

# ゲームグラフィックス特論

---

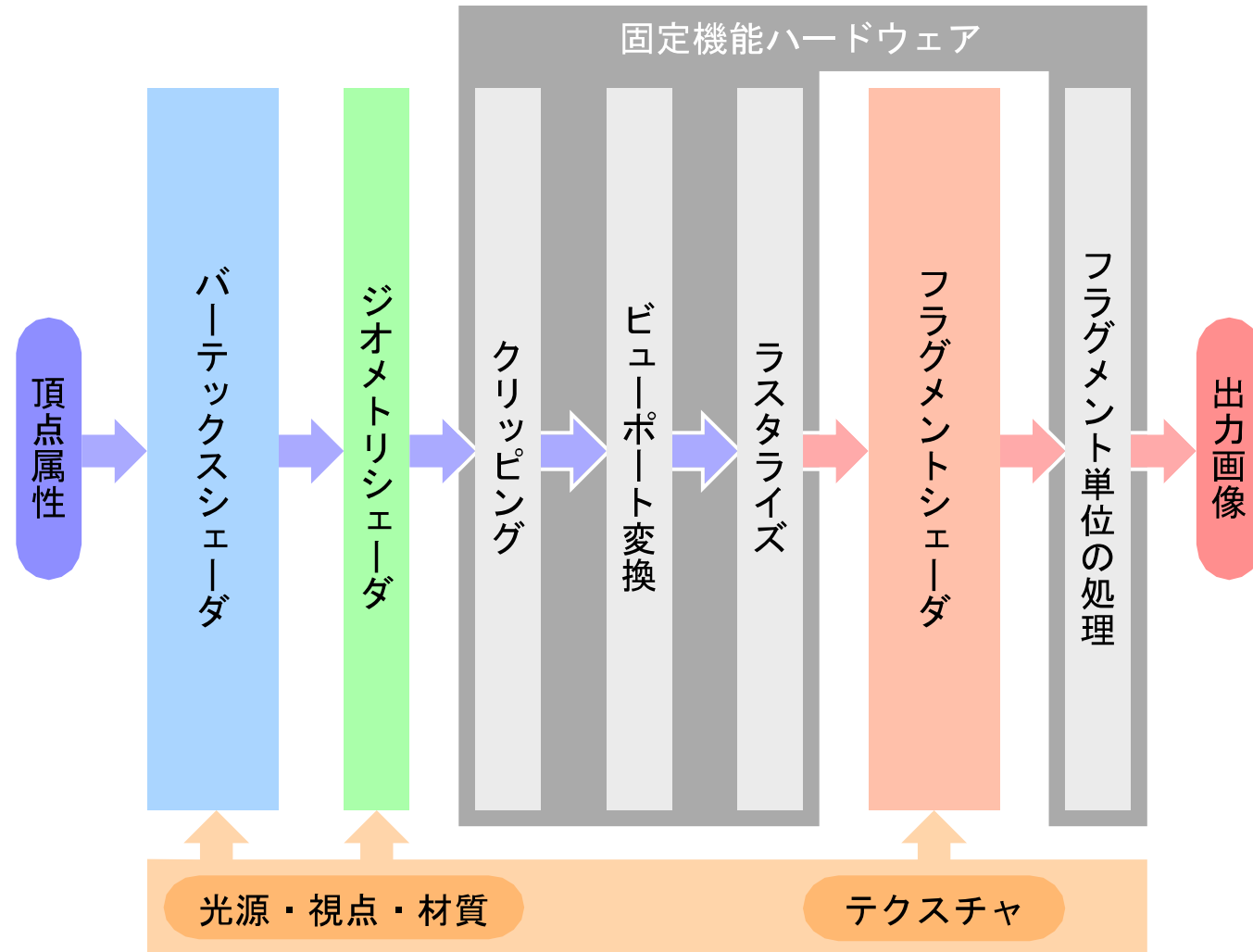
## 第14回 ジオメトリシェーダ

# 基本図形の細分化

---

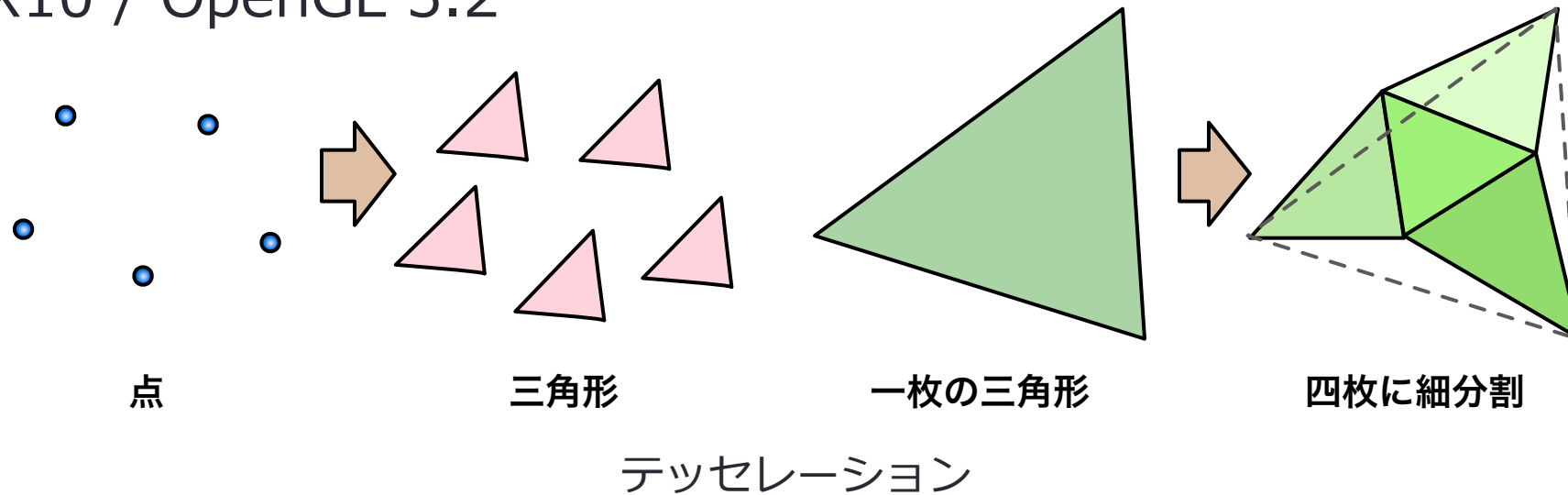
