RAČUNARSKA GRAFIKA

Predavanje broj: 12

Nastavna jedinica: OpenGL

Nastavne teme: Svetlo. Normale. Ambient. Diffuse. Specular. Emissive. Spot. materijali. Svetlo i materijali. Postavljanje svetla. Primer. Teksture: smeštanje, borders, mipmape, filteri, objekti, funkcije, ponavljanje, odsecanje.

Predavač: prof. dr Perica S. Štrbac, dipl. ing.

Literatura:

James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes: "Computer Graphics: Principles and Practice", 2nd ed. in C, Addison-Wesley, 1996.

Osvetljenje, normale

- Potrebno je u temenima definisati normale koje određuju proračun za pojedine tipove svetla.
  - Normale se zadaju PRE komande glVertex komandom glNormal3{bsidf}:

```
void glNormal3{bsidf}(TYPE nx, TYPE ny, TYPE nz);
void glNormal3{bsidf}v(const TYPE *v);
```

  - Automatska normalizacija (svođenje na jedinični vektor) je potrebna kada se koriste transformacije koje narušavaju normale (npr. scale) što se postiže kao što sledi:

```
glEnable(GL_NORMALIZE)
```

- U OpenGL-u moguće je:
  - kreiranje,
  - selekcija,
  - pozicioniranje jednog ili više izvora svetlosti
- Razmatraju se sledeći tipovi svetla u OpenGLu:
  - ambient,
  - diffuse
  - specular.

Osvetljenje, ambient, difuse

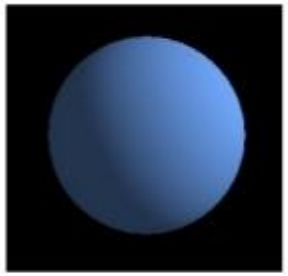
- Ambient
  je svetlo koje dolazi iz svih pravaca tako da jednako utiču na na svaku tačku objekata na sceni.
  - Površine objekata na sceni su jednako osvetljene u svim svojim tačkama.


Wireframe


1. Original Color

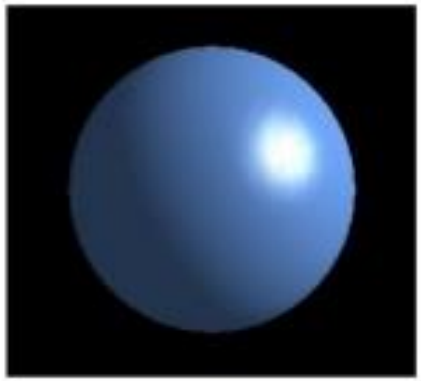

2. Original Color + Ambient

- Diffuse
  je svetlo koje dolazi iz određenog pravca i "udara" u površinu objekta a onda se odbija od površine (prema uglu pod kojim pada).
  - Ako svetlo pada pod uglom od 90 stepeni na površinu, ista će biti maksimalno osvetljena.


3. Original Color + Ambient + Diffuse

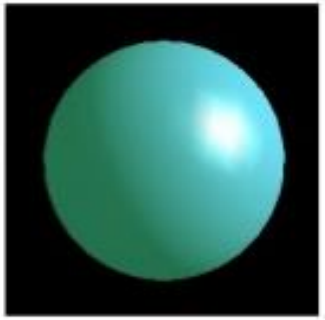
Osvetljenje, specular, emmisive

- Specular svetlo
  slično diffuse-u (direkciono je svetlo) samo što je veći uticaj osvetljenosti površine na koju pada od diffuse-a.
  - Pošto stvara svetle tačke na mestima pada zove se specular highlight.
  - Pošto je direkciono zavisi i od ugla pod kojim kamera posmatra scenu.



4. Original Color + Ambient + Diffuse + Specular

- Svetlo se formira kao kombinacija prethodnih izvora svetla.
- Ako se doda i emmisive kao sposobnost objekta da emituje svetlo dobija se slika:



5. Original Color + Ambient + Diffuse + Specular + Emissive

Osvetljenje, materijali

- Npr. da bi se formiralo svetlo zelenog lasera vrednosti boja datog svetla bi bili:
  - Ambient RGBA = (0.0, 0.07, 0.0, 1.0)
  - Diffuse RGBA = (0.0, 0.12, 0.0, 1.0)
  - Specular RGBA = (0.0, 0.98, 0.0, 1.0)

Materijali

- Za dobijanje vizuelnih efekata postoji mogućnost da se poligonima dodele osim boje i određena svojstva (materijali).
- Kao i svetla, materijali imaju tri osnovna svojstva: ambient, diffuse i specular.
- Zavisno od tri atributa svetla i tri atributa materijala pikseli dobijaju završnu boju.
- Boja piksela se dobija proračunima parametara svetla i materijala, npr:
  - ambient svetlo : (0.5, 0.5, 0.5, 1.0)
  - ambient materijal : (0.5, 1.0, 0.5, 1.0)

  završna vrednost se dobija kao što sledi:
  (0.5*0.5, 0.5*1.0, , 0.5*0.5, 1.0*1.0) = (0.25, 0.50, 0.25, 1.0)
  - Isto je i sa difuse i specular atributima.

Materijali

- Sledi kod koji koristi svetlo i materijal:

```
//belo svetlo
GLfloat ambientLight[] = { 1.0f, 1.0f, 1.0f, 1.0f };
//omoguci svetla
glEnable(GL_LIGHTING);
//ambient light postavljen preko vektora ambientLight
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,ambientLight);
//sivi materijal
GLfloat gray[] = { 0.75f, 0.75f, 0.75f, 1.0f };
//materijal ambient i difuse za prednje lice poligona
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, gray);
//crtanje trugla na koga ce biti primenjeno prethodno
glBegin(GL_TRIANGLES);
  glVertex3f(-15.0f,  0.0f,  30.0f );
  glVertex3f(  0.0f, 15.0f,  30.0f );
  glVertex3f(  0.0f,  0.0f, -56.0f );
glEnd();
```

