OpenGL (1)

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija (OpenGL je skraćenica od *Open Graphics Language*). Predstavljen je prvi put 1992. godine, i od tada OpenGL predstavlja najčešće korišćeni grafički API (API je skraćenica od *Application Programming Interface*), koji je doneo na hiljade aplikacija za sve vrste računarskh platformi. Pored OpenGL-a, trebalo bi podesiti da se radi i sa GLUT-om (GLUT je skraćenica od *Graphics Language Utility Toolkit*) i u potpunosti je kompatibilan sa OpenGL-om.

Ovde će biti objašnjeno kako omogućiti da OpenGL i GLUT rade unutar operativnog sistema Windows i uz pomoć programskog jezika Visual C++. U ovom slučaju reč je o jeziku Visual C++ 6.0, unutar programskog paketa Microsoft Visual Studio 6.0. Da bi sve radilo kako treba korisnik mora da skine sa Interneta sledeće fajlove:

- za OpenGL: opengl32.lib, glu32.lib, glaux.lib, gl.h, glaux.h, glu.h, glu32.dll i opengl32.dll;
- za GLUT: glut.dll, glut32.dll, glut.h, glut.lib i glut32.lib.

Nakon toga treba ih smestiti na tačno određene lokacije. Sve biblioteke sa ekstenzijom LIB (opengl32.lib, glu32.lib, glaux.lib, glut.lib i glut32.lib) trebalo bi smestiti na sledeću lokaciju:

<drajv>:\<VC++ putanja>\lib\opengl32.lib
<drajv>:\<VC++ putanja>\lib\glu32.lib
<drajv>:\<VC++ putanja>\lib\glaux.lib
<drajv>:\<VC++ putanja>\lib\glut.lib
<drajv>:\<VC++ putanja>\lib\glut32.lib

gde <drajv>: predstavlja particiju hard diska na kojoj je instaliran programski paket Visual C++, a <VC++ putanja> predstavlja lokaciju direktorijuma gde je instaliran programski paket Visual C++. Biblioteke sa ekstenzijom H (gl.h, glaux.h, glu.h i glut.h) trebalo bi smestiti na sledeću lokaciju:

<drajv>:\<VC++ putanja>\include\GL\gl.h
<drajv>:\<VC++ putanja>\include\GL\glaux.h
<drajv>:\<VC++ putanja>\include\GL\glu.h
<drajv>:\<VC++ putanja>\include\GL\glut.h

gde <drajv>: predstavlja particiju hard diska na kojoj je instaliran programski paket Visual C++, a <VC++ putanja> predstavlja lokaciju direktorijuma gde je instaliran programski paket Visual C++. Biblioteke sa ekstenzijom DLL (glu32.dll, opengl32.dll, glut.dll i glut32.dll) trebalo bi smestiti u sistemske direktorijume unutar dirketorijuma gde se nalazi instaliran operativni sistem. Kada je reč o klasičnim Windows operativnim sistemima, onda je to lokacija:

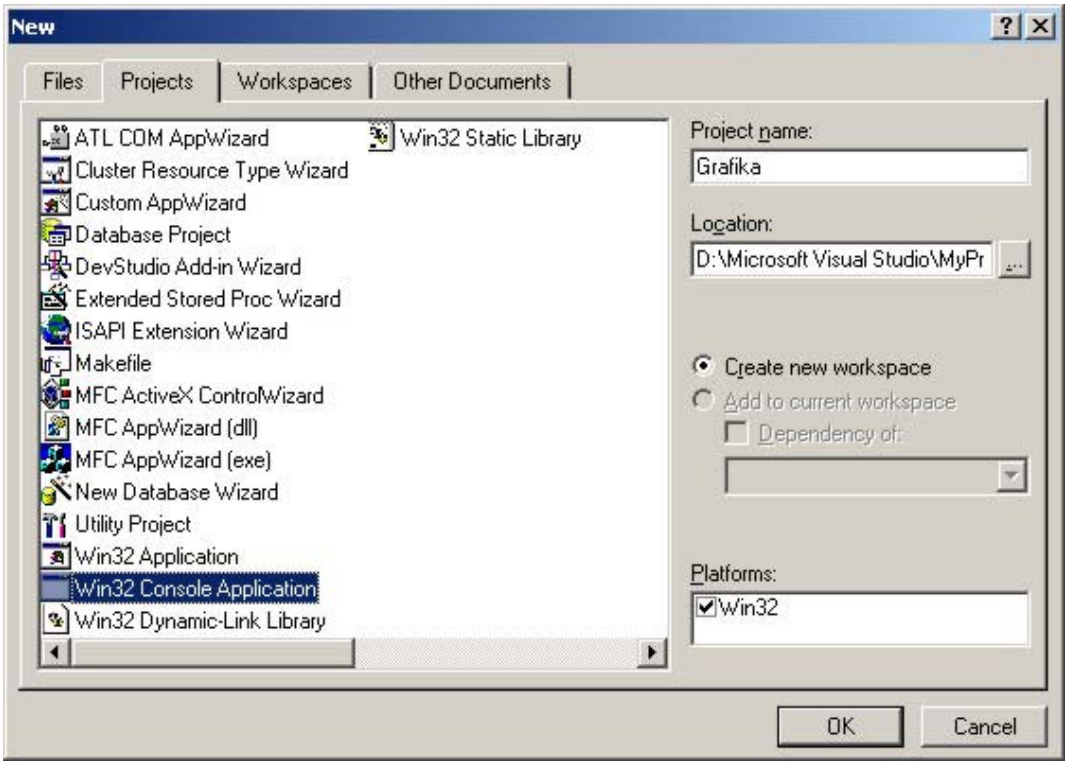
C:\Windows\System\glu32.dll
C:\Windows\System\opengl32.dll
C:\Windows\System\glu32.dll
C:\Windows\System\glut32.dll

Kada je reč o NT mašinama, onda je to lokacija:

C:\Winnt\System32\glu32.dll
C:\Winnt\System32\opengl32.dll
C:\Winnt\System32\glu32.dll
C:\Winnt\System32\glut32.dll

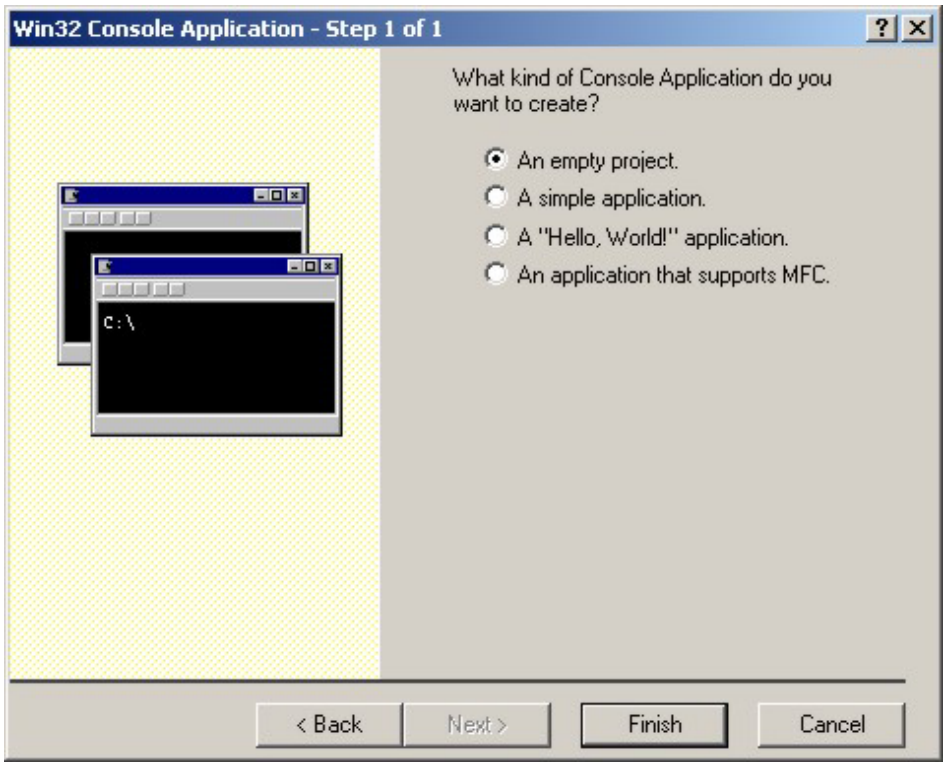
Nakon ovoga sledi posao oko kreiranja projekta unutar programa Visual C++. Procedura je sledeća:

1. Treba startovati program VC++.
2. Nakon toga treba aktivirati ***File > New*** i pojaviće se dijalog koji je prikazan na slici 1.



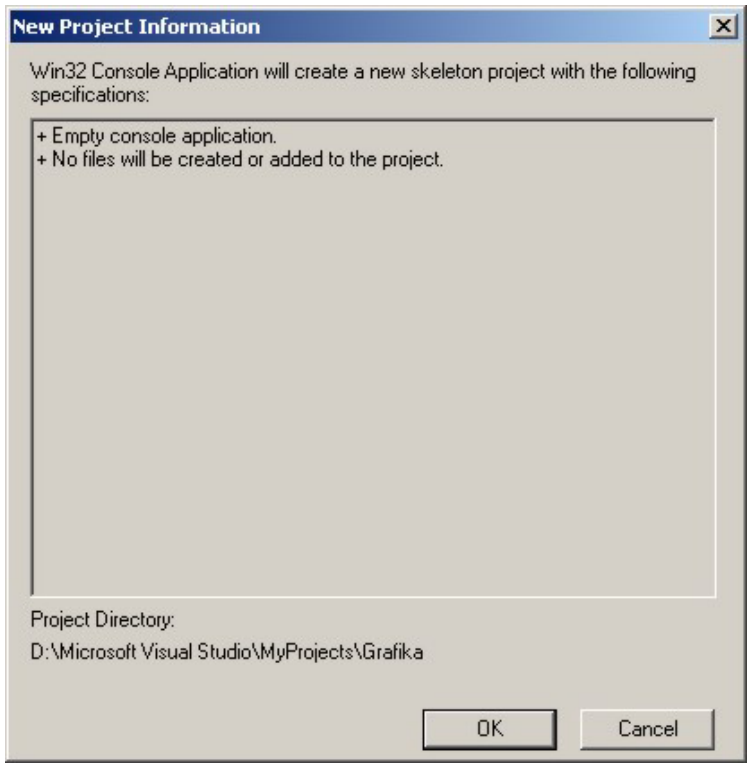
Slika 1

3. U tom dijalogu treba aktivirati karticu *Projects* i treba izabrati *Win32 Console Applications*.
4. U polju *Project name* treba ukucati naziv projekta. U ovom slučaju projekat je nazvan **Grafika**. Nakon pritiska na taster OK korisnik ide dalje. Kao rezultat pojavljuje se dijalog na slici 2.



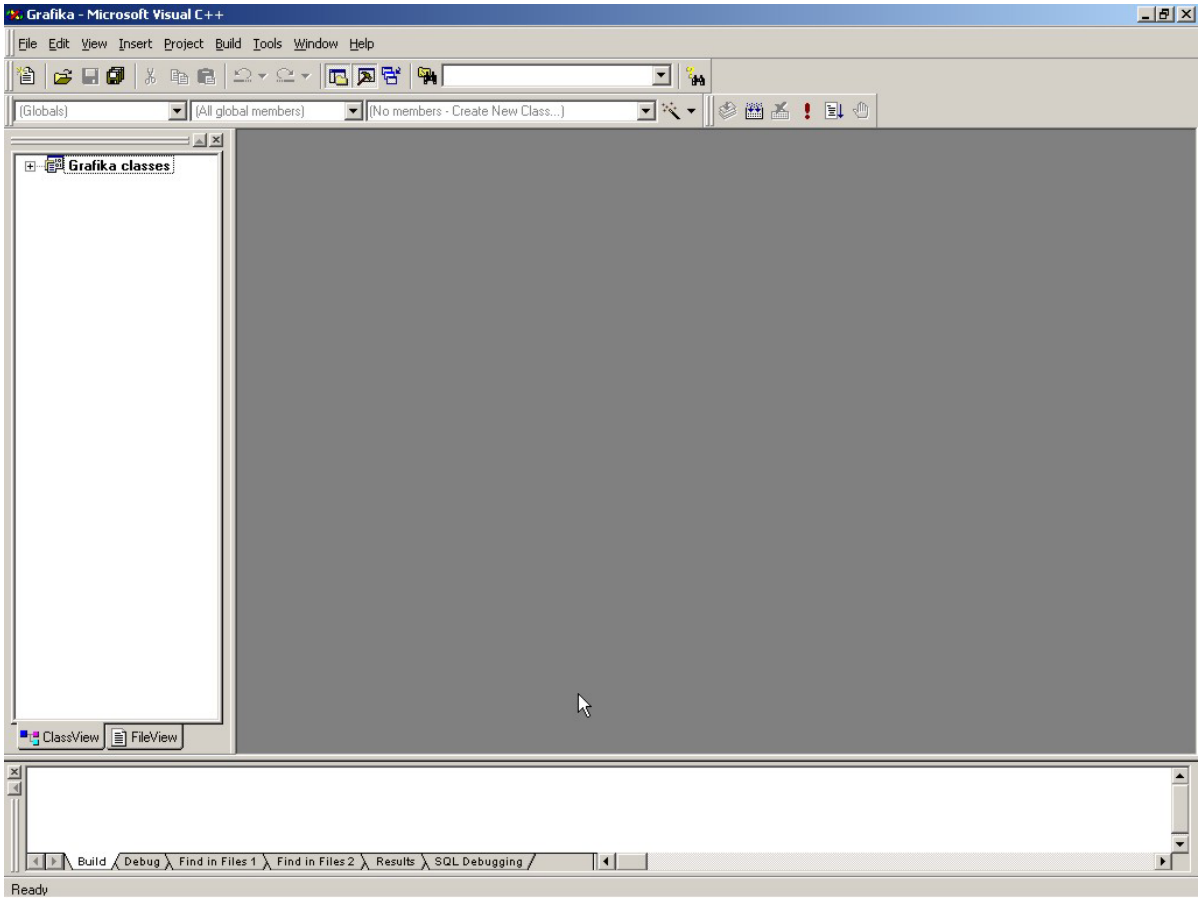
Slika 2

5. U dijalogu pod nazivom *Win32 Console Application – Step 1 of 1* treba izabrati *An empty project*. Nakon toga treba aktivirati taster Finish. Kao rezultat ovoga pojavljuje se slika 3.



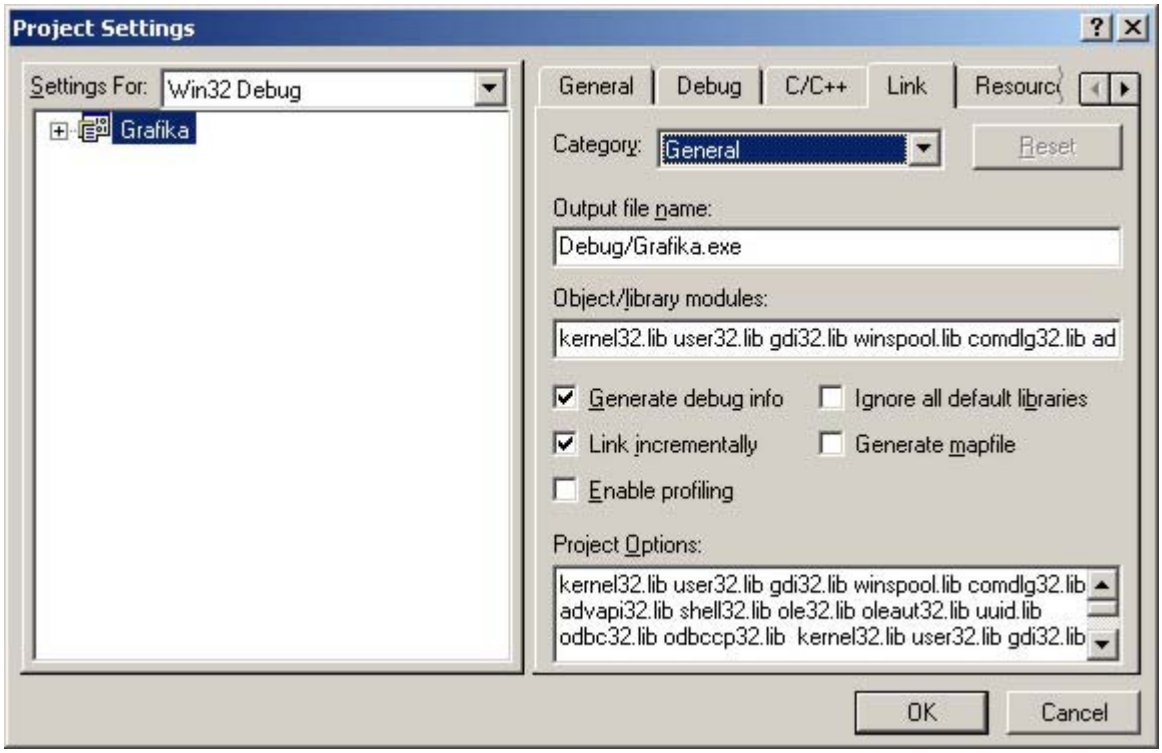
Slika 3

6. Pritiskom na taster OK završava se priča oko kreiranja novog projekta. Izgled radnog ekrana programa VC++ prikazan je na slici 4.



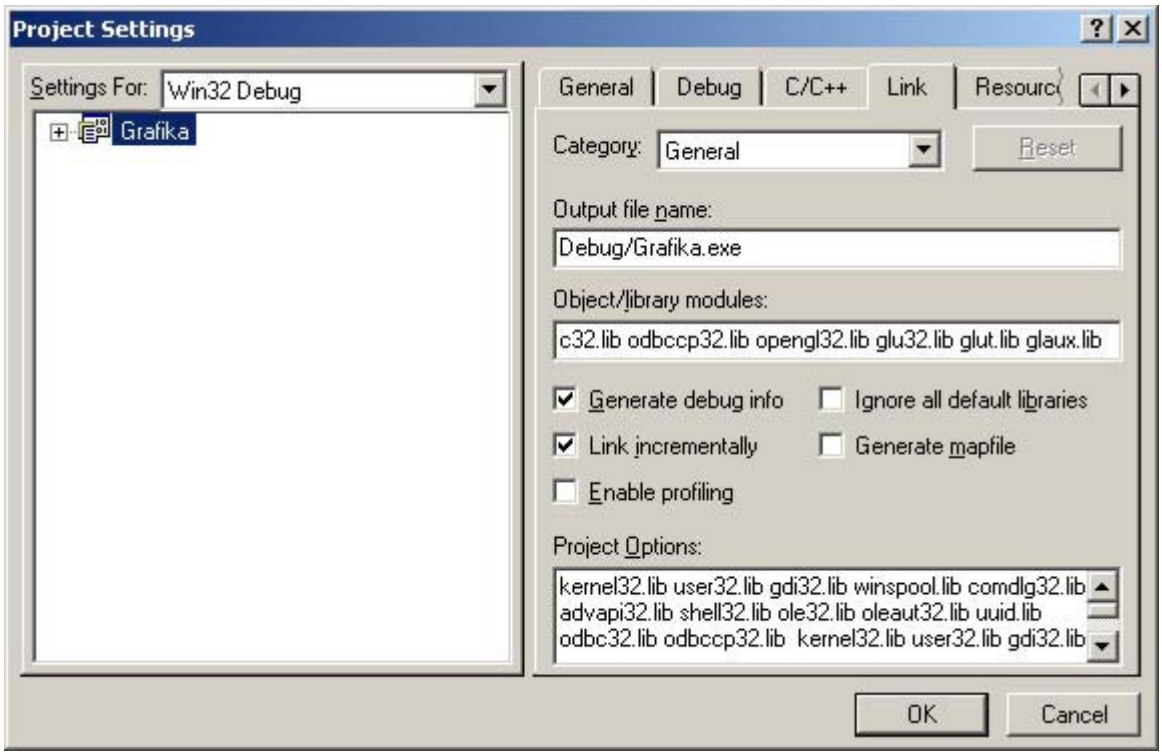
Slika 4

7. Sledeći korak je aktiviranje *Project* padajućeg menija i aktiviranje komande *Settings*. To isto može da se postigne i pritiskom na kombinaciju tastera **Alt+F7**. Unutar dijaloga *Project Settings* treba aktivirati karticu *Link*. Kao rezultat pojavljuje se slika 5.



Slika 5

8. Treba podesitiObject/library modules. Unutar ovog modula treba ubaciti četiri biblioteke: *opengl32.lib*, *glu32.lib*, *glaux.lib* i *glut.lib* (slika 6). Neće sve biblioteke da trebaju uvek, ali od viška glava ne boli.



Slika 6

9. Sada je sve spremno za rad. Korisnik može slobodno da koristi pogodnosti OpenGL-a i GLUT-a.

OpenGL (2)

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazan jednostavan program koji omogućava iscrtavanje kocke u boji.

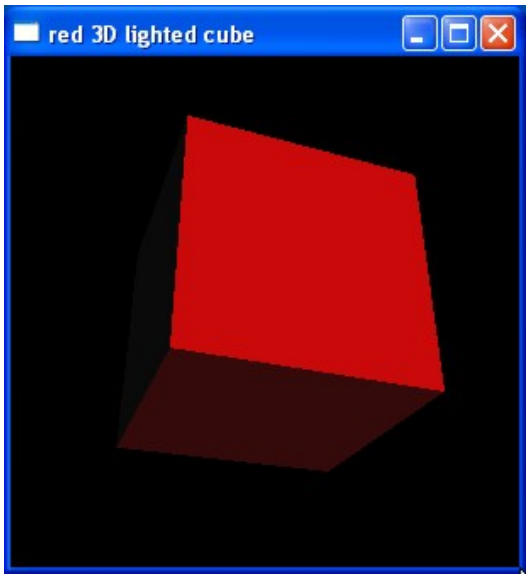
Na početku svakog programa sledi spisak fajlova koje se smeštaju u heder (zaglavlje) programa i koji nose sa sobom potrebne informacije o funkcijama.

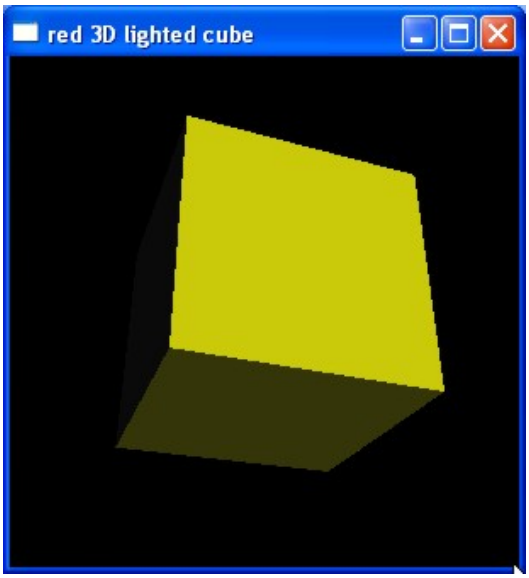
```
#include <GL/glut.h>

/* Definisanje crvenog difuzionog svetla */
GLfloat light_diffuse[] = {1.0, 0.0, 0.0, 1.0};

/* Postavljanje svetla u beskonačnosti */
GLfloat light_position[] = {1.0, 1.0, 1.0, 0.0};

/* Definisanje normala za svih 6 strana kocke */
GLfloat n[6][3] = {
  {-1.0, 0.0, 0.0}, {0.0, 1.0, 0.0}, {1.0, 0.0, 0.0},
  {0.0, -1.0, 0.0}, {0.0, 0.0, 1.0}, {0.0, 0.0, -1.0} };

/* Definisanje verteksa (temena) kocke */
GLint faces[6][4] = {
  {0, 1, 2, 3}, {3, 2, 6, 7}, {7, 6, 5, 4},
  {4, 5, 1, 0}, {5, 6, 2, 1}, {7, 4, 0, 3} };

/* Popunjavanje pomoću verteksa */
GLfloat v[8][3];

void
drawBox(void)
{
  int i;

  for (i = 0; i < 6; i++) {
    glBegin(GL_QUADS);
    glNormal3fv(&n[i][0]);
    glVertex3fv(&v[faces[i][0]][0]);
    glVertex3fv(&v[faces[i][1]][0]);
    glVertex3fv(&v[faces[i][2]][0]);
    glVertex3fv(&v[faces[i][3]][0]);
    glEnd();
  }
}

void
display(void)
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  drawBox();
  glutSwapBuffers();
}

void
init(void)
{

  /* Podešavanje podataka o verteksima kocke */
  v[0][0] = v[1][0] = v[2][0] = v[3][0] = -1;
  v[4][0] = v[5][0] = v[6][0] = v[7][0] = 1;
```

```
  v[0][1] = v[1][1] = v[4][1] = v[5][1] = -1;
  v[2][1] = v[3][1] = v[6][1] = v[7][1] = 1;
  v[0][2] = v[3][2] = v[4][2] = v[7][2] = 1;
  v[1][2] = v[2][2] = v[5][2] = v[6][2] = -1;

  /* Omogućava primenu jednog OpenGL svetlosnog izvora */
  glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
  glLightfv(GL_LIGHT0, GL_POSITION, light_position);
  glEnable(GL_LIGHT0);
  glEnable(GL_LIGHTING);

  /* Upotreba depth buffer-a za uklanjanje nevidljivih površina */
  glEnable(GL_DEPTH_TEST);

  /* Podešavanje pogleda na kocku */
  glMatrixMode(GL_PROJECTION);
  gluPerspective(40.0, 1.0, 1.0, 10.0);
  glMatrixMode(GL_MODELVIEW);
  gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.);

  /* Podešavanje ugla kocke */
  glTranslatef(0.0, 0.0, -1.0);
  glRotatef(60, 1.0, 0.0, 0.0);
  glRotatef(-20, 0.0, 0.0, 1.0);
}

int
main(int argc, char **argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
  glutCreateWindow("red 3D lighted cube");
  glutDisplayFunc(display);
  init();
  glutMainLoop();
  return 0;
}
```

Kao rezultat se pojavljuje prozor sa prikazanom kockom i sa pripadajućim parametrima. Na prvoj slici je originalna kocka, a na drugoj je žuta kocka jer je promenjen jedan broj u definiciji difuzionog svetla:

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





Sledi kompletan listing programa:

```
#include <GL/glut.h>
GLfloat light_diffuse[] = {1.0, 0.0, 0.0, 1.0};
GLfloat light_position[] = {1.0, 1.0, 1.0, 0.0};
GLfloat n[6][3] = {
  {-1.0, 0.0, 0.0}, {0.0, 1.0, 0.0}, {1.0, 0.0, 0.0},
  {0.0, -1.0, 0.0}, {0.0, 0.0, 1.0}, {0.0, 0.0, -1.0} };
GLint faces[6][4] = {
  {0, 1, 2, 3}, {3, 2, 6, 7}, {7, 6, 5, 4},
  {4, 5, 1, 0}, {5, 6, 2, 1}, {7, 4, 0, 3} };
GLfloat v[8][3];

void
drawBox(void)
{
  int i;
  for (i = 0; i < 6; i++) {
    glBegin(GL_QUADS);
    glNormal3fv(&n[i][0]);
    glVertex3fv(&v[faces[i][0]][0]);
    glVertex3fv(&v[faces[i][1]][0]);
    glVertex3fv(&v[faces[i][2]][0]);
    glVertex3fv(&v[faces[i][3]][0]);
    glEnd();
  }
}

void
display(void)
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  drawBox();
  glutSwapBuffers();
}

void
init(void)
{

  v[0][0] = v[1][0] = v[2][0] = v[3][0] = -1;
  v[4][0] = v[5][0] = v[6][0] = v[7][0] = 1;
  v[0][1] = v[1][1] = v[4][1] = v[5][1] = -1;
  v[2][1] = v[3][1] = v[6][1] = v[7][1] = 1;
  v[0][2] = v[3][2] = v[4][2] = v[7][2] = 1;
  v[1][2] = v[2][2] = v[5][2] = v[6][2] = -1;

  glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
  glLightfv(GL_LIGHT0, GL_POSITION, light_position);
  glEnable(GL_LIGHT0);
  glEnable(GL_LIGHTING);
  glEnable(GL_DEPTH_TEST);
  glMatrixMode(GL_PROJECTION);
```

```
  gluPerspective(40.0, 1.0, 1.0, 10.0);
  glMatrixMode(GL_MODELVIEW);
  gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.);
  glTranslatef(0.0, 0.0, -1.0);
  glRotatef(60, 1.0, 0.0, 0.0);
  glRotatef(-20, 0.0, 0.0, 1.0);
}

int
main(int argc, char **argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
  glutCreateWindow("red 3D lighted cube");
  glutDisplayFunc(display);
  init();
  glutMainLoop();
  return 0;
}
```

OpenGL (3)

Pripremio Dragan Cvetković

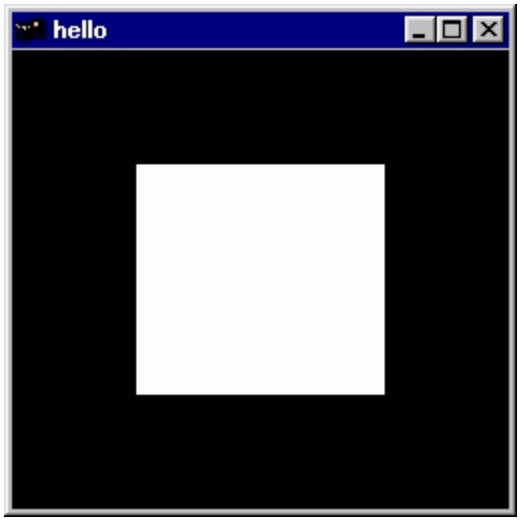
OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazan jednostavan i uobičajen program koji omogućava iscrtavanje kvadrata u beloj boji na crnoj pozadini. Treba napomenuti da korisnik može da zadaje proizvoljnu boju, kao i ostale parametre.

Na početku svakog programa sledi spisak fajlova koje se smeštaju u heder (zaglavlje) programa i koji nose sa sobom potrebne informacije o funkcijama.

```
/* Jednostavan program za upoznavanje sa OpenGL-om */
#include <GL/glut.h>

void display(void)
{

/* "Čiste" se svi pikseli */
   glClear (GL_COLOR_BUFFER_BIT);

/* Iscrtava se beli pravougaonik sa koordinatama temena
 * (0.25, 0.25, 0.0) i (0.75, 0.75, 0.0) */
   glColor3f (1.0, 1.0, 1.0);
   glBegin(GL_POLYGON);
      glVertex3f (0.25, 0.25, 0.0);
      glVertex3f (0.75, 0.25, 0.0);
      glVertex3f (0.75, 0.75, 0.0);
      glVertex3f (0.25, 0.75, 0.0);
   glEnd();

/* Počinje procesiranje OpenGL rutina */
   glFlush ();
}

void init (void)
{

/* Bira se boja za prebrisavanje */
   glClearColor (0.0, 0.0, 0.0, 0.0);

/* Inicijalizuju se koordinate pogleda */
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}

/*
 * Definiše se inicijalna veličina prozora, pozicija i ekranski mod.
 * Otvara se prozor sa hello naslovom.
 * Pozivaju se inicijalne rutine.
 */
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (250, 250);
   glutInitWindowPosition (100, 100);
```

```
   glutCreateWindow ("hello");
   init ();
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
}
```

Kao rezultat se pojavljuje prozor sa prikazanim kvadratom i sa pripadajućim parametrima. Korisnik može da menja sve dostupne parametre, kao što su veličina, pozicija, boja, itd.



Sledi kompletan listing programa:

```
#include <GL/glut.h>

void display(void)
{

   glClear (GL_COLOR_BUFFER_BIT);

   glColor3f (1.0, 1.0, 1.0);
   glBegin(GL_POLYGON);
      glVertex3f (0.25, 0.25, 0.0);
      glVertex3f (0.75, 0.25, 0.0);
      glVertex3f (0.75, 0.75, 0.0);
      glVertex3f (0.25, 0.75, 0.0);
   glEnd();

   glFlush ();
}

void init (void)
{

   glClearColor (0.0, 0.0, 0.0, 0.0);

   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}
```

```
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (250, 250);
   glutInitWindowPosition (100, 100);
   glutCreateWindow ("hello");
   init ();
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
}
```

Povezivanje OpenGL-a sa Visual Studiom.NET

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija (OpenGL je skraćenica od *Open Graphics Language*). Predstavljen je prvi put 1992. godine, i od tada OpenGL predstavlja najčešće korišćeni grafički API (API je skraćenica od *Application Programming Interface*), koji je doneo na hiljade aplikacija za sve vrste računarskh platformi. Pored OpenGL-a, trebalo bi podesiti da se radi i sa GLUT-om (GLUT je skraćenica od *Graphics Language Utility Toolkit*) i u potpunosti je kompatibilan sa OpenGL-om.

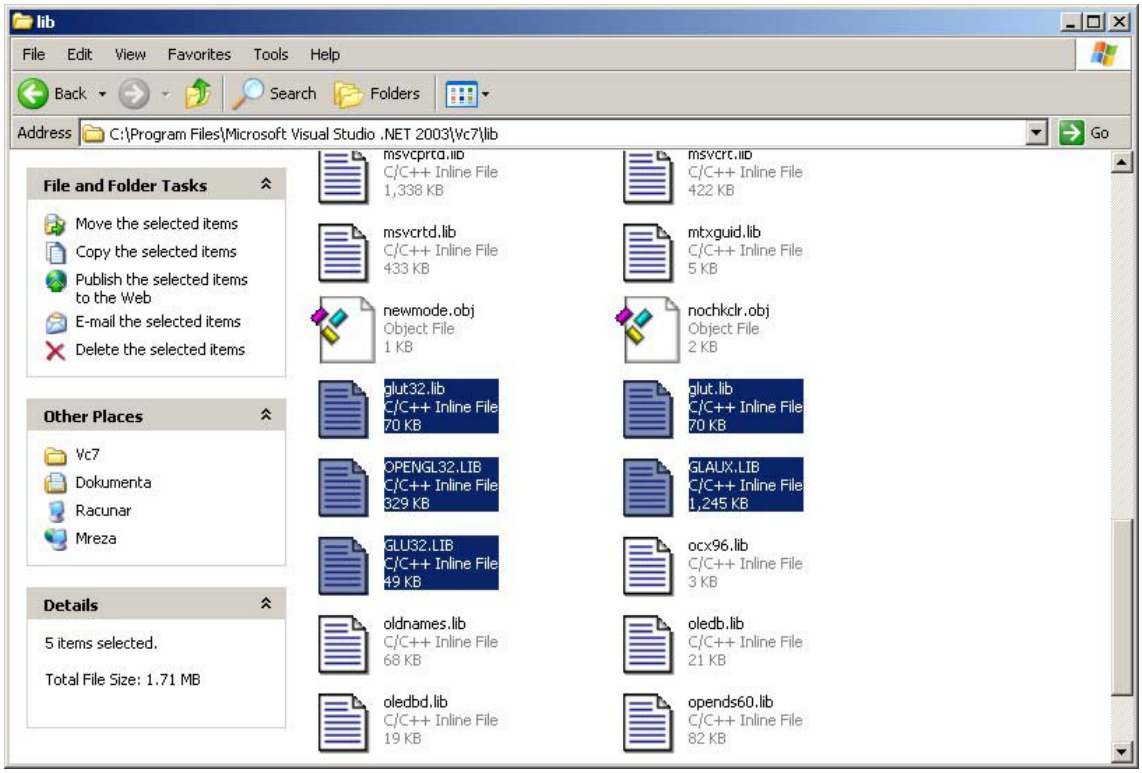
Ovde će biti objašnjeno kako omogućiti da OpenGL i GLUT rade unutar operativnog sistema Windows i uz pomoć programskog jezika Visual C++. U ovom slučaju reč je o jeziku Visual C++.NET, unutar programskog paketa Microsoft Visual Studio.NET. Da bi sve radilo kako treba korisnik mora da skine sa Interneta sledeće fajlove:

- za OpenGL: opengl32.lib, glu32.lib, glaux.lib, gl.h, glaux.h, glu.h, glu32.dll i opengl32.dll;
- za GLUT: glut.dll, glut32.dll, glut.h, glut.lib i glut32.lib.

Nakon toga treba ih smestiti na tačno određene lokacije. Sve biblioteke sa ekstenzijom LIB (opengl32.lib, glu32.lib, glaux.lib, glut.lib i glut32.lib) trebalo bi smestiti na sledeću lokaciju:

C:\Program Files\Microsoft Visual Studio.NET 2003\Vc7\lib\opengl32.lib
C:\Program Files\Microsoft Visual Studio.NET 2003\Vc7\lib\glu32.lib
C:\Program Files\Microsoft Visual Studio.NET 2003\Vc7\lib\glaux.lib
C:\Program Files\Microsoft Visual Studio.NET 2003\Vc7\lib\glut.lib
C:\Program Files\Microsoft Visual Studio.NET 2003\Vc7\lib\glut32.lib

s tim što treba napomenuti da je instalacija odrađena na uobičajeni način, što znači da ništa nije menjano po pitanju podrazumevanih putanja za inatalaciju paketa Visual Studio.NET. Na slici 1 je prikazana lokacija LIB direktorujama i iskopiran sadržaj unutar njega.

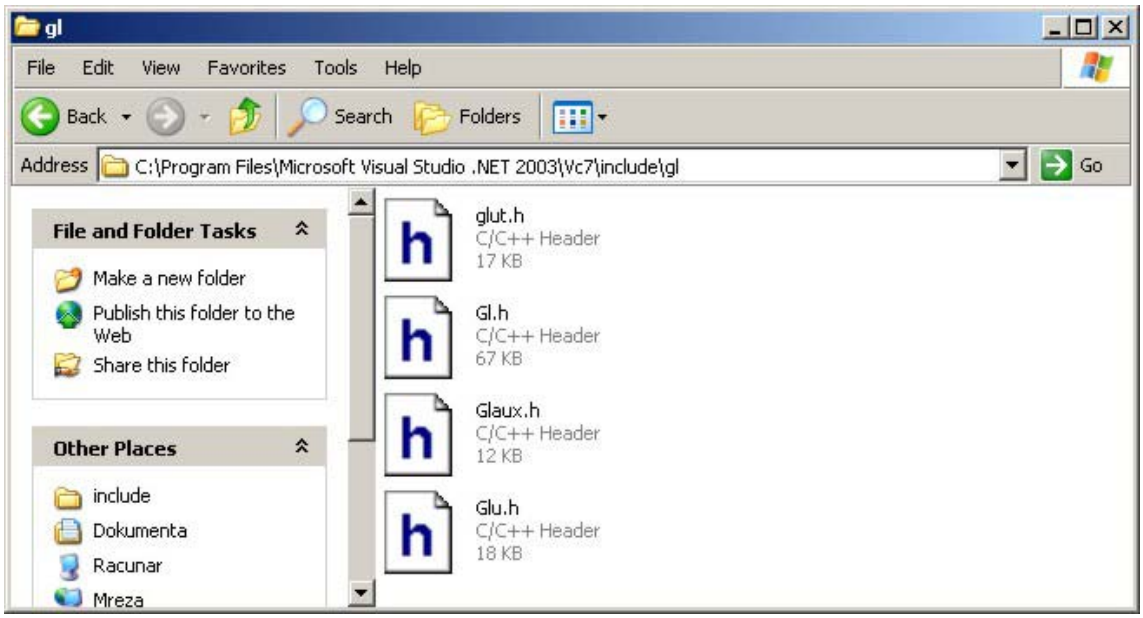


Slika 1

Biblioteke sa ekstenzijom H (gl.h, glaux.h, glu.h i glut.h) trebalo bi smestiti na sledeću lokaciju:

C:\Program Files\Microsoft Visual Studio.NET 2003\Vc7\include\GL\gl.h
C:\Program Files\Microsoft Visual Studio.NET 2003\Vc7\include\GL\glaux.h
C:\Program Files\Microsoft Visual Studio.NET 2003\Vc7\include\GL\glu.h
C:\Program Files\Microsoft Visual Studio.NET 2003\Vc7\include\GL\glut.h

s tim što treba napomenuti da je instalacija odrađena na uobičajeni način, što znači da ništa nije menjano po pitanju podrazumevanih putanja za inatalaciju paketa Visual Studio.NET. Treba napomenuti jednu stvar. Ako direktorijum GL nije formiran, onda korisnik treba da ga formira i da u njega iskopira .H fajlove. Na slici 2 je prikazana lokacija GL direktorujama i iskopiran sadržaj unutar njega.



Slika 2

Biblioteke sa ekstenzijom DLL (glu32.dll, opengl32.dll, glut.dll i glut32.dll) trebalo bi smestiti u sistemske direktorijume unutar dirketorijuma gde se nalazi instaliran operativni sistem. Kada je reč o klasičnim Windows operativnim sistemima, onda je to lokacija:

C:\Windows\System\glu32.dll
C:\Windows\System\opengl32.dll
C:\Windows\System\glu32.dll
C:\Windows\System\glut32.dll

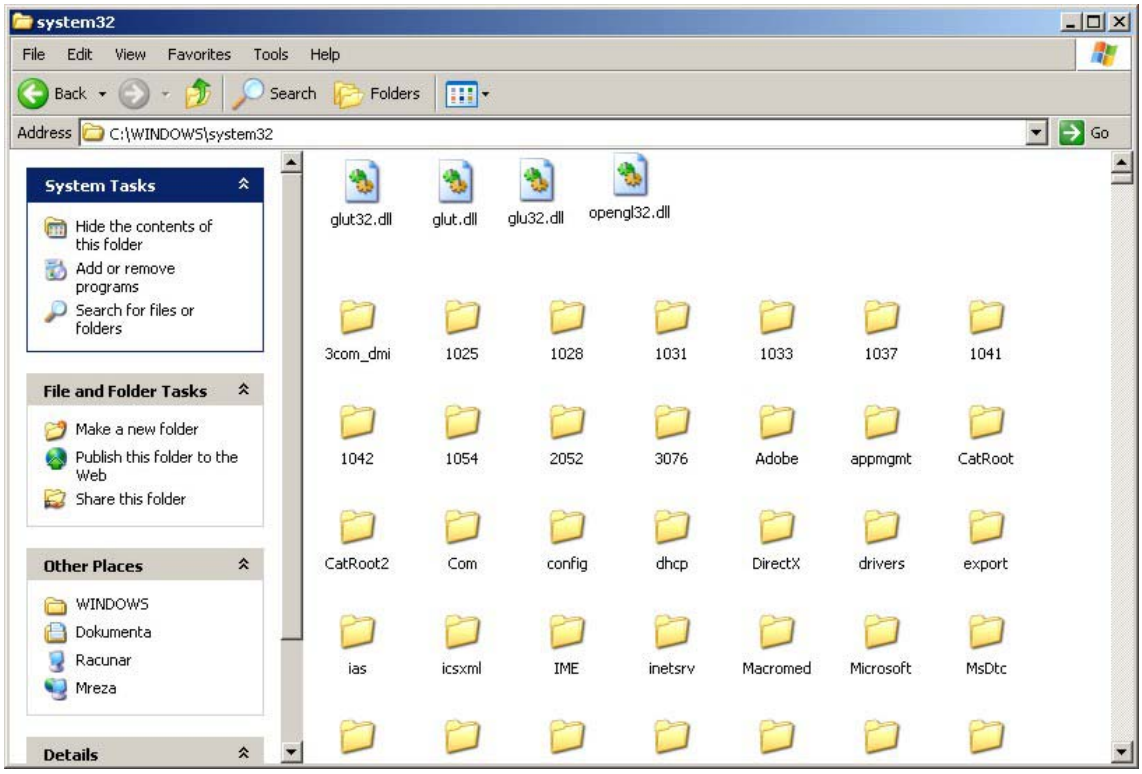
Kada je reč o NT mašinama, onda je to lokacija:

C:\Winnt\System32\glu32.dll
C:\Winnt\System32\opengl32.dll
C:\Winnt\System32\glu32.dll
C:\Winnt\System32\glut32.dll

ili

C:\Windows\System32\glu32.dll
C:\Windows\System32\opengl32.dll
C:\Windows\System32\glu32.dll
C:\Windows\System32\glut32.dll

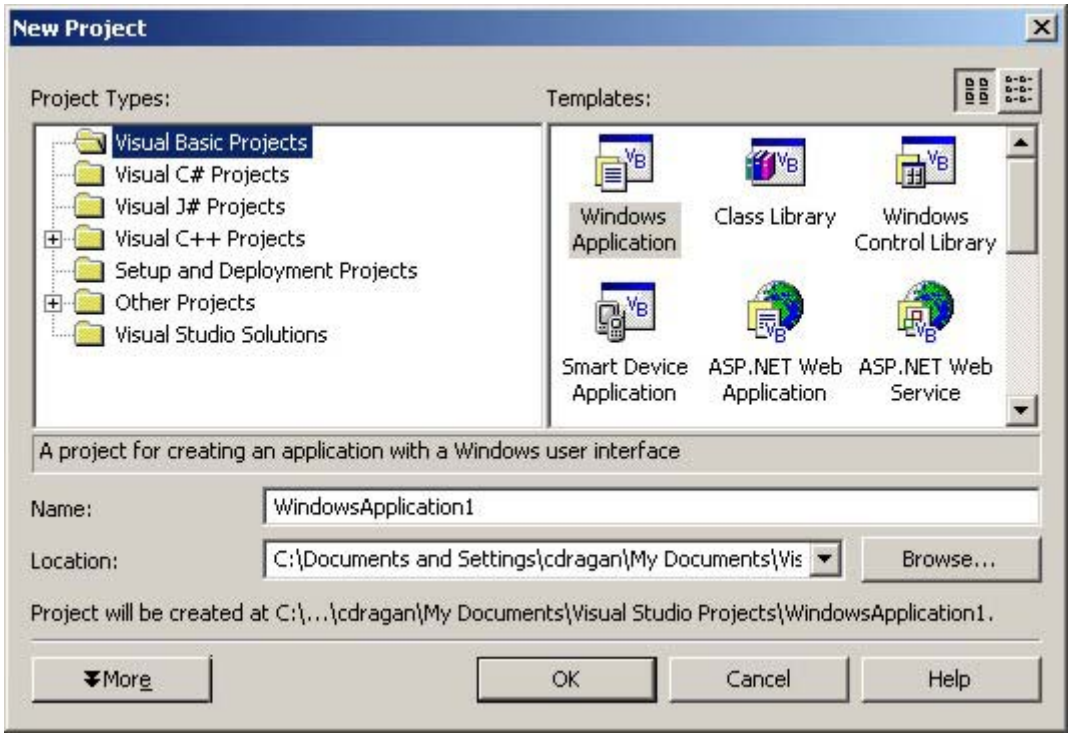
Na slici 3 prikazan je iskopiran sadržaj u System32 direktorijum.



