Vježba L02

Jakov Spahija

28. svibnja 2021.

Sadržaj

1. Vectors and Matrices

Prijepis 1.1. Vector4 operatori

```
28        T& operator[](uint32_t i){
29              return v[i];
30        }
31        const T& operator[](uint32_t i) const {
32              return (const T)(v[i]);
33        }
34
35        Vector4<T> inline operator*(const Matrix4x4<T>& mat){
36              return Vector4<T>{
37                          v[0] * mat[0][0] + v[1] * mat[1][0] + v[2] * mat
                                [2][0] + v[3] * mat[3][0],
38                          v[0] * mat[0][1] + v[1] * mat[1][1] + v[2] * mat
                                [2][1] + v[3] * mat[3][1],
39                          v[0] * mat[0][2] + v[1] * mat[1][2] + v[2] * mat
                                [2][2] + v[3] * mat[3][2],
40                          v[0] * mat[0][3] + v[1] * mat[1][3] + v[2] * mat
                                [2][3] + v[3] * mat[3][3],
41              };
42        }
43
44        Vector4<T> inline operator-(const Vector4<T>& vec){
45              return Vector4<T>(
46                    v[0] - vec[0],
47                          v[1] - vec[1],
48                          v[2] - vec[2],
49                          v[3] - vec[3]
50              );
51        }
52
53        Vector4<T> inline operator/(const T& k) {
54              return Vector4<T>(
55                    v[0]/k,
56                    v[1]/k,
57                    v[2]/k,
58                    v[3]/k
59                    );
60        }
61
62        std::string ToString() const {
63              std::stringstream output;
64              output << "[" << v[0] << ", " << v[1] << ", " << v[2] << ", "
                      << v[3] << "]";
65              return output.str();
66        }
```

Prijepis 1.2. Vector4 operacije

```
68
69  template<typename T>
70  T Length(Vector4<T>& vec){
71        return std::sqrt(vec[0]* vec[0]+ vec[1]* vec[1]+ vec[2]* vec[2]+ vec
              [3]* vec[3]);
72  }
73
74  template<typename T>
75  T Dot(Vector4<T>& v1,Vector4<T>& v2){
76        return v1[0]*v2[0]+v1[1]*v2[1]+v1[2]*v2[2];
77  }
78
79  template<typename T>
80  Vector4<T> Cross(Vector4<T>& v1,Vector4<T>& v2){
81        return  Vector4<T>{
82              (v1[1] * v2[2]) - (v1[2] * v2[1]),
83              (v1[2] * v2[0]) - (v1[0] * v2[2]),
84              (v1[0] * v2[1]) - (v1[1] * v2[0]),
85              T()
86        };
87  }
88
89  template<typename T>
90  Vector4<T> Normalize(Vector4<T>& vec){
91        return Vector4<T>(
92              vec[0] / Length(vec),
93                    vec[1] / Length(vec),
94                    vec[2] / Length(vec),
95                    vec[4] / Length(vec)
96        );
97  }
98
99  typedef Vector4<float> float4;
```

Prijepis 1.3. Matrix4x4 operator[]

```
26        T* operator[](uint32_t i) {
27              return M[i];
28        }
29        const T* operator[](uint32_t i) const {
30              return (const T*)M[i];
31        }
```

Prijepis 1.4. Matrix4x4 mul

```
50  template<typename T>
51  inline Matrix4x4<T> Multiply(const Matrix4x4<T>& mat1,const Matrix4x4<T>&
        mat2){
52        Matrix4x4<T> mat {0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0};
53        for(int i = 0; i < 4; i++){
54              for(int j = 0; j < 4; j++){
55                    for(int k = 0; k < 4; k++){
56                          mat[i][j] += mat1[i][k] * mat2[k][j];
57                    }
58              }
59        }
60        return mat;
61  }
```

Terminal

```
u: [1, 1, 1, 0]
||u||: 1.73205
v: [1, 1, 1, 0]
w = u x v [0, 0, 0, 0]
u . v = 3
A:
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15

B:
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15

C = Multiply(A, B):
56 62 68 74
152 174 196 218
248 286 324 362
344 398 452 506

D = A*B:
56 62 68 74
152 174 196 218
248 286 324 362
344 398 452 506

u = v * M: [14, 15, 16, 0]

v.w = 1
u = v * M: [18, 18, 18, 1]
```

