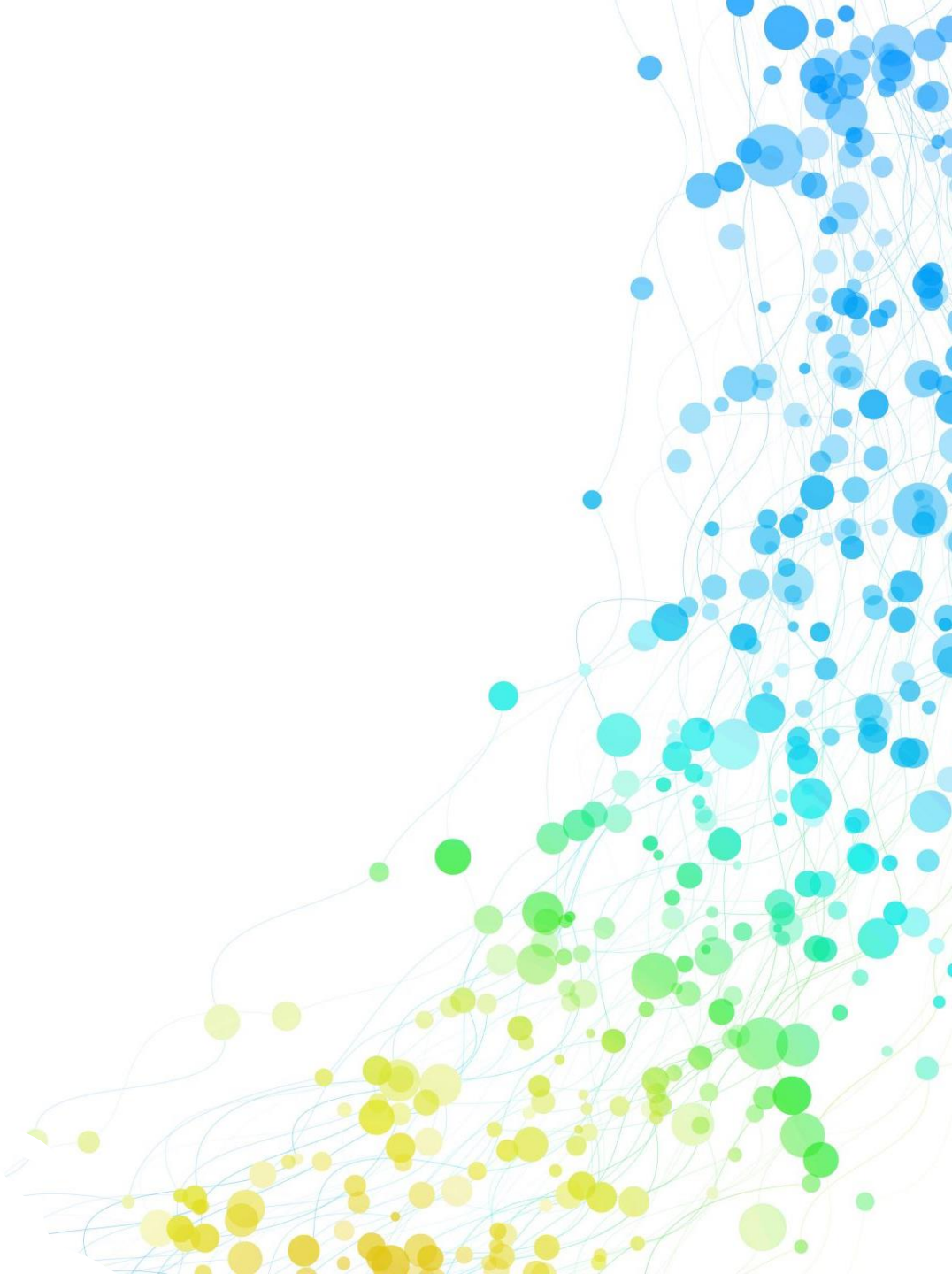
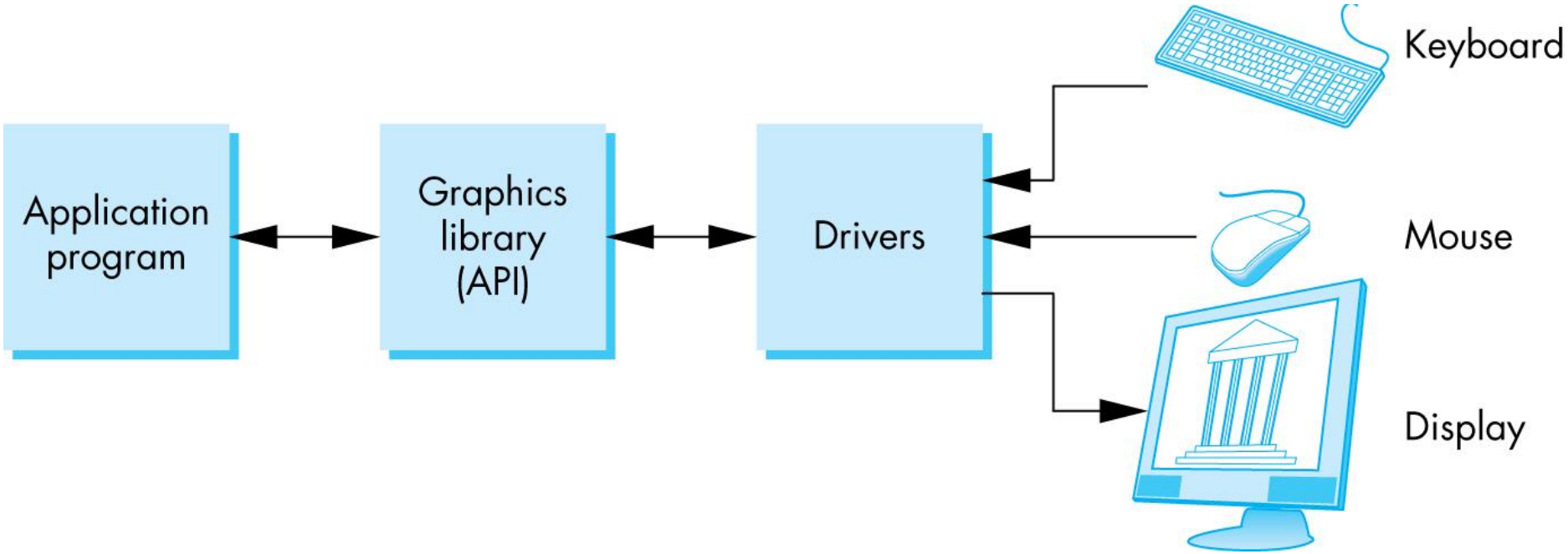
Graphics Programming

