Why do we need a Tessellation step right in the pipeline?
• You can perform adaptive subdivision based on a variety of criteria (size, curvature, etc.)
• You can provide coarser models, but have finer ones displayed (= geometric compression)
• You can apply detailed displacement maps without supplying equally detailed geometry
• You can adapt visual quality to the required level of detail
• You can create smoother silhouettes
• You can do all of this, and someone else will supply the patterns!

What built-in patterns can the Tessellation shaders produce?
- Lines
- Triangles
- Quads (subsequently broken into triangles)

Tessellation Shader Organization
The Tessellation Control Shader (TCS) transforms the input coordinates to a regular surface representation. It also computes the required tessellation level based on distance to the eye, screen space spanning, hull curvature, or displacement roughness. There is one invocation per output vertex.

The Fixed-Function Tessellation Primitive Generator (TPG) generates semi-regular $u$-$v$-$w$ coordinates in specific patterns. (In fact, if it had been up to me, this would have been called the Tessellation Pattern Generator.)

The Tessellation Evaluation Shader (TES) evaluates the surface in $uvw$ coordinates. It interpolates attributes and applies displacements. There is one invocation per generated vertex.

There is a new "Patch" primitive – it is the face and its neighborhood:
```
gBegin( GL_PATCHES );
gVertex3f( … );
gVertex3f( … );
gEnd( );
```

In the OpenGL Program
```
gBegin( GL_PATCHES );
gVertex3f( … );
gVertex3f( … );
gEnd( );
```
These have no implied topology – they will be given to you in an array. It’s up to your shader to interpret the order.
```
gPatchParameteri( GL_PATCH_VERTICES, num );  // # of vertices in each patch
GLuint tcs = glCreateShader( GL_TESS_CONTROL_SHADER );
GLuint tes = glCreateShader( GL_TESS_EVALUATION_SHADER );
```

Check the OpenGL extension:
```
"GL_ARB_tessellation_shader"
```
TCS Inputs

`gl_in[ ]` is an array of structures:

```plaintext
struct {
  vec4 gl_Position;
  float gl_PointSize;
  float gl_ClipDistance[6];
} gl_in[ ];
```

`gl_InInvocationID` tells you which output vertex you are working on. This must be the index into the `gl_in[ ]` array.

`gl_PatchVerticesIn` is the number of vertices in each patch and the dimension of `gl_in[ ]`

`gl_PrimitiveID` is the number of primitives since the last glBegin() (the first one is 0)

TCS Outputs

`gl_out[ ]` is an array of structures:

```plaintext
struct {
  vec4 gl_Position;
  float gl_PointSize;
  float gl_ClipDistance[6];
} gl_out[ ];
```

All invocations of the TCS have read-only access to all the output information.

```plaintext
layout( vertices = n ) out;
```

Used to specify the number of vertices output to the TPG

```plaintext
gl_TessLevelOuter[4]` is an array containing up to 4 edges of tessellation levels

`gl_TessLevelInner[2]` is an array containing up to 2 edges of tessellation levels

In the TCS

User-defined variables defined per-vertex are qualified as “out”

User-defined variables defined per-patch are qualified as “patch out”

Defining how many vertices this patch will output:

```plaintext
layout( vertices = 16 ) out;
```

Tessellation Primitive Generator

Is “fixed-function”, i.e., you can’t change its operation except by setting parameters

Consumes all vertices from the TCS and emits tessellated triangles, quads, or lines

Outputs positions as coordinates in barycentric (u,v,w)

All three coordinates (u,v,w) are used for triangles

Just (u,v) are used for quads and isolines

TES Inputs

Reads one vertex of 0 <= (u,v,w) <= 1 coordinates in variable vec3 `gl_TessCoord`

User-defined variables defined per-vertex are qualified as “out”

User-defined variables defined per-patch are qualified as “patch out”

```plaintext
`gl_in[ ]` is an array of structures coming from the TCS:

struct {
  vec4 gl_Position;
  float gl_PointSize;
  float gl_ClipDistance[6];
} gl_in[ ];
```

Tessellation Primitive Generator (TPG)

- Is “fixed-function”, i.e., you can’t change its operation except by setting parameters
- Consumes all vertices from the TCS and emits vertices for the triangles, quads, or lines patterns
- TPG outputs a series of vertices as coordinates in barycentric (u,v,w) parametric space
- All three coordinates (u,v,w) are used for triangles
- Just (u,v) are used for quads and isolines