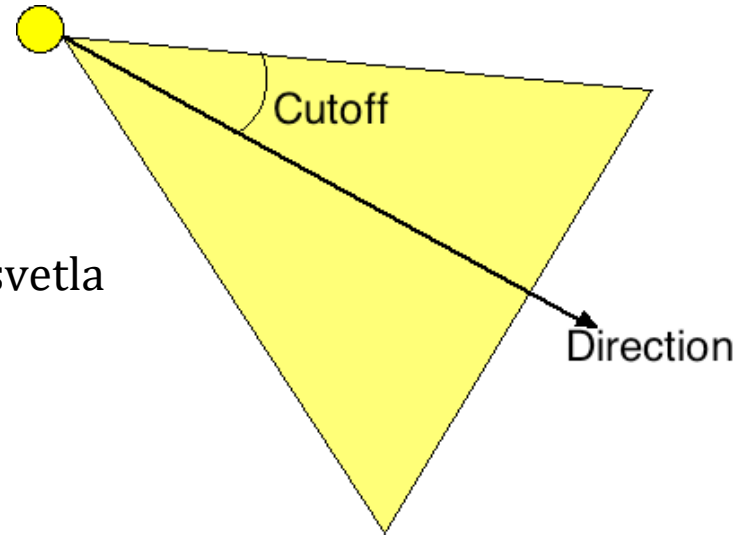
Materijali, color tracking

- Brži kod za davanje osobina materijala je color tracking:

```
// Enable color tracking
glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT,GL_AMBIENT_AND_DIFFUSE);
// Boja se dodeljuje naredbom glColor()
glcolor3f(0.75f, 0.75f, 0.75f );
glBegin(GL_TRIANGLES);
   glVertex3f(-15.0f,0.0f,30.0f );
   glVertex3f(0.0f, 15.0f, 30.0f );
   glVertex3f(0.0f, 0.0f, -56.0f );
glEnd();
```

Svetlo i materijal

- OpenGL garantuje 8 izvora svetla što je dato u konstanti MAX_LIGHTS .
- Specular parametar se koristi za defnisanje stepena svetlucavosti materijala i njenu boju (metalik).
- Specular se definiše iz dve komponente:
  - specular - boja svetlucavosti
  - shininess - intenzitet svetlucavosti (od 1 do 128).



Spot svetlo

- Spot svetla su precizno usmereni izvori svetla
  - Defnisani su:
    - lokacijom odakle se emituju,
    - destinacijom - fokusom svetla
    - uglom kupe koju "prave" (parametar je polovina ugla)

Postavljanje svetla GL_LIGHT0

- Kreiranje spot svetla

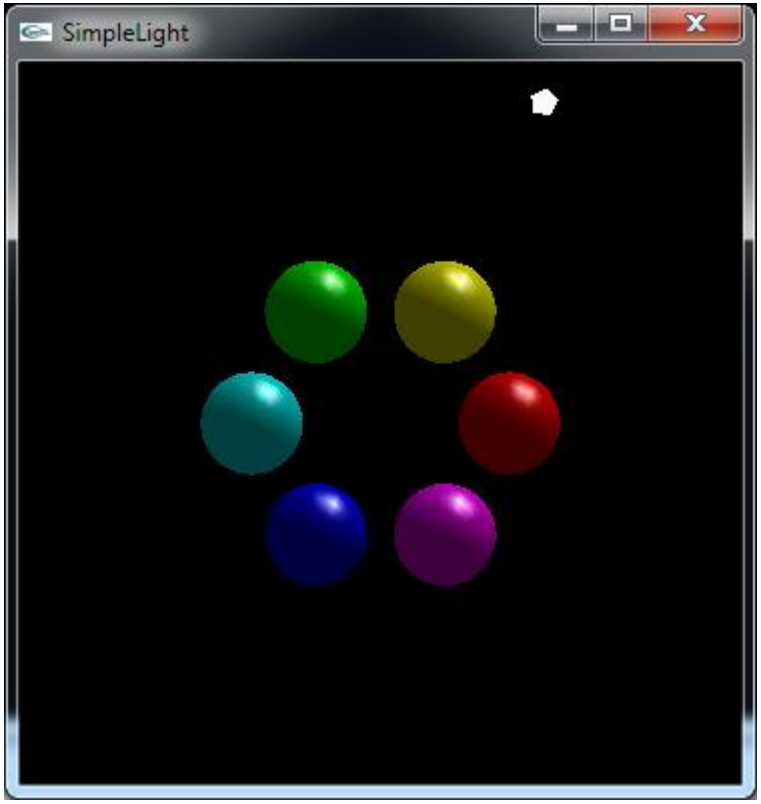
```
GLfloat lightPos[] = { 0.0f, 0.0f, 75.0f, 1.0f }; //poz. izvora svetla
GLfloat specular[] = { 1.0f, 1.0f,  1.0f, 1.0f };
GLfloat speccol[] =  { 1.0f, 1.0f,  1.0f, 1.0f }; //boja svetlucavosti
GLfloat ambientLight[] = { 0.5f, 0.5f, 0.5f, 1.0f };
GLfloat spotDir[] = { 0.0f, 0.0f, -1.0f }; // destinacija svetla
glEnable(GL_DEPTH_TEST); // za uklanjanje skrivenih poligona
glFrontFace(GL_CCW); // standardno CCW smer za prednje lice poligona
glEnable(GL_CULL_FACE); // culling zadnjih strana poligona
// Enable lighting
glEnable(GL_LIGHTING);
// opis svetla : light 0
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientLight);
// Svetlo sastavljamo od difuse i specular komponenti
glLightfv(GL_LIGHT0,GL_DIFFUSE,ambientLight);
glLightfv(GL_LIGHT0,GL_SPECULAR,specular);
glLightfv(GL_LIGHT0,GL_POSITION,lightPos); // pozicija u prostoru
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir); // destinacija
```

