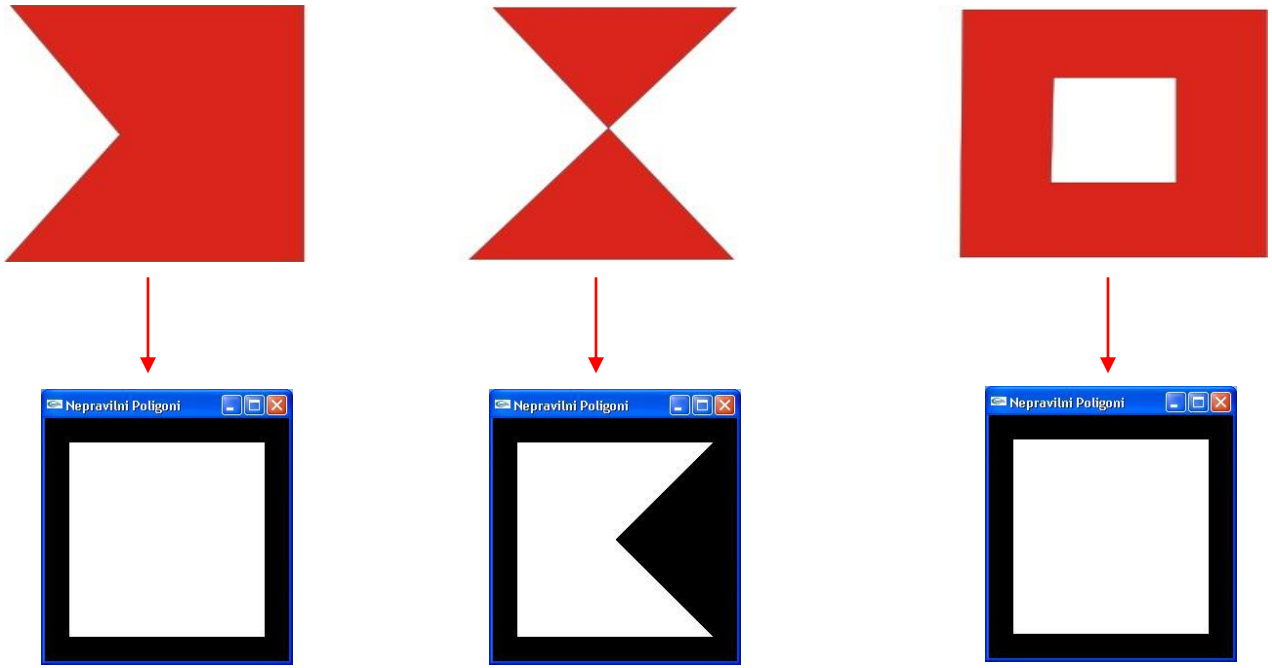
OpenGL – poligoni

- Poligon – oblast ograničena zatvorenom linijom
  - mora biti konveksan i planaran
  - ne sme presecati samog sebe
  - ne sme imati rupu
- Primer nepravilnih poligona i kako ih OpenGL crta (zavisi od implementacije):



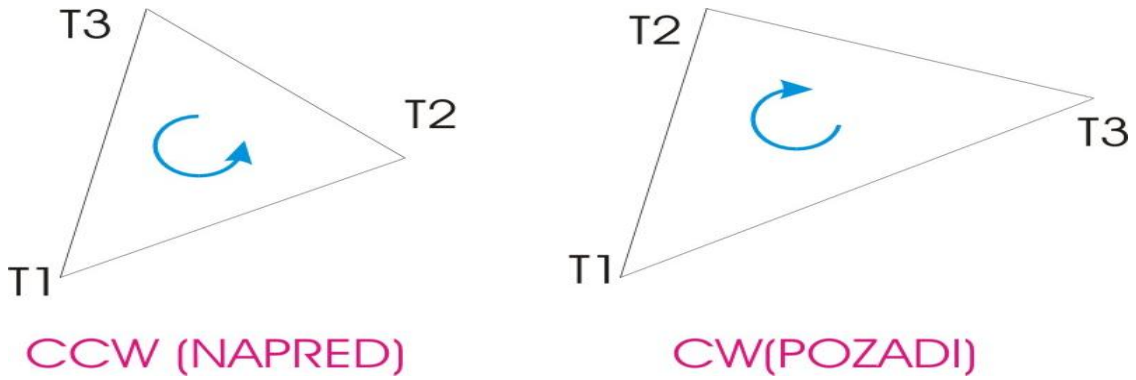
OpenGL – poligoni

- Svaki poligon ima dve lica – prednje i zadnje
- Podrazumevano je da se oba lica crtaju

- **`void glPolygonMode(GLenum face, GLenum mode);`**
  - Specificira kako se crtaju lica poligona
  - **`face`** – lice
    - **GL_FRONT** – prednja strana (lice)
    - **GL_BACK** – zadnja strana (lice)
    - **GL_FRONT_AND_BACK** – obe strane
  - **`mode`** – mod iscrtavanja strane
    - **GL_POINT** – crtaju se samo tačke
    - **GL_LINE** – crtaju se ivice poligona
    - **GL_FILL** – crta se popunjen poligon

OpenGL – poligoni

- Lice poligona se određuje na osnovu zadate orijentacije i redosleda zadavanja temena
- Na primer, neka je orijentacija lica u suprotnom smeru od kazaljki na satu (CCW)



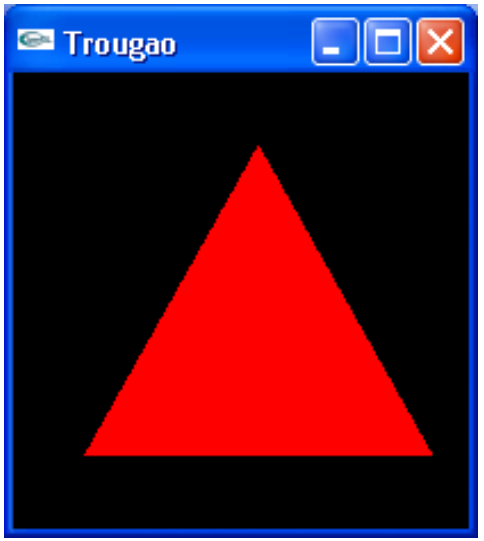
- **`void glFrontFace(GLenum face);`**
  - Određuje orijentaciju poligona
  - **`face`**
    - GL_CW – ako su temena lica zadata u CW smeru
    - GL_CCW – ako su temena lica zadata u CCW smeru

OpenGL – senčenje

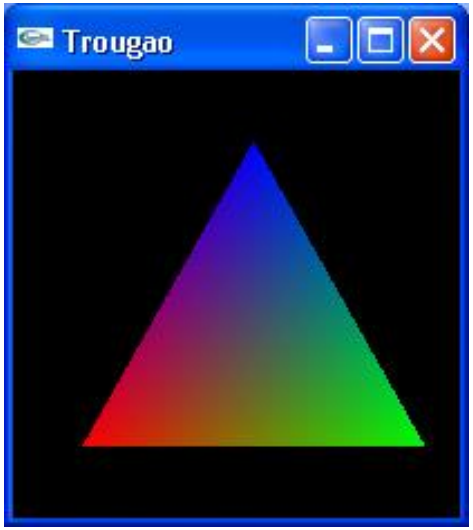
- Linije ili popunjeni poligoni se mogu crtati
  - Uvek jednobojno – **flat shading** (constant shading)
  - Nijansirano – **smooth shading** (koristi se *Gouraud* shading)
- Flat shading
  - Svi pikseli koji pripadaju poligonu imaju istu boju
  - Boja poligona određena je bojom **prvog temena** koja je zadata (boja ostalih temena ne utiče)
- Smooth shading
  - Boja svakog temena se uzima u obzir
  - Boje piksela u poligoni se interpoliraju između boja svih temena kojima je poligon zadat
  - Dva susedna piksela imaju neznatno različitu boju
- **`void glShadeModel(GLenum mode);`**
  - **`mode`**
    - GL_FLAT
    - GL_SMOOTH

OpenGL – senčenje

```
glBegin(GL_POLYGON);
    glColor3f(1.0f, 0.0f, 0.0f);
    glVertex2f(0.0f, 0.0f);
    glColor3f(0.0f, 1.0f, 0.0f);
    glVertex2f(1.0f, 0.0f);
    glColor3f(0.0f, 0.0f, 1.0f);
    glVertex2f(0.5f, 0.866f);
glEnd();
```