```plaintext
triangle pattern
```
TES Output Topologies: the Quad Pattern

(u=0,v=0) (u=1,v=0) (u=1,v=1) (u=0,v=1)

OL1
OL0
OL2
OL3

IL0
IL1

u
v

TES Output Topologies: the Isolines Pattern

OL0 == 1 implies that you just want to draw a single curve

OL0 (Top line not drawn)

TES Output Topologies: the Triangle Pattern

(u=0,v=0,w=1) (u=1,v=0,w=0) (u=0,v=1,w=0)

OL1
IL0

u + v + w = 1

Examples

In these examples:

1. We are using glman to run them. The only necessary input files are the glman .glib file and the shader files. If you aren’t using glman, you can do this from a full OpenGL program.

2. All of the surface examples use the Geometry Shader triangle-shrink shader. This isn’t necessary, but is educational to really see how much and where the surfaces have been tessellated.

Example: A Bézier Curve

P(u) = (1−u)^3P_0 + 3u(1−u)^2P_1 + 3u^2(1−u)P_2 + u^3P_3

1. You program the Tessellation Control Shader to decide how much to tessellate the curve based on screen area, curvature, etc.

Can even tessellate non-uniformly if you want

The OpenGL tessellation can also do 1D curves. Just set OL0 = 1.
2. The Tessellation Primitive Generator generates \( u, v, w \) values for as many subdivisions as the TCS asked for.

3. The Tessellation Evaluation Shader computes the \( x, y, z \) coordinates based on the TPG's \( u \) values

\[
P(u) = (1-u)^3 P_0 + 3u(1-u)^2 P_1 + 3u^2(1-u) P_2 + u^3 P_3
\]

In an OpenGL Program

```glsl
[GL_PATCH_VERTICES, 4];
[GL_PATCHES ];
[glVertex3f, x0, y0, z0];
[glVertex3f, x1, y1, z1];
[glVertex3f, x2, y2, z2];
[glVertex3f, x3, y3, z3];
END( );
```

In a .glib File

```glsl
[OpenGL, GLIB];
[Perspective 70];
Vertex beziercurve.vert
Fragment beziercurve.frag
TessControl beziercurve.tcs
TessEvaluation beziercurve.tes
Program BezierCurve \u003c0 1 5\u003e \u003c3 5 50\u003e
Color 1. 0.5 0. 1.
[NumPatch/vertices 4 ];
[glBegin, gl_patches ];
[glVertex 0. 0. 0.];
[glVertex 1. 1. 1.];
[glVertex 2. 1. 0.];
[glVertex 3. 0. 1.];
gend
```

In the TCS Shader

```glsl
#version 400
getExtension GL_ARB_tessellation_shader: enable
uniform int uOuter0, uOuter1;
layout( vertices = 4 ) out;
void main( ){
  gl_out[gl_InvocationID].gl_Position = gl_in[gl_InvocationID].gl_Position;
  gl_TessLevelOuter[0] = float( uOuter0 );
  gl_TessLevelOuter[1] = float ( uOuter1 );
}
```

In the TES Shader

```glsl
#version 400
[getExtension GL_ARB_tessellation_shader: enable]
layout( isolines, equal_spacing ) in;
void main( ){
  vec4 p0 = gl_in[0].gl_Position;
  vec4 p1 = gl_in[1].gl_Position;
  vec4 p2 = gl_in[2].gl_Position;
  vec4 p3 = gl_in[3].gl_Position;
  float u = gl_TessCoord.x;
  // the basis functions:
  float b0 = (1.-u) * (1.-u) * (1.-u);
  float b1 = 3. * u * (1.-u) * (1.-u);
  float b2 = 3. * u * u * (1.-u);
  float b3 = u * u * u;
  gl_Position = b0*p0 + b1*p1 + b2*p2 + b3*p3;
}
```

Assigning the intermediate \( p_i \)'s is here to make the code more readable. We assume that the compiler will optimize this away.
Example: A Bézier Curve

Outer1 = 5

Outer1 = 50

Example: A Bézier Surface

Bézier Surface Parametric Equations

\[ P(u, v) = \left[ (1-u)^3 \right] 3u(1-u)^2 3u^2(1-u) u^3 \]  
\[ P_{00} P_{10} P_{20} P_{30} P_{01} P_{11} P_{21} P_{31} P_{02} P_{12} P_{22} P_{32} P_{03} P_{13} P_{23} P_{33} \]  
\[ (1-v)^3 \]  
\[ 3v(1-v)^2 3v^2(1-v) v^3 \]

In an OpenGL Program

