

CS770/870 Fall 2015

Advanced GLSL

Based on material from
Bailey notes from Oregon State and
web sources cited in slides and others to be documented later.

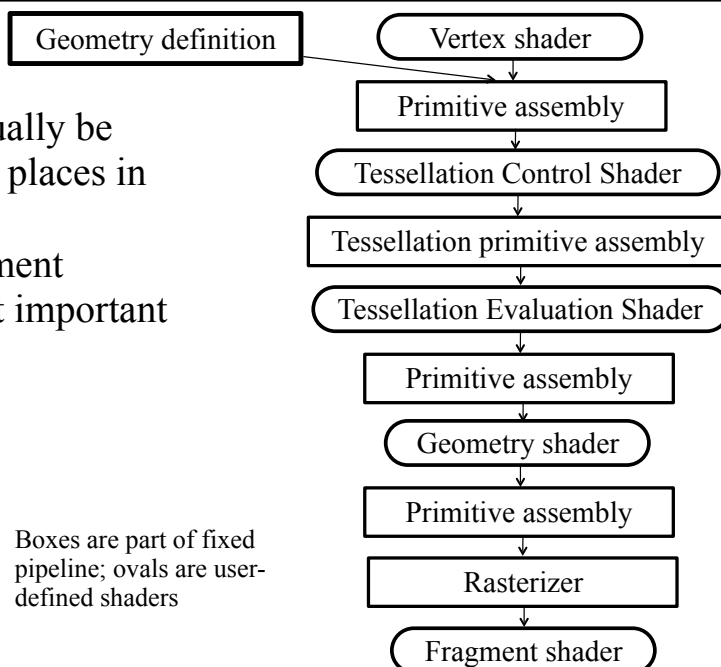
05/08/17

CS770/870 Spring 2017 Bergeron

1

Expanded Graphics Pipeline

- Shaders can actually be inserted in more places in pipeline
- Vertex and fragment shaders are most important

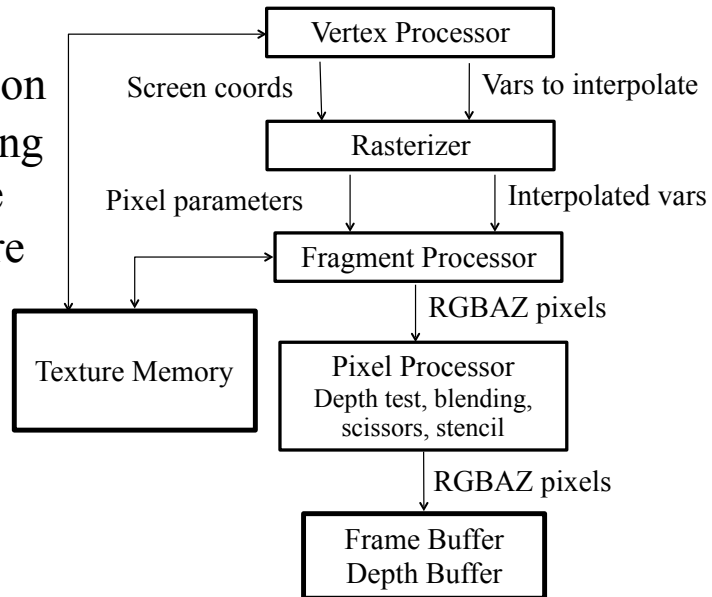


CS770/870 Fall 2015 Bergeron

2

Fragment Processing Pipeline

- *pixel*: rgbaz for a frame buffer location
- *fragment*: everything needed to compute pixel: rgbaz, texture coords, and more



Topics of Interest

- **Tessellation**
 - tessellation control shader
 - tessellation evaluation shader
- **Geometry** shader
- Textures v. **samplers**
- Framebuffer objects
- **WorkGroups**
- Advanced vertex shader functionality
 - `drawArraysInstanced`, `drawArraysBaseInstanced`, `multiDrawArrays`, `drawArraysIndirect`

Tessellation

- **Optional Vertex Processing**
 - vertex patch data is subdivided
 - generates new vertices
 - need to specify new vertex attributes: position, color, normal, texture coords, etc.

