Advanced image synthesis

Romain Vergne – 2014/2015

What is it?

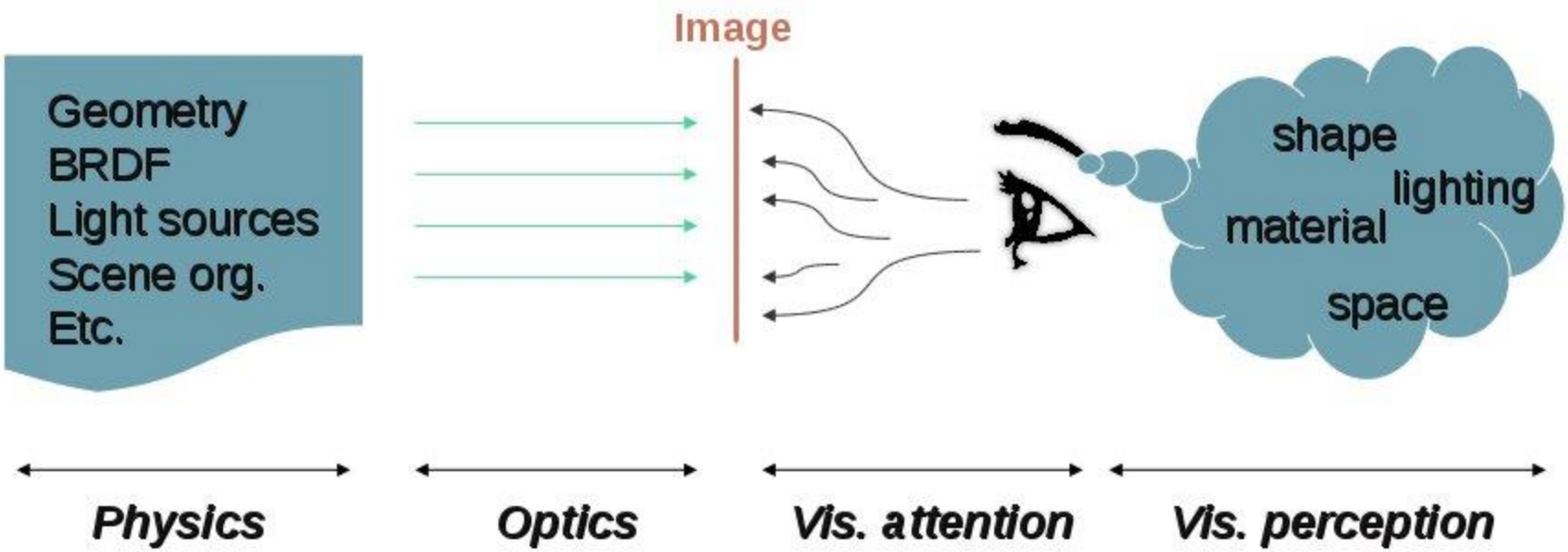


Image
Geometry
BRDF
Light sources
Scene org.
Etc.
shape
lighting
material
space
Physics
Optics
Vis. attention
Vis. perception

For what?

Video games

@Battlefield

For what?

Animations

For what?

Movies / special effects

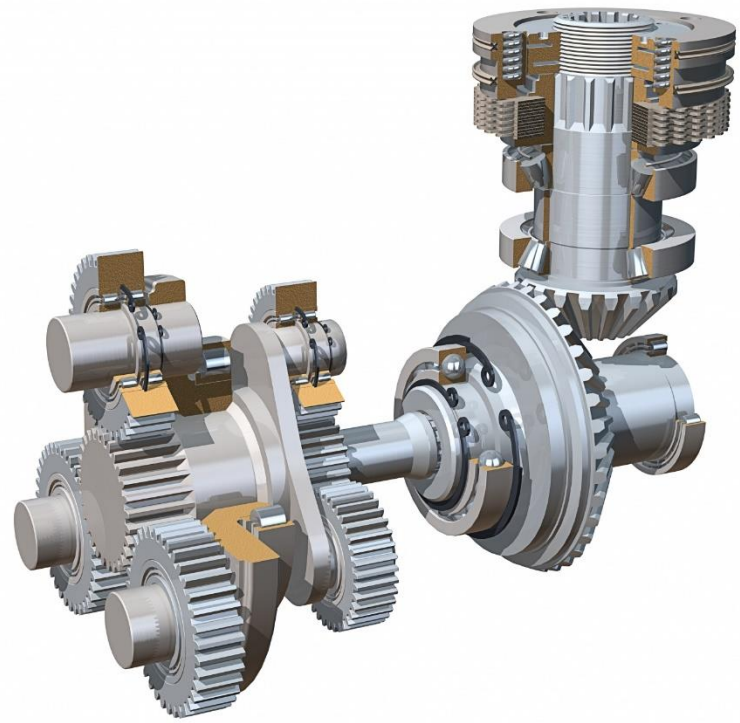
@Exodus

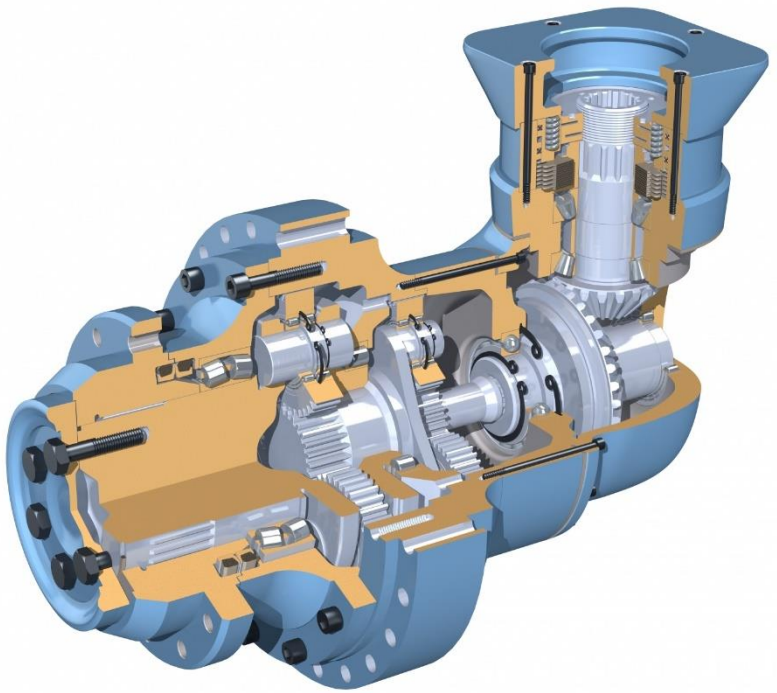
For what?

Advertising

For what?

Computer aided design

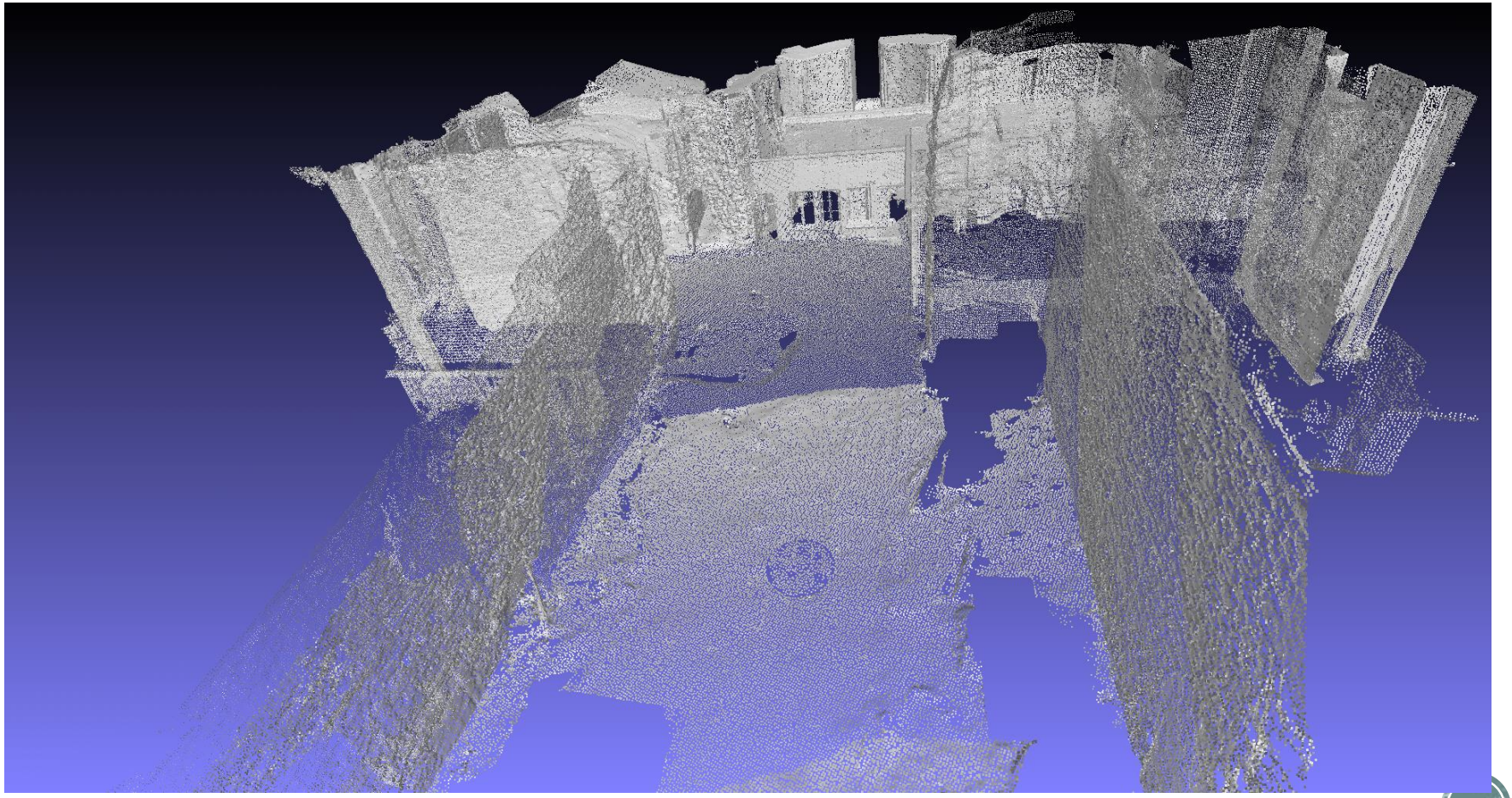




http://www.starjammer-engineering.com/cad.php

For what?

