MODERN OPENGL

APRIL 2012

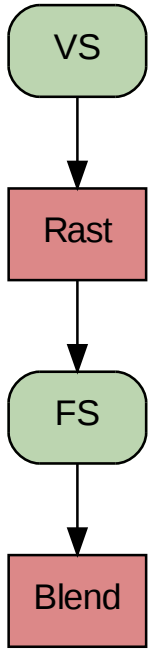
1 Evolution of the OpenGL Pipe

2 Whirlwind Tour of OpenGL 4.2

3 Demos & Code Walkthroughs

```glsl
// Vertex Shader
varying vec4 vPosition;
varying vec2 vTexCoord;
varying vec3 vNormal;
void main() {
  vPosition = gl_ModelViewMatrix * gl_Vertex;
  vNormal = gl_NormalMatrix * gl_Normal;
  vTexCoord = gl_MultiTexCoord0.st;
  gl_Position = ftransform();
}
```

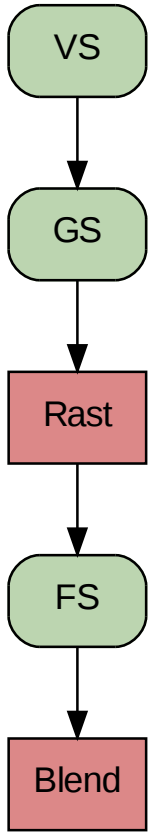
```glsl
// Fragment Shader
uniform sampler2D Baboon;
uniform vec3 LightDir;
varying vec2 vTexCoord;
varying vec3 vNormal;
void main() {
  float df = dot(LightDir,vNormal);
  vec3 c = df * texture2D(Baboon, vTexCoord).rgb;
  gl_FragColor = vec4(c, 1);
}
```



```glsl
// Vertex Shader
in vec4 Position;
in vec2 TexCoord;
out vec4 vPosition;
out vec2 vTexCoord;
uniform mat4 ModelViewMatrix, MVP;
void main() {
    vPosition = ModelViewMatrix * Position;
    vTexCoord = TexCoord;
    gl_Position = MVP * Position;
}
```

```glsl
// Fragment Shader
uniform sampler2D Baboon;
uniform vec3 LightDir = vec3(0,0,1);
in vec2 gTexCoord;
in vec3 gNormal;
void main() {
    float df = dot(LightDir,gNormal);
    vec3 c = df*texture2D(Baboon, gTexCoord).rgb;
    gl_FragColor = vec4(c, 1);
}
```

```glsl
// Geometry Shader
layout(triangles) in;
layout(triangle_strip, max_vertices = 3) out;
in vec3 vPosition[3];
in vec2 vTexCoord[3];
out vec3 gNormal;
out vec2 gTexCoord;

void main()
{
    vec3 A = vPosition[2] - vPosition[0];
    vec3 B = vPosition[1] - vPosition[0];
    gNormal = normalize(cross(A, B));

    for (int j = 0; j < 3; j++) {
        gTexCoord = vTexCoord[j];
        gl_Position = gl_in[j].gl_Position;
        EmitVertex();
    }
    EndPrimitive();
}
```



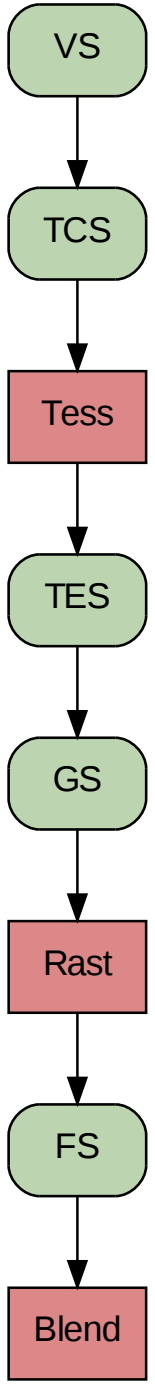
```glsl
// Vertex Shader
in vec4 Position;
in vec2 TexCoord;
out vec4 vPosition;
out vec2 vTexCoord;
uniform mat4 ModelViewMatrix, MVP;
void main() {
    vPosition = ModelViewMatrix * Position;
    vTexCoord = TexCoord;
    gl_Position = MVP * Position;
}
```

```glsl
// Tessellation Control Shader
layout(triangles) in;
layout(triangle_strip, max_vertices = 3) out;
in vec3 vPosition[3];
in vec2 vTexCoord[3];
out vec3 gNormal;
out vec2 gTexCoord;

void main()
{
    for (int j = 0; j < 3; j++) {
        gTexCoord = vTexCoord[j];
        gl_Position = gl_in[j].gl_Position;
        EmitVertex();
    }
    EndPrimitive();
}
```

```glsl
// Tessellation Evaluation Shader
uniform sampler2D Baboon;
in vec2 gTexCoord;
in vec3 gNormal;

void main() {
    float df = dot(vec3(0,0,1),gNormal);
    vec3 c = df*texture2D(Baboon, gTexCoord).rgb;
    gl_FragColor = vec4(c, 1);
}
```

```glsl
// Geometry Shader
layout(triangles) in;
layout(triangle_strip, max_vertices = 3) out;
in vec3 vPosition[3];
in vec2 vTexCoord[3];
out vec3 gNormal;
out vec2 gTexCoord;

void main()
{
    vec3 A = vPosition[2] - vPosition[0];
    vec3 B = vPosition[1] - vPosition[0];
    gNormal = normalize(cross(A, B));

    for (int j = 0; j < 3; j++) {
        gTexCoord = vTexCoord[j];
        gl_Position = gl_in[j].gl_Position;
        EmitVertex();
    }
    EndPrimitive();
}
```

