Ray Casting of Trimmed NURBS Surfaces on the GPU

Hans-Friedrich Pabst  Jan P. Springer  André Schollmeyer
Robert Lenhardt  Christian Lessig  Bernd Fröhlich

Bauhaus University Weimar · Faculty of Media · Virtual Reality Systems Group

IEEE Symposium on Interactive Ray Tracing 2006
Overview

► GPUCAST system
  ► Framework for single pass ray casting on the GPU
  ► Generic library: algorithms, data structures
  ► Type/value transform iterators on the GPU
  ► Shader metaprogramming
  ► Scene graph integration

► Publication
  ► Pabst, Springer, Schollmeyer, Lenhardt, Lessig, Froehlich: *Ray Casting of Trimmed NURBS Surfaces on the GPU*
Motivation

- Trimmed NURBS surfaces
  - CAD standard
- Ray casting
  - Direct rendering
  - Pixel-accurate
- GPU
  - Lots of gigaflops per value

Main Goal

Interactive rendering of trimmed NURBS surfaces using ray casting on commodity hardware.
Motivation

- Trimmed NURBS surfaces
  - CAD standard
- Ray casting
  - Direct rendering
  - Pixel-accurate
- GPU
  - Lots of gigaflops per value

Main Goal

Interactive rendering of trimmed NURBS surfaces using ray casting on commodity hardware.
NURBS

- NURBS surfaces provide local and explicit control
- Primitives included, e.g. curve, sphere, cone, cube
- Compact representation
- Continuity between curves and patches
- Trimming allows complex boundaries and topologies
Outline

- Integrate NURBS primitives into hardware graphics pipeline
- Ray-NURBS intersection and accurate trimming on the GPU
- Demonstration
- Results and conclusions
Surface Rendering: Algorithm Overview

- Preprocessing
  - Create bounding volume (convex hull)
  - Send vertices and parametric data

- For each surface
  - Transform convex hull
  - Rasterize convex hull
    - Compute ray-surface intersection
    - Trimming
    - Shading

- Result
Surface Rendering: Algorithm Overview

- **Preprocessing**
  - Create bounding volume (convex hull)
  - Send vertices and parametric data

- For each surface
  - Transform convex hull
  - Rasterize convex hull
  - Compute ray-surface intersection
  - Trimming
  - Shading
Surface Rendering: Algorithm Overview

- Preprocessing
  - Create bounding volume (convex hull)
  - Send vertices and parametric data

- For each surface
  - Transform convex hull
  - Rasterize convex hull
    - Compute ray-surface intersection
    - Trimming
    - Shading
Surface Rendering: Algorithm Overview

- **Preprocessing**
  - Create bounding volume (convex hull)
  - Send vertices and parametric data

- For each surface
  - Transform convex hull
  - Rasterize convex hull
  - Compute ray-surface intersection

- **Trimming**
  - Shading
Surface Rendering: Algorithm Overview

- **Preprocessing**
  - Create bounding volume (convex hull)
  - Send vertices and parametric data

- **For each surface**
  - Transform convex hull
  - Rasterize convex hull
  - Compute ray-surface intersection

- **Trimming**
- **Shading**
Numeric Intersection Computation

- Evaluation: \((uv) \rightarrow (x, y, z)\)
- Solving: \((x, y, z) \rightarrow (uv)\)

- Methods: general root finding vs. geometrical context
  - Subdivision
  - Numerical (iterative)
  - Algebraic
  - Hybrid

- Newton Iteration
  - Only parameters of one step necessary
  - Only function values and partial derivatives needed
  - Quadratic convergence
Initial Values for the Newton Iteration

**Problem:** An approximate solution is needed to get a solution

→ Information about geometrical context can be used

- Two complementary approaches: subdivision and \( uv \)-texturing
- Motivation: good initial values will result in fast convergence
Subdivision of the Convex Hull

- Quadratically increasing tightness
- Trade-off between ray casting and standard graphics pipeline
  - Minimizes number of fragments/rays
  - Number of vertices increased
- Adaptive subdivision
  - Minimizes number of generated vertices
- Union of all convex hulls
  - Approximation of the surface
uv-Texturing