Cultural heritage



https://fredericduvivier.wordpress.com

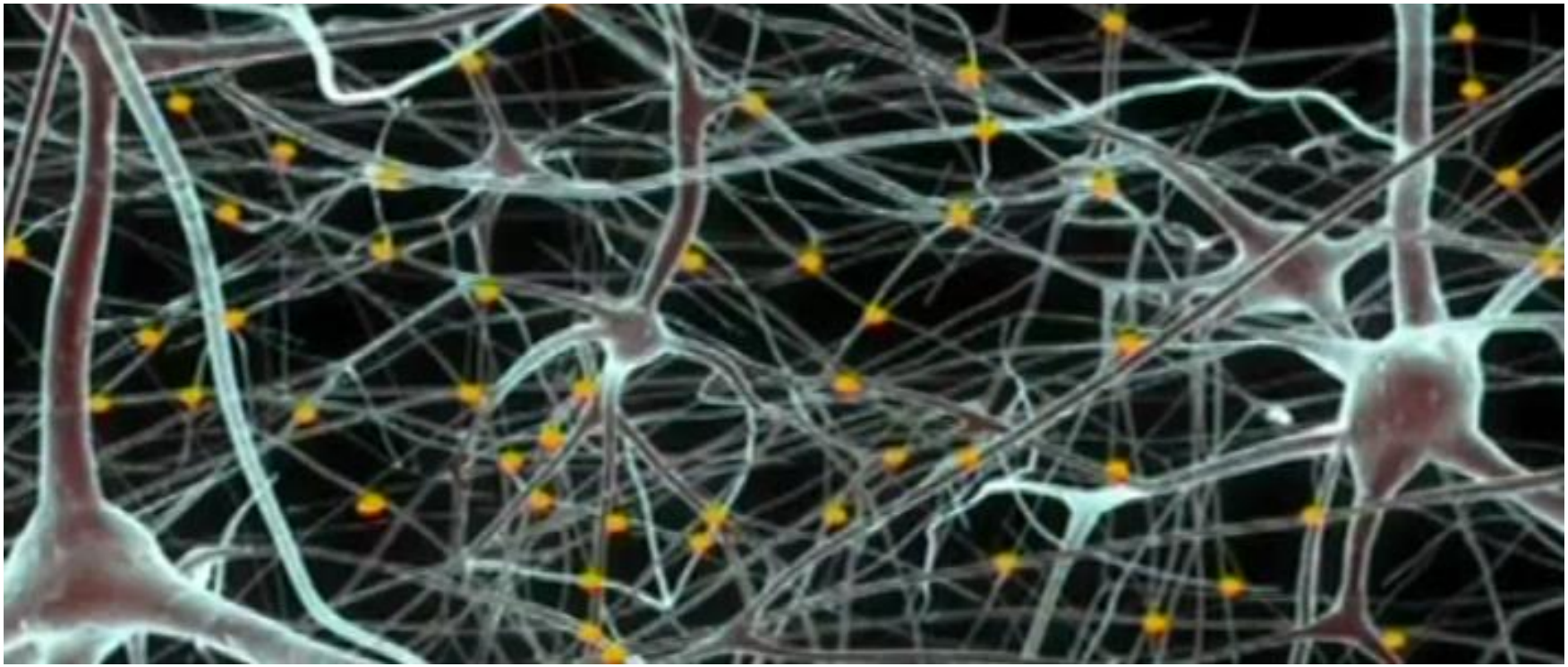
For what?

Education

http://www.nec-display-solutions.com/p/edu/en/Study3DLearning.xhtml

For what?

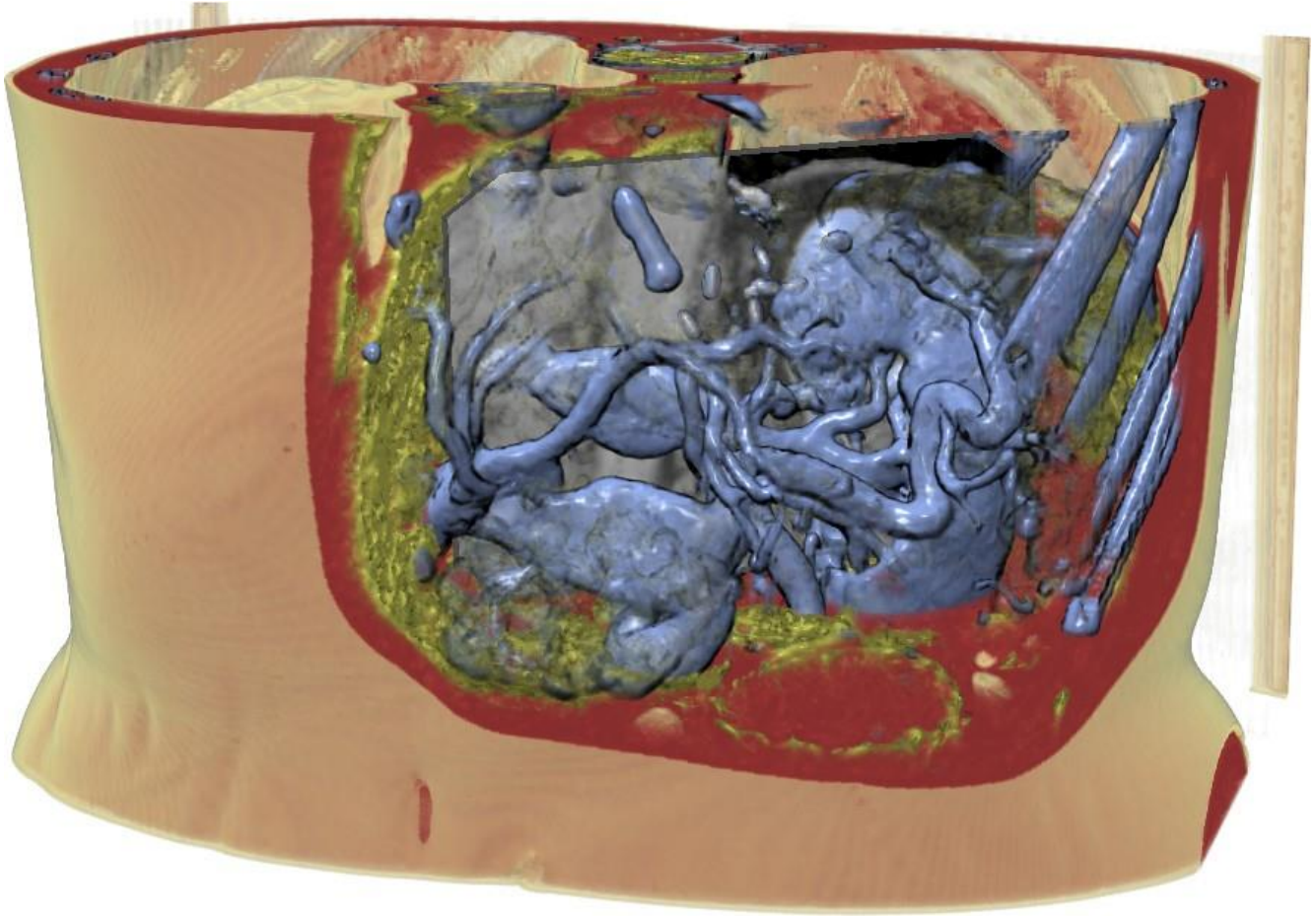
Scientific visualization



http://datavisualization.ch/showcases/scientific-visualizations-on-ted/

For what?

Medical visualization



[Burns et al. 2007]

For what?

Simulation

http://printf.eu/le-meilleur-simulateur-de-vol/

For what?

Virtual reality

http://www.telegraph.co.uk

More than you would expect…

http://vimeo.com/9553622

What you will learn

Lectures

- Graphics pipelines
- Ray casting
- Ray tracing
- Shading & material appearance
- Global illumination

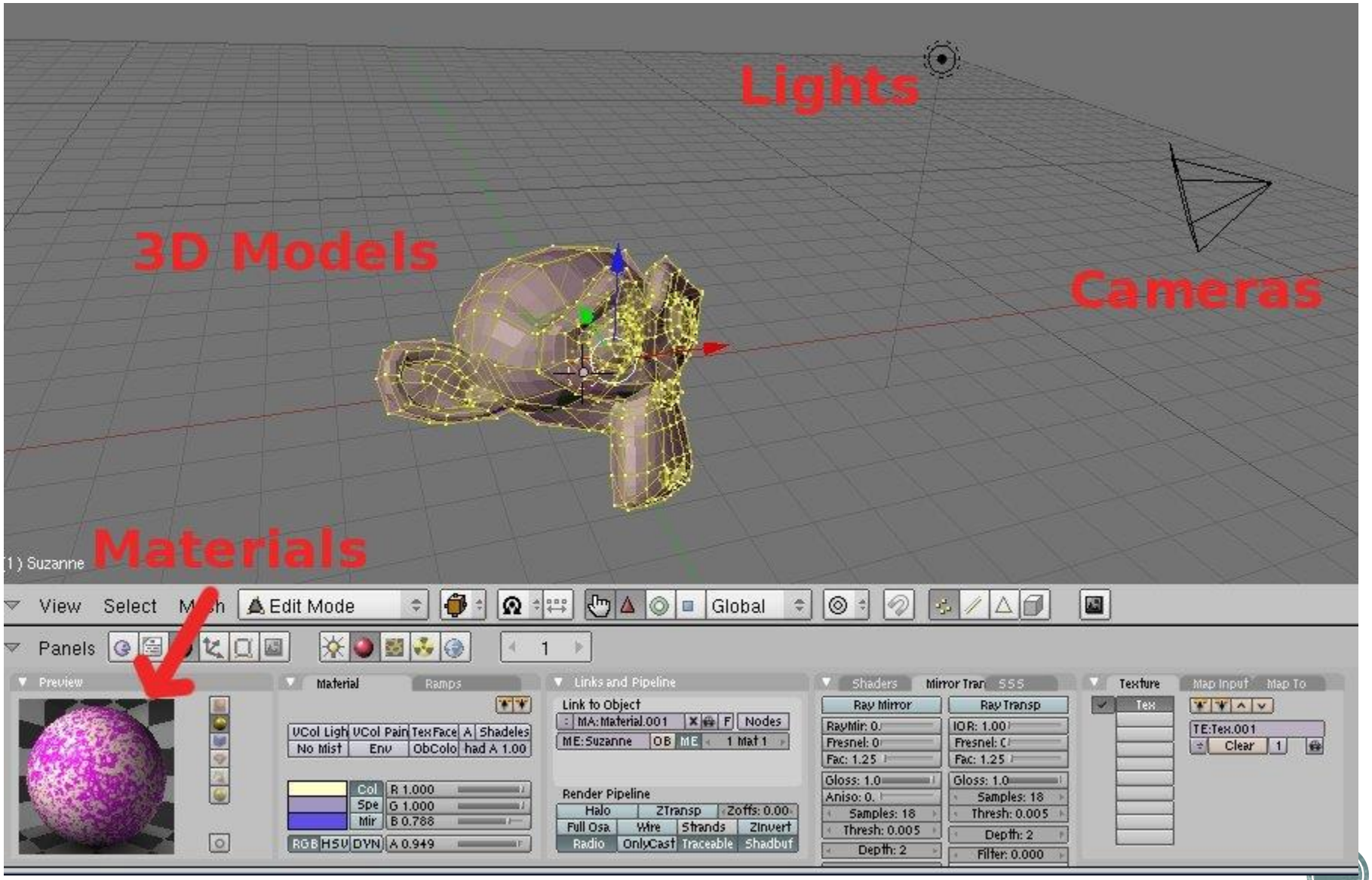
Exercices

- Everything will be done in GLSL
  - (https://www.opengl.org/documentation/glsl/)
- In Gratin
  - (https://gforge.inria.fr/projects/gratin/)

