RAČUNARSKA GRAFIKA

Oznaka predmeta: RG

Predavanje broj: 10

Nastavna jedinica: Z-buffer. OGL. GLUT

Nastavne teme: Z-buffer algoritam. OpenGL: uvod, aplikacije, biblioteke. Tok podataka: geometrijski, slike, pipeline. Sintaksa komandi. State mašina. Glut: dizajn, struktura programa, inicijalizacija, callback funkcije. Greške. Baferi. Boja. Senčenje.

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.

Z-bafer algoritam

- Inicijalizacija:

```
for( i=0; i<Xmax; i++)
  for( j=0; j<Ymax; j++)
  {
    depth[i,j] = Zmin; //npr. Zmin = -1000
    screen[i,j]= background;
  }
```

- Funkcija za postavljanje piksela:

```
void SetPixel(int x, int y, int z, int v);
{
  if (z>depth[x,y])
  {
   depth[x,y]  = z;
   screen[x,y] = v;
  }
}
```

Z-bafer–računanje z koordinate

- Osnovni problem z-bafer algoritma:
  - izračunavanje z koordinate za svaku tačku koja se prikazuje pri sken-konverziji poligona
- Polazi se od jednačine ravni kroz 3 tačke:

$$\det\begin{bmatrix} x-x1 & y-y1 & z-z1 \\ x2-x1 & y2-y1 & z2-z1 \\ x3-x1 & y3-y1 & z3-z1 \end{bmatrix} = 0 \Rightarrow Ax+By+Cz+D=0$$

$$\Rightarrow z = \frac{-D-Ax-By}{C}$$

- Pošto se crta tačka po tačka idući duž X-pravca (x se inkrementira za 1)
  - pod pretpostavkom ortogonalne projekcije
    - ako je u tački $(x,y) \Rightarrow z$, tada je u $(x+1,y) \Rightarrow z'=z-A/C$
    - A/C je konstanta za ceo poligon i računa se samo jednom
  - ako se radi sa projekcijom sa perspektivom

$$\frac{x'}{x} = \frac{d}{d-z}, \quad \frac{y'}{y} = \frac{d}{d-z} \Rightarrow \text{(iz jednačine ravni)} \quad z = \frac{-D-Ax'-By'}{Cd-Ax'-By'} \cdot d$$

    - međusobni odnos z-koordinata 2 tačke P i R određuje koja se vidi: $z_P > z_R \Rightarrow$ P se vidi (smer ka nama je ka +z)

Z-bafer –prednosti i nedostaci

- Prednosti:
  - jednostavnost, nezavisnost od scene, mogućnost HW realizacije
- Nedostaci:
  - veličina z-bafera, izračunavanje za svaku tačku prilikom sken-konverzije
- Primer
  - ako ekran ima rezoluciju 1024x768 piksela, odrediti:
    a) kapacitet memorije za Z-bafer (*depth buffer*)
    ako je potrebna rezolucija od 256 nivoa dubine;
  - b) kapacitet video-memorije (*screen buffer*)
    za istovremeni prikaz 16M boja;
- Rešenje

  a) Za 256 nivoa dubine potrebno je 8 bita po pikselu za Z-bafer ($256 = 2^8$)
  te ukupni kapacitet memorije za *depth buffer* je 1024x768x8bit = 768 KB

  b) Za istovremeni prikaz 16M boja (24 bita po pikselu = 3 bajta po pikselu),
  screen buffer zauzima kapacitet 1024x768x3 = 2304 KB

OpenGL – uvod

- U nekoliko prethodnih lekcija bio je dat po primer rešenja u OpenGl-u (koristi se biblioteka GLUT) koji se odnosio na datu lekciju.
- Šta je OpenGL?
  - OpenGL je API (Applications Programming Interface)
  - Softverski interfejs ka grafičkom hardveru
  - Sloj između programera i grafičkog hardvera
  - Omogućuje opis scena na najnižem nivou
  - OpenGL je proceduralni interfejs ka hardveru
  - Platformski je nezavisan
- Istorijat :
  - 1980 – IrisGL (Silicon Graphics)
  - 1992 – OpenGL verzija 1.0
  - 2001 – OpenGL verzija 1.3
  - …
  - 2014 – OpenGL 4.5

OpenGL - uvod

- Specifikacija OpenGL standarda
  - Specifikaciju piše ARB (Architecture Review Board)
  - Članovi ARB-a su DEC, HP, Intel, IBM, Microsoft...
- Implementacija biblioteka
  - OpenGL biblioteke može napisati bilo ko i ako biblioteke prođu testove mogu se nazvati OpenGL, u suprotnom ne (MESA – open source grafička biblioteka)
- Mogućnosti
  - crtanje tačaka, linija, trouglova, poligona (3D)
  - manipulacija slika (2D)
  - teksture
  - osvetljenje ...
- OpenGL ne obezbeđuje (ali postoje biblioteke koje ovo obezbeđuju)
  - Podršku za opis objekata i scene
  - Podršku za parametrizovani opis površina (NURBS i sl.)
  - Rad sa prozorima
  - Senčenje...

