Računarska grafika
OpenGL 1

OpenGL

- Open Graphics Library
- API - biblioteka koja omogućava pristup grafičkom hardveru
- Sadrži preko 250 različitih funkcija koje korisnik koristi da bi definisao objekte, slike, operacije i da bi producirao interaktivnu 3D scenu.
- Nativni OpenGL ne uključuje funkcije koje kreiraju ili upravljaju prozorima na kojima se crta ili funkcije koje procesiraju input od korisnika.
- Sve komande (procedure i funkcije) biblioteke GL počinju prefiksom gl, sve konstante - prefiksom GL_.

Prednosti OpenGL-a

- Industrijski standard - potpuno otvoren, nezavisan od proizvođača
- Stabilnost
- Pouzdanost i prenosivost
- Razvoj
- Skalabilnost - aplikacije se mogu prilagoditi skoro svakoj klasi uređaja
- Jednostavna upotreba
- Dobra dokumentacija

GLUT (The OpenGL Utility Toolkit)

- GLUT biblioteka je skup funkcija za OpenGL programe, koje većinom obavljaju ulazno-izlazne operacije.
- Navedeno uključuje definiciju prozora, kontrolu i upravljanje prozorom i očitavanje inputa korisnika sa tastature i miša. Također su dostupne funkcije za prikaz nekih primitiva poput kocke, kugle i čajnika.
- Odgovarajuće komande i konstante biblioteke GLUT imaju prefiks glut (GLUT_).

Inicijalizacija programa - `glutInit*` rutine

Prije svega, u svakom OpenGL programu potrebno je izvršiti inicijalizaciju GLUT stanja.

Inicijalizacija se vrši pozivom `glutInit*` rutina. Nijedna druga `glut*` ili `gl*` funkcija ne smije biti pozvana prije ovih rutina.

`glutInit`

- Vrši inicijalizaciju GLUT biblioteke i započinje sesiju sa *window* sistemom.

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

- Funkciji je potrebno proslijediti argumente sa komandne linije onakve kakvi su došli u naš program (argumenti `argc` i `argv` od `main`-a).
- Funkcija nema povratni tip, ali će modifikovati svoje ulazne argumente na način da će ekstraktovati sve one namijenjene GLUT-u (koje GLUT biblioteka zna tumačiti).
- Ova funkcija može terminirati program (ukoliko ne može uspostaviti sesiju sa *window* sistemom, nedostaje podrška za OpenGL ili su argumenti nevalidni).
- Funkciju pozvati tačno jednom!