What you will NOT learn

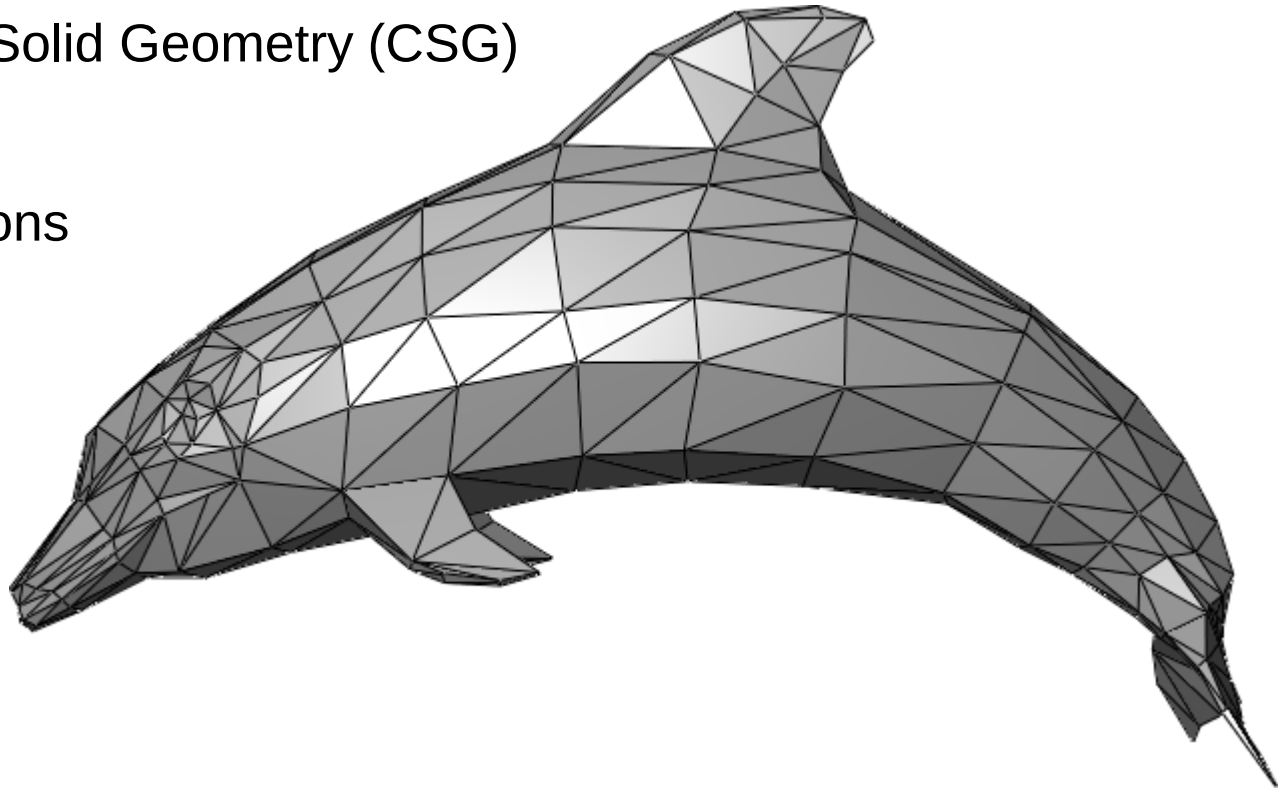
- OpenGL (https://www.opengl.org/)
- Object modeling (http://stephaneginier.com/sculptgl/)
- Hierarchical modeling (scene graphs)
- Skinning for animation (http://billbaxter.com/courses/290/html/)
- Particle systems (http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-837-computer-graphics-fall-2012/lecture-notes/MIT6_837F12_Lec07.pdf)
- …

Simplified pipeline



3D models

- Meshes
- Particle / splats
- Boundary representation
- Constructive Solid Geometry (CSG)
- Billboards
- Implicit functions
- …



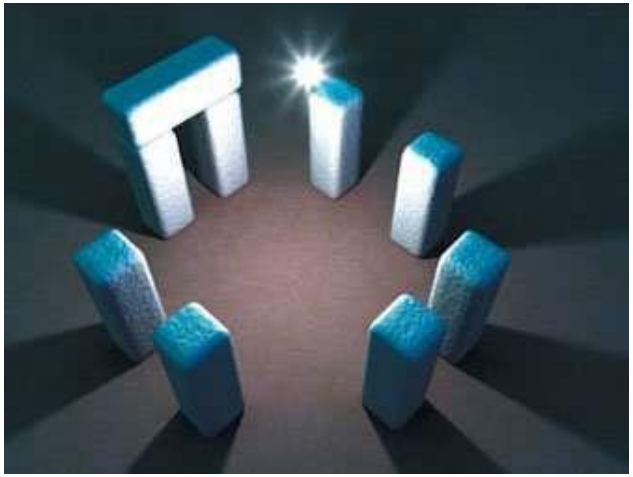
Materials

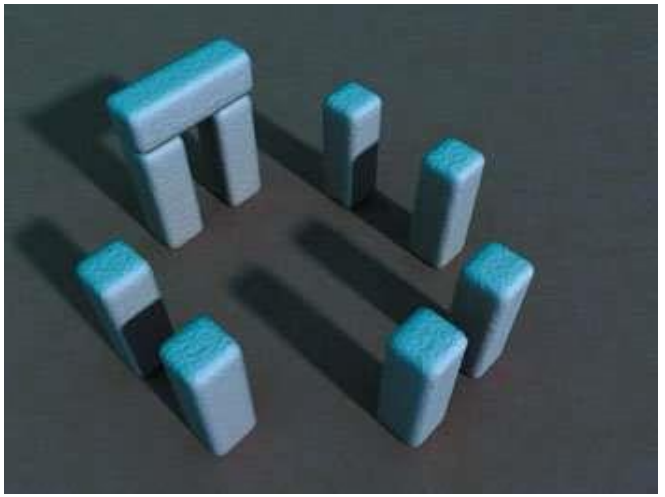
- Matte
- Glossy
- Transparent
- Translucent
- Mirror
- …

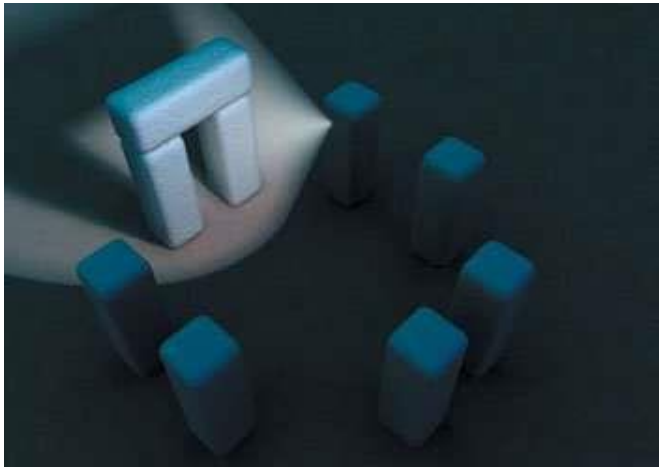
http://paulw.deviantart.com/art/18-Free-VRay-Materials-85537243

Lights

- Point light
- Directional light
- Spot light
- Area light
- …





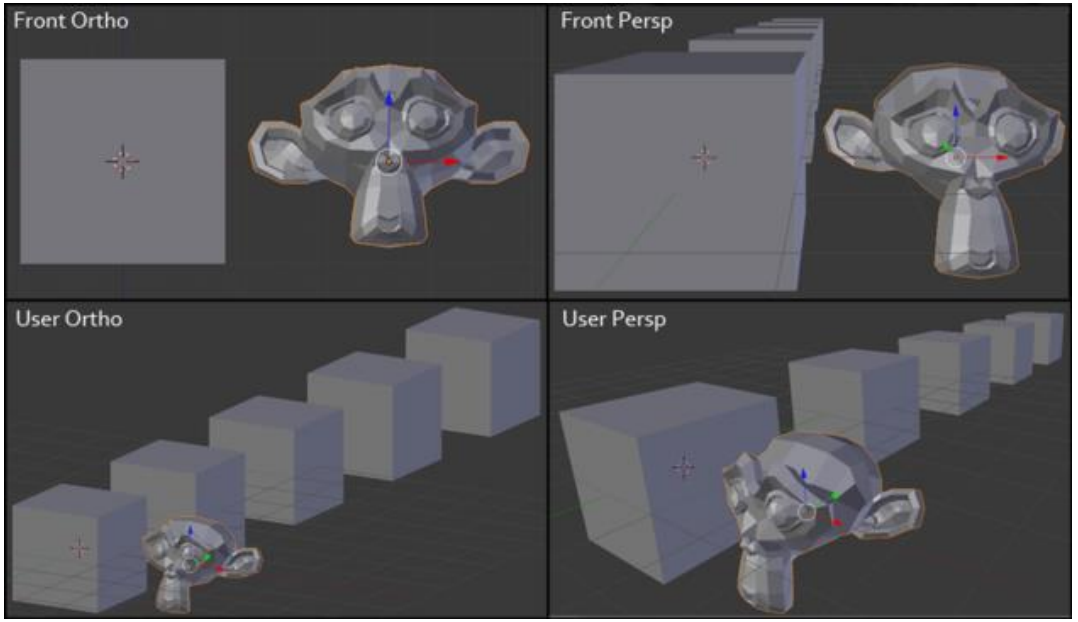


http://digital-lighting.150m.com/ch02lev1sec2.html

Cameras

- Orthographic
- Perspective
- Fish eye
- Multi-perspective
- Lens properties
- …

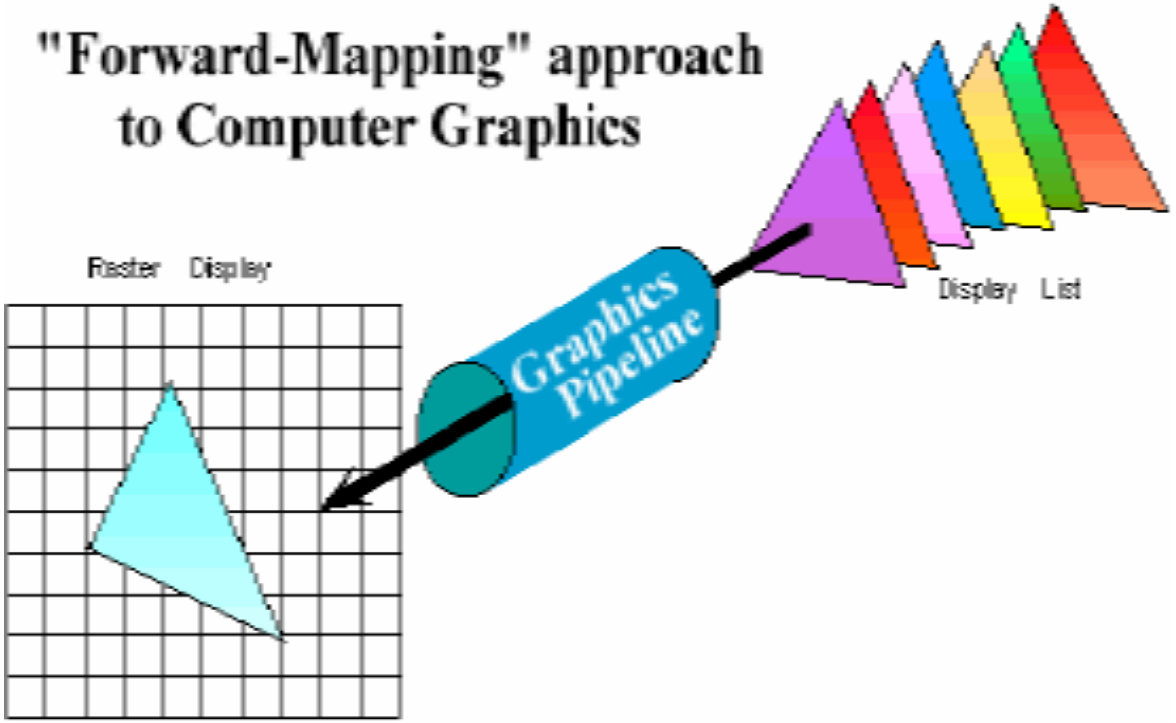
http://wiki.blender.org/index.php/Doc:2.4/Manual/3D_interaction/Navigating/3D_View



http://psd.fanextra.com/articles/50-fantastic-examples-of-fish-eye-photography/

Rasterization pipeline

- For each triangle
  - For each pixel
    - Does triangle cover pixel?
    - Kip closest hit