2. Vertices and Models

Prijepis 2.1. Model.h

```
6  class Model {
7  private:
8        std::vector<Vertex> _vertices;
9  public:
10       Model() {}
11       Model( const Vertex& vtx1,
12                    const Vertex& vtx2,
13                    const Vertex& vtx3,
14                    const Vertex& vtx4)
15       {
16             _vertices.push_back(vtx1);
17             _vertices.push_back(vtx2);
18             _vertices.push_back(vtx3);
19             _vertices.push_back(vtx4);
20       }
21
22       std::string ToString() const {
23             std::stringstream info;
24             info << std::endl;
25             for (auto it = begin(_vertices); it < end(_vertices); ++it) {
26                   info << it.operator*().ToString() << std::endl;
27             }
28             return info.str();
29       }
30
31       void AddVertex(const Vertex& vertex) {
32             _vertices.push_back(vertex);
33       }
34       Vertex* GetVerticesAddress(uint32_t i) {
35             return &(this->_vertices[i]);
36       }
37       uint32_t GetVertexCount() {
38             return _vertices.size();
39       }
40 };
```

Terminal

```
Model:

Pozicija: [-1, 1, 0, 1] Boja: [255, 0, 0, 0]
Pozicija: [1, 1, 0, 1] Boja: [0, 255, 0, 0]
Pozicija: [1, -1, 0, 1] Boja: [0, 0, 255, 0]
Pozicija: [-1, -1, 0, 1] Boja: [125, 125, 125, 0]
```

3. Transforms

3.1 World Transform

$$S = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}; R_z = \begin{pmatrix} cos(\theta) & 0 & -sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ sin(\theta) & 0 & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}; T = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix};$$

$$W = S * R * T$$

World Transform

```
6  float4x4 CreateScaleMatrix(float sx = 1, float sy = 1, float sz = 1) {
7        return float4x4{
8              sx,0,0,0,
9              0,sy,0,0,
10             0,0,sz,0,
11             0,0,0,1
12       };
13 }
14
15 float4x4 CreateRotateZMatrix(float angle = 0.0f) {
16       return float4x4{
17             std::cos(angle),std::sin(angle),0,0,
18             -std::sin(angle),std::cos(angle),0,0,
19             0,0,1,0,
20             0,0,0,1
21       };
22 }
23
24 float4x4 CreateTranslateMatrix(float tx = 0.0f, float ty = 0.0f, float tz =
       0.0f) {
25       return float4x4{
26             1,0,0,0,
27             0,1,0,0,
28             0,0,1,0,
29             tx,ty,tz,1
30       };
31 }
```

3.2 Camera Transform

Matrica pogleda, se sastavlja poznavajući poziciju kamere *eye*, poziciju točke na koju se gleda *target*, vektor orijentacije prema gore *up*.

Nova orijentacija kamere su vektori $\mathbf{r}$, $\mathbf{u}$, $\mathbf{v}$, a pozicija kamere $\mathbf{t} = \mathbf{eye}$.

$\mathbf{v} = (\mathbf{target} - \mathbf{eye})/\|\mathbf{target} - \mathbf{eye}\|$
$\mathbf{r} = (\mathbf{up} \times \mathbf{v})/\|\mathbf{up} \times \mathbf{v}\|$
$\mathbf{u} = (\mathbf{r} \times \mathbf{v})$

$$M = \underbrace{\begin{pmatrix} r_x & u_x & v_x & 0 \\ r_y & u_y & v_y & 0 \\ r_z & u_z & v_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\text{matrica prijelaza}} \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -t_x & -t_y & -t_z & 1 \end{pmatrix}}_{\text{translacija}} = \begin{pmatrix} r_x & u_x & v_x & 0 \\ r_y & u_y & v_y & 0 \\ r_z & u_z & v_z & 0 \\ -\mathbf{t}\cdot\mathbf{r} & -\mathbf{t}\cdot\mathbf{r} & -\mathbf{t}\cdot\mathbf{r} & 1 \end{pmatrix}$$

View Matrix

```
33  float4x4 CreateViewMatrix(float4& eye, float4& target, float4& up) {
34        float4 v = target - eye;
35        v = Normalize(v);
36        float4 r = Cross(up, v);
37        r = Normalize(r);
38        float4 u = Cross(v, r);
39        return float4x4{
40              r.x, u.x, v.x, 0,
41              r.y, u.y, v.y, 0,
42              r.z, u.z, v.z, 0,
43              -Dot(eye, r), -Dot(eye,u), -Dot(eye,v), 1
44        };
45  }
```

Matrica projekcije se sastavlja pomoću:

- n - near plane
- f - far plane
- r - aspect ratio
- α - FOV

$$P = \begin{pmatrix} \frac{1}{r\tan(\frac{\alpha}{2})} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan(\frac{\alpha}{2})} & 0 & 0 \\ 0 & 0 & \frac{f}{f-n} & 0 \\ 0 & 0 & -\frac{nf}{f-n} & 0 \end{pmatrix}$$