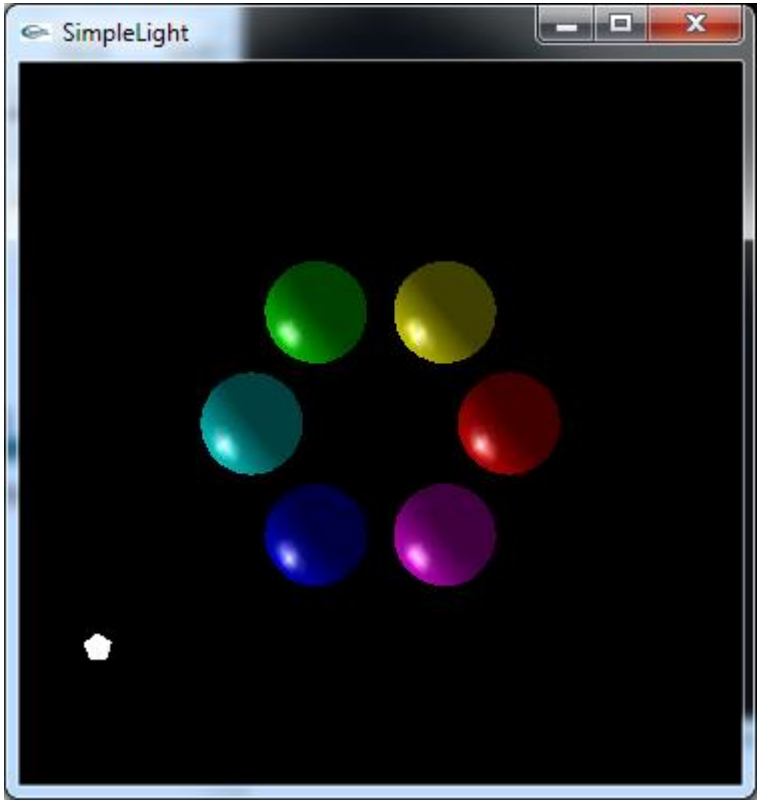
Postavljanje svetla GL_LIGHT0

```
// parametri za spot svetlo
// polovina ugla kupe svetla = 60
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,60.0f );
// omogucavanje svetla
glEnable(GL_LIGHT0);
// color tracking
glEnable(GL_COLOR_MATERIAL);
// svojsta materijala prednjeg lica poligona
glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
// dva parametra za spot
//    boja svetlucavosti
glMaterialfv(GL_FRONT, GL_SPECULAR,speccol );
//    intenzitet svetlucavosti
glMateriali(GL_FRONT, GL_SHININESS,128);
// boja pozadine
glClearColor(0.0f, 0.0f, 0.0f, 1.0f );
// sada bi se crtali objekti
// na koje bi bilo primenjeno prethodno
```

Primer svetla koje kruži oko objekata

- Ideja je da se programira izvor svetlosti koji kruži oko kuglica različitih boja i gde se vidi efekat izvora svetlosti na tim kuglicama.





Primer svetla koje kruži oko objekata 1/9

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include "GL/glut.h"// paziti kod mene je u projektu

//prototipovi
void myKeyboardFunc(unsigned char key, int x, int y);
void mySpecialKeyFunc(int key, int x, int y);
void drawScene(void);
void initRendering();
void resizeWindow(int w, int h);

// vrednosti za kontrolu osobina materijala
float Noemit[4] = { 0.0, 0.0, 0.0, 1.0 };      //crna, nema em.
float SphShininess = 50;  // specular exponent kuglica.
float SphAmbDiff[6][4] = {// ambient/diffuse boja kuglica
{ 0.5, 0.0, 0.0, 1.0 },// Red
{ 0.5, 0.5, 0.0, 1.0 },// Yellow
{ 0.0, 0.5, 0.0, 1.0 },// Green
{ 0.0, 0.5, 0.5, 1.0 },// Cyan
{ 0.0, 0.0, 0.5, 1.0 },// Blue
{ 0.5, 0.0, 0.5, 1.0 }// Magenta
};
```

Primer svetla koje kruži oko objekata 2/9

```
float SphSpecular[4] = { 1, 1, 1, 1.0 };

// vrednosti osvetljenja
float ambientLight[4] = { 0.2, 0.2, 0.2, 1.0 };
float Lt0amb[4]  = { 0.3, 0.3, 0.3, 1.0 };
float Lt0diff[4] = { 1.0, 1.0, 1.0, 1.0 };
float Lt0spec[4] = { 1.0, 1.0, 1.0, 1.0 };

float zeroPos[4] = { 0, 0, 0, 1 }; // ishodiste (x,y,z,w)
float dirI[4]    = { 1, 0, 0, 0 }; // u beskonacnosti

int LightIsPositional = 0;
int RunMode = 1;              // izvor svetla se krece ili ne (1 ili 0)

float CurrentAngle = 0.0f;// tekuci ugao u stepenima
float AnimateStep = 0.5f;// korak animacije
// reakcija na koriscenje normalnih tastera
void myKeyboardFunc(unsigned char key, int x, int y)
{
  switch (key) {
    case 'r':
      RunMode = 1 - RunMode;// prebaci na drugu vrednost 0 <--> 1
```

Primer svetla koje kruži oko objekata 3/9

```
      if (RunMode == 1) {
        glutPostRedisplay();
      }
      break;
    case 's':
      RunMode = 1;
      drawScene();
      RunMode = 0;
      break;
    case 'l':// lokalni light mode
      LightIsPositional = 1 - LightIsPositional;
      if (RunMode == 0) {
        drawScene();
      }
      break;
    case 27:// Escape key
      exit(1);
  }
}
// reakcija na koriscenje specijalnih tastera
void mySpecialKeyFunc(int key, int x, int y)
{
```

Primer svetla koje kruži oko objekata 4/9

```
  switch (key) {
    case GLUT_KEY_UP:
      if (AnimateStep < 5.0) {   // gornje ogranicenje brzine
        AnimateStep *= sqrt(2.0);// povecaj korak
      }
      break;
    case GLUT_KEY_DOWN:
      if (AnimateStep>1.0e-3) {  // donje ogranicenje brzine
        AnimateStep /= sqrt(2.0);// umanji korak
      }
      break;
  }
}
// crtanje scene
void drawScene(void)
{
  int i;
  // obrisi bafer boje i bafer dubine
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  if (RunMode == 1) {
    // podesi parameter animacije
    CurrentAngle += AnimateStep;
```

Primer svetla koje kruži oko objekata 5/9

```
  if (CurrentAngle > 360.0)
    CurrentAngle -= 360.0*floor(CurrentAngle / 360.0);// da nema ow.
}
// rotacija
glMatrixMode(GL_MODELVIEW);// modelview
glLoadIdentity(); // ucitavanje matrice identiteta
// postavljanje svetla na datu poziciju
glPushMatrix();
  glRotatef(CurrentAngle, 0.0, 0.0, 1.0);// rotacija oko z ose
  glTranslatef(7.0, 0.0, 0.0);// translacija
  // kuglica emituje svetlost (emissive)
  glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, Lt0spec);
  glutSolidSphere(0.3, 5, 5); // crtanje kuglice
  glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, Noemit);// nema em.
  if (LightIsPositional == 1) {
     glLightfv(GL_LIGHT0, GL_POSITION, zeroPos);// poz. u ish.
  }
  else {
    glLightfv(GL_LIGHT0, GL_POSITION, dirI);// poz. u beskon.
  }
glPopMatrix();
```

Primer svetla koje kruži oko objekata 6/9

