2D Rendering

The rendering system document describes the rendering system. This document will discuss how specifically 2D rendering works and how to create new vertex attribute formats.

The 2D renderer contains a pointer to an overriding shader which, if valid, will override the material shader provided upon submission of the 2D geometry. The 2D renderer contains a map of shader program pointers mapped to a list of batch entries. A **batch entr**y is a struct containing the list of subtextures, a list of texture units, the tint, and the model matrix.

To support multiple vertex attribute formats, different vertex arrays and buffers need to be used. The **shader** being used to draw the submission contains the **UBO**s required by that shader program but also the **vertex array** whose vertex attributes satisfy the format declared in the shader program.

When the 2D geometry is submitted, a new batch entry is simply added to the back of the shader entry’s queue. When a flush call is made, the renderer iterates over all shader entries, and sets the necessary vertex data by editing the VBO once all batch entries for a shader have been accessed and binds the textures.

**Z ordering** of the 2D geometry is based on the z value of the local position of the 2D components (Text & MeshRender2D). Transform2D does not have this z value because it is not needed and with an orthographic projection, there is no depth with varying z values. Depth testing is enabled so the z-ordering is still determined by the z value of the local position. The value, however, **must be between 0 and 1.**

**New Vertex Format**

1. Create the new vertex in vertex.h
2. Create a new vertex array with the vertex buffer that supports the new vertex format, the index buffer will **NOT change**
3. Create the shader program which supports the vertex attribute format and attach the vertex array created in step 2
4. Update the vertex list loader function in renderer2D to create that vector of vertices and editing the VBO of the shader program passed
5. Update the resource loader to return the size of the newly created vertex
6. Update the 2D renderer to call the appropriate vertex list loader function when the current shader entry matches with vertex array of the shader program. Function name: **generateVertexList**.
7. Now we can attach the shader to a material and use the material to render with the vertex format.