Slika 3

Nakon ovoga sledi posao oko kreiranja projekta unutar programa Visual C++. Procedura je sledeća:

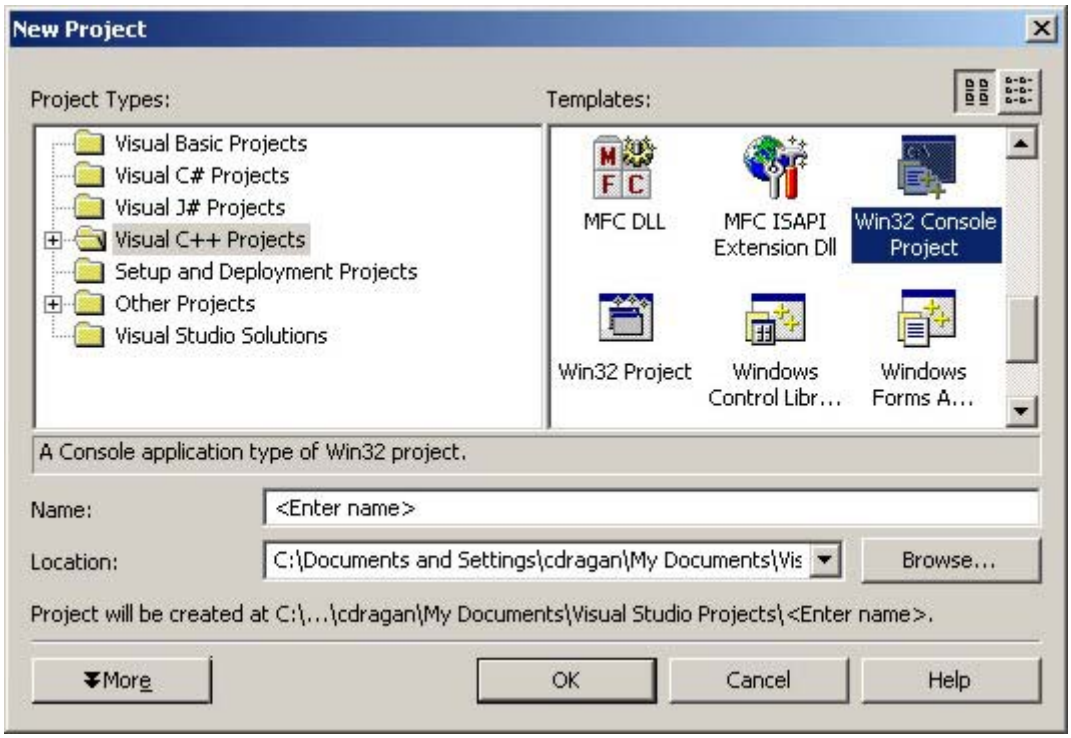
1. Treba startovati program Visual Studio.NET.
2. Nakon toga treba aktivirati ***File > New > Project*** i pojaviće se dijalog koji je prikazan na slici 4.



Slika4

3. U tom dijalogu ne levoj strani treba aktivirati *Visual C++ Projects* opciju (slika 5), a sa desen strane treba aktivirati opciju *Win32 Console Project* (slika 6).

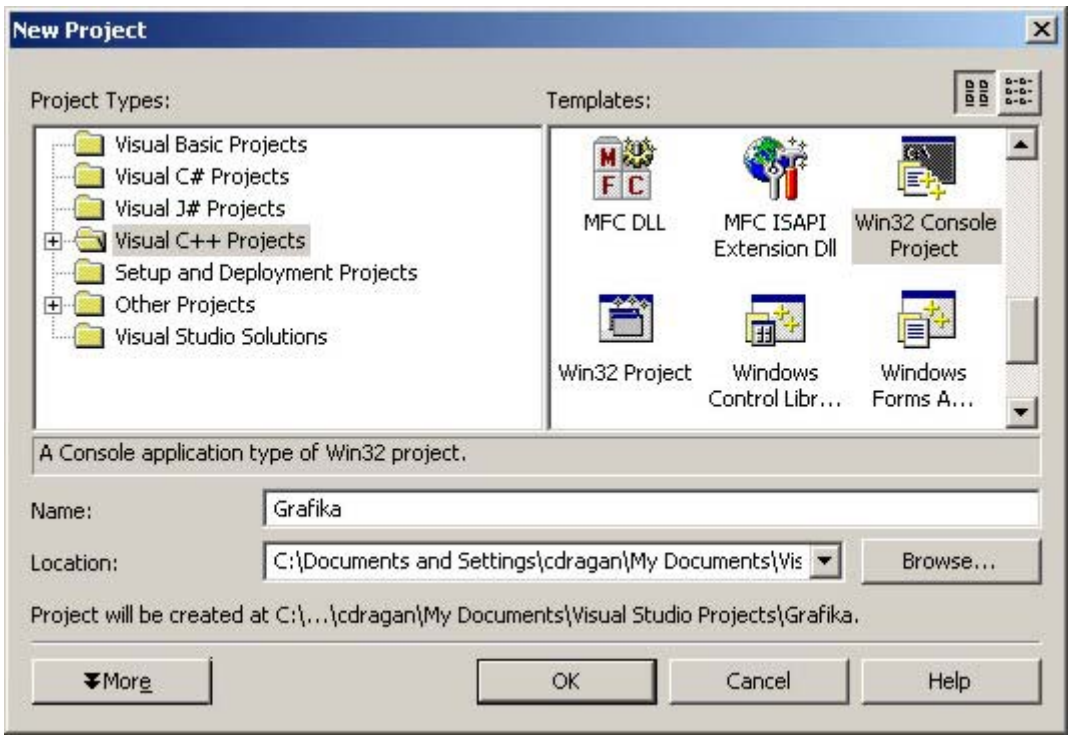
Slika 5



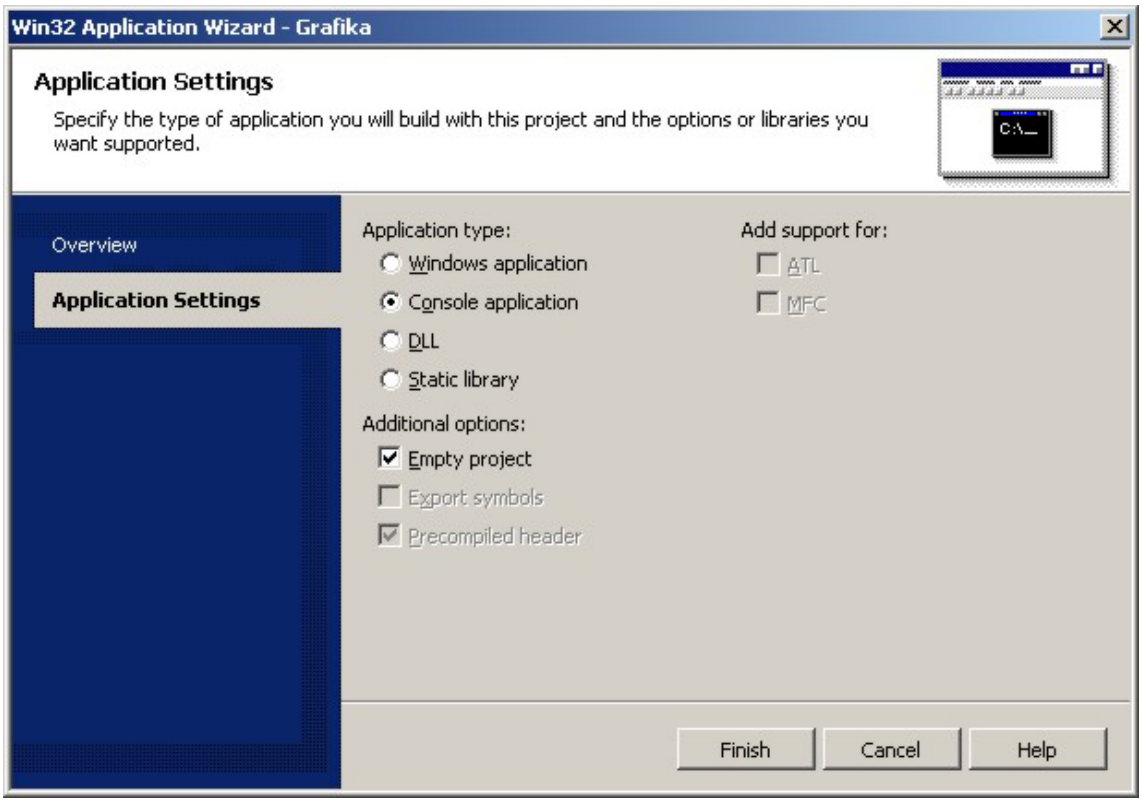
Slika 6

4. U polju *Name* treba ukucati naziv projekta. U ovom slučaju projekat je nazvan **Grafika**. U polju *Location* treba definisati putanju gde će se odgovarajući fajlovi smestiti. Ako korisnik nema posebnih želja ovi fajlovi se smeštaju u direktorijum *Visual Studio Projects* unutar direktorijuma *My Documents* (u ovom slučaju je *C:\Documents and Settings\cdragan\My Documents\Visual Studio Projects\Grafika*, slika 7). Pritiskom na taster OK ide se dalje.

5. Pojavljuje se prozor *Win32 Application Wizard* (slika 8). Treba aktivirati karticu *Application Settings* sa leve strane. U njoj bi trebalo da bude aktivirana opcija *Console application* unutar oblasti *Application type*, a u oblasti *Additional options* treba markirati polje *Empty project*.

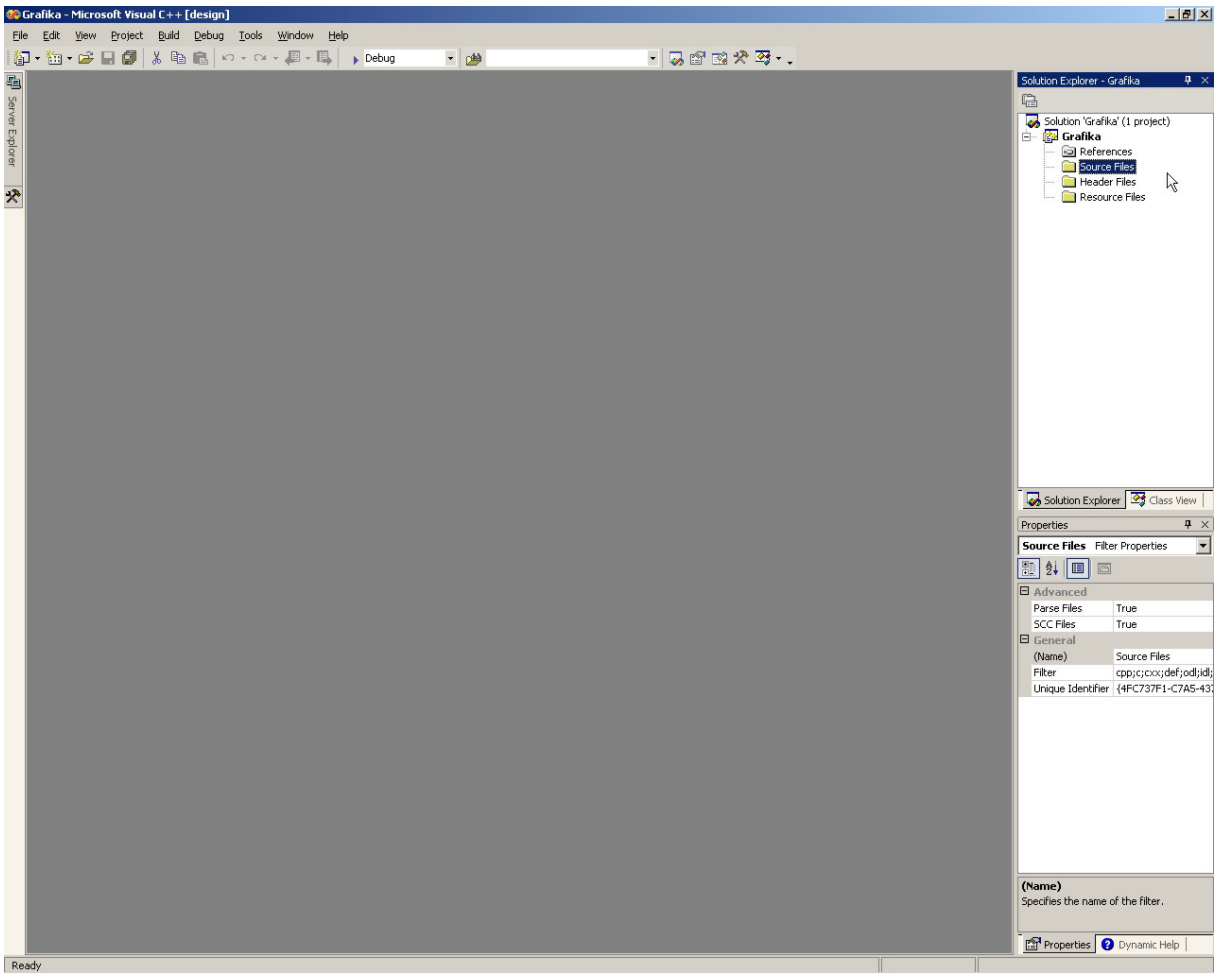


Slika 7

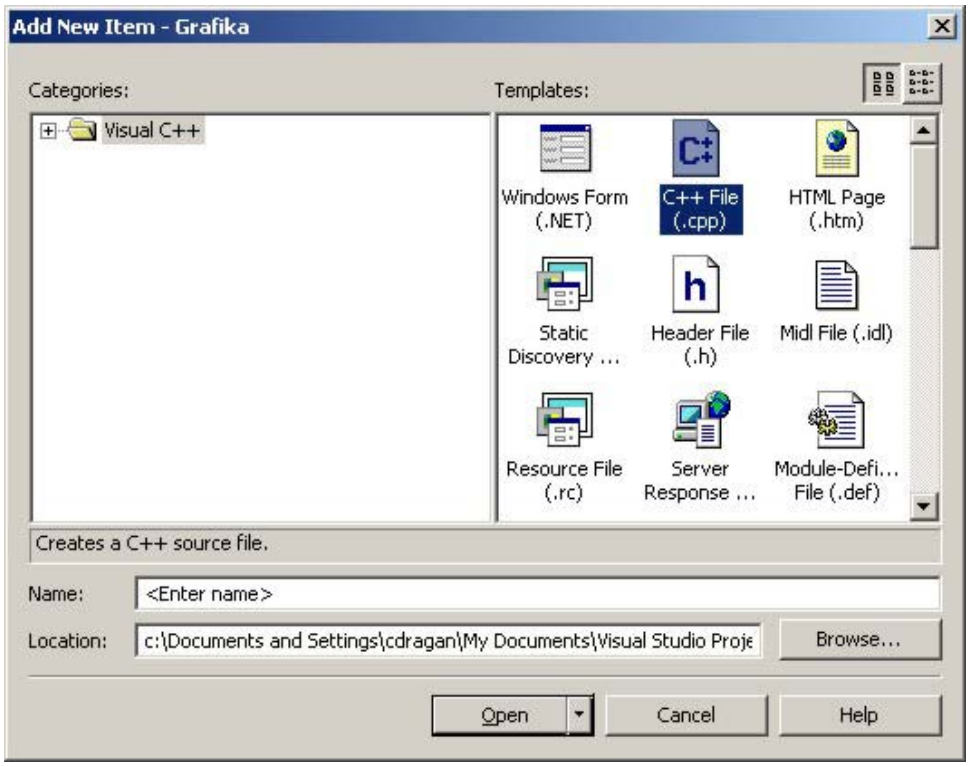


Slika 8

6. Nakon toga treba aktivirati taster *Finish*. Kao rezultat ovoga pojavljuje se slika 9.
7. Na ovaj način novi projekat je „otvoren“. Trebalo bi markirati desnim tasterom miša *Source Files* u desnom gornjem uglu prozora, i u kontekstnom meniju treba aktivirati *Add* > *Add Item* > *C++ file* (taj prozor je prikazan na slici 10).

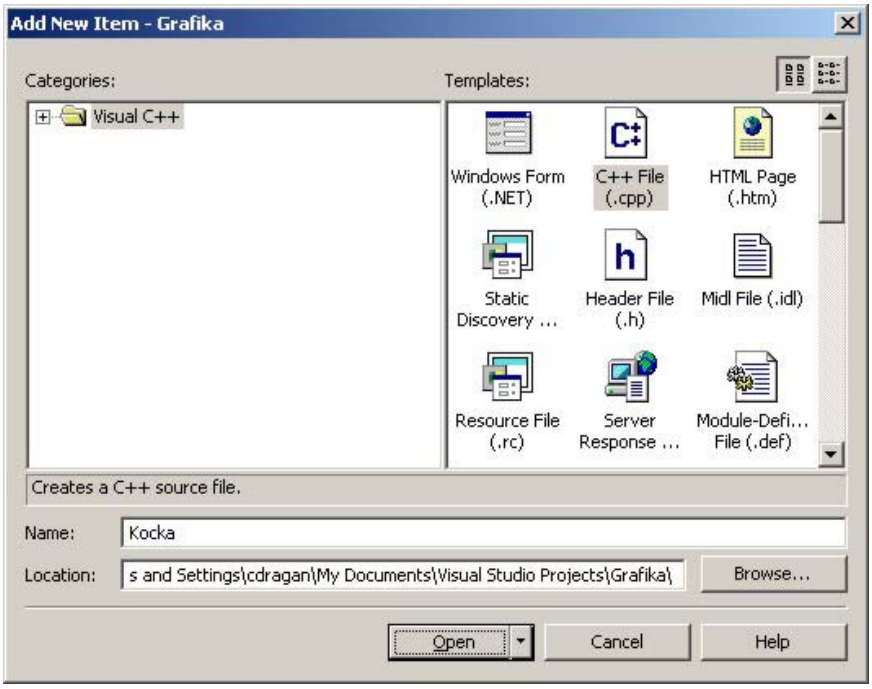


Slika 9



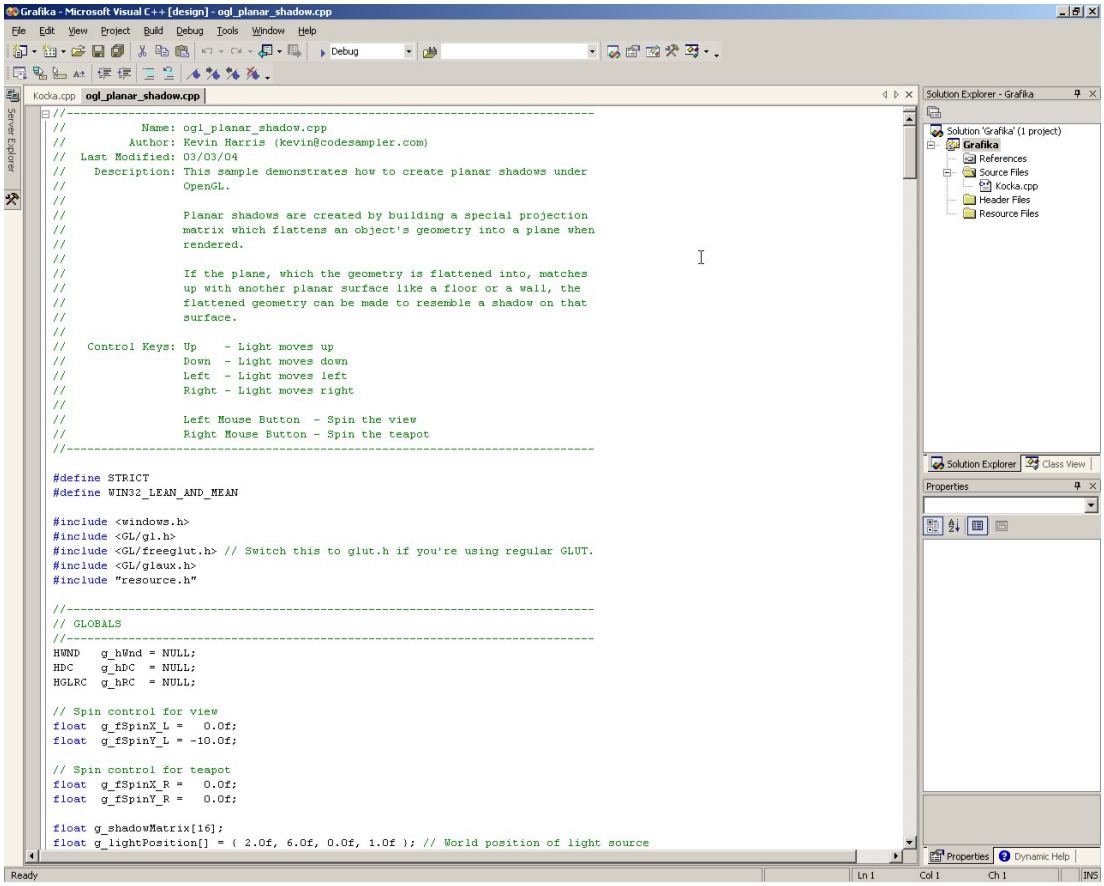
Slika 10

8. U polju *Name* treba definisati naziv fajla koji će „bitisati“ unutar otvorenog projekta. U ovom slučaju nazvan je fajl **Kocka** (slika 11).



Slika 11

9. Korisnik sada može da iskopira sadržaj kôda bilo kog programa, koji treba da se odradi unutar otvorenog projekta. Tek sada se dolazi do kompletnog izgleda radnog ekrana sa učitanim kôdom odgovarajućeg programa. Taj ekran je prikazan na slici 12.



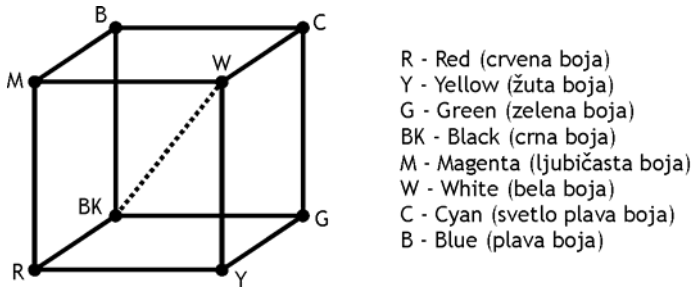
Slika 12

10. Sada je sve spremno za rad. Korisnik može slobodno da koristi pogodnosti OpenGL-a i GLUT-a. Za kompajliranje programa dovoljno je aktivirati funkcijski taster **F5**.

OpenGL (5) – RGB model boje u obliku kocke

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazan jednostavan program koji omogućava da se iscrta kocka u čijim temenima se nalaze tačke određene boje kako je te predviđeno RGB modelom boja. RGB model u obliku kocke prikazan je na slici 1.



Slika 1

Komanda koja omogućava definisanje boja unutar OpenGL-a je **glColor3f (R, G, B)**, s tim što vrednosti nijansi boja „idu“ od 0 do 1: R (crvena) od 0.0 do 1.0, G (zelena) od 0.0 do 1.0 i B (plava) od 0.0 do 1.0. Sledi spisak osnovnih boja:

glColor3f (0.0, 0.0, 0.0) – CRNA boja (*black*)
glColor3f (1.0, 0.0, 0.0) – CRVENA boja (*red*)
glColor3f (0.0, 1.0, 0.0) – ZELENA boja (*green*)
glColor3f (1.0, 1.0, 0.0) – ŽUTA boja (*yellow*)
glColor3f (0.0, 0.0, 1.0) – PLAVA boja (*blue*)
glColor3f (1.0, 0.0, 1.0) – LJUBIČASTA boja (*magenta*)
glColor3f (0.0, 1.0, 1.0) – SVETLO PLAVA boja (*cyan*)
glColor3f (1.0, 1.0, 1.0) – BELA boja (*white*)

Bitan je redosled u sintaksi što se definisanja boja. Crta objekte A i B crvenom bojom, a objekat C se iscrtava plavom bojom. Četvrta linija koja setuje zelenu boju za crtanje je čisto gubljenje vremena i prostora.

```
set_current_color (red);
draw_object (A);
draw_object (B);
set_current_color (green);
set_current_color (blue);
draw_object (C);
```

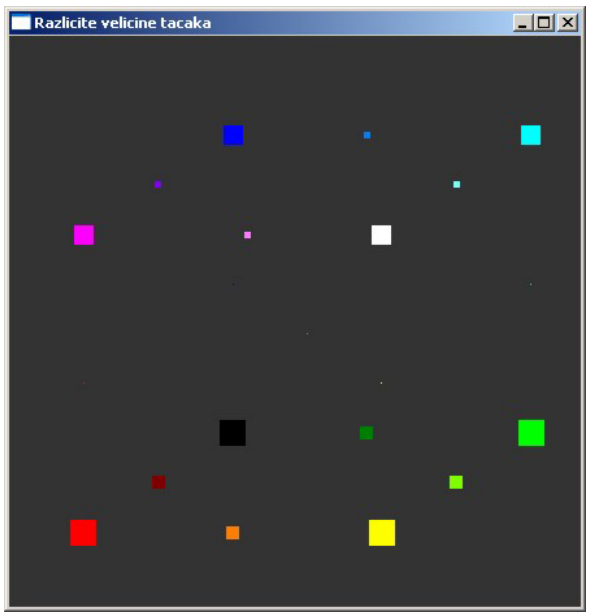
Tačke su prezentirane kompletom koordinata koje formiraju TEME ili VERTEKS. Temena su specificirana od strane korisnika kao dvodimenzionalne tačke (samo X i Y koordinata), s tim što se podrazumeva da je Z koordinata jednaka 0 (nuli). Sintaksa za kreiranje 2D tačaka (tačaka u ravni) je:

```
glBegin(GL_POINTS);
   glVertex2f (x, y);
glEnd();
```

Sintaksa za kreiranje 3D tačaka (tačaka u prostoru) je:

```
glBegin(GL_POINTS);
   glVertex3f (x, y, z);
glEnd();
```

Komanda za podešavanje veličine same tačke je **glPointSize (20.0)**. Vrednost u zagradi je vrednost u PIKSELIMA, tako da ako je vrednost 1 (1.0), to znači da se iscrtava kvadrat dužine stranice 1 piksel, ako je vrednost 2, onda se iscrtava kvadrat čija je stranica dužine 2 piksela, itd (slika 2).

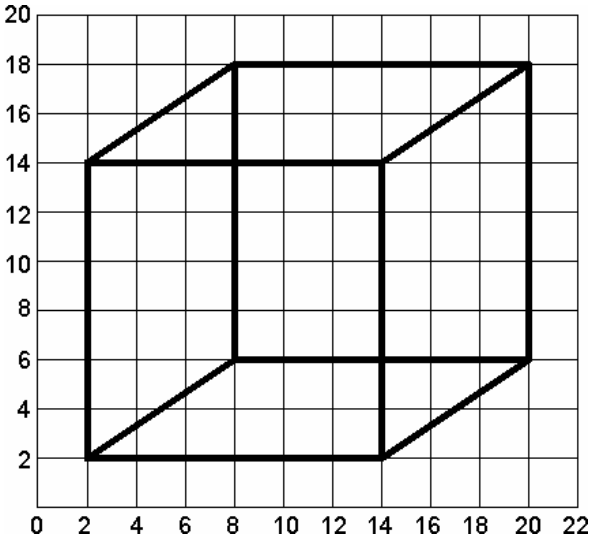


Slika 2

U OpenGL-u termin LINE znači da je reč o LINIJSKOM SEGMENTU, što znači da nije reč o pravoj, koja u matematici predstavlja liniju kojoj se ne zna ni početna ni krajnja tačka. Ovo je najlakši način da se definiše povezana serija pravolinijskih segmenata, čak i zatvorenih. U svim slučajevima, pravolinijski segment je definisan koordinatama krajnjih tačaka. Tri komande koje omogućavaju iscrtavanje linija su:

- **GL_LINES** – par temena koji interpretiraju individualni pravolinijski segment
- **GL_LINE_STRIP** – serija povezanih pravolinijskih segmenata
- **GL_LINE_LOOP** – isto kao prethodna komanda, sa dodatnim segmentom između krajnjeg i početnog temena

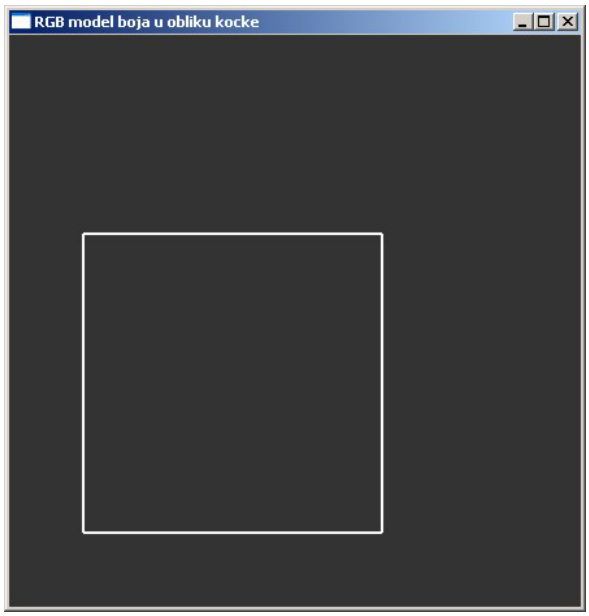
Komanda za podešavanje širine ("debljine") same linije je **glLineWidth (2.0)**. Vrednost u zagradi je vrednost u PIKSELIMA, tako da ako je vrednost 1 (1.0), to znači da se iscrtava linija čija je širina ("debljina") 1 piksel, ako je vrednost 2, onda se iscrtava linija čija je širina ("debljina") 2 piksela, itd.



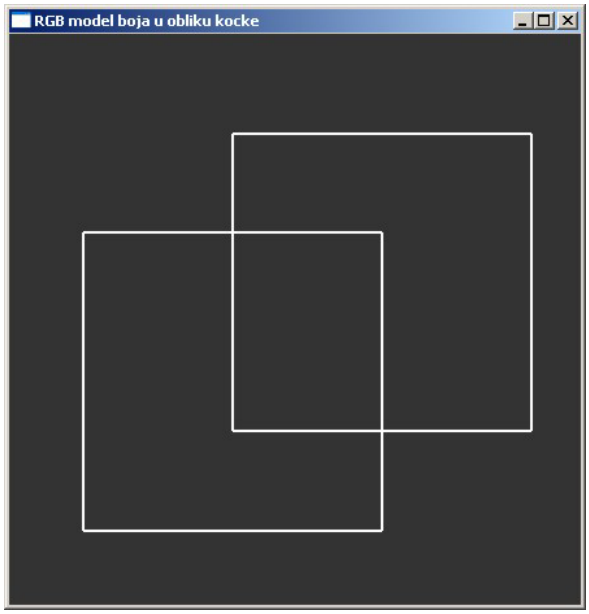
Slika 3

Na slici 3 prikazana je geometrija buduće kocke. Sledi sintaksa programa sa komentarima uz propratne slike.

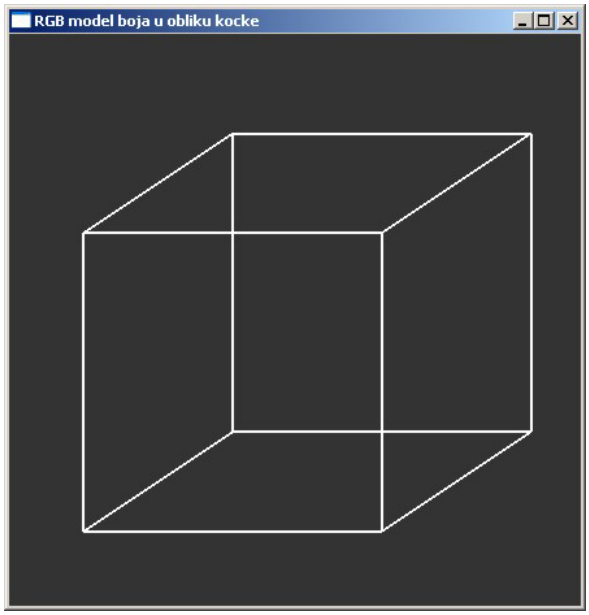
```
/*
 * Ovde je uradjen RGB model boja u obliku kocke
 *       Dragan Cvetkovic, 14.12.2004.
 */

#include <GL/glut.h>

void display(void)

{
/* brise sve piksele iz bafera  */
   glClear (GL_COLOR_BUFFER_BIT);

/* Crta se prednji kvadrat RGB kocke */
   glColor3f (1.0, 1.0, 1.0);
   glLineWidth (2.0);
   glBegin(GL_LINE_LOOP);
      glVertex2f (2.0, 2.0);
      glVertex2f (14.0, 2.0);
      glVertex2f (14.0, 14.0);
      glVertex2f (2.0, 14.0);
   glEnd();
```



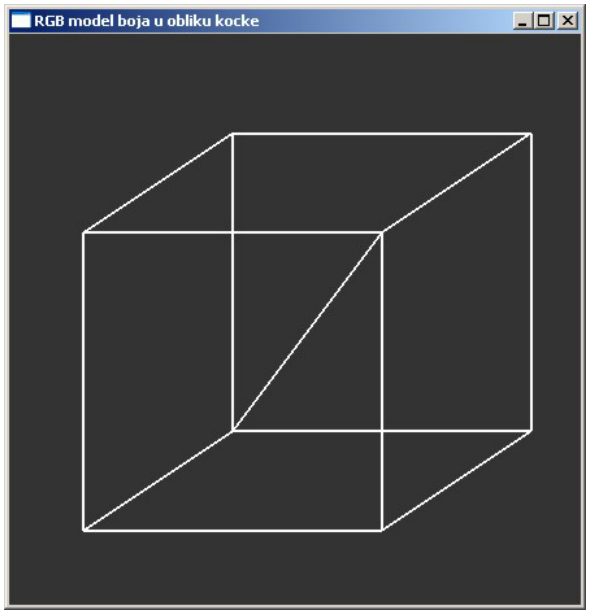

```
/* Crta se zadnji kvadrat RGB kocke */
   glColor3f (1.0, 1.0, 1.0);
   glBegin(GL_LINE_LOOP);
      glVertex2f (8.0, 6.0);
      glVertex2f (20.0, 6.0);
      glVertex2f (20.0, 18.0);
      glVertex2f (8.0, 18.0);
   glEnd();
```



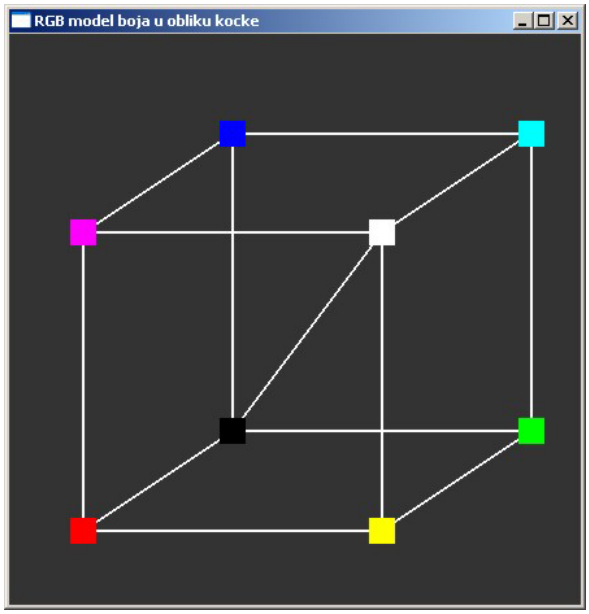

```
/* Crtaju se bocne stranice RGB kocke */
   glColor3f (1.0, 1.0, 1.0);
   glBegin(GL_LINES);
      glVertex2f (2.0, 2.0);
      glVertex2f (8.0, 6.0);
        glVertex2f (14.0, 2.0);
      glVertex2f (20.0, 6.0);
        glVertex2f (2.0, 14.0);
      glVertex2f (8.0, 18.0);
        glVertex2f (14.0, 14.0);
      glVertex2f (20.0, 18.0);
   glEnd();
```



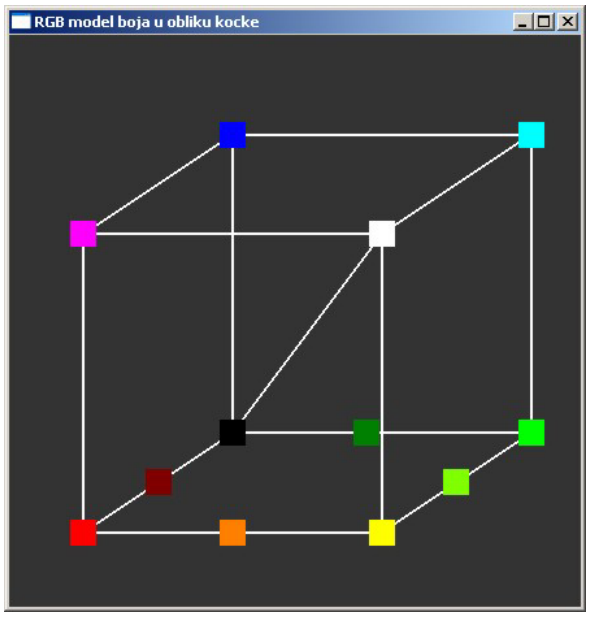

```
/* Crta se prostorna dijagonala od crne ka beloj boji */
   glColor3f (1.0, 1.0, 1.0);
   glBegin(GL_LINES);
      glVertex2f (8.0, 6.0);
      glVertex2f (14.0, 14.0);
   glEnd();
```



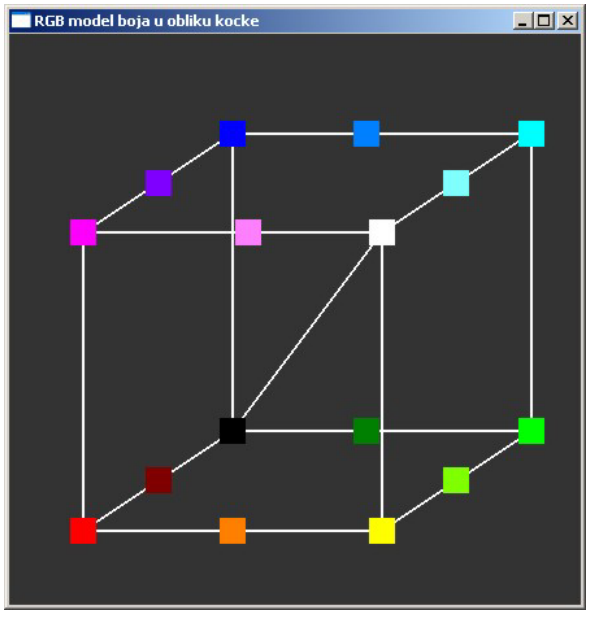

```
/* Crtaju se temena RGB kocke */
   glPointSize (20.0);
   glBegin(GL_POINTS);
        glColor3f (1.0, 0.0, 0.0);
      glVertex2f (2.0, 2.0);
        glColor3f (1.0, 1.0, 0.0);
      glVertex2f (14.0, 2.0);
        glColor3f (0.0, 1.0, 0.0);
      glVertex2f (20.0, 6.0);
        glColor3f (0.0, 0.0, 0.0);
      glVertex2f (8.0, 6.0);
        glColor3f (1.0, 1.0, 1.0);
      glVertex2f (14.0, 14.0);
        glColor3f (1.0, 0.0, 1.0);
      glVertex2f (2.0, 14.0);
        glColor3f (0.0, 1.0, 1.0);
      glVertex2f (20.0, 18.0);
        glColor3f (0.0, 0.0, 1.0);
      glVertex2f (8.0, 18.0);
```



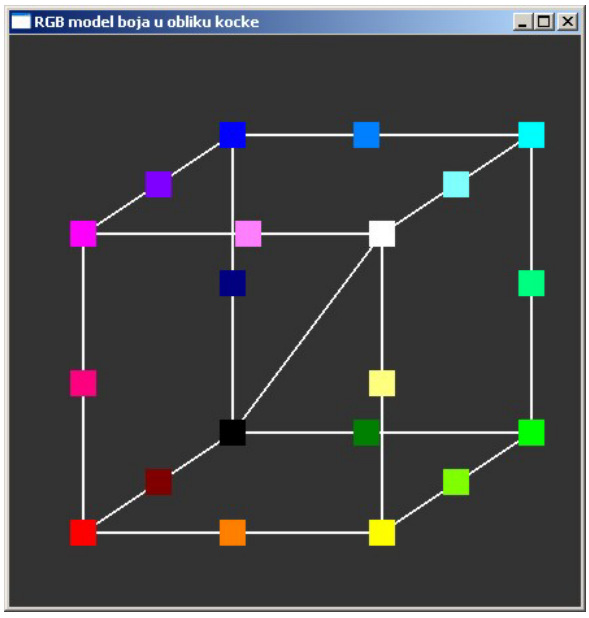
```
/* Crtaju se nijanse boja na sredinama stranica donje osnove */
        glColor3f (0.5, 1.0, 0.0);
      glVertex2f (17.0, 4.0);
        glColor3f (1.0, 0.5, 0.0);
      glVertex2f (8.0, 2.0);
        glColor3f (0.0, 0.5, 0.0);
      glVertex2f (13.4, 6.0);
        glColor3f (0.5, 0.0, 0.0);
      glVertex2f (5.0, 4.0);
```



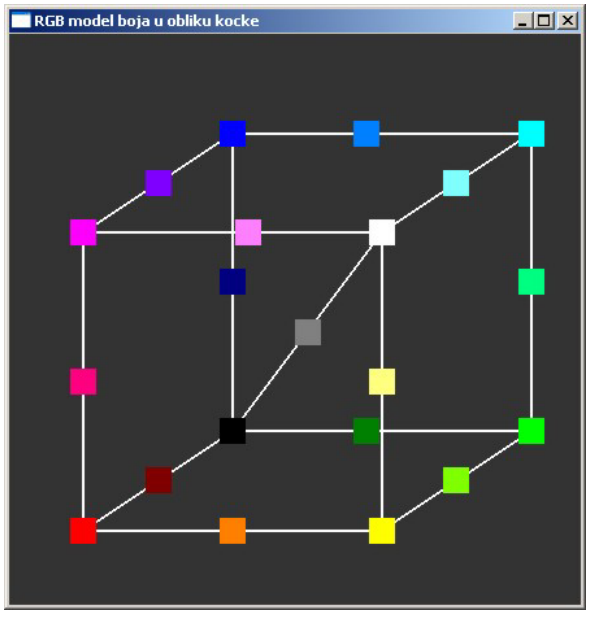

```
/* Crtaju se nijanse boja na sredinama stranica gornje osnove */
        glColor3f (0.5, 1.0, 1.0);
      glVertex2f (17.0, 16.0);
        glColor3f (0.0, 0.5, 1.0);
      glVertex2f (13.4, 18.0);
        glColor3f (0.5, 0.0, 1.0);
      glVertex2f (5.0, 16.0);
        glColor3f (1.0, 0.5, 1.0);
      glVertex2f (8.6, 14.0);
```



```
/* Crtaju se nijanse boja na sredinama vertikalnih stranica */
        glColor3f (1.0, 1.0, 0.5);
      glVertex2f (14.0, 8.0);
        glColor3f (0.0, 1.0, 0.5);
      glVertex2f (20.0, 12.0);
        glColor3f (0.0, 0.0, 0.5);
      glVertex2f (8.0, 12.0);
        glColor3f (1.0, 0.0, 0.5);
      glVertex2f (2.0, 8.0);
```




```
/* Crta se siva nijansa boje na sredini prostorne dijagonale */
        glColor3f (0.5, 0.5, 0.5);
      glVertex2f (11.0, 10.0);
   glEnd();
```




```
/* Izvrsava GL komande u konacnom vremenu */
   glFlush ();
}

void init (void)
```

```
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
   glClearColor (0.2, 0.2, 0.2, 0.0);

/* Definisu se vrednosti pogleda */
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(-1.0, 22.0, -1.0, 22.0, -1.0, 1.0);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * "RGB model boja u obliku kocke" u naslovu prozora.
 * Pozivaju se inicijalne rutine. Registruje se Callback
 * funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (440, 440);
   glutInitWindowPosition (400, 200);
   glutCreateWindow ("RGB model boja u obliku kocke");
   init ();
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
/* ANSI C zahteva odgovor kroz pritisak bilo kog tastera. */
}
```

Sledi kompletan listing programa bez propratnih komentara:

```
#include <GL/glut.h>
void display(void)
{
   glClear (GL_COLOR_BUFFER_BIT);
   glColor3f (1.0, 1.0, 1.0);
   glLineWidth (2.0);
   glBegin(GL_LINE_LOOP);
      glVertex2f (2.0, 2.0);
      glVertex2f (14.0, 2.0);
      glVertex2f (14.0, 14.0);
      glVertex2f (2.0, 14.0);
   glEnd();
   glColor3f (1.0, 1.0, 1.0);
   glBegin(GL_LINE_LOOP);
      glVertex2f (8.0, 6.0);
      glVertex2f (20.0, 6.0);
      glVertex2f (20.0, 18.0);
      glVertex2f (8.0, 18.0);
   glEnd();
   glColor3f (1.0, 1.0, 1.0);
```

```
glBegin(GL_LINES);
   glVertex2f (2.0, 2.0);
   glVertex2f (8.0, 6.0);
     glVertex2f (14.0, 2.0);
   glVertex2f (20.0, 6.0);
     glVertex2f (2.0, 14.0);
   glVertex2f (8.0, 18.0);
     glVertex2f (14.0, 14.0);
   glVertex2f (20.0, 18.0);
glEnd();
glColor3f (1.0, 1.0, 1.0);
glBegin(GL_LINES);
   glVertex2f (8.0, 6.0);
   glVertex2f (14.0, 14.0);
glEnd();
glPointSize (20.0);
glBegin(GL_POINTS);
     glColor3f (1.0, 0.0, 0.0);
   glVertex2f (2.0, 2.0);
     glColor3f (1.0, 1.0, 0.0);
   glVertex2f (14.0, 2.0);
     glColor3f (0.0, 1.0, 0.0);
   glVertex2f (20.0, 6.0);
     glColor3f (0.0, 0.0, 0.0);
   glVertex2f (8.0, 6.0);
     glColor3f (1.0, 1.0, 1.0);
   glVertex2f (14.0, 14.0);
     glColor3f (1.0, 0.0, 1.0);
   glVertex2f (2.0, 14.0);
     glColor3f (0.0, 1.0, 1.0);
   glVertex2f (20.0, 18.0);
     glColor3f (0.0, 0.0, 1.0);
   glVertex2f (8.0, 18.0);
     glColor3f (0.5, 1.0, 0.0);
   glVertex2f (17.0, 4.0);
     glColor3f (1.0, 0.5, 0.0);
   glVertex2f (8.0, 2.0);
     glColor3f (0.0, 0.5, 0.0);
   glVertex2f (13.4, 6.0);
     glColor3f (0.5, 0.0, 0.0);
   glVertex2f (5.0, 4.0);
     glColor3f (0.5, 1.0, 1.0);
   glVertex2f (17.0, 16.0);
     glColor3f (0.0, 0.5, 1.0);
   glVertex2f (13.4, 18.0);
     glColor3f (0.5, 0.0, 1.0);
   glVertex2f (5.0, 16.0);
     glColor3f (1.0, 0.5, 1.0);
   glVertex2f (8.6, 14.0);
     glColor3f (1.0, 1.0, 0.5);
   glVertex2f (14.0, 8.0);
     glColor3f (0.0, 1.0, 0.5);
   glVertex2f (20.0, 12.0);
     glColor3f (0.0, 0.0, 0.5);
   glVertex2f (8.0, 12.0);
     glColor3f (1.0, 0.0, 0.5);
   glVertex2f (2.0, 8.0);
     glColor3f (0.5, 0.5, 0.5);
```

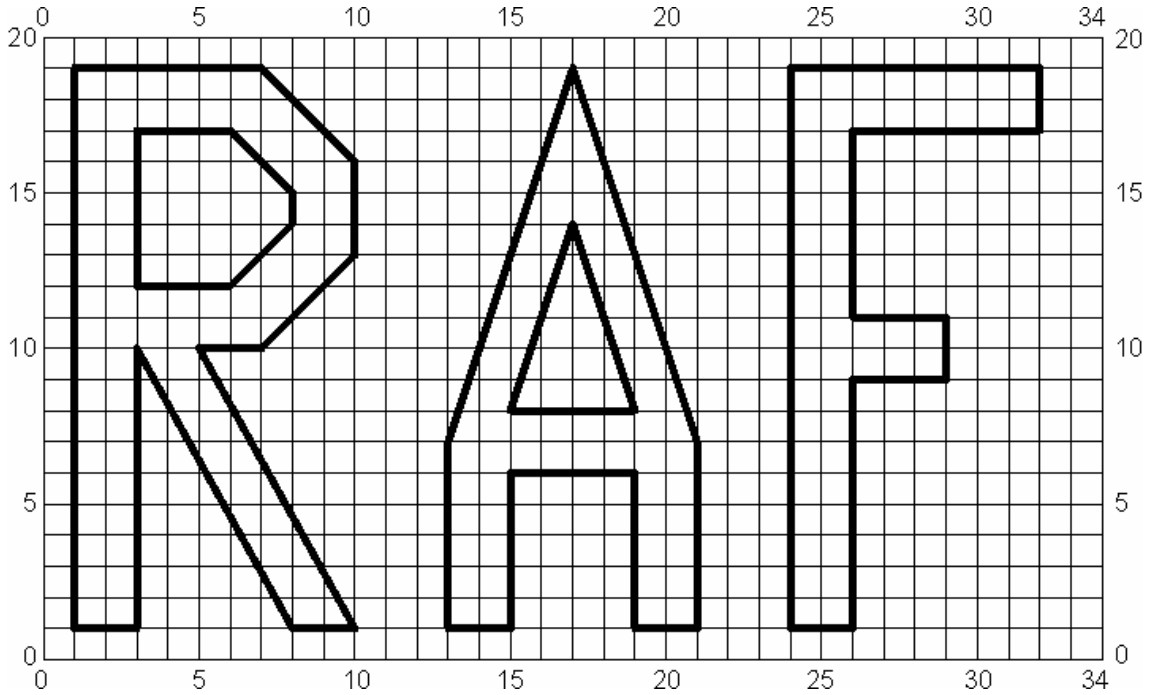
```
      glVertex2f (11.0, 10.0);
   glEnd();
   glFlush ();
}
void init (void)
{
   glClearColor (0.2, 0.2, 0.2, 0.0);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(-1.0, 22.0, -1.0, 22.0, -1.0, 1.0);
}
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (440, 440);
   glutInitWindowPosition (400, 200);
   glutCreateWindow ("RGB model boja u obliku kocke");
   init ();
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
}
```

OpenGL (6)

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazan program koji omogućavaju iscrtavanje dečijeg crteža i natpisa RAF pomoću poligona.

