OpenGL:

Osvetjenje i tako to:

```
void SetLightingParams()

{

    GLfloat ambientModel[] = { 0.3f, 0.3f, 0.3f, 1.0f };


    // Подешавање амбијенталног модела осветљења

    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientModel);


    // Локални посматрач (рачунање углова према стварном положају камере)

    glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);


    // Двострано осветљење (осветљава и задњу страну полигона)

    glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);


    // Омогућавање система осветљења

    glEnable(GL_LIGHTING);

}

void SetLightParams(int glLight)

{

    GLfloat ambient[]  = { 0.0f, 0.0f, 0.0f, 1.0f };

    GLfloat diffuse[]  = { 1.0f, 1.0f, 1.0f, 1.0f };

    GLfloat specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };

    GLfloat position[] = { 0.0f, 0.0f, 0.0f, 1.0f }; // позиција у координатном почетку
    GLfloat direction[] = { 0.0f, 0.0f, -1.0f };     // усмерено ка -Z

    glLightfv(glLight, GL_AMBIENT, ambient);
    glLightfv(glLight, GL_DIFFUSE, diffuse);
    glLightfv(glLight, GL_SPECULAR, specular);
    glLightfv(glLight, GL_POSITION, position);

    // Spotlight параметри
    glLightfv(glLight, GL_SPOT_DIRECTION, direction);
    glLightf(glLight, GL_SPOT_CUTOFF, 45.0f); // купа од 45°

    // Мала атенуација (слабљење интензитета са растојањем)
    glLightf(glLight, GL_CONSTANT_ATTENUATION, 1.0f);
    glLightf(glLight, GL_LINEAR_ATTENUATION, 0.05f);
    glLightf(glLight, GL_QUADRATIC_ATTENUATION, 0.001f);

    glEnable(glLight);
}
void SetMaterial(float diffuse[])
{
    GLfloat ambient[]  = { diffuse[0] * 0.5f, diffuse[1] * 0.5f, diffuse[2] * 0.5f, 1.0f };
    GLfloat specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };
    GLfloat emission[] = { 0.0f, 0.0f, 0.0f, 1.0f };
    GLfloat shininess  = 64.0f;

    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, ambient);
    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, diffuse);
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, specular);
    glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, emission);
    glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, shininess);
}
void CYourView::OnDraw(CDC* pDC)
{
    // Подеси модел осветљења
    SetLightingParams();

    // Подеси једно светло (GL_LIGHT0)
    SetLightParams(GL_LIGHT0);

    // Подеси материјал, нпр. црвени објекат
    GLfloat redDiffuse[] = { 1.0f, 0.0f, 0.0f, 1.0f };
    SetMaterial(redDiffuse);

    // Сада исцртај објекат
    glutSolidTeapot(1.0); // пример
}
```

PrepareScene, DrawScene, Reshape, DestroyScene, LoadTexture, DrawAxes, DrawEnvCube - uvek isto:

```
void CGLRenderer::PrepareScene(CDC *pDC)
{
	wglMakeCurrent(pDC->m_hDC, m_hrc);
	//--------------------------------
	glClearColor(1.0, 1.0, 1.0, 1.0);
	glEnable(GL_DEPTH_TEST);
	env = LoadTexture("assets/env.png"); //ovo se menja u zavisnosti od naziva teksture koju
ucitavam
	brick = LoadTexture("assets/brick.png");
	//glEnable(GL_TEXTURE_2D);
	//--------------------------------
	wglMakeCurrent(NULL, NULL);
}

void CGLRenderer::DrawScene(CDC *pDC)
{
	wglMakeCurrent(pDC->m_hDC, m_hrc);
	//--------------------------------
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity();
	//Ovde idu funkcije za crtanje
	glFlush();
	SwapBuffers(pDC->m_hDC);
	//--------------------------------
	wglMakeCurrent(NULL, NULL);
}

void CGLRenderer::Reshape(CDC *pDC, int w, int h)
```

```
{
	wglMakeCurrent(pDC->m_hDC, m_hrc);
	//---------------------------------
	glViewport(0, 0, (GLsizei)w, (GLsizei)h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluPerspective(50, (double)w / (double)h, 0.1, 2000);
	glMatrixMode(GL_MODELVIEW);
	//---------------------------------
	wglMakeCurrent(NULL, NULL);
}

void CGLRenderer::DestroyScene(CDC *pDC)
{
	wglMakeCurrent(pDC->m_hDC, m_hrc);
	glDeleteTextures(1, &brick);
	glDeleteTextures(1, &env);
	wglMakeCurrent(NULL,NULL);
	if(m_hrc)
	{
		wglDeleteContext(m_hrc);
		m_hrc = NULL;
	}
}

UINT CGLRenderer::LoadTexture(char* fileName)
```

```
{
	UINT texID;
	DImage img;
	img.Load(CString(fileName));

	glPixelStorei(GL_UNPACK_ALIGNMENT, 4);
	glGenTextures(1, &texID);
	glBindTexture(GL_TEXTURE_2D, texID);

	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
	glTexParameteri(GL_PROXY_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR_MIPMAP_LINEAR);

	gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGBA, img.Width(), img.Height(), GL_RGBA,
GL_UNSIGNED_BYTE, img.GetDIBBits());
	return texID;
}

void CGLRenderer::DrawAxes()
{
	glLineWidth(2.0);
	glBegin(GL_LINES);

	glColor3d(0.0, 0.0, 1.0);
	glVertex3d(0.0, 0.0, 0.0);
	glVertex3d(50.0, 0.0, 0.0);

	glColor3d(1.0, 0.0, 0.0);
	glVertex3d(0.0, 0.0, 0.0);
	glVertex3d(0.0, 50.0, 0.0);

	glColor3d(0.0, 1.0, 0.0);
	glVertex3d(0.0, 0.0, 0.0);
	glVertex3d(0.0, 0.0, 50.0);

	glEnd();
}

void CGLRenderer::DrawEnvCube(double a)
{
	glEnable(GL_TEXTURE_2D);

	//prednja
	glBindTexture(GL_TEXTURE_2D, m_texEnv[0]);
	glBegin(GL_QUADS);
	glColor3d(1.0, 1.0, 1.0);

	glTexCoord2d(0, 1);
	glVertex3d(-a / 2, -a / 2, -a / 2);
	glTexCoord2d(0, 0);
	glVertex3d(-a / 2, a / 2, -a / 2);
```

```
glTexCoord2d(1, 0);

glVertex3d(a / 2, a / 2, -a / 2);

glTexCoord2d(1, 1);

glVertex3d(a / 2, -a / 2, -a / 2);


glEnd();


//leva

glBindTexture(GL_TEXTURE_2D, m_texEnv[2]);

glBegin(GL_QUADS);

glColor3d(1.0, 1.0, 1.0);


glTexCoord2d(0, 0);

glVertex3d(-a / 2, a / 2, a / 2);

glTexCoord2d(0, 1);

glVertex3d(-a / 2, -a / 2, a / 2);

glTexCoord2d(1, 1);

glVertex3d(-a / 2, -a / 2, -a / 2);

glTexCoord2d(1, 0);

glVertex3d(-a / 2, a / 2, -a / 2);


glEnd();


//donja

glBindTexture(GL_TEXTURE_2D, m_texEnv[5]);

glBegin(GL_QUADS);
```

```
glColor3d(1.0, 1.0, 1.0);


glTexCoord2d(0, 0);
glVertex3d(-a / 2, -a / 2, -a / 2);
glTexCoord2d(1, 0);
glVertex3d(a / 2, -a / 2, -a / 2);
glTexCoord2d(1, 1);
glVertex3d(a / 2, -a / 2, a / 2);
glTexCoord2d(0, 1);
glVertex3d(-a / 2, -a / 2, a / 2);


glEnd();


//zadnja
glBindTexture(GL_TEXTURE_2D, m_texEnv[1]);
glBegin(GL_QUADS);
glColor3d(1.0, 1.0, 1.0);


glTexCoord2d(0, 0);
glVertex3d(a / 2, a / 2, a / 2);
glTexCoord2d(0, 1);
glVertex3d(a / 2, -a / 2, a / 2);
glTexCoord2d(1, 1);
glVertex3d(-a / 2, -a / 2, a / 2);
glTexCoord2d(1, 0);
glVertex3d(-a / 2, a / 2, a / 2);
```

```
glEnd();


//desna

glBindTexture(GL_TEXTURE_2D, m_texEnv[3]);

glBegin(GL_QUADS);

glColor3d(1.0, 1.0, 1.0);


glTexCoord2d(0, 0);

glVertex3d(a / 2, a / 2, -a / 2);

glTexCoord2d(0, 1);

glVertex3d(a / 2, -a / 2, -a / 2);

glTexCoord2d(1, 1);

glVertex3d(a / 2, -a / 2, a / 2);

glTexCoord2d(1, 0);

glVertex3d(a / 2, a / 2, a / 2);


glEnd();


//gore

glBindTexture(GL_TEXTURE_2D, m_texEnv[4]);

glBegin(GL_QUADS);

glColor3d(1.0, 1.0, 1.0);


glTexCoord2d(0, 0);

glVertex3d(-a / 2, a / 2, a / 2);
```

```
        glTexCoord2d(0, 1);
        glVertex3d(-a / 2, a / 2, -a / 2);
        glTexCoord2d(1, 1);
        glVertex3d(a / 2, a / 2, -a / 2);
        glTexCoord2d(1, 0);
        glVertex3d(a / 2, a / 2, a / 2);

        glEnd();

        glDisable(GL_TEXTURE_2D);
}

//Piramida

void GLRenderer::DrawPyramid(double side, double height, int n)
{
  double angleStep = (2 * piconst) / n;
  double currAngle = 0;

  double halfAngle = angleStep / 2;
  double r = (side / 2) / tan(halfAngle);
  double L = sqrt(r * r + height * height);
  double R = sqrt(r * r + (side / 2) * (side / 2));
  float ny = r / L;
  float nr = height / L;
```

```
    glBegin(GL_TRIANGLES);
    {
        for (int i = 0; i < n; i++)
        {
            glNormal3f(nr * cos(currAngle + (angleStep / 2)), ny, -nr * sin(currAngle + (angleStep /
2)));

            float x1 = R * cos(currAngle);
            float x2 = R * cos(currAngle + angleStep);
            float z1 = -R * sin(currAngle);
            float z2 = -R * sin(currAngle + angleStep);

            glVertex3f(0, height, 0);
            glVertex3f(x1, 0, z1);
            glVertex3f(x2, 0, z2);

            currAngle += angleStep;
        }
    }
    glEnd();

    currAngle = 0;
    glBegin(GL_TRIANGLE_FAN);
    {
        glNormal3f(0.0, -1.0, 0.0);
        glVertex3f(0.0, 0.0, 0.0);
        for (int i = 0; i < n + 1; i++)
        {
            glVertex3f(R * cos(currAngle), 0.0, R * sin(currAngle));
            currAngle += angleStep;
        }
    }
    glEnd();
}

//Valjak:

void GLRenderer::DrawRoller(float r, float h, int n)
{
    glColor3f(1.0, 0.0, 0.0);
    float angleStep = (piconst * 2) / n;
    float texStep = 1.0 / (float)n;
    float currAngle = 0;
    float currTex = 0;

    glBegin(GL_QUAD_STRIP);
    {
        for (int i = 0; i < n + 1; i++)
        {
            float x = r * cos(currAngle);
            float z = -r * sin(currAngle);
        glNormal3f(x / r, 0, z / r);

        glTexCoord2f(currTex, 0);
        glVertex3f(x, h / 2, z);

        glTexCoord2f(currTex, 1);
        glVertex3f(x, -h / 2, z);

        currAngle += angleStep;
        currTex += texStep;
    }
}
glEnd();

glDisable(GL_TEXTURE_2D);

currAngle = 0;
glBegin(GL_TRIANGLE_FAN);
{
    glNormal3f(0.0, 1.0, 0.0);
    glVertex3f(0.0, h / 2, 0.0);

    for (int i = 0; i < n + 1; i++)
    {
        float x = r * cos(currAngle);
        float z = -r * sin(currAngle);
        glVertex3f(x, h / 2, z);

        currAngle += angleStep;
    }
}
glEnd();

currAngle = 0;
glBegin(GL_TRIANGLE_FAN);
{
    glNormal3f(0.0, -1.0, 0.0);
    glVertex3f(0.0, -h / 2, 0.0);

    for (int i = 0; i < n + 1; i++)
    {
        float x = r * cos(currAngle);
        float z = r * sin(currAngle);

        glVertex3f(x, -h / 2, z);

        currAngle += angleStep;
    }
}
glEnd();
```

```
   glEnable(GL_TEXTURE_2D);
}
```

Sfera bez teksture - uvek isto:

```
void CGLRenderer::DrawSphere(double r, int nSeg)
{
	double stepAlpha = 3.1415 / nSeg;
	double stepBeta = 2 * 3.1415 / nSeg;

	double alpha, beta;

	alpha = -3.1415 / 2;

	for (int i = 0; i < nSeg; i++)
	{
		beta = 0.0;
		glBegin(GL_QUAD_STRIP);

		for (int j = 0; j < nSeg + 1; j++)
		{
			double x1 = r * cos(alpha) * cos(beta);
			double y1 = r * sin(alpha);
			double z1 = r * cos(alpha) * sin(beta);

			glVertex3d(x1, y1, z1);
```

```
            double x2 = r * cos(alpha + stepAlpha) * cos(beta);
            double y2 = r * sin(alpha + stepAlpha);
            double z2 = r * cos(alpha + stepAlpha) * sin(beta);

            glVertex3d(x2, y2, z2);

            beta += stepBeta;
        }

        glEnd();
        alpha += stepAlpha;
    }
}
```

