GPU Tessellation

Damon Rocco

Definition

- Tessellation: The filling of a plane with polygons such that there is no overlap or gap.
- In computer graphics objects are rendered as a triangular mesh
  - A triangle is always a plane
- Thus, the triangles in a mesh can be tessellated to increase detail

Why Tessellate?

- In software tessellation provides an interesting way of enhancing detail
- In hardware tessellation allows a simple mesh to be sent down to the GPU, converted to a complex mesh, and then displayed
  - Decrease memory to the card
  - Increase rendering performance by decreasing the number of polygons through the full pipeline

Why Tessellate?

- With programmable tessellation what objects get tessellated can be decided by the programmer
- This allows for objects closer to the screen to be tessellated while objects far away are not
  - Detail actually increases as objects get closer instead of just getting bigger
  - Resources aren’t wasted on meshes that are too far away for tessellation effects to be viewed

Tessellation Hardware Roadmap

- Xbox 360: ATI Xenos graphics chipset
  - The first geometry tessellation unit
  - “Programmable”
- ATI Radeon HD 5xxx series
  - The first DX11 card
- NVIDIA “Fermi”
  - NVIDIA’s first DX11 card

Tessellation Software Roadmap

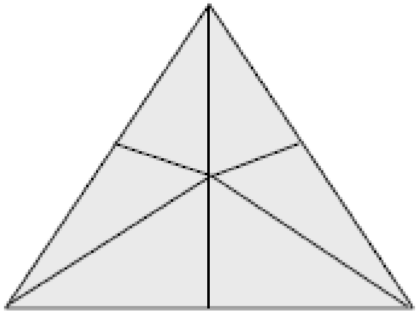
- DirectX 9: *Dynamic Terrain Rendering on GPU Using Real-Time Tessellation*. Natalya Tatarchuk. ATI Research published in ShaderX7
- DirectX 10: *Instanced Tessellation in DirectX10*. Andrei Tatarinov. GDC '08
- DirectX 11: New pipeline features three tessellation stages, 2 are programmable

Dynamic Terrain Rendering on GPU Using Real-Time Tessellation

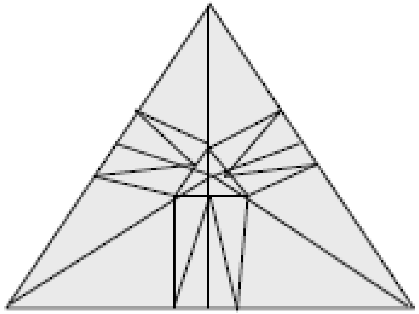
Natalya Tatarchuk

The Work

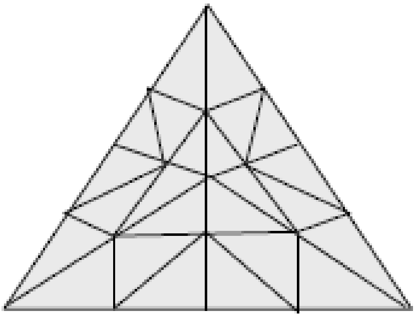
- Goal: Create a software library that would allow tessellation exclusively on ATI cards.
- Product: ATI GPU Tessellation library. An ATI only library that worked in conjunction with DirectX 9 (also 10.x) and was capable of tessellating meshes



Tessellation Level 1.0



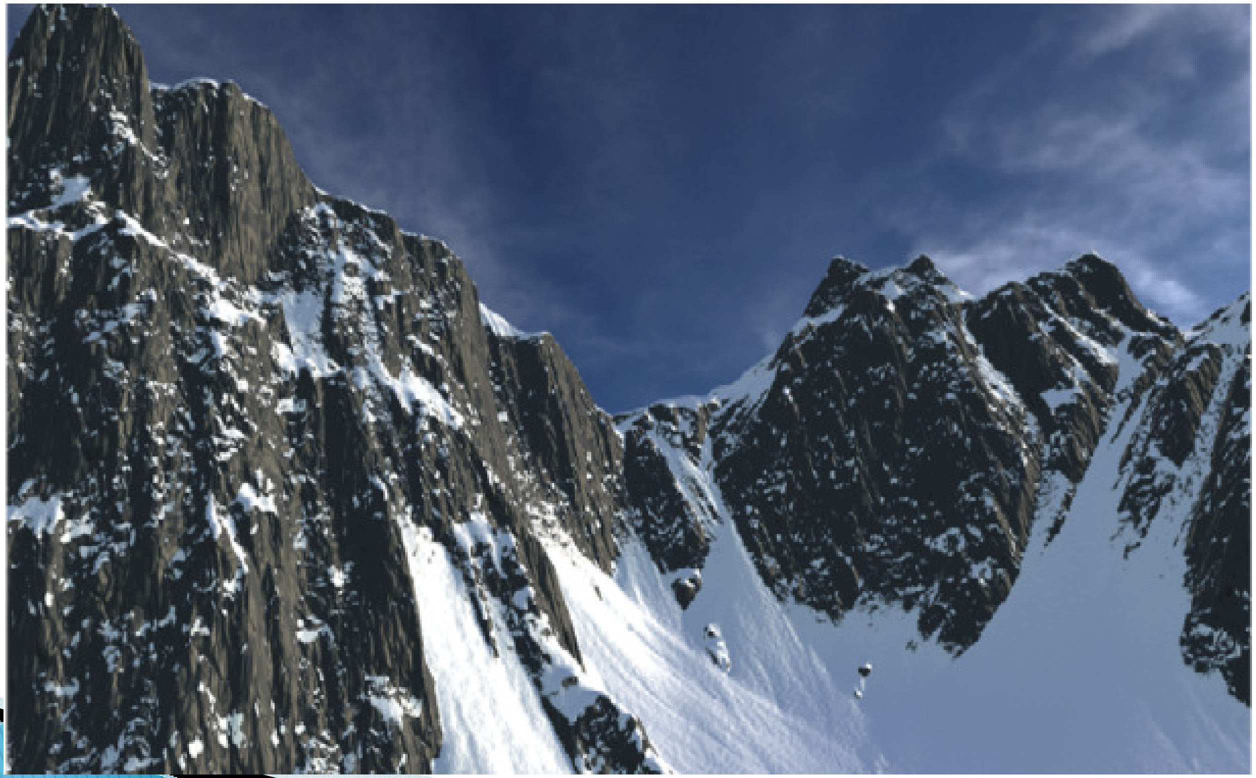
Tessellation Level 1.5



Tessellation Level 3.0

Paper Results

- A flat 900 polygon grid becomes a 1,000,000 polygon mountain-scape at inter-actable rates