Naš prvi OpenGL program

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

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    return 0;
}
```

glutInitWindowPosition

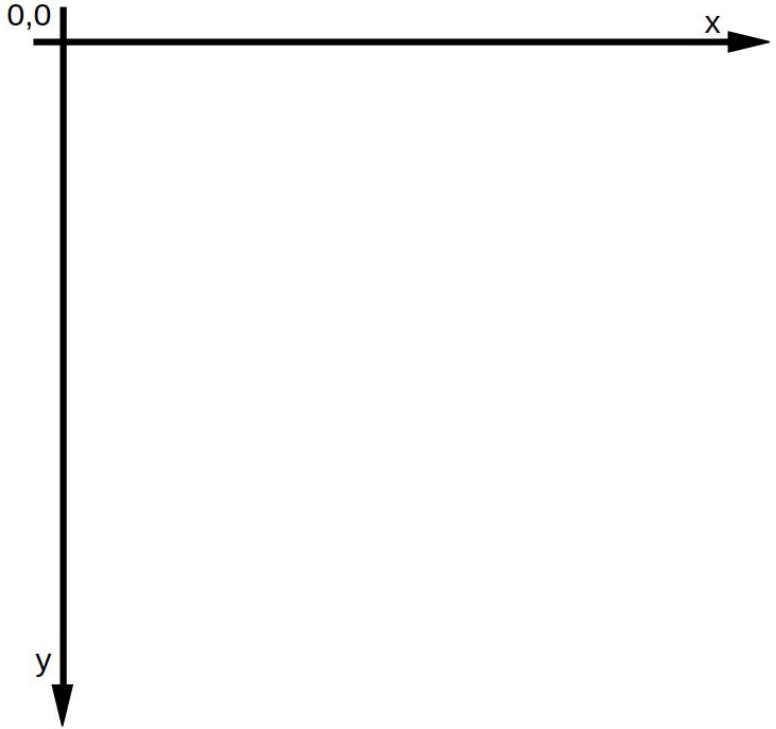
- Podešava poziciju prozora koji će biti korišten za crtanje.

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

- `x` i `y` su koordinate gornjeg lijevog ugla prozora za crtanje.
- Ukoliko se funkcija ne pozove, GLUT će kreirati prozor za crtanje u gornjem lijevom uglu prozora.

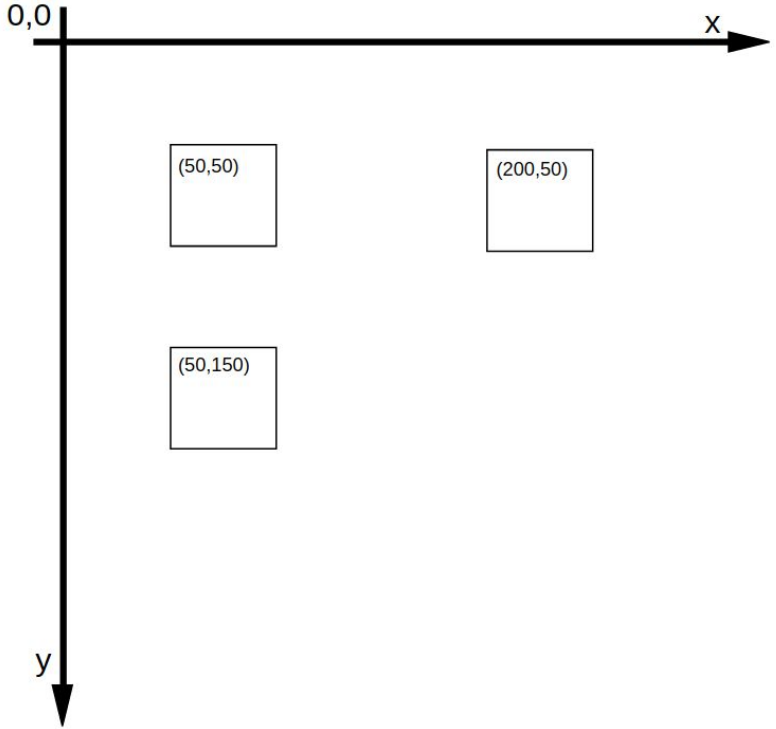
glutInitWindowPosition

- Koordinatni sistem našeg laptopa ili računara možemo zamisliti na sljedeći način:



glutInitWindowPosition

- Koordinatni sistem našeg laptopa ili računara možemo zamisliti na sljedeći način:



glutInitWindowSize

- Podešava veličinu prozora koji će biti korišten za crtanje.

```
void glutInitWindowSize(int width, int height);
```

- `width` i `height` su širina i visina prozora izražene u pikselima.
- Ukoliko se funkcija ne pozove, GLUT će koristiti svoje predefinisane vrijednosti za veličinu prozora (300, 300).

glutInitDisplayMode

- Služi za davanje postavki prozora koji će biti korišten za crtanje.

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

- Argument `mode` predstavlja vrijednost jedne ili više predefinisanih konstanti uvezanih operatorom OR (|). Svaka konstanta je jedna bitmaska:
  - GLUT_SINGLE
  - GLUT_DOUBLE
  - GLUT_RGB
  - GLUT_DEPTH
  - ...

Naš prvi OpenGL program

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

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(400, 300);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    return 0;
}
```

Kreiranje prozora

OpenGL koristi dvije vrste prozora:

- top-level prozore
- pod-prozore

Za početak, bavit ćemo se samo top-level prozorima.

glutCreateWindow

- Kreira top-level prozor imena `name`.

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

- ASCII niz karaktera koji se bude koristio kao argument funkcije bit će ispisan u statusnoj traci prozora za crtanje.
- Povratna vrijednost je unikatni ID prozora (koji zasad možemo ignorisati).
- Svaki kreirani prozor ima sa sobom asociran unikatni OpenGL context, koji čuva stanje prozora (ali se trenutno ne moramo time zamarati).
- Sam poziv funkcije neće prikazati prozor, ali tek nakon poziva ove funkcije možemo pozivati OpenGL funkcije.

glutMainLoop

- Ova funkcija započinje GLUT event processing petlju.

