DX11 Tessellation

Tianyun Ni

NVIDIA

Tessellation Agenda

- Motivation
- How it works
- Tessellation Schemes
- Watertight, LODs, perf
- Examples
  - Metro2033
  - Terrain Fractal detail
- DS shading

Geometric Realism in Film

- ***Geometric complexity is key to realism***
- **Pixels are meticulously shaded**
- **Geometric detail is substantial**
- **Enables richer content and animation**

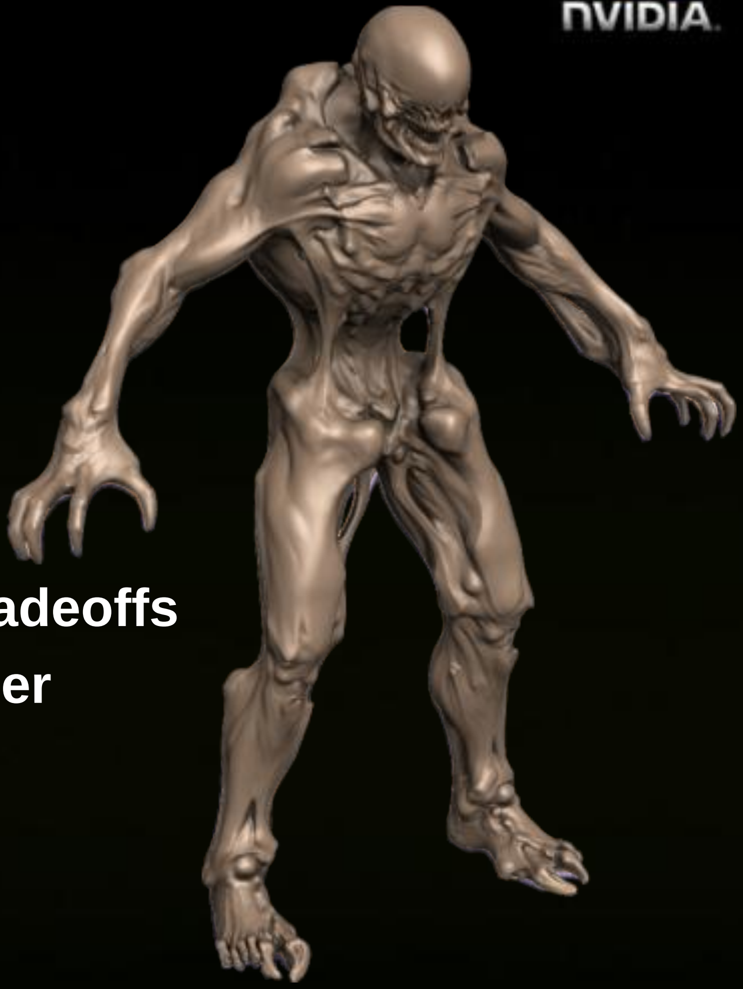
The Problem of Geometric Realism in Games

- Pixels are meticulously shaded,
  but geometric detail is modest

Image from Far Cry® 2,
courtesy of Ubisoft

Tessellation – What and Why

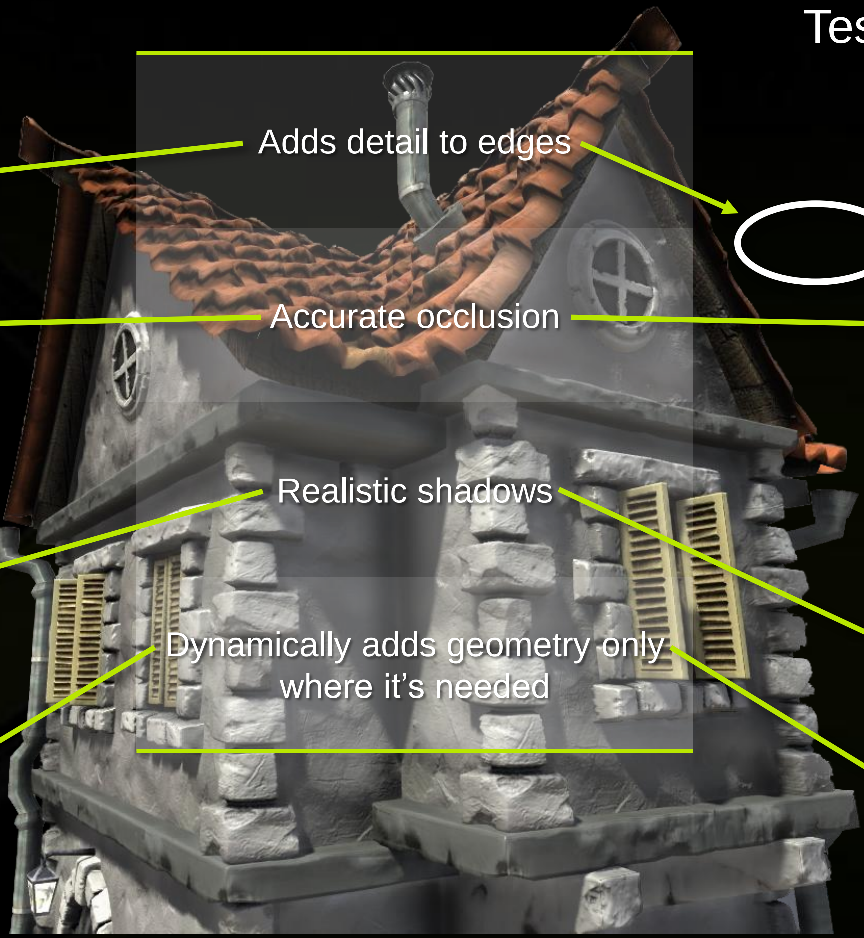
- Memory footprint & BW savings
  - Store coarse geometry, expand on-demand
  - Enables more complex animations
- Scalability
  - Dynamic LOD allows for performance/quality tradeoffs
  - Scale into the future – resolution, compute power
- Computational efficiency
  - Dynamic LOD
  - GPU animate and expand compact representation



