II kolokvijum 2015

Definisati perspektivnu projekciju sa FOV = 40° i ispuniti funkcije **PrepareScene**(), **DrawScene**() i **Reshape**() odgovarajućim OpenGL funkcijskim pozivima kako bi se omogućilo dalje crtanje. Napomena: Nacrtati trougao u koordinatnom početku i preći na sledeću tačku tek kada trougao bude vidljiv. [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);
glCullFace(GL_BACK);
glEnable(GL_CULL_FACE);

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(40, (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();
// …
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. [10 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;
}
```

U funkciji void CGLRenderer::**PrepareScene**(CDC *pDC), učitati teksture: TSC0.jpg – TSC5.jpg, M0.jpg – M5.jpg, S0.jpg – S5.jpg, i postaviti sve potrebne početne vrednosti parametara. Izvršiti dealokaciju učitanih tekstura u odgovarajućoj funkciji. [5 poena]

```
T[0] = LoadTexture("TSC0.png");
T[1] = LoadTexture("TSC1.png");
T[2] = LoadTexture("TSC2.png");
T[3] = LoadTexture("TSC3.png");
T[4] = LoadTexture("TSC4.png");
T[5] = LoadTexture("TSC5.png");

M[0] = LoadTexture("M0.jpg");
M[1] = LoadTexture("M1.jpg");
M[2] = LoadTexture("M2.jpg");
M[3] = LoadTexture("M3.jpg");
M[4] = LoadTexture("M4.jpg");
M[5] = LoadTexture("M5.jpg");

S[0] = LoadTexture("S0.jpg");
S[1] = LoadTexture("S1.jpg");
S[2] = LoadTexture("S2.jpg");
S[3] = LoadTexture("S3.jpg");
S[4] = LoadTexture("S4.jpg");
S[5] = LoadTexture("S5.jpg");

glEnable(GL_TEXTURE_2D);
```

```
protected:
UINT T[6];        // Zemlja
UINT S[6];        // Svemir
UINT M[6];        // Mesec
```

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

glDeleteTextures(6, T);
glDeleteTextures(6, M);
glDeleteTextures(6, S);

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

Napisati funkciju void CGLRenderer::**DrawPatch**(double R, int n), kojom se iscrtava četvorostrana zakrivljena površina (patch) koja predstavlja 1/6 površine sfere (Sl.2). Patch je iscrtava parametarski po x,y∈[-1,+1], pri čemu je n broj podeoka po pravcu na koliko je podeljen patch. Na Sl.2, n = 20. Iz parametarskih koordinata prelazi se u polarne po jednačinama:

$$\varphi = \tan^{-1} x$$

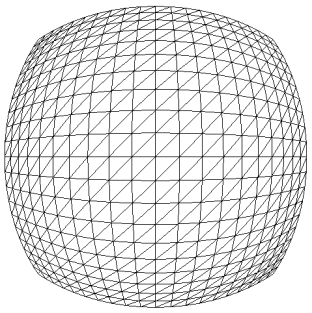
$$\theta = \tan^{-1}(y \cdot \cos\varphi)$$

a iz polarnih u Dekartove po jednačinama:

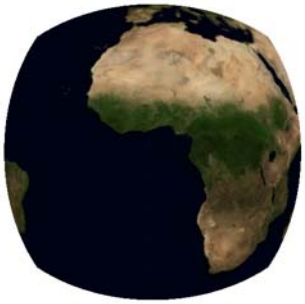
$$x = R \cdot \cos\theta \cdot \sin\varphi$$

$$z = R \cdot \cos\theta \cdot \cos\varphi$$

$$y = R \cdot \sin\theta$$



Sl.2



Sl.3

U temenima generisati teksturne koordinate i normale. Teksturne koordinate treba da omoguće mapiranje čitave teksture na čitavu površinu patch-a (Sl.3). Normale se postavljaju radijalno. [25 poena]

