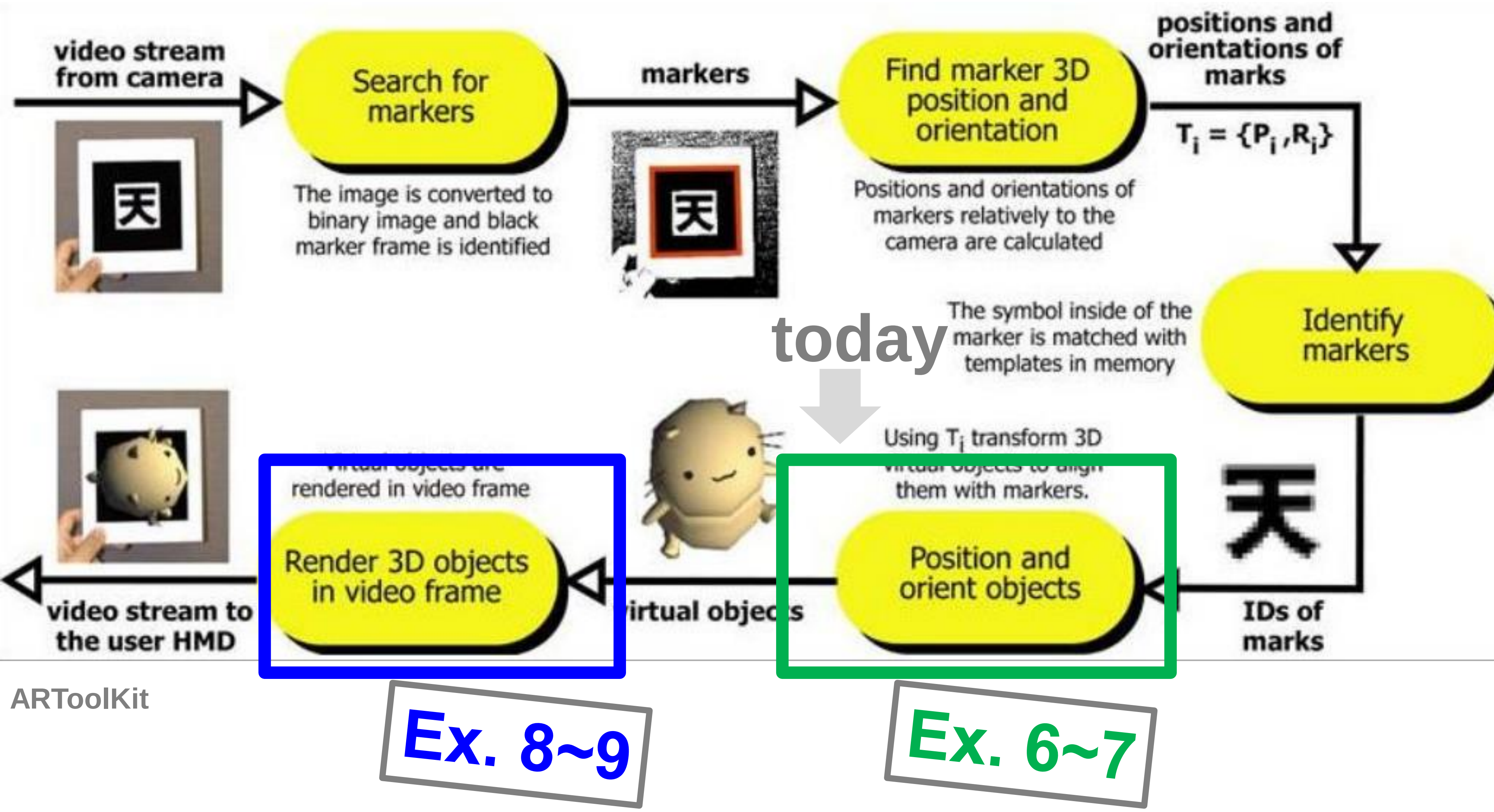
Introduction to Augmented Reality

Tutorial 6: OpenGL

Andreas Langbein, Adnane Jadid, David Plecher

Fachgebiet Augmented Reality
Technische Universität München

Marker-based Tracking


video stream from camera
Search for markers
The image is converted to binary image and black marker frame is identified
markers
Find marker 3D position and orientation
Positions and orientations of markers relatively to the camera are calculated
positions and orientations of marks
$T_i = \{P_i, R_i\}$
Identify markers
The symbol inside of the marker is matched with templates in memory
today
IDs of marks
Using T_i transform 3D virtual objects to align them with markers.
Position and orient objects
virtual objects
Virtual objects are rendered in video frame
Render 3D objects in video frame
video stream to the user HMD
Ex. 8~9
Ex. 6~7