```
glPatchParameteri(GL_PATCH_VERTICES, 16);
glBegin(GL_PATCHES);
glVertex3f( x00, y00, z00 );
glVertex3f( x10, y10, z10 );
glVertex3f( x20, y20, z20 );
glVertex3f( x30, y30, z30 );
glVertex3f( x01, y01, z01 );
glVertex3f( x11, y11, z11 );
glVertex3f( x21, y21, z21 );
glVertex3f( x31, y31, z31 );
glVertex3f( x02, y02, z02 );
glVertex3f( x12, y12, z12 );
glVertex3f( x22, y22, z22 );
glVertex3f( x32, y32, z32 );
glVertex3f( x03, y03, z03 );
glVertex3f( x13, y13, z13 );
glVertex3f( x23, y23, z23 );
glVertex3f( x33, y33, z33 );
glEnd();
```

This order is not set by OpenGL. It is set by you. Pick a convention yourself and stick to it. GLSL doesn’t care as long as you are consistent.

In the .glib File

```
In the TCS Shader
```

```
#version 400
#extension GL_ARB_tessellation_shader : enable
uniform float uOuter02, uOuter13, uInner0, uInner1;
layout( vertices = 16 )  out;
void main( ){
  gl_out[ gl_InvocationID ].gl_Position = gl_in[ gl_InvocationID ].gl_Position;
  gl_TessLevelOuter[0] = gl_TessLevelOuter[2] = uOuter02;
  gl_TessLevelInner[0] = uInner0;
  gl_TessLevelInner[1] = uInner1;
}
```
In the TES Shader

```cpp
layout ( quads, equal_spacing, ccw)  in;
out vec3 teNormal;

void main(  ) {
    vec4 p00 = gl_in[0].gl_Position;
    vec4 p10 = gl_in[1].gl_Position;
    vec4 p20 = gl_in[2].gl_Position;
    vec4 p30 = gl_in[3].gl_Position;
    vec4 p01 = gl_in[4].gl_Position;
    vec4 p11 = gl_in[5].gl_Position;
    vec4 p21 = gl_in[6].gl_Position;
    vec4 p31 = gl_in[7].gl_Position;
    vec4 p02 = gl_in[8].gl_Position;
    vec4 p12 = gl_in[9].gl_Position;
    vec4 p22 = gl_in[10].gl_Position;
    vec4 p03 = gl_in[12].gl_Position;
    vec4 p13 = gl_in[13].gl_Position;
    vec4 p23 = gl_in[14].gl_Position;
    vec4 p33 = gl_in[15].gl_Position;

    float u = gl_TessCoord.x;
    float v = gl_TessCoord.y;

    Assigning the intermediate pij’s is here to make the code more readable. We assume that the compiler will optimize this away.

    In the TES Shader – Computing the Position, given a u and v

    // the basis functions:
    float bu0 = (1.-u) * (1.-u) * (1.-u);
    float bu1 = 3. * u * (1.-u) * (1.-u);
    float bu2 = 3. * u * u * (1.-u);
    float bu3 = u * u * u;
    float dbu0 = -3. * (1.-u) * (1.-u);
    float dbu1 =  3. * (1.-u) * (1.-3.*u);
    float dbu2 =  3. * u *      (2.-3.*u);
    float dbu3 =  3. * u *      u;
    float bv0 = (1.-v) * (1.-v) * (1.-v);
    float bv1 = 3. * v * (1.-v) * (1.-v);
    float bv2 = 3. * v * v * (1.-v);
    float bv3 = v * v * v;
    float dbv0 = -3. * (1.-v) * (1.-v);
    float dbv1 =  3. * (1.-v) * (1.-3.*v);
    float dbv2 =  3. * v *      (2.-3.*v);
    float dbv3 =  3. * v *      v;

    // finally, we get to compute something:
    gl_Position = bu0 * ( bv0*p00 + bv1*p01 + bv2*p02 + bv3*p03 )
                  + bu1 * ( bv0*p10 + bv1*p11 + bv2*p12 + bv3*p13 )
                  + bu2 * ( bv0*p20 + bv1*p21 + bv2*p22 + bv3*p23 )
                  + bu3 * ( bv0*p30 + bv1*p31 + bv2*p32 + bv3*p33 );
}
```

In the TES Shader – Computing the Normal, given a u and v

