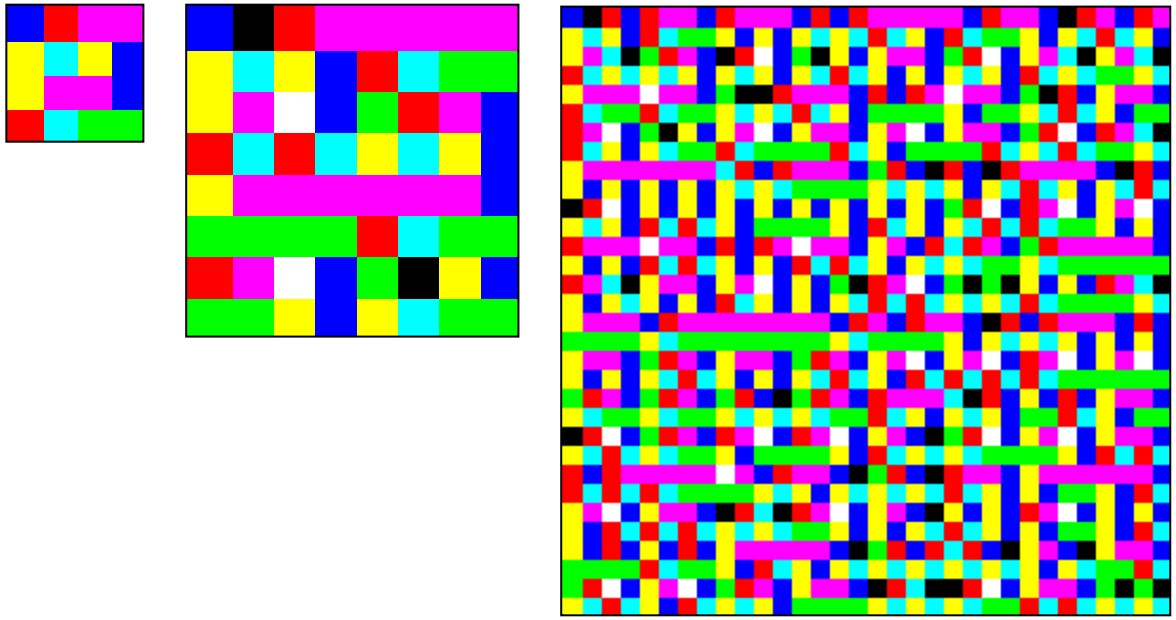
OpenGL II

MipMap

- MipMap

```
glTexImage2D(GL_TEXTURE_2D, level, GL_RGBA32F, ...);
```

```
glFramebufferTexture2D(GL_FRAMEBUFFER, ..., TexID, mipLevel);
```



Geometria generálása

- Üres vertex array

```
glBufferData(GL_ARRAY_BUFFER, dataSize, NULL, GL_STATIC_DRAW);
```

- Leképzés a host memóriába

```
float* vertices = (float*)glMapBuffer(GL_ARRAY_BUFFER, GL_WRITE_ONLY);
```