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

- Funkcija mora biti pozvana najviše jednom u svakom OpenGL programu. Ona interno ima beskonačnu petlju, što znači da funkcija nikad neće završiti, stoga je potrebno pozvati je nakon inicijalizacije i kreiranja prozora.
- Petlja će se terminirati kada korisnik zatvori top-level prozor ili ubije proces.

Nas prvi OpenGL program

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

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(400, 300);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutCreateWindow("Moj prozor");
    glutMainLoop();
    return 0;
}
```

Vjerovatno se pitate...

Poziv funkcije `glutMainLoop()` je obično zadnja linija main funkcije svakog OpenGL programa. Kako išta više uraditi ako smo zaglavili u petlji?

Callback funkcije

Prije pozivanja `glutMainLoop` potrebno je podesiti *callback* funkcije koje će GLUT pozivati u određenim trenucima:

- Kada se javi potreba za osvježavanjem prozora
- Kada se izabere stavka iz menija
- Kada korisnik pošalje input putem tastature
- Kada korisnik pošalje input putem miša
- Kada istekne neki *timeout*
- ...

glutDisplayFunc

- Postavlja *display callback* funkciju za trenutni prozor.

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

- Kad god se javi potreba za osvježavanjem prozora, poziva se *display* funkcija.
- Postoji način da forsiramo osvježavanje prozora, ali ćemo se time baviti naknadno.
- Funkcija nema argumenata.
- Kada se prozor kreira, ne postoji njegova *default display* funkcija. Onaj ko piše program dužan je podesiti *display* funkciju prije prikazivanja prozora (prije pozivanja `glutMainLoop` funkcije).

Display callback funkcija

- Unutar ove funkcije se radi rendering*.
- Svaka *display* funkcija obično ima tri dijela:
    1. Čisti trenutni prozor pozivom `glClear()`
    2. Poziva druge OpenGL funkcije koje će vršiti rendering.
    3. Šalje zahtjev za prikazivanje kreirane slike na ekran pozivom `glFlush()`

* rendering je proces kojim računar kreira sliku od modela. OpenGL je samo jedan primjer rendering sistema.

glClear

- Prije svega, potrebno je očistiti trenutni prozor, jer će se “crtanje” svaki put raditi iznova.
- Zapravo se čisti *framebuffer**.

```
void glClear(GLbitfield mask);
```

- Funkcija kao parametar uzima *bitmask*-u, koja predstavlja *buffer*-e koje treba očistiti. *Bitmask*-a je zapravo kombinacija potencijalno više vrijednosti povezanih operacijom OR.

* framebuffer - mjesto za pohranu piksela, tj. dio memorije kojim upravlja grafički hardver i prosljeđuje vrijednosti dalje onome ko prikazuje scenu (ekran).

Vrijednosti za `GLbitfield` parametar

Naziv buffer-a	Konstanta
Color Buffer	`GL_COLOR_BUFFER_BIT`
Depth Buffer	`GL_DEPTH_BUFFER_BIT`
Stencil Buffer	`GL_STENCIL_BUFFER_BIT`

- *Color buffer* je zapravo onaj na koji najčešće crtamo.
- *Depth buffer* se koristi za regulisanje vidljivosti objekata u 3D sceni.
- *Stencil buffer* se koristi za ograničavanje površine dostupne za crtanje.

Za početak, koristit ćemo samo *color buffer*.

glClearColor

- Funkcija podešava boju pozadine (na koju će OpenGL očistiti *color buffer*-e).

