# Literature review

## Shading Rig: Dynamic Art-directable Stylised Shading for 3D Characters

Used function to define shapes. Cannot do rectangular shapes, for mechanical objects & characters. Less control than be Bezier curves.(has stats on performance). Bezier curves were not used due to real time performance goal.

## Fast, memory efficient and resolution independent rendering of cubic Bezier curves using tessellation shaders

Beziers curves have been rendered on the CPU. Curves can be less than a pixel which is “computationally expensive”.

Their method creates an area to be rendered where the curve can be. This area is made bigger than the bounds of the curve to avoid elements not being rendered as they go off the bounds. Method is based on the method in 3. Breaks down cubic to the simpler quadradic.

## Resolution Independent Curve Rendering using Programmable Graphics Hardware

For coordinates u and v, the function f(u,v) = u^2 – v can determine whether the coordinates are inside or outside a quadratic Bezier curve by checking if f(u,v) < 0. Their method requires finding the inflection points on the curve. The 5 curve types require different functions to determine these. The curve type can be determined. Once the inflection points are found, a general algorithm can render any cubic Bezier curve. Some B-Splines have both on curve and off curve points. On curve points are included in Bezier curves and can be rendered but off curve edges are divided into 2 on curve points.

Some pre-processing must be done on the CPU so dynamic points would impose a performance penalty. Current implementation only has solid colour fill. Gradient and texture fill are possible targets for expanding the technique.

NB: this paper’s technique is patented.

## Scalable Scalable Vector Graphics: Automatic Translation of Interactive SVGs to a Multithread VDOM for Fast Rendering

This solution uses multithreaded CPU.

## Random-Access Rendering of General Vector Graphics

## The A -buffer, an antialiased hidden surface method

Contains history of Z-buffer.

## GLSL Essentials

Pre 2002, render pipelines were fixed. After word, GPUs allowed for their stages to be programmable. The instructions are given in a vendor specific pseudo assembler language. Typically, these programmable stages, referred to as shaders, are written in a higher-level language such as HLSL, Cg or GLSL.

Initially, only vertex shaders and fragment shaders were available. Vertex shaders applied transformations and other application specific operations on the vertices given to the GPU. These run once per vertex. They can access other data provided to the GPU such as texture data. Fragment shader can access the same data and can be programmed but it is run per fragment created during rasterization.

## Unity profiler - https://docs.unity3d.com/Manual/ProfilerGPU.html

## Unreal profiler - <https://docs.unrealengine.com/4.27/en-US/TestingAndOptimization/PerformanceAndProfiling/GPU/>

## Nvidia profiler - https://developer.nvidia.com/nvidia-visual-profiler