# GLSL

## OpenGL Shading Language

CS3314 Programming Languages
Dec 10, 2019
Yuxi Luo

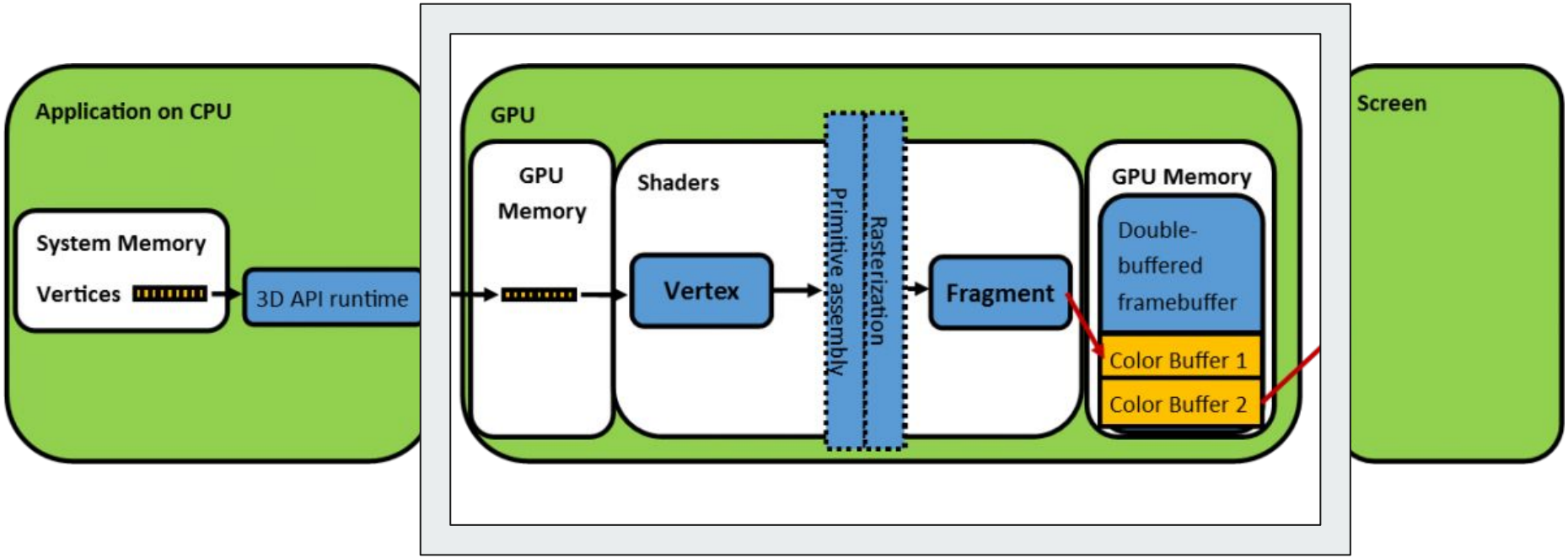
# Overview

- Background
- Language
- Demo
- Reference

# Background

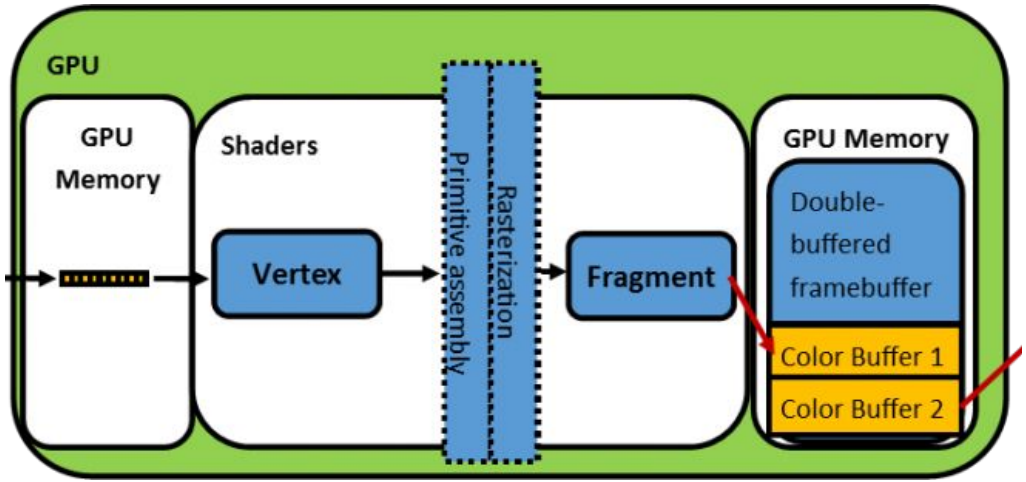
# Graphics Pipeline

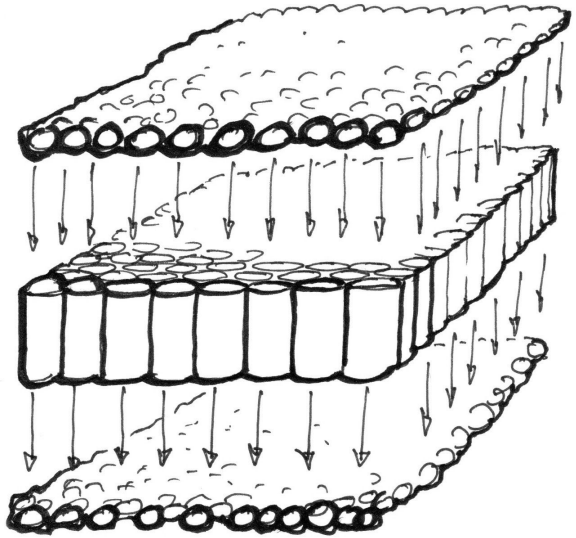
- Purpose: Display (3D) geometric objects on 2D screen



# Graphics Pipeline

- Purpose: Display 3D objects on 2D screen



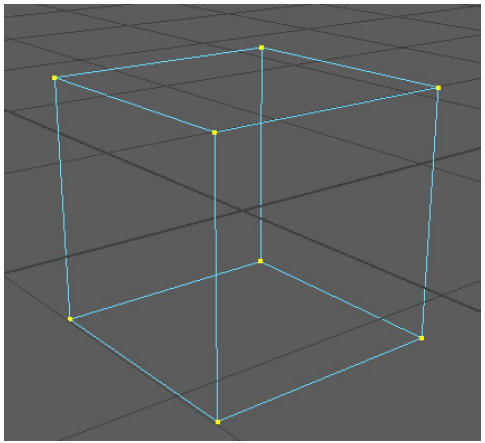



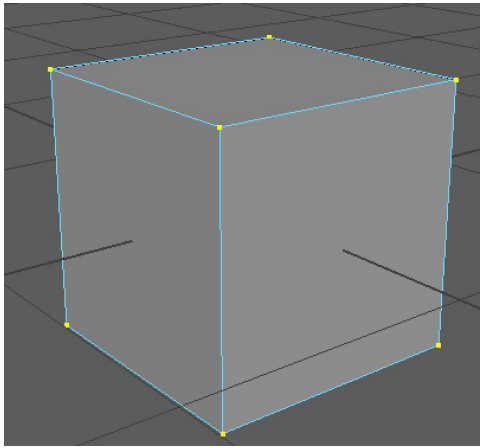