Cone (kupa) bez teksture - uvek isto:

```
void CGLRenderer::DrawCone(double r, double h, int nSeg)
{
    double stepAlpha = 2 * 3.1415 / nSeg;

    glBegin(GL_TRIANGLE_FAN);

    double alpha = 0.0;
    glVertex3d(0.0, h, 0.0);
```

```
        for (int i = 0; i < nSeg + 1; i++)

        {

                double x1 = r * cos(alpha);

                double z1 = r * sin(alpha);


                glVertex3d(x1, 0.0, z1);


                alpha += stepAlpha;

        }


        glEnd();

}
```

sfera sa teksturom nekom prelepljenom - pauk:

```
void CGLRenderer::DrawSphere(double r, int nSeg, double texU, double texV, double texR)

{

        double stepAlpha = 3.1415 / nSeg;

        double stepBeta = 2 * 3.1415 / nSeg;


        double alpha, beta;


        alpha = -3.1415 / 2;


        for (int i = 0; i < nSeg; i++)

        {
```

```
		beta = 0.0;

		glBegin(GL_QUAD_STRIP);


		for (int j = 0; j < nSeg + 1; j++)

		{

			double x1 = r * cos(alpha) * cos(beta);

			double y1 = r * sin(alpha);

			double z1 = r * cos(alpha) * sin(beta);


			double tx1 = x1 / r * texR + texU; //Ovo se racuna samo kada u tekstu
kaze da treba nekako da se nalepi tekstura na sferu

			double ty1 = z1 / r * texR + texV; //Ovo se racuna samo kada u tekstu kaze
da treba nekako da se nalepi tekstura na sferu


			glTexCoord2d(tx1, ty1);  //Ovo se racuna samo kada u tekstu kaze da
treba nekako da se nalepi tekstura na sferu

			glVertex3d(x1, y1, z1);


			double x2 = r * cos(alpha + stepAlpha) * cos(beta);

			double y2 = r * sin(alpha + stepAlpha);

			double z2 = r * cos(alpha + stepAlpha) * sin(beta);


			double tx2 = x2 / r * texR + texU;

			double ty2 = z2 / r * texR + texV;


			glTexCoord2d(tx2, ty2);

			glVertex3d(x2, y2, z2);
```

```
                beta += stepBeta;
            }

            glEnd();
            alpha += stepAlpha;
        }
}
```

Cone(kupa) sa teksturom preko - pauk:

```
void CGLRenderer::DrawCone(double r, double h, int nSeg, double texU, double texV, double
texR)
{
        double stepAlpha = 2 * 3.1415 / nSeg;

        glBegin(GL_TRIANGLE_FAN);

        double alpha = 0.0;
        glTexCoord2d(0.75, 0.75); //Ovo se menja u zavisnosti od teksture
        glVertex3d(0.0, h, 0.0);

        for (int i = 0; i < nSeg + 1; i++)
        {
            double x1 = r * cos(alpha);
```

```
                double z1 = r * sin(alpha);

                double tx1 = x1 / r * texR + texU;
                double ty1 = z1 / r * texR + texV;

                glTexCoord2d(tx1, ty1);
                glVertex3d(x1, 0.0, z1);

                alpha += stepAlpha;
        }

        glEnd();
}
```

