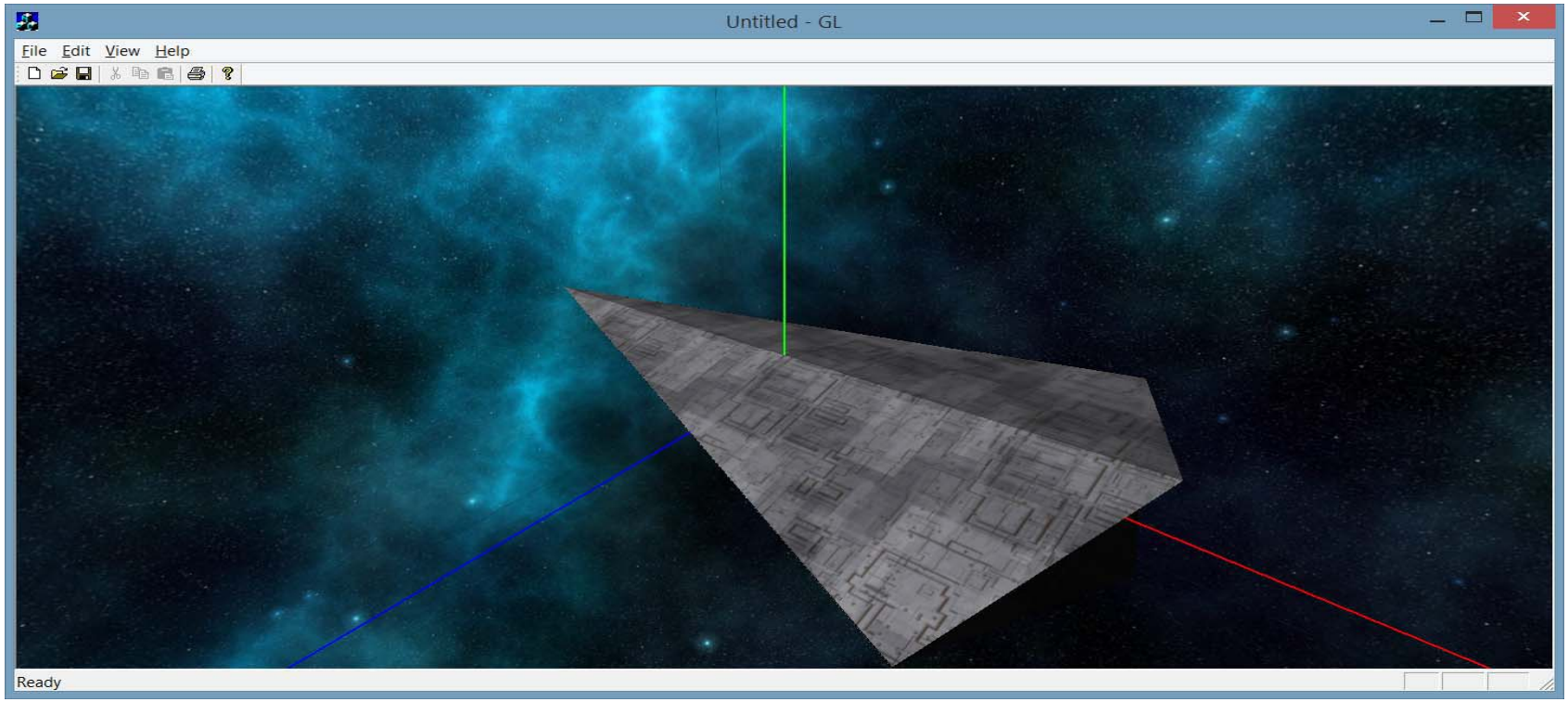
II kolokvijum 2018



Definisati perspektivnu projekciju sa FOV = 50° i ispuniti funkcije **PrepareScene**(), **DrawScene**() i **Reshape**() odgovarajućim OpenGL funkcijskim pozivima kako bi se omogućilo dalje crtanje. Nacrtati tri linije dužine 10, koje kreću iz koordinatnog početka i poklapaju sa koordinatnim osama. Neka je linija duž X-ose crvena, linija duž Y-ose zelena, a duž Z-ose plava. Preći na sledeću tačku tek kada koordinatne ose budu vidljive. [10 poena]

```
void CGLRenderer::PrepareScene(CDC *pDC)
{
   wglMakeCurrent(pDC->m_hDC, m_hrc);
   glClearColor(1.0, 1.0, 1.0, 1.0);
   glEnable(GL_DEPTH_TEST);
   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::DrawScene(CDC *pDC)
{
   wglMakeCurrent(pDC->m_hDC, m_hrc);
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();
   // …
   gluLookAt (10, 10, 10, 0, 0, 0, 0, 1, 0);
   // Tripod
            glLineWidth(2.0);
            glBegin(GL_LINES);
                     glColor3f(1, 0, 0);
                     glVertex3f(0, 0, 0);
                     glVertex3f(10, 0, 0);
                     glColor3f(0, 1, 0);
                     glVertex3f(0, 0, 0);
                     glVertex3f(0, 10, 0);
                     glColor3f(0, 0, 1);
                     glVertex3f(0, 0, 0);
                     glVertex3f(0, 0, 10);
            glEnd();
   glFlush();
   SwapBuffers(pDC->m_hDC);
   wglMakeCurrent(NULL, NULL);
}
```

