Adaptive NURBS Tessellation with CUDA

Presenter: Dr. Qiu Jie

Multi-plAtform Game Innovation Centre (MAGIC), Nanyang Technological University (NTU), Singapore
About MAGIC

- Established in Nanyang Technological University (NTU), Singapore on 1 Nov 2012
- Cross-disciplinary research with 26 professors, 33 researchers, 11 PhD
- To translate scientific ideas into technological products and services
- To increase the capability of Future Studios in media & entertainment
- Launched Future Studios Research Lab (FSR Lab) on 21 Jan 2014
Adaptive NURBS Tessellation

- Tessellation
  - Convert NURBS to a set of polygons

- Adaptive tessellation
  - According to
    - Curvature
    - View (camera distance to object)

Non-adaptive tessellation

Adaptive tessellation

NURBS surface
Tessellating NURBS surface
Adaptive NURBS Tessellation

• How to be adaptive?
  – Divide the surface into patches
  – Choose accurate, effective criterion to tessellate each patch

• How to handle the connection of neighbor parts?
  – Extract neighbour info of each patch to check boundary consistency
  – Stitch gaps if not consistent

• How to speed up?
  – Implementation on GPU
Proposed Framework

Bézier Conversion → Tessellation Factors Calculation → Tessellation → Stitch Gap

To be adaptive
Handle boundaries

GPU implementation
Bézier Conversion

convert NURBS surface to a set of Bézier patches

More accurate for tessellation factor calculation
Tessellation Factors Calculation

Bézier Conversion → Tessellation Factors Calculation → Tessellation → Stitch Gap

n
m
Tessellation Factors Calculation

- One traditional method

\[ \text{tess} \propto \max_i \left( \left\| \Delta^2 \left( w_i P_i \right) \right\| + (r - \varepsilon) \left| \Delta^2 w_i \right| \right) \]

- Related to the origin of the coordinate system
- Heuristic method: Center of bounding box as origin

\( P_i \): control points
\( w_i \): weights of control points
Tessellation Factors Calculation

- Our method
  - Using a new objective function to find origin $O$

$$\min_{O} \left( \sum_{j=0}^{n} (P_j - O)^2 + \sum_{i=0}^{n-2} \Delta^2 w_i \left( \frac{\Delta^2 w_i p_i}{\Delta^2 w_i} - O \right)^2 \right)$$

<table>
<thead>
<tr>
<th>Model</th>
<th>Old method ($\epsilon = 0.1$)</th>
<th>Our method ($\epsilon = 0.1$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>42</td>
<td>36</td>
</tr>
<tr>
<td>2</td>
<td>142</td>
<td>120</td>
</tr>
<tr>
<td>3</td>
<td>417</td>
<td>360</td>
</tr>
</tbody>
</table>

# of vertices
Tessellation

\[
S(s,t) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} w_{i,j} P_{i,j} B_i(s) B_j(t)}{\sum_{i=0}^{n} \sum_{j=0}^{m} w_{i,j} B_i(s) B_j(t)}
\]
Stitch Gap

Problem: inconsistent tessellation factors between patches gap
Stitch Gap

Re-design boundary connections

1 side

4 sides
Implementation on GPU – level of parallelism

- Thread assignment
Implementation on GPU – data management

Bézier Conversion

Tessellation Factors Calculation

Tessellation

Stitch Gap

Minimize data transfer
Implementation on GPU

- NURBS surface to Bézier patches
  - Compute Bézier control points
Implementation on GPU

• Copy control points needed by each block to shared memory
  – Avoid duplicate memory read for global memory

4 X 259 = 1036 points for a block
19 X 19 = 361 points for a block
Implementation on GPU

Bézier Conversion

Tessellation Factors Calculation

Store tessellation factors of all patches in order to extract neighbor info

Tessellation

Stitch Gap
Implementation on GPU

- Calculate tessellation factors
  - Calculate tessellation factors (2 directions)
  - Store them in global memory
Implementation on GPU

1. Bézier Conversion
2. Tessellation Factors Calculation
3. Tessellation
4. Stitch Gap

Pre-compute blending functions
Implementation on GPU

- Tessellation using regular pattern
  - Fast evaluation
    - Each row has the same $B_i(s)$
    - Each column has the same $B_j(t)$

\[
S(s,t) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} w_{i,j} P_{i,j} B_i(s) B_j(t)}{\sum_{i=0}^{n} \sum_{j=0}^{m} w_{i,j} B_i(s) B_j(t)}
\]
Implementation on GPU

- Bézier Conversion
- Tessellation Factors Calculation
- Tessellation
- Stitch Gap

Parallel scan to determine vertex/index offset for each patch
Implementation on GPU

- **Vertex offset**
  - Determine # of vertices for each patch based on tessellation factors
  - Determine vertex offset for each patch

<table>
<thead>
<tr>
<th>Patch ID</th>
<th># vertices</th>
<th>vertex offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(#v)_0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>(#v)_1</td>
<td>(#v)_0</td>
</tr>
<tr>
<td>2</td>
<td>(#v)_2</td>
<td>(#v)_0 + (#v)_1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>k</td>
<td>(#v)_k</td>
<td>SUM((#v)<em>0, (#v)</em>{k-1})</td>
</tr>
</tbody>
</table>
Implementation on GPU

• Transition region and index offset
  – Determine # of indices on each patch (include transition regions)
  – Determine index offset for each patch

<table>
<thead>
<tr>
<th>Patch ID</th>
<th># indices</th>
<th>index offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(#id)_0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>(#id)_1</td>
<td>(#id)_0</td>
</tr>
<tr>
<td>2</td>
<td>(#id)_2</td>
<td>(#id)_0 + (#id)_1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>k</td>
<td>(#id)_k</td>
<td>SUM(#id)_0, (#id)_k-1</td>
</tr>
</tbody>
</table>
Implementation on GPU

- Minimize data transfer
- Store patch info for easy neighbour info extraction
- Regular tessellation pattern
- Parallel scan
Video Demonstration

load NURBS surface
## Results

<table>
<thead>
<tr>
<th></th>
<th>Head</th>
<th>Old man face</th>
<th>Stegosaurus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patch No.</td>
<td>360</td>
<td>3828</td>
<td>12312</td>
</tr>
<tr>
<td>Triangle No.</td>
<td>2298</td>
<td>10216</td>
<td>25530</td>
</tr>
</tbody>
</table>
Results

- CPU (intel i7-2600 3.4GHz) vs GPU (GTX590 512 CUDA cores 160GB/s bandwidth)
Results

- Car model
  - Error tolerance 0.05
  - 407 surfaces
  - 3,081,930 triangles
  - 10.4 ms
Results

- Submarine model
  - Error tolerance 0.05
  - 365 surfaces
  - 2,235,934 triangles
  - 9.16 ms
Acknowledgement

- This research is supported by Multi-plAtform Game Innovation Centre (MAGIC), funded by the Singapore National Research Foundation under its IDM Futures Funding Initiative and administered by the Interactive & Digital Media Programme Office, Media Development Authority of Singapore, and the National Science Foundation of China (Grant no. 60933008, 61272300).
Thank you!