TECHNISCHE UNIVERSITÄT DRESDEN

FACULTY OF COMPUTER SCIENCE
INSTITUTE OF SOFTWARE AND MULTIMEDIA TECHNOLOGY
CHAIR OF COMPUTER GRAPHICS AND VISUALIZATION
PROF. DR. STEFAN GUMHOLD

Minor Thesis

Spline-Tube Rendering

Mirko Salm
(Mat.-No.: 3753374)

Tutor: Dr. Sebastian Grottel

Dresden, 09.03.2015

Aufgabenstellung

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die von mir am heutigen Tag dem Prüfungsausschuss der Fakultät Informatik eingereichte Arbeit zum Thema:

Spline-Tube Rendering

vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Dresden, den 09.03.2015

Mirko Salm

Contents

Nomenclature

SDF	signed distance field

1 Introduction

- scientific vis works with things
- to render things one can use tubes
- cubic tubes are nice because smooth
- variable radius usefull to vis things of things

1.1 Thesis Structure

2 Related topics

2.1 Splines and generalized cylinders

- what are they good for? Why nice?
- Cubic Hermite spline
- Catmull Rom spline
- Bézier curves
- construction scheme using linear interpolations
- transform to points+tangents form (Hermite form)
- transform to polynomial in power form
- quadratic splines
- limitations, only 3 dofs, possible forms
- approx cubic by quadratic
- estimate tangents by central/forward/backward differences

2.2 Tessellation and rasterization

- given a mathematical description of a surface
- approximate surface with polygone/triangle mesh
- project triangles to screen and rasterize

-> convert mathematical descr. of triangle to set of fragments

- usually done using the GPU
- GPU itself can create new triangles using the tessellation unit of the pipeline
- different approaches for parametric and implicit surface descriptions
- parametric: derive vertex positions directly from paramters
- implicit: not possible to directly infer vertex positions from formula, apply Marching Cubes and Co
- dynamic tessellation to ensure view independent approx quality

2.3 Ray-based rendering

- ray-scene intersection routine

- only coherent primary rays: ray-casting

-> also simulated by rasterization

- also non-primary, non-coherent rays: ray-tracing, path-tracing

-> can not be realized using rasterization

2.4 Rootfinding

- bisectioning

- newton

- + polynomial division

- WeiDuKe

- interval/affine-arithmetic

3 Previous Work

3.1 Rendering Tubes from Discrete Curves Using Hardware Tessellation

- sequence of points define curve -> interpolated with Catmull-Rom splines
- points are sent to GPU, geometry created in hull-shader/tessellator/domain-shader
- approx the curve with linear segments, adaptively adds more segments in regions with large derivatives
- LOD reduces only cross-section vertex count down to three (triangular cross-section)
- radius is scaled up over distance to reduce aliasing
- apparently no MSAA is used in addition but AA is supposedly eliminated (?)
- also no opacity reduction applied when reaching sub-pixel diameters
- benchmark compares the tessellator-approach to a naive one where the geo is generated on CPU and send to GPU
- strongly reduced banwidth requirements
- AA approach (scaling radius) is compared to no AA
- nearly no fps drop

3.2 Rendering Generalized Cylinders using the A-Buffer

- dynamic adaptive screne-space tessellation of curves with varying radii
- using an A-Buffer for AA purposes (also adaptively supersamples its Phong lighting model)
- software-based implementation
- positions und radii are interpolated using cubic splines (Catmull-Rom)
- other properties like color, opacity, reflectance are linearly interpolated
- coarse tessellation followed by length- and breadthwise refinement using different constraints
- position constraint I: based on the angle between the screenspace tangent vectors of start/end-point of sub-segment

- position constraint II: maintains a desired degree of linearity in the z-component
- necessary to ensure that a curved segment is adequately subdivided even when viewed from an angle that makes its screen pro jection close to linear (important for quality of normales and texcoords)
- radius contraint: ensure smooth variation in the radius of a segment
- inflection point constraint: ensures that inflection point lie between sub-segment (on their vertices)
- 3 quality levels for breadthwise refinement, higher levels better approximate tube
- chosen depending on screenspace size and quality requirements
- backface culling for each segment
- normal calculation must consider changing radius
- high screenspace curvature can lead to ill-formed polygons
- handled by splitting the offending polygon into an equivalent pair of triangles
- variety of effects like length- and breadthwise opacity variation and a global illumination model that approx AO based

on the assumption that the paintstroke (tube) is embedded in a layer of a homogeneous medium
- benchmarks are pretty deprecated (thesis from 1997)
- overall approach appears annoyingly fiddly

3.3 Fast Ray Tracing of Arbitrary Implicit Surfaces with Interval and Affine Arithmetic

- hmmmmm

3.4 sphere/distance-bound ray-marching

- pretty slow, since distance is computed from roots of cubic polynomial at each step
- but works with smooth blending
- also used by GPU-Based Hyperstreamlines for Diffusion Tensor Imaging

4 Tessellation based approach

- equidistant static tessellation on CPU
- expands circles of vertices at each sample point in local tangent frame defined by tangents
- dynamic refinement on GPU
- refine silhouette based on view angle
- refine each triangle based on screenspace sizes of edges
- very generic approach
- bulge artifacts in regions of high curvature and relatively large radius
- could be resolved by using a more involved base tessellation scheme
- tessellate chain of capsules

5 Ray-based approaches

5.1 Disk-based

- intersection of ray with planes/disks placed and oriented along the curve
- results in high-order polynomials
- root finding either slow oder unreliable, or both
- bulge aritfacts

5.2 Sphere-based

- intersection of ray with spheres placed along the curve
- results in t(l) = polyA - sqrt(polyB) (for front intersections)
- interested in global minumum (first/smalles t (ray parameter))
- possible to derive high-order polynomial from t(x)’
- many useless roots
- same perf problems as disk-based poly
- alternative approach:
- search roots of polyB -> enclose valid/interesting regions of t(x)’
- search roots of t(x)’ in this regions
- use bisectioning between extrema for robust and fast root finding
- necessary extrema are found as roots from derivatives

6 Evaluation and discussion

- tessellation:

+ fast

+ good quality, in many cases indistinguishable from ray-casting

+ easy implementation (even from scratch)

- only primary rays

- bulge artifacts

- ray-based (sphere)

- mediocre performance

+ high quality

+ straightforward implementation (difficult from scratch, though)

+ supports incoherent rays -> suitable for high quality renderings

+ no bulge artifacts

7 Conclusion

- for interactive visualisation: tessellation
- high quality stills: ray-based (sphere)
- combine things to get a better thing
- further investigate things

Bibliography