What’s New in OpenGL for OS X

Session 507

Chris Niederauer
GPU Software

Overview

What’s new in OpenGL for OS X

- OpenGL feature support update
- Key features
- Compute and OpenGL
- Migrating to Core Profile

OpenGL Features

OpenGL Feature Support Update

Available in OpenGL Core Profile on OS X

- Framebuffer objects
- Vertex array objects
- Instancing
- Primitive restart
- Uniform buffer objects
- Geometry shaders
- Floating point textures
- Multisample textures
- Texture buffer objects
- Transform feedback
- Seamless cube maps
- Mip-map generation
- Sync objects

OpenGL Feature Support Update

Available in OS X Mavericks

NEW

- Framebuffer objects
- Vertex array objects
- Instancing
- Primitive restart
- Uniform buffer objects
- Geometry shaders
- Floating point textures
- Multisample textures
- Texture buffer objects
- Transform feedback
- Seamless cube maps
- Mip-map generation
- Sync objects

OpenGL Feature Support Update

Available in OS X Mavericks

- Framebuffer objects
- Vertex array objects
- Instancing
- Primitive restart
- Uniform buffer objects
- Geometry shaders
- Floating point textures
- Multisample textures
- Texture buffer objects
- Transform feedback
- Seamless cube maps
- Mip-map generation
- Sync objects

Now available on OS X:

- Texture swizzle
- Separate shader objects
- Explicit attribute location
- Sampler objects
- ES2 compatibility
- Texture storage
- Texture barrier
- Extended blend support
- More texture formats
- More vertex attribute types

NEW

OpenGL Feature Support Update

Available in OS X Mavericks

- Framebuffer objects
- Vertex array objects
- Instancing
- Primitive restart
- Uniform buffer objects
- Geometry shaders
- Floating point textures
- Multisample textures
- Texture buffer objects
- Transform feedback
- Seamless cube maps
- Mip-map generation
- Sync objects

Now available on OS X:

- Texture swizzle
- Separate shader objects
- Explicit attribute location
- Sampler objects
- ES2 compatibility
- Texture storage
- Texture barrier
- Extended blend support
- More texture formats
- More vertex attribute types

And on modern GPUs:

- Tessellation shaders
- Texture gather
- Shader subroutines
- Sample shading
- Draw indirect
- Multiple viewports
- 64-bit processing
- and more…

OpenGL Feature Support Update

Available in OS X Mavericks

- Framebuffer objects
- Vertex array objects
- Instancing
- Primitive restart
- Uniform buffer objects
- Geometry shaders
- Floating point textures
- Multisample textures
- Texture buffer objects
- Transform feedback
- Seamless cube maps
- Mip-map generation
- Sync objects

Now available on OS X:

- Texture swizzle
- Separate shader objects
- Explicit attribute location
- Sampler objects
- ES2 compatibility
- Texture storage
- Texture barrier
- Extended blend support
- More texture formats
- More vertex attribute types

And on modern GPUs:

- Tessellation shaders
- Texture gather
- Shader subroutines
- Sample shading
- Draw indirect
- Multiple viewports
- 64-bit processing
- and more...

Tessellation Shaders

GL_ARB_tessellation_shader

Tessellation Shaders

Overview

- Use the GPU to tessellate geometry for you
  - Submit coarse geometry
  - GPU performs tessellation
- Defined using shaders
  - How course or fine
  - Positions, etc., of new vertices

Tessellation Shaders
Concept

- Benefits
  - Dynamically increase polygon density
  - Able to significantly decrease vertex bandwidth
- Common usage techniques
  - Displacement mapping
  - Terrain rendering
  - High-order surfaces
- Availability
  - Uses new pipeline stage on modern GPUs
  - Check for `GL_ARB_tessellation_shader` using `glGetStringi`

Tessellation Shaders

Unigine Heaven 4.0

Benchmark
Camera
Environment
Tessellation
Settings
Quality
Sound
Quit

Tessellation Shaders

Unigine Heaven 4.0

Benchmark
Camera
Environment
Tessellation
Settings
Quality
Sound
Quit

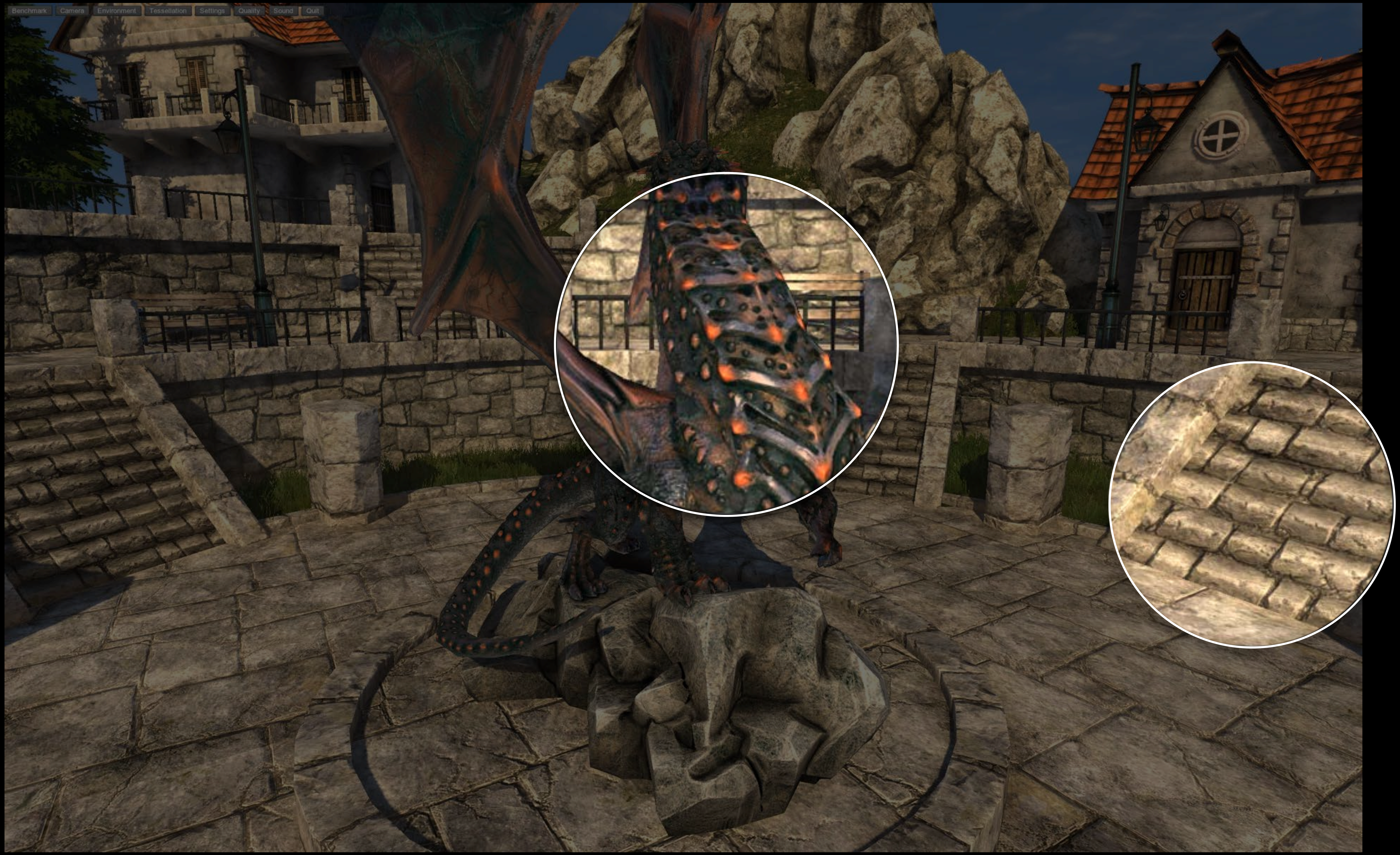
Tessellation Shaders

Unigine Heaven 4.0

Benchmark
Camera
Environment
Tessellation
Settings
Quality
Sound
Quit

Tessellation Shaders

Unigine Heaven 4.0



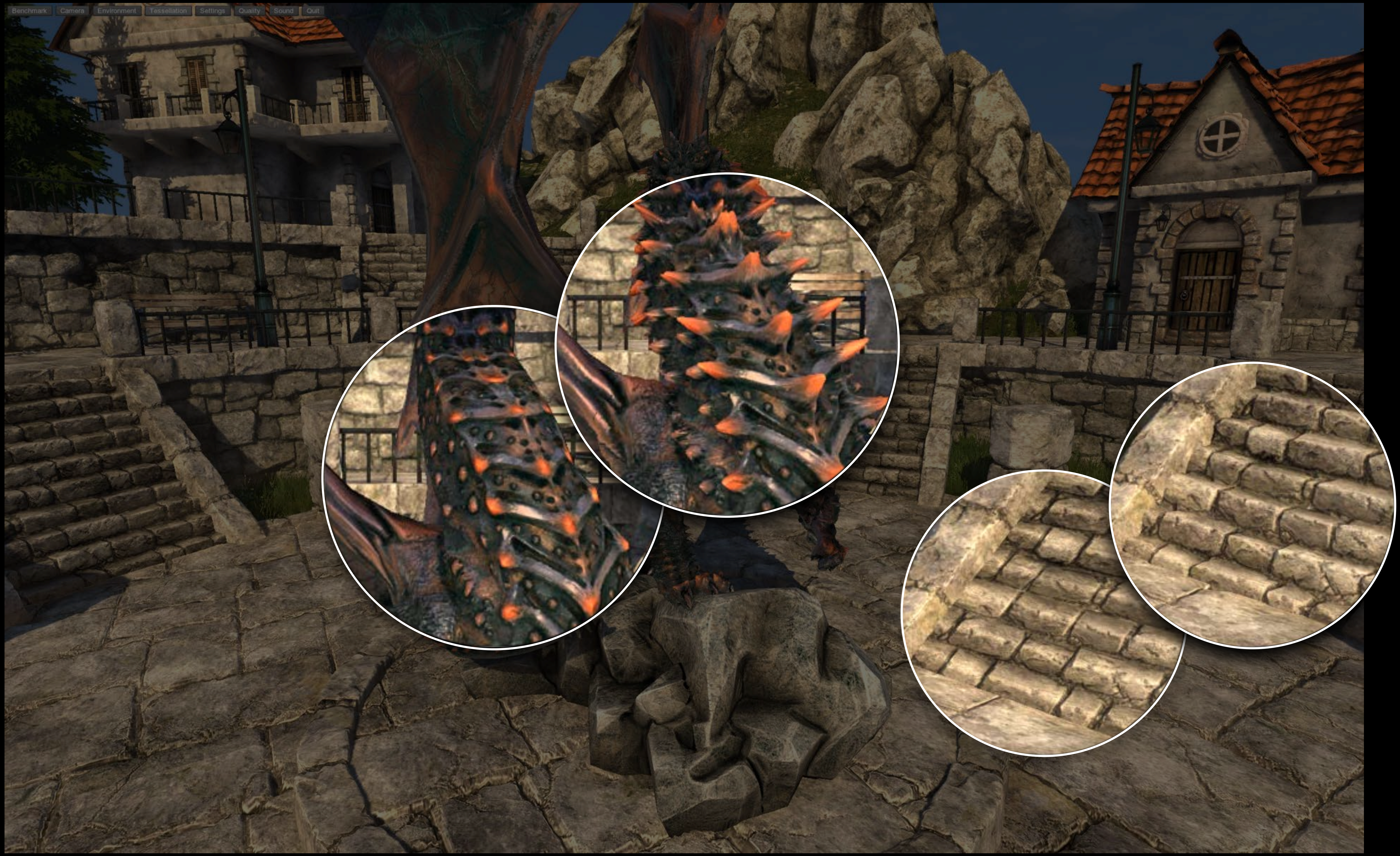
Tessellation Shaders

Unigine Heaven 4.0

Benchmark
Camera
Environment
Tessellation
Settings
Quality
Sound
Quit

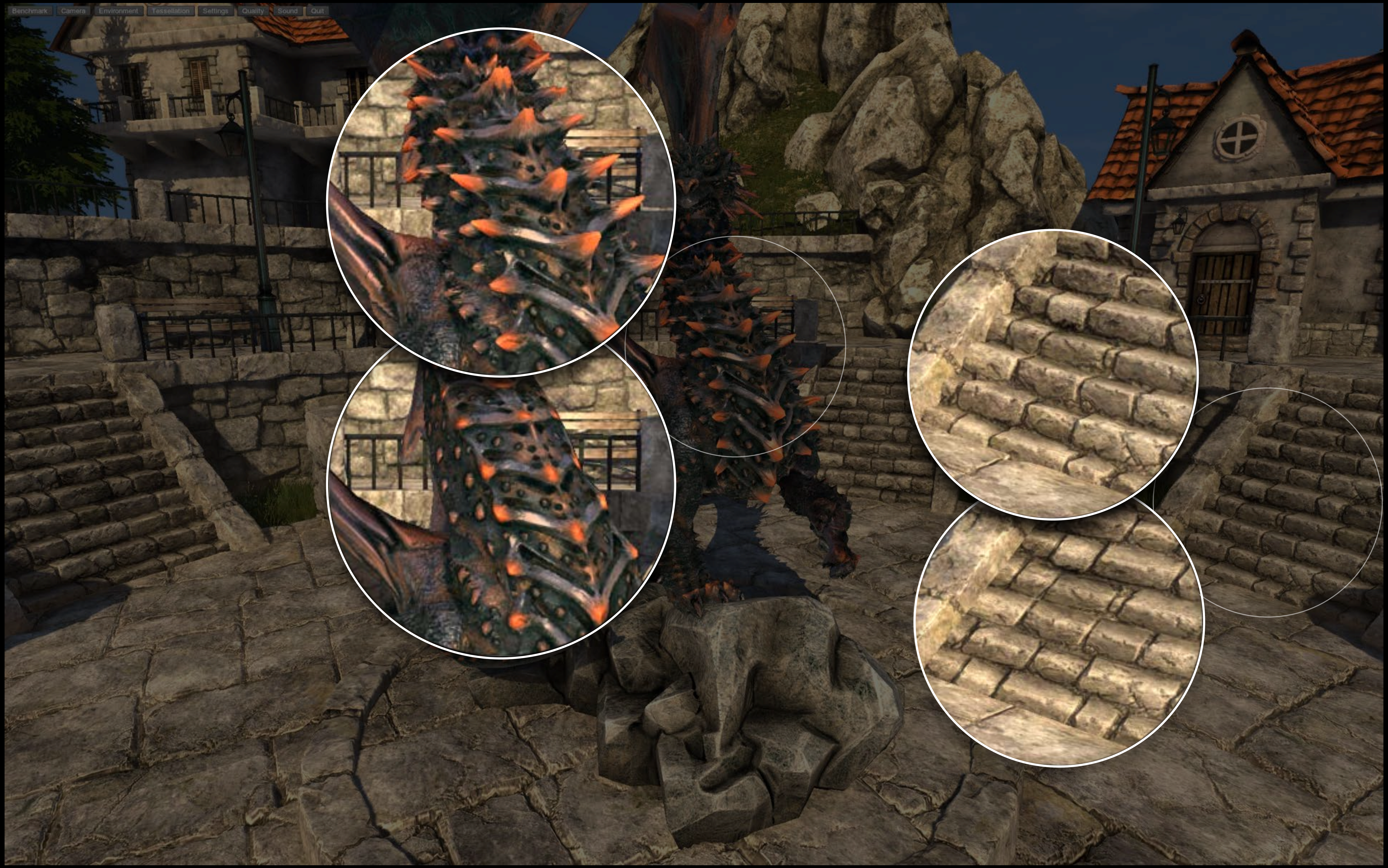
Tessellation Shaders

Unigine Heaven 4.0



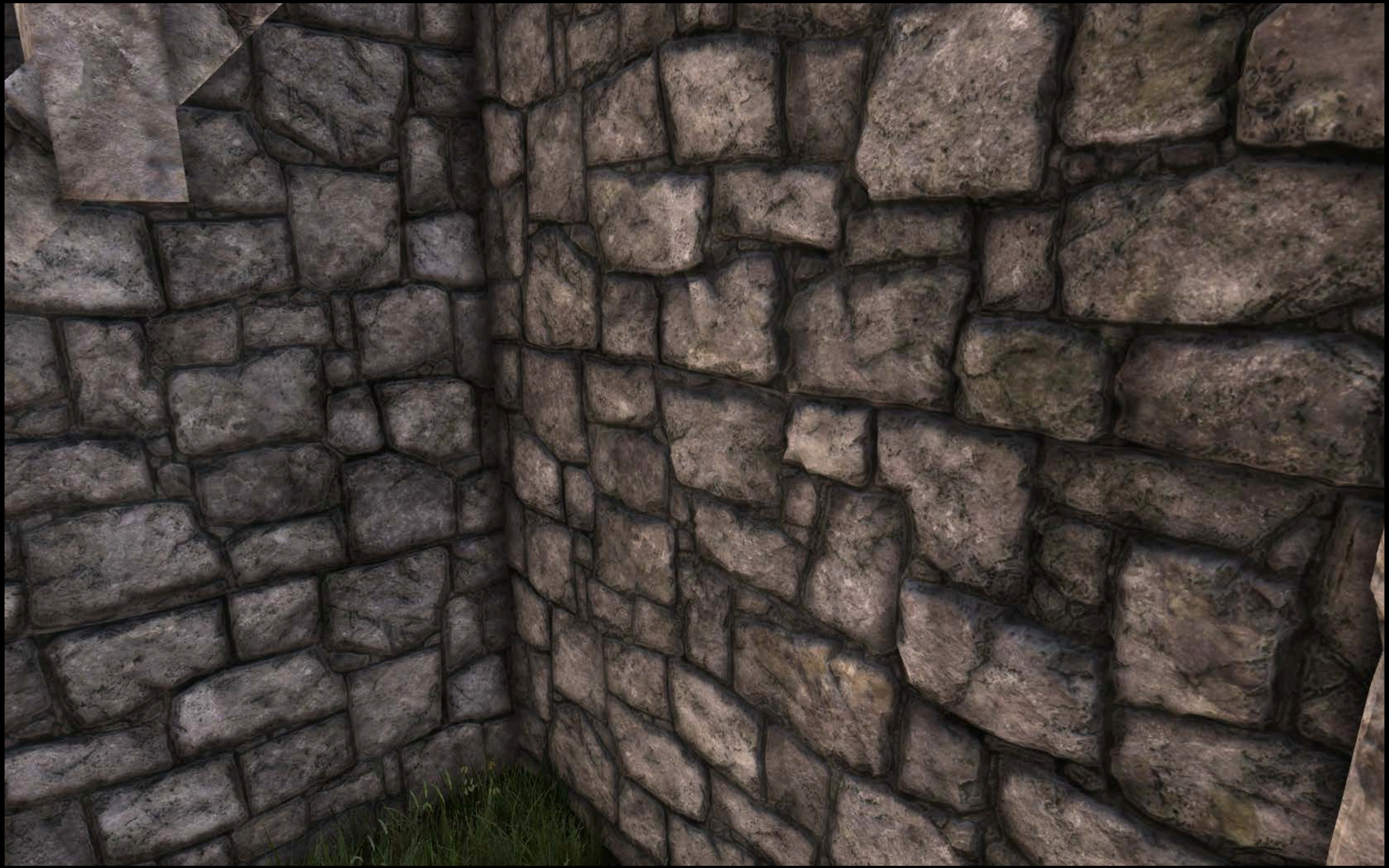
Tessellation Shaders

Unigine Heaven 4.0



Tessellation Shaders

Dynamically generate geometry



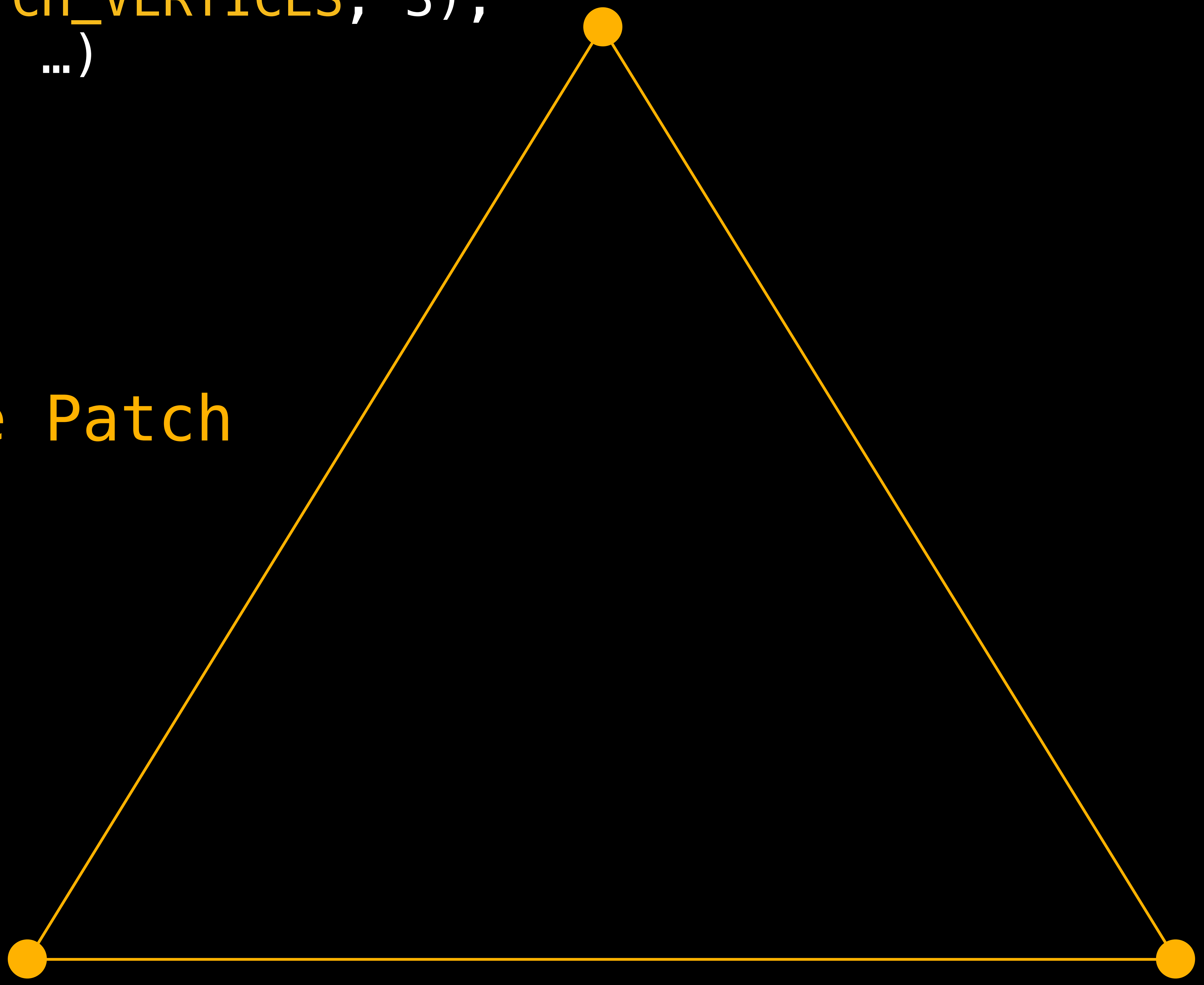
Tessellation Shaders

Dynamically generate geometry

How Tessellation Works

Start with a patch

```
glPatchParameteri(GL_PATCH_VERTICES, 3);
glDrawArrays(GL_PATCHES, …)
```