- Felszabadítás az OpenGL számára

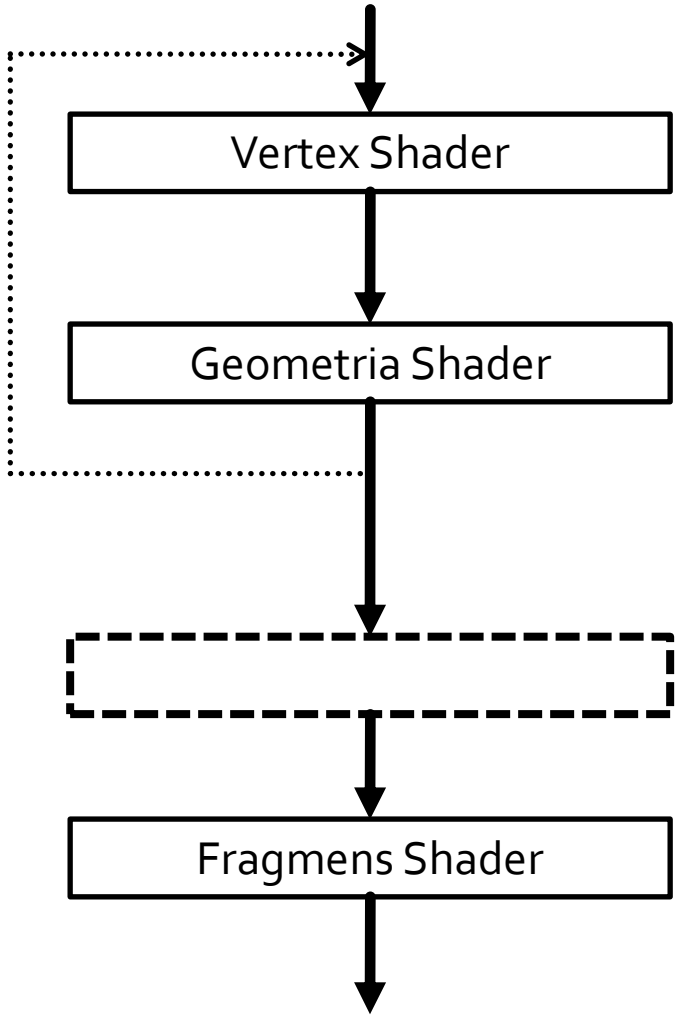
```
glUnmapBuffer(GL_ARRAY_BUFFER);
```

Geometria shader

- Opcionális lépcső
- Primitíveken dolgozik

- Bemenet: egy primitív
- Kimenet: egy vagy több

- A shader kimenete visszaköthető



Geometria shader

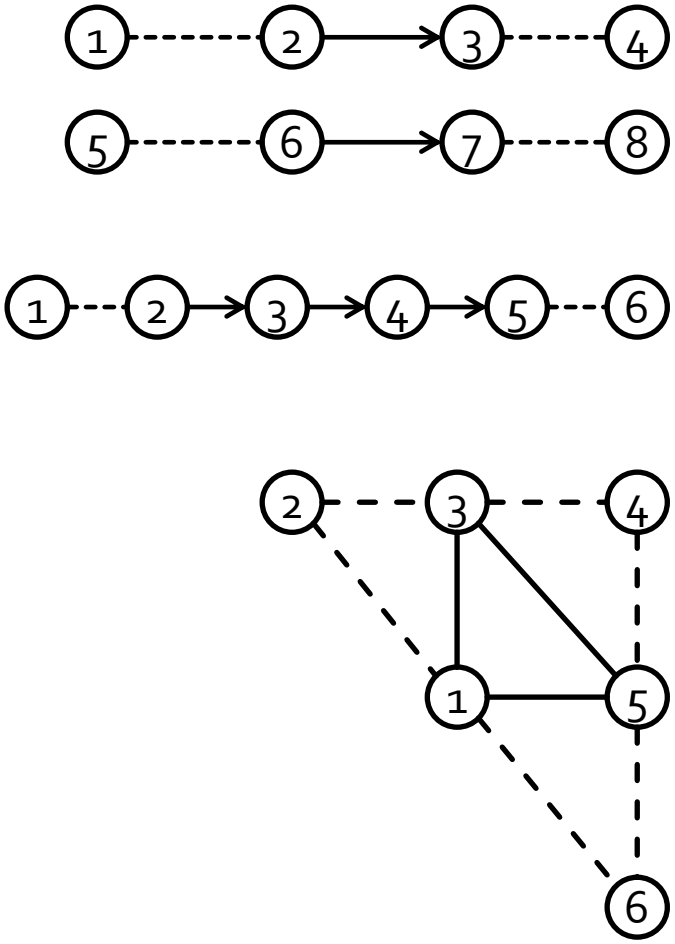
- Bementi primitívek

```
glProgramParameteri(shader, GL_GEOMETRY_INPUT_TYPE, tipus);
```

  - Pont
    - GL_POINTS
  - Szakasz
    - GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP
  - Háromszög
    - GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN
  - Adjacencia információ