Tessellation Adds Rich Detail to Games

Tessellation OFF ✘

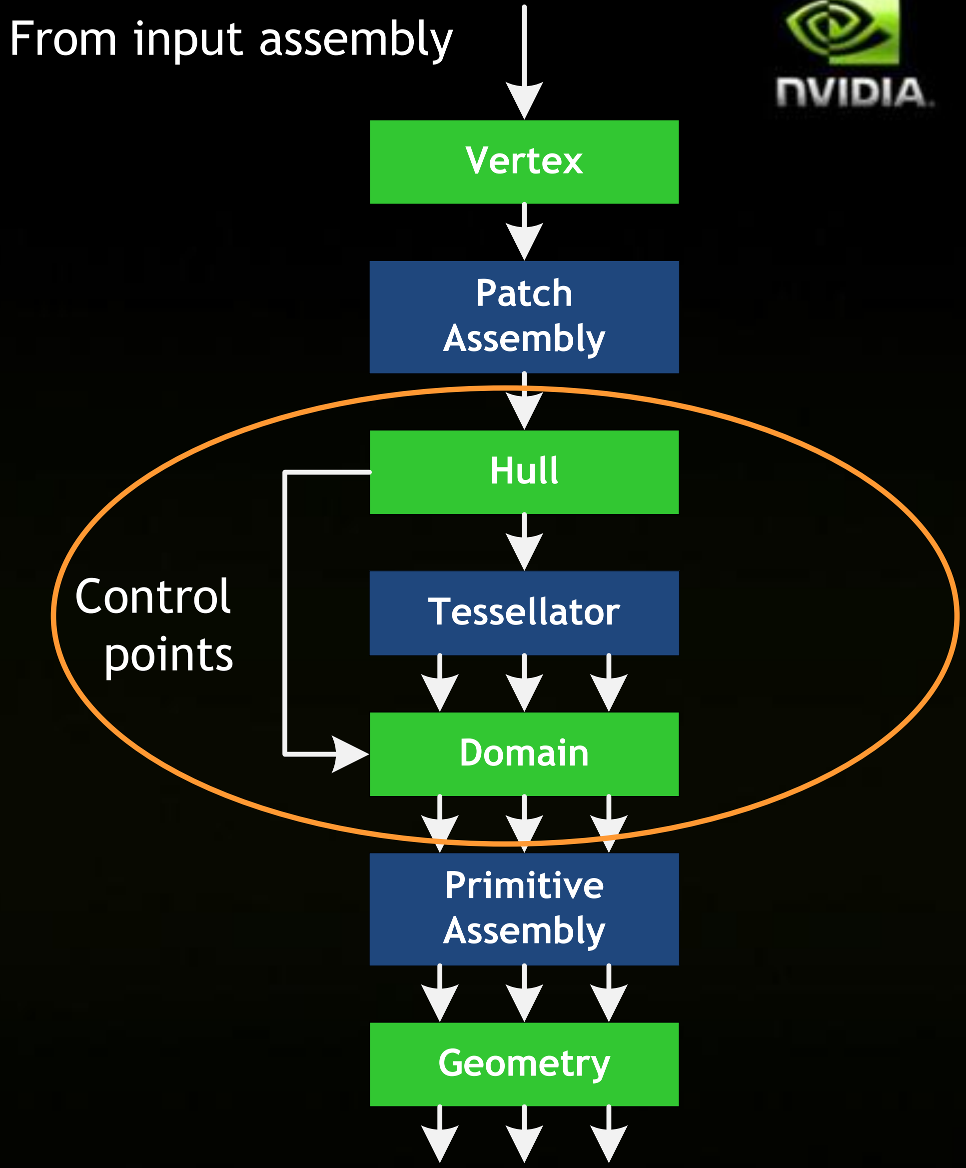
Tessellation ON ✓



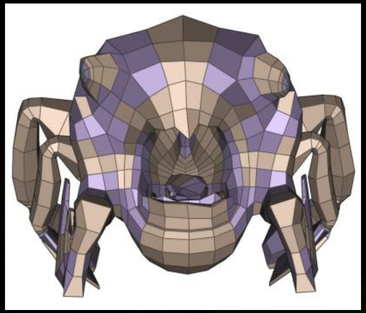
Unigine Heaven Benchmark

Tessellation in DirectX 11

- **Hull shader**
  - Runs pre-expansion
  - Explicitly parallel across control points
- **Fixed function tessellation stage**
  - Configured by LOD output from HS
  - Produces triangles and lines
  - Expansion happens here
- **Domain shader**
  - Runs post-expansion
  - Maps (u,v) to (x,y,z,w)
  - Implicitly parallel

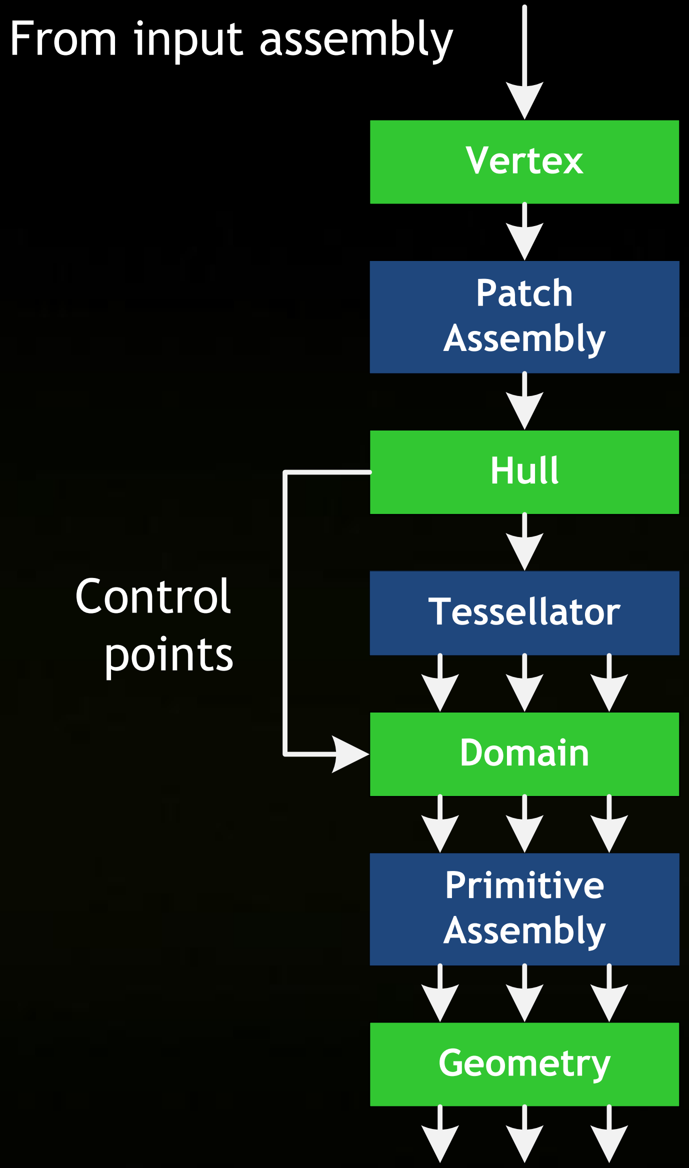


Life of a patch

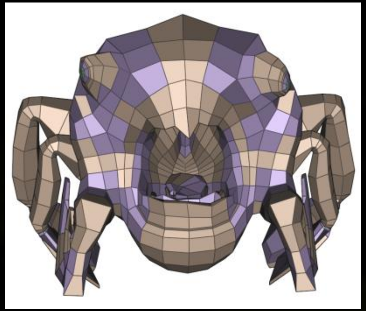


Input Mesh
(a collection of patch primitives)
Displacement Map
Normal Map (optional)

- Patch
  - Represent the a face and its 1-ring.
  - Only primitive type that is supported when tessellation stages are enabled.
  - Arbitrary number of vertices between 1 and 32
  - No implied topology.