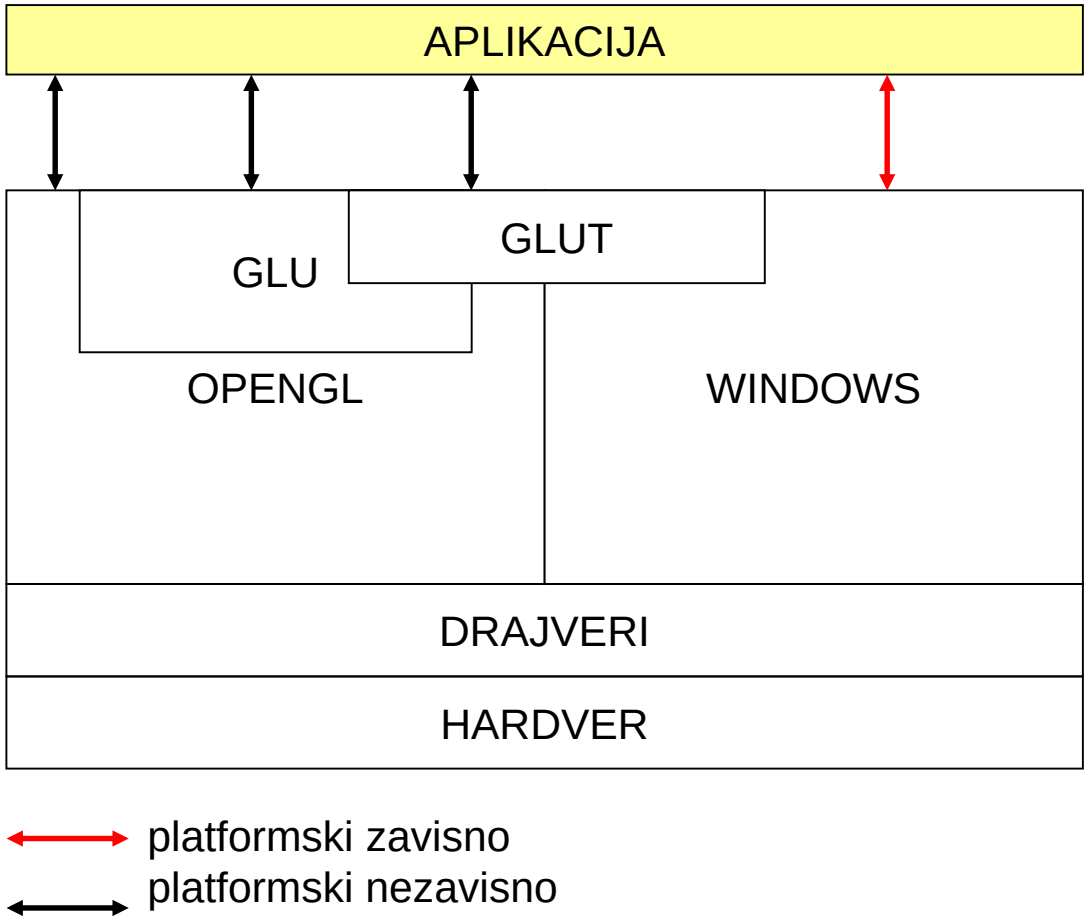
Postojeće OpenGL aplikacije

- 3D animacija i modelovanje
  - Alias/Wavefront Maya
  - Kinetix 3D Studio MAX
- CAD/CAM alati
  - Bentley Systems MicroStation
  - Pro/ENGINEER
  - AutoCAD
- Igre (tehnologija OpenGL je standard na linux-u i “windows”-u)
  - Unreal
  - Quake
  - HalfLife ...
  - igre na mobilnim telefonima
  - ...

OpenGL biblioteke

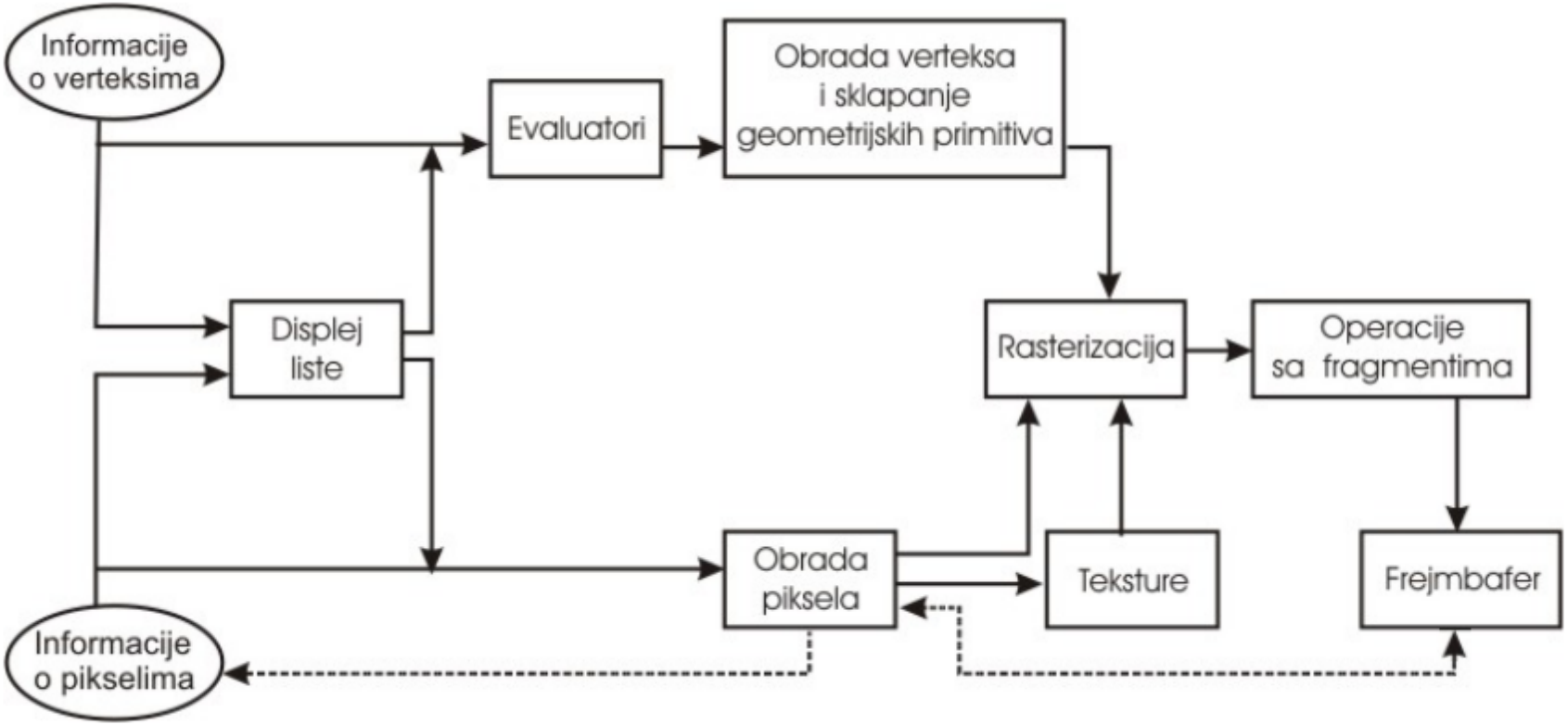
- OpenGl – **gl.h**
  - Osnovna zaglavlja gde su deklarisani tipovi podataka(GLfloat...)
- OpenGL Utility Library (GLU) – **glu.h**
  - Sastavni deo OpenGL distribucije
  - Transformacija koordinata
  - Osnovni objekti
  - NURBS
- GLUT (OpenGL Utility Toolkit)– **glut.h**
  - Rutine za rad sa prozorima (platformski nezavisne)
  - Nije deo OpenGL distribucije
  - Blizanac verzija je freeglut.
- Open Inventor
  - Objekti za rad sa modelima
  - Objektno orijentisan
  - Nije deo OpenGL distribucije

Struktura OpenGL aplikacije



Tok podataka

- Blok dijagram OpenGL sistema



Tok podataka – geometrijski podaci

- Svi objekti su opisani pomoću tačaka-temena (vertex)
- Tačka se opisuje sa nekoliko parametara
  - Koordinate (x, y, z, w)
  - Podaci o vektoru normale u tački
  - Podaci o teksturi koja se koristi
  - Boja (RGBA ili indeks)
    - R-red
    - G-green
    - B-blue
    - A- alfa kanal za providnost
  - Podaci o ivici (edge-flag)
- Svi ovi podaci procesiraju se zajedno

Tok geometrijskih podataka

- Faza “Evaluatori”
  - Objekti mogu biti predstavljeni pomoću evaluatora (npr. Bezier-ove funkcije)
  - U ovoj fazi se te informacije konvertuju u tačke

- Faze:
  - “Obrada verteksa”
    - Transformacije koordinata i normala
    - Računa se osvetljenje i menja se boja tačaka
    - Generišu se koordinate za teksture

  i
  - “Sklapanje geometrijskih primitiva”
    - Vrši se odsecanje primitive prema zadatom prozoru
    - Primenjuje se perspektiva na scenu