ARToolKit

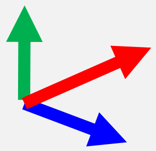
Solution for the Previous Tutorial

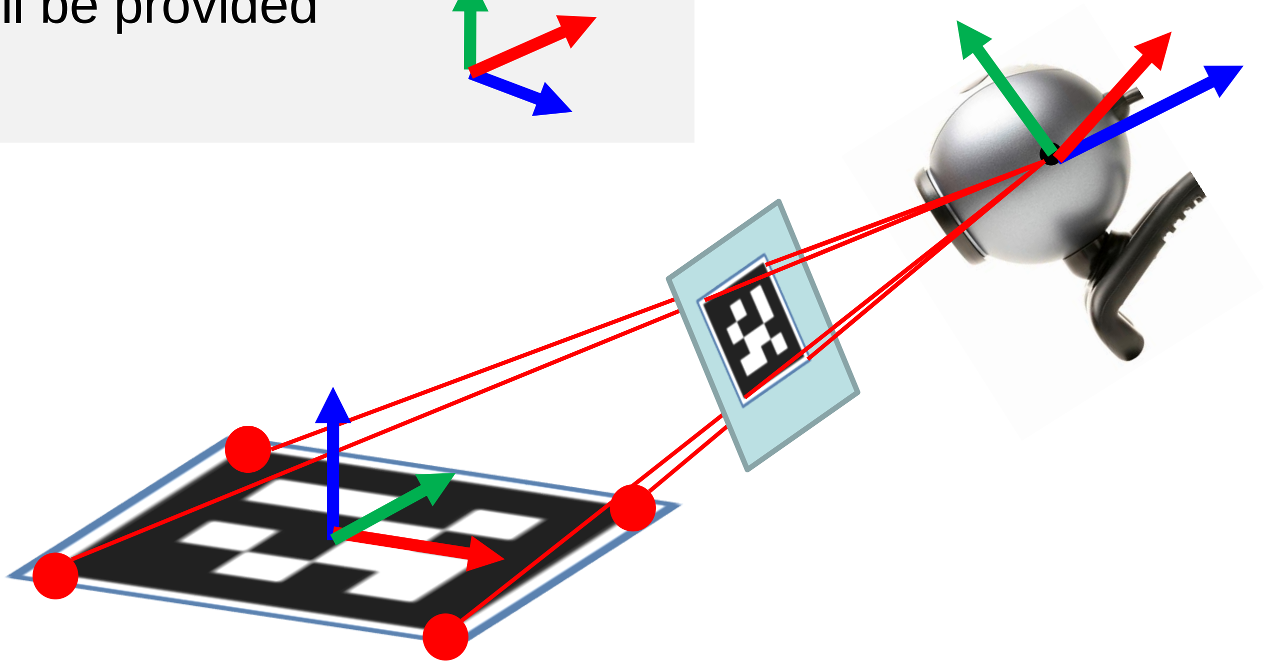
Ex. 5

Marker-Pose Estimation

Pose = Position + Orientation

A code will be provided



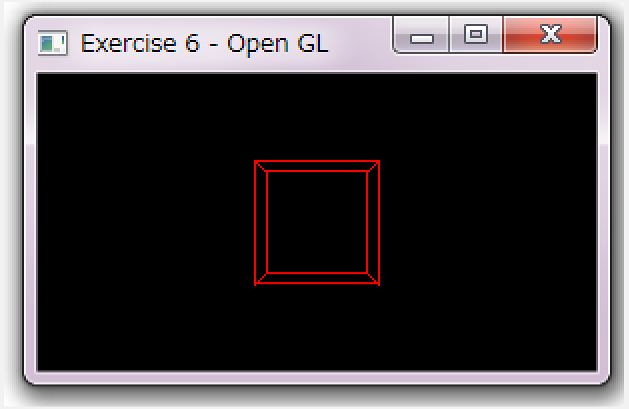


Code walkthrough

Today's Tutorial

Ex. 6

OpenGL basics



Resources

Documentation:

http://www.glprogramming.com/red/

Further information

- http://www.opengl.org/
- http://www.opengl.org/registry/
- http://www.opengl.org/documentation/implementations/
- http://nehe.gamedev.net/

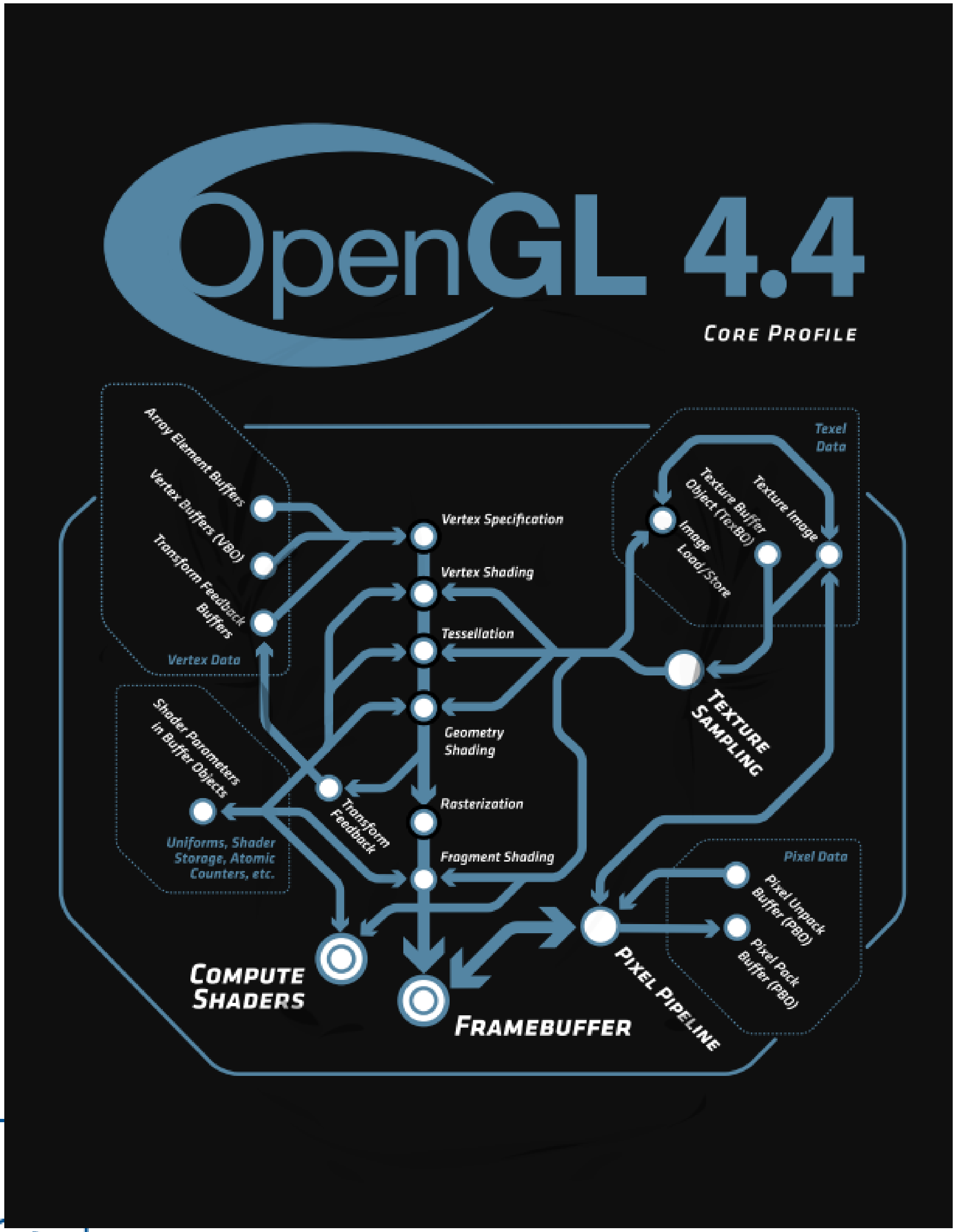
Another resource

An Introduction to OpenGL Programming
@SIGGRAPH 2013
http://www.meetup.com/SV-SIGGRAPH/events/16253618/

by Ed Angel & Dave Shreiner

– A comprehensive tutorial focused on modern ***Shader-based*** OpenGL programming (See course materials for more detail)