# Shaders

- Vertex Shader
  - Control the position of each vertex
- Fragment Shader
  - Control the color of each shape

- Application -> Vertex Shader -> Fragment Shader -> Display

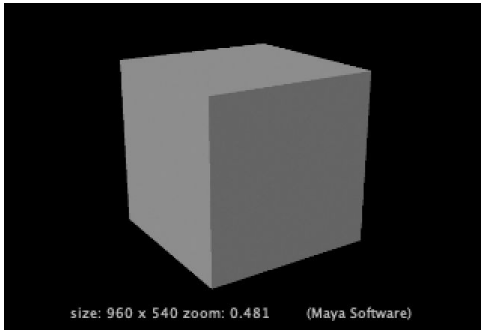
# Shaders

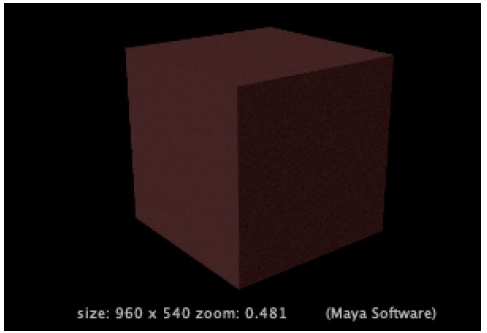




- Vertex Shader

- Fragment Shader



# Definition

- GLSL
  - OpenGL Shading Language
  - Interact with shaders (vertex & fragment)
- OpenGL
  - Cross-platform graphics API: interact directly with GPU
  - Register geometric object information to GPU

