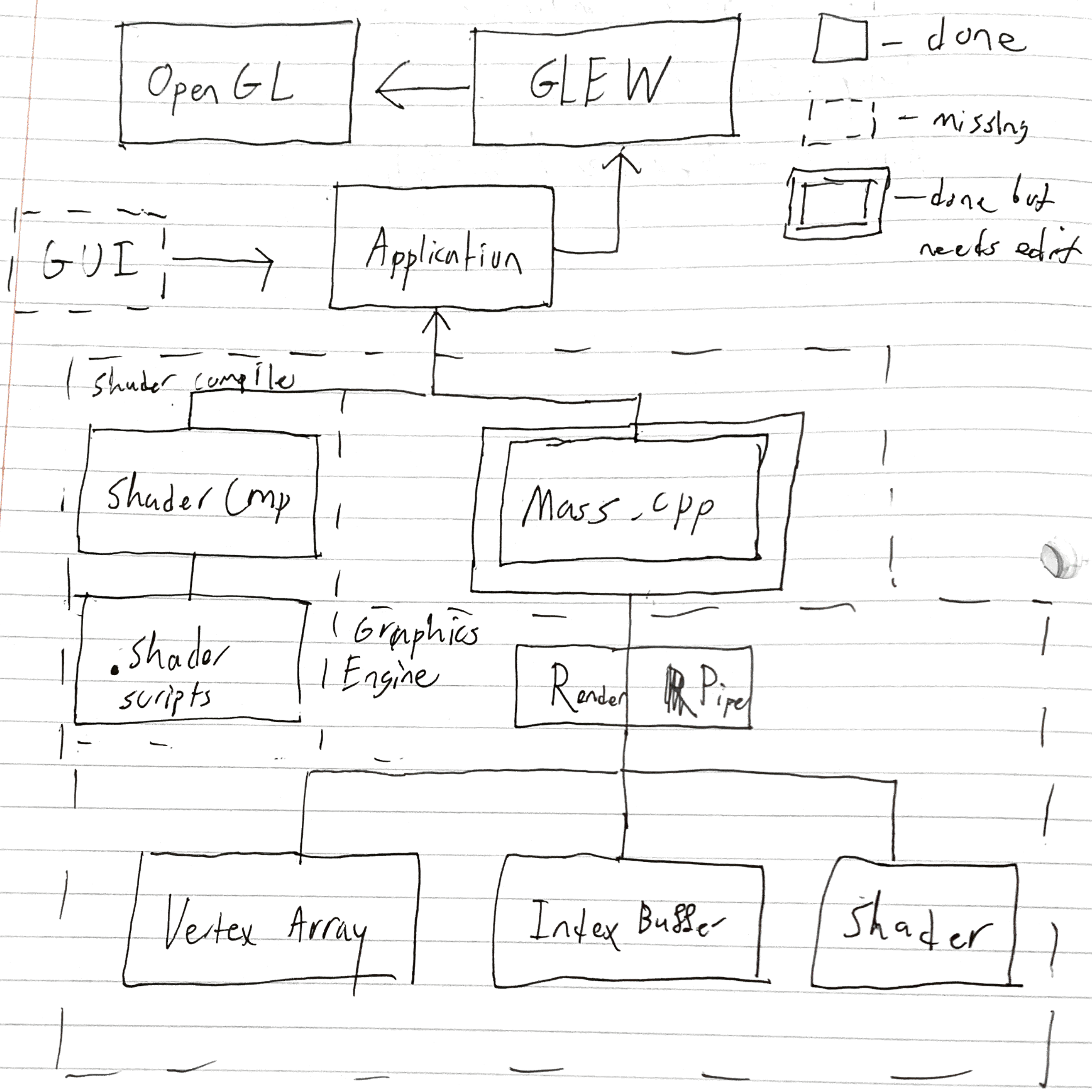
N-Body simulation



Vertex ~~Array~~ Buffer (How do we store Position?)

What is it?

- Store of Verticies that will be stored in the GPU's VRAM.

"Verticies" does not mean points ~~rathe~~ can be any data that will be used to render, ~~ie~~, eg. position, color, opacity(?) etc.

How does the engine handle it?

- we store VBO's (vertex buffer objects) as objects with 3 important peices of data

void * - Data (void* can be filled by anything)

unsigned int - size (# of Bytes used in memory.)