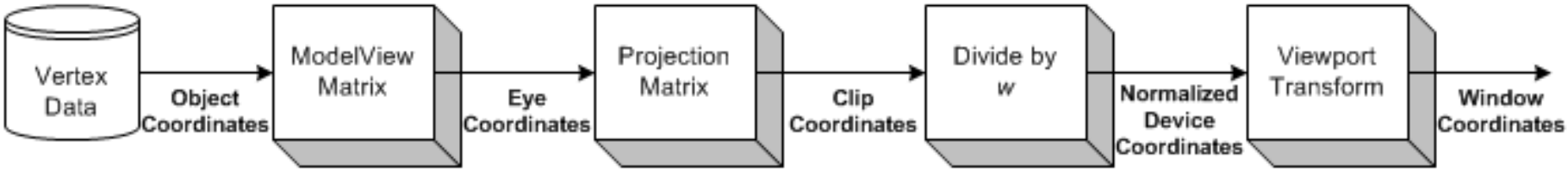


Rasterization pipeline

- Project vertices to 2D
- Rasterize triangle
- Compute per-pixel color
- Test visibility

Rasterization pipeline

- Project vertices to 2D
- Rasterize triangle
- Compute per-pixel color
- Test visibility



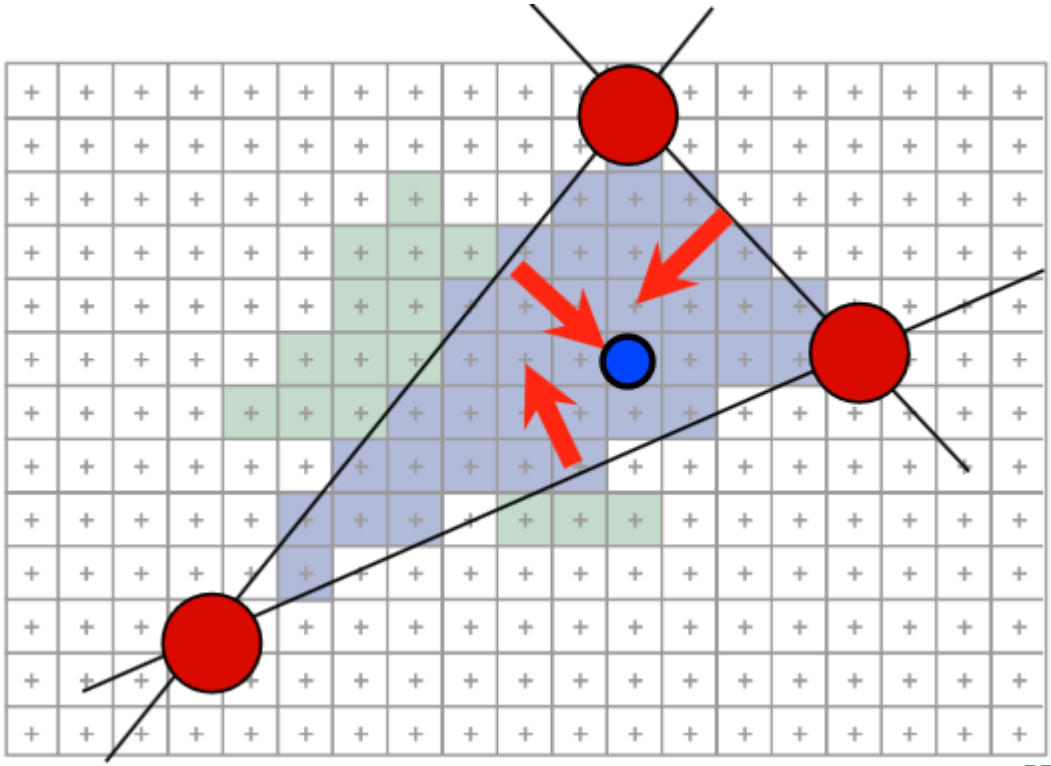
- M = model matrix
- V = view matrix
- P = projection matrix
- PVM = ModelViewProjection matrix

Read: http://romain.vergne.free.fr/teaching/IS/SI02-transformations.html

Rasterization pipeline

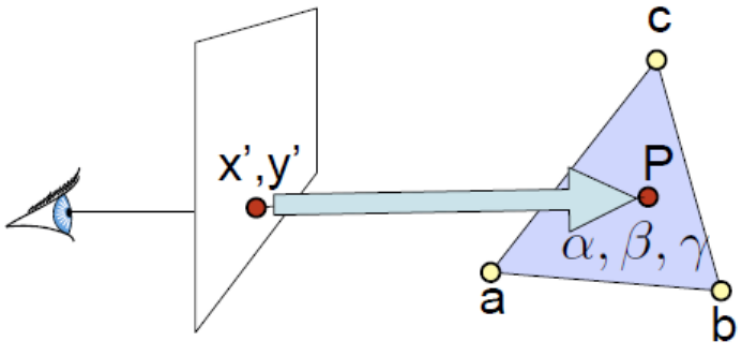
- Project vertices to 2D
- Rasterize triangle
- Compute per-pixel color
- Test visibility

- For each pixel
  - Test 3 edge equations
  - If all pass, draw



Rasterization pipeline

- Project vertices to 2D
- Rasterize triangle
- Compute per-pixel color
- Test visibility

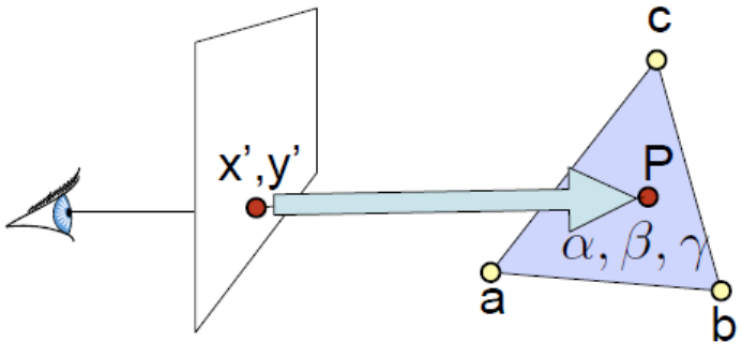


$$P(\alpha, \beta, \gamma) = \alpha \boldsymbol{a} + \beta \boldsymbol{b} + \gamma \boldsymbol{c}$$

$$\alpha + \beta + \gamma = 1, \quad \alpha, \beta, \gamma \geq 0$$

Rasterization pipeline

- Project vertices to 2D
- Rasterize triangle
- Compute per-pixel color
- Test visibility



$$P(\alpha, \beta, \gamma) = \alpha \boldsymbol{a} + \beta \boldsymbol{b} + \gamma \boldsymbol{c}$$

$$\alpha + \beta + \gamma = 1, \quad \alpha, \beta, \gamma \geq 0$$

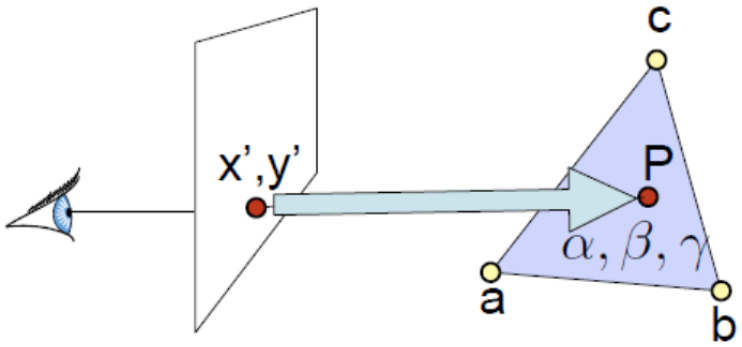
projective equivalence (up to scale) — 2D homogenous coordinates — projected vertices — barycentric coordinates

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \sim \begin{pmatrix} P'_x \\ P'_y \\ P'_w \end{pmatrix} = \begin{pmatrix} a'_x & b'_x & c'_x \\ a'_y & b'_y & c'_y \\ a'_z & b'_z & c'_z \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}$$

a', b', c' are the projected homogeneous coordinates

Rasterization pipeline

- Project vertices to 2D
- Rasterize triangle
- Compute per-pixel color
- Test visibility



$$P(\alpha, \beta, \gamma) = \alpha \boldsymbol{a} + \beta \boldsymbol{b} + \gamma \boldsymbol{c}$$