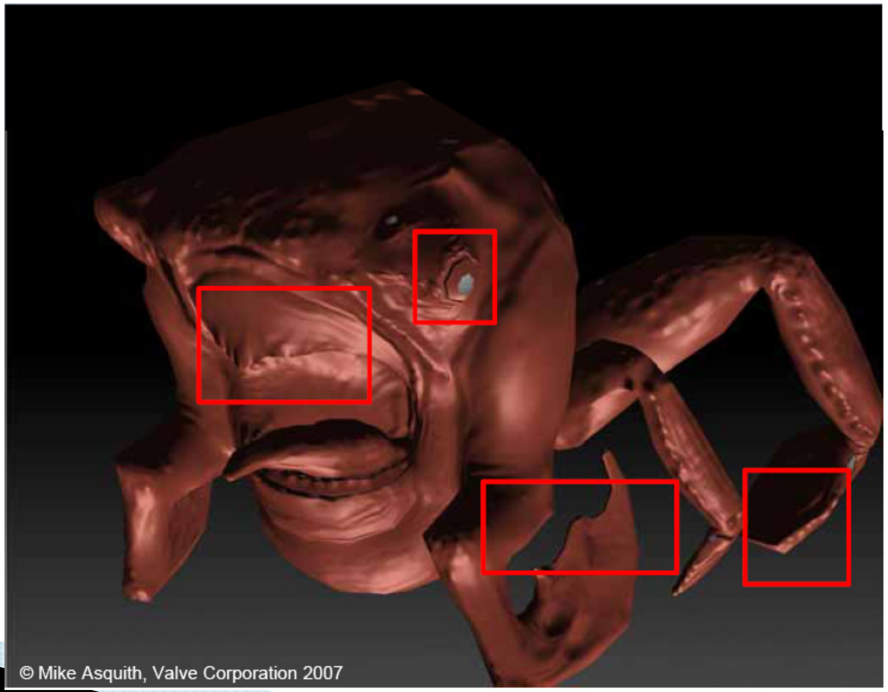


Instanced Tessellation in DirectX10

Andrei Tatarinov

Introduction

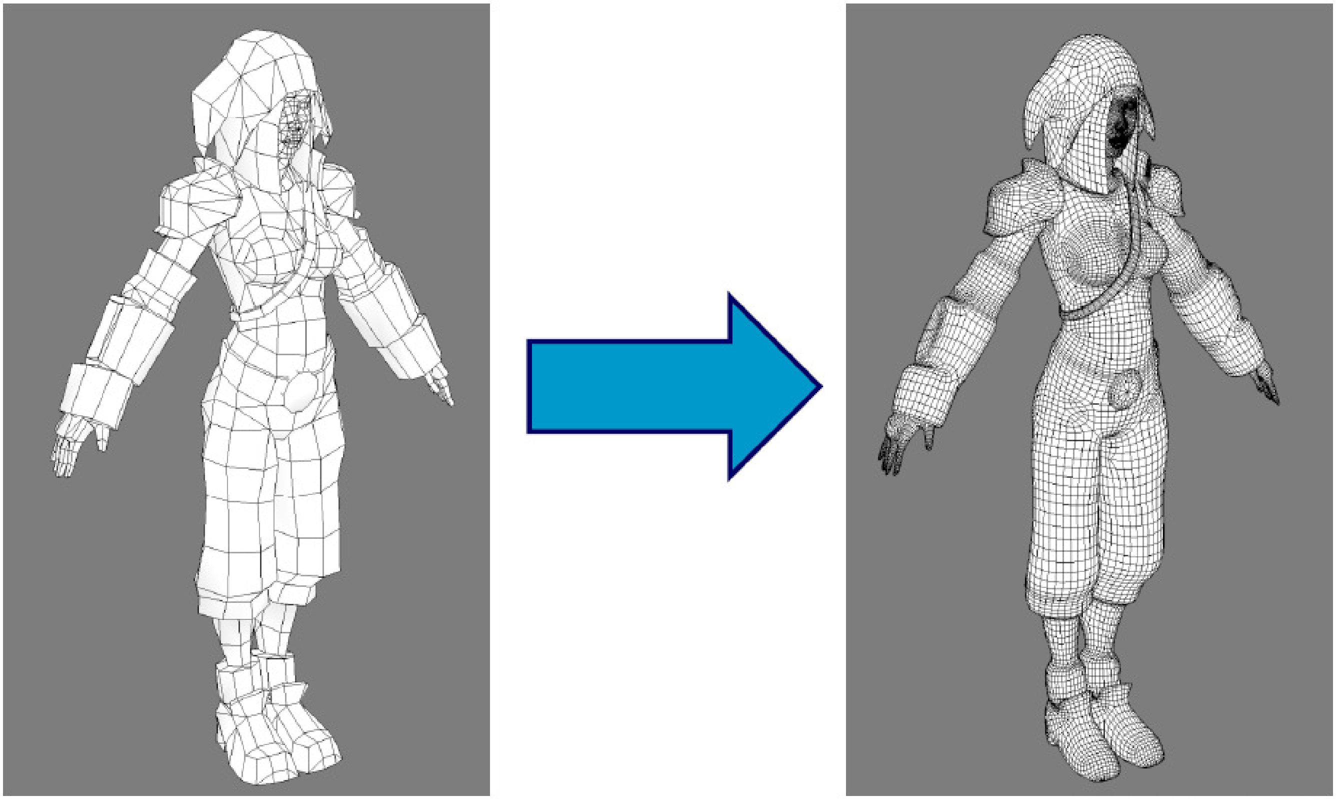
- In the past complex shading models were used to hide lack of detail in a polygonal mesh
- They can only do so much…



© Mike Asquith, Valve Corporation 2007

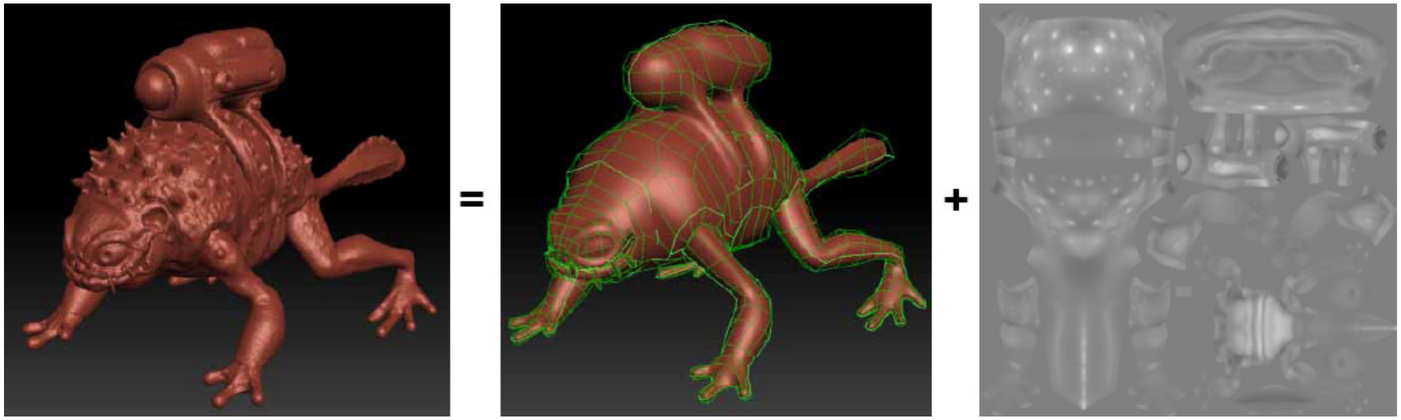
Introduction

- The solution is to use on-the-card tessellation to increase the physical detail in the meshes



Introduction

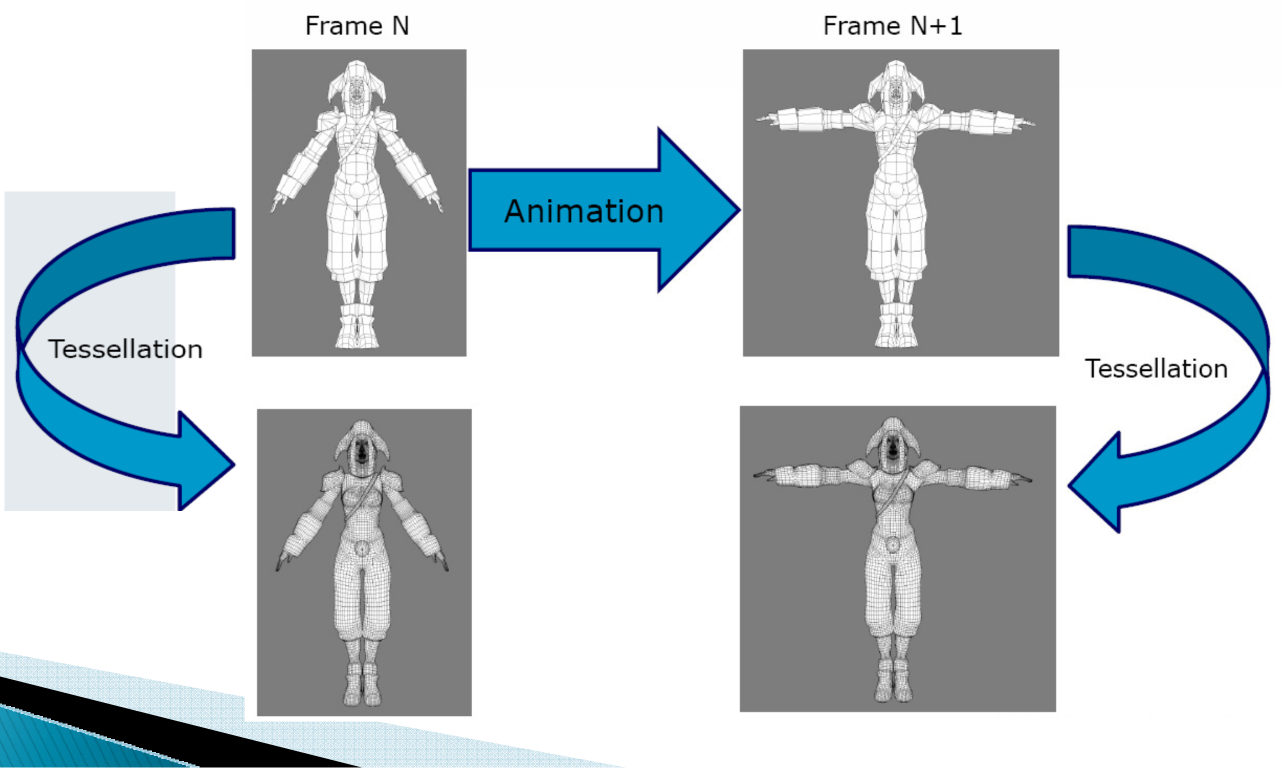
- A highly detailed mesh can be sent to the card as a simple mesh and a displacement map