~~render_id~~

unsigned int - render_id (renderer the VBO must bind to for the buffer to render)

Example VBO Data

$$\text{float [] Points =} \quad \{ \underbrace{\begin{matrix}4, & 4, & 0, \\ -4, & 4, & 0, \\ -4, & -4, & 0, \\ 4, & -4, & 0,\end{matrix}}_{\text{vertex pos}} \quad \underbrace{\begin{matrix}1, & 0, & 0, \\ 0, & 1, & 0, \\ 0, & 0, & 1, \\ 1, & 1, & 0\end{matrix}}_{\text{vertex color}} \}$$

✗ All data ~~the~~ about an object is stored in the VBO

Vertex Buffer Layout

Problem: Each object has a vertex buffer of data but we don't know where each peice of data is allocated or what it's used for

How the competer sees ~~a~~ a VBO

4, 4, 0, 1, 0, 0, -4, 4, 0, 0, 1, 0, -4, -4, 0, 0, 0, 1

How we want it to pocess a VBO

$$\begin{aligned}&\{[4,4,0],\ [1,0,0]\},\\ &\{\underbrace{[-4,4,0],}_{\text{position}}\ \underbrace{[0,1,0]}_{\text{color}}\}\end{aligned}$$

How? VBL! Vertex Buffer Layout

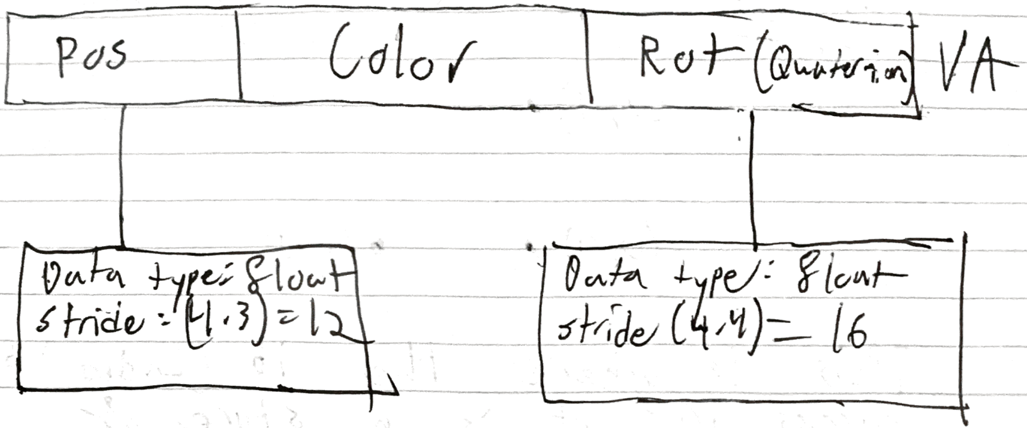
What is a VBL?

A VBL is a List ~~of~~ data perces and how they should be read from the buffer.

The peices of data in the above example would be position and color of each vertex. Each can be thought of as a series of 3 ~~number~~ floats

The VBL processes this as ~~m~~ each vertex will ~~and~~ take 24 bytes

A VBL adds structure to our data and can be thought of as the meta data for a ~~W~~ VA.



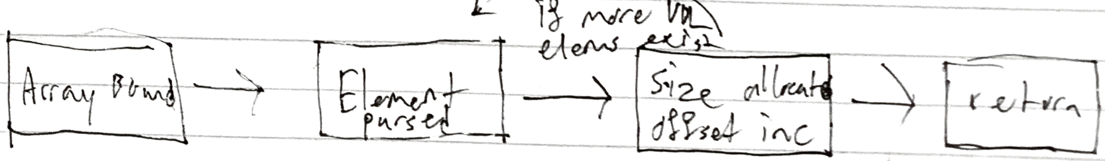
Total stride = 16+12 = 26

Vertex Array (How do we synthesize?)

Currently we have our data and structure but no way to combine the VA does this task.

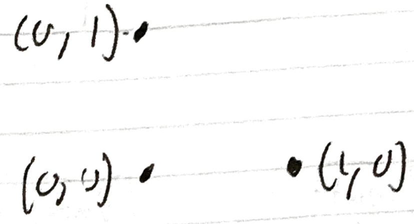
The VA acts as glue btwn the VB, VBL and opengl.

The VA binds our VB to our GPU and gives the renderer the VBL to process everything based on the strike.



Index Buffers (We have data; how is it used!)

say the Renderer is passed the following data, (1, 0) (0,0) (0,1). These give us points but fail to tell us how they connect



This is where the ib (Index Buffer) comes in; it is a store of connections between the <u>indicies</u> in the array of points;