Implementation

Tianyun Ni
Overview

• This course section discusses real-time rendering of efficient substitutes for subdivision surfaces.
  – The advent of DirectX 11
  – Recent theoretical results
• Implementation on current hardware
• Practical implementation issues
Current Authoring Pipeline

1. Control Cage, or SubD Mesh
2. Smooth Surface
3. Displaced Surface
4. Generate LODs
5. Polygon Mesh

GPU
Direct3D 11 Pipeline For Real-time Tessellation Rendering

Control Cage + Displacement Map → Optimally Tessellated Mesh

GPU
Direct3D 11 Tessellation Pipeline

• Save memory and bandwidth
  - Memory is the critical bottleneck to render highly detailed surfaces

<table>
<thead>
<tr>
<th></th>
<th>Level 8</th>
<th>Level 16</th>
<th>Level 32</th>
<th>Level 64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular Triangle Mesh</td>
<td>16MB</td>
<td>59MB</td>
<td>236MB</td>
<td>943MB</td>
</tr>
<tr>
<td>D3D11 compact</td>
<td>1.9MB</td>
<td>7.5MB</td>
<td>30MB</td>
<td>118MB</td>
</tr>
</tbody>
</table>
Direct3D 11 Tessellation Pipeline

Control Cage + Displacement Map

GPU

Optimally Tessellated Mesh
Tessellation Process
Direct3D 11 Graphics Rendering Pipeline
Direct3D 11 Tessellation

Vertex Shader
Vertex Shader

- Transforms control cage vertices
**Vertex Shader**

- **Applications**
  - Realistic animation: skinning, morph targets, etc

- **Physical Simulation**: hair simulation, particle system, soft body deformation
Direct3D 11 Tessellation

- New Primitive Type: **Patch**
- Vertex Shader
- Hull Shader
Hull Shader

- New patch primitive type
  - The only supported primitive when tessellation is enabled
  - Arbitrary vertex count (up to 32)
  - No implied topology
- Computes control points
- Computes tessellation factors
Adaptive Tessellation

- Curvature, view-dependent Level of Detail
Control Points Evaluation

• In all cases we can evaluate a control point as a weighted sum: $P_j = \sum (W_{ij} \times V_i)$

• Pre-compute the weights for each topological connection type and store this information in a texture

• The hull shader invokes multiple threads and use one thread to compute one control point.
Control Points Evaluation

```cpp
ACC_CONTROL_POINT SubDToParametricPatchHS(
    InputPatch<CONTROL_POINT_OUTPUT, M> p,
    uint tid : SV_OutputControlPointID,
    uint pid : SV_PrimitiveID )
{
    ACC_CONTROL_POINT output;

    int topo = getPatchConnectivityID(pid);
    int num = getVertexCount(pid);

    float3 output.pos = float3(0,0,0);
    for (int i=0; i< num; i++)
    {
        int idx = getVertexIDInPatch(pid,i);
        int index = fetchIndexInStencil(topo, idx, tid);
        output.pos += p[i] * gStencil.Load(int3((index, 0,0)));
    }
    return output;
}
```

- Compute one control point per thread
- Connectivity type ID
- The number of vertices in the patch primitive
Direct3D 11 Tessellation

Vertex Shader

New Primitive Type: **Patch**

Hull Shader

Tessellator

+
Tessellator

- Fixed function stage, but configurable
- Domains:
  - Triangle
  - Quad
  - Isolines
- Spacing:
  - Discrete
  - Continuous(fractional)
  - Pow2
Tessellator

Level 5

Level 5.4

Level 6.6
Direct3D 11 Tessellation

- **Vertex Shader**
- **New Primitive Type: Patch**
- **Hull Shader**
- **Domain Shader**
- **Tessellator**
Domain Shader

- Evaluate surface given parametric UV{W} coordinates
- Apply displacements
- One invocation per generated vertex
Tessellation Process
Patch Construction Schemes

• PN Triangles

by Alex Vlachos, Jörg Peters, Chas Boyd, and Jason Mitchell

PN Triangles on Direct3D 11 Pipeline

Hull Shader

HS input:

Tessellator

Tessellation factors

Domain Shader

DS Input from Tessellator:

• uvw coordinates for one vertex

DS Output:
Patch Construction Schemes

• ACC Patches

*by Charles Loop and Scott Schaefer*

ACC Patch on Direct3D 11 Pipeline

**HS input:**
- uv coordinates for one vertex

**Tessellation factors**

**Domain Shader**

**Hull Shader**

**Tessellator**

DS Input from Tessellator:
- uv coordinates for one vertex
Patch Construction Schemes