Prvi program omogućava iscrtavanje naziva RAF. Na sledećoj slici prikazana je geometrija gde korisnik može da pronađe željene koordinate.



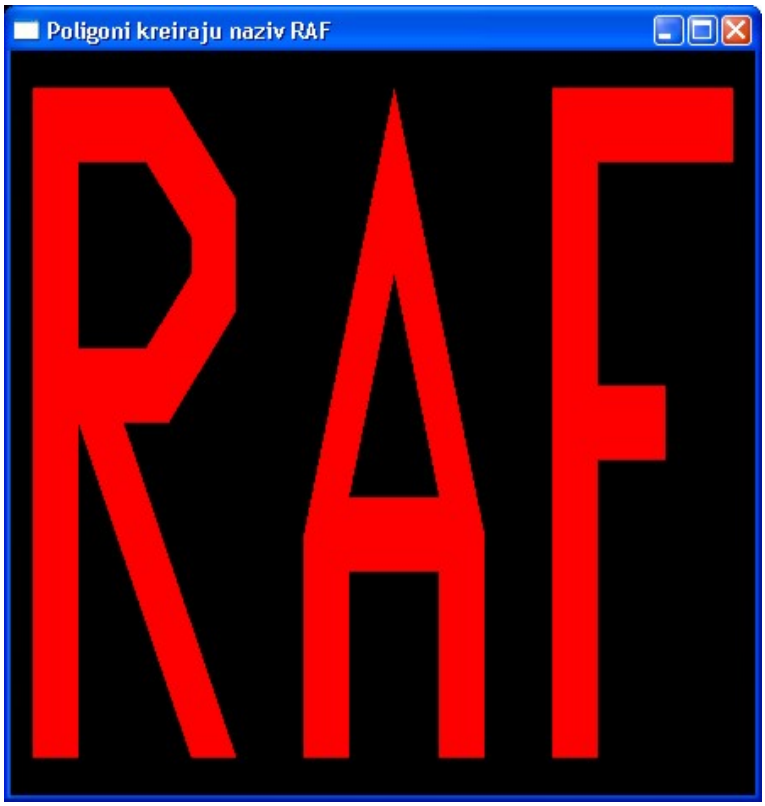
Poligoni za kreiranje naziva RAF

```
/*
 * Crtanje poligona kako bi se kreirao naziv RAF
 *      Dragan Cvetkovic, 16.12.2004.
 */

#include <GL/glut.h>

void display(void)
{

/* Brise sve piksele iz bafera  */
   glClear (GL_COLOR_BUFFER_BIT);

/* Crta se leva ivica slova R */
   glColor3f (1.0, 0.0, 0.0);
   glLineWidth (2.0);
   glRectf (1.0, 1.0, 3.0, 19.0);
   glRectf (3.0, 10.0, 7.0, 19.0);

/* Crta se gornji deo slova R */
   glBegin(GL_POLYGON);
      glVertex2f (7.0, 10.0);
      glVertex2f (10.0, 13.0);
      glVertex2f (10.0, 16.0);
      glVertex2f (7.0, 19.0);
   glEnd();
```

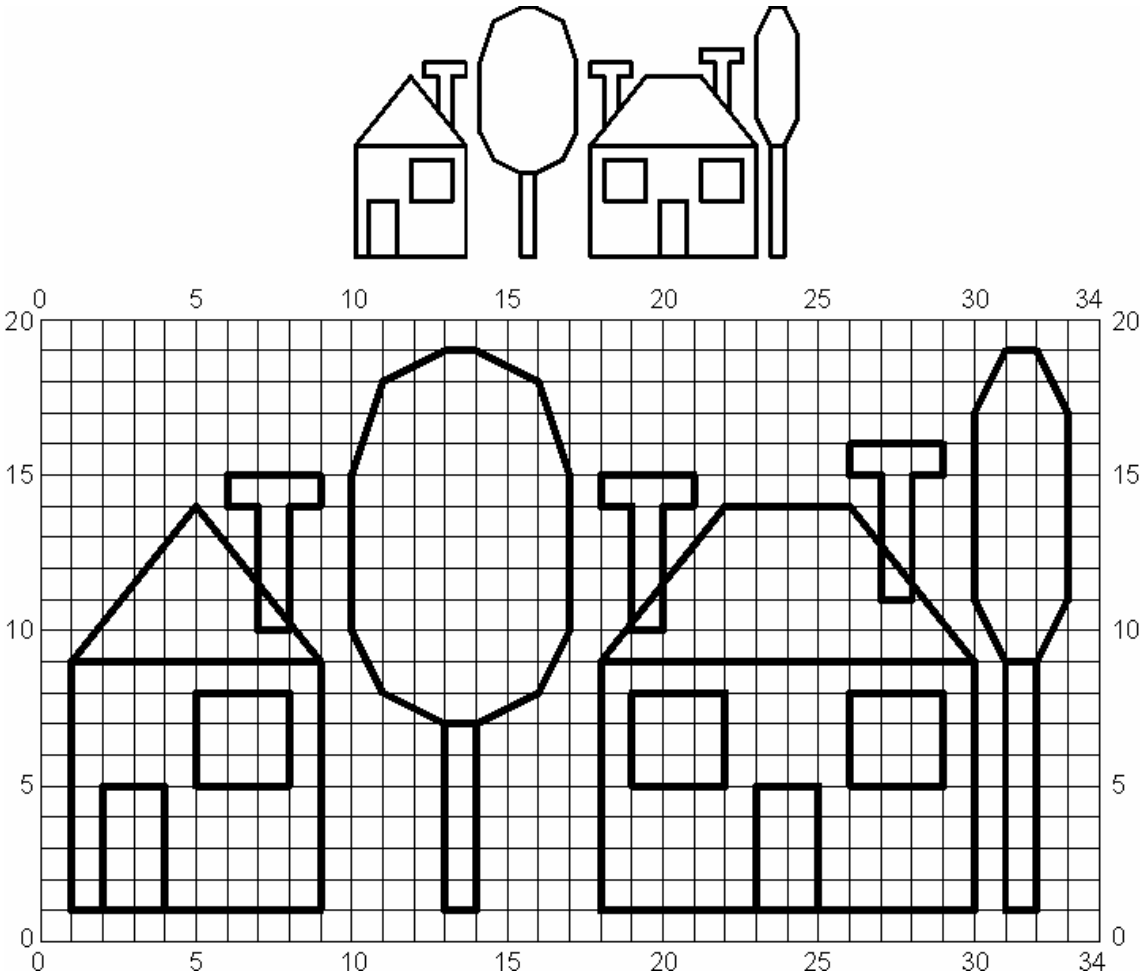
```
/* Crta se pomocna stopica slova R */
   glBegin(GL_POLYGON);
      glVertex2f (8.0, 1.0);
      glVertex2f (10.0, 1.0);
      glVertex2f (5.0, 10.0);
      glVertex2f (3.0, 10.0);
   glEnd();

/* Crta se donji deo slova A */
   glRectf (13.0, 1.0, 21.0, 7.0);

/* Crta se gornji deo slova A */
   glBegin(GL_TRIANGLES);
      glVertex2f (13.0, 7.0);
      glVertex2f (21.0, 7.0);
      glVertex2f (17.0, 19.0);
   glEnd();

/* Crtaju se delovi slova F */
   glRectf (24.0, 1.0, 26.0, 19.0);
   glRectf (26.0, 17.0, 32.0, 19.0);
   glRectf (26.0, 9.0, 29.0, 11.0);

/* Crtaju se supljine u slovima R i A */
   glColor3f (0.0, 0.0, 0.0);
   glBegin(GL_POLYGON);
      glVertex2f (3.0, 12.0);
      glVertex2f (6.0, 12.0);
      glVertex2f (8.0, 14.0);
      glVertex2f (8.0, 15.0);
      glVertex2f (6.0, 17.0);
      glVertex2f (3.0, 17.0);
   glEnd();

   glRectf (15.0, 1.0, 19.0, 6.0);

   glBegin(GL_TRIANGLES);
      glVertex2f (15.0, 8.0);
      glVertex2f (19.0, 8.0);
      glVertex2f (17.0, 14.0);
   glEnd();

/* Izvrsava GL komande u konacnom vremenu */
   glFlush ();
}

void init (void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
   glClearColor (0.0, 0.0, 0.0, 0.0);

/* Definisu se vrednosti pogleda */
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(0.0, 33.0, 0.0, 20.0, -1.0, 1.0);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
```

```
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * "Poligoni kreiraju naziv RAF" u naslovu prozora.
 * Pozivaju se inicijalne rutine. Registruje se Callback
 * funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (440, 440);
   glutInitWindowPosition (400, 200);
   glutCreateWindow ("Poligoni kreiraju naziv RAF");
   init ();
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
/* ANSI C zahteva odgovor kroz pritisak bilo kog tastera. */
}
```

Kada se ovaj program startuje, posle kompajliranja i linkovanja, dobija se kao izlaz prozor koji je prikazan na donjoj slici.



Drugi program omogućava iscrtavanje dečijeg crteža koji se sastoji od kućica i drveća. Na sledećoj slici prikazana je geometrija gde korisnik može da pronađe željene koordinate.



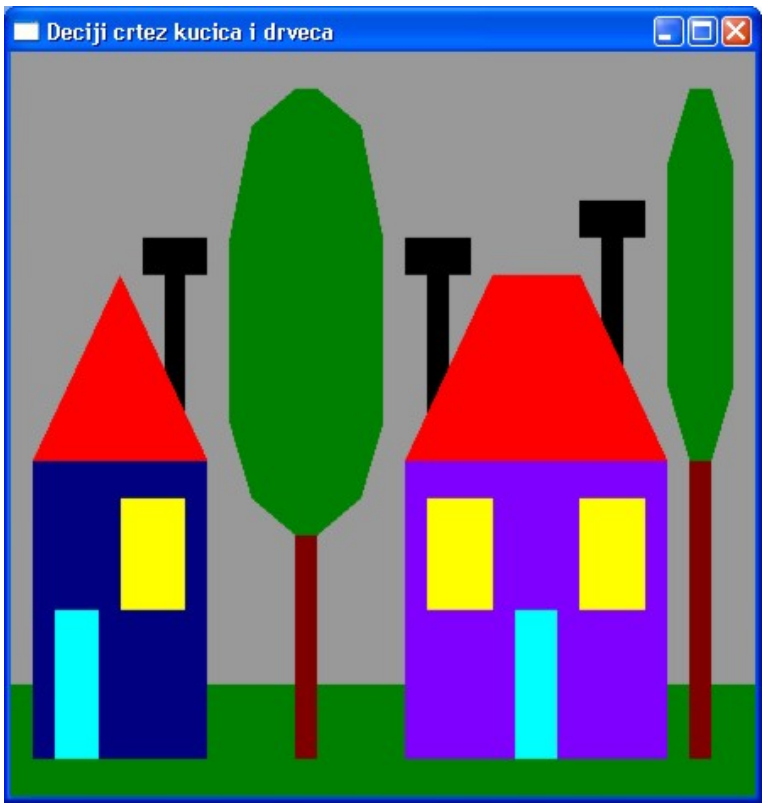
Dečiji crtež kućica i drveća

```
/*
 * Crtanje decijeg crteza kucica i drveca
 *    Dragan Cvetkovic, 16.12.2004.
 */

#include <GL/glut.h>

void display(void)
{

/* brise sve piksele iz bafera  */
   glClear (GL_COLOR_BUFFER_BIT);

/* Crta se tamno zelena trava */
   glColor3f (0.0, 0.5, 0.0);
   glLineWidth (1.0);
   glRectf (0.0, 0.0, 34.0, 3.0);

/* Crta se krosnja veceg drveta */
   glBegin(GL_POLYGON);
      glVertex2f (13.0, 7.0);
      glVertex2f (14.0, 7.0);
      glVertex2f (16.0, 8.0);
      glVertex2f (17.0, 10.0);
      glVertex2f (17.0, 15.0);
      glVertex2f (16.0, 18.0);
      glVertex2f (14.0, 19.0);
      glVertex2f (13.0, 19.0);
```

```
      glVertex2f (11.0, 18.0);
      glVertex2f (10.0, 15.0);
      glVertex2f (10.0, 10.0);
      glVertex2f (11.0, 8.0);
   glEnd();

/* Crta se krosnja manjeg drveta */
   glBegin(GL_POLYGON);
      glVertex2f (31.0, 9.0);
      glVertex2f (32.0, 9.0);
      glVertex2f (33.0, 11.0);
      glVertex2f (33.0, 17.0);
      glVertex2f (32.0, 19.0);
      glVertex2f (31.0, 19.0);
      glVertex2f (30.0, 17.0);
      glVertex2f (30.0, 11.0);
   glEnd();

/* Crtaju se stabla drveca braonkastom bojom,
 * koja predstavlja nijansu izmedju crne i crvene boje
 */
   glColor3f (0.5, 0.0, 0.0);
   glRectf (13.0, 1.0, 14.0, 7.0);
   glRectf (31.0, 1.0, 32.0, 9.0);

/* Crtaju se dimnjaci crnom bojom,
 * sa leve na desnu stranu, redom
 */
   glColor3f (0.0, 0.0, 0.0);
   glRectf (7.0, 10.0, 8.0, 14.0);
   glRectf (6.0, 14.0, 9.0, 15.0);

   glRectf (19.0, 10.0, 20.0, 14.0);
   glRectf (18.0, 14.0, 21.0, 15.0);

   glRectf (27.0, 11.0, 28.0, 15.0);
   glRectf (26.0, 15.0, 29.0, 16.0);

/* Crta se leva kuca tamno plavom bojom,
 * koja predstavlja nijansu izmedju crne i plave boje
 */
   glColor3f (0.0, 0.0, 0.5);
   glRectf (1.0, 1.0, 9.0, 9.0);

/* Crtaju se svetlo plava vrata na levoj kuci */
   glColor3f (0.0, 1.0, 1.0);
   glRectf (2.0, 1.0, 4.0, 5.0);

/* Crta se zuti prozor na levoj kuci */
   glColor3f (1.0, 1.0, 0.0);
   glRectf (5.0, 5.0, 8.0, 8.0);

/* Crta se crveni krov na levoj kuci */
   glColor3f (1.0, 0.0, 0.0);
   glBegin(GL_TRIANGLES);
      glVertex2f (1.0, 9.0);
      glVertex2f (9.0, 9.0);
      glVertex2f (5.0, 14.0);
   glEnd();

/* Crta se desna kuca tamno ljubicastom bojom,
 * koja predstavlja nijansu izmedju crne i ljubicaste boje
 */
   glColor3f (0.5, 0.0, 1.0);
```

```
   glRectf (18.0, 1.0, 30.0, 9.0);

/* Crtaju se svetlo plava vrata na desnoj kuci */
   glColor3f (0.0, 1.0, 1.0);
   glRectf (23.0, 1.0, 25.0, 5.0);

/* Crtaju se zuti prozori na desnoj kuci */
   glColor3f (1.0, 1.0, 0.0);
   glRectf (19.0, 5.0, 22.0, 8.0);
   glRectf (26.0, 5.0, 29.0, 8.0);

/* Crta se crveni krov na levoj kuci */
   glColor3f (1.0, 0.0, 0.0);
   glBegin(GL_POLYGON);
      glVertex2f (18.0, 9.0);
      glVertex2f (30.0, 9.0);
      glVertex2f (26.0, 14.0);
      glVertex2f (22.0, 14.0);
   glEnd();

/* Izvrsava GL komande u konacnom vremenu */
   glFlush ();
}

void init (void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
   glClearColor (0.6, 0.6, 0.6, 0.0);

/* Definisu se vrednosti pogleda */
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(0.0, 34.0, 0.0, 20.0, -1.0, 1.0);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * "Deciji crtez kucica i drveca" u naslovu prozora.
 * Pozivaju se inicijalne rutine. Registruje se Callback
 * funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (440, 440);
   glutInitWindowPosition (400, 200);
   glutCreateWindow ("Deciji crtez kucica i drveca");
   init ();
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
/* ANSI C zahteva odgovor kroz pritisak bilo kog tastera. */
}
```

Kada se ovaj program startuje, posle kompajliranja i linkovanja, dobija se kao izlaz prozor koji je prikazan na donjoj slici.

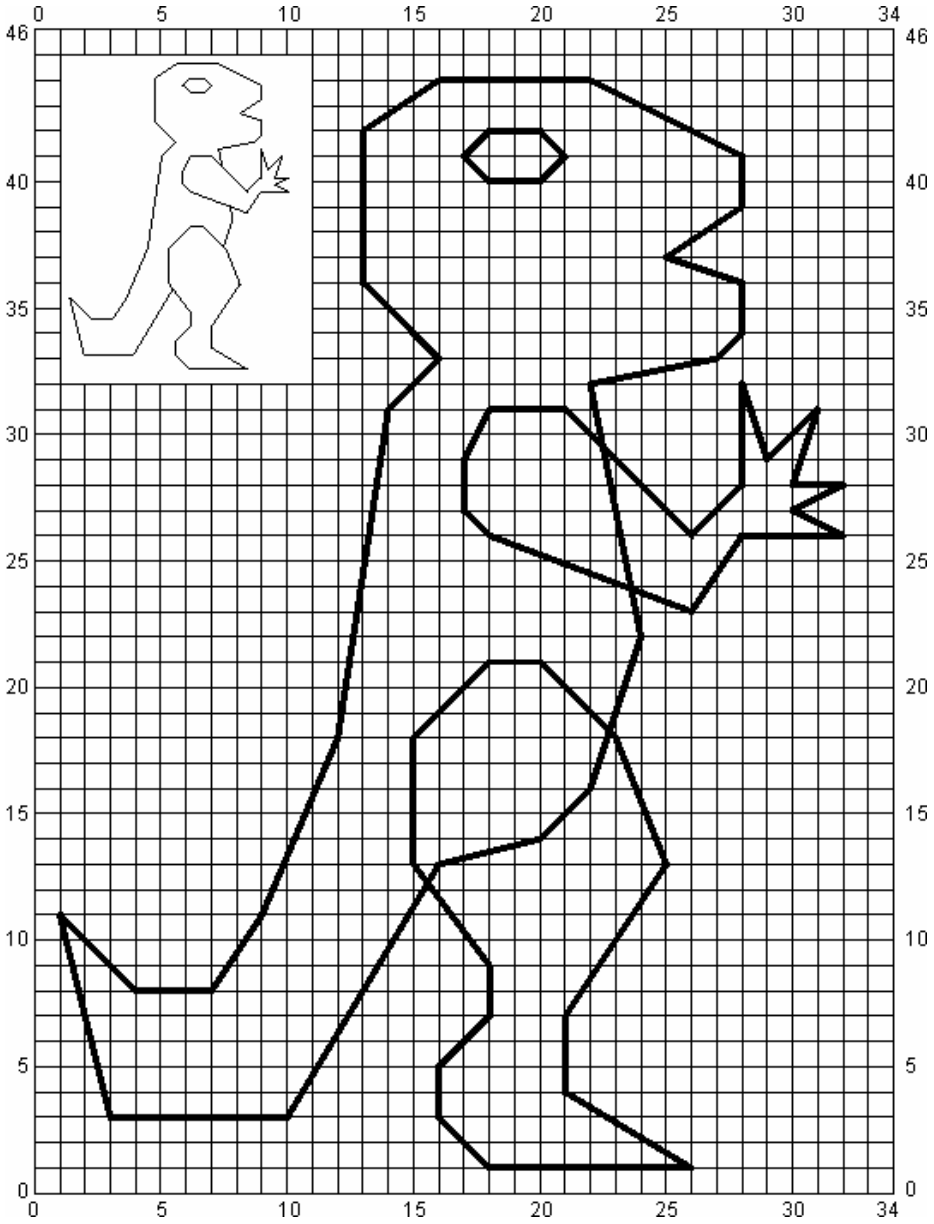

Deciji crtez kucica i drveca

OpenGL (7)

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazan program koji omogućavaju iscrtavanje dečijeg crteža dinosaurusa T-Rex.

Na sledećoj slici prikazana je geometrija gde korisnik može da pronađe željene koordinate.



Poligoni za iscrtavanje dinosaurusa T-Rex

```
/*
 * "Ispunjen" poligon u obliku dinosaurusa T-Rex
 *          Dragan Cvetkovic, 21.12.2004.
 */

#include <GL/glut.h>

void display(void)

{
```

```
/* Brise sve piksele iz bafera  */
   glClear (GL_COLOR_BUFFER_BIT);

/* Crta se vrh repa */
   glColor3f (0.0, 0.4, 0.0);
   glBegin(GL_POLYGON);
      glVertex2f (1.0, 11.0);
      glVertex2f (3.0, 3.0);
      glVertex2f (4.0, 8.0);
   glEnd();

/* Crta se 1. nastavak repa */
   glColor3f (0.0, 0.4, 0.0);
   glBegin(GL_POLYGON);
      glVertex2f (3.0, 3.0);
      glVertex2f (10.0, 3.0);
      glVertex2f (7.0, 8.0);
      glVertex2f (4.0, 8.0);
   glEnd();

/* Crta se 2. nastavak repa */
   glColor3f (0.0, 0.4, 0.0);
   glBegin(GL_POLYGON);
      glVertex2f (7.0, 8.0);
      glVertex2f (10.0, 3.0);
      glVertex2f (16.0, 13.0);
        glVertex2f (12.0, 18.0);
      glVertex2f (9.0, 11.0);
   glEnd();

/* Crta se trup */
   glColor3f (0.0, 0.4, 0.0);
   glBegin(GL_POLYGON);
      glVertex2f (16.0, 13.0);
      glVertex2f (20.0, 14.0);
      glVertex2f (22.0, 16.0);
        glVertex2f (24.0, 22.0);
        glVertex2f (22.0, 32.0);
      glVertex2f (16.0, 33.0);
        glVertex2f (14.0, 31.0);
        glVertex2f (12.0, 18.0);
   glEnd();

/* Crta se donja polovina glave */
   glColor3f (0.0, 0.4, 0.0);
   glBegin(GL_POLYGON);
      glVertex2f (16.0, 33.0);
      glVertex2f (22.0, 32.0);
      glVertex2f (27.0, 33.0);
        glVertex2f (28.0, 34.0);
      glVertex2f (28.0, 36.0);
        glVertex2f (25.0, 37.0);
        glVertex2f (13.0, 42.0);
        glVertex2f (13.0, 36.0);
   glEnd();

/* Crta se gornja polovina glave */
   glColor3f (0.0, 0.4, 0.0);
   glBegin(GL_POLYGON);
      glVertex2f (13.0, 42.0);
      glVertex2f (25.0, 37.0);
      glVertex2f (28.0, 39.0);
        glVertex2f (28.0, 41.0);
      glVertex2f (22.0, 44.0);
```

```
        glVertex2f (16.0, 44.0);
    glEnd();

/* Crta se donji deo noge */
    glColor3f (0.0, 0.5, 0.0);
    glBegin(GL_POLYGON);
       glVertex2f (18.0, 7.0);
       glVertex2f (16.0, 5.0);
       glVertex2f (16.0, 3.0);
         glVertex2f (18.0, 1.0);
       glVertex2f (26.0, 1.0);
         glVertex2f (21.0, 4.0);
    glEnd();

/* Crta se cevanica noge */
    glColor3f (0.0, 0.5, 0.0);
    glBegin(GL_POLYGON);
       glVertex2f (18.0, 7.0);
       glVertex2f (21.0, 4.0);
       glVertex2f (21.0, 7.0);
         glVertex2f (18.0, 9.0);
    glEnd();

/* Crta se gornji deo noge */
    glColor3f (0.0, 0.5, 0.0);
    glBegin(GL_POLYGON);
       glVertex2f (21.0, 7.0);
       glVertex2f (25.0, 13.0);
       glVertex2f (23.0, 18.0);
         glVertex2f (20.0, 21.0);
       glVertex2f (18.0, 21.0);
         glVertex2f (15.0, 18.0);
         glVertex2f (15.0, 13.0);
         glVertex2f (18.0, 9.0);
    glEnd();

/* Crta se rameni deo ruke */
    glColor3f (0.0, 0.5, 0.0);
    glBegin(GL_POLYGON);
       glVertex2f (26.0, 23.0);
       glVertex2f (26.0, 26.0);
       glVertex2f (21.0, 31.0);
         glVertex2f (18.0, 31.0);
       glVertex2f (17.0, 29.0);
         glVertex2f (17.0, 27.0);
         glVertex2f (18.0, 26.0);
    glEnd();

/* Crta se podlaktica */
    glColor3f (0.0, 0.5, 0.0);
    glBegin(GL_POLYGON);
       glVertex2f (26.0, 23.0);
       glVertex2f (28.0, 26.0);
       glVertex2f (28.0, 28.0);
         glVertex2f (26.0, 26.0);
    glEnd();

/* Crta se osnovica dlana */
    glColor3f (0.0, 0.5, 0.0);
    glBegin(GL_POLYGON);
       glVertex2f (28.0, 26.0);
       glVertex2f (30.0, 26.0);
       glVertex2f (30.0, 29.0);
         glVertex2f (28.0, 29.0);
```

```
   glEnd();

/* Crtaju se prsti */
   glColor3f (0.0, 0.5, 0.0);
   glBegin(GL_TRIANGLES);
      glVertex2f (30.0, 26.0);
      glVertex2f (32.0, 26.0);
      glVertex2f (30.0, 27.0);

      glVertex2f (30.0, 27.0);
      glVertex2f (32.0, 28.0);
      glVertex2f (30.0, 28.0);

        glVertex2f (30.0, 28.0);
      glVertex2f (31.0, 31.0);
      glVertex2f (29.0, 29.0);

        glVertex2f (29.0, 29.0);
      glVertex2f (28.0, 32.0);
      glVertex2f (28.0, 28.0);
   glEnd();

/* Crta se oko crvene boje */
   glColor3f (1.0, 0.0, 0.0);
   glBegin(GL_POLYGON);
      glVertex2f (17.0, 41.0);
      glVertex2f (18.0, 40.0);
      glVertex2f (20.0, 40.0);
        glVertex2f (21.0, 41.0);
        glVertex2f (20.0, 42.0);
        glVertex2f (18.0, 42.0);
   glEnd();

/* Izvrsava GL komande u konacnom vremenu */
   glFlush ();
}

void init (void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 * Ovde je izabrana svetlija siva boja!
 */
   glClearColor (0.6, 0.6, 0.6, 0.0);

/* Definisu se vrednosti pogleda */
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(0.0, 33.0, 0.0, 45.0, -1.0, 1.0);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * "Poligon u obliku T-Rexa" u naslovu prozora.
 * Pozivaju se inicijalne rutine. Registruje se Callback
 * funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
```

```
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (400, 400);
   glutInitWindowPosition (400, 200);
   glutCreateWindow ("Poligon u obliku dinosaurusa - T-Rex");
   init ();
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
/* ANSI C zahteva odgovor kroz pritisak bilo kog tastera. */
}
```

Kada se ovaj program startuje, posle kompajliranja i linkovanja, dobija se kao izlaz prozor koji je prikazan na donjoj slici.

OpenGL (8)

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazan program koji omogućava crtanje linija, s tim što će se obratiti pažnja na tri različite komande koje su »vezane« za ovu problematiku. Pored ovoga, prikazaće se koordinate osnovnih boja u RGB modelu boja.

Komanda koja definiše boje koje će se primenjivati na sintaksu ispod je:

glColor3f (R, G, B)

gde se vrednosti komponenti nalaze u granicama od 0.0 do 1.0: **R od 0.0 do 1.0**, **G od 0.0 do 1.0** i **B od 0.0 do 1.0**. Ispod sledi spisak osnovnih boja koje se koriste u OpenGL-u:

glColor3f (0.0, 0.0, 0.0) – CRNA boja (black)
glColor3f (1.0, 0.0, 0.0) – CRVENA boja (red)
glColor3f (0.0, 1.0, 0.0) – ZELENA boja (green)
glColor3f (1.0, 1.0, 0.0) – ŽUTA boja (yellow)
glColor3f (0.0, 0.0, 1.0) – PLAVA boja (blue)
glColor3f (1.0, 0.0, 1.0) – LJUBIČASTA boja (magenta)
glColor3f (0.0, 1.0, 1.0) – SVETLO PLAVA boja (cyan)
glColor3f (1.0, 1.0, 1.0) – BELA boja (white)

U OpenGL-u termin LINE znači da je reč o LINIJSKOM SEGMENTU, što znači da nije reč o pravoj, koja u matematici predstavlja liniju kojoj se ne zna ni početna ni krajnja tačka. Ovo je najlakši način da se definiše povezana serija pravolinijskih segmenata, čak i zatvorenih. U svim slučajevima, pravolinijski segment je definisan koordinatama krajnjih tačaka. Po pitanju ove problematike u igri su tri komande:

- **GL_LINES** – par temena koji interpretiraju individualni pravolinijski segment
- **GL_LINE_STRIP** – serija povezanih pravolinijskih segmenata
- **GL_LINE_LOOP** – isto kao prethodna komanda, sa dodatnim segmentom između krajnjeg i početnog temena

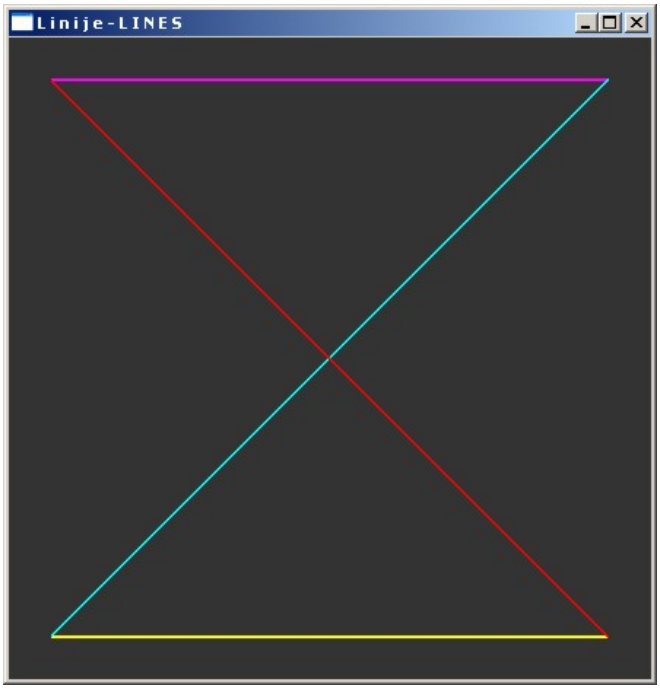
Pored ove tri komande u opticaju je i komanda za podešavanje širine (“debljine”) same linije čija je sintaksa **glLineWidth (2.0)**. Vrednost u zagradi je vrednost u PIKSELIMA, tako da ako je vrednost 1 (1.0), to znači da se iscrtava linija čija je širina (“debljina”) 1 piksel, ako je vrednost 2, onda se iscrtava linija čija je širina (“debljina”) 2 piksela, itd.

Slede primeri primene ovih komandi, s tim što treba napomenuti da su koordinate tačaka iste, ali će odziv biti različiti u zavisnosti od primenjene komande.

```
/*
 * Ovde su uradjene linije u raznim bojama komandom GL_LINES
 *              14.02.2004.
 */

#include <GL/glut.h>
void display(void)
{

/* brise sve piksele iz bafera  */
   glClear (GL_COLOR_BUFFER_BIT);

/* Crtaju se linije razlicite boje */
   glLineWidth (2.0);
   glBegin(GL_LINES);
```

```
  glColor3f (1.0, 1.0, 0.0);
    glVertex2f (1.0, 1.0);
    glVertex2f (14.0, 1.0);
  glColor3f (1.0, 0.0, 1.0);
    glVertex2f (14.0, 14.0);
    glVertex2f (1.0, 14.0);
  glColor3f (0.0, 1.0, 1.0);
    glVertex2f (1.0, 1.0);
    glVertex2f (14.0, 14.0);
  glColor3f (1.0, 0.0, 0.0);
    glVertex2f (14.0, 1.0);
    glVertex2f (1.0, 14.0);
  glEnd();

/* Izvrsava GL komande u konacnom vremenu */
  glFlush ();
}

void init (void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
  glClearColor (0.2, 0.2, 0.2, 0.0);

/* Definisu se vrednosti pogleda */
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(0.0, 15.0, 0.0, 15.0, -1.0, 1.0);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * "L i n i j e - L I N E S" u naslovu prozora.
 * Pozivaju se inicijalne rutine. Registruje se Callback
 * funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize (440, 440);
  glutInitWindowPosition (400, 200);
  glutCreateWindow ("L i n i j e - L I N E S");
  init ();
  glutDisplayFunc(display);
  glutMainLoop();
  return 0;
/* ANSI C zahteva odgovor kroz pritisak bilo kog tastera. */
}
```

Odziv ove sintakse je prikazan na slici 1.

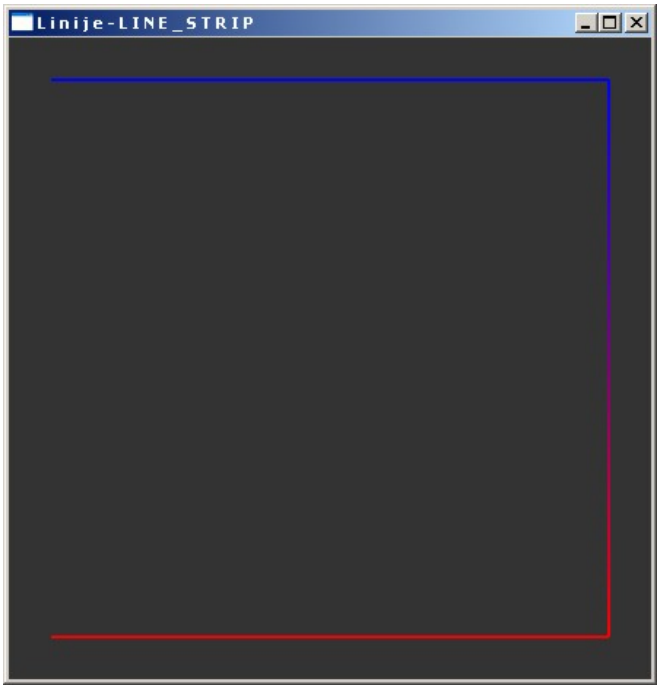


Slika 1

Ako se primeni komanda GL_LINE_STRIP pojaviće se slika 2. Sintaksa programa sledi:

```
/*
 * Crtanje linije komandom GL_LINE_STRIP
 *     Dragan Cvetkovic, 15.12.2004.
 */

#include <GL/glut.h>
void display(void)
{
/* brise sve piksele iz bafera  */
   glClear (GL_COLOR_BUFFER_BIT);

/* Crtaju se tri pravolinijska segmenta, s tim sto je pocetni
 * segment crvene boje, treci segment je plave boje, a srednji
 * (vertikalni) segment je u nijansama izmedju ove dve boje
 */
   glLineWidth (2.0);
   glBegin(GL_LINE_STRIP);
   glColor3f (1.0, 0.0, 0.0);
      glVertex2f (1.0, 1.0);
      glVertex2f (14.0, 1.0);
   glColor3f (0.0, 0.0, 1.0);
      glVertex2f (14.0, 14.0);
      glVertex2f (1.0, 14.0);
   glEnd();

/* Izvrsava GL komande u konacnom vremenu */
   glFlush ();
}

void init (void)
{
```

```
/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
  glClearColor (0.2, 0.2, 0.2, 0.0);

/* Definisu se vrednosti pogleda */
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(0.0, 15.0, 0.0, 15.0, -1.0, 1.0);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * "L i n i j e - L I N E _ S T R I P" u naslovu prozora.
 * Pozivaju se inicijalne rutine. Registruje se Callback
 * funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize (440, 440);
  glutInitWindowPosition (400, 200);
  glutCreateWindow ("L i n i j e - L I N E _ S T R I P");
  init ();
  glutDisplayFunc(display);
  glutMainLoop();
  return 0;
/* ANSI C zahteva odgovor kroz pritisak bilo kog tastera. */
}
```



Slika 2

Ako se primeni komanda GL_LINE_LOOP pojaviće se slika 3. Sintaksa programa sledi:

```
/*
 * Model kvadrata komandom GL_LINE_LOOP
 *    Dragan Cvetkovic, 15.12.2004.
 */

#include <GL/glut.h>
void display(void)
{
/* brise sve piksele iz bafera  */
  glClear (GL_COLOR_BUFFER_BIT);

/* Crta se kvadrat, s tim sto je pocetna
 * stranica (donja) crvene boje, a treca
 * stranica (gornja horizontalna) plave
 * boje, a vertikalne stranice imaju boje
 * koje predstavljaju prelaze izmedju ove dve
 */
  glLineWidth (2.0);
  glBegin(GL_LINE_LOOP);
  glColor3f (1.0, 0.0, 0.0);
    glVertex2f (1.0, 1.0);
    glVertex2f (14.0, 1.0);
  glColor3f (0.0, 0.0, 1.0);
    glVertex2f (14.0, 14.0);
    glVertex2f (1.0, 14.0);
  glEnd();

/* Izvrsava GL komande u konacnom vremenu */
  glFlush ();
}

void init (void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
  glClearColor (0.2, 0.2, 0.2, 0.0);

/* Definisu se vrednosti pogleda */
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(0.0, 15.0, 0.0, 15.0, -1.0, 1.0);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * "L i n i j e - L I N E _ L O O P" u naslovu prozora.
 * Pozivaju se inicijalne rutine. Registruje se Callback
 * funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
```

```
int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize (440, 440);
  glutInitWindowPosition (400, 200);
  glutCreateWindow ("L i n i j e - L I N E _ L O O P");
  init ();
  glutDisplayFunc(display);
  glutMainLoop();
  return 0;
/* ANSI C zahteva odgovor kroz pritisak bilo kog tastera. */
}
```

Slika 3

Sledi sintaksa programa gde se vidi primena i promena različitih širina linija. Sledi sintaksa programa, a rezultat je prikazan na slici 4.

```
/*
 * Crtaju se linije razlicite sirine ("debljine") i boje
 *            Dragan Cvetkovic, 14.12.2004.
 */
#include <GL/glut.h>
void display(void)
{
/* brise sve piksele iz bafera  */
  glClear (GL_COLOR_BUFFER_BIT);
/* Crta se linija sirine 1 i crne boje */
  glLineWidth (1.0);
  glBegin(GL_LINES);
  glColor3f (0.0, 0.0, 0.0);
  glVertex2f (1.0, 1.0);
  glVertex2f (14.0, 1.0);
  glEnd();
/* Crta se linija sirine 2 i crvene boje */
```

```
   glLineWidth (2.0);
   glBegin(GL_LINES);
   glColor3f (1.0, 0.0, 0.0);
   glVertex2f (1.0, 3.0);
   glVertex2f (14.0, 3.0);
   glEnd();
/* Crta se linija sirine 3 i plave boje */
   glLineWidth (3.0);
   glBegin(GL_LINES);
   glColor3f (0.0, 1.0, 0.0);
   glVertex2f (1.0, 5.0);
   glVertex2f (14.0, 5.0);
   glEnd();

/* Crta se linija sirine 4 i zute boje */
   glLineWidth (4.0);
   glBegin(GL_LINES);
   glColor3f (1.0, 1.0, 0.0);
   glVertex2f (1.0, 7.0);
   glVertex2f (14.0, 7.0);
   glEnd();
/* Crta se linija sirine 5 i plave boje */
   glLineWidth (5.0);
   glBegin(GL_LINES);
   glColor3f (0.0, 0.0, 1.0);
   glVertex2f (1.0, 9.0);
   glVertex2f (14.0, 9.0);
   glEnd();
/* Crta se linija sirine 6 i ljubicaste boje */
   glLineWidth (6.0);
   glBegin(GL_LINES);
   glColor3f (1.0, 0.0, 1.0);
   glVertex2f (1.0, 11.0);
   glVertex2f (14.0, 11.0);
   glEnd();
/* Crta se linija sirine 7 i svetlo plave boje */
   glLineWidth (7.0);
   glBegin(GL_LINES);
   glColor3f (0.0, 1.0, 1.0);
   glVertex2f (1.0, 13.0);
   glVertex2f (14.0, 13.0);
   glEnd();
/* Crta se linija sirine 8 i bele boje */
   glLineWidth (8.0);
   glBegin(GL_LINES);
   glColor3f (1.0, 1.0, 1.0);
   glVertex2f (1.0, 15.0);
   glVertex2f (14.0, 15.0);
   glEnd();
/* Izvrsava GL komande u konacnom vremenu */
   glFlush ();
}
void init (void)
{
/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
```

```
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
   glClearColor (0.2, 0.2, 0.2, 0.0);
/* Definisu se vrednosti pogleda */
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(0.0, 15.0, 0.0, 16.0, -1.0, 1.0);
}
/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * "Razlicite sirine i boje linija" u naslovu prozora.
 * Pozivaju se inicijalne rutine. Registruje se Callback
 * funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (440, 440);
   glutInitWindowPosition (400, 200);
   glutCreateWindow ("Razlicite sirine i boje linija");
   init ();
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
/* ANSI C zahteva odgovor kroz pritisak bilo kog tastera. */
}
```

Slika 4

Ovo su predložene varijante, ali korisnik može parametre da menja proizvoljno, kako bi došao do željenog rezultata.

OpenGL (9)

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazan program koji omogućava crtanje trouglova, s tim što će se obratiti pažnja na tri različite komande koje su »vezane« za ovu problematiku. Pored ovoga, prikazaće se koordinate osnovnih boja u RGB modelu boja.

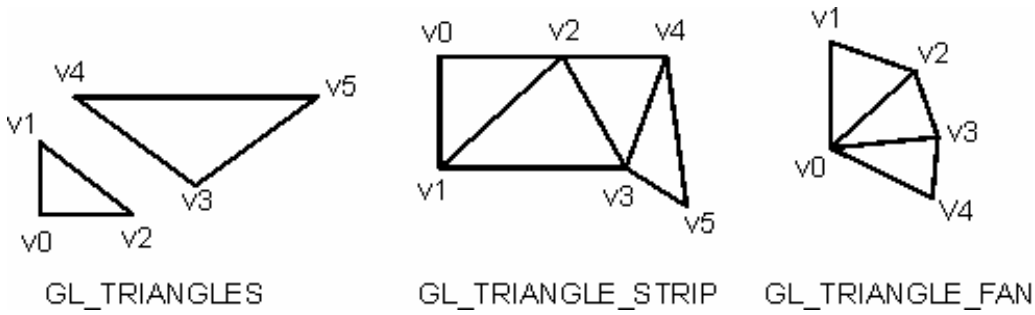
Komanda koja definiše boje koje će se primenjivati na sintaksu ispod je:

glColor3f (R, G, B)

gde se vrednosti komponenti nalaze u granicama od 0.0 do 1.0: **R od 0.0 do 1.0**, **G od 0.0 do 1.0** i **B od 0.0 do 1.0**. Ispod sledi spisak osnovnih boja koje se koriste u OpenGL-u:

glColor3f (0.0, 0.0, 0.0) – CRNA boja (black)
glColor3f (1.0, 0.0, 0.0) – CRVENA boja (red)
glColor3f (0.0, 1.0, 0.0) – ZELENA boja (green)
glColor3f (1.0, 1.0, 0.0) – ŽUTA boja (yellow)
glColor3f (0.0, 0.0, 1.0) – PLAVA boja (blue)
glColor3f (1.0, 0.0, 1.0) – LJUBIČASTA boja (magenta)
glColor3f (0.0, 1.0, 1.0) – SVETLO PLAVA boja (cyan)
glColor3f (1.0, 1.0, 1.0) – BELA boja (white)

U OpenGL-u termin TRIANGLE znači da je reč o trougaonim površima ili površinama ili trouglovima, s tim što treba napomenuti da su ti trouglovi proizvoljnog oblika i definišu se pomoću koordinata svojih temena. U svakom slučaju, trougao je definisan koordinatama svojih temena, tj. krajnjih tačaka svojih stranica.



