Game Developers Conference® | **March 9-13, 2010** | Moscone Center | San Francisco, CA

THINK SERVICES
A DIVISION OF UNITED BUSINESS MEDIA LLC

GDC 10

Learn. Network. Inspire.

www.GDConf.com

Game Developers Conference®
March 9-13, 2010
Moscone Center
San Francisco, CA
www.GDConf.com

Direct3D 11 Tessellation: More Detail, Less Storage

Tianyun Ni, NVIDIA

Tessellation on Characters

© Kenneth Scott, id Software

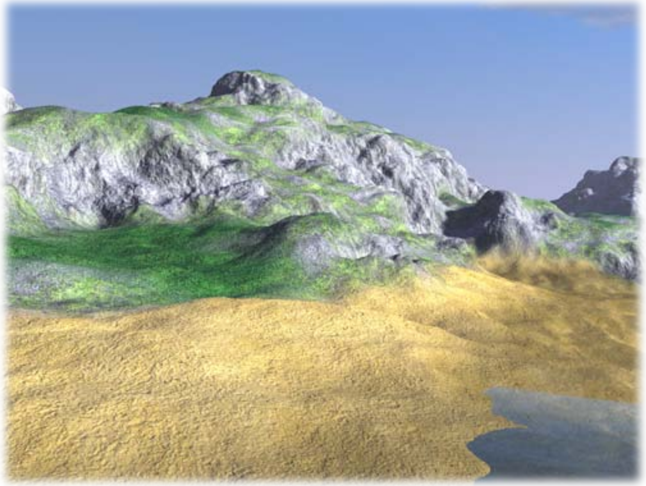
© Mike Asquith, Valve

© Bay Raitt

Tessellation on Environmental Objects

Screenshot from Unigine website

Tessellation in other areas...