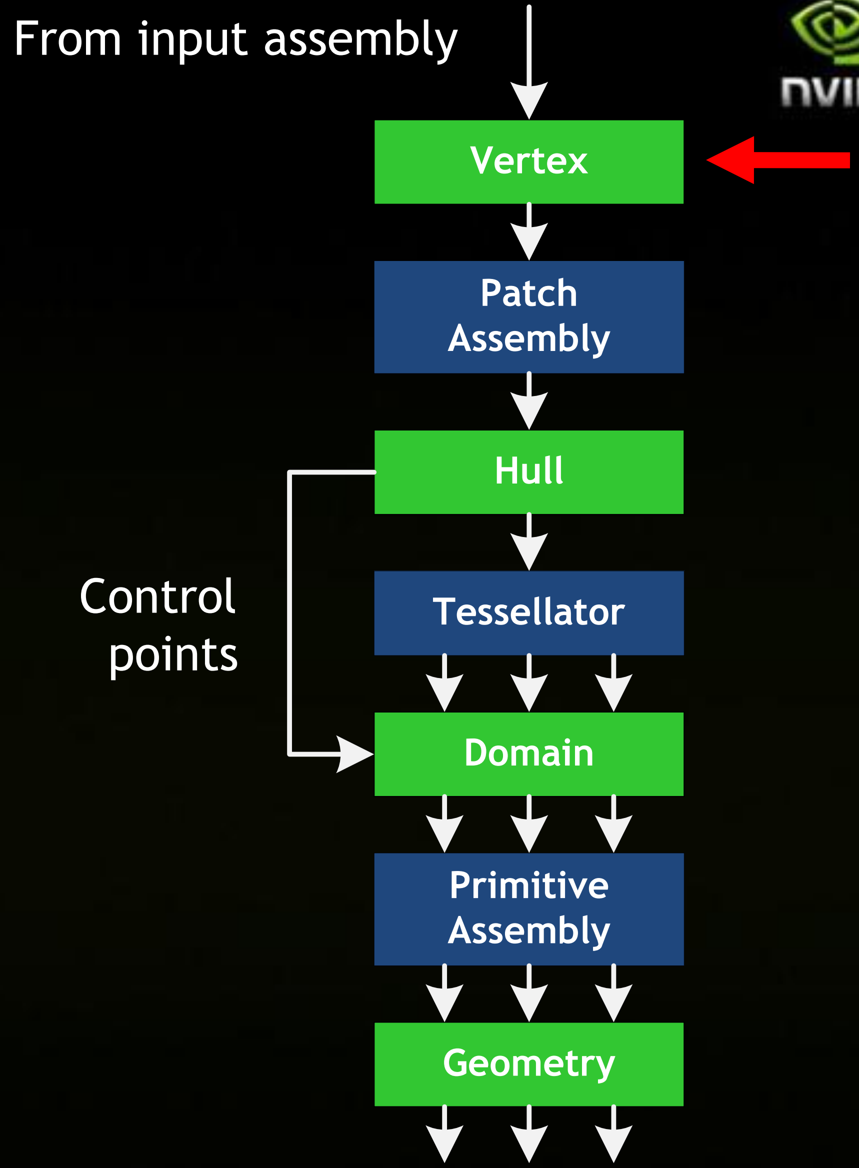
Life of a patch



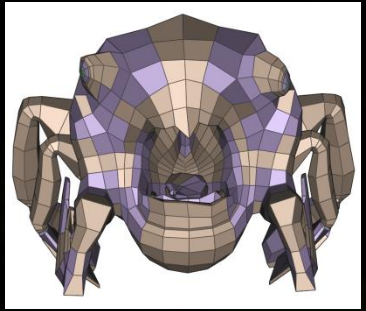
Input Mesh
(a collection of patch primitives)
Displacement Map
Normal Map (optional)

Skinning,…

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

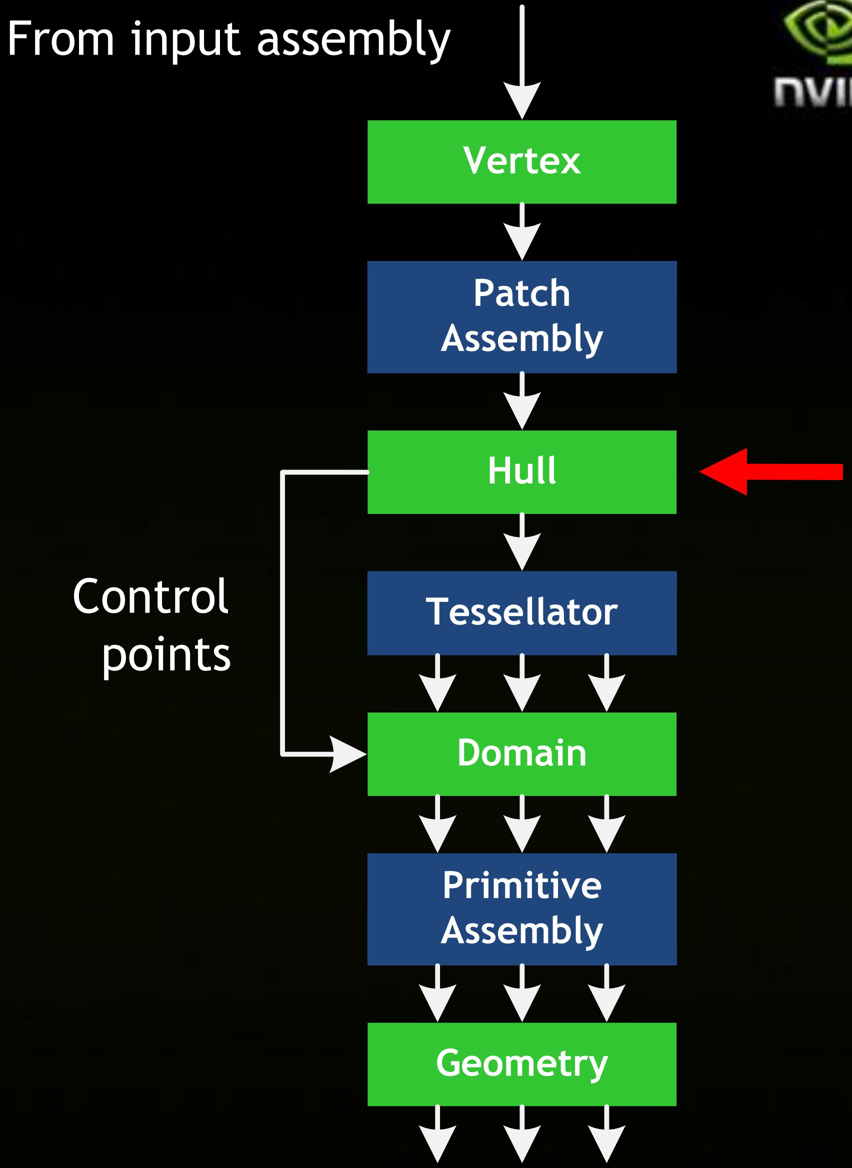


Life of a patch



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

- **Hull Shader**
  - **Control Point Phase (optional)**
    - **Compute Control points (optional)**
    - **Explicitly parallel**
  - **Constant Phase**
    - **Compute LODs**
    - **Compute per patch information**
    - **Pseudo parallel (fxc dependent)**



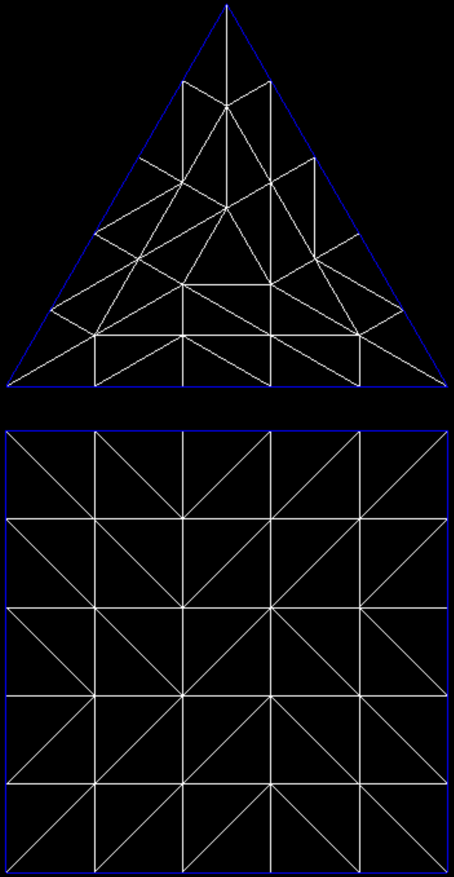
Life of a patch

- Tessellator
  - Where expansion happens
- Let lod be the TessFactor at each edge and interior
- Number of triangles on a triangle domain

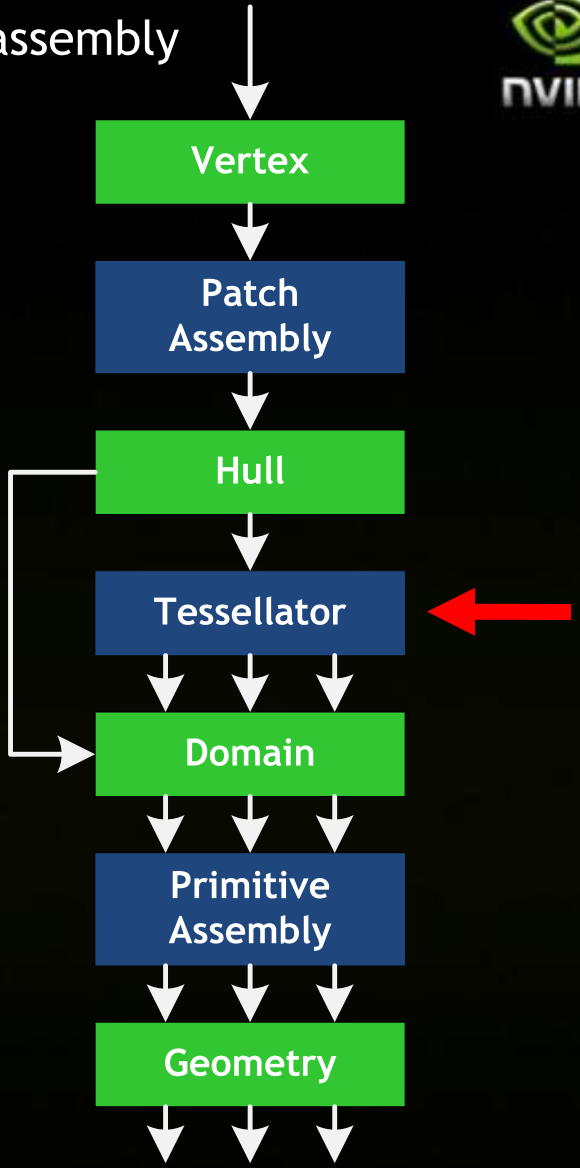
  $1+6*\sum_{i=1}^{lod/2}(2*i)$, If lod is odd