- Trades ALU operations for memory bandwidth
- ALUs scale faster than bandwidth

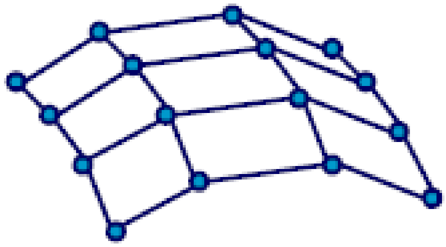
Introduction

- Tessellation can also be used by Animators to make their job easier
  - Animate a low polygon mesh
  - Tessellate and get detail for free



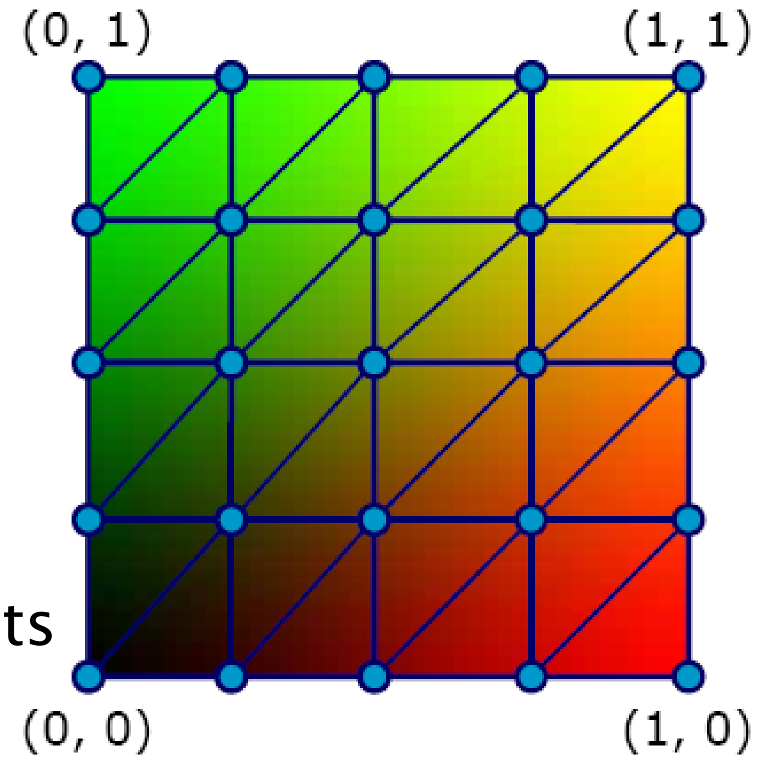
Implementation

- New “patch” primitive defined by a set of control points
- Operation called refinement generates triangle from each patch



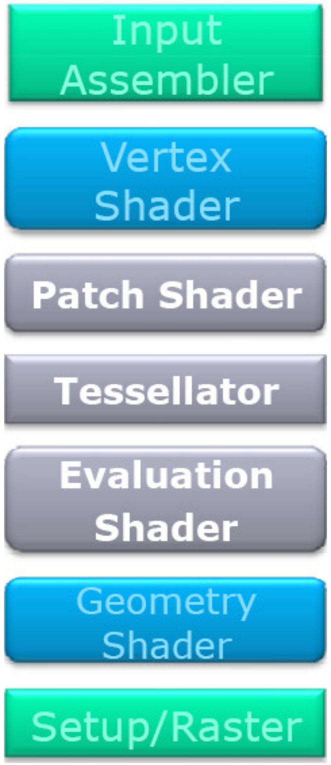
Implementation

- Per-patch operations
  - Level of Detail computation
  - Transfer of Basis
    - Bezier -> B-spline
    - B-spline -> NURBS
    - Etc.
- Generating topology
  - Generates a set of (u,v)-points in the tessellation domain



The "Future"

- Programmable hardware tessellation
  - 3 stages
    - 2 programmable shaders
    - 1 fixed function configurable tessellator
  - New primitive "patch"
    - Curved surface
    - Easily converted to triangles

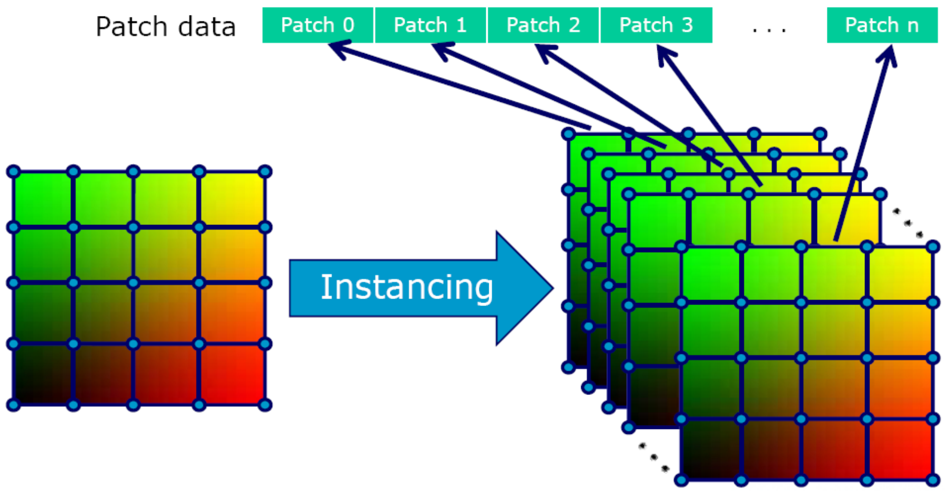


Why Wait?

- Programmable tessellation can be imitated using DirectX 10 features:
  - Geometry Shader and Instancing 2.0

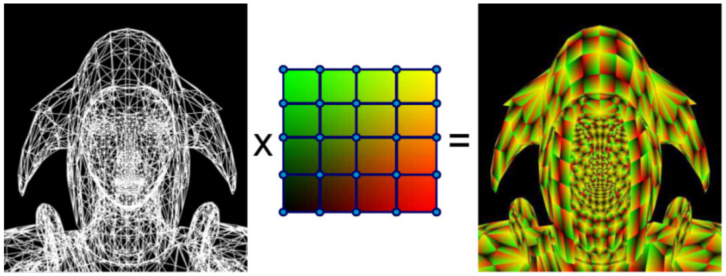
Implementation Details

- Geometry Shader cannot do tessellation itself
  - Outputs triangles serially
  - Maximum output size of 1024 scalars
    - 16x16 grid of float4s
- Instead we can save small pre-tessellated patches as instances

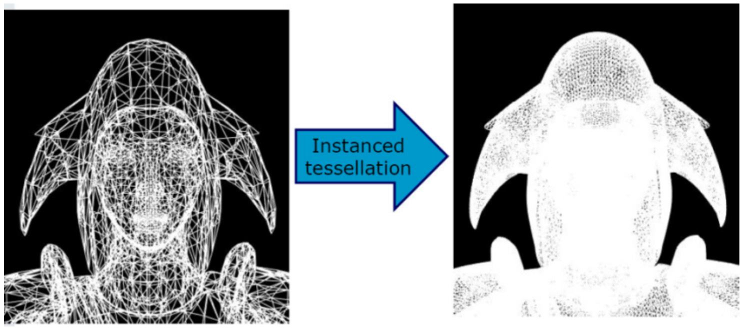


Implementation Details

- The pre-tessellated patches represent the results of tessellating every input patch

