Matrisen

2025-02-25

Notes

set up mesh shader that takes in parameters for curves and lines save the parameters after model view projection and lighting calculations

make a global resource bank with buffers and images, dont bundle them together and put structs everywhere and member variables everywhere

Use uniforms for global scene data and depending on material for texture and samplers use push constants in

Algorithm

pass primitve control points to the GPU eg. point along a curve fill, stroke The CPU is responsible for calculating the points from more high level geometry such as circle arrow etc.

the GPU will generate meshes (using mesh shader) to tesselate the geometry up to a certain resolution to small triangles will be inneficent, too large triangles will be inneficent for the rasterizer

the next step for the GPU is for the fragment shader to fill in pixels for its triangle, this can be done in one of three ways: fill all, fill inside a curve boundry fill only the curve boundry with a certain thicknes I think every possible geometry can be reduced to these three if tesselation is done right.

FRAGMENTSHADER

if pixel > boundry **then**
  discard
else
  fill in pixel