GPU 内部で基本図形を生成する

# ジオメトリシェーダの追加（第2回）

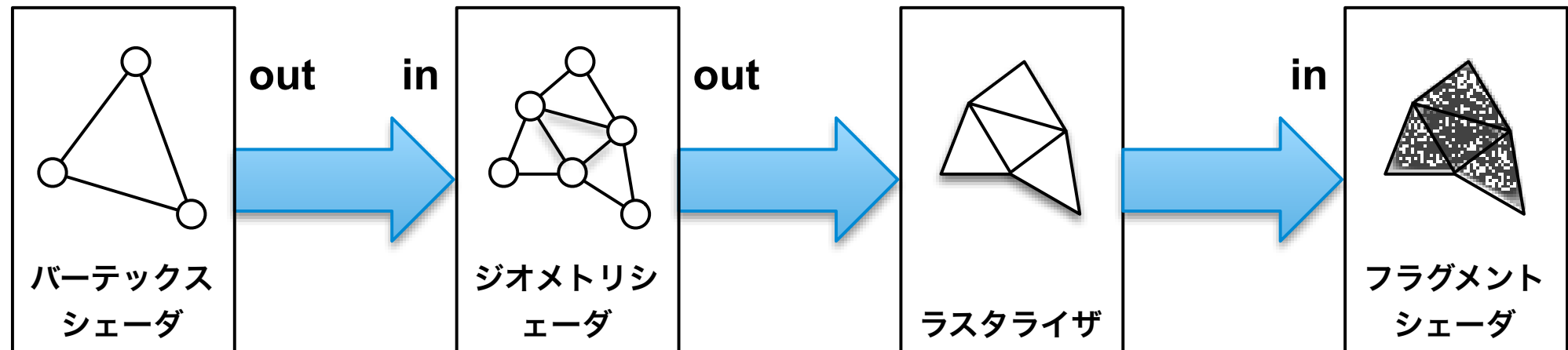
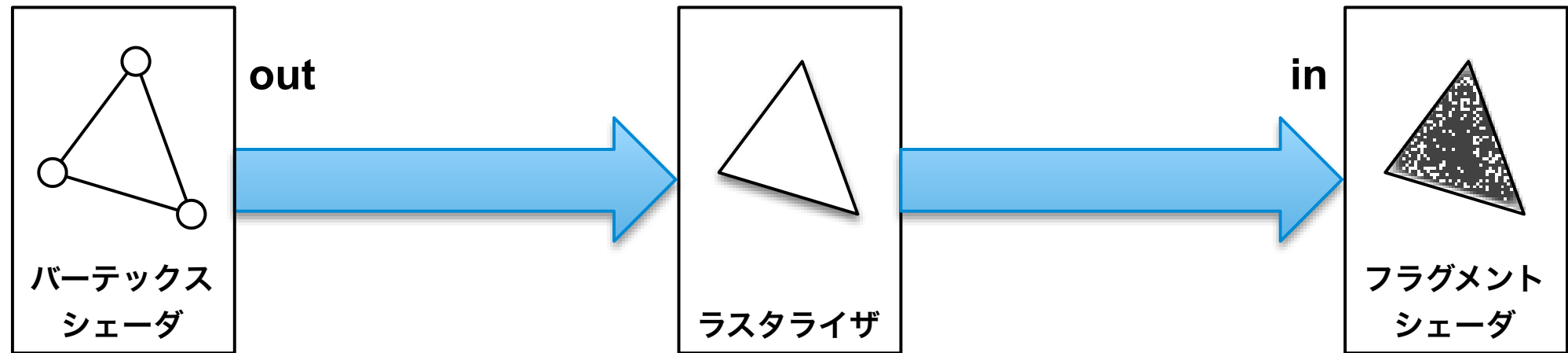