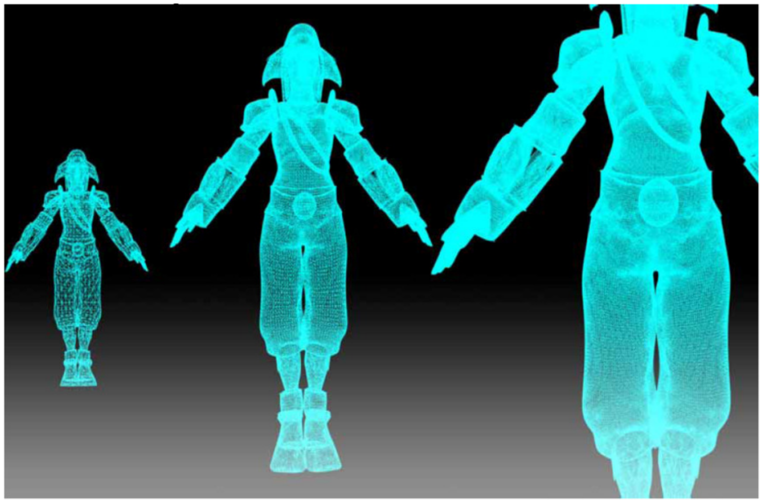


- This data is combined in the vertex shader to produce the desired effects



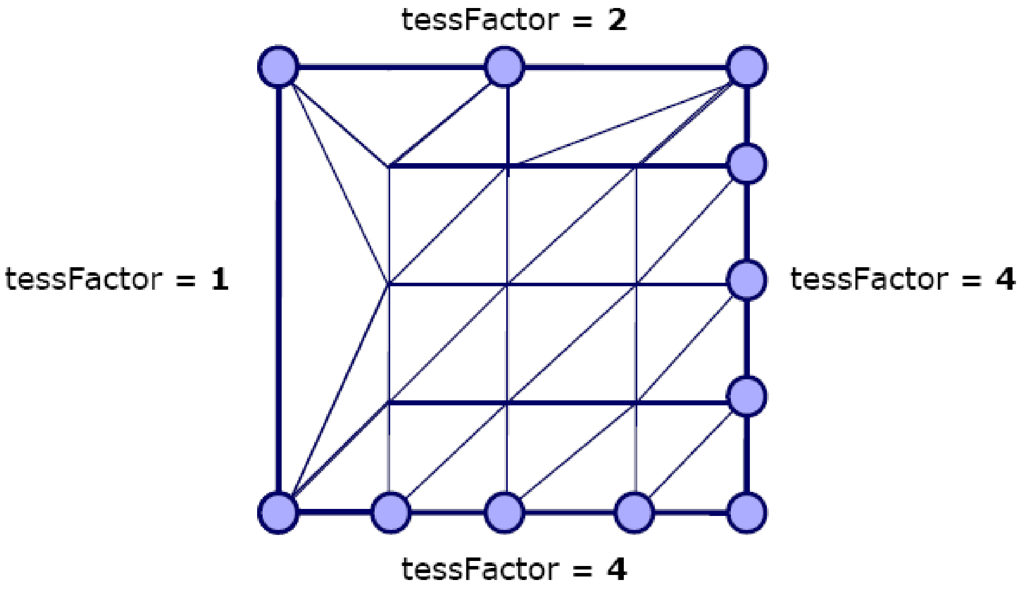
Implementation Details

- Vertex shader inputs are too small to handle an entire mesh
  - Must be bound to shared buffers
- The tessellation mesh instances are of varying quality, a LoD factor is used to determine which mesh to select



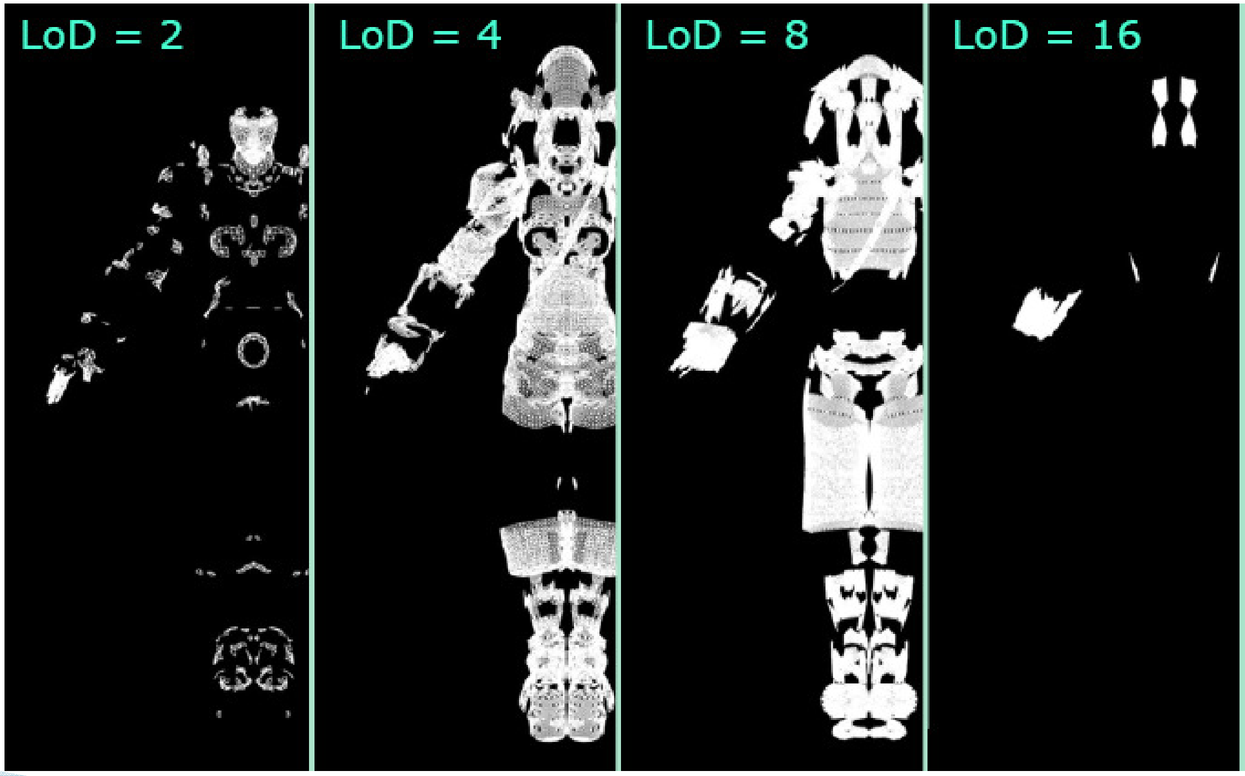
Implementation Details

- Tessellation factors can change across a mesh as each patch edge has its own tessFactor.



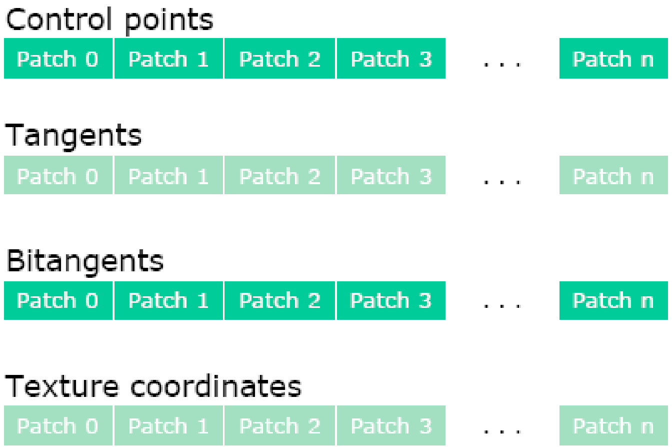
Implementation Details

- Per-patch tessellation factors requires multiple draw calls (it won't in DX11)



Implementation Details

- Since all of the meshes are stored in vertex buffers the only inputs are the primitive id and the vertex id
- U = Vertex ID mod LoD
- V = Vertex ID div LoD
- LoD is based on vertex position



