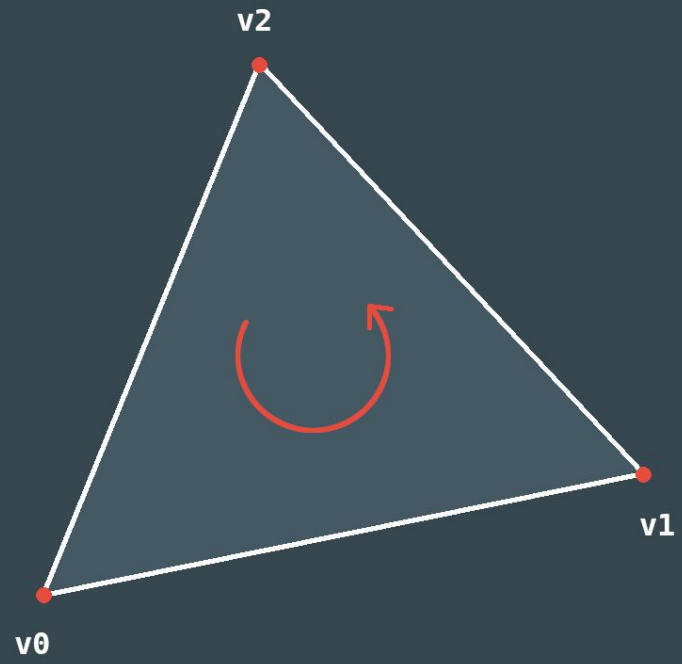
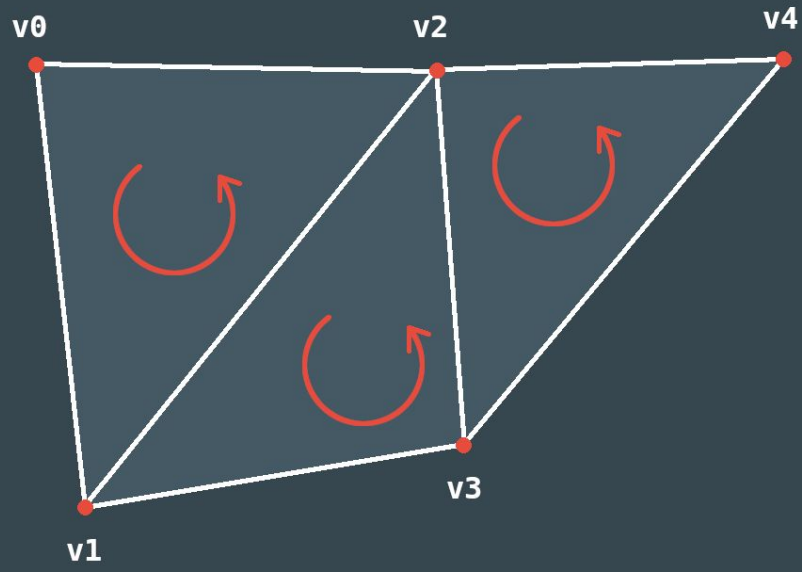
Računarska grafika

Čas 02 - Hello Triangle

Orijentacija poligona

Lice poligona određeno je normalom čiji pravac zavisi od redosleda definisanja temena i određuje se pravilom desne ruke (moguće menjati).


v2
v1
v0

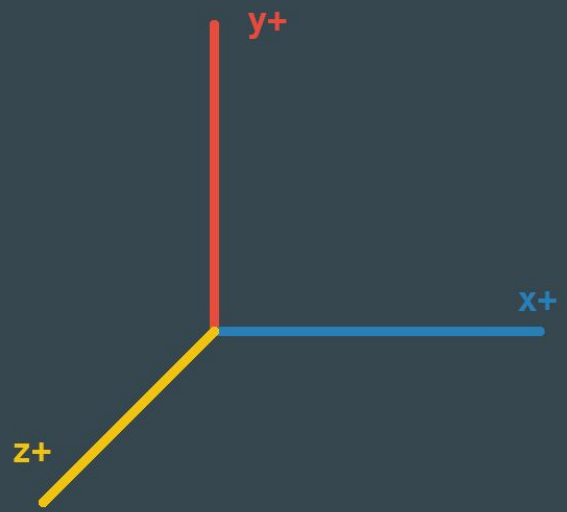

v0
v2
v4
v3
v1

Triangle strip

Prethodno prikazani triangle strip iscrtan je u redosledu: v0, v1, v2, pa zatim v2, v1, v3, pa v2, v3, v4 ...