Terrain Rendering

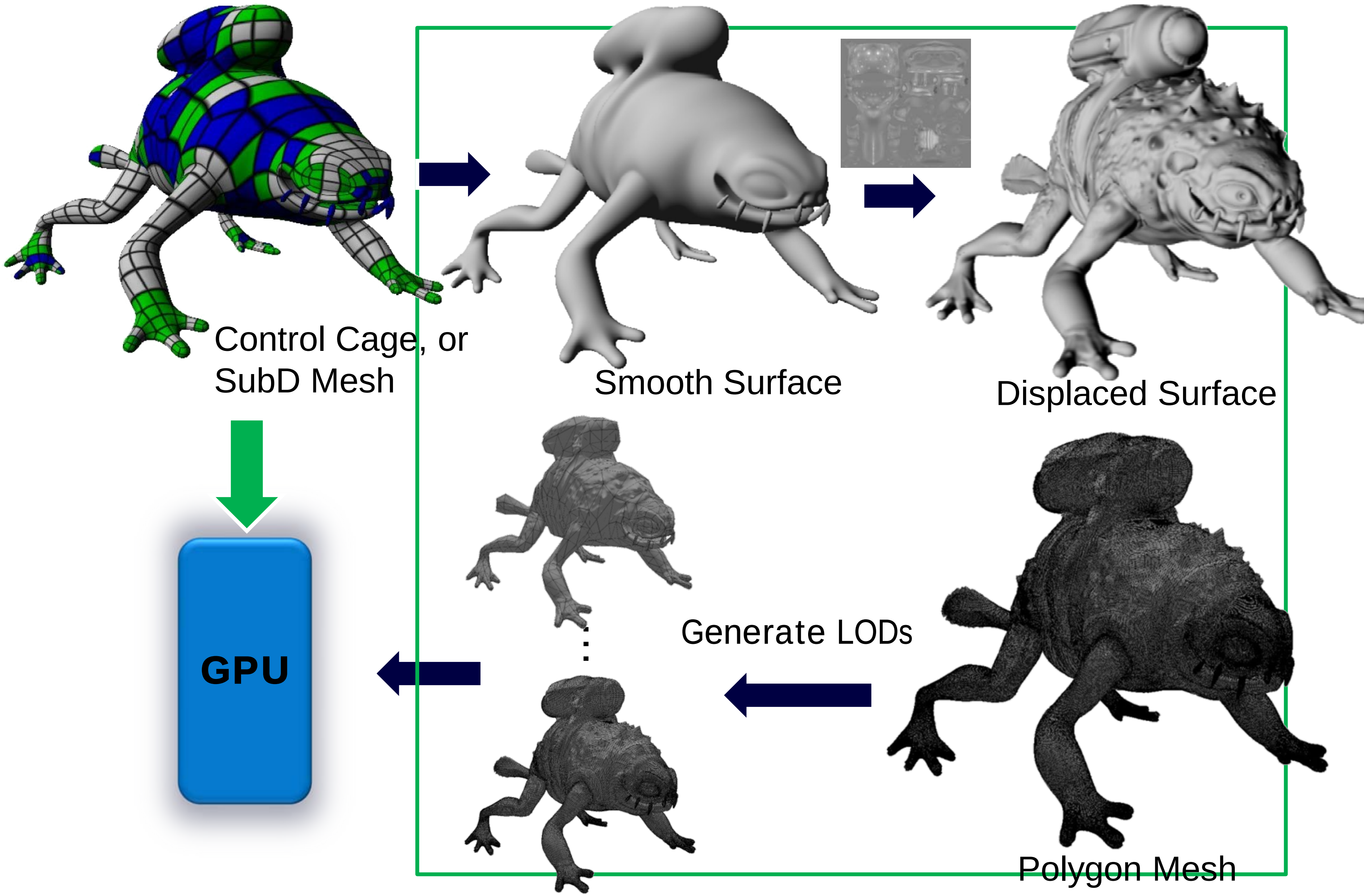

Hair


Ocean

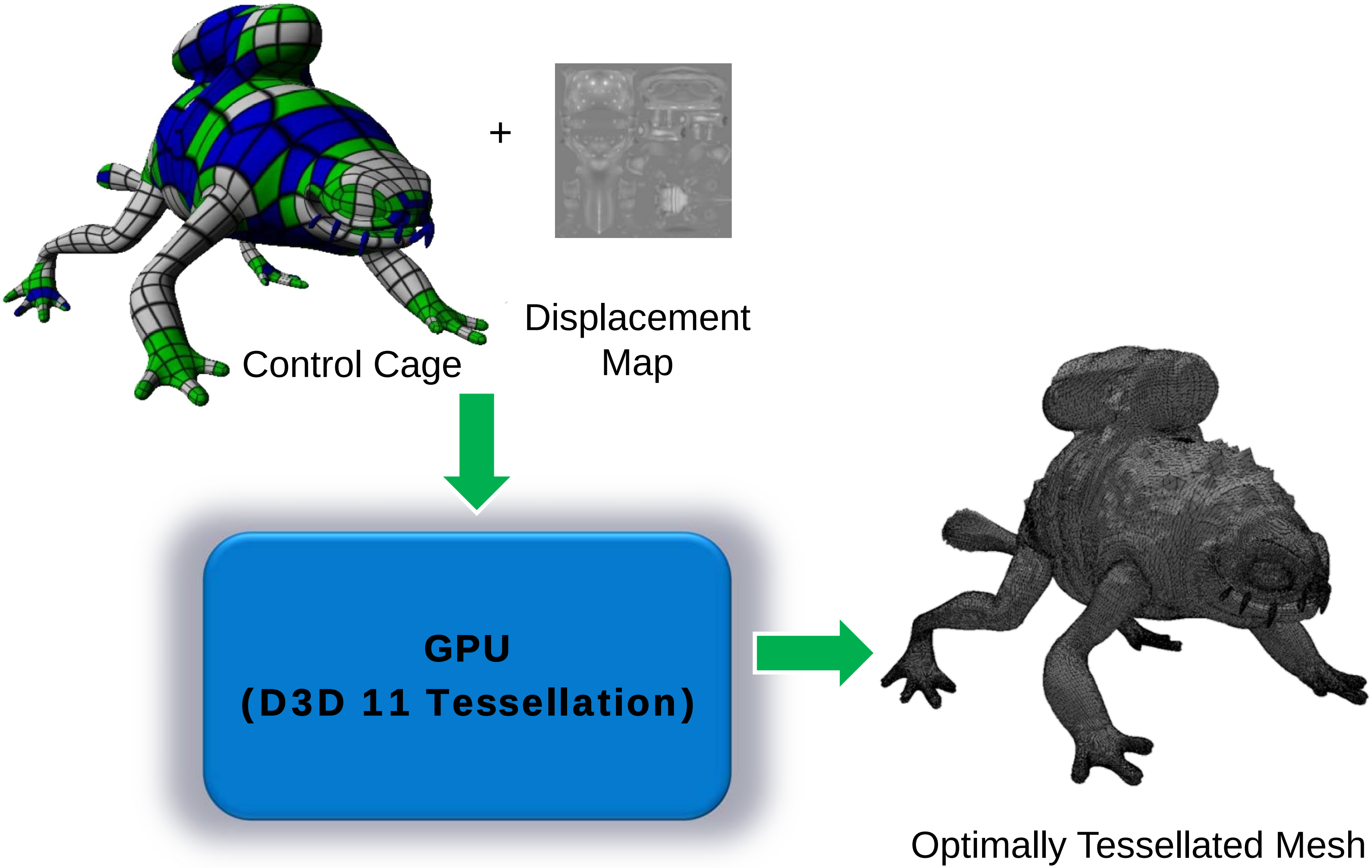

Grass

Other Possible Areas

Current Authoring Pipeline

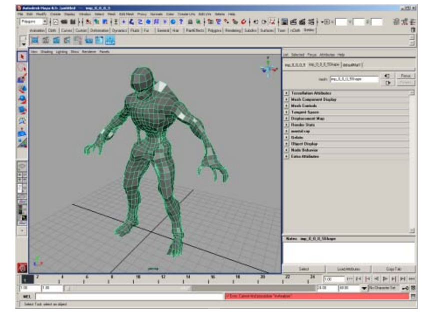


Direct3D 11 Pipeline For Real-time Tessellation Rendering



Content Creation Pipeline

- Modeling Tools
  - Base surface (control cage)