Control points

Patch 0 | Patch 1 | Patch 2 | Patch 3 | . . . | Patch n

Tangents

Patch 0 | Patch 1 | Patch 2 | Patch 3 | . . . | Patch n

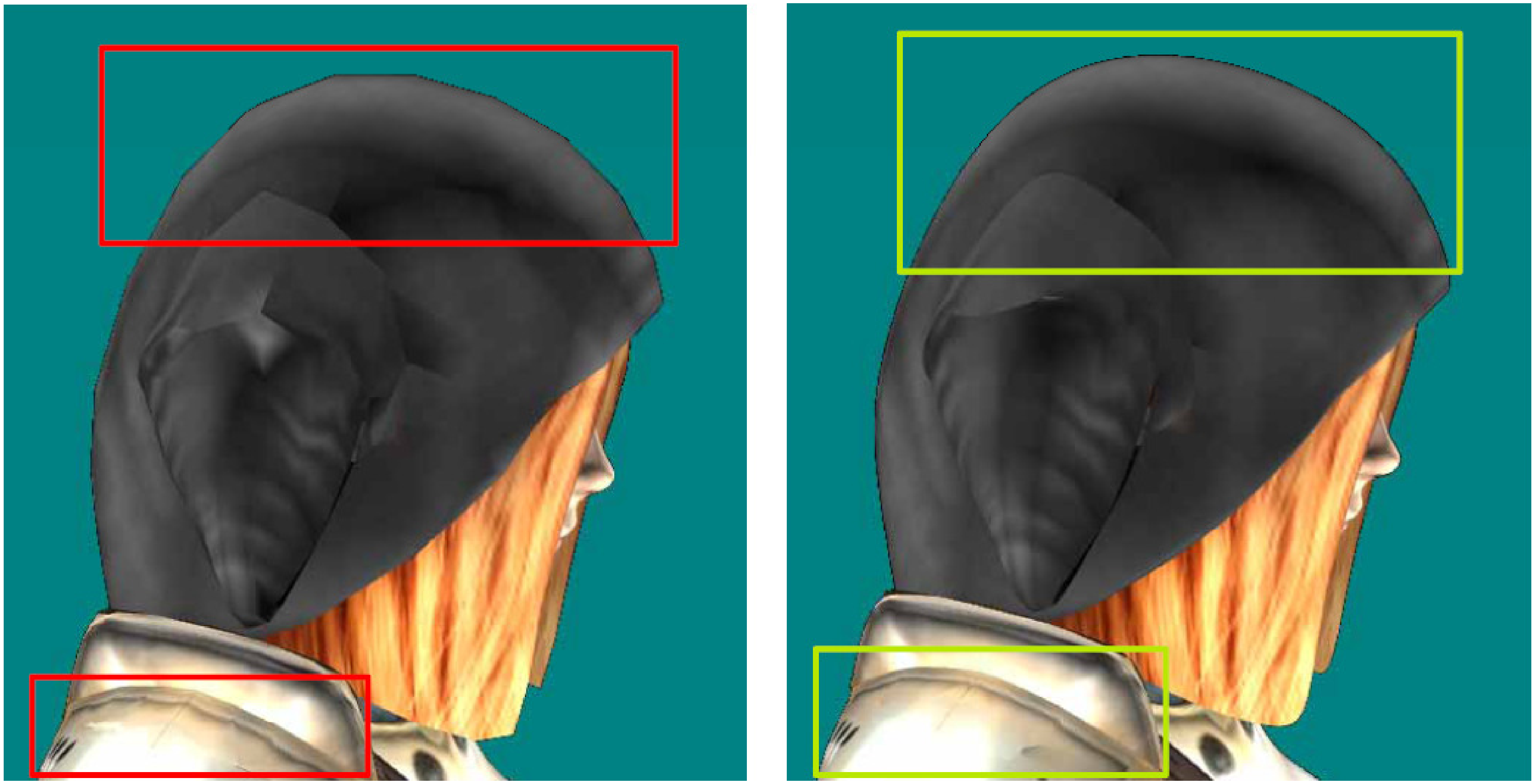
Bitangents

Patch 0 | Patch 1 | Patch 2 | Patch 3 | . . . | Patch n

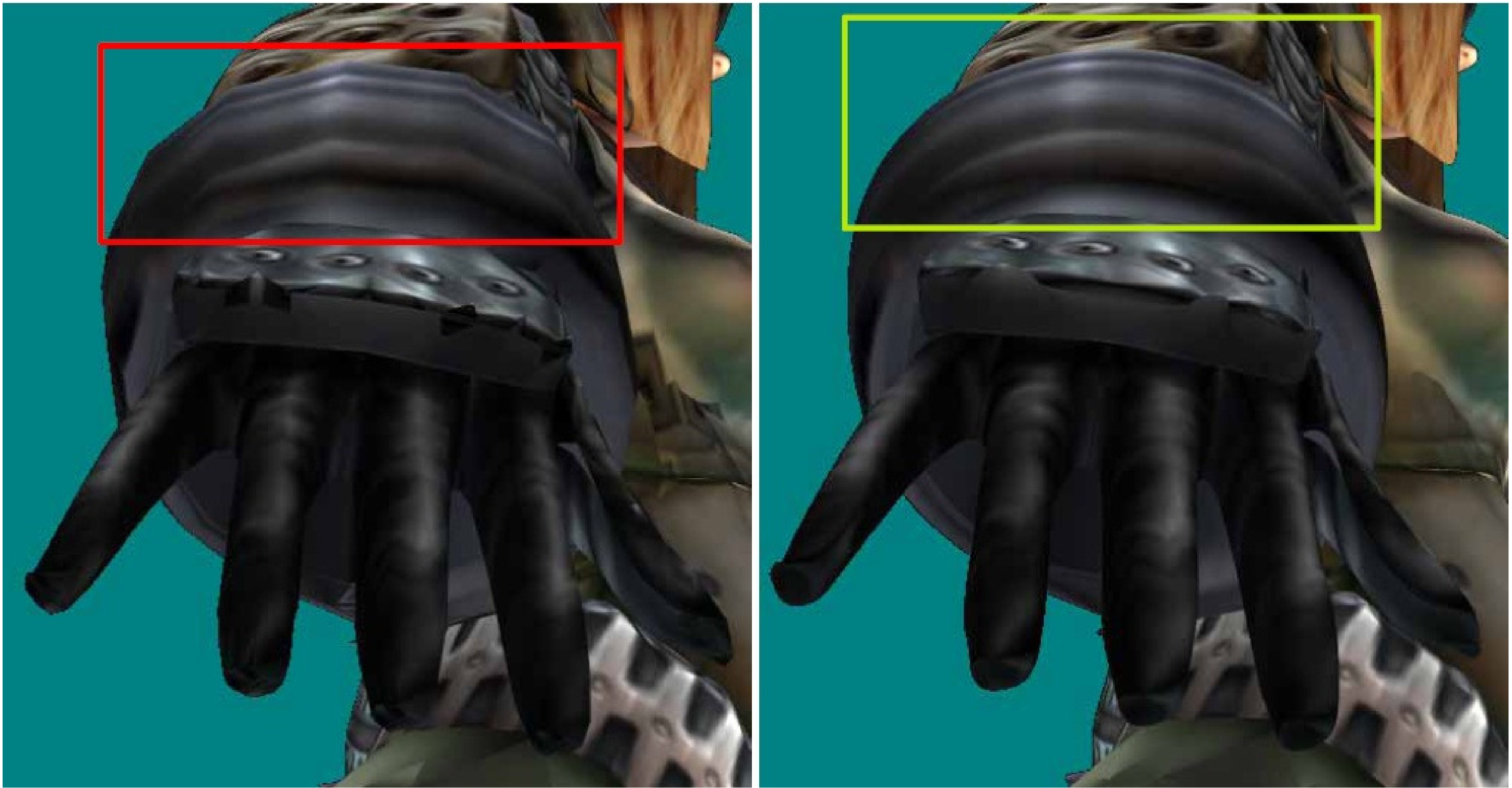
Texture coordinates

Patch 0 | Patch 1 | Patch 2 | Patch 3 | . . . | Patch n

Results

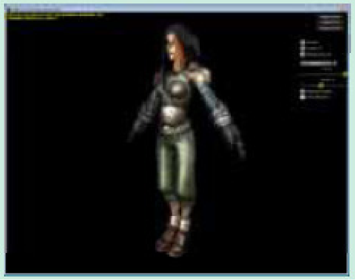
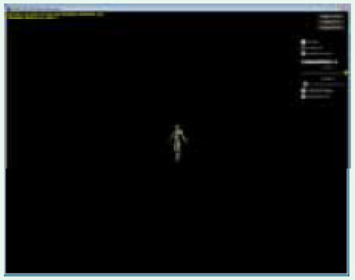


Results



Results

Mesh: 6118 patches, 256 vertices each, 8800 GT

	39.32 FPS	36.70 FPS
	230.61 FPS	40.60 FPS

	Coarse mesh	Height map	Diffuse map	**Total size**
Dynamically tessellated mesh	3 728 KBs	4 096 KBs	4 096 KBs	**11 920 KBs**

	Detailed mesh	Diffuse map	**Total size**
Pretessellated mesh	48 944 KBs	4 096 KBs	**53 040 KBs**