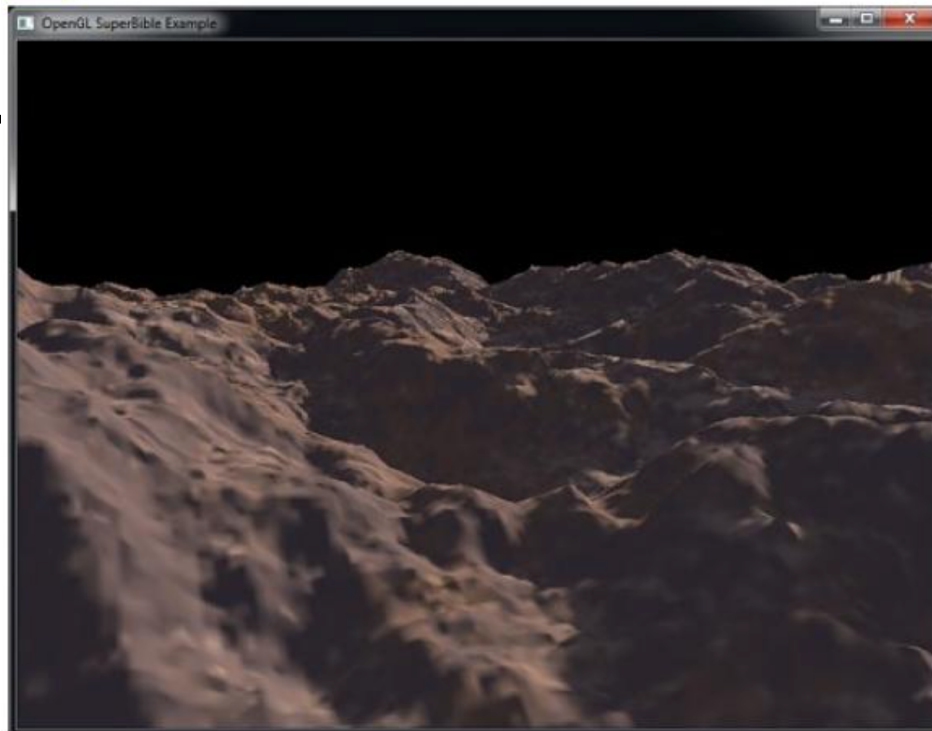
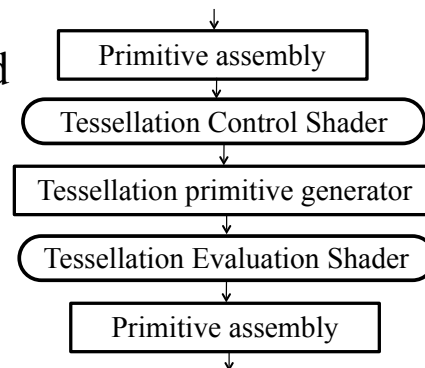