- Sculpting Tools
  - Detailed mesh

- Baker Tools
  - Normal, displacement, occlusion, and other maps

Direct3D11 Tessellation Pipeline

Patch Primitive

↓

Input Assembler

Vertex Shader

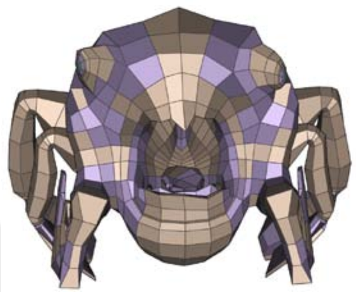
Hull Shader

Tessellator

Domain Shader

Setup/Raster

Pixel Shader

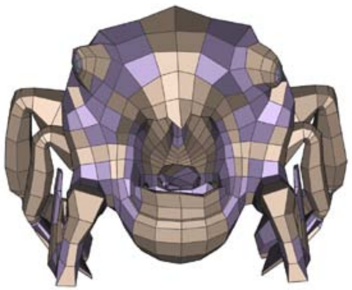


Input Mesh
(a collection of patch primitives)
Displacement Map
Normal Map (optional)

Direct3D11 Tessellation Pipeline

Patch Primitive

- Input Assembler
- Vertex Shader
- Hull Shader
- Tessellator
- Domain Shader
- Setup/Raster
- Pixel Shader



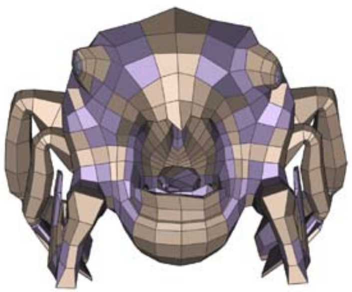
Input Mesh
(a collection of patch primitives)
Displacement Map
Normal Map (optional)

Skinning,…

```
struct    VERTEX
{
    float3 vPosition          : POSITION;
    float2 vUV                : TEXCOORD0;
    float3 vTangent           : TANGENT;
    uint4  vBones             : BONES;
    float4 vWeights           : WEIGHTS;
};
```

Direct3D11 Tessellation Pipeline

Patch Primitive



Input Mesh
(a collection of patch primitives)

Input Assembler

Vertex Shader — **Skinning,…**

Hull Shader — **•Compute Control Points (optional)** **•Compute LOD**

Tessellator — **Geometry expansion**

Domain Shader

Setup/Raster

Pixel Shader

Tessellation Patterns

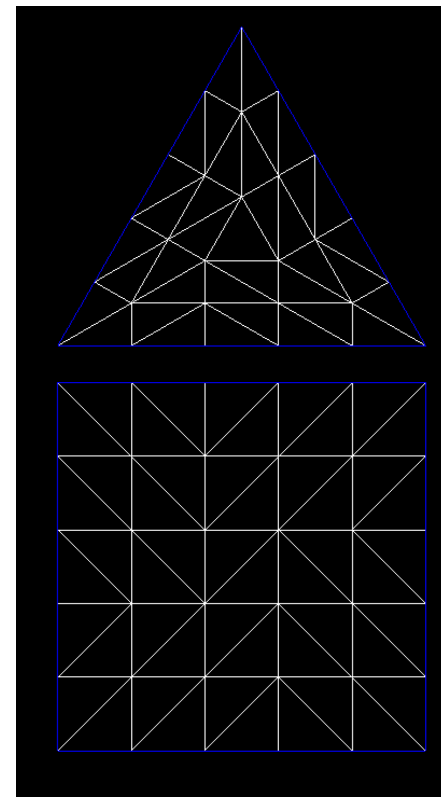
- Let lod be the TessFactor at each edge
- Number of triangles on a triangle domain

  $1+6*\sum_{i=1}^{lod/2}(2*i)$, If lod is odd

  $6*\sum_{i=1}^{lod/2}(2*i-1)$, If lod is even

- Number of triangles on a quad domain

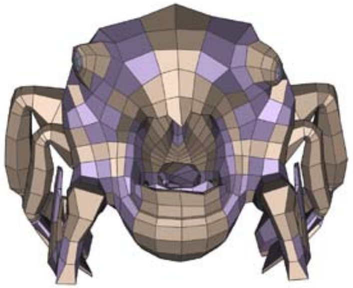
  $2*lod*lod$



Direct3D11 Tessellation Pipeline

Patch Primitive

Input Assembler



Input Mesh
(a collection of patch primitives)

Vertex Shader — **Skinning**

Hull Shader — **•Compute Control Points** **•Compute LOD**

Tessellator — **Geometry expansion**

Patch Surface

Domain Shader — **•Surface evaluation** **•Displacement mapping**

Setup/Raster

Pixel Shader — **•Normal mapping (move to DS stage?)** **•Shading calculation**

High-detailed Mesh

Tessellation Schemes

- Various tessellation schemes differ at
  - Number of vertices in the patch primitive
  - Control points computations (in Hull Shader)
    - Pass through or higher order parametric patch
  - Surface evaluation (in Domain Shader)
    - Barycentric interpolation or higher order parametric patch