Projection Matrix

```
47  float4x4 CreateProjectionMatrix(float angle, float ratio, float n, float f)
        {
48      return float4x4{
49              1 / (ratio * tan(angle / 2)), 0, 0, 0,
50              0, 1 / (tan(angle / 2)), 0, 0,
51              0, 0, f / (f - n), 1,
52              0, 0, -(n * f) / (f - n), 0
53      };
54  }
```

3.3 Screen Mapping

Normaliziran prostor NDC se projicira na zaslon, to uključuje sljedeće parametre:

- w - visina zaslona
- h - širina zaslona
- x_t - gornje-lijeva x kooridnata zaslona
- y_t - gornje-lijeva y kooridnata zaslona
- d_m - minimalna z vrijednost NDC mapiranja
- d_M - maximalna z vrijednost NDC mapiranja

$$S = \begin{pmatrix} \frac{w}{2} & 0 & 0 & 0 \\ 0 & -\frac{h}{2} & 0 & 0 \\ 0 & 0 & d_M - d_m & 0 \\ x_t + \frac{w}{2} & y_t + \frac{h}{2} & d_m & 1 \end{pmatrix}$$

Viewport Matrix

```
56  float4x4 CreateViewportMatrix(int x, int y, float width, float height, float
        zmin, float zmax) {
57      return float4x4{
58              width / 2, 0, 0, 0,
59              0, -height / 2, 0, 0,
60              0, 0, zmax - zmin, 0,
61              x + width / 2, y + height / 2, zmin, 1
62      };
63  }
```

Terminal

```
Model:

Pozicija: [-1, 1, 0, 1] Boja: [255, 0, 0, 0]
Pozicija: [1, 1, 0, 1] Boja: [0, 255, 0, 0]
Pozicija: [1, -1, 0, 1] Boja: [0, 0, 255, 0]
Pozicija: [-1, -1, 0, 1] Boja: [125, 125, 125, 0]

Scale Matrix:
10 0 0 0
0 10 0 0
0 0 10 0
0 0 0 1

RotateZ Matrix:
0.707107 0.707107 0 0
-0.707107 0.707107 0 0
0 0 1 0
0 0 0 1

Translate Matrix:
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1

World Transform Matrix:
7.07107 7.07107 0 0
-7.07107 7.07107 0 0
0 0 10 0
0 0 0 1

World Object:
Pozicija: [-14.1421, 0, 0, 1] Boja: [255, 0, 0, 0]
Pozicija: [0, 14.1421, 0, 1] Boja: [0, 255, 0, 0]
Pozicija: [14.1421, 0, 0, 1] Boja: [0, 0, 255, 0]
Pozicija: [0, -14.1421, 0, 1] Boja: [125, 125, 125, 0]

View Transform Matrix:
1 0 0 0
0 1 0 0
0 0 1 0
-0 -0 5 1

View Object:
Pozicija: [-14.1421, 0, 5, 1] Boja: [255, 0, 0, 0]
Pozicija: [0, 14.1421, 5, 1] Boja: [0, 255, 0, 0]
Pozicija: [14.1421, 0, 5, 1] Boja: [0, 0, 255, 0]
Pozicija: [0, -14.1421, 5, 1] Boja: [125, 125, 125, 0]

Projection Matrix:
1.08253 0 0 0
0 1.73205 0 0
0 0 1.001 1
0 0 -0.1001 0

Clip Object:
```

```
Pozicija: [-15.3093, 0, 4.9049, 5] Boja: [255, 0, 0, 0]
Pozicija: [0, 24.4949, 4.9049, 5] Boja: [0, 255, 0, 0]
Pozicija: [15.3093, 0, 4.9049, 5] Boja: [0, 0, 255, 0]
Pozicija: [0, -24.4949, 4.9049, 5] Boja: [125, 125, 125, 0]

Normalized Object:
Pozicija: [-3.06186, 0, 0.980981, 1] Boja: [255, 0, 0, 0]
Pozicija: [0, 4.89898, 0.980981, 1] Boja: [0, 255, 0, 0]
Pozicija: [3.06186, 0, 0.980981, 1] Boja: [0, 0, 255, 0]
Pozicija: [0, -4.89898, 0.980981, 1] Boja: [125, 125, 125, 0]

Viewport Matrix:
20 0 0 0
0 -12.5 0 0
0 0 1 0
20 12.5 0 1

Screen Object:
Pozicija: [-41.2373, 12.5, 0.980981, 1] Boja: [255, 0, 0, 0]
Pozicija: [20, -48.7372, 0.980981, 1] Boja: [0, 255, 0, 0]
Pozicija: [81.2373, 12.5, 0.980981, 1] Boja: [0, 0, 255, 0]
Pozicija: [20, 73.7372, 0.980981, 1] Boja: [125, 125, 125, 0]
```