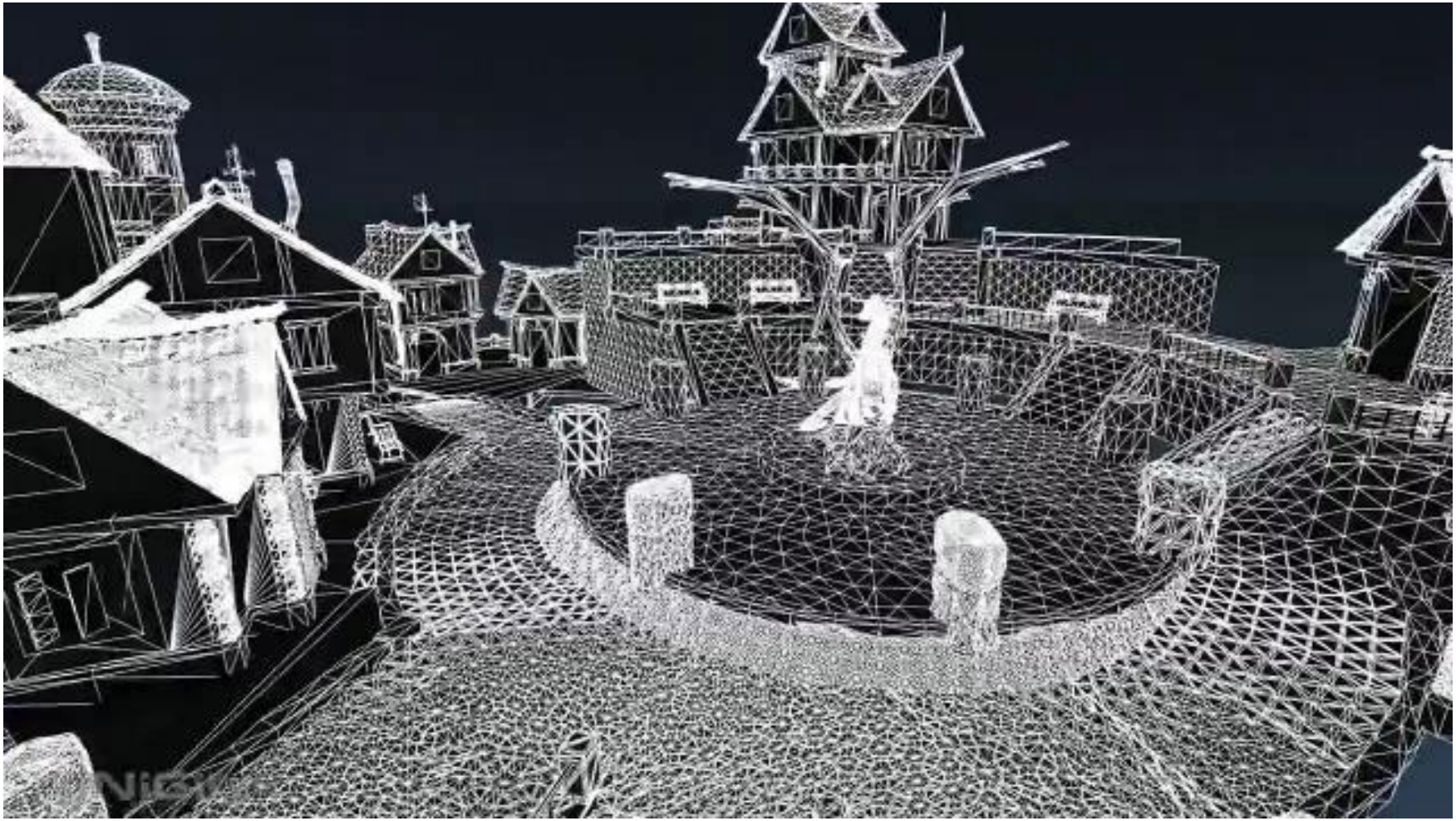
# Hardware-Tessellation

Matthias Labschütz, Peter Houska

Institute of Computer Graphics
and Algorithms

**Vienna University of Technology**

# Tessellation

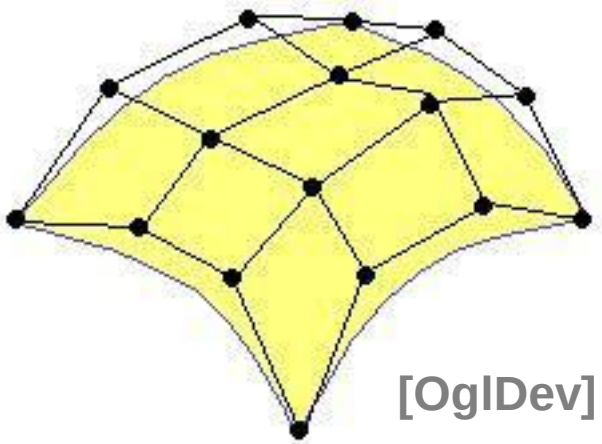


[Unigine Heaven Benchmark]

