3D Моделирање

OpenGL – 3D. Пројекција. Камера.

Да би се могло цртати у тродимензионалном окружењу, а потом исцртано и исправно представити на екрану, потребно је поставити камеру, дефинисати пројекцију и подесити *viewport*. Функције којима се врше ова почетна подешавања су:

- `gluLookAt(double eyeX, double eyeY, double eyeZ, double centerX, double centerY, double centerZ, double upX, double upY, double upZ)` – Функција се користи за подешавање камере. Прва три аргумента дефинишу координате положаја камере. Друга три дефинишу координате тачке ка којој је усмерена камера, док последња три аргумента дефинишу вектор „на горе“.
- `gluPerspective(double favy, double aspect, double zNear, double zFar)` – Дефинише се начин приказивања модела, односно начин пресликавања координата модела у нормализоване координате на екрану. Први аргумент представља угао видљивости у степенима у правцу y осе. Други параметар би требало да се поклопи са односом висине и ширине екрана који се користи при подешавању *viewport* -а. Трећи и четврти аргумент представљају удаљеност камере од најближе и најдаље равни приказа. Треба нагласити да су трећи и четврти аргумент увек позитивне вредности.
- `glViewPort(int x, int y, int width, int height)` – Користи се за постављање оквира за приказ. Прва два аргумента представљају координате доњег левог угла у пикселима, а трећи и четврти аргумент представљају ширину и висину оквира за приказ, такође у пикселима. Тиме се одређује афина трансформација нормализованих координата у координате прозора. Ако су (x_{nd}, y_{nd}) нормализоване координате, тада су координате прозора једнаке:

$$x_w = (x_{nd} + 1)\left(\frac{width}{2}\right) + x$$
$$y_w = (y_{nd} + 1)\left(\frac{height}{2}\right) + y$$

```
void initGL()
{
   glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Set background
color to black and opaque
   glShadeModel(GL_FLAT);

   glEnable(GL_DEPTH_TEST); // Enable depth testing for z-
culling

}
void reshape(GLsizei width, GLsizei height)
{
   if (height == 0) height = 1;
   GLfloat aspect = (GLfloat)width / (GLfloat)height;
```

```
    // Set the viewport to cover the new window
    glViewport(0, 0, width, height);

    // Set the aspect ratio of the clipping volume to match the
viewport
    glMatrixMode(GL_PROJECTION);  // To operate on the
Projection matrix
    glLoadIdentity();             // Reset
    gluPerspective(80.0f, aspect, 0.1f, 50.0f);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(camera.m_x,camera.m_y,camera.m_z,
              0.0,0.0,0.0,
              0.0,1.0,0.0);
}
```

Матрице трансформације

Класа *Matrix4x4* осим метода за рад са матрицама садржи и следеће методе:

- `void LoadTranslate(double dx, double dy, double dz)` – На основу задатих параметара поставља вредности матрице тако да се добије матрица транслације за вектор $[dx\ dy\ dz]^T$.
- `void LoadScale(double sx, double sy, double sz)` – На основу задатих параметара поставља вредности матрице тако да се добије матрица скалирања и то по x оси sx пута, а по y оси sy пута и по *z* оси *sz* пута.
- `void LoadRotateZ(double alpha)` – На основу задатог угла поставља вредности матрице тако да се добије матрица ротације за угао $alpha$ око осе која пролази кроз координатни почетак, а нормална је на раван *xOy*.
- `void LoadRotateX(double alpha)` – На основу задатог угла поставља вредности матрице тако да се добије матрица ротације за угао $alpha$ око осе која пролази кроз координатни почетак, а нормална је на раван *yOz*.
- `void LoadRotateY(double alpha)` – На основу задатог угла поставља вредности матрице тако да се добије матрица ротације за угао $alpha$ око осе која пролази кроз координатни почетак, а нормална је на раван *xOz*.