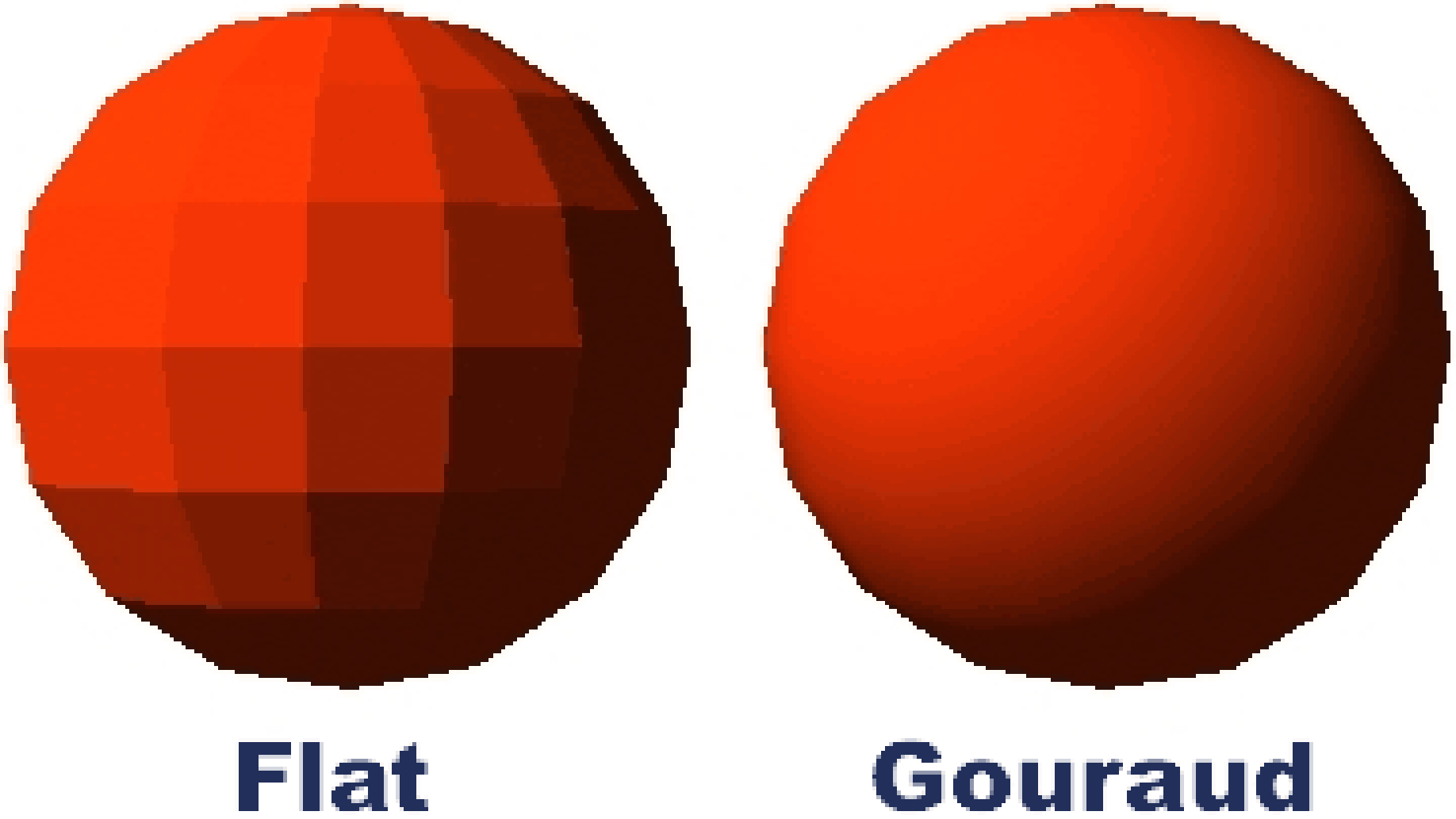


GL_FLAT

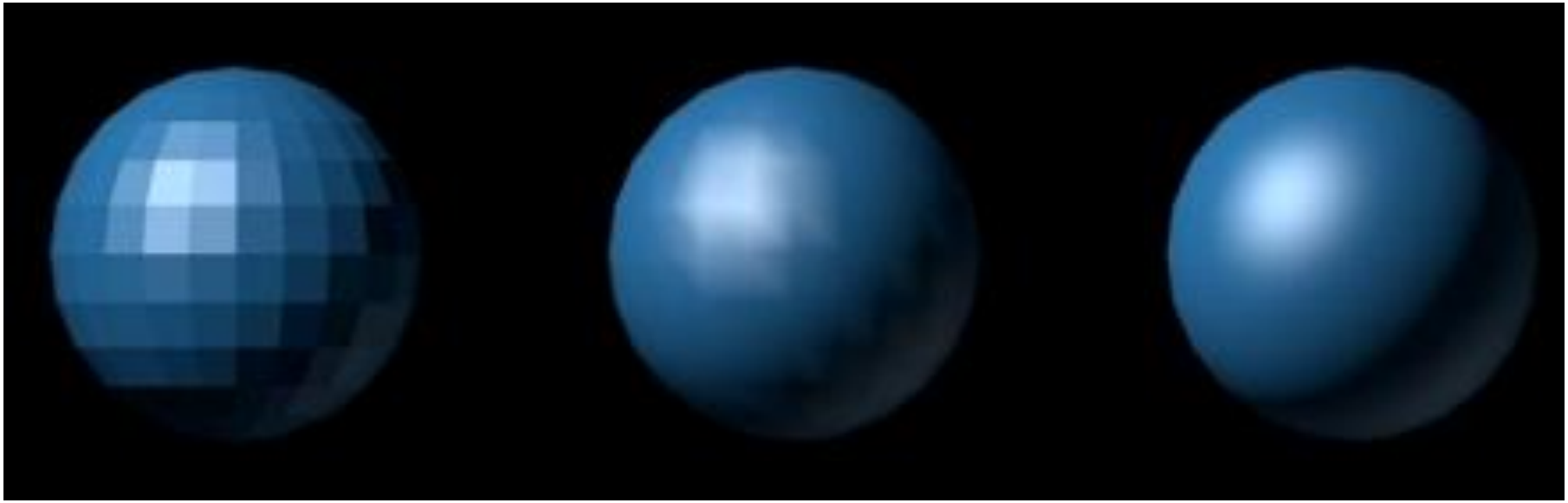


GL_SMOOTH

OpenGL – senčenje



OpenGL – senčenje

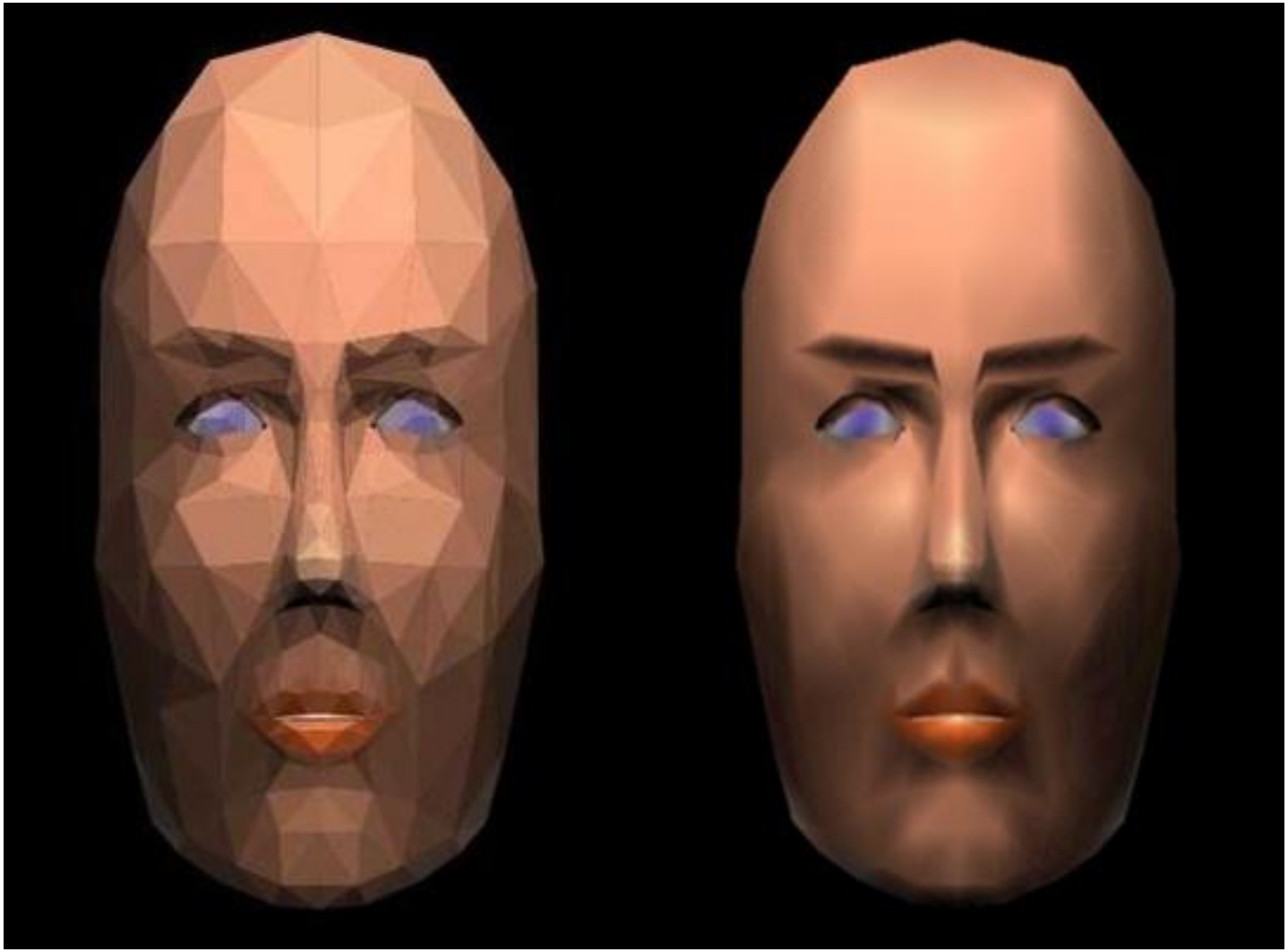


Flat

Gouraud

Phong
NIJE PODRŽANO

OpenGL – senčenje



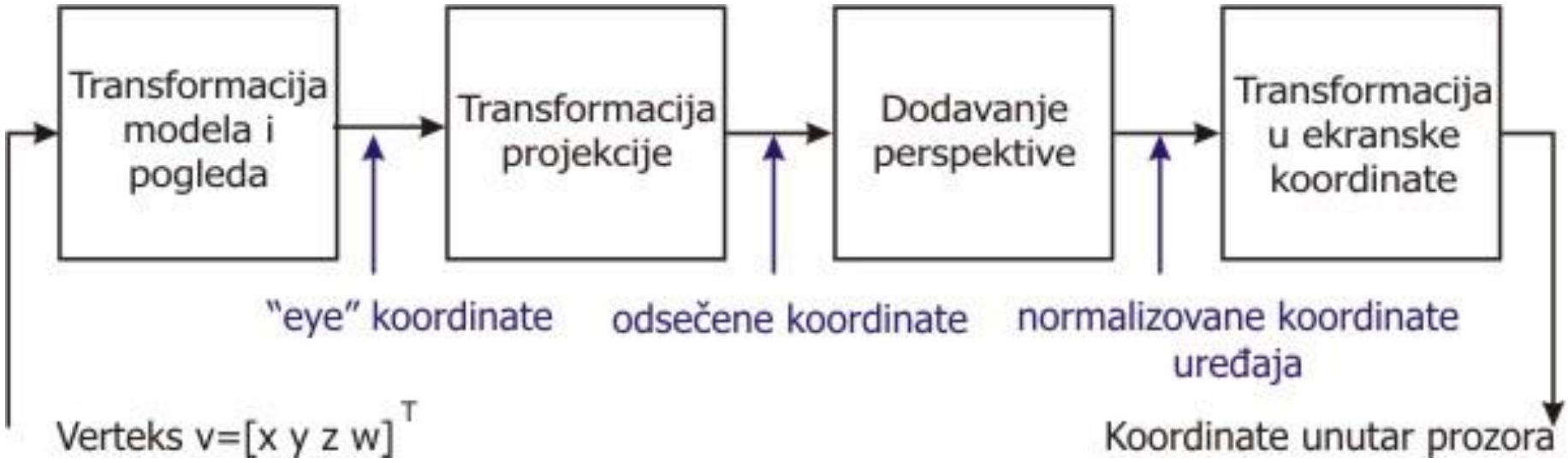
FLAT GOURAUD

OpenGL – transformacije

- Služe za:
  - jednostavno formiranje scene
    - raspoređivanje i skaliranje objekata
    - pozicioniranje kamere
  - zadavanje parametara projekcije
    - ortogonalna
    - sa perspektivom
    - programski definisana
- Koriste se matrice 4x4
- Predefinisane i programski zadate
- Svaka vrsta matrice ima svoj stek

OpenGL – transformacije

- Obrada temena



OpenGL – transformacije

- Primer
  - v – teme koje se transformiše
  - v` – teme posle transformacije
  - M – matrica transformacije

$$v` = \begin{bmatrix} x` \\ y` \\ z` \\ w` \end{bmatrix} = M \cdot v = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

OpenGL – transformacije

- **`void glMatrixMode(GLenum mode);`**
  - Bira aktivnu matricu
  - Parmetar **`mode`**
    - **`GL_MODELVIEW`** – označava matricu za transformaciju modela i pogleda
    - **`GL_PROJECTION`** – označava matricu za transformaciju projekcije
    - **`GL_TEXTURE`** – označava maticu za teksture
  - Sve transformacije utiču samo na aktivnu matricu
  - Koja je matrica trenutno izabrana može se proveriti preko **`glGet()`** sa argumentom **`GL_MATRIXMODE`**
  - Na početku rada automatski je selektovana matrica za transformaciju modela i pogleda

OpenGL – transformacije

- **`void glLoadIdentity(void);`**
  - Postavlja aktivnu matricu na jediničnu

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **`void glLoadMatrix{f d}(const TYPE *M);`**
  - Postavlja vrednost elemenata tekuće matrice na zadatu (M)
  - Primer (definisanje sopstvene projekcije)

```
glMatrixMode(GL_PROJECTION);
glLoadMatrix(mojaProjekcija);
```

OpenGL – transformacije

- **`void glMultMatrix{f d}(const TYPE *M);`**
  - Množi aktivnu matricu zadatom matricom
  - Parametar M
    - 4x4 matrica
    - Zadaje se kao niz od 16 vrednosti (double ili float)
    - matrica kolona
  - Množenje se uvek obavlja sa desne strane
  - Primer
    - C – tekuća matrica transformacije
    - M – matrica kojom se množi
    - Posle ove naredbe biće C ← CM

OpenGL – transformacije

- Primer

```
glMatrixMode(GL_MODELVIEW);
glMultMatrix(P);
glMultMatrix(Q);
glMultMatrix(R);
glBegin(GL_POINTS);
      glVertex3fv(v);
glEnd();
```

- U trenutku crtanja, matica transformacije je PQR
- Crta se transformisani verteks v`=PQRv=P(Q(Rv))
- Dakle, prvo se primenjuje transformacija koja je poslednja zadata

OpenGL – transformacije

- Ponekad je korisno dohvatiti sadržaj matrice:

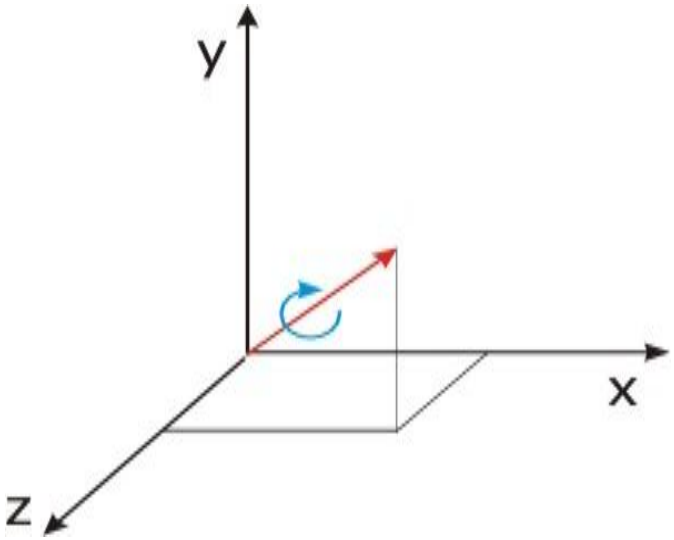
```
void glGetFloatv(GLenum what, GLfloat *m);
```

  - Parametar **`what`**
    - Određuje koja se matrica traži
    - Vrednost parametra **`mode`** kod **`glMatrixMode`**
  - Parametar **`m`**
    - Pointer na niz od 16 elemenata (matrica 4x4)
    - U njega će biti upisana matica
  - Primer (dohvata matricu projekcije)

```
GLfloat m[16];
glGetFloatv(GL_PROJECTION, m);
```

OpenGL – transformacije

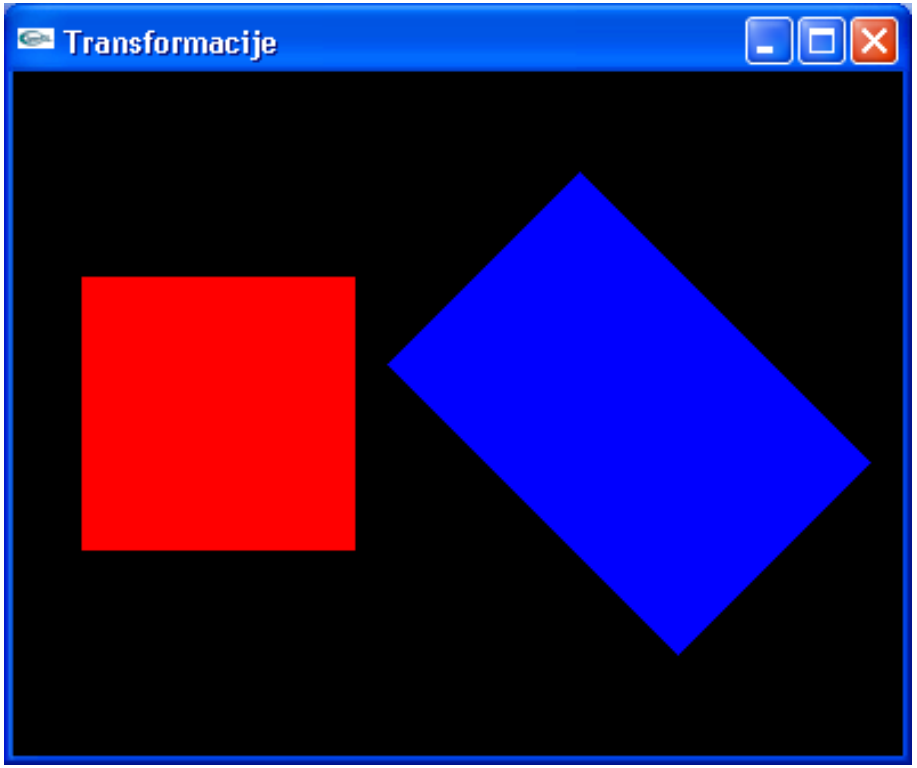
- Jednostavnije – koristiti mehanizam transformacije koordinatnog sistema vezanog za objekat
- **`void glTranslate{f d}(TYPE x, TYPE y, TYPE z);`**
- **`void glRotate{f d}(TYPE angle,`**
  `TYPE x, TYPE y, TYPE z);`
- **`void glScale{f d}(TYPE x, TYPE y, TYPE z);`**



Ugao (**`angle`**) je zadat u stepenima

OpenGL – transformacije

```
void nacrtaj_kvadrat(void) {
    glBegin(GL_POLYGON);
        glVertex2f(-1.0, -1.0);
        glVertex2f( 1.0, -1.0);
        glVertex2f( 1.0,  1.0);
        glVertex2f(-1.0,  1.0);
    glEnd();
}

glClear(GL_COLOR_BUFFER_BIT);
glLoadIdentity();
glColor3f(1.0, 0.0, 0.0);
nacrtaj_kvadrat();
glColor3f(0.0, 0.0, 1.0);
glTranslatef(3.0, 0.0, 0.0);
glRotatef(45.0, 0.0, 0.0, 1.0);
glScalef(1.0, 1.5, 1.0);
nacrtaj_kvadrat();
```



OpenGL – transformacije

- OpenGL čuva sve matrice na stekovima
- Za svaku vrstu transformacije postoji poseban stek
- Primena neke transformacije modifikuje
  samo matricu koja je na vrhu odgovarajućeg steka
- **`glLoadMatrix`**, **`glMultMatrix`** i **`glLoadIdentity`**
  utiču samo na vrh steka
- Postoje dve naredbe za manipulaciju stekom
  - **`glPushMatrix()`** – pomera sve matrice na steku
    jedan nivo naniže; matrica na vrhu steka se duplira
  - **`glPopMatrix()`** – skida matricu sa vrha steka
    i uništava je; ukoliko je stek imao samo jednu matricu → greška

OpenGL – transformacije

- Dubina steka se može saznati pomoću

```
glGetIntegerv(GL_MODELVIEW_STACK_DEPTH, &depth);
glGetIntegerv(GL_PROJECTION_STACK_DEPTH, &depth);
```

- Maksimalna dubina steka se može saznati pomoću

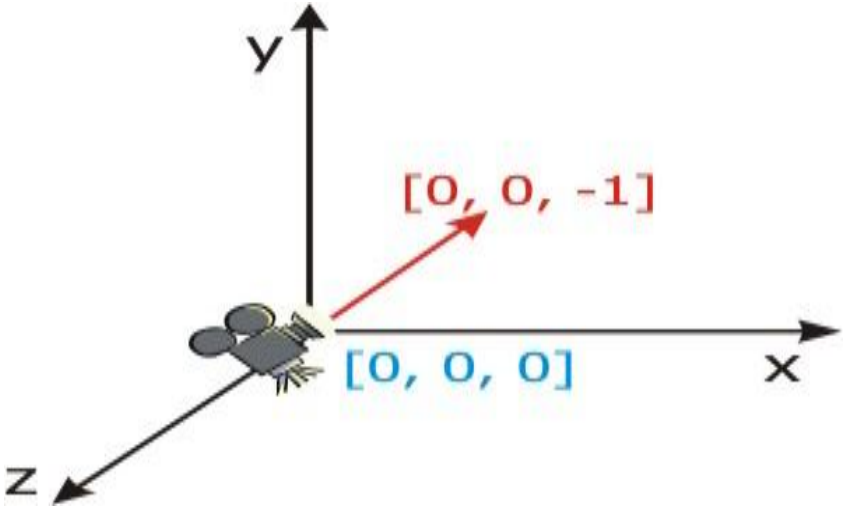
```
glGetIntegerv(GL_MAX_MODELVIEW_STACK_DEPTH, &depth);
glGetIntegerv(GL_MAX_PROJECTION_STACK_DEPTH, &depth);
```

OpenGL – transformacije pogleda

- Kamera se može pozicionirati na dva načina
  - Pomeranjem svih objekata u sceni (lokacija kamere fiksna – transformacija modela)
  - Pomeranjem kamere (transformacija pogleda)
- Pomeranje kamere
  - Ekvivalentno je pomeranju svih objekata u suprotnom pravcu
- Transformacije pogleda treba da se izvrše pre transformacija modela

OpenGL – transformacije pogleda

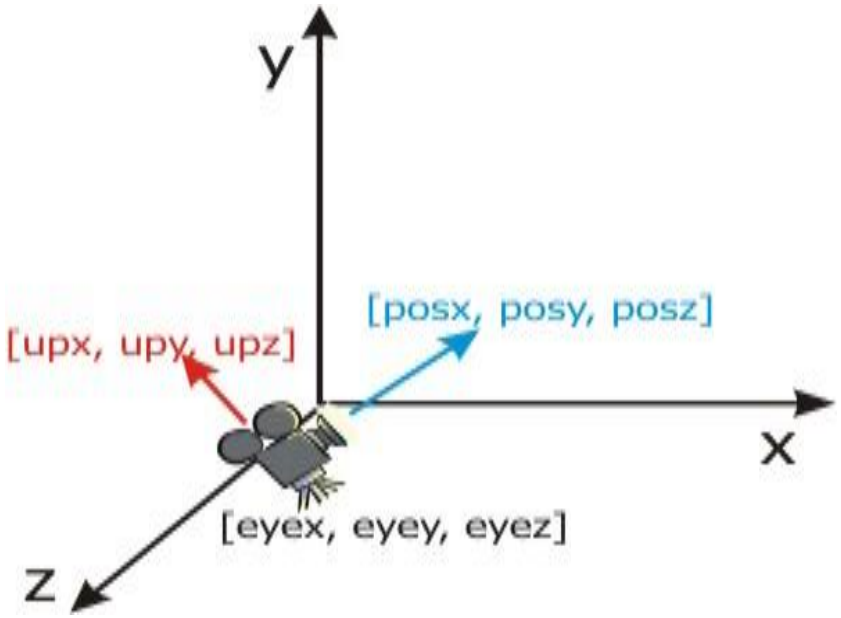
- Kamera “gleda” u neku tačku
  - podrazumeva da je kamera orijentisana u pravcu vektora koji čine pozicija kamere i posmatrana tačka
- Podrazumevani položaj kamere
  - Pozicija kamere je [0, 0, 0]
  - Kamera “gleda” u tačku [0, 0, -1]



OpenGL – transformacije pogleda

- Jednostavno pozicioniranje kamere

```
void gluLookAt(
       GLdouble eyex, GLdouble eyey, GLdouble eyez,
       GLdouble posx, GLdouble posy, GLdouble posz,
       GLdouble upx, GLdouble upy, GLdouble upz
      );
```