# ジオメトリシェーダ

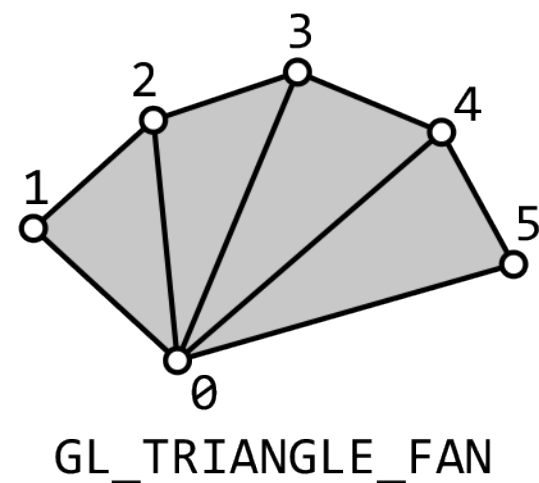
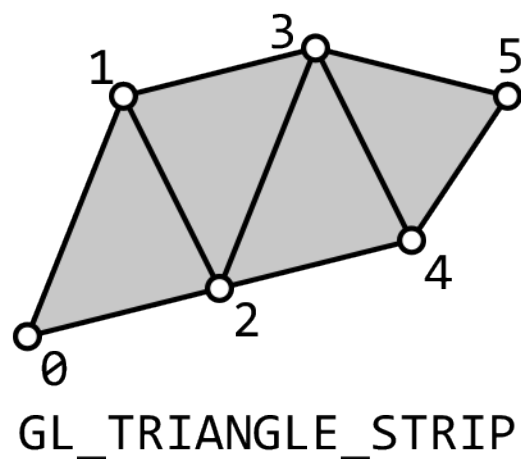
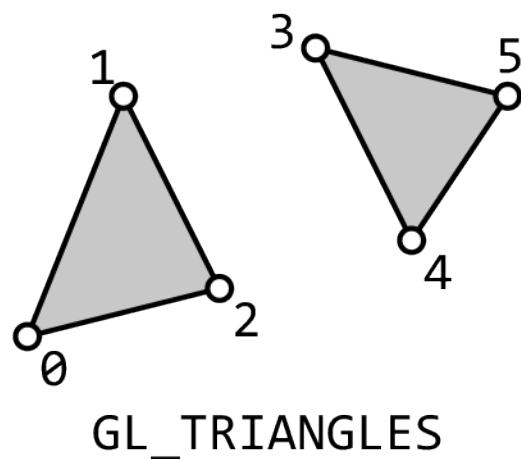
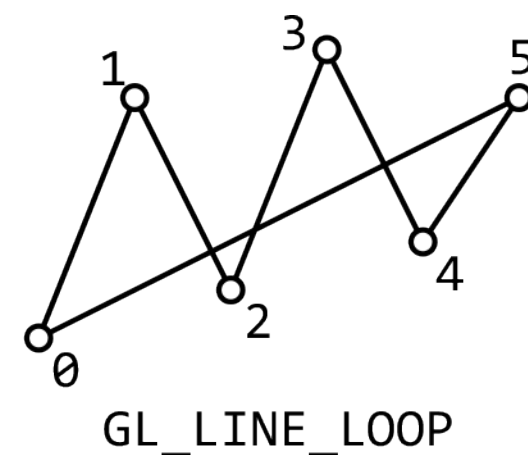
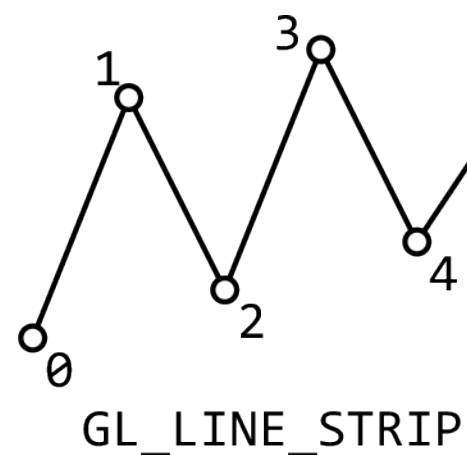
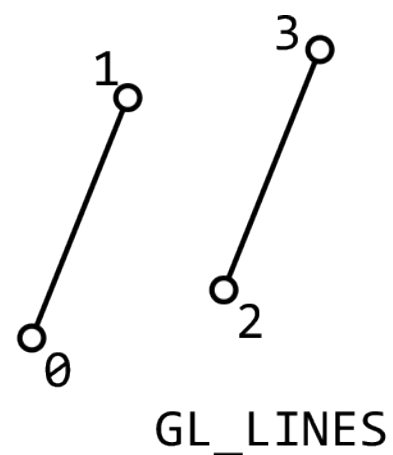
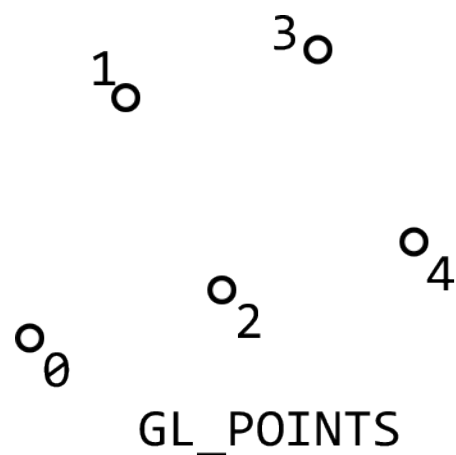
- ジオメトリデータの生成や細分化を行う
  - テッセレーション (Tessellation)
  - テッセレータという固定機能ハードウェアを制御する
  - オプション (使用しなくても良い)
  - Direct X10 / OpenGL 3.2