trougao, bez teksture, za teksturu samo koristiti zakomentarisane linije:

```
void CGLRenderer::DrawTriangle(float d1, float d2, float rep)
{
        double a1 = atan2(d2, d1);
        double d3 = sqrt(d1 * d1 + d2 * d2);
        double y = d1 * cos(a1) / d3;
        double x = d1 * sin(a1) / d3;
        glBegin(GL_TRIANGLES);
        glColor3f(1.0, 1.0, 1.0);
        glNormal3f(0, 0, 1.0);
        //glTexCoord2f(0.5 * rep, 0.0);
```

```
	glVertex3f(0.0, 0.0, 0.0);

	//glTexCoord2f((0.5 + x) * rep, y * rep);

	glVertex3f(d1, 0.0, 0.0);

	//glTexCoord2f(0.5 * rep, rep);

	glVertex3f(d1, d2, 0.0);

	glEnd();

}


//Polygon


void CGLRenderer::DrawPolygon(POINTF* points, POINTF* texCoords, int n)

{

	//po preporuci

	glBegin(GL_TRIANGLE_FAN);

	for (int i = 0; i < n; i++)

	{

		glTexCoord2f(texCoords[i].x, texCoords[i].y); // tacka tesksture

		glVertex2f(points[i].x, points[i].y); // tacka temena

	}

	glEnd();

}




//Bager 2019
```

```
void CGLRenderer::DrawPolygon(POINTF* points, POINTF* texCoords, int n)

{

        //po preporuci

        glBegin(GL_TRIANGLE_FAN);

        for (int i = 0; i < n; i++)

        {

                glTexCoord2f(texCoords[i].x, texCoords[i].y); // tacka tesksture

                glVertex2f(points[i].x, points[i].y); // tacka temena

        }

        glEnd();

}


void CGLRenderer::DrawExtrudedPolygon(POINTF* points, POINTF* texCoords, int n, float zh,
float r, float g, float b)

{

        glPushMatrix();

        glTranslatef(0, 0, -zh / 2); // udaljeno kao

        DrawPolygon(points, texCoords, n); // sa slike 1, crta taj poligon

        glTranslatef(0, 0, zh); // nakon toga,pozicioniramo za drugu stranu

        DrawPolygon(points, texCoords, n); // nacrta drugu stranu

        glPopMatrix();

        //uzmemo boju kojom hocemo da crtamo

        glColor3f(r, g, b);

        glBegin(GL_QUAD_STRIP);

        // treba sada obe strane,leva i desna strana da se spoje

        for (int i = 0; i < n; i++)
        {
                glVertex3f(points[i].x, points[i].y, -zh/2);
                glVertex3f(points[i].x, points[i].y, zh/2);
        }

        glVertex3f(points[0].x, points[0].y, -zh / 2);
        glVertex3f(points[0].x, points[0].y, zh / 2);
        glEnd();
        glColor3f(1, 1, 1);
}
void CGLRenderer::DrawBase()
{
        POINTF vertex[8];
        vertex[0] = POINTF({0,0.5});
        vertex[1] = POINTF({0,1.5});
        vertex[2] = POINTF({0.5,2});
        vertex[3] = POINTF({7.5,2});
        vertex[4] = POINTF({8,1.5});
        vertex[5] = POINTF({8,0.5});
        vertex[6] = POINTF({7.5,0});
        vertex[7] = POINTF({0.5,0});
        //gledamo sa slike, teskture 16*16
        //0/16, i 15/16 - predstavlja kocku skroz dole levo,odatle ucitavamo teksturu za Base
        POINTF textures[8];
        //obavezno 16 sa tackom,inace nece da radi
        textures[0] = POINTF({ 0/16.,15 / 16. }); // skroz levo dole
```

```
	textures[1] = POINTF({ 0/16.,13 / 16. }); // 2 kocke na gore,sa leve strane dole
	textures[2] = POINTF({ 1 / 16.,12 / 16. }); // gornji levi deo gusenice
	textures[3] = POINTF({ 15 / 16.,12 / 16. }); // skroz desno,desni deo gornji gusenice
	textures[4] = POINTF({ 16 / 16.,13 / 16. }); // pa naredne dve,spusti do donje desne,
	textures[5] = POINTF({ 16 / 16.,15 / 16. });
	textures[6] = POINTF({ 15/ 16.,16 / 16. }); // donji deo desnog kraja gusenice,dole desno
skroz kockica
	// i poslednju spoji sa skroz levo,predzadnjom,dole
	textures[7] = POINTF({ 1 / 16,16 / 16 });

	DrawExtrudedPolygon(vertex, textures, 8, 5, 0, 0, 0); // visina prizme 5, 8- tacaka
}
void CGLRenderer::DrawBody()
{
	//podeok je 0.5
	POINTF vertex[5];
	//gledano sa slike 16x16.
	vertex[0] = POINTF({-2,0}); // leva donja
	vertex[1] = POINTF({-2,4}); // leva gornja
	vertex[2] = POINTF({0,4}); // desna donja
	vertex[3] = POINTF({2,2}); // desna gornja
	vertex[4] = POINTF({2,0}); //zadnja tacka

	POINTF textures[5];
	textures[0] = POINTF({8/16.,8/16.}); // donja leva
	//pa ide na gore
	textures[1] = POINTF({8/16.,0/16.}); // gornja leva
	textures[2] = POINTF({12/16.,0/16.}); // gore desno
	//kosa crta
	textures[3] = POINTF({16/16.,4/16.}); // desno
	textures[4] = POINTF({16/16.,8/16.}); // poslednja je dole desno skroz
	DrawExtrudedPolygon(vertex, textures, 5, 4, 0.96, 0.5, 0.12); // visina 4
}
void CGLRenderer::DrawArm(double zh)
{
	POINTF vertex[8];
	//pocinjemo od leve strane
	vertex[0] = POINTF({-1,-0.5});
	vertex[1] = POINTF({-1,0.5});
	vertex[2] = POINTF({-0.5,1});
	vertex[3] = POINTF({1,1});
	vertex[4] = POINTF({7,0.5});
	vertex[5] = POINTF({7,-0.5});
	vertex[6] = POINTF({1,-1});
	vertex[7] = POINTF({-0.5,-1});

	POINTF textures[8];
	textures[0] = POINTF({0/16.,11/16.}); // leva skroz,donja
	textures[1] = POINTF({0/16.,9/16.}); // pa gornja
	textures[2] = POINTF({1/16.,8/16.});
	textures[3] = POINTF({4/16.,8/16.});
	textures[4] = POINTF({16/16.,9/16.});//pa skroz desno
	textures[5] = POINTF({16/16.,11/16.});
	textures[6] = POINTF({4/16.,12/16.}); // vrati na levo dole
	textures[7] = POINTF({1/16.,12/16.}); // donja leva

	DrawExtrudedPolygon(vertex, textures, 8, zh, 0.96, 0.5, 0.12);
}
void CGLRenderer::DrawFork()
{
	POINTF vertex[6];
	vertex[0] = POINTF({-1,-1}); // donja leva tacka
	vertex[1] = POINTF({-1,1.5});
	vertex[2] = POINTF({-0.5,2});
	vertex[3] = POINTF({2.5,2});
	vertex[4] = POINTF({3,1.5});//desno
	vertex[5] = POINTF({3,-1});

	POINTF textures[6];
	textures[0] = POINTF({0/16.,6/16.}); // dole levo
	textures[1] = POINTF({0/16.,1/16.});
	textures[2] = POINTF({1/16.,1/16.});
	textures[3] = POINTF({7/16.,0/16.});
	textures[4] = POINTF({8/16.,1/16.});
	textures[5] = POINTF({8/16.,6/16.});

	DrawExtrudedPolygon(vertex, textures, 6, 1, 0.7, 0.7, 0.7);
}
```

```
void CGLRenderer::DrawExcavator()
{
	glBindTexture(GL_TEXTURE_2D, excavator);
	glPushMatrix();
	glTranslatef(-4, 0, 0);
	DrawBase();
	glPopMatrix();

	//primeni se transformacija za telo,odnosno rotacija
	glRotatef(angle1, 0, 1, 0);
	glPushMatrix();
	glTranslatef(0, 2, 0); // podignemo malo
	DrawBody();
	glPopMatrix();

	//duzi deo "ruke"
	glRotatef(90, 0, 0, 1);
	glTranslatef(3, -1, -1.5); // ka desno od kabine
	glRotatef(angle2, 0, 0, 1);
	DrawArm(1);

	//nastavlja da crta od centra prvog dela ruke
	glRotatef(-90, 0, 0, 1); // zarotiran za 90 stepeni,ali to i ne mora odmah
	glTranslatef(0, 6.5, 0);  // samo podigne po Y osi
	glRotatef(angle3, 0, 0, 1); // primeni transformaciju za slucaj rotiranja
	DrawArm(1.5); // nacrta ddrugi deo ruke
	//i na kraju viljuska
	glRotatef(-90, 0, 0, 1);
	glTranslatef(0, 6.5, 0);
	glRotatef(angle4, 0, 0, 1);
	DrawFork();
}
void CGLRenderer::setLight()
{
	GLfloat light1_ambient[] = { 0.5,0.5,0.5,1 };
	GLfloat light1_diffuse[] = { 1.0,1.0,1.0,1.0 };
	GLfloat light1_specular[] = { 1.0,1.0,1.0,1.0 };
	GLfloat light1_position[] = { -2.0,-2.0,1.0,1.0 };
	GLfloat light1_direction[] = { -1.0,-1.0,0.0 };

	glLightfv(GL_LIGHT1, GL_AMBIENT, light1_ambient);
	glLightfv(GL_LIGHT1, GL_DIFFUSE, light1_diffuse);
	glLightfv(GL_LIGHT1, GL_SPECULAR, light1_specular);
	glLightfv(GL_LIGHT1, GL_POSITION, light1_position);
	glEnable(GL_LIGHT1);
}

void CMy24012019View::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
	//pogled
	if (nChar == VK_RIGHT) m_glRenderer.ugaoX += 0.1;
	if (nChar == VK_LEFT) m_glRenderer.ugaoX -= 0.1;

	if (nChar == VK_UP) m_glRenderer.ugaoY += 0.1;

	if (nChar == VK_DOWN) m_glRenderer.ugaoY -= 0.1;

	if (nChar == VK_ADD) {

		m_glRenderer.r += 1;

	}

	if (nChar == VK_SUBTRACT) {

		m_glRenderer.r -= 1;

	}

	//nakon pomeranja ,izracuna se vrednost pozicije kamere odakle gledamo

	m_glRenderer.xEye = sin(m_glRenderer.ugaoY) * sin(m_glRenderer.ugaoX) *
m_glRenderer.r;

	m_glRenderer.yEye = cos(m_glRenderer.ugaoY) * m_glRenderer.r;

	m_glRenderer.zEye = sin(m_glRenderer.ugaoY) * cos(m_glRenderer.ugaoX) *
m_glRenderer.r;

	//rotacije za bager

	if (nChar == '1') m_glRenderer.angle1 += 5;

	if (nChar == '2') m_glRenderer.angle1 -= 5;

	if (nChar == '3') m_glRenderer.angle2 += 5;

	if (nChar == '4') m_glRenderer.angle2 -= 5;

	if (nChar == '5') m_glRenderer.angle3 += 5;

	if (nChar == '6') m_glRenderer.angle3 -= 5;

	if (nChar == '7') m_glRenderer.angle4 += 5;

	if (nChar == '8') m_glRenderer.angle4 -= 5;

	Invalidate();

	CView::OnKeyDown(nChar, nRepCnt, nFlags);

}
```

```
UINT B[6];
	UINT excavator;
	float angle1 = 0; // ugao kabine
	float angle2 = 0; // ugao prvog dela ruke
	float angle3 = 0; // ugao duzeg dela ruke
	float angle4 = 0; // ugao viljuske
	//pogled
	float xEye = 0;
	float yEye = 0;
	float zEye = 0;
	//ugao kojim se pomeramo oko bagera dok ga gledamo
	float ugaoX = 180;
	float ugaoY = 30;
	float r = 20;



//Rubikova kocka 2014

void CMy17012014View::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
	switch (nChar)
	{
	case 'E': m_glRenderer.ugaoPosmatraca -= 5.0f;
		break;
	case 'Q': m_glRenderer.ugaoPosmatraca += 5.0f;
		break;
	case VK_NUMPAD1: m_glRenderer.ugao[0] += 5.0f;
		break;
	case VK_NUMPAD2: m_glRenderer.ugao[1] += 5.0f;
		break;
	case VK_NUMPAD3: m_glRenderer.ugao[2] += 5.0f;
		break;
	}
	Invalidate();

	CView::OnKeyDown(nChar, nRepCnt, nFlags);
}

//modifikovano
void CGLRenderer::DrawCube(float a, int i, int j, int k, float texPom)
{
	glBegin(GL_QUADS);

	// donja
	SetMaterial(1.0, 1.0, 0.3);
	//ispod normale
	glNormal3f(0.0, -1.0, 0.0);
	//potrebno je izdvojiti deo teskture
	glTexCoord2f(1.0 - (k + 1) * texPom, 1.0 - i * texPom);
	glVertex3f(a / 2, -a / 2, a / 2);
```

```
glTexCoord2f(1.0 - k * texPom, 1.0 - i * texPom);

glVertex3f(-a / 2, -a / 2, a / 2);

glTexCoord2f(1.0 - k * texPom, 1.0 - (i + 1) * texPom);

glVertex3f(-a / 2, -a / 2, -a / 2);

glTexCoord2f(1.0 - (k + 1) * texPom, 1.0 - (i + 1) * texPom);

glVertex3f(a / 2, -a / 2, -a / 2);


// prednja,koju vidimo

SetMaterial(1.0, 0.2, 0.2);

glNormal3f(0.0, 0.0, 1.0); // na desno

glTexCoord2f((k + 1) * texPom, (j + 1) * texPom);

glVertex3f(a / 2, -a / 2, a / 2);

glTexCoord2f((k + 1) * texPom, j * texPom);

glVertex3f(a / 2, a / 2, a / 2);

glTexCoord2f(k * texPom, j * texPom);

glVertex3f(-a / 2, a / 2, a / 2);

glTexCoord2f(k * texPom, (j + 1) * texPom);

glVertex3f(-a / 2, -a / 2, a / 2);


// gornja

SetMaterial(1.0, 1.0, 1.0);

glNormal3f(0.0, 1.0, 0.0);

glTexCoord2f((k + 1) * texPom, 1.0 - i * texPom);

glVertex3f(a / 2, a / 2, a / 2);

glTexCoord2f((k + 1) * texPom, 1.0 - (i + 1) * texPom);

glVertex3f(a / 2, a / 2, -a / 2);
```

```
glTexCoord2f(k * texPom, 1.0 - (i + 1) * texPom);

glVertex3f(-a / 2, a / 2, -a / 2);

glTexCoord2f(k * texPom, 1.0 - i * texPom);

glVertex3f(-a / 2, a / 2, a / 2);


// zadnja

SetMaterial(1.0, 0.7, 0.2);

glNormal3f(0.0, 0.0, -1.0);

glTexCoord2f(1.0- (k + 1) * texPom, (j + 1) * texPom);

glVertex3f(a / 2, -a / 2, -a / 2);

glTexCoord2f(1.0 - k * texPom, (j + 1) * texPom);

glVertex3f(-a / 2, -a / 2, -a / 2);

glTexCoord2f(1.0 - k * texPom, j * texPom);

glVertex3f(-a / 2, a / 2, -a / 2);

glTexCoord2f(1.0 - (k + 1) * texPom, j * texPom);

glVertex3f(a / 2, a / 2, -a / 2);


// leva

SetMaterial(0.0, 0.7, 0.1);

glNormal3f(-1.0, 0.0, 0.0);

glTexCoord2f(1.0 - i * texPom, j * texPom);

glVertex3f(-a / 2, a / 2, a / 2);

glTexCoord2f(1.0 - (i + 1) * texPom, j * texPom);

glVertex3f(-a / 2, a / 2, -a / 2);

glTexCoord2f(1.0 - (i + 1) * texPom, (j + 1) * texPom);

glVertex3f(-a / 2, -a / 2, -a / 2);
```

```
        glTexCoord2f(1.0 - i * texPom, (j + 1) * texPom);

        glVertex3f(-a / 2, -a / 2, a / 2);


        // desna

        SetMaterial(0.4, 0.2, 1.0);

        glNormal3f(1.0, 0.0, 0.0);

        glTexCoord2f(i * texPom, j * texPom);

        glVertex3f(a / 2, a / 2, a / 2);

        glTexCoord2f(i * texPom, (j + 1) * texPom);

        glVertex3f(a / 2, -a / 2, a / 2);

        glTexCoord2f((i + 1) * texPom, (j + 1) * texPom);

        glVertex3f(a / 2, -a / 2, -a / 2);

        glTexCoord2f((i + 1) * texPom, j * texPom);

        glVertex3f(a / 2, a / 2, -a / 2);


        glEnd();

}

void CGLRenderer::DrawRubikCube(double a, int count)

{

        float polaStranica = count / 2.0;

        float prviPomeraj = polaStranica * a + floor(polaStranica) * a * 0.05 - a / 2;

        //da bi se obezb razmak

        float razmak = 1.05 * a;

        float pomerajTeksture = 1.0 / count;  // isto kao i razmak izmedju kocki,da se napravi
razmak