Napisati funkciju UINT CGLRenderer::**LoadTexture**(char* fileName), koja učitava teksturu sa datim imenom (fileName) i vraća ID kreirane teksture. Korišćenjem ove funkcije u okviru **PrepareScene**() učitati teksture: ShipT1.png, front.jpg, left.jpg, right.jpg, back.jpg, top.jpg i bot.jpg. [15 poena]

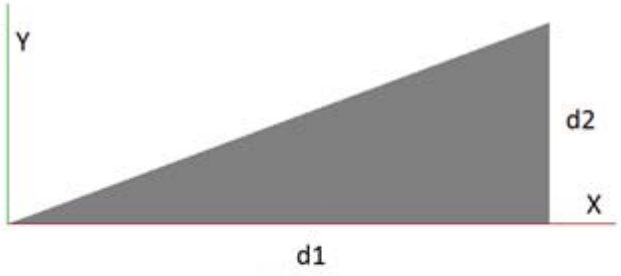
```
UINT CGLRenderer::LoadTexture(char* fileName)
{
	UINT texID;
	DImage img;
	img.Load(CString(fileName));

	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_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
	glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);

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

```
void CGLRenderer::PrepareScene(CDC *pDC)
{
   wglMakeCurrent(pDC->m_hDC, m_hrc);
   glClearColor(1.0, 1.0, 1.0, 1.0);
   glEnable(GL_DEPTH_TEST);

   m_texShip = LoadTexture("ShipT1.png");
   m_texSpace[0] = LoadTexture("front.jpg");
   m_texSpace[1] = LoadTexture("left.jpg");
   m_texSpace[2] = LoadTexture("right.jpg");
   m_texSpace[3] = LoadTexture("back.jpg");
   m_texSpace[4] = LoadTexture("top.jpg");
   m_texSpace[5] = LoadTexture("bot.jpg");

   glEnable(GL_TEXTURE_2D);

   wglMakeCurrent(NULL, NULL);
}
```

```
protected:
    UINT m_texShip;
    UINT m_texSpace[6];
```

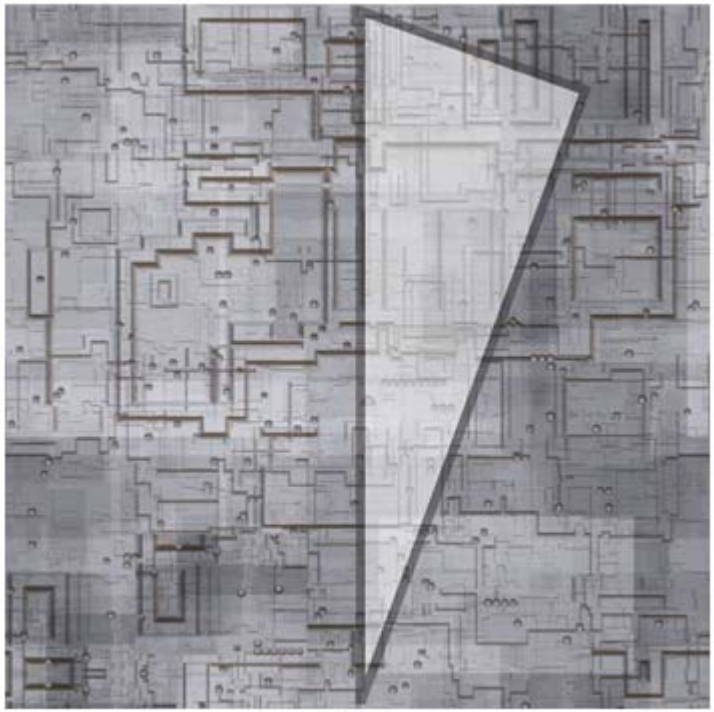
```
void CGLRenderer::DestroyScene(CDC *pDC)
{
 wglMakeCurrent(pDC->m_hDC, m_hrc);
 glDeleteTextures(1, &m_texShip);
 glDeleteTextures(6, m_texSpace);