3RD WEEK, 2022



The Programmer's Interface

- Programmer sees the graphics system through a software interface
  - Application Programming Interface (API)



API Contents

API의 목적

- Functions that specify what we need to form an image
  - Objects
  - Viewer
  - Light Sources
  - Materials

- Other information
  - Input from devices such as mouse and keyboard
  - Capabilities of system

Object Specification

- Most APIs support a limited set of primitives including
  - Points
  - Line segments
  - Polygons
  - Some curves and surfaces
    - Quadrics
    - Parametric polynomials

- All are defined through locations in space or vertices

Example (GPU based)

1. • Put geometric data in an array

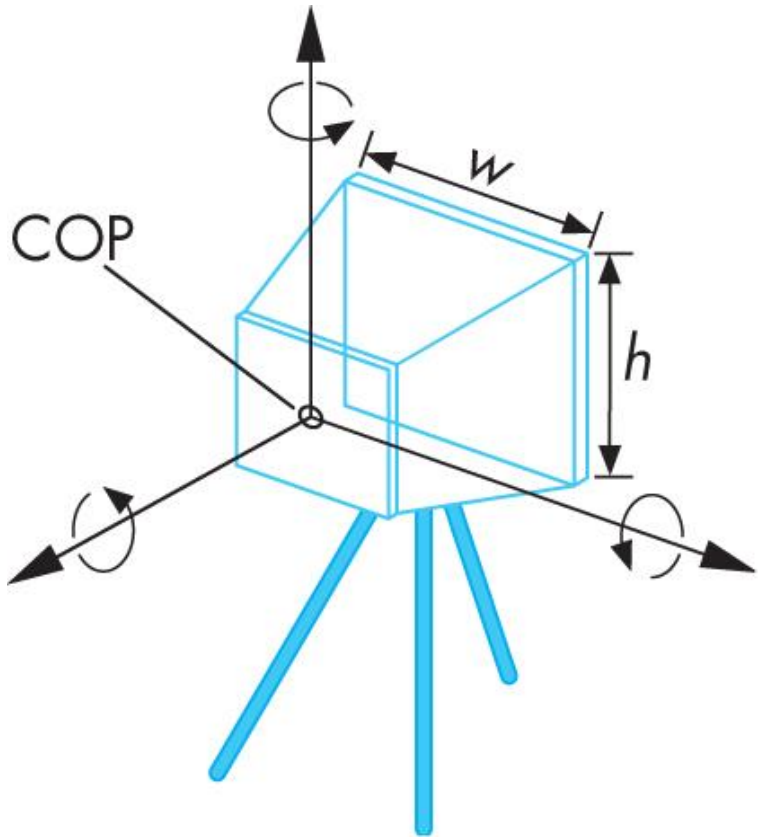
```
vec3 points[3];
points[0] = vec3(0.0f, 0.0f, 0.0f);
points[1] = vec3(0.0f, 1.0f, 0.0f);
points[2] = vec3(0.0f, 0.0f, 1.0f);
```