- Pm Patches

*by Ashish Myles, Tianyun Ni and Jörg Peters*

*“Fast Parallel Construction of Smooth Surfaces from Meshes with Tri/Quad/Pent Facets”, Symposium on Geometry Processing (SGP), Copenhagen, Denmark, January 2 - 4, 2008.*
Pm – Patch on Direct3D 11 Pipeline

**HS input:**

- Hull Shader input

**Tessellation factors**

**Domain Shader**

- UV, or uvw coordinates for one vertex

**Tessellator**

- One tessellated vertex

**DS Input from Tessellator:**

- UV, or uvw coordinates for one vertex

**DS Output:**
Instanced Tessellation on Current Hardware

- **Input Assembler**
- **Vertex Shader**
- **Hull Shader**
- **Tessellator**
- **Domain Shader**
- **Geometry Shader**
- **Setup/Raster**
- **Pixel Shader**
- **Output Merger**

---

- **Rendering Pass 1:** Animation
- **Rendering Pass 2:** Patch Construction
- **Rendering Pass 3:** Compute TF

---

**Rendering Pass 1:**
- Input Assembler
- Vertex Shader
- Domain Shader
- Tessellator
- Geometry Shader
- Setup/Raster
- Pixel Shader
- Output Merger

**Rendering Pass 2:**
- Input Assembler
- Vertex Shader
- Domain Shader
- Tessellator
- Geometry Shader

**Rendering Pass 3:**
- Input Assembler
- Vertex Shader
Instanced Tessellation on Current Hardware

<table>
<thead>
<tr>
<th>Patch</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

Patch index buffer

2 4 5

0 1

3 6
Adaptive Tessellation on Current Hardware

• **Generic Mesh Refinement On GPU**
  
  [Link](http://iparla.labri.fr/publications/2005/BS05/GenericMeshRefinementOnGPU.pdf)

• **Generic Adaptive Mesh Refinement**
  
  [Link](http://http.developer.nvidia.com/GPUGems3/gpugems3_ch05.html)

• **Semi-uniform Adaptive Patch Tessellation**
  
  [Link](http://sintef.org/project/Heterogeneous%20Computing/preprints/topofix-draft.pdf)
Instanced Tessellation on Current Hardware

• In the vertex shader
  - Load patch index from bucket’s patch list
  - Load edge tessellation level to stitch boundaries
  - Load control points to evaluate surface
  - Apply Displacement Mapping
Watertight Tessellation

• Floating point precision issue
  - Addition is not always commutative as it should be
  - Special care has to be taken to obtain watertight results to prevent cracks
  - Tessellation process involves 3 stages
    - Patch Construction
    - Surface Evaluation
    - Displacement Mapping
Watertight Patch Construction

• Control Points Evaluation
  - Control points type: corner, edge, face
  - In all cases we can evaluate a control point as a weighted sum: \[ P_j = \sum (W_{ij} \cdot V_i) \]
  - Needs consistent ordering of the summation

- Different patch types
Positions:  
\[ P(u,v) = \sum b_i B_i(u,v) \]

Problem:  
\[ a+b+c \neq c+b+a \]

Solution 1:  
All computations need to done in consistent parametric orientation

Solution 2:  
Symmetric evaluation
Watertight Displacement

- **Displacement Mapping**
  - *Sample* displacement value from a texture
  - Displace vertex along its *normal*
Watertight Displacement

• Watertight normals
  - Cross product of a pair of tangnet, bitangent vectors
  - All three vectors should be co-planar
  - **Problem**: \( \text{cross}(\text{tanU}, \text{tanV}) \neq \text{cross}(\text{tanV}, \text{tanU}) \)
  - Discontinuities occur at shared corners and edges
  - Define corner and edge ownership
**Problem:**
Non-watertight Texture Seams
- Due to bilinear discontinuities
- Varying floating point precision on different regions of the texture map
- Seamless parameterization removes bilinear artifacts, but does not solve floating point precision issues

**Solution:**
Define patch ownership of the seams
Watertight Displacement

• Store 4 texture coordinates per vertex (0: interior, 1,2: edges, 3: corner)
  – 16 per quad patch, 12 per triangle patch
  – The corresponding texture coordinate can be selected using the value of the parametric coordinate UV
  – Compute Barycentric interpolation of texture coordinates
Watertight Surface Evaluation

Suggested Reading:

• **Tessellation of Subdivision Surfaces in DirectX 11** (Gamefest 08)

• **Next-Generation Rendering of Subdivision Surfaces** (Siggraph08)
Live Demo

Instanced Tessellation

GPU Patch Construction
Q & A

Tianyun Ni
tni@nvidia.com

Ignacio Castaño
icastano@nvidia.com