OpenGL is a huge API



The latest specification
762 pages

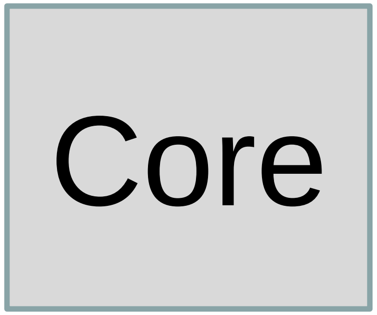
>7000 commands

Thus bit Confusing at First Sight …

gl.h	glx.h	wgl.h
glext.h	glxext.h	wglext.h
glcorearb.h	glu.h	
freeglut.h	glut.h	glew.h
opengl32.lib	glu32.lib	
glut32.lib	glew32.lib	

Basic Structure of OpenGL API

Core (GL_*, gl*)

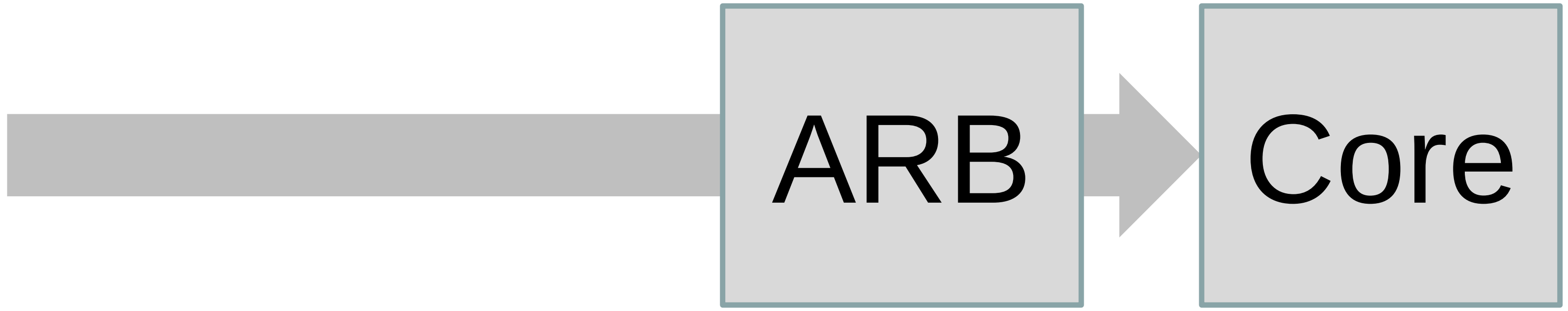


Basic Structure of OpenGL API

Core

Extensions

- ARB
  - Extensions officially approved by the OpenGL **A**rchitecture **R**eview **B**oard
  - (GL_ARB_*)

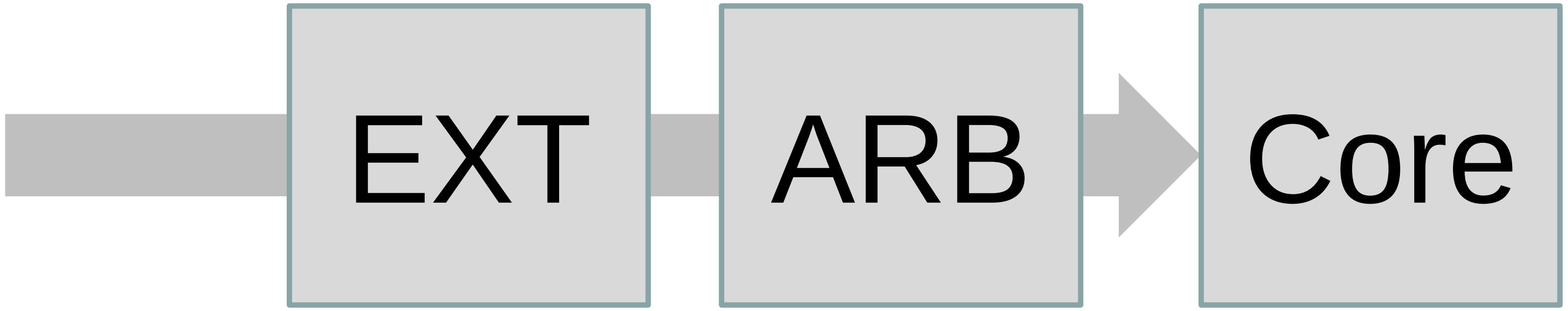


Basic Structure of OpenGL API

Core

Extensions

- ARB
- EXT
  - Extensions agreed upon by multiple OpenGL vendors
  - (GL_EXT_*)

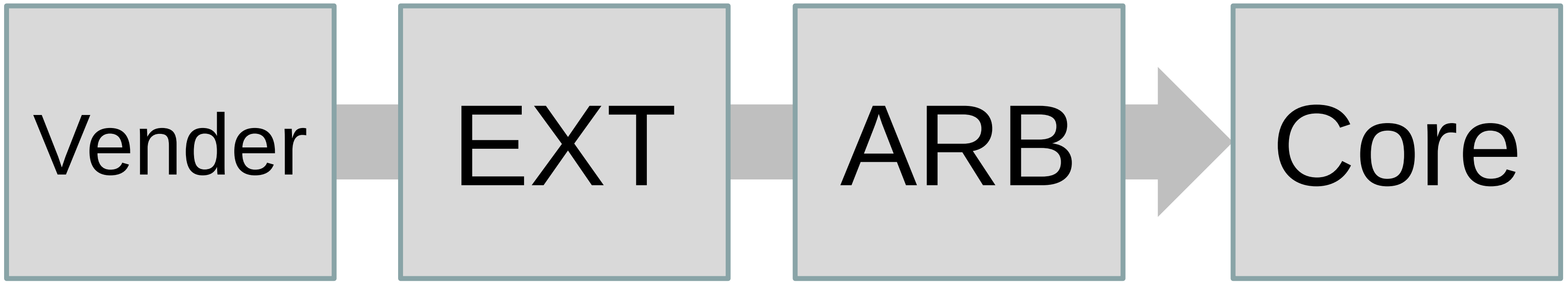


Basic Structure of OpenGL API

Core

Extensions

- ARB
- EXT
- Vender
  - Extensions provided by a single OpenGL vendor
  - (GL_SGI_*, GL_ATI_*, GL_NV_*, GL_INTEL_*, ...)