Задаци

Креирате следеће функције:

1. `void CreateCube(vector< vector<Vector3D> > &cube)` – Креира потребне чворове за цртање коцке чија је страница дужине 2 тако да је координатни почетак уједно и средиште коцке.
2. `void DrawCube(vector< vector<Vector3D> > &cube)` – За задате податке о чворовима коцке црта шест полигона који представљају странице те коцке.
3. `void CreatePyramid(vector<Vector3D> &pyramid)` – Креира чворове потребне за дефинисање праве четворостране пирамиде са следећим карактеристикама: дужина странице квадрата у основи једнака је 1; висина пирамиде је такође једнака 1. Пирамида треба да се нађе у првом октанту координатног система и то тако да један чвор основе пирамиде буде у координатном почетку.
4. `void DrawPyramid(vector<Vector3D> &pyramid)` – За дате податке о чворовима четворостране пирамиде врши њено исцртавање.
5. `void CreateCone(vector<Vector3D> &cone, int n)` – Креира чворове потребне за дефинисање праве купе полупречника основе 1 и висине 1. Центар круга у основи купе треба да буде у координатном почетку и да основа купе лежи у равни xOz, а да се врх купе нађе на позитивном делу y осе. Аргумент n представља поделу кругова у основи купе.
6. `void DrawCone(vector<Vector3D> &cone)` – За дате податке о чворовима купе, врши њено исцртавање.
7. Нацртати правоугаоник за који су задати центар, вектор нормале и дужине страница, при чему се зна да су странице дужине *a* паралелне са *xOy* равни.

Решење

Направљена је прво помоћна функција која за задати вектор тачака црта полигон.

```
void DrawPolygon(vector<Vector3D> &polygon)
{
    glBegin(GL_POLYGON);
        for(unsigned int i = 0; i < polygon.size(); i++)
            glVertex3d(polygon[i].m_x, polygon[i].m_y, polygon[i].m_z);
    glEnd();
}
```

1. Дефинисано је осам чворова коцке тако да се коцка простире по све три осе од -1 до 1.

```
void CreateCube(vector< vector<Vector3D> > &cube)
{
    cube.resize(2);
    cube[0].resize(4);
    cube[1].resize(4);

    cube[0][0] = Vector3D(-1.0, -1.0, -1.0);
    cube[0][1] = Vector3D(-1.0, -1.0,  1.0);
    cube[0][2] = Vector3D( 1.0, -1.0,  1.0);
    cube[0][3] = Vector3D( 1.0, -1.0, -1.0);
    cube[1][0] = Vector3D(-1.0,  1.0, -1.0);
    cube[1][1] = Vector3D(-1.0,  1.0,  1.0);
    cube[1][2] = Vector3D( 1.0,  1.0,  1.0);
    cube[1][3] = Vector3D( 1.0,  1.0, -1.0);
}
```

2. Црта се шест полигона тако да сваки представља једну страну коцке.

```
void DrawCube(vector< vector<Vector3D> > &cube)
{
    vector<Vector3D> poly;
    poly.resize(4);

    poly[0] = cube[0][0];
    poly[1] = cube[0][1];
    poly[2] = cube[0][2];
    poly[3] = cube[0][3];
    DrawPolygon(poly);

    poly[0] = cube[1][0];
    poly[1] = cube[1][1];
    poly[2] = cube[1][2];
    poly[3] = cube[1][3];
    DrawPolygon(poly);

    for(unsigned int i = 0; i < cube[0].size()-1; i++)
    {
        poly[0] = cube[0][i];
        poly[1] = cube[0][i+1];
        poly[2] = cube[1][i+1];
```

```
        poly[3] = cube[1][i];
        DrawPolygon(poly);
    }
}
```