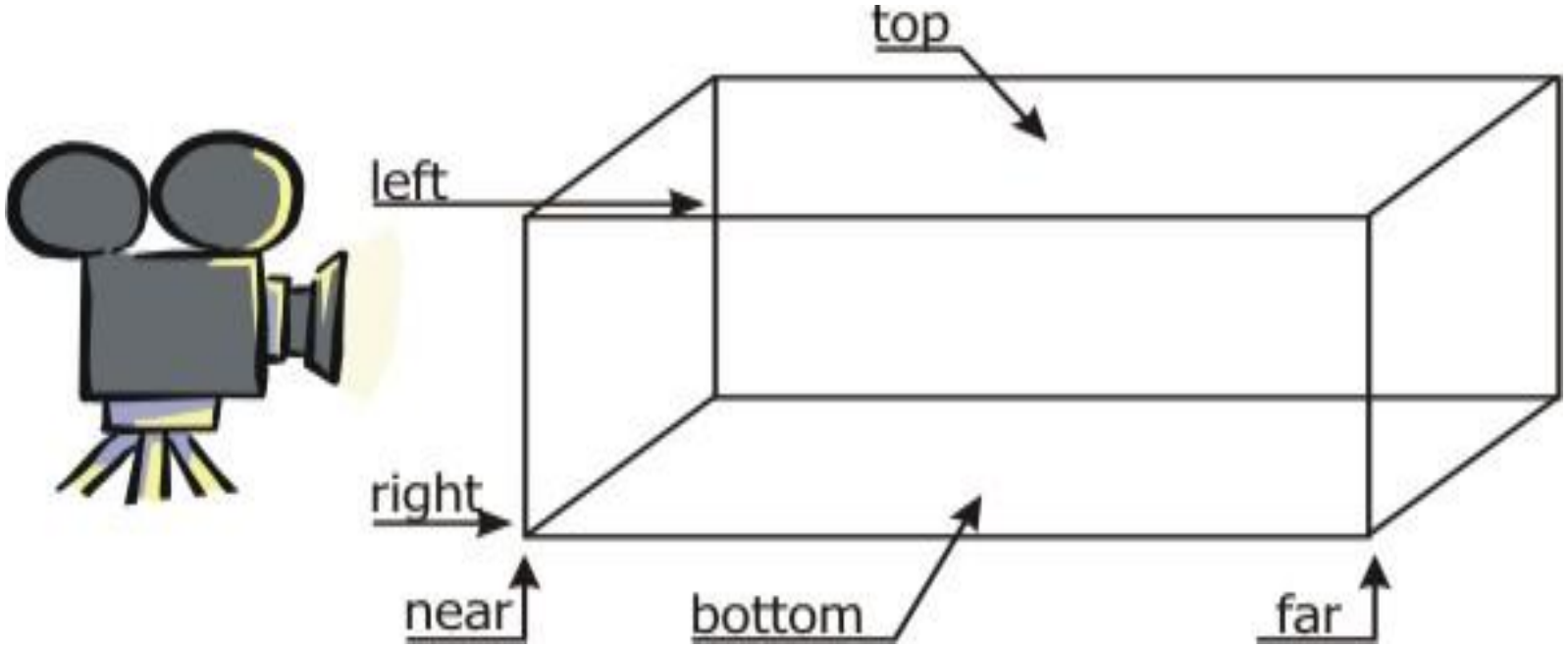


OpenGL – transformacije projekcije

- Definišu vidljivi prostor (eng. *view frustum*)
  - Objekti unutar prostora se crtaju
  - Objekti koji su van prostora se odesecaju
- Određuju kako se objekti projektuju na ekran
- Transformacije projekcije mogu biti
  - Ortografska (paralelna) projekcija
  - Projekcija sa perspektivom
  - Korisnički definisana projekcija
- Mora se izabrati matrica projekcije: **`glMatrixMode(GL_PROJECTION);`**
- Pomoću naknadnih rotacija i translacija moguće je promeniti orijentaciju prostora za odsecanje

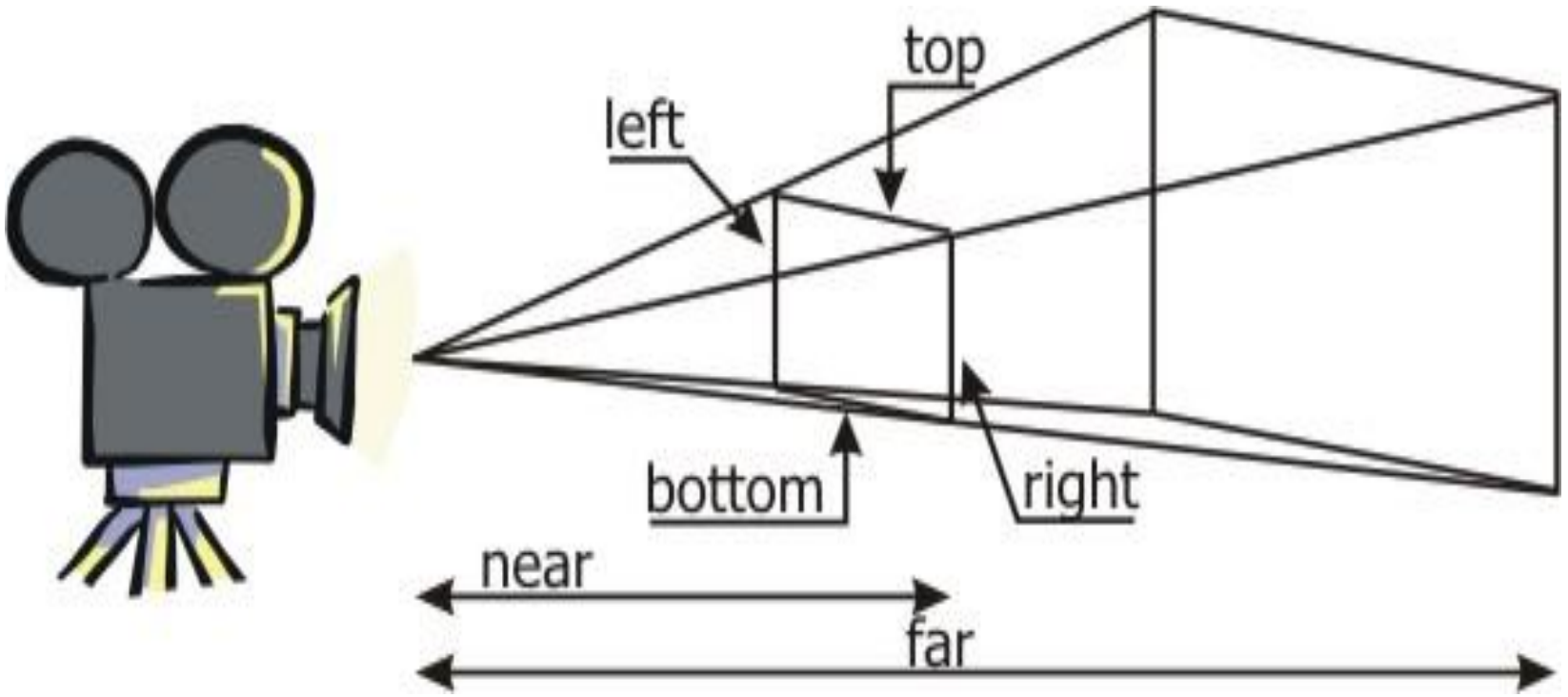
OpenGL – transformacije projekcije

- **`void glOrtho(GLdouble left, GLdouble right,`**
  `GLdouble bottom, GLdouble top,`
  `GLdouble near, GLdouble far`
  `);`
  - Postavlja paralelnu projekciju
  - Parametri određuju prostor za odsecanje



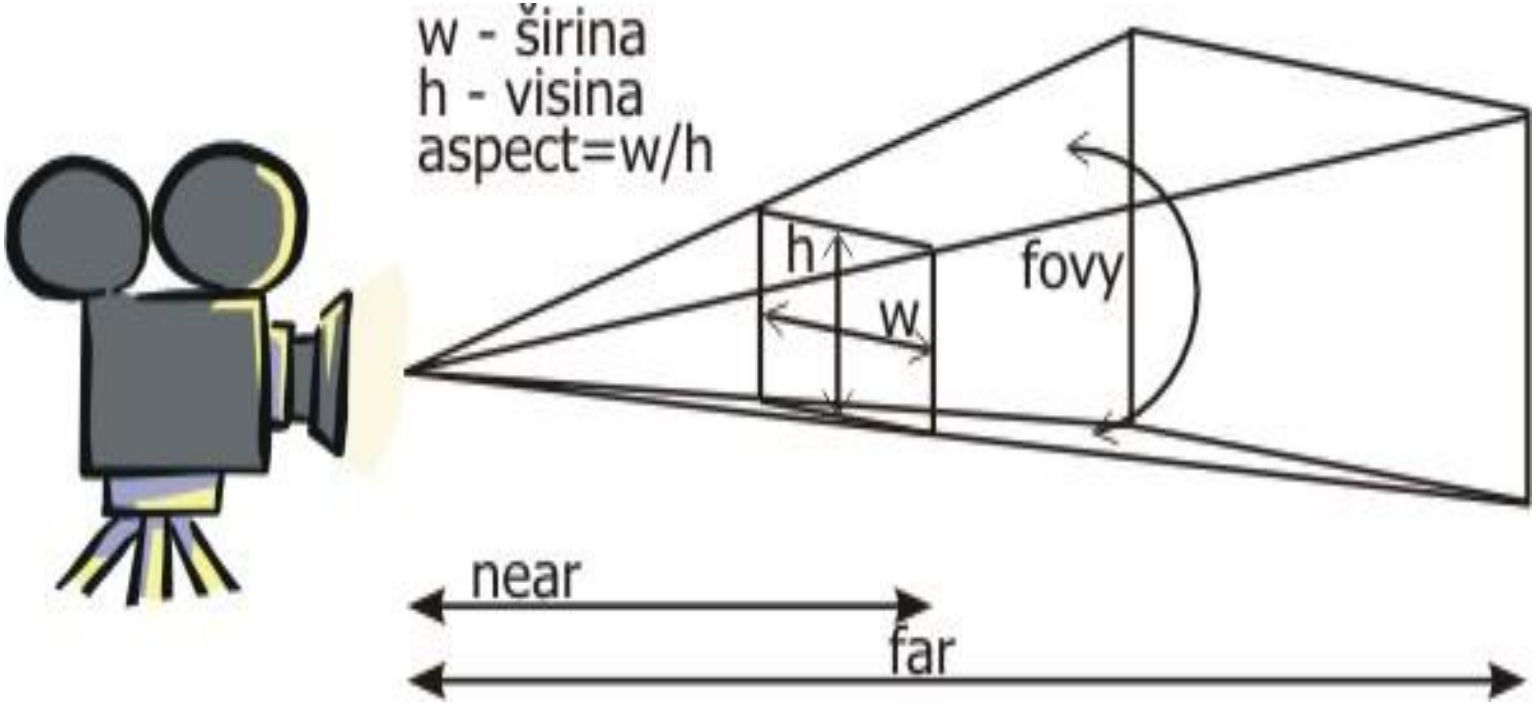
OpenGL – transformacije projekcije

- **`void glFrustum(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far);`**
  - Postavlja perspektivnu projekciju
  - Parametri određuju prostor za odsecanje

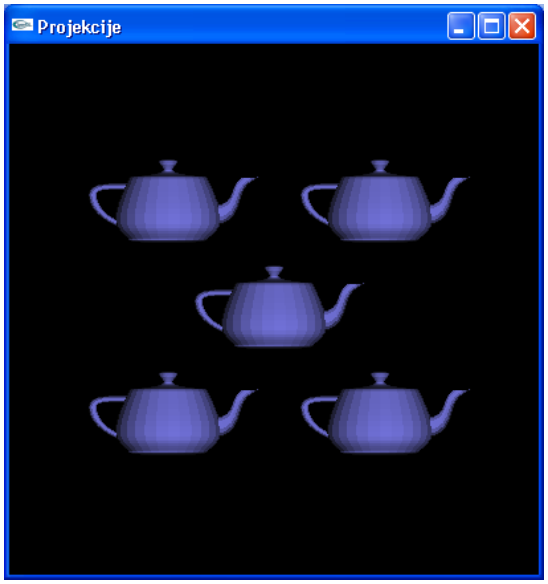


OpenGL – transformacije projekcije

- **`void gluPerspective(GLdouble fovy,`**
  `GLdouble aspect,`
  `GLdouble near,`
  `GLdouble far);`
  - Postavlja perspektivnu projekciju
  - Projekcija je uvek simetrična



OpenGL – transformacije projekcije




```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-5, 5, -5, 5, -10, 10);
// Centralni cajnik, z = -3.0
glTranslatef(0.0, 0.0, -3.0);
glutSolidTeapot(1.0);
// Gornji desni cajnik, z = -5.0
glTranslatef(2.0, 2.0, -2.0);
glutSolidTeapot(1.0);
// Gornji levi cajnik, z = -6.0
```


```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(-2, 2, -2, 2, 1.0, 10);
glTranslatef(-4.0, 0.0, -1.0);
glutSolidTeapot(1.0);
// Donji levi cajnik, z = -7.0
glTranslatef(0.0, -4.0, -1.0);
glutSolidTeapot(1.0);
// Donji desni cajnik, z = -8.0
glTranslatef(4.0, 0.0, -1.0);
glutSolidTeapot(1.0);
```

OpenGL – prikazni prozor (viewport)

- Cela scena se crta samo unutar tog prostora
- Taj prostor je uvek pravougaonog oblika
- Njegova veličina je izražena
  u ekranskim koordinatama
- Zadaje se relativno u odnosu na donji levi ugao ekrana
  (tj. sistemskog prozora)
- Podrazumevano je da prikazni prozor
  zauzima ceo sistemski prozor