  $6*\sum_{i=1}^{lod/2}(2*i-1)$, If lod is even
- Number of triangles on a quad domain

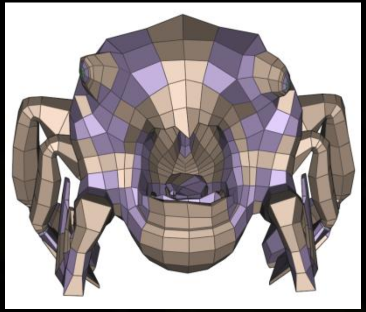
  2*lod*lod



From input assembly



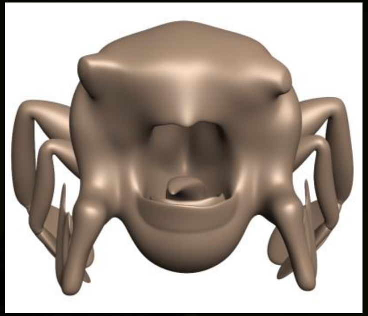
Life of a patch



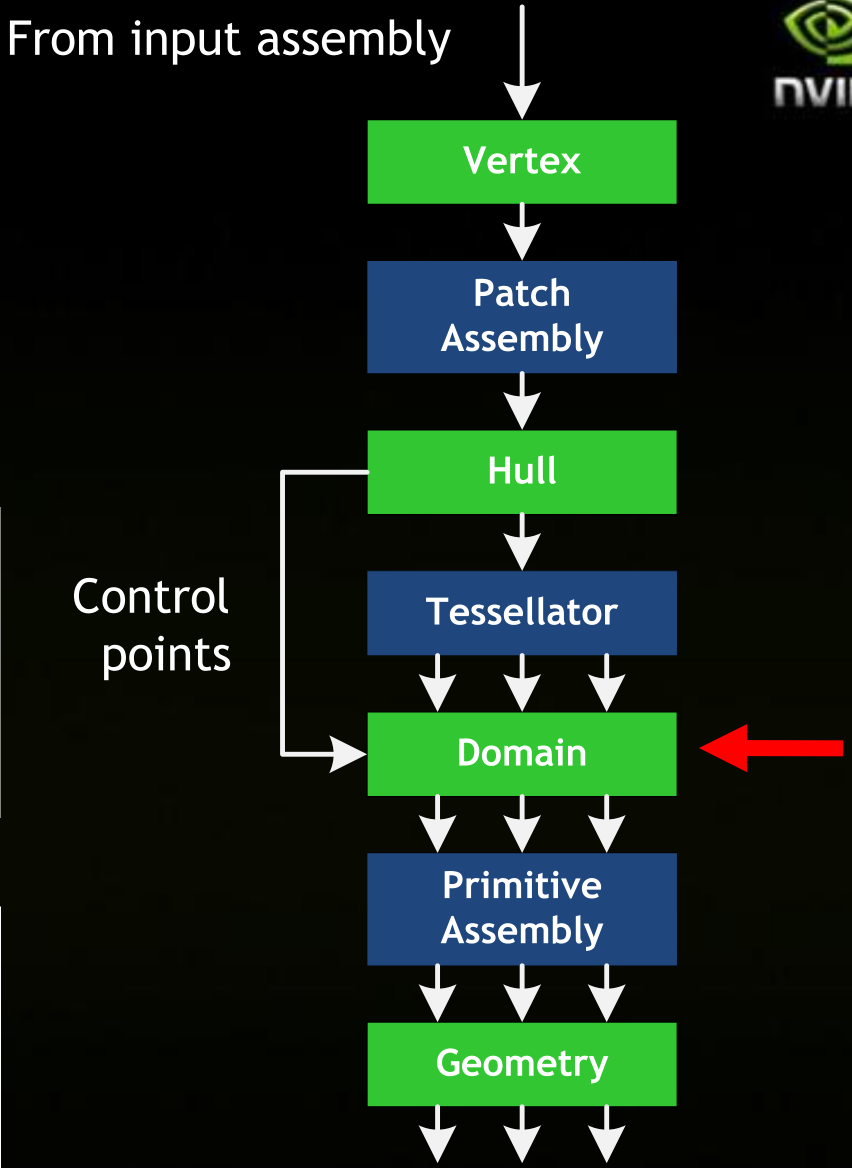
Input Mesh
(a collection of patch primitives)
Displacement Map
Normal Map (optional)

- **Domain Shader**
  - **Surface Evaluation**
  - **Displacement mapping**
  - **Implicitly parallel (on thread per vertex)**
  - **Vertex shader tasks**
    - **Vertex projection**
    - **Normal transformation**
    - **…**

Patch Surface



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

Linear/Flat

Phong Tessellation

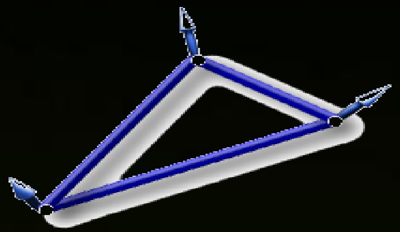
PN Triangles

Catmull-Clark (approximated)

Local Construction Schemes

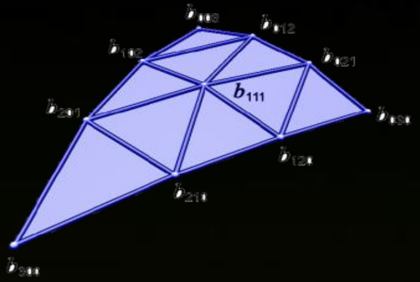
PN Triangles

- **Key features:**
  - **Cubic Bezier patches**
  - **Quadratic normal variation**
- **Easy to implement**
- **Hard edges not handled**

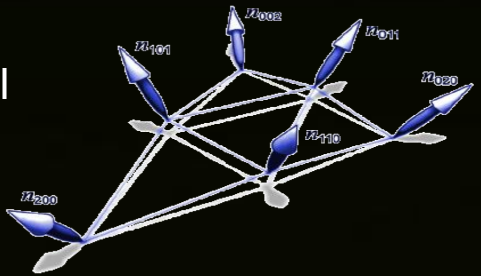


One input triangle

Geometry patch



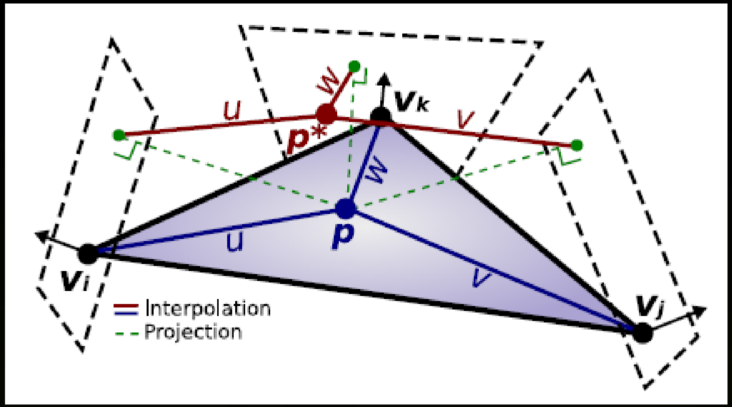
Normal patch



- **“Curved PN Triangles”, by Alex Vlachos, Jörg Peters, Chas Boyd, and Jason Mitchell, I3D 2001.**
- **“PN Quads”, by Jörg Peters, 2008.**

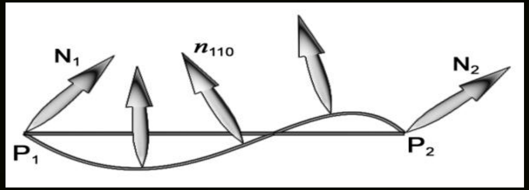
Phong Tessellation

- Key features:
  - Quadratic geometry interpolation
  - Linear normal variation (phong shading)



