Računarska grafika

Čas 03 - Depth, Face-Cull

Depth testing

Podrazumevani režim iscrtavanja ne uzima u obzir koji fragment se nalazi ispred, a koji iza, već se prikazuje onaj koji je poslednji nacrtan (upisan u *color buffer*).

Kako bi se fragmenti koji su ispred ostalih fragmenata tako i prikazali, nezavisno od redosleda iscrtavanja, potrebno je uključiti *depth test*:

```
// Enable depth testing
glEnable(GL_DEPTH_TEST);
```

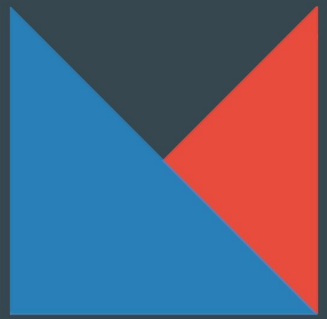
Depth testing

Ukoliko je *depth testing* uključen, pre upisa fragmenta u *color buffer* OpenGL testira vrednost “dubine” tog fragmenta u *depth buffer*-u*. Ukoliko fragment prođe test, vrednost njegove “dubine” se ažurira u *depth buffer*-u i taj fragment se upisuje u *color buffer*.

* *Depth* i *color buffer* su istih dimenzija

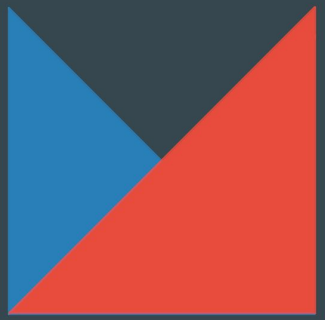
Depth testing - off (default)

```
float VerticesRed[] = {
    0.0f, 0.0f, 0.0f, 1.0f, 0.0f, 0.0f
    0.5f, 0.0f, 0.0f, 1.0f, 0.0f, 0.0f
    0.5f, 0.5f, 0.0f, 1.0f, 0.0f, 0.0f
};
// …Draw red…
float VerticesBlue[] = {
    0.0f, 0.0f, -1.0f, 0.0f, 0.0f, 1.0f
    0.0f, 0.5f, -1.0f, 0.0f, 0.0f, 1.0f
    0.5f, 0.0f, -1.0f, 0.0f, 0.0f, 1.0f
};
// …Draw blue…
```



Depth testing - on

```
// Enable depth testing
glEnable(GL_DEPTH_TEST);
glBegin(GL_TRIANGLES);
float VerticesRed[] = {
    0.0f, 0.0f, 0.0f, 1.0f, 0.0f, 0.0f
    0.5f, 0.0f, 0.0f, 1.0f, 0.0f, 0.0f
    0.5f, 0.5f, 0.0f, 1.0f, 0.0f, 0.0f
};
// …Draw red…
float VerticesBlue[] = {
    0.0f, 0.0f, -1.0f, 0.0f, 0.0f, 1.0f
    0.0f, 0.5f, -1.0f, 0.0f, 0.0f, 1.0f
    0.5f, 0.0f, -1.0f, 0.0f, 0.0f, 1.0f
};
// …Draw blue…
```



