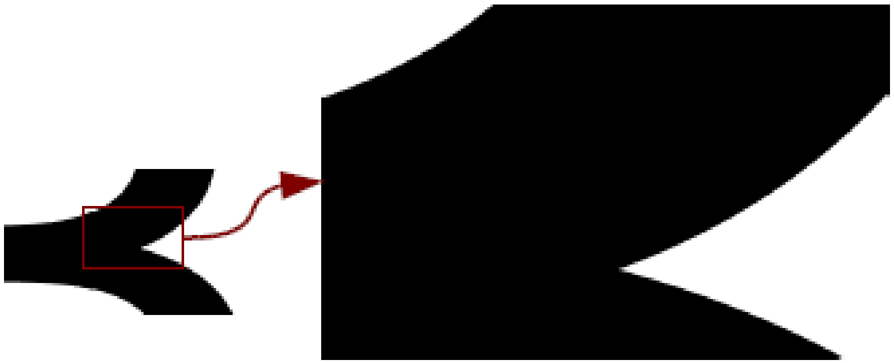
Resolution Independent NURBS Curves Rendering using Programmable Graphics Pipeline

Rami Santina
CCT International

Graphicon 2011
Moscow, Russia
September 2011

Motivation

- Visualize NURBS Curves:
  - Resolution independence
  - Fast Rendering, and pre-processing.
- Minimal memory storage.
- Resolution independent UI and CAD drawing in a 3D Scene.
- On embedded devices!



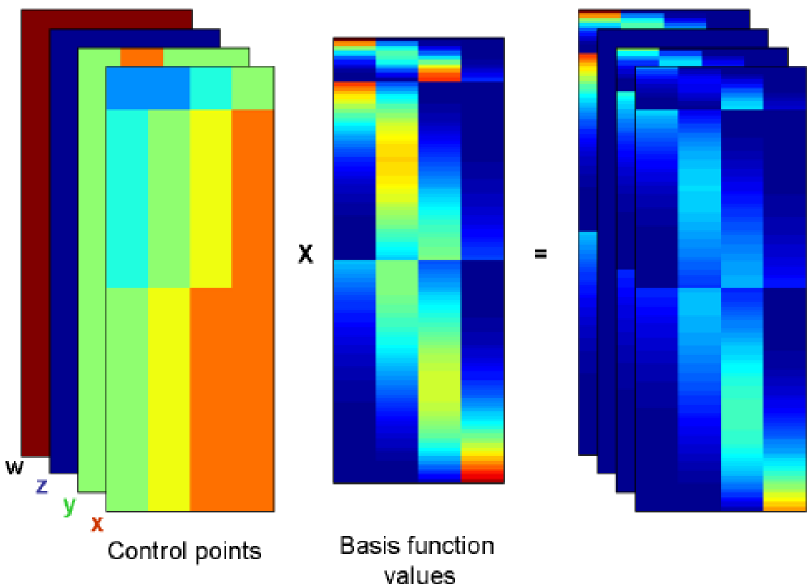
Background

- NURBS Curves:
  - Provides additional DoF namely, Weights.
  - Fewer Control Points to describe complex shapes
  - Widely used in CAD.
- NURBS Visualization:
  - Heavy pre-processing.
  - Common approach: Convert to Bezier data. (SVG...) - Post Design Visualization

$$C(x) = \frac{\sum_{i=0}^{n} N_{i,D}(x) w_i P_i}{\sum_{j=0}^{n} N_{j,D}(x) w_j}$$

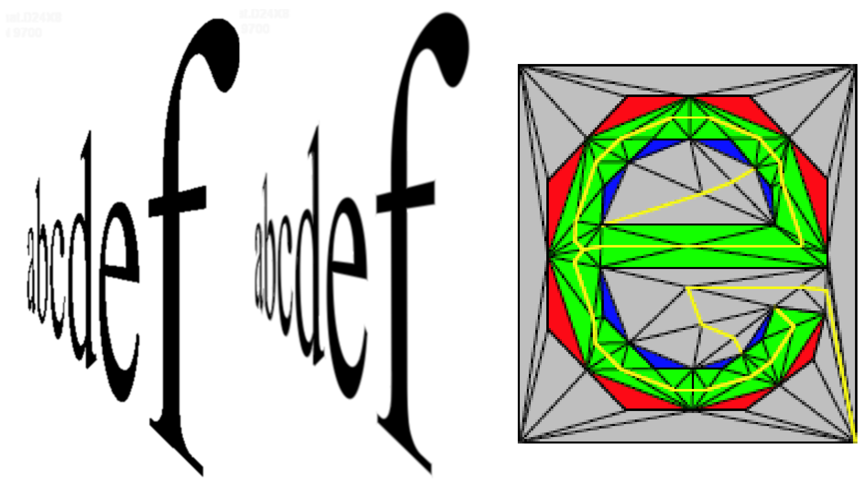
Related Work

GPU NURBS Rendering, using textures



Direct evaluation of nurbs curves and surfaces on the gpu, Krishnamurthy et al. 2007

