



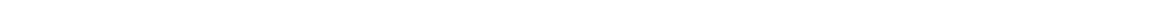
Instanced Tessellation

Tianyun Ni
tni@nvidia.com

Ignacio Castano
icastano@nvidia.com

Document Change History

Version	Date	Responsible	Reason for Change
1.0	07/06/09	tmi	Initial release



1. Abstract

This sample presents an efficient instanced tessellation approach for approximating Catmull-Clark (ACC) subdivision surfaces while DirectX11 class hardware is unavailable. This sample implements three recent ACC schemes[3,4,5], and their adaptive tessellation.

2. Implementation of ACC Schemes

Catmull-Clark subdivision has become a standard modeling tool for decades. A subdivision surface starts with a base mesh (also called input mesh, or control cage), constructed by an artist. This mesh approximates the desired surface. The traditional recursive Catmull-Clark implementation requires at least 4 refinement steps, which indicates multiple passes on the modern GPU and large memory bandwidth to pass through the intermediate refined meshes. A direct patch representation is desired instead. [3,4,5,6] provide such solutions. The general algorithm is that each face (formed by four vertices in Figure 1.a: $V_i, V_{i+1}, V_{i+2}, V_{i-1}$) in an input mesh is converted to a polynomial patch representation.

We consider three possible faces in an input mesh:

- a regular quad: a quad where all 4 vertices have 4 neighbors, and the adjacent facets are quads only.
- an irregular quad: a quad that is not regular.
- a triangle

A regular quad is converted into a bi-cubic patch by the standard B-spline to Bezier conversion rules (Figure 1 b).

An irregular quad can be converted to one of the following:

- a Bicubic patch (Figure 1 c.1) and a pair of tangent patches (Figure 1 c.2) [1], short as Bezier ACC.
- a Gregory tensor product patch that has 20 control points. (Figure 1 e), short as Gregory ACC.
- c-patch with 24 control points (Figure 1 d), short as Pm ACC. c-patch[6] is a special case of Pm-patch[5] where each face in an input mesh is a quad.

A triangle can be converted to one of the following:

- a Gregory triangular patch that has 15 control points, also short as Gregory ACC.
- a Pm-patch with 19 control points, also short as Pm ACC.

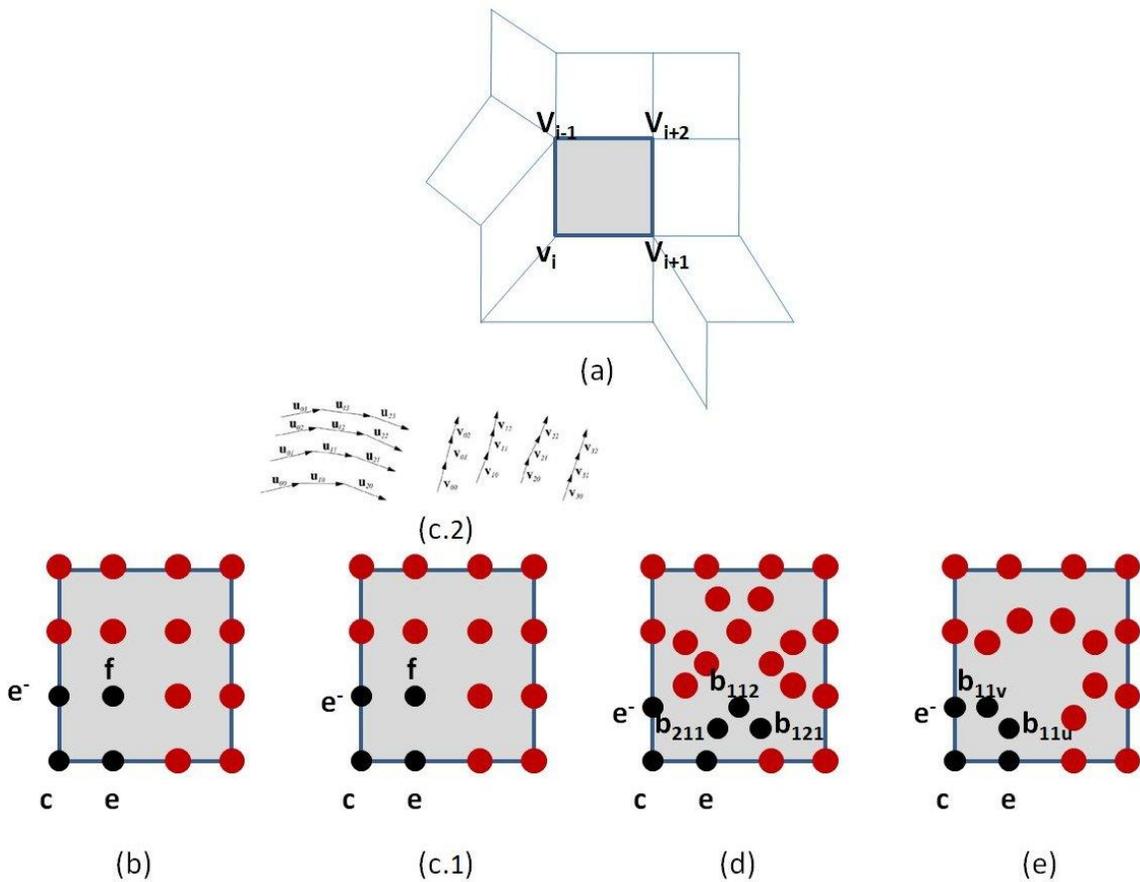
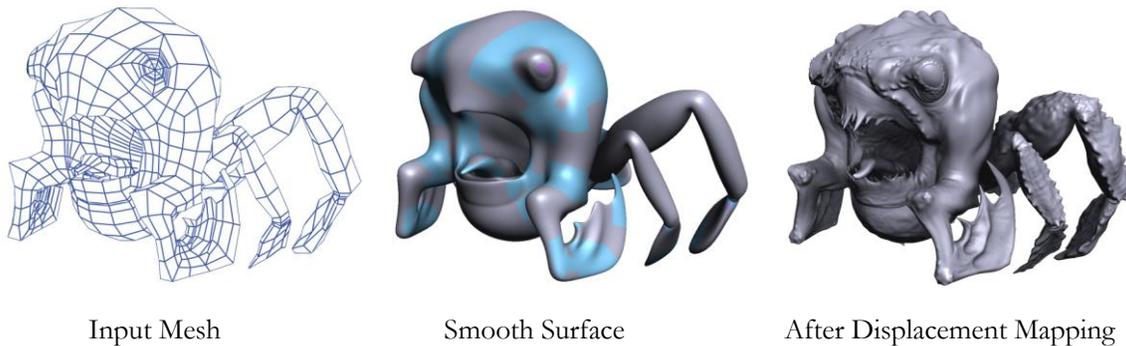