Figure 8.12: Terrain rendered using tessellation

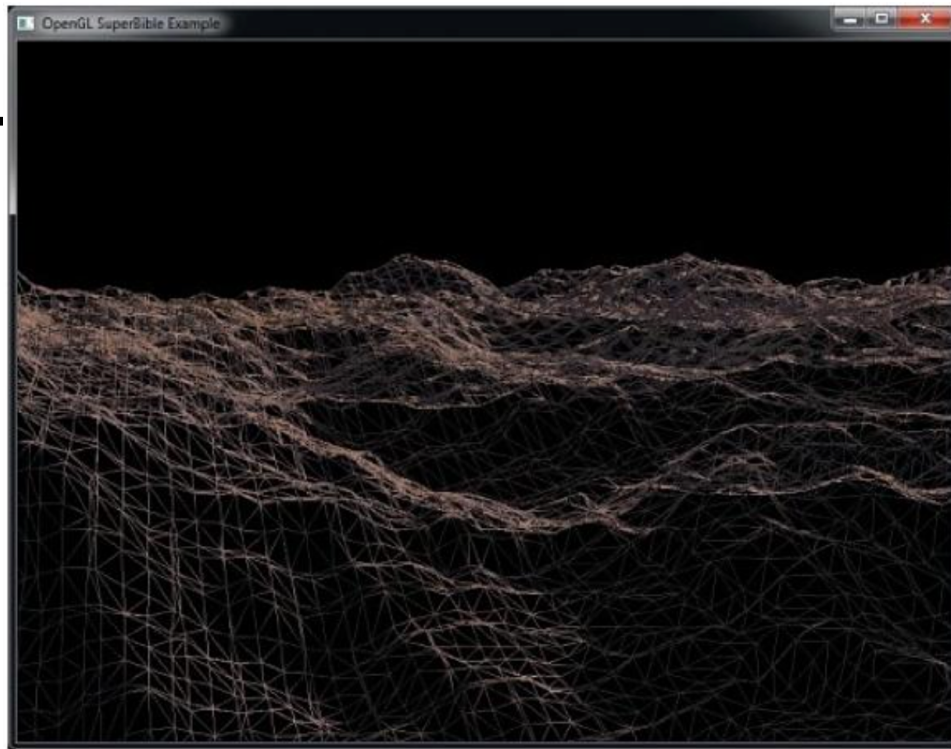


Figure 8.13: Tessellated terrain in wireframe
From OpenGL SuperBible CS770/870 Fall 2015 Bergeron

7

Patches

- OpenGL patch *primitive* is a block of vertices in which each successive n vertices constitutes 1 patch
 - `glPatchParameteri( patchId, 6 );` // 6 verts per patch
 - vertices in patch often thought of as *control points*
 - max patch size is impl-dependent, but is ≥ 32
- No particular order is required in a patch; it depends on how the patch is used
- Can have all the attribute information associated with any vertex block: position, color, normals, texture coords, etc.

Tessellation Levels

- Patches have *inner* and *outer edges*
- Tessellation level independently set for inner/outer edge types
- A level n means that the edge will be partitioned into n segments by the tessellation.
- *outer* edges usually have higher n than *inner* edges
 - Allows for better stitching between patches.

<http://www.geeks3d.com/20100730/test-first-contact-with-opengl-4-0-gpu-tessellation/>

CS770/870 Fall 2015 Bergeron

9

Fractional Tessellation Levels

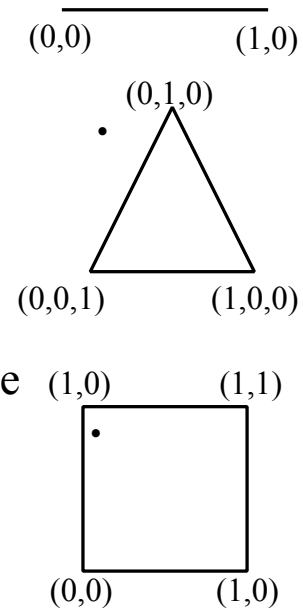
- Level specification is float
- 3 options for fractional component
 - *equal_spacing* : all segments same length
 - *fractional_odd_spacing* : let n = next-lower *odd* integer (for 4.6, that is 7); make segment size segments and split the remaining space in 2.
 - *fractional_even_spacing* : same as odd, but n is next-lower *even* integer
 - get smoother transitions between levels
- <http://www.geeks3d.com/20130128/know-your-opengl-tessellation-spacing-modes/>

CS770/870 Fall 2015 Bergeron

10

Tessellation Coordinates

- Similar to texture coordinates
- Depend on data type: line, triangle, quad
- triangles use *barycentric* coordinates
 - represent any point inside by linear combination of vertex coordinates
 - subdivision points inside triangle can be defined on an *abstract* triangle with these coordinates and later transformed by actual Cartesian coordinates.