Resolution Independent, Bezier Curves

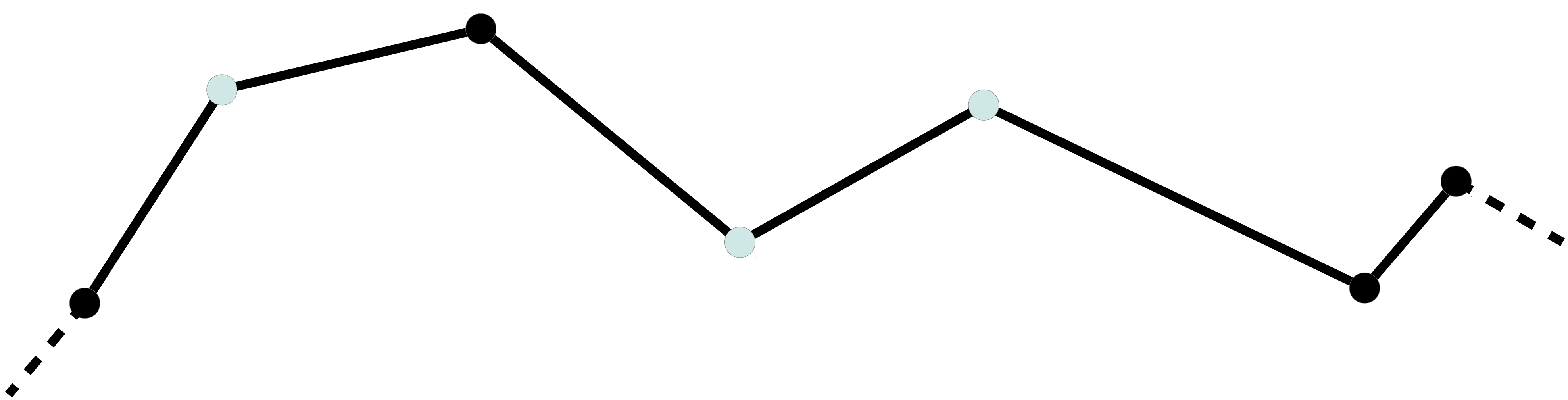


Resolution independent curve rendering using programmable graphics hardware, Loop Blinn 2005

Images from referenced authors/papers

Our Method: Input

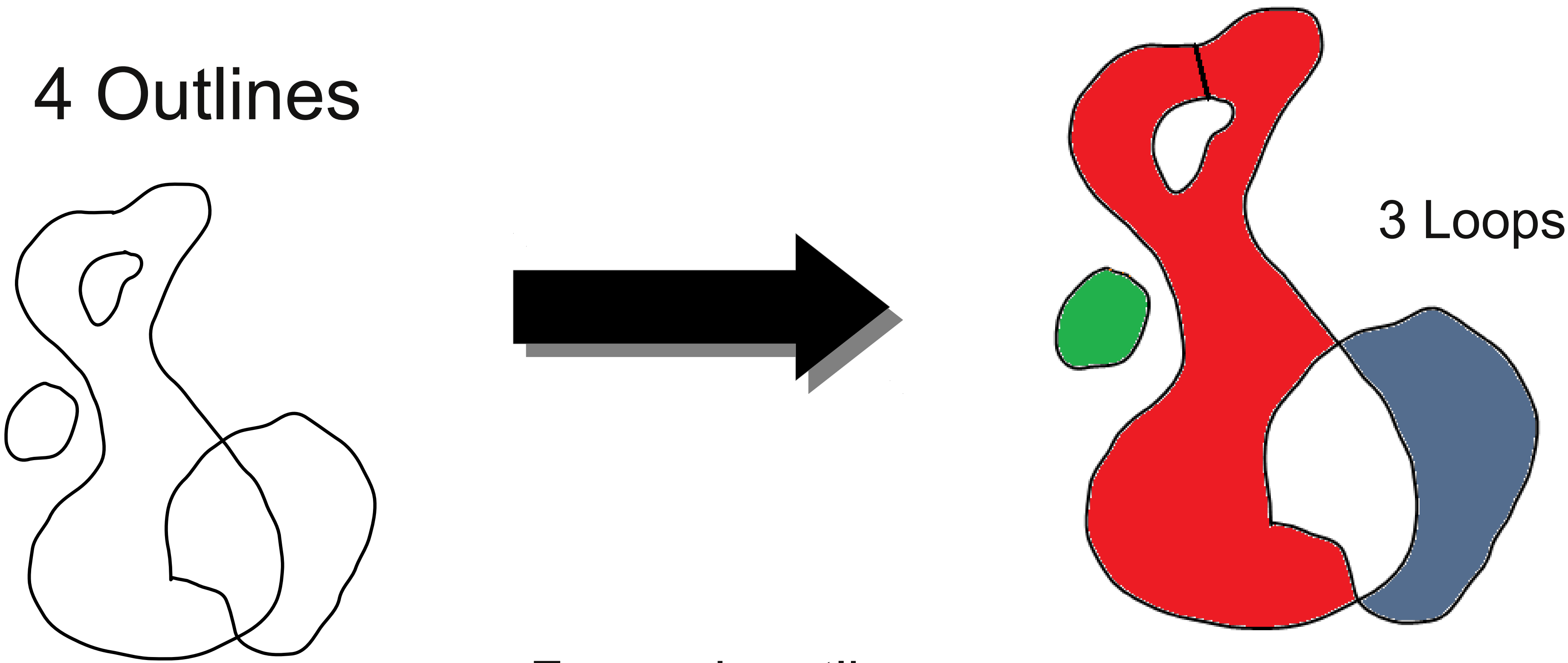
- Set of outlines (shape's boundaries).
  - Vertices are of two types: off-curve, on-curve.
  - Each vertex has x, y, z, w as attributes.