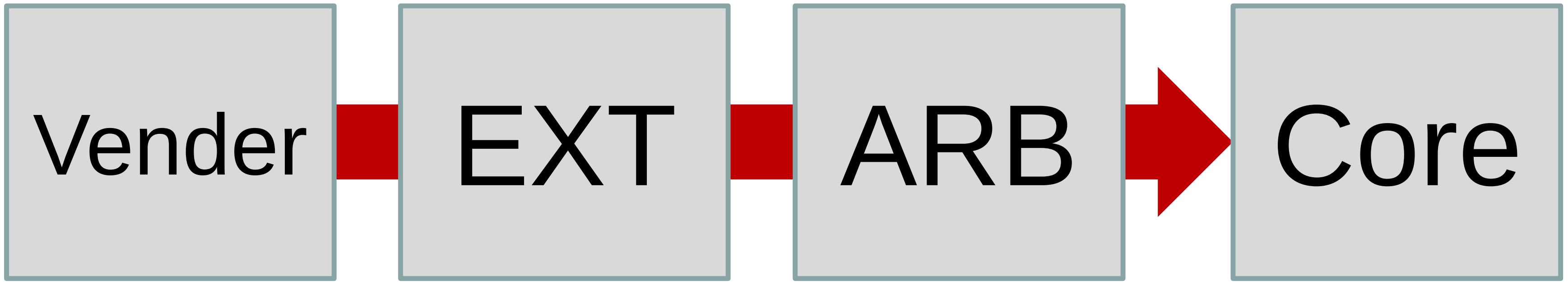
Basic Structure of OpenGL API

Core

Extensions

- ARB
- EXT
- Vender

“Almost all of the new functionality in OpenGL 1.1 and 1.2 showed up first as OpenGL extensions.” *http://www.opengl.org/*

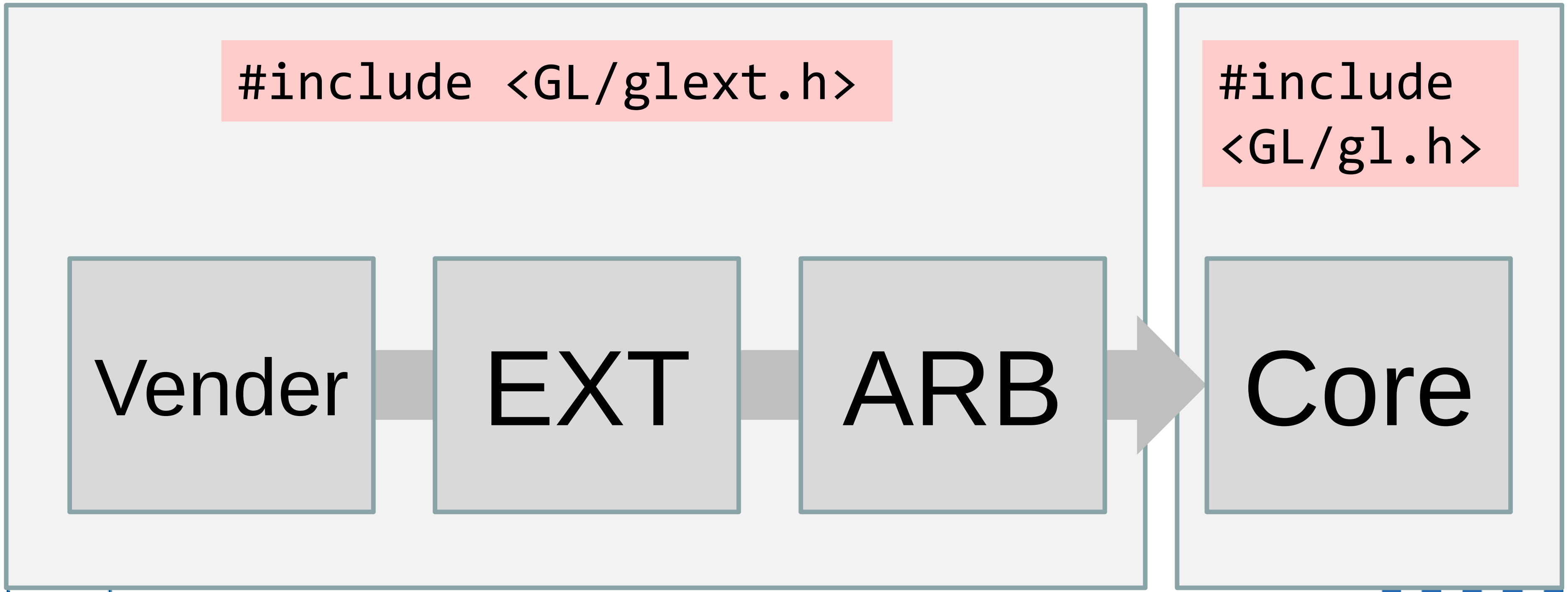


Basic Structure of OpenGL API

OpenGL® Registry provides latest header files

```
#include <GL/glu.h>
```

OpenGL Utility Library (GLU)
e.g. gluLookAt
`glu32.lib`



A Naïve Way to Set Up OpenGL

Include core/extension/utility headers
gl.h/glext.h/glu.h …

Link `opengl32.lib`
(for windows, even for x64 programs…)
And,…

Manually load API functions

via a **platform-specific** API call

Loading OpenGL API

```
glActiveTextureARB =
  (PFNGLACTIVETEXTUREARBPROC)wglGetProcAddress
                                     ("glActiveTextureARB");
```

```
glActiveTextureARB =
  (PFNGLACTIVETEXTUREARBPROC)glXGetProcAddress
                                     ("glActiveTextureARB");
```