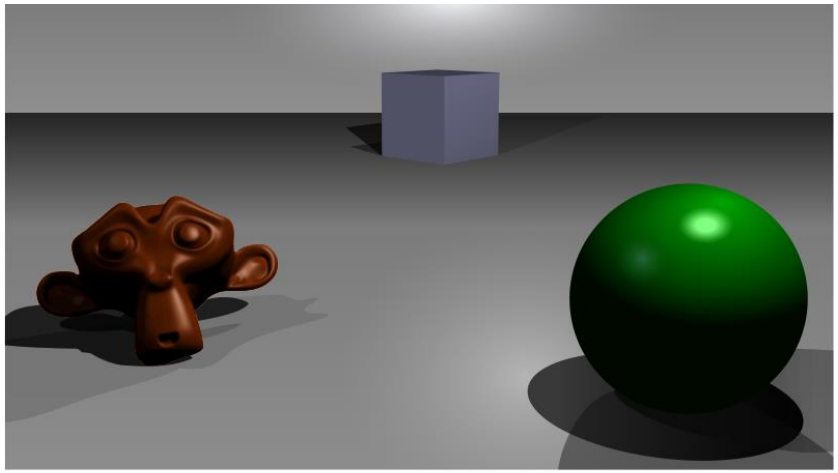
$$\alpha + \beta + \gamma = 1, \quad \alpha, \beta, \gamma \geq 0$$

$$\begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} \sim \begin{pmatrix} a'_x & b'_x & c'_x \\ a'_y & b'_y & c'_y \\ a'_w & b'_w & c'_w \end{pmatrix}^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

a', b', c' are the projected homogeneous coordinates

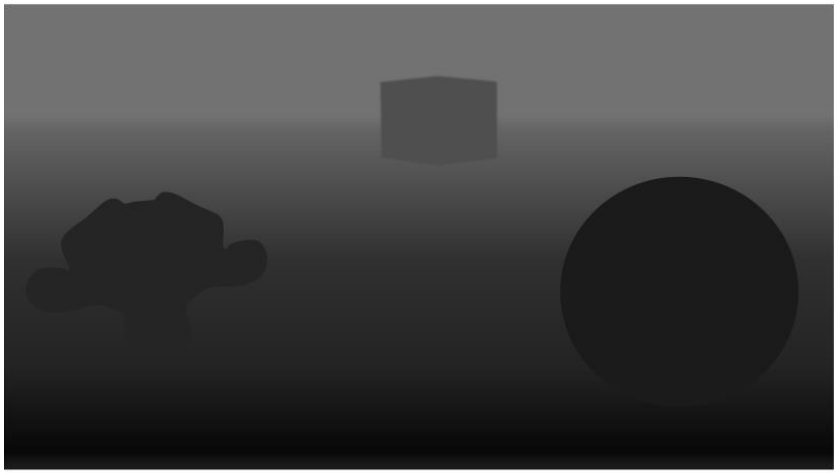
Rasterization pipeline

- Project vertices to 2D
- Rasterize triangle
- Compute per-pixel color
- Test visibility



A simple three-dimensional scene

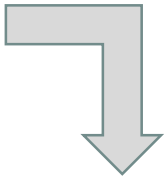
- Store minimum distance to camera for each pixel in z-buffer
- If new_z<zbuffer[x,y]
  - Zbuffer[x,y] = new_z
  - Framebuffer[x,y] = new_color



Z-buffer representation

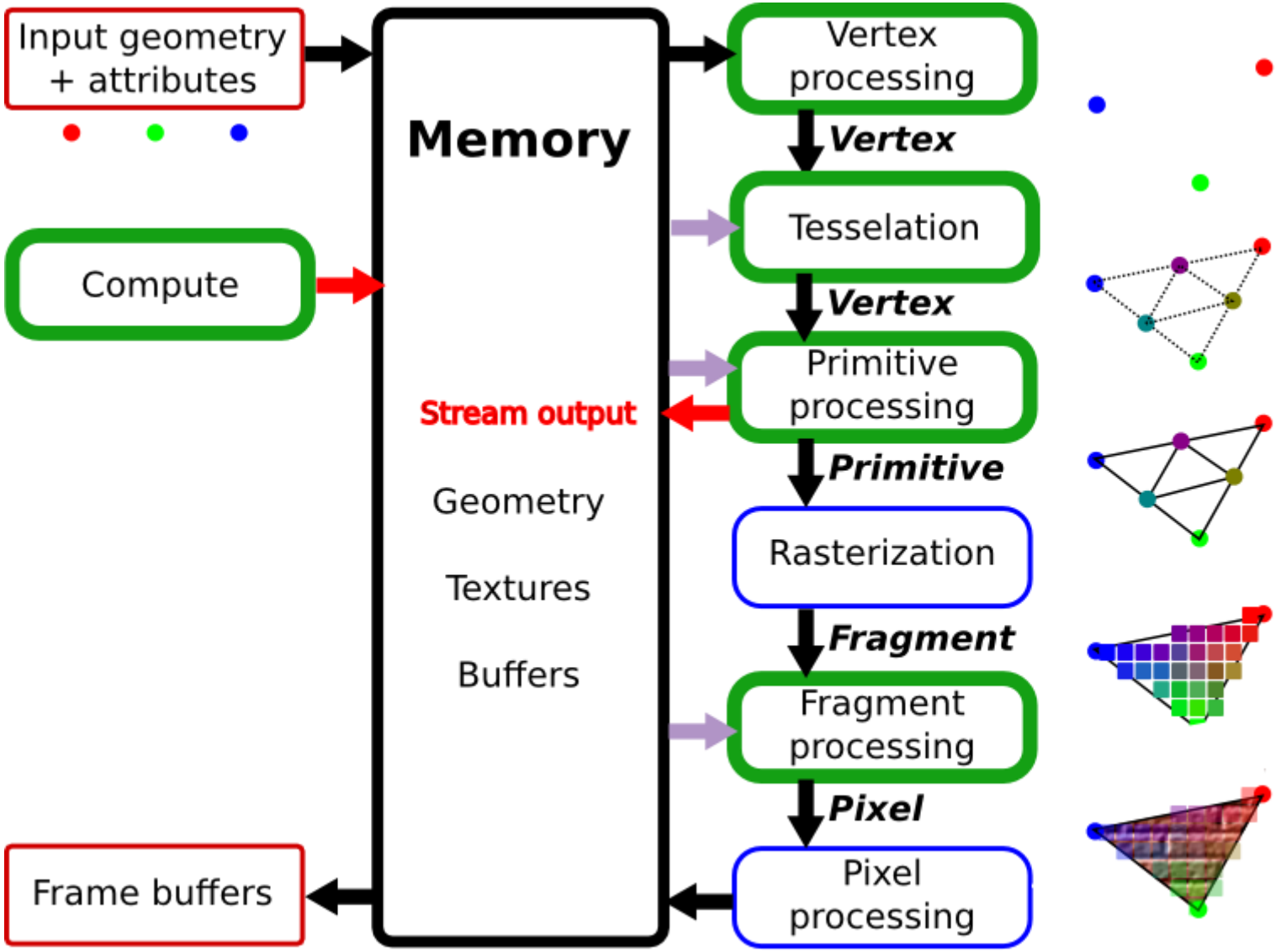
Rasterization pipeline

- For each triangle
  - For each pixel
    - Does triangle cover pixel?
    - Kip closest hit

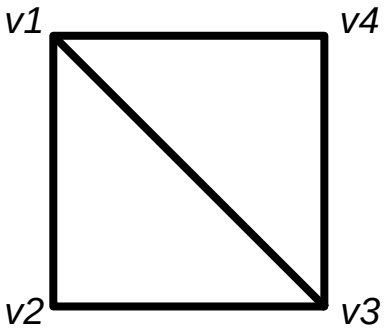


- For each triangle
  - Compute projection
  - Compute interpolation matrix
  - Compute Bbox, clip bbox to screen limits
  - For each pixel x,y in bbox
    - Test edge functions
    - If all Ei > 0
      - Compute barycentrics
      - Interpolate z from vertices
      - If z < zbuffer[x,y]
        - Interpolate attributes (color, normal)
        - Framebuffer[x,y] = resulting color

Modern graphics pipeline



Modern graphics pipeline



Input geometry
v in [-0.5,0.5]

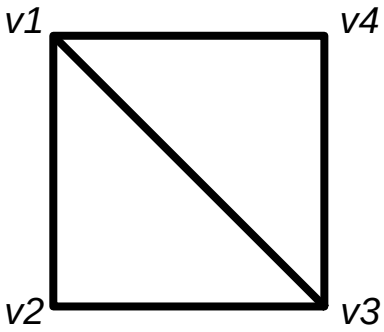
```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
}
```

Vertex shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    outBuffer0 = vec4(1,0,0,1);
}
```

Fragment shader

Modern graphics pipeline



Input geometry
v in [-0.5,0.5]

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
}
```

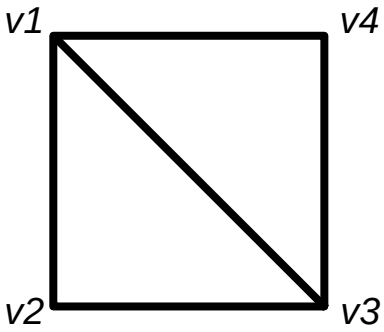
Vertex shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    outBuffer0 = vec4(1,0,0,1);
}
```

Fragment shader

GLSL Version

Modern graphics pipeline



Input geometry
v in [-0.5,0.5]

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
}
```

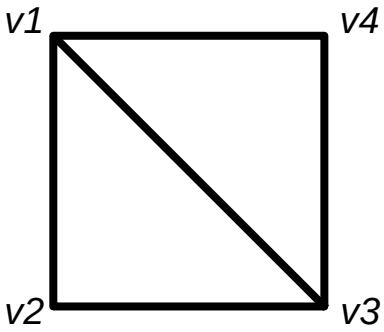
Vertex shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    outBuffer0 = vec4(1,0,0,1);
}
```

Fragment shader

Per vertex attributes

Modern graphics pipeline



Input geometry
v in [-0.5,0.5]

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
}
```

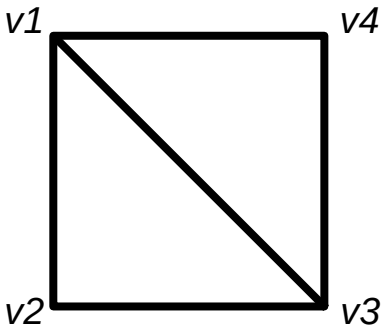
Vertex shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    outBuffer0 = vec4(1,0,0,1);
}
```

Fragment shader

Output framebuffer

Modern graphics pipeline



Input geometry
v in [-0.5,0.5]

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
}
```

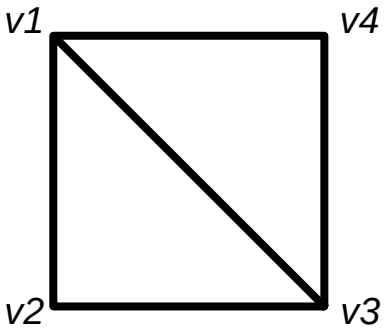
Vertex shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    outBuffer0 = vec4(1,0,0,1);
}
```

Fragment shader

Per object/frame (global) variables

Modern graphics pipeline



Input geometry
v in [-0.5,0.5]

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
}
```

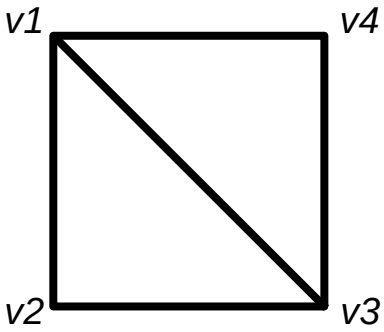
Vertex shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    outBuffer0 = vec4(1,0,0,1);
}
```

Fragment shader

Built-in variables: gl_*

Modern graphics pipeline



Input geometry
v in [-0.5,0.5]

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

10 void main() {
11     mat4 mdv   = view*model;
12     mat4 mvp   = proj*mdv;
13
14     gl_Position = mvp*vec4(inVertex,1);
15 }
16
```

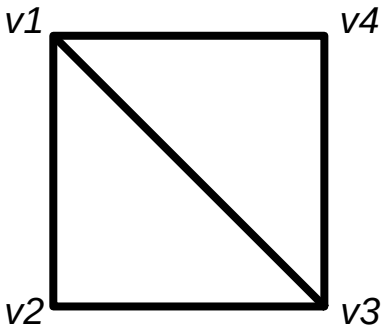
Vertex shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

