Lesson 12 – Tessellation Shaders

PV227 – GPU Rendering

Jiří Chmelík, Jan Čejka
Fakulta informatiky Masarykovy univerzity

15. 12. 2015

Tessellation Shaders

- new programmable stage (optional), requires OpenGL 4.0
- between vertex shader and geometry shader,
- use the correct spelling :-)

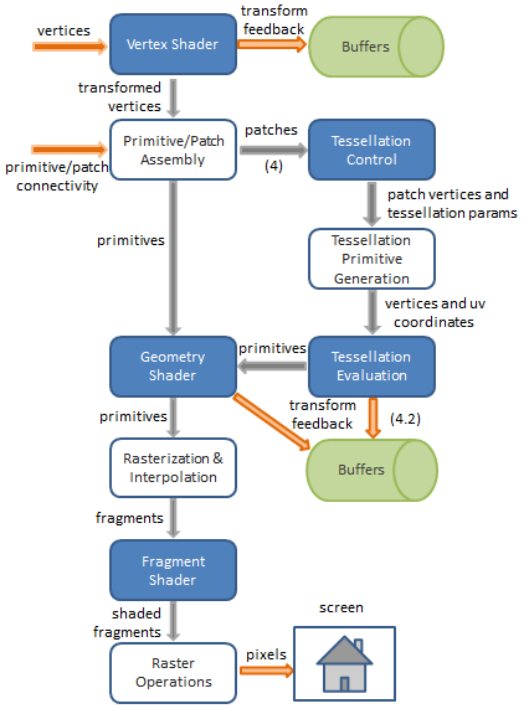


Figure: Taken from lighthouse3d.com

Tessellation Shaders (cont.)

- Tessellation Control Shader (TCS)
  - Hull Shader in HLSL
  - optional, programmable
- Primitive generation
  - fixed
- Tessellation Evaluation Shader (TES)
  - Domain Shader in HLSL
  - required, programmable

Tessellation Shaders — Primitive Generation

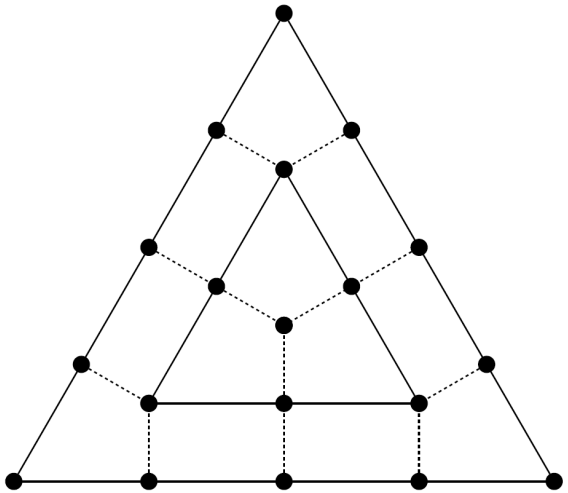


Figure: Triangle

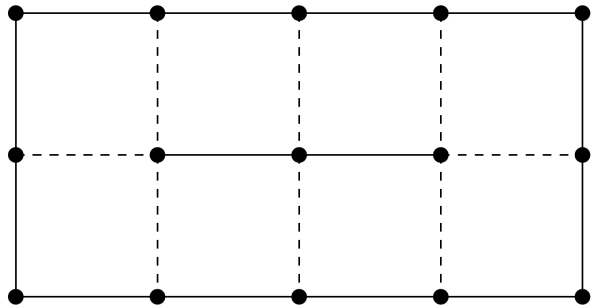


Figure: Quad

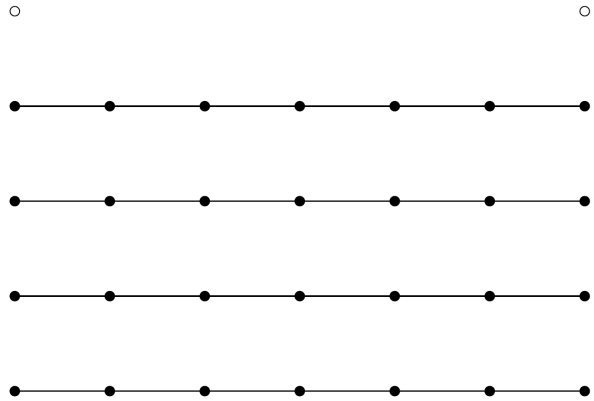


Figure: Lines

Images taken from OpenGL Specification

Tessellation Shaders — Shaders

- Tessellation Evaluation Shader
  - computes the data of each generated vertex
  - similar to vertex shaders

- Tessellation Control Shader
  - computes the level of the tessellation (the density of the mesh)
  - computes the data of the control points and other patch data

Patches

- New primitive, only for tessellation shaders
- Consist of 1 – 32 vertices
- Individual patches, no strips

```
1 glPatchParameteri(GL_PATCH_VERTICES, 16);
2 glDrawArrays(GL_PATCHES, ...);
```

- OpenGL does **not** define the mapping between input vertices and control points, the programmer does!