Geometria shader

- Adjacencia
  - Szakasz
    - GL_LINES_ADJACENCY
    - GL_LINE_STRIP_ADJACENCY
  - Háromszög
    - GL_TRIANGLES_ADJACENCY
    - GL_TRIANGLE_STRIP_ADJACENCY



Geometria shader

- Kimeneti primitívek

```
glProgramParameteri(shader, GL_GEOMETRY_OUTPUT_TYPE, tipus);
glProgramParameteri(shader, GL_GEOMETRY_VERTICES_OUT, darab);
```

  - Pont
    - GL_POINTS
  - Szakasz
    - GL_LINE_STRIP
  - Háromszög
    - GL_LINE_STRIP

Geometria shader

- Speciális bemeneti változók
  - gl_ClipDistance[] : vágási információk
  - gl_PointSize[] : vertex méret a vertex shaderből
  - gl_Position : vertex pozíció
  - gl_PrimitiveIDIn : a feldolgozott primitív sorszáma
- Speciális kimeneti változók
  - A bemeneti változók
  - gl_Layer : melyik rétegbe tegye a fragmens shader

    (pl. cube map rendereléshez)

Geometria shader

- Primitívek generálása
  - Vertex információk beállítása
  - Vertex lezárása

```
EmitVertex();
```

  - Primitív lezárása

```
EndPrimitive();
```

Geometria shader

- Példa

```
#version 130
#extension GL_EXT_geometry_shader4 : enable

in vec2 vTexCoord[];
out vec2 fTexCoord;

void main(void){
 for(int i=0; i < gl_VerticesIn; ++i){
   gl_Position = gl_PositionIn[i];
   fTexCoord = vTexCoord[i];
   EmitVertex();
 }
 EndPrimitive();

 for(int i=0; i < gl_VerticesIn; ++i){
   gl_Position = gl_PositionIn[i].yxzw;
   fTexCoord = vTexCoord[i].yx;
   EmitVertex();
 }
 EndPrimitive();
}
```

Geometria shader

- Primitívek újrafeldolgozása
  - Transform feedback

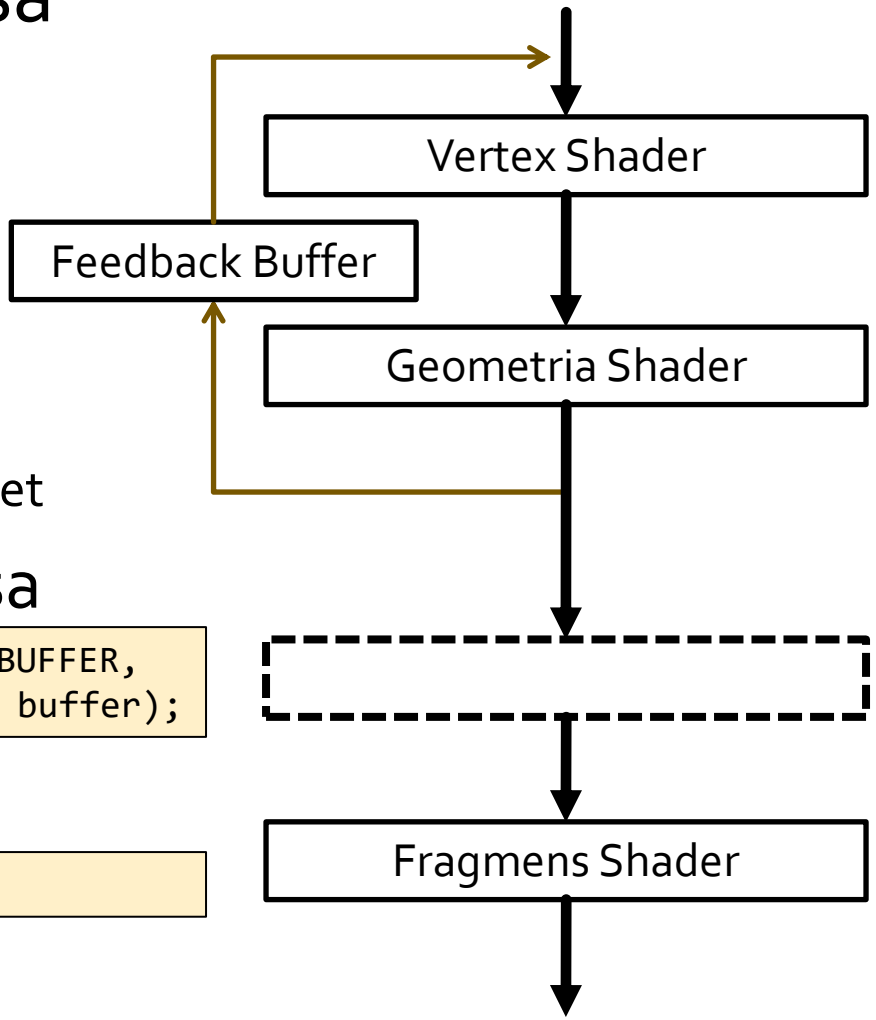
```
glBeginTransformFeedback(mode);
// glDrawArrays(...);
glEndTransformFeedback();
```