        //broj strana,na koliko se deli delova
```

```
	//Kockice crtamo od gore na dole.

	//OpenGL - BELE boje

	//prva kockica koja se crta je Donja Leva kockica BELE OpenGL

	//sa prvom For petljom i unutrasnjom K petljom,crtamo celu gornju stranu

	for (int i = 0; i < count; ++i)

	{

		glPushMatrix();

		glRotatef(ugao[i % count], 1.0, 0.0, 0.0); // sad rotiramo po x

		glTranslatef(-prviPomeraj, prviPomeraj, prviPomeraj);

		//crta na dole,ova for petlja

		for (int j = 0; j < count; ++j)

		{

			glPushMatrix();

			//jedna stranica, od donje leve do gornje desne

			//sa i = 0 , k od 0 do 2, nacrtamo leve 3 kocke gornje stranice

			for (int k = 0; k < count; ++k)

			{

				DrawCube(a, k, j, i, pomerajTeksture); // posto menjamo
translacije po x i z menjamo i koordinate

				glTranslatef(0.0, 0.0, -razmak); // ovo je bilo dole

			}

			glPopMatrix();

			//na dole,po Y osi

			glTranslatef(0.0, -razmak, 0.0);

		}
```

```
        glPopMatrix();
        //za sledecu kocku pomeri za razmak po X osi
        glTranslatef(razmak, 0.0, 0.0); // ovo je bilo gore
    }
}
void CGLRenderer::SetWhiteLight()
{
    glEnable(GL_LIGHTING);
    //globalno
    float global_ambient[] = { 0.5, 0.5, 0.5, 1.0};
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, global_ambient);
    glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);

    //tackasti izvor LIGHT0
    float ambient0[] = { 0.1, 0.1, 0.1, 1.0 };
    float diffuse0[] = { 0.8, 0.8, 0.8, 1.0 };
    float specular0[] = { 1.0, 1.0, 1.0, 1.0 };
    glLightfv(GL_LIGHT0, GL_AMBIENT, ambient0);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse0);
    glLightfv(GL_LIGHT0, GL_SPECULAR, specular0);
    glEnable(GL_LIGHT0);

    //Usmereni izvor LIGHT1
    float ambient1[] = { 0.4, 0.4, 0.4, 1.0 };
    float diffuse1[] = { 0.8, 0.8, 0.8, 1.0 };
    float specular1[] = { 1.0, 1.0, 1.0, 1.0 };
```

```
	glLightfv(GL_LIGHT1, GL_AMBIENT, ambient1);
	glLightfv(GL_LIGHT1, GL_DIFFUSE, diffuse1);
	glLightfv(GL_LIGHT1, GL_SPECULAR, specular1);

	//nakon toga,podesavamo poziciju usmerenog izvora svetlosti
	float light1_position[] = { 0.0, 0.0, 0.0 };
	glLightfv(GL_LIGHT1, GL_POSITION, light1_position);

	float light1_direction[] = { 0.0, 0.0, 1.0 };
	glLightfv(GL_LIGHT1, GL_SPOT_DIRECTION, light1_direction);
	glLightf(GL_LIGHT1, GL_SPOT_CUTOFF, 13.0);
	glLightf(GL_LIGHT1, GL_SPOT_EXPONENT, 2.0);
	glLightf(GL_LIGHT1, GL_CONSTANT_ATTENUATION, 1.0);
	glLightf(GL_LIGHT1, GL_LINEAR_ATTENUATION, 0.01);
	glLightf(GL_LIGHT1, GL_QUADRATIC_ATTENUATION, 0.001);
	glEnable(GL_LIGHT1);
}
void CGLRenderer::SetMaterial(float r, float g, float b)
{
	//ambijentalna boja 20% difuzione broje:
	float ambient[] = { 0.2f * r, 0.2f * g, 0.2f * b, 1.0f };
	float diffuse[] = { r, g, b, 1.0f };
	float specular[] = { r, g, b, 1.0f };
	//nema emisionu komponentu
	float shininess = 64.0f;
	glMaterialfv(GL_FRONT, GL_AMBIENT, ambient);
	glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuse);
	glMaterialfv(GL_FRONT, GL_SPECULAR, specular);
	glMaterialf(GL_FRONT, GL_SHININESS, shininess);
}
void CGLRenderer::DrawScene(CDC *pDC)
{
	wglMakeCurrent(pDC->m_hDC, m_hrc);
	//---------------------------------
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity();
	//gluLookAt(20, 20, 20, 0, 0, 0, 0, 1, 0);
	//rotiramo pre crtanja,zahtevamo u 1. zadatku
	glTranslatef(0.0, 0.0, -20);
	glRotatef(45.0, 1.0, 0.0, 0.0);

	float light0_position[] = { 5.0, 20.0, 0.0, 1.0 };
	glLightfv(GL_LIGHT0, GL_POSITION, light0_position);

	//koristimo transformaciju za rotiranje pogleda na scenu po Y osi
	glRotatef(ugaoPosmatraca, 0.0, 1.0, 0.0);
	//nezavisno od preth. transformacija,crtamo kocku
	glPushMatrix();
	DrawRubikCube(2.0, 3);
	glPopMatrix();
	glFlush();

	SwapBuffers(pDC->m_hDC);

	//---------------------------------

	wglMakeCurrent(NULL, NULL);

}



//Lampa 2012


void CGLRenderer::DrawScene(CDC *pDC)

{

	wglMakeCurrent(pDC->m_hDC, m_hrc);

	//---------------------------------

	glClear(GL_DEPTH_BUFFER_BIT | GL_COLOR_BUFFER_BIT);

	glLoadIdentity();

	//postavljamo posmatraca

	gluLookAt(3, 2, 3, 0, 1, 0, 0, 1, 0); // y =1 , gleda u gornju povrsinu


	glRotated(rotation, 0.0, 1.0, 0.0);


	//Postavljamo poziciono svetlo

	//ostale vr. su podrazmevane

	float light_position[] = { 6,4,6,1.0 };

	glLightfv(GL_LIGHT0, GL_POSITION, light_position);

	glEnable(GL_LIGHT0);
	glEnable(GL_LIGHTING);
	//nakon podesavanja svetlosti,moze da se pocne crtanje

	Kocka(1.0);
	glTranslated(0.0, 0.5, 0.0);
	Telo(0.2, 0.1, 0.05, 1.0, 16);

	glTranslated(0.0, 1.0, 0.0);

	glBindTexture(GL_TEXTURE_2D, fabric);
	glEnable(GL_TEXTURE_2D);
	Abazur(0.5, 0.25, 0.2, 10);

	glFlush();
	SwapBuffers(pDC->m_hDC);
	//---------------------------------
	wglMakeCurrent(NULL, NULL);
}

void CGLRenderer::Abazur(double r1, double r2, double h, int step)
{
	//broj koraka aproksimacije,za 1
	double alfaStep = (2 * pi) / step;
	double alfa = 0;
	double texture = 0;
	double textStep = 1.0 / step; // pomeranje za teksturu
	//kao valjak ,samo bez gornjeg i donjeg poklopca
	glBegin(GL_QUAD_STRIP);
	for (int i = 0; i <= step; i++)
	{
		//cvor gore
		glNormal3f(cos(alfa),0,sin(alfa)); // normala
		glTexCoord2f(texture, 1.0);
		glVertex3f(r2 * cos(alfa), h, r2 * sin(alfa));
		//cvor dole
		glNormal3f(cos(alfa), 0, sin(alfa)); // normala
		glTexCoord2f(texture, 0.0);
		glVertex3f(r1 * cos(alfa), 0.0, r1 * sin(alfa));

		alfa += alfaStep;
		texture += textStep;
	}
	glEnd();
}
void CGLRenderer::Telo(double r1, double r2, double r3, double h, int step)
{
	glBindTexture(GL_TEXTURE_2D, metal1);
	glEnable(GL_TEXTURE_2D);
	double hStep = h / 4;
	glPushMatrix();
	//jednostavnije nacrtati iz 4 dela
	//donji deo
	Abazur(r1, r2, hStep, step);

	glTranslated(0.0, hStep, 0.0);

	//pa na taj dodje jos jedan,koji je jos uzi

	Abazur(r2, r3, hStep, step);

	glTranslated(0.0, hStep, 0.0);

	//najuzi deo

	Abazur(r3, r3, hStep, step);

	glTranslated(0.0, hStep, 0.0);

	//i na kraju gornji deo

	Abazur(r3, r2, hStep, step);

	glPopMatrix();

}




//Ajfelov toranj 2013


void CGLRenderer::DrawScene(CDC *pDC)

{

	wglMakeCurrent(pDC->m_hDC, m_hrc);

	//---------------------------------

	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

	glLoadIdentity();

	gluLookAt(40, 25, 30, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);


	float light_position[] = { 100.0, 70.0, 90.0, 1.0 };
	float spot_direction[] = { -1.0, -1.0, -1.0 };

	glLightfv(GL_LIGHT0, GL_POSITION, light_position);

	glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spot_direction);

	//---------------------------------

	DrawAxis(100);

	glPushMatrix();

	//malo pomerimo ka nama

	glTranslatef(0,0,10);

	glEnable(GL_TEXTURE_2D);


	//Donji deo tornja

	FillVA(buff1, 4.8, 8, 3.0); // b = 2.4 , h = 1.5 , a = 4

	FillVATex(bufft1, 0.215, 0); // x1  = 0.215 , x2 =0

	//ucitamo prvu teksturu

	glBindTexture(GL_TEXTURE_2D,T[0]);

	//DrawPyramid(buff,bufft);

	glTranslatef(10, 0.0, 0);

	glRotatef(rotTower, 0.0, 1.0, 0.0);

	DrawPyramid(4.8, 8, 3.0, bufft1);


	//terasica donja

	FillVATex(bufft4, 0.0, 0.0);

	glBindTexture(GL_TEXTURE_2D, T[3]);

	glTranslatef(0, 1.5, 0);

	//sada treba naopako da se nacrta,zato je prvo 5.28 pa 4.8 , prva vrednost ako je
veca,onda je donji deo gore
```

```
	DrawPyramid(5.28, 4.8, 0.5, bufft4);


	//dalje crtamo 2 deo ,isto kao prvi,samo manji za pola

	FillVATex(bufft2, 0.215,0);  // x1  = 0.215 , x2 =0

	glBindTexture(GL_TEXTURE_2D, T[1]);

	glTranslatef(0, 1.5, 0);

	DrawPyramid(2.4, 4.8, 3, bufft2);


	//gornja terasica

	FillVATex(bufft4, 0.0, 0.0);

	glBindTexture(GL_TEXTURE_2D, T[3]);

	glTranslatef(0, 1.5, 0);

	//sada treba naopako da se nacrta,zato je prvo 5.28 pa 4.8 , prva vrednost ako je
veca,onda je donji deo gore

	DrawPyramid(2.69, 2.4, 0.5, bufft4);


	// i poslednji deo,onaj duzi

	FillVATex(bufft3, 0.45, 0.05);  // x1  = 0.215 , x2 =0

	glBindTexture(GL_TEXTURE_2D, T[2]);

	glTranslatef(0, 8, 0);

	DrawPyramid(0, 2.16, 16, bufft3); // visina veca


	glDisable(GL_TEXTURE_2D);

	glPopMatrix();

	glFlush();

	//---------------------------------
```

```
        SwapBuffers(pDC->m_hDC);

        wglMakeCurrent(NULL, NULL);

}


//Parametar x1 defini e s teksturnu koordinatu gornjeg levog temena zarubljene piramide, a
parametar x2 donjeg levog temena

void CGLRenderer::FillVATex(float* buff, float x1, float x2)

{

        int count = 0;

        for (int i = 0; i < 4; i++)

        {

                buff[count++] = 0;

                buff[count++] = x1;

                buff[count++] = 1;

                buff[count++] = x2;

                buff[count++] = 1;

                buff[count++] = 1.0 - x2;

                buff[count++] = 0;

                buff[count++] = 1.0 - x1;

        }

}

void CGLRenderer::FillVA(float* buff, float a, float b, float h)

{

        //ovu funkciju pisemo na osnovu DrawPyramid.


        //Podesavamo koordinate za teksture
```

```
float x = a / 2.0;

float y = h / 2.0;

float z = b / 2.0;

int count = 0;

//donja stranica x,delimo sa 2 , jer crtamo iz centra


//prednja

//prva tacka za Texture

buff[count++] = -x;

buff[count++] = y;

//a trecu koristimo za Vertex

buff[count++] = x;


buff[count++] = -z;

buff[count++] = -y;

buff[count++] = z;

buff[count++] = z;

buff[count++] = -y;

buff[count++] = z;

buff[count++] = x;

buff[count++] = y;

buff[count++] = x;


//desna

buff[count++] = x;

buff[count++] = y;
```

```
buff[count++] = x;
buff[count++] = z;
buff[count++] = -y;
buff[count++] = z;
buff[count++] = z;
buff[count++] = -y;
buff[count++] = -z;
buff[count++] = x;
buff[count++] = y;
buff[count++] = -x;

//zadnja
buff[count++] = x;
buff[count++] = y;
buff[count++] = -x;
buff[count++] = z;
buff[count++] = -y;
buff[count++] = -z;
buff[count++] = -z;
buff[count++] = -y;
buff[count++] = -z;
buff[count++] = -x;
buff[count++] = y;
buff[count++] = -x;

//leva
```

```
	buff[count++] = -x;
	buff[count++] = y;
	buff[count++] = -x;
	buff[count++] = -z;
	buff[count++] = -y;
	buff[count++] = -z;
	buff[count++] = -z;
	buff[count++] = -y;
	buff[count++] = z;
	buff[count++] = -x;
	buff[count++] = y;
	buff[count++] = x;
}
void CGLRenderer::DrawPyramid(double a, double b, double h, float* bufft)
{
	double x = a / 2.0;
	double y = h / 2.0; // visina
	double z = b / 2.0;
	int countt = 0;
	int count = 0;
	SetMaterial();

	glBegin(GL_QUADS);
	glNormal3f(0.0, 0.0, 1.0);// sa prednje strane crtamo
	glTexCoord2f(bufft[countt++], bufft[countt++]); // donja
	glVertex3d(-x, y, x);
```

```
glTexCoord2f(bufft[countt++], bufft[countt++]); // gornja
glVertex3d(-z, -y, z);
glTexCoord2f(bufft[countt++], bufft[countt++]); // gornja
glVertex3d(z, -y, z);
glTexCoord2f(bufft[countt++], bufft[countt++]); // donja
glVertex3d(x, y, x);

//desna strana, po x , ide na desno dakle
glNormal3f(1.0, 0.0, 0.0);
glTexCoord2f(bufft[countt++], bufft[countt++]);
glVertex3d(x, y, x);
glTexCoord2f(bufft[countt++], bufft[countt++]);
glVertex3d(z, -y, z);
glTexCoord2f(bufft[countt++], bufft[countt++]);
glVertex3d(z, -y, -z);
glTexCoord2f(bufft[countt++], bufft[countt++]);
glVertex3d(x, y, -x);

//zadnja
glNormal3f(0.0, 0.0, -1.0); // -z osa
glTexCoord2f(bufft[countt++], bufft[countt++]);
glVertex3d(x, y, -x);
glTexCoord2f(bufft[countt++], bufft[countt++]);
glVertex3d(z, -y, -z);
glTexCoord2f(bufft[countt++], bufft[countt++]);
glVertex3d(-z, -y, -z);
```

```
	glTexCoord2f(bufft[countt++], bufft[countt++]);
	glVertex3d(-x, y, -x);

	//leva
	glNormal3f(-1.0, 0.0, 0.0); // -X osa
	glTexCoord2f(bufft[countt++], bufft[countt++]);
	glVertex3d(-x, y, -x);
	glTexCoord2f(bufft[countt++], bufft[countt++]);
	glVertex3d(-z, -y, -z);
	glTexCoord2f(bufft[countt++], bufft[countt++]);
	glVertex3d(-z, -y, z);
	glTexCoord2f(bufft[countt++], bufft[countt++]);
	glVertex3d(-x, y, x);
	glEnd();
}

//podesimo materijal i svetlost
void CGLRenderer::SetMaterial()
{
	GLfloat mat_ambient[] = { 0.75, 0.75, 0.75, 1.0 };
	GLfloat mat_diffuse[] = { 0.9, 0.9, 0.7, 1.0 };
	GLfloat mat_specular[] = { 0.1, 0.1, 0.1, 1.0 };
	GLfloat mat_emission[] = { 0.0, 0.0, 0.0, 0.0 };
	GLfloat mat_shininess = 0.3; // malo smo mu dali odsjaj

	glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
	glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
	glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