2. • Send array to GPU

3. • Tell GPU to render as triangle

Camera Specification

- Six degrees of freedom
  - Position of center of lens
  - Orientation
- Lens
- Film size
- Orientation of film plane



Lights and Materials

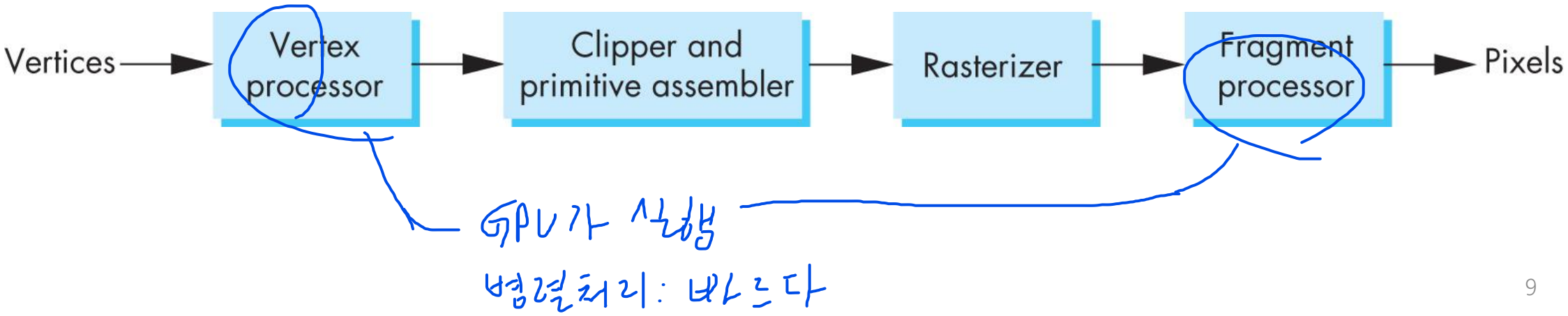
- Types of lights
  - Point vs. directional light sources
  - Spotlight s
  - Near and far sources
  - Color properties
- Material properties
  - Absorption: color properties
  - Scattering: diffuse and Specular components

OpenGL

- A platform-independent API that was
  - Easy to use
  - Close enough to the hardware to get excellent performance
  - Focus on rendering
  - Omitted windowing and input to avoid window system dependencies

Modern OpenGL

- Performance is achieved by using GPU rather than CPU
- Control GPU through programs called shaders
- Application's job is to send data to GPU
- GPU does all rendering



OpenGL 3.1

- Totally shader-based
  - No default shaders
  - Each application must provide both a vertex and a fragment shader
- No immediate mode
- Few state variables
- Most 2.5 functions deprecated
- Backward compatibility not required

Retained Mode Graphics

유지하다

- Put all vertex and attribute data in array
- Send array to GPU to be rendered immediately
- Almost OK but problem is we would have to send array over each time we need another render of it
- Better to send array over and store on GPU for multiple renderings

Other Versions

- OpenGL ES
  - Embedded systems
  - Version 1.0 simplified OpenGL 2.1
  - Version 2.0 simplified OpenGL 3.1
    - Shader-based
- WebGL
  - JavaScript implementation of ES 2.0
  - Supported on newer browsers
- OpenGL 4.1 and 4.2
  - Add geometry shaders and tessellator