Slika 1

Po pitanju ove problematike u igri su tri komande:

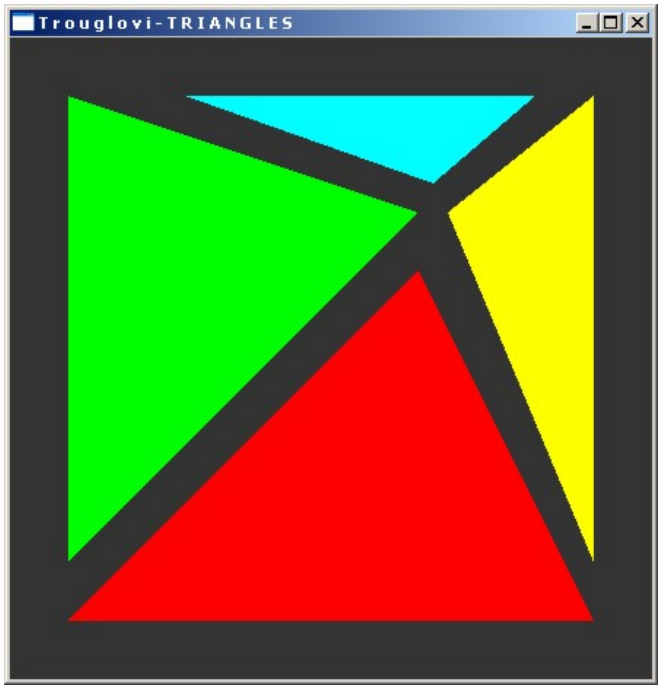
- **GL_TRIANGLES** – crta seriju trouglova (trostranih poligona) koristeći temena v0, v1, v2, onda v3, v4, v5, itd. Ako se ne definišu sva tri temena, onda se prvo ili prvo i drugo teme zanemaruju, kao i da nisu definisana (slika 1).
- **GL_TRIANGLE_STRIP –** crta seriju trouglova (trostranih poligona) koristeći temena v0, v1, v2, onda v2, v1, v3 (OBRATITI PAŽNJU NA REDOSLED), onda v2, v3, v4, itd. Redosled osigurava crtanje trouglova iste orijentacije, tako da mogu da kreiraju korektan deo površi koja se sastoji od trougaonih segmenata. Ispravna orijentacija je veoma važna za neke operacije (slika 1).
- **GL_TRIANGLE_FAN –** radi slično komandi GL_TRIANGLE_STRIP, jedino što se temena definišu u sledećem rasporedu: v0, v1, v2, onda, v0, v2, v3, onda v0, v3, v4, itd (slika 1).

Slede primeri primene ovih komandi. Ako se primeni komanda GL_TRIANGLES pojaviće se slika 3. Sintaksa programa sledi:

```
/*
 * Crtaju se proizvoljni trouglovi komandom TRIANGLES
 *          Dragan Cvetkovic, 15.12.2004.
 */

#include <GL/glut.h>

void display(void)

{
/* brise sve piksele iz bafera  */
   glClear (GL_COLOR_BUFFER_BIT);

/* Crtaju se proizvoljni trouglovi */
   glLineWidth (2.0);
   glBegin(GL_TRIANGLES);
/* Crta se crveni proizvoljan trougao */
           glColor3f (1.0, 0.0, 0.0);
           glVertex2f (2.0, 2.0);
      glVertex2f (20.0, 2.0);
      glVertex2f (14.0, 14.0);
/* Crta se zeleni proizvoljan trougao */
      glColor3f (0.0, 1.0, 0.0);
           glVertex2f (2.0, 4.0);
      glVertex2f (14.0, 16.0);
      glVertex2f (2.0, 20.0);
/* Crta se zuti proizvoljan trougao */
           glColor3f (1.0, 1.0, 0.0);
           glVertex2f (20.0, 20.0);
      glVertex2f (20.0, 4.0);
      glVertex2f (15.0, 16.0);
/* Crta se svetlo plavi proizvoljan trougao */
           glColor3f (0.0, 1.0, 1.0);
           glVertex2f (6.0, 20.0);
      glVertex2f (18.0, 20.0);
      glVertex2f (14.5, 17.0);
   glEnd();

/* Izvrsava GL komande u konacnom vremenu */
   glFlush ();
}

void init (void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
   glClearColor (0.2, 0.2, 0.2, 0.0);

/* Definisu se vrednosti pogleda */
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(0.0, 22.0, 0.0, 22.0, -1.0, 1.0);
```

```
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * "T r o u g l o v i - T R I A N G L E S" u naslovu prozora.
 * Pozivaju se inicijalne rutine. Registruje se Callback
 * funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize (440, 440);
  glutInitWindowPosition (400, 200);
  glutCreateWindow ("T r o u g l o v i - T R I A N G L E S");
  init ();
  glutDisplayFunc(display);
  glutMainLoop();
  return 0;
/* ANSI C zahteva odgovor kroz pritisak bilo kog tastera. */
}
```

Odziv ove sintakse je prikazan na slici 2.



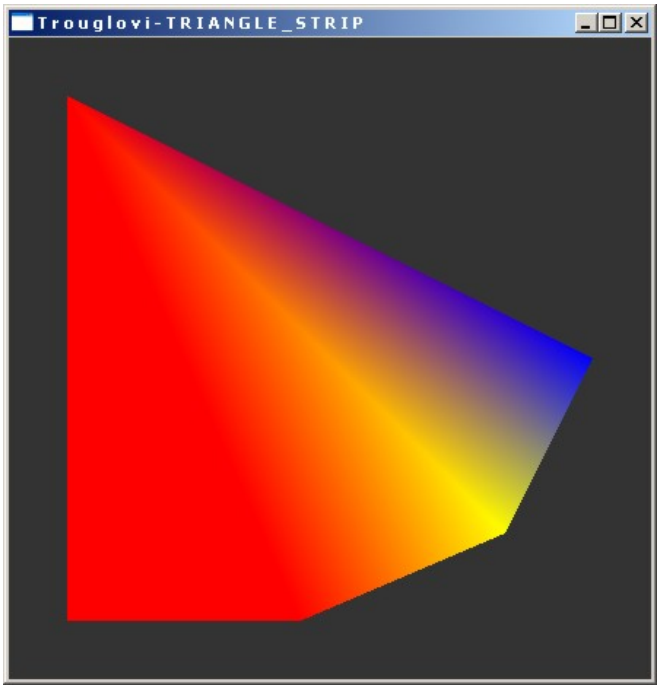
Slika 2

Ako se primeni komanda GL_TRIANGLE_STRIP pojaviće se slika 3. Sintaksa programa sledi:

```
/*
 * Crtaju se proizvoljni trouglovi komandom TRIANGLE_STRIP
 *          Dragan Cvetkovic, 15.12.2004.
 */

#include <GL/glut.h>
```

```
void display(void)

{
/* brise sve piksele iz bafera  */
   glClear (GL_COLOR_BUFFER_BIT);

/* Crtaju se proizvoljni trouglovi */
   glLineWidth (2.0);
   glBegin(GL_TRIANGLE_STRIP);

/* Crta se crveni trougao */
   glColor3f (1.0, 0.0, 0.0);
      glVertex2f (2.0, 2.0);
      glVertex2f (10.0, 2.0);
      glVertex2f (2.0, 20.0);

/* Crta se zuti trougao, kao nastavak crvenog */
   glColor3f (1.0, 1.0, 0.0);
      glVertex2f (17.0, 5.0);

/* Crta se plavi trougao, kao nastavak zutog */
   glColor3f (0.0, 0.0, 1.0);
      glVertex2f (20.0, 11.0);
   glEnd();

/* Izvrsava GL komande u konacnom vremenu */
   glFlush ();
}

void init (void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
   glClearColor (0.2, 0.2, 0.2, 0.0);

/* Definisu se vrednosti pogleda */
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(0.0, 22.0, 0.0, 22.0, -1.0, 1.0);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * "T r o u g l o v i _ T R I A N G L E S _ S T R I P" u naslovu prozora.
 * Pozivaju se inicijalne rutine. Registruje se Callback
 * funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
```

```
   glutInitWindowSize (440, 440);
   glutInitWindowPosition (400, 200);
   glutCreateWindow ("T r o u g l o v i - T R I A N G L E _ S T R I P");
   init ();
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
/* ANSI C zahteva odgovor kroz pritisak bilo kog tastera. */
}
```

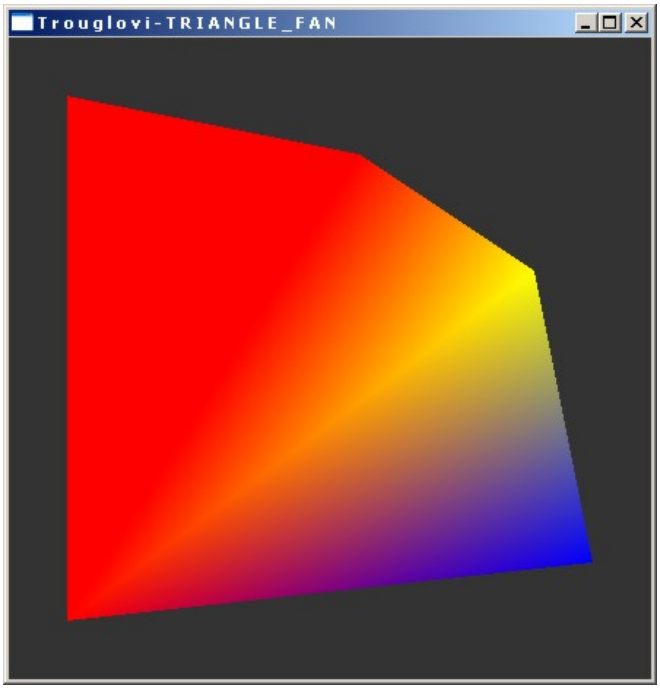


Slika 3

Ako se primeni komanda GL_TRAINGLE_FAN pojaviće se slika 4. Sintaksa programa sledi:

```
/*
 * Crtaju se proizvoljni trouglovi komandom TRIANGLE_FAN
 *              Dragan Cvetkovic, 15.12.2004.
 */

#include <GL/glut.h>

void display(void)

{
/* brise sve piksele iz bafera  */
   glClear (GL_COLOR_BUFFER_BIT);

/* Crtaju se proizvoljni trouglovi */
   glLineWidth (2.0);
   glBegin(GL_TRIANGLE_FAN);

/* Crta se crveni trougao */
   glColor3f (1.0, 0.0, 0.0);
     glVertex2f (2.0, 2.0);
     glVertex2f (2.0, 20.0);
     glVertex2f (12.0, 18.0);
```

```
/* Crta se zuti trougao, kao nastavak crvenog */
   glColor3f (1.0, 1.0, 0.0);
     glVertex2f (18.0, 14.0);

/* Crta se plavi trougao, kao nastavak zutog */
   glColor3f (0.0, 0.0, 1.0);
     glVertex2f (20.0, 4.0);
   glEnd();

/* Izvrsava GL komande u konacnom vremenu */
   glFlush ();
}

void init (void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
   glClearColor (0.2, 0.2, 0.2, 0.0);

/* Definisu se vrednosti pogleda */
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(0.0, 22.0, 0.0, 22.0, -1.0, 1.0);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * "T r o u g l o v i _ T R I A N G L E S _ F A N" u naslovu prozora.
 * Pozivaju se inicijalne rutine. Registruje se Callback
 * funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (440, 440);
   glutInitWindowPosition (400, 200);
   glutCreateWindow ("T r o u g l o v i - T R I A N G L E _ F A N");
   init ();
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
/* ANSI C zahteva odgovor kroz pritisak bilo kog tastera. */
}
```



Slika 4

U tekstu **OpenGL(08)** prikazano je kako se radi sa različitim širinama („debljinama“) linija, tako da korisnik može da pročita i to, ako hoće da primeni različite širine linija. Ovo su predložene varijante, ali korisnik može parametre da menja proizvoljno, kako bi došao do željenog rezultata.

OpenGL (10)

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazan program koji omogućava crtanje trouglova, s tim što će se obratiti pažnja na tri različite komande koje su »vezane« za ovu problematiku. Pored ovoga, prikazaće se koordinate osnovnih boja u RGB modelu boja.

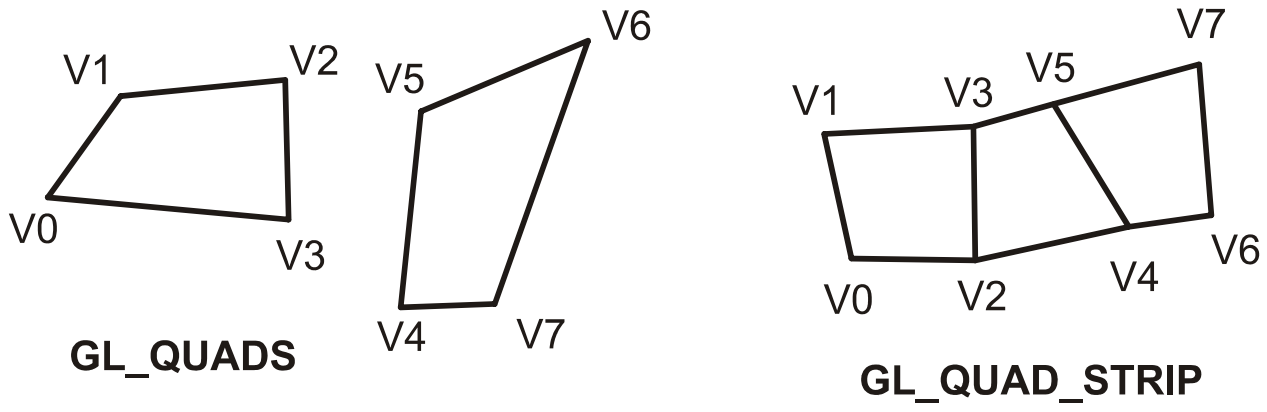
Komanda koja definiše boje koje će se primenjivati na sintaksu ispod je:

glColor3f (R, G, B)

gde se vrednosti komponenti nalaze u granicama od 0.0 do 1.0: **R od 0.0 do 1.0**, **G od 0.0 do 1.0** i **B od 0.0 do 1.0**. Ispod sledi spisak osnovnih boja koje se koriste u OpenGL-u:

glColor3f (0.0, 0.0, 0.0) – CRNA boja (black)
glColor3f (1.0, 0.0, 0.0) – CRVENA boja (red)
glColor3f (0.0, 1.0, 0.0) – ZELENA boja (green)
glColor3f (1.0, 1.0, 0.0) – ŽUTA boja (yellow)
glColor3f (0.0, 0.0, 1.0) – PLAVA boja (blue)
glColor3f (1.0, 0.0, 1.0) – LJUBIČASTA boja (magenta)
glColor3f (0.0, 1.0, 1.0) – SVETLO PLAVA boja (cyan)
glColor3f (1.0, 1.0, 1.0) – BELA boja (white)

U OpenGL-u termin QUAD znači da je reč o četvorougaonim površima ili površinama ili četvorouglovima, s tim što treba napomenuti da su ti četvorouglovi proizvoljnog oblika i definišu se pomoću koordinata svojih temena. U svakom slučaju, četvorougao je definisan koordinatama svojih temena, tj. krajnjih tačaka svojih stranica.



Slika 1

Po pitanju ove problematike u igri su dve komande:

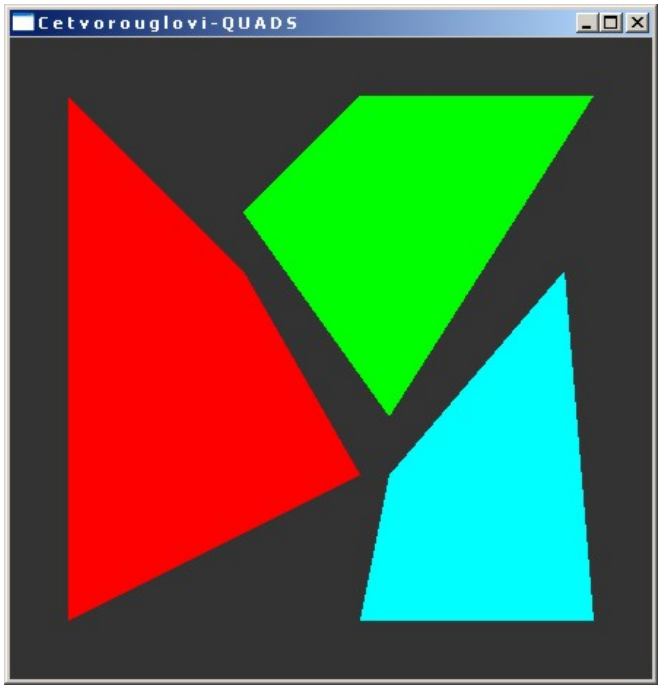
- **GL_QUADS** – crta seriju četvorouglova (četvorougaonih poligona) koristeći temena v0, v1, v2, v3, onda v4, v5, v6, v7, itd. Ako se ne definišu sva 4 temena, onda se prvo ili prvo i drugo ili prvo, drugo i treće teme zanemaruju kao da nisu ni definisana (slika 1, levo).
- **GL_QUAD_STRIP –** crta seriju četvorouglova (četvorougaonih poligona) počinjući temenima v0, v1, v3, v2, onda v2, v3, v5, v4, onda v4, v5, v7, v6, itd. Ako se ne definišu sva 4 temena, onda se prvo ili prvo i drugo ili prvo, drugo i treće teme zanemaruju kao da nisu ni definisana (slika 1, desno).

Slede primeri primene ovih komandi. Ako se primeni komanda GL_QUADS pojaviće se slika 2. Sintaksa programa sledi:

```
/*
 * Crtanje cetvorouglova komandom QUADS
 *     Dragan Cvetkovic, 15.12.2004.
 */

#include <GL/glut.h>

void display(void)

{
/* brise sve piksele iz bafera  */
  glClear (GL_COLOR_BUFFER_BIT);

/* Crtaju se proizvoljni cetvorouglovi */
        glLineWidth (2.0);
        glBegin(GL_QUADS);

/* Crta se crveni cetovorugao */
          glColor3f (1.0, 0.0, 0.0);
          glVertex2f (2.0, 2.0);
          glVertex2f (2.0, 20.0);
          glVertex2f (8.0, 14.0);
          glVertex2f (12.0, 7.0);

/* Crta se svetlo plavi cetovorugao */
    glColor3f (0.0, 1.0, 1.0);
    glVertex2f (12.0, 2.0);
    glVertex2f (20.0, 2.0);
    glVertex2f (19.0, 14.0);
    glVertex2f (13.0, 7.0);

/* Crta se zeleni cetovorugao */
    glColor3f (0.0, 1.0, 0.0);
    glVertex2f (13.0, 9.0);
    glVertex2f (20.0, 20.0);
    glVertex2f (12.0, 20.0);
    glVertex2f (8.0, 16.0);
  glEnd();

/* Izvrsava GL komande u konacnom vremenu */
  glFlush ();
}

void init (void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
  glClearColor (0.2, 0.2, 0.2, 0.0);

/* Definisu se vrednosti pogleda */
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(0.0, 22.0, 0.0, 22.0, -1.0, 1.0);
```

```
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * "C e t v o r o u g l o v i - Q U A D S" u naslovu prozora.
 * Pozivaju se inicijalne rutine. Registruje se Callback
 * funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize (440, 440);
  glutInitWindowPosition (400, 200);
  glutCreateWindow ("C e t v o r o u g l o v i - Q U A D S");
  init ();
  glutDisplayFunc(display);
  glutMainLoop();
  return 0;
/* ANSI C zahteva odgovor kroz pritisak bilo kog tastera. */
}
```

Odziv ove sintakse je prikazan na slici 2.



Slika 2

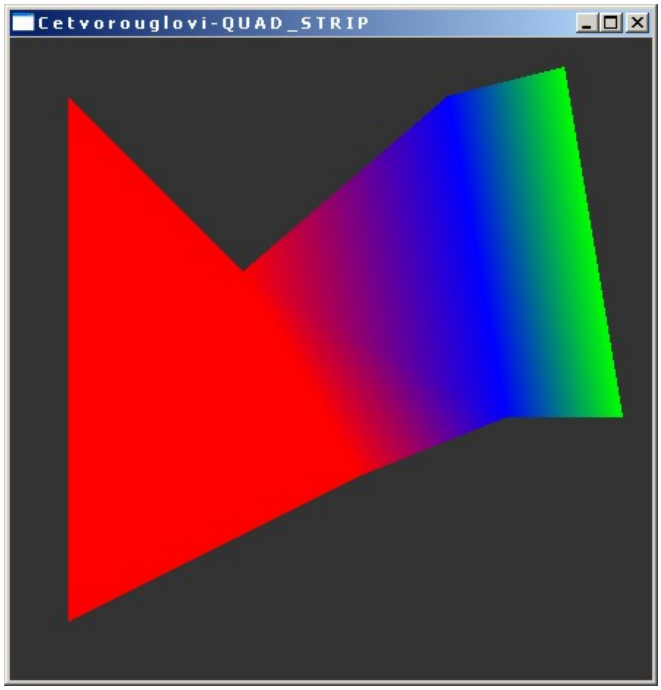
Ako se primeni komanda GL_QUAD_STRIP pojaviće se slika 3. Sintaksa programa sledi:

```
/*
 * Crtanje cetvorouglova komandom QUAD_STRIP
 *     Dragan Cvetkovic, 15.12.2004.
 */

#include <GL/glut.h>
```

```
void display(void)

{
/* brise sve piksele iz bafera  */
   glClear (GL_COLOR_BUFFER_BIT);

/* Crtaju se proizvoljni cetvorouglovi */
    glLineWidth (2.0);
    glBegin(GL_QUAD_STRIP);

/* Crta se crveni cetovorugao */
      glColor3f (1.0, 0.0, 0.0);
      glVertex2f (2.0, 2.0);
      glVertex2f (2.0, 20.0);
      glVertex2f (12.0, 7.0);
      glVertex2f (8.0, 14.0);

/* Crta se svetlo plavi cetovorugao */
      glColor3f (0.0, 0.0, 1.0);
      glVertex2f (17.0, 9.0);
      glVertex2f (15.0, 20.0);

/* Crta se zeleni cetovorugao */
      glColor3f (0.0, 1.0, 0.0);
      glVertex2f (21.0, 9.0);
      glVertex2f (19.0, 21.0);
   glEnd();

/* Izvrsava GL komande u konacnom vremenu */
   glFlush ();
}

void init (void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
   glClearColor (0.2, 0.2, 0.2, 0.0);

/* Definisu se vrednosti pogleda */
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(0.0, 22.0, 0.0, 22.0, -1.0, 1.0);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * "C e t v o r o u g l o v i - Q U A D _ S T R I P" u naslovu prozora.
 * Pozivaju se inicijalne rutine. Registruje se Callback
 * funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)
```

```
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize (440, 440);
  glutInitWindowPosition (400, 200);
  glutCreateWindow ("C e t v o r o u g l o v i - Q U A D _ S T R I P");
  init ();
  glutDisplayFunc(display);
  glutMainLoop();
  return 0;
/* ANSI C zahteva odgovor kroz pritisak bilo kog tastera. */
}
```

Odziv ove sintakse je prikazan na slici 3.



Slika 3

U tekstu **OpenGL(08)** prikazano je kako se radi sa različitim širinama („debljinama“) linija, tako da korisnik može da pročita i to, ako hoće da primeni različite širine linija. Ovo su predložene varijante, ali korisnik može parametre da menja proizvoljno, kako bi došao do željenog rezultata.

OpenGL (11)

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazan program koji omogućava crtanje proizvoljnog poligona, s tim što će se najpre odraditi poligon pomoću linija, a kasnije pomoću ispunjenih površi.

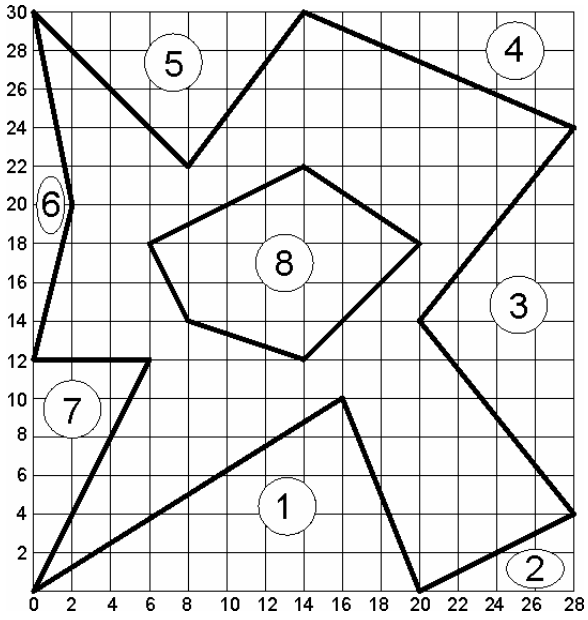
Komanda koja definiše boje koje će se primenjivati na sintaksu ispod je:

glColor3f (R, G, B)

gde se vrednosti komponenti nalaze u granicama od 0.0 do 1.0: **R od 0.0 do 1.0**, **G od 0.0 do 1.0** i **B od 0.0 do 1.0**. Ispod sledi spisak osnovnih boja koje se koriste u OpenGL-u:

glColor3f (0.0, 0.0, 0.0) – CRNA boja (black)
glColor3f (1.0, 0.0, 0.0) – CRVENA boja (red)
glColor3f (0.0, 1.0, 0.0) – ZELENA boja (green)
glColor3f (1.0, 1.0, 0.0) – ŽUTA boja (yellow)
glColor3f (0.0, 0.0, 1.0) – PLAVA boja (blue)
glColor3f (1.0, 0.0, 1.0) – LJUBIČASTA boja (magenta)
glColor3f (0.0, 1.0, 1.0) – SVETLO PLAVA boja (cyan)
glColor3f (1.0, 1.0, 1.0) – BELA boja (white)

Na slici 1 prikazana je geometrija željenog poligona koji treba nacrtati i kao konturu i kao ispunjene površine proizvoljne boje.



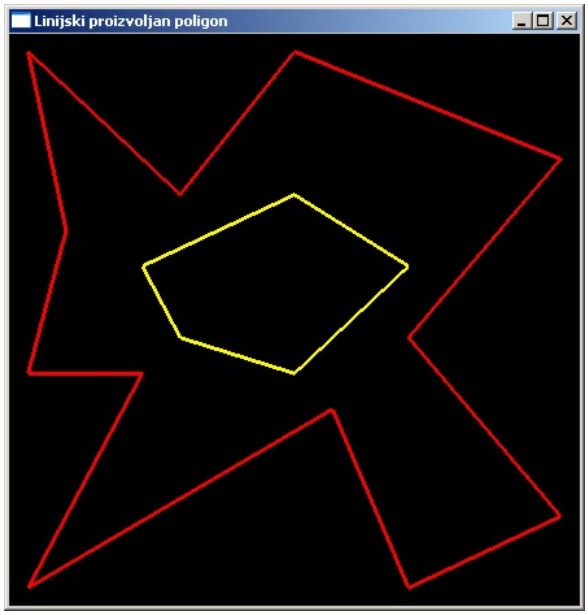
Slika 1

Sledi primer primene komandi koje omogućavaju iscrtavanje poligona pomoću linija. Rezultat donje sintakse prikazan je na slici 2.

```
/*
 * Crtanje proizvoljnog poligona linijama
 *    Dragan Cvetkovic, 14.02.2005.
 */

#include <GL/glut.h>
```

```
void display(void)

{
/* brise sve piksele iz bafera  */
   glClear (GL_COLOR_BUFFER_BIT);

/* Crta se glavna kontura */
   glColor3f (1.0, 0.0, 0.0);
   glLineWidth (3.0);
   glBegin(GL_LINE_LOOP);
      glVertex2f (0.0, 0.0);
      glVertex2f (16.0, 10.0);
      glVertex2f (20.0, 0.0);
      glVertex2f (28.0, 4.0);
     glVertex2f (20.0, 14.0);
      glVertex2f (28.0, 24.0);
      glVertex2f (14.0, 30.0);
      glVertex2f (8.0, 22.0);
     glVertex2f (0.0, 30.0);
      glVertex2f (2.0, 20.0);
      glVertex2f (0.0, 12.0);
      glVertex2f (6.0, 12.0);
   glEnd();

/* Crta se unutrasnja rupa */
   glColor3f (1.0, 1.0, 0.0);
   glLineWidth (3.0);
   glBegin(GL_LINE_LOOP);
      glVertex2f (14.0, 12.0);
      glVertex2f (20.0, 18.0);
      glVertex2f (14.0, 22.0);
      glVertex2f (6.0, 18.0);
      glVertex2f (8.0, 14.0);
   glEnd();

/* Izvrsava GL komande u konacnom vremenu */
   glFlush ();
}

void init (void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
   glClearColor (0.0, 0.0, 0.0, 0.0);

/* Definisu se vrednosti pogleda */
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(-1.0, 29.0, -1.0, 31.0, -1.0, 1.0);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
```

```
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * "Linijski proizvoljan poligon" u naslovu prozora.
 * Pozivaju se inicijalne rutine. Registruje se Callback
 * funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize (440, 440);
  glutInitWindowPosition (400, 200);
  glutCreateWindow ("Linijski proizvoljan poligon");
  init ();
  glutDisplayFunc(display);
  glutMainLoop();
  return 0;
/* ANSI C zahteva odgovor kroz pritisak bilo kog tastera. */
}
```



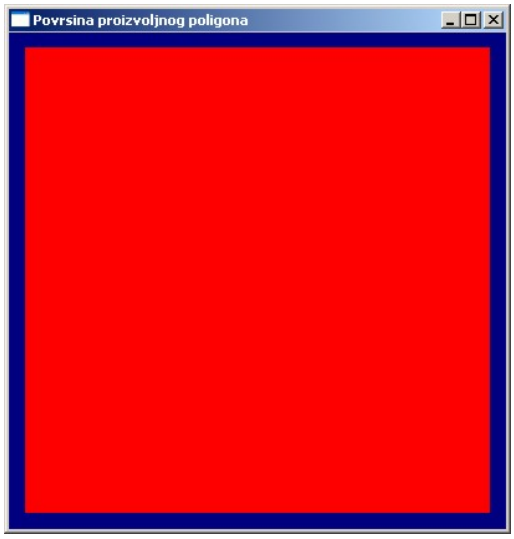
Slika 2

Sledi sintaksa programa koji iscrtava definisani poligon pomoću ispunjenih površina. Izabrane su plava i crvena boja kako bi se videle razlike.

```
/*
 * "Ispunjen" proizvoljan poligon
 *          14.02.2005.
 */

#include <GL/glut.h>

void display(void)

{
/* Brise sve piksele iz bafera  */
  glClear (GL_COLOR_BUFFER_BIT);

/* Crta se glavna kontura */
```

```
  glColor3f (1.0, 0.0, 0.0);
  glBegin(GL_POLYGON);
    glVertex2f (0.0, 0.0);
    glVertex2f (28.0, 0.0);
    glVertex2f (28.0, 30.0);
    glVertex2f (0.0, 30.0);
  glEnd();
```

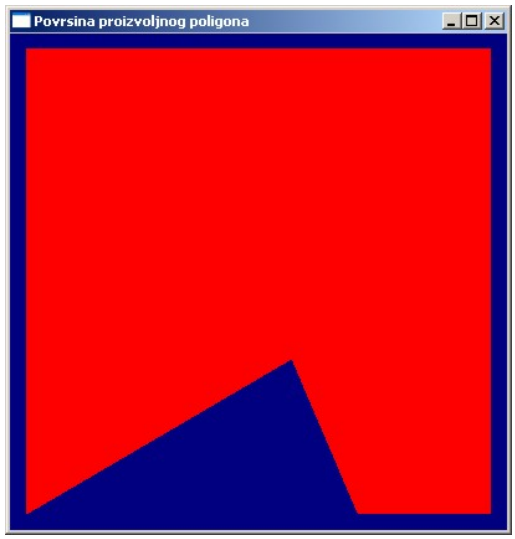
/ Rezultat gornje sintakse je prikazan na slici 3 */*



Slika 3

```
/*
 * Oduzima se trougao od 0 do 20 po X osi
 * (povrsina je na slici 1 oznacena brojem 1)
 */
  glColor3f (0.0, 0.0, 0.5);
  glBegin(GL_POLYGON);
    glVertex2f (0.0, 0.0);
    glVertex2f (20.0, 0.0);
    glVertex2f (16.0, 10.0);
  glEnd();
```

/ Rezultat gornje sintakse je prikazan na slici 4 */*

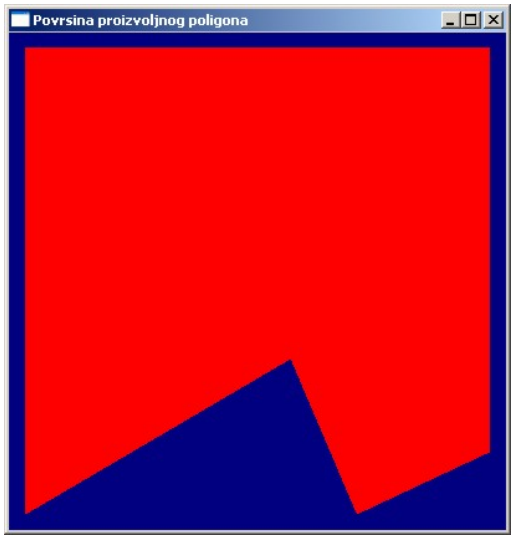


Slika 4

```
/*
```

```
 * Oduzima se trougao u donjem desnom uglu
 * (povrsina je na slici 1 oznacena brojem 2)
 */
  glColor3f (0.0, 0.0, 0.5);
  glBegin(GL_POLYGON);
    glVertex2f (20.0, 0.0);
    glVertex2f (28.0, 0.0);
    glVertex2f (28.0, 4.0);
  glEnd();
```

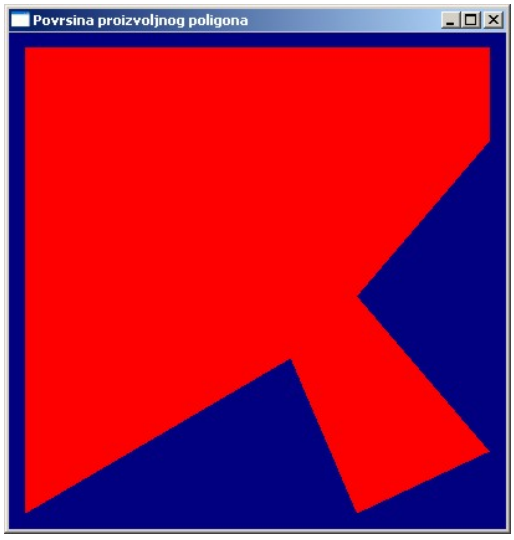
/ Rezultat gornje sintakse je prikazan na slici 5 */*



Slika 5

```
/*
 * Oduzima se trougao uz desnu ivicu prozora
 * (povrsina je na slici 1 oznacena brojem 3)
 */
  glColor3f (0.0, 0.0, 0.5);
  glBegin(GL_POLYGON);
    glVertex2f (28.0, 4.0);
    glVertex2f (28.0, 24.0);
    glVertex2f (20.0, 14.0);
  glEnd();
```

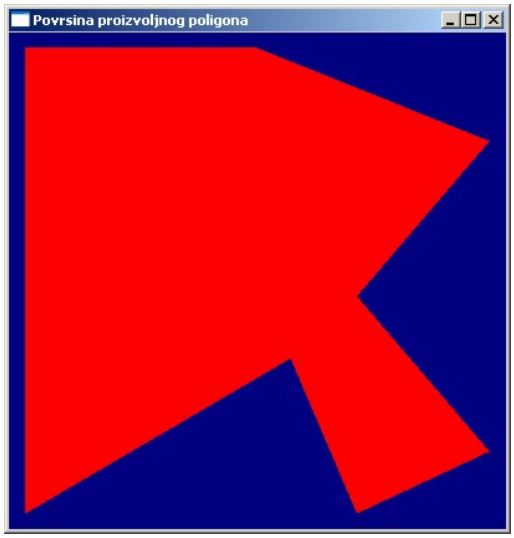
/ Rezultat gornje sintakse je prikazan na slici 6 */*



Slika 6

```
/*
 * Oduzima se trougao u gornjem desnom uglu
 * (povrsina je na slici 1 oznacena brojem 4)
 */
  glColor3f (0.0, 0.0, 0.5);
  glBegin(GL_POLYGON);
    glVertex2f (28.0, 24.0);
    glVertex2f (28.0, 30.0);
    glVertex2f (14.0, 30.0);
  glEnd();
```

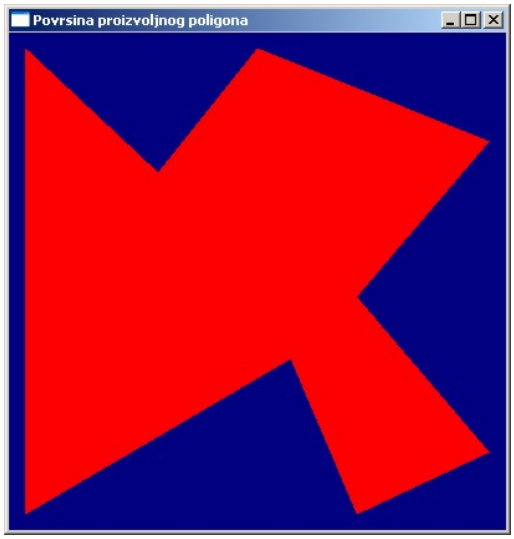
/ Rezultat gornje sintakse je prikazan na slici 7 */*



Slika 7

```
/*
 * Oduzima se trougao u gornjem levom uglu
 * (povrsina je na slici 1 oznacena brojem 5)
 */
  glColor3f (0.0, 0.0, 0.5);
  glBegin(GL_POLYGON);
    glVertex2f (14.0, 30.0);
    glVertex2f (0.0, 30.0);
    glVertex2f (8.0, 22.0);
  glEnd();
```

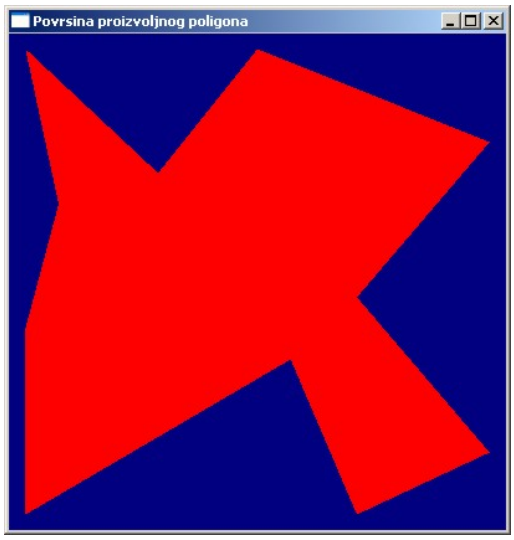
/ Rezultat gornje sintakse je prikazan na slici 8 */*



Slika 8

```
/*
 * Oduzima se trougao uz levu ivicu prozora
 * (povrsina je na slici 1 oznacena brojem 6)
 */
  glColor3f (0.0, 0.0, 0.5);
  glBegin(GL_POLYGON);
    glVertex2f (0.0, 30.0);
    glVertex2f (0.0, 12.0);
    glVertex2f (2.0, 20.0);
  glEnd();
```

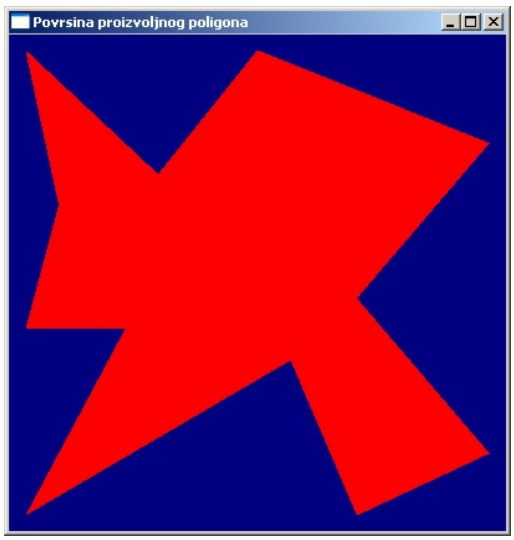
/ Rezultat gornje sintakse je prikazan na slici 9 */*



Slika 9

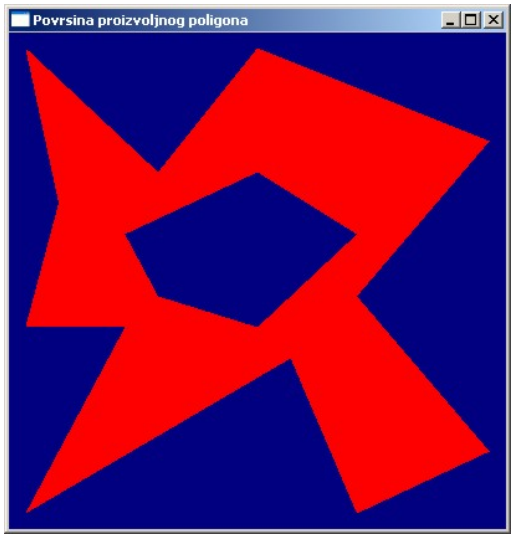
```
/*
 * Oduzima se trougao u donjem levu uglu
 * (povrsina je na slici 1 oznacena brojem 7)
 */
  glColor3f (0.0, 0.0, 0.5);
  glBegin(GL_POLYGON);
    glVertex2f (0.0, 0.0);
    glVertex2f (6.0, 12.0);
    glVertex2f (0.0, 12.0);
  glEnd();
```

/ Rezultat gornje sintakse je prikazan na slici 10 */*



Slika 10

```
/*
 * Oduzima se poligon u sredini (rupa)
 * (povrsina je na slici 1 oznacena brojem 8)
 */
   glColor3f (0.0, 0.0, 0.5);
   glBegin(GL_POLYGON);
      glVertex2f (14.0, 12.0);
      glVertex2f (20.0, 18.0);
      glVertex2f (14.0, 22.0);
      glVertex2f (6.0, 18.0);
      glVertex2f (8.0, 14.0);
   glEnd();
/* Rezultat gornje sintakse je  prikazan na slici 11  */
```



Slika 11

```
/* Izvrsava GL komande u konacnom vremenu */
   glFlush ();
}

void init (void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
```

```
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 * Ovde je izabrana tamno plava na sredini izmedju crne i plave!
 */
  glClearColor (0.0, 0.0, 0.5, 0.0);

/* Definisu se vrednosti pogleda */
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(-1.0, 29.0, -1.0, 31.0, -1.0, 1.0);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * "Povrsina proizvoljnog poligona" u naslovu prozora.
 * Pozivaju se inicijalne rutine. Registruje se Callback
 * funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize (400, 400);
  glutInitWindowPosition (400, 200);
  glutCreateWindow ("Povrsina proizvoljnog poligona");
  init ();
  glutDisplayFunc(display);
  glutMainLoop();
  return 0;
/* ANSI C zahteva odgovor kroz pritisak bilo kog tastera. */
}
```

Na kraju ovoga priložena je samo “suva” sintaksa gornjeg programa, kako bi se izbegle moguće nesuglasice i eventualni problemi.

```
#include <GL/glut.h>
void display(void)
{
  glClear (GL_COLOR_BUFFER_BIT);
  glColor3f (1.0, 0.0, 0.0);
  glBegin(GL_POLYGON);
    glVertex2f (0.0, 0.0);
    glVertex2f (28.0, 0.0);
    glVertex2f (28.0, 30.0);
    glVertex2f (0.0, 30.0);
  glEnd();
  glColor3f (0.0, 0.0, 0.5);
  glBegin(GL_POLYGON);
    glVertex2f (0.0, 0.0);
    glVertex2f (20.0, 0.0);
    glVertex2f (16.0, 10.0);
  glEnd();
  glColor3f (0.0, 0.0, 0.5);
  glBegin(GL_POLYGON);
    glVertex2f (20.0, 0.0);
```

```
    glVertex2f (28.0, 0.0);
    glVertex2f (28.0, 4.0);
  glEnd();
  glColor3f (0.0, 0.0, 0.5);
  glBegin(GL_POLYGON);
    glVertex2f (28.0, 4.0);
    glVertex2f (28.0, 24.0);
    glVertex2f (20.0, 14.0);
  glEnd();
  glColor3f (0.0, 0.0, 0.5);
  glBegin(GL_POLYGON);
    glVertex2f (28.0, 24.0);
    glVertex2f (28.0, 30.0);
    glVertex2f (14.0, 30.0);
  glEnd();
  glColor3f (0.0, 0.0, 0.5);
  glBegin(GL_POLYGON);
    glVertex2f (14.0, 30.0);
    glVertex2f (0.0, 30.0);
    glVertex2f (8.0, 22.0);
  glEnd();
  glColor3f (0.0, 0.0, 0.5);
  glBegin(GL_POLYGON);
    glVertex2f (0.0, 30.0);
    glVertex2f (0.0, 12.0);
    glVertex2f (2.0, 20.0);
  glEnd();
  glColor3f (0.0, 0.0, 0.5);
  glBegin(GL_POLYGON);
    glVertex2f (0.0, 0.0);
    glVertex2f (6.0, 12.0);
    glVertex2f (0.0, 12.0);
  glEnd();
  glColor3f (0.0, 0.0, 0.5);
  glBegin(GL_POLYGON);
    glVertex2f (14.0, 12.0);
    glVertex2f (20.0, 18.0);
    glVertex2f (14.0, 22.0);
        glVertex2f (6.0, 18.0);
    glVertex2f (8.0, 14.0);
  glEnd();
  glFlush ();
}
void init (void)
{
  glClearColor (0.0, 0.0, 0.5, 0.0);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(-1.0, 29.0, -1.0, 31.0, -1.0, 1.0);
}
int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize (400, 400);
  glutInitWindowPosition (400, 200);
  glutCreateWindow ("Povrsina proizvoljnog poligona");
```

```
   init ();
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
}
```

Korisniku se ostavlja da sam bira boje, kao i put kojim će da realizuje priloženi poligon.

OpenGL (12)

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazani programi koji omogućavaju crtanje određenih 3D primitiva. Reč je o sledećim primitivima:

- ***Sphere*** – lopta,
- ***Cube*** – kocka,
- ***Cone*** – kupa,
- ***Torus*** – torus,
- ***Dodecahedron*** – dodekaedar ili dvanaestostranični model,
- ***Tetrahedron*** – tetraedar,
- ***Octahedron*** – oktaedar,
- ***Icosahedron*** – ikosaedar ili dvadesetostranični model, i
- ***Teapot*** – čajnik.

Da bi se kreirali primitivi, trebalo bi razmotriti nekoliko komandi »vezanih« za 3D okruženje unutar OpenGL-a.