	glMaterialfv(GL_FRONT, GL_EMISSION, mat_emission);
	glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess);
}
void CGLRenderer::SetLightModel()
{
	glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_FALSE);
	glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);

	float light_ambient[] = { 0.5, 0.5, 0.5, 1.0 };
	float light_diffuse[] = { 0.2, 0.2, 0.2, 1.0 };
	float light_specular[] = { 0.8, 0.7, 0.7, 1.0 };

	glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
	glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
	glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);

	glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 1.0);
	glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);
	glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, 2.0);

	glEnable(GL_LIGHTING);
	glEnable(GL_LIGHT0);
}
```

```
//Zemlja 2015


void CGLRenderer::DrawScene(CDC *pDC)
{
	wglMakeCurrent(pDC->m_hDC, m_hrc);
	//---------------------------------
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity();
	glDisable(GL_DEPTH_TEST);
	glDisable(GL_LIGHTING);

	glPushMatrix();
	glRotated(rot2, 1, 0, 0);
	glRotated(rot1, 0, 1, 0);
	DrawSpace(75, 20);
	glPopMatrix();
	// u centru nacrta zemlju,da bi
	glEnable(GL_DEPTH_TEST); //da ne bi bile prozirne
	// 8. deo zad. Direkcioni izvor svetlosti
	//amb, dif, spe za light0
	if (lighting)
	{
		glEnable(GL_LIGHT0);
```

```
            glEnable(GL_LIGHTING);
        }
        else
        {
            glDisable(GL_LIGHTING);
            glDisable(GL_LIGHT0);
        }


        GLfloat amb[] = { 1, 1, 1, 1.0 };
        GLfloat dif[] = { 1, 1, 1, 1 };
        GLfloat spe[] = { 1, 1, 1, 1 };
        GLfloat pos[] = { 0,0,1,0 }; // u pravcu pozitibne Z-ose
        glLightfv(GL_LIGHT0, GL_POSITION, pos);
        glLightfv(GL_LIGHT0, GL_DIFFUSE, dif);
        glLightfv(GL_LIGHT0, GL_AMBIENT, amb);
        glLightfv(GL_LIGHT0, GL_SPECULAR, spe);




        glTranslated(0, 0, -dist); // udaljimo zemlju po Z osi,da bude manja,sa Q je
    priblizimo,odnosno sa E udaljimo
        glRotated(rot1+180, 0, 1, 0); // gore i dole rotacija zemlje
        glRotated(rot2+180 , 1, 0, 0); // na levo i desno rotacija


        // +180 stepeni stalvjamo jer crta zemlju naopako,pa da bi izgledala ok
```

```
	DrawEarth(3, 20);


	glTranslated(-15, 0, 0);
	glRotated(rotMesecX, 1, 0, 0);
	glRotated(rotMesecY,0,1,0);
	DrawMoon(1, 20);
	//---------------------------------


	//---------------------------------
	glFlush();
	SwapBuffers(pDC->m_hDC);
	wglMakeCurrent(NULL, NULL);
}

void CGLRenderer::DrawPatch(double R, int n)
{
	double x = -1;
	double y = -1;
	double step = (double)2.0 / n;
	//za svaki podeok:
	for (int i = 0; i < n; i++)
	{
		glBegin(GL_QUAD_STRIP);
		for (int j =0 ; j <= n; j++)
		{
			//prelazak u polarne,formula sa blanketa
```

```
        double fi = atan(x);

        double teta = atan(y*cos(fi));


        //prelazak u dekartove:

        double x1 = cos(teta)*sin(fi);

        double z1 = cos(teta) * cos(fi);

        double y1 = sin(teta);

        //generise teksturne koordinate i normale

        glNormal3f(x1,y1,z1);

        glTexCoord2d((double)j/n,(double)i/n);

        glVertex3d(R*x1,R*y1,R*z1);


        //fi se ne menja

        teta = atan((y+step)*cos(fi));

        x1 = cos(teta) * sin(fi);

        z1 = cos(teta) * cos(fi);

        y1 = sin(teta);

        glNormal3d(x1, y1, z1);

        glTexCoord2d((double)(j) / n, (double)(i + 1) / n);

        glVertex3d(R * x1, R * y1, R * z1);


        x += step;

}

x = -1.0;

y += step;

glEnd();
```

```
	}
}
void CGLRenderer::DrawMoon(double R, int tes)
{
	glPushMatrix();
	glBindTexture(GL_TEXTURE_2D, M[0]);
	DrawPatch(R, tes);
	glRotated(90, 0, 1, 0);
	glBindTexture(GL_TEXTURE_2D, M[1]);
	DrawPatch(R, tes);
	glRotated(90, 0, 1, 0);
	glBindTexture(GL_TEXTURE_2D, M[2]);
	DrawPatch(R, tes);
	glRotated(90, 0, 1, 0);
	glBindTexture(GL_TEXTURE_2D, M[3]);
	DrawPatch(R, tes);
	glPopMatrix();

	glPushMatrix();
	glRotated(90, 1, 0, 0);
	glBindTexture(GL_TEXTURE_2D, M[4]);
	DrawPatch(R, tes);
	glRotated(180, 1, 0, 0);
	glBindTexture(GL_TEXTURE_2D, M[5]);
	DrawPatch(R, tes);
	glPopMatrix();
```

```
}
void CGLRenderer::DrawSpace(double R, int tes)
{
	glPushMatrix();
	glBindTexture(GL_TEXTURE_2D, S[0]);
	DrawPatch(R, tes);
	glRotated(90, 0, 1, 0);
	glBindTexture(GL_TEXTURE_2D, S[1]);
	DrawPatch(R, tes);
	glRotated(90, 0, 1, 0);
	glBindTexture(GL_TEXTURE_2D, S[2]);
	DrawPatch(R, tes);
	glRotated(90, 0, 1, 0);
	glBindTexture(GL_TEXTURE_2D, S[3]);
	DrawPatch(R, tes);
	glPopMatrix();

	glPushMatrix();
	glRotated(90, 1, 0, 0);
	glBindTexture(GL_TEXTURE_2D, S[4]);
	DrawPatch(R, tes);
	glRotated(180, 1, 0, 0);
	glBindTexture(GL_TEXTURE_2D, S[5]);
	DrawPatch(R, tes);
	glPopMatrix();
```

```
}

void CGLRenderer::DrawEarth(double R, int tes)

{

        // prvo matricu pushamo

        // pratimo redosled kako je zadatao,prvo prve 4 na desno

        glPushMatrix();

        glBindTexture(GL_TEXTURE_2D,T[0]);

        DrawPatch(R,tes);

        glRotated(90,0,1,0);

        glBindTexture(GL_TEXTURE_2D, T[1]);

        DrawPatch(R, tes);

        glRotated(90, 0, 1, 0);

        glBindTexture(GL_TEXTURE_2D, T[2]);

        DrawPatch(R, tes);

        glRotated(90, 0, 1, 0);

        glBindTexture(GL_TEXTURE_2D, T[3]);

        DrawPatch(R, tes);

        glPopMatrix();

        //za preostale 2 radimo sl:

        glPushMatrix();

        glRotated(90, 1, 0, 0);

        glBindTexture(GL_TEXTURE_2D, T[4]);

        DrawPatch(R, tes);

        glRotated(180, 1, 0, 0);

        glBindTexture(GL_TEXTURE_2D, T[5]);

        DrawPatch(R, tes);
        glPopMatrix();
}



//2011 nzm sta je ovo, ali ima prizma i jos neke stvari

void CGLRenderer::DrawScene(CDC *pDC)
{
        wglMakeCurrent(pDC->m_hDC, m_hrc);
        //--------------------------------
        glClear(GL_DEPTH_BUFFER_BIT | GL_COLOR_BUFFER_BIT);
        glLoadIdentity();
        //tackasti izvor,iza posmatraca
        float light_position[] = { 0.5,0.5,1.0, 1.0 };
        if(light)
                glEnable(GL_LIGHT0);
        glLightfv(GL_LIGHT0, GL_POSITION, light_position);
        glEnable(GL_LIGHTING);

        gluLookAt(0, 0, 10, 0, 0, 0, 0, 1, 0);
        //rotiramo pogled
        glRotated(viewAngle, 0, 1.0, 0);
        glRotated(viewAngleY, 1.0, 0, 0);

        glColor3f(1.0, 1.0, 1.0);
        DrawCompoundBody();
```

```
	glEnable(GL_TEXTURE_2D);
	glBindTexture(GL_TEXTURE_2D, texture);
	DrawWings();


	glFlush();
	SwapBuffers(pDC->m_hDC);
	//---------------------------------
	wglMakeCurrent(NULL, NULL);
}
#pragma region Brod


//n-to strana prizma
void CGLRenderer::DrawBody(double r, double h1, double h2, int n)
{
	double piramidHeight = (h1 - h2)/2;
	//telo
	DrawCilindar(r, h2, n);
	//posle toga crtamo piramide na vrhu i dnu
	glPushMatrix();
	glTranslated(0.0, h2, 0.0);
	//na vrhu
	DrawCone(r, piramidHeight, n);
	glPopMatrix();

	glPushMatrix();
	glRotated(180.0, 1.0, 0.0, 0.0);
```

```
        //na dnu
        DrawCone(r, piramidHeight, n);
        glPopMatrix();
}
void CGLRenderer::DrawCilindar(double r, double h, int step)
{
        double alfa = 0.0;
        double alfaStep = (2*pi) / step; // kao i prethodni
        double texture = 0.0;
        double textStep = 1.0 / step;

        double textStartX = 0.25;
        double textStartY = 0;
        double textEndX = 0.5;
        double textEndY = -0.5;
        double textStepX = (textEndX - textStartX) / step;
        double textStepY = (textEndY - textStartY) / step;

        glBegin(GL_QUAD_STRIP);
        for (int i = 0; i <= step; i++)
        {
                //gornji cvor
                glNormal3f(cos(alfa), 0, sin(alfa)); // normala
                glTexCoord2f(textStartX, texture);
                glVertex3f(r * cos(alfa), h, r * sin(alfa));
                //cvor dole
                glNormal3f(cos(alfa), 0, sin(alfa)); // normala
                glTexCoord2f(textEndX, texture);
                glVertex3f(r * cos(alfa), 0.0, r * sin(alfa));

                alfa += alfaStep;
                texture += textStartY;
        }
        glEnd();
}
void CGLRenderer::DrawCone(double r, double h, int step)
{
        double alfa = 0.0;
        double alfaStep = (2 * pi) / step;
        double texture = 0.0;
        double textStep = 1.0 / step;

        double textureStartX = 0.125;
        double textureStartY = -0.25;
        //vrh
        glBegin(GL_TRIANGLE_FAN);
        glNormal3f(0.0, 1.0, 0.0);
        glTexCoord2f(textureStartX, textureStartY);
        glVertex3f(0.0, h, 0.0);

        for (int i = 0; i <= step; i++)
        {
                glNormal3f(cos(alfa), 0.0, sin(alfa));

                glTexCoord2f(textureStartX + 0.125 * cos(alfa), textureStartY + 0.25 * sin(alfa));

                glVertex3f(r * cos(alfa), 0.0, r * sin(alfa));


                alfa += alfaStep;

                texture += textStep;

        }

        glEnd();

}

void CGLRenderer::DrawCompoundBody()

{

        //najveci deo

        glEnable(GL_TEXTURE_2D);

        glPushMatrix();

        glRotated(90.0, 1.0, 0.0, 0.0);

        glTranslated(0.0, -1.0, 0.0);

        DrawBody(1.0, 2.5, 2.0, 8);

        glPopMatrix();


        //2 sa strane

        glDisable(GL_TEXTURE_2D);

        glPushMatrix();

        glRotated(-90.0, 0.0, 0.0, 1.0);

        glTranslated(0.0, -1.5, 0.0);

        DrawBody(0.5, 3.5, 3.0, 8);

        glPopMatrix();
        glPushMatrix();
        glRotated(-90.0, 0.0, 0.0, 1.0);
        glTranslated(0.0, -2.5, 0.0);
        DrawBody(0.25, 5.0, 5.0, 8);
        glPopMatrix();

        glEnable(GL_TEXTURE_2D);
}
void CGLRenderer::DrawWing()
{
        glPushMatrix();
        glTranslated(0, 0.75, 0);
        glRotated(-20.0, 1.0, 0, 0);
        glTranslated(0, -0.75, 0);

        glBegin(GL_TRIANGLES);
        glTexCoord2d(0, -0.5);
        glVertex3d(-4, 0.75, 0);
        glTexCoord2d(0.5, -0.5);
        glVertex3d(0, 0.75, 0);
        glTexCoord2d(0.5, -1.0);
        glVertex3d(0, 2.75, 0);
        glEnd();

        glBegin(GL_QUADS);
```

```
glTexCoord2d(0.5, -0.5);

glVertex3d(0, 0.75, 0);

glTexCoord2d(0.75, -0.5);

glVertex3d(2, 0.75, 0);

glTexCoord2d(0.75, -1.0);

glVertex3d(2, 2.75, 0);

glTexCoord2d(0.5, -1.0);

glVertex3d(0, 2.75, 0);

glEnd();


glPopMatrix();


glBegin(GL_QUADS);

glTexCoord2d(0.75, -0.5);

glVertex3d(2, 0.75, 0);

glTexCoord2d(0.5, -0.5);

glVertex3d(0, 0.75, 0);

glTexCoord2d(0.5, 0);

glVertex3d(0, -0.75, 0);

glTexCoord2d(0.75, 0);

glVertex3d(2, -0.75, 0);

glEnd();


glPushMatrix();

glScaled(1.0, -1.0, 1.0);

glTranslated(0, 0.75, 0);
```

```
	glRotated(-20.0, 1.0, 0, 0);
	glTranslated(0, -0.75, 0);

	glBegin(GL_TRIANGLES);
	glTexCoord2d(0, -0.5);
	glVertex3d(-4, 0.75, 0);
	glTexCoord2d(0.5, -0.5);
	glVertex3d(0, 0.75, 0);
	glTexCoord2d(0.5, -1.0);
	glVertex3d(0, 2.75, 0);
	glEnd();

	glBegin(GL_QUADS);
	glTexCoord2d(0.5, -0.5);
	glVertex3d(0, 0.75, 0);
	glTexCoord2d(0.75, -0.5);
	glVertex3d(2, 0.75, 0);
	glTexCoord2d(0.75, -1.0);
	glVertex3d(2, 2.75, 0);
	glTexCoord2d(0.5, -1.0);
	glVertex3d(0, 2.75, 0);
	glEnd();

	glPopMatrix();
}
void CGLRenderer::DrawWings()
```

```
{
        glPushMatrix();
        glTranslated(2.6, 0.0, 0.0);
        glRotated(90.0, 0.0, 1.0, 0.0);
        glTranslated(-1.0, 0, 0);
        DrawWing();
        glPopMatrix();

        glPushMatrix();
        glRotated(180, 0, 0, 1.0);
        glTranslated(2.6, 0.0, 0.0);
        glRotated(90.0, 0.0, 1.0, 0.0);
        glTranslated(-1.0, 0, 0);
        DrawWing();
        glPopMatrix();
}

//April 2017

void GLRenderer::DrawRoller(float r, float h, int n)
{
  glColor3f(1.0, 0.0, 0.0);
  float angleStep = (piconst * 2) / n;
  float texStep = 1.0 / (float)n;
  float currAngle = 0;
  float currTex = 0;
```

```
glBegin(GL_QUAD_STRIP);
{
   for (int i = 0; i < n + 1; i++)
   {
      float x = r * cos(currAngle);
      float z = -r * sin(currAngle);

      glNormal3f(x / r, 0, z / r);

      glTexCoord2f(currTex, 0);
      glVertex3f(x, h / 2, z);

      glTexCoord2f(currTex, 1);
      glVertex3f(x, -h / 2, z);

      currAngle += angleStep;
      currTex += texStep;
   }
}
glEnd();

glDisable(GL_TEXTURE_2D);

currAngle = 0;
glBegin(GL_TRIANGLE_FAN);
```

```
{
    glNormal3f(0.0, 1.0, 0.0);
    glVertex3f(0.0, h / 2, 0.0);