Tessellation Schemes

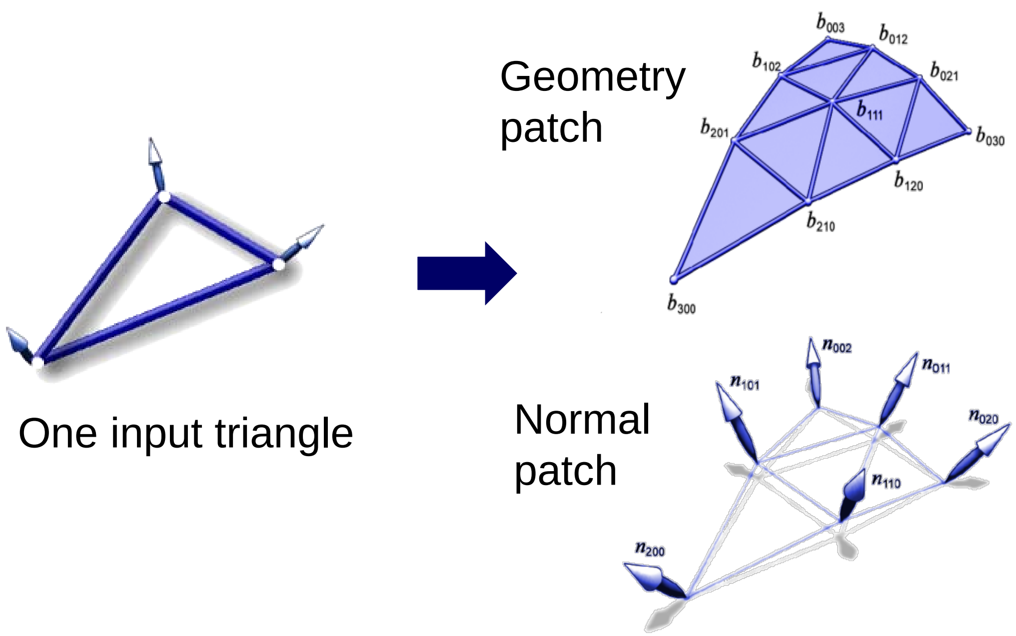
- Choose appropriate schemes for your art assets
  - Tradeoff between performance and visual quality
- Linear interpolation
  - for rendering pebble roads, brick walls, terrain, …
- Local construction schemes
  - PN , Phong Tessellation
- Approximating Catmull-Clark Schemes

Local Construction Schemes

- PN , Phong Tessellation
- Can be applied to tri/quad meshes
- Pros:
  - Fits well with production pipeline
  - Simple, fast
    - less ALU ops
    - 3 or 4 vertices in a patch primitive
- Cons:
  - lower quality surfaces
  - No support in sculpting tools for Displacement Maps creation

PN Schemes

- "**Curved PN Triangles**", by Alex Vlachos, Jörg Peters, Chas Boyd, and Jason Mitchell, I3D 2001.

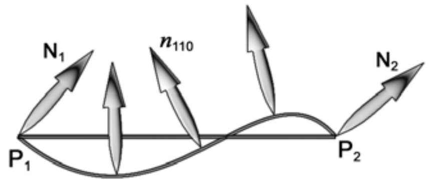


- "**PN Quads**", by Jörg Peters, 2008.
  http://www.cise.ufl.edu/submit/files/file_020f70fe71888f602530143e2e326be2.pdf

  The same formulae except for computing interior control points

Phong Tessellation

- Simpler than PN Triangles
  - uses quadratic geometry patch and phong shading
- Can not handle inflection points



- Needs a relatively dense mesh to start with
- Siggraph 2008 Asia paper, by Tamy Boubekeur and Marc Alexa

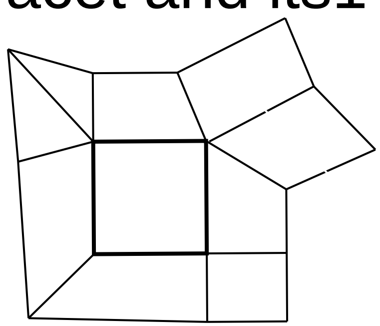
http://perso.telecom-paristech.fr/~boubek/papers/PhongTessellation/

Phong Tessellation

Screenshot from Metro 2033

Approximating Catmull-Clark Subdivision Surfaces Schemes

- Provides movie-quality surfaces
  - Catmull-Clark subdivision surfaces are extensively used in movie production and modeling & sculpting tools
- Suitable for quadrilateral meshes with few triangles in it
- Approximation rather than interpolation
- Requires the mesh info of a facet and its1-ring neighborhood



Approximating Catmull-Clark Subdivision Surfaces (ACC)