Komanda koja definiše boje koje će se primenjivati na sintaksu ispod je:

glColor3f (R, G, B)

gde se vrednosti komponenti nalaze u granicama od 0.0 do 1.0: **R od 0.0 do 1.0**, **G od 0.0 do 1.0** i **B od 0.0 do 1.0**. Ispod sledi spisak osnovnih boja koje se koriste u OpenGL-u:

glColor3f (0.0, 0.0, 0.0) – CRNA boja (black)
glColor3f (1.0, 0.0, 0.0) – CRVENA boja (red)
glColor3f (0.0, 1.0, 0.0) – ZELENA boja (green)
glColor3f (1.0, 1.0, 0.0) – ŽUTA boja (yellow)
glColor3f (0.0, 0.0, 1.0) – PLAVA boja (blue)
glColor3f (1.0, 0.0, 1.0) – LJUBIČASTA boja (magenta)
glColor3f (0.0, 1.0, 1.0) – SVETLO PLAVA boja (cyan)
glColor3f (1.0, 1.0, 1.0) – BELA boja (white)

Sledeća komanda je komanda koja omogućava senčenje, tj. ispunu 3D primitiva. Reč je o komandi

glShadeModel (GL_FLAT / GL_SMOOTH);

Poligon ili 3D primitiv može da bude ispunjen jednom bojom (FLAT) ili sa mnogo boja (SMOOTH ili GOURAUD). Ovde se bira ili GL_FLAT ili GL_SMOOTH. Kod FLAT senčenja boja svakog verteksa se duplira kroz sve vertekse primitiva, a kod SMOOTH senčenja boja svakog verteksa se posmatra zasebno.

Sledeća komanda koja se koristi je

glLoadIdentity ();

Ova komanda se koristi za inicijalizaciju tekuće matrice projekcije, tako da JEDINO definisane transformacije projekcije imaju efekta. Nakon ovoga može da se koristi komanda **glFrustum()** sa argumentima koji definišu parametre transformisanja projekcije.

Sledeća komanda koja se koristi je

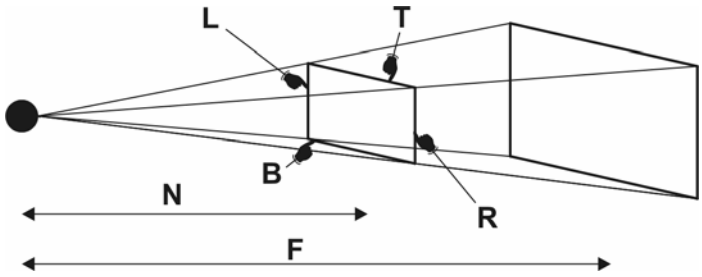
gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

Argumenti ove komande definišu položaj kamere (čovečijeg oka), gde "gleda" (nišani) kamera i vektor kamere (*up vector*). U gornjoj komandi kamera je smeštena u tački (0,0,5), "nišani" tačku (0,0,0) i definiše vektor sa (0,1,0). VEKTOR definiše jedinstvenu orijentaciju kamere.

Sledeća komanda koja se koristi je

glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);

Ova komanda omogućava definisanje zapreminskog pogleda u perspektivi. Uopštena sintaksa za ovu komandu je **glFrustum (L, R, B, T, N, F);** a objašnjenje je prikazano na donjoj slici.



Sledeća komanda je

glTranslatef (1.0, 2.0, 3.0);

Ova komanda definiše translaciju duž tri ose tekućeg koordinatnog sistema. Prvi broj (1.0) definiše transliranje duž X ose, drugi broj (2.0) definiše transliranje duž Y ose i treći broj (3.0) definiše transliranje duž Z ose.

Sledeća komanda je

glScalef (1.0, 2.0, 1.0);

Ova komanda definiše faktore skaliranja duž tri ose tekućeg koordinatnog sistema. Ako su sve tri vrednosti 1.0, onda ova komanda nema efekta. Ako su faktori različiti, onda se originalni objekti transformišu, jer ova komanda omogućava različito skaliranje duž pojedinih osa.

Sledeća komanda koja se koristi je

glRotatef (45.0, 0.0, 0.0, 1.0);

Ova komanda definiše rotiranje oko osa tekućeg koordinatnog sistema. Prvi broj predstavlja vrednost ugla u stepenima, a preostala tri broja predstavljaju koordinate tačke kroz koju prolazi zrak iz koordinatnog početka. Rotiranje se odvija u suprotnom smeru od smera kretanja kazaljke na satu. Gornja sintaksa vrši rotaciju objekta oko Z ose za ugao od 45 stepeni u smeru suprotnom od smera kretanja kazaljke na satu.

Komanda koja definiše LOPTU u potpunosti je

glutWireSphere (I, II, III);

Ova komanda definiše žičani model lopte, gde su argumenti:

- **I** – poluprečnik lopte
- **II** – broj segmenata oko Z ose
- **III** – broj segmenata duž Z ose

Sledeća komanda

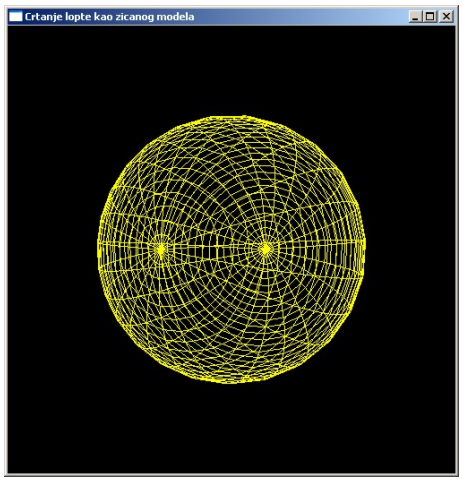
glutSolidSphere (I, II, III);

definiše loptu kao solid, a argumenti su isti kao kod prethodne komande.

Evo nekoliko primera kako će izgledati lopta sa različitim paramaterima, a kompletna sintaksa sledi ispod. Za komandu

glutWireSphere (1.0, 25, 40);

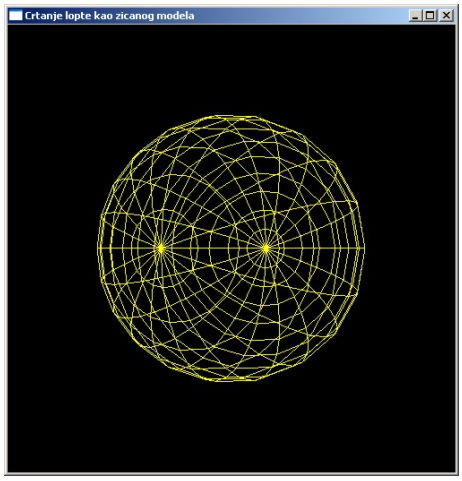
dobija se sledeća slika.



Za komandu

glutWireSphere (1.0, 20, 15);

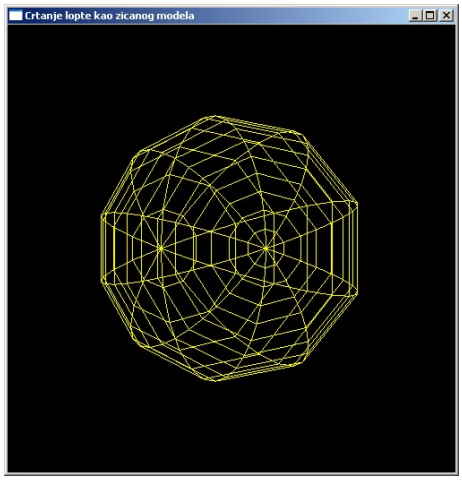
dobija se sledeća slika.



Za komandu

glutWireSphere (1.0, 10, 15);

dobija se sledeća slika.

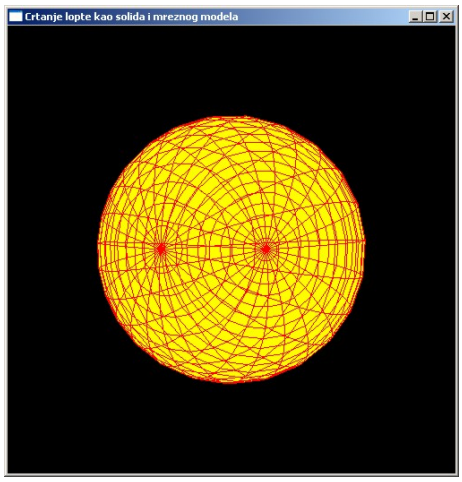


Postoji varijanta i da se iskombinuju i žičani i osenčeni model lopte. Moguća sintaksa za jedan od načina je:

glutSolidSphere (1.0, 25, 40);

glColor3f (1.0, 0.0, 0.0);
glutWireSphere (1.0, 25, 25);

a rezultat je prikazan na donjoj slici.



Celokupna sintaksa za gornju sliku je:

```
#include <GL/glut.h>
void init(void)
{
/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
glClearColor (0.0, 0.0, 0.0, 0.0);

/* Model sencenja - FLAT ili SMOOTH */
glShadeModel (GL_FLAT);
}
void display(void)
{
/* brise sve piksele iz bafera  */
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 0.0);

/* "Praznjenje" matrice */
glLoadIdentity ();

/* Transformacija pogleda */
gluLookAt (2.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

/* Transformacija modela - skaliranje */
glScalef (2.0, 2.0, 2.0);

/* Crtanje 3D entiteta */
glutSolidSphere (1.0, 25, 40);
glColor3f (1.0, 0.0, 0.0);
glutWireSphere (1.0, 25, 25);

/* Izvrsava GL komande u konacnom vremenu */
glFlush ();
}
void reshape (int w, int h)
```

```
{

/* Definisu se vrednosti pogleda */
glMatrixMode (GL_PROJECTION);
glLoadIdentity ();

/* Definise projekciju u perspektivi */
glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
glMatrixMode (GL_MODELVIEW);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * u naslovu prozora. Pozivaju se inicijalne rutine.
 * Registruje se Callback funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)

{
glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
glutInitWindowPosition (300, 100);
glutCreateWindow ("Crtanje lopte kao solida i mreznog modela");
init ();
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutMainLoop();
return 0;
}
```

Korisniku se ostavlja da sam bira boje, kao i put kojim će da realizuje 3D primitive. U sledećem tekstu će biti obrađeni i ostali 3D primitivi.

OpenGL (13)

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazani programi koji omogućavaju crtanje određenih 3D primitiva. Reč je o sledećim primitivima:

- ***Sphere*** – lopta,
- ***Cube*** – kocka,
- ***Cone*** – kupa,
- ***Torus*** – torus,
- ***Dodecahedron*** – dodekaedar ili dvanaestostranični model,
- ***Tetrahedron*** – tetraedar,
- ***Octahedron*** – oktaedar,
- ***Icosahedron*** – ikosaedar ili dvadesetostranični model, i
- ***Teapot*** – čajnik.

Da bi se kreirali primitivi, trebalo bi razmotriti nekoliko komandi »vezanih« za 3D okruženje unutar OpenGL-a.

Komanda koja definiše boje koje će se primenjivati na sintaksu ispod je:

glColor3f (R, G, B)

gde se vrednosti komponenti nalaze u granicama od 0.0 do 1.0: **R od 0.0 do 1.0**, **G od 0.0 do 1.0** i **B od 0.0 do 1.0**. Ispod sledi spisak osnovnih boja koje se koriste u OpenGL-u:

glColor3f (0.0, 0.0, 0.0) – CRNA boja (black)
glColor3f (1.0, 0.0, 0.0) – CRVENA boja (red)
glColor3f (0.0, 1.0, 0.0) – ZELENA boja (green)
glColor3f (1.0, 1.0, 0.0) – ŽUTA boja (yellow)
glColor3f (0.0, 0.0, 1.0) – PLAVA boja (blue)
glColor3f (1.0, 0.0, 1.0) – LJUBIČASTA boja (magenta)
glColor3f (0.0, 1.0, 1.0) – SVETLO PLAVA boja (cyan)
glColor3f (1.0, 1.0, 1.0) – BELA boja (white)

Komanda koja definiše KOCKU u potpunosti je

glutWireCube (I);

Ova komanda definiše žičani model lopte, gde je argument:

- **I** – dužina stranice kocke

Sledeća komanda

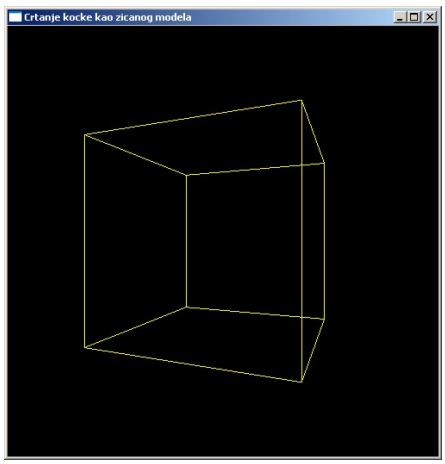
glutSolidCube (I);

definiše loptu kao solid, a argumenti su isti kao kod prethodne komande.

Evo nekoliko primera kako će izgledati kocka sa različitim paramaterima, a kompletna sintaksa sledi ispod. Za komandu

glutWireCube (1.0);

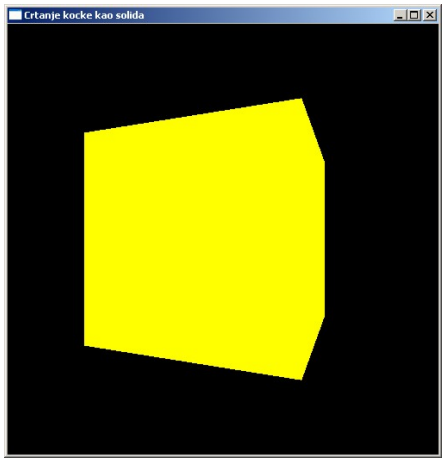
dobija se sledeća slika.



Za komandu

glutSolidCube (1.0);

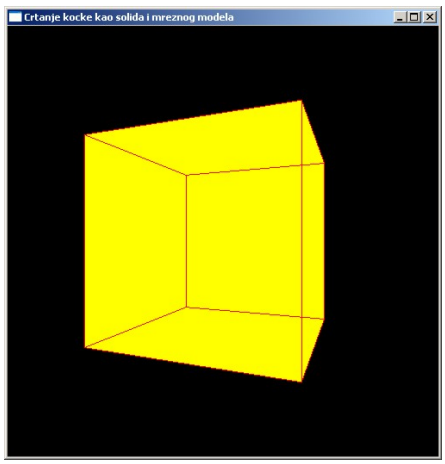
dobija se sledeća slika.



Postoji varijanta i da se iskombinuju i žičani i osenčeni model kocke. Moguća sintaksa za jedan od načina je:

glutSolidCube (1.0);
glColor3f (1.0, 0.0, 0.0);
glutWireCube (1.0);

a rezultat je prikazan na donjoj slici.

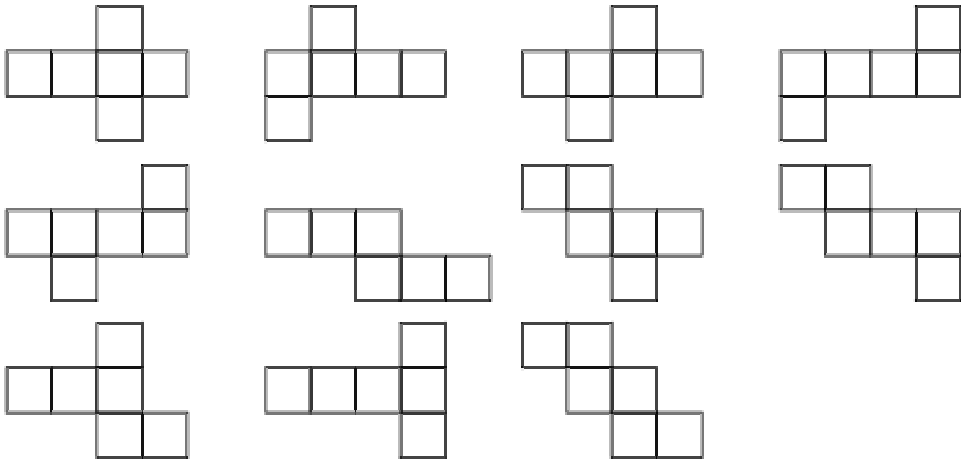


Celokupna sintaksa za gornju sliku je:

```
#include <GL/glut.h>
void init(void)

{
/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
glClearColor (0.0, 0.0, 0.0, 0.0);

/* Model sencenja - FLAT ili SMOOTH */
glShadeModel (GL_FLAT);
}

void display(void)

{
/* brise sve piksele iz bafera  */
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 0.0);

/* "Praznjenje" matrice */
glLoadIdentity ();

/* Transformacija pogleda */
gluLookAt (2.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

/* Transformacija modela - skaliranje */
glScalef (3.0, 3.0, 3.0);

/* Crtanje 3D entiteta */
glutSolidCube (1.0);
glColor3f (1.0, 0.0, 0.0);
glutWireCube (1.0);

/* Izvrsava GL komande u konacnom vremenu */
glFlush ();
}

void reshape (int w, int h)

{

// glViewport (0, 0, (GLsizei) w, (GLsizei) h);
/* Definisu se vrednosti pogleda */
glMatrixMode (GL_PROJECTION);
glLoadIdentity ();

/* Definise projekciju u perspektivi */
glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
glMatrixMode (GL_MODELVIEW);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
```

```
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * u naslovu prozora. Pozivaju se inicijalne rutine.
 * Registruje se Callback funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)
{
glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
glutInitWindowPosition (300, 100);
glutCreateWindow ("Crtanje kocke kao solida i mreznog modela");
init ();
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutMainLoop();
return 0;
}
```

Izgledi otvorene mreže za kocku prikazani su na donjoj slici.



A model kocke je prikazan na donjoj slici.

Korisniku se ostavlja da sam bira boje, kao i put kojim će da realizuje 3D primitive. U sledećem tekstu će se nastaviti sa obradom 3D primitiva.

OpenGL (14)

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazani programi koji omogućavaju crtanje određenih 3D primitiva. Reč je o sledećim primitivima:

- ***Sphere*** – lopta,
- ***Cube*** – kocka,
- ***Cone*** – kupa,
- ***Torus*** – torus,
- ***Dodecahedron*** – dodekaedar ili dvanaestostranični model,
- ***Tetrahedron*** – tetraedar,
- ***Octahedron*** – oktaedar,
- ***Icosahedron*** – ikosaedar ili dvadesetostranični model, i
- ***Teapot*** – čajnik.

Da bi se kreirali primitivi, trebalo bi razmotriti nekoliko komandi »vezanih« za 3D okruženje unutar OpenGL-a.

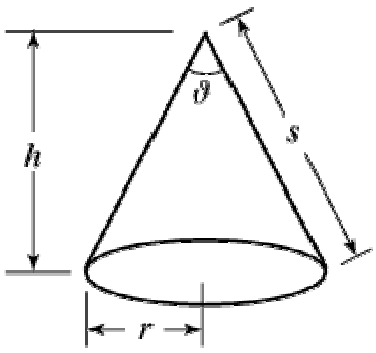
Komanda koja definiše boje koje će se primenjivati na sintaksu ispod je:

glColor3f (R, G, B)

gde se vrednosti komponenti nalaze u granicama od 0.0 do 1.0: **R od 0.0 do 1.0**, **G od 0.0 do 1.0** i **B od 0.0 do 1.0**. Ispod sledi spisak osnovnih boja koje se koriste u OpenGL-u:

glColor3f (0.0, 0.0, 0.0) – CRNA boja (black)
glColor3f (1.0, 0.0, 0.0) – CRVENA boja (red)
glColor3f (0.0, 1.0, 0.0) – ZELENA boja (green)
glColor3f (1.0, 1.0, 0.0) – ŽUTA boja (yellow)
glColor3f (0.0, 0.0, 1.0) – PLAVA boja (blue)
glColor3f (1.0, 0.0, 1.0) – LJUBIČASTA boja (magenta)
glColor3f (0.0, 1.0, 1.0) – SVETLO PLAVA boja (cyan)
glColor3f (1.0, 1.0, 1.0) – BELA boja (white)

Geometrija kupe prikazana je na donjoj slici.



Komanda koja definiše KUPU u potpunosti je

glutWireCone (I, II, III, IV);

Ova komanda definiše žičani model kupe, gde su argumenti:
I – poluprečnik osnove kupe
II – visina kupe
III – broj segmenata oko Z ose
IV – broj segmenata duž Z ose

Sledeća komanda

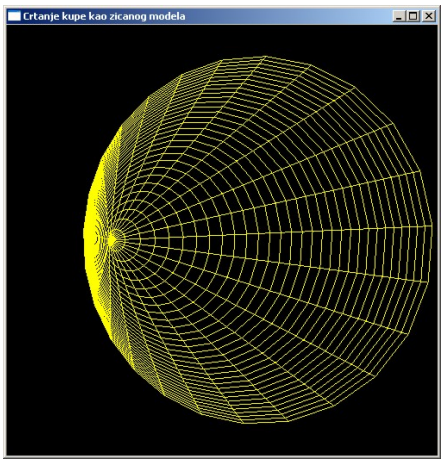
glutSolidCone (I, II, III, IV);

definiše kupu kao solid, a argumenti su isti kao kod prethodne komande.

Evo nekoliko primera kako će izgledati kupa sa različitim paramaterima, a kompletna sintaksa sledi ispod. Za komandu

glutWireCone (1.0, 0.9, 25, 30);

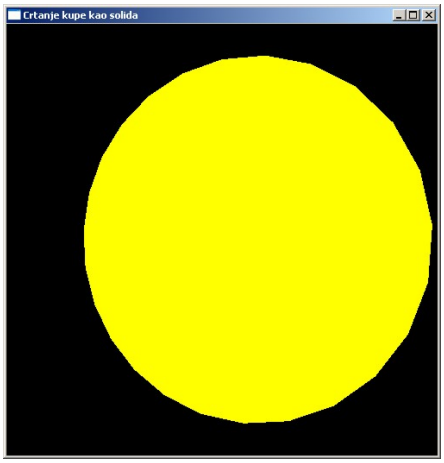
dobija se sledeća slika.



Za komandu

glutSolidCone (1.0, 0.9, 25, 30);

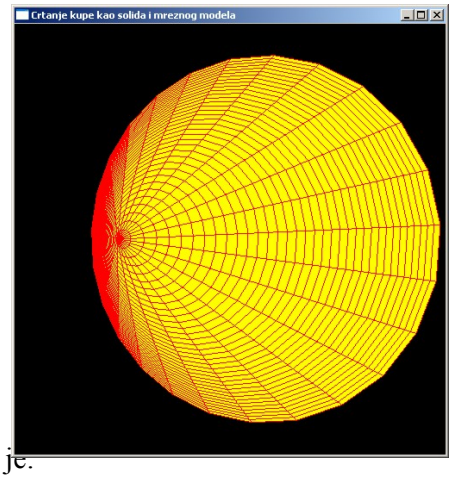
dobija se sledeća slika.



Postoji varijanta i da se iskombinuju i žičani i osenčeni model kupe. Moguća sintaksa za jedan od načina je:

glutSolidCone (1.0, 0.9, 25, 30);
glColor3f (1.0, 0.0, 0.0);
glutWireCone (1.0, 0.9, 25, 35);

a rezultat je prikazan na donjoj slici.



Celokupna sintaksa za gornju sliku je:

```
#include <GL/glut.h>
void init(void)
{
/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
glClearColor (0.0, 0.0, 0.0, 0.0);

/* Model sencenja - FLAT ili SMOOTH */
glShadeModel (GL_FLAT);
}

void display(void)

{
/* brise sve piksele iz bafera  */
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 0.0);

/* "Praznjenje" matrice */
glLoadIdentity ();

/* Transformacija pogleda */
gluLookAt (2.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

/* Transformacija modela - skaliranje */
glScalef (3.0, 3.0, 3.0);

/* Crtanje 3D entiteta */
glutSolidCone (1.0, 0.9, 25, 30);
glColor3f (1.0, 0.0, 0.0);
glutWireCone (1.0, 0.9, 25, 35);

/* Izvrsava GL komande u konacnom vremenu */
glFlush ();
}

void reshape (int w, int h)

{
```

```
/* Definisu se vrednosti pogleda */
glMatrixMode (GL_PROJECTION);
glLoadIdentity ();

/* Definise projekciju u perspektivi */
glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
glMatrixMode (GL_MODELVIEW);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * u naslovu prozora. Pozivaju se inicijalne rutine.
 * Registruje se Callback funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)

{
glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
glutInitWindowPosition (300, 100);
glutCreateWindow ("Crtanje kupe kao solida i mreznog modela");
init ();
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutMainLoop();
return 0;
}
```

Korisniku se ostavlja da sam bira boje, kao i put kojim će da realizuje 3D primitive. U sledećem tekstu će se nastaviti sa obradom 3D primitiva.

OpenGL (15)

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazani programi koji omogućavaju crtanje određenih 3D primitiva. Reč je o sledećim primitivima:

- ***Sphere*** – lopta,
- ***Cube*** – kocka,
- ***Cone*** – kupa,
- ***Torus*** – torus,
- ***Dodecahedron*** – dodekaedar ili dvanaestostranični model,
- ***Tetrahedron*** – tetraedar,
- ***Octahedron*** – oktaedar,
- ***Icosahedron*** – ikosaedar ili dvadesetostranični model, i
- ***Teapot*** – čajnik.

Da bi se kreirali primitivi, trebalo bi razmotriti nekoliko komandi »vezanih« za 3D okruženje unutar OpenGL-a.

Komanda koja definiše boje koje će se primenjivati na sintaksu ispod je:

glColor3f (R, G, B)

gde se vrednosti komponenti nalaze u granicama od 0.0 do 1.0: **R od 0.0 do 1.0**, **G od 0.0 do 1.0** i **B od 0.0 do 1.0**. Ispod sledi spisak osnovnih boja koje se koriste u OpenGL-u:

glColor3f (0.0, 0.0, 0.0) – CRNA boja (black)
glColor3f (1.0, 0.0, 0.0) – CRVENA boja (red)
glColor3f (0.0, 1.0, 0.0) – ZELENA boja (green)
glColor3f (1.0, 1.0, 0.0) – ŽUTA boja (yellow)
glColor3f (0.0, 0.0, 1.0) – PLAVA boja (blue)
glColor3f (1.0, 0.0, 1.0) – LJUBIČASTA boja (magenta)
glColor3f (0.0, 1.0, 1.0) – SVETLO PLAVA boja (cyan)
glColor3f (1.0, 1.0, 1.0) – BELA boja (white)

Komanda koja definiše TORUS u potpunosti je

glutWireTorus (I, II, III, IV);

Ova komanda definiše žičani model torusa, gde su argumenti:
I – unutrašnji poluprečnik torusa,
II – spoljašnji poluprečnik torusa,
III – broj strana za svaku radijalnu sekciju, i
IV – broj radijalnih segmenata torusa.

Sledeća komanda

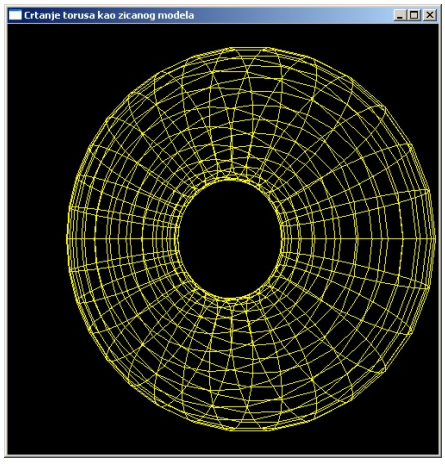
glutSolidTorus (I, II, III, IV);

definiše torus kao solid, a argumenti su isti kao kod prethodne komande.

Evo nekoliko primera kako će izgledati torus sa različitim paramaterima, a kompletna sintaksa sledi ispod. Za komandu

glutWireTorus (0.5, 1.0, 25, 30);

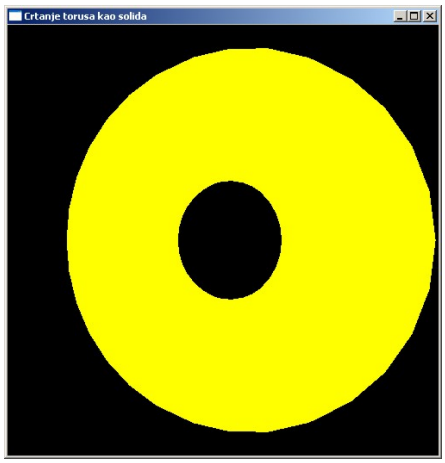
dobija se sledeća slika.



Za komandu

glutSolidTorus (0.5, 1.0, 25, 30);

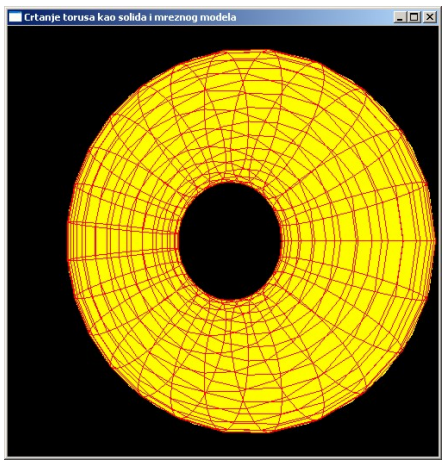
dobija se sledeća slika.



Postoji varijanta i da se iskombinuju i žičani i osenčeni model torusa. Moguća sintaksa za jedan od načina je:

```
glutSolidTorus (0.5, 1.0, 25, 30);
glColor3f (1.0, 0.0, 0.0);
glutWireTorus (0.5, 1.0, 25, 25);
```

a rezultat je prikazan na donjoj slici.

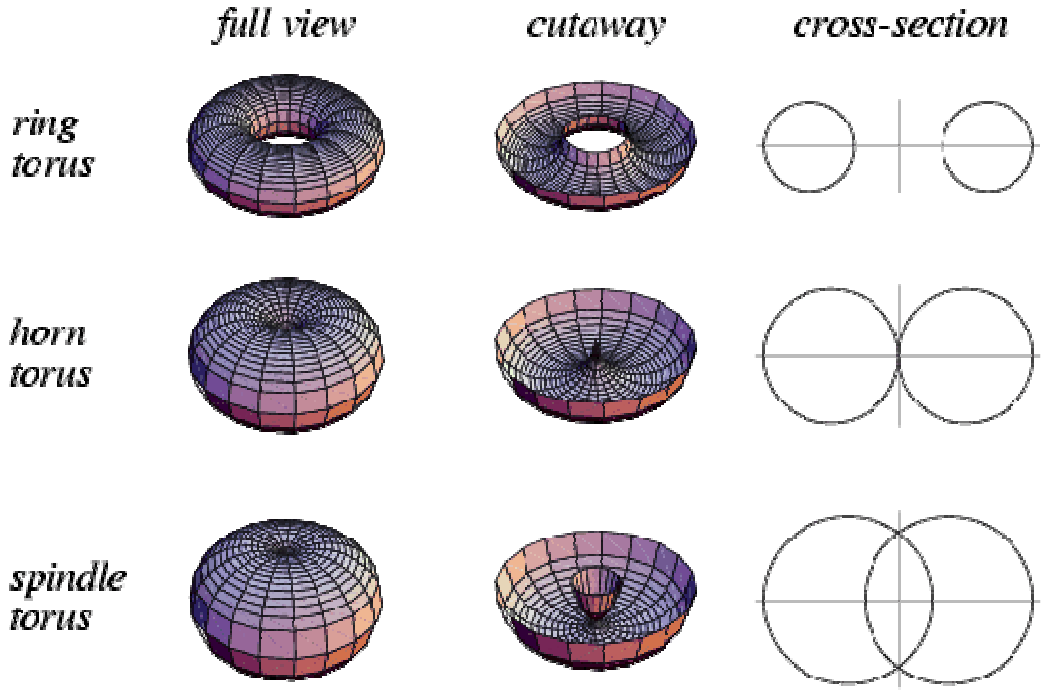


Celokupna sintaksa za gornju sliku je:

```
#include <GL/glut.h>
void init(void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */

glClearColor (0.0, 0.0, 0.0, 0.0);

/* Model sencenja - FLAT ili SMOOTH */
glShadeModel (GL_FLAT);

}
void display(void)
{

/* brise sve piksele iz bafera  */

glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 0.0);

/* "Praznjenje" matrice */

glLoadIdentity ();

/* Transformacija pogleda */

gluLookAt (2.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

/* Transformacija modela - skaliranje */

glScalef (2.0, 2.0, 2.0);

/* Crtanje 3D entiteta */

glutSolidTorus (0.5, 1.0, 25, 30);
glColor3f (1.0, 0.0, 0.0);
glutWireTorus (0.5, 1.0, 25, 25);

/* Izvrsava GL komande u konacnom vremenu */

glFlush ();
}
void reshape (int w, int h)
{

/* Definisu se vrednosti pogleda */

glMatrixMode (GL_PROJECTION);
glLoadIdentity ();

/* Definise projekciju u perspektivi */
```

```
glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
glMatrixMode (GL_MODELVIEW);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * u naslovu prozora. Pozivaju se inicijalne rutine.
 * Registruje se Callback funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */

int main(int argc, char** argv)
{
glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
glutInitWindowPosition (300, 100);
glutCreateWindow ("Crtanje torusa kao solida i mreznog modela");
init ();
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutMainLoop();
return 0;
}
```

Na narednoj slici prikayane su tri vrste torusa, što je direktna funkcija vrednosti dva poluprečnika.



Korisniku se ostavlja da sam bira boje, kao i put kojim će da realizuje 3D primitive. U sledećem tekstu će se nastaviti sa obradom 3D primitiva.

OpenGL (16)

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazani programi koji omogućavaju crtanje određenih 3D primitiva. Reč je o sledećim primitivima:

- ***Sphere*** – lopta,
- ***Cube*** – kocka,
- ***Cone*** – kupa,
- ***Torus*** – torus,
- ***Dodecahedron*** – dodekaedar ili dvanaestostranični model,
- ***Tetrahedron*** – tetraedar,
- ***Octahedron*** – oktaedar,
- ***Icosahedron*** – ikosaedar ili dvadesetostranični model, i
- ***Teapot*** – čajnik.

Da bi se kreirali primitivi, trebalo bi razmotriti nekoliko komandi »vezanih« za 3D okruženje unutar OpenGL-a.

Komanda koja definiše boje koje će se primenjivati na sintaksu ispod je:

glColor3f (R, G, B)

gde se vrednosti komponenti nalaze u granicama od 0.0 do 1.0: **R od 0.0 do 1.0**, **G od 0.0 do 1.0** i **B od 0.0 do 1.0**. Ispod sledi spisak osnovnih boja koje se koriste u OpenGL-u:

glColor3f (0.0, 0.0, 0.0) – CRNA boja (black)
glColor3f (1.0, 0.0, 0.0) – CRVENA boja (red)
glColor3f (0.0, 1.0, 0.0) – ZELENA boja (green)
glColor3f (1.0, 1.0, 0.0) – ŽUTA boja (yellow)
glColor3f (0.0, 0.0, 1.0) – PLAVA boja (blue)
glColor3f (1.0, 0.0, 1.0) – LJUBIČASTA boja (magenta)
glColor3f (0.0, 1.0, 1.0) – SVETLO PLAVA boja (cyan)
glColor3f (1.0, 1.0, 1.0) – BELA boja (white)

Komanda koja definiše DODEKAEDAR u potpunosti je

glutWireDodecahedron ();

Ova komanda definiše žičani model dodekaedra (dvanaestostraničnog modela) čiji je poluprečnik $\sqrt{3}$. Sledeća komanda

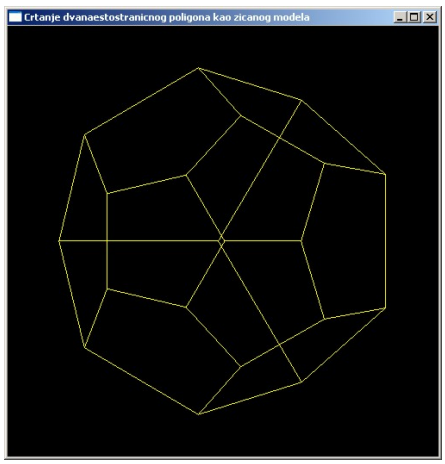
glutSolidDodecahedron ();

definiše dodekaedar kao solid, a komentar je isti kao kod prethodne komande.

Evo nekoliko primera kako će izgledati dodekaedar sa različitim paramaterima, a kompletna sintaksa sledi ispod. Za komandu

glutWireDodecahedron ();

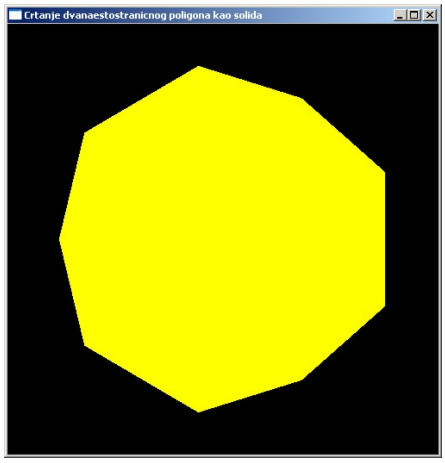
dobija se sledeća slika.



Za komandu

glutSolidDodecahedron ();

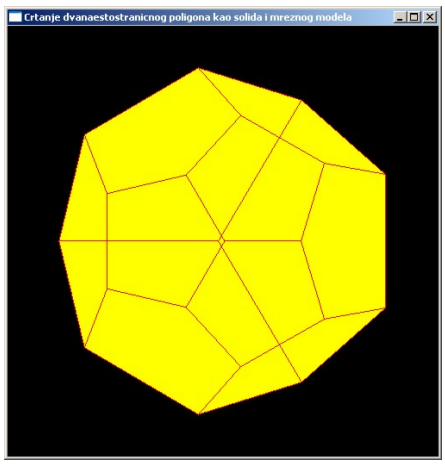
dobija se sledeća slika.



Postoji varijanta i da se iskombinuju i žičani i osenčeni model dodekaedra. Moguća sintaksa za jedan od načina je:

```
glutSolidDodecahedron ();
glColor3f (1.0, 0.0, 0.0);
glutWireDodecahedron ();
```

a rezultat je prikazan na donjoj slici.

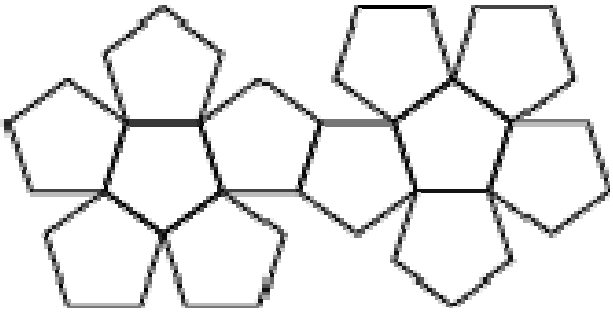


Celokupna sintaksa za gornju sliku je:

```
#include <GL/glut.h>
void init(void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */

glClearColor (0.0, 0.0, 0.0, 0.0);

/* Model sencenja - FLAT ili SMOOTH */

glShadeModel (GL_FLAT);
}
void display(void)
{

/* brise sve piksele iz bafera  */

glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 0.0);

/* "Praznjenje" matrice */

glLoadIdentity ();

/* Transformacija pogleda */

gluLookAt (2.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

/* Transformacija modela - skaliranje */

glScalef (1.5, 1.5, 1.5);

/* Crtanje 3D entiteta */

glutSolidDodecahedron ();
glColor3f (1.0, 0.0, 0.0);
glutWireDodecahedron ();

/* Izvrsava GL komande u konacnom vremenu */

glFlush ();
}
void reshape (int w, int h)
{

/* Definisu se vrednosti pogleda */

glMatrixMode (GL_PROJECTION);
glLoadIdentity ();

/* Definise projekciju u perspektivi */
```

```
glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
glMatrixMode (GL_MODELVIEW);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * u naslovu prozora. Pozivaju se inicijalne rutine.
 * Registruje se Callback funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */

int main(int argc, char** argv)
{
glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
glutInitWindowPosition (300, 100);
glutCreateWindow ("Crtanje dvanaestostranicnog poligona kao solida i mreznog modela");
init ();
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutMainLoop();
return 0;
}
```

Na narednoj slici (levi deo) prikazana je mreža dodekaedra kako bi se željeni model napravio, a na desnoj strani te slike prikazan je “sklopljen” model.



Korisniku se ostavlja da sam bira boje, kao i put kojim će da realizuje 3D primitive. U sledećem tekstu će se nastaviti sa obradom 3D primitiva.

OpenGL (17)

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazani programi koji omogućavaju crtanje određenih 3D primitiva. Reč je o sledećim primitivima:

- ***Sphere*** – lopta,
- ***Cube*** – kocka,
- ***Cone*** – kupa,
- ***Torus*** – torus,
- ***Dodecahedron*** – dodekaedar ili dvanaestostranični model,
- ***Tetrahedron*** – tetraedar,
- ***Octahedron*** – oktaedar,
- ***Icosahedron*** – ikosaedar ili dvadesetostranični model, i
- ***Teapot*** – čajnik.

Da bi se kreirali primitivi, trebalo bi razmotriti nekoliko komandi »vezanih« za 3D okruženje unutar OpenGL-a.

Komanda koja definiše boje koje će se primenjivati na sintaksu ispod je:

glColor3f (R, G, B)

gde se vrednosti komponenti nalaze u granicama od 0.0 do 1.0: **R od 0.0 do 1.0**, **G od 0.0 do 1.0** i **B od 0.0 do 1.0**. Ispod sledi spisak osnovnih boja koje se koriste u OpenGL-u:

glColor3f (0.0, 0.0, 0.0) – CRNA boja (black)
glColor3f (1.0, 0.0, 0.0) – CRVENA boja (red)
glColor3f (0.0, 1.0, 0.0) – ZELENA boja (green)
glColor3f (1.0, 1.0, 0.0) – ŽUTA boja (yellow)
glColor3f (0.0, 0.0, 1.0) – PLAVA boja (blue)
glColor3f (1.0, 0.0, 1.0) – LJUBIČASTA boja (magenta)
glColor3f (0.0, 1.0, 1.0) – SVETLO PLAVA boja (cyan)
glColor3f (1.0, 1.0, 1.0) – BELA boja (white)

Komanda koja definiše TETRAEDAR u potpunosti je

glutWireTetrahedron ();

Ova komanda definiše žičani model tetraedra (četvorostraničnog modela) čiji je poluprečnik $\sqrt{3}$. Sledeća komanda

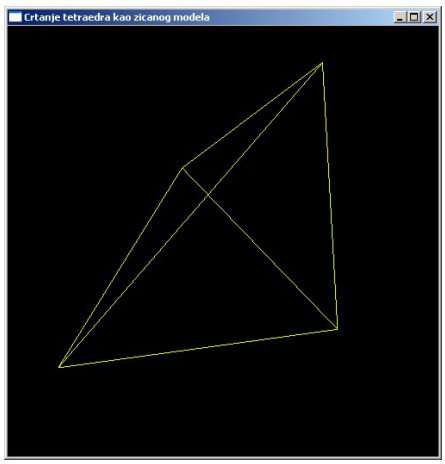
glutSolidTetrahedron ();

definiše tetraedar kao solid, a komentar je isti kao kod prethodne komande.

Evo nekoliko primera kako će izgledati tetraedar sa različitim paramaterima, a kompletna sintaksa sledi ispod. Za komandu

glutWireTetrahedron ();

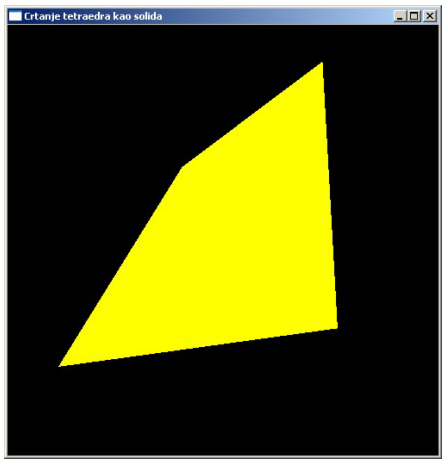
dobija se sledeća slika.



Za komandu

glutSolidTetrahedron ();

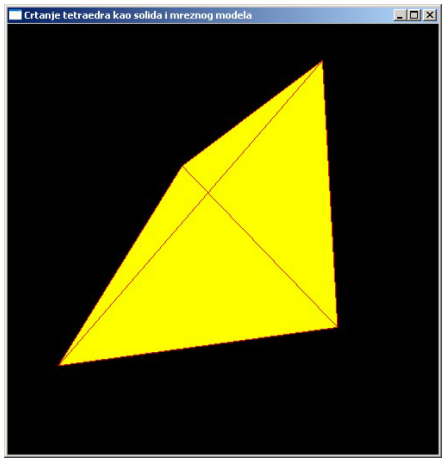
dobija se sledeća slika.



Postoji varijanta i da se iskombinuju i žičani i osenčeni model dodekaedra. Moguća sintaksa za jedan od načina je:

glutSolidTetrahedron ();
glColor3f (1.0, 0.0, 0.0);
glutWireTetrahedron ();

a rezultat je prikazan na donjoj slici.

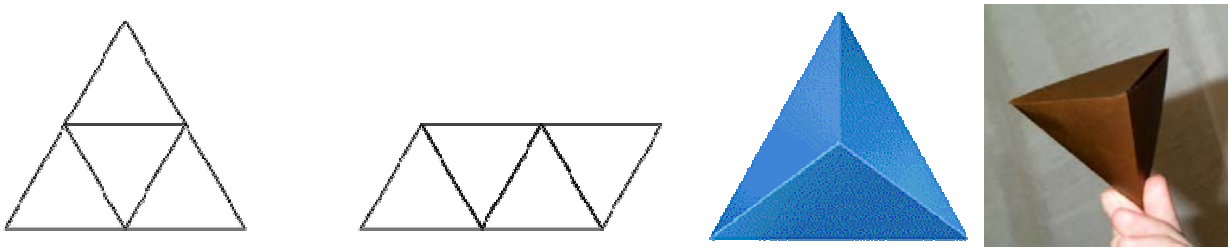


Celokupna sintaksa za gornju sliku je:

```
#include <GL/glut.h>

void init(void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */

glClearColor (0.0, 0.0, 0.0, 0.0);

/* Model sencenja - FLAT ili SMOOTH */
glShadeModel (GL_FLAT);
}

void display(void)
{

/* brise sve piksele iz bafera  */
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 0.0);

/* "Praznjenje" matrice */
glLoadIdentity ();

/* Transformacija pogleda */
gluLookAt (2.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

/* Transformacija modela - skaliranje */
glScalef (3.0, 3.0, 3.0);

/* Crtanje 3D entiteta */
glutSolidTetrahedron ();
glColor3f (1.0, 0.0, 0.0);
glutWireTetrahedron ();

/* Izvrsava GL komande u konacnom vremenu */
glFlush ();
}

void reshape (int w, int h)
{

/* Definisu se vrednosti pogleda */
glMatrixMode (GL_PROJECTION);
glLoadIdentity ();

/* Definise projekciju u perspektivi */
glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
glMatrixMode (GL_MODELVIEW);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
```

```
 * u naslovu prozora. Pozivaju se inicijalne rutine.
 * Registruje se Callback funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)

{
glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
glutInitWindowPosition (300, 100);
glutCreateWindow ("Crtanje tetraedra kao solida i mreznog modela");
init ();
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutMainLoop();
return 0;
}
```

Na narednoj slici (levi deo) prikazana je mreža tetraedra kako bi se željeni model napravio, a na desnoj strani te slike prikazan je "sklopljen" model.