    for (int i = 0; i < n + 1; i++)
    {
        float x = r * cos(currAngle);
        float z = -r * sin(currAngle);

        glVertex3f(x, h / 2, z);

        currAngle += angleStep;
    }
}
glEnd();

currAngle = 0;
glBegin(GL_TRIANGLE_FAN);
{
    glNormal3f(0.0, -1.0, 0.0);
    glVertex3f(0.0, -h / 2, 0.0);

    for (int i = 0; i < n + 1; i++)
    {
        float x = r * cos(currAngle);
        float z = r * sin(currAngle);
```

```
            glVertex3f(x, -h / 2, z);

            currAngle += angleStep;
        }
    }
    glEnd();

    glEnable(GL_TEXTURE_2D);
}


void GLRenderer::DrawFigure(float h, float dx, float alpha, float beta, float dt, CString arTex[])
{
    int n = 20;

    glPushMatrix();
    glRotatef(dt * (alpha * (180 / piconst)), 0.0, 1.0, 0.0);
    DrawRoller(dx, 0.2 * h, n);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(0.0, -0.2 * h, 0.0);
    glRotatef(dt * (alpha * (180 / piconst)), 0.0, 1.0, 0.0);
    DrawRoller(10 * dx, 0.2 * h, n);
    glPopMatrix();
```

```
  for (int i = 0; i < 4; i++)
  {
    glPushMatrix();
    glTranslatef(0.0, -0.1 * h - 0.2 * h - h / 2, 0.0);
    glRotatef(dt * (alpha * (180 / piconst)) + i * 90, 0.0, 1.0, 0.0);
    glTranslatef(3 * dx, 0.0, 0.0);
    glRotatef(dt * (beta * (180 / piconst)), 0.0, 1.0, 0.0);

    DrawRoller(dx, h, n);

    glTranslatef(0.0, -h / 2 - 0.1 * h, 0.0);
    DrawRoller(2 * dx, 0.2 * h, n);
    glPopMatrix();
  }

}


//Decembar 2018 hram

void GLRenderer::DrawScene(CDC* pDC)
{
  wglMakeCurrent(pDC->m_hDC, m_hrc);
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glLoadIdentity();
```

```
    SetLight();


    glTranslatef(0.0, 0.0, -40.0);
    glRotatef(xRot, 1.0, 0.0, 0.0);
    glRotatef(yRot, 0.0, 1.0, 0.0);


    //DrawBox(3, 4, 5);
    //DrawCylinder(0.5, 8, 20);
    //DrawColumn(0.5, 8, 20);
    DrawTemple();


    glFlush();
    SwapBuffers(pDC->m_hDC);
    wglMakeCurrent(NULL, NULL);
}

void GLRenderer::DestroyScene(CDC* pDC)
{
    wglMakeCurrent(pDC->m_hDC, m_hrc);


    wglMakeCurrent(NULL, NULL);
    if (m_hrc)
    {
        wglDeleteContext(m_hrc);
        m_hrc = NULL;
    }
}

void GLRenderer::RotateScene(int x, int y)
{
    xRot += x;
    if (xRot < 0)
        xRot += 360;
    else if (xRot > 360)
        xRot -= 360;

    yRot += y;
    if (yRot < 0)
        yRot += 360;
    else if (yRot > 360)
        yRot -= 360;
}


void GLRenderer::SetLight()
{
    float ambient[] = { 0.2, 0.2, 0.2, 1.0 };
    float diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
    float specular[] = { 1.0, 1.0, 1.0, 1.0 };
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    float position[] = { 0.0, 0.0, 0.0, 1.0 }; //postavlja svetlo na poziciji posmatraca, tako da uvek
prati poziciju posmatraca
    glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, specular);
    glLightfv(GL_LIGHT0, GL_POSITION, position);
}

void GLRenderer::DrawBox(double a, double b, double c)
{

    glBegin(GL_QUADS);
    {
        glNormal3f(0.0, 0.0, 1.0);
        glVertex3f(-a / 2, b / 2, c / 2);
        glVertex3f(-a / 2, -b / 2, c / 2);
        glVertex3f(a / 2, -b / 2, c / 2);
        glVertex3f(a / 2, b / 2, c / 2);

        glNormal3f(1.0, 0.0, 0.0);
        glVertex3f(a / 2, b / 2, c / 2);
        glVertex3f(a / 2, -b / 2, c / 2);
        glVertex3f(a / 2, -b / 2, -c / 2);
        glVertex3f(a / 2, b / 2, -c / 2);
}


        glNormal3f(0.0, 0.0, -1.0);
        glVertex3f(a / 2, b / 2, -c / 2);
        glVertex3f(a / 2, -b / 2, -c / 2);
        glVertex3f(-a / 2, -b / 2, -c / 2);
        glVertex3f(-a / 2, b / 2, -c / 2);

        glNormal3f(-1.0, 0.0, 0.0);
        glVertex3f(-a / 2, b / 2, -c / 2);
        glVertex3f(-a / 2, -b / 2, -c / 2);
        glVertex3f(-a / 2, -b / 2, c / 2);
        glVertex3f(-a / 2, b / 2, c / 2);

        glNormal3f(0.0, 1.0, 0.0);
        glVertex3f(a / 2, b / 2, c / 2);
        glVertex3f(a / 2, b / 2, -c / 2);
        glVertex3f(-a / 2, b / 2, -c / 2);
        glVertex3f(-a / 2, b / 2, c / 2);

        glNormal3f(0.0, -1.0, 0.0);
        glVertex3f(-a / 2, -b / 2, c / 2);
        glVertex3f(-a / 2, -b / 2, -c / 2);
        glVertex3f(a / 2, -b / 2, -c / 2);
        glVertex3f(a / 2, -b / 2, c / 2);
    }
    glEnd();
}
```

```
void GLRenderer::DrawCylinder(double r, double h, int steps)
{
   double angStep = (2 * piconst) / (double)steps;
   double currAng = 0;
   glBegin(GL_QUAD_STRIP);
   {
      for (int i = 0; i < steps + 1; i++)
      {
         double x = r * cos(currAng);
         double z = -r * sin(currAng);
         double nx = x / r;
         double nz = z / r;
         glNormal3f(nx, 0.0, nz);
         glVertex3f(x, h / 2.0, z);
         glVertex3f(x, -h / 2.0, z);

         currAng += angStep;
      }
   }
   glEnd();
}

void GLRenderer::DrawColumn(double r, double h, int steps)
{
   DrawCylinder(r, h, steps);
```

```
    glPushMatrix();
    glTranslatef(0.0, h / 2.0 + 0.25 * r, 0.0);
    DrawBox(2.5 * r, 0.5 * r, 2.5 * r);
    glPopMatrix();
    glPushMatrix();
    glTranslatef(0.0, -h / 2.0 - 0.15 * r, 0.0);
    DrawBox(3 * r, 0.3 * r, 3 * r);
    glTranslatef(0.0, -0.3 * r, 0.0);
    DrawBox(4 * r, 0.3 * r, 4 * r);
    glPopMatrix();
}

void GLRenderer::DrawTemple()
{
    glPushMatrix();
    glTranslatef(-10.5, 0.0, 4.5);
    for (int i = 0; i < 2; i++)
    {
        glPushMatrix();
        glTranslatef(0.0, 0.0, -i * 9.0);
        for (int j = 0; j < 8; j++)
        {
            glPushMatrix();
            glTranslatef(j * 3, 0.0, 0.0);
            DrawColumn(0.5, 8, 20);
            glPopMatrix();
        }
        glPopMatrix();
    }
    glPushMatrix();
    glTranslatef(0.0, 0.0, -3.0);
    DrawColumn(0.5, 8, 20);
    glTranslatef(0.0, 0.0, -3.0);
    DrawColumn(0.5, 8, 20);
    glPopMatrix();
    glPushMatrix();
    glTranslatef(7 * 3, 0.0, -3.0);
    DrawColumn(0.5, 8, 20);
    glTranslatef(0.0, 0.0, -3.0);
    DrawColumn(0.5, 8, 20);
    glPopMatrix();
    glPopMatrix();

}




//Jan 2017 torus


void GLRenderer::DrawScene(CDC* pDC)
{
    wglMakeCurrent(pDC->m_hDC, m_hrc);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glLoadIdentity();

    glEnable(GL_LIGHTING);

    glEnable(GL_LIGHT0);

    //glEnable(GL_TEXTURE_2D);

    double l = 0;


    float ambient[] = { 0.0, 0.0, 0.0, 1.0 };

    float diffuse[] = { 0.7, 0.7, 0.0, 1.0 }; //pise da se nalazi usmereni izvor svetlosti, zute boje

    float specular[] = { 1.0, 1.0, 0.6, 1.0 }; // sto 0.6, tj. sto uopste ove vrednosti?

    GLfloat position[] = { 5.0, 5.0, 10.0, 1.0 };  // usmeren je ka tacki koja je 10 jedinica ispred
postamtraca??? sta znaci to?

    float direction[] = { -5.0, -5.0, -10.0 }; // ovo il ono dole sto kaze ugao kupe koji definise
osvetljeni prostor je 20 stepeni

                                          // pogled posmatraca je usmeren u centru torusa koji se otkotrljao
duz x ose za 20 jedinica ???????/

    glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);

    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);

    glLightfv(GL_LIGHT0, GL_SPECULAR, specular);


    glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, direction);


    glLightfv(GL_LIGHT0, GL_POSITION, position);

    glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 20.0);

    glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, 2.0);
    double theta = atan(l / 10);   // cemu sluzi ovo??

    glRotatef(theta * (180 / piconst), 0.0, 1.0, 0.0);  // i bez to daje istu sliku

    glTranslatef(0.0, 0.0, -10.0);
    glRotatef(rotX, 1.0, 0.0, 0.0);
    glRotatef(rotY, 0.0, 1.0, 0.0);

    SetThorusMat();
    /*int t = LoadTexture(".\\Resources\\brick.png");
    glBindTexture(GL_TEXTURE_2D, t);*/
    RotateThorus(l);

    glFlush();
    SwapBuffers(pDC->m_hDC);
    wglMakeCurrent(NULL, NULL);
}

void GLRenderer::DrawThorus(double R, double r)
{
    int n = 50;   // sta je n??
    double currB = 0;
    double angleStep = (2 * piconst) / n;

    double* vertices = new double[(n + 1) * (n + 1) * 3]; // sto ima ovoliko ovih tacaka i kod
normala i kod tex?
```

```
double* normals = new double[(n + 1) * (n + 1) * 3];

double* textures = new double[(n + 1) * (n + 1) * 2];

UINT* indices = new UINT[n * n * 4];


double textureStep = 1.0 / (float)n;


for (int i = 0; i < n + 1; i++)

{

    double currA = 0;

    for (int j = 0; j < n + 1; j++)

    {

        double nextA = currA + angleStep;

        double nextB = currB + angleStep;

        int baseInd = i * (n + 1) * 3 + j * 3;


        vertices[baseInd] = (R + r * cos(currA)) * cos(currB);

        vertices[baseInd + 1] = (R + r * cos(currA)) * sin(currB);

        vertices[baseInd + 2] = -r * sin(currA);


        normals[baseInd] = cos(currA) * cos(currB);

        normals[baseInd + 1] = cos(currA) * sin(currB);

        normals[baseInd + 2] = -sin(currA);


        int baseTexInd = i * (n + 1) * 2 + j * 2;

        textures[baseTexInd] = i * textureStep;

        textures[baseTexInd + 1] = j * textureStep;
        currA += angleStep;
    }
    currB += angleStep;
}

for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        int baseInd = i * n * 4 + j * 4;  //??????
        indices[baseInd] = i * (n + 1) + j;
        indices[baseInd + 1] = indices[baseInd] + 1;
        indices[baseInd + 3] = (i + 1) * (n + 1) + j;
        indices[baseInd + 2] = indices[baseInd + 3] + 1;
        int a = 2;
    }
}

glEnableClientState(GL_VERTEX_ARRAY);
glEnableClientState(GL_NORMAL_ARRAY);
glEnableClientState(GL_TEXTURE_COORD_ARRAY);
glNormalPointer(GL_DOUBLE, 0, normals);
glTexCoordPointer(2, GL_DOUBLE, 0, textures);
glVertexPointer(3, GL_DOUBLE, 0, vertices);
glDrawElements(GL_QUADS, n * n * 4, GL_UNSIGNED_INT, indices);
```

```
    glDisableClientState(GL_VERTEX_ARRAY);

    glDisableClientState(GL_NORMAL_ARRAY);

    glDisableClientState(GL_TEXTURE_COORD_ARRAY);

}


void GLRenderer::RotateView(int x, int y)

{

    rotX += x;

    if (rotX > 360)

        rotX -= 360;

    else if (rotX < 0)

        rotX += 360;


    rotY += y;

    if (rotY > 360)

        rotY -= 360;

    else if (rotY < 0)

        rotY += 360;

}


UINT GLRenderer::LoadTexture(char* fileName)

{

    UINT texId;

    DImage img;

    img.Load(CString(fileName));

    glGenTextures(1, &texId);
    glBindTexture(GL_TEXTURE_2D, texId);

    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);

    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);

    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_COLOR, GL_MODULATE);

    gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGBA, img.Width(), img.Height(), GL_RGBA,
GL_UNSIGNED_BYTE, img.GetDIBBits());

    return texId;

}


void GLRenderer::SetThorusMat()

{

    float ambient[] = { 0.2, 0.0, 0.2, 1.0 }; // "vrlo tamna ambijetalna komponenta" gde smo
postavili ljubicastu boju?

    float diffuse[] = { 0.8, 0.0, 0.8, 1.0 }; //ovde?

    float specular[] = { 1.0, 0.6, 1.0, 1.0 };

    int shininess = 120;


    glMaterialfv(GL_FRONT, GL_AMBIENT, ambient);

    glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuse);

    glMaterialfv(GL_FRONT, GL_SPECULAR, specular);

    glMateriali(GL_FRONT, GL_SHININESS, shininess);

}


void GLRenderer::RotateThorus(double L)

{
    glTranslatef(L, 0.0, 0.0);

    double R = 2; //ovo znaci iz DRAWScene teksta, da se otkotrljao za 20 jedinica?

    double angle = L / R;

    glRotatef(-angle, 0.0, 0.0, 1.0);

    DrawThorus(R, 0.5);

}


//Januar 2018


void GLRenderer::DrawScene(CDC* pDC)

{

    wglMakeCurrent(pDC->m_hDC, m_hrc);

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glLoadIdentity();


    glTranslatef(0.0, -6, 0.0);

    glRotatef(xRot, 1.0, 0.0, 0.0);

    glRotatef(yRot, 0.0, 1.0, 0.0);


    glEnable(GL_TEXTURE_2D);

    LoadTexture(".\\Resources\\OpenGL.bmp");

    glColor3f(0.5, 0.4, 0.2);


    glBegin(GL_LINES);

    {

        glVertex3f(0.0, 0.0, 0.0);
        glVertex3f(10.0, 0.0, 0.0);


        glVertex3f(0.0, 0.0, 0.0);
        glVertex3f(0.0, 10.0, 0.0);


        glVertex3f(0.0, 0.0, 0.0);
        glVertex3f(0.0, 0.0, 10.0);
    }
    glEnd();


    //DrawSide(4, 5, 4, 5, 5, 3);
    //DrawBox(3, 4, 5, 5, 7, 3, 4, 9, 7);
    //DrawBasket(4, 5, 0.5);
    DrawFigure(0.5, 10, 5, 6, 30, 10, -45);


    glFlush();
    SwapBuffers(pDC->m_hDC);
    wglMakeCurrent(NULL, NULL);