Demo

Takeaways

- Tessellation presents a means of significantly increasing detail without a performance cost
- Tessellation is possible with DirectX 10
- Some of the hiccups with the DirectX 10 implementation will be fixed by future hardware implementations

Introduction to the Direct3D 11 Graphics Pipeline

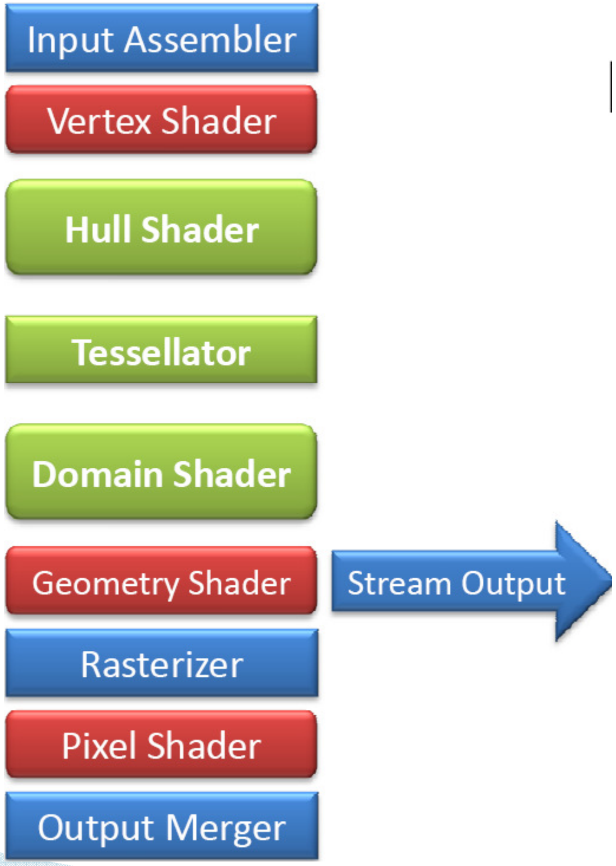
Kevin Gee

DirectX 11 Tessellation

- D3D11 HW Feature
  - Required for DirectX 11 compatibility
- D3D11 Only
  - No direct backwards compatibility
- Fundamental primitive is “patch”
  - Not triangles
- Superset of Xbox 360 tessellation

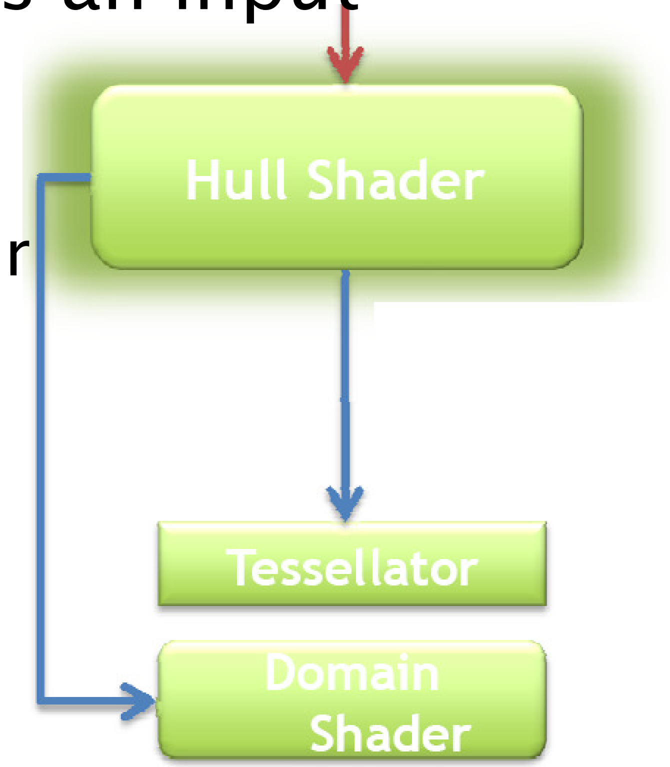
New Pipeline

- 3 new stages added for Tessellation



Hull Shader

- Programmable
- Takes patch control points as an input
- 2 Output paths:
  - Output of basis converted control points to Domain Shader
  - Output of the control points, a tessellation factor, and tessellation modes to the Tessallator



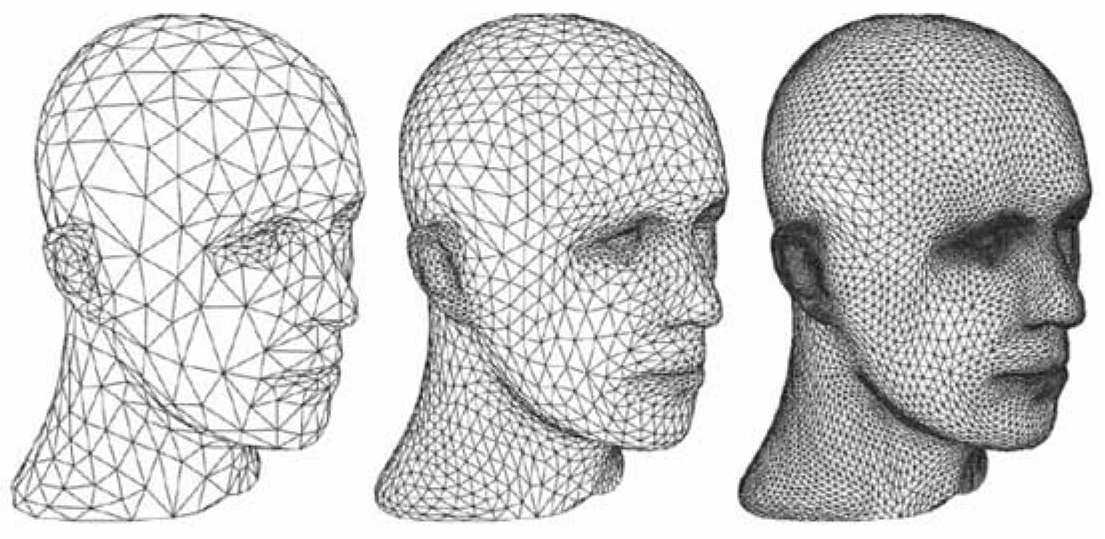
Fixed-Function Tessellator

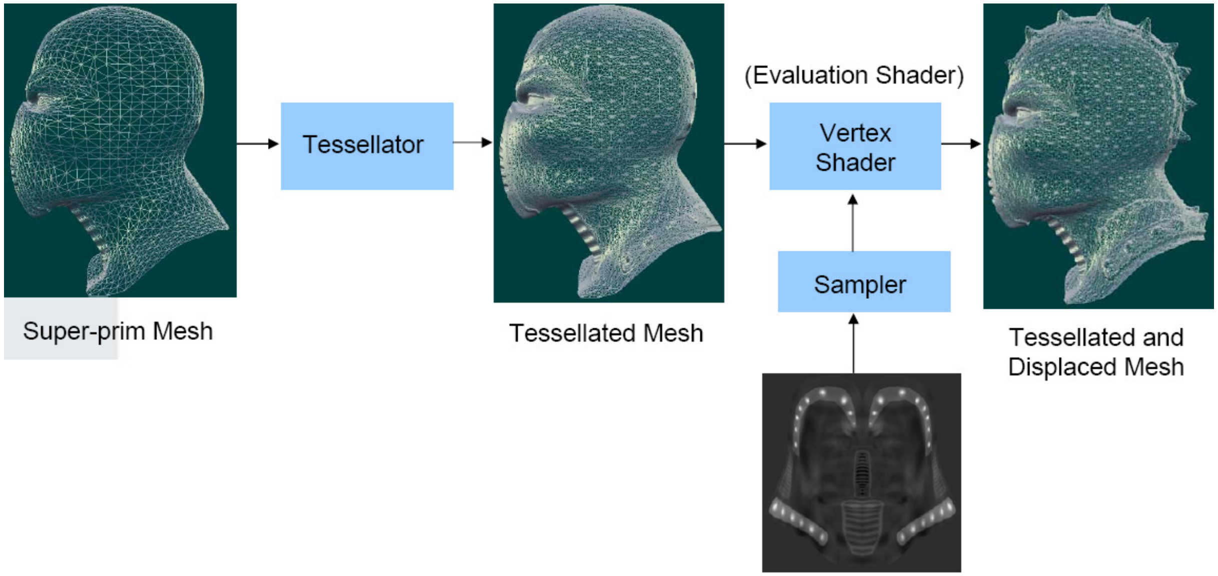
- Tessellator operates on a per patch basis
- Again 2 output paths:
  - Outputs U,V points to Domain Shader for further shading
  - Outputs topology for primitive assembly