OpenGL – prikazni prozor (viewport)

```
void glViewport(GLint x, GLint y,
                GLsizei width, GLsizei height);
```

- (x, y) – koordinate donjeg levog ugla
- (width, height) – dimenzije prostora

– Prostor je pravougaonik određen zadatim parametrima

– Oprez!

```
gluPerspective(fovy, 1.0, near, far);
glViewPort(0, 0, 400, 200);
```

Viewport je dva puta širi nego duži Projekciona ravan je iste širine i dužine

Slika će biti izobličena

OpenGL – baferi

- Hardverska podrška/implementacija
- Standard ne zahteva postojanje svih bafera

naziv	opis
z – buffer	sadrži udaljenost svakog piksela na ekranu po z – osi
alpha buffer	sadrži providnost svakog piksela
accumulation buffer	pamćenje i kombinovanje slika (na pr. *antialiasing*)
color buffer	čuva podatke o boji (RGB mod)
index buffer	čuva podatke o boji (u indeksnom modu)
stencil buffer	ima više primena (na primer - da spreči crtanje na određenom delu ekrana)

OpenGL – baferi

- Postavljanje vrednosti kojom se “čisti” bafer:
  - boja (color buffer)
    - **glClearColor(GLclampf r, GLclampf g, GLclampf b, GLclampf alpha)**
  - dubina (z- buffer)
    - **glClearDepth(GLclampd depth)**
  - stensil
    - **glClearStencil(GLint s)**
  - akumulacija
    - **glClearAccum(GLclampf r, GLclampf g, GLclampf b, GLclampf alpha)**
  - indeks
    - **glClearIndex(GLfloat index)**

OpenGL – baferi

- glClear(GLint mask)
  - mask je bit-maska koja određuje koji baferi se čiste

bafer	maska
color	**GL_COLOR_BUFFER_BIT**
depth	**GL_DEPTH_BUFFER_BIT**
stencil	**GL_STENCIL_BUFFER_BIT**
accumulation	**GL_ACCUM_BUFFER_BIT**

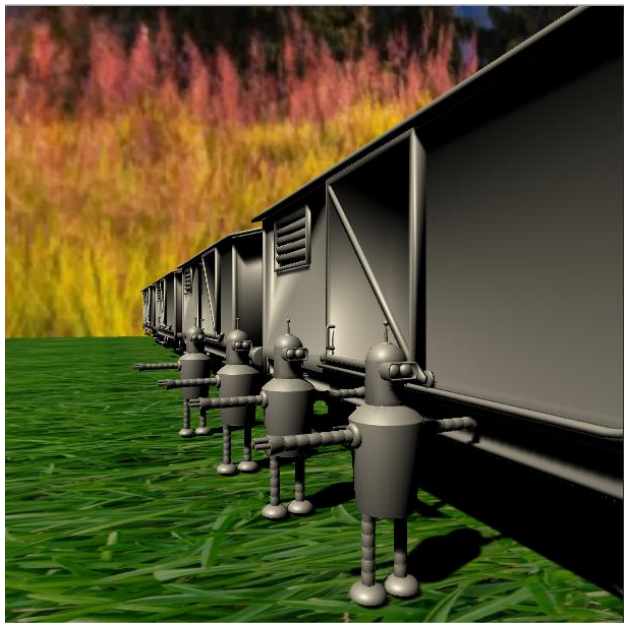
Primer :

```
glClearColor(0.0, 0.0, 0.0, 0.0);
glClearDepth(0.0);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

Bafer slike (Frame buffer)

- Konceptualni kontejner
  - kolor bafer (RGBA = 32 bpp)
  - bafer dubine (Z-buffer, depth buffer) (24 bpp)
  - stensil bafer (stencil buffer) (8 bpp)

} 32 bpp

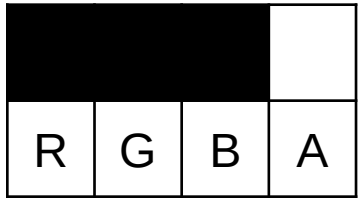


Kolor bafer

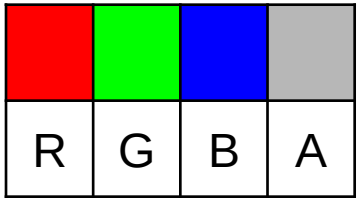
Bafer dubine

Kolor bafer

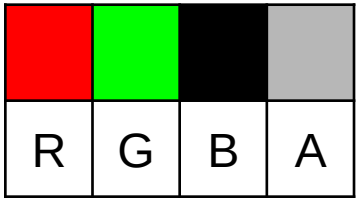
- RGB formira boju (aproksimacija)
- A – dodatni atribut (prozirnost, ...)
- Ograničena kontrola
  - omogućavanje kanala za pisanje
  - Primer: omogući RGA, spreči B
  - `glColorMask(GL_TRUE, GL_TRUE, GL_FALSE, GL_TRUE)`



Trenutni sadržaj



Novi podaci

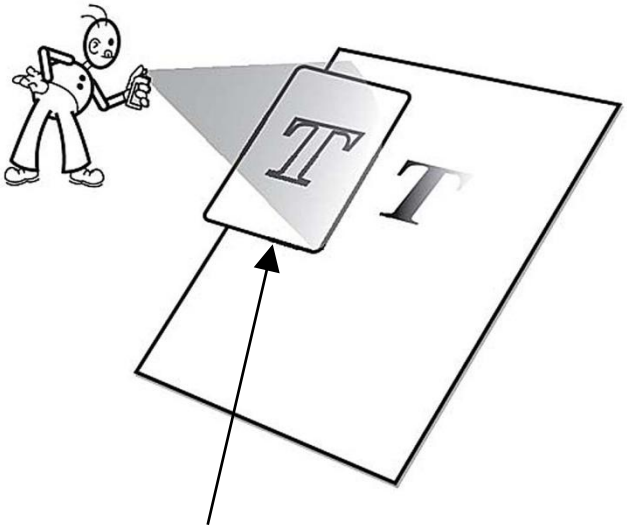


Novi sadržaj

Bafer dubine

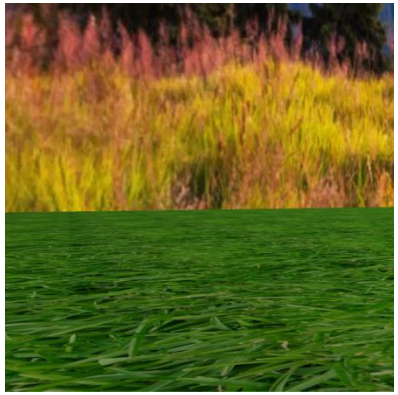
- Automatski (hardver, bez intervencije programera)
- Ograničena kontrola:
  - (ne) pisanje u bafer (glDepthMask(true / false))
  - Test funkcija (`glEnable / glDisable  GL_DEPTH_TEST`)
    (`Less | Greater | LEqual | GEqual | Equal | Not Equal | Always`)

Stensil bafer

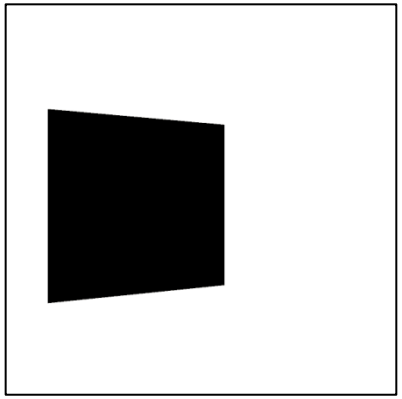


Ovo je “stensil”

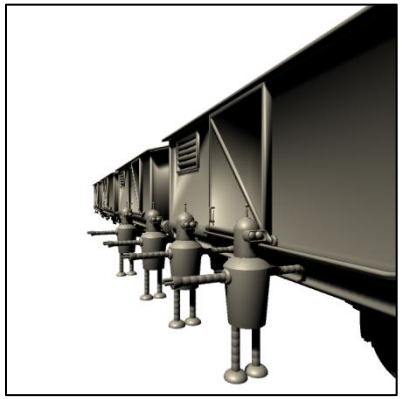
Definicija stensila: tanak karton (...) sa urezanim obrascem, korišćen da se formira specifičan oblik na podlozi ispod.



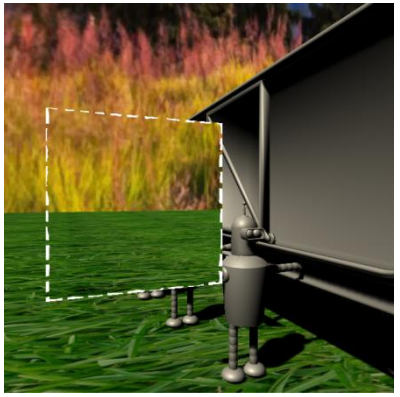
Tekući kolor bafer



Stensil



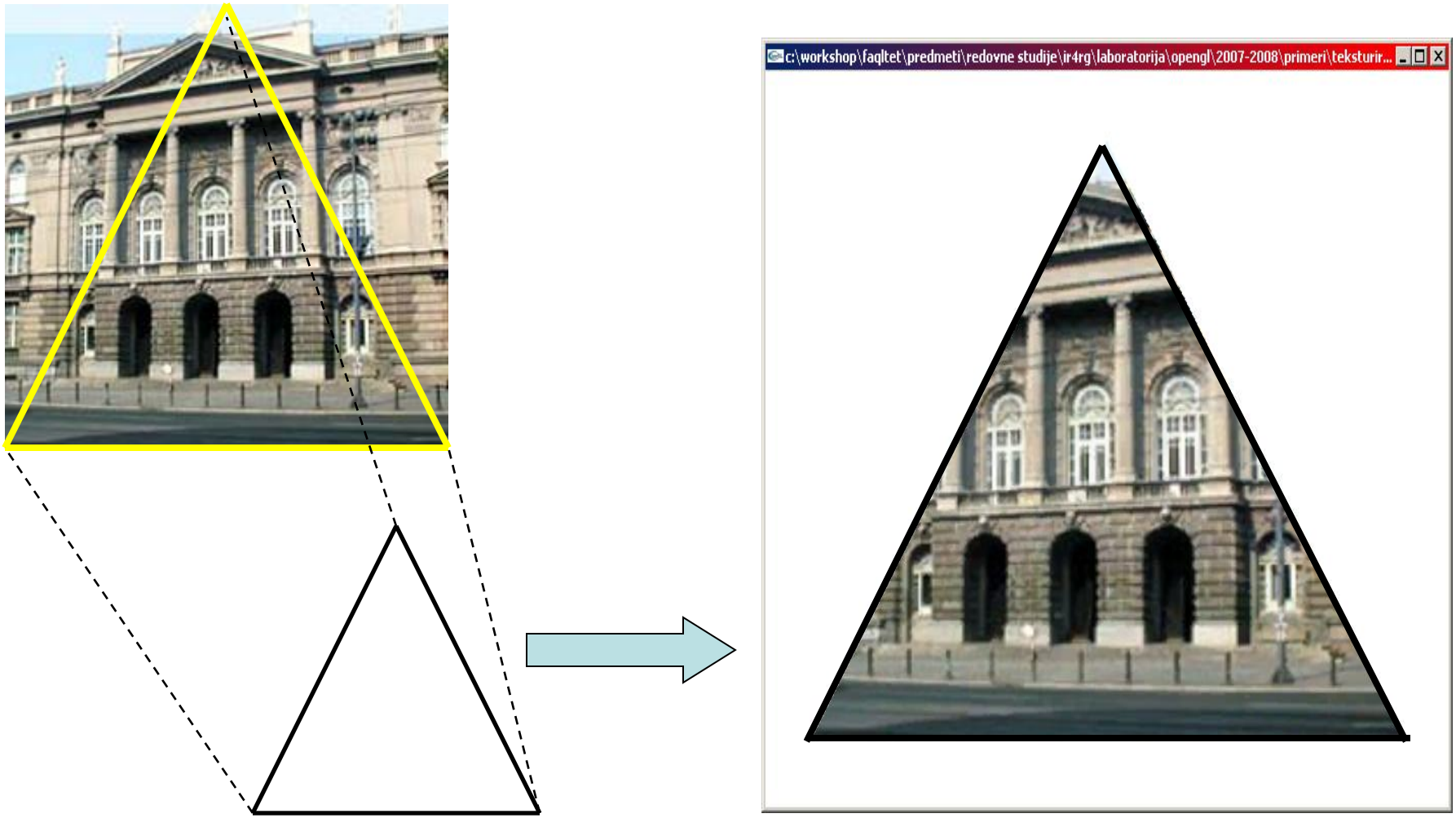
Novo crtanje



Rezultat

Jedna od mogućih primena: **shadow volumes**

OpenGL: Primena tekstura



Primena tekstura
- terminologija -

- Piksel (pixel) : kovanica nastala od Picture Cell ili Picture Element. Označava najsitniji (nedeljivi) element slike prikazivača

- Teksel (texel): kovanica nastala od Texture Cell ili Texture Element, po uzoru na piksel. Označava najsitniji (nedeljivi) deo teksture, koja se često koristi kao sinonim za digitalnu sliku

- Mapiranje teksture (texture mapping): proces popunjavanja unutrašnjosti poligona teksturom. Tokom ovog procesa, jedan teksel može da se preslika na jedan ili više piksela, ali može i obrnuto – da se više teksela preslika na jedan piksel

- MIP-mapiranje (MIP-mapping): potiče od reči MIP-mapa (ili mipmapa), gde je MIP akronim sa latinskog jezika multum in parvo, što u slobodnom prevodu znači "mnogo u malom". Tehnika kojom se ubrzava proces mapiranja tekstura kada se više teksela mapira na jedan piksel, uz izbegavanje neželjenih vizuelnih artefakata

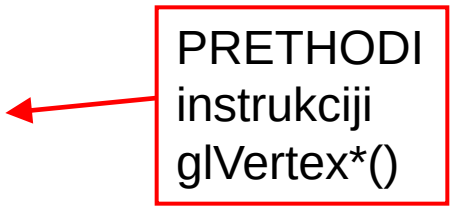
Primena tekstura

- Tekstura je uzorak kojim se popunjava unutrašnjost poligona
- Najčešće se zadaje u vidu 1D ili 2D digitalne slike (bitmape ili piksmape)
- OpenGL podržava nekoliko različitih formata za teksture
  - GL_RGB, GL_RGBA, GL_LUMINANCE,...
  - GL_EXT_BGR, ...
- OpenGL podržava veliki broj formata za interno skladištenje teksture

Primena tekstura

- Koordinata u prostoru tekstura se zadaje instrukcijom **`glTexCoord*(...)`**
- Primer:

```
glBegin(GL_TRIANGLES);
      glTexCoord2d(0, 0);
      glVertex2i(-1, -1);
      glTexCoord2d(1, 0);
      glVertex2i(1, -1);
      glTexCoord2d(0.5, 1);
      glVertex2i(0, 1);
glEnd();
```



Primena tekstura

- Zadata koordinata može da bude van opsega [0, 1]. Može da se definiše način preslikavanja u tom slučaju (CLAMP, WRAP, MIRROR)

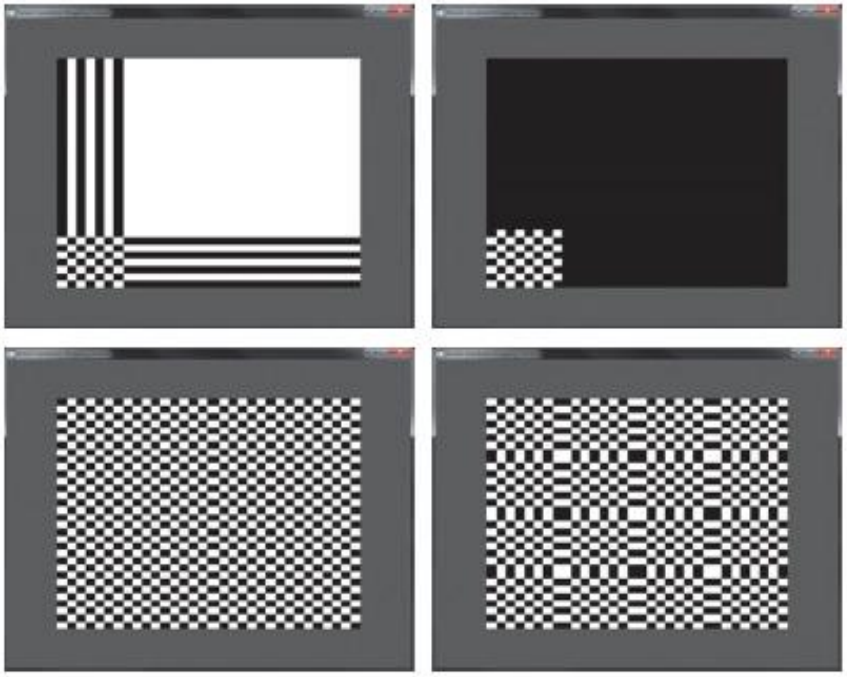
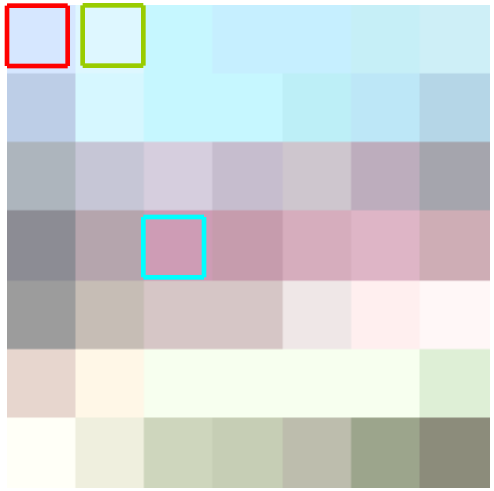


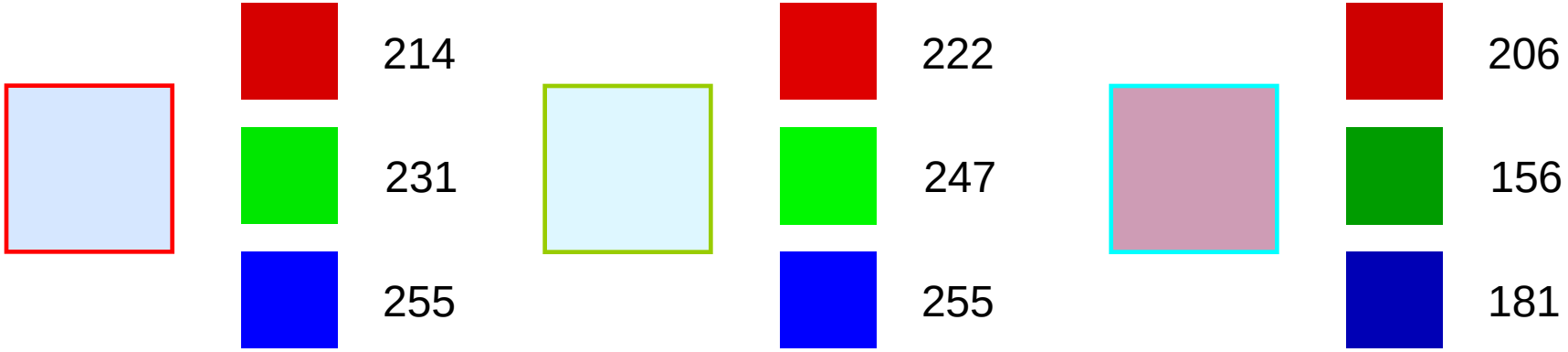
Figure 6.6 Effect of different texture wrapping modes (GL_CLAMP_TO_EDGE (top left), GL_CLAMP_TO_BORDER (top right), GL_REPEAT (bottom left), and GL_MIRRORED_REPEAT (bottom right).)

Primena tekstura

- GL_RGB

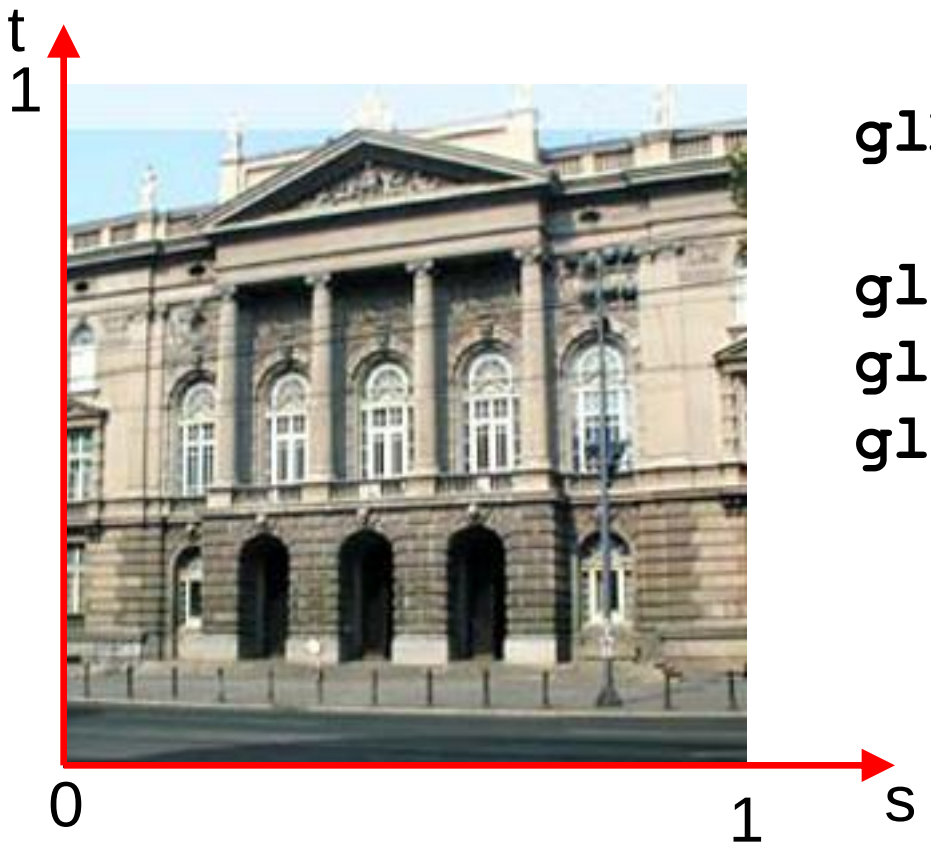


214	231	255	222	247	255	...
...						
...						
...						206
...						



Primena tekstura

- Tekstura poseduje svoj koordinatni prostor (odnosno matricu transformacije)