3. Основа праве четворостране пирамиде се налази у равни xOz и то тако да се један чвор основе налази у координатном почетку, а да цела пирамида лежи у првом октанту. Дужина основе једнака је 1 као и висина.

```
void CreatePyramid(vector<Vector3D> &pyramid)
{
    pyramid.resize(5);

    pyramid[0] = Vector3D(0.0, 0.0, 0.0);
    pyramid[1] = Vector3D(0.0, 0.0, 1.0);
    pyramid[2] = Vector3D(1.0, 0.0, 1.0);
    pyramid[3] = Vector3D(1.0, 0.0, 0.0);
    pyramid[4] = Vector3D(0.5, 1.0, 0.5);
}
```

4. Прво се црта основа пирамиде. У вектору $\overrightarrow{piramid}$ налази се пет тачака, а прве четири одређују основу пирамиде, док је пета тачка врх пирамиде. Осова као и преостале четири стране пирамиде се цртају као полигони и то користећи помоћну функцију за цртање полигона.

```
void DrawPyramid(vector<Vector3D>  &pyramid)
{
    vector<Vector3D> poly;
    poly.resize(4);

    poly[0] = pyramid[0];
    poly[1] = pyramid[1];
    poly[2] = pyramid[2];
    poly[3] = pyramid[3];
    DrawPolygon(poly);

    poly.resize(3);

    poly[0] = pyramid[0];
    poly[1] = pyramid[1];
    poly[2] = pyramid[4];
    DrawPolygon(poly);

    poly[0] = pyramid[1];
    poly[1] = pyramid[2];
    poly[2] = pyramid[4];
    DrawPolygon(poly);

    poly[0] = pyramid[2];
    poly[1] = pyramid[3];
    poly[2] = pyramid[4];
    DrawPolygon(poly);
```

```
    poly[0] = pyramid[3];
    poly[1] = pyramid[0];
    poly[2] = pyramid[4];
    DrawPolygon(poly);
}
```

5. Купа

Чворови основе се креирају као и код кружнице. Тако ће првих n тачака вектора $\overrightarrow{cone}$ бити чворови основе, а последњи чвор ће представљати врх купе. Центар основе је у координатном почетку. Основа је полупречника 1 и налази се у равни xOz, док се врх купе налази на y оси и то на висини 1.

```
void CreateCone(vector<Vector3D> &cone, int n)
{
    cone.resize(n+1);
    Matrix4x4 MRotate;
    MRotate.loadRotateY(2.0*M_PI/n);

    cone [0] = Vector3D(1.0, 0.0, 0.0);
    for(unsigned int i = 1; i < cone.size()-1; i++)
        cone [i] = MRotate.Transform(cone [i-1]);

    cone [n] = Vector3D(0.0, 1.0, 0.0);
}
```

6. Исцртавање купе

Основа купе се црта као полигон без последње тачке вектора $\overrightarrow{cone}$. Омотач купе се може нацртати од троуглова састављених од сваке две узастопне тачке на основи заједно са врхом купе. Скуп таквих троуглова може се нацртати модом $GL\_TRIANGLE\_FAN$.

```
void DrawCone (vector<Vector3D> &cone)
{
    vector<Vector3D> poly = cone;
    poly.resize(cone.size()-1);
    DrawPolygon(poly);

    int n = cone.size();
    glBegin(GL_TRIANGLE_FAN);
        glVertex3d(cone [n-1].m_x, cone [n-1].m_y, cone [n-1].m_z);
        for(int i = 0; i < n-1; i++)
            glVertex3d(cone [i].m_x, cone [i].m_y, cone [i].m_z);
    glEnd();

    poly.resize(3);
    poly[0] = cone [n-1];
    poly[1] = cone [n-2];
    poly[2] = cone [0];

    DrawPolygon(poly);
}
```

7.Нацртати правоугаоник за који су задати центар, вектор нормале и дужине страница, при чему се зна да су странице дужине *a* паралелне са *xOy* равни.

