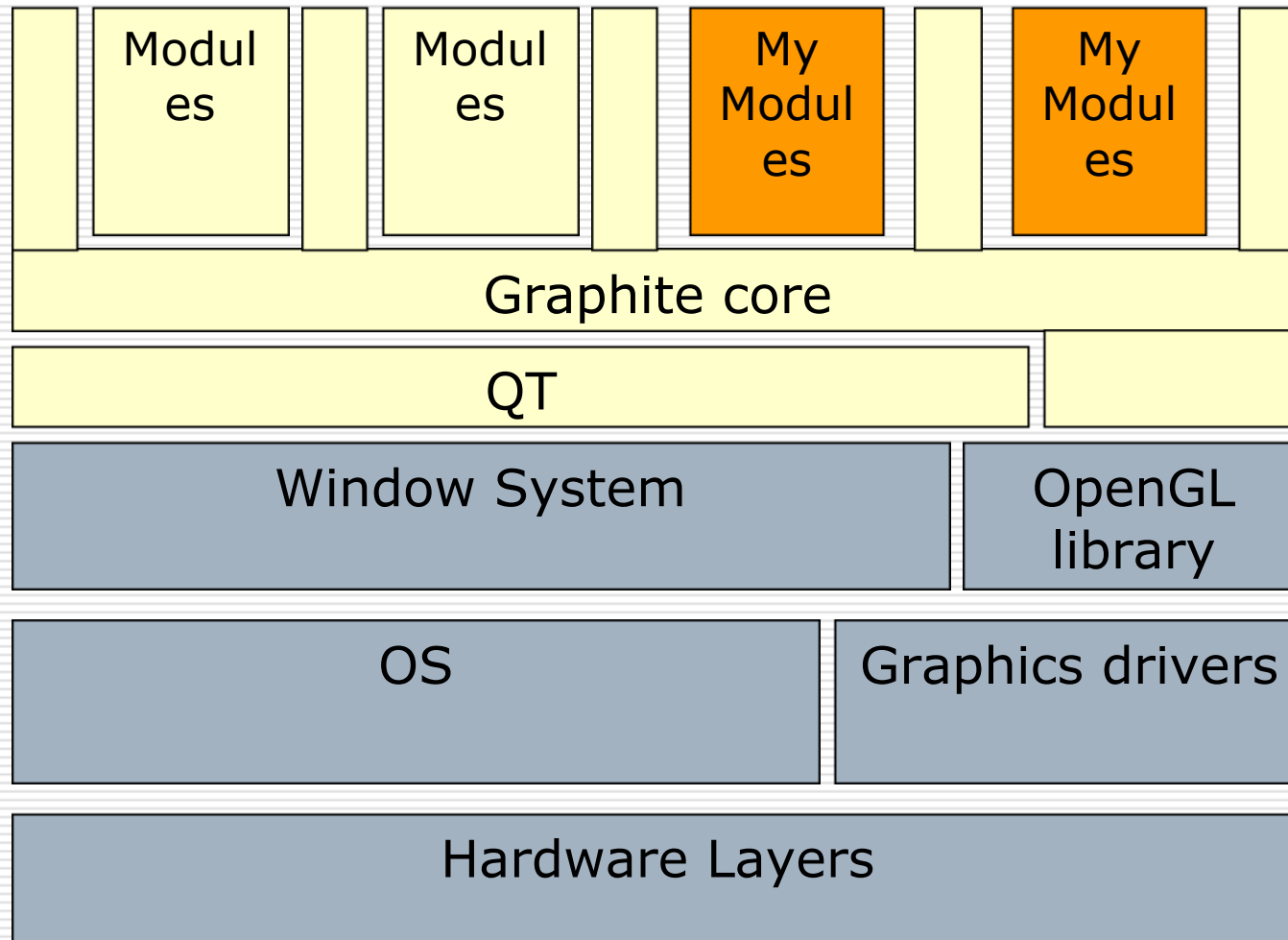
Qt based open architecture graphics  
framework

New version last week!