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

Napisati funkciju void CGLRenderer::**DrawTriangle**(float d1, float d2, float rep), kojom se iscrtava pravougli trougao, kateta dužina **d1** i **d2** (Sl.2). Parametar rep definiše koliko se puta ponavlja tekstura na površini trougla. Na Sl.3 prikazano je kako izgleda mapiranje teksture, ukoliko je broj ponavljanja 1. U temenima trougla definisati teksturne koordinate i normale, tako da prikaz bude korektan. Na Sl.4 prikazano je izgleda teksturisan trougao, ako je broj ponavljanja teksture 3. Voditi računa da se teksturne koordinate tačno proračunaju na osnovu dužina kateta. [15 poena]

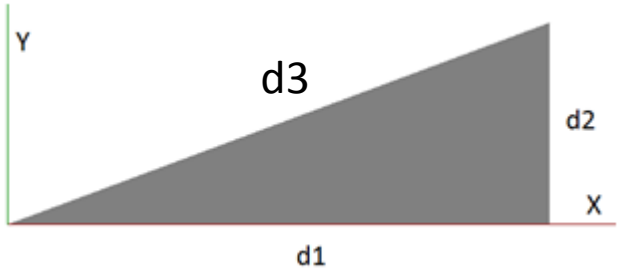


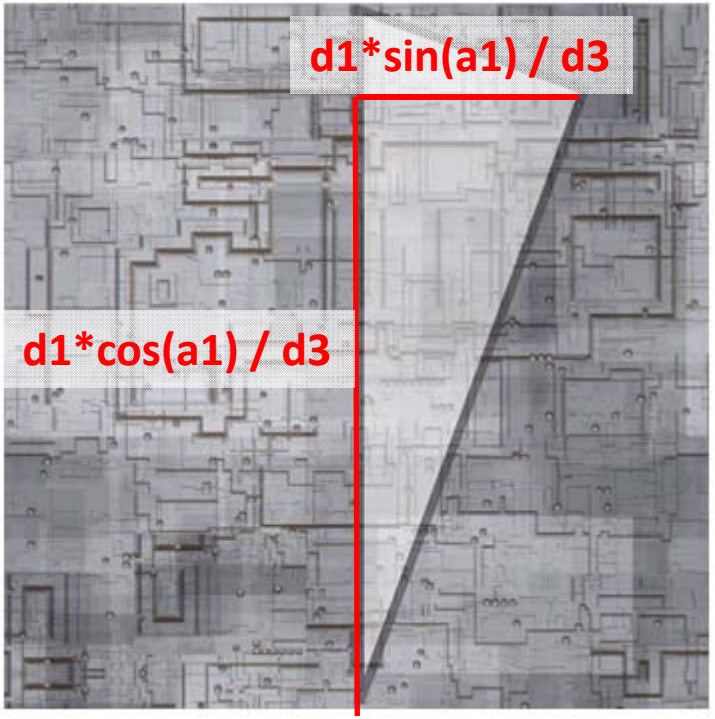
Sl.2

Sl.4

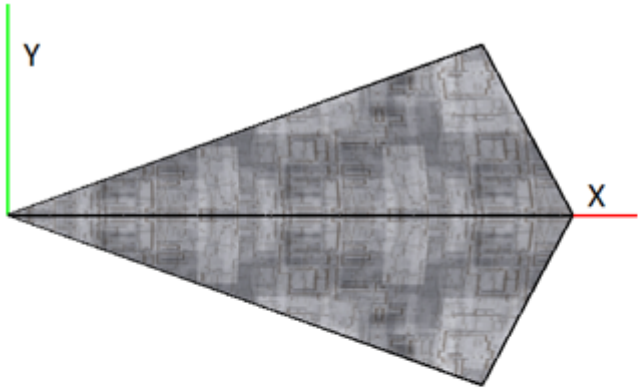

Sl.3

```
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, 1.0*rep);
        glVertex3f(d1, d2, 0.0);
        glEnd();
}
```



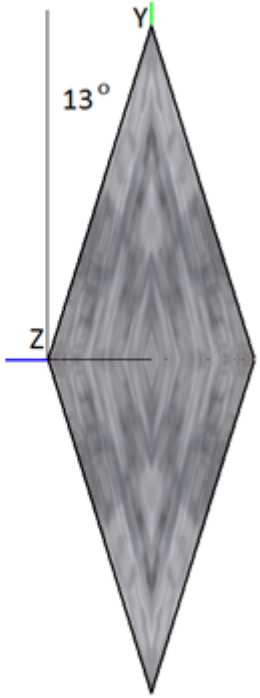


Napisati funkciju void CGLRenderer::**DrawShip**(), koja iscrtava svemirski brod sastavljen od 4 trougla, nacrtanih prethodnom funkcijom sa katetama dužina 5.8 i 2.15. Gornji