Our Method: Input

- Convert curved parts of the outlines to a set of triplets.
  - Each triplet has one off-curve → *curved* triangle
- The inner regions → *non-curved* triangles. (Triangulation)

Modified Delaunay Triangulation.



For each outline:
- Triangulation done Independently.
- No cleanup phase.
- No extra triangles.

Rendering – Quadratic Curve

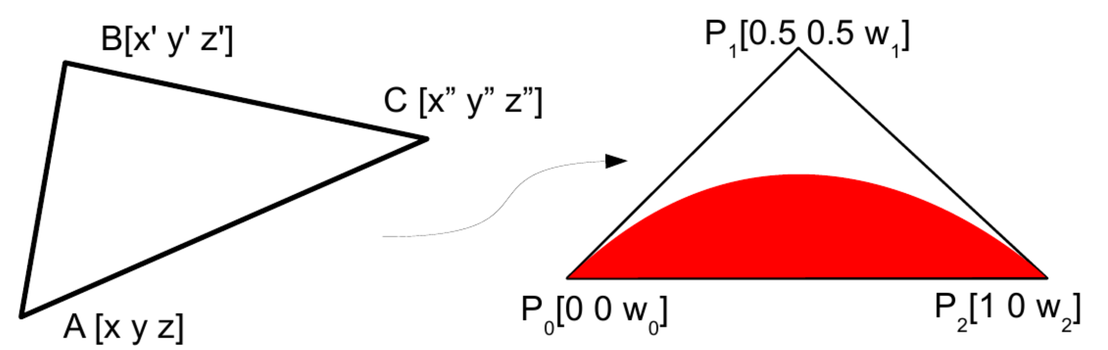
- Let the control points be:

$$p_0 = [0\ 0\ w_0],\ p_1 = [\tfrac{1}{2}\ \tfrac{1}{2}\ w_1] \text{ and } p_2 = [1\ 0\ w_2]$$

- Perform a Triple Knot insertion.

  K= [0 0 0 1 1 1]

- Assign *Pi* as texture Coordinates to each *curved* triangle



Rendering – Quadratic Curve

- Derive the implicit form of the curve.

$$f = v - \frac{w_1 u(1-u)}{(w_0 - 2w_1 + w_2)u^2 + 2(w_1 - w_0)u + w_0}$$

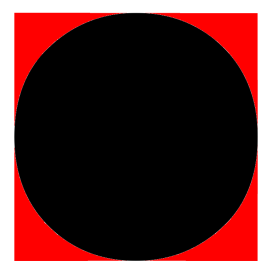
- Using the implicit function we can check if a fragment is **in** (*f* < *0*) or **out**.
- Setting P_1=[1/2 -1/2] for **out**, we can always render **in.**

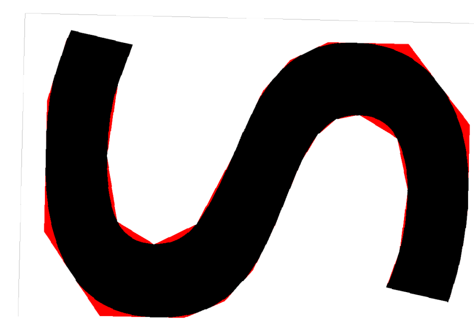
Rendering – Quadratic Curve

- Each curved triangle can be manipulated using W_1

Equiv to LoopBlinn2005

- We note that the Curve is **Aliased**.