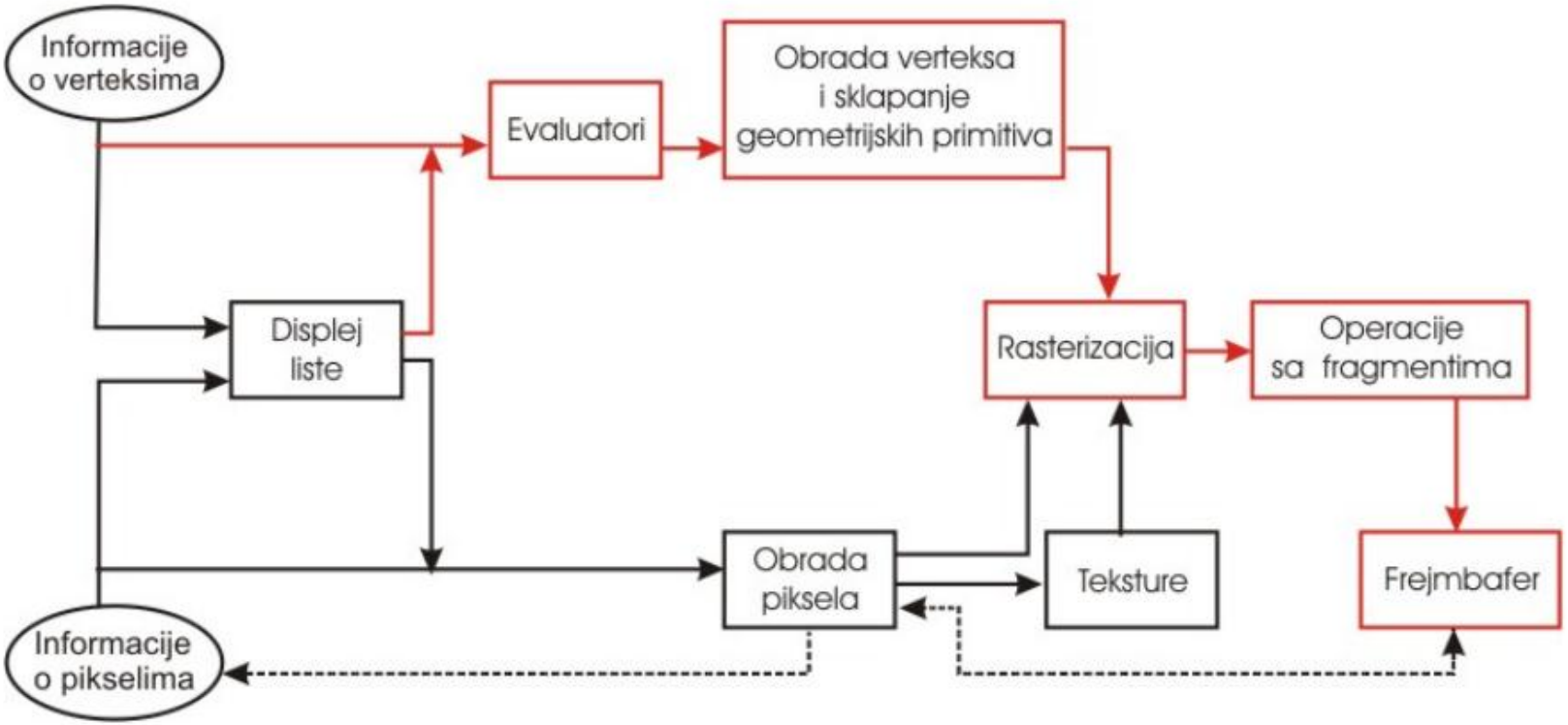
Tok geometrijskih podataka

- Faza “Rasterizacija”
  - Primitive se konvertuju u fragmente
  - Na objekte se primenjuju njihove osobine (širina linije, stil ...)
  - Određuju se pikseli koje primitiva zauzima na ekranu
  - Dubina se dodeljuje svakom pikselu (z-buffer)

- Faza “Operacija sa fragmentima”
  - U slučaju da postoje teksture, svakom fragmentu se pridružuje teksel (texture-pixel)
  - Nad fragmentima se primenjuju testovi
    - scissor test (odsecanje)
    - alpha test (providnost)
    - stencil test (za senčenje)
    - depth-buffer test (test dubine)
  - Fragmentu se dodaje boja pa se upisuje u bafer

Tok geometrijskih podataka

- Prikazivanje trougla (bez teksture)

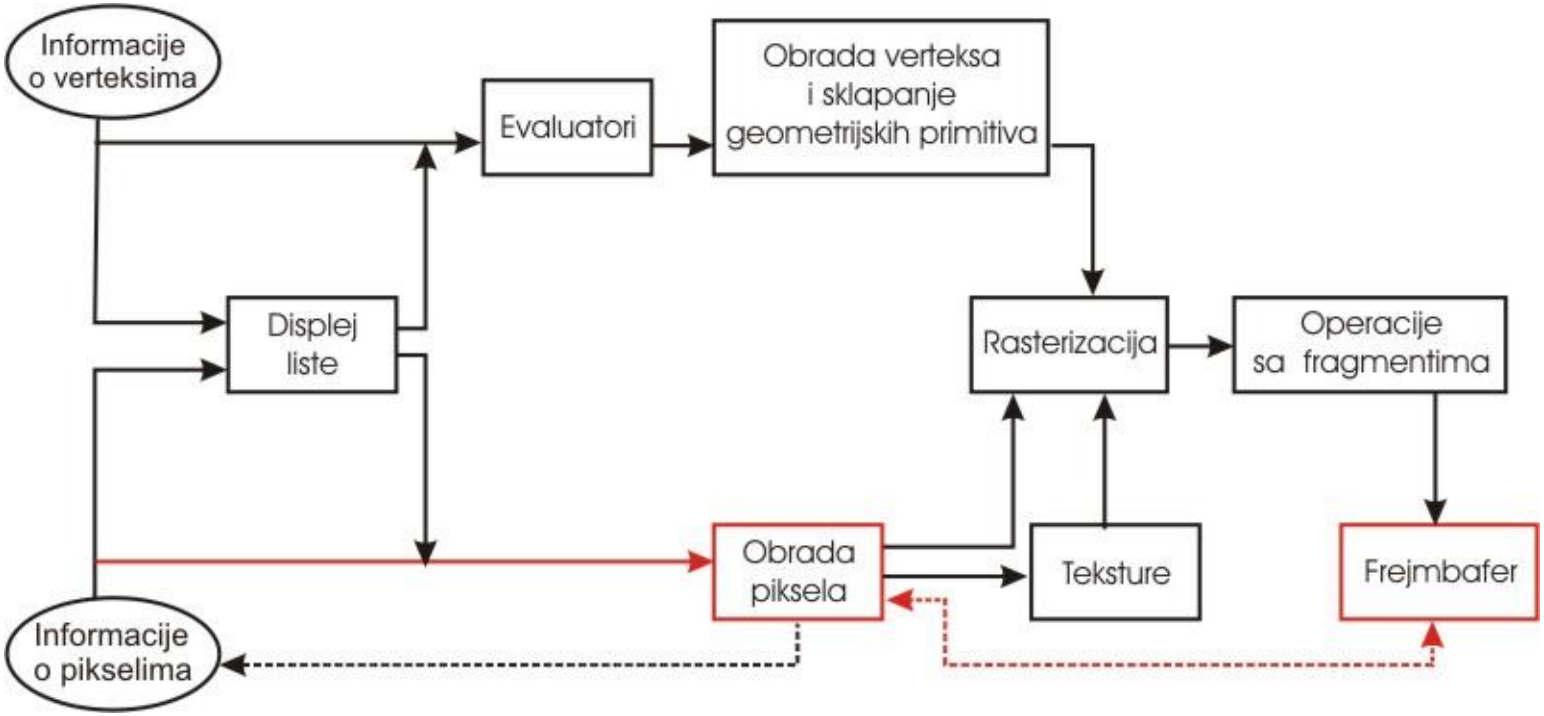


Tok podataka o slici (pikseli)