# Why Tessellation?

- Continuous & View-dependent Level Of Detail
  - e.g. PN-Triangles (using Bézier patches instead of triangles)
- Displacement mapping
  - not just modification of lighting (→ bump mapping, etc.)
  - changes objects‘ silhouette
- Requires OpenGL 4.0

  or ARB_tessellation_shader

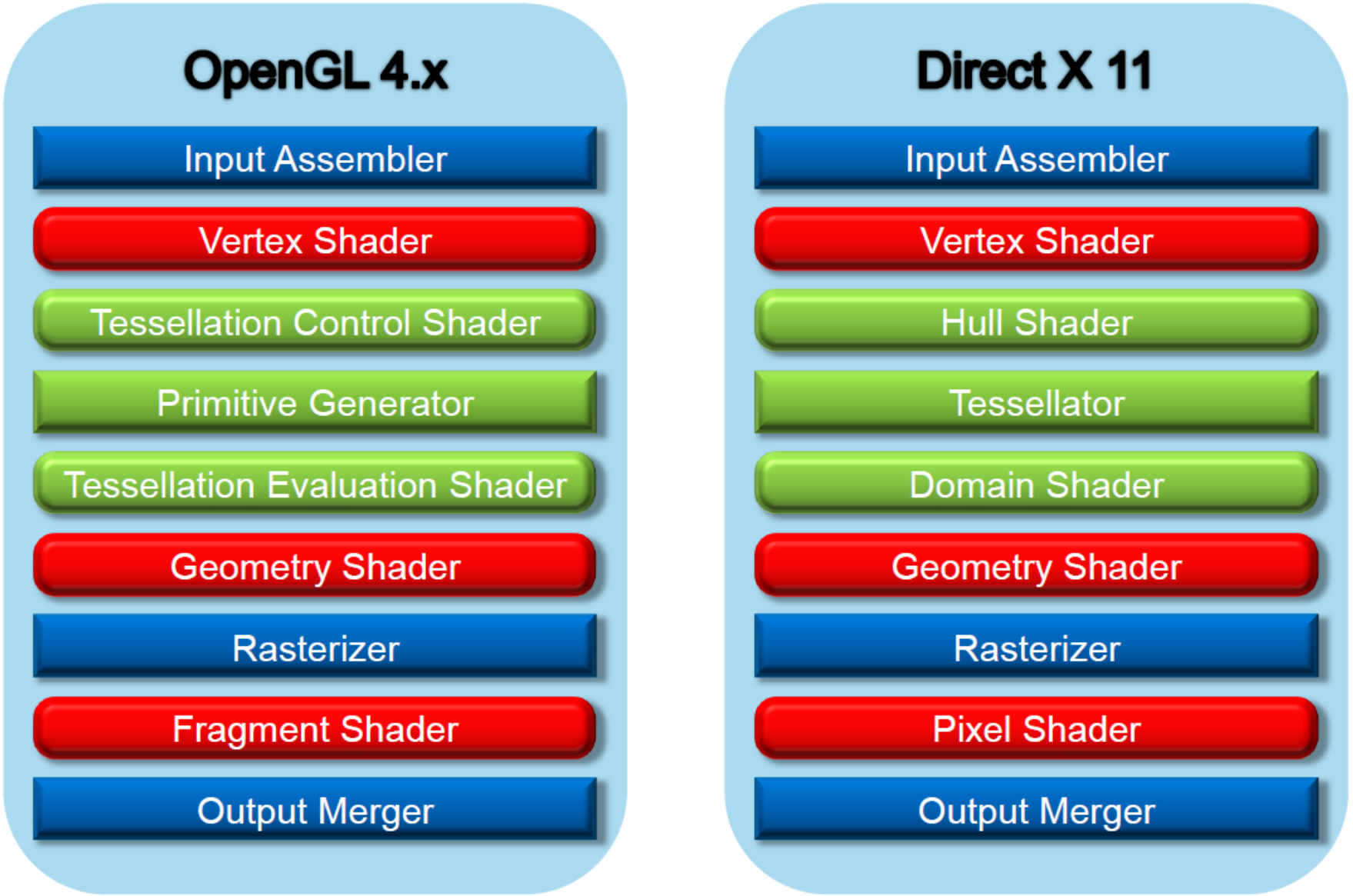


[OglDev]

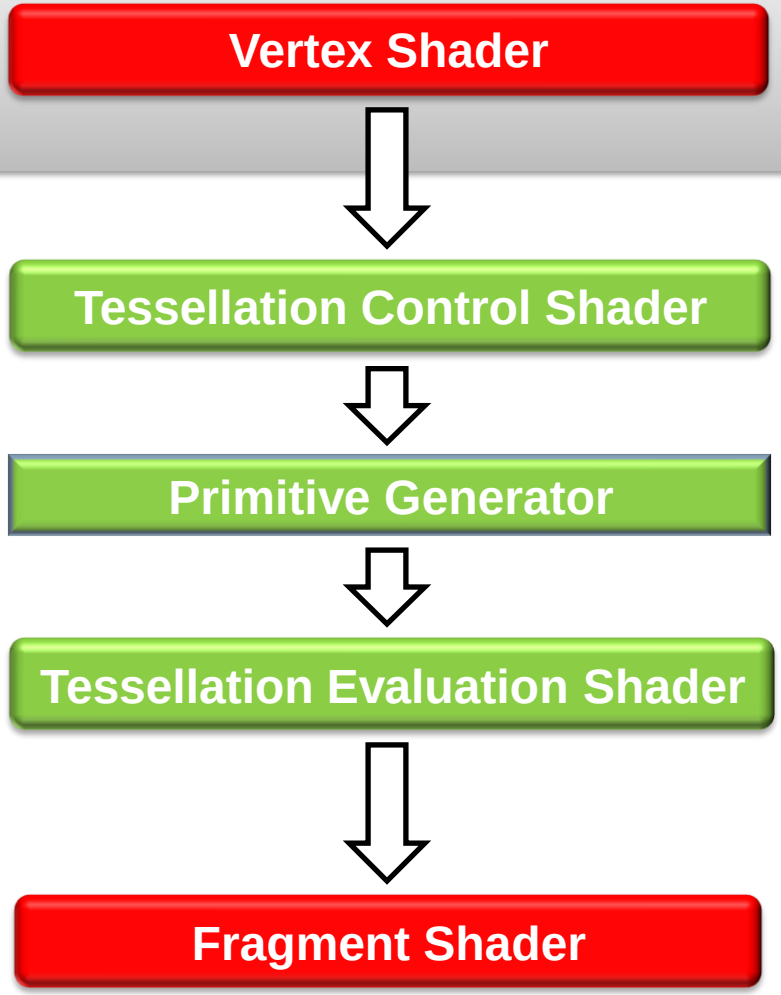
# Example: Displacement Mapping

[UnHe]

