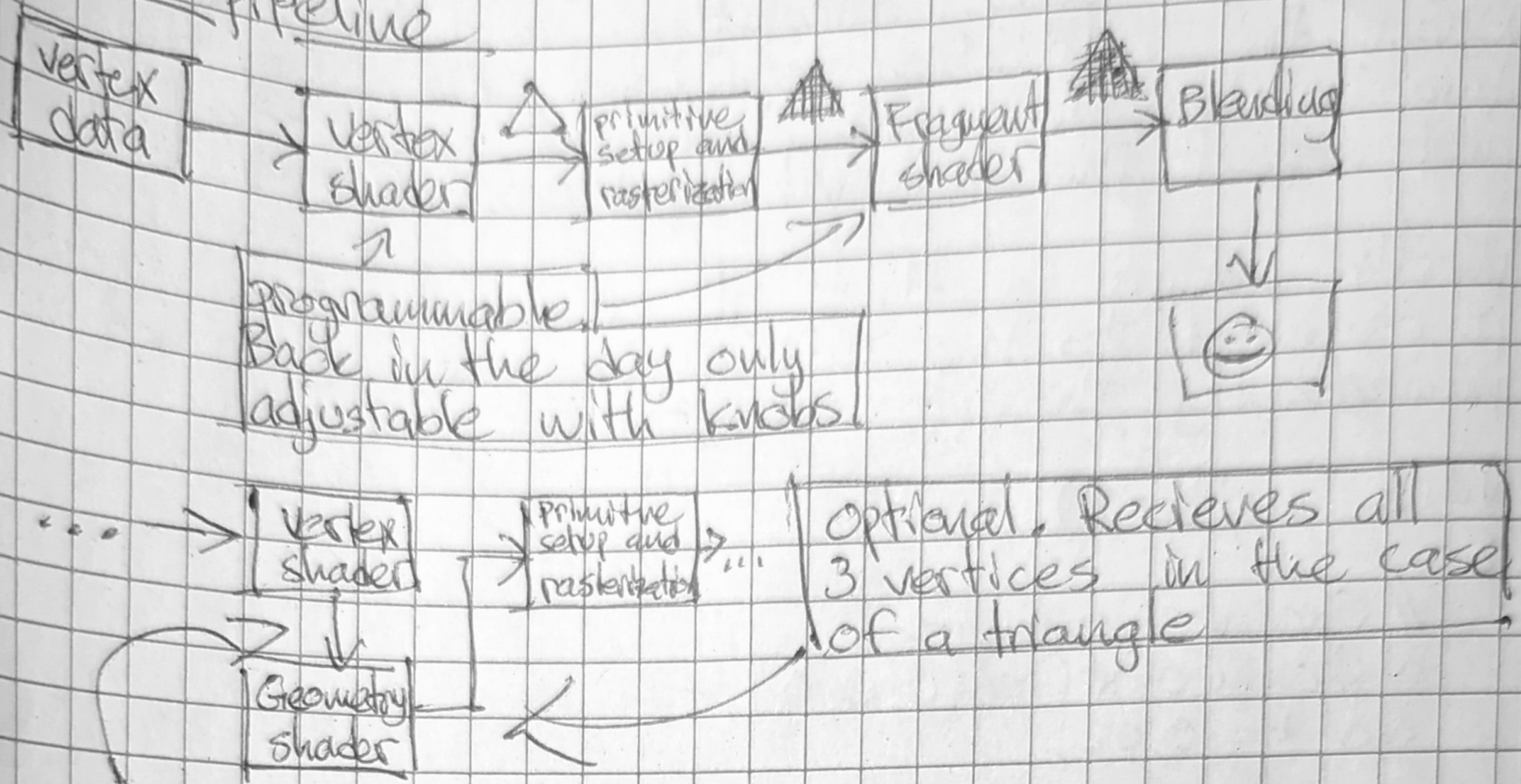
0- Introduction to modern OpenGL

GPU pipeline



can also have a tessellation shader after the vertex shader, either with or without the geometry shader in front.

Today we can replace the vertex, tessellation and geometry shaders with a mesh shader, which does everything.

Fragment shader:

First render target, can render to multiple at once

```
#version 330 core

layout(location=0) out vec4 color;

void main()
{
  color = vec4(1,0,0,1);
}
```

Vertex shader:

can recieve multiple inputs/attributes

```
#version 330 core;

layout(location=0) in vec3 pos;
```

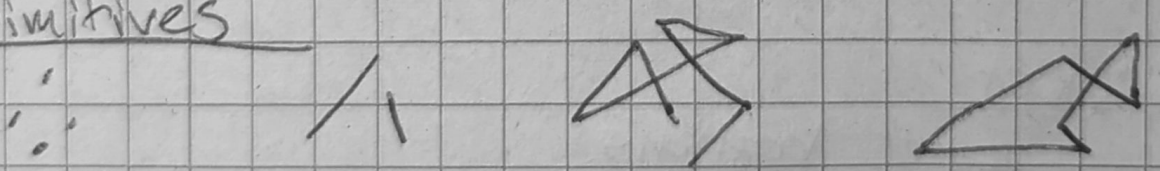
```
uniform mat4 mvp;

void main()
{
    gl_Position = mvp * vec4(pos, 1);
}
```

compile shaders:

we want to compile shaders at runtime so they can be optimized for the hardware

```
GLuint program = glCreateProgram();

// Compile vertex shader
char *vsSource = ReadFromFile("shader.vert");
GLuint vs = glCreateShader();
glShaderSource(vs, 1, vsSource, nullptr);
glCompileShader(vs);
glAttachShader(program, vs);
delete [] vsSource;

// same for fragment shader
...

glLinkProgram(program);
```