i donji deo sastoje se od po dva trougla, čije se hipotenuze preklapaju (Sl.5). Dva trougla su nagnuta u odnosu na Y-osu za 13° (Sl.6), a hipotenuza je nagnuta 4.75° u odnosu na X-osu (Sl.7). [20 poena]



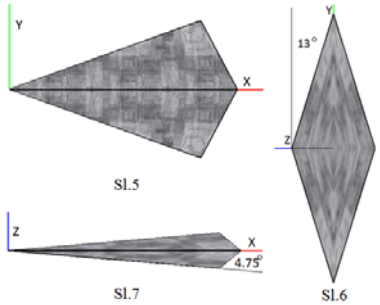
Sl.5

Sl.7



Sl.6

```
void CGLRenderer::DrawShip()
{
glBindTexture(GL_TEXTURE_2D, m_texShip);
double a1 = atan2(2.15, 5.8);
float rep = 3.0;
// Gornji levo
glPushMatrix();
glRotatef(-4.75, 0, 1, 0);
glRotatef(13, 1, 0, 0);
glRotatef(-a1*toDeg, 0, 0, 1);
DrawTriangle(5.8, 2.15, rep);
glPopMatrix();
// Gornji desno
glPushMatrix();
glScalef(1, -1, 1);
glRotatef(-4.75, 0, 1, 0);
glRotatef(13, 1, 0, 0);
glRotatef(-a1*toDeg, 0, 0, 1);
DrawTriangle(5.8, 2.15, rep);
glPopMatrix();
```




```
// Donji deo
glPushMatrix();
glRotatef(180, 1, 0, 0);
// Levo
glPushMatrix();
glRotatef(-4.75, 0, 1, 0);
glRotatef(13, 1, 0, 0);
glRotatef(-a1*toDeg, 0, 0, 1);
DrawTriangle(5.8, 2.15, rep);
glPopMatrix();
// Desno
glPushMatrix();
glScalef(1, -1, 1);
glRotatef(-4.75, 0, 1, 0);
glRotatef(13, 1, 0, 0);
glRotatef(-a1*toDeg, 0, 0, 1);
DrawTriangle(5.8, 2.15, rep);
glPopMatrix();
glPopMatrix();
}
```