Komanda koja definiše OKTAEDAR u potpunosti je

glutWireOctahedron ();

Ova komanda definiše žičani model oktaedra (osmostraničnog modela) čiji je poluprečnik 1. Sledeća komanda

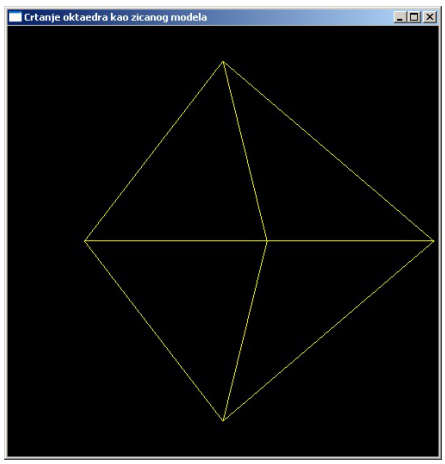
glutSolidOctahedron ();

definiše oktaedar kao solid, a komentar je isti kao kod prethodne komande.

Evo nekoliko primera kako će izgledati oktaedar sa različitim paramaterima, a kompletna sintaksa sledi ispod. Za komandu

glutWireOctahedron ();

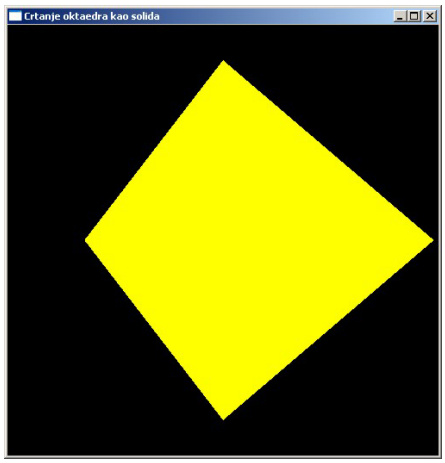
dobija se sledeća slika.



Za komandu

glutSolidOctahedron ();

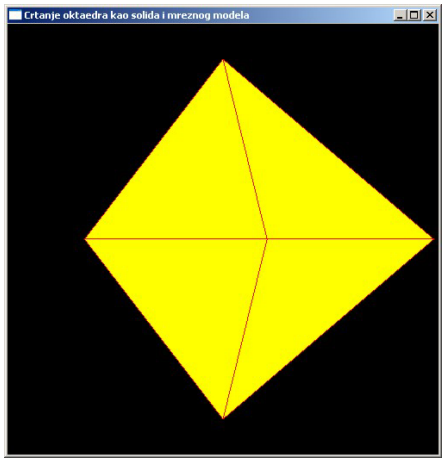
dobija se sledeća slika.



Postoji varijanta i da se iskombinuju i žičani i osenčeni model oktaedra. Moguća sintaksa za jedan od načina je:

glutSolidOctahedron ();
glColor3f (1.0, 0.0, 0.0);
glutWireOctahedron ();

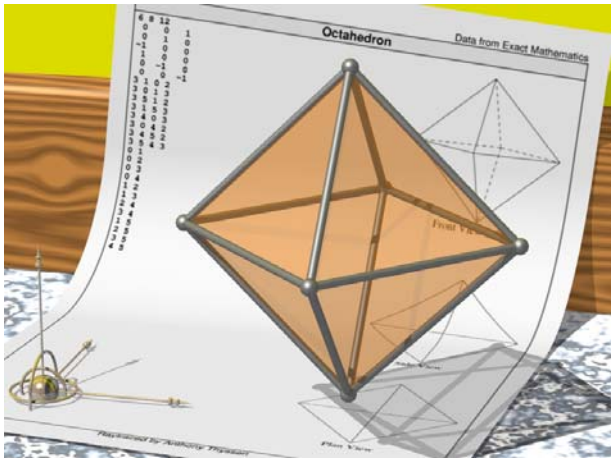
a rezultat je prikazan na donjoj slici.

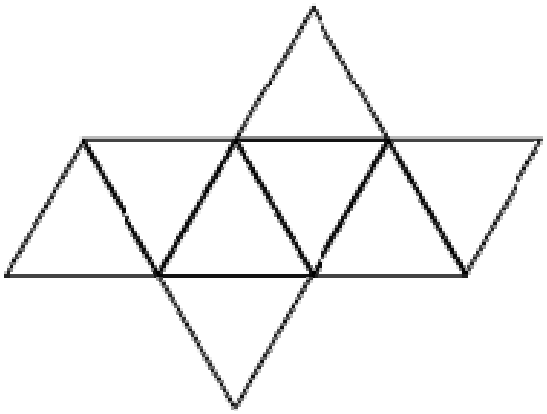


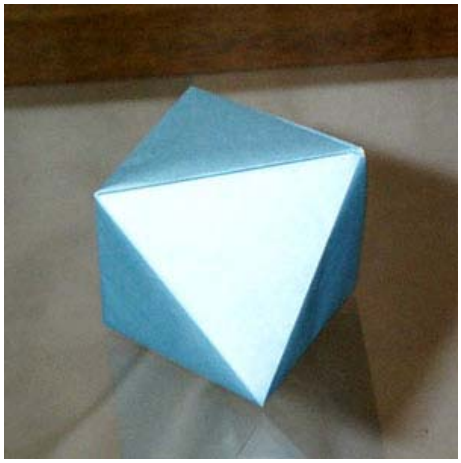
Celokupna sintaksa za gornju sliku je:

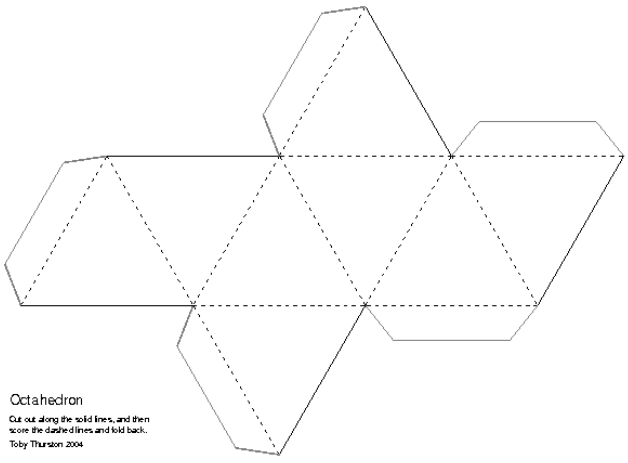
```
#include <GL/glut.h>

void init(void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
glClearColor (0.0, 0.0, 0.0, 0.0);

/* Model sencenja - FLAT ili SMOOTH */
glShadeModel (GL_FLAT);
}

void display(void)
{
/* brise sve piksele iz bafera  */
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 0.0);

/* "Praznjenje" matrice */
glLoadIdentity ();

/* Transformacija pogleda */
gluLookAt (2.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

/* Transformacija modela - skaliranje */
glScalef (3.0, 3.0, 3.0);

/* Crtanje 3D entiteta */
glutSolidOctahedron ();
glColor3f (1.0, 0.0, 0.0);
glutWireOctahedron ();

/* Izvrsava GL komande u konacnom vremenu */
glFlush ();
}

void reshape (int w, int h)
{

/* Definisu se vrednosti pogleda */
glMatrixMode (GL_PROJECTION);
glLoadIdentity ();

/* Definise projekciju u perspektivi */
glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
glMatrixMode (GL_MODELVIEW);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * u naslovu prozora. Pozivaju se inicijalne rutine.
 * Registruje se Callback funkcija za prikaz grafike.
```

```
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)
{
glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
glutInitWindowPosition (300, 100);
glutCreateWindow ("Crtanje oktaedra kao solida i mreznog modela");
init ();
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutMainLoop();
return 0;
}
```









Korisniku se ostavlja da sam bira boje, kao i put kojim će da realizuje 3D primitive. U sledećem tekstu će se nastaviti sa obradom 3D primitiva.

OpenGL (18)

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazani programi koji omogućavaju crtanje određenih 3D primitiva. Reč je o sledećim primitivima:

- ***Sphere*** – lopta,
- ***Cube*** – kocka,
- ***Cone*** – kupa,
- ***Torus*** – torus,
- ***Dodecahedron*** – dodekaedar ili dvanaestostranični model,
- ***Tetrahedron*** – tetraedar,
- ***Octahedron*** – oktaedar,
- ***Icosahedron*** – ikosaedar ili dvadesetostranični model, i
- ***Teapot*** – čajnik.

Da bi se kreirali primitivi, trebalo bi razmotriti nekoliko komandi »vezanih« za 3D okruženje unutar OpenGL-a.

Komanda koja definiše boje koje će se primenjivati na sintaksu ispod je:

glColor3f (R, G, B)

gde se vrednosti komponenti nalaze u granicama od 0.0 do 1.0: **R od 0.0 do 1.0**, **G od 0.0 do 1.0** i **B od 0.0 do 1.0**. Ispod sledi spisak osnovnih boja koje se koriste u OpenGL-u:

glColor3f (0.0, 0.0, 0.0) – CRNA boja (black)
glColor3f (1.0, 0.0, 0.0) – CRVENA boja (red)
glColor3f (0.0, 1.0, 0.0) – ZELENA boja (green)
glColor3f (1.0, 1.0, 0.0) – ŽUTA boja (yellow)
glColor3f (0.0, 0.0, 1.0) – PLAVA boja (blue)
glColor3f (1.0, 0.0, 1.0) – LJUBIČASTA boja (magenta)
glColor3f (0.0, 1.0, 1.0) – SVETLO PLAVA boja (cyan)
glColor3f (1.0, 1.0, 1.0) – BELA boja (white)

Komanda koja definiše IKOSAEDAR u potpunosti je

glutWireIcosahedron ();

Ova komanda definiše žičani model ikosaedra (dvanaestostraničnog modela) čiji je poluprečnik 1. Sledeća komanda

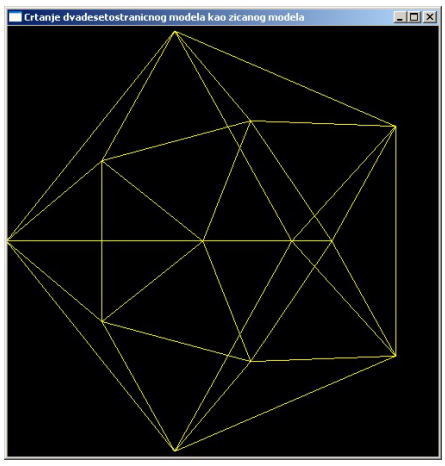
glutSolidIcosahedron ();

definiše ikosaedar kao solid, a komentar je isti kao kod prethodne komande.

Evo nekoliko primera kako će izgledati ikosaedar sa različitim paramaterima, a kompletna sintaksa sledi ispod. Za komandu

glutWireIcosahedron ();

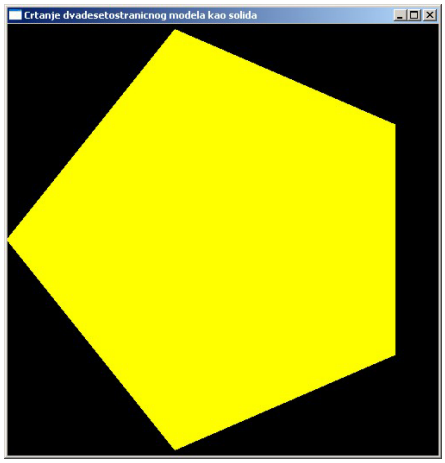
dobija se sledeća slika.



Za komandu

glutSolidIcosahedron ();

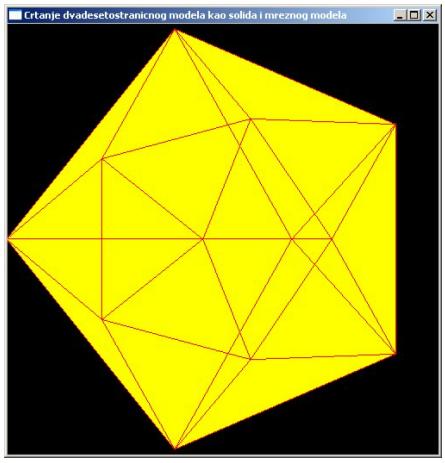
dobija se sledeća slika.



Postoji varijanta i da se iskombinuju i žičani i osenčeni model dodekaedra. Moguća sintaksa za jedan od načina je:

glutSolidIcosahedron ();
glColor3f (1.0, 0.0, 0.0);
glutWireIcosahedron ();

a rezultat je prikazan na donjoj slici.

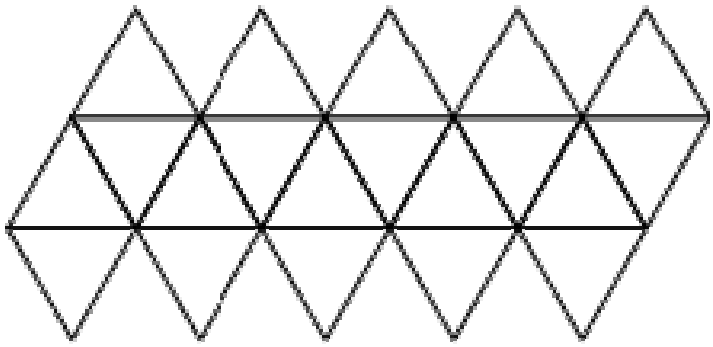


Celokupna sintaksa za gornju sliku je:

```
#include <GL/glut.h>
void init(void)

{
/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
glClearColor (0.0, 0.0, 0.0, 0.0);

/* Model sencenja - FLAT ili SMOOTH */
glShadeModel (GL_FLAT);
}

void display(void)

{
/* brise sve piksele iz bafera  */
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 0.0);

/* "Praznjenje" matrice */
glLoadIdentity ();

/* Transformacija pogleda */
gluLookAt (2.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

/* Transformacija modela - skaliranje */
glScalef (3.0, 3.0, 3.0);

/* Crtanje 3D entiteta */
glutSolidIcosahedron ();
glColor3f (1.0, 0.0, 0.0);
glutWireIcosahedron ();

/* Izvrsava GL komande u konacnom vremenu */
glFlush ();
}

void reshape (int w, int h)
{

/* Definisu se vrednosti pogleda */
glMatrixMode (GL_PROJECTION);
glLoadIdentity ();

/* Definise projekciju u perspektivi */
glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
glMatrixMode (GL_MODELVIEW);
}

/* Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * u naslovu prozora. Pozivaju se inicijalne rutine.
 * Registruje se Callback funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
```

```
int main(int argc, char** argv)

{
glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
glutInitWindowPosition (300, 100);
glutCreateWindow ("Crtanje dvadesetostranicnog modela kao solida i mreznog modela");
init ();
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutMainLoop();
return 0;
}
```

Na narednoj slici (levi deo) prikazana je mreža tetraedra kako bi se željeni model napravio, a na desnoj strani te slike prikazan je "sklopljen" model.



Komanda koja definiše ČAJNIK u potpunosti je

glutWireTeapot (I);

Ova komanda definiše žičani model oktaedra (osmostraničnog modela) čija je veličina definisana vrednošću parametra I. Sledeća komanda

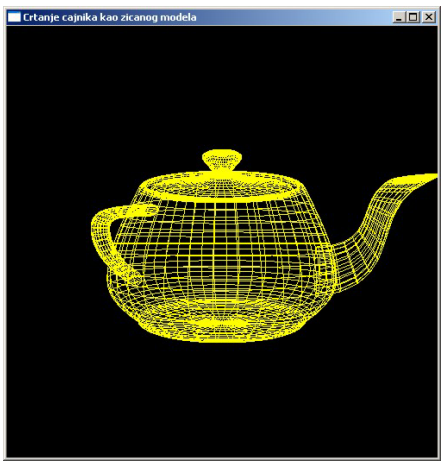
glutSolidTeapot (I);

definiše čajnik kao solid, a komentar je isti kao kod prethodne komande.

Evo nekoliko primera kako će izgledati čajnik sa različitim paramaterima, a kompletna sintaksa sledi ispod. Za komandu

glutWireTeapot (1.0);

dobija se sledeća slika.



Za komandu

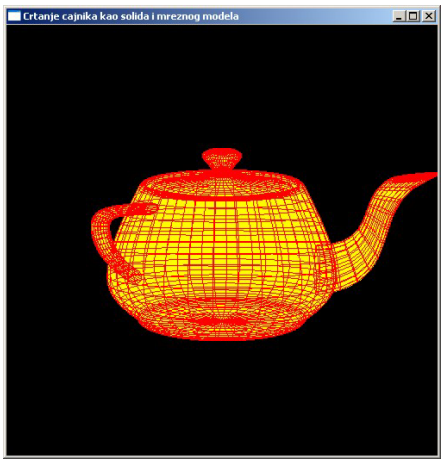
glutSolidTeapot (1.0);

dobija se sledeća slika.

Postoji varijanta i da se iskombinuju i žičani i osenčeni model oktaedra. Moguća sintaksa za jedan od načina je:

glutSolidTeapot (1.0);
glColor3f (1.0, 0.0, 0.0);
glutWireTeapot (1.0);

a rezultat je prikazan na donjoj slici.



Celokupna sintaksa za gornju sliku je:

```
#include <GL/glut.h>
void init(void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
*/ Kada su prve tri vrednosti iste onda je rec o sivoj boji.

glClearColor (0.0, 0.0, 0.0, 0.0);

/* Model sencenja - FLAT ili SMOOTH */
glShadeModel (GL_FLAT);
}
```

```
void display(void)
{

/* brise sve piksele iz bafera  */
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 0.0);

/* "Praznjenje" matrice */
glLoadIdentity ();

/* Transformacija pogleda */
gluLookAt (2.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

/* Transformacija modela - skaliranje */
glScalef (1.8, 1.8, 1.8);

/* Crtanje 3D entiteta */
glutSolidTeapot (1.0);
glColor3f (1.0, 0.0, 0.0);
glutWireTeapot (1.0);

/* Izvrsava GL komande u konacnom vremenu */
glFlush ();
}
void reshape (int w, int h)
{

/* Definisu se vrednosti pogleda */
glMatrixMode (GL_PROJECTION);
glLoadIdentity ();

/* Definise projekciju u perspektivi */
glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
glMatrixMode (GL_MODELVIEW);
}

/* Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * u naslovu prozora. Pozivaju se inicijalne rutine.
 * Registruje se Callback funkcija za prikaz grafike.
*/ Startuje se program i sam proces.
int main(int argc, char** argv)
{
glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
glutInitWindowPosition (300, 100);
glutCreateWindow ("Crtanje cajnika kao solida i mreznog modela");
init ();
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutMainLoop();
return 0;
}
```

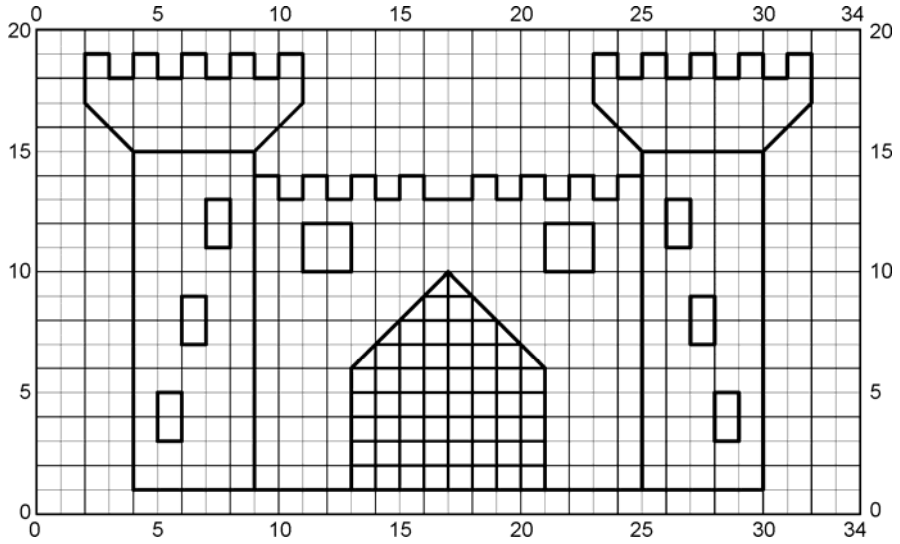
Korisniku se ostavlja da sam bira boje, kao i put kojim će da realizuje 3D primitive.

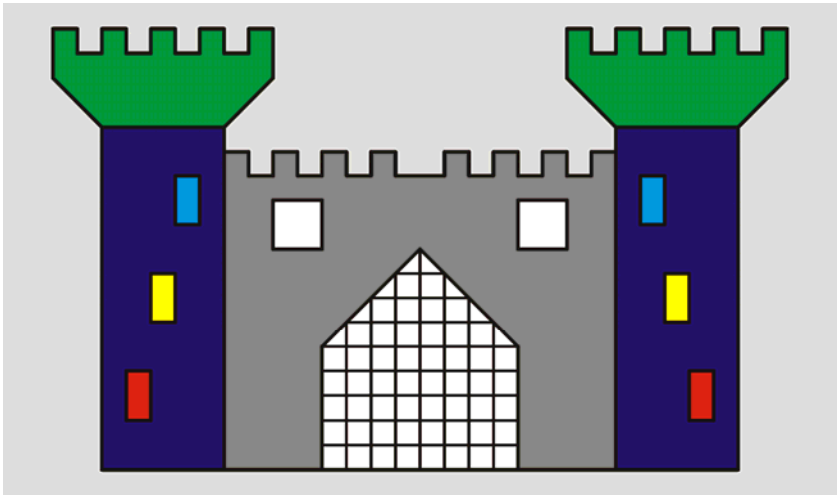
OpenGL (19)

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazani programi koji omogućavaju iscrtavanje dečijih crteža zamka i rođendanske torte. Za oba crteža prikazana je geometrija u obliku crteža i prateće mreže, a prikazana je i zahtevana „raspodela“ boja koju treba isprogramirati.

Prvi program omogućava iscrtavanje dečijeg crteža zamka. Na sledećim slikama prikazane su geometrija gde korisnik može da pronađe željene koordinate, kao i orijentaciona preraspodela boja.





Poligoni za kreiranje dečijeg crteža zamka

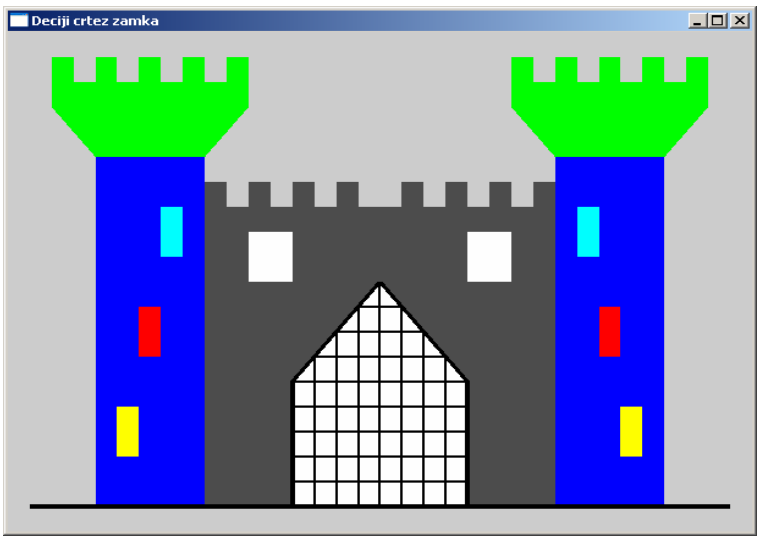
```
/* Crtanje zamka */

#include <GL/glut.h>

void display(void)
{

/* brise sve piksele iz bafera */
   glClear (GL_COLOR_BUFFER_BIT);
```

```
/* Crtaju se tamnoplave kule */
      glColor3f (0.0, 0.0, 1.0);
      glRectf (4.0, 1.0, 9.0, 15.0);
      glRectf (25.0, 1.0, 30.0, 15.0);

/* Crta se levi zeleni vrh kule */
      glColor3f (0.0, 1.0, 0.0);
      glRectf (2.0, 18.0, 3.0, 19.0);
      glRectf (4.0, 18.0, 5.0, 19.0);
      glRectf (6.0, 18.0, 7.0, 19.0);
      glRectf (8.0, 18.0, 9.0, 19.0);
      glRectf (10.0, 18.0, 11.0, 19.0);
      glRectf (2.0, 17.0, 11.0, 18.0);

      glBegin(GL_POLYGON);
      glVertex2f (2.0, 17.0);
      glVertex2f (4.0, 15.0);
      glVertex2f (9.0, 15.0);
      glVertex2f (11.0, 17.0);
      glEnd();

/* Crta se desni zeleni vrh kule */
      glColor3f (0.0, 1.0, 0.0);
      glRectf (23.0, 18.0, 24.0, 19.0);
      glRectf (25.0, 18.0, 26.0, 19.0);
      glRectf (27.0, 18.0, 28.0, 19.0);
      glRectf (29.0, 18.0, 30.0, 19.0);
      glRectf (31.0, 18.0, 32.0, 19.0);
      glRectf (23.0, 17.0, 32.0, 18.0);

      glBegin(GL_POLYGON);
      glVertex2f (23.0, 17.0);
      glVertex2f (25.0, 15.0);
      glVertex2f (30.0, 15.0);
      glVertex2f (32.0, 17.0);
      glEnd();

/* Crta se srednji deo zamka */
      glColor3f (0.3, 0.3, 0.3);
      glRectf (9.0, 1.0, 25.0, 13.0);
      glRectf (9.0, 13.0, 10.0, 14.0);
      glRectf (11.0, 13.0, 12.0, 14.0);
      glRectf (13.0, 13.0, 14.0, 14.0);
      glRectf (15.0, 13.0, 16.0, 14.0);
      glRectf (18.0, 13.0, 19.0, 14.0);
      glRectf (20.0, 13.0, 21.0, 14.0);
      glRectf (22.0, 13.0, 23.0, 14.0);
      glRectf (24.0, 13.0, 25.0, 14.0);

/* Crtaju se beli prozori zamka */
      glColor3f (1.0, 1.0, 1.0);
      glRectf (11.0, 10.0, 13.0, 12.0);
      glRectf (21.0, 10.0, 23.0, 12.0);

      glBegin(GL_POLYGON);
      glVertex2f (13.0, 1.0);
      glVertex2f (21.0, 1.0);
      glVertex2f (21.0, 6.0);
```

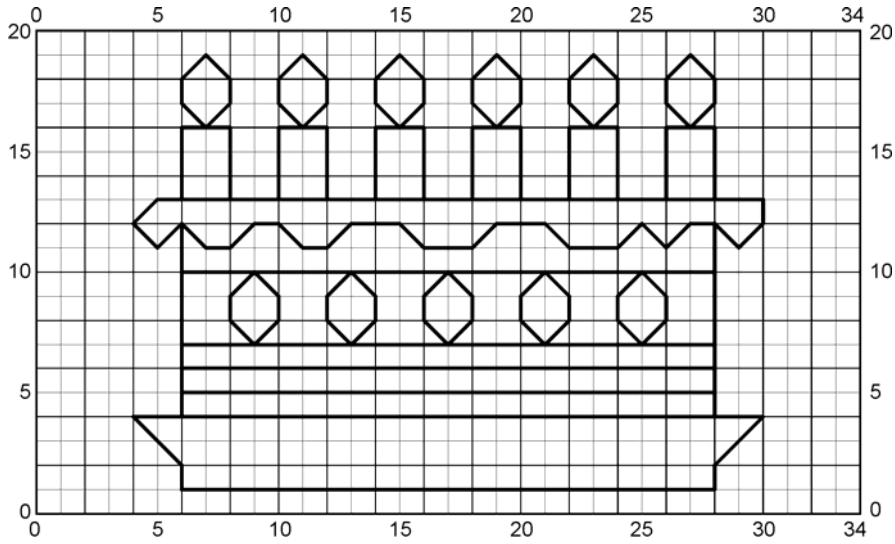
```
      glVertex2f (17.0, 10.0);
      glVertex2f (13.0, 6.0);
      glEnd();

/* Crtaju se svetloplavi prozori kula */
      glColor3f (0.0, 1.0, 1.0);
      glRectf (7.0, 11.0, 8.0, 13.0);
      glRectf (26.0, 11.0, 27.0, 13.0);

/* Crtaju se crveni prozori kula */
      glColor3f (1.0, 0.0, 0.0);
      glRectf (6.0, 7.0, 7.0, 9.0);
      glRectf (27.0, 7.0, 28.0, 9.0);

/* Crtaju se zuti prozori kula */
      glColor3f (1.0, 1.0, 0.0);
      glRectf (5.0, 3.0, 6.0, 5.0);
      glRectf (28.0, 3.0, 29.0, 5.0);

/* Crta se kapije kontura sirine 4 i crne boje */
    glLineWidth (4.0);
    glBegin(GL_LINES);
    glColor3f (0.0, 0.0, 0.0);
       glVertex2f (33.0, 1.0);
       glVertex2f (1.0, 1.0);
    glEnd();

    glBegin(GL_LINE_STRIP);
    glColor3f (0.0, 0.0, 0.0);
       glVertex2f (13.0, 1.0);
       glVertex2f (13.0, 6.0);
       glVertex2f (17.0, 10.0);
       glVertex2f (21.0, 6.0);
       glVertex2f (21.0, 1.0);
    glEnd();

/* Crtaju se resetke sirine 2 i crne boje */
    glLineWidth (2.0);
    glBegin(GL_LINES);
    glColor3f (0.0, 0.0, 0.0);
       glVertex2f (14.0, 1.0);
       glVertex2f (14.0, 7.0);
       glVertex2f (15.0, 1.0);
       glVertex2f (15.0, 8.0);
       glVertex2f (16.0, 1.0);
       glVertex2f (16.0, 9.0);
       glVertex2f (17.0, 1.0);
       glVertex2f (17.0, 10.0);
       glVertex2f (18.0, 1.0);
       glVertex2f (18.0, 9.0);
       glVertex2f (19.0, 1.0);
       glVertex2f (19.0, 8.0);
       glVertex2f (20.0, 1.0);
       glVertex2f (20.0, 7.0);
       glVertex2f (13.0, 2.0);
       glVertex2f (21.0, 2.0);
       glVertex2f (13.0, 3.0);
       glVertex2f (21.0, 3.0);
```

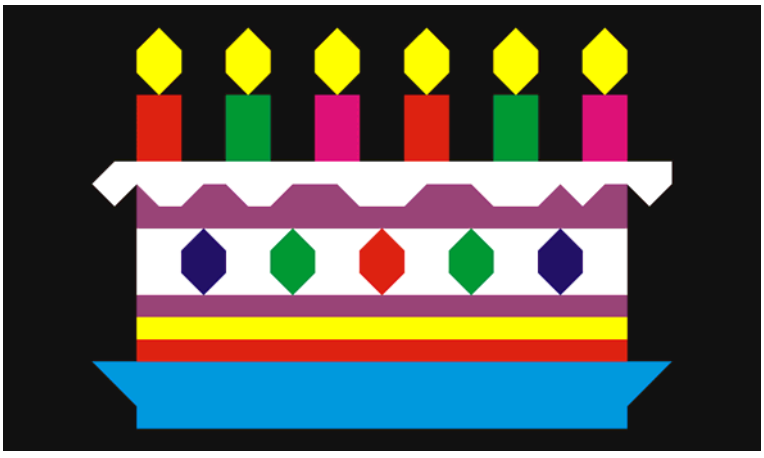
```
      glVertex2f (13.0, 4.0);
      glVertex2f (21.0, 4.0);
      glVertex2f (13.0, 5.0);
      glVertex2f (21.0, 5.0);
      glVertex2f (13.0, 6.0);
      glVertex2f (21.0, 6.0);
      glVertex2f (14.0, 7.0);
      glVertex2f (20.0, 7.0);
      glVertex2f (15.0, 8.0);
      glVertex2f (19.0, 8.0);
      glVertex2f (16.0, 9.0);
      glVertex2f (18.0, 9.0);
   glEnd();

/* Izvrsava GL komande u konacnom vremenu */
   glFlush ();
}

void init (void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
   glClearColor (0.8, 0.8, 0.8, 0.0);

/* Definisu se vrednosti pogleda */
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(0.0, 34.0, 0.0, 20.0, -1.0, 1.0);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * "Deciji crtez zamka" u naslovu prozora. Pozivaju se inicijalne rutine.
* Registruje se Callback  funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (440, 440);
   glutInitWindowPosition (400, 200);
   glutCreateWindow ("Deciji crtez zamka");
   init ();
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
/* ANSI C zahteva odgovor kroz pritisak bilo kog tastera. */
}
```

Kada se ovaj program startuje, posle kompajliranja i linkovanja, dobija se kao izlaz prozor koji je prikazan na donjoj slici.



Drugi program omogućava iscrtavanje dečijeg crteža rođendanske torte. Na sledećim slikama prikazane su geometrija gde korisnik može da pronađe željene koordinate, kao i orijentaciona preraspodela boja.





Poligoni za kreiranje dečijeg crteža rođendanske torte

```
/* Crtanje torte  */

#include <GL/glut.h>
```

```
void display(void)
{

/* brise sve piksele iz bafera  */
   glClear (GL_COLOR_BUFFER_BIT);

/* Crta se tanjir torte */
      glColor3f (0.0, 1.0, 1.0);
      glBegin(GL_TRIANGLES);
            glVertex2f (6.0, 2.0);
            glVertex2f (6.0, 4.0);
            glVertex2f (4.0, 4.0);
      glEnd();
      glBegin(GL_TRIANGLES);
            glVertex2f (28.0, 2.0);
            glVertex2f (30.0, 4.0);
            glVertex2f (28.0, 4.0);
      glEnd();

      glRectf (6.0, 1.0, 28.0, 4.0);

/* Crta se crna linija na tanjiru */
      glColor3f (0.0, 0.0, 0.0);
      glLineWidth(3);
      glBegin(GL_LINES);
            glVertex2f (6.0, 1.9);
            glVertex2f (28.0, 1.9);
      glEnd();

/* Crtaju se slojevi torte */
      glColor3f (1.0, 0.0, 0.0);
      glRectf (6.0, 4.0, 28.0, 5.0);
      glColor3f (1.0, 1.0, 0.0);
      glRectf (6.0, 5.0, 28.0, 6.0);
      glColor3f (1.0, 0.0, 1.0);
      glRectf (6.0, 6.0, 28.0, 7.0);
      glColor3f (1.0, 1.0, 1.0);
      glRectf (6.0, 7.0, 28.0, 10.0);
      glColor3f (1.0, 0.0, 1.0);
      glRectf (6.0, 10.0, 28.0, 12.0);

/* Definisanje slaga torte */
      glColor3f (1.0, 1.0, 1.0);
      glRectf (5.0, 12.0, 30.0, 13.0);

      glBegin(GL_TRIANGLES);
            glVertex2f (5.0, 13.0);
            glVertex2f (4.0, 12.0);
            glVertex2f (5.0, 12.0);
      glEnd();

      glBegin(GL_TRIANGLES);
            glVertex2f (4.0, 12.0);
            glVertex2f (5.0, 12.0);
            glVertex2f (5.0, 11.0);
      glEnd();

      glBegin(GL_TRIANGLES);
```

```
            glVertex2f (5.0, 12.0);
            glVertex2f (5.0, 11.0);
            glVertex2f (6.0, 12.0);
      glEnd();

      glBegin(GL_TRIANGLES);
            glVertex2f (25.0, 12.0);
            glVertex2f (26.0, 11.0);
            glVertex2f (27.0, 12.0);
      glEnd();

      glBegin(GL_TRIANGLES);
            glVertex2f (28.0, 12.0);
            glVertex2f (29.0, 11.0);
            glVertex2f (30.0, 12.0);
      glEnd();

      glBegin(GL_POLYGON);
            glVertex2f (6.0, 12.0);
            glVertex2f (7.0, 11.0);
            glVertex2f (8.0, 11.0);
            glVertex2f (9.0, 12.0);
      glEnd();

      glBegin(GL_POLYGON);
            glVertex2f (10.0, 12.0);
            glVertex2f (11.0, 11.0);
            glVertex2f (12.0, 11.0);
            glVertex2f (13.0, 12.0);
      glEnd();

      glBegin(GL_POLYGON);
            glVertex2f (15.0, 12.0);
            glVertex2f (16.0, 11.0);
            glVertex2f (18.0, 11.0);
            glVertex2f (19.0, 12.0);
      glEnd();

      glBegin(GL_POLYGON);
            glVertex2f (21.0, 12.0);
            glVertex2f (22.0, 11.0);
            glVertex2f (24.0, 11.0);
            glVertex2f (25.0, 12.0);
      glEnd();

/* Definisanje crvenih svecica */
      glColor3f (1.0, 0.0, 0.0);
      glRectf (6.0, 13.0, 8.0, 16.0);
      glRectf (18.0, 13.0, 20.0, 16.0);

/* Definisanje zelenih svecica */
      glColor3f (0.0, 1.0, 0.0);
      glRectf (10.0, 13.0, 12.0, 16.0);
      glRectf (22.0, 13.0, 24.0, 16.0);

/* Definisanje tamnoplavih svecica */
      glColor3f (0.0, 0.0, 1.0);
      glRectf (14.0, 13.0, 16.0, 16.0);
```

```
      glRectf (26.0, 13.0, 28.0, 16.0);

/* Crtanje prvog plamena svece */
      glColor3f (1.0, 1.0, 0.0);
      glBegin(GL_POLYGON);
            glVertex2f (7.0, 16.0);
            glVertex2f (8.0, 17.0);
            glVertex2f (8.0, 18.0);
            glVertex2f (7.0, 19.0);
            glVertex2f (6.0, 18.0);
            glVertex2f (6.0, 17.0);
      glEnd();

/* Crtanje drugog plamena svece */
      glBegin(GL_POLYGON);
            glVertex2f (11.0, 16.0);
            glVertex2f (12.0, 17.0);
            glVertex2f (12.0, 18.0);
            glVertex2f (11.0, 19.0);
            glVertex2f (10.0, 18.0);
            glVertex2f (10.0, 17.0);
      glEnd();

/* Crtanje treceg plamena svece */
      glBegin(GL_POLYGON);
            glVertex2f (15.0, 16.0);
            glVertex2f (16.0, 17.0);
            glVertex2f (16.0, 18.0);
            glVertex2f (15.0, 19.0);
            glVertex2f (14.0, 18.0);
            glVertex2f (14.0, 17.0);
      glEnd();

/* Crtanje cetvrtog plamena svece */
      glBegin(GL_POLYGON);
            glVertex2f (19.0, 16.0);
            glVertex2f (20.0, 17.0);
            glVertex2f (20.0, 18.0);
            glVertex2f (19.0, 19.0);
            glVertex2f (18.0, 18.0);
            glVertex2f (18.0, 17.0);
      glEnd();

/* Crtanje petog plamena svece */
      glBegin(GL_POLYGON);
            glVertex2f (23.0, 16.0);
            glVertex2f (24.0, 17.0);
            glVertex2f (24.0, 18.0);
            glVertex2f (23.0, 19.0);
            glVertex2f (22.0, 18.0);
            glVertex2f (22.0, 17.0);
      glEnd();

/* Crtanje sestog plamena svece */
      glBegin(GL_POLYGON);
            glVertex2f (27.0, 16.0);
            glVertex2f (28.0, 17.0);
            glVertex2f (28.0, 18.0);
```

```
            glVertex2f (27.0, 19.0);
            glVertex2f (26.0, 18.0);
            glVertex2f (26.0, 17.0);
      glEnd();

/* Crtanje prvog znaka */
      glColor3f (0.0, 0.0, 1.0);
      glBegin(GL_POLYGON);
            glVertex2f (9.0, 7.0);
            glVertex2f (10.0, 8.0);
            glVertex2f (10.0, 9.0);
            glVertex2f (9.0, 10.0);
            glVertex2f (8.0, 9.0);
            glVertex2f (8.0, 8.0);
      glEnd();

/* Crtanje petog znaka */
      glBegin(GL_POLYGON);
            glVertex2f (25.0, 7.0);
            glVertex2f (26.0, 8.0);
            glVertex2f (26.0, 9.0);
            glVertex2f (25.0, 10.0);
            glVertex2f (24.0, 9.0);
            glVertex2f (24.0, 8.0);
      glEnd();

/* Crtanje drugog znaka */
      glColor3f (0.0, 1.0, 0.0);
      glBegin(GL_POLYGON);
            glVertex2f (13.0, 7.0);
            glVertex2f (14.0, 8.0);
            glVertex2f (14.0, 9.0);
            glVertex2f (13.0, 10.0);
            glVertex2f (12.0, 9.0);
            glVertex2f (12.0, 8.0);
      glEnd();

/* Crtanje cetvrtog znaka */
      glBegin(GL_POLYGON);
            glVertex2f (21.0, 7.0);
            glVertex2f (22.0, 8.0);
            glVertex2f (22.0, 9.0);
            glVertex2f (21.0, 10.0);
            glVertex2f (20.0, 9.0);
            glVertex2f (20.0, 8.0);
      glEnd();

/* Crtanje treceg znaka */
      glColor3f (1.0, 0.0, 0.0);
      glBegin(GL_POLYGON);
            glVertex2f (17.0, 7.0);
            glVertex2f (18.0, 8.0);
            glVertex2f (18.0, 9.0);
            glVertex2f (17.0, 10.0);
            glVertex2f (16.0, 9.0);
            glVertex2f (16.0, 8.0);
      glEnd();
```

```
/* Izvrsava GL komande u konacnom vremenu */
    glFlush ();
}

void init (void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
 */
    glClearColor (0.0, 0.0, 0.0, 0.0);

/* Definisu se vrednosti pogleda */
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, 34.0, 0.0, 20.0, -1.0, 1.0);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod (single buffer i RGBA).
* Otvara se prozor sa tekstom  "Deciji crtez torte" u naslovu prozora. Pozivaju se inicijalne
* rutine. Registruje se Callback funkcija za prikaz grafike. Startuje se program i sam proces.
 */
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (440, 440);
    glutInitWindowPosition (400, 200);
    glutCreateWindow ("Deciji crtez torte");
    init ();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;

/* ANSI C zahteva odgovor kroz pritisak bilo kog tastera. */
}
```

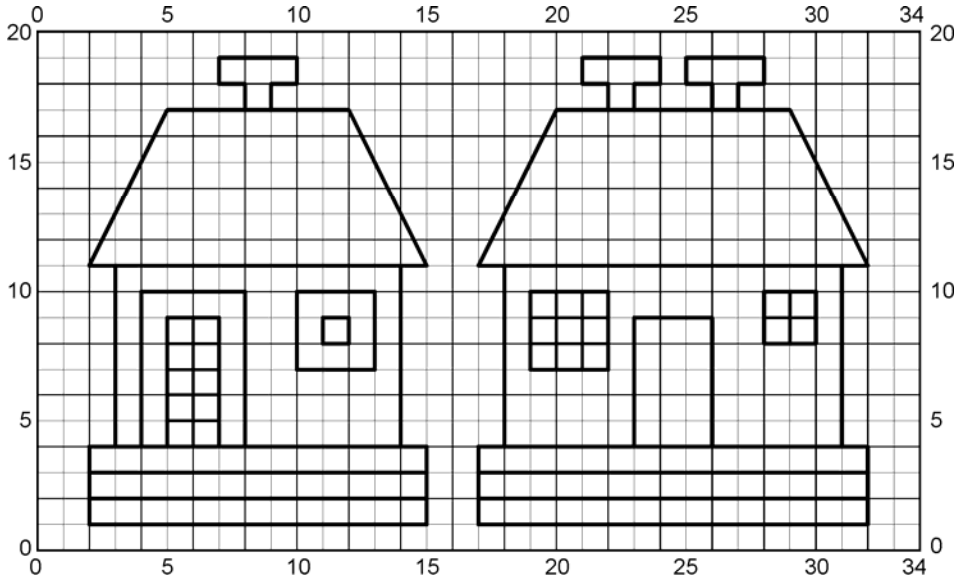
Kada se ovaj program startuje, posle kompajliranja i linkovanja, dobija se kao izlaz prozor koji je prikazan na donjoj slici.

OpenGL (20)

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazan program koji omogućava iscrtavanje dečijeg crteža dve kućice. Za crtež prikazana je geometrija u obliku crteža i prateće mreže, a prikazana je i moguća „raspodela“ boja koju treba isprogramirati.

Program omogućava iscrtavanje dečijeg crteža kućica. Na sledećim slikama prikazane su geometrija gde korisnik može da pronađe željene koordinate.