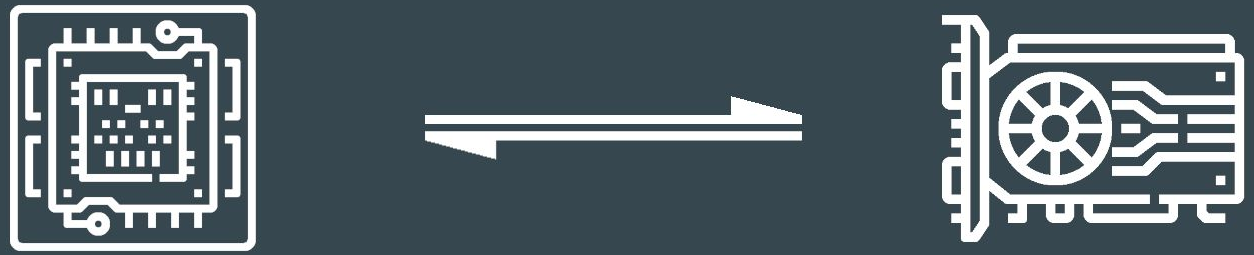
Koordinatni sistem

OpenGL koristi Dekartov desni pravougli koordinatni sistem sa koordinatnim početkom u sredini OpenGL kontrole.



Client - Server

Arhitektura OpenGL-a zasnovana je na ***client-server*** arhitekturi. Aplikacija pisana da koristi OpenGL API je “klijent” i izvršava se na CPU, dok implementacija samog OpenGL *engine*-a predstavlja “server” koji se izvršava na GPU.



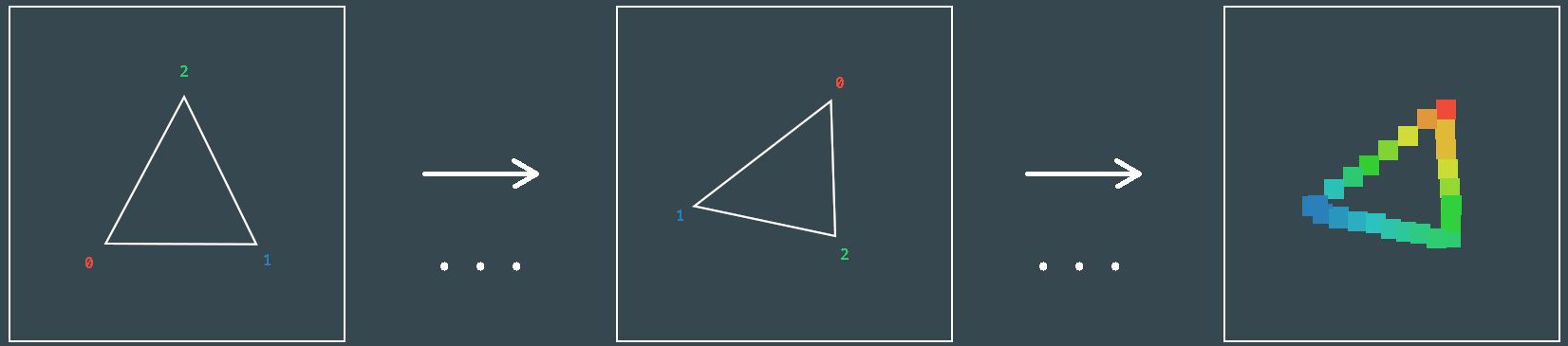
CPU
(client)

GPU
(server)

Client - Server

Modelovanje, *render*-ovanje (iscrtavanje) i interakcija su saradnički procesi između CPU klijentskog programa i serverskog GPU dela, te je važan deo dizajn procesa upravo adekvatna raspodela poslova i planiranje najboljeg načina slanja podataka CPU <-> GPU. Ne postoji standardni “najbolji” način koji pokriva sve slučaje.