```
  // crtanje 6 kuglica razlicitih boja
  for (i = 0; i<6; i++) {
   glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,SphAmbDiff[i]);
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR,  SphSpecular);
    glMaterialf (GL_FRONT_AND_BACK, GL_SHININESS, SphShininess);
    glPushMatrix();
     glRotatef(60.0*(float)i, 0.0, 0.0, 1.0);// svaka rel. na 60 stepeni
     glTranslatef(2.5, 0.0, 0.0);
     glutSolidSphere(1.0, 20, 20);
    glPopMatrix();
  }
  // isprazni pipeline, swap-uj bafere
  glFlush();
  glutSwapBuffers();

  if (RunMode == 1) {
    glutPostRedisplay();// Trigger an automatic redraw for animation
  }
}
```

Primer svetla koje kruži oko objekata 7/9

```
// inicijalizacija renderovanja
void initRendering()
{
  glEnable(GL_DEPTH_TEST);
  glEnable(GL_LIGHTING);
  glEnable(GL_LIGHT0);
  // ambient light
  glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientLight);
  // Light 0
  glLightfv(GL_LIGHT0, GL_AMBIENT,  Lt0amb  );
  glLightfv(GL_LIGHT0, GL_DIFFUSE,  Lt0diff );
  glLightfv(GL_LIGHT0, GL_SPECULAR, Lt0spec );
}
// reakcija na promenu velicine prozora
void resizeWindow(int w, int h)
{
  float viewWidth  = 7.0; // postavljanje koeficijenta za w
  float viewHeight = 7.0; // postavljanje koeficijenta za h
  glViewport(0, 0, w, h);
  h = (h == 0) ? 1 : h;
  w = (w == 0) ? 1 : w;
```

Primer svetla koje kruži oko objekata 8/9

```
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  if (h < w) {
    viewWidth *= (float)w / (float)h;
  }
  else {
    viewHeight *= (float)h / (float)w;
  }
    gluOrtho2D(-viewWidth, viewWidth, -viewHeight, viewHeight);
  }

// postavljanje OpenGL, definisanje reakcije na dogadjaje ulazak
// glavnu petlju osluskivanja dogadjaja
int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  // RGB mod, dupli bafer, kontrola dubine
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
  // pozicija i velicina glut prozora
  glutInitWindowPosition(10, 60);
  glutInitWindowSize(360, 360);
```

Primer svetla koje kruži oko objekata 9/9

```
    glutCreateWindow("SimpleLight");
    // inicijalizacija rendering scene
    initRendering();
    // postavljanje callback funkcija
    glutKeyboardFunc(myKeyboardFunc); // reak. na "normalne" tastere
    glutSpecialFunc(mySpecialKeyFunc);// reak. na "special" tastere
    // callback funkcija za promenu velicine prozora
    glutReshapeFunc(resizeWindow);
    // moze se registrovati callback f-ja za slucaj kada glut
    // nista ne radi
    // glutIdleFunc( myIdleFunction );
    // callback funkcija iscrtavanja scene
    glutDisplayFunc(drawScene);
    fprintf(stdout, "Press \"r\" to run,  \"s\" to single step.\n");
    // glavna glut petlja
    glutMainLoop();
    return(0);
}
```

Teksture

- Ideja je da se slika ili deo slike mapira na neki poligon.
- Specificiranje teksture

```
void glTexImage2D(GLenum target, GLint level, GLint components,
                  GLsizei width, GLsizei height, GLint border,
                  GLenum format, GLenum type, const GLvoid *pixels);
```

target - GL_TEXTURE_2D ili GL_PROXY_TEXTURE_2D,

level – nivo detalja, 0 za osnovnu mapu, n za n-ti nivo mipmape,

components – broj komponenti boja (1,..,4), ili predefinisano (npr. GL_RGB, GL_RGBA)

width – širina teksturne mape, mora biti $2^N+2*(\text{border})$

height – visina teksturne mape, mora biti $2^M+2*(\text{border})$

border – širina okvira (0 ili 1),

format – format piksela (GL_RGB, GL_RGBA, GL_ALPHA, ...),

type – tip podataka za piksele (GL_UNSIGNED_BYTE, GL_BYTE, GL_BITMAP, GL_UNSIGNED_SHORT, GL_SHORT, GL_UNSIGNED_INT, GL_INT, i GL_FLOAT),

*pixels – pokazivač na bitmapu

Način smeštanja bitmape kao teksture

Specificiranje načina smeštanja bitmape koja se koristi kao tekstura

- Sledeća funkcija mora se pozvati pre specifikacije teksture, tj. pre glTexImage.