# Language

# GLSL

- C-style language
- Built-in math library (vector, matrix, …)
- Control branches, function definitions
- [DOES NOT SUPPORT RECURSION]

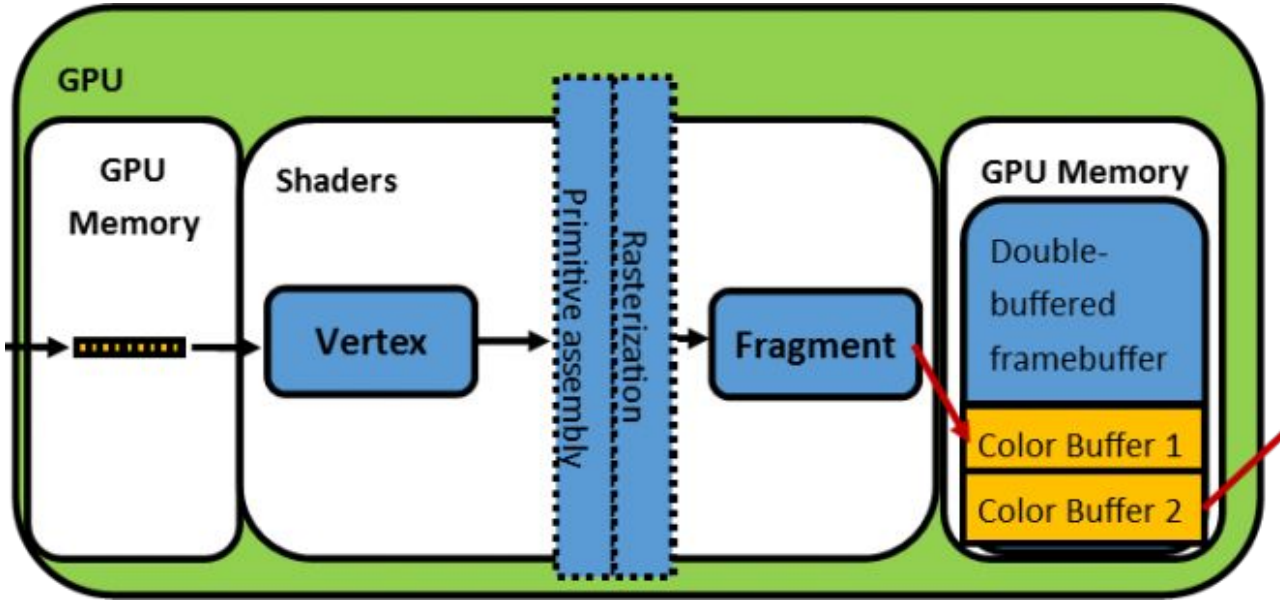
# Data Type

- Scalars: bool | int | uint | float | double
- Vectors: vecn (e.g vec3)
  - vec3 pos = vec3(0.3, 0.3, 0.3);
  - pos.z = pos.x + pos.y;
  - vec3 anotherPos = pos.zyy; // swizzling
- Matrices

# Qualifiers

- attribute:
  - shared between application and vertex shader
  - vertex-specific data (e.g vertex position)
- uniform:
  - shared between application and shader
  - object properties (e.g light positions, projection matrix)
- varying:
  - shared between vertex shader and fragment shader
  - information from vertex shader (e.g effects of lighting)