- Faza “Obrada piksela”
  - Vrši se konverzija slike u odgovarajući oblik
  - Po potrebi se vrši i transformacija slike (skaliranje i sl.)
  - Slika potom ide u frejmbafer ili u memoriju tekstura
  - Dohvatanje slike iz frejmbafera
    - Pikseli se mogu prebacivati i iz frejmbafera u operativnu memoriju
    - Vrše se potrebne transformacije (skaliranje i sl.)
- “Teksture”
  - U memoriji za teksture se čuvaju slike koje će se koristiti kao teksture
  - One mogu biti iz operativne memorije ili iz frejmbafera

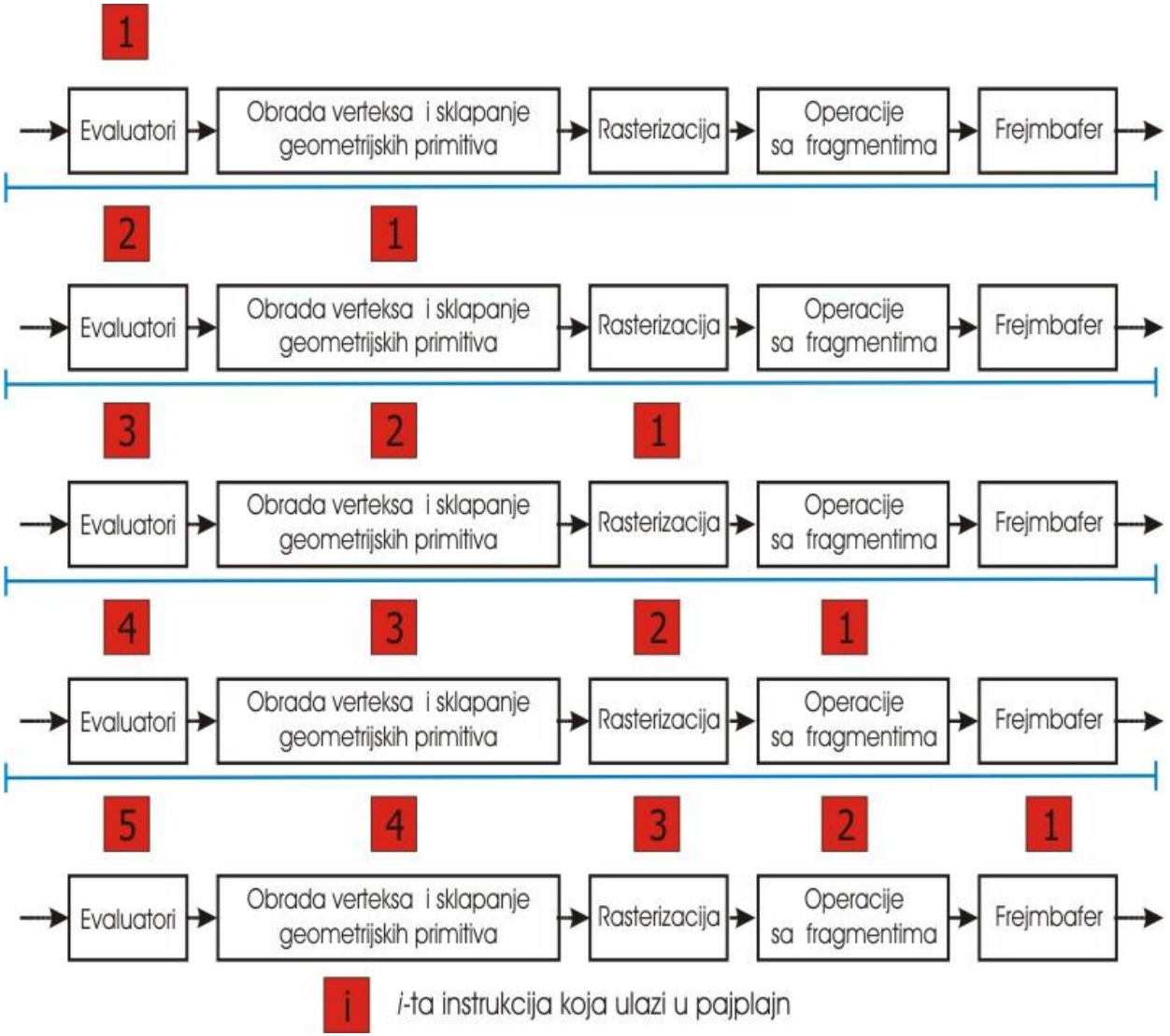
Tok podataka o slici

- Prikazivanje slike



Tok podataka – pajplajn

- OpenGL je pogodan za pajplajn obradu podataka

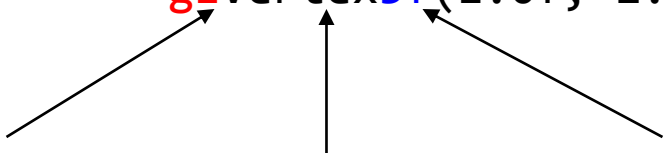


Sintaksa OpenGL komandi

- Sve OpenGL komande imaju prefiks i sufiks
- U slučaju procedura početno slovo svake reči je veliko

Tip	Prefiks	Primer
poziv procedure	`gl`	`glEnd()`
konstanta	`GL_`	`GL_POINTS`
tip podataka	`GL`	`GLfloat`

Tipična komanda : `glVertex3f(1.0f, 2.0f, 1.0f)`



prefiks

naziv komande

broj i tip argumenata
(tri float argumenta)

Sintaksa OpenGL komandi

Sufiksi komandi i tipovi podataka

OpenGL tip	Interna predstava	C tip	Sufiks
`GLbyte`	8-bitni ceo broj	char	`b`
`GLshort`	16-bitni ceo broj	short	`s`
`GLint, GLsizei`	32-bitni ceo broj	long	`i`
`GLfloat, GLclampf`	32-bitni broj u pok. zarezu	float	`f`
`GLdouble, GLclampd`	64-bitni broj u pok. zarezu	double	`d`
`GLubyte, GLboolean`	8-bitni pozitivan ceo broj	unsigned char	`ub`
`GLushort`	16-bitni pozitivan ceo broj	unsigned short	`us`
`GLuint, GLenum, GLbitfield`	32-bitni pozitivan ceo broj	unsigned long unsigned int	`ui`

Sintaksa OpenGL komandi

- Sufiks v označava da se funkciji predaje parameter koji je niz u kome se nalaze dati argument/i
- Komande

```
glColor3f(1.0f, 1.0f, 1.0f);
```

i

```
GLfloat color[] = {1.0f, 1.0f, 1.0f};
glColor3fv(color);
```

imaju identičan efekat

- Na pojedinim sistemima v-sintaksa je brža:
  - Proceduri se predaje samo jedan parametar
  - Hardver može posedovati mod u kome je u stanju da brzo prebaci blok podataka iz memorije u video kartu (burst mode)
  - Podatke zbog toga treba organizovati sekvencijalno u memoriji

