


Doodle here...





Introduction

OpenGL

What is OpenGL?

- OpenGL is an API (Application Programming Interface)
- Set of specifications developed by the Khronos Group (<https://www.khronos.org/>)



- Cross Platform Graphics Library
- Usually developed by the graphic cards manufacturer
- Update your graphic drivers

OpenGL – Core vs Compatibility

- Old OpenGL – uses fixed function pipeline
 - Not a lot of control
 - Most of things hidden from the programmer
 - Inefficient
- New OpenGL – version 3.3 or higher
- New OpenGL is split in two:

Core

Compatibility
(to maintain legacy)

- Core:
 - Modern version
 - More flexibility and control for the programmer

OpenGL – State Machine

- OpenGL is a large **State Machine**:
 - a collection of variables that define how OpenGL should currently operate
- Context
 - What is most commonly referred to as the state of OpenGL
- We change the state of OpenGL by changing some context variable that sets how OpenGL should draw.
- State-changing functions
 - Functions that change the context of OpenGL
- State-using functions
 - Functions that perform some operations based on the current state of OpenGL.
- Objects
 - A collection of objects that represents a subset of OpenGL' state



Drawing in OpenGL – basic steps

1. Create buffer
2. Add data to the buffer (vertices)
3. Create shaders
4. Define the vertex format for the mesh vertices
5. Draw the mesh
 - Set the shader as current
 - Set the buffer as current
 - Draw the buffer



Drawings in OpenGL

- GL_POINTS
- GL_LINES
- GL_LINE_STRIP
- GL_LINE_LOOP
- GL_TRIANGLES
- GL_TRIANGLE_STRIP
- GL_TRIANGLE_FAN



What is GLSL?

- OpenGL Shading Language (GLSL)
- GLSL is a mini-program also known as a **shader program**
- Enables the programmer to harness the power of the Graphics Processing Unit (GPU)
- It is both fundamental and integral to OpenGL

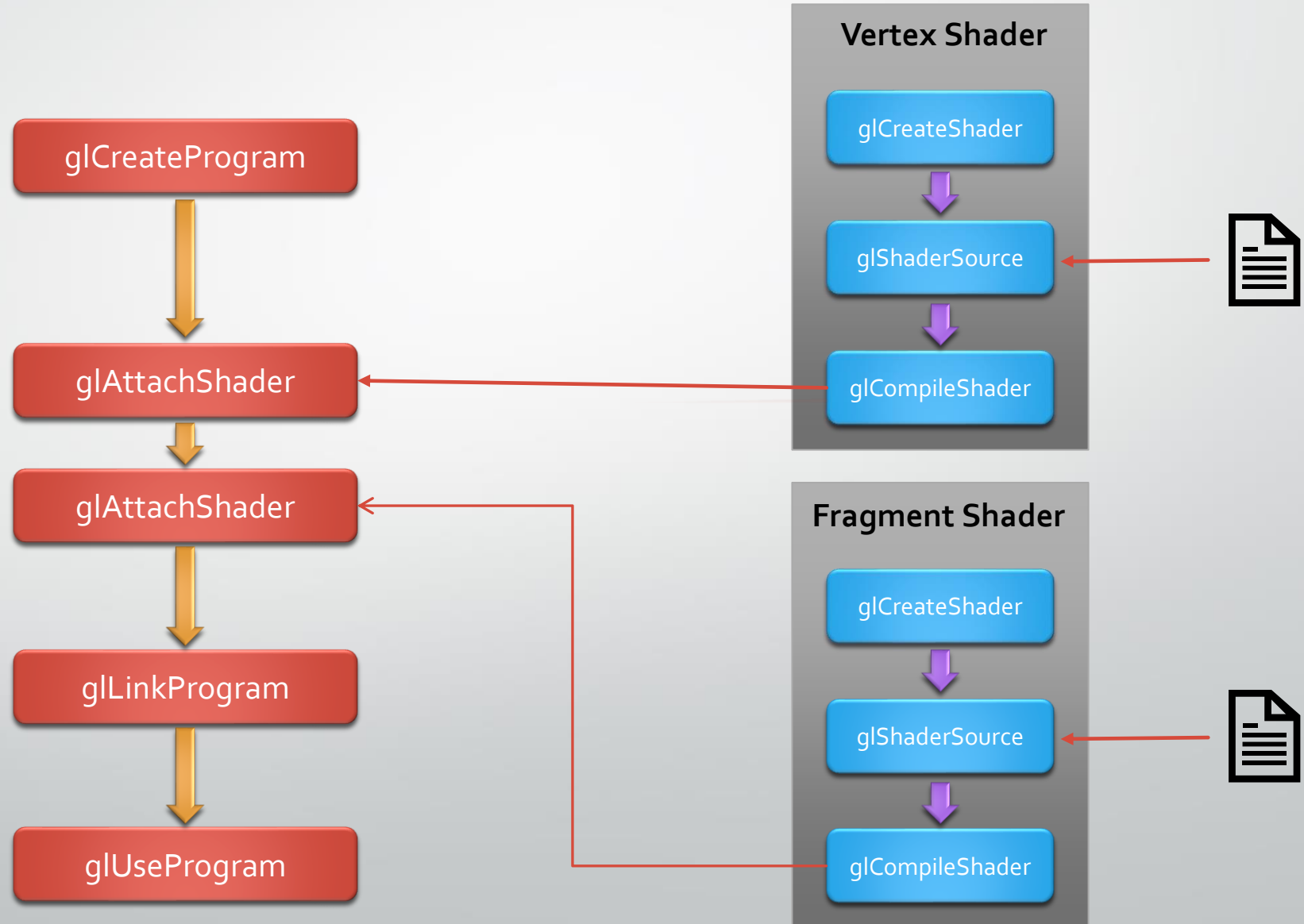
GLSL – Common data types

- Built-in Simple types: float, double, bool, int, uint
- Vectors:
 - Floats: vec2, vec3, vec4
 - Double: dvec2, dvec3, dvec4
 - Booleans: bvec2, bvec3, bvec4
 - Ints: ivec2, ivec3, ivec4
 - Unsigned ints: uvec2, uvec3, uvec4
- Square matrices:
 - Floats: mat2, mat3, mat4
 - Double: dmat2, dmat3, dmat4
- Non-Square matrices:
 - Floats: mat(2,3,4) × (2,3,4)
 - Double: mat(2,3,4) × (2,3,4)