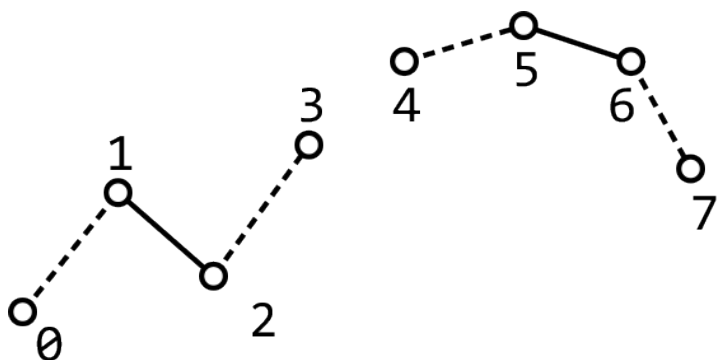
# ジオメトリシェーダの役割



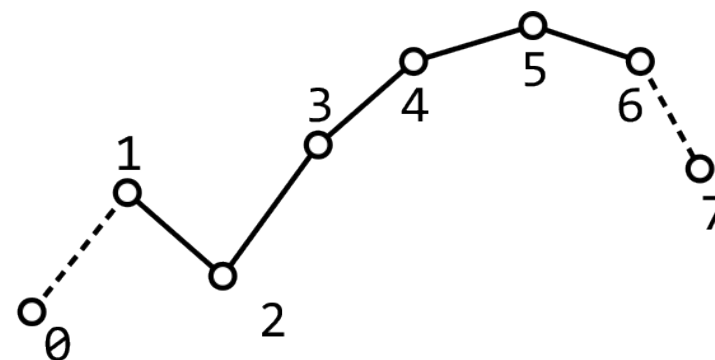
# OpenGL の基本図形



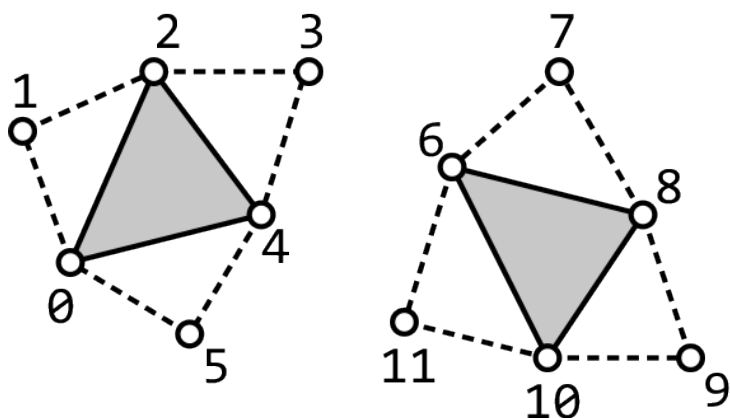
## 追加された基本図形



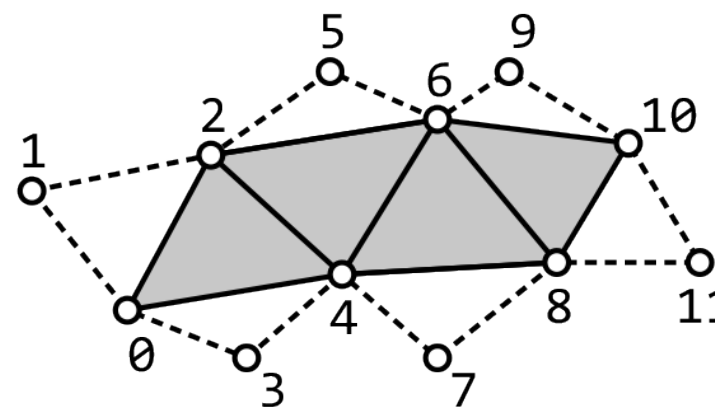
GL\_LINES\_ADJACENCY



GL\_LINE\_STRIP\_ADJACENCY