```cpp
vec4 dpdu = dbu0 * ( bv0*p00 + bv1*p01 + bv2*p02 + bv3*p03 )
            + dbu1 * ( bv0*p10 + bv1*p11 + bv2*p12 + bv3*p13 )
            + dbu2 * ( bv0*p20 + bv1*p21 + bv2*p22 + bv3*p23 )
            + dbu3 * ( bv0*p30 + bv1*p31 + bv2*p32 + bv3*p33 );
vec4 dpdv = bu0 * ( dbv0*p00 + dbv1*p01 + dbv2*p02 + dbv3*p03 )
            + bu1 * ( dbv0*p10 + dbv1*p11 + dbv2*p12 + dbv3*p13 )
            + bu2 * ( dbv0*p20 + dbv1*p21 + dbv2*p22 + dbv3*p23 )
            + bu3 * ( dbv0*p30 + dbv1*p31 + dbv2*p32 + dbv3*p33 );

telnormal = normalize( cross( dpdu.xyz, dpdv.xyz ) );
```

Example: A Bézier Surface

```cpp
uOuter02 = uOuter13 = 5
uInner0    = uInner1   = 5
```

Example: Whole-Sphere Subdivision

```cpp
spheresubd.glib
```

Using the x, y, z, and w to specify the center and radius of the sphere
Making the Whole-Sphere Subdivision Adapt to Screen Coverage

spheresubd.tcs, I

#include <GL_ARB_tessellation_shader>

#version 400 compatibility

in float uDetail;

uniform float uScale;

taxiclip = uScale;

taxiin = uDetail;

taxiout = uDetail;

extreme points of the sphere

Example: Whole-Sphere Subdivision

spheresubd.tes

// Using x, y, z and w to specify the center and radius of the sphere.

Example: Whole-Sphere Subdivision

Example: Whole-Sphere Subdivision

Making the Whole-Sphere Subdivision Adapt to Screen Coverage

spheresubd.tcs, II

// Using the x, y, z and w to specify the center and radius of the sphere.

Example: Whole-Sphere Subdivision

Example: Whole-Sphere Subdivision

Making the Whole-Sphere Subdivision Adapt to Screen Coverage
Example: PN Triangles

General idea: turn each triangle into a triangular Bézier patch. Create the Bézier control points by using the vertices normals at the corner vertices. The Bézier patch equation can then be interpolated to any level of tessellation.

Spherical coordinates

Making the Whole-Sphere Subdivision Adapt to Screen Coverage

Example: PN Triangles

pntriangles.tex

pntriangles.tex, II

pntriangles.tex, III

pntriangles.tex, IV

pntriangles.tex, V

pntriangles.tex, VI

Example: PN Triangles

pntriangles.tex, VII

pntriangles.tex, VIII

Example: PN Triangles

pntriangles.tex, IX

pntriangles.tex, X

Example: PN Triangles

pntriangles.tex, XI

Example: PN Triangles

pntriangles.tex, XII

Example: PN Triangles

pntriangles.tex, XIII

Example: PN Triangles

pntriangles.tex, XIV

Example: PN Triangles

pntriangles.tex, XV

Example: PN Triangles

pntriangles.tex, XVI

Example: PN Triangles

pntriangles.tex, XVII

Example: PN Triangles

pntriangles.tex, XVIII

Example: PN Triangles

pntriangles.tex, XIX

Example: PN Triangles

pntriangles.tex, XX

Example: PN Triangles

pntriangles.tex, XXI

Example: PN Triangles

pntriangles.tex, XXII

Example: PN Triangles

pntriangles.tex, XXIII

Example: PN Triangles

pntriangles.tex, XXIV

Example: PN Triangles

pntriangles.tex, XXV

Example: PN Triangles

pntriangles.tex, XXVI

Example: PN Triangles

pntriangles.tex, XXVII

Example: PN Triangles

pntriangles.tex, XXVIII

Example: PN Triangles

pntriangles.tex, XXIX

Example: PN Triangles

pntriangles.tex, XXX

Example: PN Triangles

pntriangles.tex, XXXI

Example: PN Triangles

pntriangles.tex, XXXII

Example: PN Triangles

pntriangles.tex, XXXIII

Example: PN Triangles

pntriangles.tex, XXXIV

Example: PN Triangles

pntriangles.tex, XXXV

Example: PN Triangles

pntriangles.tex, XXXVI

Example: PN Triangles

pntriangles.tex, XXXVII

Example: PN Triangles

pntriangles.tex, XXXVIII

Example: PN Triangles

pntriangles.tex, XXXIX

Example: PN Triangles

pntriangles.tex, XL

Example: PN Triangles

pntriangles.tex, XLI

Example: PN Triangles

pntriangles.tex, XLII

Example: PN Triangles

pntriangles.tex, XLIII

Example: PN Triangles

pntriangles.tex, XLIV

Example: PN Triangles

pntriangles.tex, XLV

Example: PN Triangles

pntriangles.tex, XLVI

Example: PN Triangles

pntriangles.tex, XLVII

Example: PN Triangles

pntriangles.tex, XLVIII

Example: PN Triangles

pntriangles.tex, XLIX

Example: PN Triangles

pntriangles.tex, L

Example: PN Triangles

pntriangles.tex, LI

Example: PN Triangles

pntriangles.tex, LII

Example: PN Triangles

pntriangles.tex, LIII