This is troublesome…

→ Use an extension loading library

The OpenGL Extension Wrangler Library

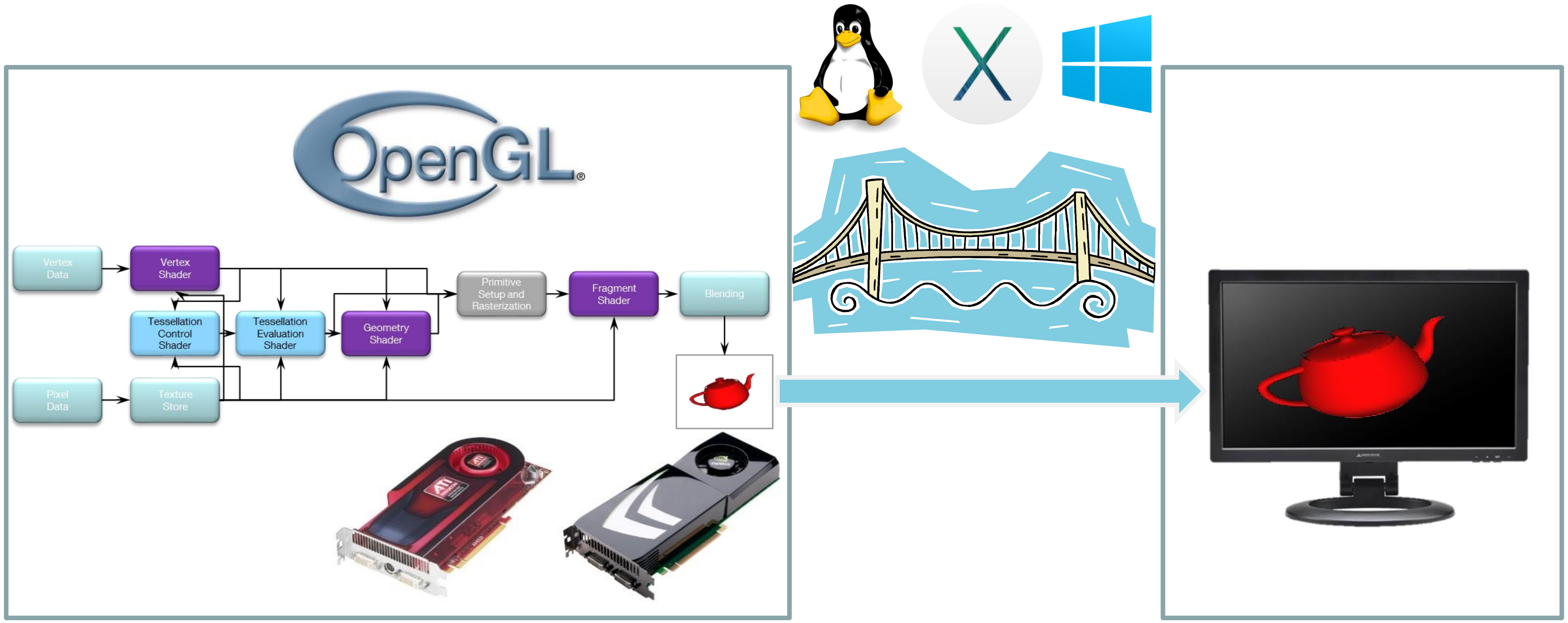
GLEW (The OpenGL Extension Wrangler Library)

Automatically loads supported extensions
cross-platform
glew.h, glew32.lib
http://glew.sourceforge.net/

Window System Management

OpenGL = a graphic rendering engine

=> No mechanisms for creating a rendering surface



OpenGL Window System Interface

Some toolkits exist

- GLUT/freeGLUT, SDL, ...

OpenGL Window System Interface

GLFW

– http://www.glfw.org/

For Mac OS users: brew install glfw

We use GLFW in our exercise

GLFW

Pros

- Can write your own rendering loop
- Cross-platform
- Support for OpenGL 3.2+ with profiles and flags,
- Thread, mutex
- Supports various input devices

Cons:

No primitive rendering functions

→ We provide a code snippet

Easy OpenGL Life with GLFW

Include GLFW header
glfw3.h

Link `opengl32.lib and glfwdll.lib`

```
#include <opencv/cv.h>
#include <opencv/highgui.h>

#define GLFW_INCLUDE_GLU // add support for GLU with GLFW
//#include <GL/glew.h> // if necessary, include before the glfw header
#include <glfw/glfw3.h> /// this also includes other openGL headers

#include "DrawPrimitives.h"
```

Include GLFW in your Visual Studio

Create a new project

Add the ***glfw\include*** directory to
the *Additional Include Directories* in Project Properties
→ Configuration Properties
→ C/C++
→ General

Add ***glfwdll.lib*** to
the *Additional Dependencies* in Project Properties
→ Configuration Properties
→ Linker
→ Input