  ```
  void glClearColor(GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha);
  ```

- Parametri funkcije su vrijednosti u opsegu [0,1]. Prva tri definišu RGB komponente boje, dok `alpha` predstavlja prozirnost.
- Ukoliko se funkcija ne pozove, OpenGL će koristiti default boju - crnu (0,0,0,0).
- Boja pozadine je primjer OpenGL stanja (interno čuva neke vrijednosti), što znači da je dovoljno jednom pozvati funkciju, a ne svaki put pri renderiranju. Zbog toga se obično pozivi ove i sličnih funkcija vrše u funkciji inicijalizacije koja se poziva jednom, prije `glutMainLoop`.

glFlush

- Tek pozivom ove funkcije se uloženi trud zapravo prikazuje na ekranu :-)

  ```
  void glFlush(void);
  ```

- Bez poziva `glFlush` prozor će konstantno ostati prazan.

`void display(void)`

- Poslije čišćenja *buffer*-a, a prije *flush*-iranja, dozvoljeno je vršiti proizvoljne pozive funkcija koje nam OpenGL stavlja na raspolaganje.
- Za početak ćemo se baviti nekim jednostavnim primjerima, kao što su crtanje tačaka, linija i poligona na različitim pozicijama na ekranu i u različitim bojama i veličinama.

Crtanje tačaka

- Za crtanje tačke, potrebno je specificirati njene koordinate (3D):

```
void glVertex3f(float x, float y, float z);
```

- Za crtanje u 2D prostoru, može se koristiti funkcija glVertex2f, koja zapravo z-koordinatu postavlja na vrijednost 0:

```
void glVertex2f(float x, float y);
```

- Dostupne su još dvije varijante:

```
float v[3] = { x, y, z }; glVertex3fv( &v[0] );
float v[2] = { x, y }; glVertex2fv( &v[0] );
```

Crtanje tačaka

- Osim koordinata, svaka tačka je određena bojom.

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

- Parametri `r`, `g` i `b` predstavljaju vrijednosti RGB komponenti. Za poziv sa vrijednostima (0,0,0), svaki naredni *vertex* će imati crnu boju, dok će za (1,1,1) boja biti bijela.
- Naravno, sve druge boje je moguće dobiti različitim kombinacijama tri vrijednosti, pri čemu svaka vrijednost treba biti u opsegu [0,1].
- Jedan ili više poziva ove funkcije definiše i boje linija, trouglova, četverouglova i poligona.

Crtanje tačaka

- Inicijalno, svaka tačka (*vertex*) predstavlja jedan piksel. Veličinu tačke moguće je promijeniti pozivom `glPointSize`.

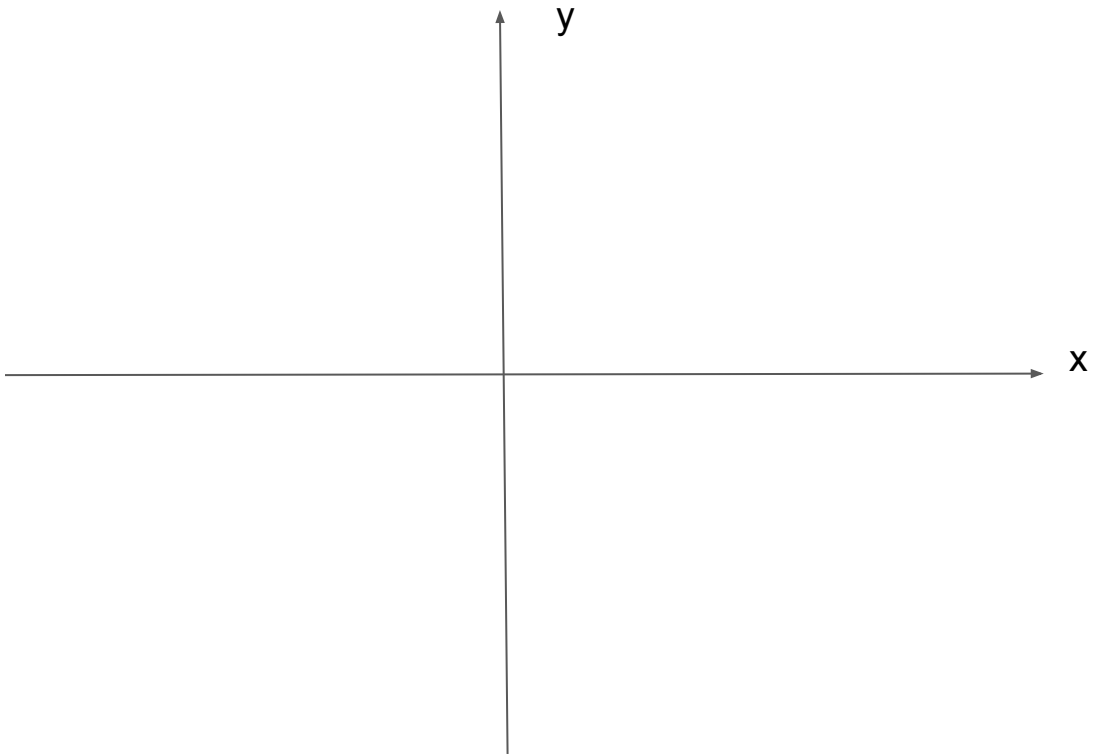
  ```
  void glPointSize(GLfloat size);
  ```

- Vrijednosti manje ili jednake nuli se smatraju nevalidnim.
- Ovu funkciju je potrebno pozvati prije `glBegin`.

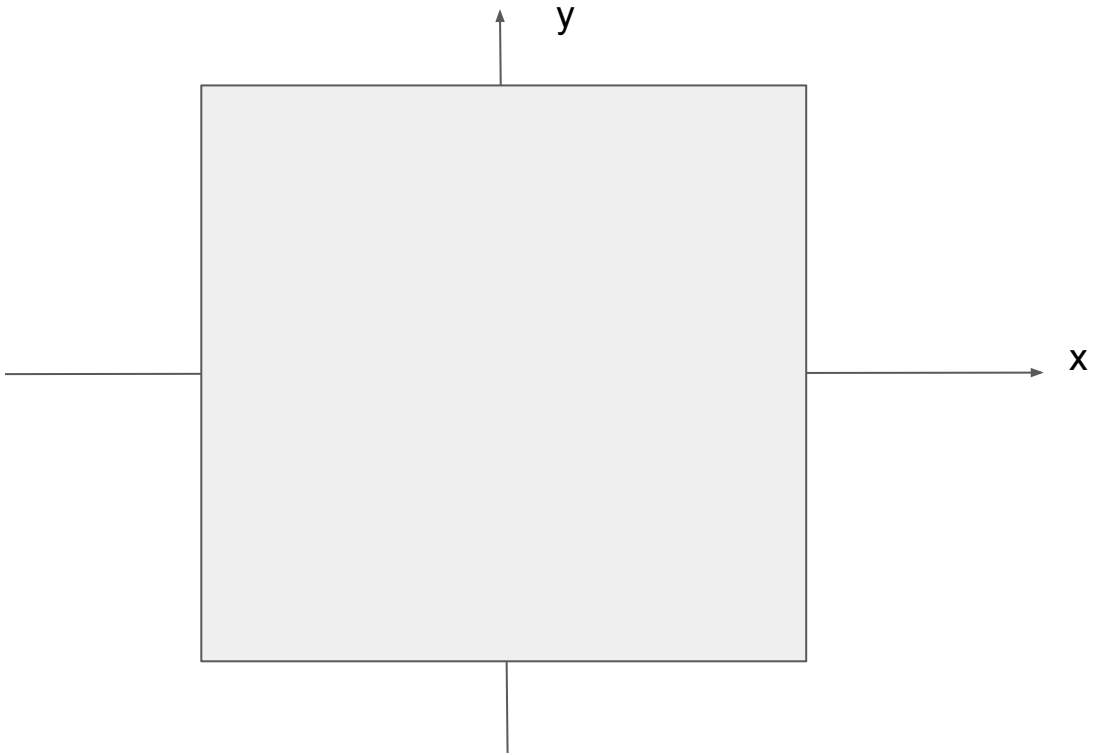
Crtanje tačaka

- Piksele možemo posmatrati kao male četverouglove. Da bi tačke bile nacrtane kao ispunjene kružnice, potrebno je to omogućiti pozivom `glEnable` sa argumentom `GL_POINT_SMOOTH`.

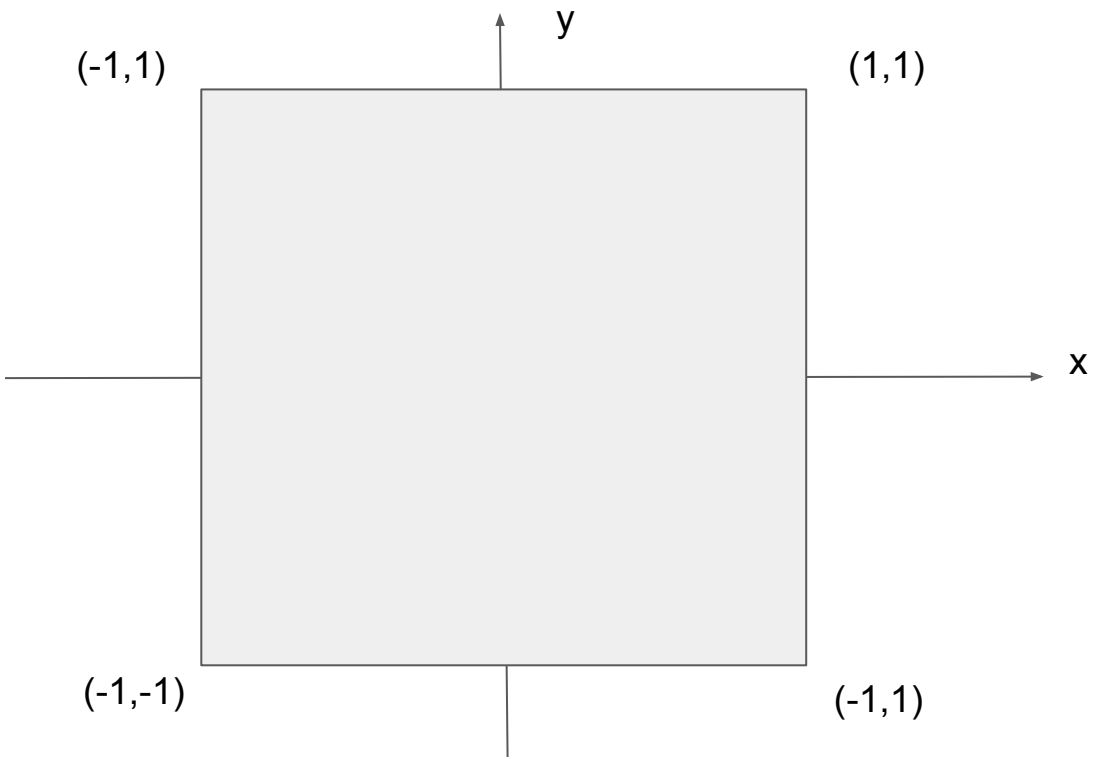
Prostor za crtanje



Prostor za crtanje



Prostor za crtanje



Crtanje nezavisnih i zavisnih tačaka

- Svi pozivi `glVertex*` moraju biti između poziva `glBegin` i `glEnd`, koji zapravo signaliziraju početak i kraj crtanja.