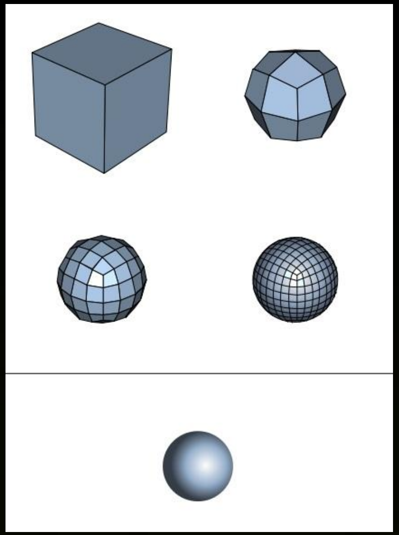
- Simpler than PN Triangles

- Can not handle inflection points
  - Needs a relatively dense mesh to start with

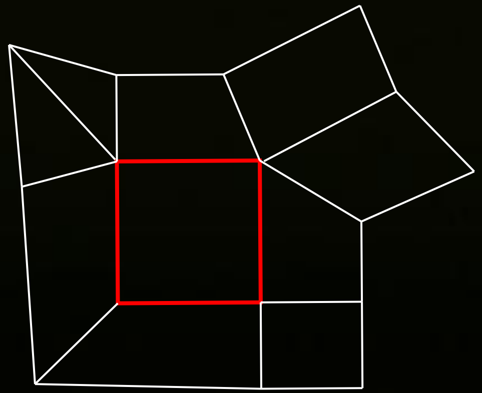


Paper Siggraph 2008 Asia, by Tamy Boubekeur & Marc Alexa

Catmull-Clark Subdivision Surfaces



- 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
- Approximation (ACC)
  - Approximation rather than interpolation
  - Requires the mesh info of a facet and its1-ring neighborhood



ACC references

- 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.

Tessellation Schemes Comparison

	# of vertices in a patch primitive	# of control points	Base mesh	Surface fairness
Phong	3 Or 4	6 Or 9	△ □	Artifacts at inflection points and high curvature area
PN	3 Or 4	10+6 Or 16+9	△ □	Artifacts at high curvature area
Gregory ACC	16 to 32	15 Or 20	△ □	Similar to CC surfaces

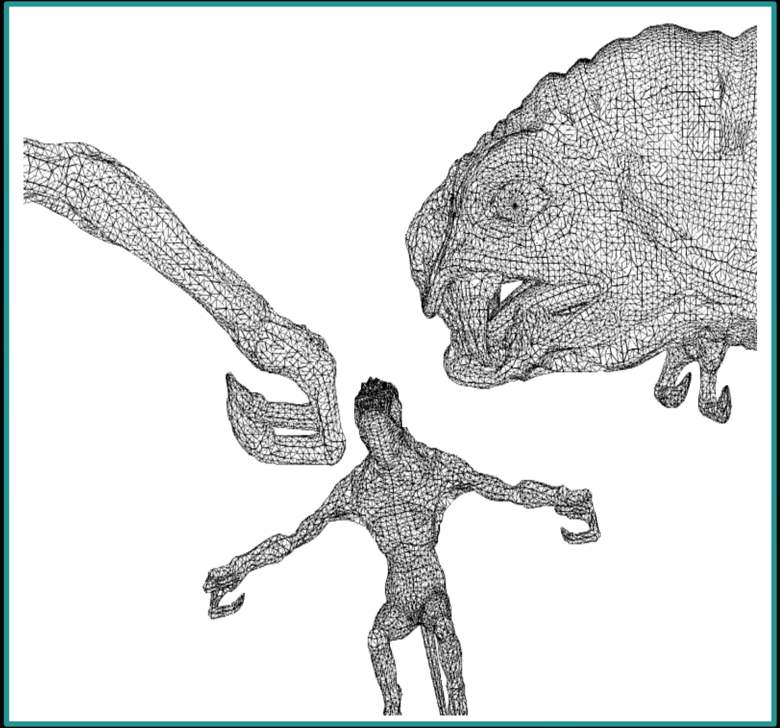
Choose appropriate schemes for your art assets.
Tradeoff between performance and visual quality

Water Tightness

- Control Points cracks
  - Problem: floating point precision issues a+b+c != c+b+a
  - Require consistent evaluations at corners and edges
- Displacement cracks
  - Problem: Bilinear discontinuities
  - Define patch ownership of the texture coordinates
- Normals
  - Problem: cross(tanU,tanV) ≠cross(tanV, tanU)
  - Discontinuities occur at shared corners and edges

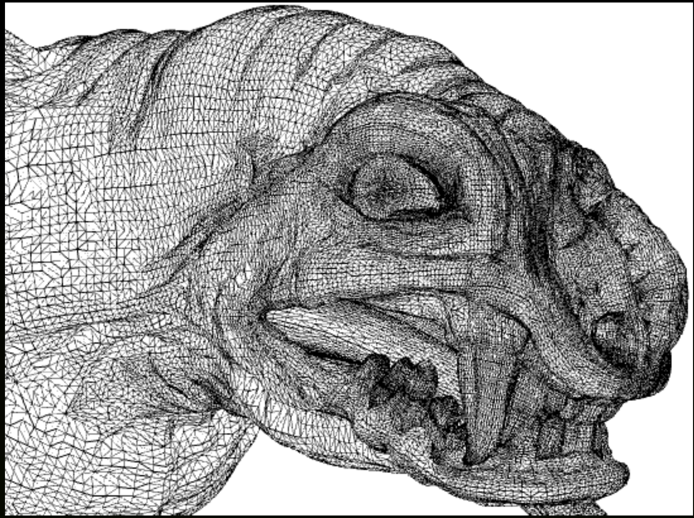
LOD computation

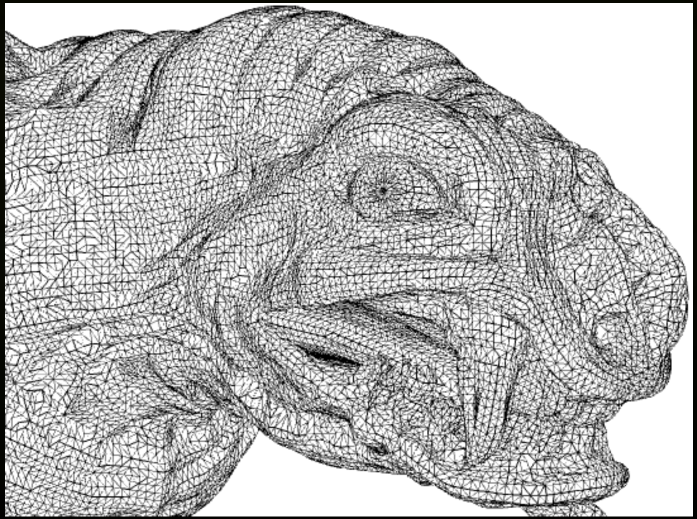
- LOD heuristics
  - Object to camera distance
  - Screen resolution
  - Silhouette
  - Displacement density
- Performance/quality balance control
- Smooth LOD transitions



Screen Space LOD Computation

- Generate exactly the amount of geometry needed for a given view
  - No under/oversampling
- Uniform sampling of the surface improves shading
- Triangles of roughly the same size => hw efficiency
  - 4-8 pixels/tri on screen for high end





Optimization tips

- Per-object culling (based on bounding box)
  - Frustum
  - Occlusion
- Per-patch culling (in the hull shader, based on tight-bound Displaced Bezier patches)
  - Frustum, Backface, Occlusion (?)
  - Set tessellation factor to 0
- Do not use tessellation factor = 1

Metro 2033: Tessellation in characters

Displacement mapping enables film-level geometric complexity in real-time

Screenshots from Metro 2033

Metro 2033 tessellation

- Reuse of DX9/DX10 assets
  - Phong Tessellation + Displacement maps
- LOD criteria:

  TESS_FACT = LEN * NP * Q / DIST

  - Where LEN is edge length in world space
  - NP is number of pixels on the screen
  - Q is quality constant
  - DIST is distance from observer to edge center