Elementi koda

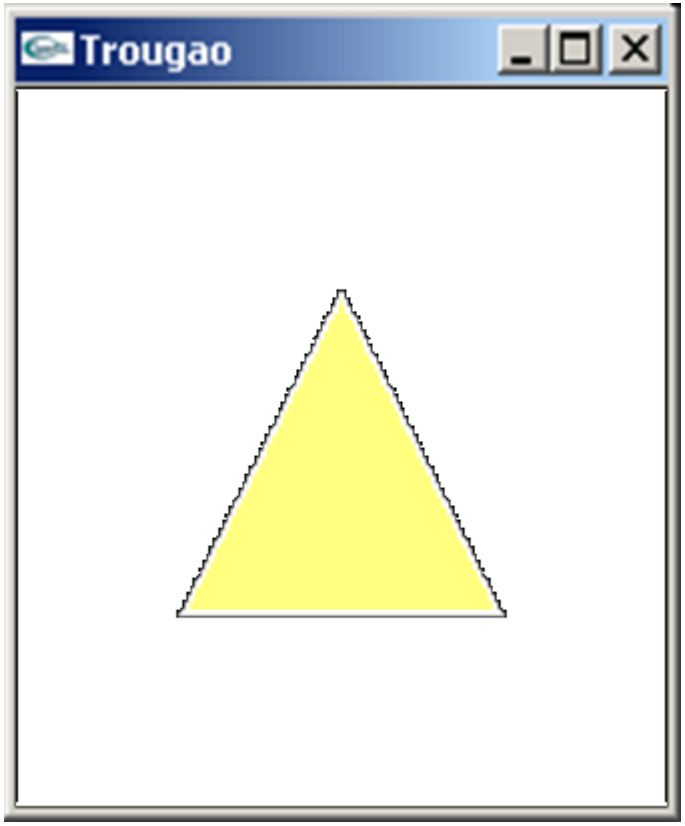
```
//nalazenje i uvoz potrebnih biblioteka
#pragma comment(lib,"opengl32.lib")
#pragma comment(lib,"glu32.lib")
#pragma comment(lib,"glut32.lib")
//ako hocemo isklucenje konzolnog prozora
#pragma comment(linker,"/subsystem:windows /entry:mainCRTStartup")
//ukljucenje zaglavlja glut-a
#include <GL/glut.h>
// Callback funkcija za iscrtavanje
void display() {
  //cisti prozor i postavlja boju pozadine
  //koja je specificirana naredbom glClearColor()
  glClear(GL_COLOR_BUFFER_BIT);
  //početak koda koji će definisati objekat
  //koji se crta – u ovom primeru poligon
  glBegin(GL_POLYGON);
    //specificira vrhove poligona koji će biti nacrtan
    glVertex2f(0.0, 0.0);
    glVertex2f(1.0, 0.0);
    glVertex2f(0.5, 0.866);
  //označava kraj koda koji će definisati objekat koji se crta
  glEnd();
  //forsira izvršavanje komandi koje su prethodno zadate
  glFlush();
}
```

Elementi koda

```
// callback funkcija za obradu dogadjaja tastature
void kbd( unsigned char key, int x, int y)
{
  if (key==27) exit(0);
}
int main(int argc, char** argv)
{
  // inicijalizacija GLUTa
  glutInit(&argc,argv);
  // postavljanje velicine gl prozora
  glutInitWindowSize(200, 200);
  // kreiranje gl prozora
  glutCreateWindow("Trougao");
  // specificira koord. sistem u
  // kome OpenGL crta sliku, kao i način mapiranja slike na ekran
  glOrtho(-0.5, 1.5, -0.5, 1.4, -1.0, 1.0);
  // postavljanje funkcije za iscrtavanje
  glutDisplayFunc(display);
  // postavljanje callback funkcije za obradu dogadjaja tastature
  glutKeyboardFunc(kbd);
  // beskonacna petlja iscrtavanja i obrade dogadjaja
  glutMainLoop(); return 0;
}
```

Pseudo primer

```
void main() {
    Inicijalizuj_prozor();
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 0.0);
    glOrtho(-0.5, 1.5,
            -0.5, 1.4,
            -1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex2f(0.0, 0.0);
        glVertex2f(1.0, 0.0);
        glVertex2f(0.5, 0.866);
    glEnd();
    glFlush();
    Održavaj_prozor_na_ekranu();
}
```



OpenGL – *state* mašina

- OpenGL se može posmatrati i kao *stejt* mašina
- OpenGL možete postaviti u različita stanja (modove) u kojima on ostaje sve dok ih ne promenite
- Stanja se kontrolišu pomoću atributa
  - Primer : tekuća boja
    - Tekuća boja se može postaviti na bilo koju vrednost
    - Svi objekti koji se crtaju imaju tekuću boju
- Dve vrste atributa (neformalna podela):
  - Promenljive stanja (boja, debljina linije ...)
  - Kontrolni atributi (kontrolišu koje mogućnosti OpenGL-a su trenutno aktivne)
- Uključenje i isključenje mogućnosti:
  - glEnable(GLenum cap) – uključuje neku od mogućnosti OpenGL-a
  - glDisable(GLenum cap) – isključuje neku od mogućnosti OpenGL-a

```
glEnable(GL_DEPTH_TEST); // Uključuje z-buffer testiranje
glEnable(GL_LIGHTING);   // Uključuje osvetljenje
```

  - Osobine koje se menjaju pomoću ovih komandi imaju samo dve vrednosti (uključeno - isključeno)

OpenGL – state mašina

- Provera da li je mogućnost uključena

```
GLboolean isLightOn;
isLightOn = glIsEnabled(GL_LIGHTING);
```

- Većina OpenGL rutina radi sa komplikovanijim stanjima (stanja sa više parametara)

  ```
  glColor3f(float r, float g, float b)
  ```

  postavlja tri realna broja koji predstavljaju boju

  - Sva tri broja zajedno čine GL_CURRENT_COLOR stanje

- Svako od komplikovanih stanja se može očitati ili promeniti.

OpenGL – state mašina

- Promenljive stanja se dohvataju pomoću

```
void glGetBooleanv(GLenum pname, GLboolean *params);
void glGetIntegerv(GLenum pname, GLint *params);
void glGetFloatv(GLenum pname, GLfloat *params);
void glGetDoublev(GLenum pname, GLdouble *params);
void glGetPointerv(GLenum pname, GLvoid **params);
```

- Ove naredbe dohvataju promenljive stanja odgovarajućeg tipa

```
// Dohvata tri promenljive (tekuća boja)
GLfloat boja[4];
glGetFloatv(GL_CURRENT_COLOR, boja);
```

- Za pojedine promenljive postoje posebne naredbe za dohvatanje (npr. glGetLight)

PRIMER 1/3

```
#pragma comment( lib, "opengl32.lib")
#pragma comment( lib, "glu32.lib")
#pragma comment( lib, "glut32.lib")
#include <cmath>
#include <GL/glut.h>

bool flag = false;

static float fRotAngle = -60.0f;
GLdouble znear =4.0;

void kuglica(GLfloat x, GLfloat y, GLfloat z)
{
  glPushMatrix();
    glTranslatef( x,y,z );
    glRotatef( fRotAngle, 1.0f, 0.0f, 1.0f );
    glutWireSphere(0.5,8,8);
  glPopMatrix();
}

GLfloat xleft = -5.5;
GLfloat alpha = 0;
int masa = 6;
```