```
void glPixelStore{if}( GLenum pname, GLfloat param );
```

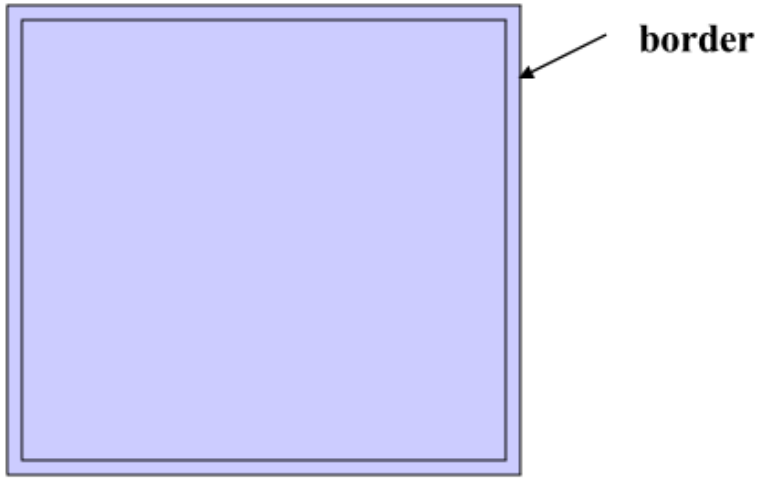
  - pname – simboličko ime parametra koji se postavlja (GL_PACK_SWAP_BYTES, GL_PACK_LSB_FIRST, GL_PACK_ROW_LENGTH, ... GL_UNPACK_ALIGNMENT ,. ..)
  - param – vrednost parametra

- Koristićemo samo GL_UNPACK_ALIGNMENT, koji može imati jednu od sledećih vrednosti:
  - 1 – poravnanje po bajtovima (byte-alignment),
  - 2 – svaki red mora imati paran broj bajtova,
  - 4 – poravnanje po rečima (word alignment)
  - 8 – poravnanje po dvostrukim rečima

- Ako je linija (vrsta) u bitmapi kraća od celobrojnog umnoška bajtova (zadatog u ovom parametru), podrazumeva se da je dopunjena do pune dužine.
  Npr. BMP slike imaju poravnanje po rečima.

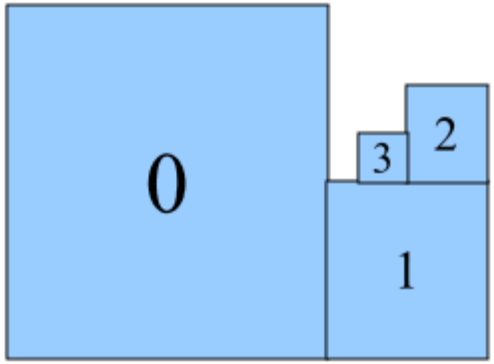
Okviri tekstura (borders)

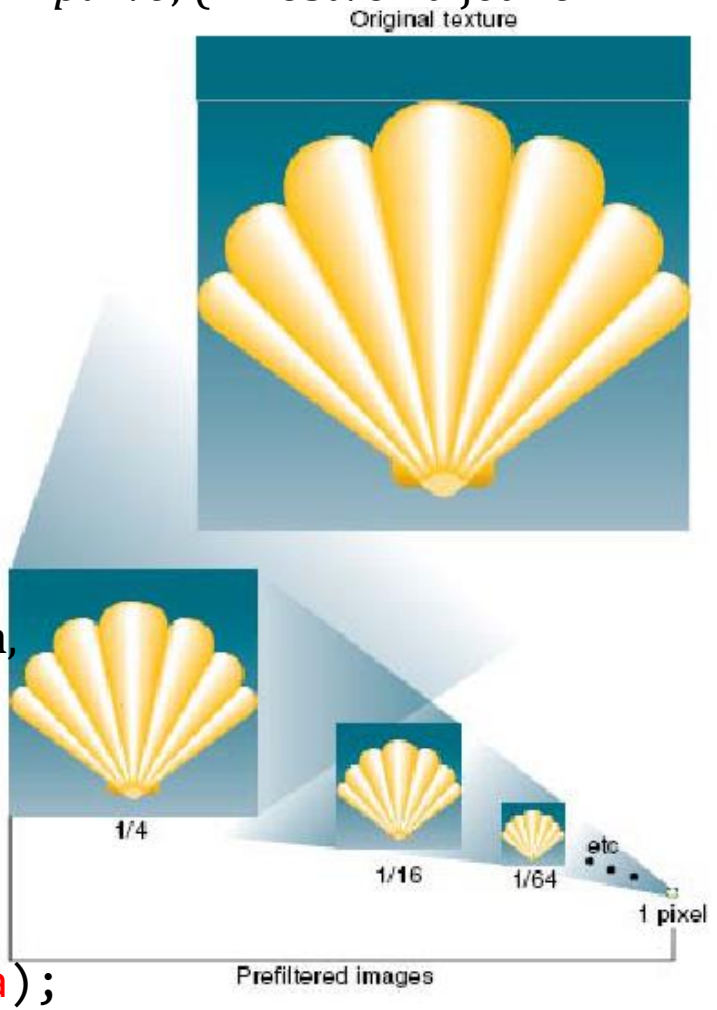
- Da bi se koristile veće teksture nego što to omogućuje OpenGL i konkretna grafička kartica, mapa se može podeliti na više segmenata, koji se zatim učitavaju kao zasebne teksture.
- Prilikom računanja boje piksela, obično se ne uzima u obzir samo jedan teksel (piksel u teksturi), već se računa srednja vrednost nekoliko okolnih teksela (ili neka složenija funkcija).
- Da bi se korektno iscrtavale ivice, tj. da se ne bi videli "šavovi", sa svakim segmentom potrebno je učitati još jedan red (i kolonu) piksela, koji se ne iscrtavaju, već se koriste samo u funkcijama izračunavanja vrednosti piksela.



Višestruki nivoi detaljnosti - Mipmape

- Termin mipmap kreirao je Lance Williams 1983. u svom radu "Pyramidal Parametrics" kao skraćenicu latinske fraze *multim im parvo*, (mnoštvo na jednom mestu).





- Za automatsko generisanje Mipmapa od strane OpenGL-a potrebno je da se ima najdetaljnija mapa, (nivo 0) i ide pozivom funkcije:

```
int gluBuild2DMipmaps(
  GLenum target, GLint components,
  GLint width, GLint height,
  GLenum format, GLenum type, void *data);
```

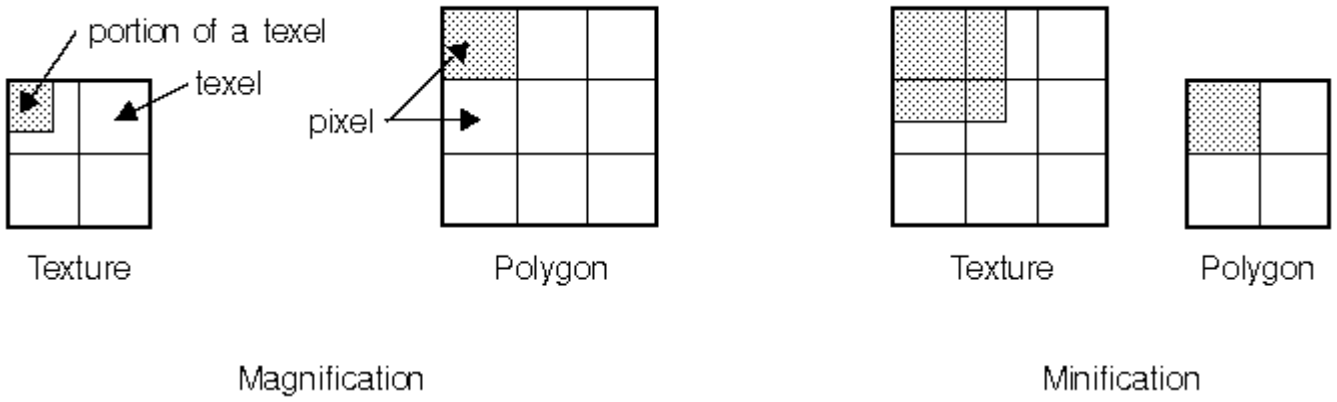
Parametri ove funkcije identični su istim takvim koji se koriste u glTexImage2D.

Zadavanje mipmapa