```
void CGLRenderer::DrawPatch(double R, int n)
{
double delta = 2. / (double)n;
double y = 1.0;
for (int i = 0; i < n; i++)
{
glBegin(GL_TRIANGLE_STRIP);
double x = -1.0;
for (int j = 0; j < n+1; j++)
{
double phi, theta;
inverseTSC(x, y, phi, theta);

double xd, yd, zd;
xd = R * cos(theta) * sin(phi);
yd = R * sin(theta);
zd = R * cos(theta) * cos(phi);

glNormal3f(xd / R, yd / R, zd / R);
glTexCoord2f( ( x + 1.0) / 2.0,
                        (-y + 1.0) / 2.0 );
glVertex3f(xd, yd, zd);

inverseTSC(x, y - delta, phi, theta);

xd = R * cos(theta) * sin(phi);
yd = R * sin(theta);
zd = R * cos(theta) * cos(phi);

glNormal3f(xd / R, yd / R, zd / R);
glTexCoord2f( ( x + 1.0) / 2.0,
                        (-y + delta + 1.0) / 2.0);
glVertex3f(xd, yd, zd);
x += delta;
}
glEnd();
y -= delta;
}
}

void CGLRenderer::inverseTSC(double x,
double y, double& phi, double& theta)
{
phi = atan(x);
theta = atan( y * cos(phi) );
}
```

Korišćenjem funkcije DrawPatch() napisati funkciju void CGLRenderer::**DrawEarth**(double R, int tes), koja icrtava čitavu površinu Zemlje, sastavljenu od 6 patch-eva i na nju primenjuje teksture TSC0 – TSC5. Redosled primena tekstura prikazan je na Sl.4. [10 poena]

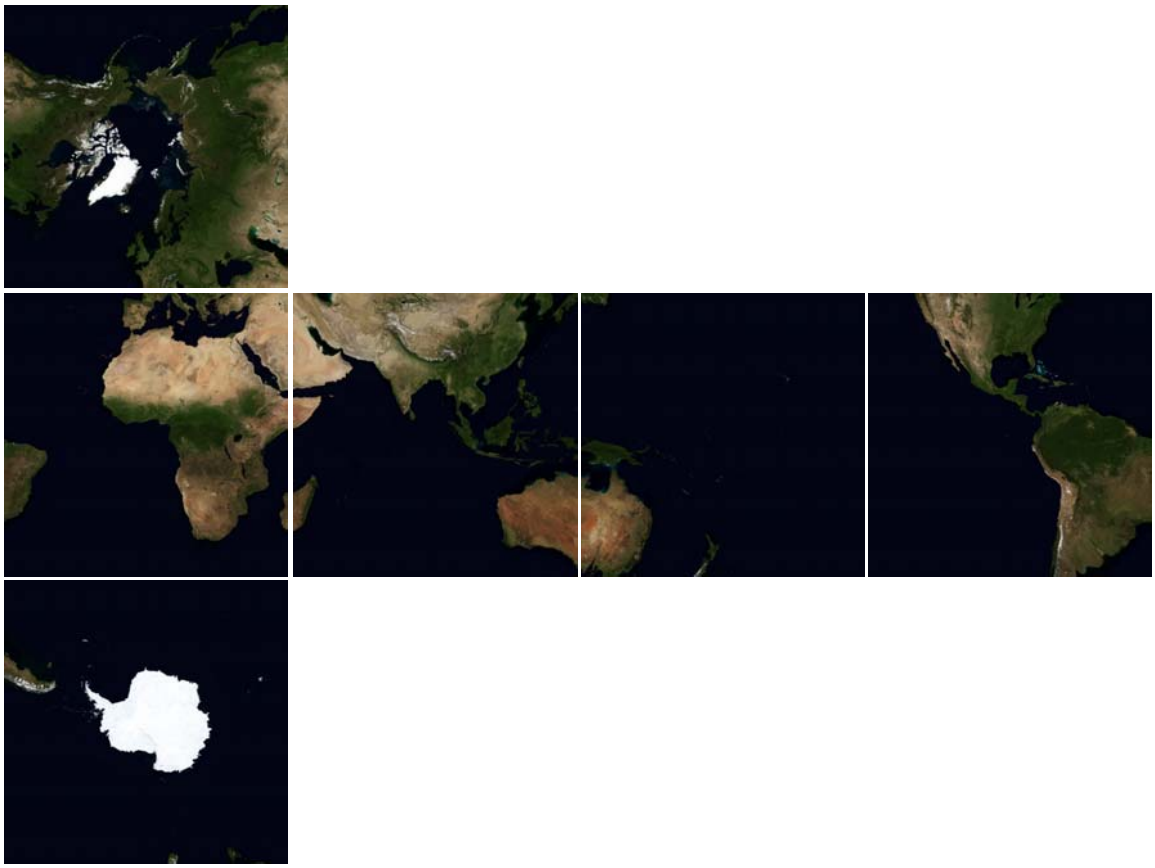
```
void CGLRenderer::DrawEarth(double R, int tes)
{
glPushMatrix();
for (int i = 0; i < 4; i++)
{
glBindTexture(GL_TEXTURE_2D, T[i]);
DrawPatch(R, tes);
glRotated(90, 0.0, 1.0, 0.0);
}
glPopMatrix();