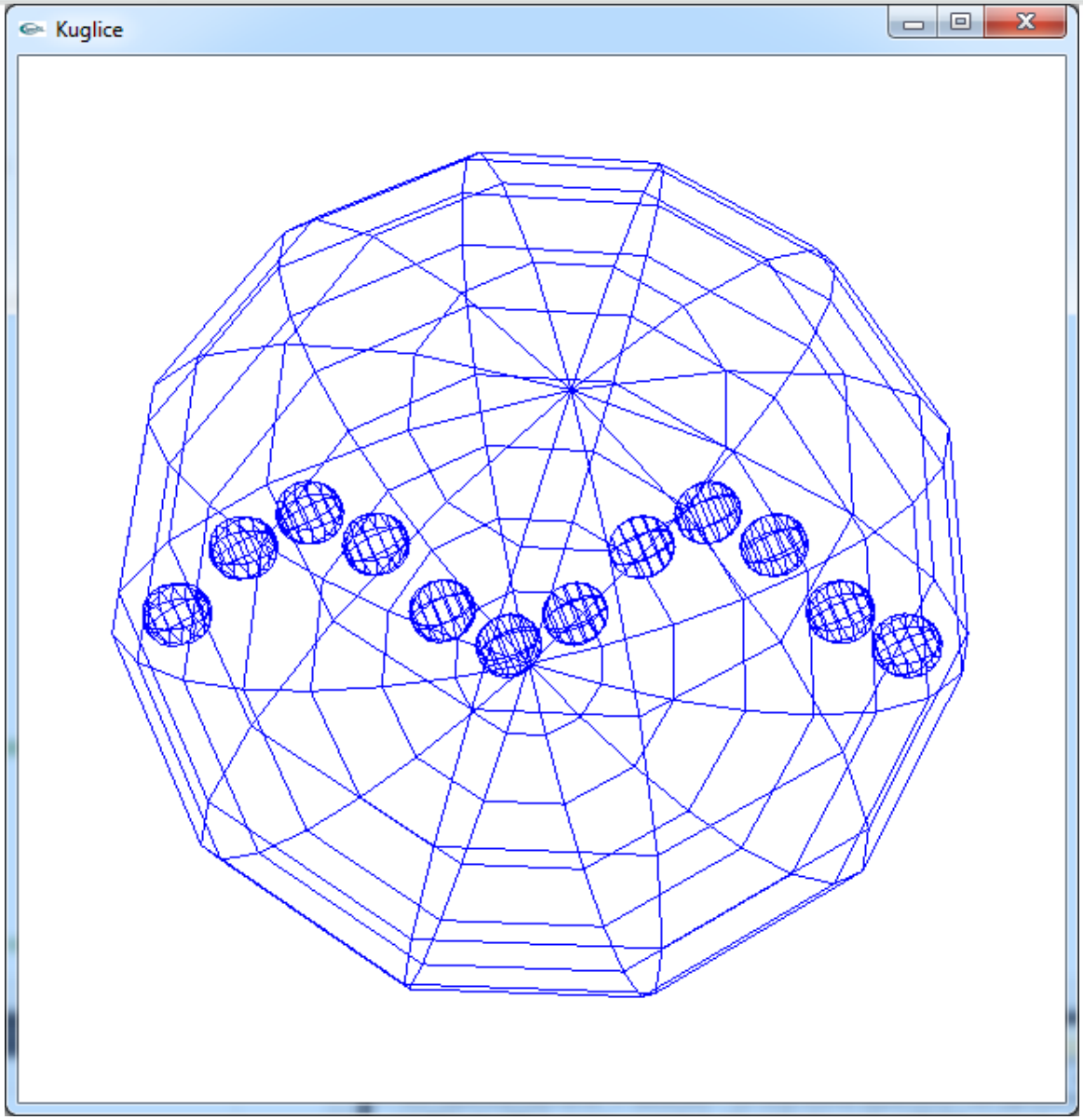
PRIMER 2/3

```
void Redraw(){
  glMatrixMode( GL_PROJECTION );
     glLoadIdentity();
  glFrustum( -2.0f, 2.0f, -2.0f, 2.0f, znear, 40.0f );

  glMatrixMode(GL_MODELVIEW);
  glClearColor(1.0,1.0,1.0,1.0);
  glColor3f(0.0,0.0,1.0);
  glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

  glLoadIdentity();
  gluLookAt(  0.0, 0.0, 15.0,    0.0, 0.0, 0.0,     0.0, 1.0, 0.0  );

  glPushMatrix();
    glTranslatef( 0.0f, 0.0f, -10.0f );
    glRotatef( fRotAngle/masa, 1.0f, 0.0f, 1.0f );
      glutWireSphere(10,12,12);
  glPopMatrix();

  for(int i=0; i<12; i++) kuglica(xleft+i,sin(0.035*(alpha+i*30.0)),0);
  fRotAngle+= 0.5f; if(fRotAngle>masa*360.0) fRotAngle = 0;
  glutSwapBuffers();
}
```

PRIMER 3/3

```
void Keyboard( unsigned char uKey, int x, int y ){
  switch ( uKey ){
    case 0x1B: exit( 0 );break;
    case 'q' : znear+=0.1;break;
    case 'a' : znear-=0.1;break;
    case 'w' : alpha+=3.0f;break;
    case 's' : alpha-=3.0f;break;
  }
}
void Timer( int iValue ){
  glutPostRedisplay();  glutTimerFunc( 1, Timer, iValue );
}
int main( int argc, char **argv ){
   glutInitDisplayMode( GLUT_RGB | GLUT_DEPTH | GLUT_DOUBLE );
   glutInitWindowPosition( 0, 0 );
   glutInitWindowSize( 600, 600 );
   glutInit( &argc, argv );
   glutCreateWindow( "Kuglice" );
   glutKeyboardFunc( Keyboard );
   glutDisplayFunc( Redraw );
   //glutFullScreen();
   glutTimerFunc( 1, Timer, 50 );
   glutMainLoop();return 0;
}
```

Izlaz



GLUT

- GLUT – OpenGL Utility Toolkit, platformski je nezavisan, obezbeđuje sledeće funkcionalnosti:
  - Više prozora za OpenGL rendering
  - Obrada događaja pomoću “callback” funkcija
  - Sofisticirani ulazni uređaji
  - Tajmere i “idle” rutinu
  - Pop-up meniji
  - Rutine za generisanje raznih objekata (čajnik, torus...)
  - Podrška za fontove (“bitmap” i “stroke”)
  - Razne funkcije za manipulisanje sa prozorima
- GLUT funkcije imaju prefiks glut
- Organizovan je kao skup apija zaduženih za :
  - inicijalizaciju - započinjanje obrade događaja
  - rad sa funkcijama za obradu događaja - rad sa prozorima
  - rad sa menijima - rad sa fontovima - rad sa geometrijskim objektima

GLUT – struktura programa

- Struktura tipičnog programa koji koristi GLUT :

  1) Inicijalizacija OpenGL

  2) Postavljanje osobina prozora

  3) Kreiranje menija

  4) Registrovanje callback funkcija

  5) Ulazak u glavnu GLUT petlju za obradu događaja

  tako da je tipična main rutina kao što sledi:

```
int main(int argc, char *argv[]) {
    glutInit(argc, argv);     // inicijalizuje GLUT biblioteku
    glutInitDisplayMode(mode);// određuje OpenGL osobine
    postaviOsobineProzora();  // pozicija, dimenzije prozora
    glutCreateWindow("Ime aplikacije");// kreira prozor
    napraviMenije();              // kreira menije
    napraviCallbackFunkcije();    // callback funkcije
    glutMainLoop();               // pokreće aplikaciju
    return 0;                     // nikad se ne izvršava
}
```