# New Shader Stages in the Pipeline



# Example: Simple Displacement Mapping

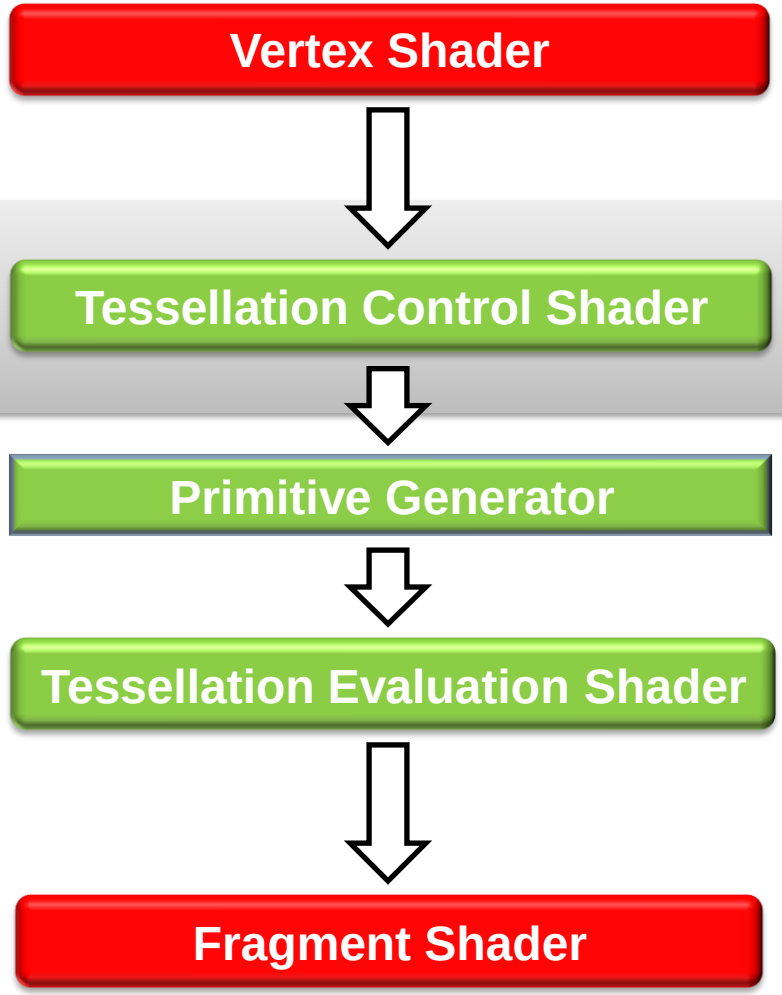


# Vertex Shader

```
#version 410 core

// matrix that transforms from object space to world space (WS)
uniform mat4 modelMatrix;

in vec4 in_Position_VS;  // attribute 0: object space vertex position
in vec2 in_TextCoord_VS; // attribute 1: texture coordinate

// variables to pass down information from VS to TCS
out vec4 in_Position_CS;
out vec2 in_TextCoord_CS;

void main(void) {
   in_Position_CS = modelMatrix * in_Position_VS; // transform vertex to WS
   in_TextCoord_CS = in_TextCoord_VS;
}
```

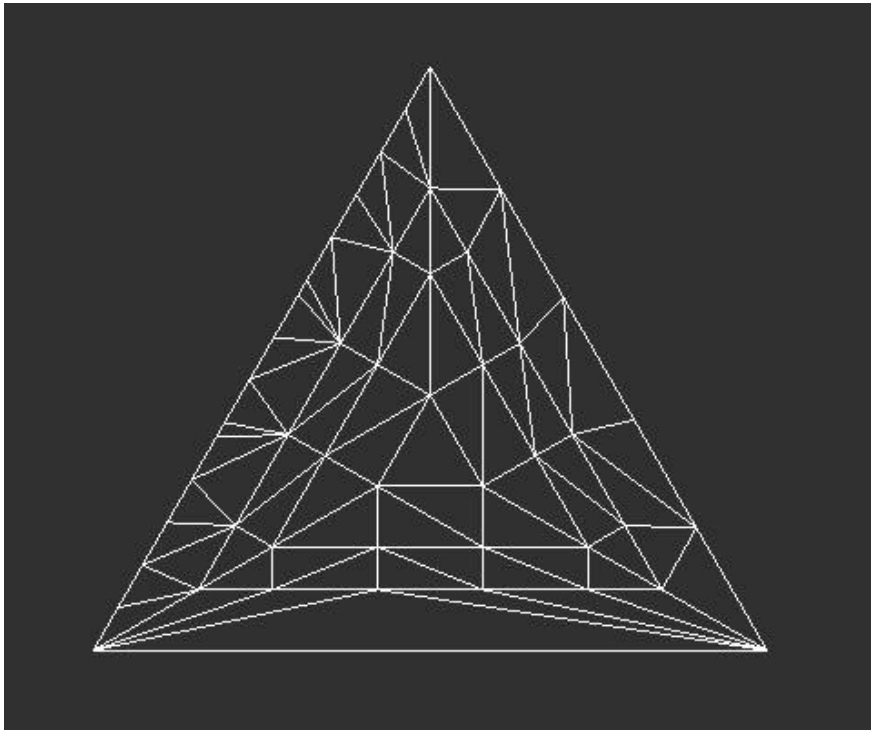


# Shader Flow


Vertex Shader
Tessellation Control Shader
Primitive Generator
Tessellation Evaluation Shader
Fragment Shader