- Idea: interpolated guess for each ray
- Associate an initial value (vertex attribute) with each vertex
  - Complement outer control points
  - Mapping parameter range between
- Subdivision-aware
- Subdivision increases quality
- Problem
  - Good heuristic for points of the control mesh
  - Invalid for some edges/faces of the convex hull
Trimming: Algorithm Overview

- Ray casting in parameter domain
- Similar to point-in-polygon test
- Bézier form provides exact representation of NURBS curves
- Accurate intersection computation using Bézier Clipping
Bézier Clipping

- Numerical root finding algorithm (subdivision)
- Makes use of the convex hull property

- Transformation into local equidistant coordinate system, invariant with respect to the intersection points
Compute convex hull intersections with $t$-axis
Split curve at $t_{\text{min}}$ and $t_{\text{max}}$ ("clipping")
Interval contraction driven by clipping and subdivision
Iterative Bézier Clipping

**Problem:** Subdivision implies recursive processing of sub-intervals

- Only two intervals at a time are needed
- Only one scalar value needed to represent remaining interval
Iterative Bézier Clipping (cont.)

▶ Favors re-computation over storing values
▶ Consists of a state machine inside a loop to simulate function calls
▶ Intersection test takes advantage of Bernstein-Bézier form

▶ Properties
  ▶ Iterative depth-first algorithm
  ▶ Enumerates roots in ascending order

"This is the first implementation of a subdivision-like single pass algorithm on current graphics hardware."
Direct Trimming

- Interactive manipulation of existing control points possible
- Complements the direct rendering of surfaces
- Can also be used for trimming triangulated patches
Limitations

- Hardware and tool chain
  - Registers, writeable memory
  - Compiler, graphics driver, debugging
- Algorithm
  - Artifacts (ray-surface intersection)
  - Trimming without acceleration data structure
- Large models
  - Usually one program per surface
  - Limited degree ($\approx 6 \times 6$):
    \[ M + 2N \leq 19 \text{ with } N \leq M \]
The Trimmed Utah Teapot

Iteration + Manipulation
## Results

<table>
<thead>
<tr>
<th>Figure</th>
<th>Triangles</th>
<th>Subdivision</th>
<th>FPS₄</th>
<th>FPS₈</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duck</td>
<td>3732</td>
<td>1 × 1</td>
<td>18</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>15648</td>
<td>2 × 2</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>63602</td>
<td>4 × 4</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>Teapot</td>
<td>3092</td>
<td>2 × 2</td>
<td>33</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>12698</td>
<td>4 × 4</td>
<td>36</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>51160</td>
<td>8 × 8</td>
<td>30</td>
<td>25</td>
</tr>
<tr>
<td>Teapot&lt;sup&gt;Orient&lt;/sup&gt;</td>
<td>3092</td>
<td>2 × 2</td>
<td>22</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>12698</td>
<td>4 × 4</td>
<td>24</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>51160</td>
<td>8 × 8</td>
<td>22</td>
<td>18</td>
</tr>
<tr>
<td>Teapot&lt;sup&gt;NV&lt;/sup&gt;</td>
<td>3092</td>
<td>2 × 2</td>
<td>18</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>12698</td>
<td>4 × 4</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>51160</td>
<td>8 × 8</td>
<td>17</td>
<td>15</td>
</tr>
</tbody>
</table>

Resolution 1280 × 1024, screen covering 80% of width, GPU GeForce 7900 GT
### Results

<table>
<thead>
<tr>
<th>Figure</th>
<th>Triangles</th>
<th>Subdivision</th>
<th>$\text{FPS}_4$</th>
<th>$\text{FPS}_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duck</td>
<td>3732</td>
<td>$1 \times 1$</td>
<td>18</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>15648</td>
<td>$2 \times 2$</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>63602</td>
<td>$4 \times 4$</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>Teapot</td>
<td>3092</td>
<td>$2 \times 2$</td>
<td>33</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>12698</td>
<td>$4 \times 4$</td>
<td>36</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>51160</td>
<td>$8 \times 8$</td>
<td>30</td>
<td>25</td>
</tr>
<tr>
<td>Teapot$^{\text{Orient}}$</td>
<td>3092</td>
<td>$2 \times 2$</td>
<td>22</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>12698</td>
<td>$4 \times 4$</td>
<td>24</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>51160</td>
<td>$8 \times 8$</td>
<td>22</td>
<td>18</td>
</tr>
<tr>
<td>Teapot$^{\text{NV}}$</td>
<td>3092</td>
<td>$2 \times 2$</td>
<td>18</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>12698</td>
<td>$4 \times 4$</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>51160</td>
<td>$8 \times 8$</td>
<td>17</td>
<td>15</td>
</tr>
</tbody>
</table>