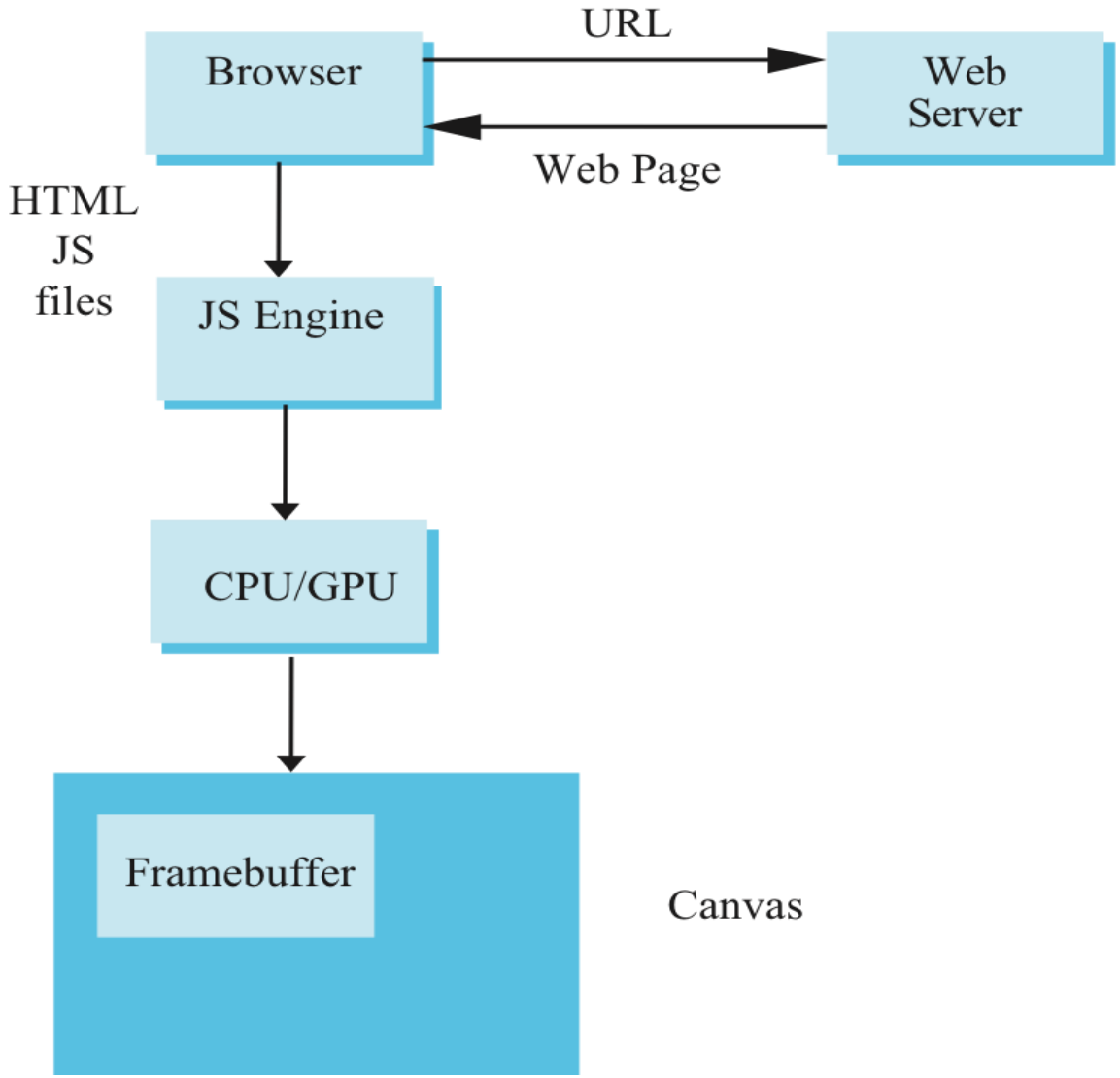
GLSL

- OpenGL Shading Language
- C-like with
  - Matrix and vector types (2, 3, 4 dimensional)
  - Overloaded operators
  - C++ like constructors
- Similar to Nvidia's Cg and Microsoft HLSL
- Code sent to shaders as source code
- New OpenGL functions to compile, link and get information to shaders

WebGL and GLSL

- WebGL requires _____s and is based less on a state machine model than a data flow model
- Most state variables, attributes and related pre 3.1 OpenGL functions have been deprecated
- Action happens in shaders
- Job of application is to get data to GPU