Metro 2033: Tessellation in characters

Metro 2033: Artifacts on Hard Edges

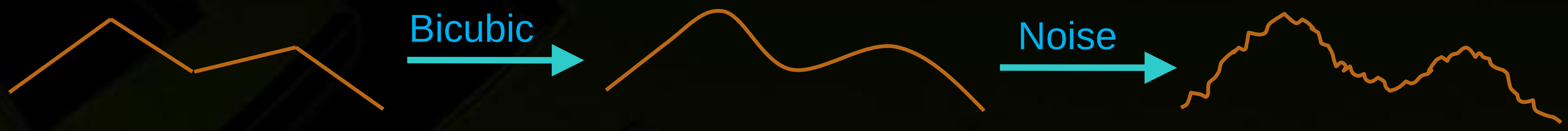
Transitional Polygons

Terrain Tessellation

- Flat quads; regular grid
- Height map; vertical displacement; sample in DS
- Challenges:
  - Existing data from DX9/DX10
  - A wide range of LODs

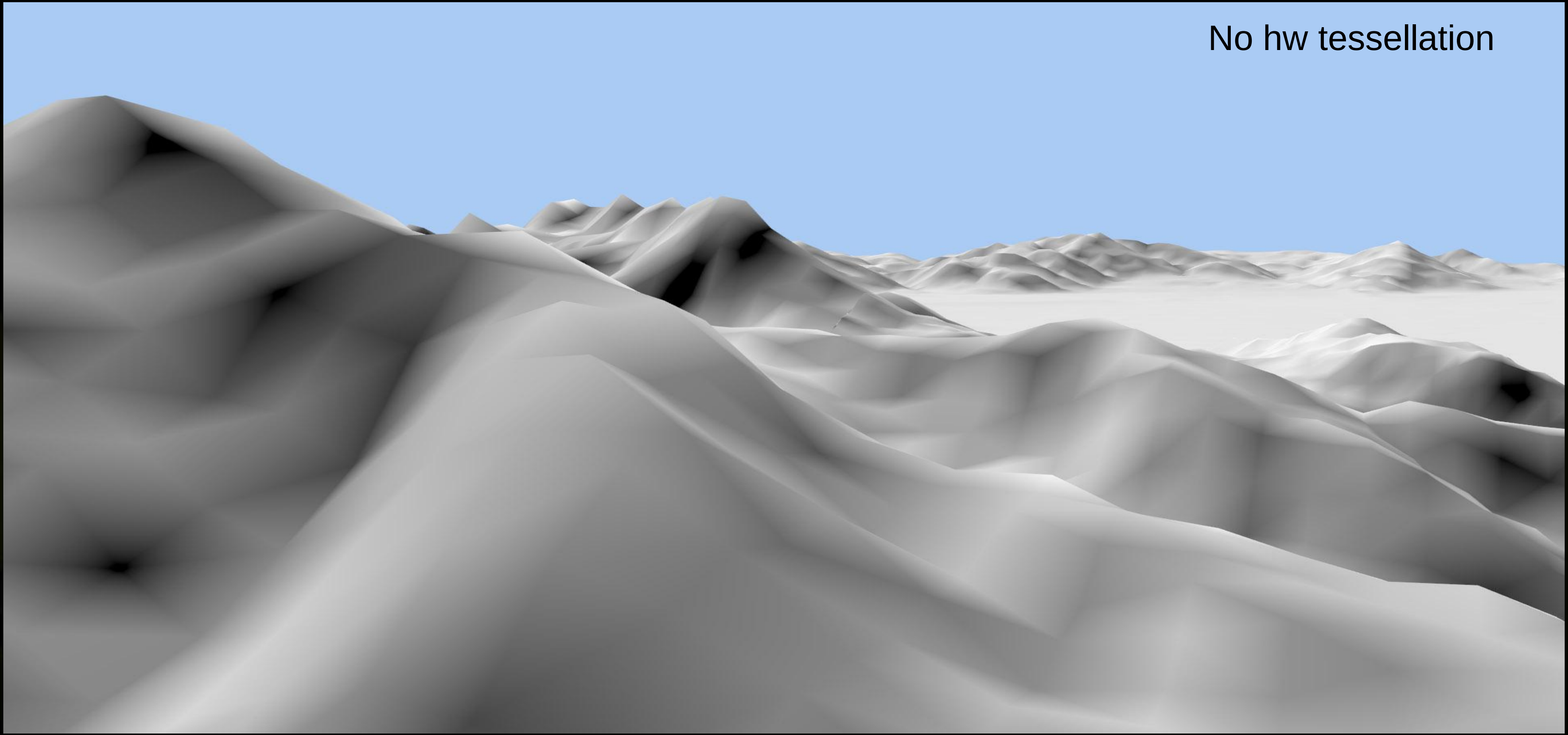
Data Solution: Fractal "Amplification"

- Coarse height map defines topographic shape
  - Upsample with bicubic
- Fractal detail map adds high-LOD detail (fBm)