---

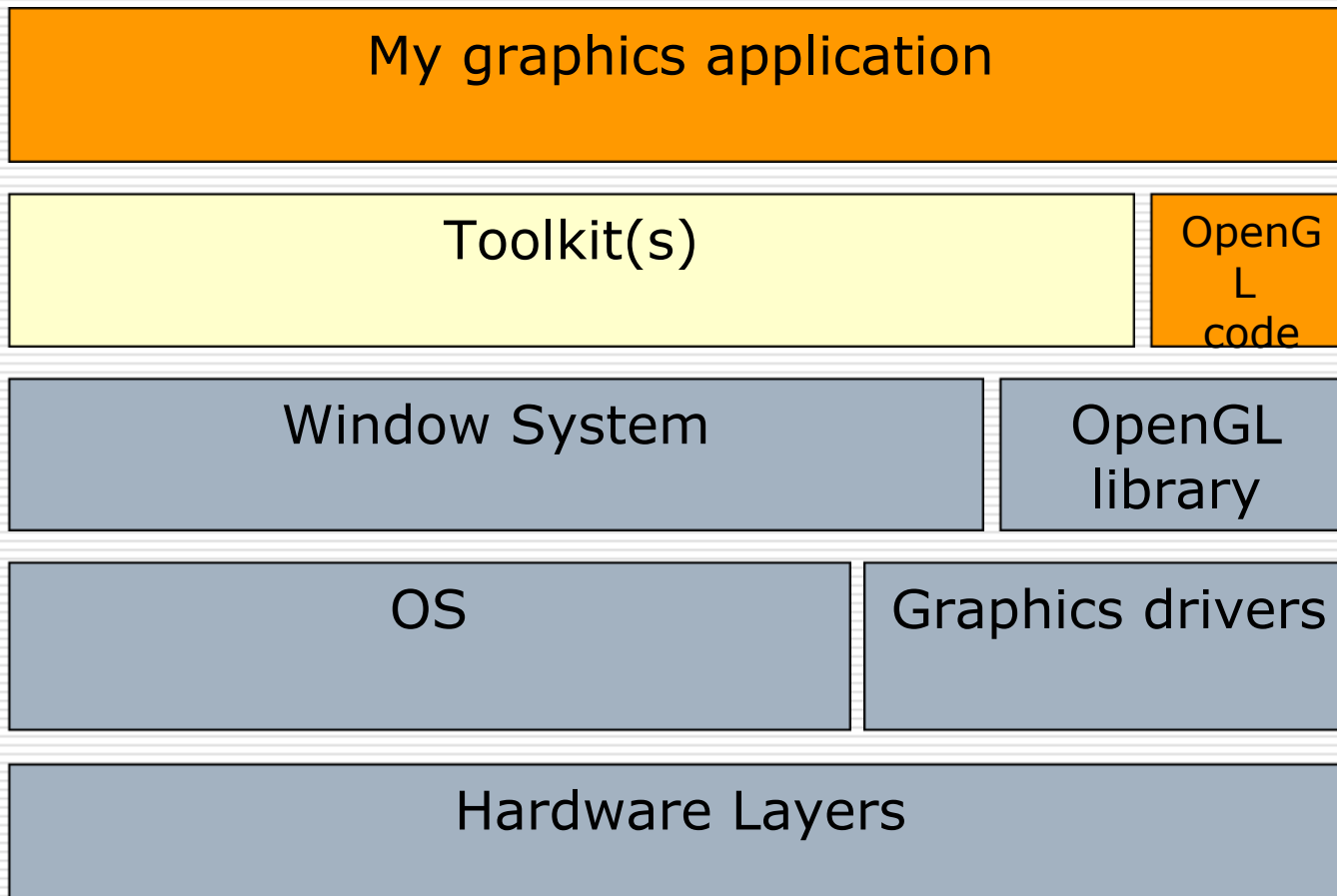
# Graphite

---



# Architecture

---



# Graphite

---

## Pros:

- Portable
  - Modular
  - Powerful  
(rich set of modules)
  - Scalable
  - CG support
  - Python scripting support
-