Depth buffer

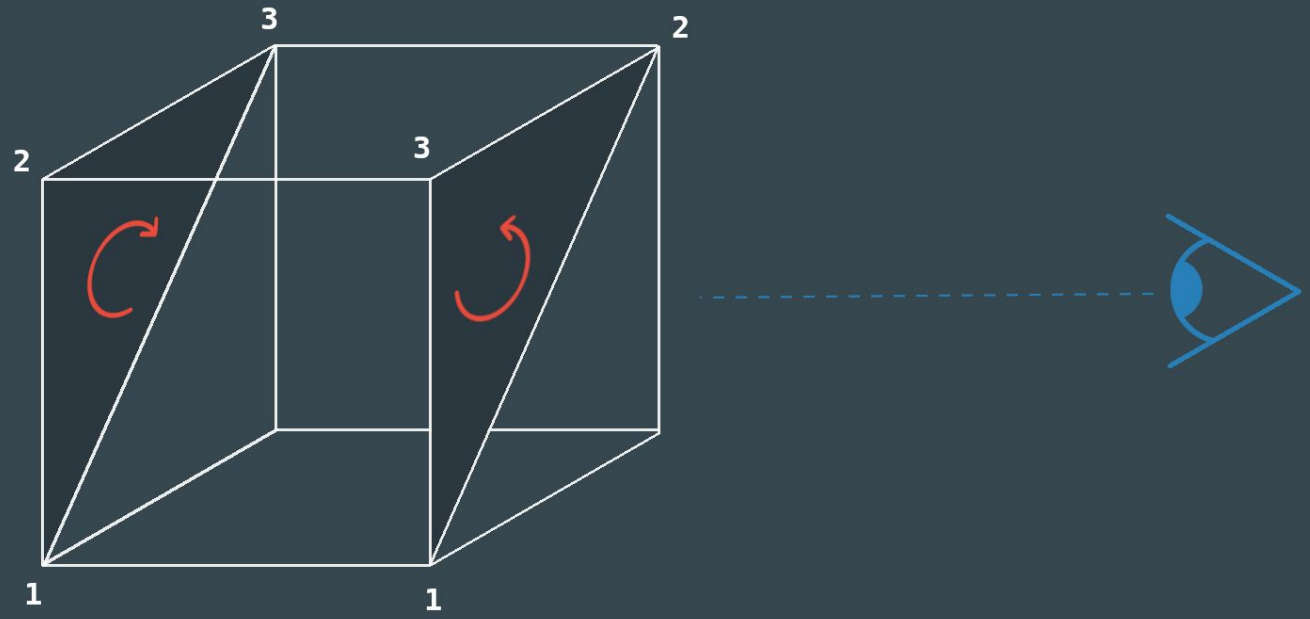
1	∞	∞	∞	∞	0
1	1	∞	∞	0	0
1	1	1	0	0	0
1	1	0	0	0	0
1	0	0	0	0	0
0	0	0	0	0	0