Triangle Barycentric Coordinates

- Given triangle defined by $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2$ where $\mathbf{v}_i = (\mathbf{x}_i, \mathbf{y}_i)$
- Can represent every point in triangle with

$$\mathbf{x} = \lambda_0 \mathbf{x}_0 + \lambda_1 \mathbf{x}_1 + \lambda_2 \mathbf{x}_2$$

$$\mathbf{y} = \lambda_0 \mathbf{y}_0 + \lambda_1 \mathbf{y}_1 + \lambda_2 \mathbf{y}_2$$
 where $\lambda_0 + \lambda_1 + \lambda_2 = 1$
- Given Cartesian coordinates, can get barycentric:

$$\lambda_0 = ((\mathbf{y}_1 - \mathbf{y}_2)(\mathbf{x} - \mathbf{x}_2) + (\mathbf{x}_2 - \mathbf{x}_1)(\mathbf{y} - \mathbf{y}_2)) / \det(\mathbf{T})$$

$$\lambda_1 = ((\mathbf{y}_2 - \mathbf{y}_0)(\mathbf{x} - \mathbf{x}_2) + (\mathbf{x}_0 - \mathbf{x}_2)(\mathbf{y} - \mathbf{y}_2)) / \det(\mathbf{T})$$

$$\lambda_2 = 1 - \lambda_0 - \lambda_1$$
 where $\mathbf{T} = \begin{vmatrix} \mathbf{x}_0 - \mathbf{x}_2 & \mathbf{x}_1 - \mathbf{x}_2 \\ \mathbf{y}_0 - \mathbf{y}_2 & \mathbf{y}_1 - \mathbf{y}_2 \end{vmatrix}$

Tessellation Level Specifications

- Inner level[2] = [2.0, 4.0]
 - *quad*: u direction edges split into 2 parts, v into 4
 - *triangle*: all edges split into 2; 2nd element not used
- Outer level[4] = [2.0, 3.0, 4.0, 5.0]
 - *quad*: left edge split in 2 parts, bottom into 3, right into 4, and top into 5.
 - *triangle*: edge between (0,0,1) and (0,1,0) split into 2
edge between (0,0,1) and (1,0,0) split into 3
edge between (1,0,0) and (0,1,0) split into 4
(last value not used).

<http://www.informit.com/articles/article.aspx?p=2120983>

Tessellation Primitive Generator

- Fixed-function stage: not programmable by user,
- User provides parameters (directly or via a shader)
 - tessellation levels
 - spacing parameters for tessellated vertices
 - data type: triangles, quads, isolines, points
- Determines **only**
 - # vertices to generate, their order, primitive type
- It does not use the actual input vertex information
 - all output is relative to an abstract normalized (0,1) primitive object

Tessellation Control Shader (TCS)

- Processes exactly one vertex of a patch
- Can communicate with other TCS instances for same patch
 - can share information
 - must synchronize
 - share output information
- Outputs (mostly) go directly to the Tessellation Evaluation Shader (TES)
 - patch output inner and outer tessellation levels goes to the tessellation primitive generator

Tessellation Evaluation Shader (TES)

- Input
 - abstract patch from tessellation primitive generator
 - actual vertex data for **entire** patch (from TCS or VS)
- Output
 - A single vertex from the tessellation
- Assumption: not verified
 - if a patch has 6 vertexes, 16 TES instances are created and all patch data and each knows which it is supposed to produce as output.

Geometry Shader

- Geometry shader takes input after vertex processing (including tessellation) and prior to vertex post-processing
- Geometry shader input: OpenGL *Primitive* stream
 - points
 - lines, lines_adjacency
 - triangles, triangle_adjacency
- Note that geometry shader has the **entire** triangle, not just 1 vertex at a time.
- Each geometry shader only processes 1 kind of Primitive

• http://web.engr.oregonstate.edu/~mjb/cs519/Handouts/geometry_shaders.1pp.pdf

CS770/870 Fall 2015 Bergeron

Stencil Buffer

- Like a depth buffer where values act like a cardboard stencil, that can either allow colors through, prevent them.
- OpenGL's stencils have more options than cardboard!

• <https://open.gl/depthstencils>