```
makeImages(); glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
glGenTextures(1, &texName); glBindTexture(GL_TEXTURE_2D, texName);
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_NEAREST);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,
                                        GL_NEAREST_MIPMAP_NEAREST);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, 32, 32, 0,
             GL_RGBA, GL_UNSIGNED_BYTE, mipmapImage32);
glTexImage2D(GL_TEXTURE_2D, 1, GL_RGBA, 16, 16, 0,
             GL_RGBA, GL_UNSIGNED_BYTE, mipmapImage16);
glTexImage2D(GL_TEXTURE_2D, 2, GL_RGBA, 8, 8, 0,
             GL_RGBA, GL_UNSIGNED_BYTE, mipmapImage8);
glTexImage2D(GL_TEXTURE_2D, 3, GL_RGBA, 4, 4, 0,
             GL_RGBA, GL_UNSIGNED_BYTE, mipmapImage4);
glTexImage2D(GL_TEXTURE_2D, 4, GL_RGBA, 2, 2, 0,
             GL_RGBA, GL_UNSIGNED_BYTE, mipmapImage2);
glTexImage2D(GL_TEXTURE_2D, 5, GL_RGBA, 1, 1, 0,
             GL_RGBA, GL_UNSIGNED_BYTE, mipmapImage1);
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
glEnable(GL_TEXTURE_2D);
```

Filtriranje

- Mape tekstura jesu pravougaone kao stepen broja 2, ali nakon mapiranja na poligone i transformacija u screen koordinate, dolazi do distorzije
  - jednom pikselu na ekranu može da odgovara mali deo texela (uvećanje - magnification)
  - jednom pikselu na ekranu može da odgovaraju ogromne kolekcije teksela (umanjenje - minification).
- Način na koji se vrši usrednjavanje ili interpolacija određuje se pozivima sledećih funkcija:
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);



Vrednosti filtera i njihova značenja

- GL_TEXTURE_MAG_FILTER — GL_NEAREST, GL_LINEAR

  GL_TEXTURE_MIN_FILTER — GL_NEAREST,
  GL_NEAREST_MIPMAP_NEAREST,
  GL_NEAREST_MIPMAP_LINEAR,
  GL_LINEAR,
  GL_LINEAR_MIPMAP_NEAREST,
  GL_LINEAR_MIPMAP_LINEAR
- GL_NEAREST – bira se teksel najbliži centru piksela
- GL_NEAREST_MIPMAP_NEAREST – bira se teksel najbliži centru piksela iz odgovarajućeg nivoa mipmape (teksel najbliže veličine sa datim pikselom)
- GL_NEAREST_MIPMAP_LINEAR – uvodi se linearna težinska funkcija 2x2 teksela najbliža centru piksela sa odgovarajućeg nivoa mipmape
- GL_LINEAR – težinsko linearno usrednjivanje 2x2 teksela najbliža centru piksela
- GL_LINEAR_MIPMAP_NEAREST – biraju se dva nivoa mipmape čije veličine teksela najviše odgovaraju veličini piksela na ekranu, a zatim se u njima bira teksel koji je najbliži centru piksela
- GL_LINEAR_MIPMAP_LINEAR – vrši se težinsko usrednjavanje dva susedna nivoa mipmape. Najsloženiji i najbolji metod.

Kreiranje i korišćenje objekata tekstura

- Imenovanje objekta teksture

```
void glGenTextures(GLsizei n, GLuint *textureNames);
```

  - n – broj imena tekstura koje treba generisati
  - textureNames – vektor u koji se smeštaju imena (identifikatori). 0 je rezervisano ime.

- Provera da li je neko ime iskorišćeno vrši se funkcijom:

```
GLboolean glIsTexture(GLuint textureName);
```

  - Ako je ime generisano pomoću glGenTextures, ali nije povezano (bar jedan poziv glBindTexture), ili je obrisana, funkcija vraća GL_FALSE.

- Povezivanje teksture