Example: PN Triangles

pntriangles.tex, LIV

Example: PN Triangles

pntriangles.tex, LV

Example: PN Triangles

pntriangles.tex, LVI

Example: PN Triangles

pntriangles.tex, LVII

Example: PN Triangles

pntriangles.tex, LVIII

Example: PN Triangles

pntriangles.tex, LIX

Example: PN Triangles

pntriangles.tex, LX

Example: PN Triangles

pntriangles.tex, LXI

Example: PN Triangles

pntriangles.tex, LXII

Example: PN Triangles

pntriangles.tex, LXIII

Example: PN Triangles

pntriangles.tex, LXIV

Example: PN Triangles

pntriangles.tex, LXV

Example: PN Triangles

pntriangles.tex, LXVI

Example: PN Triangles

pntriangles.tex, LXVII

Example: PN Triangles

pntriangles.tex, LXVIII

Example: PN Triangles

pntriangles.tex, LXIX

Example: PN Triangles

pntriangles.tex, LXX

Example: PN Triangles

pntriangles.tex, LXXI

Example: PN Triangles

pntriangles.tex, LXXII

Example: PN Triangles

pntriangles.tex, LXXIII

Example: PN Triangles

pntriangles.tex, LXXIV

Example: PN Triangles

pntriangles.tex, LXXV

Example: PN Triangles

pntriangles.tex, LXXVI

Example: PN Triangles

pntriangles.tex, LXXVII

Example: PN Triangles

pntriangles.tex, LXXVIII

Example: PN Triangles

pntriangles.tex, LXXIX

Example: PN Triangles

pntriangles.tex, LXXX

Example: PN Triangles

pntriangles.tex, LXXXI

Example: PN Triangles

pntriangles.tex, LXXXII

Example: PN Triangles

pntriangles.tex, LXXXIII

Example: PN Triangles

pntriangles.tex, LXXXIV

Example: PN Triangles

pntriangles.tex, LXXXV

Example: PN Triangles

pntriangles.tex, LXXXVI

Example: PN Triangles

pntriangles.tex, LXXXVII

Example: PN Triangles

pntriangles.tex, LXXXVIII

Example: PN Triangles

pntriangles.tex, LXXXIX

Example: PN Triangles

pntriangles.tex, XC

Example: PN Triangles

pntriangles.tex, XCI

Example: PN Triangles

pntriangles.tex, XCI

Example: PN Triangles

pntriangles.tex, XCV

Example: PN Triangles

pntriangles.tex, XCVI

Example: PN Triangles

pntriangles.tex, XCVII

Example: PN Triangles

pntriangles.tex, XCVIII

Example: PN Triangles

pntriangles.tex, XCIX

Example: PN Triangles

pntriangles.tex, C

Example: PN Triangles

pntriangles.tex, CI

Example: PN Triangles

pntriangles.tex, CII

Example: PN Triangles

pntriangles.tex, CIII

Example: PN Triangles

pntriangles.tex, CIV

Example: PN Triangles

pntriangles.tex, CV

Example: PN Triangles

pntriangles.tex, CVI

Example: PN Triangles

pntriangles.tex, CVII

Example: PN Triangles

pntriangles.tex, CVIII

Example: PN Triangles

pntriangles.tex, CIX

Example: PN Triangles

pntriangles.tex, CX

Example: PN Triangles

pntriangles.tex, CXI

Example: PN Triangles

pntriangles.tex, CXII

Example: PN Triangles

pntriangles.tex, CXIII

Example: PN Triangles

pntriangles.tex, CXIV

Example: PN Triangles

pntriangles.tex, CXV

Example: PN Triangles

pntriangles.tex, CXVI

Example: PN Triangles

pntriangles.tex, CXVII

Example: PN Triangles

pntriangles.tex, CXVIII

Example: PN Triangles

pntriangles.tex, CXIX

Example: PN Triangles

pntriangles.tex, LXX

Example: PN Triangles

pntriangles.tex, LXXX

Example: PN Triangles

pntriangles.tex, LXXXI

Example: PN Triangles

pntriangles.tex, LXXXII

Example: PN Triangles

pntriangles.tex, LXXXIII

Example: PN Triangles

pntriangles.tex, LXXXIV

Example: PN Triangles

pntriangles.tex, LXXXV

Example: PN Triangles

pntriangles.tex, LXXXVI

Example: PN Triangles

pntriangles.tex, LXXXVII

Example: PN Triangles

pntriangles.tex, LXXXVIII

Example: PN Triangles

pntriangles.tex, LXXXIX

Example: PN Triangles

pntriangles.tex, XC

Example: PN Triangles

pntriangles.tex, XCI

Example: PN Triangles

pntriangles.tex, XCV

Example: PN Triangles

pntriangles.tex, XCVI

Example: PN Triangles

pntriangles.tex, XCVII

Example: PN Triangles

pntriangles.tex, XCVIII

Example: PN Triangles

pntriangles.tex, XCVII

Example: PN Triangles

pntriangles.tex, XCVIII

Example: PN Triangles

pntriangles.tex, XCVII

Example: PN Triangles

pntriangles.tex, XCVIII

Example: PN Triangles

pntriangles.tex, XCVII

Example: PN Triangles

pntriangles.tex, XCVIII

Example: PN Triangles

pntriangles.tex, XCVII

Example: PN Triangles

pntriangles.tex, XCVIII
Example: PN Triangles