```
glMatrixMode(GL_TEXTURE)

glScale*()
glTranslate*()
glRotate*()
```

Primena tekstura

- OpenGL posmatra teksture kao "imenovane" objekte
- Pravljenje objekta teksture (ili tekstura)
  - `void glGenTextures(GLsizei n,GLuint *textureNames);`
- Brisanje objekta teksture (ili tekstura)
  - `void glDeleteTextures(GLsizei n,`
    `const GLuint *textureNames);`
- Aktiviranje objekta teksture za korišćenje
  - `void glBindTexture(GLenum target,`
    `GLuint textureName);`
  - `target = { GL_TEXTURE_1D, ..._2D } (OpenGL 1.1)`
  - ne može da se koristi između **`glBegin()`** i **`glEnd()`**

Primena tekstura

- Zadavanje sadržaja objektu teksture
  - **`void glTexImage1D(GLenum target, GLint level, GLint internalFormat, GLsizei width, GLint border, GLenum format, GLenum type, const GLvoid *pixels);`**
  - **`void glTexImage2D(GLenum target, GLint level, GLint internalFormat, GLsizei width, GLsizei height, GLint border, GLenum format, GLenum type, const GLvoid *pixels);`**
- Parametrom **`type`** se precizira tumačenje podataka zadatih sa **`pixels`**

Primena tekstura

- Zadavanje parametara aktivnom objektu teksture
  - **`void glTexParameter{if}(GLenum target, GLenum pname, TYPE param);`**
  - **`void glTexParameter{if}v(GLenum target, GLenum pname, TYPE *param);`**

Naziv parametra (OpenGL 1.1)	Vrednost
`GL_TEXTURE_WRAP_S`	`GL_CLAMP, GL_REPEAT`
`GL_TEXTURE_WRAP_T`	`GL_CLAMP, GL_REPEAT`
`GL_TEXTURE_MAG_FILTER`	`GL_NEAREST, GL_LINEAR`

`GL_TEXTURE_MIN_FILTER`	`GL_NEAREST, GL_LINEAR,`
	`GL_NEAREST_MIPMAP_NEAREST,`
	`GL_NEAREST_MIPMAP_LINEAR,`
	`GL_LINEAR_MIPMAP_NEAREST,`
	`GL_LINEAR_MIPMAP_LINEAR`

`GL_TEXTURE_BORDER_COLOR`	`Četiri vrednosti u opsegu 0-1`
`GL_TEXTURE_PRIORITY`	`Vrednost 0-1`

Primena tekstura

- Jedinica za teksturiranje
  - deo OpenGL sistema koji vrši proračune potrebne za crtanje svakog elementa teksture
  - može se konfigurisati
  - jedna jedinica može da radi nad jednom teksturom
  - OpenGL implementacija mora da ima najmanje jednu jedinicu
- Primena rezultata iz jedinice za teksturiranje preko TexEnv:
  - `void glTexEnv{if}(GLenum target, GLenum pname, TYPE param);`
  - `void glTexEnv{if}v(GLenum target, GLenum pname, TYPE *param);`
  - `target – mora biti GL_TEXTURE_ENV`

pname	param
`GL_TEXTURE_ENV_MODE`	`GL_DECAL, GL_REPLACE, GL_MODULATE, GL_BLEND`
`GL_TEXTURE_ENV_COLOR`	`Boja (RGB, RGBA)`

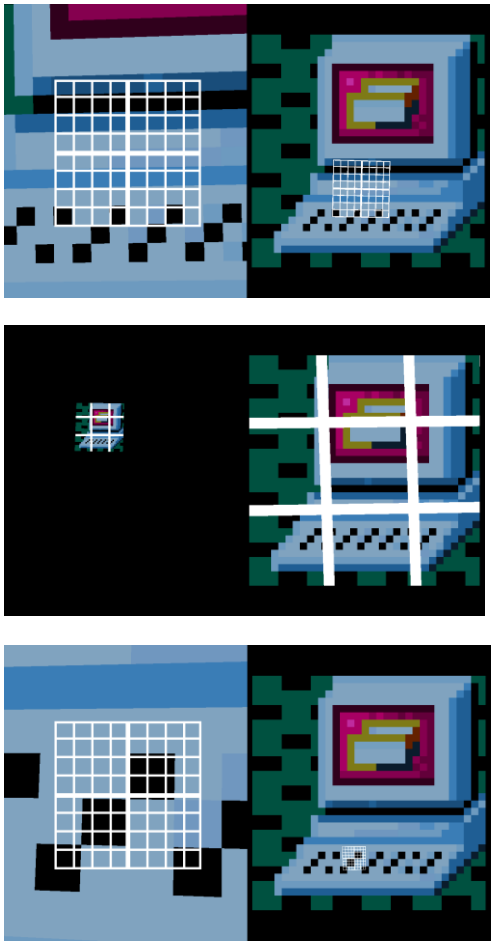
Primena tekstura - primer

```
int ID;
glGenTextures(1, &ID);
glBindTexture(GL_TEXTURE_2D, ID);

glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA,
             TextureImage->sizeX,
             TextureImage->sizeY,
             0, GL_RGB, GL_UNSIGNED_BYTE,
             TextureImage->data);

glTexParameteri(GL_TEXTURE_2D,
             GL_TEXTURE_MIN_FILTER,GL_LINEAR);

glTexParameteri(GL_TEXTURE_2D,
             GL_TEXTURE_MAG_FILTER,GL_LINEAR);