Нека су дати вектор нормале $\vec{N}$ и вектор центра правоугаоника $\overrightarrow{center}$, као и дужине страница *a* и *b*. Да бисмо нацртали задати правоугаоник, потребно је да одредимо векторе страница $\vec{a}$ и $\vec{b}$, након чега ћемо тачке правоугаоника добити на следећи начин:

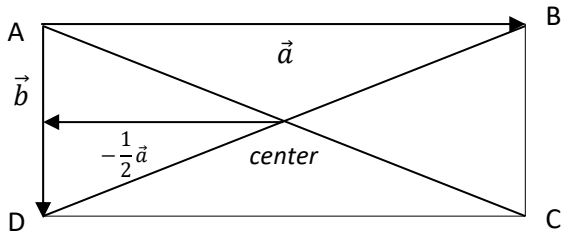
$$\vec{A} = \overrightarrow{center} - \frac{1}{2} \cdot \vec{a} - \frac{1}{2} \cdot \vec{b}$$

$$\vec{B} = \vec{A} + \vec{a}$$

$$\vec{C} = \vec{B} + \vec{b}$$

$$\vec{D} = \vec{A} + \vec{b}$$

што се лако може уочити са слике 1.



Слика 1

Остало је још одредити векторе страница $\vec{a}$ и $\vec{b}$.

Познато је да вектор $\vec{a}$ припада равни *xOy*. Осим што је вектор нормале $\vec{N}$ нормалан на вектор $\vec{a}$, знамо и да је *z* оса нормална на вектор $\vec{a}$ (јер вектор $\vec{a}$ припада равни *xOy* а *z* оса је нормална на све векторе равни *xOy*), тако да вектор $\vec{a}$ можемо добити на следећи начин:

$$\vec{a} = a \cdot \frac{\vec{N} \times \vec{z}}{\left\|\vec{N} \times \vec{z}\right\|}$$

при чему је $\vec{z} = (0,0,1)$.

Вектор $\vec{b}$ је нормалан на векторе $\vec{a}$ и $\vec{N}$, тако да се добија као векторски производ претходна два вектора, који се након тога нормализује и множи дужином странице *b*.

$$\vec{b} = b \cdot \frac{\vec{a} \times \vec{N}}{\left\|\vec{a} \times \vec{N}\right\|}$$

Домаћи

Креирати следеће геометријске облике:

1. **Ваљак**
   a. `void CreateCylinder(vector< vector<Vector3D> > &cylinder, int n)` – Креира чворове потребне за дефинисање ваљка полупречника основе 1, висине 1 и то тако да једна основа ваљка буде у равни $y = 0$, а друга у равни $y = 1$, а да центри основа леже на y оси. Аргумент n представља поделу кругова у основи ваљка.
   b. `void DrawCylinder(vector< vector<Vector3D> > &cylinder)` – За дате податке о чворовима ваљка, врши се његово исцртавање.
2. **Четворострана права зарубљена пирамида**
   a. `void CreateTruncatedPyramid(vector<vector<Vector3D> &truncatedPyramid, int n)` – Креира чворове четворостране праве зарубљене пирамиде са следећим карактеристикама: једна основа се налази у равни $y = 0$, а друга у равни $y = 1$; квадрат у основи је странице 1 и један његов чвор се налази у координатном почетку, а страница друге основе треба да буде дужине 0.5. Цела пирамида треба да се нађе у првом октанту координатног система.
   b. `void DrawTruncatedPyramid(vector< vector<Vector3D> > &truncatedPyramid, int n)` – За дате податке о чворовима зарубљене пирамиде, врши њено исцртавање.
3. **Права зарубљена купа**
   a. `void CreateTruncatedCone(vector< vector<Vector3D> > &truncatedPyramid, int n)` - Креира чворове потребне за дефинисање праве зарубљене купе полупречника основа 1 и 0.5, висином 1 и то тако да једна основа ваљка буде у равни $y = 0$, а друга у равни $y = 1$. Центри основа треба да се нађу на y оси. Аргумент n представља поделу кругова у основи ваљка.
   b. `void DrawTruncatedCone(vector< vector<Vector3D> > &truncatedPyramid, int n)` – За дате податке о чворовима зарубљене купе, врши се њено исцртавање.