Tessellation Control Shader in GLSL

- Consumes one patch, generates one patch, like geometry shader
- Unlike geometry shaders, TCS is executed ones per output vertex.
  - Number of generated vertices defined by `layout(vertices = 4) out;`
  - Use `gl_InvocationID` to get the index of the vertex for which this TCS is executed.
- Array of per vertex input data from the vertex shader
  - Example: `in vec4 position_vs[];`
  - Every TCS has access to each per vertex input data (read-only)
- Array of per vertex output data into the TES
  - Example: `out vec4 position_tcs[];`
  - Every TCS has **readonly** access to each per vertex output data
  - Only the one for which TCS is executed can **write** the data of its vertex
  - Use `barrier();` to make sure the data written by TCS are visible to other TCS.

Tessellation Control Shader in GLSL

- Per patch data, marked as `out patch`, passed into TES
  - Example: `out patch int materialIdx;`
- Tessellation levels defining the density of the mesh
  - `gl_TessLevelInner[2]` describes the density inside the patch
  - `gl_TessLevelOuter[4]` describes the density at the boundary of the patch
  - When set to 0, the whole patch is discarded
- TCS is optional, when missing:
  - Per vertex data passes through from vertex shader into TES
  - The number of patch vertices stays the same
  - Tessellation levels defined from C++ code using `glPatchParameterfv`

Tessellation Evaluation Shader in GLSL

- Defines the patch topology: `layout (...) in;`
  - triangles / quads / isolines
  - fractional_odd_spacing / fractional_even_spacing / equal_spacing
  - cw / ccw
  - point_mode / (nothing)
- Array of per vertex input data from the TCS
  - Example: `in vec4 position_vs[];`
  - Every TES has access to each per vertex input data (readonly)
- Per patch data, marked as `in patch`, from TES
  - Example: `in patch int materialIdx;`
- Coordinate of the tessellated vertex in the patch `vec3 gl_TessCoord`
  - triangles uses 3 coordinates (xyz)
  - quads and isolines use 2 coordinates (xy)

