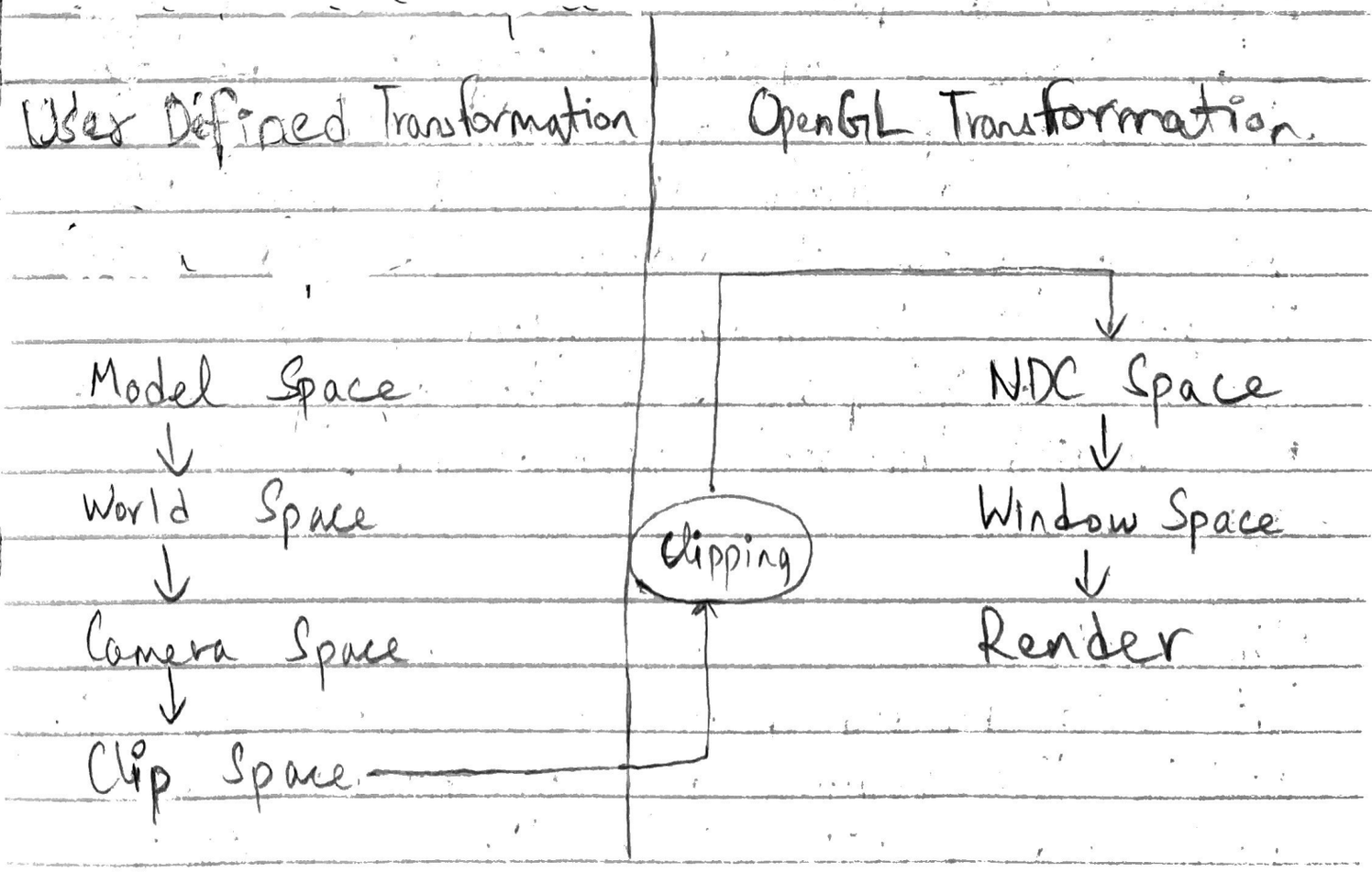
VERTEX TRANSFORMATION IN RASTERIZATION PIPELINE

Rasterization is one of the way of rendering 2D image from 3D world described in terms of 3D triangle primitives. Rasterization pipeline is the series of transformations that vertices of these 3D triangles go through before being finally rendered as a 2D pixel on the screen. Today, specialized hardwares like GPUs exist that perform these per vertex computation fast and thus we call them hardware rasterizers. OpenGL is an api for accessing hardware rasterizers. Following is an overview of vertex transformations that happen in a rasterization pipeline:



Local / Model Space

When you model your geometry (or mesh) in isolation in 3D modeling software, say a cube, that cube may have origin at (0, 0, 0) and be of certain scale, etc. This is local to the model of that cube. Vertices start out in this space.

World Space

When you put alongside one or more instances of one or more models, in a common space, that is a world space for those instances.

Camera / View Space

You have a camera occupying position in world space, looking at certain direction. You want to observe the world from the point of view of camera. This camera is brought to the origin facing -Z axis, parallel to XY plane and the world around is rotated / translated accordingly. This is camera space.

Clip Space

This is a bounded region i.e. a volume. Everything within this volume will be rendered as a final image. Theoretically this a 2x2x2

volume in OpenGL, so if either X, Y or Z coordinate isn't within [-1,1] range, then that vertex would fall outside the clipspace region and thus would not be rendered. Practically, however, clipspace coordinates of a vertex are represented a little differently: Your vertex shader is supposed to output clipspace coordinate for the vertex being processed. Clipspace coordinates have extra W coordinate, beside the usual X, Y and Z, and the vertex shader is not supposed to divide by W just yet, and that is something that happens further down the pipeline, before the invocation of fragment shaders and programmers have no control over it.

If a triangle is partially outside clipspace, then the triangle is clipped and the process is called clipping. This breaks the triangle apart into a number of smaller triangles such that the smaller triangles are entirely within clip space.

Window / Screen Space

NDC coordinates are mapped to window coordinates. Window coordinates have bottom-left position as (0,0) origin. X and Y are bounded by the viewport while the Z ranges from 0 to 1. X and Y are still floating point numbers so their

precision is not lost. Window coordinates are passed as input to the fragment shader.

Render

Vertices of triangles are fleshed out into something called "fragment"s that exist for every pixel within area of triangle. Fragments have color and depth values. Finally the fragment is written to a destination image.