# Qualifiers

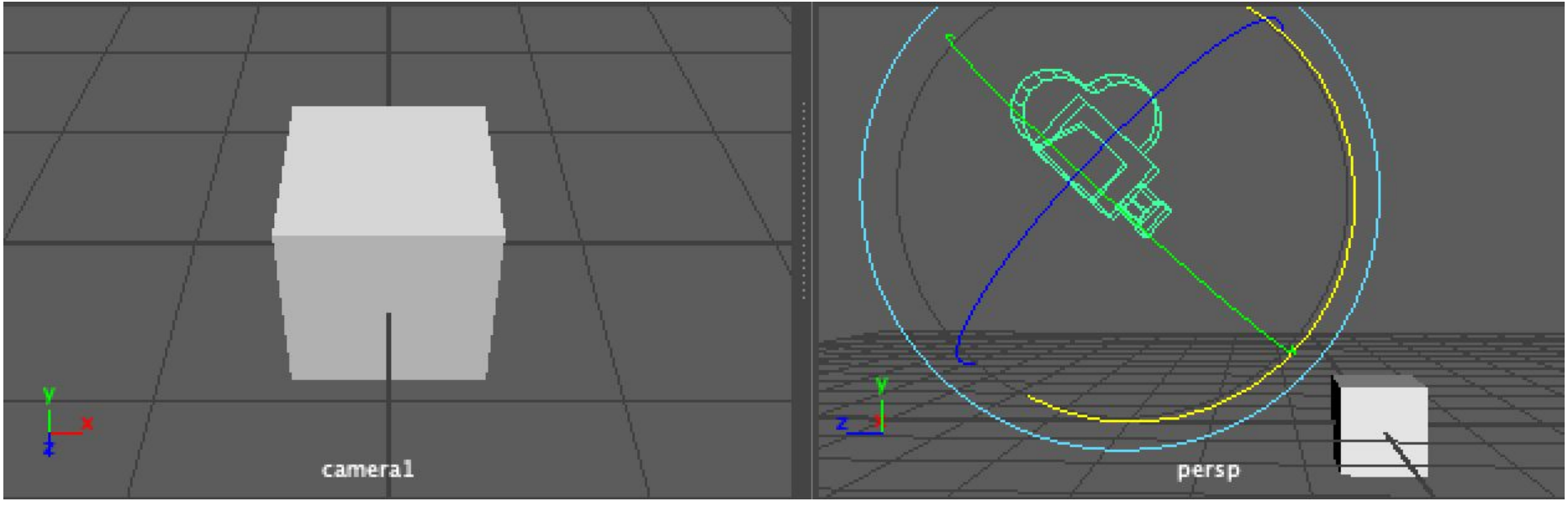

GPU
GPU Memory
Shaders
Vertex
Primitive assembly
Rasterization
Fragment
GPU Memory
Double-buffered framebuffer
Color Buffer 1
Color Buffer 2

# Sample Vertex Shader

```
1   /* GLSL 120 */
2   attribute vec3 vPosition;
3   attribute vec3 vColor;
4   varying   vec4 color;
5
6   uniform mat4 model_view;
7   uniform mat4 projection;
8
9   void main() {
10      vec4 vPosition4 = vec4(vPosition.x, vPosition.y, vPosition.z, 1.0);
11      vec4 vColor4 = vec4(vColor.r, vColor.g, vColor.b, 1.0);
12
13      gl_Position = projection * model_view * vPosition4;
14      color = vColor4;
15  }
```

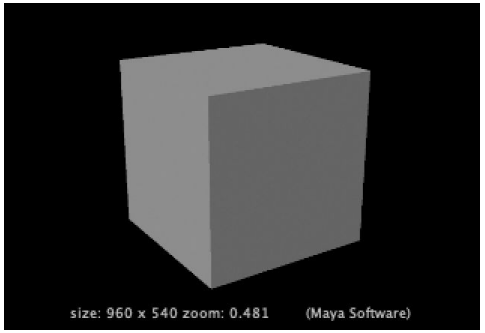
# Sample Vertex Shader

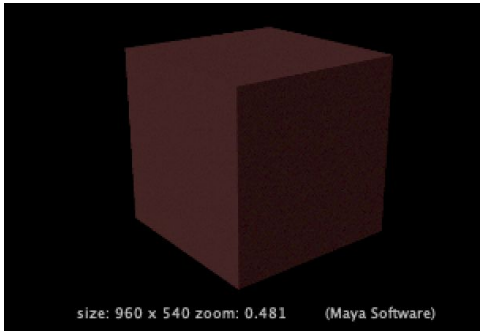
- Model - View - Projection matrices:
- Describe the relationship between the object and camera



# Sample Fragment Shader

```
1   /* GLSL 120 */
2   varying vec4 color;
3
4   void main() {
5       gl_FragColor = color;
6   }
7
```



# Demo

# References

# References

- LearnOpenGL: https://learnopengl.com/
- The Book of Shaders: https://thebookofshaders.com/
- GLSL Qualifiers: http://www.shaderific.com/glsl-qualifiers