# Tessellation Control Shader (TCS)

- Defines how much tessellation a patch (~ polygon) gets.
  - `gl_TessLevel`
- Can filter vertex data from the vertex shader (access to all vertices per patch)
- Outputs control points to the Evaluation Shader



[OglDev]

# Application Code: Shader attachment

- Basically the same procedure as for VS/GS/FS, just replace constants:

```
...
glCreateShader(GL_TESS_CONTROL_SHADER);
...
glCreateShader(GL_TESS_EVALUATION_SHADER);
...
```

- Don’t forget

```
glAttachShader(prog, tess_control_shader);
glAttachShader(prog, tess_evaluation_shader);
```



# Application Code: Draw

- Use primitive mode GL_PATCHES for rendering, e.g.:

```
glDrawArrays(GL_PATCHES, first_vert, vert_cnt);
glDrawElements(GL_PATCHES, idx_buffer_len, GL_UNSIGNED_INT, 0);
```

- Configure number of vertices in patch

```
glPatchParameteri(GL_PATCH_VERTICES, n);
```

- Query for the maximum allowable number n

```
GLint MaxPatchVertices;
glGetIntegerv(GL_MAX_PATCH_VERTICES, &MaxPatchVertices);
```



# Tessellation Control Shader (1/2)

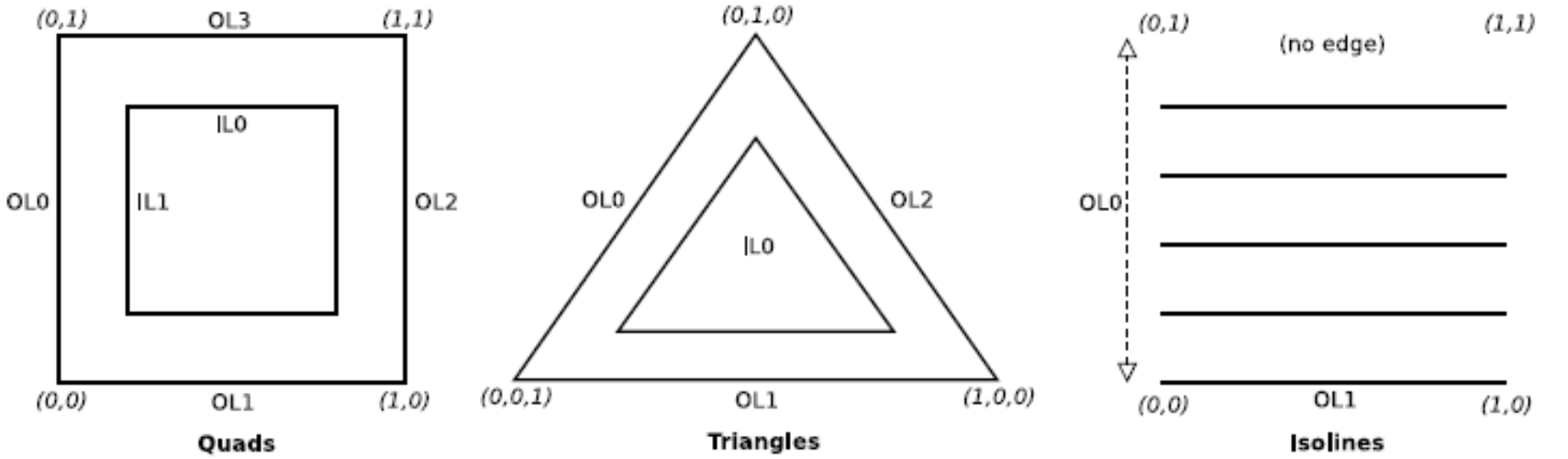
```
#version 410 core

// define the number of Vertices in the output patch
// (can be different from the input patch size)
layout (vertices = 3) out;

// attributes of the input Vertices (from Vertex Shader)
in vec4 in_Position_CS[];
in vec2 in_TextCoord_CS[];

// attributes of the output Vertices (to Tessellation Evaluation Shader)
out vec4 in_Position_ES[];
out vec2 in_TextCoord_ES[];
```



# TCS: Built in parameters

- Active vertex:
  - `gl_InvocationID`
- Tessellation level
  - `gl_TessLevelOuter[]`
  - `gl_TessLevelInner[]`

[OglSpec]