 9 void main() {
10     outBuffer0 = vec4(1,0,0,1);
11 }
12
```

Fragment shader

Vertex projection

Modern graphics pipeline



Input geometry
v in [-0.5,0.5]

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
}
```

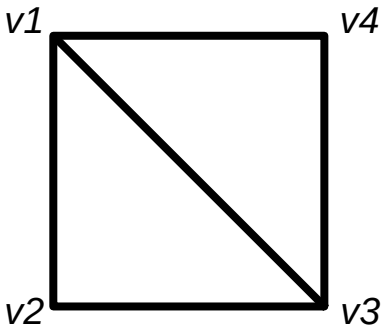
Vertex shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    outBuffer0 = vec4(1,0,0,1);
}
```

Fragment shader

Output color

Modern graphics pipeline



Input geometry
v in [-0.5,0.5]

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
}
```

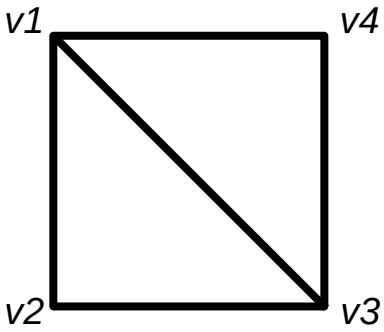
Vertex shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    outBuffer0 = vec4(1,0,0,1);
}
```

Fragment shader

Resulting rendering?

Modern graphics pipeline



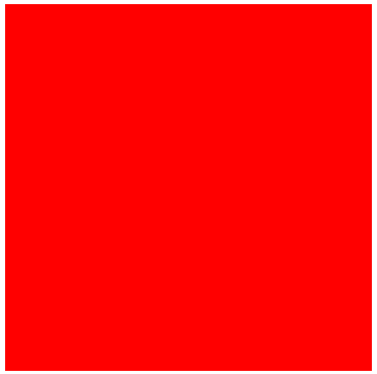
Input geometry
v in [-0.5,0.5]

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
}
```

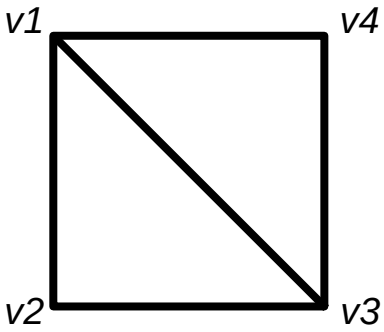
Vertex shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    outBuffer0 = vec4(1,0,0,1);
}
```

Fragment shader



Modern graphics pipeline



Input geometry
v in [-0.5,0.5]

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(location=0) out vec3 outVar;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
    outVar = inVertex+0.5;
}
```

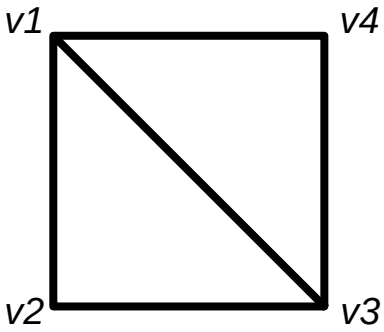
Vertex shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(location=0) in vec3 inVar;

void main() {
    outBuffer0 = vec4(inVar,1);
}
```

Fragment shader

Defining new attributes.
Rasterization is done automatically!

Modern graphics pipeline



Input geometry
v in [-0.5,0.5]

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(location=0) out vec3 outVar;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
    outVar = inVertex+0.5;
}
```

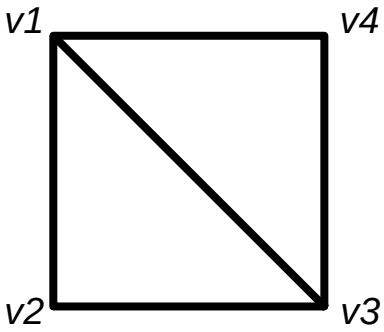
Vertex shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(location=0) in vec3 inVar;

void main() {
  outBuffer0 = vec4(inVar,1);
}
```

Fragment shader

Resulting rendering?

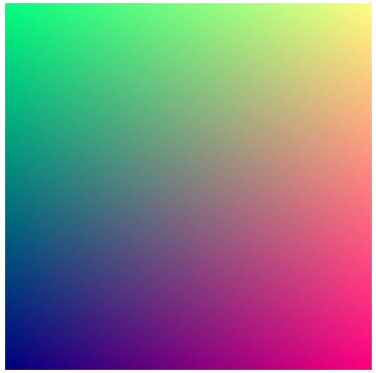
Modern graphics pipeline



Input geometry
v in [-0.5,0.5]

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(location=0) out vec3 outVar;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
    outVar = inVertex+0.5;
}
```

Vertex shader



```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(location=0) in vec3 inVar;

void main() {
    outBuffer0 = vec4(inVar,1);
}
```

Fragment shader

Modern graphics pipeline

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
}
```

Vertex shader

```
#version 420

layout(triangles) in;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(line_strip, max_vertices = 4) out;

void main() {
    gl_Position = gl_in[0].gl_Position; EmitVertex();
    gl_Position = gl_in[1].gl_Position; EmitVertex();
    gl_Position = gl_in[2].gl_Position; EmitVertex();
    gl_Position = gl_in[0].gl_Position; EmitVertex();
    EndPrimitive();
}
```

Geometry shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    outBuffer0 = vec4(1,0,0,1);
}
```

Fragment shader

Modern graphics pipeline

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
}
```

Vertex shader

```
#version 420

layout(triangles) in;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(line_strip, max_vertices = 4) out;

void main() {
    gl_Position = gl_in[0].gl_Position; EmitVertex();
    gl_Position = gl_in[1].gl_Position; EmitVertex();
    gl_Position = gl_in[2].gl_Position; EmitVertex();
    gl_Position = gl_in[0].gl_Position; EmitVertex();
    EndPrimitive();
}
```

Geometry shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    outBuffer0 = vec4(1,0,0,1);
}
```

Fragment shader

Input data type

Modern graphics pipeline

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
}
```

Vertex shader

```
#version 420

layout(triangles) in;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(line_strip, max_vertices = 4) out;

void main() {
    gl_Position = gl_in[0].gl_Position; EmitVertex();
    gl_Position = gl_in[1].gl_Position; EmitVertex();
    gl_Position = gl_in[2].gl_Position; EmitVertex();
    gl_Position = gl_in[0].gl_Position; EmitVertex();
    EndPrimitive();
}
```

Geometry shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    outBuffer0 = vec4(1,0,0,1);
}
```

Fragment shader

Output data type

Modern graphics pipeline

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
}
```

Vertex shader

```
#version 420

layout(triangles) in;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(line_strip, max_vertices = 4) out;

void main() {
    gl_Position = gl_in[0].gl_Position; EmitVertex();
    gl_Position = gl_in[1].gl_Position; EmitVertex();
    gl_Position = gl_in[2].gl_Position; EmitVertex();
    gl_Position = gl_in[0].gl_Position; EmitVertex();
    EndPrimitive();
}
```

Geometry shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    outBuffer0 = vec4(1,0,0,1);
}
```

Fragment shader

Create new vertices

Modern graphics pipeline

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
}
```

Vertex shader

```
#version 420

layout(triangles) in;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(line_strip, max_vertices = 4) out;

void main() {
    gl_Position = gl_in[0].gl_Position; EmitVertex();
    gl_Position = gl_in[1].gl_Position; EmitVertex();
    gl_Position = gl_in[2].gl_Position; EmitVertex();
    gl_Position = gl_in[0].gl_Position; EmitVertex();
    EndPrimitive();
}
```

Geometry shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    outBuffer0 = vec4(1,0,0,1);
}
```

Fragment shader

End of this primitive

Modern graphics pipeline

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
}
```

Vertex shader

```
#version 420

layout(triangles) in;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(line_strip, max_vertices = 4) out;

void main() {
    gl_Position = gl_in[0].gl_Position; EmitVertex();
    gl_Position = gl_in[1].gl_Position; EmitVertex();
    gl_Position = gl_in[2].gl_Position; EmitVertex();
    gl_Position = gl_in[0].gl_Position; EmitVertex();
    EndPrimitive();
}
```

Geometry shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    outBuffer0 = vec4(1,0,0,1);
}
```

Fragment shader

Resulting rendering?

Modern graphics pipeline

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
}
```

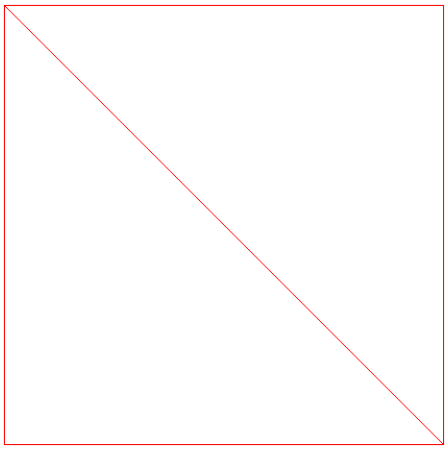
Vertex shader

```
#version 420

layout(triangles) in;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(line_strip, max_vertices = 4) out;

void main() {
    gl_Position = gl_in[0].gl_Position; EmitVertex();
    gl_Position = gl_in[1].gl_Position; EmitVertex();
    gl_Position = gl_in[2].gl_Position; EmitVertex();
    gl_Position = gl_in[0].gl_Position; EmitVertex();
    EndPrimitive();
}
```

Geometry shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    outBuffer0 = vec4(1,0,0,1);
}
```

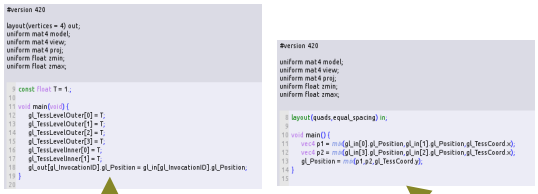
Fragment shader



Modern graphics pipeline

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
}
```

Vertex shader



Tessellation control shader

Tessellation eval shader

```
#version 420

layout(triangles) in;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(line_strip, max_vertices = 4) out;

void main() {
    gl_Position = gl_in[0].gl_Position; EmitVertex();
    gl_Position = gl_in[1].gl_Position; EmitVertex();
    gl_Position = gl_in[2].gl_Position; EmitVertex();
    gl_Position = gl_in[0].gl_Position; EmitVertex();
    EndPrimitive();
}
```

Geometry shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

void main() {
    outBuffer0 = vec4(1,0,0,1);
}
```

Fragment shader

```
#version 420

layout(vertices = 4) out;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

const float T = 1.;

void main(void) {
    gl_TessLevelOuter[0] = T;
    gl_TessLevelOuter[1] = T;
    gl_TessLevelOuter[2] = T;
    gl_TessLevelOuter[3] = T;
    gl_TessLevelInner[0] = T;
    gl_TessLevelInner[1] = T;
    gl_out[gl_InvocationID].gl_Position = gl_in[gl_InvocationID].gl_Position;
}
```

```
#version 420

uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(quads,equal_spacing) in;

void main() {
    vec4 p1 = mix(gl_in[0].gl_Position,gl_in[1].gl_Position,gl_TessCoord.x);
    vec4 p2 = mix(gl_in[3].gl_Position,gl_in[2].gl_Position,gl_TessCoord.x);
    gl_Position = mix(p1,p2,gl_TessCoord.y);
}
```