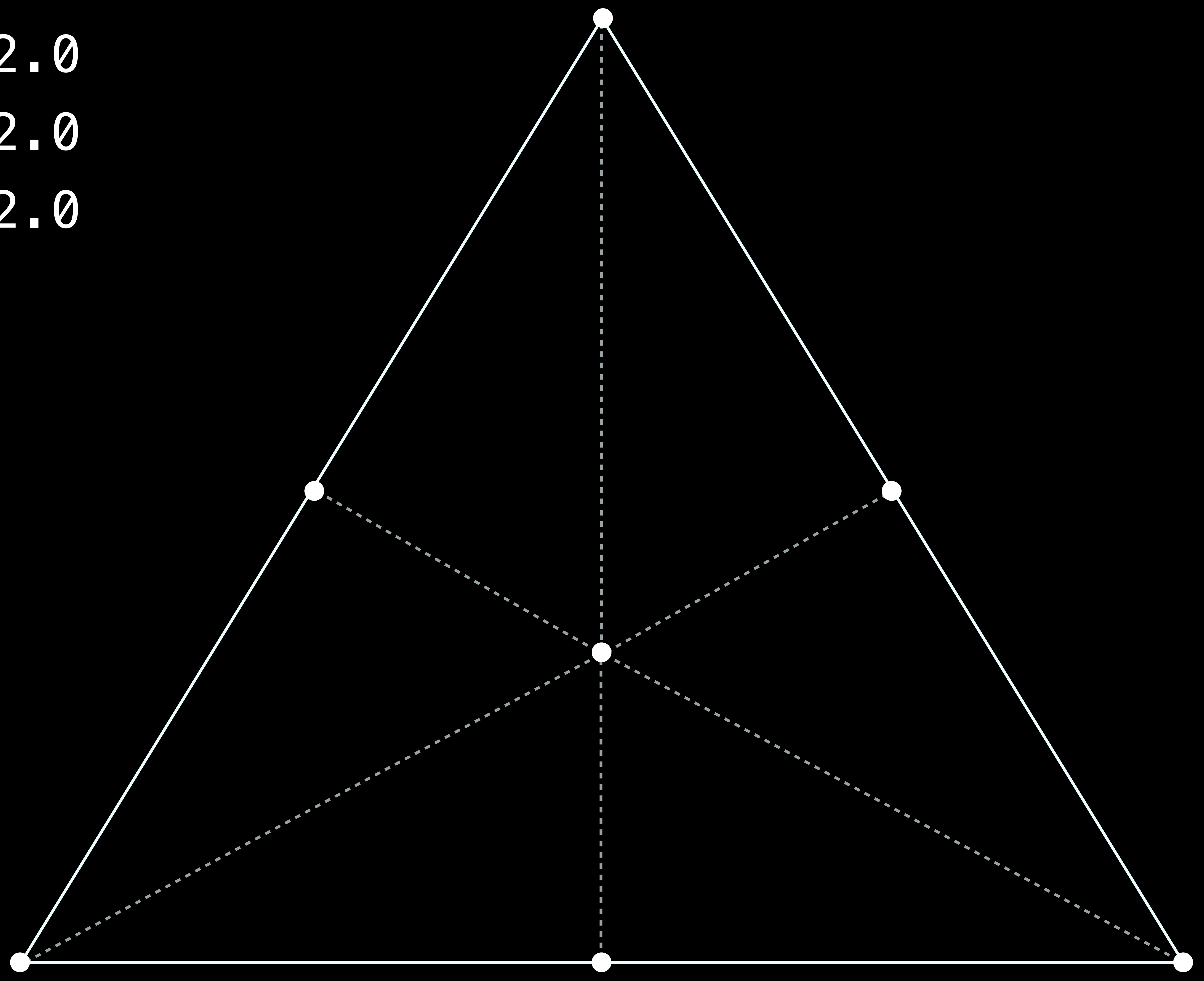


Triangle Patch

How Tessellation Works

Set the outer tessellation levels

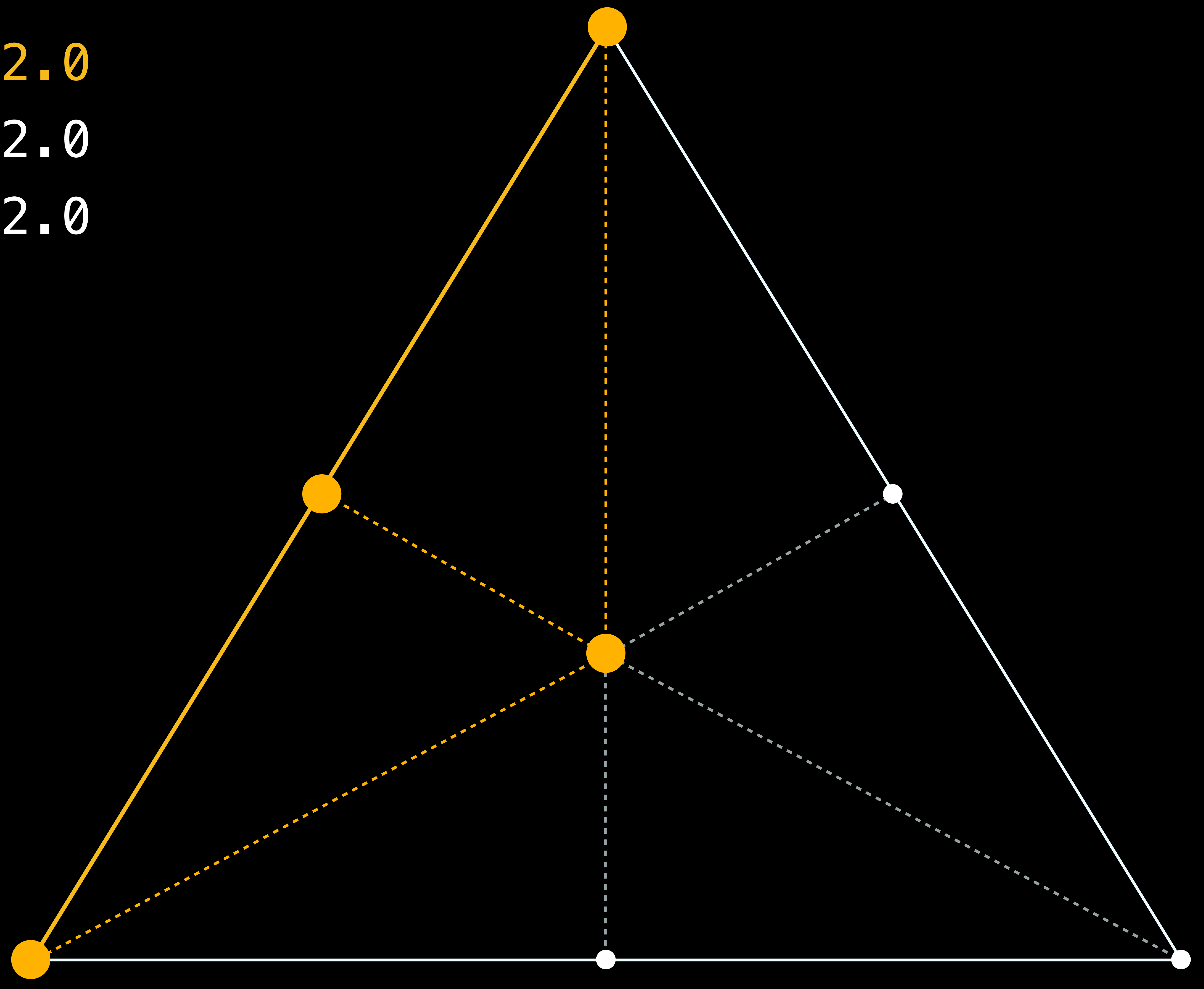
```
gl_TessLevelOuter[0] = 2.0
gl_TessLevelOuter[1] = 2.0
gl_TessLevelOuter[2] = 2.0
```



How Tessellation Works

Set the outer tessellation levels

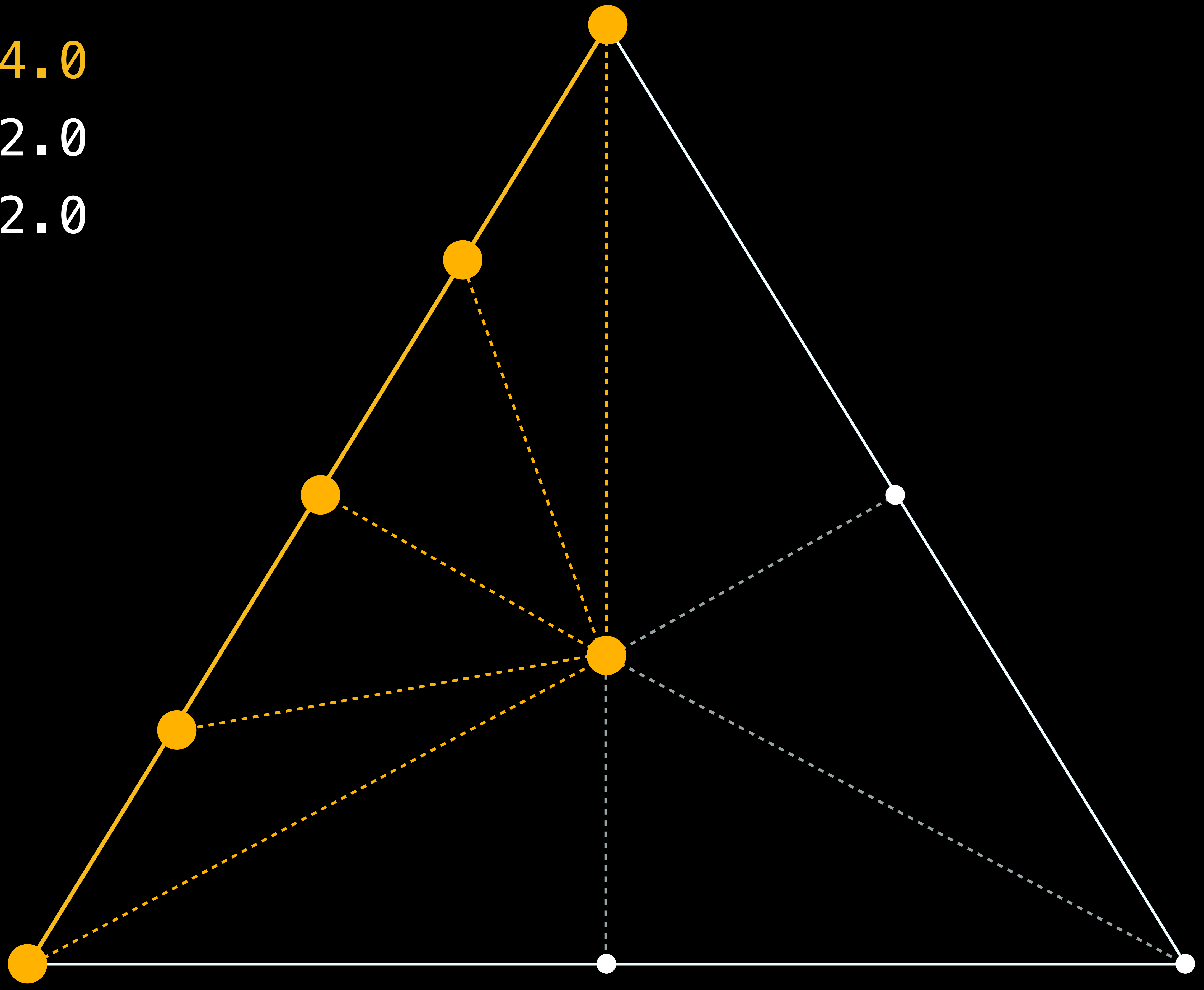
```
gl_TessLevelOuter[0] = 2.0
gl_TessLevelOuter[1] = 2.0
gl_TessLevelOuter[2] = 2.0
```



How Tessellation Works

Set the outer tessellation levels

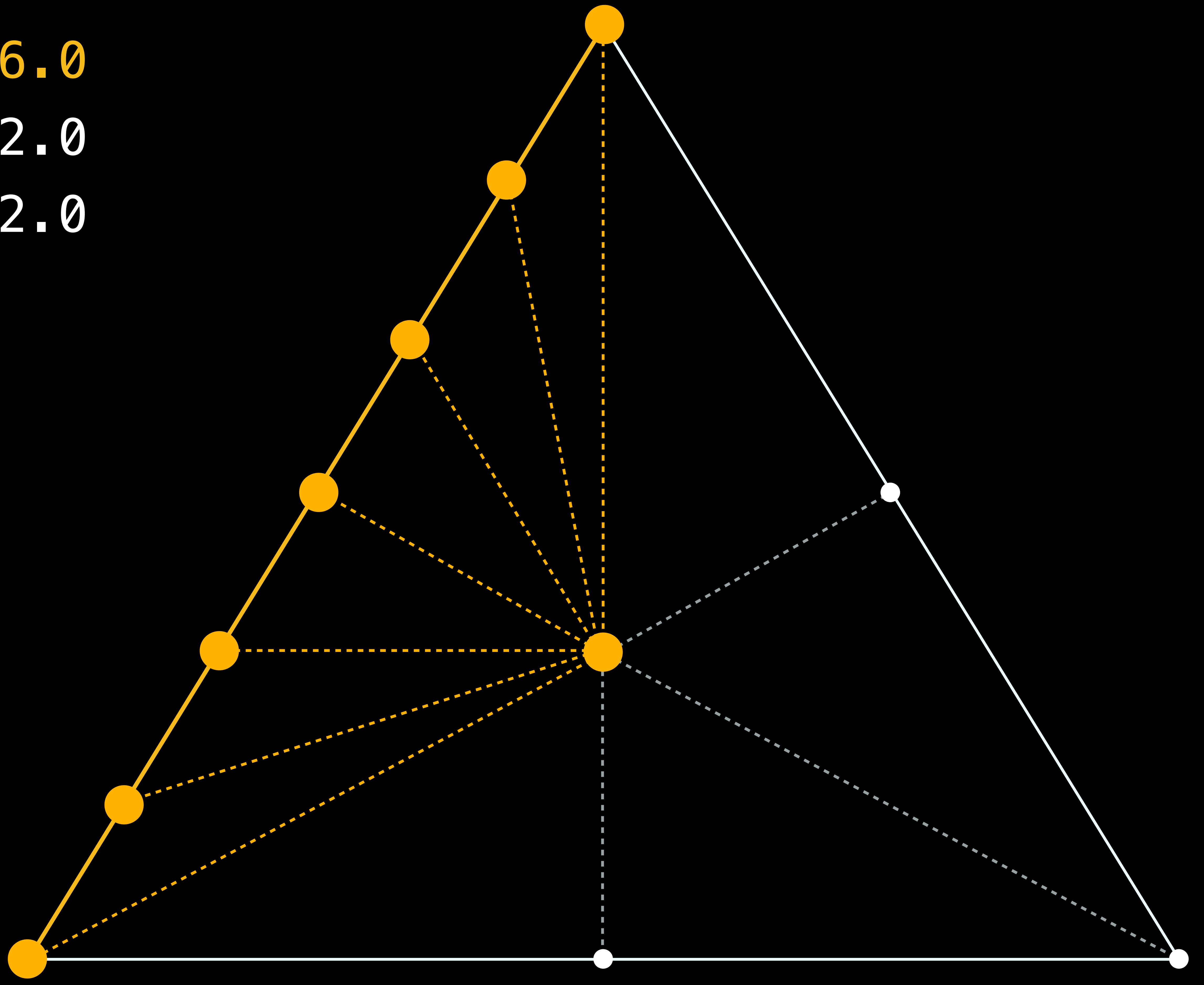
```
gl_TessLevelOuter[0] = 4.0
gl_TessLevelOuter[1] = 2.0
gl_TessLevelOuter[2] = 2.0
```



How Tessellation Works

Set the outer tessellation levels

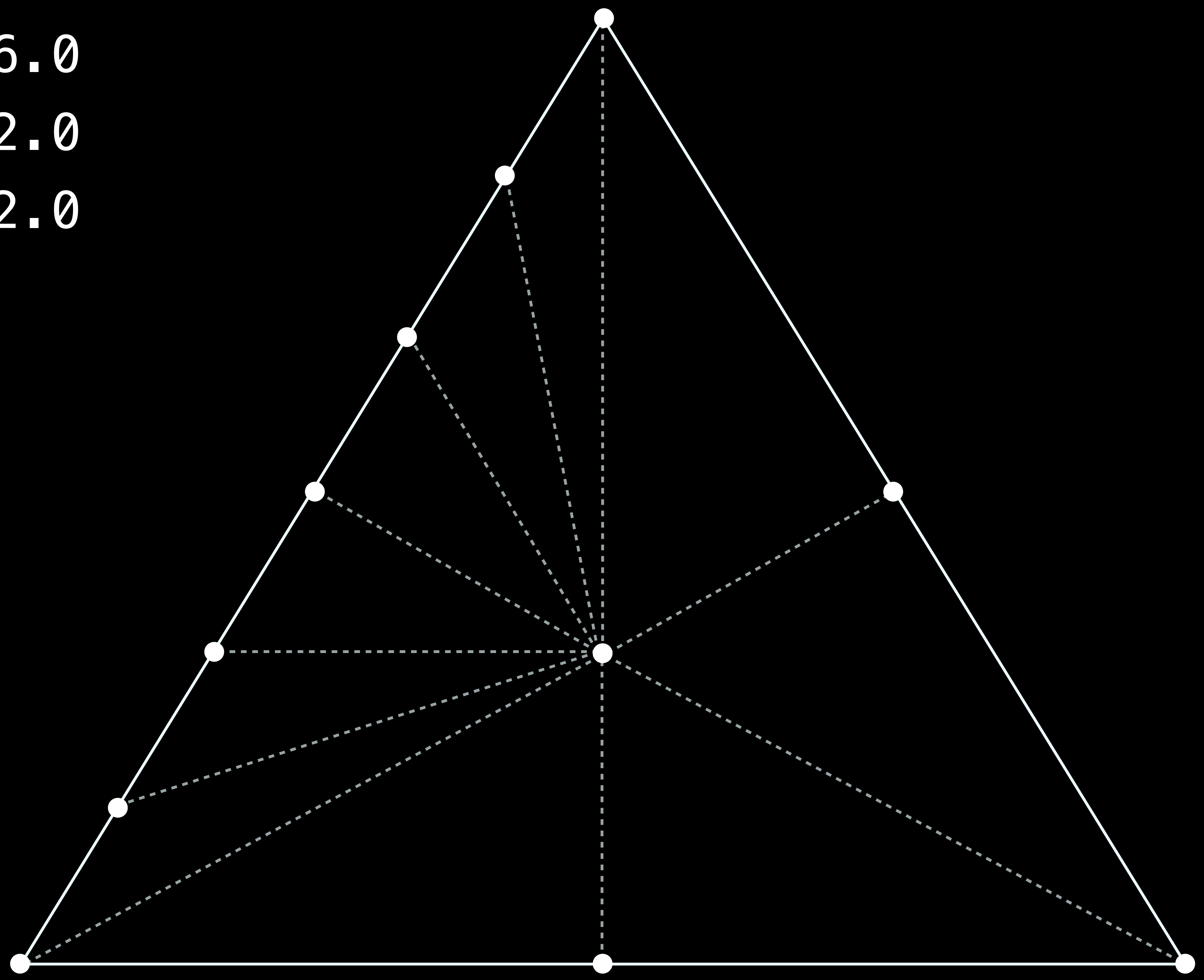
```
gl_TessLevelOuter[0] = 6.0
gl_TessLevelOuter[1] = 2.0
gl_TessLevelOuter[2] = 2.0
```



How Tessellation Works

Set the outer tessellation levels

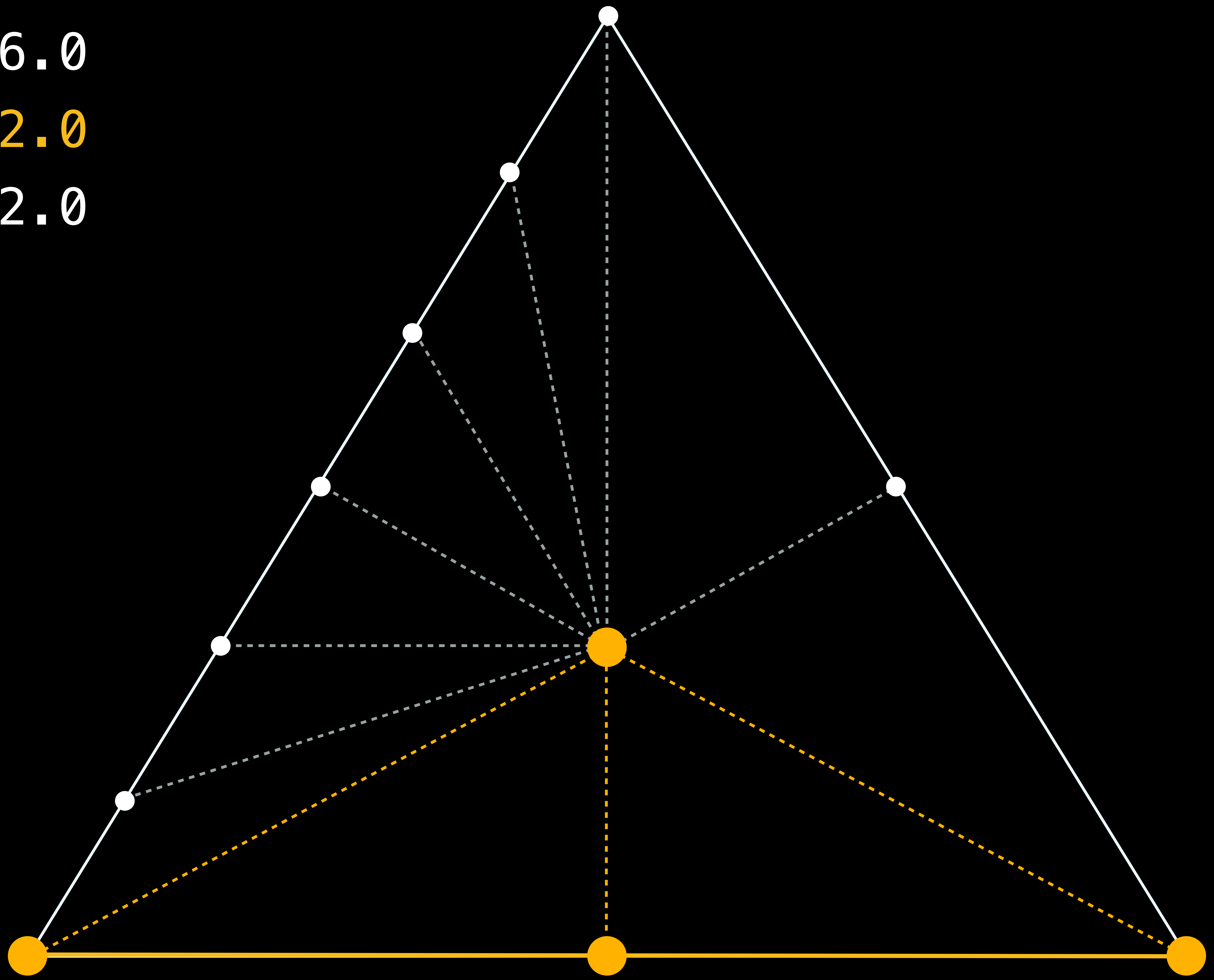
```
gl_TessLevelOuter[0] = 6.0
gl_TessLevelOuter[1] = 2.0
gl_TessLevelOuter[2] = 2.0
```



How Tessellation Works

Set the outer tessellation levels

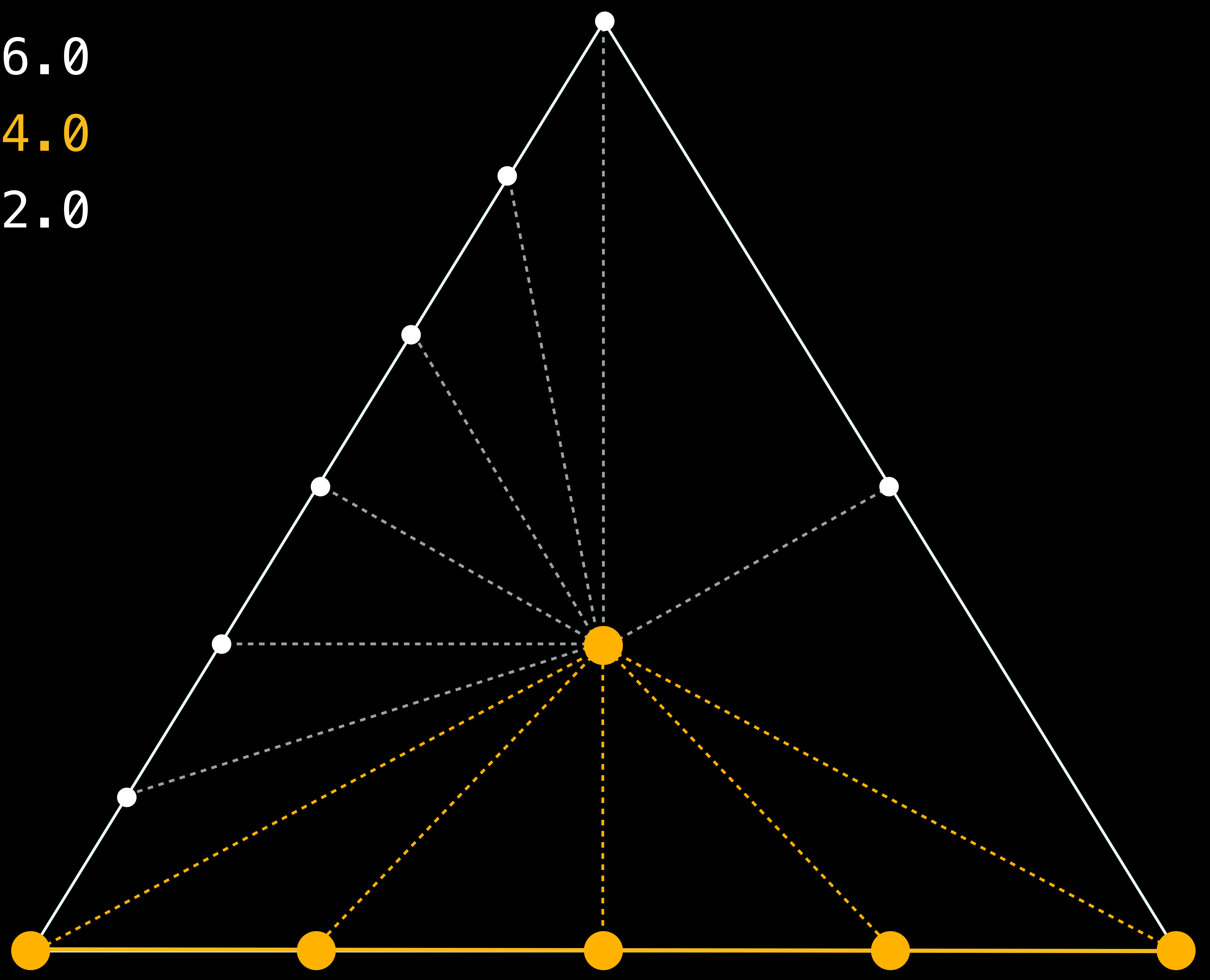
```
gl_TessLevelOuter[0] = 6.0
gl_TessLevelOuter[1] = 2.0
gl_TessLevelOuter[2] = 2.0
```



