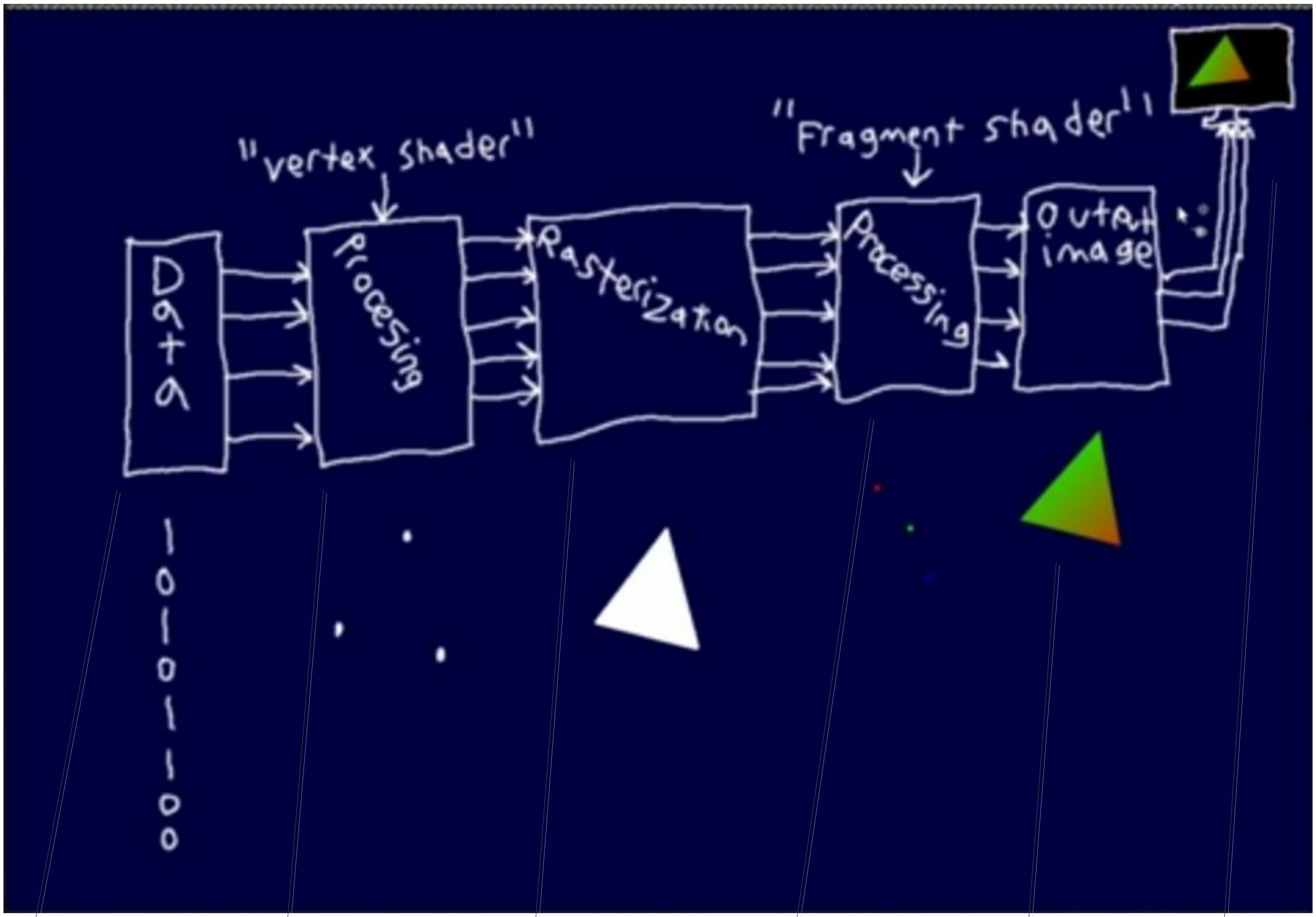
Graphics Pipeline: OpenGL

Processing happens in parallel… many parallel processes.
OpenGL programming involves telling the Graphical Processing Unit (GPU) how to interpret, arrange, and color incoming data. **Interpret Data** = **Mesh Creation**. Pixel **Placement** = **Vertex Shader**. Pixel **Colors** = **Fragment shader**.



Shaders

Step 1: Binary Data (could be any binary data)…

Writing the C++ openGL application involves telling the GPU How it must interpret data, eg: how big one unit of data is, etc…

This is called **mesh creation**

Step 2: Vertex Shader: Process data to try to make sense of it. It's processed in parallel to try to convert into points in 3d space.

Writing the C++ openGL application also involves telling the GPU how the final pixels should be arranged. This is done by a **vertex shader**.

Step 3: Rasterization: Connect / fill-in points to make a bunch of triangles

Step 4: Fragment Shader: Processing each pixel of rasterized triangles to generate color for each of these pixels.

Writing the C++ openGL application also involves telling the GPU how to treat the pixel, with color, in a **fragment shader**

Step 5: Output image is assembled in memory until the image is complete… then…

Step 6: The image is finally sent to the display buffer and can be seen

openGL3 is the first tutorial to cover this process