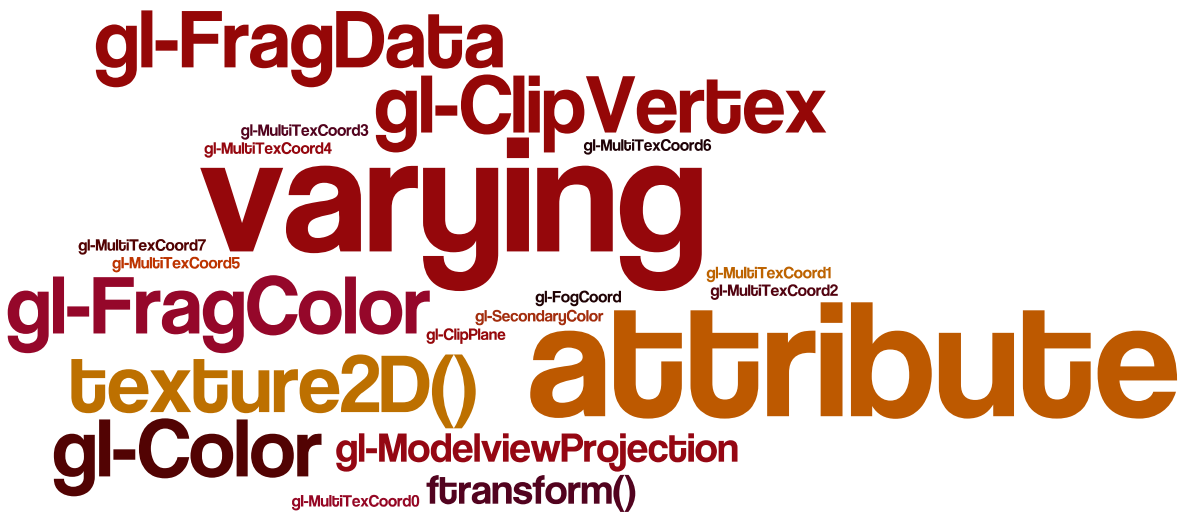
```glsl
// Fragment Shader
# LOOK AT BARREL DISTORTATION.
# USE CHRISTOPHE'S INTERFACES.
# EXPLICIT BINDINGS.
uniform sampler2D Baboon;
uniform vec3 LightDir = vec3(0,0,1);
in vec2 gTexCoord;
in vec3 gNormal;
void main() {
    float df = dot(LightDir,gNormal);
    vec3 c = df*texture(Baboon, gTexCoord).rgb;
    gl_FragColor = vec4(c, 1);
}
```



```
QGLFormat format;
format.setVersion(4,0);
format.setProfile(QGLFormat::CoreProfile);
QGLWidget *myWidget = new QGLWidget(format);
```

```
int attribs[] = {
    GLX_CONTEXT_MAJOR_VERSION_ARB, 4,
    GLX_CONTEXT_MINOR_VERSION_ARB, 0,
    GLX_CONTEXT_FLAGS_ARB, GLX_CONTEXT_FORWARD_COMPATIBLE_BIT_ARB,
    NULL
};
GLXContext glc = glXCreateContextAttribs(diplay, config, NULL, True, attribs);
```