How Tessellation Works

Set the outer tessellation levels

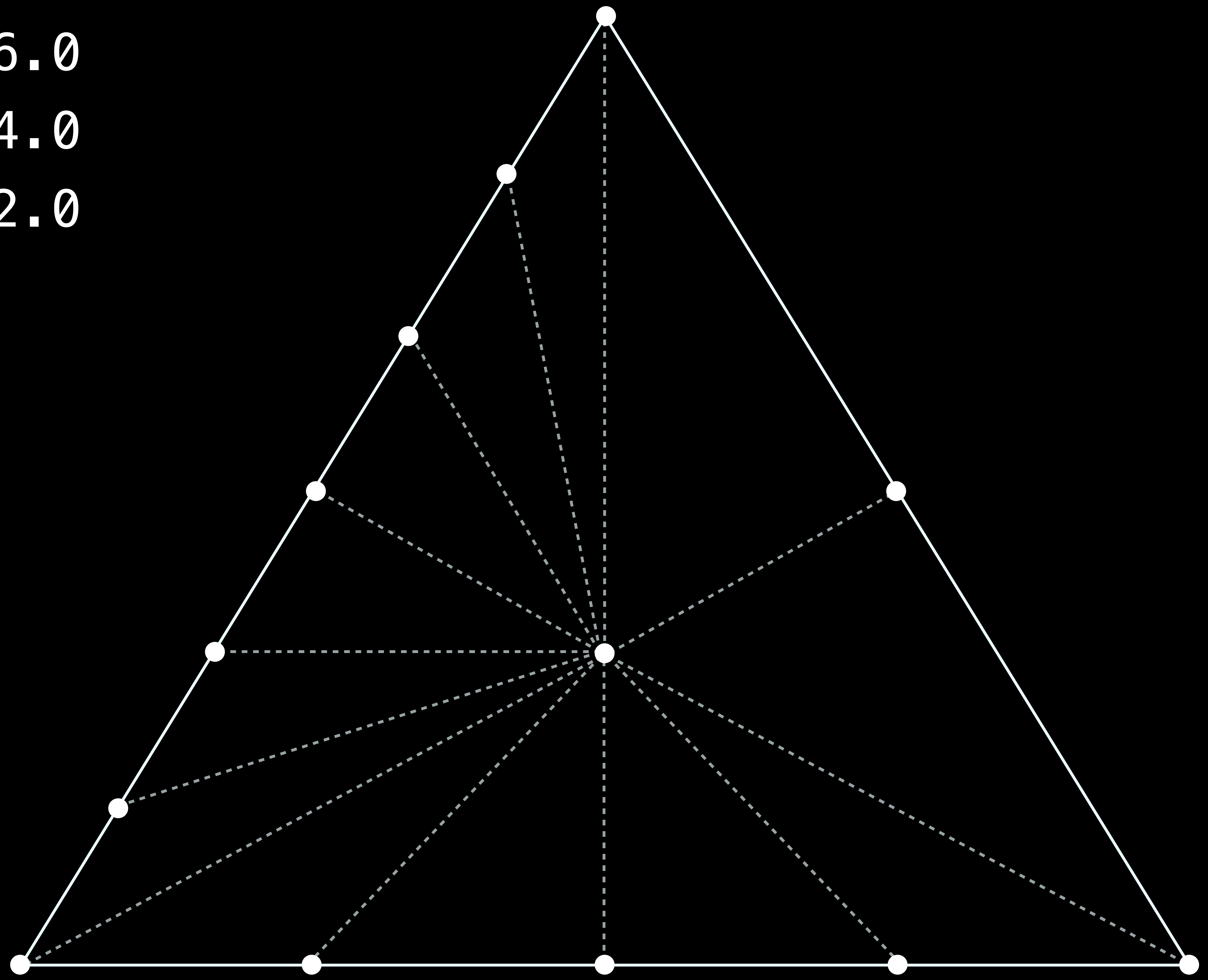
```
gl_TessLevelOuter[0] = 6.0
gl_TessLevelOuter[1] = 4.0
gl_TessLevelOuter[2] = 2.0
```



How Tessellation Works

Set the outer tessellation levels

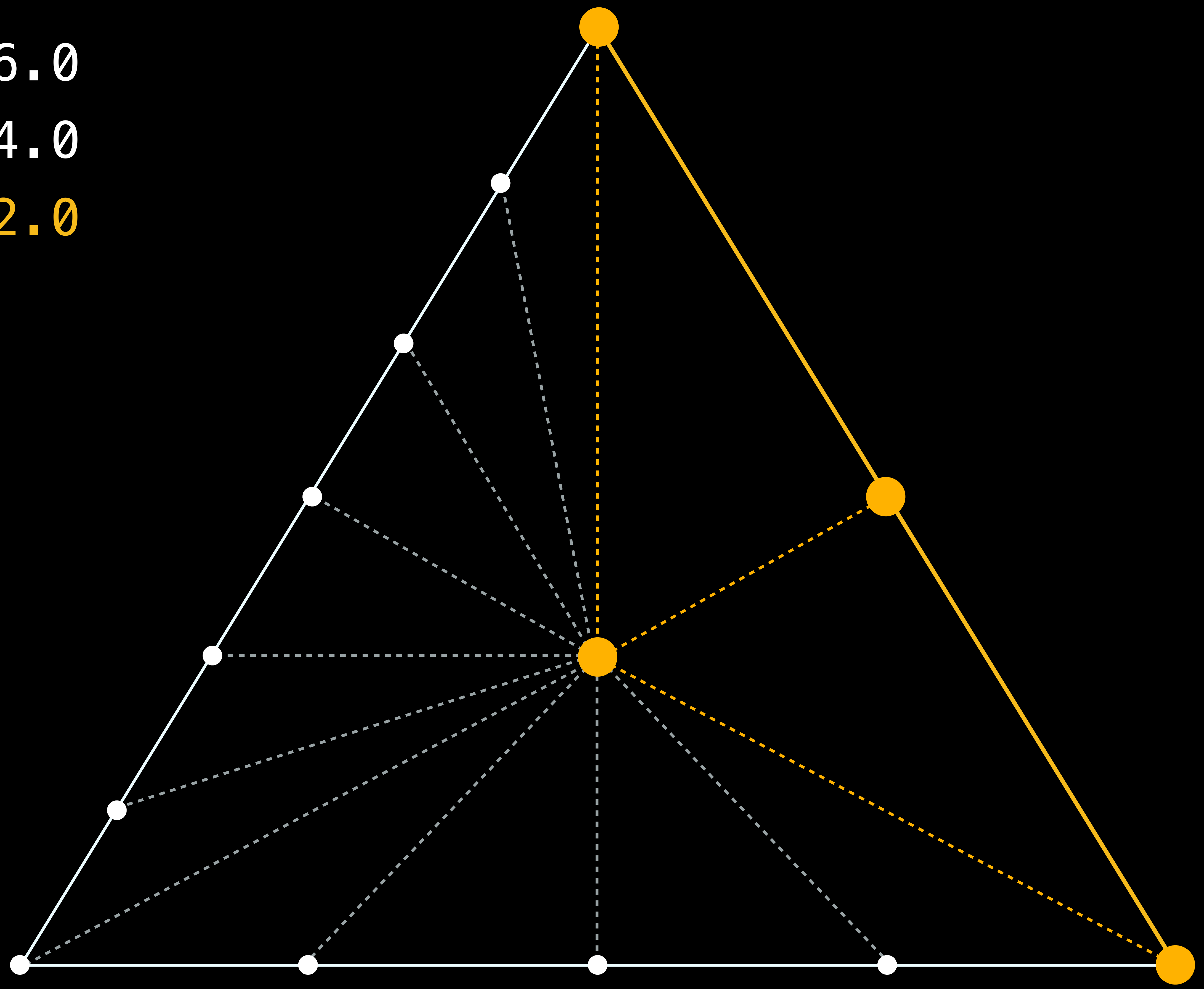
```
gl_TessLevelOuter[0] = 6.0
gl_TessLevelOuter[1] = 4.0
gl_TessLevelOuter[2] = 2.0
```



How Tessellation Works

Set the outer tessellation levels

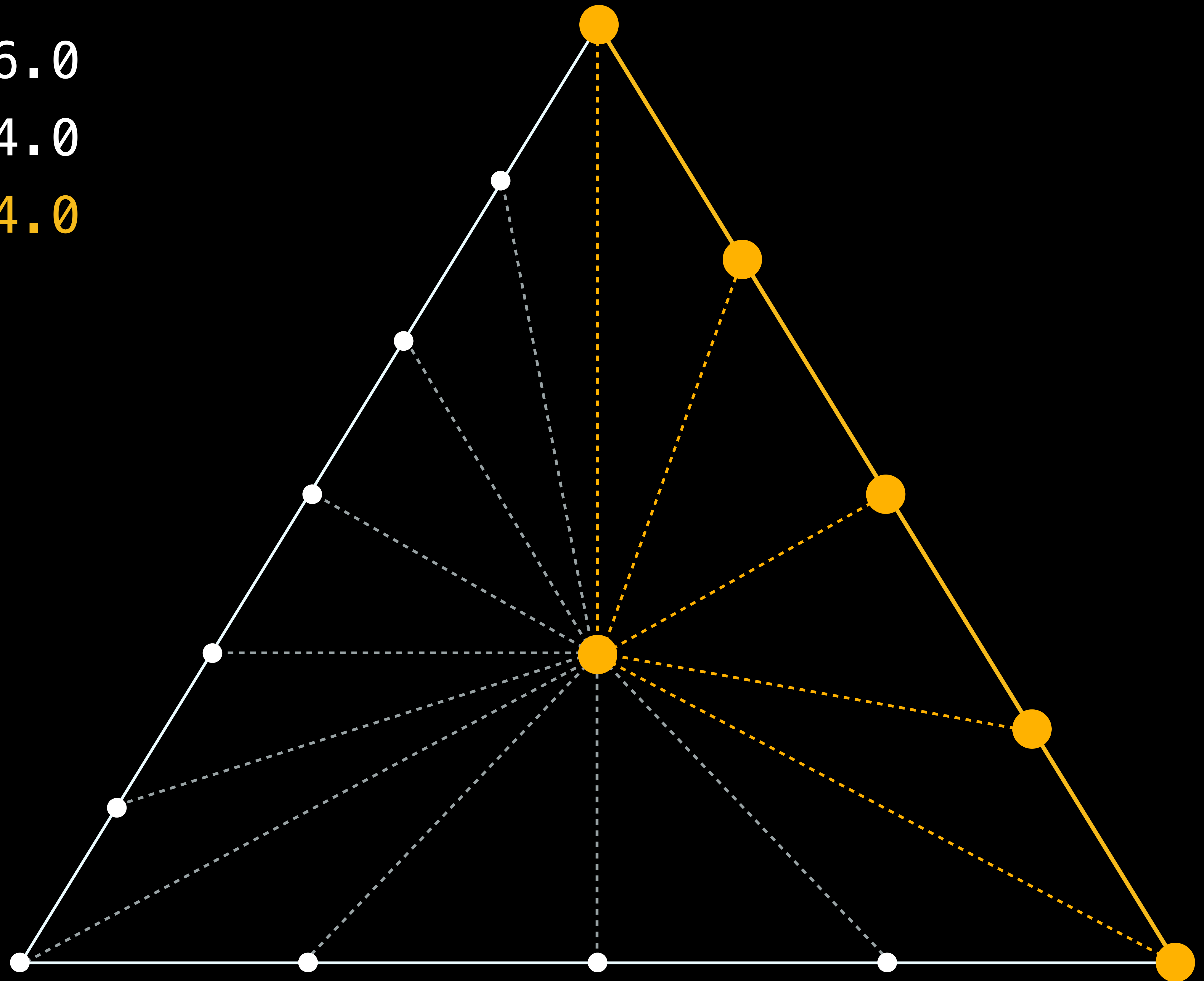
```
gl_TessLevelOuter[0] = 6.0
gl_TessLevelOuter[1] = 4.0
gl_TessLevelOuter[2] = 2.0
```



How Tessellation Works

Set the outer tessellation levels

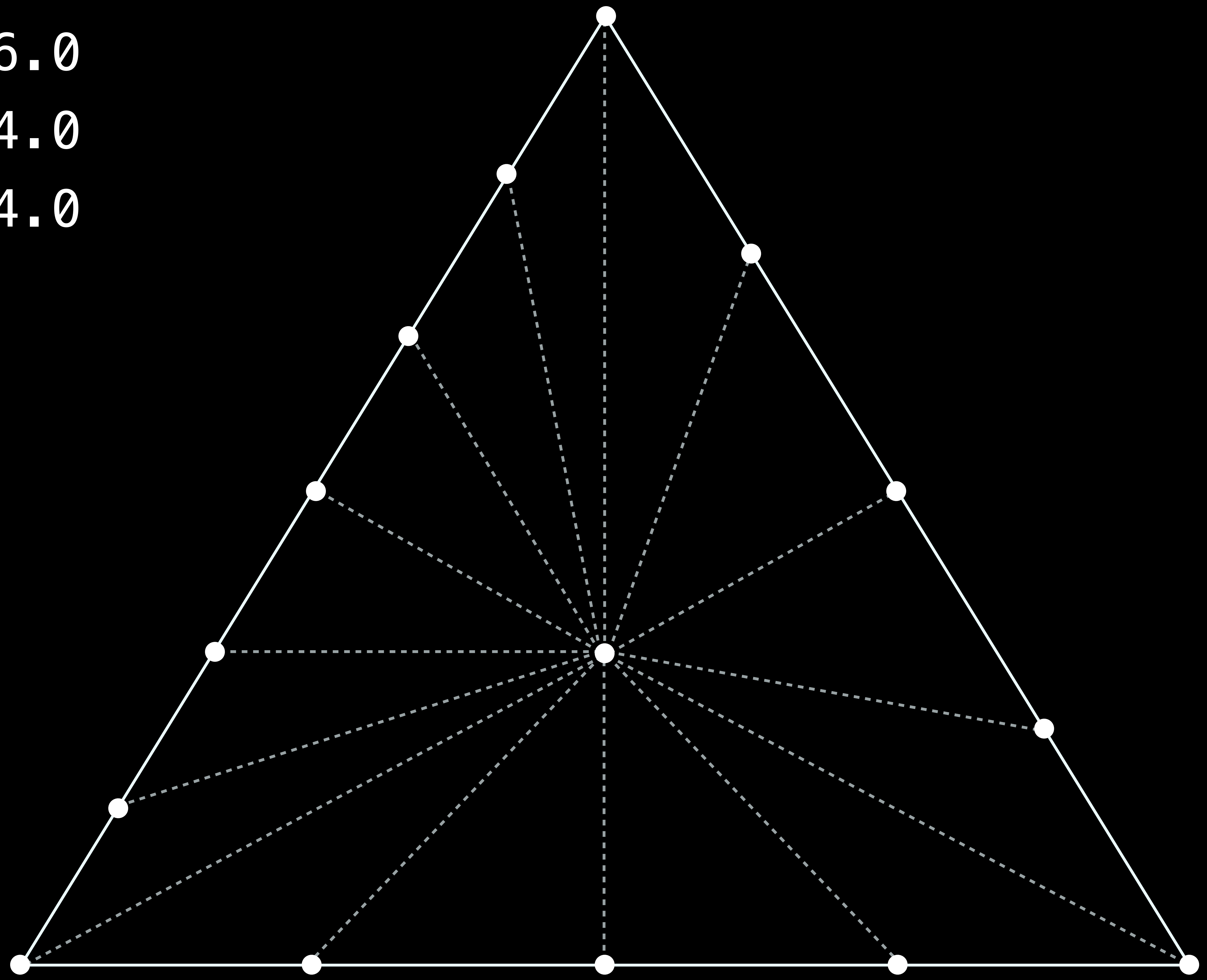
```
gl_TessLevelOuter[0] = 6.0
gl_TessLevelOuter[1] = 4.0
gl_TessLevelOuter[2] = 4.0
```



How Tessellation Works

Set the outer tessellation levels

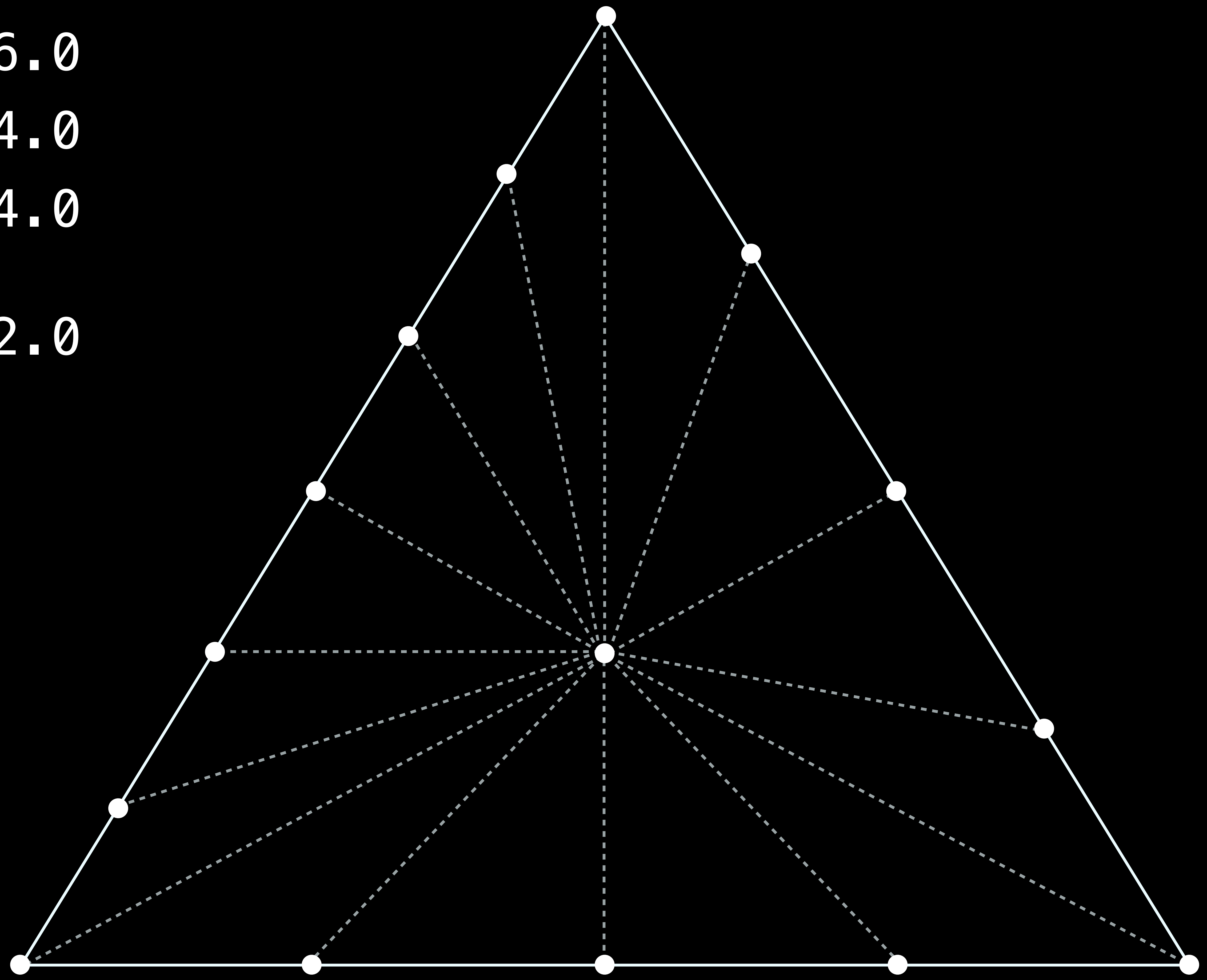
```
gl_TessLevelOuter[0] = 6.0
gl_TessLevelOuter[1] = 4.0
gl_TessLevelOuter[2] = 4.0
```



How Tessellation Works

Set the inner tessellation levels

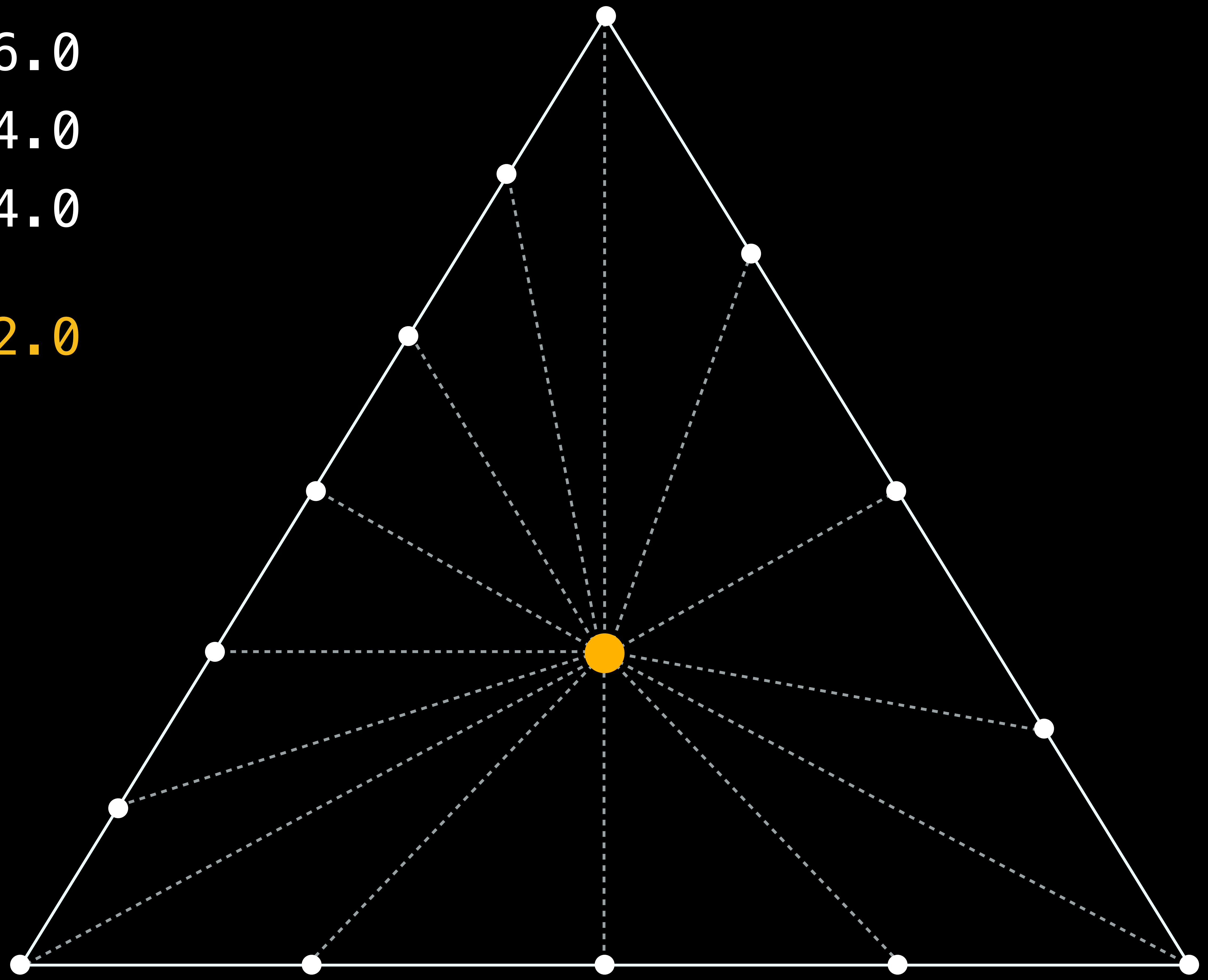
```
gl_TessLevelOuter[0] = 6.0
gl_TessLevelOuter[1] = 4.0
gl_TessLevelOuter[2] = 4.0

gl_TessLevelInner[0] = 2.0
```