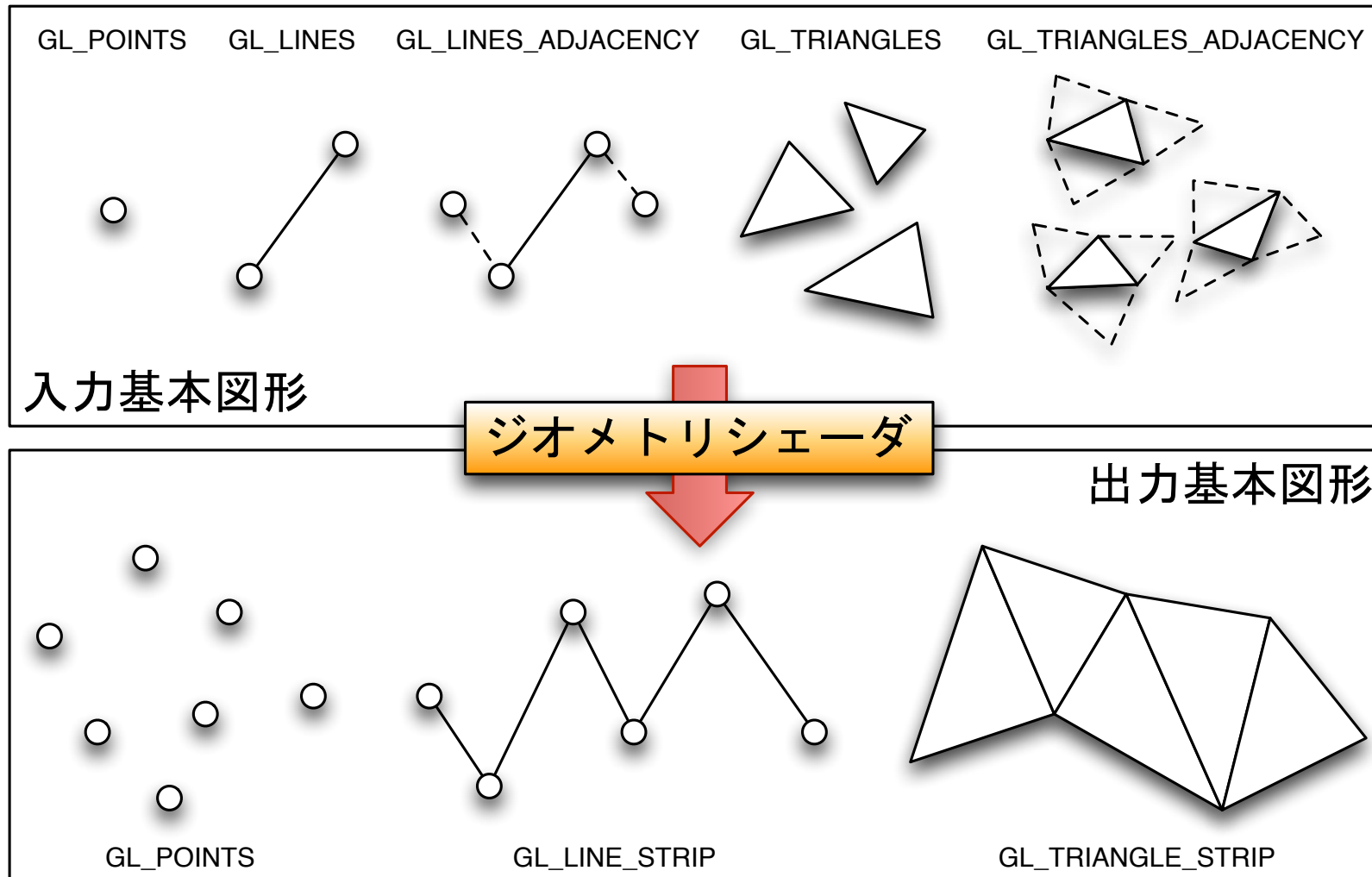


GL\_TRIANGLES\_ADJACENCY



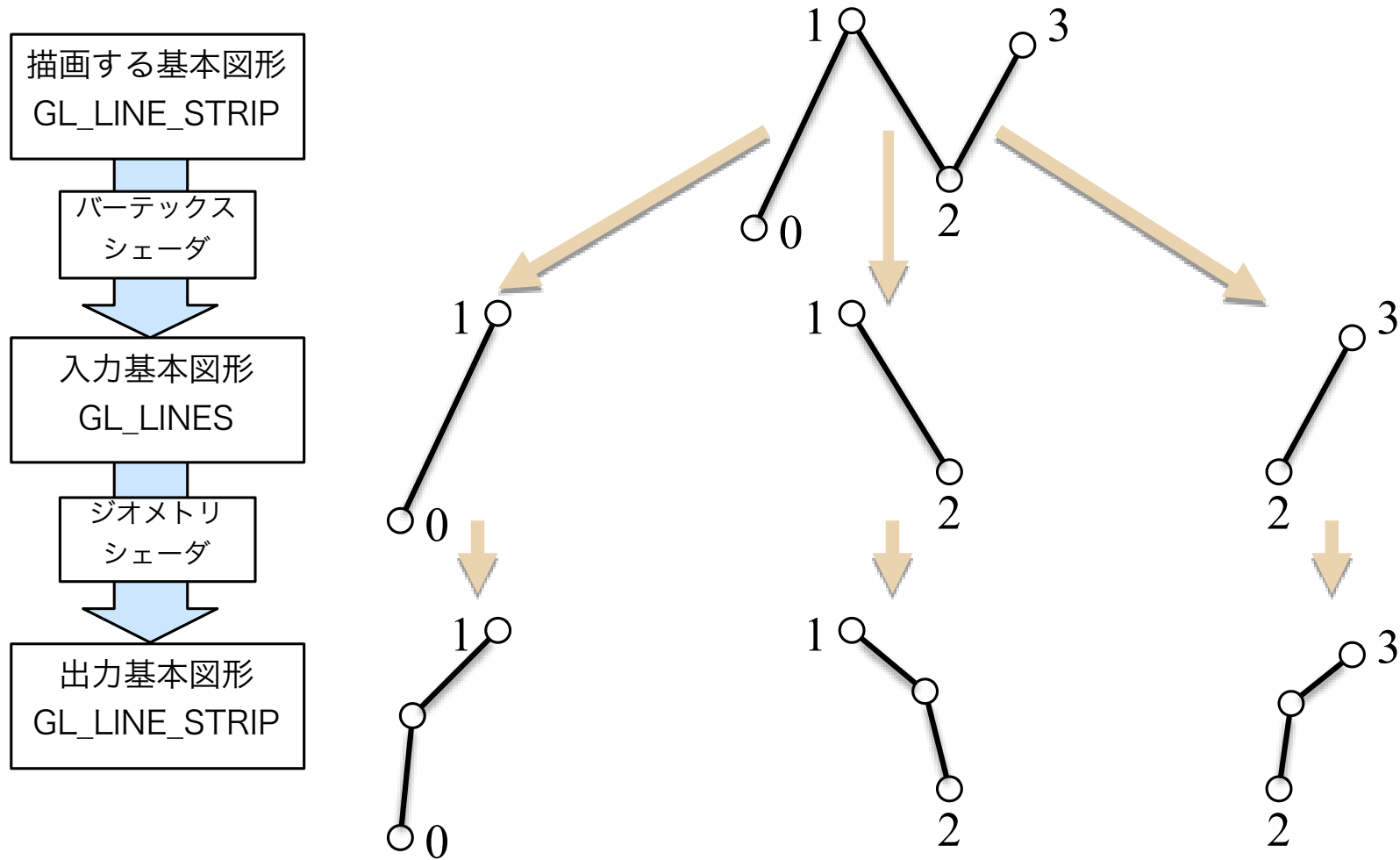
GL\_TRIANGLE\_STRIP\_ADJACENCY

# ジオメトリシェーダの入力と出力

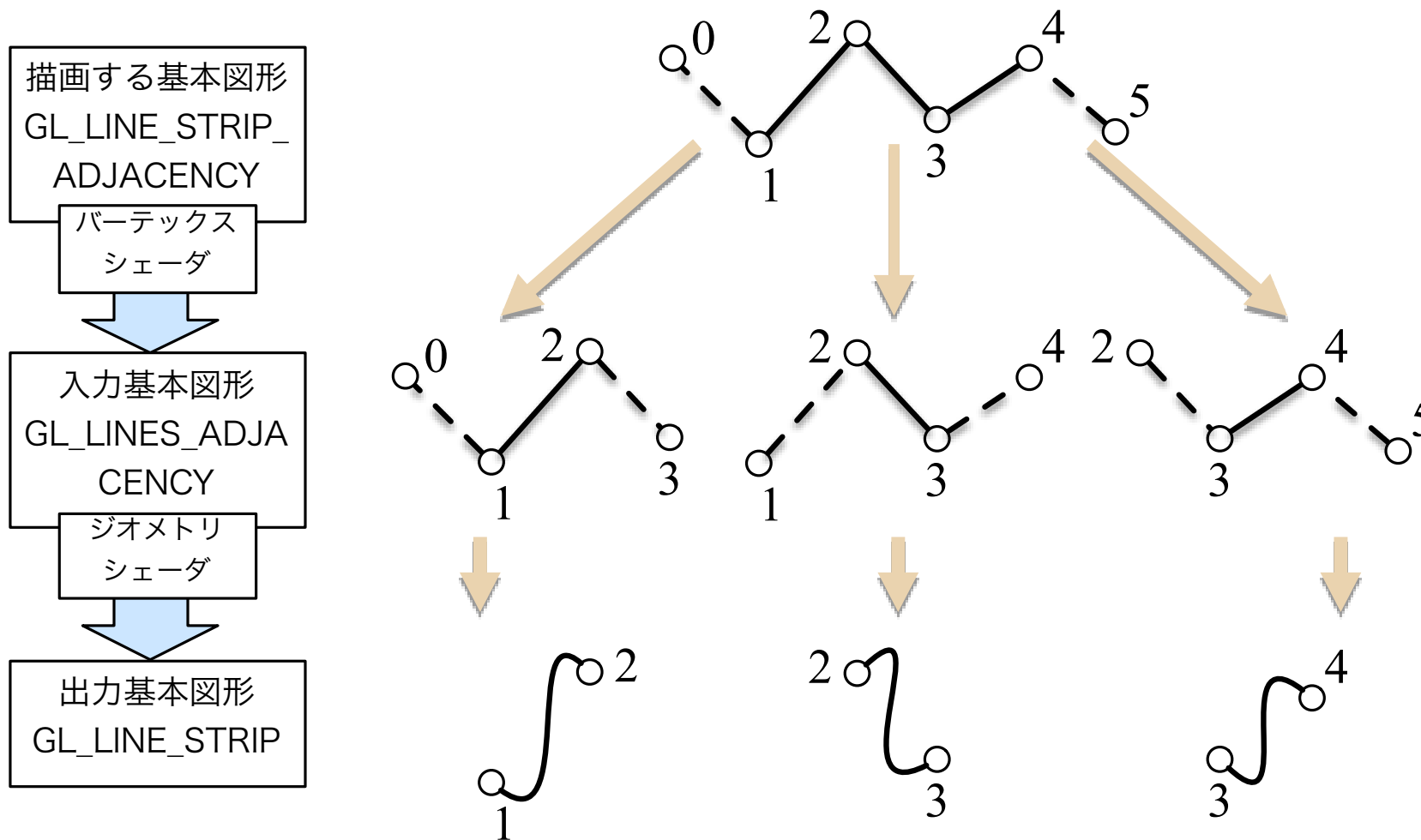




# 描画図形とジオメトリシェーダの入力



# 描画図形とジオメトリシェーダの入力



# 指定可能な入力基本図形

実際に描画する基本図形	指定可能な入力基本図形	頂点数
GL_POINTS	GL_POINTS	1
GL_LINES	GL_LINES	2
GL_LINE_STRIP		
GL_LINE_LOOP		
GL_LINES_ADJACENCY	GL_LINES_ADJACENCY	4
GL_LINE_STRIP_ADJACENCY		
GL_TRIANGLES	GL_TRIANGLES	3
GL_TRIANGLE_FAN		
GL_TRIANGLE_STRIP		
GL_TRIANGLES_ADJACENCY	GL_TRIANGLES_ADJACENCY	6
GL_TRIANGLE_STRIP_ADJACENCY		