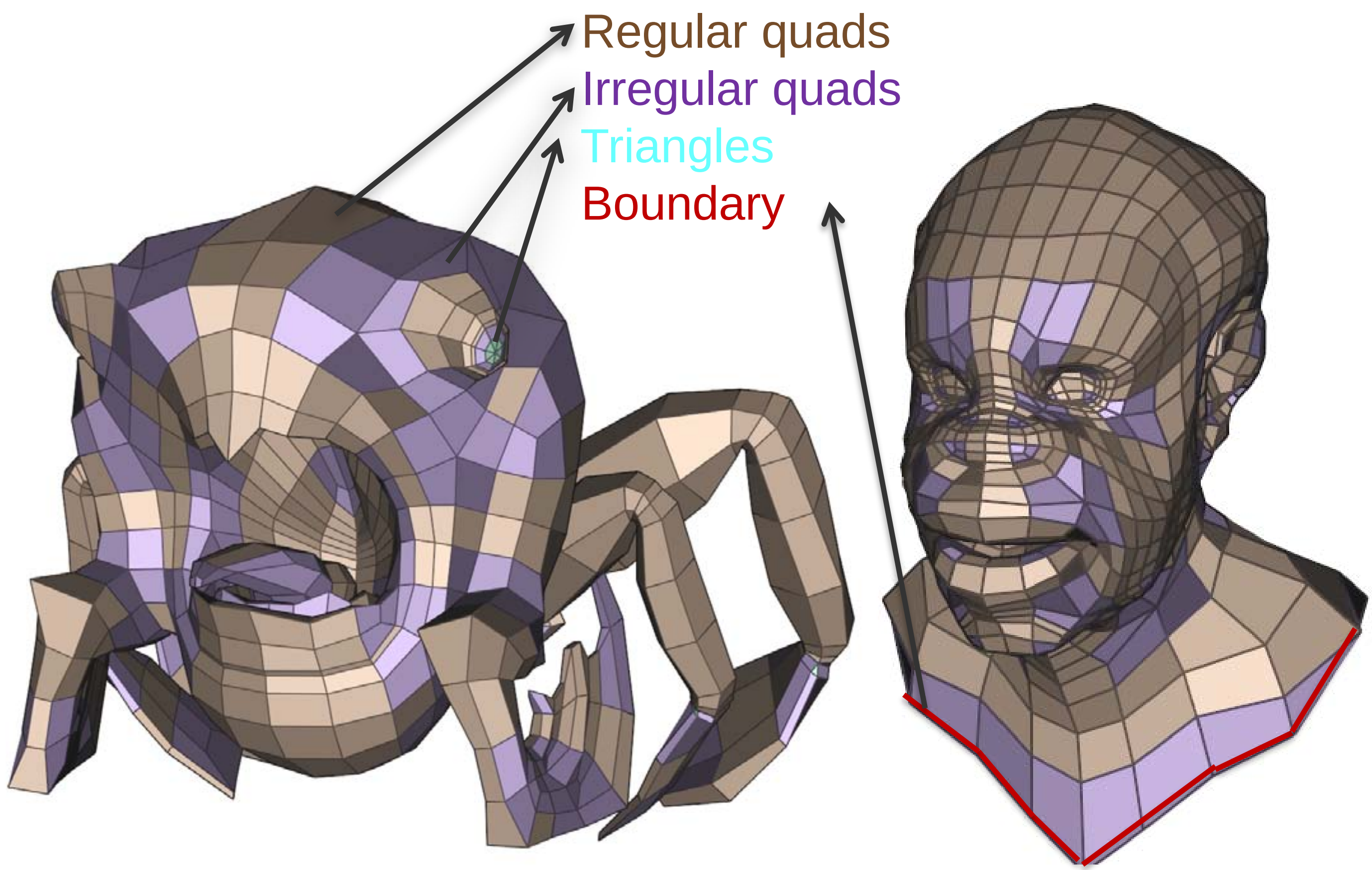
- **Approximating Catmull-Clark Subdivision Surface with Bicubic Patches**” *by Charles Loop and Scott Schaefer,* ACM Transactions on Graphics, Vol. 27 No. 1 Article 8 March 2008.

  http://research.microsoft.com/en-us/um/people/cloop/msrtr-2007-44.pdf
- “**Approximating Subdivision Surface with Gregory Patches for hardware Tessellation**” *by Charles Loop, Scott Schaefer, Tianyun Ni, Ignacio Castano,* Siggraph Asia 2009.

  http://research.microsoft.com/en-us/um/people/cloop/sga09.pdf
  - Extends previous work to a more general mesh that contain quads, triangles and meshes with boundary.
  - Reduces number of control points for faster surface construction and evaluation.

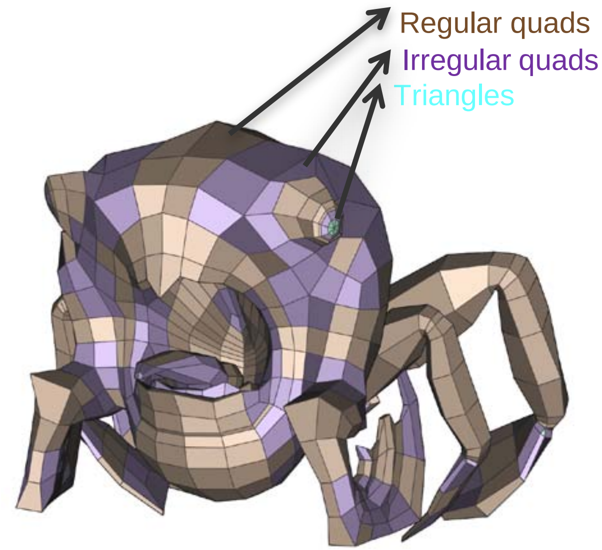
Approximating Catmull-Clark Subdivision Surfaces Using Gregory Patches

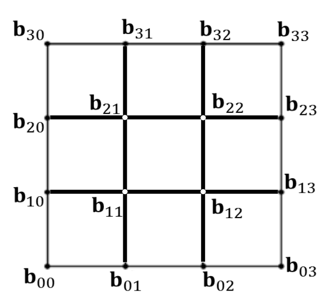
- Flexible ACC scheme for general input mesh



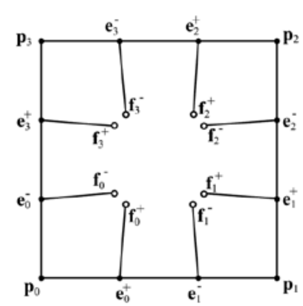
Approximating Catmull-Clark Subdivision Surfaces Using Gregory Patches

- Convert each face of an input mesh to a gregory patch
  - Regular quad ➔ Bicubic Bézier patch
  - Irregular quad ➔ Tensor-product gregory patch
  - Triangle ➔ Triangular gregory patch