Figure 1: Patch Conversion

Each ACC scheme needs to derive three types of control points: corner points edge points, and interior points (“c”, “e”, and “f/b” in Figure 1 respectively).

We implemented all abovementioned ACC schemes, and a feature to compare them with Catmull-Clark subdivision surfaces. We will explain how to use our sample to view the resulting images from each scheme and the comparison of the correspondent Catmull-Clark subdivision surface in Section 5.

3. Instanced Tessellation Approach

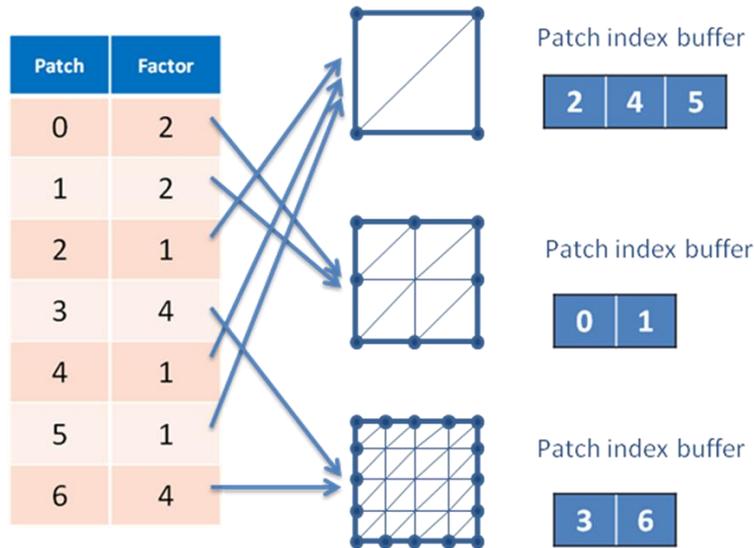


The overall instanced tessellation process consists of 3 major steps:

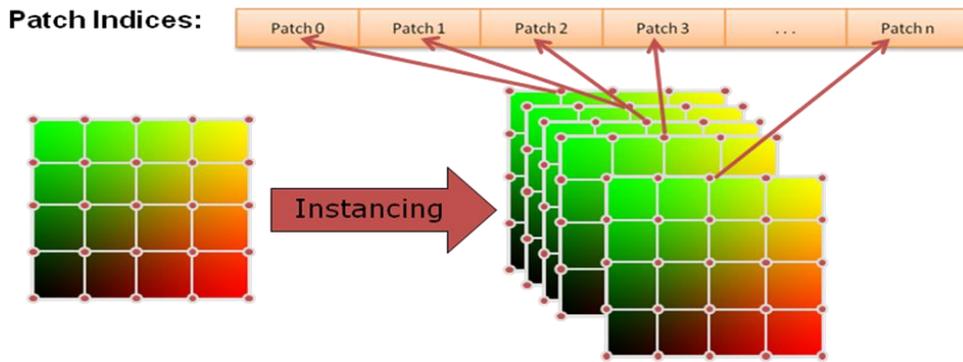
Step 1: An input mesh is first converted into a set of parametric patches by using an ACC scheme. The patches are represented by their control points. This step is processed in our internal baker tool. All control points are stored in a Bezier file and we take it as the input to our sample.

Step 2: The parametric patches are evaluated based on their types as well as control points. This evaluation takes place in the vertex shader and generates a smooth surface. In the above figure, blue shows the areas converted from irregular quads, and purple indicates the areas converted from triangles. Instead of one patch evaluation per draw call, we use instancing technique to greatly save state changes overheads and significantly improve performance.

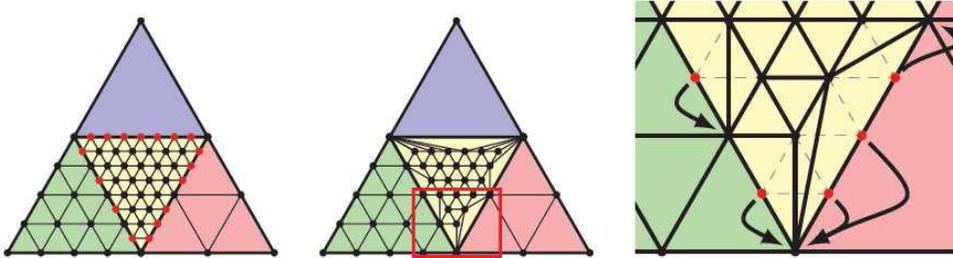
The idea behind Instanced Tessellation technique is to define a generic refinement pattern [1] that is replicated for each face of the input mesh. A generic refinement pattern is just a regular tessellation of the tessellation domain. The index and vertex buffers of this refinement pattern can be pre-computed for the preset tessellation level. This uniform tessellation solution only allows rendering tessellated meshes with a fixed level of detail. To improve performance, a desired tessellation needs to be view-dependent and curvature-dependent. That means we only tessellate the surfaces where needed. For example, using a higher level of tessellation pattern for the patches closer to the camera, and those patches define more curved surfaces.



In adaptive tessellation, tessellations factors differ among patches. We pre-compute tessellation patterns for all possible tessellation factors. For simplicity, all tessellation factors have to be power of 2. Each tessellation pattern is stored into one bucket. Multiple draw calls are issued so that each draw call renders the patches that are in the same bucket. Then the patches within each bucket are rendered simultaneously using a different refinement pattern with the corresponding level of detail.



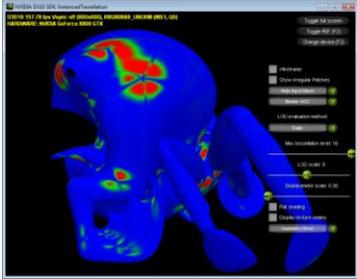
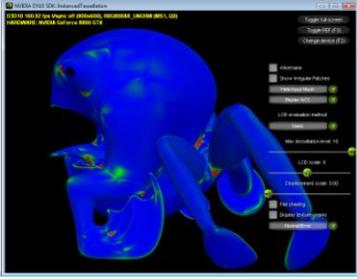
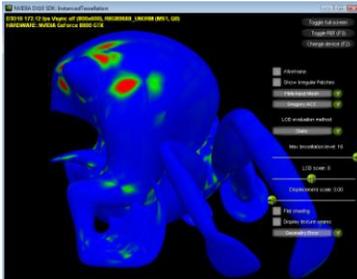
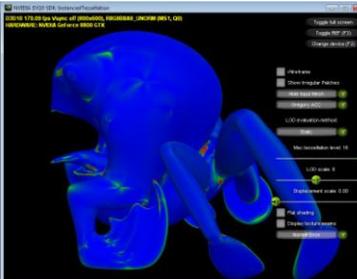
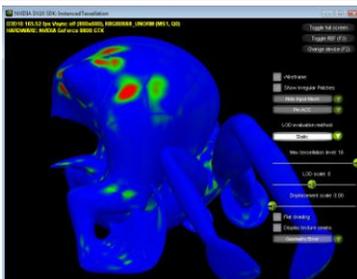
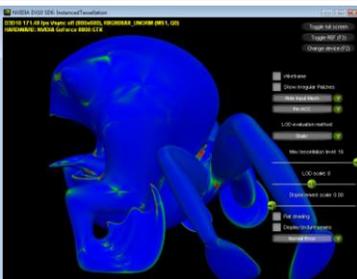
The main problem of this approach is that the resulting mesh will have T-Junctions along the edges of patches with different tessellation levels. Several solutions have been proposed to solve this problem. We use a simple approach provided by [2], where T-Junctions are eliminated by using a snap function that moves boundary vertices and collapses triangles to make adjacent patches match up correctly. This T-junction elimination process is illustrated in the figure below.