  ```
  void glBegin(GLenum mode);

  void glEnd(void);
  ```

- Parametar `mode` predstavlja jednu od 10 simboličkih konstanti, a definiše način na koji će se specificirane tačke (*vertex*-i) interpretirati.

Crtanje tačaka

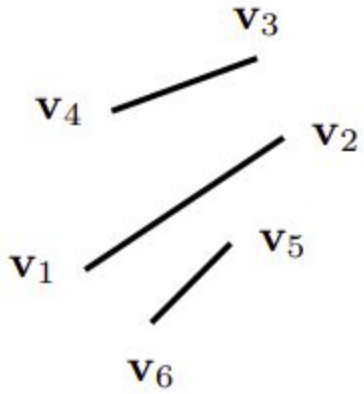
Konstanta	Značenje
`GL_POINTS`	Tretira svaki *vertex* kao zasebnu tačku.

Primjer 1

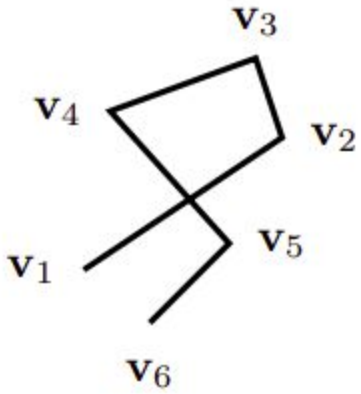
Crtanje linija

Konstanta	Značenje
`GL_LINES`	Tretira parove *vertex*-a kao nezavisne linije. Parove kreira od uzastopnih *vertex*-a.
`GL_LINE_STRIP`	Radi kao GL_LINES, ali spaja linije medjusobno.
`GL_LINE_LOOP`	Radi kao GL_LINE_STRIP, ali spaja prvu i posljednju liniju, čime efektivno kreira poligon.

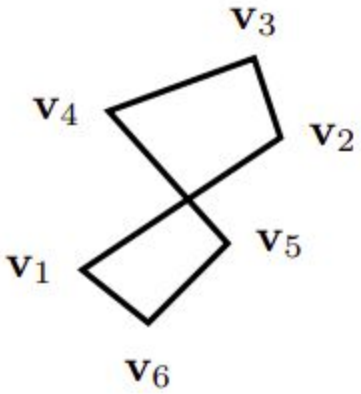
Crtanje linija



GL_LINES



GL_LINE_STRIP



GL_LINE_LOOP

Crtanje linija

- Pri crtanju linija, moguće je zadati debljinu linije.