    - Feedback mód
      - Megadja a használható primitíveket
  - Feedback buffer kiválasztása

```
glBindBufferBase(GL_TRANSFORM_FEEDBACK_BUFFER,
                                    index, buffer);
```

  - Tulajdonságok kiválasztása

```
glTransformFeedbackVaryings(...);
```



Geometria shader

- Információ a geometria shader működéséről
  - Primitive query

```
GLuint outputQuery;
glGenQueries(1, &outputQuery);
glBeginQuery(mode, outputQuery);

...

glEndQuery(mode);
```

  - Query mód
    - GL_PRIMITIVES_GENERATED
      - Mennyi primitívet állított elő a geometria shader
    - GL_TRANSFORM_FEEDBACK_PRIMITIVES_WRITTEN
      - Mennyi primitívet tudott a feedback bufferbe írni a shader

Geometria shader

- Információ a geometria shader működéséről
  - A Query eredményének lekérdezése

```
GLuint outPointCount = 0;
GLuint succeded = 0;

while(!succeded){
  glGetQueryObjectiv(outputQuery, GL_QUERY_RESULT_AVAILABLE, &succeded);
}

glGetQueryObjectiv(outputQuery, GL_QUERY_RESULT, &outPointCount);
```

Atomikus számláló

- Számláló létrehozása

```
glGenBuffers(1, &atomicBuffer);
glBindBuffer(GL_ATOMIC_COUNTER_BUFFER, atomicBuffer);
glBufferData(GL_ATOMIC_COUNTER_BUFFER, sizeof(GLuint), NULL, GL_DYNAMIC_DRAW);
glBindBuffer(GL_ATOMIC_COUNTER_BUFFER, 0);
```

- Számláló inicializálása

```
glBindBuffer(GL_ATOMIC_COUNTER_BUFFER, atomicBuffer);
GLuint* ptr = (GLuint*)glMapBufferRange(GL_ATOMIC_COUNTER_BUFFER, 0,
                                        sizeof(GLuint),
                                        GL_MAP_WRITE_BIT |
                                        GL_MAP_INVALIDATE_BUFFER_BIT |
                                        GL_MAP_UNSYNCHRONIZED_BIT);
ptr[0] = 0;
glUnmapBuffer(GL_ATOMIC_COUNTER_BUFFER);
```

- Számláló bekötése

```
glBindBufferBase(GL_ATOMIC_COUNTER_BUFFER, 0, atomicBuffer);
```

Atomikus számláló

- Számláló a shaderben
  - GLSL 420-tól elérhető
  - Deklaráció

```
layout(binding = 0, offset = 0) uniform atomic_uint counter;
```

  - Műveletek

```
uint atomicCounter(atomic_uint c);
uint atomicCounterIncrement(atomic_uint c);
uint atomicCounterDecrement(atomic_uint c);
```

Atomikus számláló