Domain Shader

- Either receives a set of points from the Hull Shader or the Tessellator.
- The Domain Shader is invoked once per point
- It outputs verticies

Video





Super-prim Mesh

Tessellator

Tessellated Mesh

(Evaluation Shader)

Vertex Shader

Sampler

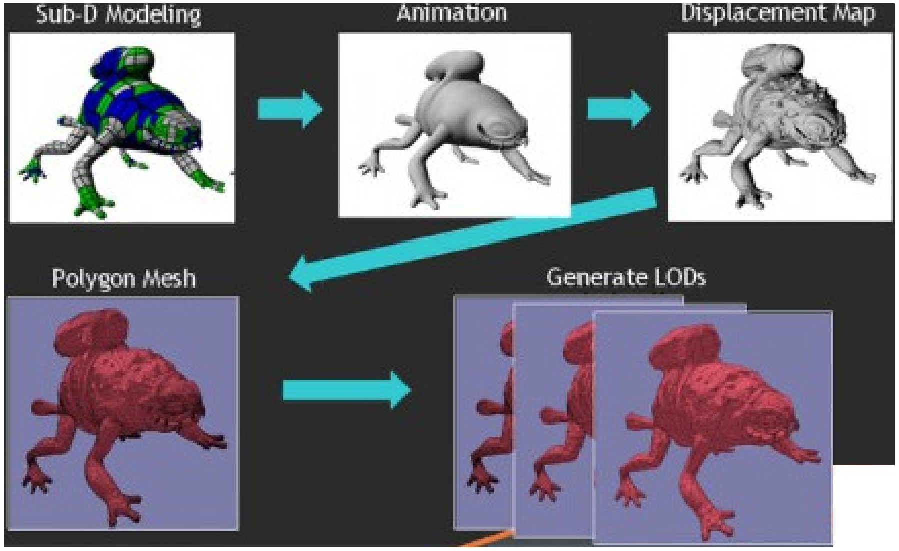
Tessellated and Displaced Mesh

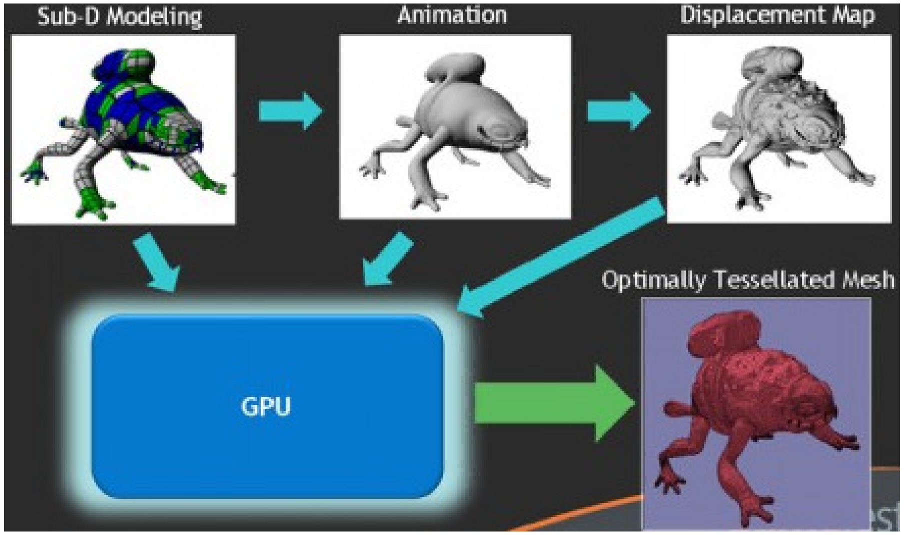
Displacement Map

Images courtesy of ATI

Applications

- Creates a new authoring pipeline
  - 1-pass process from input to optimally tessellated mesh
  - Makes both animation and real-time applications faster
  - Allows for a higher level of detail





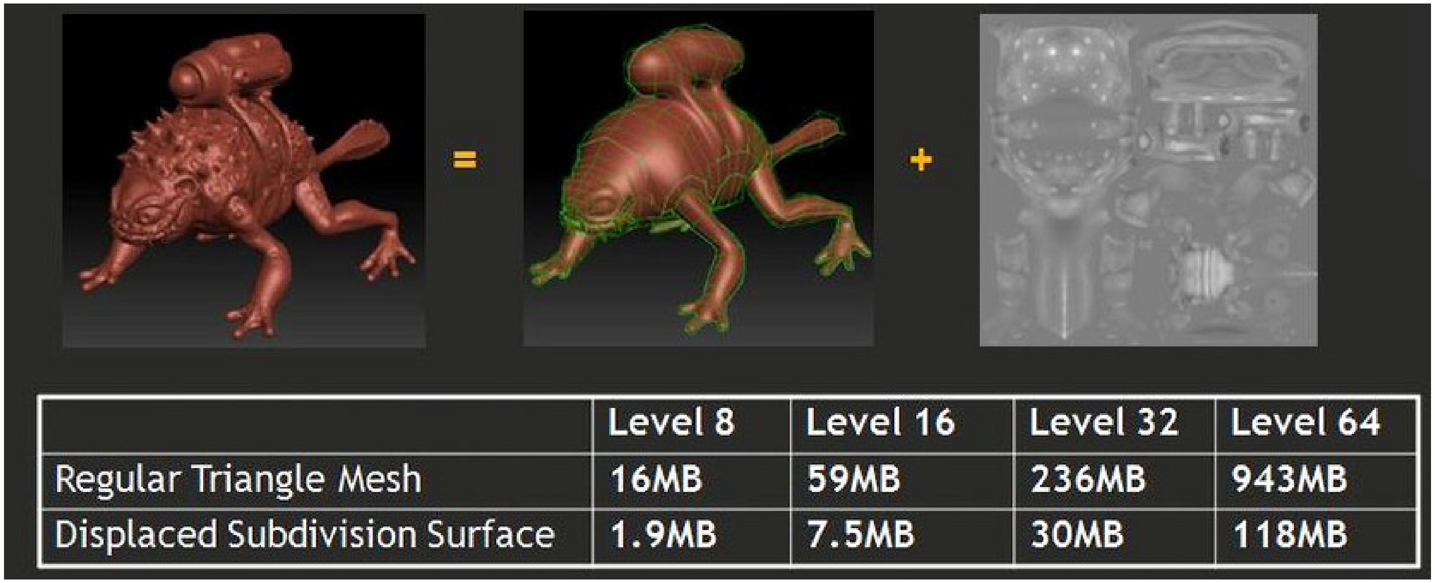
Images courtesy of NVIDIA Gamefest 2008

Tessellation of Displaced Subdivision Surfaces in DX11

Ignacio Castano

Introduction

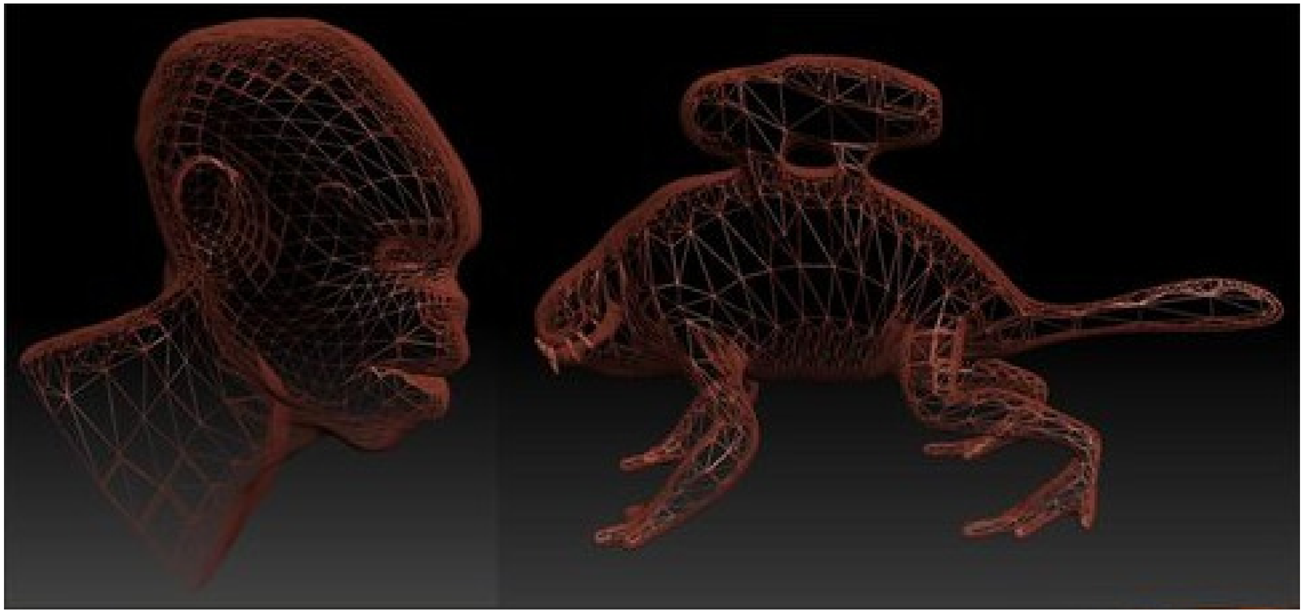
- Tessellation lets us send down low polygon meshes to save memory and bandwidth which are the main bottlenecks