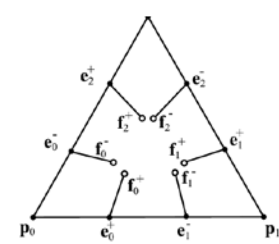




Bicubic Bezier patch



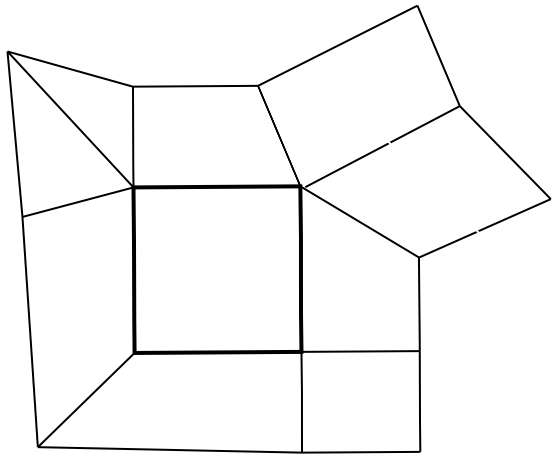
Tensor-product gregory patch



Triangular gregory patch

Control Point Computation

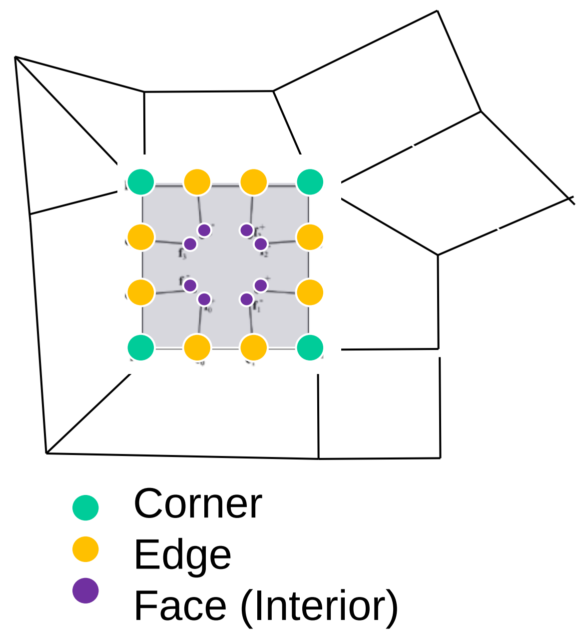
- HS input: a patch primitive (a facet and its neighborhood)



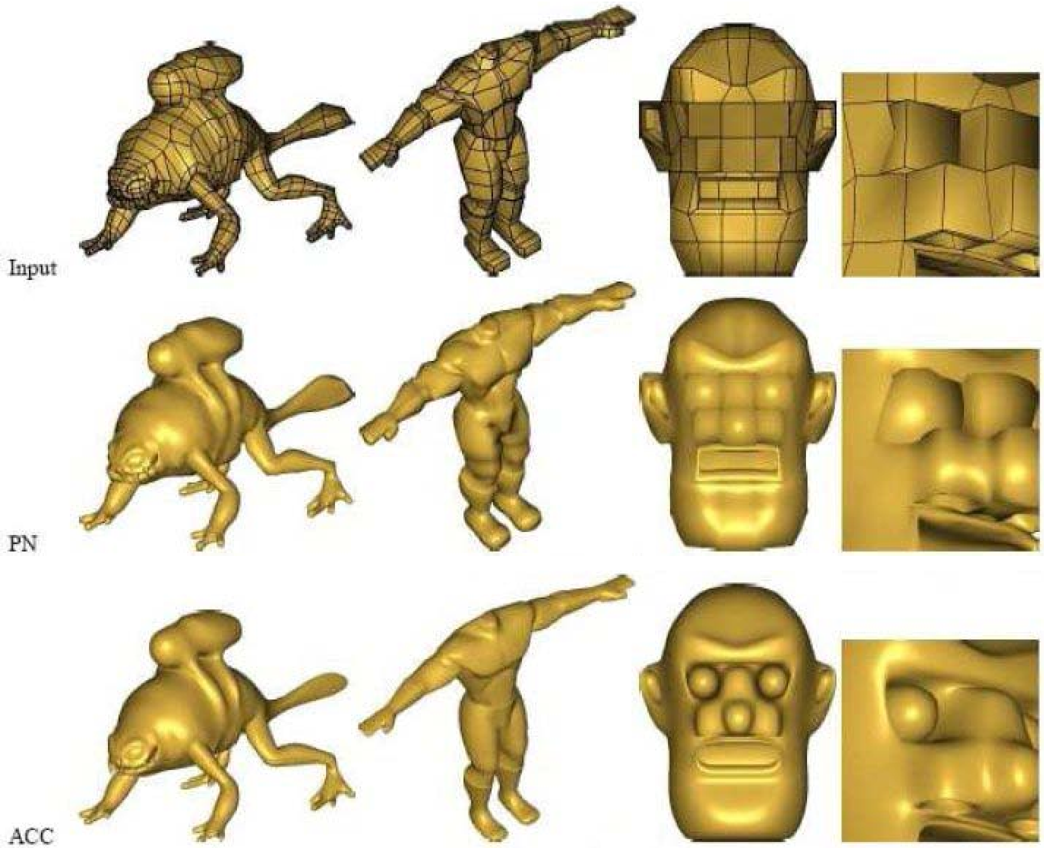
- Each control point is a weighted sum of positions of all vertices in a patch primitive:

  $P_j = \sum(W_{ij} * V_i)$

  the set of weights defined in a stencil rule



Tessellation Schemes Comparison



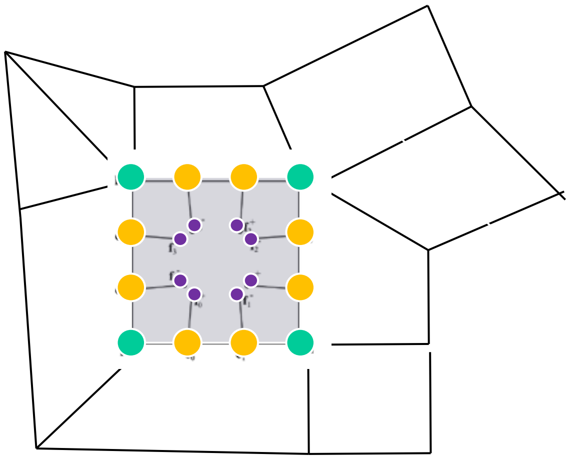
- Tradeoff between quality and speed
- PN
  - Easier to implement
  - Faster
- ACC
  - Better visual quality

Optimization Tips

- Separate regular patches and irregular patches
  - Up to three draw calls
  - Each draw call is for one patch type (regular patch, irregular patch, and a triangular patch)
- For environmental objects (such as trees and rocks)
  - Pack control points into a vertex buffer
  - HS pass these control points down to DS as attributes

Optimization Tips

- Precompute weights
  - Simplify control points computation in the hull shader
- Preprocess to find out the number of vertices whose weights are non-zeroes

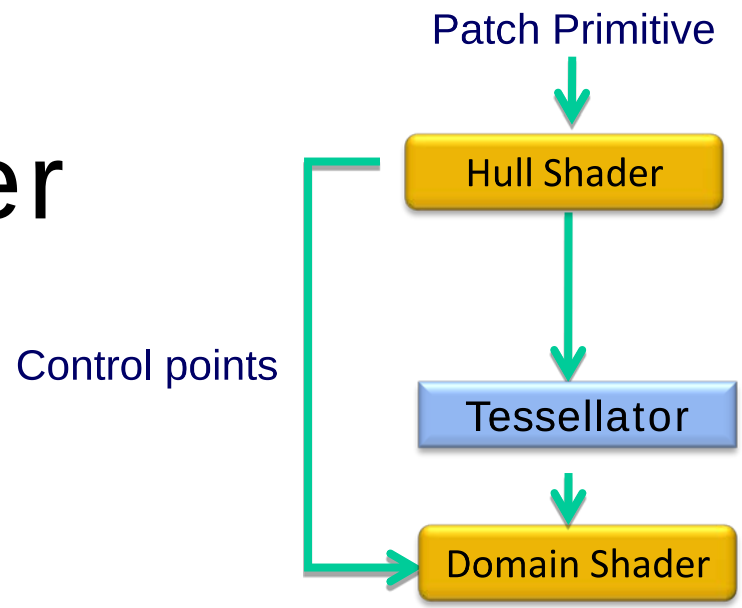


Corner
Edge
Face (Interior)

Water-tight Control Point Computation

- Cracks may occur due to floating point precision issue
  - a+b+c != c+b+a
- Corner and edge control points need to be evaluated “consistently”
- Sum terms must be added in the same order

Hull Shader

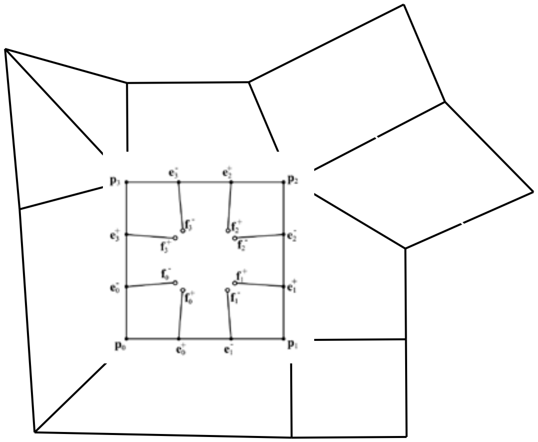


Parametric domain

Number of control points per patch

```
[domain("quad")]
[partitioning("fractional_even")]
[outputtopology("triangle_cw")]
[outputcontrolpoints(20)]
[patchconstantfunc("SubDToGregoryConstantsHS")]
CONTROL_POINT SubDToGregoryHS( InputPatch<VS_CONTROL_POINT_OUTPUT,
                                          primitive_size> p,
                                          uint cpid : SV_OutputControlPointID,
                                          uint pid : SV_PrimitiveID )
{
    CONTROL_POINT output;

    /* compute control point per thread here */

    return output;
}
```