Poligoni za kreiranje dečijeg crteža kućica

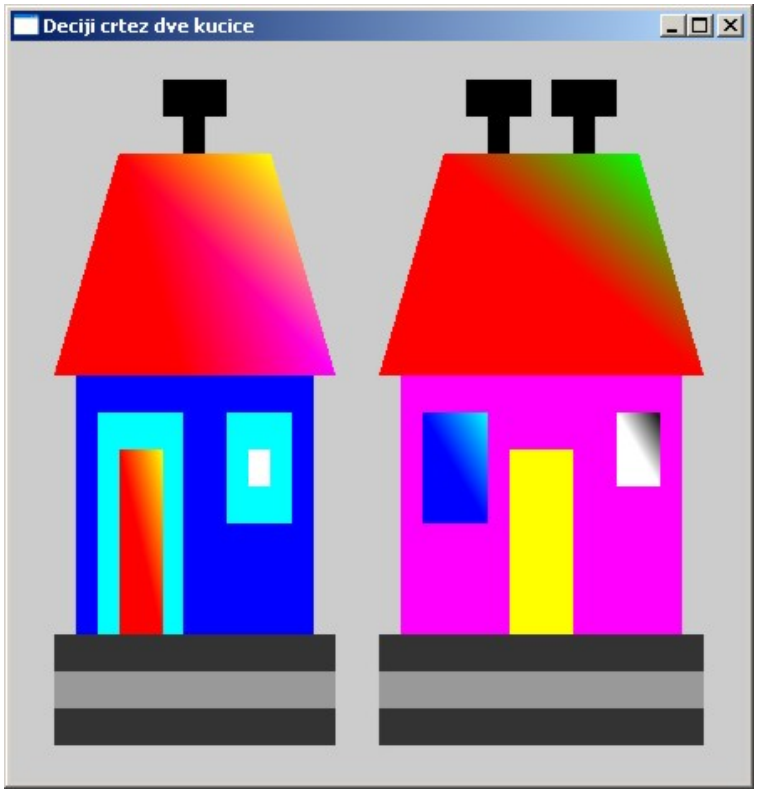
```
/*
 * Crtanje dve kucice
 */

#include <GL/glut.h>

void display(void)

{
/* brise sve piksele iz bafera  */
   glClear (GL_COLOR_BUFFER_BIT);

/* Crtaju se temelji obe kuce */
      glColor3f (0.2, 0.2, 0.2);
      glRectf (2.0, 1.0, 15.0, 4.0);
      glRectf (17.0, 1.0, 32.0, 4.0);
      glColor3f (0.6, 0.6, 0.6);
      glRectf (2.0, 2.0, 15.0, 3.0);
      glRectf (17.0, 2.0, 32.0, 3.0);

/* Crta se zid leve kuce */
      glColor3f (0.0, 0.0, 1.0);
      glRectf (3.0, 4.0, 14.0, 11.0);

/* Definisanje vrata i prozora leve kuce */
```

```
      glColor3f (0.0, 1.0, 1.0);
      glRectf (4.0, 4.0, 8.0, 10.0);
      glRectf (10.0, 7.0, 13.0, 10.0);
      glColor3f (1.0, 1.0, 1.0);
      glRectf (11.0, 8.0, 12.0, 9.0);

/* Definisanje vrata sa nijansama boja */
      glBegin(GL_TRIANGLE_FAN);

/* Crta se crveni trougao */
   glColor3f (1.0, 0.0, 0.0);
      glVertex2f (5.0, 9.0);
      glVertex2f (5.0, 4.0);
      glVertex2f (7.0, 4.0);

/* Crta se zuti trougao */
   glColor3f (1.0, 1.0, 0.0);
      glVertex2f (7.0, 9.0);

   glEnd();

/* Definisanje krova leve kuce */
      glBegin(GL_TRIANGLE_FAN);

/* Crta se crveni trougao */
   glColor3f (1.0, 0.0, 0.0);
      glVertex2f (5.0, 17.0);
      glVertex2f (2.0, 11.0);
      glVertex2f (8.0, 11.0);

/* Crta se ljubicasti trougao */
   glColor3f (1.0, 0.0, 1.0);
      glVertex2f (15.0, 11.0);

/* Crta se zuti trougao */
   glColor3f (1.0, 1.0, 0.0);
      glVertex2f (12.0, 17.0);

   glEnd();


/* Crta se dimnjak leve kuce */
      glColor3f (0.0, 0.0, 0.0);
      glRectf (8.0, 17.0, 9.0, 18.0);
      glRectf (7.0, 18.0, 10.0, 19.0);

/* Crta se zid desne kuce */
      glColor3f (1.0, 0.0, 1.0);
      glRectf (18.0, 4.0, 31.0, 11.0);

/* Definisanje krova desne kuce */
      glBegin(GL_TRIANGLE_FAN);

/* Crta se crveni trougao */
   glColor3f (1.0, 0.0, 0.0);
      glVertex2f (20.0, 17.0);
      glVertex2f (17.0, 11.0);
      glVertex2f (32.0, 11.0);
```

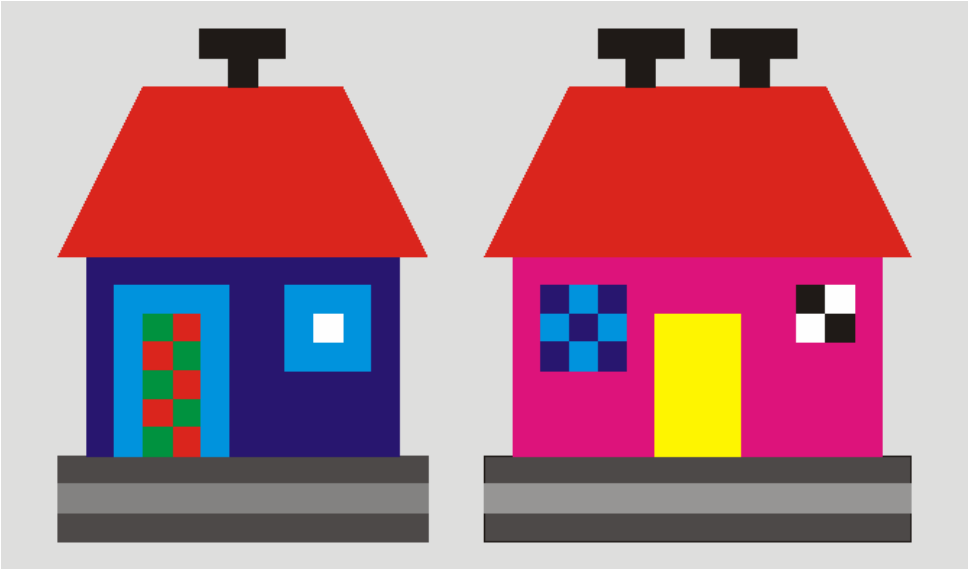
```
/* Crta se zeleni trougao */
   glColor3f (0.0, 1.0, 0.0);
      glVertex2f (29.0, 17.0);

   glEnd();

/* Crtaju se dimnjaci leve kuce */
      glColor3f (0.0, 0.0, 0.0);
      glRectf (22.0, 17.0, 23.0, 18.0);
      glRectf (21.0, 18.0, 24.0, 19.0);
      glRectf (26.0, 17.0, 27.0, 18.0);
      glRectf (25.0, 18.0, 28.0, 19.0);

/* Crtaju se vrata desne kuce */
      glColor3f (1.0, 1.0, 0.0);
      glRectf (23.0, 4.0, 26.0, 9.0);

/* Definisanje levog prozora */
      glBegin(GL_TRIANGLE_FAN);

/* Crta se tamnoplavi trougao */
   glColor3f (0.0, 0.0, 1.0);
      glVertex2f (19.0, 10.0);
      glVertex2f (19.0, 7.0);
      glVertex2f (22.0, 7.0);

/* Crta se svetloplavi trougao */
   glColor3f (0.0, 1.0, 1.0);
      glVertex2f (22.0, 10.0);

   glEnd();

/* Definisanje desnog prozora */
      glBegin(GL_TRIANGLE_FAN);

/* Crta se beli trougao */
   glColor3f (1.0, 1.0, 1.0);
      glVertex2f (28.0, 10.0);
      glVertex2f (28.0, 8.0);
      glVertex2f (30.0, 8.0);

/* Crta se crni trougao */
   glColor3f (0.0, 0.0, 0.0);
      glVertex2f (30.0, 10.0);

   glEnd();

/* Izvrsava GL komande u konacnom vremenu */
   glFlush ();
}

void init (void)
{

/* Definise RGB i Alpha vrednosti kada je bafer "ociscen" od boja.
 * Uobicajene vrednosti sa sve cetiri velicine je 0.0.
 * Kada su prve tri vrednosti iste onda je rec o sivoj boji.
```

```
 */
    glClearColor (0.8, 0.8, 0.8, 0.0);

/* Definisu se vrednosti pogleda */
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, 34.0, 0.0, 20.0, -1.0, 1.0);
}

/*
 * Deklarise se pocetna velicina prozora, i Display mod
 * (single buffer i RGBA). Otvara se prozor sa tekstom
 * "Deciji crtez dve kucice" u naslovu prozora.
 * Pozivaju se inicijalne rutine. Registruje se Callback
 * funkcija za prikaz grafike.
 * Startuje se program i sam proces.
 */
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (440, 440);
    glutInitWindowPosition (400, 200);
    glutCreateWindow ("Deciji crtez dve kucice");
    init ();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
/* ANSI C zahteva odgovor kroz pritisak bilo kog tastera. */
}
```

Kada se ovaj program startuje, posle kompajliranja i linkovanja, dobija se kao izlaz prozor koji je prikazan na donjoj slici.



Za vežbu korisnik može da iskoristi sledeću sliku kako bi napisao program koji će da zadovolji prikazanu raspodelu boja na pomenutoj slici.



OpenGL (21)

Pripremio Dragan Cvetković

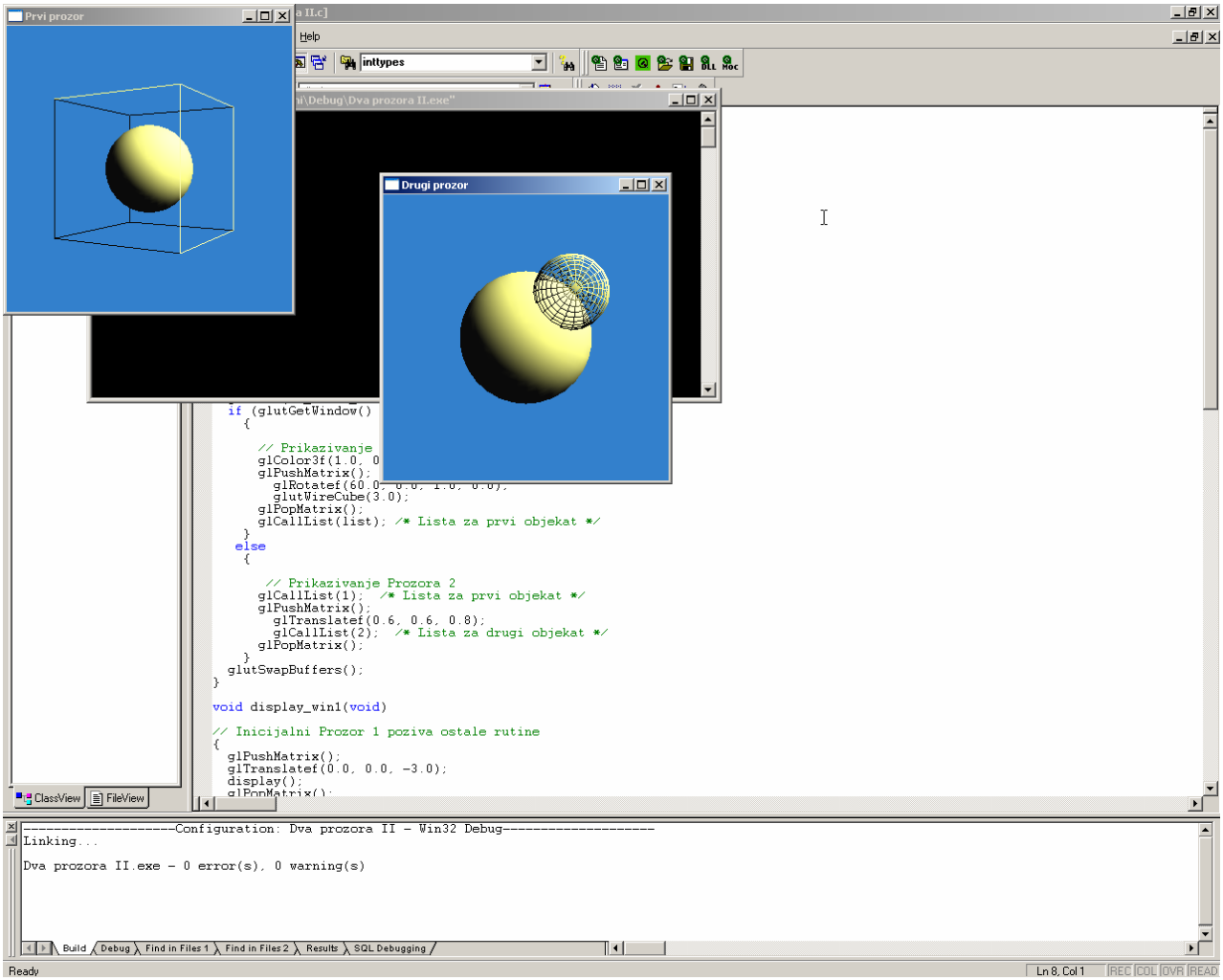
OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazan program koji omogućava kreiranje dva nezavisna prozora sa različitim sadržajem, kao i sa pratećim tekstualnim prozorom koji prikazuje koji je prozor aktivan.

```
/* Otvaranje dva prozora */

#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <math.h>
#include <GL/glut.h>

GLfloat light_diffuse[] = {2.0, 2.0, 1.0, 1.0};
GLfloat light_position[] = {5.0, 5.0, 4.0, 0.0};
GLUquadricObj *qobj;

// Promenljive prozora
int win1, win2;

// Prikazivanje liste
int list = 1;

void display(void)

// Prikazivanje rutina za oba prozora
{
  glClearColor(0.2, 0.5, 0.8, 1.0);
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  if (glutGetWindow() == win1)
    {

      // Prikazivanje Prozora 1 i kocke
      glColor3f(1.0, 0.0, 0.0);
      glPushMatrix();
        glRotatef(60.0, 0.0, 1.0, 0.0);
        glutWireCube(3.0);
      glPopMatrix();
      glCallList(list); /* Lista za prvi objekat */
    }
   else
    {

       // Prikazivanje Prozora 2
   glCallList(1);  /* Lista za prvi objekat */
      glPushMatrix();
        glTranslatef(0.6, 0.6, 0.8);
    glCallList(2); /* Lista za drugi objekat */
      glPopMatrix();
    }
  glutSwapBuffers();
}

void display_win1(void)
```

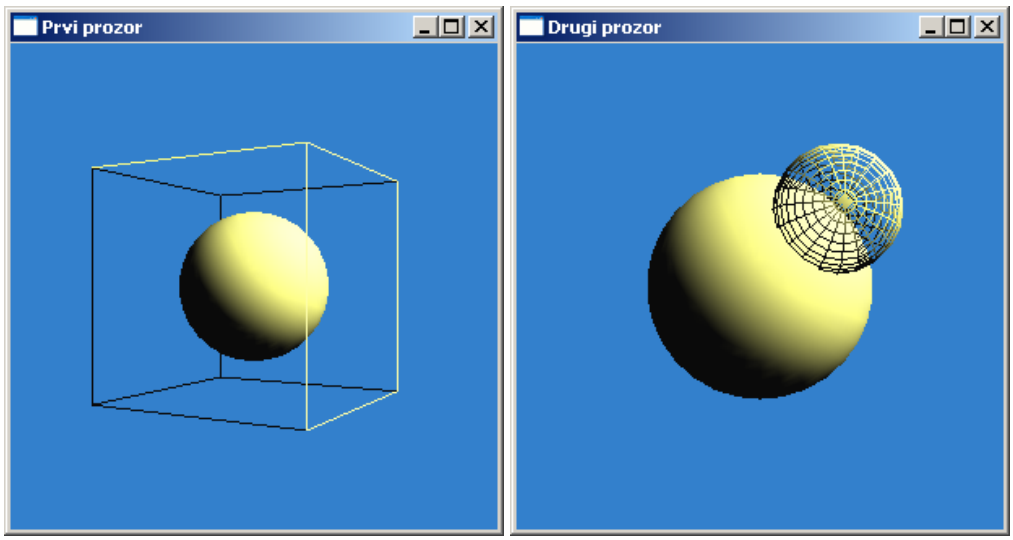
```
// Inicijalni Prozor 1 poziva ostale rutine
{
  glPushMatrix();
  glTranslatef(0.0, 0.0, -3.0);
  display();
  glPopMatrix();
}

// Ova funkcija govori koji je prozor aktivan, a koji nije!
void which_window(int state)
{
  printf("%s  Prozor %d \n",
    state == GLUT_LEFT ? "Izlazak" : "Ulazak",
    glutGetWindow());
}

// Jednostavna funkcija misa koja prikazuje
// tekucu koordinatu misa unutar aktivnog prozora!
void mouse(int button, int state, int x, int y)
{
  printf("button: %d %s %d,%d\n", button, state
       == GLUT_UP ? "UP" : "down", x, y);
}

void init(void)
{
 // Definise loptu kao solid
  gluQuadricDrawStyle(qobj, GLU_FILL);
  glNewList(1, GL_COMPILE);  /* Lista za prvi objekat */
    gluSphere(qobj,  1.0,  30,  30);
  glEndList();

 // Definise loptu kao zicani model
  gluQuadricDrawStyle(qobj, GLU_LINE);
 glNewList(2, GL_COMPILE);  /* Lista za drugi objekat */
    gluSphere(qobj,  0.5,  20,  20);
  glEndList();

  glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
  glLightfv(GL_LIGHT0, GL_POSITION, light_position);
  glEnable(GL_LIGHTING);
  glEnable(GL_LIGHT0);
  glEnable(GL_DEPTH_TEST);

  glMatrixMode(GL_PROJECTION);
  gluPerspective( 40.0, 1.0, 1.0,  20.0);
  glMatrixMode(GL_MODELVIEW);
  gluLookAt(0.0, 0.0, 5.0,  0.0, 0.0, 0.0,  0.0, 1.0, 0.);
  glTranslatef(0.0, 0.0, -1.0);
}

int main(int argc, char **argv)
{
  qobj = gluNewQuadric();
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
```

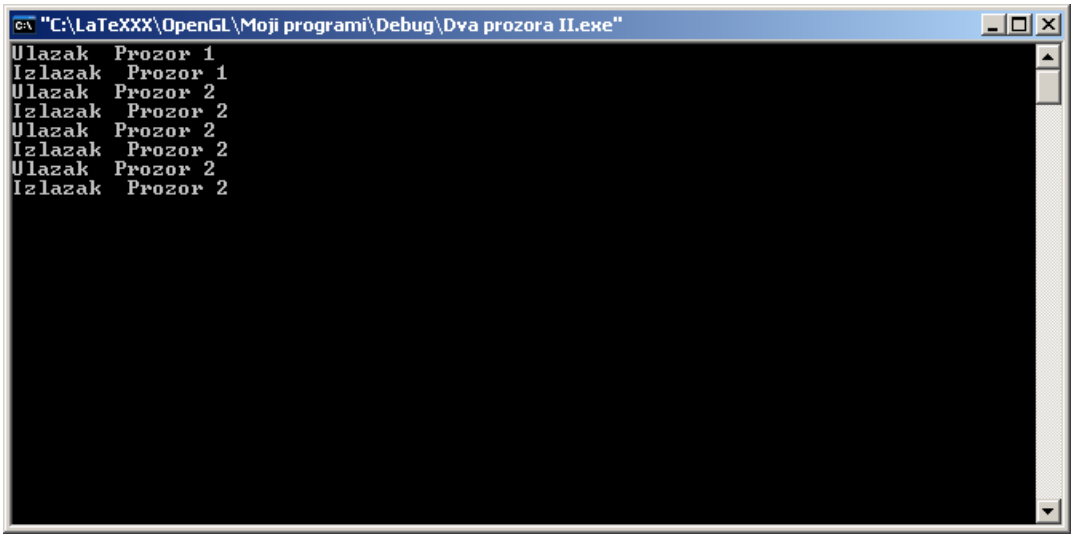
```
// Kreira Prozor 1
   win1 = glutCreateWindow("Prvi prozor");
   glutInitWindowPosition (400, 200);
   glutEntryFunc(which_window);
   glutMouseFunc(mouse);
   init();
   glutDisplayFunc(display_win1);

// Kreira Prozor 2
   win2 = glutCreateWindow("Drugi prozor");
   glutEntryFunc(which_window);
   glutMouseFunc(mouse);
   init();
   glutDisplayFunc(display);

   glutMainLoop();
   return 0;
}
```

Kada se ovaj program startuje, posle kompajliranja i linkovanja, dobija se kao izlaz prozor koji je prikazan na donjoj slici.



Ispod su prikazani pojedinačni prozori, kao i prateći tekstualni prozor koji prikazuje aktivan prozor.


Prvi prozor
Drugi prozor


"C:\LaTeXXX\OpenGL\Moji programi\Debug\Dva prozora II.exe"
Ulazak Prozor 1
Izlazak Prozor 1
Ulazak Prozor 2
Izlazak Prozor 2
Ulazak Prozor 2
Izlazak Prozor 2
Ulazak Prozor 2
Izlazak Prozor 2

OpenGL (22)

Pripremio Dragan Cvetković

OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazano matematičko rešenje zadatog problema (prevođenje kućice na traženi položaj) uz poštovanje zadataih parametara. Na kraju sledi kompletan listing programa koji to oslikava.

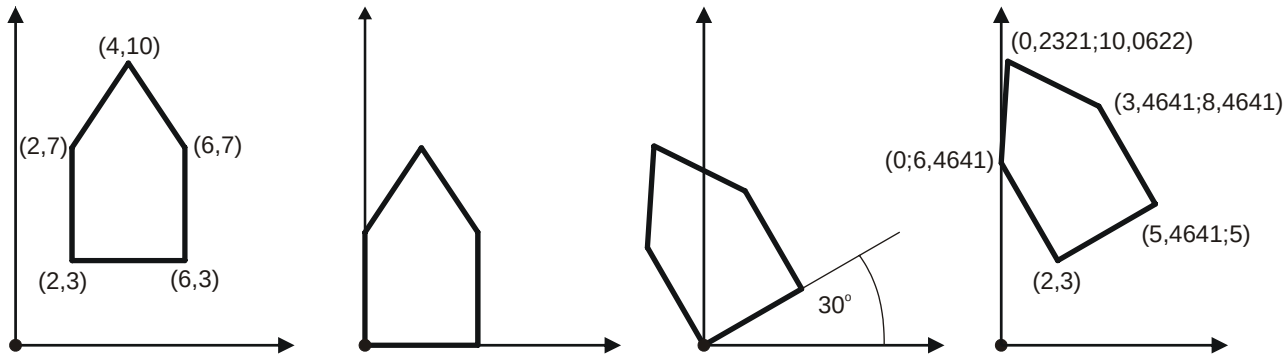
.

2D transformacije – Zadatak 1.

U XY koordinatnom sistemu kućica je određena koordinatama: A(2,3), B(6,3), C(6,7), D(4,10) i E(2,7). Kućica se pomera sa tačkom A u koordinatni početak, u toj tački se rotira za 30 stepeni u smeru suprotnom od smera kretanja kazaljke na satu i na kraju se kućica translira na početni položaj. Odrediti:

1. Kompozitnu matricu transformacija
2. Nove koordinate kućice

Rešenje



Kompozitna matrica transformacija je:

$$M = T(2,3)\cdot R(30^o)\cdot T(-2,-3)$$

$$M = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos 30^o & -\sin 30^o & 0 \\ \sin 30^o & \cos 30^o & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & 2 \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 3 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & \frac{7}{2}-\sqrt{3} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 2-\frac{3\sqrt{3}}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

Treba odrediti nove koordinate tačaka, posle transformacija. Na gornjem crtežu su prikazane koordinate iz programa AutoCAD, a sada treba proveriti da li su to zaista te koordinate.

Koordinata A'

$$A' = \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & \frac{7}{2}-\sqrt{3} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 2-\frac{3\sqrt{3}}{2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} \qquad A' = \begin{bmatrix} \frac{\sqrt{3}}{2}\cdot 2-\frac{1}{2}\cdot 3+\frac{7}{2}-\sqrt{3} \\ \frac{1}{2}\cdot 2+\frac{\sqrt{3}}{2}\cdot 3+2-\frac{3\sqrt{3}}{2} \\ 0\cdot 2+0\cdot 3+1 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

Koordinata B'

$$B' = \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & \frac{7}{2}-\sqrt{3} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 2-\frac{3\sqrt{3}}{2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 6 \\ 3 \\ 1 \end{bmatrix} \qquad B' = \begin{bmatrix} \frac{\sqrt{3}}{2}\cdot 6-\frac{1}{2}\cdot 3+\frac{7}{2}-\sqrt{3} \\ \frac{1}{2}\cdot 6+\frac{\sqrt{3}}{2}\cdot 3+2-\frac{3\sqrt{3}}{2} \\ 0\cdot 6+0\cdot 3+1 \end{bmatrix} = \begin{bmatrix} 2\sqrt{3}+2 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 5,4641 \\ 5 \\ 1 \end{bmatrix}$$

Koordinata C'

$$C' = \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & \frac{7}{2}-\sqrt{3} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 2-\frac{3\sqrt{3}}{2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 6 \\ 7 \\ 1 \end{bmatrix} \qquad C' = \begin{bmatrix} \frac{\sqrt{3}}{2}\cdot 6-\frac{1}{2}\cdot 7+\frac{7}{2}-\sqrt{3} \\ \frac{1}{2}\cdot 6+\frac{\sqrt{3}}{2}\cdot 7+2-\frac{3\sqrt{3}}{2} \\ 0\cdot 6+0\cdot 7+1 \end{bmatrix} = \begin{bmatrix} 2\sqrt{3} \\ 5+2\sqrt{3} \\ 1 \end{bmatrix} = \begin{bmatrix} 3,4641 \\ 8,4641 \\ 1 \end{bmatrix}$$

Koordinata D'

$$D' = \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & \frac{7}{2}-\sqrt{3} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 2-\frac{3\sqrt{3}}{2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 10 \\ 1 \end{bmatrix} \qquad D' = \begin{bmatrix} \frac{\sqrt{3}}{2}\cdot 4-\frac{1}{2}\cdot 10+\frac{7}{2}-\sqrt{3} \\ \frac{1}{2}\cdot 4+\frac{\sqrt{3}}{2}\cdot 10+2-\frac{3\sqrt{3}}{2} \\ 0\cdot 4+0\cdot 10+1 \end{bmatrix} = \begin{bmatrix} \sqrt{3}-\frac{3}{2} \\ 4+\frac{7\sqrt{3}}{2} \\ 1 \end{bmatrix} = \begin{bmatrix} 0,2321 \\ 10,0622 \\ 1 \end{bmatrix}$$

Koordinata E'

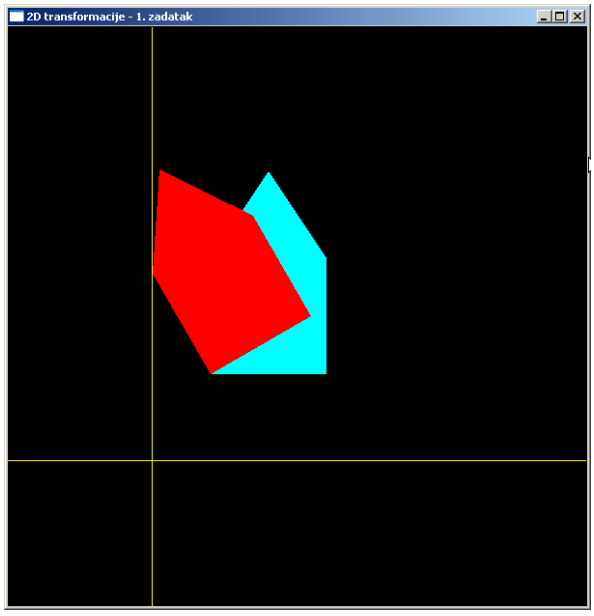
$$E' = \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & \frac{7}{2}-\sqrt{3} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 2-\frac{3\sqrt{3}}{2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 7 \\ 1 \end{bmatrix} \qquad E' = \begin{bmatrix} \frac{\sqrt{3}}{2}\cdot 2-\frac{1}{2}\cdot 7+\frac{7}{2}-\sqrt{3} \\ \frac{1}{2}\cdot 2+\frac{\sqrt{3}}{2}\cdot 7+2-\frac{3\sqrt{3}}{2} \\ 0\cdot 2+0\cdot 7+1 \end{bmatrix} = \begin{bmatrix} 0 \\ 3+2\sqrt{3} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 6,4641 \\ 1 \end{bmatrix}$$

Sledi kompletan listing programa koji ovaj problem rešava:

```
/*
* Program koji resava zadatak 1
*/
#include <GL/glut.h>
void init(void)
{
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_FLAT);
}

//      Funkcija koja crta koordinatne ose
void nacrtaj_koordinatne_ose()
{
      // Iscrtavanje X ose
      glBegin(GL_LINES);
            glVertex2f(-10.0, 0.0);
            glVertex2f(20.0, 0.0);
      glEnd();

      // Iscrtavanje Y ose
      glBegin(GL_LINES);
            glVertex2f(0.0, -10.0);
            glVertex2f(0.0, 20.0);
      glEnd();
}

void nacrtaj_polygon(void)
{
   glBegin (GL_POLYGON);
   glVertex2f(2.0, 3.0);
   glVertex2f(6.0, 3.0);
   glVertex2f(6.0, 7.0);
   glVertex2f(4.0, 10.0);
   glVertex2f(2.0, 7.0);
   glEnd();

   glFlush ();
}

void display(void)
{
   glPushMatrix();
   glClear (GL_COLOR_BUFFER_BIT);
   glColor3f (1.0, 1.0, 0.0);
   glPolygonMode(GL_FRONT, GL_FILL);
   nacrtaj_koordinatne_ose ();

  // Iscrtavanje inicijalnog poligona
   glColor3f (0.0, 1.0, 1.0);
   nacrtaj_polygon();

      glColor3f (1.0, 0.0, 0.0);
      glTranslatef(2.0, 3.0, 0.0);
      glRotatef (30.0, 0.0, 0.0, 1.0);
      glTranslatef(-2.0, -3.0, 0.0);
      // Iscrtavanje krajnjeg poligona
      nacrtaj_polygon();
      glPopMatrix();
   glFlush ();
}

void reshape (int w, int h)
{
glViewport (0, 0, (GLsizei) w, (GLsizei) h);
```

```
glMatrixMode (GL_PROJECTION);
   glLoadIdentity ();
   if (w <= h)
      glOrtho (-5.0, 15.0, -5.0*(GLfloat)h/(GLfloat)w,
         15.0*(GLfloat)h/(GLfloat)w, -1.0, 1.0);
   else
      glOrtho (-5.0*(GLfloat)w/(GLfloat)h,
         15.0*(GLfloat)w/(GLfloat)h, -5.0, 15.0, -1.0, 1.0);
glMatrixMode(GL_MODELVIEW);
}

int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (600, 600);
   glutInitWindowPosition (100, 100);
   glutCreateWindow ("2D transformacije - 1. zadatak");
   init ();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutMainLoop();
   return 0;
}
```

Kao rezultat rada ovog programa pojavljuje se slika 1.



Slika 1

OpenGL (23)

Pripremio Dragan Cvetković

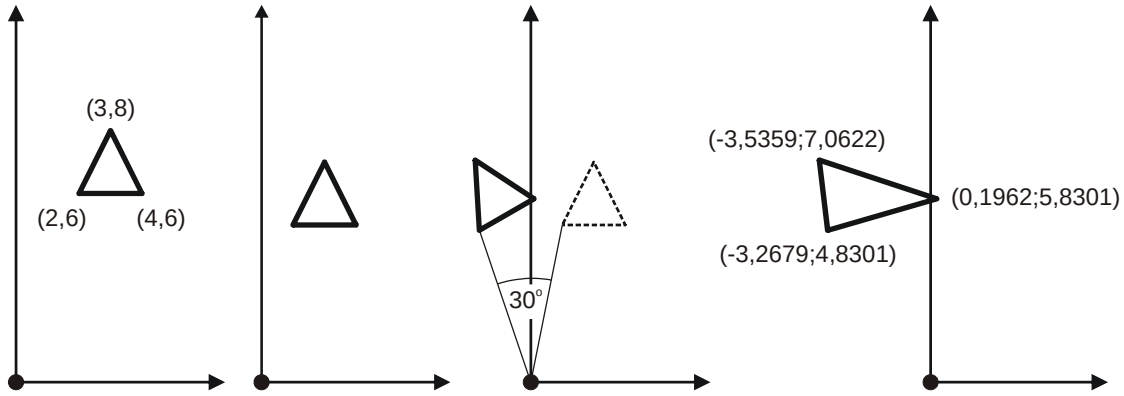
OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazano matematičko rešenje zadatog problema (prevođenje trougla na traženi položaj) uz poštovanje zadataih parametara. Na kraju sledi kompletan listing programa koji to oslikava.

2D transformacije – Zadatak 2.

U XY koordinatnom sistemu trougao je određen koordinatama: A(2,6), B(4,6) i C(3,8). Trougao se pomera za tačku (-1,-1), u novoj tački se rotira za 30 stepeni u smeru suprotnom od smera kretanja kazaljke na satu i na kraju se trougao skalira faktorima s_x=2 i s_y=1. Odrediti:

1. Kompozitnu matricu transformacija
2. Nove koordinate trougla

Rešenje



Kompozitna matrica transformacija je:

$$M = S(2,1) \cdot R(30^o) \cdot T(-1,-1)$$

$$M = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos 30^o & -\sin 30^o & 0 \\ \sin 30^o & \cos 30^o & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} \sqrt{3} & -1 & 0 \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} \sqrt{3} & -1 & 1-\sqrt{3} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & -\frac{1}{2}-\frac{\sqrt{3}}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

Treba odrediti nove koordinate tačaka, posle transformacija. Na gornjem crtežu su prikazane koordinate iz programa AutoCAD, a sada treba proveriti da li su to zaista te koordinate.

Koordinata A'

$$A' = \begin{bmatrix} \sqrt{3} & -1 & 1-\sqrt{3} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & -\frac{1}{2}-\frac{\sqrt{3}}{2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 6 \\ 1 \end{bmatrix} \qquad A' = \begin{bmatrix} \sqrt{3}\cdot 2 - 1\cdot 6 + 1 - \sqrt{3} \\ \frac{1}{2}\cdot 2 + \frac{\sqrt{3}}{2}\cdot 6 - \frac{1}{2} - \frac{\sqrt{3}}{2} \\ 0\cdot 2 + 0\cdot 6 + 1 \end{bmatrix} = \begin{bmatrix} \sqrt{3}-5 \\ \frac{1}{2}+\frac{5\sqrt{3}}{2} \\ 1 \end{bmatrix} = \begin{bmatrix} -3,2679 \\ 4,8301 \\ 1 \end{bmatrix}$$

Koordinata B'

$$B' = \begin{bmatrix} \sqrt{3} & -1 & 1-\sqrt{3} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & -\frac{1}{2}-\frac{\sqrt{3}}{2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 6 \\ 1 \end{bmatrix} \qquad B' = \begin{bmatrix} \sqrt{3}\cdot 4 - 1\cdot 6 + 1 - \sqrt{3} \\ \frac{1}{2}\cdot 4 + \frac{\sqrt{3}}{2}\cdot 6 - \frac{1}{2} - \frac{\sqrt{3}}{2} \\ 0\cdot 4 + 0\cdot 6 + 1 \end{bmatrix} = \begin{bmatrix} 3\sqrt{3}-5 \\ \frac{3}{2}+\frac{5\sqrt{3}}{2} \\ 1 \end{bmatrix} = \begin{bmatrix} 0,1962 \\ 5,8301 \\ 1 \end{bmatrix}$$

Koordinata C'

$$C' = \begin{bmatrix} \sqrt{3} & -1 & 1-\sqrt{3} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & -\frac{1}{2}-\frac{\sqrt{3}}{2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 8 \\ 1 \end{bmatrix} \qquad C' = \begin{bmatrix} \sqrt{3}\cdot 3 - 1\cdot 8 + 1 - \sqrt{3} \\ \frac{1}{2}\cdot 3 + \frac{\sqrt{3}}{2}\cdot 8 - \frac{1}{2} - \frac{\sqrt{3}}{2} \\ 0\cdot 3 + 0\cdot 8 + 1 \end{bmatrix} = \begin{bmatrix} 2\sqrt{3}-7 \\ 1+\frac{7\sqrt{3}}{2} \\ 1 \end{bmatrix} = \begin{bmatrix} -3,5359 \\ 7,0622 \\ 1 \end{bmatrix}$$

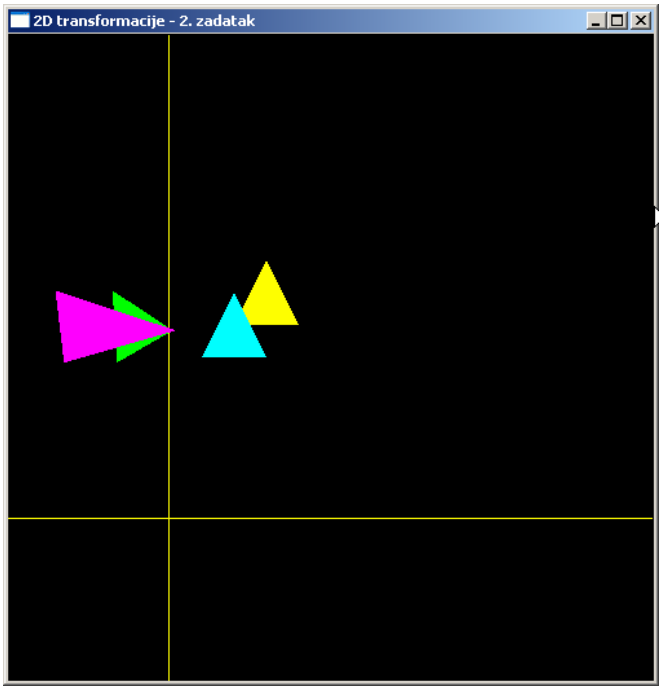
Sledi kompletan listing programa koji ovaj problem rešava:

```
/*
* Program koji resava zadatak 1
*/
#include <GL/glut.h>

void init(void)
{
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_FLAT); }
//      Funkcija koja crta koordinatne ose
void crtaj_koordinatne_ose()
```

```
{
	//	Crtanje X ose
	glBegin(GL_LINES);
		glVertex2f(-10.0, 0.0);
		glVertex2f(20.0, 0.0);
	glEnd();

	// Crtanje Y ose
	glBegin(GL_LINES);
		glVertex2f(0.0, -10.0);
		glVertex2f(0.0, 20.0);
	glEnd();
}

void crtaj_trougao(void)
{
   glBegin (GL_POLYGON);
   glVertex2f(2.0, 6.0);
   glVertex2f(4.0, 6.0);
   glVertex2f(3.0, 8.0);
   glEnd();

   glFlush ();
}

void display(void)
{
	// Crtanje prvog trougla
	glPushMatrix();
	glClear (GL_COLOR_BUFFER_BIT);
	glColor3f (1.0, 1.0, 0.0);
	glPolygonMode(GL_FRONT, GL_FILL);
	crtaj_koordinatne_ose ();
	glLineWidth (2.5);
	crtaj_trougao();

	// Crtanje drugog trougla posle translacije
	glColor3f (0.0, 1.0, 1.0);
	glTranslatef (-1.0, -1.0, 0.0);
	crtaj_trougao();

	// Crtanje treceg trougla posle rotacije
	glLoadIdentity ();
	glColor3f (0.0, 1.0, 0.0);
	glRotatef (30.0, 0.0, 0.0, 1.0);
	glTranslatef (-1.0, -1.0, 0.0);
	crtaj_trougao();

	// Crtanje cetvrtog trougla posle skaliranja
	glLoadIdentity ();
	glColor3f (1.0, 0.0, 1.0);
	glScalef (2,1,0);
	glRotatef (30.0, 0.0, 0.0, 1.0);
	glTranslatef (-1.0, -1.0, 0.0);
	crtaj_trougao();
	glPopMatrix();
   glFlush (); }

void reshape (int w, int h)
{
glViewport (0, 0, (GLsizei) w, (GLsizei) h);
glMatrixMode (GL_PROJECTION);

   glLoadIdentity ();
```

```
   if (w <= h)
      glOrtho (-5.0, 15.0, -5.0*(GLfloat)h/(GLfloat)w,
         15.0*(GLfloat)h/(GLfloat)w, -1.0, 1.0);
   else
      glOrtho (-0.5*(GLfloat)w/(GLfloat)h,
         15.0*(GLfloat)w/(GLfloat)h, -5.0, 15.0, -1.0, 1.0);

glMatrixMode(GL_MODELVIEW); }

int main(int argc, char** argv) {
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (500, 500);
   glutInitWindowPosition (100, 100);
   glutCreateWindow ("2D transformacije - 2. zadatak");
   init ();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutMainLoop();
   return 0; }
```

Kao rezultat rada ovog programa pojavljuje se slika 1.



Slika 1

OpenGL (24)

Pripremio Dragan Cvetković

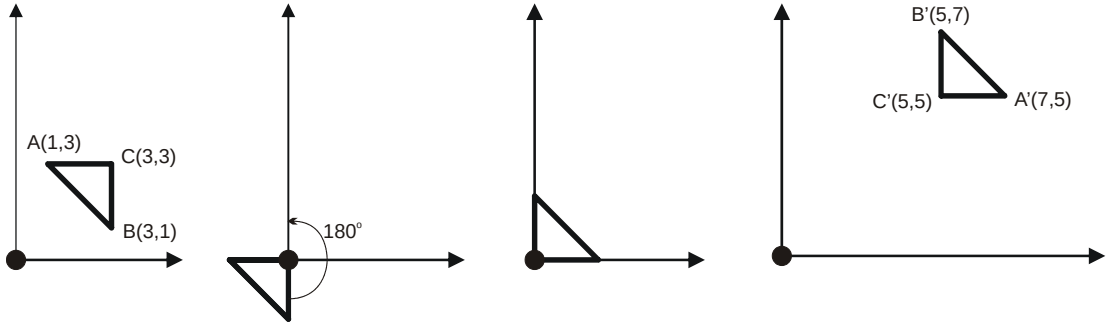
OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazano matematičko rešenje zadatog problema (prevođenje trougla na traženi položaj) uz poštovanje zadataih parametara. Na kraju sledi kompletan listing programa koji to oslikava.

2D transformacije – Zadatak 3.

U XY koordinatnom sistemu trougao je određen koordinatama: A(1,3), B(3,1) i C(3,3). Koordinatni sistem se nalazi u donjem levom uglu monitora. Odrediti kompozitnu matricu transformacije kako bi se postigao efekat promene položaja trougla ABC u položaj A'B'C' gde su koordinate tačaka A'(7,5), B'(5,7) i C'(5,5). Znači. Treba

1. odrediti kompozitnu matricu transformacija, i
2. proveriti da li su dobijene potrebne koordinate trougla.

Rešenje



Rešenje se svodi na sledeće korake:

1. Translacija temena C u koordinatni početak;
2. Rotacija oko koordinatnog početka za 180^o
3. Translacija temena C u tačku C'

Kompozitna matrica transformacija je:

$$M = T(5,5) \cdot R(180^o) \cdot T(-3,-3)$$

$$M = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos 180^o & -\sin 180^o & 0 \\ \sin 180^o & \cos 180^o & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} -1 & 0 & 5 \\ 0 & -1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} -1 & 0 & 8 \\ 0 & -1 & 8 \\ 0 & 0 & 1 \end{bmatrix}$$

Treba odrediti nove koordinate tačaka, posle transformacija. Na gornjem crtežu su prikazane koordinate koje su date zadatkom, a sada treba proveriti da li su to zaista te koordinate.

Koordinata A'

$$A' = \begin{bmatrix} -1 & 0 & 8 \\ 0 & -1 & 8 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} \qquad A' = \begin{bmatrix} (-1)\cdot 1+8 \\ (-1)\cdot 3+8 \\ 0\cdot 1+0\cdot 3+1 \end{bmatrix} = \begin{bmatrix} 7 \\ 5 \\ 1 \end{bmatrix}$$

Koordinata B'

$$B' = \begin{bmatrix} -1 & 0 & 8 \\ 0 & -1 & 8 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix} \qquad B' = \begin{bmatrix} (-1)\cdot 3+8 \\ (-1)\cdot 1+8 \\ 0\cdot 1+0\cdot 3+1 \end{bmatrix} = \begin{bmatrix} 5 \\ 7 \\ 1 \end{bmatrix}$$

Koordinata C'

$$C' = \begin{bmatrix} -1 & 0 & 8 \\ 0 & -1 & 8 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 3 \\ 1 \end{bmatrix} \qquad C' = \begin{bmatrix} (-1)\cdot 3+8 \\ (-1)\cdot 3+8 \\ 0\cdot 1+0\cdot 3+1 \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \\ 1 \end{bmatrix}$$

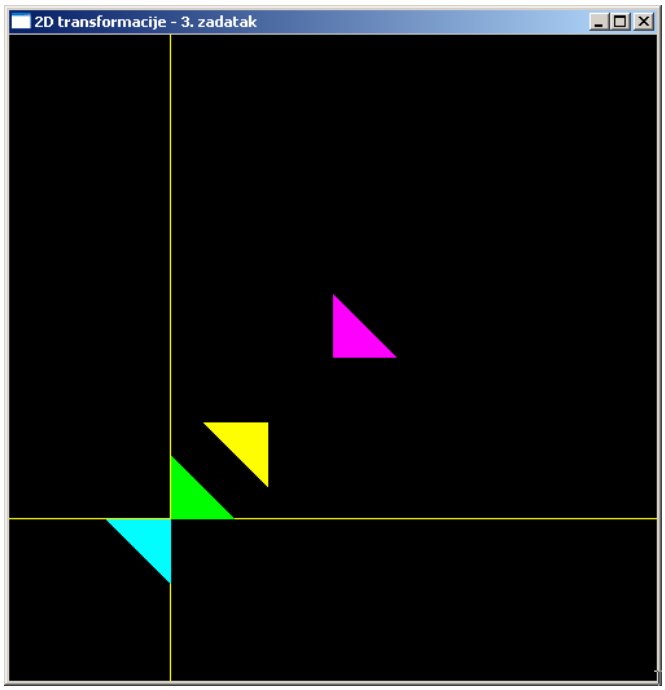
Sledi kompletan listing programa koji ovaj problem rešava:

```
/*
* Program koji resava zadatak 3
*/
#include <GL/glut.h>

void init(void)
{
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_FLAT);
}

//      Funkcija koja crta koordinatne ose
void crtaj_koordinatne_ose()
{
      //      Crtanje X ose
      glBegin(GL_LINES);
            glVertex2f(-10.0, 0.0);
            glVertex2f(20.0, 0.0);
      glEnd();

      // Crtanje Y ose
      glBegin(GL_LINES);
            glVertex2f(0.0, -10.0);
            glVertex2f(0.0, 20.0);
      glEnd();
}
```

```
void crtaj_trougao(void)
{
   glBegin (GL_POLYGON);
   glVertex2f(1.0, 3.0);
   glVertex2f(3.0, 3.0);
   glVertex2f(3.0, 1.0);
   glEnd();

   glFlush ();
}

void display(void)
{
      // Crtanje prvog trougla
      glPushMatrix();
      glClear (GL_COLOR_BUFFER_BIT);
      glColor3f (1.0, 1.0, 0.0);
      glPolygonMode(GL_FRONT, GL_FILL);
      crtaj_koordinatne_ose ();
      glLineWidth (2.5);
      crtaj_trougao();

      // Crtanje drugog trougla posle translacije
      glColor3f (0.0, 1.0, 1.0);
      glTranslatef (-3.0, -3.0, 0.0);
      crtaj_trougao();

      // Crtanje treceg trougla posle rotacije
      glLoadIdentity ();
      glColor3f (0.0, 1.0, 0.0);
      glRotatef (180.0, 0.0, 0.0, 1.0);
      glTranslatef (-3.0, -3.0, 0.0);
      crtaj_trougao();

      // Crtanje cetvrtog trougla posle skaliranja
      glLoadIdentity ();
      glColor3f (1.0, 0.0, 1.0);
      glTranslatef (5.0, 5.0, 0.0);
      glRotatef (180.0, 0.0, 0.0, 1.0);
      glTranslatef (-3.0, -3.0, 0.0);
      crtaj_trougao();

      glPopMatrix();
   glFlush ();
}

void reshape (int w, int h)
{
glViewport (0, 0, (GLsizei) w, (GLsizei) h);
glMatrixMode (GL_PROJECTION);

   glLoadIdentity ();
   if (w <= h)
      glOrtho (-5.0, 15.0, -5.0*(GLfloat)h/(GLfloat)w,
         15.0*(GLfloat)h/(GLfloat)w, -1.0, 1.0);
   else
      glOrtho (-0.5*(GLfloat)w/(GLfloat)h,
```

```
        15.0*(GLfloat)w/(GLfloat)h, -5.0, 15.0, -1.0, 1.0);

glMatrixMode(GL_MODELVIEW);
}

int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (500, 500);
   glutInitWindowPosition (100, 100);
   glutCreateWindow ("2D transformacije - 3. zadatak");
   init ();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutMainLoop();
   return 0;
}
```

Kao rezultat rada ovog programa pojavljuje se slika 1.