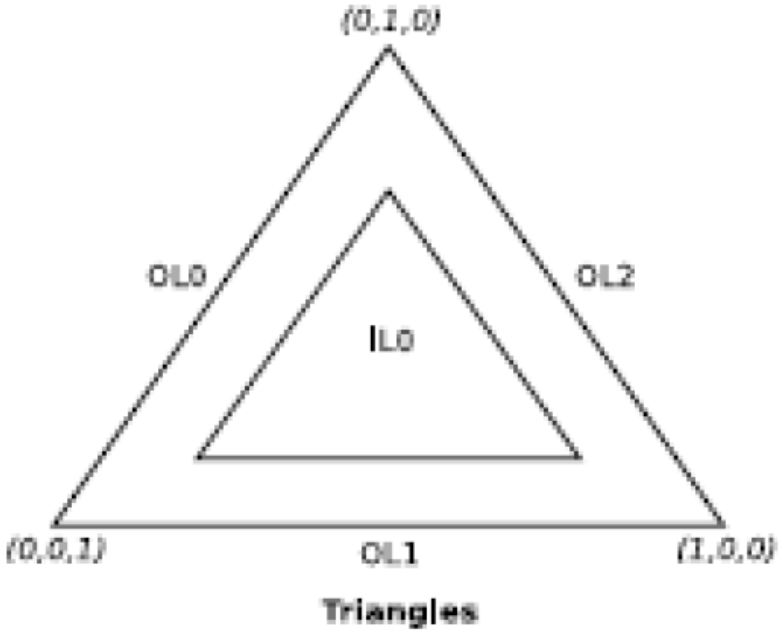

IL 0-1 inner tessellation levels; OL 0-3 outer tessellation levels

# Tessellation Control Shader (2/2)

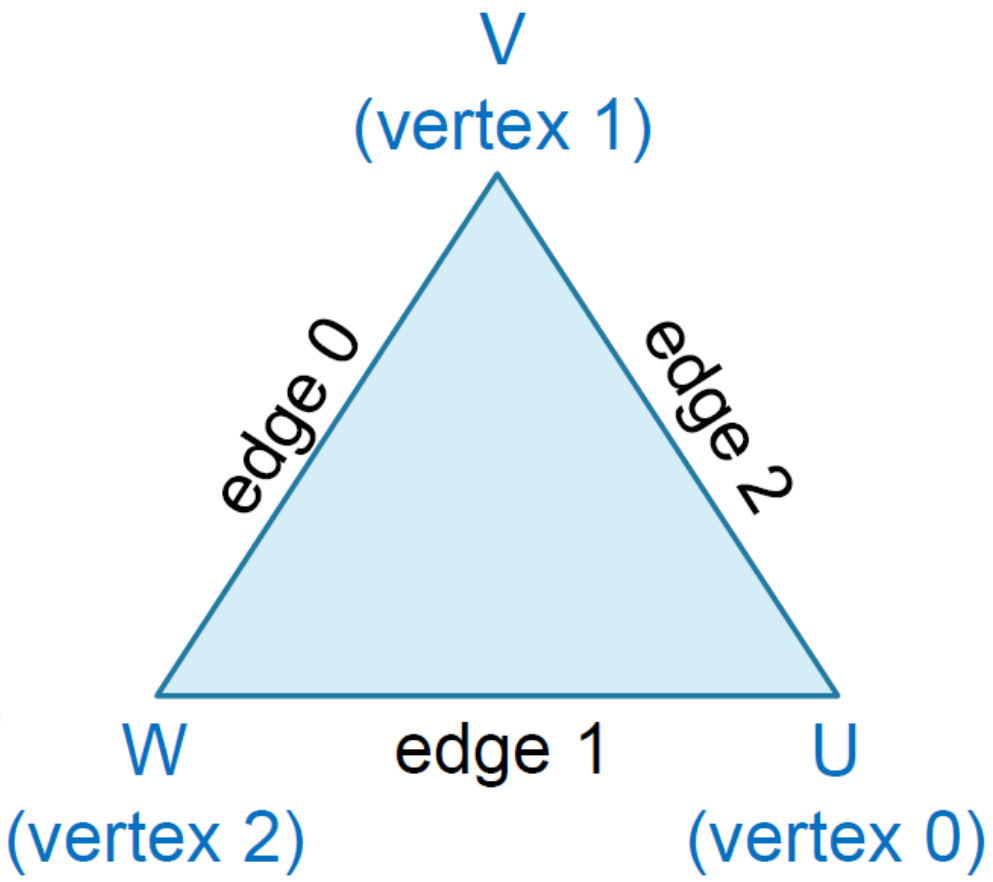
```
void main(void) {
    // Set the control points (vertices) of the output patch
    in_Position_ES[gl_InvocationID] = in_Position_CS[gl_InvocationID];
    in_TextCoord_ES[gl_InvocationID] = in_TextCoord_CS[gl_InvocationID];

    // the next snippet just sketches the calculations...
    // based on the vertex distances to the camera, we choose the TLs
    // see [OglDev] for an example implementation

    // Calculate the tessellation levels
    if (gl_InvocationID == 0) {
        gl_TessLevelOuter[0] = calc_TL(in_Position_CS[1], in_Position_CS[2]);
        gl_TessLevelOuter[1] = calc_TL(in_Position_CS[2], in_Position_CS[0]);
        gl_TessLevelOuter[2] = calc_TL(in_Position_CS[0], in_Position_CS[1]);
        gl_TessLevelInner[0] = calc_inner_TL( ... );
    }
}
```



# Tessellation Control Shader - Indices




[OglSpec]