Step 3: Displacement Mapping also happens in the vertex shader right after the position and normal are evaluated. The final lit pixel is calculated in the pixel shader.

5. Running the sample

- “Wireframe” checkbox: check it to show the wireframe
- “Show Irregular Patches” checkbox: check it to show irregular quadrilateral patches in blue, and irregular triangular patches in purple.
- “Input Mesh” combo box: make selections to see the input mesh.
- “Bezier ACC, Gregory ACC, Pm ACC” combo box. Make selections to show Instanced Tessellation using the chosen ACC scheme.
- Two LOD evaluation methods: static, or dynamic. Static method (uniform tessellation) uniformly tessellates each patch while dynamic method (adaptive tessellation) wisely choose a LOD for each patch according to viewing distance from it and its curvature.
- The slider for the tessellation level: choose any tessellation level for uniform tessellation.
- The slider for LOD scale: adjust it to scale adaptive tessellation.
- Displacement Scale: scales the displaced value in displacement mapping
- “Flat Shading” checkbox: check it to turn on flat shading
- Display Texture seams
- Turn on/off error comparison with Catmull-Clark (CC) subdivision surfaces. We compare the above three ACC schemes with CC subdivision due to the popularity of CC in standard modeling packages and feature films. Two error measurements are provided here. “Geometry Error” measures the difference of the surfaces positions between an ACC scheme and CC subdivision. To correctly view “Geometry Error”, make sure that “Displacement Scale” is set to 0. The other error measurement is called “Normal Error” that shows the normal angle variations between an ACC scheme and CC subdivision. Since the regular patches in any ACC scheme reproduces CC subdivision surfaces, both geometry error and normal error are zeroes, indicated in blue.

	Geometry Error	Normal Error
Bezier ACC		
Gregory ACC		
Pm ACC		

References

- [1] “**Generic Mesh Refinement on Gpu**”, Tamy Boubekeur and Christophe Schlick., In *ACM SIGGRAPH/Eurographics Graphics Hardware*, 2005.
 - [2] “**Semi-Uniform Adaptive Patch Tessellation.**”, Christopher Dyken, Martin Reimers, and Johan Seland, *Computer Graphics Forum*, http://folk.uio.no/erikd/pdf/topofix_draft.pdf.
 - [3] “**Approximating Catmull-Clark Subdivision Surfaces with Bicubic Patches**”, Charles Loop, Scott Schaefer, *ACM Transactions on Graphics*, Volume 27, issue 1, 2008.
 - [4] “**Approximating Subdivision Surfaces with Gregory Patches for Hardware Tessellation**”, Charles Loop, Scott Schaefer, Tianyun Ni, Ignacio Castano, to appear in *SIGGRAPH Asia*, 2009.
 - [5] “**Fast Parallel Construction of Smooth Surfaces from Meshes with Tri/Quad/Pent Facets**”, A. Myles, T. Ni, J. Peters In *Symposium on Geometry Processing*, July 2 - 4, 2008, Copenhagen, Denmark, pages 1–8. Blackwell, 2008.
 - [6] “**GPU Smoothing of Quad Meshes**”, T. Ni, Y. Yeo, A. Myles, V. Goel, J. Peters, In *IEEE International Conference on Shape Modeling and Applications*, 2008.
-

Notice

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent or patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. NVIDIA Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

Trademarks

NVIDIA, the NVIDIA logo, GeForce, and NVIDIA Quadro are trademarks or registered trademarks of NVIDIA Corporation in the United States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2009 NVIDIA Corporation. All rights reserved.

**NVIDIA.**

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com