```
void glLineWidth(GLfloat width);
```

- *Default* vrijednost za debljinu linije je 1.0.
- Validne vrijednosti za parametar `width` su veće od 0.0.
- Poziv ove funkcije se mora obaviti prije poziva `glBegin`.

Crtanje linija

- Pri crtanju linija, moguće je zadati i stil linije.

```
void glLineStipple(GLint factor, GLushort pattern);
```

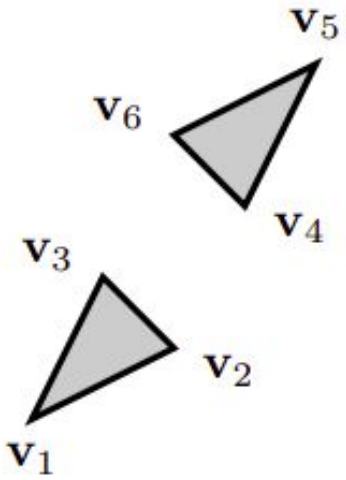
- *Default* vrijednosti za parametre su 1 i 0xFFFF, respektivno.
- Drugi parametar specificira *pattern* koji određuje koji piksel će se iscrtati, a koji ne.
- Prvi parametar specificira množilac svakog bita u *pattern*-u.
- Poziv ove funkcije se mora obaviti prije poziva `glBegin`.
- Potrebno uključiti ovu funkcionalnost pozivom `glEnable(GL_LINE_STIPPLE)`.

Primjer 2

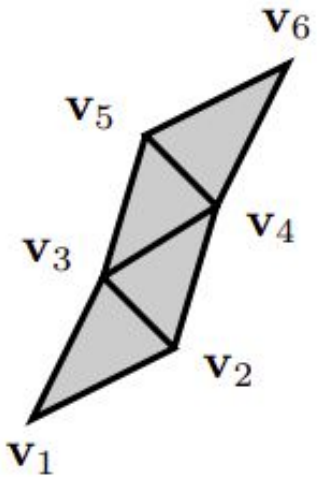
Crtanje trouglova

Konstanta	Značenje
`GL_TRIANGLES`	Tretira trojke *vertex*-a kao nezavisne trouglove. Trojke kreira od uzastopnih *vertex*-a.
`GL_TRIANGLE_STRIP`	Radi kao GL_TRIANGLES, ali spaja trouglove medjusobno.
`GL_TRIANGLE_FAN`	Radi slično kao GL_TRIANGLE_STRIP, ali definiše prvu tačku kao zajedničku za sve trouglove.

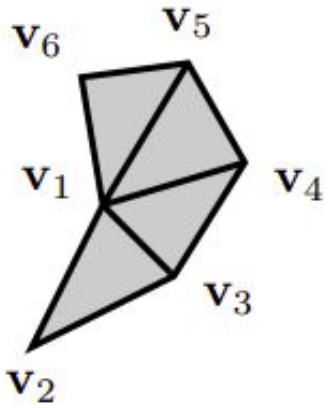
Crtanje trouglova



GL_TRIANGLES



GL_TRIANGLE_STRIP

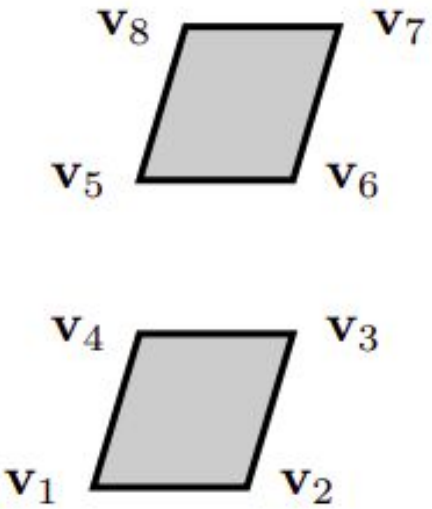


GL_TRIANGLE_FAN

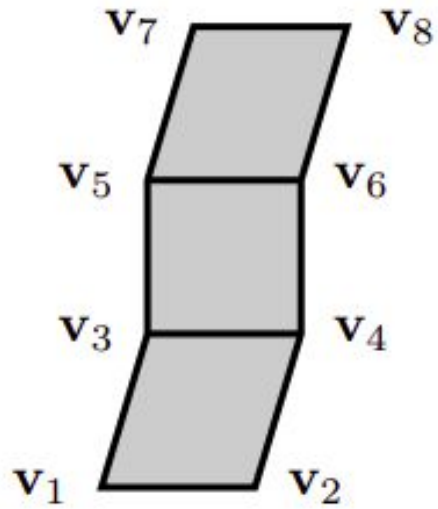
Crtanje četverouglova

Konstanta	Značenje
`GL_QUADS`	Kreira četvorougao od svaka četiri uzastopno definisana *vertex*-a, povezujući ih suprotno od smjera kazaljke na satu.
`GL_QUAD_STRIP`	Radi slično kao GL_QUADS, ali spaja četverouglove medjusobno. Osim toga, tačke uvijek povezuje kao parove sa lijeva na desno.

Crtanje četverouglova



GL_QUADS

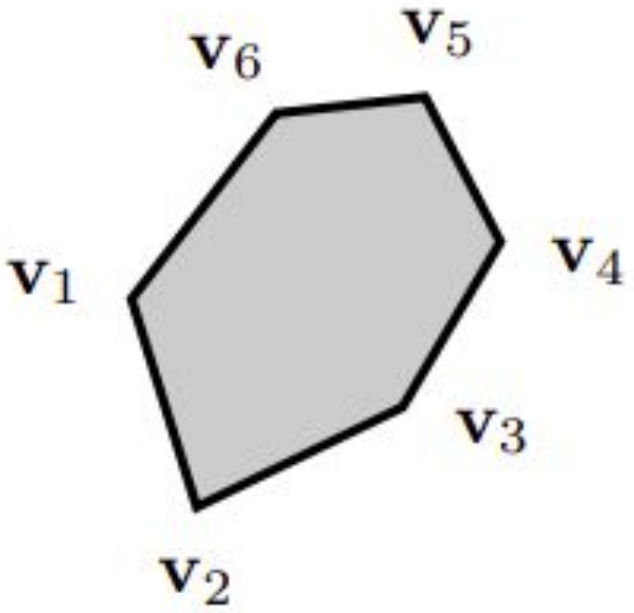


GL_QUAD_STRIP

Crtanje poligona

Konstanta	Značenje
`GL_POLYGON`	Kreira poligon on svih *vertex*-a definisanih do poziva `glEnd()`.

Crtanje poligona



Primjer 3

Crtanje poligona

- Pri crtanju poligona, moguće je specificirati način crtanja

```
void glPolygonMode(GLenum face, GLenum mode);
```

- Prvi parametar ćemo zasada zanemariti, jer ima smisla samo kada postoji više pogleda na isti objekat.
- Drugi parametar može biti jedna od tri konstante koja određuje kako će se poligon iscrtati.
- Ovu funkciju je potrebno pozvati prije `glBegin`.

Crtanje poligona

Konstanta	Značenje
`GL_FILL`	*Default* način crtanja. Sadržaj poligona je ispunjen određenom bojom.
`GL_POINT`	Iscrtat će se samo vrhovi poligona.
`GL_LINE`	Iscrtat će se vrhovi poligona povezani linijama (poligon je prazan).

Primjer 4

Hvala na pažnji!