How Tessellation Works

Set the inner tessellation levels

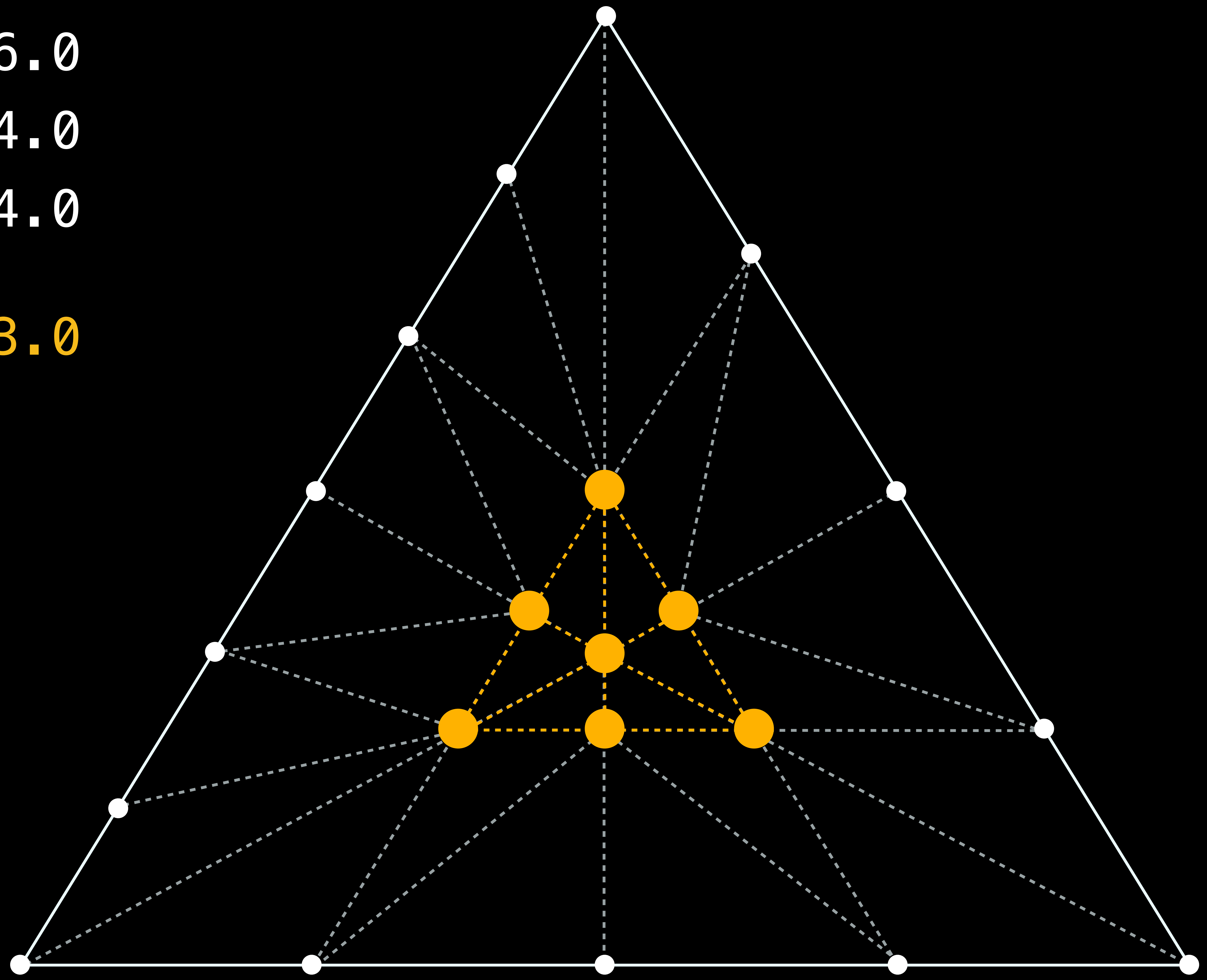
```
gl_TessLevelOuter[0] = 6.0
gl_TessLevelOuter[1] = 4.0
gl_TessLevelOuter[2] = 4.0

gl_TessLevelInner[0] = 2.0
```



How Tessellation Works

Set the inner tessellation levels

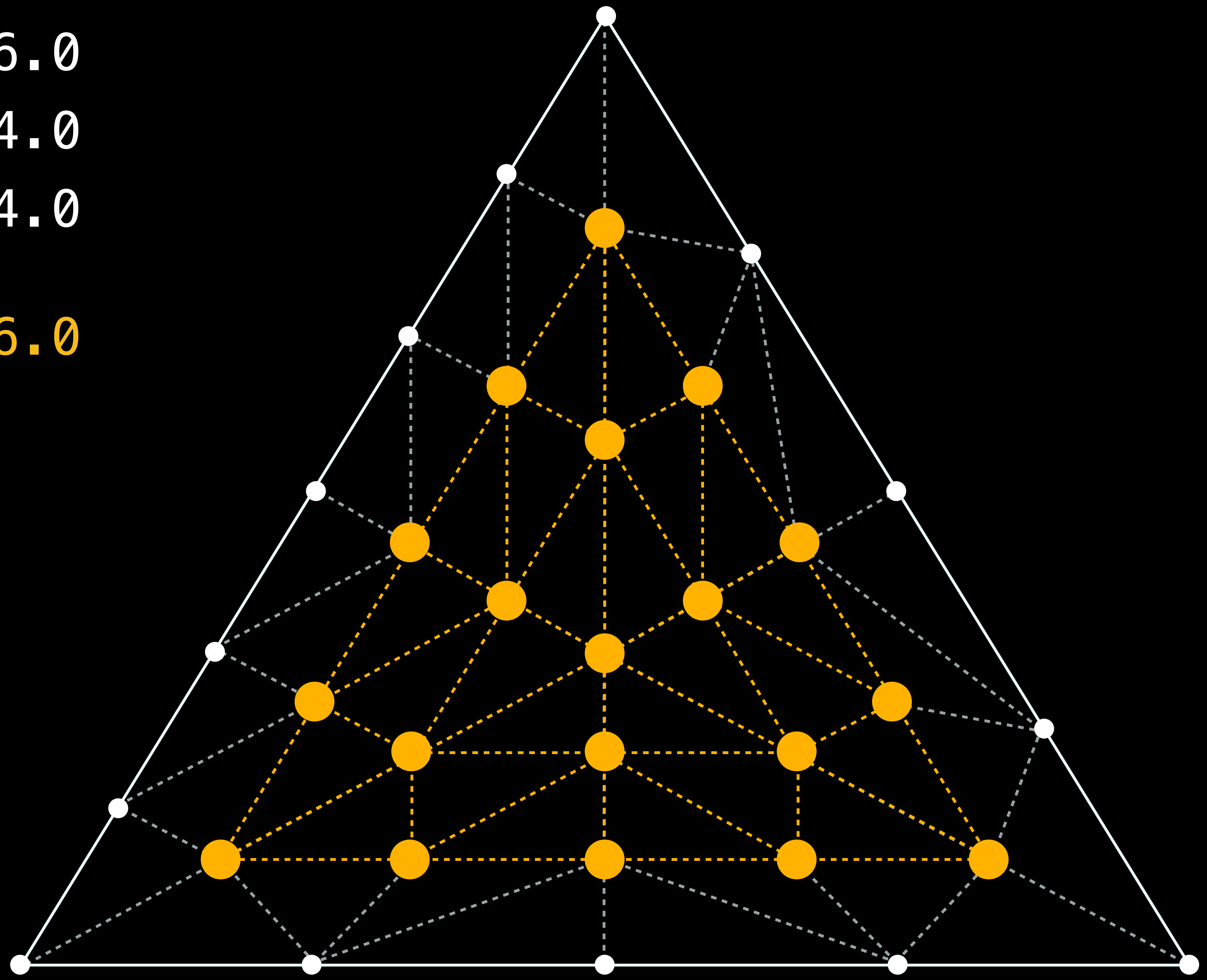
```
gl_TessLevelOuter[0] = 6.0
gl_TessLevelOuter[1] = 4.0
gl_TessLevelOuter[2] = 4.0

gl_TessLevelInner[0] = 3.0
```



How Tessellation Works

Set the inner tessellation levels

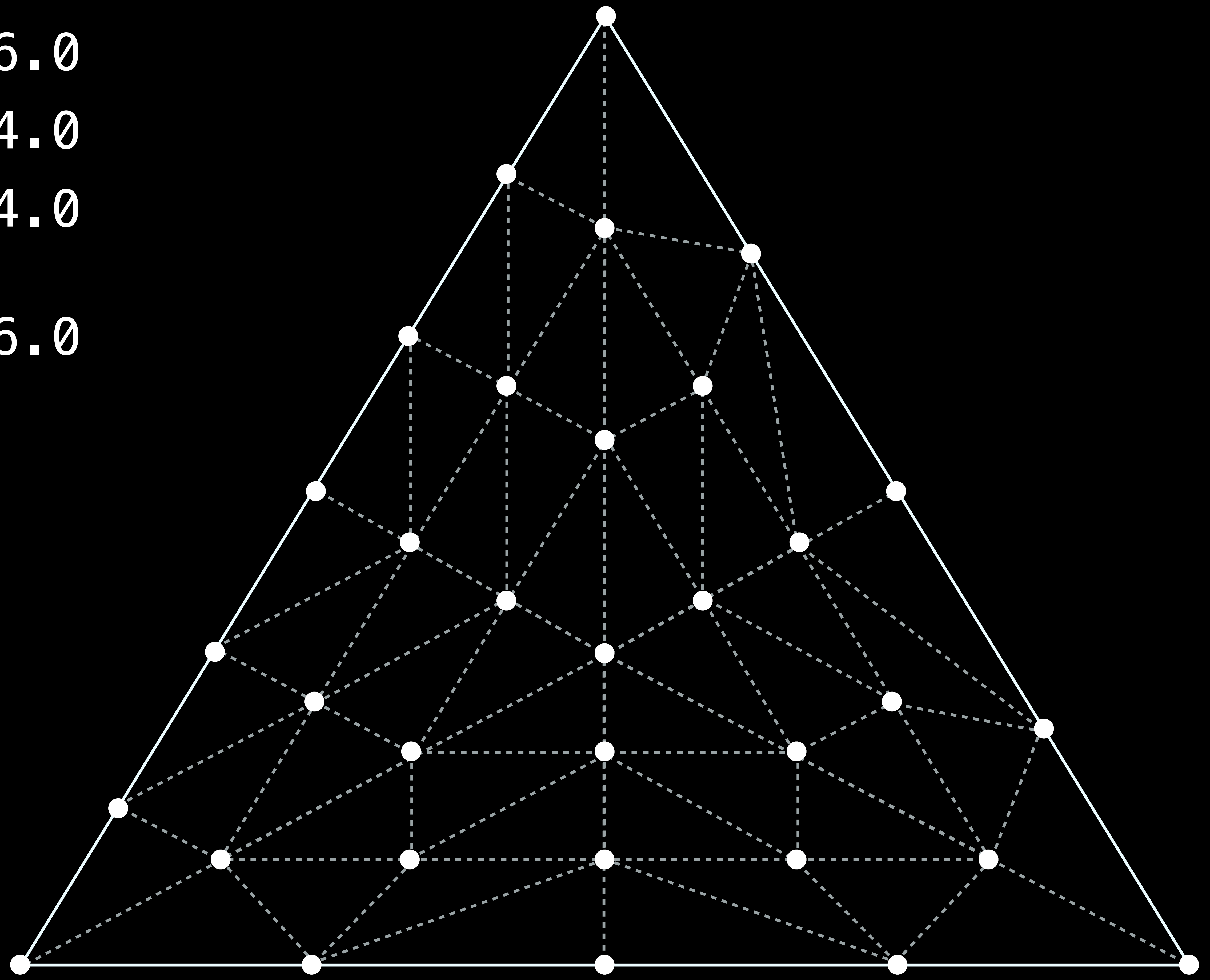
```
gl_TessLevelOuter[0] = 6.0
gl_TessLevelOuter[1] = 4.0
gl_TessLevelOuter[2] = 4.0

gl_TessLevelInner[0] = 6.0
```



How Tessellation Works

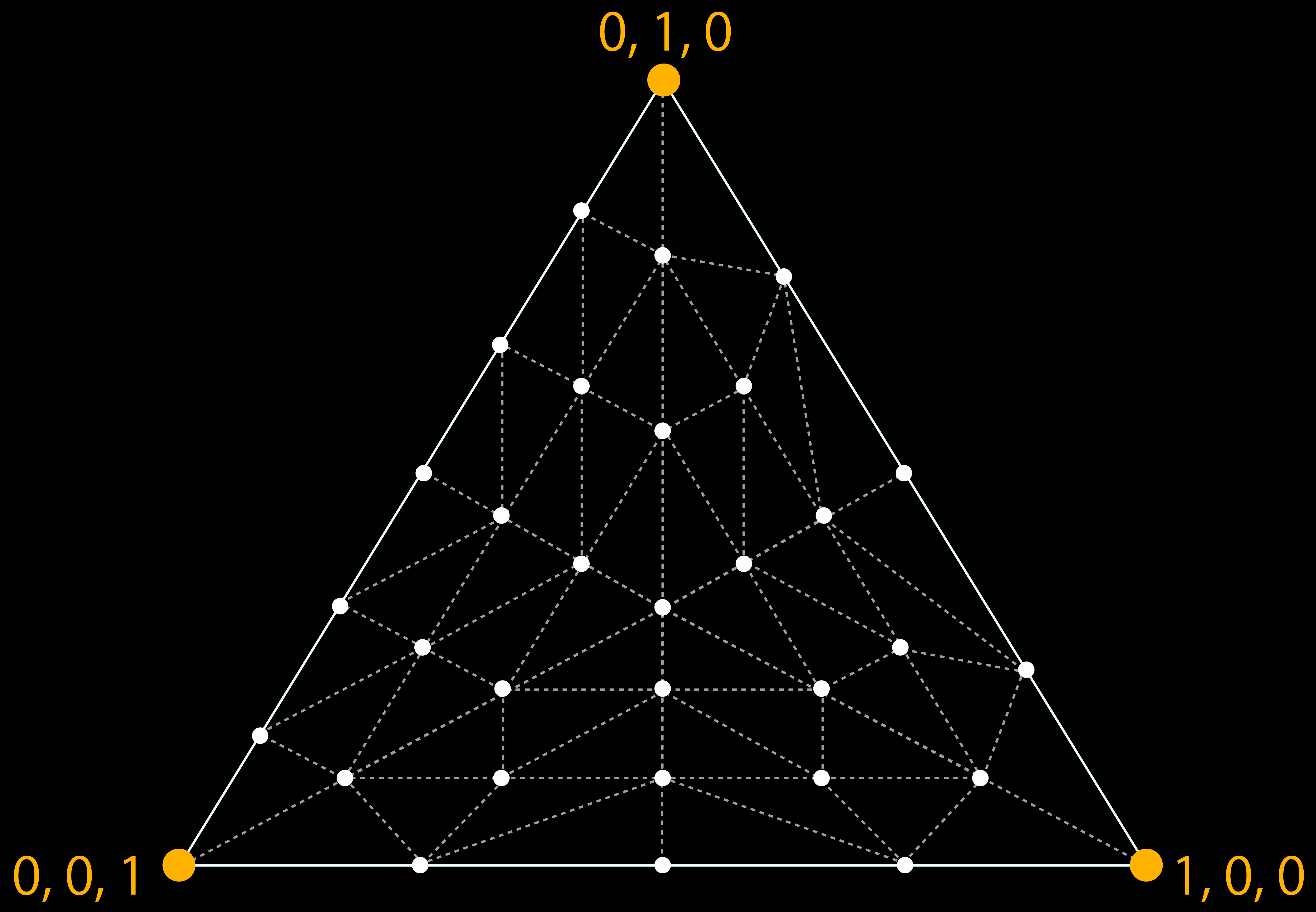
Evaluate vertex attributes into tessellated primitives

```
gl_TessLevelOuter[0] = 6.0
gl_TessLevelOuter[1] = 4.0
gl_TessLevelOuter[2] = 4.0

gl_TessLevelInner[0] = 6.0
```