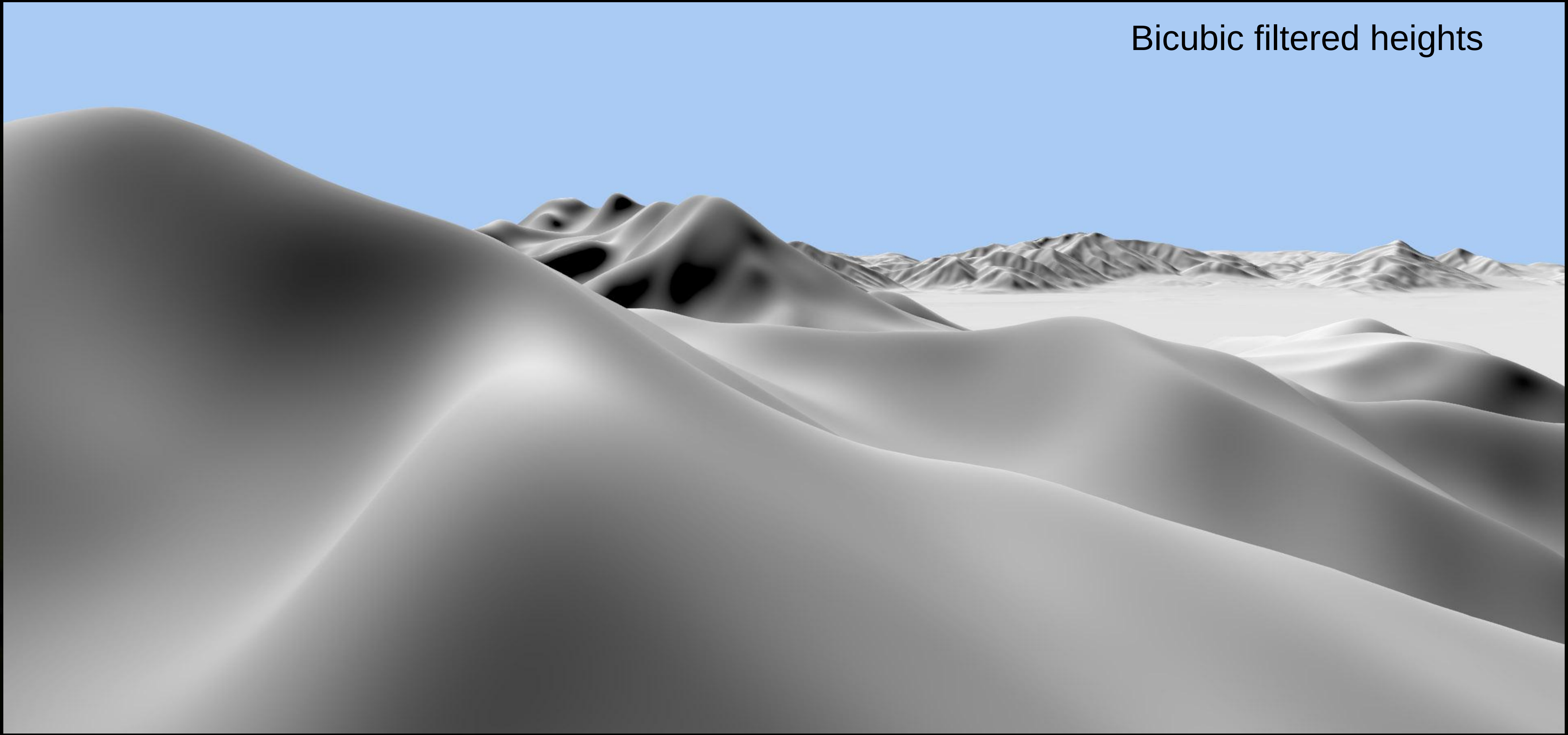


- Cheap memory requirements
- Can reuse coarse assets from DX9 or DX10 engine

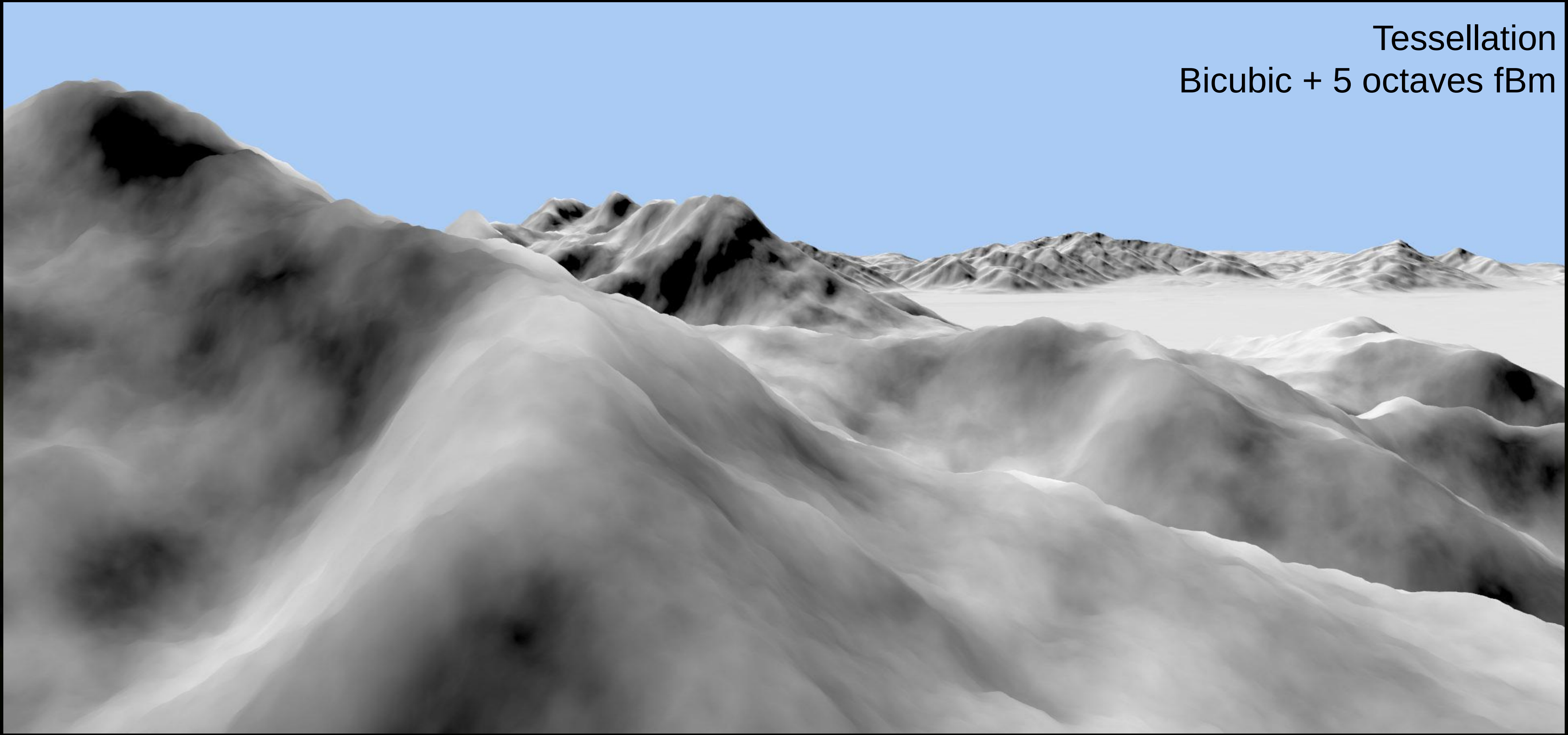
Fractal “Amplification” - Results



Fractal “Amplification” - Results

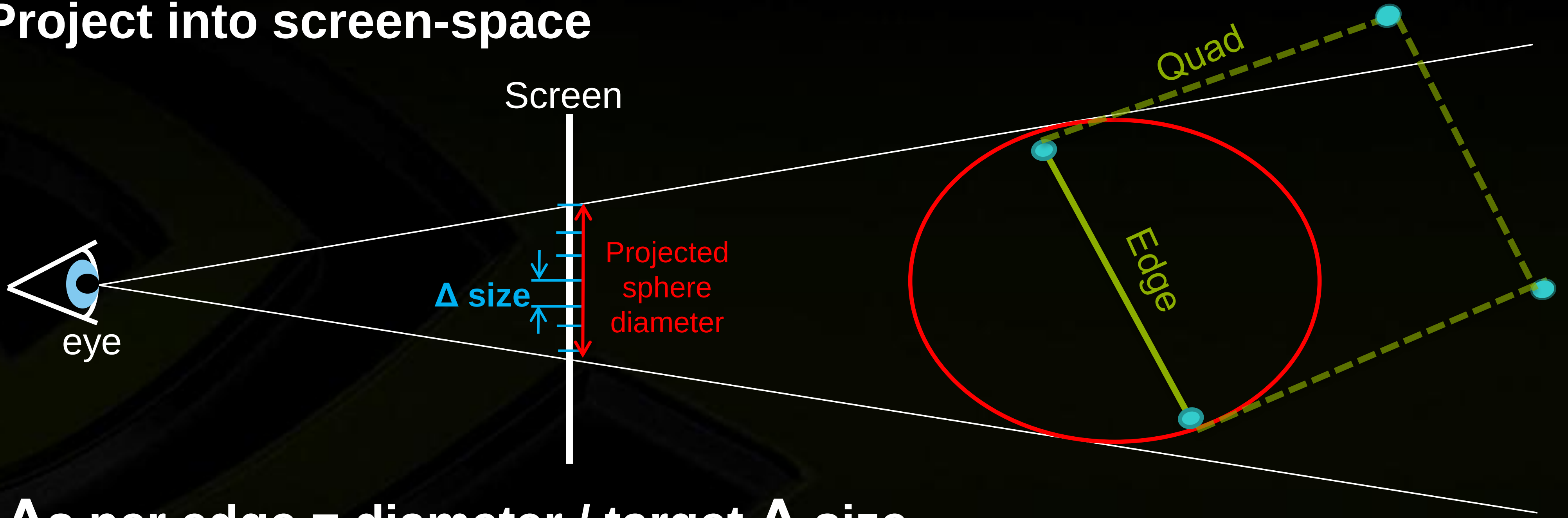


Fractal “Amplification” - Results



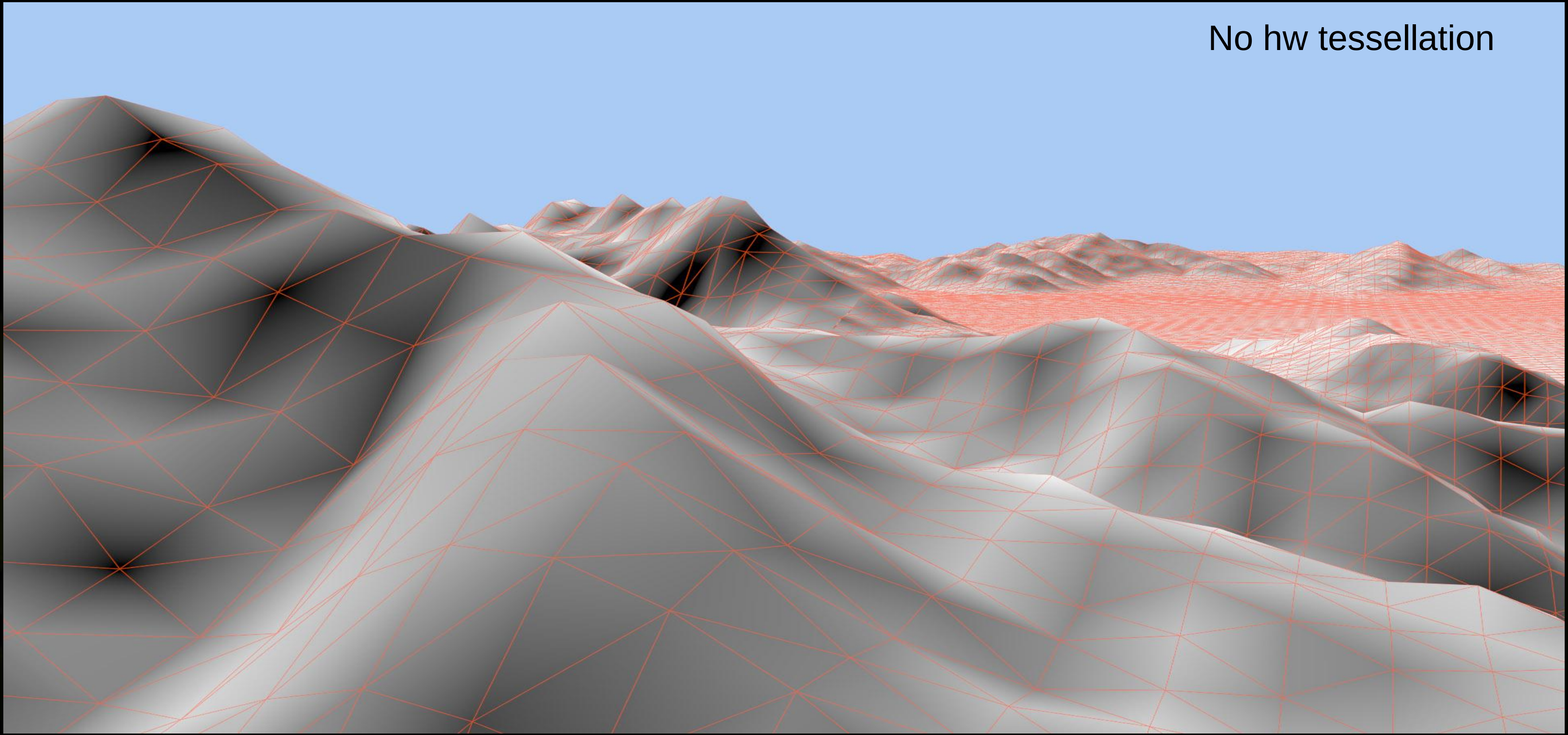
Screen-space-based LOD (hull shader)

- **Enclose quad patch edge in bounding sphere**
- **Project into screen-space**

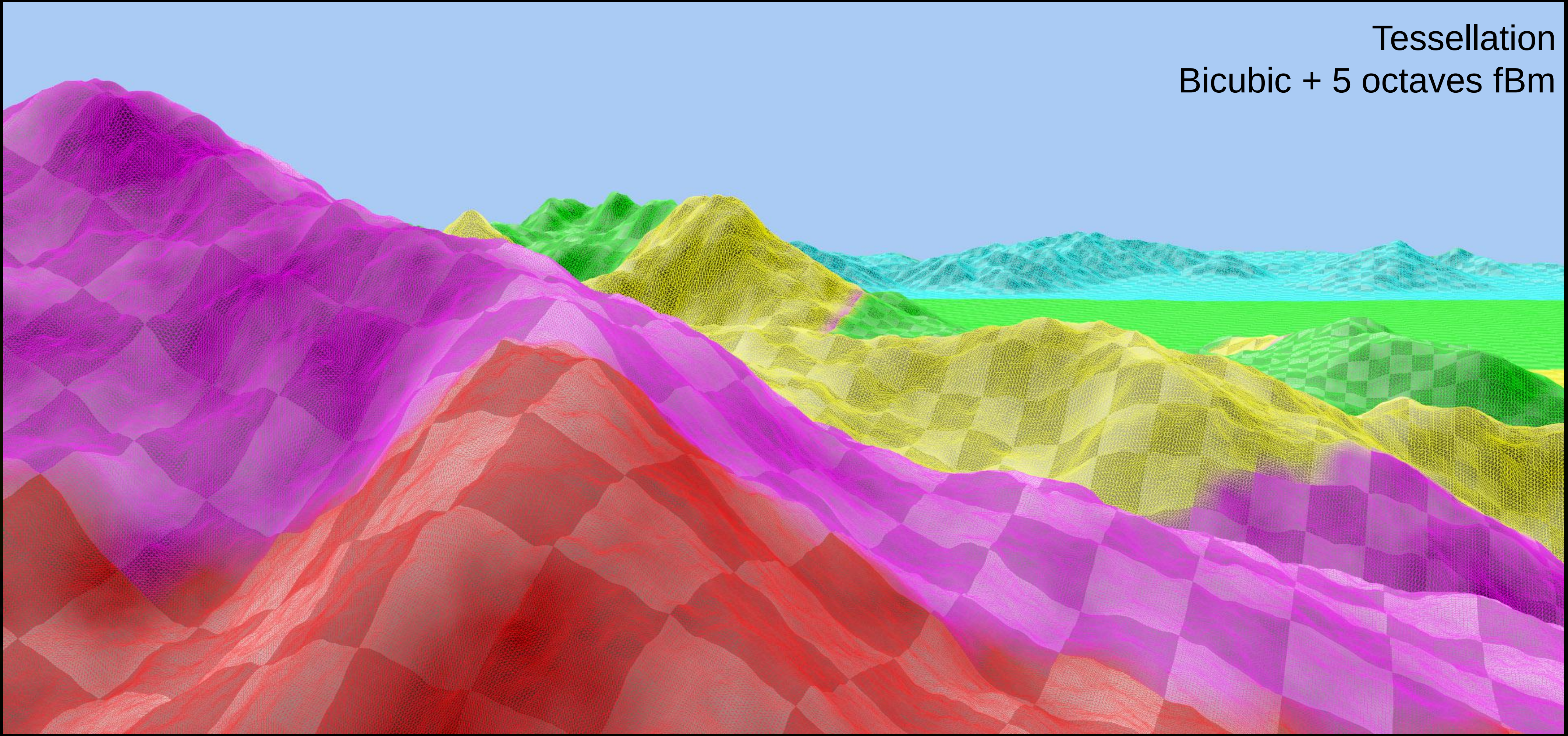


- **Δs per edge = diameter / target Δ size**
- **(diameter & target size in pixels)**
- **Fully independent of patch size**

Fractal “Amplification” - Results



Fractal “Amplification” - Results



HAWX 2 Results

HAWX 2 Results

HAWX 2 Results

HAWX 2 Results

HAWX 2 Results

HAWX 2 Results

Shading at different frequencies

- Hoist lower-frequency computation from PS to DS
  - E.g. ambient/volumetric lighting
- Shading in object space sometimes better
  - More uniform surface sampling
  - Less aliasing under animation
- In general, compute complex things as early in the pipeline as possible
  - VS possible? … HS possible? … DS possible? … If not, then PS
  - Try to minimize number of attributes coming to PS stage

Shading in the Domain Shader

Conclusions on Tessellation

- Direct3D11 Tessellation enables visual detail
  - Several tessellation schemes with flexible LOD control
  - Changes to content creation pipeline
  - Tessellation HW is very powerful, but still need to use it wisely
- It is possible to re-use DX9/DX10 content with no extra work
  - Local schemes, Fractal noise functions,…
- Tessellation is not only about visual detail, but faster shading
  - Shading in DS

It's time to bring games to the next level!!!

Thanks

- To all the people I borrowed material from: Miguel Sainz, Kirill Dmitriev, Yury Uralsky, Iain Cantlay, Jon Jansen, Cem Cebenoyan, Sarah Tariq, Tim Tcheblokov, Evgeny Makarov, THQ, 4A Games, Ubisoft, ID, Pixar, Disney…

- For questions/comments please contact us:

  - tni@nvidia.com

Other use cases: Grass

Other use cases: Hair

Content Creation Pipeline

- Modeling Tools
  - Base surface
  - (control cage)
- Sculpting Tools
  - Detailed mesh
- Baker Tools
  - Normal, displacement, occlusion, and other maps…

Some baker tools can be automated… talk to us!