Resolution $1280 \times 1024$, screen covering 80% of width, GPU Geforce 7900 GT
### Results

<table>
<thead>
<tr>
<th>Figure</th>
<th>Triangles</th>
<th>Subdivision</th>
<th>FPS$_4$</th>
<th>FPS$_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duck</td>
<td>3732</td>
<td>$1 \times 1$</td>
<td>18</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>15648</td>
<td>$2 \times 2$</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>63602</td>
<td>$4 \times 4$</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>Teapot</td>
<td>3092</td>
<td>$2 \times 2$</td>
<td>33</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>12698</td>
<td>$4 \times 4$</td>
<td>36</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>51160</td>
<td>$8 \times 8$</td>
<td>30</td>
<td>25</td>
</tr>
<tr>
<td>Teapot$^{\text{Orient}}$</td>
<td>3092</td>
<td>$2 \times 2$</td>
<td>22</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>12698</td>
<td>$4 \times 4$</td>
<td>24</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>51160</td>
<td>$8 \times 8$</td>
<td>22</td>
<td>18</td>
</tr>
<tr>
<td>Teapot$^{\text{NV}}$</td>
<td>3092</td>
<td>$2 \times 2$</td>
<td>18</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>12698</td>
<td>$4 \times 4$</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>51160</td>
<td>$8 \times 8$</td>
<td>17</td>
<td>15</td>
</tr>
</tbody>
</table>

Resolution 1280 × 1024, screen covering 80% of width, GPU GeForce 7900 GT
<table>
<thead>
<tr>
<th>Figure</th>
<th>Triangles</th>
<th>Subdivision</th>
<th>FPS(_4)</th>
<th>FPS(_8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duck</td>
<td>3732</td>
<td>1 × 1</td>
<td>18</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>15648</td>
<td>2 × 2</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>63602</td>
<td>4 × 4</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>Teapot</td>
<td>3092</td>
<td>2 × 2</td>
<td>33</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>12698</td>
<td>4 × 4</td>
<td>36</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>51160</td>
<td>8 × 8</td>
<td>30</td>
<td>25</td>
</tr>
<tr>
<td>Teapot(_{Orient})</td>
<td>3092</td>
<td>2 × 2</td>
<td>22</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>12698</td>
<td>4 × 4</td>
<td>24</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>51160</td>
<td>8 × 8</td>
<td>22</td>
<td>18</td>
</tr>
<tr>
<td>Teapot(_{NV})</td>
<td>3092</td>
<td>2 × 2</td>
<td>18</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>12698</td>
<td>4 × 4</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>51160</td>
<td>8 × 8</td>
<td>17</td>
<td>15</td>
</tr>
</tbody>
</table>

Resolution 1280 × 1024, screen covering 80% of width, GPU Geforce 7900 GT
Conclusions

- Direct rendering
  - Ideal solution for CAD
  - Low CPU overhead
  - Minimal storage

- Pixel-accurate
  - Silhouettes
  - Interpenetrations
  - Normals

- Future Work
  - Reliable ray-surface intersection test
  - Trimming acceleration data structure

"Higher order primitives will complement triangles as the primary rendering primitive."
Conclusions

- Direct rendering
  - Ideal solution for CAD
  - Low CPU overhead
  - Minimal storage
- Pixel-accurate
  - Silhouettes
  - Interpenetrations
  - Normals
- Future Work
  - Reliable ray-surface intersection test
  - Trimming acceleration data structure

"Higher order primitives will complement triangles as the primary rendering primitive."
The End.

Thank you for your attention.