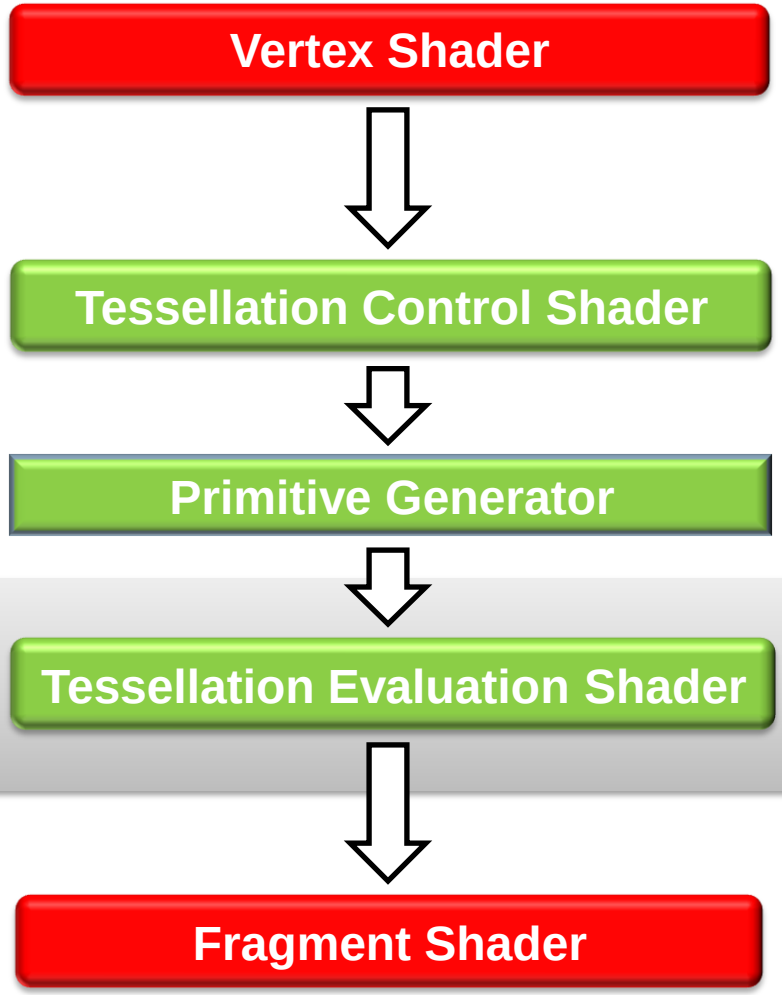


# Tessellation Control Shader (TCS)

- Per Vertex output:
  - e.g.: `out vec2 texCoord[];`
  - Can only write to active ID `texCoord[gl_InvocationID]`
- Per Patch output:
  - e.g.: `patch out vec4 data;`
- Synchronization:
  - When reading the current output variables
  - `barrier()`



# Shader Flow


Vertex Shader
Tessellation Control Shader
Primitive Generator
Tessellation Evaluation Shader
Fragment Shader

# Tessellation Evaluation Shader (TES)

- Takes the result of the tessellation
  - Abstract position (weighting) inside the patch between [0,1]
- Calculates interpolated position and other per-vertex data
- Generates one Vertex



# Tessellation Evaluation Shader (1/3)

```
#version 410 core

// tell PG to emit triangles in counter-clockwise order
// with equal spacing (try fractional_even_spacing and
// fractional_odd_spacing, too!)
layout(triangles, equal_spacing, ccw) in;

uniform mat4 projMatrix;
uniform mat4 viewMatrix;

uniform sampler2D dispTexture; // texture for displacement values
uniform float displacement_factor;

// these vertex attributes are passed down from the TCS
in vec4 in_Position_ES[];
in vec2 in_TextCoord_ES[];

out vec2 in_TextCoord_FS; // the only attribute we'll need later in the FS
```

# Tessellation Evaluation Shader (TES)

- Parameters:
  - `layout(param1, param2, ...) in;`
  - Patch type: `isolines, triangles, quads`
  - Spacing: between Vertices
  - Primitive ordering `cw` or `ccw` for face culling
- Inputs:
  - Per vertex and per patch input from the Tessellation Control Shader (TCS)
  - `gl_TessCoord` location in the abstract patch



# Tessellation Evaluation Shader (2/3)

```
// Interpolate values v0-v2 based on the barycentric coordinates
// of the current vertex within the triangle
vec2 interpolate2D(vec2 v0, vec2 v1, vec2 v2) {
     return  vec2(gl_TessCoord.x) * v0 +
             vec2(gl_TessCoord.y) * v1 +
             vec2(gl_TessCoord.z) * v2;
}

// Interpolate values v0-v2 based on the barycentric coordinates
// of the current vertex within the triangle
vec3 interpolate3D(vec3 v0, vec3 v1, vec3 v2) {
     return  vec3(gl_TessCoord.x) * v0 +
             vec3(gl_TessCoord.y) * v1 +
             vec3(gl_TessCoord.z) * v2;
}
```