```
void glBindTexture(GLenum target, GLuint textureName);
```

    - target – npr. GL_TEXTURE_1D ili GL_TEXTURE_2D
    - textureName – ime teksture (generisano glGenTextures funkcijom)
  - Kada se koristi prvi put – kreira se novi objekat teksture i dodeljuje mu se ime.
  - Kada se poziva za prethodno kreirani objekat – objekat teksture postaje aktivan.
  - Kada se pozove za vrednost 0 – OpenGL prestaje da koristi objekte tekstura i vraća se na korišćenje neimenovanih default tekstura.

Funkcije tekstura

- Brisanje objekata tekstura
```
void glDeleteTextures(GLsizei n, const GLuint *textureNames);
```
  - Briše n objekata tekstura imena zadatih vektorom textureNames.
  - Ako se obriše objekat koji je trenutno aktivan (povezan), postaje aktivna default tekstura (kao da je pozvana glBindTexture za 0).
- Tekstura može u potpunosti da zameni boju modela, ali se može koristiti i za modulaciju boje ili se mešati sa bojom objekta. To se definiše funkcijom:
```
void glTexEnv{if} (GLenum target, GLenum pname, TYPE param);
void glTexEnv{if}v(GLenum target, GLenum pname, TYPE *param);
```
  - target - GL_TEXTURE_ENV
  - pname - GL_TEXTURE_ENV_MODE
  - param – može imati jednu od sledećih vrednosti:
    - GL_DECAL – tekstura u potpunosti menja boju fragmenata,
    - GL_REPLACE – za RGB interni format je isti GL_DECAL,
    - GL_MODULATE – boja fragmenta se moduliše teksturom,
    - GL_BLEND – vrednost boje koristi se kao alpha kanal za mešanje,
    - GL_ADD - vrednost boje je suma boja teksture i fragmenta.

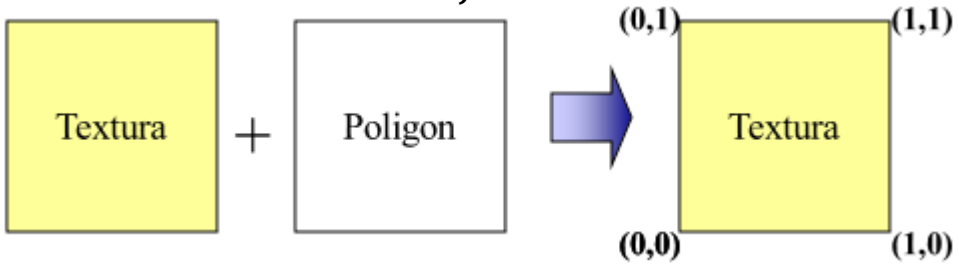
Dodela koordinata tekstura

- Koordinate tekstura određuje koji teksel u teksturi odgovara datom temenu. Između temena vrši se interpolacija.
  - Zadaju se funkcijom `glTexCoord` u okviru glBegin - glEnd bloka.

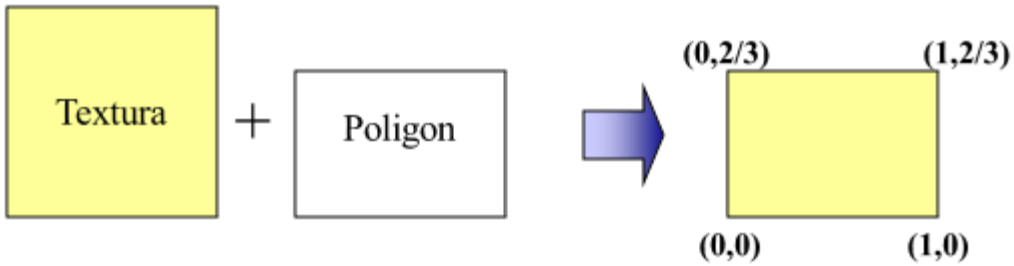
```
void glTexCoord{1234}{sifd} (TYPE s, TYPE t); //s,t,r,q
void glTexCoord{1234}{sifd}v(TYPE *coords);
```

Koordinate tekstura su normalizovane [0.0, 1.0]. Ako se zada vrednost teksturne koordinate veće od 1.0, tada se koristi samo razlomljeni deo.

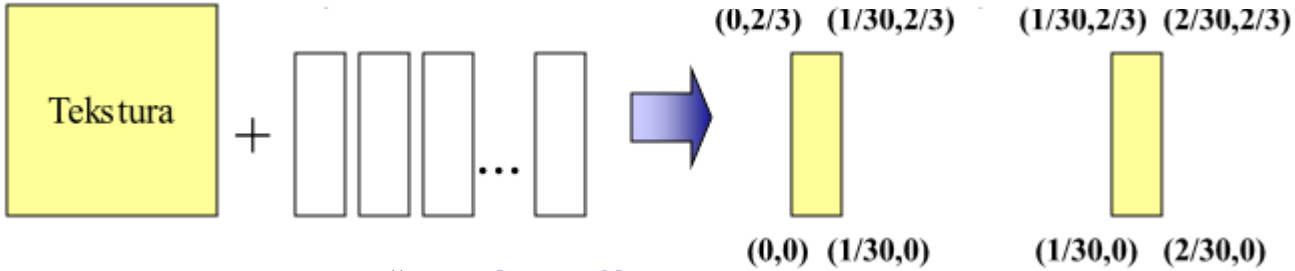
- Čitava tekstura mapira se na pravougaonu površinu



- Tekstura je viša od objekta



- Tekstura se omotava oko objekta

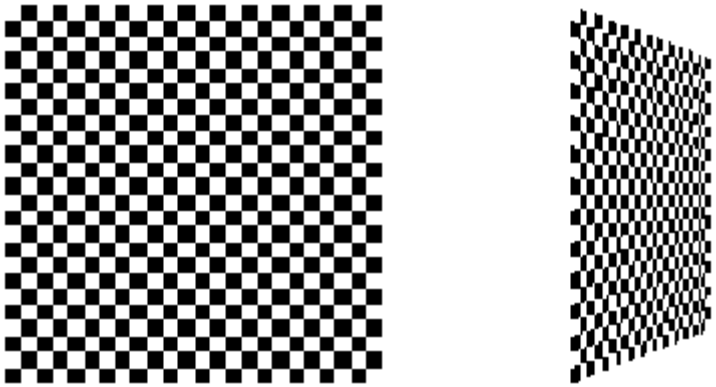


Ponavljanje tekstura

- Ponavljanje tekstura se dobija funkcijom
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

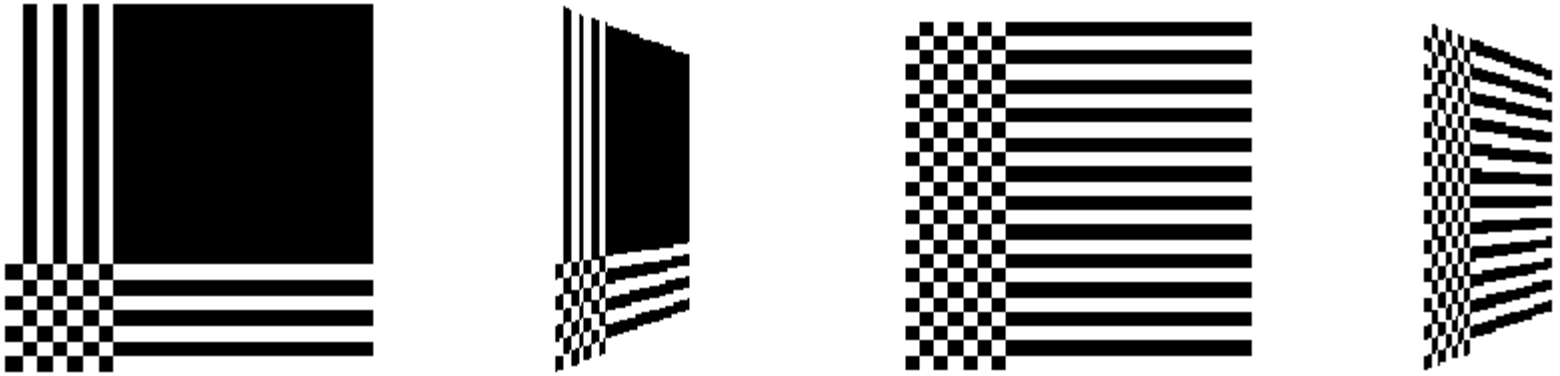
```
glBegin(GL_QUADS);
 glTexCoord2f(0.0, 0.0); glVertex3f(-2.0, -1.0,  0.0);
 glTexCoord2f(0.0, 3.0); glVertex3f(-2.0,  1.0,  0.0);
 glTexCoord2f(3.0, 3.0); glVertex3f( 0.0,  1.0,  0.0);
 glTexCoord2f(3.0, 0.0); glVertex3f( 0.0, -1.0,  0.0);