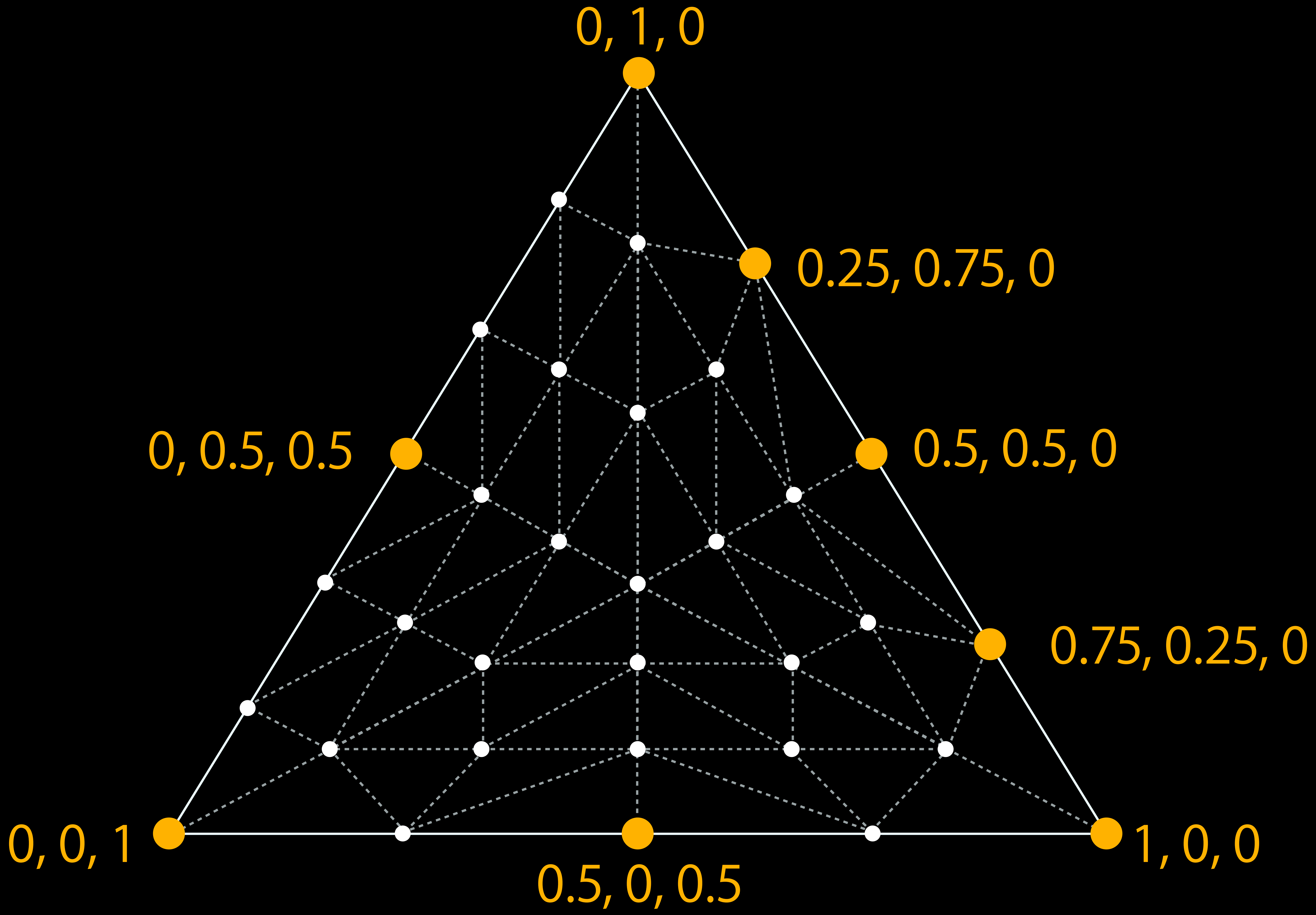
How Tessellation Works

Evaluate vertex attributes into tessellated primitives



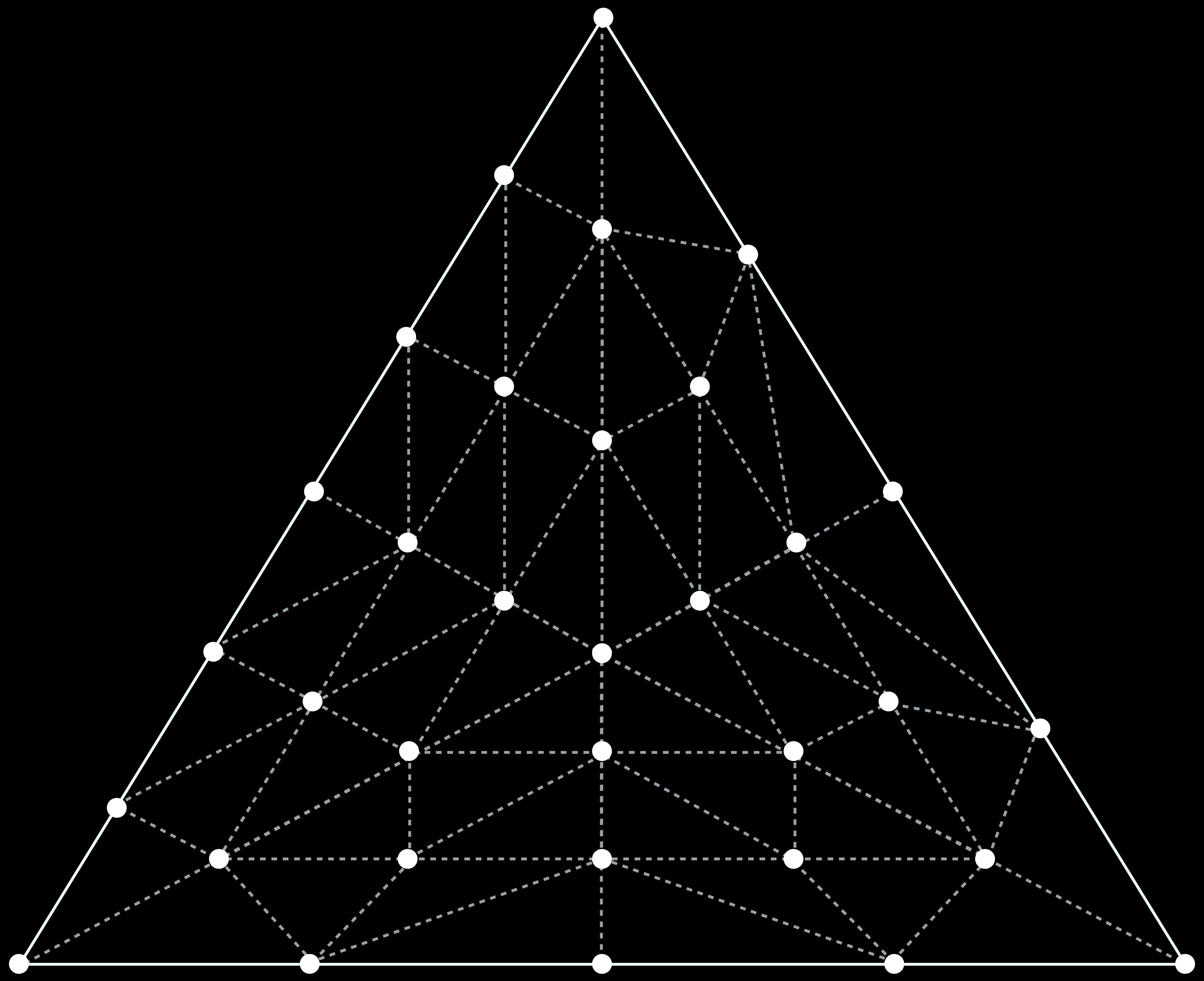
How Tessellation Works

Evaluate vertex attributes into tessellated primitives



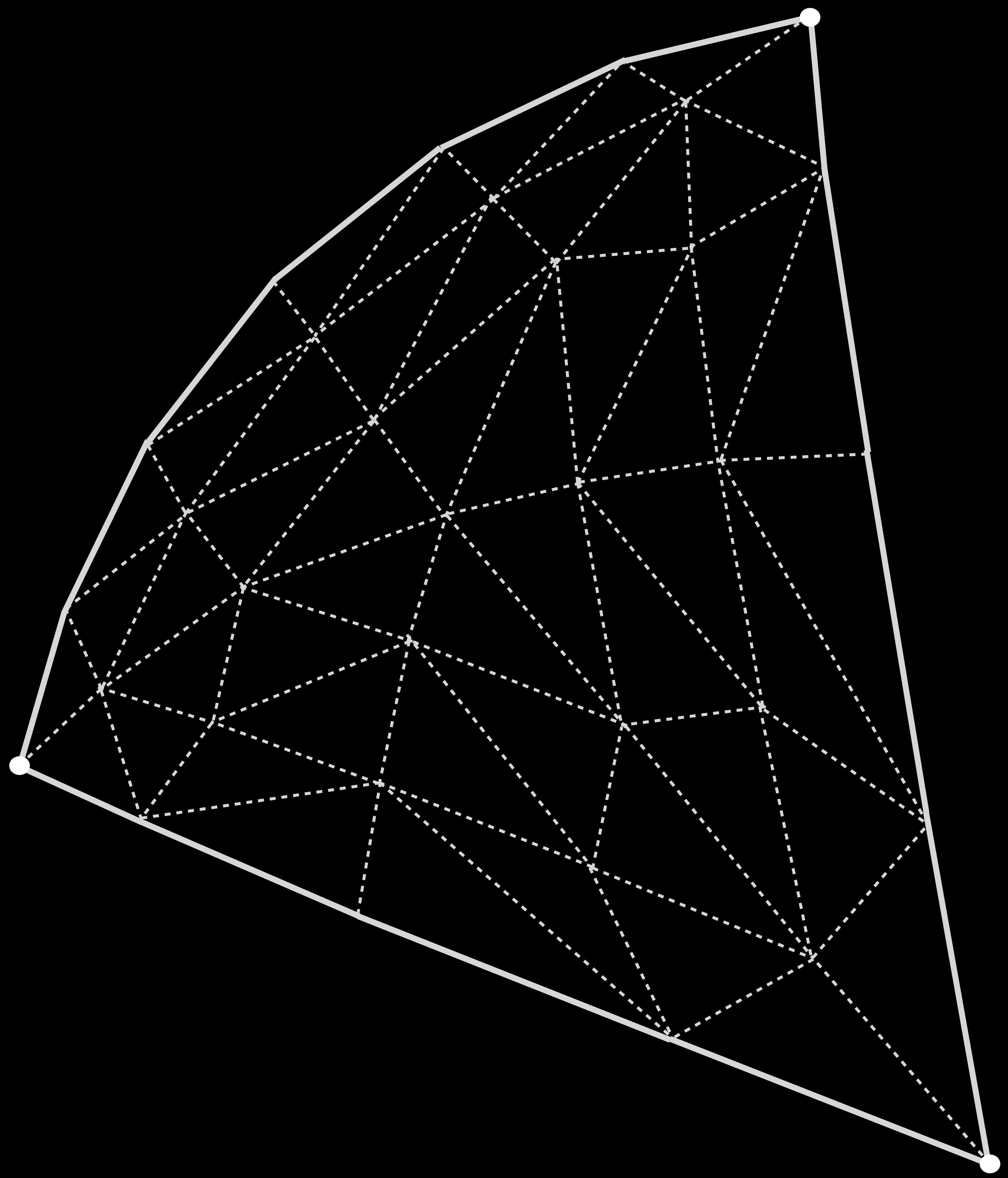
How Tessellation Works

Evaluate vertex attributes into tessellated primitives



How Tessellation Works

Evaluate vertex attributes into tessellated primitives



OpenGL 4 Pipeline

Vertex Shader
Tessellation Shaders
Geometry Shader
Fragment Shader

OpenGL 4 Pipeline

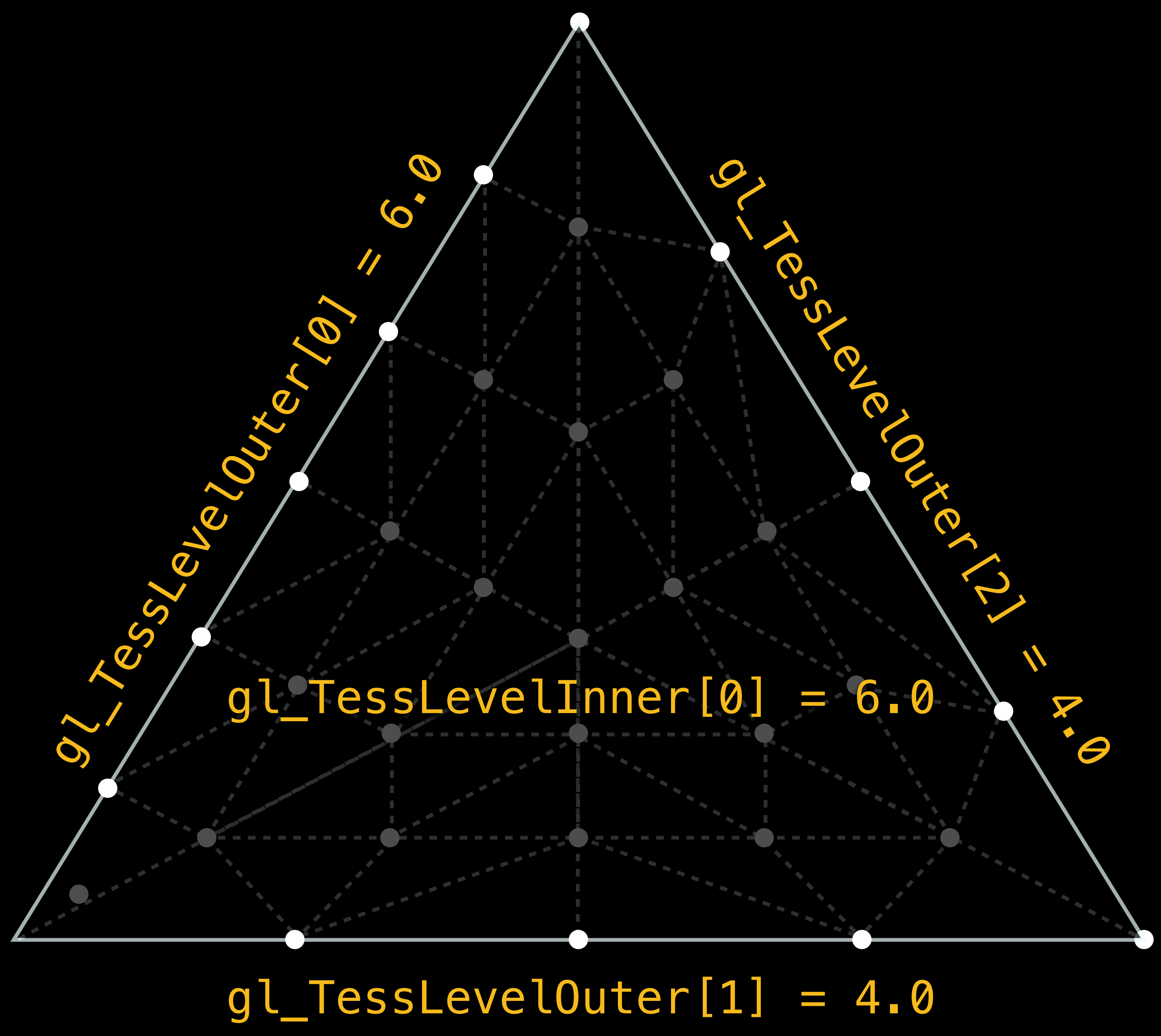
Vertex Shader
Tessellation Shaders
Geometry Shader
Fragment Shader

OpenGL 4 Pipeline

Vertex Shader
Tessellation Control Shader
Primitive Generator
Tessellation Evaluation Shader
Tessellation Stages
Geometry Shader

Tessellation Control Shader

- "Controls" how much to tessellate a patch
- Inputs:
  - `GL_PATCHES` from vertex shader
  - Array of control points
- Outputs:
  - `gl_TessLevelOuter[] = { 6.0, 4.0, 4.0, .. }`
  - `gl_TessLevelInner[] = { 6.0, .. }`


gl_TessLevelOuter[0] = 6.0
gl_TessLevelOuter[2] = 4.0
gl_TessLevelInner[0] = 6.0
gl_TessLevelOuter[1] = 4.0

- Tip: Match `gl_TessLevelOuter` on adjacent patches for crack-free tessellation

Control Shader

Triangle control shader example

```
#version 400

layout(vertices=3) out;  // Using 3 control points for triangle

in  vec4 vPos[3];
out vec4 ctrlPos[3];

void main ()
{
    ctrlPos[gl_InvocationID] = vPos[gl_InvocationID];

    if(gl_InvocationID == 0) {
        gl_TessLevelOuter[0] = MyCalcOuterLOD(vPos[0], vPos[1]);
        gl_TessLevelOuter[1] = MyCalcOuterLOD(vPos[1], vPos[2]);
        gl_TessLevelOuter[2] = MyCalcOuterLOD(vPos[2], vPos[0]);

        gl_TessLevelInner[0] = MyCalcInnerLOD(vPos[0], vPos[1], vPos[2]);
    }
}
```

Control Shader

Triangle control shader example

```
#version 400
layout(vertices=3) out;  // Using 3 control points for triangle

in  vec4 vPos[3];
out vec4 ctrlPos[3];
void main ()
{
   ctrlPos[gl_InvocationID] = vPos[gl_InvocationID];

   if(gl_InvocationID == 0) {
      gl_TessLevelOuter[0] = MyCalcOuterLOD(vPos[0], vPos[1]);
      gl_TessLevelOuter[1] = MyCalcOuterLOD(vPos[1], vPos[2]);
      gl_TessLevelOuter[2] = MyCalcOuterLOD(vPos[2], vPos[0]);

      gl_TessLevelInner[0] = MyCalcInnerLOD(vPos[0], vPos[1], vPos[2]);
   }
}
```

Control Shader

Triangle control shader example

```
#version 400

layout(vertices=3) out;  // Using 3 control points for triangle

in  vec4 vPos[3];
out vec4 ctrlPos[3];

void main ()
{
   ctrlPos[gl_InvocationID] = vPos[gl_InvocationID];

   if(gl_InvocationID == 0) {
      gl_TessLevelOuter[0] = MyCalcOuterLOD(vPos[0], vPos[1]);
      gl_TessLevelOuter[1] = MyCalcOuterLOD(vPos[1], vPos[2]);
      gl_TessLevelOuter[2] = MyCalcOuterLOD(vPos[2], vPos[0]);

      gl_TessLevelInner[0] = MyCalcInnerLOD(vPos[0], vPos[1], vPos[2]);
   }
}
```

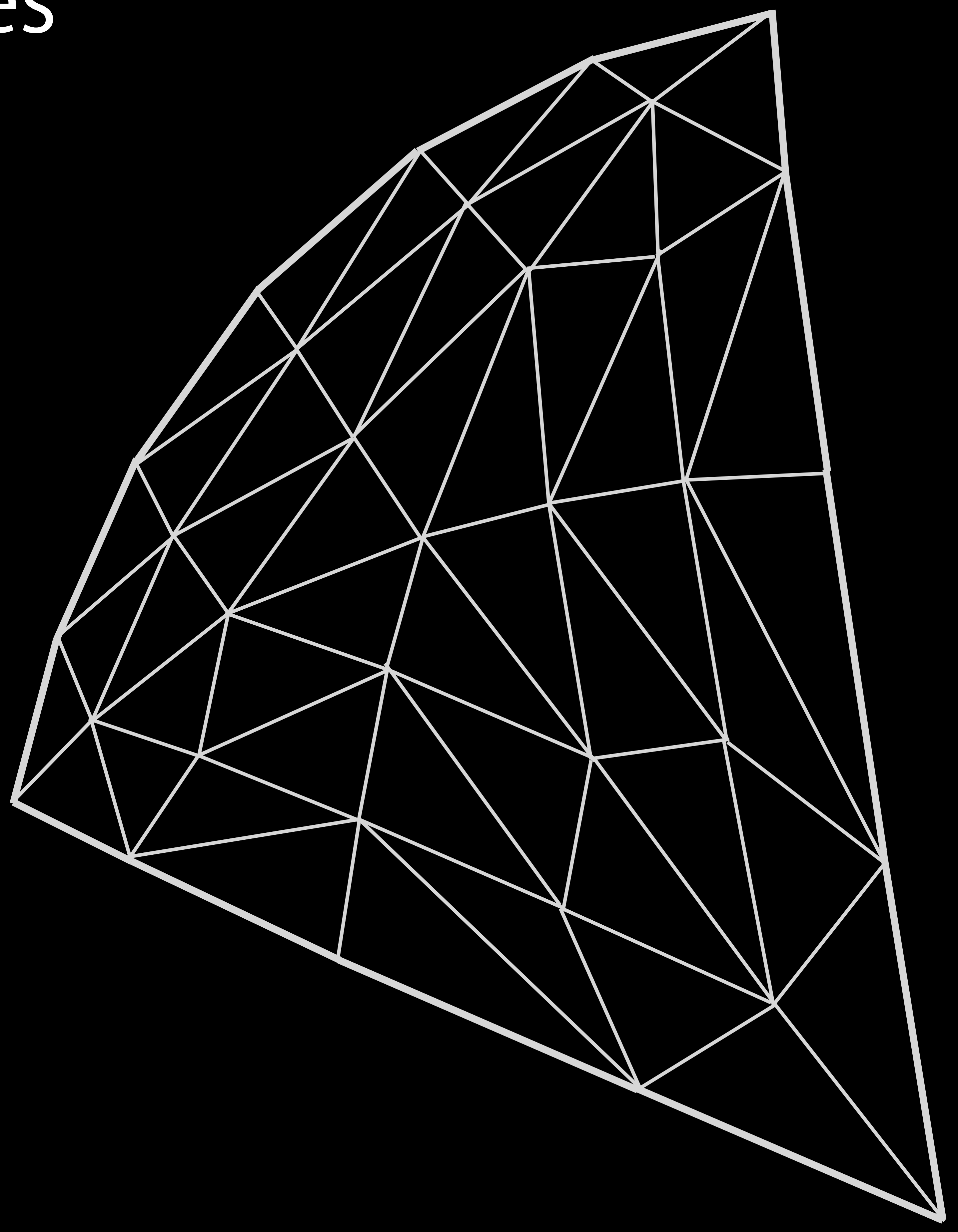
Control Shader

Triangle control shader example

```
#version 400

layout(vertices=3) out;  // Using 3 control points for triangle

in  vec4 vPos[3];
out vec4 ctrlPos[3];

void main ()
{
   ctrlPos[gl_InvocationID] = vPos[gl_InvocationID];
   if(gl_InvocationID == 0) {
      gl_TessLevelOuter[0] = MyCalcOuterLOD(vPos[0], vPos[1]);
      gl_TessLevelOuter[1] = MyCalcOuterLOD(vPos[1], vPos[2]);
      gl_TessLevelOuter[2] = MyCalcOuterLOD(vPos[2], vPos[0]);

      gl_TessLevelInner[0] = MyCalcInnerLOD(vPos[0], vPos[1], vPos[2]);
   }
}
```

Tessellation Evaluation Shader

- “Evaluates” new position and other attributes
- Runs for each tessellated vertex
- Inputs:
  - Original patch
  - Tessellation coordinates
- Evaluation shader outputs:
  - `gl_Position`
  - MyTexCoord
  - And other attributes



Evaluation Shader

Triangle evaluation shader example

```
#version 400

layout(triangles, fractional_odd_spacing) in;
uniform mat4 mvp;

in vec4 ctrlPos[3];  // Input vertex data from control

void main ()
{
   vec4 position = ctrlPos[0] * gl_TessCoord[0] + // Barycentric coordinates
                   ctrlPos[1] * gl_TessCoord[1] +
                   ctrlPos[2] * gl_TessCoord[2];

   gl_Position = mvp * MyCustomDisplacement(position);
}
```

Evaluation Shader

Triangle evaluation shader example

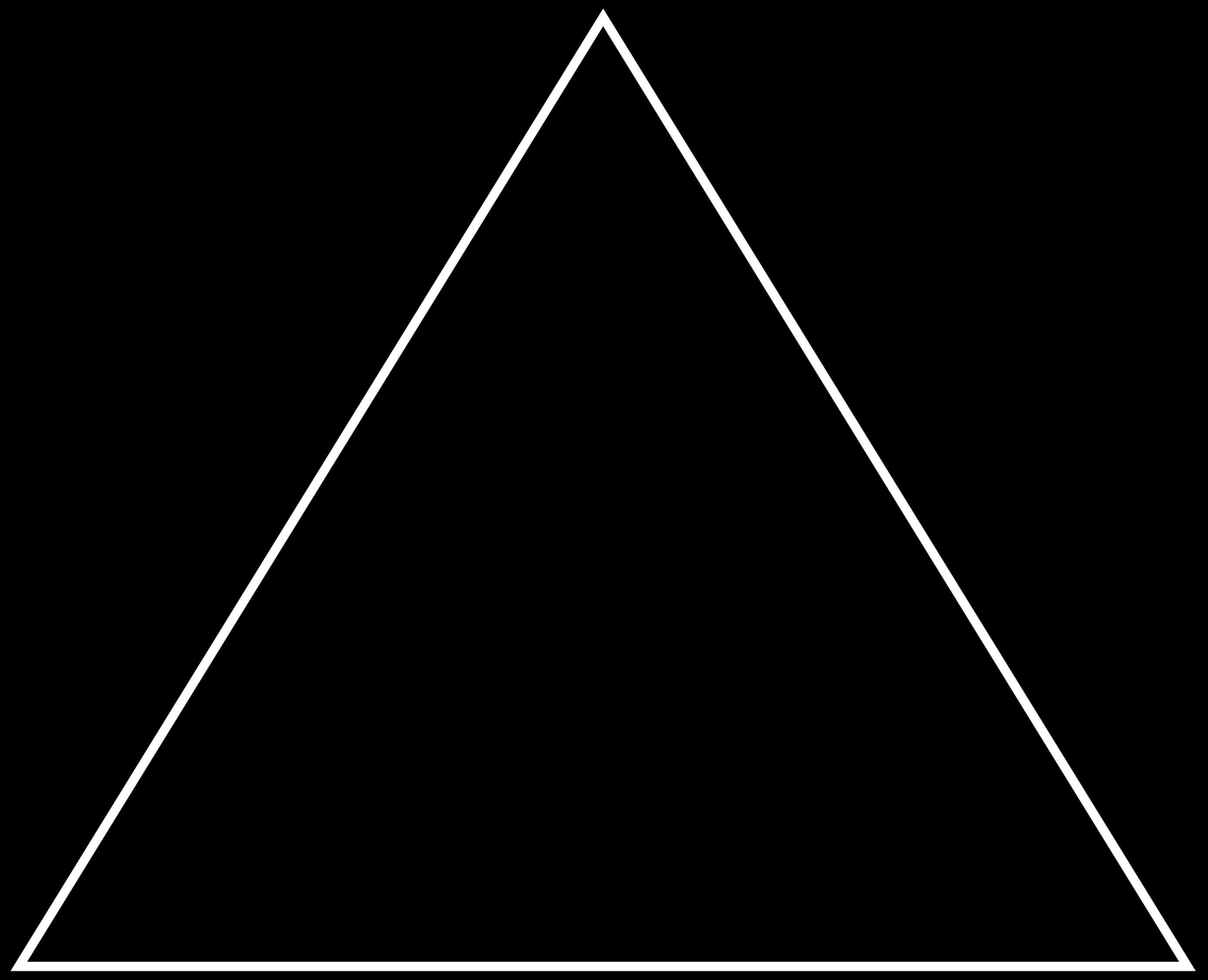
```
#version 400

layout(triangles, fractional_odd_spacing) in;
uniform mat4 mvp;

in vec4 ctrlPos[3];  // Input vertex data from control

void main ()
{
   vec4 position = ctrlPos[0] * gl_TessCoord[0] + // Barycentric coordinates
                   ctrlPos[1] * gl_TessCoord[1] +
                   ctrlPos[2] * gl_TessCoord[2];

   gl_Position = mvp * MyCustomDisplacement(position);
}
```

Evaluation Shader

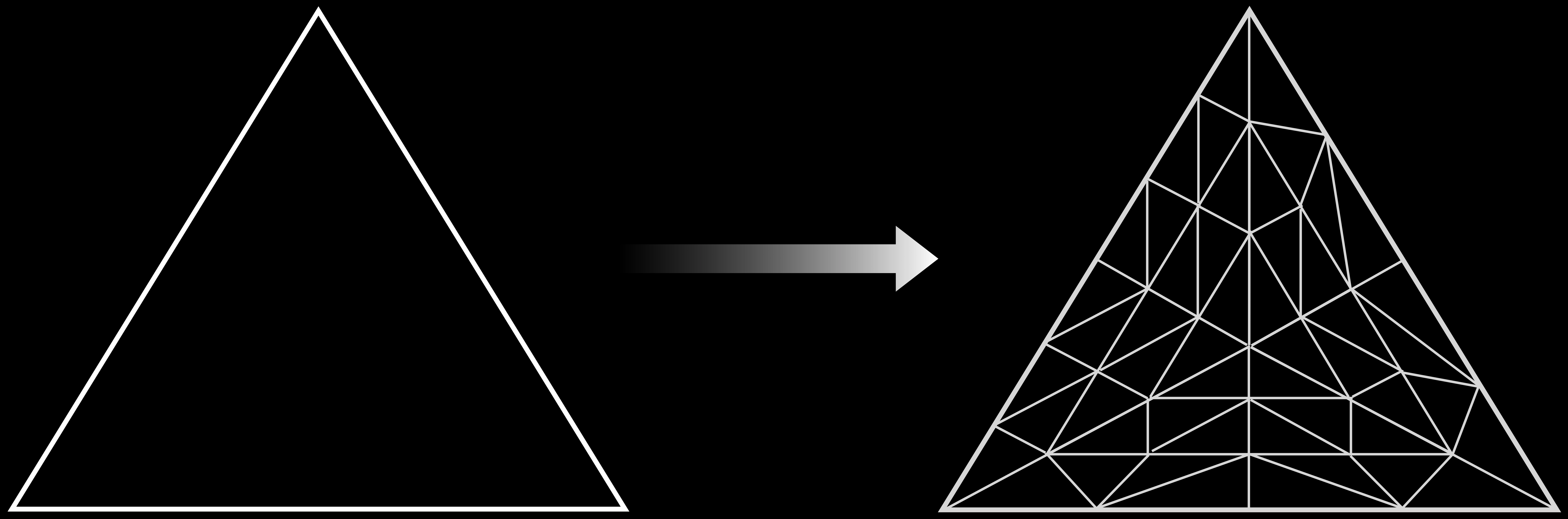
Triangle evaluation shader example

```
#version 400

layout(triangles, fractional_odd_spacing) in;
uniform mat4 mvp;

in vec4 ctrlPos[3];  // Input vertex data from control

void main ()
{
    vec4 position = ctrlPos[0] * gl_TessCoord[0] + // Barycentric coordinates
                    ctrlPos[1] * gl_TessCoord[1] +
                    ctrlPos[2] * gl_TessCoord[2];

    gl_Position = mvp * MyCustomDisplacement(position);
}
```

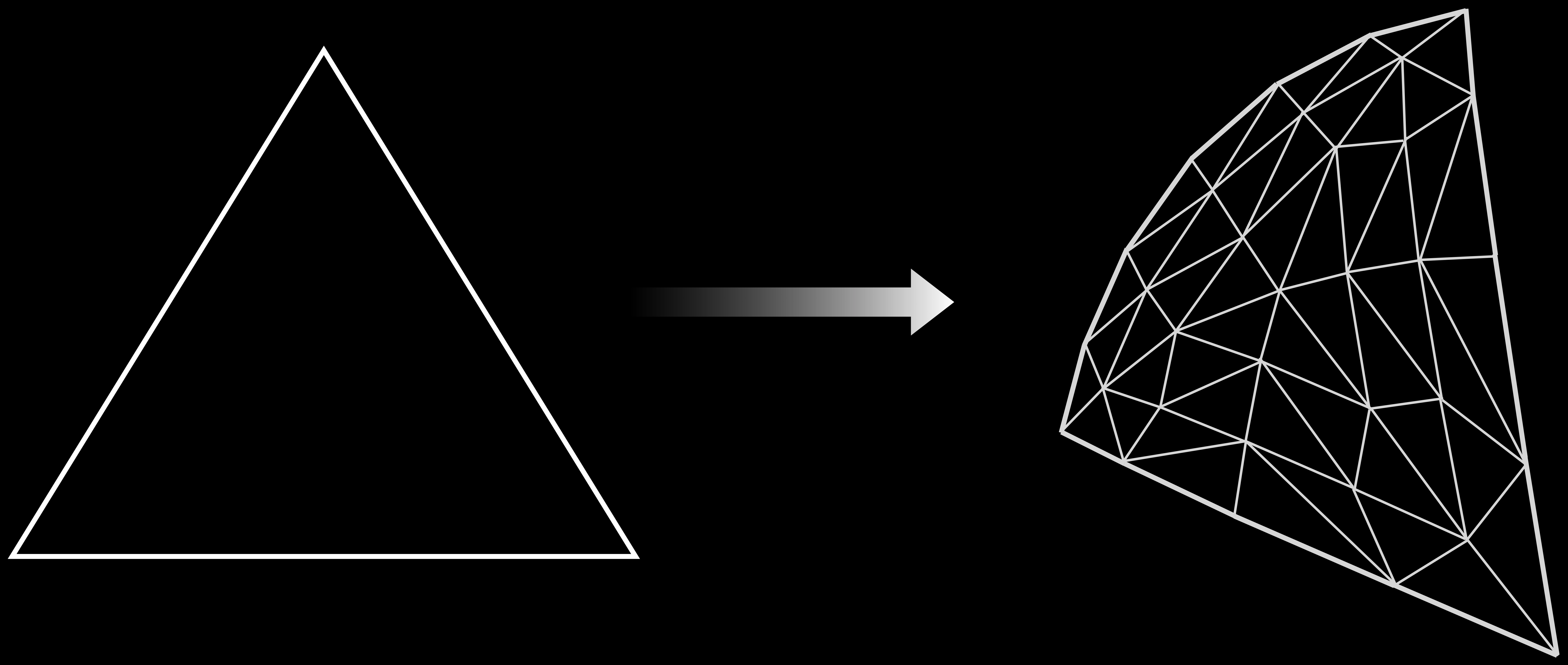
Tessellated Triangle Patch



Tessellated Triangle Patch



Tessellated Triangle Patch



Control Shader

Quad control shader example

```
#version 400

layout(vertices=4) out;  // Using 4 control points for quad

in  vec4 vPos[4];
out vec4 ctrlPos[4];

void main () {
   ctrlPos[gl_InvocationID] = vPos[gl_InvocationID];

   if(gl_InvocationID == 0) {
      gl_TessLevelOuter[0] = MyCalcOuterLOD(vPos[0], vPos[1]);
      gl_TessLevelOuter[1] = MyCalcOuterLOD(vPos[1], vPos[2]);
      gl_TessLevelOuter[2] = MyCalcOuterLOD(vPos[2], vPos[3]);
      gl_TessLevelOuter[3] = MyCalcOuterLOD(vPos[3], vPos[0]);

      gl_TessLevelInner[0] = MyCalcInnerLOD(vPos[1], vPos[3]);
      gl_TessLevelInner[1] = MyCalcInnerLOD(vPos[0], vPos[2]);
   }
}
```