Pajplajn



Buffer Data

Vertex Shader

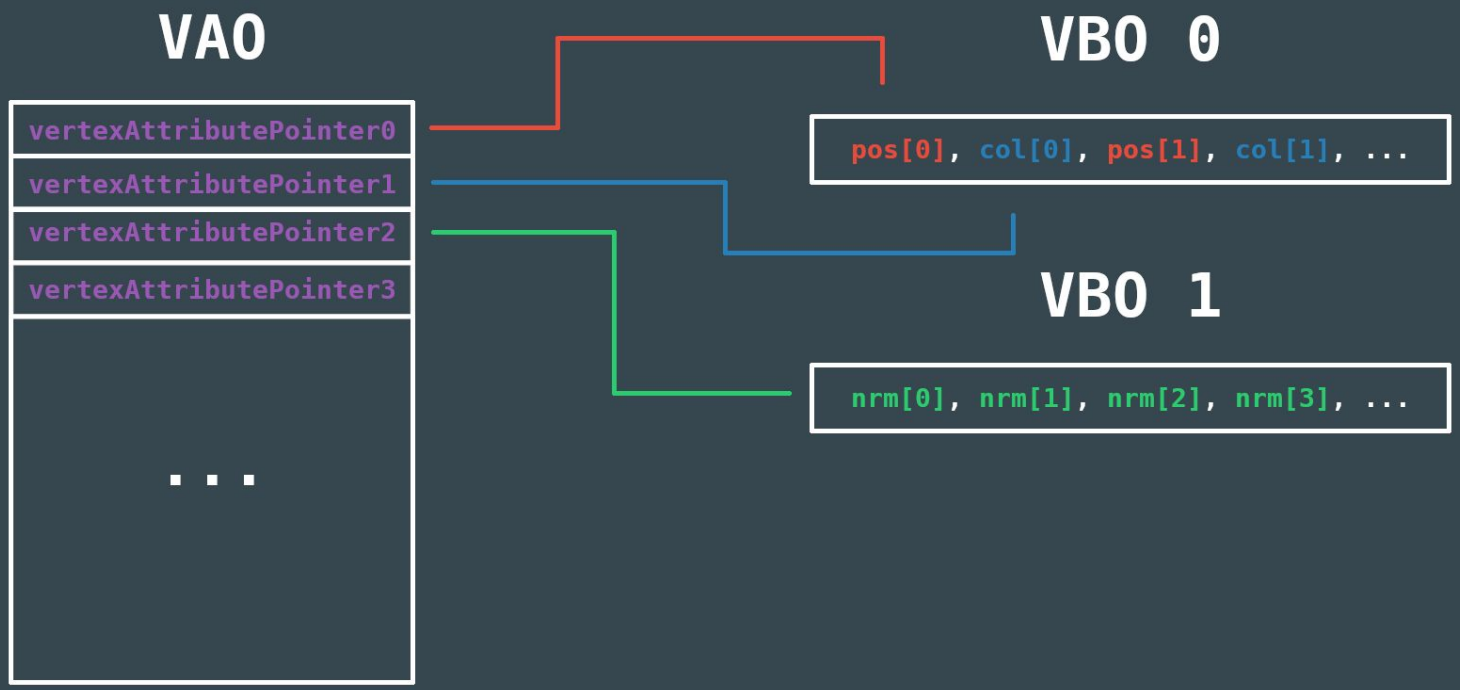
Fragment Shader

VAO, VBO

- **V**ertex**B**uffer**O**bject - buffer *vertex*-a koji se čuva na GPU
- **V**ertex**A**rray**O**bject - objekat u kom se čuva jedan ili više VBO-a

Iscrtavanje putem VAO i VBO takođe nudi značajno bolje performanse u odnosu na *immediate-mode rendering* (iscrtavanje).