glEnable(GL_TEXTURE_2D);
```

MIP-mapiranje

- Filtriranje teksture: proces uzorkovanja teksture.
- Formalno, tekstura je funkcija (1D, 2D, ...)
- Vrednost funkcije u datoj tački u prostoru je boja odgovarajućeg teksela.
- Složen proces: učestanost uzorkovanja u prostoru teksture može biti sasvim drugačija od učestanosti uzorkovanja u prostoru sintetizovane slike
- Prostorne koordinate objekata variraju nezavisno od koordinata teksture, rešetka piksela, koja je u prostoru sintetizovane slike najčešće pravougaonog oblika,može imati različite projekcije (otiske – footprint) u prostoru teksture.

prostor teksture | prostor ekrana

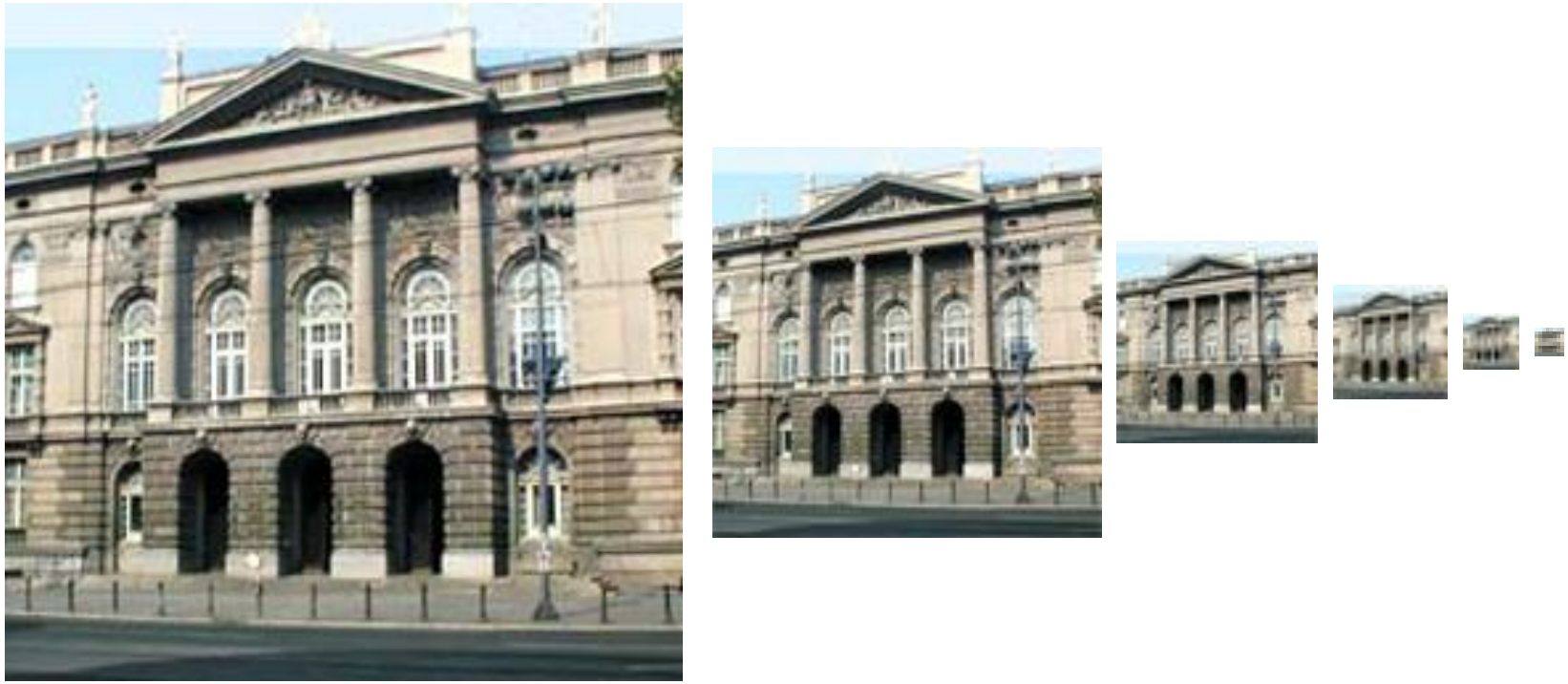


Najjednostavniji slučajevi filtriranja su:

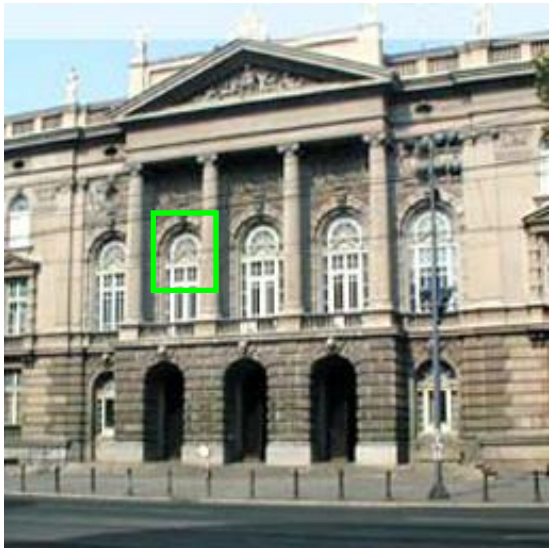
1. više teksela se preslikava na jedan piksel
2. više teksela se ne preslikava na jedan piksel

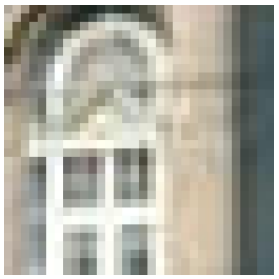
MIP-mapiranje

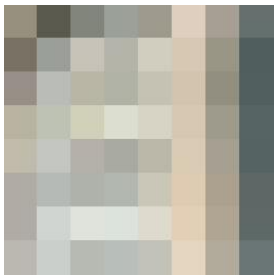
- MIP-mapa
  - kolekcija slika, progresivno opadajuće veličine (do 1x1)
  - svakoj slici se pridružuje nivo (ceo broj)
  - tradicionalno: najveća slika označava nivoom 0
  - najčešće: slika na nivou i+1 predstavlja umanjenu sliku nivoa i
  - tehnika zahteva oko 33% više memorije u odnosu na sliku nivoa 0
  - koristi se kod ubrzavanja **minifikacije**
- Omogućava **trilinearno** filtriranje (bilinearno + MIP-mapa)

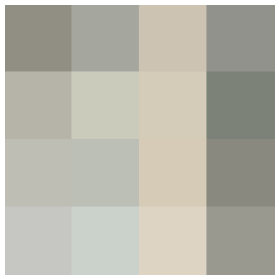


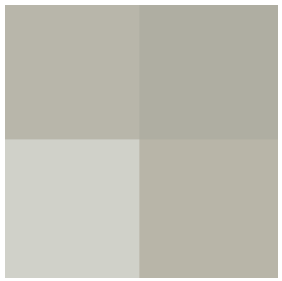
MIP-mapiranje

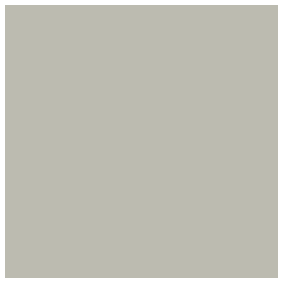












MIP-mapiranje

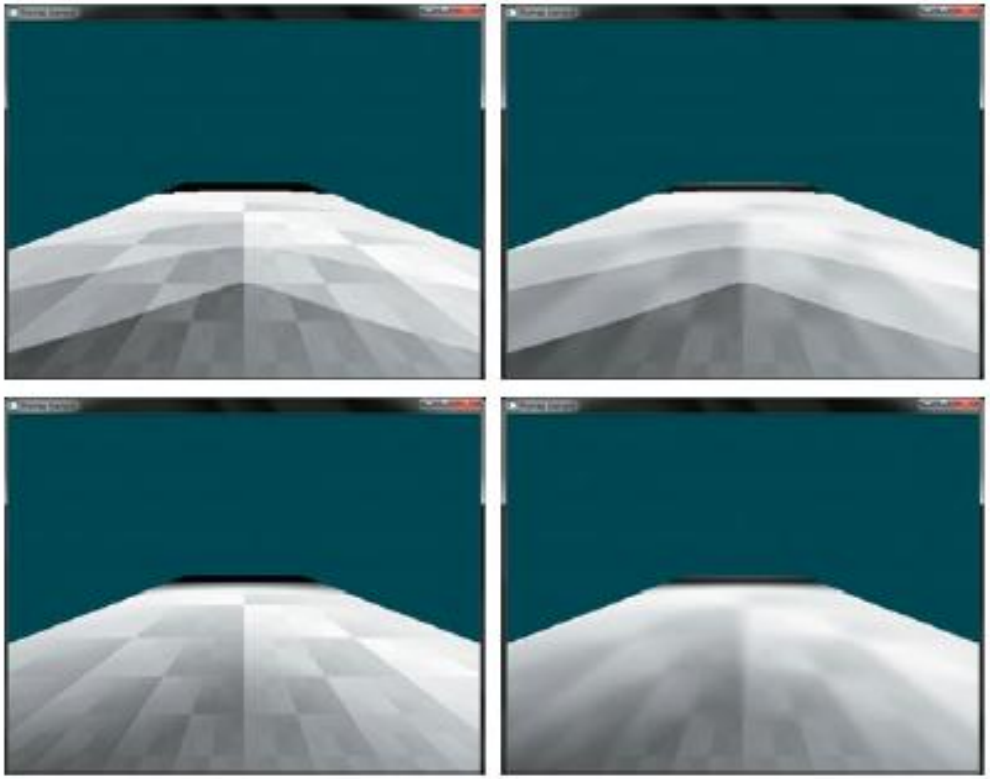
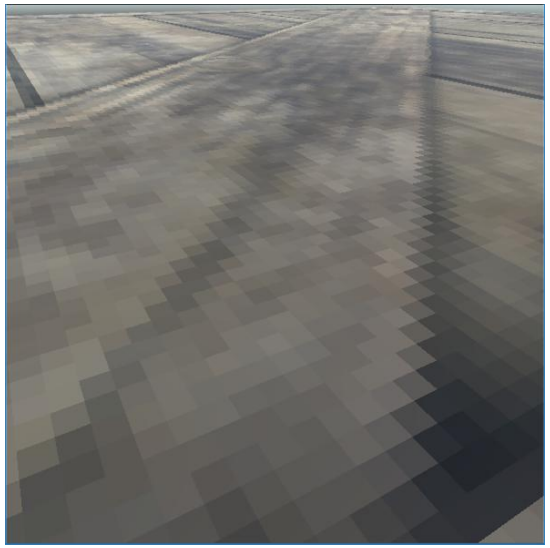


Figure 6.18 Effects of minification mipmap filters (GL_NEAREST_MIPMAP_NEAREST (top left), GL_LINEAR_MIPMAP_NEAREST (top right), GL_NEAREST_MIPMAP_LINEAR (bottom left), and GL_LINEAR_MIPMAP_LINEAR (bottom right).)

Preuzeto iz Orange Book, 3. izdanje, strana 335.

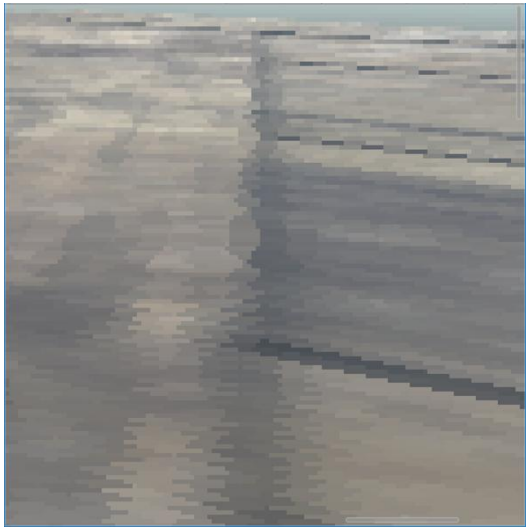
Filtriranje tekstura



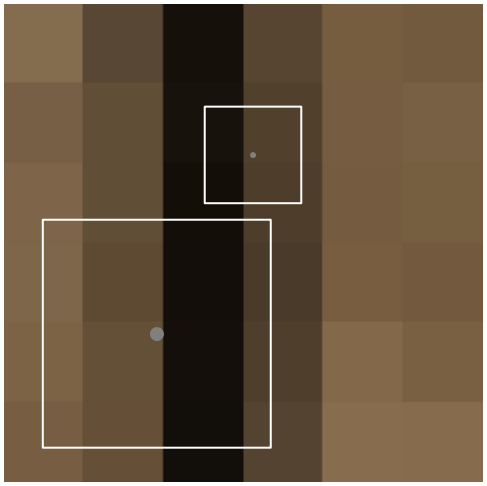
Point (nearest neighbour)

Bilinear

Trilinear

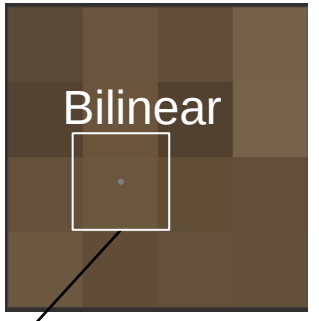


Filtriranje tekstura



Point

Bilinear



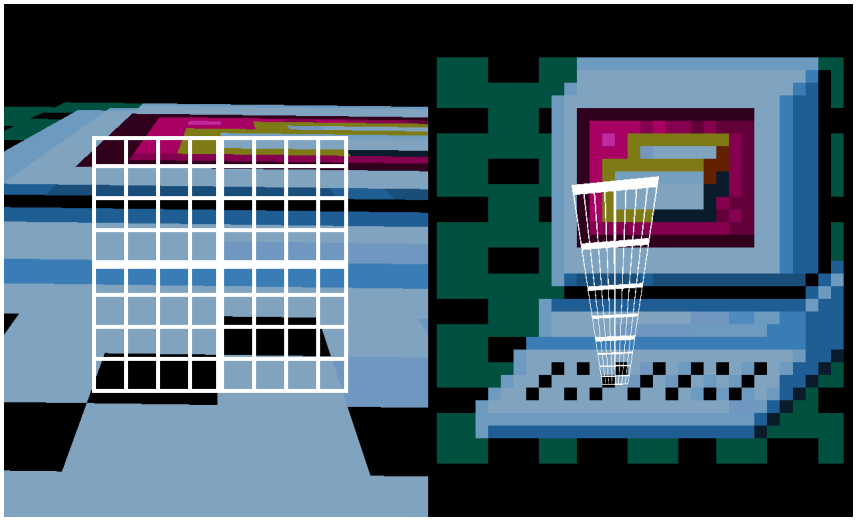
Linear

Trilinear

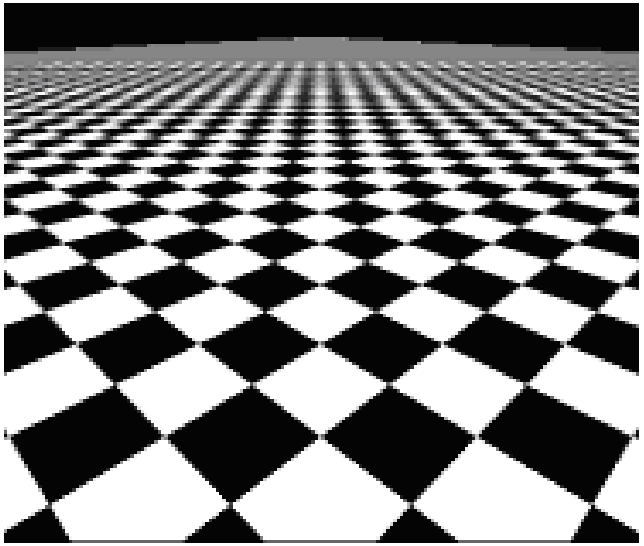
Anizotropno filtriranje

- Anizotropija: pojava kod koje se određen fenomen različito manifestuje u zavisnosti od pravca.
- Anizotropno filtriranje teksture: proces uzorkovanja (funkcije) teksture kada se rešetka piksela značajno deformiše prilikom projekcije u prostor teksture.

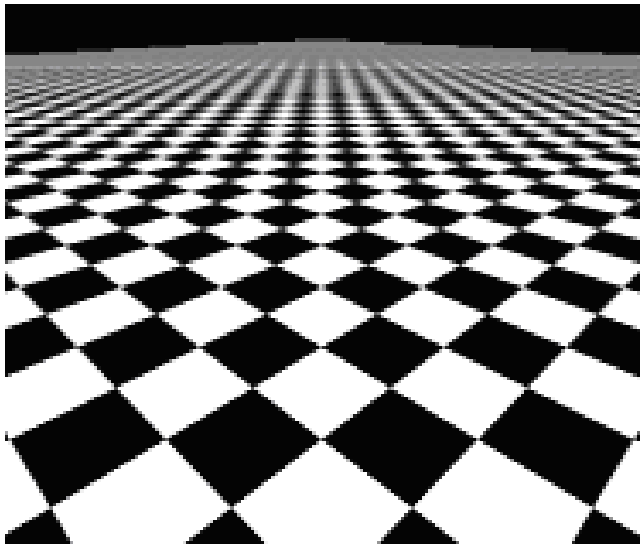
Rešetka je do te mere deformisana
da gornja vrsta rešetke
pokriva nekoliko redova teksela,
a krajnje donja vrsta rešetke
samo jedan red teksela
(slično i za kolone).



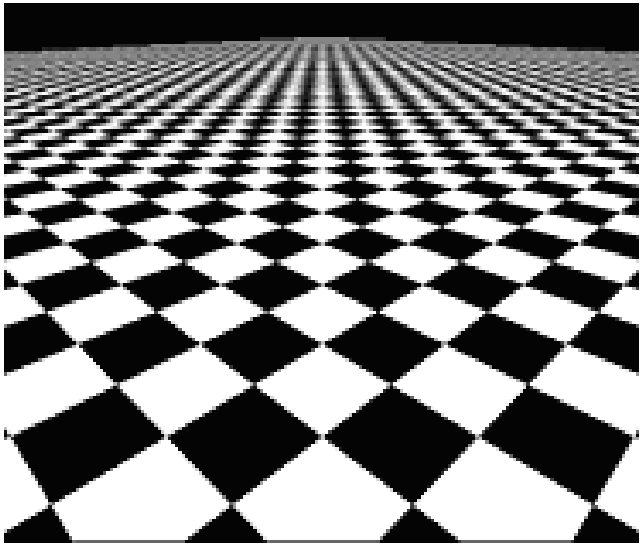
Anizotropno filtriranje



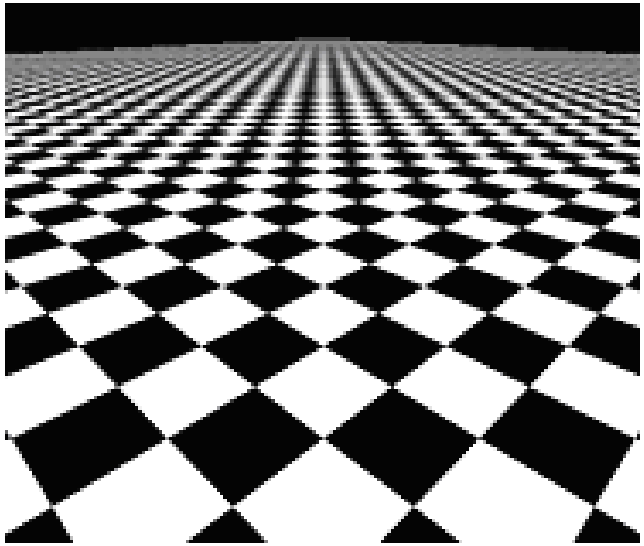
izotropno bilinearno filtriranje



izotropno trilinearno filtriranje



anizotropno bilinearno filtriranje



anizotropno trilinearno filtriranje

Anizotropno filtriranje

Trilinearno (bilinearno + MIP mapa)

Anizotropno

Slike preuzete sa besplatne enciklopedije Wikipedia (©Angus Dorbie), napravljene pomoću Google Earth