Control Shader

Quad control shader example

```
#version 400
layout(vertices=4) out;  // Using 4 control points for quad

in  vec4 vPos[4];
out vec4 ctrlPos[4];
void main () {
    ctrlPos[gl_InvocationID] = vPos[gl_InvocationID];

    if(gl_InvocationID == 0) {
        gl_TessLevelOuter[0] = MyCalcOuterLOD(vPos[0], vPos[1]);
        gl_TessLevelOuter[1] = MyCalcOuterLOD(vPos[1], vPos[2]);
        gl_TessLevelOuter[2] = MyCalcOuterLOD(vPos[2], vPos[3]);
        gl_TessLevelOuter[3] = MyCalcOuterLOD(vPos[3], vPos[0]);

        gl_TessLevelInner[0] = MyCalcInnerLOD(vPos[1], vPos[3]);
        gl_TessLevelInner[1] = MyCalcInnerLOD(vPos[0], vPos[2]);
    }
}
```

Control Shader

Quad control shader example

```
#version 400

layout(vertices=4) out;  // Using 4 control points for quad

in  vec4 vPos[4];
out vec4 ctrlPos[4];

void main () {
   ctrlPos[gl_InvocationID] = vPos[gl_InvocationID];

   if(gl_InvocationID == 0) {
      gl_TessLevelOuter[0] = MyCalcOuterLOD(vPos[0], vPos[1]);
      gl_TessLevelOuter[1] = MyCalcOuterLOD(vPos[1], vPos[2]);
      gl_TessLevelOuter[2] = MyCalcOuterLOD(vPos[2], vPos[3]);
      gl_TessLevelOuter[3] = MyCalcOuterLOD(vPos[3], vPos[0]);

      gl_TessLevelInner[0] = MyCalcInnerLOD(vPos[1], vPos[3]);
      gl_TessLevelInner[1] = MyCalcInnerLOD(vPos[0], vPos[2]);
   }
}
```

Control Shader

Quad control shader example

```
#version 400

layout(vertices=4) out;  // Using 4 control points for quad

in  vec4 vPos[4];
out vec4 ctrlPos[4];

void main () {
    ctrlPos[gl_InvocationID] = vPos[gl_InvocationID];
    if(gl_InvocationID == 0) {
        gl_TessLevelOuter[0] = MyCalcOuterLOD(vPos[0], vPos[1]);
        gl_TessLevelOuter[1] = MyCalcOuterLOD(vPos[1], vPos[2]);
        gl_TessLevelOuter[2] = MyCalcOuterLOD(vPos[2], vPos[3]);
        gl_TessLevelOuter[3] = MyCalcOuterLOD(vPos[3], vPos[0]);

        gl_TessLevelInner[0] = MyCalcInnerLOD(vPos[1], vPos[3]);
        gl_TessLevelInner[1] = MyCalcInnerLOD(vPos[0], vPos[2]);
    }
}
```

Evaluation Shader

Quad evaluation shader example

```
#version 400

layout(quads, fractional_odd_spacing) in;
uniform mat4 mvp;

in vec4 ctrlPos[4];  // Input vertex data from control

void main ()
{
    vec4 a = mix(ctrlPos[0], ctrlPos[1], gl_TessCoord.x);  // UV coordinates
    vec4 b = mix(ctrlPos[2], ctrlPos[3], gl_TessCoord.x);
    vec4 position = mix(a, b, gl_TessCoord.y);

    gl_Position = mvp * MyCustomDisplacement(position);
}
```

Evaluation Shader

Quad evaluation shader example

```
#version 400

layout(quads, fractional_odd_spacing) in;
uniform mat4 mvp;

in vec4 ctrlPos[4];  // Input vertex data from control

void main ()
{
    vec4 a = mix(ctrlPos[0], ctrlPos[1], gl_TessCoord.x);  // UV coordinates
    vec4 b = mix(ctrlPos[2], ctrlPos[3], gl_TessCoord.x);
    vec4 position = mix(a, b, gl_TessCoord.y);

    gl_Position = mvp * MyCustomDisplacement(position);
}
```

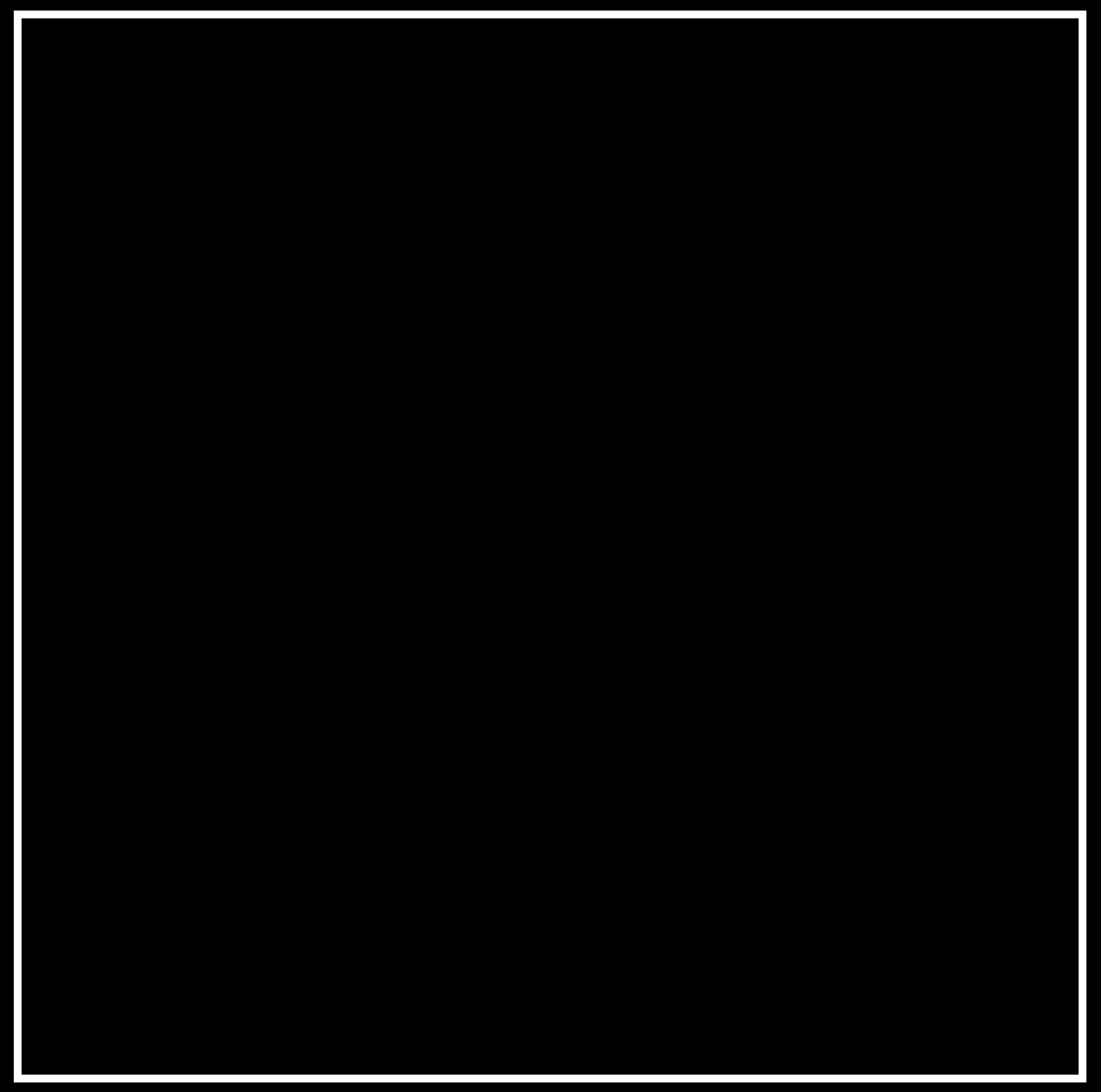
Evaluation Shader

Quad evaluation shader example

```
#version 400

layout(quads, fractional_odd_spacing) in;
uniform mat4 mvp;

in vec4 ctrlPos[4];  // Input vertex data from control

void main ()
{
   vec4 a = mix(ctrlPos[0], ctrlPos[1], gl_TessCoord.x);  // UV coordinates
   vec4 b = mix(ctrlPos[2], ctrlPos[3], gl_TessCoord.x);
   vec4 position = mix(a, b, gl_TessCoord.y);

   gl_Position = mvp * MyCustomDisplacement(position);
}
```

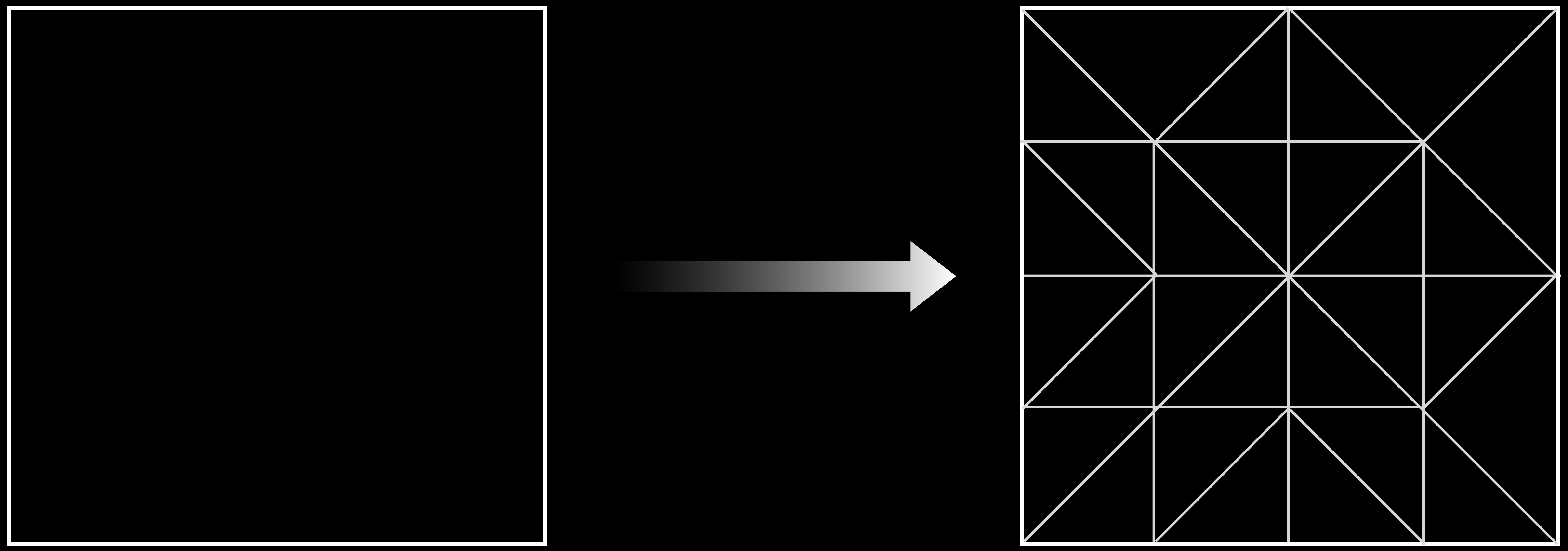
Tessellation Shader

Works with quads too



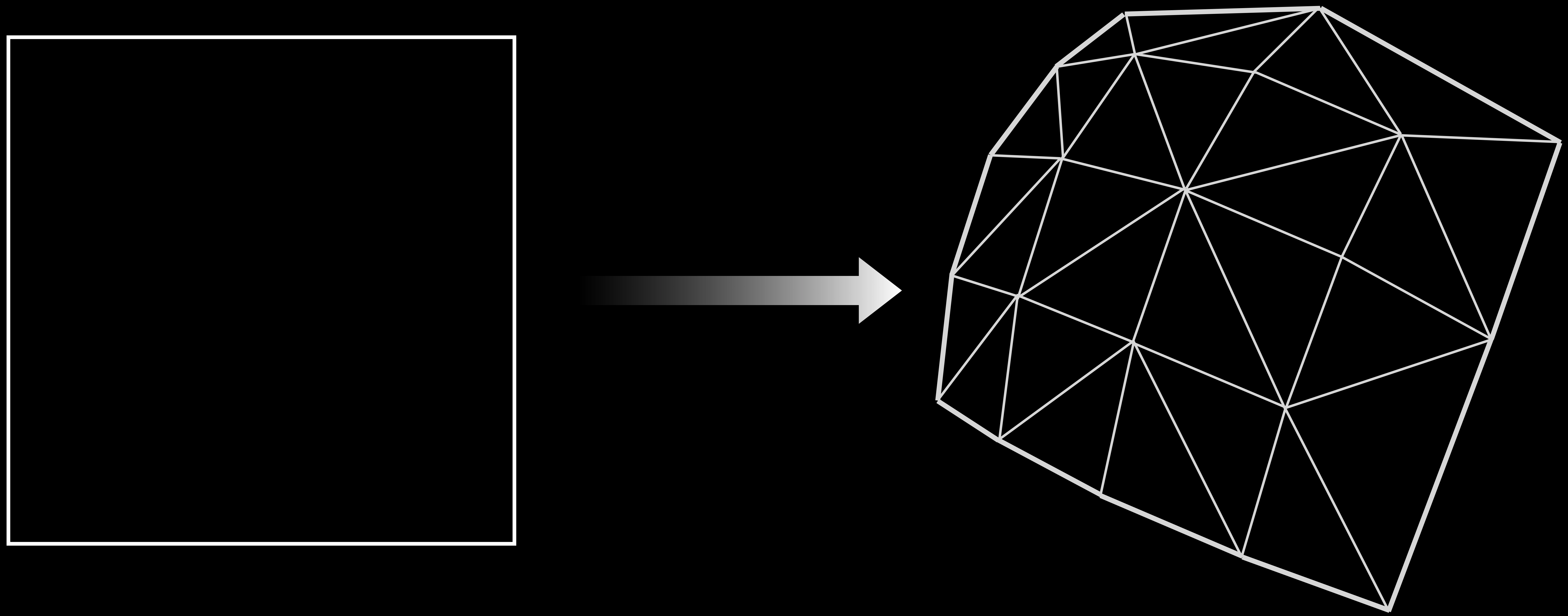
Tessellation Shader

Works with quads too



Tessellation Shader

Works with quads too



Tessellation Shader

Summary

- Add detail where you need it
  - Triangles, quads, arbitrary geometry
- Generate geometry on GPU
  - Instead of submitting it
- Available on modern hardware
  - Check for `GL_ARB_tessellation_shader` using `glGetStringi`
- Match outer patches for crack-free tessellation

Instancing

GL_ARB_draw_instanced
GL_ARB_instanced_arrays

Instancing

Overview

- Useful when drawing many similar objects
- Thousands of copies (instances) with single draw call
- Performance boost
- Each instance can have different parameters
    - Position offset, color, skeletal attributes…
    - Defined in external buffer
- Guaranteed support in Core Profile contexts on OS X

Instancing

Two forms

Instancing

Two forms

- Instanced arrays: `GL_ARB_instanced_arrays`
  - Instance parameters in a vertex array
  - Use a divisor for repeating attributes

Instancing

Two forms

- Instanced arrays: `GL_ARB_instanced_arrays`
  - Instance parameters in a vertex array
  - Use a divisor for repeating attributes
- Shader instance ID: `GL_ARB_draw_instanced`
  - `gl_InstanceID` variable for instance drawn in vertex shader

Instancing

Two forms

- Instanced arrays: `GL_ARB_instanced_arrays`
  - Instance parameters in a vertex array
  - Use a divisor for repeating attributes
- Shader instance ID: `GL_ARB_draw_instanced`
  - `gl_InstanceID` variable for instance drawn in vertex shader
- Both instancing methods available in iOS 7

Uniform Buffer Objects

GL_ARB_uniform_buffer_object

Uniform Buffer Objects

Overview

- Buffer object to store uniform data
- Benefits
  - Faster than calls to `glUniform`
  - Share uniform data among different GLSL shaders
  - Quickly switch between uniform sets in shaders
  - Access GPU generated data
- Uses
  - Skinning
  - Character animation
  - Instancing with `gl_InstanceID`

Uniform Buffer Objects

Shader usage example

```
#version 150

#define MY_DATA_SIZE 16

// UBO interface block definition
layout(std140) uniform MyUBO
{
    vec4  my_data[MY_DATA_SIZE];
    ivec2 another_var;
} MyBlock;

void main ()
{
    // Example read from UBO block
    vec4 uboData = MyBlock.my_data[offset];

    // ...
}
```

Uniform Buffer Objects

Shader usage example

```
#version 150

#define MY_DATA_SIZE 16

// UBO interface block definition
layout(std140) uniform MyUBO
{
    vec4  my_data[MY_DATA_SIZE];
    ivec2 another_var;
} MyBlock;

void main ()
{
    // Example read from UBO block
    vec4 uboData = MyBlock.my_data[offset];

    // ...
}
```

Uniform Buffer Objects

Shader usage example

```
#version 150

#define MY_DATA_SIZE 16

// UBO interface block definition
layout(std140) uniform MyUBO
{
    vec4  my_data[MY_DATA_SIZE];
    ivec2 another_var;
} MyBlock;

void main ()
{
    // Example read from UBO block
    vec4 uboData = MyBlock.my_data[offset];

    // ...
}
```

Uniform Buffer Objects

API setup

```
GLuint prog_id = MyLinkProgram(...);

glGenBuffers(1, &ubo_id);

// Data needs to match GLSL shader specified-layout (i.e. std140)
glBindBuffer(GL_UNIFORM_BUFFER, ubo_id);
glBufferData(GL_UNIFORM_BUFFER, dataSize, data, GL_STATIC_DRAW);

GLuint block_index = glGetUniformBlockIndex(prog_id, "MyUBO");

#define BINDING_IDX  0  // [0, GL_MAX_UNIFORM_BUFFER_BINDINGS)
glUniformBlockBinding(prog_id, block_index, BINDING_IDX);
glBindBufferBase(GL_UNIFORM_BUFFER, BINDING_IDX, ubo_id);
```

Uniform Buffer Objects

API setup

```
GLuint prog_id = MyLinkProgram(...);

glGenBuffers(1, &ubo_id);

// Data needs to match GLSL shader specified-layout (i.e. std140)
glBindBuffer(GL_UNIFORM_BUFFER, ubo_id);
glBufferData(GL_UNIFORM_BUFFER, dataSize, data, GL_STATIC_DRAW);

GLuint block_index = glGetUniformBlockIndex(prog_id, "MyUBO");

#define BINDING_IDX  0  // [0, GL_MAX_UNIFORM_BUFFER_BINDINGS)
glUniformBlockBinding(prog_id, block_index, BINDING_IDX);
glBindBufferBase(GL_UNIFORM_BUFFER, BINDING_IDX, ubo_id);
```

Uniform Buffer Objects

API setup

```
GLuint prog_id = MyLinkProgram(...);

glGenBuffers(1, &ubo_id);

// Data needs to match GLSL shader specified-layout (i.e. std140)
glBindBuffer(GL_UNIFORM_BUFFER, ubo_id);
glBufferData(GL_UNIFORM_BUFFER, dataSize, data, GL_STATIC_DRAW);

GLuint block_index = glGetUniformBlockIndex(prog_id, "MyUBO");

#define BINDING_IDX  0  // [0, GL_MAX_UNIFORM_BUFFER_BINDINGS)
glUniformBlockBinding(prog_id, block_index, BINDING_IDX);
glBindBufferBase(GL_UNIFORM_BUFFER, BINDING_IDX, ubo_id);
```

Uniform Buffer Objects

API setup

```
GLuint prog_id = MyLinkProgram(...);

glGenBuffers(1, &ubo_id);

// Data needs to match GLSL shader specified-layout (i.e. std140)
glBindBuffer(GL_UNIFORM_BUFFER, ubo_id);
glBufferData(GL_UNIFORM_BUFFER, dataSize, data, GL_STATIC_DRAW);

GLuint block_index = glGetUniformBlockIndex(prog_id, "MyUBO");
```

```
#define BINDING_IDX  0  // [0, GL_MAX_UNIFORM_BUFFER_BINDINGS)
glUniformBlockBinding(prog_id, block_index, BINDING_IDX);
glBindBufferBase(GL_UNIFORM_BUFFER, BINDING_IDX, ubo_id);
```

Uniform Buffer Objects

Summary

- Upload many uniform values all at once
- Tip: Split frequently modified uniforms into a separate UBO
    - Or orphan buffers with `glBufferData(GL_UNIFORM_BUFFER, …, NULL)`
    - Or double buffer
- Each UBO block size is limited to `GL_MAX_UNIFORM_BLOCK_SIZE` (64KB)

Texture Buffer Objects

GL_ARB_texture_buffer_object

Texture Buffer Objects

Overview

- Buffer object to store 1D array of data as texels
- Benefits
  - Access GPU generated data
  - Access a large amount of data within a shader
  - Uses GPU's texture cache
- Uses
  - Skinning
  - Character animation
  - Instancing with `gl_InstanceID`

Texture Buffer Objects

Shader usage example

```
#version 150

// Texture buffer objects use new sampler types
uniform samplerBuffer MyTBO;

void main ()
{
    // Texel read example from TBO into theColor.rgba
    vec4 theColor = texelFetch(MyTBO, offset);

    // ...
}
```

Texture Buffer Objects

Shader usage example

```
#version 150

// Texture buffer objects use new sampler types
uniform samplerBuffer MyTBO;

void main ()
{
    // Texel read example from TBO into theColor.rgba
    vec4 theColor = texelFetch(MyTBO, offset);

    // ...
}
```

Texture Buffer Objects

Shader usage example

```
#version 150

// Texture buffer objects use new sampler types
uniform samplerBuffer MyTBO;

void main ()
{
    // Texel read example from TBO into theColor.rgba
    vec4 theColor = texelFetch(MyTBO, offset);

    // ...
}
```

Texture Buffer Objects

Shader usage example

```
#version 150

// Texture buffer objects use new sampler types
uniform samplerBuffer MyTBO;

void main ()
{
   // Matrix read from TBO into theMatrix
   mat4x4 theMatrix( texelFetch(MyTBO, gl_InstanceID*4 + 0),
                     texelFetch(MyTBO, gl_InstanceID*4 + 1),
                     texelFetch(MyTBO, gl_InstanceID*4 + 2),
                     texelFetch(MyTBO, gl_InstanceID*4 + 3) );

   // ...
}
```

Texture Buffer Objects

API setup

```
GLuint prog_id = MyLinkProgram(…);

glGenBuffers(1, &tbo_id);
glGenTextures(1, &tex_id);

// Data needs to match glTexBuffer() format (i.e. GL_RGBA32F)
glBindBuffer(GL_TEXTURE_BUFFER, tbo_id);
glBufferData(GL_TEXTURE_BUFFER, dataSize, data, GL_STATIC_DRAW);

GLint tex_unit = 0;  // [0, GL_MAX_TEXTURE_IMAGE_UNITS)
glActiveTexture(GL_TEXTURE0 + tex_unit);
glBindTexture(GL_TEXTURE_BUFFER, tex_id);
glTexBuffer(GL_TEXTURE_BUFFER, GL_RGBA32F, tbo_id);

GLint tbo_loc = glGetUniformLocation(prog_id, "MyTBO");
glUniform1i(tbo_loc, tex_unit);
```