# ジオメトリシェーダの入出力基本図形の指定

- 入力基本図形の指定

- `glProgramParameteri(program, GL_GEOMETRY_INPUT_TYPE, input);`
- `input` は `GL_POINTS`, `GL_LINES`, `GL_LINES_ADJACENCY`, `GL_TRIANGLES`, `GL_TRIANGLES_ADJACENCY`

- 出力基本図形の指定

- `glProgramParameteri(program, GL_GEOMETRY_OUTPUT_TYPE, output);`
- `output` は `GL_POINTS`, `GL_LINE_STRIP`, `GL_TRIANGLE_STRIP`

# ジオメトリシェーダが出力可能な頂点数の算出

- ジオメトリシェーダが出力できる頂点数には上限がある
  - `GLint vertices, components;`
- 頂点数の上限値の取得
  - `glGetIntegerv(GL_MAX_GEOMETRY_OUTPUT_VERTICES, &vertices);`
- 要素数の上限値の取得
  - `glGetIntegerv(GL_MAX_GEOMETRY_TOTAL_OUTPUT_COMPONENTS, &components);`
- 一つの頂点には `gl_Position` の他に複数の頂点属性を設定できる
  - 多くの頂点属性を持っていると出力できる頂点数が減る
- `GL_GEOMETRY_VERTICES_OUT` に指定可能な実際に出力する頂点数の最大値は `components ÷ (頂点属性の要素数)` と `vertices` の少ない方
  - `glProgramParameteri(program, GL_GEOMETRY_VERTICES_OUT, 少ない方);`

# ジオメトリシェーダの使用設定

↑ソースプログラムのコンパイル

```
// ジオメトリシェーダに入力する基本図形の指定
glProgramParameteri(program, GL_GEOMETRY_INPUT_TYPE, input);
// ジオメトリシェーダから出力する基本図形の指定
glProgramParameteri(program, GL_GEOMETRY_OUTPUT_TYPE, output);
// ジオメトリシェーダが出力可能な頂点数と要素数
GLint vertices, components;
// ジオメトリシェーダが出力可能な頂点数の最大値を得る
glGetIntegerv(GL_MAX_GEOMETRY_OUTPUT_VERTICES, &vertices);
// ジオメトリシェーダが出力可能な要素数の最大値を得る
glGetIntegerv(GL_MAX_GEOMETRY_TOTAL_OUTPUT_COMPONENTS, &components);
components /= 12; // このジオメトリシェーダプログラムの out 変数が vec4 型 3 個の場合
// ジオメトリシェーダが出力可能な頂点の最大数を設定する
if (vertices > components) vertices = components;
glProgramParameteri(program, GL_GEOMETRY_VERTICES_OUT, vertices);
```

入力基本図形

出力基本図形

↓オブジェクトプログラムのリンク

## ジオメトリシェーダの使用設定の補足

- `GL_MAX_GEOMETRY_OUTPUT_VERTICES`
  - ジオメトリシェーダが出力可能な頂点数 (`EmitVertex()` の実行数) 最大値
- `GL_MAX_GEOMETRY_TOTAL_OUTPUT_COMPONENTS`
  - ジオメトリシェーダが出力可能な要素数 (`out` 変数の要素数の合計) の最大値
- `GL_GEOMETRY_VERTICES_OUT`
  - ジオメトリシェーダプログラムによって出力する可能性のある最大出力頂点数
- 入出力基本図形や最大出力頂点数はシェーダプログラム内で設定できる

```
layout(triangles) in;  
layout(triangle_strip, max_vertices = 10) out;
```

# バーテックスシェーダ

```
#version 410

uniform mat4 mc;                // モデルビュー投影変換

layout (location = 0) in vec4 pv;    // 頂点位置
layout (location = 1) in vec4 cv;    // 頂点色

out vec4 vc;                     // ジオメトリシェーダに送る頂点色

void main(void)
{
    vc = cv;                      // 頂点色をジオメトリシェーダに送る
    gl_Position = mc * pv;        // 頂点位置をジオメトリシェーダに送る
}
```



# ジオメトリシェーダ

```
#version 410
```

```
layout (triangles) in;
```

```
layout (triangle_strip, max_vertices = 16) out;
```

```
in vec4 vc[]; // バーテックスシェーダから受け取る頂点色
```

```
out vec4 cf; // ラスタライザに送る頂点色
```

```
void main(void)
```

```
{  
    for (int i = 0; i < gl_in.length(); ++i)
```

```
    {  
        cf = vc[i]; // ラスタライザに頂点色を送る  
        gl_Position = gl_in[i].gl_Position; // ラスタライザに頂点位置を送る  
        EmitVertex();
```

```
    }  
    EndPrimitive();
```

```
}
```

ジオメトリシェーダに入力 (バーテックスシェーダから出力) する図形要素

ジオメトリシェーダから出力する図形要素

ジオメトリシェーダの in 変数は  
入力基本図形の頂点数分の配列

一度に送られてくる頂点の数

図形要素の区切り

バーテックスシェーダから出力している  
gl\_Position も入力基本図形の頂点数分の配列

# フラグメントシェーダ

```
#version 410

in vec4 cf; // ジオメトリシェーダから出力されてラスタライザで補間された画素色

layout (location = 0) out vec4 fc; // フラグメントの色

void main(void)
{
    fc = cf;
}
```

## gl\_in.length()

- バーテックスシェーダから受け取る頂点属性の数
  - ジオメトリシェーダには一度にこの数の頂点が渡される
- 入力基本形状 GL\_GEOMETRY\_INPUT\_TYPE で決まる