}


void GLRenderer::RotateScene(int x, int y)
{
    xRot += x;
    if (xRot > 360)
        xRot -= 360;
    else if (xRot < 0)
```

```
        xRot += 360;

    yRot += y;
    if (yRot > 360)
        yRot -= 360;
    else if (yRot < 0)
        yRot += 360;
}

void GLRenderer::DrawSide(float x, float y, int nPartX, int nPartY, int nTexX, int nTexY)
{
    float xOffset = x / nPartX;
    float yOffset = y / nPartY;

    float xTexStep = (float)nTexX / (float)nPartX;
    float yTexStep = (float)nTexY / (float)nPartY;

    float currY = y / 2.0;
    float z = 0;

    float currTexY = 0.0;

    glNormal3f(0.0, 0.0, 1.0);
    for (int i = 0; i < nPartY; i++)
    {
        float currX = -x / 2.0;
        float currTexX = 0.0;

        glBegin(GL_QUAD_STRIP);

        {

            for (int i = 0; i < nPartX + 1; i++)

            {

                glTexCoord2f(currTexX, currTexY);

                glVertex3f(currX, currY, z);


                glTexCoord2f(currTexX, currTexY + yTexStep);

                glVertex3f(currX, currY - yOffset, z);


                currX += xOffset;

                currTexX += xTexStep;

            }

        }

        glEnd();


        currY -= yOffset;

        currTexY += yTexStep;

    }

}


void GLRenderer::DrawBox(float x, float y, float z, int nPartX, int nPartY, int nPartZ, int nTexX, int
nTexY, int nTexZ)

{

    glPushMatrix();
```

```
glTranslatef(0.0, 0.0, z / 2);
DrawSide(x, y, nPartX, nPartY, nTexX, nTexY);
glPopMatrix();

glPushMatrix();
glRotatef(90, 0.0, 1.0, 0.0);
glTranslatef(0.0, 0.0, x / 2);
DrawSide(z, y, nPartZ, nPartY, nTexZ, nTexY);
glPopMatrix();

glPushMatrix();
glRotatef(180, 0.0, 1.0, 0.0);
glTranslatef(0.0, 0.0, z / 2);
DrawSide(x, y, nPartX, nPartY, nTexX, nTexY);
glPopMatrix();

glPushMatrix();
glRotatef(-90, 0.0, 1.0, 0.0);
glTranslatef(0.0, 0.0, x / 2);
DrawSide(z, y, nPartZ, nPartY, nTexZ, nTexY);
glPopMatrix();

glPushMatrix();
glRotatef(90, 1.0, 0.0, 0.0);
glTranslatef(0.0, 0.0, y / 2);
DrawSide(x, z, nPartX, nPartZ, nTexX, nTexZ);
```

```
    glPopMatrix();


    glPushMatrix();
    glRotatef(-90, 1.0, 0.0, 0.0);
    glTranslatef(0.0, 0.0, y / 2);
    DrawSide(x, z, nPartX, nPartZ, nTexX, nTexZ);
    glPopMatrix();
}

void GLRenderer::DrawBasket(float w, float h, float d)
{
    glPushMatrix();
    glTranslatef(0.0, 0.0, w / 2 - d / 2);
    DrawBox(w, h, d, 10, 20, 1, 10, 20, 1);
    glPopMatrix();


    glPushMatrix();
    glRotatef(180, 0.0, 1.0, 0.0);
    glTranslatef(0.0, 0.0, w / 2 - d / 2);
    DrawBox(w, h, d, 10, 20, 1, 10, 20, 1);
    glPopMatrix();


    glPushMatrix();
    glRotatef(90, 0.0, 1.0, 0.0);
    glTranslatef(0.0, 0.0, w / 2 - d / 2);
    DrawBox(w - 2 * d, h, d, 10 - 2, 20, 1, 10 - 2, 20, 1);
    glPopMatrix();


    glPushMatrix();
    glRotatef(-90, 0.0, 1.0, 0.0);
    glTranslatef(0.0, 0.0, w / 2 - d / 2);
    DrawBox(w - 2 * d, h, d, 10 - 2, 20, 1, 10 - 2, 20, 1);
    glPopMatrix();


    glPushMatrix();
    glRotatef(90, 1.0, 0.0, 0.0);
    glTranslatef(0.0, 0.0, h / 2 - d / 2);
    DrawBox(w - 2 * d, w - 2 * d, d, 10 - 2, 10 - 2, 1, 10 - 2, 10 - 2, 1);
    glPopMatrix();
}


void GLRenderer::SetMaterial(float r, float g, float b)
{
    float diffuse[] = { r, g, b, 1.0 };
    float ambient[] = { 0.5 * r, 0.5 * g, 0.5 * b, 1.0 };
    float specular[] = { 1.0, 1.0, 1.0, 1.0 };
    float emission[] = { 0.0, 0.0, 0.0, 1.0 };
    int shininess = 15;


    glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuse);
    glMaterialfv(GL_FRONT, GL_AMBIENT, ambient);
    glMaterialfv(GL_FRONT, GL_SPECULAR, specular);
    glMaterialfv(GL_FRONT, GL_EMISSION, emission);

    glMateriali(GL_FRONT, GL_SHININESS, shininess);

}


void GLRenderer::DrawRoller(float r, float h, int n)

{

    //glColor3f(1.0, 0.0, 0.0);

    float angleStep = (piconst * 2) / n;

    float texStep = 1.0 / (float)n;

    float currAngle = 0;

    float currTex = 0;


    glBegin(GL_QUAD_STRIP);

    {

        for (int i = 0; i < n + 1; i++)

        {

            float x = r * cos(currAngle);

            float z = -r * sin(currAngle);


            glNormal3f(x / r, 0, z / r);


            glTexCoord2f(currTex, 0);

            glVertex3f(x, h / 2, z);


            glTexCoord2f(currTex, 1);

            glVertex3f(x, -h / 2, z);
            currAngle += angleStep;
            currTex += texStep;
        }
    }
    glEnd();

    glDisable(GL_TEXTURE_2D);

    currAngle = 0;
    glBegin(GL_TRIANGLE_FAN);
    {
        glNormal3f(0.0, 1.0, 0.0);
        glVertex3f(0.0, h / 2, 0.0);

        for (int i = 0; i < n + 1; i++)
        {
            float x = r * cos(currAngle);
            float z = -r * sin(currAngle);

            glVertex3f(x, h / 2, z);

            currAngle += angleStep;
        }
    }
    glEnd();
```

```
    currAngle = 0;

    glBegin(GL_TRIANGLE_FAN);

    {

        glNormal3f(0.0, -1.0, 0.0);

        glVertex3f(0.0, -h / 2, 0.0);


        for (int i = 0; i < n + 1; i++)

        {

            float x = r * cos(currAngle);

            float z = r * sin(currAngle);


            glVertex3f(x, -h / 2, z);


            currAngle += angleStep;

        }

    }

    glEnd();


    glEnable(GL_TEXTURE_2D);

}


void GLRenderer::DrawFigure(float w, float h, float d, float r, float alpha, float beta, float gama)

{

    /*int n = 20;

    glPushMatrix();
```

```
glRotatef(alpha, 0.0, 1.0, 0.0);
DrawRoller(r, 2 * w, n);

glTranslatef(0.0, w + d / 2, 0.0);
DrawBox(w, d, w, 1, 20, 1, 1, 20, 1);

glTranslatef(0.0, d / 2 + 0.75 * w, 0.0);
DrawBox(1.5, 1.5, 1.5, 1, 1, 1, 1, 1, 1);
glPopMatrix();

glPushMatrix();
glRotatef(alpha, 0.0, 1.0, 0.0);
glTranslatef(0.75 * w, w + d + 0.75 * w, 0.0);
glRotatef(beta, 0.0, 0.0, 1.0);
glTranslatef(d / 2, 0.0, 0.0);
glRotatef(-90, 0.0, 0.0, 1.0);
DrawBox(w, d, w, 1, 20, 1, 1, 20, 1);
glPopMatrix();

glPushMatrix();
glRotatef(alpha, 0.0, 1.0, 0.0);
glTranslatef(0.0, w + d + 0.75 * w, 0.0);
glRotatef(beta, 0.0, 0.0, 1.0);
glTranslatef(0.75 * w + d + 0.75 * w, 0.0, 0.0);
glRotatef(-beta, 0.0, 0.0, 1.0);
DrawBox(1.5, 1.5, 1.5, 1, 1, 1, 1, 1, 1);
```

```
glPopMatrix();


glPushMatrix();

glRotatef(alpha, 0.0, 1.0, 0.0);

glTranslatef(0.0, w + d + 0.75 * w, 0.0);

glRotatef(beta, 0.0, 0.0, 1.0);

glTranslatef(0.75 * w + d + 1.5 * w, 0.0, 0.0);

glRotatef(gama, 0.0, 0.0, 1.0);

glTranslatef(d / 2, 0.0, 0.0);

glRotatef(-90, 0.0, 0.0, 1.0);

DrawBox(w, d, w, 1, 20, 1, 1, 20, 1);

glPopMatrix();


glPushMatrix();

glRotatef(alpha, 0.0, 1.0, 0.0);

glTranslatef(0.0, w + d + 0.75 * w, 0.0);

glRotatef(beta, 0.0, 0.0, 1.0);

glTranslatef(0.75 * w + d + 0.75 * w, 0.0, 0.0);

glRotatef(gama, 0.0, 0.0, 1.0);

glTranslatef(0.75 * w + d + 3 * w, 0.0, 0.0);

glRotatef(-beta - gama, 0.0, 0.0, 1.0);

DrawBasket(6 * w, 0.5 * h, w);

glPopMatrix();*/


int n = 20;

glPushMatrix();
```

```
glRotatef(alpha, 0.0, 1.0, 0.0);

DrawRoller(r, 2 * w, n);


glTranslatef(0.0, d / 2 + w, 0.0);

DrawBox(w, d, w, 1, 20, 1, 1, 20, 1);


glTranslatef(0.0, d / 2 + 0.75 * w, 0.0);

DrawBox(1.5 * w, 1.5 * w, 1.5 * w, 1, 1, 1, 1, 1, 1);


glRotatef(-90 + beta, 0.0, 0.0, 1.0);

glTranslatef(0.0, d / 2 + 0.75 * w, 0.0);

DrawBox(w, d, w, 1, 20, 1, 1, 20, 1);


glTranslatef(0.0, d / 2 + 0.75 * w, 0.0);

glRotatef(-beta + 90, 0.0, 0.0, 1.0);

DrawBox(1.5 * w, 1.5 * w, 1.5 * w, 1, 1, 1, 1, 1, 1);


glRotatef(-90 + gama, 0.0, 0.0, 1.0);

glTranslatef(0.0, d / 2 + 0.75 * w, 0.0);

DrawBox(w, d, w, 1, 20, 1, 1, 20, 1);


glTranslatef(0.0, d / 2 + 3 * w, 0.0);

glRotatef(90 - gama, 0.0, 0.0, 1.0);

DrawBasket(6 * w, 0.5 * h, w);

glPopMatrix();
```

```
}

//Januar 2021 - teleskop

void GLRenderer::DrawScene(CDC* pDC)
{
   wglMakeCurrent(pDC->m_hDC, m_hrc);
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();

   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);

   float position[] = { 0.0, 0.0, 0.0, 1.0 };
   glLightfv(GL_LIGHT0, GL_POSITION, position);

   glTranslatef(0.0, 0.0, -10.0);
   glRotatef(xRot, 1.0, 0.0, 0.0);
   glRotatef(yRot, 0.0, 1.0, 0.0);

   //DrawCylinder(0.5, 4, 20, 5);
   DrawTelescope(0.5, 4, 20, 5);
```

```
    glFlush();
    SwapBuffers(pDC->m_hDC);
    wglMakeCurrent(NULL, NULL);
}

void GLRenderer::RotateScene(int x, int y)
{
    xRot += x;
    if (xRot > 360)
        xRot -= 360;
    else if (xRot < 0)
        xRot += 360;

    yRot += y;
    if (yRot > 360)
        yRot -= 360;
    else if (yRot < 0)
        yRot += 360;
}

void GLRenderer::DrawCylinder(float r, float h, float nr, float nh, bool axes)
{
    float texStepY = 1.0 / nh;
    float texStepX = 1.0 / nr;
    float texY = 0;
    float angStep = (2 * piconst) / nr; //za x i z(i nalazenje normala)
```

```
// float yStep = h / nh;
 //float startY = (((int)nh % 2) == 0) ? (((int)nh / 2) * yStep) : (((int)nh / 2) + 0.5) * yStep;
// float currY = startY;
 for (int i = 0; i < nh; i++)
 {
     float currAng = 0;
     float texX = 0;
     glBegin(GL_QUAD_STRIP);
     {
         for (int j = 0; j < nr + 1; j++)
         {
             float x = r * cos(currAng);
             float z = -r * sin(currAng);
             float nx = x / r;
             float nz = z / r;
             glNormal3f(nx, 0, nz);
             glTexCoord2f(texX, texY);
             glVertex3f(x, h/2, z);
             glTexCoord2f(texX, texY + texStepY);
             glVertex3f(x, -h/2, z);
             texX += texStepX;
             currAng += angStep;
         }
     }
     glEnd();
     texY += texStepY;
```

```
        // currY -= yStep;
    }

    glBegin(GL_TRIANGLE_FAN);
    {
        glNormal3f(0.0, 1.0, 0.0);
        glTexCoord2f(0.5, 0.5);
        glVertex3f(0, h / 2, 0);
        float currAng = 0;
        for (int i = 0; i < nr + 1; i++)
        {
            float x = r * cos(currAng);
            float z = -r * sin(currAng);
            float texX = 0.5 + 0.5 * cos(currAng); //zasto ovo prvo 0.5 + za x i - za y?? i sto uopste
koristimo 0.5???
            texY = 0.5 - 0.5 * sin(currAng);
            glTexCoord2f(texX, texY);
            glVertex3f(x, h / 2, z);
            currAng += angStep;
        }
    }
    glEnd();

    glBegin(GL_TRIANGLE_FAN);
    {
        glNormal3f(0.0, -1.0, 0.0);
    glTexCoord2f(0.5, 0.5);
    glVertex3f(0, -h / 2, 0);
    float currAng = 0;
    for (int i = 0; i < nr + 1; i++)
    {
        float x = r * cos(currAng);
        float z = r * sin(currAng);
        float texX = 0.5 + 0.5 * cos(currAng);
        texY = 0.5 - 0.5 * sin(currAng);
        glTexCoord2f(texX, texY);
        glVertex3f(x, -h / 2, z);
        currAng += angStep;
    }
}
glEnd();

if(axes)
{
    glDisable(GL_LIGHTING);
    glBegin(GL_LINES);
    {
        glColor3f(1.0, 0.0, 0.0);
        glVertex3f(0.0, 0.0, 0.0);
        glVertex3f(r + 1, 0.0, 0.0);

        glColor3f(0.0, 1.0, 0.0);
```

```
            glVertex3f(0.0, 0.0, 0.0);
            glVertex3f(0.0, h / 2 + 1, 0.0);


            glColor3f(0.0, 0.0, 1.0);
            glVertex3f(0.0, 0.0, 0.0);
            glVertex3f(0.0, 0.0, r + 1);
        }
        glEnd();
        glEnable(GL_LIGHTING);
    }
}

void GLRenderer::DrawTelescope(float r, float h, float nr, float nh)
{
    for (int i = 0; i < 3; i++)
    {
        glPushMatrix();
        glRotatef(i * 120, 0.0, 1.0, 0.0);
        glRotatef(alpha, 1.0, 0.0, 0.0);
        glTranslatef(0.0, -0.6 * h, 0.0);
        DrawCylinder(0.1 * r, 1.2 * h, nr, nh, false);
        glPopMatrix();
    }

    glPushMatrix();
    glRotatef(-90, 0.0, 0.0, 1.0);
    glRotatef(angleHor, 1.0, 0.0, 0.0);

    glRotatef(angleVer, 0.0, 0.0, 1.0);

    glTranslatef(0.0, (h / 2) - dHolder, 0.0);

    DrawCylinder(r, h, nr, nh, true);

    glPushMatrix();

    glTranslatef(0.0, h / 2 + 0.4 * h - dHidden, 0.0);

    DrawCylinder(0.8 * r, 0.8 * h, nr, nh, false);

    glPopMatrix();

    glTranslatef(0.0, -h / 2 + dVizor, 0.0);

    glRotatef(90, 0.0, 0.0, 1.0);

    glTranslatef(0.0, (0.2 / 2.0) * h, 0.0);

    DrawCylinder(0.1 * r, 0.2 * h, nr, nh, false);

    glTranslatef(0.0, (0.2 / 2.0) * h + (0.1 / 2.0) * r, 0.0);

    glRotatef(-90, 0.0, 0.0, 1.0);

    DrawCylinder(0.1 * r, 0.3 * h, nr, nh, false);

    glPopMatrix();

}




//Sep 2017  - dijamant neki


void GLRenderer::DrawScene(CDC* pDC)

{

    wglMakeCurrent(pDC->m_hDC, m_hrc);

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