Texture Buffer Objects

API setup

```
GLuint prog_id = MyLinkProgram(…);

glGenBuffers(1, &tbo_id);
glGenTextures(1, &tex_id);

// Data needs to match glTexBuffer() format (i.e. GL_RGBA32F)
glBindBuffer(GL_TEXTURE_BUFFER, tbo_id);
glBufferData(GL_TEXTURE_BUFFER, dataSize, data, GL_STATIC_DRAW);

GLint tex_unit = 0;  // [0, GL_MAX_TEXTURE_IMAGE_UNITS)
glActiveTexture(GL_TEXTURE0 + tex_unit);
glBindTexture(GL_TEXTURE_BUFFER, tex_id);
glTexBuffer(GL_TEXTURE_BUFFER, GL_RGBA32F, tbo_id);

GLint tbo_loc = glGetUniformLocation(prog_id, "MyTBO");
glUniform1i(tbo_loc, tex_unit);
```

Texture Buffer Objects

API setup

```
GLuint prog_id = MyLinkProgram(…);

glGenBuffers(1, &tbo_id);
glGenTextures(1, &tex_id);

// Data needs to match glTexBuffer() format (i.e. GL_RGBA32F)
glBindBuffer(GL_TEXTURE_BUFFER, tbo_id);
glBufferData(GL_TEXTURE_BUFFER, dataSize, data, GL_STATIC_DRAW);

GLint tex_unit = 0;  // [0, GL_MAX_TEXTURE_IMAGE_UNITS)
glActiveTexture(GL_TEXTURE0 + tex_unit);
glBindTexture(GL_TEXTURE_BUFFER, tex_id);
glTexBuffer(GL_TEXTURE_BUFFER, GL_RGBA32F, tbo_id);

GLint tbo_loc = glGetUniformLocation(prog_id, "MyTBO");
glUniform1i(tbo_loc, tex_unit);
```

Texture Buffer Objects

API setup

```
GLuint prog_id = MyLinkProgram(…);

glGenBuffers(1, &tbo_id);
glGenTextures(1, &tex_id);

// Data needs to match glTexBuffer() format (i.e. GL_RGBA32F)
glBindBuffer(GL_TEXTURE_BUFFER, tbo_id);
glBufferData(GL_TEXTURE_BUFFER, dataSize, data, GL_STATIC_DRAW);

GLint tex_unit = 0;  // [0, GL_MAX_TEXTURE_IMAGE_UNITS)
glActiveTexture(GL_TEXTURE0 + tex_unit);
glBindTexture(GL_TEXTURE_BUFFER, tex_id);
glTexBuffer(GL_TEXTURE_BUFFER, GL_RGBA32F, tbo_id);

GLint tbo_loc = glGetUniformLocation(prog_id, "MyTBO");
glUniform1i(tbo_loc, tex_unit);
```

Texture Buffer Objects

Summary

- Access a large data array via texture sampling
- Commonly used with instancing
- Don’t modify a TBO while it’s being used to draw
  - Double buffering, orphaning
- TBO size limited to `GL_MAX_TEXTURE_BUFFER_SIZE` (≥64MB)

Draw Indirect

GL_ARB_draw_indirect

Draw Indirect

Overview

- Specify draw call arguments from buffer object data
  - `count, instanceCount, first, baseVertex`
- Useful when generating geometry with OpenCL
  - No round-trip to CPU needed
- Available on modern hardware
  - Check for `GL_ARB_draw_indirect` using `glGetStringi`

Draw Indirect

DrawArrays example

```
//  typedef struct {
//      GLuint count, instanceCount, first;
//      GLuint reservedMustBeZero;
//  } DrawArraysIndirectCommand;


// CL kernel generated DrawArraysIndirectCommand values
// into DRAW_INDIRECT_BUFFER at indirectBufOffset.

glBindBuffer(GL_DRAW_INDIRECT_BUFFER, buf_id);
glBindVertexArray(vao_id);


// Replaces glDrawArrays(mode, first, count)
glDrawArraysIndirect(mode, indirectBufOffset);
```

Draw Indirect

DrawArrays example

```
//  typedef struct {
//      GLuint count, instanceCount, first;
//      GLuint reservedMustBeZero;
//  } DrawArraysIndirectCommand;
```

```
// CL kernel generated DrawArraysIndirectCommand values
// into DRAW_INDIRECT_BUFFER at indirectBufOffset.

glBindBuffer(GL_DRAW_INDIRECT_BUFFER, buf_id);
glBindVertexArray(vao_id);


// Replaces glDrawArrays(mode, first, count)
glDrawArraysIndirect(mode, indirectBufOffset);
```

Draw Indirect

DrawArrays example

```
// typedef struct {
//     GLuint count, instanceCount, first;
//     GLuint reservedMustBeZero;
// } DrawArraysIndirectCommand;


// CL kernel generated DrawArraysIndirectCommand values
// into DRAW_INDIRECT_BUFFER at indirectBufOffset.

glBindBuffer(GL_DRAW_INDIRECT_BUFFER, buf_id);
glBindVertexArray(vao_id);


// Replaces glDrawArrays(mode, first, count)
glDrawArraysIndirect(mode, indirectBufOffset);
```

Draw Indirect

DrawArrays example

```
// typedef struct {
//     GLuint count, instanceCount, first;
//     GLuint reservedMustBeZero;
// } DrawArraysIndirectCommand;


// CL kernel generated DrawArraysIndirectCommand values
// into DRAW_INDIRECT_BUFFER at indirectBufOffset.
```

```
glBindBuffer(GL_DRAW_INDIRECT_BUFFER, buf_id);
glBindVertexArray(vao_id);
```

```
// Replaces glDrawArrays(mode, first, count)
glDrawArraysIndirect(mode, indirectBufOffset);
```

Draw Indirect

DrawArrays example

```
//  typedef struct {
//      GLuint count, instanceCount, first;
//      GLuint reservedMustBeZero;
//  } DrawArraysIndirectCommand;


// CL kernel generated DrawArraysIndirectCommand values
// into DRAW_INDIRECT_BUFFER at indirectBufOffset.

glBindBuffer(GL_DRAW_INDIRECT_BUFFER, buf_id);
glBindVertexArray(vao_id);
```

```
// Replaces glDrawArrays(mode, first, count)
glDrawArraysIndirect(mode, indirectBufOffset);
```

Draw Indirect

DrawElements example

```
//  typedef struct {
//      GLuint count, instanceCount, firstIndex;
//      GLint baseVertex;
//      GLuint reservedMustBeZero;
//  } DrawElementsIndirectCommand;

// CL kernel generated DrawElementsIndirectCommand values
// into DRAW_INDIRECT_BUFFER at indirectBufOffset.

glBindBuffer(GL_DRAW_INDIRECT_BUFFER, buf_id);
glBindVertexArray(vao_id);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, elem_buf_id);

// Replaces glDrawElements(mode, count, elemType, elemBufOffset)
glDrawElementsIndirect(mode, elemType, indirectBufOffset);
```

Draw Indirect

DrawElements example

```
//  typedef struct {
//      GLuint count, instanceCount, firstIndex;
//      GLint baseVertex;
//      GLuint reservedMustBeZero;
//  } DrawElementsIndirectCommand;
```

```
// CL kernel generated DrawElementsIndirectCommand values
// into DRAW_INDIRECT_BUFFER at indirectBufOffset.

glBindBuffer(GL_DRAW_INDIRECT_BUFFER, buf_id);
glBindVertexArray(vao_id);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, elem_buf_id);

// Replaces glDrawElements(mode, count, elemType, elemBufOffset)
glDrawElementsIndirect(mode, elemType, indirectBufOffset);
```

Draw Indirect

DrawElements example

```
// typedef struct {
//     GLuint count, instanceCount, firstIndex;
//     GLint baseVertex;
//     GLuint reservedMustBeZero;
// } DrawElementsIndirectCommand;

// CL kernel generated DrawElementsIndirectCommand values
// into DRAW_INDIRECT_BUFFER at indirectBufOffset.

glBindBuffer(GL_DRAW_INDIRECT_BUFFER, buf_id);
glBindVertexArray(vao_id);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, elem_buf_id);

// Replaces glDrawElements(mode, count, elemType, elemBufOffset)
glDrawElementsIndirect(mode, elemType, indirectBufOffset);
```

Draw Indirect

DrawElements example

```
//  typedef struct {
//      GLuint count, instanceCount, firstIndex;
//      GLint baseVertex;
//      GLuint reservedMustBeZero;
//  } DrawElementsIndirectCommand;

// CL kernel generated DrawElementsIndirectCommand values
// into DRAW_INDIRECT_BUFFER at indirectBufOffset.

glBindBuffer(GL_DRAW_INDIRECT_BUFFER, buf_id);
glBindVertexArray(vao_id);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, elem_buf_id);

// Replaces glDrawElements(mode, count, elemType, elemBufOffset)
glDrawElementsIndirect(mode, elemType, indirectBufOffset);
```

Draw Indirect

DrawElements example

```
//  typedef struct {
//      GLuint count, instanceCount, firstIndex;
//      GLint baseVertex;
//      GLuint reservedMustBeZero;
//  } DrawElementsIndirectCommand;

// CL kernel generated DrawElementsIndirectCommand values
// into DRAW_INDIRECT_BUFFER at indirectBufOffset.

glBindBuffer(GL_DRAW_INDIRECT_BUFFER, buf_id);
glBindVertexArray(vao_id);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, elem_buf_id);
```

```
// Replaces glDrawElements(mode, count, elemType, elemBufOffset)
glDrawElementsIndirect(mode, elemType, indirectBufOffset);
```

And a Few More…

And a Few More…

NEW

`GL_ARB_separate_shader_objects`

- Enables mix-and-match between GLSL shaders

`GL_ARB_ES2_compatibility`

- Use "#version 100" GLSL shaders on OS X

`GL_NV_texture_barrier`

- Bind the same texture as both a render target and texture source

`GL_ARB_texture_swizzle`

- Support legacy formats like `GL_LUMINANCE`—Available in OS X 10.8.3

OpenGL and Compute

Using OpenCL with OpenGL

OpenGL and Compute

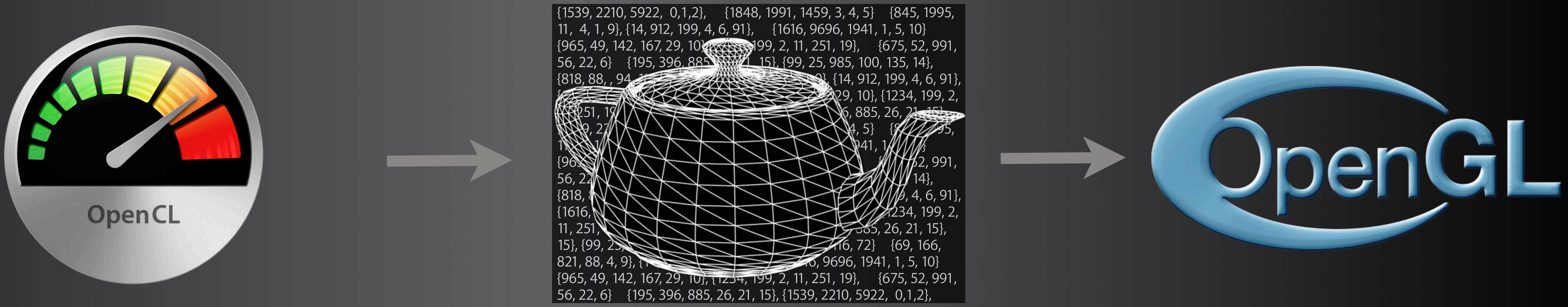
Using OpenGL with OpenCL

- Rendering pipeline of OpenGL
- Parallel compute of OpenCL
- Can share resources (buffers, textures)
  - No need to copy data to host and back
- Simple integration into render loop

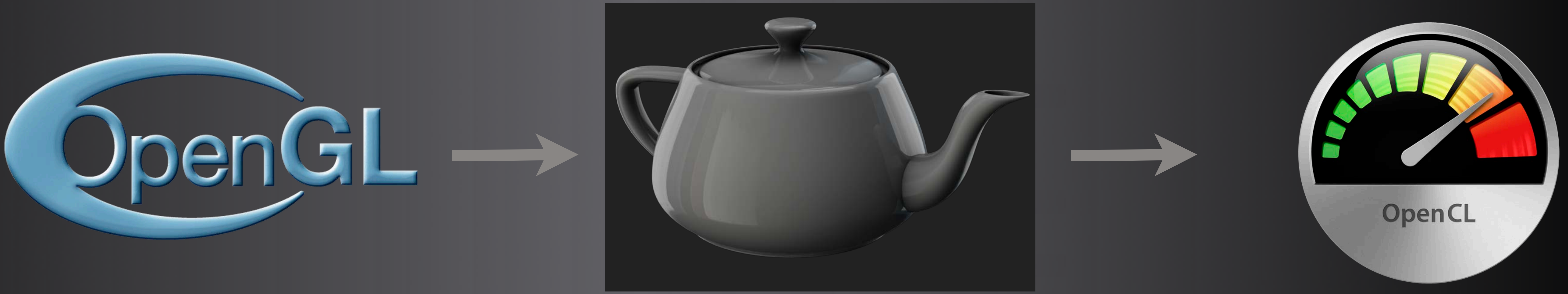
OpenGL and Compute

Interoperability

Generate or modify geometry



Post-process images



OpenGL and Compute

Example: Fill VBO in OpenCL, render in OpenGL

- One-time setup
  - Set up OpenGL and OpenCL contexts, allowing sharing
  - Set up vertex buffer object to be shared
- Every frame
  - Enqueue OpenCL commands to fill VBO
  - Flush OpenCL to ensure synchronization
  - Draw using VBO in OpenGL

OpenCL

OpenGL

OpenCL

OpenGL

`NSOpenGLPFAAcceleratedCompute`

OpenCL

OpenGL

NSOpenGLPFAAcceleratedCompute

CGLGetShareGroup

OpenCL

clGetDeviceIDs

OpenGL

NSOpenGLPFAAcceleratedCompute

CGLGetShareGroup

OpenCL

clGetDeviceIDs

clCreateContext

OpenGL

NSOpenGLPFAAcceleratedCompute

CGLGetShareGroup

OpenCL

clGetDeviceIDs

clCreateContext

OpenGL

NSOpenGLPFAAcceleratedCompute

CGLGetShareGroup

glBindBuffer

OpenCL

clGetDeviceIDs

clCreateContext

OpenGL

NSOpenGLPFAAcceleratedCompute

CGLGetShareGroup

glBindBuffer

glBufferData

OpenCL

clGetDeviceIDs

clCreateContext

OpenGL

NSOpenGLPFAAcceleratedCompute

CGLGetShareGroup

glBindBuffer

glBufferData

glFlushRenderAPPLE

OpenCL

clGetDeviceIDs

clCreateContext

clCreateFromGLBuffer

OpenGL

NSOpenGLPFAAcceleratedCompute

CGLGetShareGroup

glBindBuffer

glBufferData

glFlushRenderAPPLE

One time setup

OpenCL

OpenGL

DEVICE_FOR_CURRENT_VS_APPLE

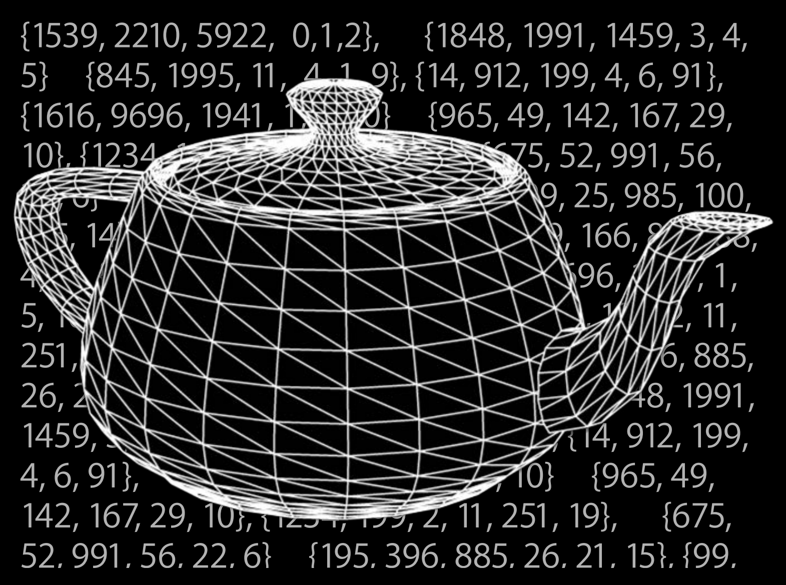
DEVICE_FOR_CURRENT_VS_APPLE

clEnqueueNDRangeKernel

DEVICE_FOR_CURRENT_VS_APPLE

clEnqueueNDRangeKernel

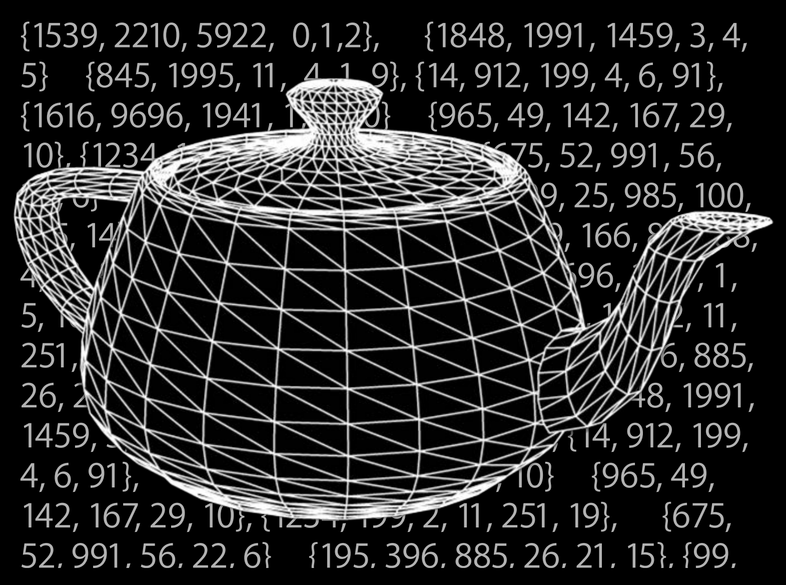
clFlush



`DEVICE_FOR_CURRENT_VS_APPLE`

`clEnqueueNDRangeKernel`

`clFlush`

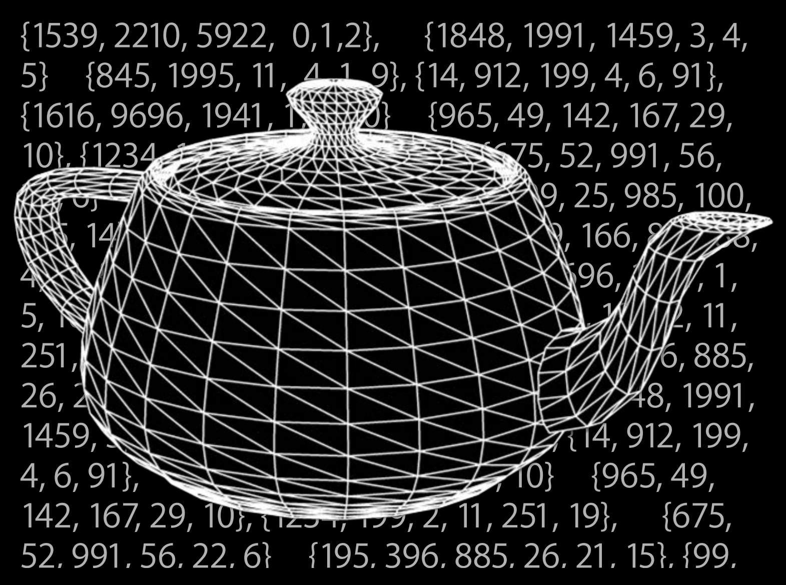


Barrier

DEVICE_FOR_CURRENT_VS_APPLE

clEnqueueNDRangeKernel

clFlush



Barrier

glDrawElementsIndirect

Vice-Versa

Example: Draw to texture in OpenGL, process in OpenCL

- One-time setup
  - Set up OpenGL and OpenCL contexts, allowing sharing (same)
  - Set up texture object to be shared
- Every frame
  - Draw to texture in OpenGL
  - Flush OpenGL to ensure synchronization
  - Enqueue OpenCL commands to process texture
- Display result
  - Flush OpenCL to ensure synchronization
  - Blit/Swap texture in OpenGL

OpenCL

OpenGL

OpenCL

OpenGL

`NSOpenGLPFAAcceleratedCompute`

OpenCL

OpenGL

NSOpenGLPFAAcceleratedCompute

CGLGetShareGroup

`clGetDeviceIDs`

`NSOpenGLPFAAcceleratedCompute`

`CGLGetShareGroup`

OpenCL

clGetDeviceIDs

clCreateContext

OpenGL

NSOpenGLPFAAcceleratedCompute

CGLGetShareGroup

OpenCL

clGetDeviceIDs

clCreateContext

OpenGL

NSOpenGLPFAAcceleratedCompute

CGLGetShareGroup

glBindTexture

OpenCL

`clGetDeviceIDs`

`clCreateContext`

OpenGL

`NSOpenGLPFAAcceleratedCompute`

`CGLGetShareGroup`

`glBindTexture`

`glTexImage2D`

clGetDeviceIDs

clCreateContext

NSOpenGLPFAAcceleratedCompute

CGLGetShareGroup

glBindTexture

glTexImage2D

glFlushRenderAPPLE

OpenCL

clGetDeviceIDs

clCreateContext

clCreateFromGLTexture

OpenGL

NSOpenGLPFAAcceleratedCompute

CGLGetShareGroup

glBindTexture

glTexImage2D

glFlushRenderAPPLE

One time setup

OpenCL

OpenGL

OpenCL

OpenGL

`glDrawElements`

glDrawElements

glFlushRenderAPPLE

glDrawElements

glFlushRenderAPPLE

Barrier

glDrawElements

glFlushRenderAPPLE

Barrier

DEVICE_FOR_CURRENT_VS_APPLE

glDrawElements

glFlushRenderAPPLE

Barrier

DEVICE_FOR_CURRENT_VS_APPLE

clEnqueueNDRangeKernel

OpenCL

OpenGL

glDrawElements

glFlushRenderAPPLE

Barrier

DEVICE_FOR_CURRENT_VS_APPLE

clEnqueueNDRangeKernel

clFlush

OpenCL

OpenGL

glDrawElements

glFlushRenderAPPLE

Barrier

DEVICE_FOR_CURRENT_VS_APPLE

clEnqueueNDRangeKernel

clFlush

Barrier

OpenCL

OpenGL

glDrawElements

glFlushRenderAPPLE

Barrier

DEVICE_FOR_CURRENT_VS_APPLE

clEnqueueNDRangeKernel

clFlush

Barrier

glBindTexture

OpenCL

OpenGL

glDrawElements

glFlushRenderAPPLE

Barrier

DEVICE_FOR_CURRENT_VS_APPLE

clEnqueueNDRangeKernel

clFlush

Barrier

glBindTexture

glBlitFramebuffer

OpenGL with OpenCL

Summary

- Best of both worlds
- Bidirectional sharing
    - Via VBO, FBO, texture objects
- Great with Draw Indirect

OpenGL with OpenCL

Summary

- Best of both worlds
- Bidirectional sharing
    - Via VBO, FBO, texture objects
- Great with Draw Indirect

Working with OpenCL

Marina
Thursday 3:15PM

Migrating to OpenGL Core Profile

OpenGL Core Profile

Overview

- Gives access to the latest GPU features
- High performance and streamlined APIs
- More control over the rendering pipeline
- Portability to and from OpenGL ES 2.0 on iOS

Migrating to Core Profile

Conceptual overview

Legacy	Core Profile
Immediate mode drawing	Vertex arrays with VBOs
Fixed function state	GLSL shaders
Matrix math via OpenGL	Custom matrix math
Older shaders	GLSL 150+

Migrating to Core Profile

Conceptual overview

Legacy	Core Profile	Core Profile with GL Kit
Immediate mode drawing	Vertex arrays with VBOs	Vertex arrays with VBOs
Fixed function state	GLSL shaders	GLKBaseEffect
Matrix math via OpenGL	Custom matrix math	GLKMath
Older shaders	GLSL 150+	GLSL 150+

Getting Started

Creating a context

```
// Opt-in to Core Profile
NSOpenGLPixelFormatAttribute attr[] =
{
   NSOpenGLPFAOpenGLProfile, NSOpenGLProfileVersion3_2Core,
   NSOpenGLColorSize, 24,
   NSOpenGLAlphaSize, 8,
   NSOpenGLPFAAccelerated,
   0
};

// Init format and get context
NSOpenGLPixelFormat* pix = [NSOpenGLPixelFormat initWithAttributes:attr];
NSOpenGLContext* ctx = [NSOpenGLContext initWithFormat:pix shareContext:nil];
```