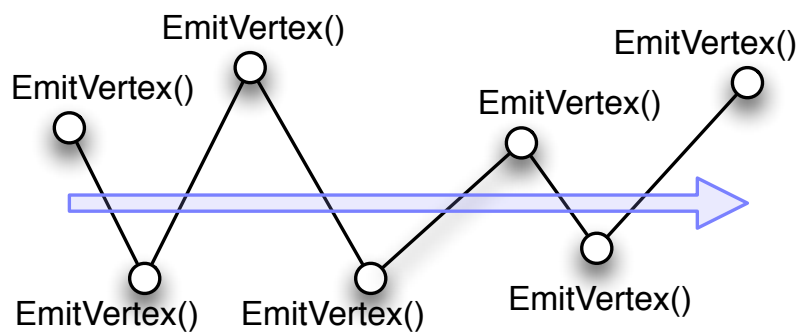
GL_GEOMETRY_INPUT_TYPE	gl_in.length()
GL_POINTS	1
GL_LINES	2
GL_LINES_ADJACENCY	4
GL_TRIANGLES	3
GL_TRIANGLES_ADJACENCY	6

# EmitVertex()

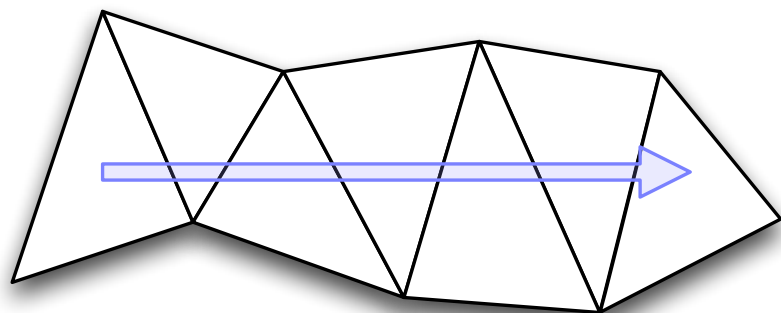
- ラスタライザに送る頂点属性を生成する
  - gl\_Position および out 変数の組
- 出力 GL\_GEOMETRY\_OUTPUT\_TYPE
  - GL\_POINTS
    - EmitVertex() を少なくとも1回実行
  - GL\_LINE\_STRIP
    - EmitVertex() を少なくとも2回実行
  - GL\_TRIANGLE\_STRIP
    - EmitVertex() を少なくとも3回実行
- EmitVertex() を実行しなければ図形は生成（表示）されない

# EndPrimitive()

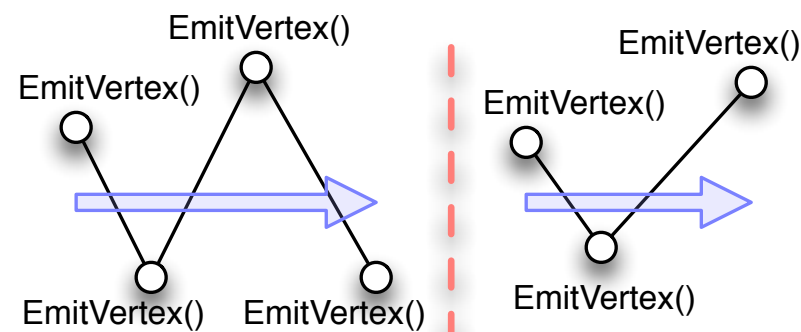
EndPrimitive() を実行しない場合



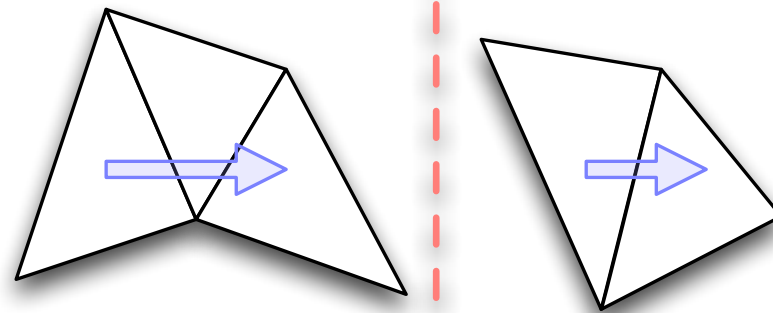
GL\_LINE\_STRIP や GL\_TRIANGLE\_STRIP を複数出力するとき



EndPrimitive() を実行した場合



EndPrimitive()



## 小テストージオメトリシェーダ

Moodle の小テストに解答してください

# 宿題

- ジオメトリシェーダを使って八面体を描いてください
  - 次のプログラムはGL\_POINTS により点を描画します.
    - <https://github.com/tokoik/ggsample14>
  - しかし, ジオメトリシェーダ ggsample14point.geom によって, この一つ一つの点は三角形に置き換えて表示されます.
  - ggsample14point.geom を書き換えて, この三角形を八面体に置き換えてください.
    - ローエンドの GPU のジオメトリシェーダから出力可能な頂点の最大数は 85 くらい.
- ggsample14point.geom 内で座標変換と陰影付け（スムーズシェーディング）を行ってください.
- ggsample14point.geom を**アップロード**してください

## 八面体

頂点位置は任意です．表示可能なサイズにしてください．

頂点の法線ベクトルは軸方向に設定してください．

GL\_TRIANGLE\_STRIP  
を二つ描けばいいと思います．