OpenGL Programming

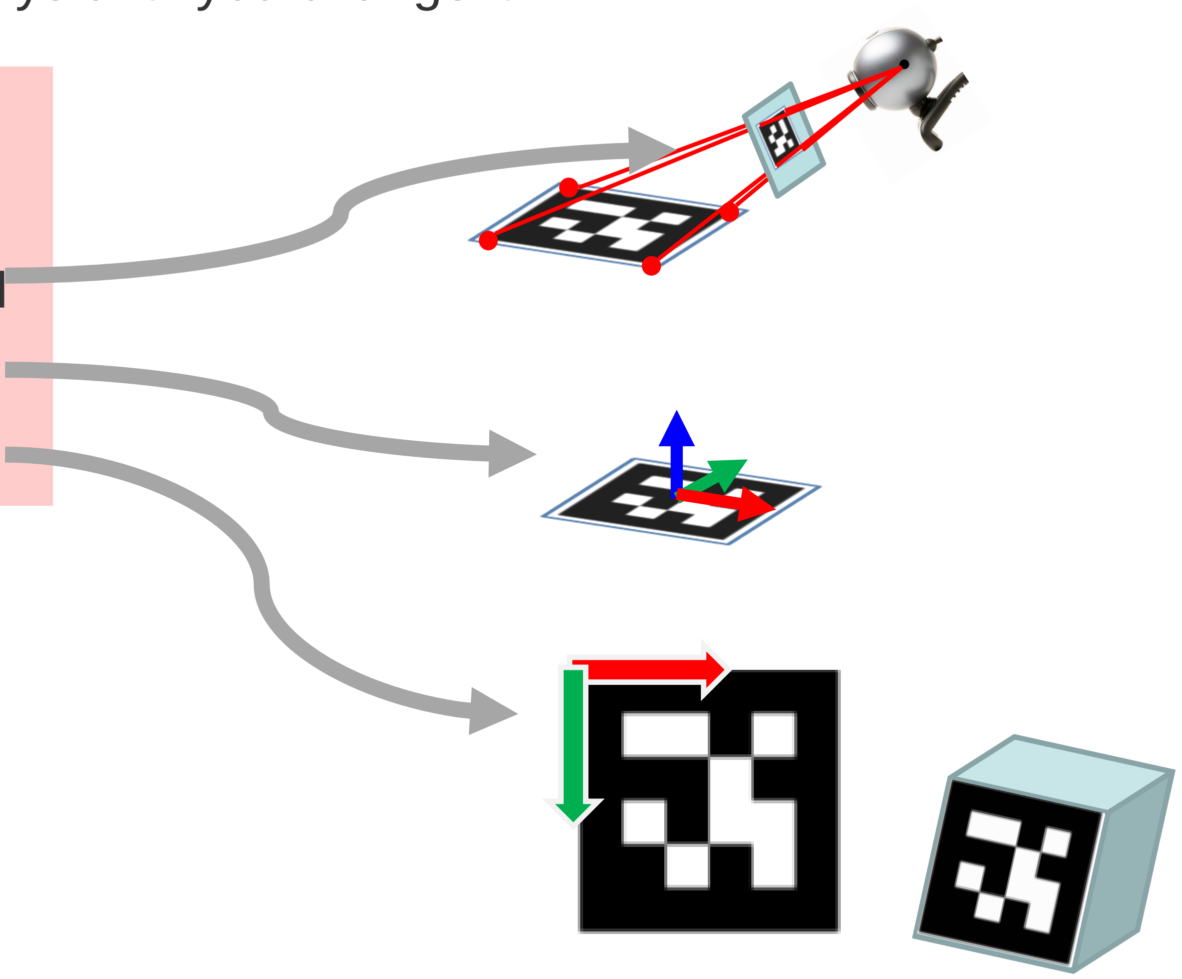
A **state machine**

– Set it once and it stays until you change it

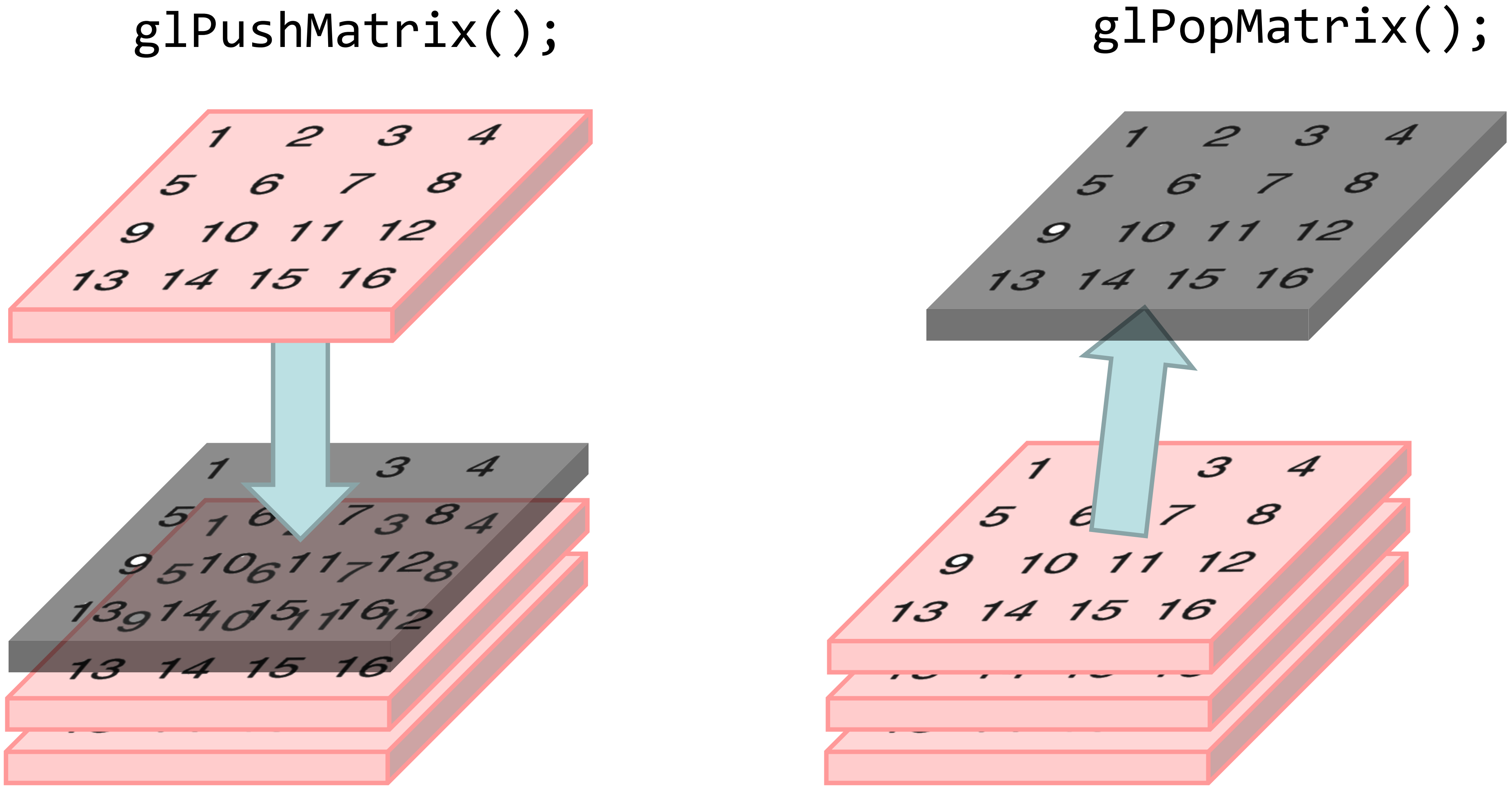
e.g. Matrices

glMatrixMode(XXX);

- GL_PROJECTION
- GL_MODELVIEW
- GL_TEXTURE



Matrix Stacks



Matrix Stack (GL_MODELVIEW)

Matrix Stacks

```
glLoadIdentity();
```

1 0 0 0

0 1 0 0

0 0 1 0

0 0 0 1

Open GL - a basic main

Initialize GLFW (window system)

- GLFWwindow* window;
- glfwInit ()
- glfwCreateWindow(width,height, “WindowName“, ...)
- glfwMakeContextCurrent(window)

Initialize OpenGL

- glEnable (GL_COLOR_MATERIAL)
- glClearColor(0.0, 0.0, 0.0, 1.0)

Enable and set depth

- glEnable(GL_DEPTH_TEST)
- glClearDepth(1.0)

Open GL - a basic main

Configure display update timing by GLFW

```
glfwSwapInterval (1)
```

Register functions for GLFW

```
glfwSetFramebufferSizeCallback(window,resize);
```

Start a rendering loop

```
while( !glfwWindowShouldClose(window) )
{
    ...
}
```

Open GL - resize (GLFWwindow* window int width, int height)

Set screen in a window

- glViewport(x, y, width, height)

Create perspective projection for a virtual camera

- glMatrixMode(GL_PROJECTION)
- glLoadIdentity()
- gluPerspective
  (angle, aspectratio, near-clipping, far-clipping)

Open GL - display (GLFWwindow* window)