Core Profile


gl-FragData
gl-ClipVertex
gl-MultiTexCoord3
gl-MultiTexCoord4
gl-MultiTexCoord6
varying
gl-MultiTexCoord7
gl-MultiTexCoord5
gl-MultiTexCoord1
gl-MultiTexCoord2
gl-FragColor
gl-FogCoord
gl-SecondaryColor
gl-ClipPlane
attribute
texture2D()
gl-Color
gl-ModelviewProjection
gl-MultiTexCoord0
ftransform()

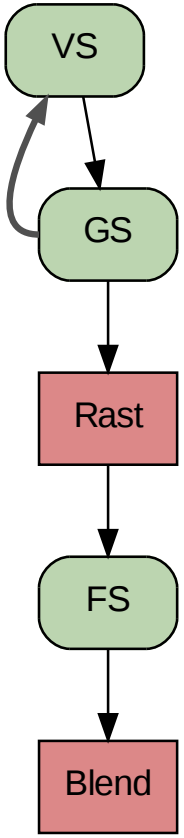
OLD


texelFetch()
texture()
gl-ViewportIndex
out
gl-TessLevelInner
textureQueryLod()
in
imageStore()
gl-ClipDistance[]
imageAtomicAdd()
layout
gl-Layer
imageLoad()
gl-TessLevelOuter
gl-InvocationID
bitfieldExtract()
gl-PrimitiveIDIn
textureSize()

NEW

```
// This goes after glCompileShader but before glLinkProgram...
const char* varyings[3] = { "vPosition", "vBirthTime", "vVelocity" };
glTransformFeedbackVaryings(programHandle, 3, varyings,
                            GL_INTERLEAVED_ATTRIBS);

// Create a query object for transform feedback:
glGenQueries(1, &PrimsWritten);

// Create VBO for input on even frames and output on odd frames:
glGenBuffers(1, &BufferA);
glBindBuffer(GL_ARRAY_BUFFER, BufferA);
glBufferData(GL_ARRAY_BUFFER, sizeof(seed_data), &seed_data[0],
GL_STREAM_DRAW);

// Create VBO for output on even frames and input on odd frames:
glGenBuffers(1, &BufferB);
glBindBuffer(GL_ARRAY_BUFFER, BufferB);
glBufferData(GL_ARRAY_BUFFER, sizeof(seed_data), 0, GL_STREAM_DRAW);
```

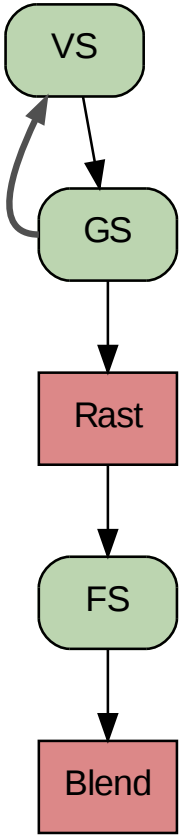
```
glEnable(GL_RASTERIZER_DISCARD);
glBindBuffer(GL_ARRAY_BUFFER, BufferA);                    // Source VBO
glVertexAttribPointer(...);
glBindBufferBase(GL_TRANSFORM_FEEDBACK_BUFFER, 0, BufferB); // Dest VBO
glBeginTransformFeedback(GL_POINTS);
glBeginQuery(GL_TRANSFORM_FEEDBACK_PRIMITIVES_WRITTEN, PrimsWritten);
glDrawArrays(GL_POINTS, 0, inCount);
glEndTransformFeedback();
glEndQuery(GL_TRANSFORM_FEEDBACK_PRIMITIVES_WRITTEN);
glGetQueryObjectuiv(Query, GL_QUERY_RESULT, &outCount);

swap(BufferA, BufferB);

glDisable(GL_RASTERIZER_DISCARD);
glBindBuffer(GL_ARRAY_BUFFER, BufferA);
glVertexAttribPointer(...);
glDrawArrays(GL_POINTS, 0, outCount);
```