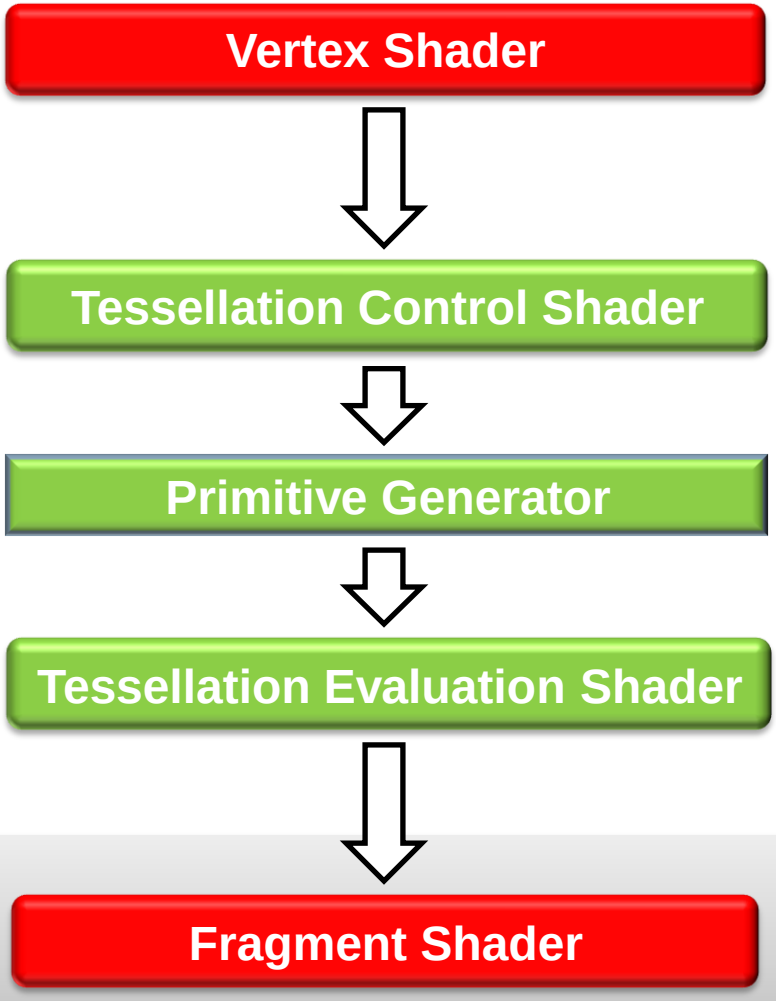


# Tessellation Evaluation Shader (3/3)

```
void main(void) {
    // Interpolate attribs of output vertex using its barycentric coords
    vec4 position = vec4( interpolate3D( in_Position_ES[0].xyz,
       in_Position_ES[1].xyz, in_Position_ES[2].xyz ), 1.0);
    vec2 textCoord = interpolate2D(in_TextCoord_ES[0], in_TextCoord_ES[1],
       in_TextCoord_ES[2]);

    // Displace the vertex along the WS y-axis
    float displacement = texture(dispTexture, textCoord).x;
    position.y += displacement * displacement_factor;

    // transform to NDC
    gl_Position = projMatrix * viewMatrix * WS_vertex_pos_tess;
    in_TextCoord_FS = textCoord; // pass texture coordinate to FS
}
```



# Shader Flow

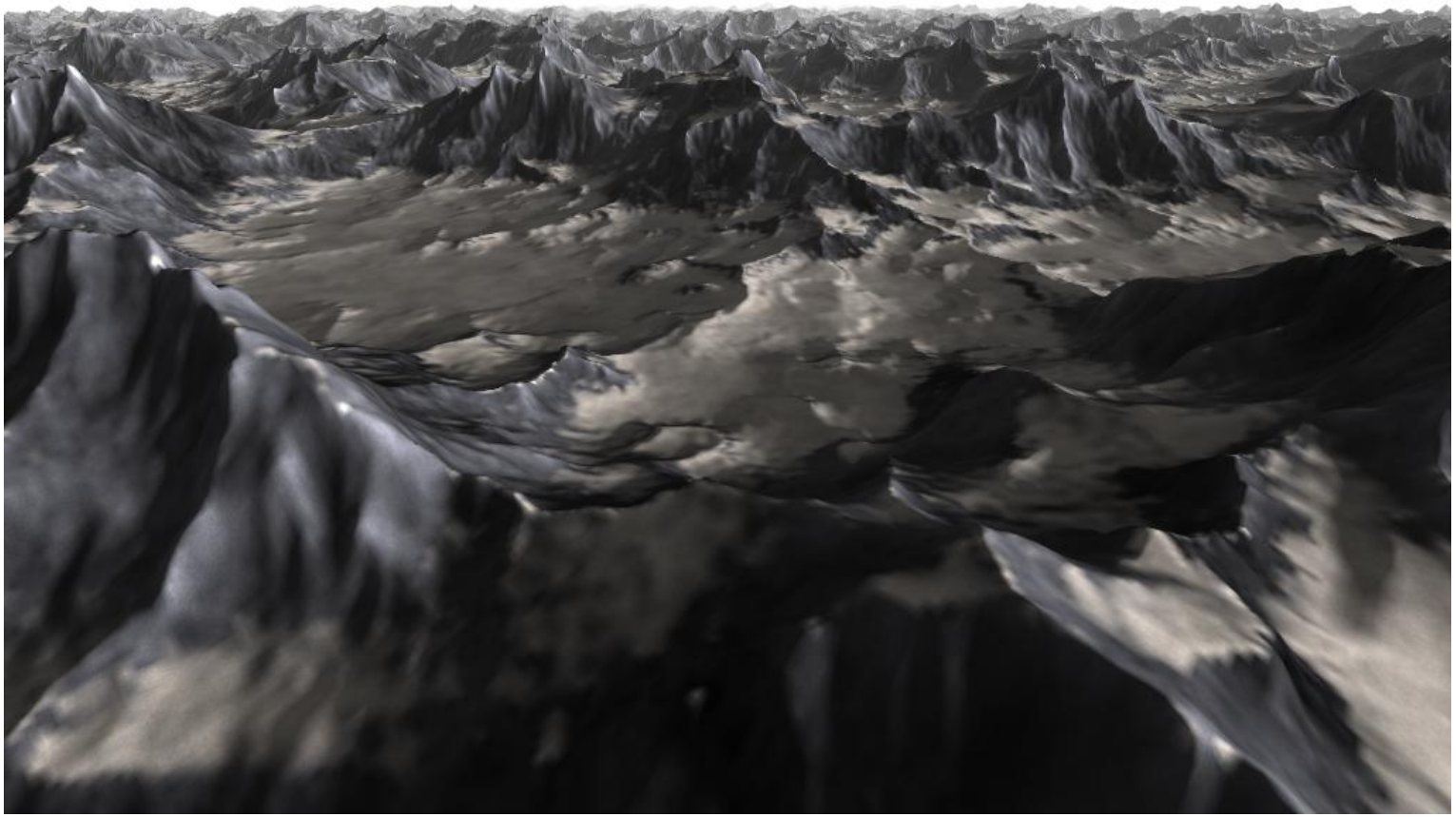

Vertex Shader
Tessellation Control Shader
Primitive Generator
Tessellation Evaluation Shader
Fragment Shader