Unutar VBO možemo čuvati podatke o poziciji, boji, normali itd...


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

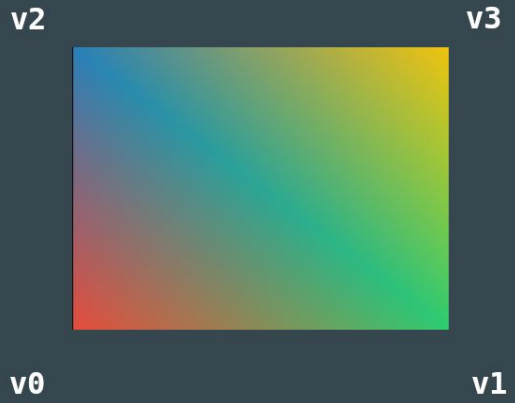
VAO - primer 0

```
float Vertices[] = {
    //     position     |      colour
    // X      Y     Z     R     G     B
    -0.5f, -0.5f, 0.0f, 0.905f, 0.298f, 0.235f, // v0
     0.5f, -0.5f, 0.0f, 0.180f, 0.8f,   0.443f, // v1
    -0.5f,  0.5f, 0.0f, 0.160f, 0.501f, 0.725f, // v2

    -0.5f,  0.5f, 0.0f, 0.160f, 0.501f, 0.725f, // v2
     0.5f, -0.5f, 0.0f, 0.180f, 0.8f,   0.443f, // v1
     0.5f,  0.5f, 0.0f, 0.945f, 0.768f, 0.058f, // v3
};
```