Old Transform Feedback (Interleaved)

```
// This goes after glCompileShader but before glLinkProgram...
const char* varyings[2] = { "vPosition", "vBirthTime" };
glTransformFeedbackVaryings(programHandle, 2, varyings,
                            GL_SEPARATE_ATTRIBS);

// Create a query object for transform feedback:
glGenQueries(1, &PrimsWritten);

// Create VBOs for input on even frames and output on odd frames:
glGenBuffers(1, &Buffer0A);
glBindBuffer(GL_ARRAY_BUFFER, Buffer0A);
glGenBuffers(1, &Buffer1A);
glBindBuffer(GL_ARRAY_BUFFER, Buffer1A);

// Create VBOs for output on even frames and input on odd frames:
glGenBuffers(1, &Buffer0B);
glBindBuffer(GL_ARRAY_BUFFER, Buffer0B);
glGenBuffers(1, &Buffer1B);
glBindBuffer(GL_ARRAY_BUFFER, Buffer1B);
```

```
glEnable(GL_RASTERIZER_DISCARD);

glBindBuffer(GL_ARRAY_BUFFER, Buffer0A);                    // Source VBO
glVertexAttribPointer(...);
glBindBuffer(GL_ARRAY_BUFFER, Buffer1A);                    // Source VBO
glVertexAttribPointer(...);
glBindBufferBase(GL_TRANSFORM_FEEDBACK_BUFFER, 0, Buffer0B); // Dest VBO
glBindBufferBase(GL_TRANSFORM_FEEDBACK_BUFFER, 1, Buffer1B); // Dest VBO
glBeginTransformFeedback(GL_POINTS);
glBeginQuery(GL_TRANSFORM_FEEDBACK_PRIMITIVES_WRITTEN, PrimsWritten);
glDrawArrays(GL_POINTS, 0, inCount);
glEndTransformFeedback();
glEndQuery(GL_TRANSFORM_FEEDBACK_PRIMITIVES_WRITTEN);
glGetQueryObjectuiv(Query, GL_QUERY_RESULT, &outCount);

swap(Buffer0A, Buffer0B);
swap(Buffer1A, Buffer1B);

glDisable(GL_RASTERIZER_DISCARD);
glBindBuffer(GL_ARRAY_BUFFER, Buffer0A);
glVertexAttribPointer(...);
glBindBuffer(GL_ARRAY_BUFFER, Buffer1A);
glVertexAttribPointer(...);
glDrawArrays(GL_POINTS, 0, outCount);
```

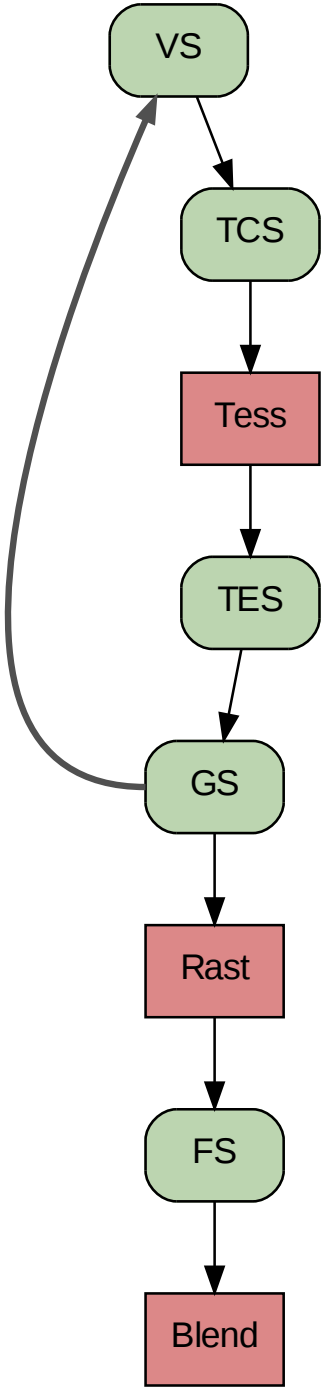


Old Transform Feedback (Separate)