# Fragment Shader

```
#version 410 core

uniform sampler2D colorTexture; // color texture

in vec2 in_TextCoord_FS; // passed down from the TES

out vec4 out_Color; // the final fragment color

void main(void) {
    out_Color = texture(colorTexture, in_TextCoord_FS);
}
```

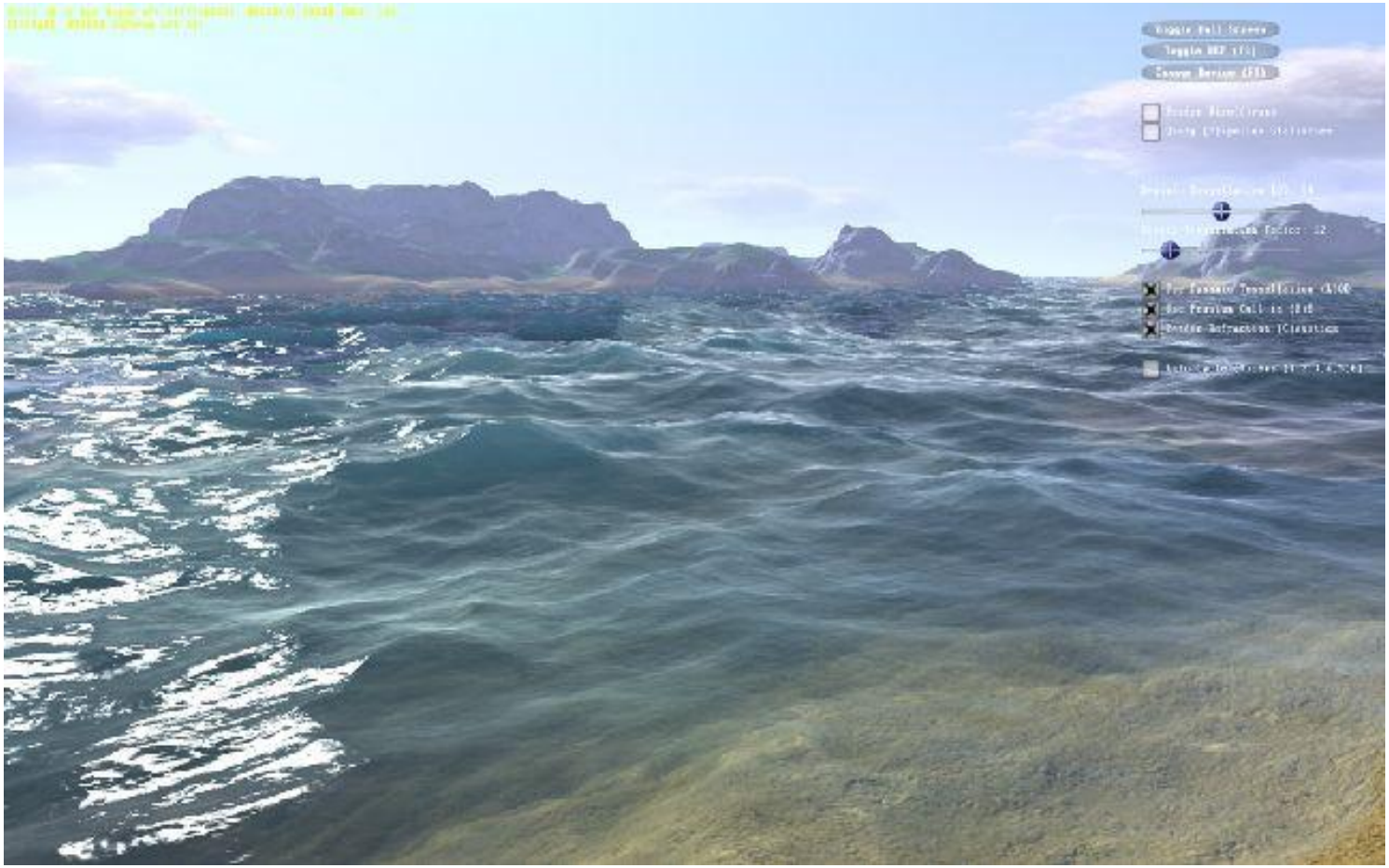


# Example Videos



http://codeflow.org/entries/2010/nov/07/opengl-4-tessellation/

# Example Videos



http://www.geforce.com/games-applications/pc-applications/fermi-water-demo

# Examples

http://www.cg.tuwien.ac.at/research/publications/2013/JAHRMANN-2013-IGR/

# References

- https://www.opengl.org/wiki/Tessellation
- Tutorials:
  - ogldev.atspace.co.uk/www/tutorial30/tutorial30.html
  - prideout.net/blog/?p=48
  - www.geeks3d.com/20100730/test-first-contact-with-opengl-4-0-gpu-tessellation/
  - web.engr.oregonstate.edu/~mjb/cs519/Handouts/tessellation.6pp.pdf
  - rastergrid.com/blog/2010/09/history-of-hardware-tessellation/