 glTexCoord2f(0.0, 0.0); glVertex3f( 1.0, -1.0,  0.0);
 glTexCoord2f(0.0, 3.0); glVertex3f( 1.0,  1.0,  0.0);
 glTexCoord2f(3.0, 3.0); glVertex3f( 2.4,  1.0, -1.4);
 glTexCoord2f(3.0, 0.0); glVertex3f( 2.4, -1.0, -1.4);
glEnd();
```



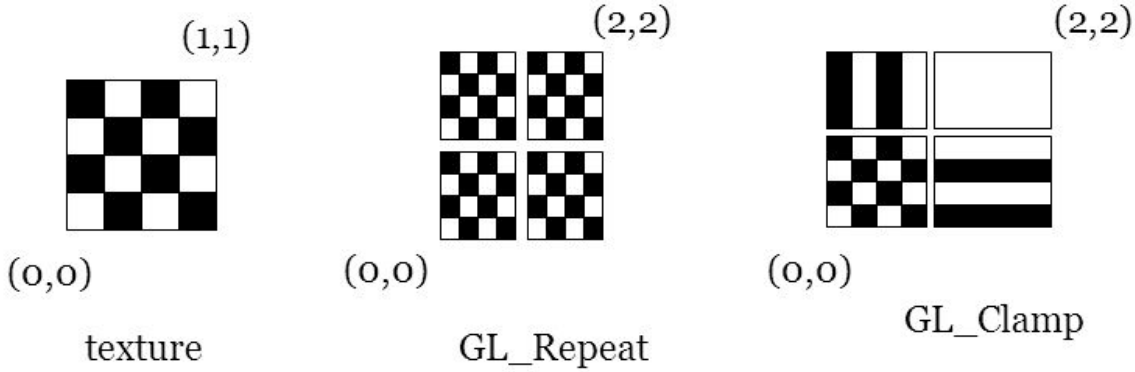
Odsecanje tekstura

- Odsecanje tekstura postiže se funkcijom:
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_ CLAMP);



glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

- Dati su primeri : tekstura je mapirana cela, sa ponavljanjem po s,t te sa odsecanjempo s,t



Primer za teksture 1/6

```
#include "RgbImage.h"  //klasa za ucitavanje bmp
#include "GL/glut.h"  //u mom projektu
const GLuint n=4;
GLuint *textureName = new GLuint[n];

void loadTextureFromFile(GLuint &texid, char *filename)//from bmp
{
  glClearColor (0.0, 0.0, 0.0, 0.0);
  glShadeModel(GL_FLAT);
  glEnable(GL_DEPTH_TEST);
  RgbImage theTexMap( filename );
  glBindTexture(GL_TEXTURE_2D, texid);
  // pixel alignment
  glPixelStorei(GL_UNPACK_ALIGNMENT, 4);
  // postavljanje interpolacije
  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_NEAREST);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
  gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGB,
       theTexMap.GetNumCols(),    theTexMap.GetNumRows(),
       GL_RGB, GL_UNSIGNED_BYTE,  theTexMap.ImageData() );
}
```

Primer za teksture 2/6

```
void initFour( char* filenames[] )//load 4 textures
{
  glGenTextures(4, textureName);
  for ( int i=0; i<4; i++ ) {
    loadTextureFromFile( textureName[i], filenames[i] );// load  tex#i
  }
}

void drawTextureQuad( int i )
{
   glBindTexture(GL_TEXTURE_2D, textureName[i]);

   glBegin(GL_QUADS);
      glTexCoord2f(0.0, 0.0); glVertex3f(-1.0, -1.0, 0.0);
      glTexCoord2f(0.0, 1.0); glVertex3f(-1.0,  1.0, 0.0);
      glTexCoord2f(1.0, 1.0); glVertex3f( 1.0,  1.0, 0.0);
      glTexCoord2f(1.0, 0.0); glVertex3f( 1.0, -1.0, 0.0);
   glEnd();
   glFlush();

}
```

Primer za teksture 3/6

```
void drawScene(void){
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glEnable(GL_TEXTURE_2D);
   glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);

   glPushMatrix();
     glTranslatef( -1.1f, 1.1f, 0.0f );   drawTextureQuad ( 0 );
   glPopMatrix();

   glPushMatrix();
     glTranslatef( 1.1f, 1.1f, 0.0f );    drawTextureQuad ( 1 );
   glPopMatrix();

   glPushMatrix();
     glTranslatef( -1.1f, -1.1f, 0.0f );   drawTextureQuad ( 2 );
   glPopMatrix();

   glPushMatrix();
     glTranslatef( 1.1f, -1.1f, 0.0f );   drawTextureQuad ( 3 );
   glPopMatrix();

   glFlush();   glDisable(GL_TEXTURE_2D);
}
```

Primer za teksture 4/6

```
void resizeWindow(int w, int h)
{
  float viewWidth = 2.2;
  float viewHeight = 2.2;
  glViewport(0, 0, w, h);
  h = (h==0) ? 1 : h;
  w = (w==0) ? 1 : w;
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  if ( h < w ) viewWidth  *= (float)w/(float)h;
  else         viewHeight *= (float)h/(float)w;

  gluOrtho2D( -viewWidth, viewWidth, -viewHeight, viewHeight );

  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
}

void keyboard (unsigned char key, int x, int y)
{
  switch (key) { case 27:exit(0);break;   }
}
```

Primer za teksture 5/6

```
char* filenameArray[4] = {
"slike/WoodGrain.bmp",
"slike/LightningTexture.bmp",
"slike/IvyTexture.bmp",
"slike/RedLeavesTexture.bmp"
};

int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
  glutInitWindowSize(500, 500);
  glutInitWindowPosition(100, 100);
  glutCreateWindow(argv[0]);
  initFour( filenameArray );
  glutDisplayFunc(drawScene);
  glutReshapeFunc(resizeWindow);
  glutKeyboardFunc(keyboard);
  glutMainLoop();
  return 0;
}
```

Primer za teksture - izlaz 6/6