# Graphite

---

## Cons:

- Proprietary (non-standard) API
  - Learning curve
  - Sparsely documented
-

# Graphite

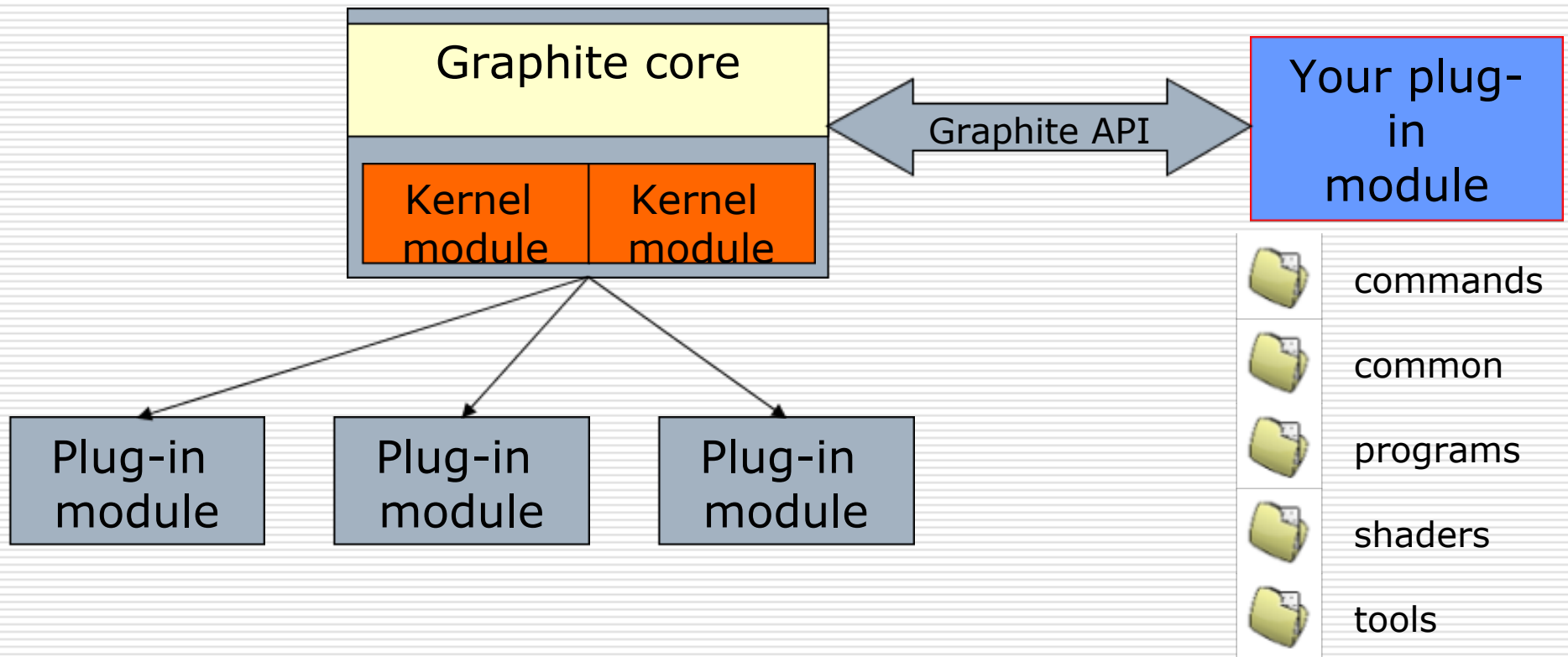
---

Structured in modules:

- Located under: `src/packages/OGF`
  - “Graphite core” manages the event loop and all the UI including the OpenGL context
  - Custom modules use a proprietary graphite API to access the “graphite core”
-

# Graphite

---



# Graphite modules

---

- Works by overriding some graphite classes
  - *quick\_start* module: a toy module to bootstrap your development
  - Shows what classes you need to override
-

# Graphite module components

---

- Common
    - Register module with graphite
  - Commands
    - Menu commands
  - Tools
    - Mouse operations
-

# Graphite modules

---

- Shaders

- Rendering
- Use Graphite rendering API
- Only object drawing
  - Transformations already setup

- Programs

- CG hardware shader programs
-

# Conclusions

---

- Nice/powerful graphics framework
  - Challenging to program
-