GLUT - inicijalizacija

```
void glutInit(int *argcp, char **argv);
```

- Služi za inicijalizaciju GLUT biblioteke
  - argcp – pokazivač na broj parametara main() funkcije
  - argv – pokazivač na listu parametara main() funkcije

```
void glutInitWindowPosition(int x, int y);
```

- Inicijalizuje poziciju prozora na ekranu
  - x – X koordinata gornjeg levog ugla prozora
  - y – Y koordinata gornjeg levog ugla prozora

```
void glutInitWindowSize(int w, int h);
```

- Inicijalizuje veličinu prozora na ekranu
  - w – širina prozora
  - h – visina prozora

```
int glutCreateWindow(char *name);
```

- Kreira osnovni prozor sa imenom *name,* pri čemu svaki prozor ima jedinstven OpenGL kontekst

GLUT – inicijalizacija

```
void glutInitDisplayMode(unsigned int mode);
```

- Postavlja inicijalni mod prikazivanja
- Poziva se (po potrebi) za svaki prozor koji se kreira
- Parametar mode (mask) se dobija logičkom OR funkcijom sledećih vrednosti :

GLUT_RGBA	RGBA mod za boju (podrazumevano)
GLUT_RGB	RGB mod za boju
GLUT_INDEX	indeksirani mod za boju
GLUT_SINGLE	prozor sa jednim baferom (podrazumevano)
GLUT_DOUBLE	dvostruko baferisani prozor
GLUT_ACCUM	prozor sa akumulacionim baferom
GLUT_ALPHA	prozor sa baferom za alfa komponentu boje
GLUT_DEPTH	prozor sa baferom za dubinu
GLUT_STENCIL	prozor sa stencil baferom
GLUT_MULTISAMPLE	prozor sa podrškom za multisampling
GLUT_STEREO	stereo prozor
GLUT_LUMINANCE	luminance mod za boju (slično indeks modu)

GLUT

```
void glutMainLoop(void);
```

- Poziva se tek pošto se obavi kompletna inicijalizacija
- Poziva se najviše jedanput iz jednog programa
- Započinje GLUT-ovu petlju za obradu događaja
- Iz nje se pozivaju korisničke rutine za obradu događaja
- Ova rutina se nikada ne završava

GLUT – callback funkcije

- *Callback* funkcije
  - Koriste se za obradu događaja (miš, tastatura ...)
  - Poziva ih GLUT kada se nešto dogodi
  - Moraju se registrovati da bi GLUT znao da ih pozove
  - Moraju imati unapred određeno zaglavlje

Primer :

```
// Funkcija koja će obrađivati pritisnute tastere
// parametar key – taster koji je pritisnut
void obradiTastaturu(unsigned char key, int x, int y) {
    if (key==27) exit(0);
    // ako je ESC ==> izlazak iz aplikacije
}
…
int main() {
    …
    glutKeyboardFunc(obradiTastaturu);
    // registrujemo callback funkciju
    …
}
```

GLUT - registracija callback funkcija

```
void glutDisplayFunc(void (*func) (void));
```

- Registruje funkciju za iscrtavanje sadržaja prozora
- Parametar func:
  - Funkcija bez argumenata koja ne vraća vrednost
  - Pozivaće se svaki put kada je potrebno ponovo iscrtati sadržaj prozora
- func() se poziva kada :
  - GLUT proceni da je potrebno ponovno iscrtavanje
  - Programer eksplicitno pozove `glutPostRedisplay();`
- Postoji posebna funkcija za svaki prozor
- Uvek se poziva funkcija koja je povezana sa prozorom koji je trenutno aktivan

GLUT - registracija callback funkcija

```
void glutReshapeFunc(void (*func) (int w, int h));
```

- Registruje funkciju koja se poziva pri promeni dimenzije prozora
- Parametar func :
  - Funkcija sa argumentima koja ne vraća vrednost
  - argument w – nova širina prozora
  - argument h – nova visina prozora
- func() se poziva kada :
  - se promeni veličina prozora
  - se prozor prvi put iscrtava na ekran
- Ukoliko je func NULL ili nije registrovana poziva se glViewport(0, 0, w, h);

GLUT - registracija callback funkcija

```
void glutKeyboardFunc(
   void (*func)(unsigned char key,int x,int y)  );
```

- Registruje funkciju koja se poziva kada se pritisne neki taster
- Argument func :
  - Funkcija sa argumentima koja ne vraća vrednost
  - argument key – pritisnuti taster (ASCII kod)
  - argument x – x koordinata miša (relativna)
  - argument y – y koordinata miša (relativna)
- Ukoliko je func NULL ili nije registrovana događaji tastature se ignorišu

GLUT - registracija callback funkcija

```
void glutMouseFunc(
        void (*func)(int button, int state, int x, int y)
);
```

- Registruje funkciju koja se poziva kada se pritisne ili otpusti dugme na mišu
- Parametar func :
  - Funkcija sa argumentima koja ne vraća vrednost
  - argument button – dugme koje pritisnuto :
    - GLUT_LEFT_BUTTON – levo dugme (0)
    - GLUT_MIDDLE_BUTTON – srednje dugme (1)
    - GLUT_RIGHT_BUTTON – desno dugme (2)
  - argument state – stanje dugmeta :
    - GLUT_UP (otpušteno, 1) GLUT_DOWN (pritisnuto, 0)
  - argumenti x i y – pozicija miša
- Ukoliko je func NULL ili nije registrovana pritisci dugmadi se ignorišu.
- Točkić miša se registruje kao: button==3(scroll up) , button==4(scroll down), reakcija je kao da je urađen GLUT_DOWN pa onda GLUT_UP.

GLUT - registracija callback funkcija

```
void glutPassiveMotionFunc(void (*func)(int x, int y));
```

- Registruje funkciju koja se poziva kada se miš pomeri a nije pritisnut nijedan taster
- Parametar func :
  - Funkcija sa argumentima koja ne vraća vrednost
  - argumenti x i y – nova pozicija miša

```
void glutMotionFunc(void (*func)(int x, int y));
```

- Registruje funkciju koja se poziva kada se miš pomeri a pritisnut je neki od tastera
- Parametar func :
  - Funkcija sa argumentima koja ne vraća vrednost
  - argumenti x i y – nova pozicija miša

Crta trougao

```
#pragma comment(lib,"opengl32.lib")
#pragma comment(lib,"glu32.lib")
#pragma comment(lib,"glut32.lib")
#include <GL/glut.h>

// Callback funkcija za
// iscrtavanje
void display()
{
  glClear(GL_COLOR_BUFFER_BIT);
  glBegin(GL_POLYGON);
    glColor3f(1.0, 0.0, 0.0);
    glVertex2f(0.0, 0.0);
    glColor3f(0.0, 1.0, 0.0);
    glVertex2f(1.0, 0.0);
    glColor3f(0.0, 0.0, 1.0);
    glVertex2f(0.5, 0.866);
  glEnd();
  glFlush();
}
```

```
// Callback funkcija za tastaturu
void kbd(unsigned char key, int x, int y)
{
    if (key==27) exit(0);
}

int main(int argc, char** argv)
{
    glutInit(&argc,argv);
    glutInitWindowSize(200, 200);
    glutCreateWindow("Trougao");
    glOrtho(-0.5,1.5,-0.5,1.4,-1.0, 1.0);
    glutDisplayFunc(display);
    glutKeyboardFunc(kbd);
    glutMainLoop();
    return 0;
}
```