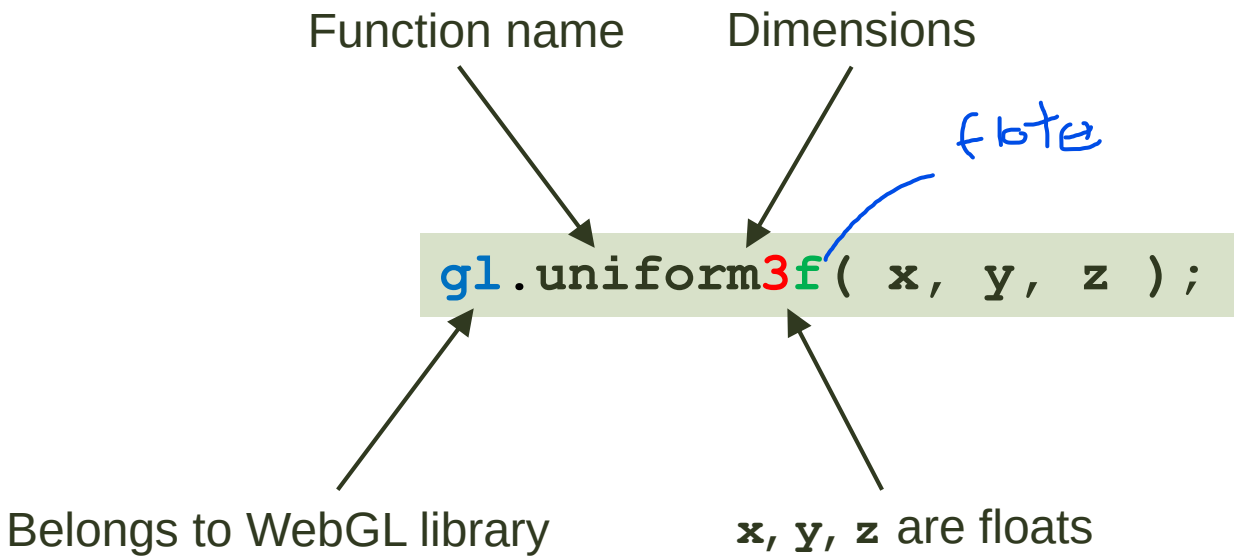
Execution in Browser



Event Loop

- Remember that the sample program specifies a render function which is an event listener or callback function
  - Every program should have a render callback
  - For a static application we need only execute the render function once
  - In a dynamic application, the render function can call itself recursively but each redrawing of the display must be triggered by an event

WebGL Function Format



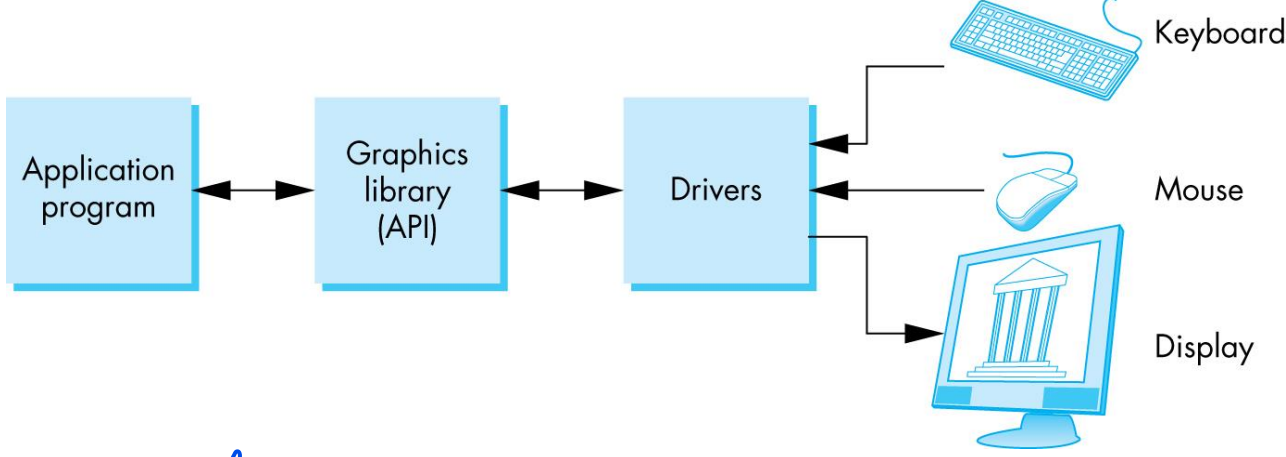

```
gl.uniform3fv( p );
```

p is a pointer to an array

WebGL Constants

- Most constants are defined in the canvas object
  - In desktop OpenGL, they were in #include files such as **`gl.h`**
- Examples
  - desktop OpenGL
    - **`glEnable(GL_DEPTH_TEST);`**
  - WebGL
    - **`gl.enable(gl.DEPTH_TEST);`**
  - **`gl.clear(gl.COLOR_BUFFER_BIT);`**

Summary



- OpenGL == API (Application Programming Interface)
  - To specify object s, a viewer, light source (s), and material s
- A platform-independent API
- Control GPU through programs called shader s
- OpenGL 3.1 == OpneGL ES 2.0 == WebGL
- Retained mode vs. immediate mode
- GLSL (OpenGL Shading Language)
  - WebGL requires shader s