Example — Patch topology

- Try `layout (...) in;` in TES and tessellation levels in TCS

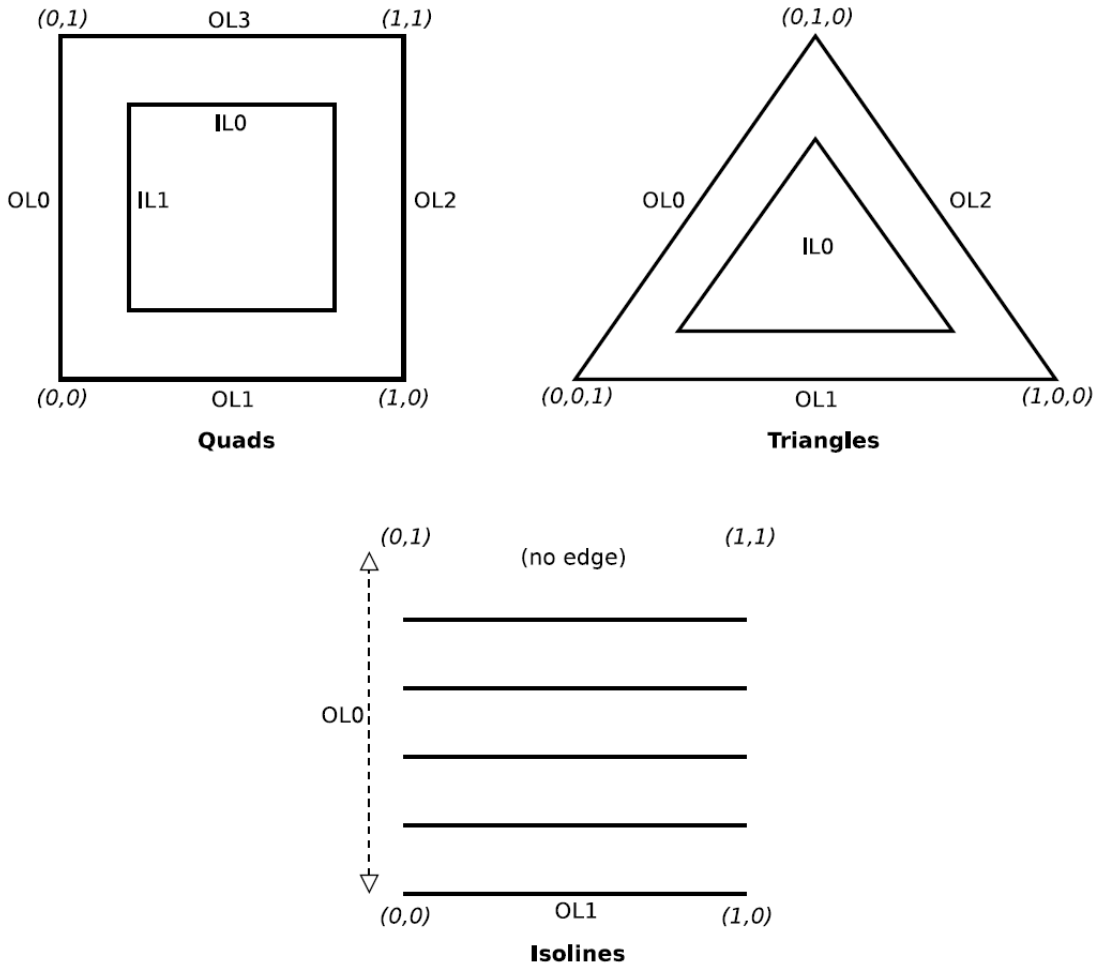
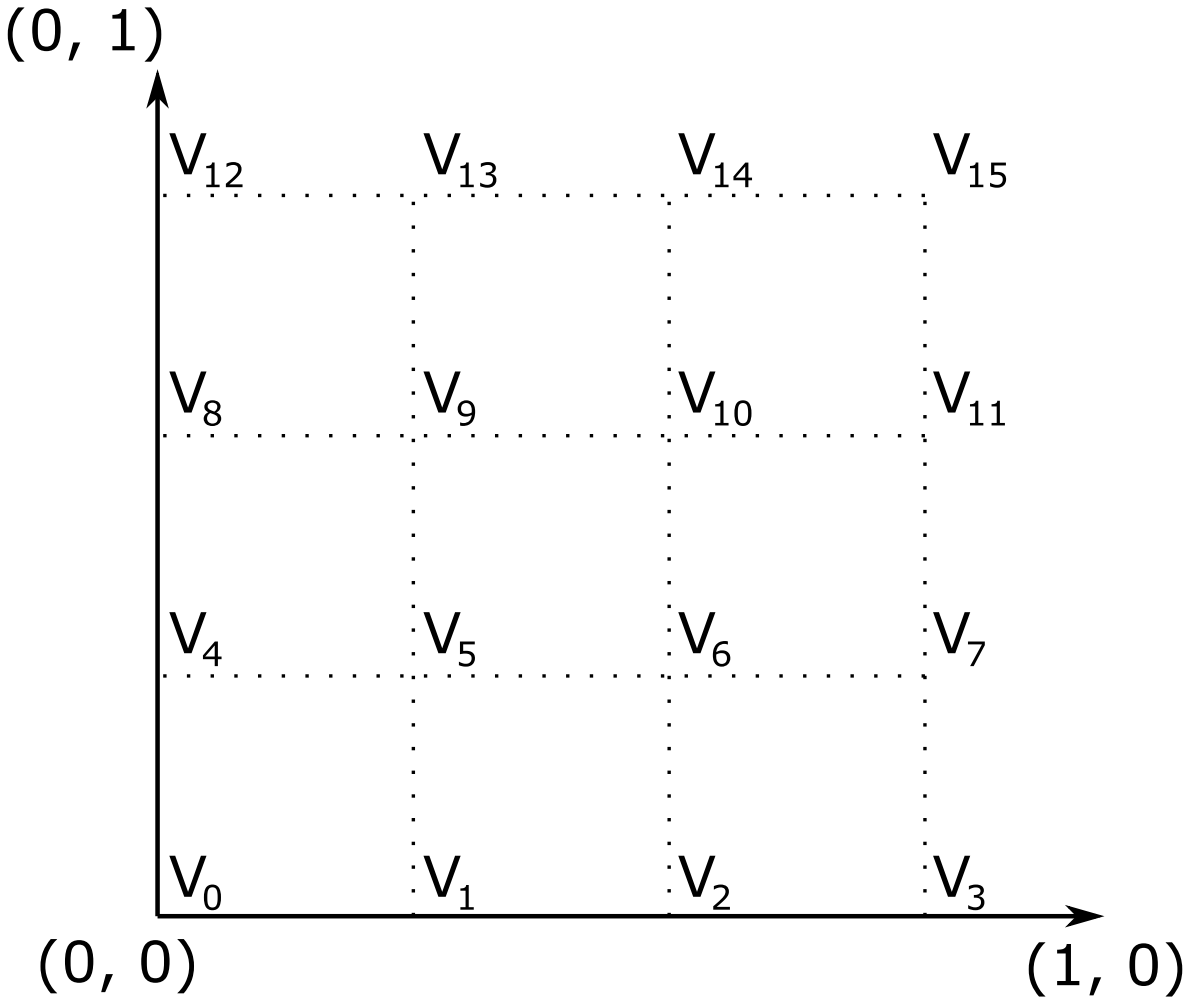


Figure: Taken from OpenGL Specification

Example — Utah teapot

- Use quads to smoothly tessellate 32 Bezier patches, each with 16 control points



- Add some displacement