Greške

- U toku rada mogu nastati razne greške
- OpenGL detektuje jedan deo grešaka
- Kada se greška detektuje
  - OpenGL pamti kod greške koja je nastala
  - Komanda koja je generisala grešku se ignoriše
  - U slučaju GL_OUT_OF_MEMORY rezultat je nepredvidiv

```
GLenum glGetError(void);
```

  - Vraća kod greške koja se dogodila
  - U slučaju da greške nema vraća GL_NO_ERROR
  - Resetuje kod greške na GL_NO_ERROR
  - Trebalo bi je pozivati barem jedanput posle crtanja scene

Greške

Vrednost	Opis
GL_NO_ERROR	greška se nije dogodila
GL_INVALID_ENUM	GLenum argument van opsega
GL_INVALID_VALUE	numerički argument van opsega
GL_INVALID_OPERATION	operacija nije dozvoljena u tekućem stanju
GL_STACK_OVERFLOW	operacija bi dovela do prepunjenja steka
GL_STACK_UNDERFLOW	operacija bi dovela do čitanja sa praznog steka
GL_NO_MEMORY	nema dovoljno memorije da se izvrši komanda

```
GLubyte* gluErrorString(GLenum errorCode);
```

- Vraća opis greške kao string, parametar errorCode je kod greške

```
GLenum errCode;
const GLubyte *errString;

if ((errCode=glGetError())!=GL_NO_ERROR) {
    errString=gluErrorString(errCode);
    fprintf(stderr,"Greska je : %s\n", errString);
}
```

Baferi

- OpenGL koristi nekoliko bafera u toku rada.
- Ovi baferi su implementirani u hardveru.

Naziv	Opis
frame buffer	čuva piksele za ispis na ekran
z – buffer	sadrži udaljenost svake tačke na ekranu po z – osi
alpha buffer	sadrži providnost svakog piksela
accumulation buffer	služi za 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)

Baferi

- Izbor vrednosti za čišćenje bafera:
  - boja (color buffer)

```
glClearColor(GLclampf r, GLclampf g,
             GLclampf b, GLclampf alpha)
```

  - dubina (z- buffer)

```
glClearDepth(GLclampd depth)
```

  - stencil

```
glClearStencil(GLint s)
```

  - akumulacija

```
glClearAccum(GLclampf r, GLclampf g,
             GLclampf b, GLclampf alpha)
```

  - indeks

```
glClearIndex(GLfloat index)
```

Baferi

- Čišćenje bafera (postavljanje izabrane vrednosti)
  - baferi se čiste pomoću komande
    `glClear(GLint mask)`
  - mask je maska koja određuje koji baferi se čiste
  - mask se dobija kombinacijom sledećih vrednosti :

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);
```

Boja

- Opis objekta je nezavisan od boje kojom se crta.
- Svi elementi koji se iscrtavaju imaju tekuću boju.
- Tekuća boja se može promeniti u bilo kom trenutku.
- Boje se opisuju pomoću RGB ili RGBA modela.
- Postoji i indeksni mod za boju :
  - Sve vrednosti boje su upisane u jedan niz
  - Boju piksela određuje indeks boje u nizu

Primer (pseudokod):

```
postavi_boju(crvena)
nacrtaj objekat(A)
postavi_boju(plava)
nacrtaj objekat(B)
postavi_boju(zelena)
postavi boju(crvena)
nacrtaj objekat(C)
```

Objekt A je crven

Objekt B je plav

nema efekta zbog sledeće linije

Objekt C je crven

Boja

Boja se zadaje pomoću jedne od sledećih komandi :

```
void glColor3{b i f d ub us ui}(TYPE r, TYPE g, Type b)
void glColor3{b i f d ub us ui}v(TYPE *color)
void glColor4{b i f d ub us ui}(TYPE r, TYPE g, Type b, TYPE a)
void glColor4{b i f d ub us ui}v(TYPE *color)
```

Primer (ekvivalentne naredbe, sve postavljaju crnu boju) :

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

glColor3b(0,0,0);

glColor3ub(0, 0, 0);

unsigned int boja = {0, 0, 0};

glColor3uiv(boja);
```

Shading

- Linije ili popunjeni poligoni se mogu crtati
  - Uvek jednobojno – flat shading (constant shading)
  - U više boja – smooth shading (goraud shading)

- Flat shading
  - Svi pikseli koji pripadaju poligonu imaju istu boju
  - Boja poligona određena je bojom prve tačke koja je zadata (boja ostalih tačaka nema uticaj)

- Smooth shading
  - Boja svakog verteksa se uzima u obzir
  - Boje piksela u poligonu se interpoliraju između boja svih verteksa kojima je poligon zadat
  - Dva susedna piksela imaju neznatno različitu boju

Shading

- Smooth shading
  - OpenGL pokušava da napravi gradijent unutar poligona (gradient fill)
  - U slučaju RGBA boje ovo je u redu
  - U slučaju indeksiranog moda gradijent se izvodi nad indeksima
    - dva susedna indeksa mogu pokazivati na različite boje
    - rešenje je da se napravi gradijent u nizu boja

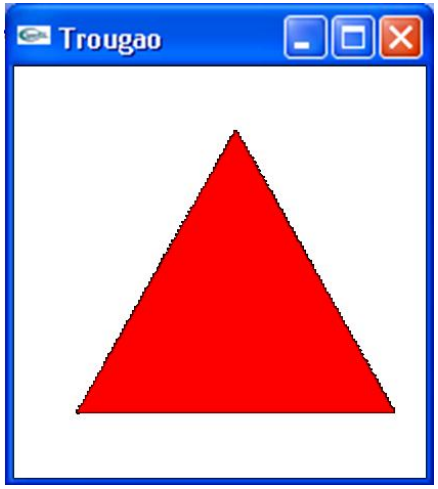
- Shading se mora zadati pomoću naredbe

```
void glShadeModel(GLenum mode);
```

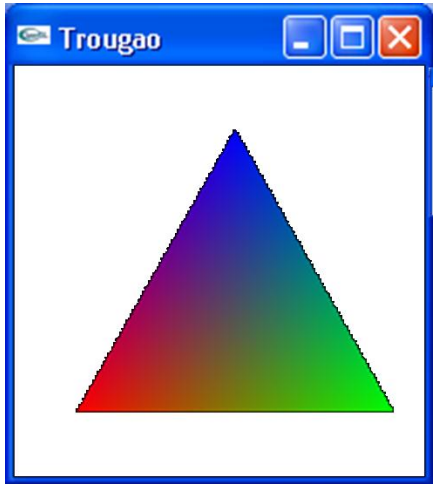
    - menja način bojenja
    - parametar mode
      - GL_FLAT
      - GL_SMOOTH

Boja

```
void display()
{
  glShadeModel(GL_FLAT);
  // ili glShadeModel(GL_SMOOTH);
  glClear(GL_COLOR_BUFFER_BIT);
  glBegin(GL_POLYGON);
    glColor3f(1.0, 0.0, 0.0);
    glVertex2f(0.0, 0.0);
    glColor3f(0.0, 1.0, 0.0);
    glVertex2f(1.0, 0.0);
    glColor3f(0.0, 0.0, 1.0);
    glVertex2f(0.5, 0.866);
  glEnd();
  glFlush();
}
```



GL_FLAT



GL_SMOOTH