```
glLoadIdentity();

glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);

glTranslatef(0.0, -3.0, 0.0);
glRotatef(xRot, 1.0, 0.0, 0.0);
glRotatef(yRot, 0.0, 1.0, 0.0);

float ambient[] = { 0.2, 0.2, 0.2, 1.0 };
float diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
float specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat position[] = { 0.5, 0.5, 1.0, 0.0 };

glColor3f(0.4, 0.5, 0.7);

glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, specular);
glLightfv(GL_LIGHT0, GL_POSITION, position);

//DrawColumn(0.5, 5);
//DrawPyramid(1, 4, 5);
//Draw2Pyramid(1, 4);
DrawFigure(0.5, 5, 1.5, 1, 0.9, 2, 20);
```

```
    glFlush();
    SwapBuffers(pDC->m_hDC);
    wglMakeCurrent(NULL, NULL);
}

void GLRenderer::DrawColumn(double side, double height)
{
    double a = side / 2;
    double h = height / 2;
    glBegin(GL_QUADS);
    {
        glNormal3f(1.0, 0.0, 0.0);
        glVertex3f(a, h, a);
        glVertex3f(a, -h, a);
        glVertex3f(a, -h, -a);
        glVertex3f(a, h, -a);

        glNormal3f(0.0, 0.0, -1.0);
        glVertex3f(a, h, -a);
        glVertex3f(a, -h, -a);
        glVertex3f(-a, -h, -a);
        glVertex3f(-a, h, -a);

        glNormal3f(-1.0, 0.0, 0.0);
        glVertex3f(-a, h, -a);
        glVertex3f(-a, -h, -a);
        glVertex3f(-a, -h, a);
        glVertex3f(-a, h, a);

        glNormal3f(0.0, 0.0, 1.0);
        glVertex3f(-a, h, a);
        glVertex3f(-a, -h, a);
        glVertex3f(a, -h, a);
        glVertex3f(a, h, a);

        glNormal3f(0.0, 1.0, 0.0);
        glVertex3f(a, h, a);
        glVertex3f(a, h, -a);
        glVertex3f(-a, h, -a);
        glVertex3f(-a, h, a);

        glNormal3f(0.0, -1.0, 0.0);
        glVertex3f(a, -h, a);
        glVertex3f(-a, -h, a);
        glVertex3f(-a, -h, -a);
        glVertex3f(a, -h, -a);
    }
    glEnd();
}

void GLRenderer::DrawPyramid(double side, double height, int n)
{
    double angleStep = (2 * piconst) / n;
    double currAngle = 0;


    double halfAngle = angleStep / 2;
    double r = (side / 2) / tan(halfAngle);
    double L = sqrt(r * r + height * height);
    double R = sqrt(r * r + (side / 2) * (side / 2));
    float ny = r / L;
    float nr = height / L;


    glBegin(GL_TRIANGLES);
    {
        for (int i = 0; i < n; i++)
        {
            glNormal3f(nr * cos(currAngle + (angleStep / 2)), ny, -nr * sin(currAngle + (angleStep /
2)));


            float x1 = R * cos(currAngle);
            float x2 = R * cos(currAngle + angleStep);
            float z1 = -R * sin(currAngle);
            float z2 = -R * sin(currAngle + angleStep);


            glVertex3f(0, height, 0);
            glVertex3f(x1, 0, z1);
            glVertex3f(x2, 0, z2);
            currAngle += angleStep;
        }
    }
    glEnd();

    currAngle = 0;
    glBegin(GL_TRIANGLE_FAN);
    {
        glNormal3f(0.0, -1.0, 0.0);
        glVertex3f(0.0, 0.0, 0.0);
        for (int i = 0; i < n + 1; i++)
        {
            glVertex3f(R * cos(currAngle), 0.0, R * sin(currAngle));
            currAngle += angleStep;
        }
    }
    glEnd();
}

void GLRenderer::Draw2Pyramid(double side, double height)
{
    int n = 5;
    DrawPyramid(side, height, n);
    glPushMatrix();
    glRotatef(180, 1.0, 0.0, 0.0);
    DrawPyramid(side, height, n);
    glPopMatrix();
}

void GLRenderer::DrawFigure(float aS, float hS, float aR, float size, float height, float offset, float
angle)
{
    glPushMatrix();
    glTranslatef(-hS / 2 + aS / 2, hS / 2, 0.0);
    DrawColumn(aS, hS);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(hS / 2 - aS / 2, hS / 2, 0.0);
    DrawColumn(aS, hS);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(0.0, hS + aS / 2, 0.0);
    glRotatef(90, 0.0, 0.0, 1.0);
    DrawColumn(aS, hS);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-hS / 2 + aS / 2 + offset, hS - aS / 2, 0.0);
    glRotatef(angle, 0.0, 1.0, 0.0);
    DrawColumn(aR, aS);
    glPopMatrix();

    glTranslatef(-hS / 2 + aS / 2 + offset, hS - aS, 0.0);

    glPushMatrix();
    glBegin(GL_LINES);
    {
        glVertex3f(0.0, 0.0, 0.0);
        glVertex3f(0.0, -2 * height, 0.0);
    }
    glEnd();

    glTranslatef(0.0, -2 * height - height, 0.0);
    glRotatef(angle, 0.0, 1.0, 0.0);
    Draw2Pyramid(size, height);
    glPopMatrix();
}

void GLRenderer::RotateView(int x, int y)
{
    xRot += x;
    if (xRot > 360)
        xRot -= 360;
    else if (xRot < 0)
        xRot += 360;
    yRot += y;

    if (yRot > 360)

        yRot -= 360;

    else if (yRot < 0)

        yRot += 360;

}



//Pauk



void CGLRenderer::DrawScene(CDC *pDC)

{

            wglMakeCurrent(pDC->m_hDC, m_hrc);

            //---------------------------------

            glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

            glLoadIdentity();


            // funkcije

            glTranslated(0.0, 0.0, -dist);

            glRotated(-alfa, 1.0, 0.0, 0.0);

            glRotated(-beta, 0.0, 1.0, 0.0);


            DrawAxes();

            DrawEnvCube(50.0);
```

```
	DrawSpider();


	glFlush();
	SwapBuffers(pDC->m_hDC);
	//---------------------------------
	wglMakeCurrent(NULL, NULL);
}

void CGLRenderer::DrawAxes()
{
	glLineWidth(2.0);
	glBegin(GL_LINES);

	glColor3d(0.0, 0.0, 1.0);
	glVertex3d(0.0, 0.0, 0.0);
	glVertex3d(50.0, 0.0, 0.0);

	glColor3d(1.0, 0.0, 0.0);
	glVertex3d(0.0, 0.0, 0.0);
	glVertex3d(0.0, 50.0, 0.0);

	glColor3d(0.0, 1.0, 0.0);
	glVertex3d(0.0, 0.0, 0.0);
	glVertex3d(0.0, 0.0, 50.0);

	glEnd();
```

```
}

void CGLRenderer::DrawEnvCube(double a)
{
        glEnable(GL_TEXTURE_2D);

        //prednja
        glBindTexture(GL_TEXTURE_2D, m_texEnv[0]);
        glBegin(GL_QUADS);
        glColor3d(1.0, 1.0, 1.0);

        glTexCoord2d(0, 1);
        glVertex3d(-a / 2, -a / 2, -a / 2);
        glTexCoord2d(0, 0);
        glVertex3d(-a / 2, a / 2, -a / 2);
        glTexCoord2d(1, 0);
        glVertex3d(a / 2, a / 2, -a / 2);
        glTexCoord2d(1, 1);
        glVertex3d(a / 2, -a / 2, -a / 2);

        glEnd();

        //leva
        glBindTexture(GL_TEXTURE_2D, m_texEnv[2]);
        glBegin(GL_QUADS);
        glColor3d(1.0, 1.0, 1.0);
```

```
glTexCoord2d(0, 0);
glVertex3d(-a / 2, a / 2, a / 2);
glTexCoord2d(0, 1);
glVertex3d(-a / 2, -a / 2, a / 2);
glTexCoord2d(1, 1);
glVertex3d(-a / 2, -a / 2, -a / 2);
glTexCoord2d(1, 0);
glVertex3d(-a / 2, a / 2, -a / 2);

glEnd();

//donja
glBindTexture(GL_TEXTURE_2D, m_texEnv[5]);
glBegin(GL_QUADS);
glColor3d(1.0, 1.0, 1.0);

glTexCoord2d(0, 0);
glVertex3d(-a / 2, -a / 2, -a / 2);
glTexCoord2d(1, 0);
glVertex3d(a / 2, -a / 2, -a / 2);
glTexCoord2d(1, 1);
glVertex3d(a / 2, -a / 2, a / 2);
glTexCoord2d(0, 1);
glVertex3d(-a / 2, -a / 2, a / 2);
```

```
glEnd();


//zadnja

glBindTexture(GL_TEXTURE_2D, m_texEnv[1]);

glBegin(GL_QUADS);

glColor3d(1.0, 1.0, 1.0);


glTexCoord2d(0, 0);

glVertex3d(a / 2, a / 2, a / 2);

glTexCoord2d(0, 1);

glVertex3d(a / 2, -a / 2, a / 2);

glTexCoord2d(1, 1);

glVertex3d(-a / 2, -a / 2, a / 2);

glTexCoord2d(1, 0);

glVertex3d(-a / 2, a / 2, a / 2);


glEnd();


//desna

glBindTexture(GL_TEXTURE_2D, m_texEnv[3]);

glBegin(GL_QUADS);

glColor3d(1.0, 1.0, 1.0);


glTexCoord2d(0, 0);

glVertex3d(a / 2, a / 2, -a / 2);

glTexCoord2d(0, 1);
```

```
}


        glVertex3d(a / 2, -a / 2, -a / 2);

        glTexCoord2d(1, 1);

        glVertex3d(a / 2, -a / 2, a / 2);

        glTexCoord2d(1, 0);

        glVertex3d(a / 2, a / 2, a / 2);


        glEnd();


        //gore

        glBindTexture(GL_TEXTURE_2D, m_texEnv[4]);

        glBegin(GL_QUADS);

        glColor3d(1.0, 1.0, 1.0);


        glTexCoord2d(0, 0);

        glVertex3d(-a / 2, a / 2, a / 2);

        glTexCoord2d(0, 1);

        glVertex3d(-a / 2, a / 2, -a / 2);

        glTexCoord2d(1, 1);

        glVertex3d(a / 2, a / 2, -a / 2);

        glTexCoord2d(1, 0);

        glVertex3d(a / 2, a / 2, a / 2);


        glEnd();


        glDisable(GL_TEXTURE_2D);

}
```

```
void CGLRenderer::DrawSphere(double r, int nSeg, double texU, double texV, double texR)
{
	double stepAlpha = 3.1415 / nSeg;
	double stepBeta = 2 * 3.1415 / nSeg;

	double alpha, beta;

	alpha = -3.1415 / 2;

	for (int i = 0; i < nSeg; i++)
	{
		beta = 0.0;
		glBegin(GL_QUAD_STRIP);

		for (int j = 0; j < nSeg + 1; j++)
		{
			double x1 = r * cos(alpha) * cos(beta);
			double y1 = r * sin(alpha);
			double z1 = r * cos(alpha) * sin(beta);

			double tx1 = x1 / r * texR + texU; //Ovo se racuna samo kada u tekstu
kaze da treba nekako da se nalepi tekstura na sferu
			double ty1 = z1 / r * texR + texV; //Ovo se racuna samo kada u tekstu kaze
da treba nekako da se nalepi tekstura na sferu
```

```
                        glTexCoord2d(tx1, ty1);  //Ovo se racuna samo kada u tekstu kaze da
treba nekako da se nalepi tekstura na sferu

                        glVertex3d(x1, y1, z1);


                        double x2 = r * cos(alpha + stepAlpha) * cos(beta);

                        double y2 = r * sin(alpha + stepAlpha);

                        double z2 = r * cos(alpha + stepAlpha) * sin(beta);


                        double tx2 = x2 / r * texR + texU;

                        double ty2 = z2 / r * texR + texV;


                        glTexCoord2d(tx2, ty2);

                        glVertex3d(x2, y2, z2);


                        beta += stepBeta;

                }


                glEnd();

                alpha += stepAlpha;

        }

}


void CGLRenderer::DrawSpiderBody()

{

        //rep

        glPushMatrix();
```

```
	glTranslated(6.5, 0.0, 0.0);
	glScaled(1.0, 0.8, 1.0);
	DrawSphere(5, 10, 0.25, 0.25, 0.24);

	glPopMatrix();

	//sise
	glPushMatrix();
	glScaled(1.0, 0.5, 1.0);
	DrawSphere(3, 10, 0.25, 0.25, 0.24);
	glPopMatrix();

	//glavudza
	glPushMatrix();
	glTranslated(-3.5, 0.0, 0.0);
	glScaled(1.0, 0.5, 1.0);
	DrawSphere(2, 10, 0.75, 0.25, 0.24);
	glPopMatrix();
}

void CGLRenderer::DrawSpider()
{
	glEnable(GL_TEXTURE_2D);
	glBindTexture(GL_TEXTURE_2D, m_texSpider);
	glColor3f(1.0, 1.0, 1.0);
```

```
glPushMatrix();

DrawSpiderBody();

glPopMatrix();


//// pipci


glPushMatrix();


glRotated(75, 0.0, 1.0, 0.0);


for (int i = 0; i < 4; i++)

{

            glRotated(-30, 0.0, 1.0, 0.0);

            glPushMatrix();

            glRotatef(45, 1.0, 0.0, 0.0);

            DrawLeg();

            glPopMatrix();

}


glPopMatrix();


glPushMatrix();


glRotated(180, 0.0, 1.0, 0.0);
```

```
	glRotated(75, 0.0, 1.0, 0.0);

	for (int i = 0; i < 4; i++)
	{
		glRotated(-30, 0.0, 1.0, 0.0);
		glPushMatrix();
		glRotatef(45, 1.0, 0.0, 0.0);
		DrawLeg();
		glPopMatrix();
	}

	glPopMatrix();

	glDisable(GL_TEXTURE_2D);
}

void CGLRenderer::DrawCone(double r, double h, int nSeg, double texU, double texV, double
texR)
{
	double stepAlpha = 2 * 3.1415 / nSeg;

	glBegin(GL_TRIANGLE_FAN);

	double alpha = 0.0;
	glTexCoord2d(0.75, 0.75);
	glVertex3d(0.0, h, 0.0);
	for (int i = 0; i < nSeg + 1; i++)
	{
		double x1 = r * cos(alpha);
		double z1 = r * sin(alpha);

		double tx1 = x1 / r * texR + texU;
		double ty1 = z1 / r * texR + texV;

		glTexCoord2d(tx1, ty1);
		glVertex3d(x1, 0.0, z1);

		alpha += stepAlpha;
	}

	glEnd();
}

void CGLRenderer::DrawLegSegment(double r, double h, int nSeg)
{
	glPushMatrix();

	DrawSphere(r, 2 * nSeg, 0.25, 0.25, 0.24);
	DrawCone(r, h, nSeg, 0.75, 0.75, 0.25);

	glPopMatrix();
```

```
}

void CGLRenderer::DrawLeg()
{
	glPushMatrix();

	DrawLegSegment(1, 10, 5);
	glTranslated(0.0, 10.0, 0.0);
	glRotated(85.0, 1.0, 0.0, 0.0);
	DrawLegSegment(1, 15, 5);

	glPopMatrix();
}
```