Modern graphics pipeline

Patch size

Tessellation control shader

```
#version 420

layout(vertices = 4) out;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

const float T = 1.;

void main(void) {
    gl_TessLevelOuter[0] = T;
    gl_TessLevelOuter[1] = T;
    gl_TessLevelOuter[2] = T;
    gl_TessLevelOuter[3] = T;
    gl_TessLevelInner[0] = T;
    gl_TessLevelInner[1] = T;
    gl_out[gl_InvocationID].gl_Position = gl_in[gl_InvocationID].gl_Position;
}

```

Tessellation eval shader

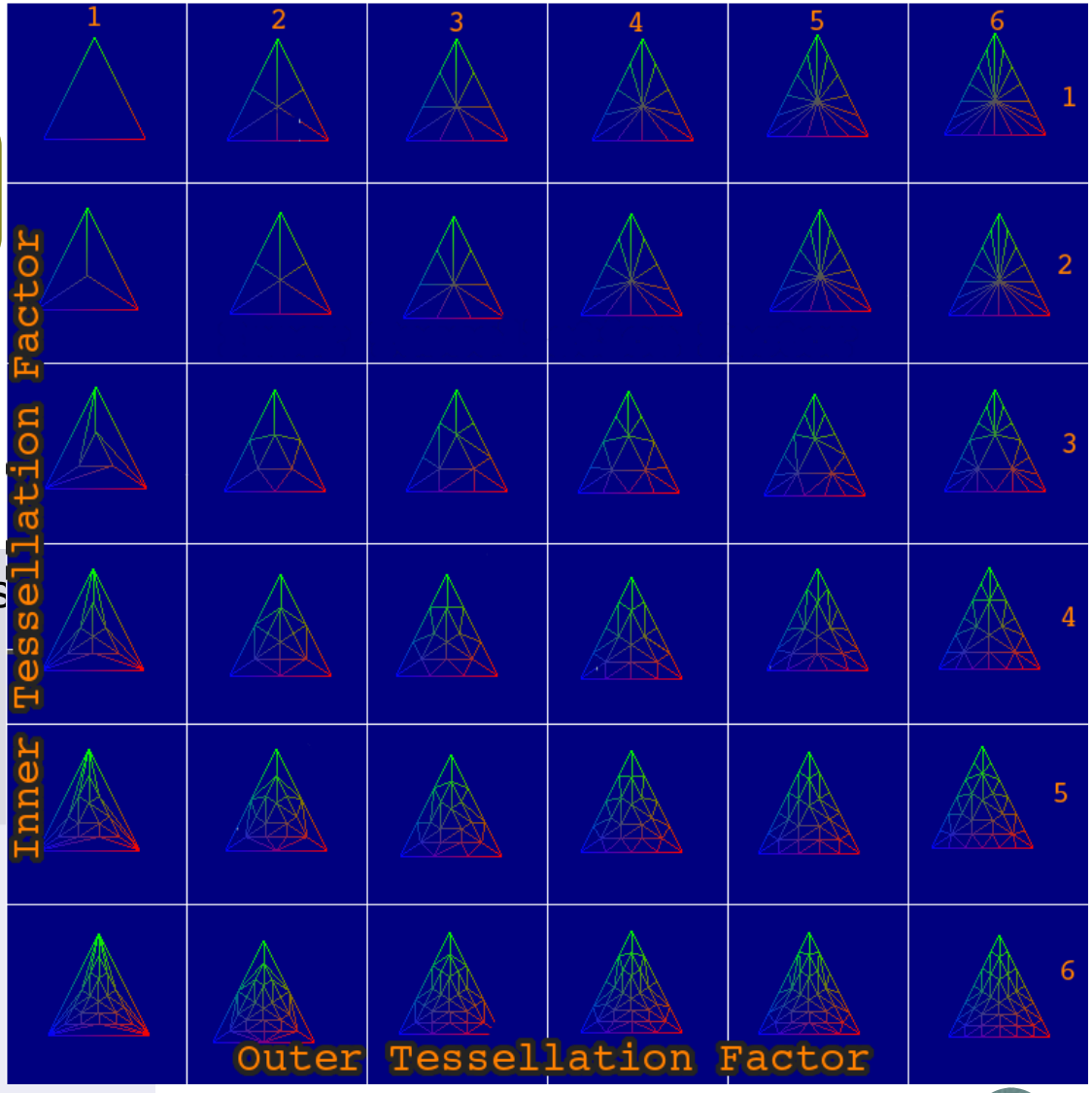
```
#version 420

uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(quads,equal_spacing) in;

void main() {
    vec4 p1 = mix(gl_in[0].gl_Position,gl_in[1].gl_Position,gl_TessCoord.x);
    vec4 p2 = mix(gl_in[3].gl_Position,gl_in[2].gl_Position,gl_TessCoord.x);
    gl_Position = mix(p1,p2,gl_TessCoord.y);
}

```

Modern graphics pipeline

Tessellation level

Tessellation control s

```
#version 420

layout(vertices = 4) out;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

const float T = 1.;

void main(void) {
    gl_TessLevelOuter[0] = T;
    gl_TessLevelOuter[1] = T;
    gl_TessLevelOuter[2] = T;
    gl_TessLevelOuter[3] = T;
    gl_TessLevelInner[0] = T;
    gl_TessLevelInner[1] = T;
    gl_out[gl_InvocationID].gl_Position = gl_in[gl_InvocationID].gl_Position;
}
```



Modern graphics pipeline

Invocation index within the patch

Tessellation control shader

```
#version 420

layout(vertices = 4) out;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

 9 const float T = 1.;
10
11 void main(void) {
12     gl_TessLevelOuter[0] = T;
13     gl_TessLevelOuter[1] = T;
14     gl_TessLevelOuter[2] = T;
15     gl_TessLevelOuter[3] = T;
16     gl_TessLevelInner[0] = T;
17     gl_TessLevelInner[1] = T;
18     gl_out[gl_InvocationID].gl_Position = gl_in[gl_InvocationID].gl_Position;
19 }
20
```

Tessellation eval shader

```
#version 420

uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

 8 layout(quads,equal_spacing) in;
 9
10 void main() {
11     vec4 p1 = mix(gl_in[0].gl_Position,gl_in[1].gl_Position,gl_TessCoord.x);
12     vec4 p2 = mix(gl_in[3].gl_Position,gl_in[2].gl_Position,gl_TessCoord.x);
13     gl_Position = mix(p1,p2,gl_TessCoord.y);
14 }
15
```

Modern graphics pipeline

Input type

Tessellation control shader

```
#version 420

layout(vertices = 4) out;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

const float T = 1.;

void main(void) {
    gl_TessLevelOuter[0] = T;
    gl_TessLevelOuter[1] = T;
    gl_TessLevelOuter[2] = T;
    gl_TessLevelOuter[3] = T;
    gl_TessLevelInner[0] = T;
    gl_TessLevelInner[1] = T;
    gl_out[gl_InvocationID].gl_Position = gl_in[gl_InvocationID].gl_Position;
}
```

Tessellation eval shader

```
#version 420

uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(quads,equal_spacing) in;

void main() {
    vec4 p1 = mix(gl_in[0].gl_Position,gl_in[1].gl_Position,gl_TessCoord.x);
    vec4 p2 = mix(gl_in[3].gl_Position,gl_in[2].gl_Position,gl_TessCoord.x);
    gl_Position = mix(p1,p2,gl_TessCoord.y);
}
```

Modern graphics pipeline

Location within the patch

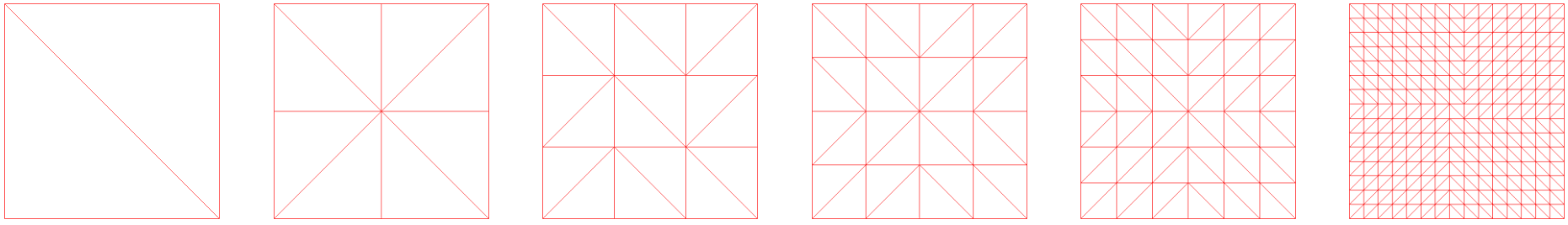
Tessellation control shader

```
#version 420

layout(vertices = 4) out;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

const float T = 1.;

void main(void) {
    gl_TessLevelOuter[0] = T;
    gl_TessLevelOuter[1] = T;
    gl_TessLevelOuter[2] = T;
    gl_TessLevelOuter[3] = T;
    gl_TessLevelInner[0] = T;
    gl_TessLevelInner[1] = T;
    gl_out[gl_InvocationID].gl_Position = gl_in[gl_InvocationID].gl_Position;
}

```

Tessellation eval shader

```
#version 420

uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(quads,equal_spacing) in;

void main() {
    vec4 p1 = mix(gl_in[0].gl_Position,gl_in[1].gl_Position,gl_TessCoord.x);
    vec4 p2 = mix(gl_in[3].gl_Position,gl_in[2].gl_Position,gl_TessCoord.x);
    gl_Position = mix(p1,p2,gl_TessCoord.y);
}

```

Modern graphics pipeline



Tessellation control shader

```
#version 420

layout(vertices = 4) out;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

const float T = 1.;

void main(void) {
    gl_TessLevelOuter[0] = T;
    gl_TessLevelOuter[1] = T;
    gl_TessLevelOuter[2] = T;
    gl_TessLevelOuter[3] = T;
    gl_TessLevelInner[0] = T;
    gl_TessLevelInner[1] = T;
    gl_out[gl_InvocationID].gl_Position = gl_in[gl_InvocationID].gl_Position;
}

```

Tessellation eval shader

```
#version 420

uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(quads,equal_spacing) in;

void main() {
    vec4 p1 = mix(gl_in[0].gl_Position,gl_in[1].gl_Position,gl_TessCoord.x);
    vec4 p2 = mix(gl_in[3].gl_Position,gl_in[2].gl_Position,gl_TessCoord.x);
    gl_Position = mix(p1,p2,gl_TessCoord.y);
}

```

Modern graphics pipeline

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(location=0) out vec3 outVar;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
    outVar = inVertex+0.5;
}
```

Vertex shader

```
#version 420

layout(vertices = 4) out;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(location=0) in vec3 inVar[];
layout(location=0) out vec3 outVar[3];

const float T = 1.;

void main(void) {
    gl_TessLevelOuter[0] = T;
    gl_TessLevelOuter[1] = T;
    gl_TessLevelOuter[2] = T;
    gl_TessLevelOuter[3] = T;
    gl_TessLevelInner[0] = T;
    gl_TessLevelInner[1] = T;
    gl_out[gl_InvocationID].gl_Position = gl_in[gl_InvocationID].gl_Position;
    outVar[gl_InvocationID] = inVar[gl_InvocationID];
}
```

Tessellation control shader