Clear buffers

- glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

Render sphere

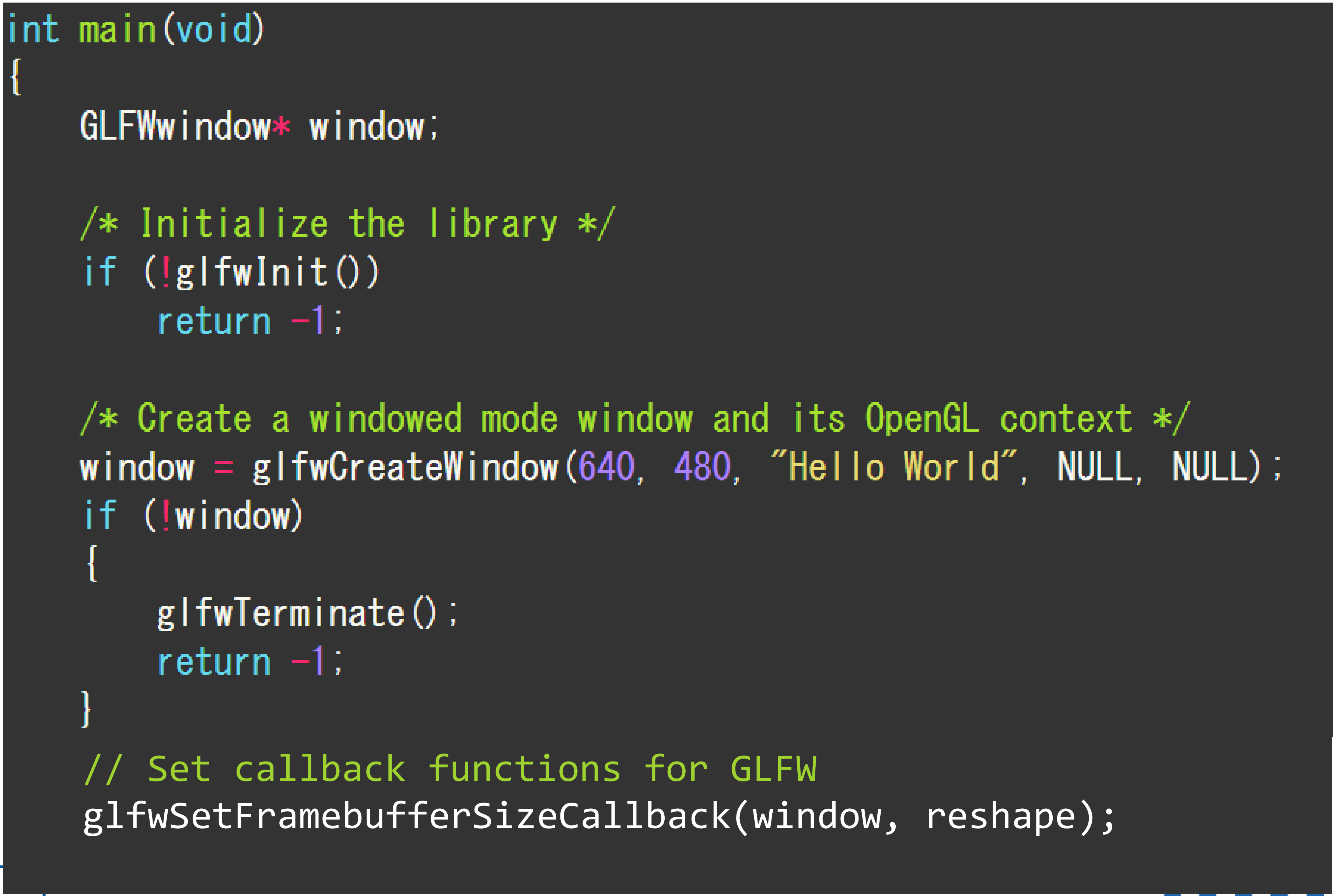
- glMatrixMode (GL_ MODELVIEW)
- glLoadIdentity()

- glTranslate(0.0, 0.0, -5.0)
- glColor4f(1.0, 0.0, 0.0, 1.0)
- drawsphere (1.0,10,10) // We provide a code snipet

Swap Buffers by GLFW

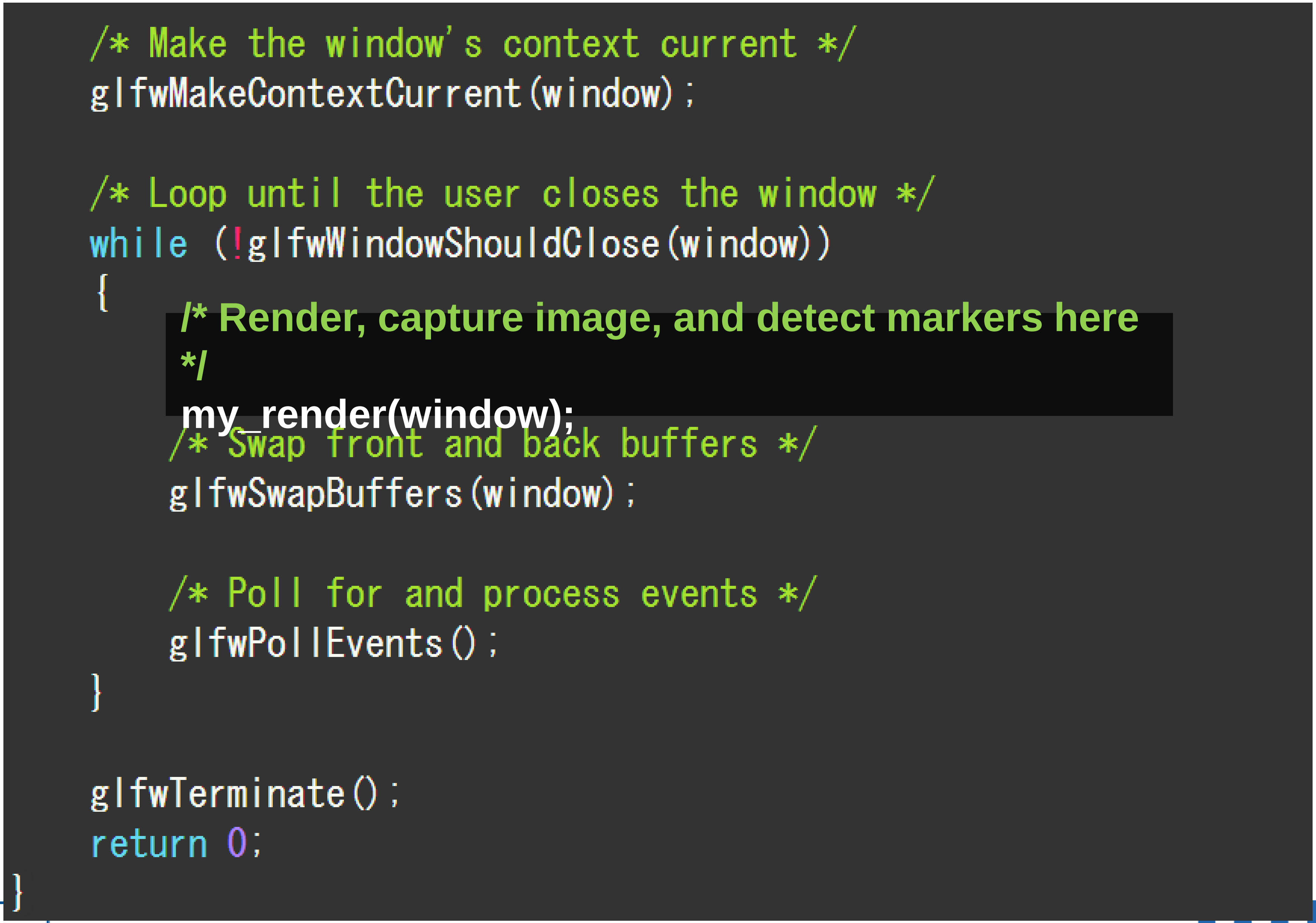
- glfwSwapBuffers

Sample Code



```
int main(void)
{
    GLFWwindow* window;

    /* Initialize the library */
    if (!glfwInit())
        return -1;

    /* Create a windowed mode window and its OpenGL context */
    window = glfwCreateWindow(640, 480, "Hello World", NULL, NULL);
    if (!window)
    {
        glfwTerminate();
        return -1;
    }
    // Set callback functions for GLFW
    glfwSetFramebufferSizeCallback(window, reshape);
```