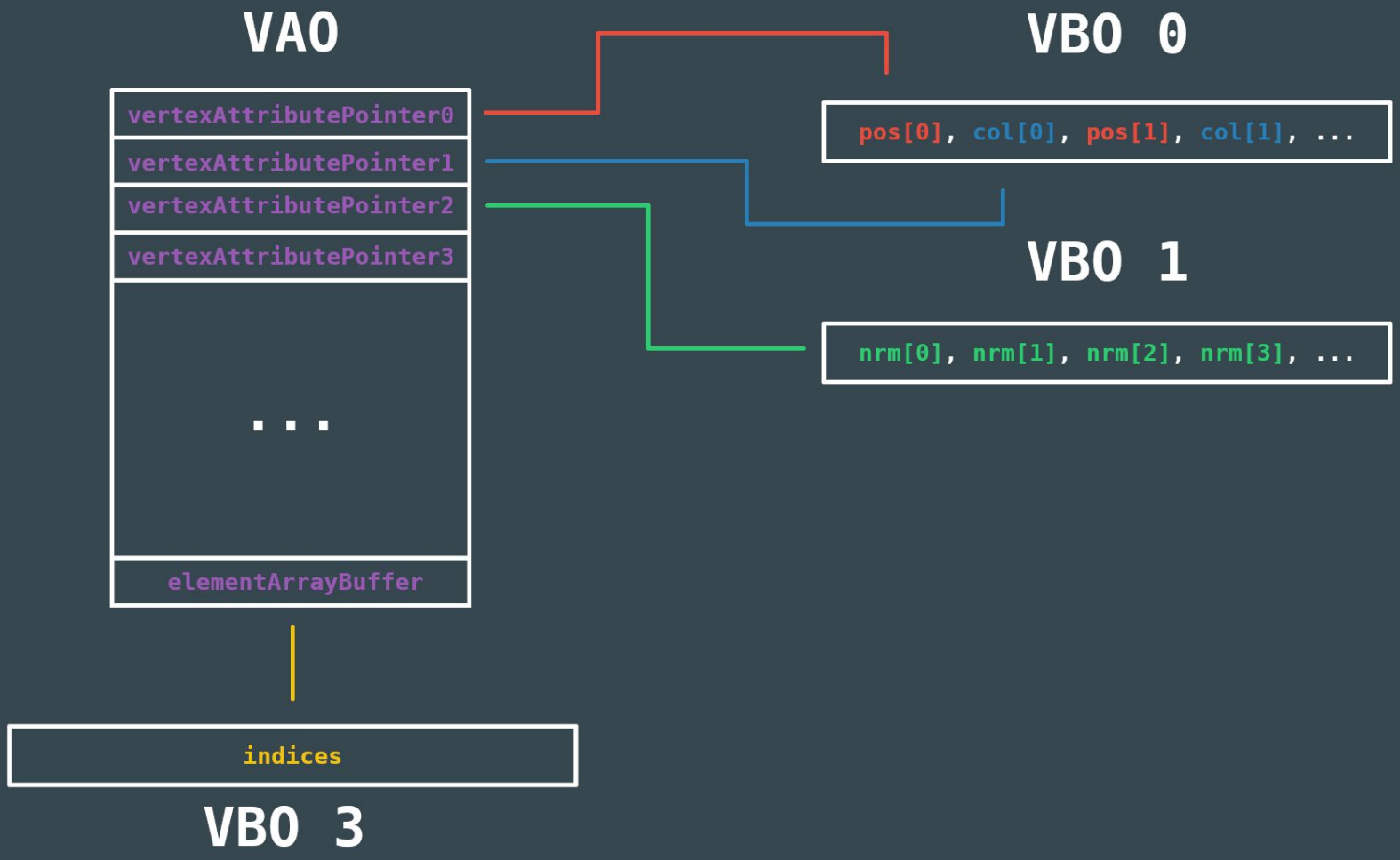
Back-face culling

OpenGL, u podrazumevanom stanju, iscrtava i prednju i zadnju stranu fragmenta (prednja strana je u pravcu normale). Ukoliko ne želimo da iscrtavamo zadnju stranu (zbog performanse) možemo podesiti uključiti *back-face culling*:

```
// Enable back-face culling
glEnable(GL_CULL_FACE);
```

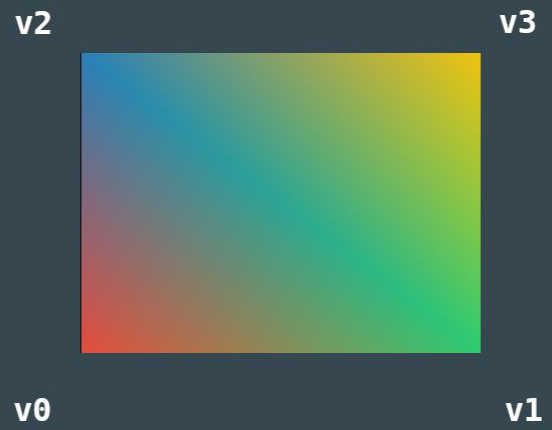
Back-face culling




VAO
vertexAttributePointer0
vertexAttributePointer1
vertexAttributePointer2
vertexAttributePointer3
...
elementArrayBuffer
indices
VBO 3
VBO 0
pos[0], col[0], pos[1], col[1], ...
VBO 1
nrm[0], nrm[1], nrm[2], nrm[3], ...

VAO - primer 1

```
uint Indices = {
    0, 1, 2,
    2, 1, 3
};
```



VAO - primer 1

```
// . . . VBO generation, buffering, vertex attrib. pointers...
// Bind second VBO(EBO) to GL_ELEMENT_ARRAY_BUFFER and load indices into it
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, VBOs[1]);
glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(Indices), Indices,
GL_STATIC_DRAW);

. . .
// Drawing
glBindVertexArray(VAO);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, VBOs[1]);
glDrawElements(GL_TRIANGLES, ArrayCount(Vertices) / 6, GL_UNSIGNED_INT,
(void*)0);
```