scribe notes overview

November 2019

1 Overview

Talk about how a rasterization based graphics engine take points (of data) and turns them into on screen pixels via a series of transformations.

Give a (brief) overview of the matrices used in the pipeline as well as their corresponding stages.

2 Algorithm Details

Skim model/camera(space) transformations. Camera is at v3f(0,0,0) looking towards -z with aspect ratio = 1 by 1. FOV will be used later on.

Talk about data preparation (matrix generation based on object model, camera view, perspective matrix.

2.1 vertex shader

Discuss data processing. Where it comes from, what processing it undergoes, where it is sent. ModelViewPerspective matrix.

2.2 rasterization

Discuss data interpolation. Specifically, perspective correct interpolation.

Q: Why does he emphasize texture interpolation if the issue exists with all interpolation?

2.3 fragment shader

Talk about what it does, how it differs from the vertex shader. Throw in some examples of [insert alg here] in vs as opposed to fs.

3 Implementation details

Talk about the heavy use of buffers and data that can be passed (position, normal, colour, TEXTURE (more on this here))
Elaborate on hardware acceleration (CPU makes the matrices but computation is done by (GP)GPU).

3.1 visibility

Discuss visibility, z buffering specifically. Add some historical visibility solving solutions (DOOM's binary space partitioning, no buffer clearing).
Perhaps talk about perspective correct interpolation here, TBD.