VAO - primer 0

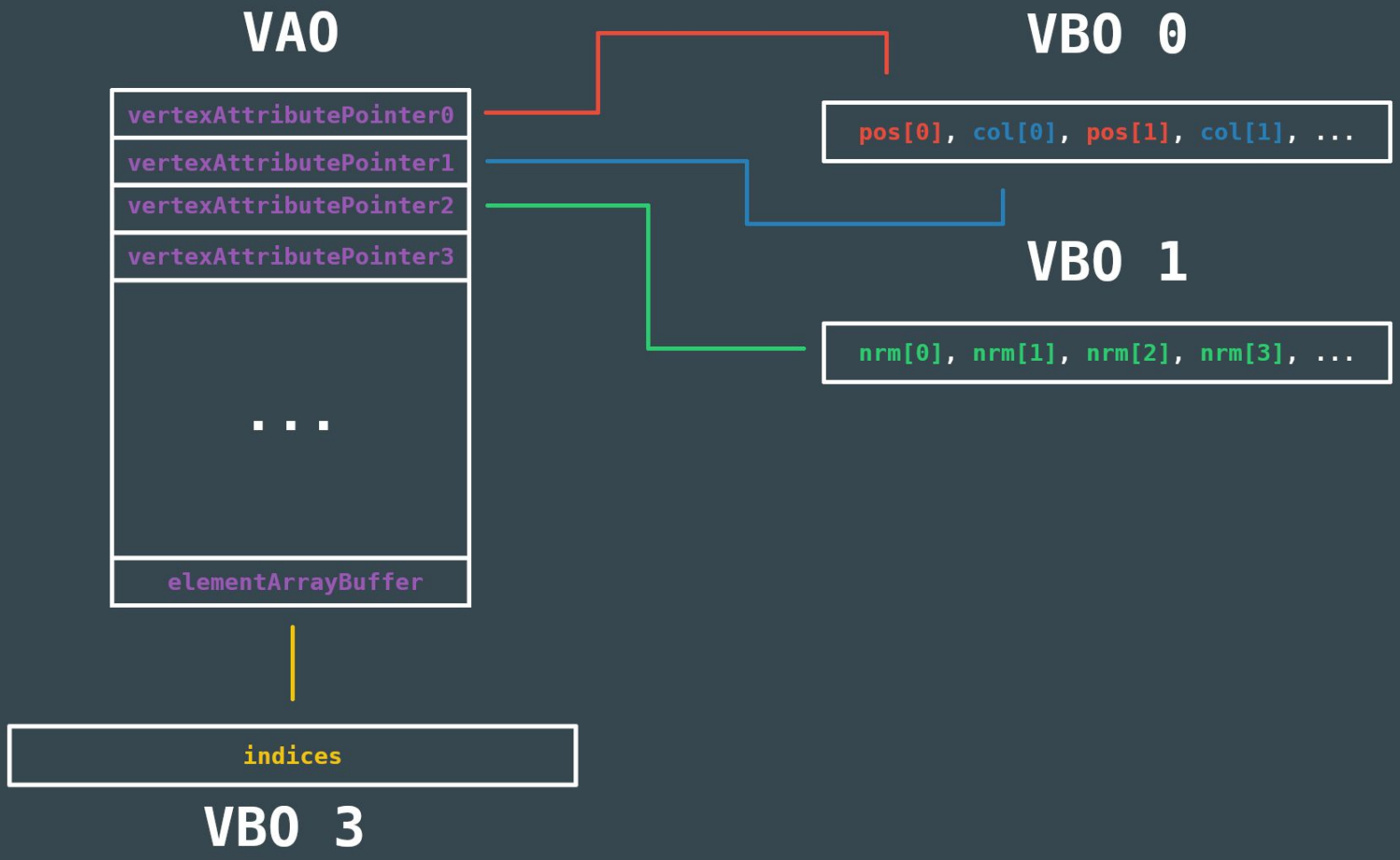
```
// Generate VBOs
unsigned VBOs[2];
int Stride = 6 * sizeof(float);
glGenBuffers(2, VBOs);

// Bind first VBO to GL_ARRAY_BUFFER and load data into it
glBindBuffer(GL_ARRAY_BUFFER, VBOs[0]);
glBufferData(GL_ARRAY_BUFFER, Vertices, GL_STATIC_DRAW);

// Set pointers to buffer segments and enable them
glVertexAttribPointer(0, 3, GL_FLOAT, false, Stride, (void*)(0));
glVertexAttribPointer(1, 3, GL_FLOAT, false, Stride, (void*)(3 *
sizeof(float)));
glEnableVertexAttribArray(0); glEnableVertexAttribArray(1);

. . .
// Drawing
glEnableClientState(GL_VERTEX_ARRAY);
glBindBuffer(GL_ARRAY_BUFFER, m_modelsVBO[0]);
glDrawArrays(GL_TRIANGLES, 0, ArrayCount(Vertices) / 6);
```

VAO - primer 0



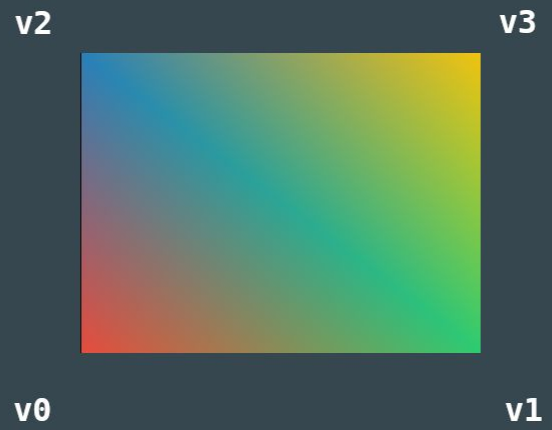
Indeksirani VAO

Kako ne bismo smanjili količinu redudantnih *vertex*-a čuvanih u memoriji, moguće je čuvati indekse *vertex*-a u VBO i tako definisati redosled njihovog korišćenja i omogućiti njihovo višestruko korišćenje.


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, Indices, GL_STATIC_DRAW);

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