**Tessellation Shaders**

Thanks to Mike Bailey (OSU)

---

**Why a tessellation shader**

- You can perform adaptive subdivision based on a variety of criteria (size, curvature, etc.)
- Your application can provide coarser models (+ 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

---

**What patterns can the Tessellation shaders produce?**

- Lines
- Triangles
- Quads (output as triangles)

---

**Another view of the Tess Shader**

---

**Figure 8.1: Schematic of OpenGL tessellation**
In the OpenGL Program

```cpp
gBegin(GL_TESS_CONTROL_SHADER);
GLControlPointTessellationEvaluator(GL_ENTITY_COUNT, 1);

gControlPointTessellationEvaluator(GL_INDEX, 0);

gControlPointTessellationEvaluator(GL_LEVEL, 0);

gControlParameter(GL_INTERPOLATION_MODE, GL_LINEAR);

gControlPointTessellationEvaluator(GL_OUTPUT, 1);

gControlPointTessellationEvaluator(GL_IN, 1);

gControlPointTessellationEvaluator(GL_IN, 0);

gControlPointTessellationEvaluator(GL_RESOLVE_MODE, GL_TESSellation_SHADER);

gEnd();
```

TCS Inputs

- `gl_in[]` is an array of structures containing:
  - `gl_Position`
  - `gl_PointSize`
  - `gl_ClipDistance[]`
- `glInvocationID` tells you which output vertex you are working on. This must be the index into the `gl_out[]` 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 last `glBegin()` (the first one is `#0`)
- `barrier()` causes all instances of TCS’s to wait here

TCS Outputs

- `gl_out[]` is an array of structures containing:
  - `gl_Position`
  - `gl_PointSize`
  - `gl_ClipDistance[]`
- All invocations of the TCS have read-only access to all the output information. `barrier()` causes all instances of TCS’s to wait here
- `layout(vertices = n) out;` used to specify the number of vertices output to the TPG
  - Defining how many vertices this patch will output:
    - `layout(vertices = 10) out;
  - `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
- User-defined variables defined per-vertex are qualified as “out”
- User-defined variables defined per-patch are qualified as “patch 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

![Triangle Quad Isolines](image)

Figure 9.1 (RB)
TES Output Topologies: the Isolines Pattern

TES Output Topologies: the Isolines Pattern

TES Output Topologies: the Triangle Pattern

TES Output Topologies: the Triangle Pattern

TES Output Topologies: the Triangle Pattern

Demo tessmodes
TES subdivision

- layout(triangles, equal_spacing, ccw) in:
  - *equal_spacing* means that the triangle edges will be subdivided into segments with equal lengths (according to the TLs).
  - *fractional_even_spacing* means if there is a fractional portion based on TLs, it is evenly split between the ends.
  - *fractional_odd_spacing* means if there is a fractional portion based on TLs, it is not evenly split between the ends.

Demo tessellation modes

Example: A Bézier Curve

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

1. The Tessellation Control Shader figures how much to tessellate the curve based on screen area, curvature, etc.
   - Can tessellate non-uniformly if desired
   - The OpenGL tessellation can also do 1D curves. Just set DUB = 1.

Example: A Bézier Curve

2. The Tessellation Primitive Generator generates \((u, v)\) values for as many subdivisions as the TCS asked for:

Example: A Bézier Curve

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

\[ P(u) = u^3(-P_0 + 3P_1 - 3P_2 + P_3) + u^2(3P_0 - 6P_1 + 3P_2) + u(-3P_0 + 3P_1) + P_0 \]
In the OpenGL Program

gPatchParameteri(GL_PATCH_VERTICES, 4);
gBegin(GL_PATCHES);
gTexCoord(tu, tv);
gVertex3f(x0, y0, z0);
gEnd();

In the TCS Shader

uniform GLboolean enable;
layout (in) float u, v;

void main()
{
    gl_TessBeginPatch(u, v);
    gl_TessBeginPoint(0);
    gl_TessVertex(0, v);
    gl_TessBeginPoint(1);
    gl_TessVertex(1, v);
    gl_TessEndPatch();
}

In the TES Shader

#version 330
layout (in) float uv);

void main()
{
    float u = gl_TessCoord.x;
    float v = gl_TessCoord.y;
    float u2 = u * u;
    float v2 = v * v;
    float u2v = u2 * v;
    float u2v2 = u2 * v2;
    float u3v = u3 * v;
    float u3v2 = u3 * v2;
    gl_Position = u3v2xyz + u2v2xyz + u2vxyz + u3vxyz + u2vxyz + u3vxyz + u2vxyz + u3vxyz;
}

Assigning the interpolated vec3 is here to make the code more readable. We assume that the compiler will optimize this away.

Example: A Bézier Curve

Outer = 5

In the OpenGL Program

gPatchParameteri(GL_PATCH_VERTICES, 11);
gBegin(GL_PATCHES);
mark

Bézier Surface Parametric Equations

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

This order is unimportant. Pick an ordering yourself and stick to it. In OpenGL, the order doesn't come as long as you are consistent.
In the TCS Shader

```
everything @ 0.0;
elevance = 0.0001;  // allow 0.0001 relevance
set the vertex colors (To be used for texturing)

float t = 0.5;
float x = (1.0 - t) * (1.0 - t) * 1.0;
float y = (1.0 - t) * t * 1.0;
float z = t * t * 1.0;
```

In the TES Shader

Assigning the intermediate g_value to make the code more readable. We assume that the compiler will optimize this away.

```
// In the TES Shader

In the TES Shader – Computing the Position

```

In the TES Shader – Computing the Normal

```

Example: A Bezier Surface

```
Bezier Patch

```

```
```
Demo Displacement