glPushMatrix();
glRotated(-90, 1.0, 0.0, 0.0);
glBindTexture(GL_TEXTURE_2D, T[4]);
DrawPatch(R, tes);
glPopMatrix();

glPushMatrix();
glRotated(90, 1.0, 0.0, 0.0);
glBindTexture(GL_TEXTURE_2D, T[5]);
DrawPatch(R, tes);
glPopMatrix();
}
```

TSC4			
TSC0	TSC1	TSC2	TSC3
TSC5			

Sl.4



Na isti način napraviti funkcije **DrawMoon**() i **DrawSpace**(), ali u njima primeniti teksture M i S, respektivno. [5 poena]

```
void CGLRenderer::DrawMoon(double R, int tes)
{
glPushMatrix();
for (int i = 0; i < 4; i++)
{
glBindTexture(GL_TEXTURE_2D, M[i]);
DrawPatch(R, tes);
glRotated(90, 0.0, 1.0, 0.0);
}
glPopMatrix();
glPushMatrix();
glRotated(-90, 1.0, 0.0, 0.0);
glBindTexture(GL_TEXTURE_2D, M[4]);
DrawPatch(R, tes);
glPopMatrix();
glPushMatrix();
glRotated(90, 1.0, 0.0, 0.0);
glBindTexture(GL_TEXTURE_2D, M[5]);
DrawPatch(R, tes);
glPopMatrix();
}
```

```
void CGLRenderer::DrawSpace(double R, int tes)
{
glDisable(GL_CULL_FACE);
glPushMatrix();
for (int i = 0; i < 4; i++)
{
glBindTexture(GL_TEXTURE_2D, S[i]);
DrawPatch(R, tes);
glRotated(90, 0.0, 1.0, 0.0);
}
glPopMatrix();
glPushMatrix();
glRotated(-90, 1.0, 0.0, 0.0);
glBindTexture(GL_TEXTURE_2D, S[4]);
DrawPatch(R, tes);
glPopMatrix();
glPushMatrix();
glRotated(90, 1.0, 0.0, 0.0);
glBindTexture(GL_TEXTURE_2D, S[5]);
DrawPatch(R, tes);
glPopMatrix();
glEnable(GL_CULL_FACE);
}
```

Popuniti funkciju void CGLRenderer::**DrawScene**(CDC *pDC), tako da u centru scene bude Zemlja (*Earth*), poluprečnika 3, na 50 jedinica od nje Mesec (*Moon*), poluprečnika 1, a centrirana u kameri bude sfera koja predstavlja Svemir (*Space*). [5 poena].

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]

```
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(m_beta, 1.0, 0.0, 0.0);
glRotated(m_alpha, 0.0, 1.0, 0.0);
DrawSpace(1.0, 20);
glPopMatrix();
```

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

float light_ambient[] = { 0.0, 0.0, 0.0, 1.0 };
float light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
float light_specular[] = { 1.0, 1.0, 1.0, 1.0 };

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

GLfloat light_position[] = { 0.0, 0.0, 1.0, 0.0 };
```

```
glEnable(GL_DEPTH_TEST);

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

glLightfv(GL_LIGHT0, GL_POSITION,
light_position);
DrawEarth(3.0, 20);

glTranslatef(-50.0, 0.0, 0.0);
glRotatef(m_moonRot, 0.0, 1.0, 0.0);
DrawMoon(1.0, 20);

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

Omogućiti animiranje scene tako da pritisak na taster:

- ← – rotira posmatrača oko Y-ose udesno oko Zemlje,
- → – rotira posmatrača oko Y-ose ulevo oko Zemlje,
- ↑ – rotira posmatrača naviše,
- ↓ – rotira posmatrača naniže,
- + – približava posmatrača centru Zemlje
- – – udaljava posmatrača od centra Zemlje
- Q/W – rotira Mesec oko njegove ose. [20 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 == 'Q')           m_glRenderer.m_moonRot  -= 5.0;
if (nChar == 'W')           m_glRenderer.m_moonRot  += 5.0;
if (nChar == 'S')           m_glRenderer.m_bLight = !m_glRenderer.m_bLight;
Invalidate();
CView::OnKeyDown(nChar, nRepCnt, nFlags);
}
```