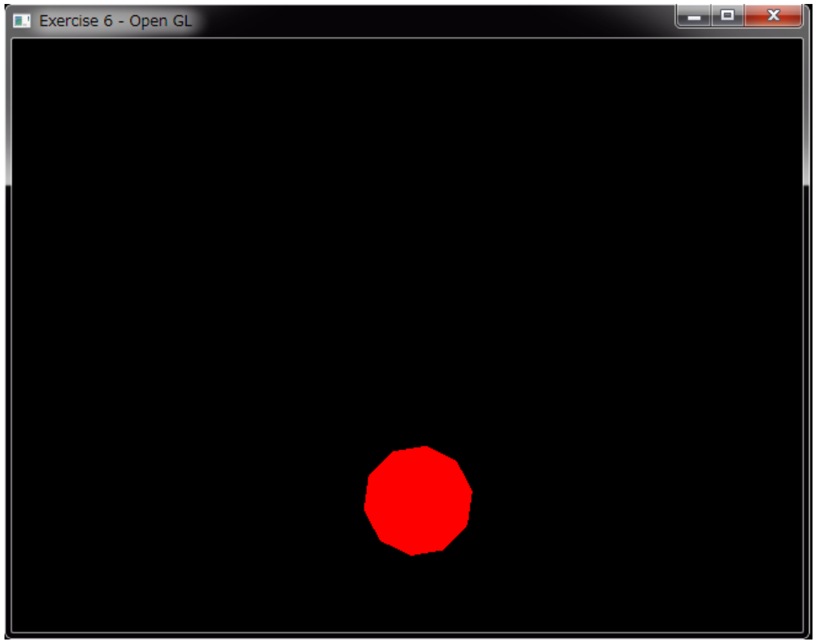
Sample Code continued



```
    /* Make the window's context current */
    glfwMakeContextCurrent(window);

    /* Loop until the user closes the window */
    while (!glfwWindowShouldClose(window))
    {
        /* Render, capture image, and detect markers here */
        my_render(window);
        /* Swap front and back buffers */
        glfwSwapBuffers(window);

        /* Poll for and process events */
        glfwPollEvents();
    }

    glfwTerminate();
    return 0;
}
```

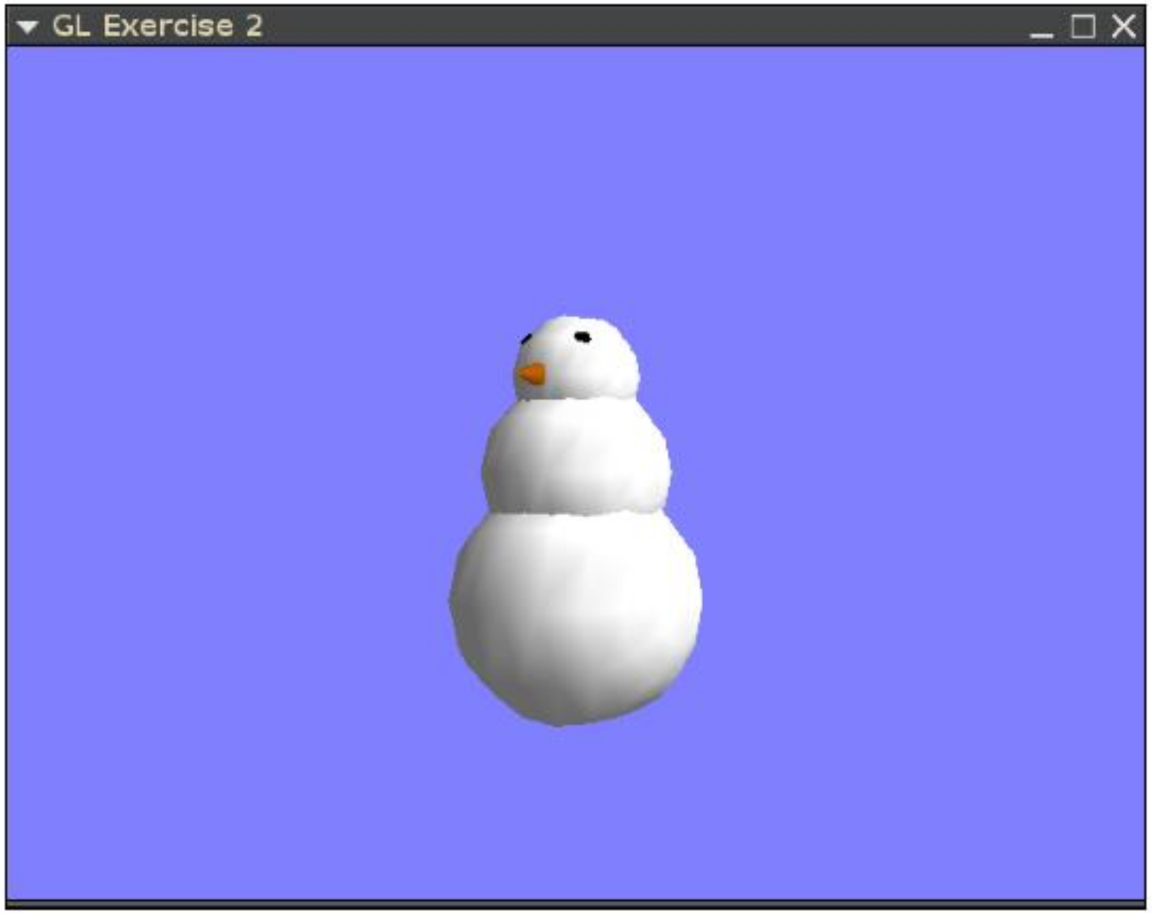
Homework

Implement OpenGL application with GLFW



Spoiler of the next tutorial

– OpenGL snowman
– Combine snowman with marker tracker



That‘s it…

- Questions