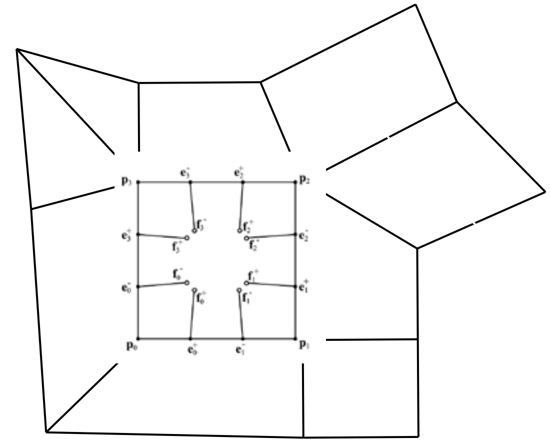
Hull Shader



```
[domain("quad")]
[partitioning("fractional_even")]
[outputtopology("triangle_cw")]
[outputcontrolpoints(20)]
[patchconstantfunc("SubDToGregoryConstantsHS")]
CONTROL_POINT SubDToGregoryHS(
InputPatch<VS_CONTROL_POINT_OUTPUT,
                                            primitive_size> p,
                                            uint cpid : SV_OutputControlPointID,
                                            uint pid : SV_PrimitiveID )
{
    CONTROL_POINT output;

    uint num = Index.Load(int3(pid, 1, 0));

    for (int i = 0; i < num; i++)
    {


    }

    return output;
}
```

the number of vertices in the patch primitive

Hull Shader



```
[domain("quad")]
[partitioning("fractional_even")]
[outputtopology("triangle_cw")]
[outputcontrolpoints(20)]
[patchconstantfunc("SubDToGregoryConstantsHS")]
CONTROL_POINT SubDToGregoryHS(
InputPatch<VS_CONTROL_POINT_OUTPUT,
                                        primitive_size> p,
                                        uint cpid : SV_OutputControlPointID,
                                        uint pid : SV_PrimitiveID )
{
    CONTROL_POINT output;

    uint topo = Index.Load(int3(pid, 0, 0));
```

connectivity type ID for the patch primitive

```
    uint num = Index.Load(int3(pid, 1, 0));

    output.pos = float3(0, 0, 0);
    for (int i = 0; i < num; i++)
    {
        uint idx = Index.Load(int3(pid, 6+i, 0));
```

for consistent computation

```
        output.pos += p[i].pos * weightsForGregoryPatches.Load(int3(cpid,
                                 topo*primitive_size+idx, 0));
    }

    return output;
}
```

Hull Shader (regular case)

- More common and much simpler
- Separate from irregular cases

```
……
[outputcontrolpoints(16)]
BEZIER_CONTROL_POINT SubDToBezierHS( InputPatch<VS_CONTROL_POINT_OUTPUT, 16> p,
                                      uint cpid : SV_OutputControlPointID,
                                      uint pid : SV_PrimitiveID )
{
    BEZIER_CONTROL_POINT output;

    output.pos = float3(0, 0, 0);

    [unroll]
    for (int i = 0; i < 16; i++)
    {
        output.pos += p[i].pos * weightsForRegularPatches [16*i+cpid];
    }

    return output;
}
```

Water-tight Displacement

- **Problem:** 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 texture coordiantes

Water-tight Normal

- Cross product of a pair of tangnet, bitangent vectors
- All three vectors should be co-planar
- Problem: cross(tanU,tanV) $\neq$cross(tanV, tanU)
- Discontinuities occur at shared corners and edges
- Define corner and edge ownership

Creases and Corners

- Add creases and corners to smooth surfaces
  - Tag the edges that generates creases
  - Modified stencil rules
    for those tagged edges

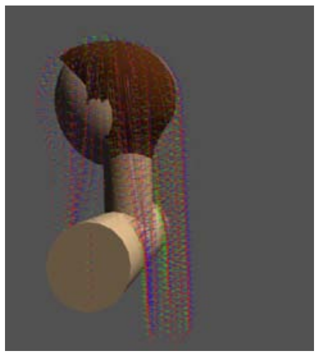
Creases and Corners References

- "**Scalar Tagged PN Triangles**",
  T. Boubekeur, P. Reuter, C. Schlick
  http://iparla.labri.fr/publications/2005/BRS05b/STPN.pdf
- "**PhongTessellation**",
  T. Boubekeur, P. Reuter, C. Schlick
  http://iparla.labri.fr/publications/2005/BRS05b/STPN.pdf
- "**Real-Time Creased Approximate Subdivision Surfaces**",
  D. Kovacs, J. Mitchell, S. Drone, D. Zorin
  http://mrl.nyu.edu/~dzorin/papers/kovacs2009rcs.pdf

Simple Tessellation

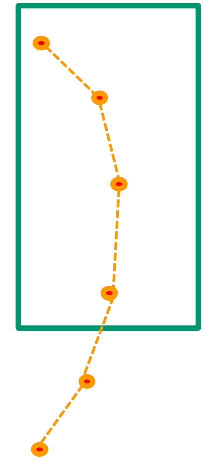
- Linear interpolation of input mesh
- Terrain rendering, Environmental objects,...
  - Pass through in HS
  - Barycentric (or bilinear) interpolation in DS

Hair Tessellation



Input

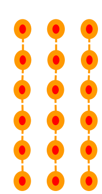
Simulated guide hair

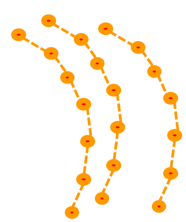


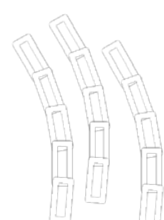
Patch primitive

Output

HS	TE	DS	GS	PS
Calculate LODs	Generate topology	Calculate vertex attributes	Expand lines to quads	Shade







Summary

- Direct3D11 Tessellation enriches visual detail with flexible LOD control
- Choose a tessellation scheme that fits your needs
- Implement it efficiently
- It’s time to bring games to the next level

Thanks