Napisati funkciju void CGLRenderer::**DrawSpaceCube**(double a), koja iscrtava kocku stranice dužine **a**, kojom se uokvirava scena (centrirana na poziciji kamere). Kocka se uvek vidi samo sa unutrašnje strane i na njenim stranicama su „nalepljene“ teksture: front.jpg, left.jpg, right.jpg, back.jpg, top.jpg i bot.jpg. Na kocu ne sme da utiče svetlost. [10 poena]

```
void CGLRenderer::DrawSpaceCube(double a)
{
// Front
glBindTexture(GL_TEXTURE_2D, m_texSpace[0]);
glBegin(GL_QUADS);
glTexCoord2f(0.0, 1.0);
glVertex3d(-a / 2, a / 2, -a / 2);
glTexCoord2f(0.0, 0.0);
glVertex3d(-a / 2, -a / 2, -a / 2);
glTexCoord2f(1.0, 0.0);
glVertex3d(a / 2, -a / 2, -a / 2);
glTexCoord2f(1.0, 1.0);
glVertex3d(a / 2, a / 2, -a / 2);
glEnd();

// Left
glBindTexture(GL_TEXTURE_2D, m_texSpace[1]);
glBegin(GL_QUADS);
glTexCoord2f(0.0, 1.0);
glVertex3d(-a / 2, a / 2, a / 2);
glTexCoord2f(0.0, 0.0);
glVertex3d(-a / 2, -a / 2, a / 2);
glTexCoord2f(1.0, 0.0);
glVertex3d(-a / 2, -a / 2, -a / 2);
glTexCoord2f(1.0, 1.0);
glVertex3d(-a / 2, a / 2, -a / 2);
glEnd();
```

```
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);
        glEnable(GL_TEXTURE_2D);
        glPushMatrix();
        glRotated(m_beta, 1.0, 0.0, 0.0);
        glRotated(m_alpha, 0.0, 1.0, 0.0);

        DrawSpaceCube(1.0);

        glPopMatrix();
        glEnable(GL_DEPTH_TEST);
        //…
}
```

Postaviti direkcioni izvor svetlosti bele boje, koji se nalazi u beskonačnosti u pravcu pozitivne Z-ose. Izvor svetlosti ne sme da utiče na Svemir, niti da se pomera sa posmatračem. Uticaj svetla uključivati/isključivati na taster S. [10 poena].

```
glTranslatef(0, 0, -m_dist);
glRotated(m_beta, 1.0, 0.0, 0.0);
glRotated(m_alpha, 0.0, 1.0, 0.0);

GLfloat light_position[] = { 0.0, 0.0, 1.0, 0.0 };
glLightfv(GL_LIGHT0, GL_POSITION, light_position);

// Tripod
//…
glEnable(GL_TEXTURE_2D);

if (m_bLight)
{
        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
}
```

Popuniti funkciju void CGLRenderer::**DrawScene**(CDC *pDC), tako da iscrtava brod u centru scene i kocku koja oslikava svemir svuda okolo (Sl.1). [5 poena].

```
        double d = sqrt(5.8*5.8 + 2.15*2.15);
        glTranslatef(-d/2., 0, 0);
        glRotatef(90, 1, 0, 0);
        DrawShip();


        glFlush();
        SwapBuffers(pDC->m_hDC);

        //-------------------------------
        wglMakeCurrent(NULL, NULL);
}
```

Omogućiti animiranje scene tako da pritisak na taster:

← – rotira posmatrača oko Y-ose udesno oko centra scene,

→ – rotira posmatrača oko Y-ose ulevo oko centra scene,

↑ – rotira posmatrača naviše,

↓ – rotira posmatrača naniže,

\+ – približava posmatrača centru scene,

– – udaljava posmatrača od centra scene. [15 poena]

```
void CGLView::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
        if (nChar == VK_RIGHT)
                m_glRenderer.m_alpha -= 5.0;
        if (nChar == VK_LEFT)
                m_glRenderer.m_alpha += 5.0;
        if (nChar == VK_UP)
                m_glRenderer.m_beta += 5.0;
        if (nChar == VK_DOWN)
                m_glRenderer.m_beta -= 5.0;
        if (nChar == VK_ADD)
                m_glRenderer.m_dist /= 1.1;
        if (nChar == VK_SUBTRACT)
                m_glRenderer.m_dist *= 1.1;
        if (nChar == 'S')
                m_glRenderer.m_bLight = !m_glRenderer.m_bLight;
        Invalidate();
        CView::OnKeyDown(nChar, nRepCnt, nFlags);
}
```