mat2 = 2x2 matrix

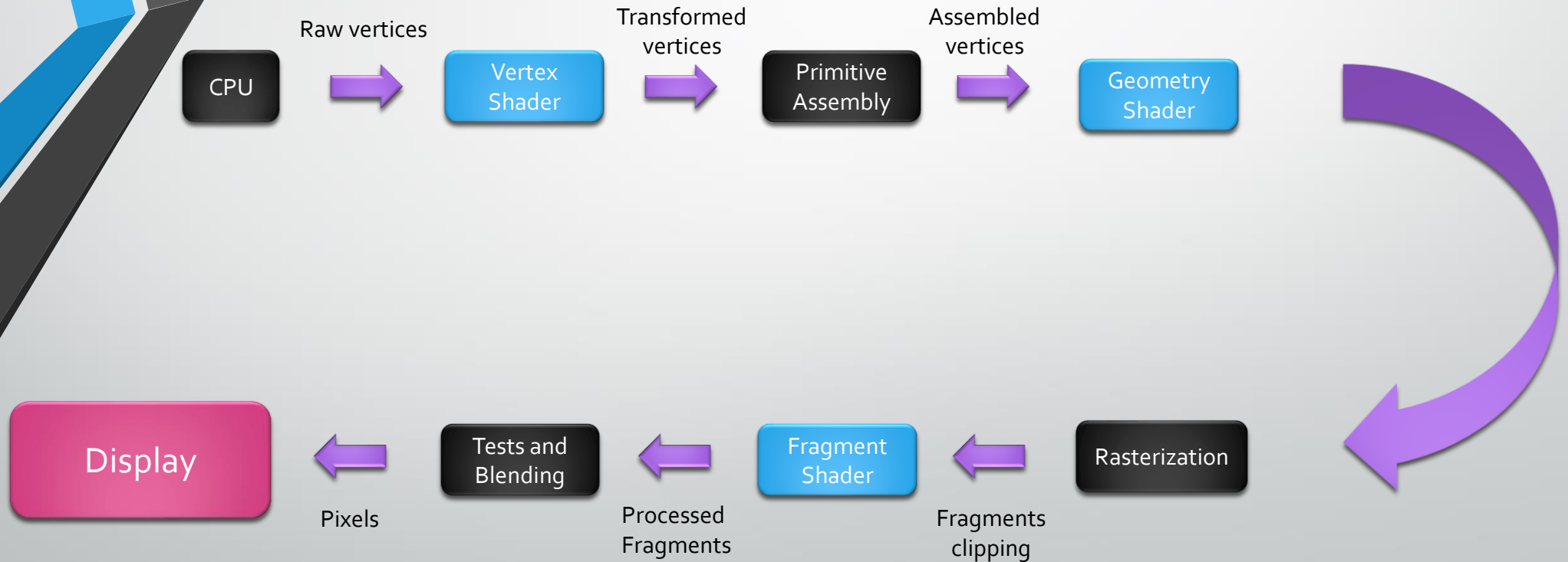
mat2x4 = 2x4 matrix

Creating a shader program



Graphics Pipeline - simplified

Transformation Process - transforming coordinates from 3D to 2D



Shader Structure

```
#version <version_number>
```

```
layout (location = 0) in <type> <in variable name>  
in <type> <in variable name>
```

```
out <type> <out variable name>
```

```
uniform <type> <uniform name>
```

```
int main()  
{  
    //do some stuff here  
    <out variable name> = stuff;  
}
```

Useful definitions

- **Vertex:** a collection of data per 3D coordinate
- **Vertex Data:** a collection of vertices
- **Vertex attributes:** vertex position, vertex colour, etc
- **Uniforms:** comes from CPU directly to the shaders, they are global and seen by all the shaders
- **layout (location = 0)** – specifies the vertex attribute location. You can use `glGetAttribLocation` to get the attribute location if you omit to set the layout
- **Vertex shader** (input a **single vertex**). The main purpose of the vertex shader is to transform 3D coordinates and allows us to do some basic processing on the vertex attributes.
- **Fragment shader:** calculates the final colour of a pixel and this is usually the stage where all the advanced OpenGL effects occur. Requires a **vec4 colour output variable**, since the fragment shader needs to generate a final colour for the screen.

Useful links

- OpenGL specifications document (PDF) (Kronos group website)-
<https://www.khronos.org/registry/OpenGL/specs/gl/glspec33.core.pdf>
- Unity DOTS vs OpenGL (YouTube video):
https://www.youtube.com/watch?v=tInal3pU19Y&ab_channel=NickCaston
- To read: Chapter 1 - Introduction to 3D Graphics & Chapter 2 - Getting Started (OpenGL Superbible – see link on the DLE)
- Shaders article focusing on the parallel processing for GPU's:
<https://antongerdelan.net/opengl/shaders.html>
- Shaders (further reading – article): <https://learnopengl.com/Getting-started/Hello-Triangle>
- To read: Shaders - <https://learnopengl.com/Getting-started/Shaders>
- To read: Getting started with GLSL and Working with GLSL in OpenGL 4 Shading Language Cookbook