```
#version 420

uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(quads,equal_spacing) in;

layout(location=0) in vec3 inVar[];
layout(location=0) out vec3 outVar;


void main() {
    vec4 p1 = mix(gl_in[0].gl_Position,gl_in[1].gl_Position,gl_TessCoord.x);
    vec4 p2 = mix(gl_in[3].gl_Position,gl_in[2].gl_Position,gl_TessCoord.x);
    gl_Position = mix(p1,p2,gl_TessCoord.y);

    vec3 v1 = mix(inVar[0],inVar[1],gl_TessCoord.x);
    vec3 v2 = mix(inVar[3],inVar[2],gl_TessCoord.x);
    outVar = mix(v1,v2,gl_TessCoord.y);
}
```

Tessellation eval shader

```
#version 420

layout(triangles) in;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(line_strip, max_vertices = 4) out;

layout(location=0) in vec3 inVar[3];
layout(location=0) out vec3 outVar;

void main() {
    gl_Position = gl_in[0].gl_Position; outVar = inVar[0]; EmitVertex();
    gl_Position = gl_in[1].gl_Position; outVar = inVar[1]; EmitVertex();
    gl_Position = gl_in[2].gl_Position; outVar = inVar[2]; EmitVertex();
    gl_Position = gl_in[0].gl_Position; outVar = inVar[0]; EmitVertex();
    EndPrimitive();
}
```

Geometry shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(location=0) in vec3 inVar;

void main() {
    outBuffer0 = vec4(inVar,1);
}
```

Fragment shader

Modern graphics pipeline

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(location=0) out vec3 outVar;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
    outVar = inVertex+0.5;
}
```

Vertex shader

```
#version 420

layout(vertices = 4) out;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(location=0) in vec3 inVar[];
layout(location=0) out vec3 outVar[3];

const float T = 1.;

void main(void) {
    gl_TessLevelOuter[0] = T;
    gl_TessLevelOuter[1] = T;
    gl_TessLevelOuter[2] = T;
    gl_TessLevelOuter[3] = T;
    gl_TessLevelInner[0] = T;
    gl_TessLevelInner[1] = T;
    gl_out[gl_InvocationID].gl_Position = gl_in[gl_InvocationID].gl_Position;
    outVar[gl_InvocationID] = inVar[gl_InvocationID];
}
```

Tessellation control shader

```
#version 420

uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(quads,equal_spacing) in;

layout(location=0) in vec3 inVar[];
layout(location=0) out vec3 outVar;


void main() {
    vec4 p1 = mix(gl_in[0].gl_Position,gl_in[1].gl_Position,gl_TessCoord.x);
    vec4 p2 = mix(gl_in[3].gl_Position,gl_in[2].gl_Position,gl_TessCoord.x);
    gl_Position = mix(p1,p2,gl_TessCoord.y);

    vec3 v1 = mix(inVar[0],inVar[1],gl_TessCoord.x);
    vec3 v2 = mix(inVar[3],inVar[2],gl_TessCoord.x);
    outVar = mix(v1,v2,gl_TessCoord.y);
}
```

Tessellation eval shader

```
#version 420

layout(triangles) in;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(line_strip, max_vertices = 4) out;

layout(location=0) in vec3 inVar[3];
layout(location=0) out vec3 outVar;

void main() {
    gl_Position = gl_in[0].gl_Position; outVar = inVar[0]; EmitVertex();
    gl_Position = gl_in[1].gl_Position; outVar = inVar[1]; EmitVertex();
    gl_Position = gl_in[2].gl_Position; outVar = inVar[2]; EmitVertex();
    gl_Position = gl_in[0].gl_Position; outVar = inVar[0]; EmitVertex();
    EndPrimitive();
}
```

Geometry shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(location=0) in vec3 inVar;

void main() {
    outBuffer0 = vec4(inVar,1);
}
```

Fragment shader

Modern graphics pipeline

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(location=0) out vec3 outVar;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
    outVar = inVertex+0.5;
}
```

Vertex shader

```
#version 420

layout(vertices = 4) out;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(location=0) in vec3 inVar[];
layout(location=0) out vec3 outVar[3];

const float T = 1.;

void main(void) {
    gl_TessLevelOuter[0] = T;
    gl_TessLevelOuter[1] = T;
    gl_TessLevelOuter[2] = T;
    gl_TessLevelOuter[3] = T;
    gl_TessLevelInner[0] = T;
    gl_TessLevelInner[1] = T;
    gl_out[gl_InvocationID].gl_Position = gl_in[gl_InvocationID].gl_Position;
    outVar[gl_InvocationID] = inVar[gl_InvocationID];
}
```

Tessellation control shader

```
#version 420

uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(quads,equal_spacing) in;

layout(location=0) in vec3 inVar[];
layout(location=0) out vec3 outVar;


void main() {
    vec4 p1 = mix(gl_in[0].gl_Position,gl_in[1].gl_Position,gl_TessCoord.x);
    vec4 p2 = mix(gl_in[3].gl_Position,gl_in[2].gl_Position,gl_TessCoord.x);
    gl_Position = mix(p1,p2,gl_TessCoord.y);

    vec3 v1 = mix(inVar[0],inVar[1],gl_TessCoord.x);
    vec3 v2 = mix(inVar[3],inVar[2],gl_TessCoord.x);
    outVar = mix(v1,v2,gl_TessCoord.y);
}
```

Tessellation eval shader

```
#version 420

layout(triangles) in;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(line_strip, max_vertices = 4) out;

layout(location=0) in vec3 inVar[3];
layout(location=0) out vec3 outVar;

void main() {
    gl_Position = gl_in[0].gl_Position; outVar = inVar[0]; EmitVertex();
    gl_Position = gl_in[1].gl_Position; outVar = inVar[1]; EmitVertex();
    gl_Position = gl_in[2].gl_Position; outVar = inVar[2]; EmitVertex();
    gl_Position = gl_in[0].gl_Position; outVar = inVar[0]; EmitVertex();
    EndPrimitive();
}
```

Geometry shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(location=0) in vec3 inVar;

void main() {
    outBuffer0 = vec4(inVar,1);
}
```

Fragment shader

Resulting rendering?

Modern graphics pipeline

```
#version 420

layout(location = 0) in vec3 inVertex;
layout(location = 1) in vec2 inTexcoord;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(location=0) out vec3 outVar;

void main() {
    mat4 mdv   = view*model;
    mat4 mvp   = proj*mdv;

    gl_Position = mvp*vec4(inVertex,1);
    outVar = inVertex+0.5;
}
```

Vertex shader

```
#version 420

layout(vertices = 4) out;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(location=0) in vec3 inVar[];
layout(location=0) out vec3 outVar[3];

const float T = 1.;

void main(void) {
    gl_TessLevelOuter[0] = T;
    gl_TessLevelOuter[1] = T;
    gl_TessLevelOuter[2] = T;
    gl_TessLevelOuter[3] = T;
    gl_TessLevelInner[0] = T;
    gl_TessLevelInner[1] = T;
    gl_out[gl_InvocationID].gl_Position = gl_in[gl_InvocationID].gl_Position;
    outVar[gl_InvocationID] = inVar[gl_InvocationID];
}
```

Tessellation control shader

```
#version 420

uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(quads,equal_spacing) in;

layout(location=0) in vec3 inVar[];
layout(location=0) out vec3 outVar;


void main() {
    vec4 p1 = mix(gl_in[0].gl_Position,gl_in[1].gl_Position,gl_TessCoord.x);
    vec4 p2 = mix(gl_in[3].gl_Position,gl_in[2].gl_Position,gl_TessCoord.x);
    gl_Position = mix(p1,p2,gl_TessCoord.y);

    vec3 v1 = mix(inVar[0],inVar[1],gl_TessCoord.x);
    vec3 v2 = mix(inVar[3],inVar[2],gl_TessCoord.x);
    outVar = mix(v1,v2,gl_TessCoord.y);
}
```

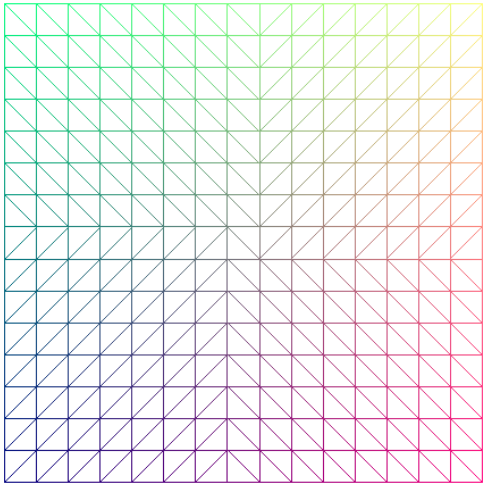
Tessellation eval shader

```
#version 420

layout(triangles) in;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(line_strip, max_vertices = 4) out;

layout(location=0) in vec3 inVar[3];
layout(location=0) out vec3 outVar;

void main() {
    gl_Position = gl_in[0].gl_Position; outVar = inVar[0]; EmitVertex();
    gl_Position = gl_in[1].gl_Position; outVar = inVar[1]; EmitVertex();
    gl_Position = gl_in[2].gl_Position; outVar = inVar[2]; EmitVertex();
    gl_Position = gl_in[0].gl_Position; outVar = inVar[0]; EmitVertex();
    EndPrimitive();
}
```

Geometry shader

```
#version 420

layout(location = 0) out vec4 outBuffer0;
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;
uniform float zmin;
uniform float zmax;

layout(location=0) in vec3 inVar;

void main() {
    outBuffer0 = vec4(inVar,1);
}
```

Fragment shader



Rasterization advantages

- Modern scenes more complicated than images
  - 1920x1080 frame (1080p)
  - 64-bit color and 32-bit depth
  - 24 Mb memory
- Rasterization can stream over triangles
  - One triangle at a time
  - Parrallelism
  - Memory optimization

Rasterization limitations

- Restricted to scan-convertible primitives (triangles)
- No unified handling of
  - Shadows
  - Reflection
  - Transparency
- Potential problem of overdraw
  - Depth complexity
  - Each pixel touched many times

References

- MIT:
  - http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-837-computer-graphics-fall-2012/lecture-notes/
- Standford:
  - http://candela.stanford.edu/cs348b-14/doku.php
- Siggraph:
  - http://blog.selfshadow.com/publications/s2014-shading-course/
  - http://blog.selfshadow.com/publications/s2013-shading-course/
- Image synthesis & OpenGL:
  - http://romain.vergne.free.fr/blog/?page_id=97
- Path tracing and global illum:
  - http://www.graphics.stanford.edu/courses/cs348b-01/course29.hanrahan.pdf
  - http://web.cs.wpi.edu/~emmanuel/courses/cs563/write_ups/zackw/realistic_raytracing.html
- GLSL / Shadertoy:
  - https://www.opengl.org/documentation/glsl/
  - https://www.shadertoy.com/
  - http://www.iquilezles.org/