Primitives

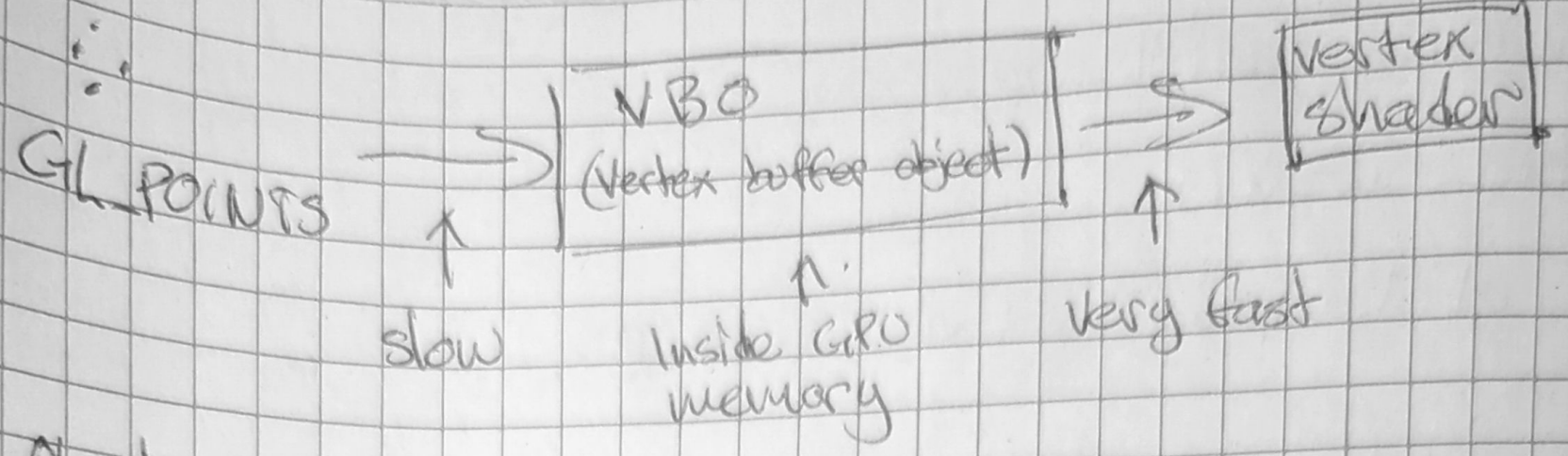


GL_POINTS GL_LINES GL_LINE_STRIP GL_LINE_LOOP

GL_TRIANGLES GL_TRIANGLE_STRIP GL_TRIANGLE_FAN

Vertex buffer object

In the old days we sent a single vertex at a time to the GPU with OpenGL call. This is really slow so now we use buffers to send many.

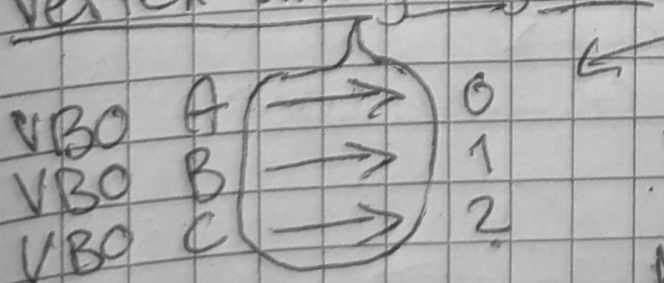



```
float positions[] = {
  -0.8, 0.4f, 0.0f,
  ...
};

GLuint buffer;
glGenBuffers(1, &buffer);
glBindBuffer(GL_ARRAY_BUFFER, buffer);
glBufferData(GL_ARRAY_BUFFER, sizeof(positions), positions,
                                                GL_STATIC_DRAW);

GLuint pos = glGetAttribLocation(program, "pos");
glEnableVertexAttribArray(pos);
glVertexAttribPointer(pos, 3, GL_FLOAT, GL_FALSE, 0,
                                        (GLvoid*)0);
```

Rendering

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glUseProgram(program);
glDrawArrays(GL_POINTS, 0, num_vertices);
glutSwapBuffers();
```

Vertex array object



VAOs stores the connections between VBOs and location in memory. Must have at least one.

```
GLuint vao;
glGenVertexArrays(1, &vao);
glBindVertexArray(vao);
```

} Need to be set before enabling vertex attribute array

Overview

- Init
  - Create VAOs
  - Create VBOs
  - Compile shaders
  - For each VAO
    - Assign VBOs to vertex attributes
- Rendering
  - For each VAO
    - call glDrawArrays