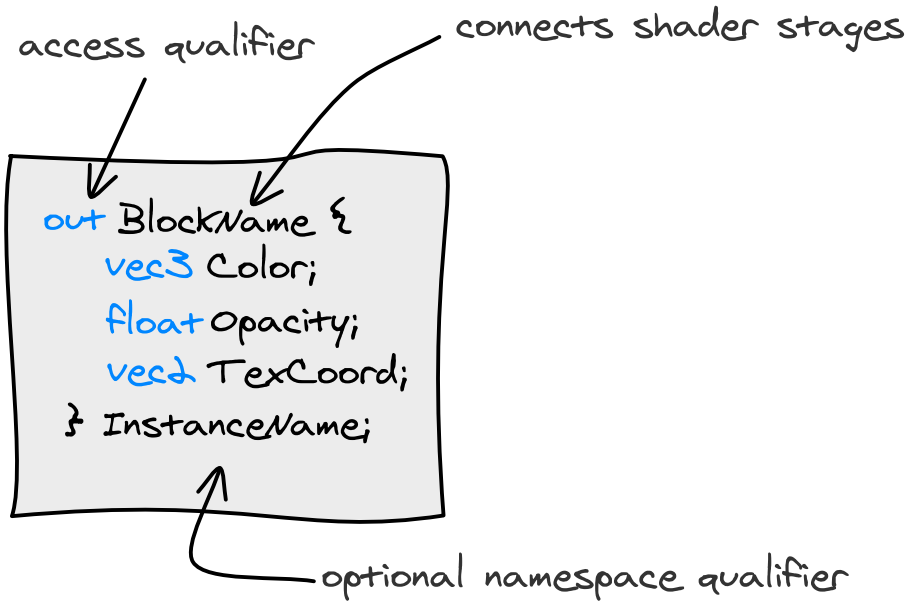
```
// This goes after glCompileShader but before glLinkProgram...
const char* varyings[4] = { "vPosition", "gl_NextBuffer", "vBirthTime", "vVelocity" };
glTransformFeedbackVaryings(programHandle, 4, varyings,
                            GL_INTERLEAVED_ATTRIBS);

// Create a query object for transform feedback:
glGenQueries(1, &PrimsWritten);

// Create VBO for input on even frames and output on odd frames:
glGenBuffers(1, &BufferA);
glBindBuffer(GL_ARRAY_BUFFER, BufferA);
glBufferData(GL_ARRAY_BUFFER, sizeof(seed_data), &seed_data[0], GL_STREAM_DRAW);

// Create VBO for output on even frames and input on odd frames:
glGenBuffers(1, &BufferB);
glBindBuffer(GL_ARRAY_BUFFER, BufferB);
glBufferData(GL_ARRAY_BUFFER, sizeof(seed_data), 0, GL_STREAM_DRAW);
```

```
glEnable(GL_RASTERIZER_DISCARD);
glBindBuffer(GL_ARRAY_BUFFER, BufferA);                    // Source VBO
glBindBufferBase(GL_TRANSFORM_FEEDBACK_BUFFER, 0, BufferB); // Dest VBO
glBeginTransformFeedback(GL_POINTS);
glBeginQuery(GL_TRANSFORM_FEEDBACK_PRIMITIVES_WRITTEN, PrimsWritten);
glDrawArrays(GL_POINTS, 0, inCount);
glEndTransformFeedback();
glEndQuery(GL_TRANSFORM_FEEDBACK_PRIMITIVES_WRITTEN);
glGetQueryObjectuiv(Query, GL_QUERY_RESULT, &outCount);

swap(BufferA, BufferB);

glDisable(GL_RASTERIZER_DISCARD);
glBindBuffer(GL_ARRAY_BUFFER, BufferA);
glDrawArrays(GL_POINTS, 0, outCount);
```

New Transform Feedback



	vert in	varying	frag out	uniform
scalar	✔	✔	✔	✔
vector	✔	✔	✔	✔
matrix	✔	✔	✖	✔
array	✔	✔	✔	✔
structure	✖	✔	✖	✔
samplers	✖	✖	✖	✔
images	✖	✖	✖	✔
atomic counters	✖	✖	✖	✔
block	✖	✔	✖	✔




```
-- Vertex Shader

out MyBlock {
    vec3 Position;
    vec3 Color[2];
    float Opacity;
} Out;

-- Geometry Shader

in MyBlock {
    vec3 Position;
    vec3 Color[2];
    float Opacity;
} In[];
```

```
-- Vertex Shader

// Built-ins:
out gl_PerVertex {
    vec4 gl_Position;
    float gl_PointSize;
    float gl_ClipDistance[];
};

// User-defined:
in MyBlock {
    float w;
} In;

void main()
{
    gl_Position = vec4(1, 0, 0, In.w);
}
```

```
// Application code
GLuint i = glGetAttribLocation(program, "MyBlock.w");
```

```
// Let the compiler decide
GLuint i = glGetAttribLocation(program, "MyBlock.w");
```

```
// Specify in application code
glCompileShader(vsHandle);
glAttachShader(programHandle, vsHandle);
glBindAttribLocation(programHandle, 3, "MyBlock.w");
glLinkProgram(programHandle);
```

```
// Declare in GLSL
in MyBlock {
    layout(location = 3) vec3 w;
}
```

Binding Vertex Attributes