Getting Started

Creating a context

```
// Opt-in to Core Profile
NSOpenGLPixelFormatAttribute attr[] =
{
    NSOpenGLPFAOpenGLProfile, NSOpenGLProfileVersion3_2Core,
    NSOpenGLColorSize, 24,
    NSOpenGLAlphaSize, 8,
    NSOpenGLPFAAccelerated,
    0
};

// Init format and get context
NSOpenGLPixelFormat* pix = [NSOpenGLPixelFormat initWithAttributes:attr];
NSOpenGLContext* ctx = [NSOpenGLContext initWithFormat:pix shareContext:nil];
```

Getting Started

Creating a context

```
// Opt-in to Core Profile
NSOpenGLPixelFormatAttribute attr[] =
{
   NSOpenGLPFAOpenGLProfile, NSOpenGLProfileVersion3_2Core,
   NSOpenGLColorSize, 24,
   NSOpenGLAlphaSize, 8,
   NSOpenGLPFAAccelerated,
   0
};
```

```
// Init format and get context
NSOpenGLPixelFormat* pix = [NSOpenGLPixelFormat initWithAttributes:attr];
NSOpenGLContext* ctx = [NSOpenGLContext initWithFormat:pix shareContext:nil];
```

Replace Immediate Mode Drawing

VAOs and VBOs

- Cache vertex data in Vertex Buffer Objects (VBOs)
- Encapsulate vertex array state with Vertex Array Objects (VAOs)

Legacy	Core Profile

`glBegin(GL_TRIANGLES) … glEnd()`	`glBindVertexArray`
	`glDrawArrays(GL_TRIANGLES, …)`
	`glDrawElements(GL_TRIANGLES, …)`

`glCallList`	

`glBitmap`
`glDrawPixels`
	`glTexSubImage2D`
	`glDrawArrays / glBlitFramebuffer`

`glVertexPointer`
`glTexCoordPointer`
`glColorPointer`
etc…
	`glVertexAttribPointer`
	`glBindAttribLocation(…, idx, "myVerts")`

`glEnableClientState(GL_COLOR_ARRAY)`	`glEnableVertexAttribArray(idx)`

Replace Matrix Transformations

Use GLKMath

- Built-in transformations deprecated
- Compute yourself—Or use GL Kit

Legacy	Core Profile with GL Kit

`glTranslate`
`glRotate`
`glScale`
	`GLKMatrix4MakeTranslate`
	`GLKMatrix4Rotate`
	`GLKMatrix4Scale`

`gluPerspective`	`GLKMatrix4MakePerspective`
`glPushMatrix` `glPopMatrix`	`GLKMatrixStackPush` `GLKMatrixStackPop`
`glLoadMatrixf`	`glUniformMatrix4fv`

GLKMath

GLKMatrix3Add
GLKMatrix3GetColumn
GLKMatrix3GetMatrix2
GLKMatrix3GetRow
GLKMatrix3Make
GLKMatrix3MakeAndTranspose
GLKMatrix3MakeRotation
GLKMatrix3MakeScale
GLKMatrix3MakeWithArray
GLKMatrix3MakeWithArrayAndTranspose
GLKMatrix3MakeWithColumns
GLKMatrix3MakeWithQuaternion
GLKMatrix3MakeWithRows
GLKMatrix3MakeXRotation
GLKMatrix3MakeYRotation
GLKMatrix3MakeZRotation
GLKMatrix3Multiply
GLKMatrix3MultiplyVector3
GLKMatrix3MultiplyVector3Array
GLKMatrix3Rotate
GLKMatrix3RotateWithVector3
GLKMatrix3RotateWithVector4
GLKMatrix3RotateX
GLKMatrix3RotateY
GLKMatrix3RotateZ
GLKMatrix3Scale
GLKMatrix3ScaleWithVector3
GLKMatrix3ScaleWithVector4
GLKMatrix3SetColumn
GLKMatrix3SetRow
GLKMatrix3Subtract
GLKMatrix3Transpose
GLKMatrix4Add
GLKMatrix4GetColumn
GLKMatrix4GetMatrix2
GLKMatrix4GetMatrix3
GLKMatrix4GetRow
GLKMatrix4Make
GLKMatrix4MakeAndTranspose
GLKMatrix4MakeFrustum
GLKMatrix4MakeLookAt
GLKMatrix4MakeOrtho
GLKMatrix4MakePerspective
GLKMatrix4MakeRotation
GLKMatrix4MakeScale
GLKMatrix4MakeTranslation
GLKMatrix4MakeWithArray
GLKMatrix4MakeWithArrayAndTranspose
GLKMatrix4MakeWithColumns
GLKMatrix4MakeWithQuaternion
GLKMatrix4MakeWithRows
GLKMatrix4MakeXRotation
GLKMatrix4MakeYRotation
GLKMatrix4MakeZRotation
GLKMatrix4Multiply
GLKMatrix4MultiplyAndProjectVector3
GLKMatrix4MultiplyAndProjectVector3Array
GLKMatrix4MultiplyVector3
GLKMatrix4MultiplyVector3Array
GLKMatrix4MultiplyVector3ArrayWithTranslation
GLKMatrix4MultiplyVector3WithTranslation
GLKMatrix4MultiplyVector4
GLKMatrix4MultiplyVector4Array
GLKMatrix4Rotate
GLKMatrix4RotateWithVector3
GLKMatrix4RotateWithVector4
GLKMatrix4RotateX
GLKMatrix4RotateY
GLKMatrix4RotateZ
GLKMatrix4Scale
GLKMatrix4ScaleWithVector3
GLKMatrix4ScaleWithVector4
GLKMatrix4SetColumn
GLKMatrix4SetRow
GLKMatrix4Subtract
GLKMatrix4Translate
GLKMatrix4TranslateWithVector3
GLKMatrix4TranslateWithVector4
GLKMatrix4Transpose
GLKQuaternionAdd
GLKQuaternionConjugate
GLKQuaternionInvert
GLKQuaternionLength
GLKQuaternionMake
GLKQuaternionMakeWithAngleAndAxis
GLKQuaternionMakeWithAngleAndVector3Axis
GLKQuaternionMakeWithArray
GLKQuaternionMakeWithVector3
GLKQuaternionMultiply
GLKQuaternionNormalize
GLKQuaternionRotateVector3
GLKQuaternionRotateVector4
GLKQuaternionSubtract
GLKVector2Add
GLKVector2AddScalar
GLKVector2AllEqualToScalar
GLKVector2AllEqualToVector2
GLKVector2AllGreaterThanOrEqualToScalar
GLKVector2AllGreaterThanOrEqualToVector2
GLKVector2AllGreaterThanScalar
GLKVector2AllGreaterThanVector2
GLKVector2Distance
GLKVector2Divide
GLKVector2DivideScalar
GLKVector2DotProduct
GLKVector2Length
GLKVector2Lerp
GLKVector2Make
GLKVector2MakeWithArray
GLKVector2Maximum
GLKVector2Minimum
GLKVector2Multiply
GLKVector2MultiplyScalar
GLKVector2Negate
GLKVector2Normalize
GLKVector2Project
GLKVector2Subtract
GLKVector2SubtractScalar
GLKVector3Add
GLKVector3AddScalar
GLKVector3AllEqualToScalar
GLKVector3AllEqualToVector3
GLKVector3AllGreaterThanOrEqualToScalar
GLKVector3AllGreaterThanOrEqualToVector3
GLKVector3AllGreaterThanScalar
GLKVector3AllGreaterThanVector3
GLKVector3CrossProduct
GLKVector3Distance
GLKVector3Divide
GLKVector3DivideScalar
GLKVector3DotProduct
GLKVector3Length
GLKVector3Lerp
GLKVector3Make
GLKVector3MakeWithArray
GLKVector3Maximum
GLKVector3Minimum
GLKVector3Multiply
GLKVector3MultiplyScalar
GLKVector3Negate
GLKVector3Normalize
GLKVector3Project
GLKVector3Subtract
GLKVector3SubtractScalar
GLKVector4Add
GLKVector4AddScalar
GLKVector4AllEqualToScalar
GLKVector4AllEqualToVector4
GLKVector4AllGreaterThanOrEqualToScalar
GLKVector4AllGreaterThanOrEqualToVector4
GLKVector4AllGreaterThanScalar
GLKVector4AllGreaterThanVector4
GLKVector4CrossProduct
GLKVector4Distance
GLKVector4Divide
GLKVector4DivideScalar
GLKVector4DotProduct
GLKVector4Length
GLKVector4Lerp
GLKVector4Make
GLKVector4MakeWithArray
GLKVector4MakeWithVector3
GLKVector4Maximum
GLKVector4Minimum
GLKVector4Multiply
GLKVector4MultiplyScalar
GLKVector4Negate
GLKVector4Normalize
GLKVector4Project
GLKVector4Subtract
GLKVector4SubtractScalar
GLKMathDegreesToRadians

GLKMath

GLKMatrix3Add
GLKMatrix3GetColumn
GLKMatrix3GetMatrix2
GLKMatrix3GetRow
GLKMatrix3Make
GLKMatrix3MakeAndTranspose
GLKMatrix3MakeRotation
GLKMatrix3MakeScale
GLKMatrix3MakeWithArray
GLKMatrix3MakeWithArrayAndTranspose
GLKMatrix3MakeWithColumns
GLKMatrix3MakeWithQuaternion
GLKMatrix3MakeWithRows
GLKMatrix3MakeXRotation
GLKMatrix3MakeYRotation
GLKMatrix3MakeZRotation
GLKMatrix3Multiply
GLKMatrix3MultiplyVector3
GLKMatrix3MultiplyVector3Array
GLKMatrix3Rotate
GLKMatrix3RotateWithVector3
GLKMatrix3RotateWithVector4
GLKMatrix3RotateX
GLKMatrix3RotateY
GLKMatrix3RotateZ
GLKMatrix3Scale
GLKMatrix3ScaleWithVector3
GLKMatrix3ScaleWithVector4
GLKMatrix3SetColumn
GLKMatrix3SetRow
GLKMatrix3Subtract
GLKMatrix3Transpose
GLKMatrix4Add
GLKMatrix4GetColumn
GLKMatrix4GetMatrix2
GLKMatrix4GetMatrix3
GLKMatrix4GetRow
GLKMatrix4Make
GLKMatrix4MakeAndTranspose
GLKMatrix4MakeFrustum
GLKMatrix4MakeLookAt
GLKMatrix4MakeOrtho
GLKMatrix4MakePerspective

GLKMatrix4MakeRotation
GLKMatrix4MakeScale
GLKMatrix4MakeTranslation
GLKMatrix4MakeWithArray
GLKMatrix4MakeWithArrayAndTranspose
GLKMatrix4MakeWithColumns
GLKMatrix4MakeWithQuaternion
GLKMatrix4MakeWithRows
GLKMatrix4MakeXRotation
GLKMatrix4MakeYRotation
GLKMatrix4MakeZRotation
GLKMatrix4Multiply
GLKMatrix4MultiplyAndProjectVector3
GLKMatrix4MultiplyAndProjectVector3Array
GLKMatrix4MultiplyVector3
GLKMatrix4MultiplyVector3Array
GLKMatrix4MultiplyVector3ArrayWithTranslation
GLKMatrix4MultiplyVector3WithTranslation
GLKMatrix4MultiplyVector4
GLKMatrix4MultiplyVector4Array
GLKMatrix4Rotate
GLKMatrix4RotateWithVector3
GLKMatrix4RotateWithVector4
GLKMatrix4RotateX
GLKMatrix4RotateY
GLKMatrix4RotateZ
GLKMatrix4Scale
GLKMatrix4ScaleWithVector3
GLKMatrix4ScaleWithVector4
GLKMatrix4SetColumn
GLKMatrix4SetRow
GLKMatrix4Subtract
GLKMatrix4Translate
GLKMatrix4TranslateWithVector3
GLKMatrix4TranslateWithVector4
GLKMatrix4Transpose
GLKQuaternionAdd
GLKQuaternionConjugate
GLKQuaternionInvert
GLKQuaternionLength
GLKQuaternionMake
GLKQuaternionMakeWithAngleAndAxis
GLKQuaternionMakeWithAngleAndVector3Axis

GLKQuaternionMakeWithArray
GLKQuaternionMakeWithVector3
GLKQuaternionMultiply
GLKQuaternionNormalize
GLKQuaternionRotateVector3
GLKQuaternionRotateVector4
GLKQuaternionSubtract
GLKVector2Add
GLKVector2AddScalar
GLKVector2AllEqualToScalar
GLKVector2AllEqualToVector2
GLKVector2AllGreaterThanOrEqualToScalar
GLKVector2AllGreaterThanOrEqualToVector2
GLKVector2AllGreaterThanScalar
GLKVector2AllGreaterThanVector2
GLKVector2Distance
GLKVector2Divide
GLKVector2DivideScalar
GLKVector2DotProduct
GLKVector2Length
GLKVector2Lerp
GLKVector2Make
GLKVector2MakeWithArray
GLKVector2Maximum
GLKVector2Minimum
GLKVector2Multiply
GLKVector2MultiplyScalar
GLKVector2Negate
GLKVector2Normalize
GLKVector2Project
GLKVector2Subtract
GLKVector2SubtractScalar
GLKVector3Add
GLKVector3AddScalar
GLKVector3AllEqualToScalar
GLKVector3AllEqualToVector3
GLKVector3AllGreaterThanOrEqualToScalar
GLKVector3AllGreaterThanOrEqualToVector3
GLKVector3AllGreaterThanScalar
GLKVector3AllGreaterThanVector3
GLKVector3CrossProduct
GLKVector3Distance
GLKVector3Divide

GLKVector3DivideScalar
GLKVector3DotProduct
GLKVector3Length
GLKVector3Lerp
GLKVector3Make
GLKVector3MakeWithArray
GLKVector3Maximum
GLKVector3Minimum
GLKVector3Multiply
GLKVector3MultiplyScalar
GLKVector3Negate
GLKVector3Normalize
GLKVector3Project
GLKVector3Subtract
GLKVector3SubtractScalar
GLKVector4Add
GLKVector4AddScalar
GLKVector4AllEqualToScalar
GLKVector4AllEqualToVector4
GLKVector4AllGreaterThanOrEqualToScalar
GLKVector4AllGreaterThanOrEqualToVector4
GLKVector4AllGreaterThanScalar
GLKVector4AllGreaterThanVector4
GLKVector4CrossProduct
GLKVector4Distance
GLKVector4Divide
GLKVector4DivideScalar
GLKVector4DotProduct
GLKVector4Length
GLKVector4Lerp
GLKVector4Make
GLKVector4MakeWithArray
GLKVector4MakeWithVector3
GLKVector4Maximum
GLKVector4Minimum
GLKVector4Multiply
GLKVector4MultiplyScalar
GLKVector4Negate
GLKVector4Normalize
GLKVector4Project
GLKVector4Subtract
GLKVector4SubtractScalar
GLKMathDegreesToRadians

GLKMath provides all your
Core Profile matrix needs

Replace Fixed-Function State

Use GLKBaseEffect

- Fixed-function lighting, materials, texturing deprecated
- Replace with light, material, texture properties in GLKBaseEffect

Legacy	Core Profile with GL Kit

`glLightfv(…, GL_POSITION, …)`
`glLightfv(…, GL_DIFFUSE, …)`
`glLightfv(…, GL_SPECULAR, …)`
	`baseEffect.light0.position`
	`baseEffect.light0.diffuseColor`
	`baseEffect.light0.specularColor`

`glEnable(GL_LIGHT0)`	`baseEffect.light0.enabled`
`glMaterialfv`	`baseEffect.material.diffuseColor`

Update Existing Shaders to #version 150+

Update OpenGL API usage

- Pass data up in generic vertex attribute arrays
- Upload matrices and current vertex state as uniforms

Legacy	Core Profile
`glEnableClientState(GL_COLOR_ARRAY)`	`glEnableVertexAttribArray(index)`
`glLoadMatrixf`	`glUniformMatrix4fv`
`glColor4fv` *etc…*	`glVertexAttrib4fv` `glUniform4fv`

`glVertexPointer`
`glTexCoordPointer`
`glColorPointer`
etc…
	`glVertexAttribPointer`

Update Existing Shaders to #version 150+

Update GLSL shaders

- Ins and outs are now explicit
- Call `glBindFragDataLocation` prior to linking

Legacy	150+ Vertex	150+ Fragment
`attribute vec4 data`	`in vec4 data`	
`varying vec2 texCoord`	`out vec2 texCoord`	`in vec2 texCoord`
`gl_FragColor`		`glBindFragDataLocation(…)` `out vec4 myColor`

Update Existing Shaders to #version 150+

Update GLSL shaders

- GLSL version now required
- Load built-ins as generic attributes and uniforms

Legacy	150+ Vertex/Fragment Shader
`#version 110`	`#version 150 / 330 / 410`

`gl_Vertex`
`gl_Normal`
`gl_MultiTexCoord0`
	`in vec4 vertPos`
	`in vec3 inNormal`
	`in vec2 texCoord`

`gl_ModelViewProjectionMatrix`
`gl_NormalMatrix`
	`uniform mat4 mvpMatrix`
	`uniform mat3 normalMatrix`

`texture2D(…)`
`texture3D(…)`
etc…
	`texture(…)`

Final Touches

Other API differences

- Include only “gl3” headers if possible
- Update to use core functions

Legacy	Core Profile

`#include <OpenGL/gl.h>`
`#include <OpenGL/glext.h>`
	`#include <OpenGL/gl3.h>`
	`#include <OpenGL/gl3ext.h>`

`glGetString(GL_EXTENSIONS)`	`glGetIntegerv(GL_NUM_EXTENSIONS)`
	`glGetStringi(GL_EXTENSIONS, <index>)`

`glSetFenceAPPLE(…)`
`glTestFenceAPPLE(…)`
	`glFenceSync(…)`
	`glWaitSync(…)`

`glGenVertexArraysAPPLE(1, &vao)`
`glBindVertexArrayAPPLE(vao)`
	`glGenVertexArrays(1, &vao)`
	`glBindVertexArray(vao)`

Migrating to OpenGL Core Profile

Piecemeal approach

Migrating to OpenGL Core Profile

Piecemeal approach

- Draw using VBOs and VAOs

Migrating to OpenGL Core Profile

Piecemeal approach

- Draw using VBOs and VAOs
- Replace matrix math using `glLoadMatrixf`
  - GLKMath is profile-agnostic

Migrating to OpenGL Core Profile

Piecemeal approach

- Draw using VBOs and VAOs
- Replace matrix math using `glLoadMatrixf`
    - GLKMath is profile-agnostic
- Update existing shaders
    - Use generic attributes, uniforms for built-ins
    - `GL_EXT_gpu_shader4` enables `glBindFragDataLocation`

Migrating to OpenGL Core Profile

Piecemeal approach

- Draw using VBOs and VAOs
- Replace matrix math using `glLoadMatrixf`
    - GLKMath is profile-agnostic
- Update existing shaders
    - Use generic attributes, uniforms for built-ins
    - `GL_EXT_gpu_shader4` enables `glBindFragDataLocation`
- Replace fixed-functions with custom shaders
    - GLKEffects only works with Core Profile

Migrating to OpenGL Core Profile

Piecemeal approach

- Draw using VBOs and VAOs
- Replace matrix math using `glLoadMatrixf`
  - GLKMath is profile-agnostic
- Update existing shaders
  - Use generic attributes, uniforms for built-ins
  - `GL_EXT_gpu_shader4` enables `glBindFragDataLocation`
- Replace fixed-functions with custom shaders
  - GLKEffects only works with Core Profile
- Lastly, switch to Core Profile and update shader versions

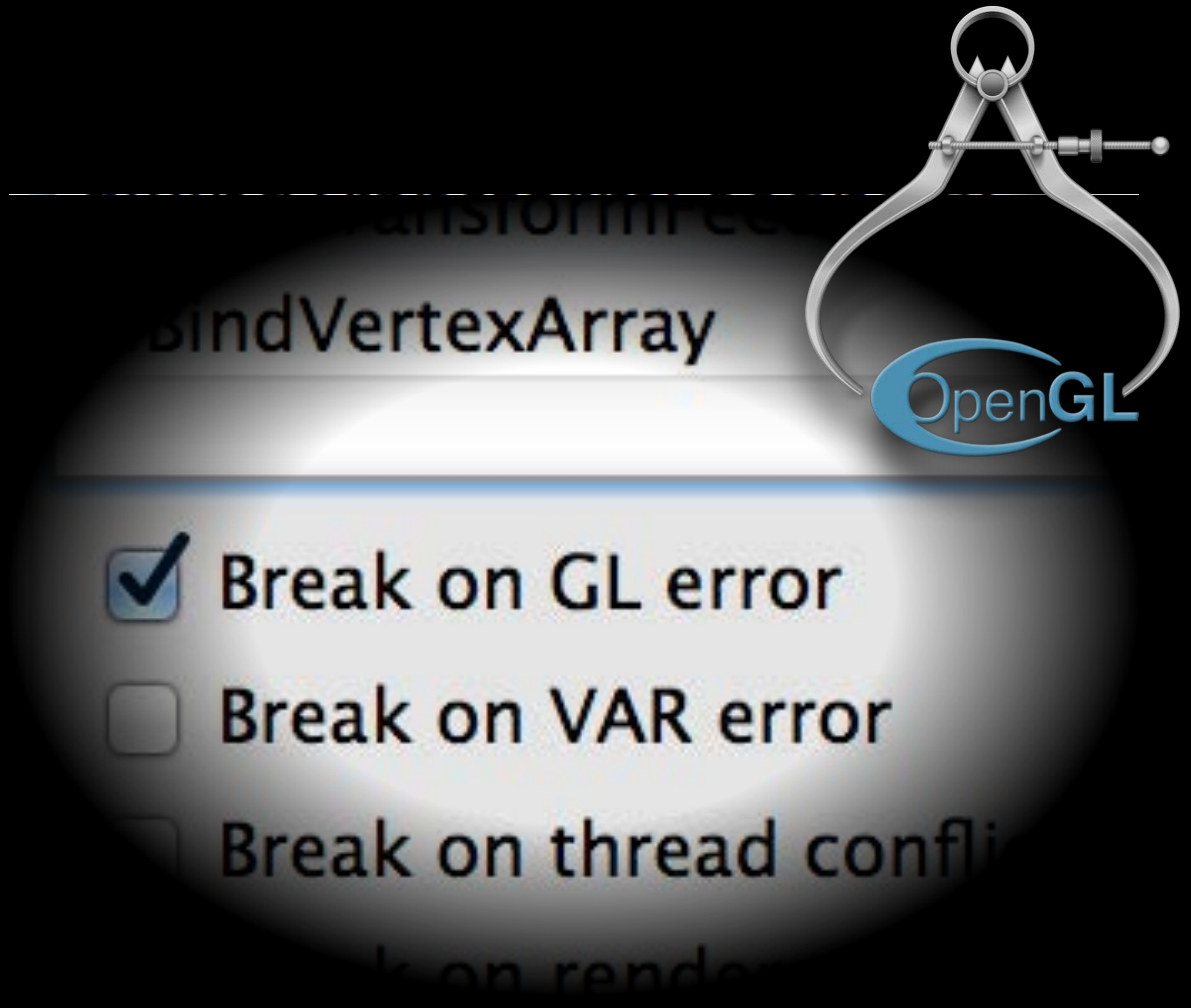
Migrating to OpenGL Core Profile

Piecemeal approach

- Draw using VBOs and VAOs
- Replace matrix math using `glLoadMatrixf`
  - GLKMath is profile-agnostic
- Update existing shaders
  - Use generic attributes, uniforms for built-ins
  - `GL_EXT_gpu_shader4` enables `glBindFragDataLocation`
- Replace fixed-functions with custom shaders
  - GLKEffects only works with Core Profile
- Lastly, switch to Core Profile and update shader versions
- Tip: Grep for legacy strings to track progress

Summary and Tips

- Access new features in Core Profile
- Debug with OpenGL Profiler
  - No need to call `glGetError`
- Use OpenGL with OpenCL for compute


BindVertexArray

- [✓] Break on GL error
- [ ] Break on VAR error
- [ ] Break on thread confli

More Information

Allan Schaffer
Graphics and Game Technologies Evangelist
aschaffer@apple.com

Documentation
OpenGL for OS X
http://developer.apple.com/opengl

Apple Developer Forums
http://devforums.apple.com

Related Sessions

Advances in OpenGL ES	Mission Thursday 9:00AM
Working with OpenCL	Marina Thursday 3:15PM

Labs

OpenGL and OpenGL ES Lab	Graphics and Games Lab A Thursday 2:00PM
OpenCL Lab	Graphics and Games Lab B Thursday 4:30PM

WWDC2013