Slika 1

OpenGL (25)

Pripremio Dragan Cvetković

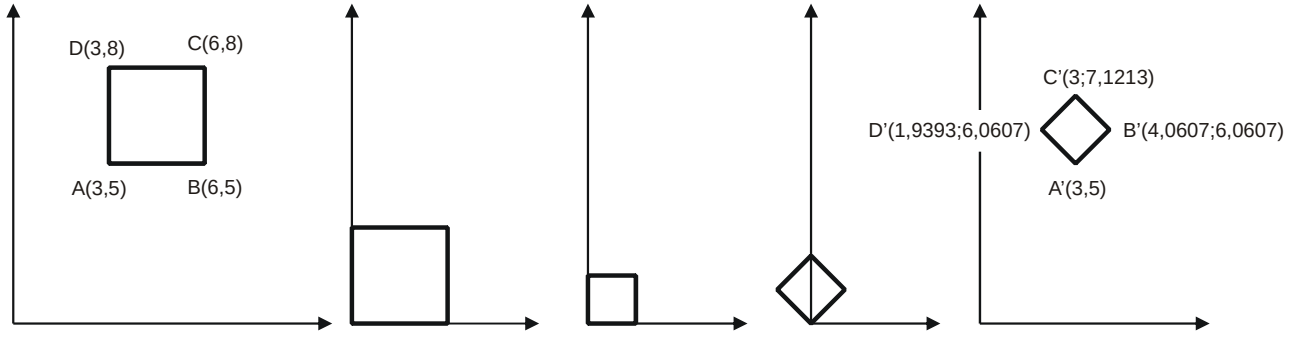
OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazano matematičko rešenje zadatog problema (prevođenje trougla na traženi položaj) uz poštovanje zadataih parametara. Na kraju sledi kompletan listing programa koji to oslikava.

2D transformacije – Zadatak 4.

U XY koordinatnom sistemu kvadrat je određen koordinatama: A(3,5), B(6,5), C(6,8) i D(3,8). Kućica se pomera sa tačkom A u koordinatni početak, u toj tački se skalira sa faktorima skaliranja ½ i po *x* i po *y* osi, u istoj tački rotira za 45 stepeni u smeru suprotnom od smera kretanja kazaljke na satu i na kraju se teme kvadrata A translira na početni položaj. Odrediti:

1. Kompozitnu matricu transformacija
2. Nove koordinate kvadrata

Rešenje



Kompozitna matrica transformacija je:

$$M = T(3,5) \cdot R(45^o) \cdot S\left(\frac{1}{2},\frac{1}{2}\right) \cdot T(-3,-5)$$

$$M = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos 45^o & -\sin 45^o & 0 \\ \sin 45^o & \cos 45^o & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 3 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 5 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & 3 \\ \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & 5 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & 3+\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & 5-2\sqrt{2} \\ 0 & 0 & 1 \end{bmatrix}$$

Treba odrediti nove koordinate tačaka, posle transformacija. Na gornjem crtežu su prikazane koordinate iz programa AutoCAD, a sada treba proveriti da li su to zaista te koordinate.

Koordinata A'

$$A' = \begin{bmatrix} \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & 3+\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & 5-2\sqrt{2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 5 \\ 1 \end{bmatrix} \qquad A' = \begin{bmatrix} \frac{3\sqrt{2}}{4} - \frac{5\sqrt{2}}{4} + 3 + \frac{\sqrt{2}}{2} \\ \frac{3\sqrt{2}}{4} + \frac{5\sqrt{2}}{4} + 5 - 2\sqrt{2} \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 1 \end{bmatrix}$$

Koordinata B'

$$B' = \begin{bmatrix} \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & 3+\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & 5-2\sqrt{2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 6 \\ 5 \\ 1 \end{bmatrix} \qquad B' = \begin{bmatrix} \frac{3\sqrt{2}}{2} - \frac{5\sqrt{2}}{4} + 3 + \frac{\sqrt{2}}{2} \\ \frac{3\sqrt{2}}{2} + \frac{5\sqrt{2}}{4} + 5 - 2\sqrt{2} \\ 1 \end{bmatrix} = \begin{bmatrix} 3+\frac{3\sqrt{2}}{4} \\ 5+\frac{3\sqrt{2}}{4} \\ 1 \end{bmatrix} = \begin{bmatrix} 4,0607 \\ 6,0607 \\ 1 \end{bmatrix}$$

Koordinata C'

$$C' = \begin{bmatrix} \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & 3+\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & 5-2\sqrt{2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 6 \\ 8 \\ 1 \end{bmatrix} \qquad C' = \begin{bmatrix} \frac{3\sqrt{2}}{2} - 2\sqrt{2} + 3 + \frac{\sqrt{2}}{2} \\ \frac{3\sqrt{2}}{2} + 2\sqrt{2} + 5 - 2\sqrt{2} \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 5+\frac{3\sqrt{2}}{2} \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 7,1213 \\ 1 \end{bmatrix}$$

Koordinata D'

$$D' = \begin{bmatrix} \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & 3+\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & 5-2\sqrt{2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 8 \\ 1 \end{bmatrix} \qquad D' = \begin{bmatrix} \frac{3\sqrt{2}}{4} - 2\sqrt{2} + 3 + \frac{\sqrt{2}}{2} \\ \frac{3\sqrt{2}}{4} + 2\sqrt{2} + 5 - 2\sqrt{2} \\ 1 \end{bmatrix} = \begin{bmatrix} 3-\frac{3\sqrt{2}}{4} \\ 5+\frac{3\sqrt{2}}{4} \\ 1 \end{bmatrix} = \begin{bmatrix} 1,9393 \\ 6,0607 \\ 1 \end{bmatrix}$$

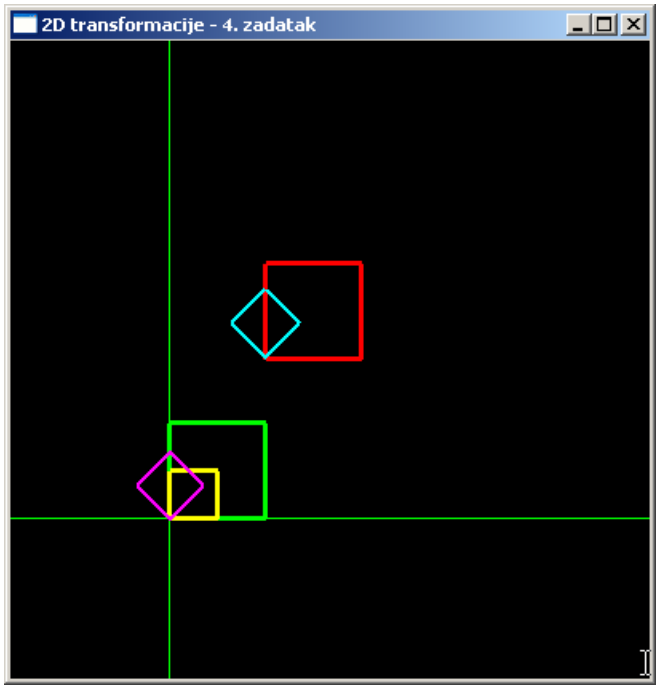
Sledi kompletan listing programa koji ovaj problem rešava:

```
/*
* Program koji resava zadatak 4
*/
#include <GL/glut.h>

void init(void)
{
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_FLAT);
}

//      Funkcija koja crta koordinatne ose
void crtaj_koordinatne_ose()
{
      //      x-osa
      glBegin(GL_LINES);
            glVertex2f(-10.0, 0.0);
            glVertex2f(20.0, 0.0);
      glEnd();

      // y-osa
      glBegin(GL_LINES);
            glVertex2f(0.0, -10.0);
            glVertex2f(0.0, 20.0);
      glEnd();
}

void crtaj_kvadrat1(void)
{
   glBegin (GL_POLYGON);
   glVertex2f(3.0, 5.0);
   glVertex2f(6.0, 5.0);
   glVertex2f(6.0, 8.0);
   glVertex2f(3.0, 8.0);
```

```
   glEnd();
   glFlush ();
}

void crtaj_kvadrat(void)
{
   glBegin (GL_POLYGON);
   glVertex2f(0.0, 0.0);
   glVertex2f(3.0, 0.0);
   glVertex2f(3.0, 3.0);
   glVertex2f(0.0, 3.0);
   glEnd();
   glFlush ();
}
void display(void)

{
   glPushMatrix();

   glClear (GL_COLOR_BUFFER_BIT);
   glColor3f (0.0, 1.0, 0.0);
   glPolygonMode(GL_FRONT, GL_LINE);
   crtaj_koordinatne_ose ();
   glPopMatrix();

   glPushMatrix();
   glLoadIdentity ();
   glLineWidth(2.5);

   // Crtanje prvog kvadrata
   glColor3f (1.0, 0.0, 0.0);
   crtaj_kvadrat1 ();

   // Crtanje drugog kvadrata posle translacije
   glColor3f (0.0, 1.0, 0.0);
   crtaj_kvadrat ();

   // Crtanje treceg kvadrata skaliranja
   glColor3f (1.0, 1.0, 0.0);
   glScalef(0.5, 0.5, 0.0);
   crtaj_kvadrat ();

   // Crtanje cetvrtog kvadrata posle rotiranja
   glColor3f (1.0, 0.0, 1.0);
   glRotatef (45.0,0.0,0.0,1.0);
   crtaj_kvadrat ();

   // Crtanje petog kvadrata posle transliranja
   glColor3f (0.0, 1.0, 1.0);
   glTranslatef(11.4, 3.0, 0.0);
   crtaj_kvadrat ();

   glPopMatrix();
   glFlush ();
}
void reshape (int w, int h)
{
glViewport (0, 0, (GLsizei) w, (GLsizei) h);

glMatrixMode (GL_PROJECTION);

   glLoadIdentity ();
   if (w <= h)
      glOrtho (-5.0, 15.0, -5.0*(GLfloat)h/(GLfloat)w,
         15.0*(GLfloat)h/(GLfloat)w, -1.0, 1.0);
```

```
   else
      glOrtho (-5.0*(GLfloat)w/(GLfloat)h,
         15.0*(GLfloat)w/(GLfloat)h, -5.0, 15.0, -1.0, 1.0);

glMatrixMode(GL_MODELVIEW);
}

int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (400, 400);
   glutInitWindowPosition (100, 100);
   glutCreateWindow ("2D transformacije - 4. zadatak");
   init ();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
     glutMainLoop();
   return 0;
}
```

Kao rezultat rada ovog programa pojavljuje se slika 1.



Slika 1

OpenGL (26)

Pripremio Dragan Cvetković

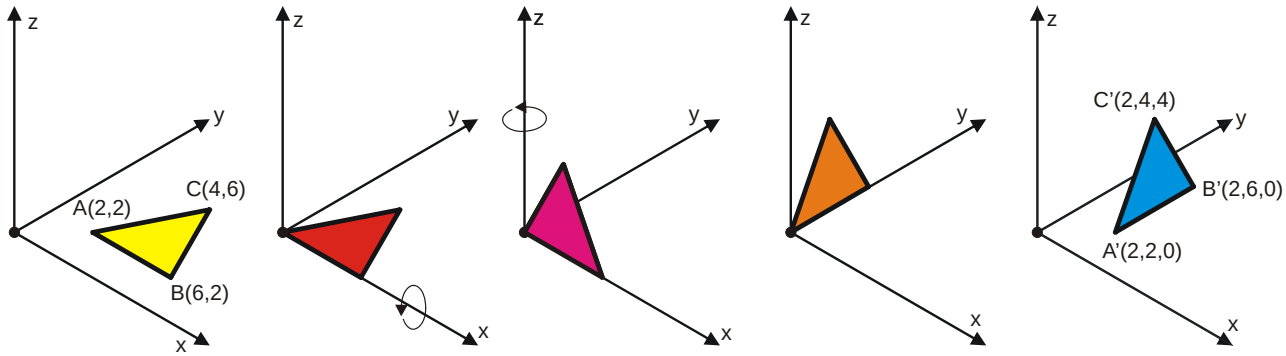
OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazano matematičko rešenje zadatog problema (prevođenje trougla na traženi položaj) uz poštovanje zadatih parametara. Na kraju sledi kompletan listing programa koji to oslikava.

3D transformacije – Zadatak 1.

U XYZ koordinatnom sistemu trougao je određen koordinatama: A(2,2,0), B(6,2,0) i C(4,6,0). Trougao se pomera sa tačkom A u koordinatni početak, u toj tački se vrši rotacija trougla oko *x* ose za 90 stepeni, u istoj tački se rotira oko *z* ose za 90 stepeni i na kraju se teme trougla A translira na početni položaj. Odrediti:

1. Kompozitnu matricu transformacija
2. Nove koordinate trougla

Rešenje



Kompozitna matrica transformacija je:

$$M = T(2,2)\cdot R_z\left(90^o\right)\cdot R_x\left(90^o\right)\cdot T(-2,-2)$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos 90^o & -\sin 90^o & 0 & 0 \\ \sin 90^o & \cos 90^o & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 90^o & -\sin 90^o & 0 \\ 0 & \sin 90^o & \cos 90^o & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} 0 & -1 & 0 & 2 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Treba odrediti nove koordinate tačaka, posle transformacija. Na gornjem crtežu su prikazane koordinate iz programa AutoCAD, a sada treba proveriti da li su to zaista te koordinate.

Koordinata A'

$$A' = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 2 \\ 0 \\ 1 \end{bmatrix} \qquad A' = \begin{bmatrix} 2 \cdot 1 \\ 1 \cdot 2 \\ 1 \cdot 2 - 2 \cdot 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 0 \\ 1 \end{bmatrix}$$

Koordinata B'

$$B' = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 6 \\ 2 \\ 0 \\ 1 \end{bmatrix} \qquad B' = \begin{bmatrix} 2 \cdot 1 \\ 1 \cdot 6 \\ 1 \cdot 2 - 2 \cdot 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \\ 0 \\ 1 \end{bmatrix}$$

Koordinata C'

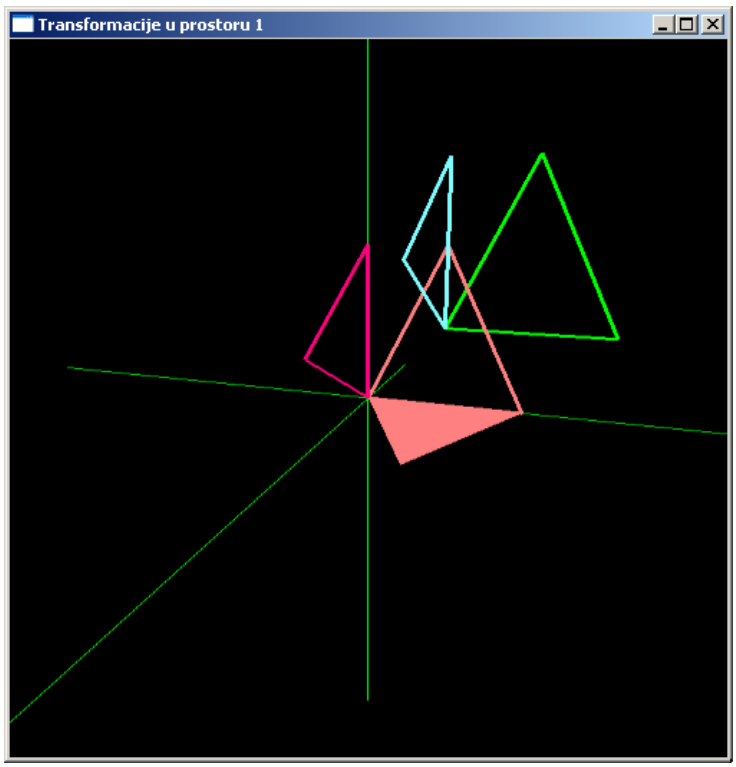
$$C' = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 6 \\ 0 \\ 1 \end{bmatrix} \qquad C' = \begin{bmatrix} 2 \cdot 1 \\ 1 \cdot 4 \\ 1 \cdot 6 - 2 \cdot 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 4 \\ 1 \end{bmatrix}$$

Sledi kompletan listing programa koji ovaj problem rešava:

```
/*
* Program koji resava zadatak 1
* Nenad Dulanovic
*/

#include <GL/glut.h>

void init(void)
{
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_FLAT);
}

//      Funkcija koja crta koordinatne ose
void draw_coordinate_grid()
{
      // x-osa
      glBegin(GL_LINES);
            glVertex2f(-10.0, 0.0);
            glVertex2f(20.0, 0.0);
      glEnd();

      // y-osa
      glBegin(GL_LINES);
            glVertex2f(0.0, -10.0);
            glVertex2f(0.0, 20.0);
      glEnd();

      // z-osa
      glBegin(GL_LINES);
            glVertex3f(0.0, 0.0, -10);
            glVertex3f(0.0, 0.0, 20);
      glEnd();
}

void crtaj_trougao(void)
{
   glBegin (GL_POLYGON);
   glVertex3f(2.0, 2.0, 0.0);
   glVertex3f(6.0, 2.0, 0.0);
   glVertex3f(4.0, 6.0, 0.0);
   glEnd();
   glFlush ();
}

void display(void)
{
   glPushMatrix();
   glClear (GL_COLOR_BUFFER_BIT);
   glColor3f (0.0, 1.0, 0.0);
   glPolygonMode(GL_FRONT, GL_LINE);
   draw_coordinate_grid ();
   glLineWidth(2.5);
   crtaj_trougao();

   glColor3f (1.0, 0.5, 0.5);
   glTranslatef (-2.0, -2.0, 0.0);
   crtaj_trougao();
   glTranslatef (2.0, 2.0, 0.0);

      glRotatef (90.0, 1.0, 0.0, 0.0);
      glTranslatef (-2.0, -2.0, 0.0);
      crtaj_trougao();
```

```
      glColor3f (1.0, 0.0, 0.5);
      glTranslatef (2.0, 2.0, 0.0);
      glRotatef (90.0, 0.0, 1.0, 0.0);

      glTranslatef (-2.0, -2.0, 0.0);
      crtaj_trougao();

      glColor3f (0.5, 1.0, 1.0);
      glTranslatef (2.0, 0.0, 2.0);
      crtaj_trougao();
      glPopMatrix();

   glFlush ();
}

void reshape (int w, int h)
{
glViewport (0, 0, (GLsizei) w, (GLsizei) h);
glMatrixMode (GL_PROJECTION);
      glLoadIdentity ();
      gluPerspective(60.0, (GLfloat) w/(GLfloat) h, 1.0, 20.0);
      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();
      gluLookAt (5.0, 5.0, 15.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
}

int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (500, 500);
   glutInitWindowPosition (100, 100);
   glutCreateWindow ("Transformacije u prostoru 1");
   init ();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutMainLoop();
   return 0;
}
```

Kao rezultat rada ovog programa pojavljuje se slika 1.



Slika 1

OpenGL (27)

Pripremio Dragan Cvetković

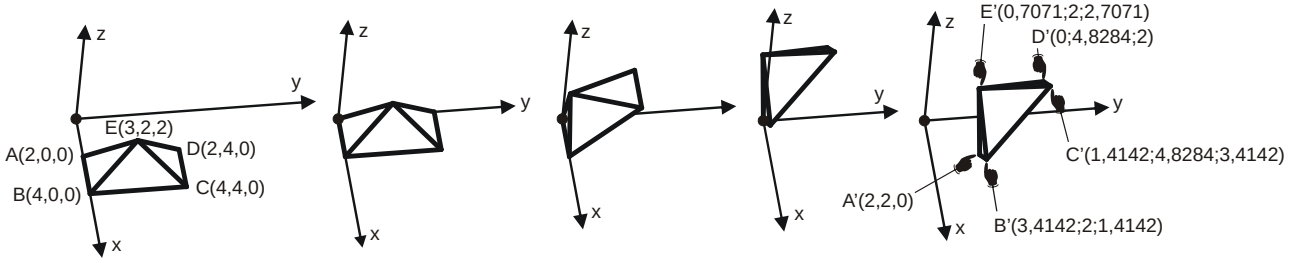
OpenGL predstavlja primarno okruženje za razvoj interaktivnih 2D i 3D grafičkih aplikacija. Ovde će biti prikazano matematičko rešenje zadatog problema (prevođenje piramide na traženi položaj) uz poštovanje zadatih parametara. Na kraju sledi kompletan listing programa koji to oslikava.

3D transformacije – Zadatak 2.

U XYZ koordinatnom sistemu četvorostrana piramida je određena temenima čije su koordinate: A(2,0,0), B(4,0,0), C(4,4,0), D(2,4,0) i E(3,2,2). Piramida se pomera sa tačkom A (teme bazisa) u koordinatni početak, u toj tački se vrši rotacija piramida oko *x* ose za 45 stepeni, u istoj tački se rotira oko *y* ose za -45 stepeni i na kraju se teme piramide A translira u tačku čije su koordinate (2,2,0). Odrediti:

1. Kompozitnu matricu transformacija
2. Nove koordinate piramide

Rešenje



Kompozitna matrica transformacija je:

$$M = T(2,2,0)\cdot R_y\left(-45^o\right)\cdot R_x\left(45^o\right)\cdot T(-2,0,0)$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(-45^o) & 0 & \sin(-45^o) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(-45^o) & 0 & \cos(-45^o) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 45^o & -\sin 45^o & 0 \\ 0 & \sin 45^o & \cos 45^o & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} & 2 \\ 0 & 1 & 0 & 2 \\ \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{1}{2} & -\frac{1}{2} & 2 \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 2 \\ \frac{\sqrt{2}}{2} & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{1}{2} & -\frac{1}{2} & 2-\sqrt{2} \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 2 \\ \frac{\sqrt{2}}{2} & \frac{1}{2} & \frac{1}{2} & -\sqrt{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Treba odrediti nove koordinate tačaka, posle transformacija. Na gornjem crtežu su prikazane koordinate iz programa AutoCAD, a sada treba proveriti da li su to zaista te koordinate.

Koordinata A'

$$A' = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{1}{2} & -\frac{1}{2} & 2-\sqrt{2} \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 2 \\ \frac{\sqrt{2}}{2} & \frac{1}{2} & \frac{1}{2} & -\sqrt{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad A' = \begin{bmatrix} 2 \cdot \frac{\sqrt{2}}{2} + 2 - \sqrt{2} \\ 2 \\ 2 \cdot \frac{\sqrt{2}}{2} - \sqrt{2} \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 0 \\ 1 \end{bmatrix}$$

Koordinata B'

$$B' = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{1}{2} & -\frac{1}{2} & 2-\sqrt{2} \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 2 \\ \frac{\sqrt{2}}{2} & \frac{1}{2} & \frac{1}{2} & -\sqrt{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad B' = \begin{bmatrix} 4 \cdot \frac{\sqrt{2}}{2} + 2 - \sqrt{2} \\ 2 \\ 4 \cdot \frac{\sqrt{2}}{2} - \sqrt{2} \\ 1 \end{bmatrix} = \begin{bmatrix} 2+\sqrt{2} \\ 2 \\ \sqrt{2} \\ 1 \end{bmatrix} = \begin{bmatrix} 3,4142 \\ 2 \\ 1,4142 \\ 1 \end{bmatrix}$$

Koordinata C'

$$C' = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{1}{2} & -\frac{1}{2} & 2-\sqrt{2} \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 2 \\ \frac{\sqrt{2}}{2} & \frac{1}{2} & \frac{1}{2} & -\sqrt{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 4 \\ 0 \\ 1 \end{bmatrix} \qquad C' = \begin{bmatrix} 2\sqrt{2} - 2 + 2 - \sqrt{2} \\ 2\sqrt{2} + 2 \\ 2\sqrt{2} + 2 - \sqrt{2} \\ 1 \end{bmatrix} = \begin{bmatrix} \sqrt{2} \\ 2\sqrt{2} + 2 \\ \sqrt{2} + 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1,4142 \\ 4,8284 \\ 3,4142 \\ 1 \end{bmatrix}$$

Koordinata D'

$$D' = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{1}{2} & -\frac{1}{2} & 2-\sqrt{2} \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 2 \\ \frac{\sqrt{2}}{2} & \frac{1}{2} & \frac{1}{2} & -\sqrt{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 4 \\ 0 \\ 1 \end{bmatrix} \qquad D' = \begin{bmatrix} \sqrt{2} - 2 + 2 - \sqrt{2} \\ 2\sqrt{2} + 2 \\ \sqrt{2} + 2 - \sqrt{2} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 2\sqrt{2} + 2 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 4,8284 \\ 2 \\ 1 \end{bmatrix}$$

Koordinata E'

$$E' = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{1}{2} & -\frac{1}{2} & 2-\sqrt{2} \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 2 \\ \frac{\sqrt{2}}{2} & \frac{1}{2} & \frac{1}{2} & -\sqrt{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 2 \\ 2 \\ 1 \end{bmatrix} \qquad E' = \begin{bmatrix} \frac{3\sqrt{2}}{2} - 1 - 1 + 2 - \sqrt{2} \\ \sqrt{2} - \sqrt{2} + 2 \\ \frac{3\sqrt{2}}{2} + 1 + 1 - \sqrt{2} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ 2 \\ \frac{\sqrt{2}}{2} + 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0,7071 \\ 2 \\ 2,7071 \\ 1 \end{bmatrix}$$

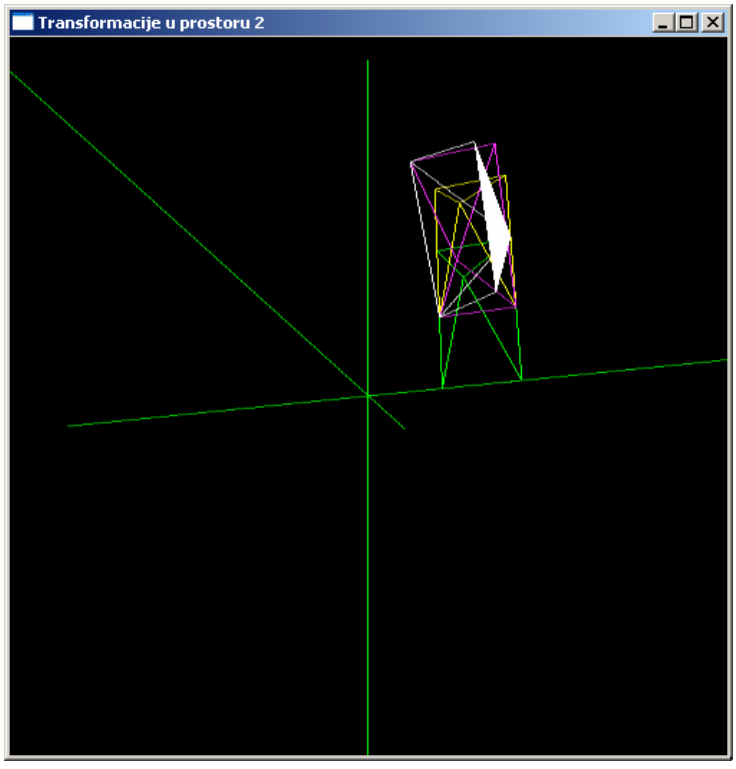
Sledi kompletan listing programa koji ovaj problem rešava:

```
/*
* Program koji resava zadatak 2
* Nenad Dulanovic
*/
```

```
#include <GL/glut.h>

void init(void)
{
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_FLAT);
}

//      Funkcija koja crta koordinatne
void draw_coordinate_grid()
{
      // x-osa
      glBegin(GL_LINES);
            glVertex2f(-10.0, 0.0);
            glVertex2f(20.0, 0.0);
      glEnd();

      // y-osa
      glBegin(GL_LINES);
            glVertex2f(0.0, -10.0);
            glVertex2f(0.0, 20.0);
      glEnd();

      // z-osa
      glBegin(GL_LINES);
            glVertex3f(0.0, 0.0, -10);
            glVertex3f(0.0, 0.0, 20);
      glEnd();
}

void draw_polygon(void)
{
   glBegin (GL_TRIANGLE_FAN);
   glVertex3f(3.0, 2.0, 2.0);
   glVertex3f(2.0, 0.0, 0.0);
   glVertex3f(4.0, 0.0, 0.0);
   glVertex3f(4.0, 4.0, 0.0);
   glVertex3f(2.0, 4.0, 0.0);
   glVertex3f(2.0, 0.0, 0.0);
   glEnd();

   glFlush ();
}

void display(void)
{
   glPushMatrix();

glClear (GL_COLOR_BUFFER_BIT);
glColor3f (0.0, 1.0, 0.0);
glPolygonMode(GL_FRONT, GL_LINE);

draw_coordinate_grid ();
draw_polygon();


      glColor3f (1.0, 1.0, 0.0);
      glTranslatef (0.0, 2.0, 0.0);
      draw_polygon();

      glColor3f (1.0, 0.2, 1.0);
      glRotatef (45.0, 1.0, 0.0, 0.0);
      draw_polygon();

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

```
      glTranslatef (2.0, 0.0, 0.0);
      glRotatef (45.0, 0.0, 1.0, 0.0);
      glTranslatef (-2.0, 0.0, 0.0);
      draw_polygon();

      glPopMatrix();

   glFlush ();
}

void reshape (int w, int h)
{
glViewport (0, 0, (GLsizei) w, (GLsizei) h);
glMatrixMode (GL_PROJECTION);

      glLoadIdentity ();
      gluPerspective(60.0, (GLfloat) w/(GLfloat) h, 1.0, 20.0);
      glMatrixMode(GL_MODELVIEW);

      glLoadIdentity();
      gluLookAt (5.0, -5.0, 15.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
}

int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (500, 500);
   glutInitWindowPosition (100, 100);
   glutCreateWindow ("Transformacije u prostoru 2");
   init ();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutMainLoop();
   return 0;
}
```

Kao rezultat rada ovog programa pojavljuje se slika 1.

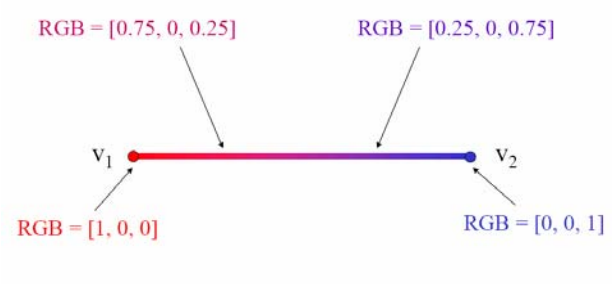


Slika 1

OpenGL – Interpolacija boje

Definicija:
– Da bi procenili vrednost (funkcije ili niza) između dve poznate vrednosti.
• OpenGL automatski interpolira boju između dva verteksa umesto nas.
• Kako to radi?

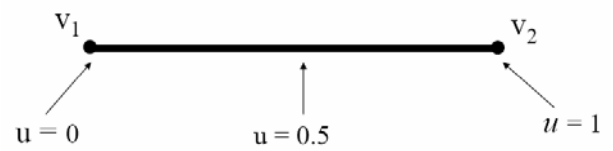
```
// crtanje linije
glBegin(GL_LINES);
     glColor3f(1.0, 0.0, 0.0); // red
     glVertex2f( 0, 0);
     glColor3f(0.0, 0.0, 1.0); // blue
     glVertex2f( 1, 0);
glEnd();
```



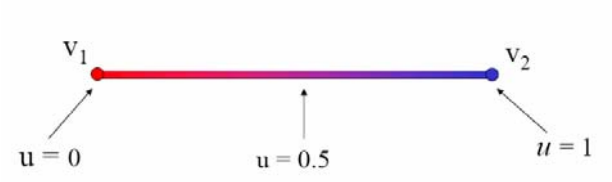
Linearna interpolacija podrazumeva:
– da znate dve konkretne vrednosti (boje u krajnjim tačkama)
– da znate da odredite nepoznatu tačku između tih vrednosti (boju između njih)
– da se process promene odvija konstantnim odnosom

• Može li se koristiti za „morph” između dve pozicije?
• Linijski segment između pozicija v1 i v2 :

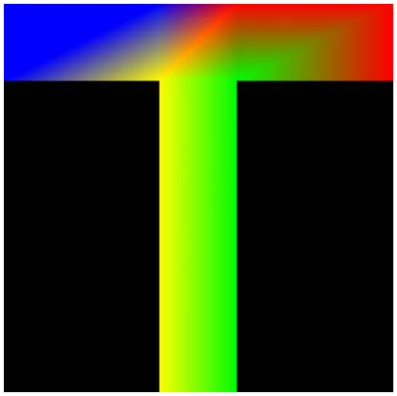
$v(u) = (1 - u) * v1 + (u) * v2$
$0 \leq u \leq 1$



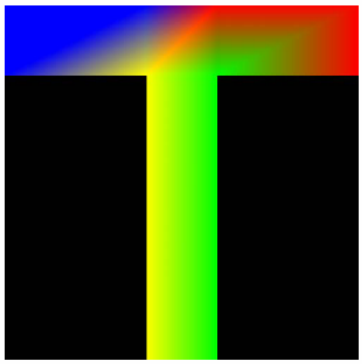
• Boja u v1 : c1 = [1, 0, 0]
• Boja u v2 : c2 = [0, 0, 1]
• Boja u v(u) : c(u) = (1 – u) * c1 + (u) * c2

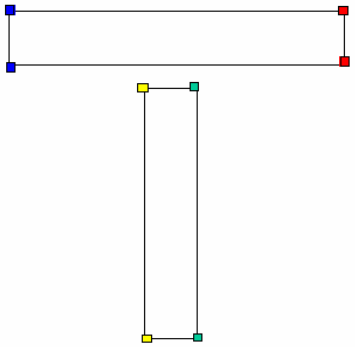


Spajanje boja susednih elemenata

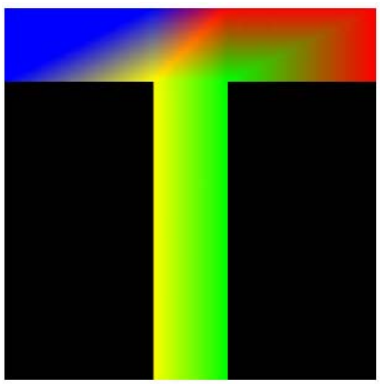
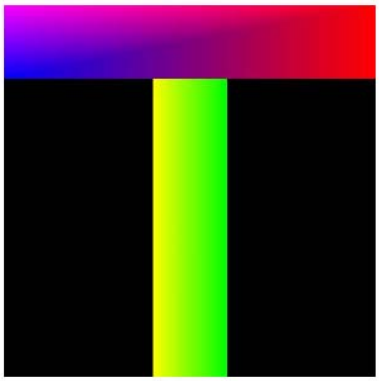


Zamislimo da su ti poligoni obojeni na sledeći način:



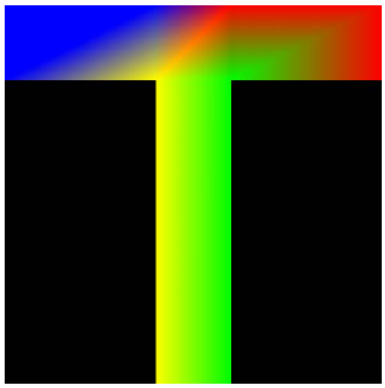
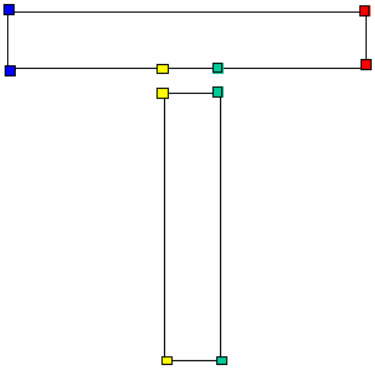


Nije dobro!

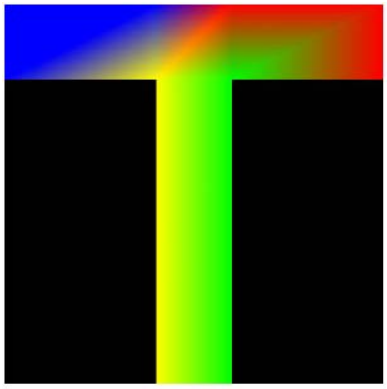
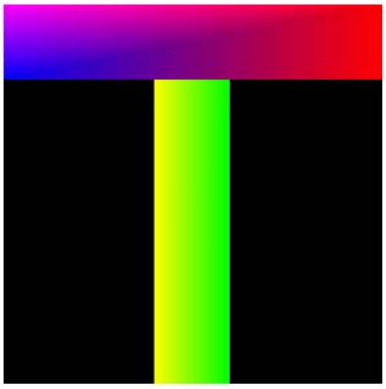



Nije dobro!

Napomena: To ne funkcioniše zbog načina na koji OpenGL renderuje poligone

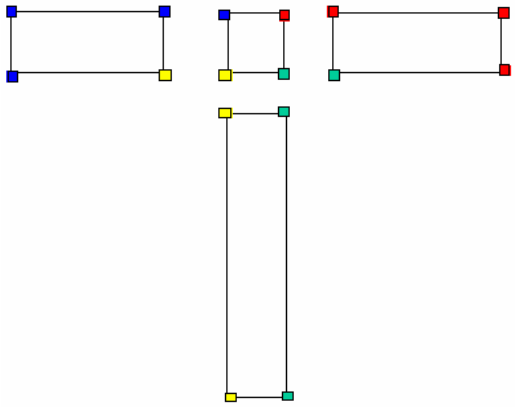



Još uvek nije dobro!

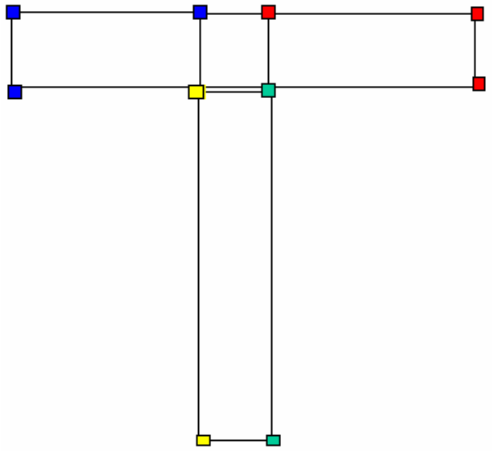



Još uvek nije dobro!

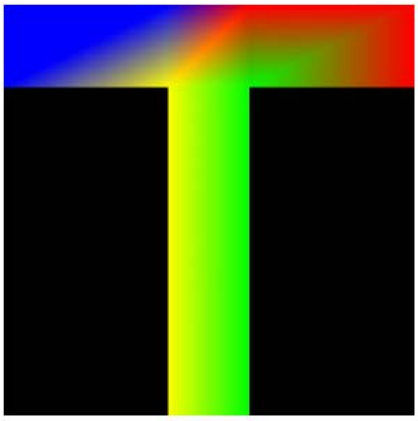
Potrebno je da kreirate više poligona:



Sad je dobro!



Sad je dobro!

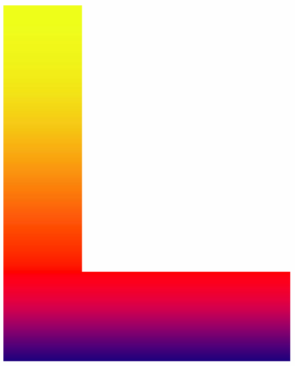


Zadatak

Nacrtati obojeno L na rasterskoj mreži koristeći:

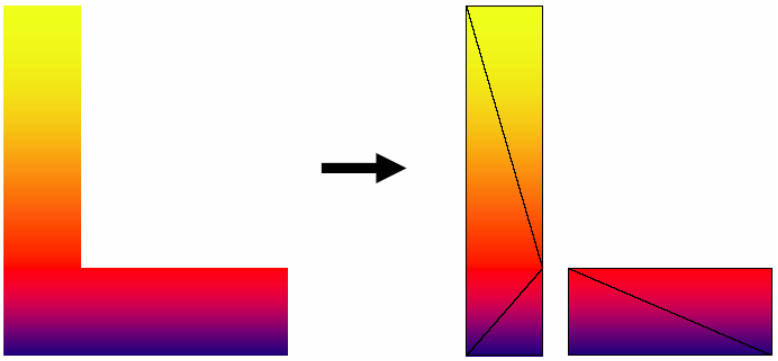
- **GL_TRIANGLE_STRIP**
- `glBegin( … )`
- `glEnd()`
- `glVertex2f(x, y)`

– `glColor3f(r, g, b)`

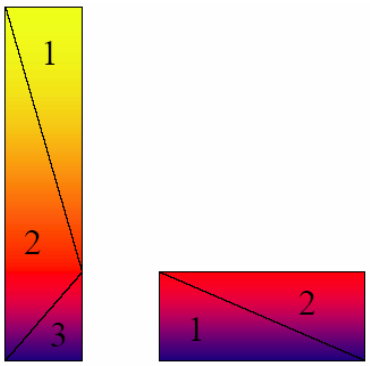


Pretpostaviti da se radi u 2D:
– gluOrtho2D(0.0, 1.0, 0.0, 1.0);

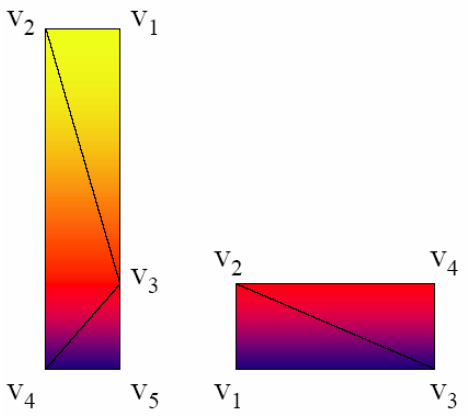
Jedan način da se to izvede je pomoću dve trake:



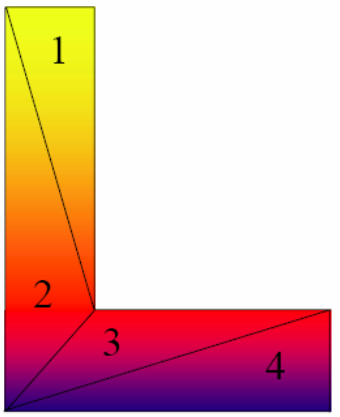
Pomoću trouglova



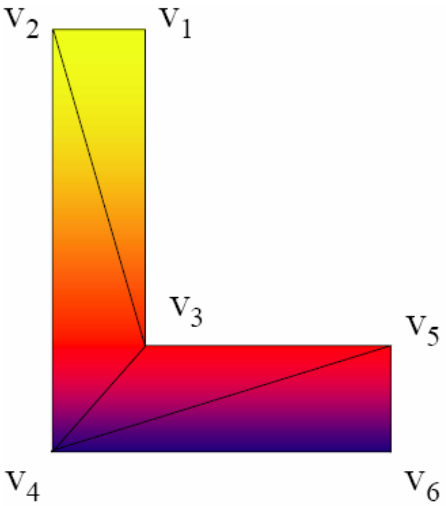
Pomoću verteksa



Bolji načina (pomoću trouglova)



Bolji način (pomoću verteksa)



Pripremio Dragan Marković

OpenGL – više nivoa detalja

Poznato je da čovekov vizuelni sistem vidi manje detalja kako se objekat udaljava i biva sve manji! Iz toga proizilazi da 3D grafički sistem treba da poseduje nekakav metod za redukovanje detalja kako se objekat smanjuje. To ne samo da čini objekat realističnijim, već i smanjuje opterećenje procesora kod renderovanja složenih (i verovatno velikih) slika koje se postavljaju na mali objekat. Tehnika koja to omogućuje je tehnika sa više detalja (levels of detail - **LOD**). OpenGL može to da ostvari pomoću metoda filtriranja pozivanjem funkcije:

```
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);
```

Međutimr, ova tehnika može lako da dovede do neželjenih efekata i načina da se to prevaziđe je da se koristi mehanizam koji korisniku omogućuje da koristi više slika za različite nivoe detalja (različite rezolucija). Ta tehnika je poznata kao **mimapiranje** ***(mipmapping)***, i izvedena je iz latinske izreke

```
multim im parvo (hence mip)
```

što znači „many things in a small place”.

Da bi koristili mipmapiranje, moramo da obezbedimo teksture koje su dimenzija stepena broja 2 između najveće mape pa sve do 1x1 mape. Na primer, ako krenemo od 64x16 teksturne mape (imajte na umu da teksturna mapa **ne mora da bude kvadratna**), tada morate da obezbedite teksturne mape za dimenzije 32x8, 16x4, 8x2, 4x1, 2x1 i 1x1.

Manje mape se obično prave od najveće mape preko nekakve interpolacije i usrednjavanja, ali OpenGL ne zahteva da to radite, i manje teksture ne moraju biti u relaciji sa većom!

Da bi specificirali te teksture vratimo se izvornom funkcijskom pozivu:

```
    glTexImage2D(GL_TEXTURE_2D, 0, 3,
                 TextureImage->sizeX, TextureImage->sizeY,
                 0, GL_RGB, GL_UNSIGNED_BYTE, TextureImage->data);
```

i koristimo različite vrednosti za parametre *level, width, height* i *image*.

Krenimo od ranije korišćene slike ***crate.bmp*** rezolucije 256x256 i kreirajmo 8 manjih teksturnih mapa. Svaka od njih je prefarbana tako da bude očigledno kada OpenGL promeni teksturnu mapu! Takođe je potrebno da promenimo funkciju koja učitava teksturne mape u smislu da ima petlju za učitavanje 9 teksturnih mapa. Da bi lakše realizovali proces nazivi fajlova su crate0.bmp, crate1.bmp do crate8.bmp. Kôd ima oblik:

```
GLvoid LoadTexture(GLvoid)
{
char file[11];
int i;
Image *TextureImage;
        for( i = 0; i<9; i++ )
    {
     sprintf(file,"crate%1d.bmp",i);  /* create the name */
    TextureImage = (Image *) malloc(sizeof(Image));
    if (TextureImage == NULL) {
        printf("Error allocating space for image");
        exit(1);
    }
```

```
    if (!ImageLoad(file, TextureImage)) {
        exit(1);
    }
          /*  2d texture, level of detail i , 3 components (red, green, blue),
*/
    /*  x size from image, y size from image,
*/
    /*  border 0 (normal), rgb color data, unsigned byte data, and finally the
data itself. */

    glTexImage2D(GL_TEXTURE_2D, i, 3,
                 TextureImage->sizeX, TextureImage->sizeY,
                 0, GL_RGB, GL_UNSIGNED_BYTE, TextureImage->data);
    free( TextureImage );
   }
```

Mehanizam učitavanja fajlova je prost i nivoi teksturne mape su vezani za promenljivu ***i*** u petlji.

Poslednji deo setovanja informiše OpenGL da želimo da koristimo mipmapiranje, tj. kada su teksturne slike manje jednostavno vršimo odabir iz raspoloživog skupa mapa.

```
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,
                    GL_NEAREST); /*  cheap scaling when image bigger than
texture */
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,
                    GL_NEAREST_MIPMAP_NEAREST);  /* use mipmaps */
```

To bi bilo sve!

`Kompajlirajte, linkujte i pokrenite program!`

Program će prikazati teksturnu mapu nanetu na pravougaonik. Možete da menjate dimenzije pomoću tastera „x,y,z” (uvećanje) , odnosno „X,Y,Z” (umanjenje).

Uočite kada uradimo smanjivanje da se koriste različite mape. Takođe pokušajte da menjate dimenziju prozora.

Filtar za funkciju minimizacije koristi najbližu teksturnu mapu za datu veličinu. Na raspolaganju su i druge funkcije koje možete da isprobate:

```
GL_NEAREST_MIPMAP_NEAREST
GL_NEAREST_MIPMAP_LINEAR
GL_LINEAR_MIPMAP_NEAREST
GL_LINEAR_MIPMAP_LINEAR
```

Napomena: Preuzmite arhivu OGLlod.rar u kojoj se nalaze izvorni kôd i devet BMP fajlova za demonstraciju opisane tehnike.

Pripremio Dragan Marković