```glsl
#version 400 compatibility
#extension GL_gpu_shader4: enable
#extension GL_geometry_shader4: enable

uniform float uShrink;
in vec3 teNormal[];
out float gLightIntensity;

const vec3 LIGHTPOS = vec3(5., 10., 10.);
vec3 V[3];
vec3 CG;

void ProduceVertex(int v)
{
    gLightIntensity = abs(dot(normalize(LIGHTPOS - V[v]), normalize(teNormal[v])));
    gl_Position = gl_ProjectionMatrix * vec4(CG + uShrink * (V[v] - CG), 1.);
    EmitVertex();
}

void main()
{
    V[0] = gl_PositionIn[0].xyz;
    V[1] = gl_PositionIn[1].xyz;
    CG = (V[0] + V[1] + V[2]) / 3.;
    ProduceVertex(0);
    ProduceVertex(1);
    ProduceVertex(2);
}
```

The Cow’s Tail is a Good Example of using PN Triangles

```glsl
uOuter = 1, uInner = 1
```

```glsl
uOuter = 2, uInner = 1
```

```glsl
uOuter = 2, uInner = 2
```

```glsl
uOuter = 2, uInner = 2
```

Notice how much improvement there is just by increasing the outer tessellation. The inner parts of the triangle are defined by the two vertex positions, but does nothing for the silhouette.

The Difference Between Tessellation Shaders and Geometry Shaders

By now, you are probably confused about when to use a Geometry Shader and when to use a Tessellation Shader. Both are capable of creating new geometry from existing geometry. See if this helps.

**Use a Geometry Shader when:**
1. You need to convert an input topology into a different output topology, such as in the silhouette and hedgehog shaders (triangles → lines) or the explosion shader (triangles → points).
2. You need some sort of geometry processing to come after the Tessellation Shader (such as how the shrink shader was used).

**Use a Tessellation Shader when:**
1. One of the built-in tessellation patterns will suit your needs.
2. You need more than 6 input vertices to define the surface being tessellated.
3. You need more output vertices than a Geometry Shader can provide.

Demonstrating the Limits of Tessellation Shaders

This tessellation is using 64x64 (the maximum allowed). This is pretty good-looking, but doesn’t come close to using the full 4096x2276 resolution available for the bump-map.