	Level 8	Level 16	Level 32	Level 64
Regular Triangle Mesh	16MB	59MB	236MB	943MB
Displaced Subdivision Surface	1.9MB	7.5MB	30MB	118MB

Scalability

- Tessellation allows for view dependent levels of detail so the GPU doesn't waste time rendering unseen triangles.

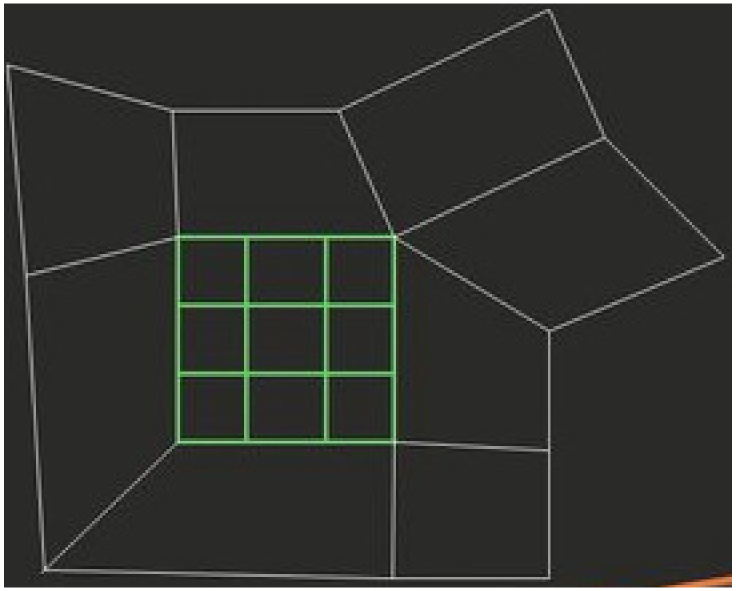


Subdivision Surfaces

- Subdivision Surfaces are a well explored technique for increasing the detail of a mesh.
- Previous GPU implementations required multiple GPU passes
- The new DirectX 11 tessellation hardware allows us to do subdivision surfaces in a single pass

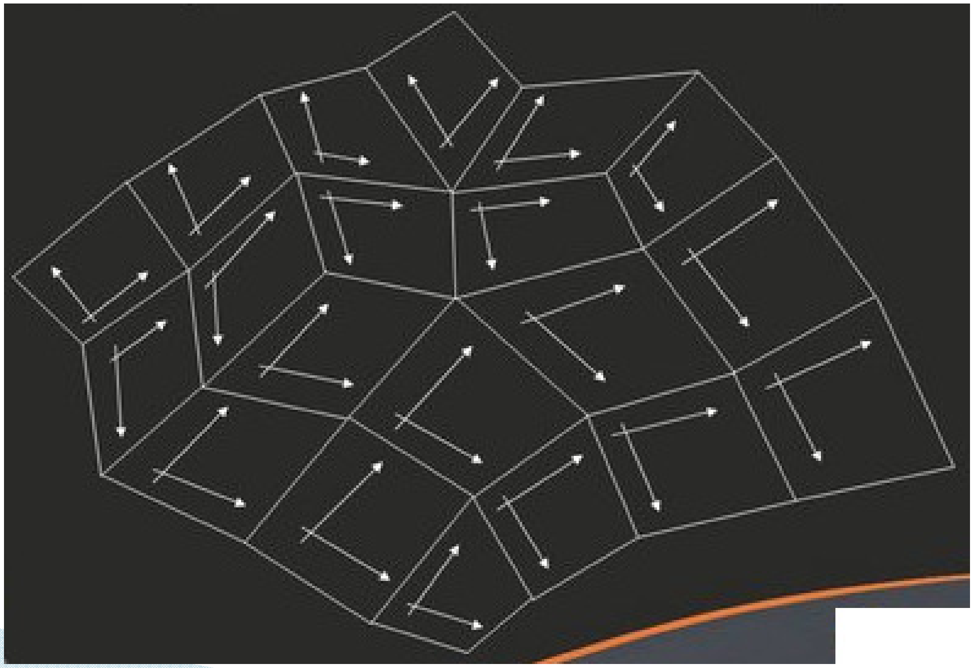
Hull Shader

- The hull shader is used for control point evaluation.
- Input is a face and its neighboorhood
- Output is a regular bicubic bezier patch of the face

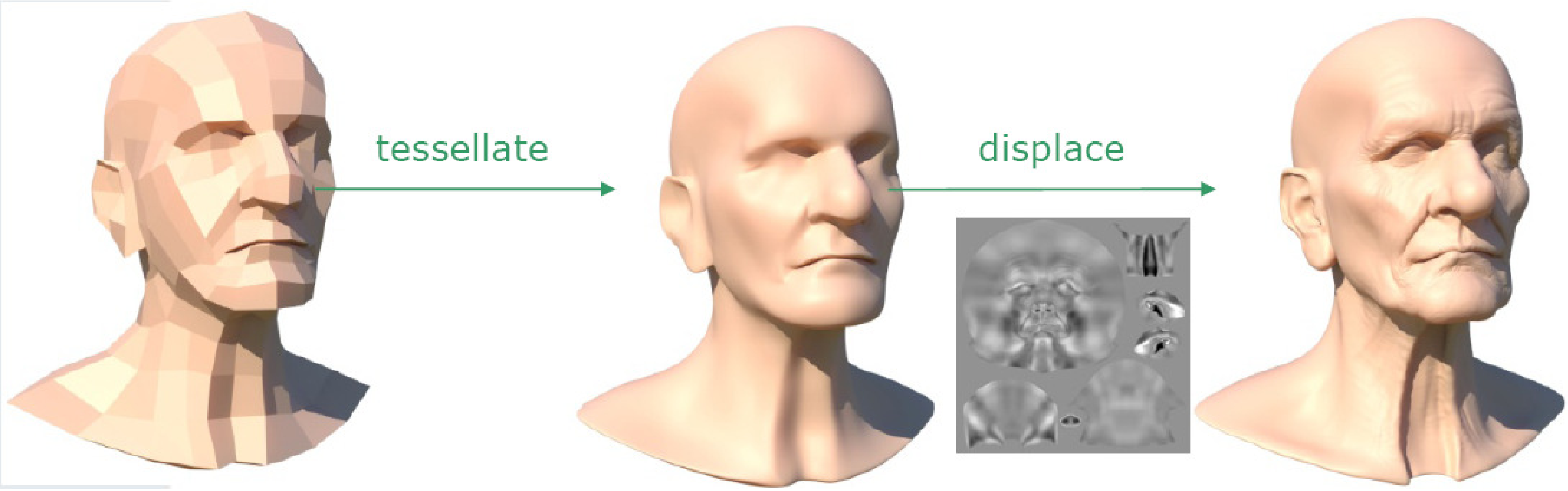


Domain Shader

- The Domain Shader evaluates the bicubic Bezier patches and corresponding tangents
- Reorders face patches for consistent adjacency
- Requires 69 instructions



Results



References

- TATARCHUK N.: Dynamic Terrain Rendering on GPU Using Real-Time Tessellation. *ShaderX*^{7} (Dec. 2008).
- Tatarinov, A.: Instanced Tessellation in DirectX10. GDC 2008. February 2008.
- Gee, K.: Introduction to the Direct3D 11 Graphics Pipeline. Nvision 2008.
- Castano, I.: Tessellation of Displaced Subdivision Surfaces in DX11. Gamefest 2008