Rendering Regions: Frag. Shader

- Compute $\nabla g(x, y)$ using the chain rule:

$$\nabla g = \begin{bmatrix} g_y^x - \frac{w_1((w_0 - w_2)u^2 - 2w_0 u + w_0)g_x^x}{(\alpha u^2 + 2\beta u + w_0)^2} \\ \\ g_y^y - \frac{w_1((w_0 - w_2)u^2 - 2w_0 u + w_0)g_x^y}{(\alpha u^2 + 2\beta u + w_0)^2} \end{bmatrix}$$

where

$$\alpha = w_0 - 2w_1 + w_2 \;,\; \beta = w_1 - w_0$$

Anti Aliasing

- Compute the signed distance:

$$e(u,v) = \frac{1}{2} - sign(v)\frac{f}{||\bigtriangledown g||}$$

- Classify:

$$class(u,v) = \begin{cases} in & e(u,v) \geq 1 \\ out & e(u,v) \leq 0 \\ boundary & \text{otherwise.} \end{cases}$$

Anti Aliasing

- For curved triangles – done.
- Non curved:
  - MSAA – General Case.
  - VBAA – Two pass rendering

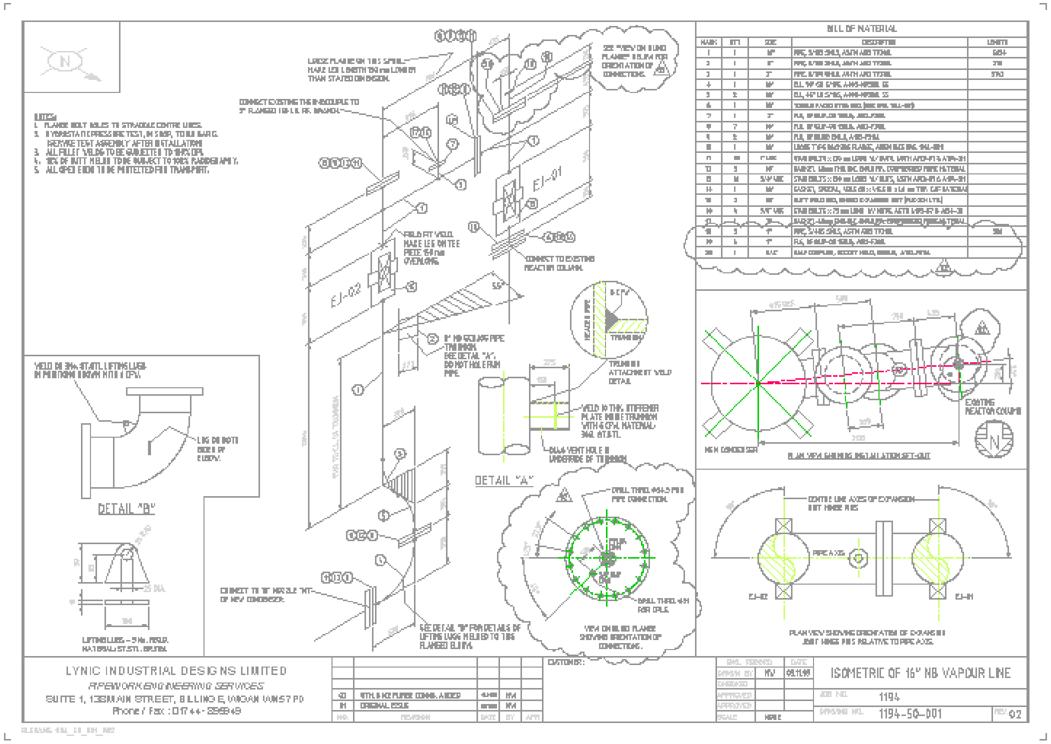
Implementation - API

- Part of Jogl Open Source Project!
- Graph API: RI shapes, Text and User Interface

http://jogamp.org

Application

- P&ID visualization: (Desktop & Mobile)



Visual Project Controls
http://c3d.com

Conclusion & Future Work

- Presented a method for rendering NURBS curve
  - Resolution Independent
  - Mobile ready! (OpenGL ES2 impl)
  - No heavy preprocessing and memory usage.
- Future work:
  - Resolution Independent User Interface and UI Design tool.
  - Resolution independent P&ID viewer

Thank you!