ELEKTROTEHNIČKI FAKULTET UNIVERZITETA U BEOGRADU

IMPLEMENTACIJA BIBLIOTEKE ZA GENERISANJE I PRIKAZIVANJE GRAFIKA PRIMENOM HTML5 I JAVASCRIPT JEZIKA

–Master rad–

Kandidat:

Miloš Šuleić 2012/3380

Mentor:

doc. dr ZoranČiča

Beograd, Oktobar 2015.

SADRŽAJ

1. UVOD

HTML5, kao standard, obuhvatio je tri nezavisne celine: HTML, CSS3 i JavaScript, time omogućujući veću integraciju i mogućnosti svake od tehnologija. Jedna od novih funkcionalnosti koje je ovaj standard doneo je *API* kojim je moguće crtati grafičke elemente direktno na HTML stranici. Element koji ovo omogućava nazvan je *Canvas.*

Iako se implementacije u nekom obimu razlikuju, svi najveći internet pregledači podržavaju *Canvas* element. Na njemu je moguće iscrtavati dvodimenzionalne, ali i trodimenzionalne objekte, uz pomoć *JavaScript-a.*

Tema ovog rada biće kreiranje *JavaScript* biblioteke za crtanje 2D i 3D grafika matematičkih funkcija, upravo korišćenjem *Canvas API*-ja. Na taj način, korisnik će lako moći da uključivanjem biblioteke u svoj sajt, iscrta na njemu funkcije visokog stepena kompleksnosti.

Ovaj rad će biti podeljen na šest poglavlja. U prvom, uvodnom poglavlju, ukazano je na oblast, cilj, motivaciju i doprinos rada. Drugo poglavlje opisuje korišćene funkcionalnosti *Canvas* elementa, uz date primere za svaku od tih funkcionalnosti. Treće poglavlje predstavlja opis koda biblioteke za crtanje grafika. U četvrtom poglavlju dati su primeri kompletnih HTML stranica na kojima su generisani grafici matematičkih funkcija korišćenjem kreirane biblioteke.. Šesto poglavlje predstavlja zaključak, u kome je izložena potencijalna korist i primena ove biblioteke. U sedmom poglavlju, kao prilog, dat je kompletan programski kod biblioteke.

2. *CANVAS* ELEMENT

HTML5 Canvas element se unutar *HTML* stranice koristi kao i drugi *tag*-ovi, kao što su npr.. `<div>`, `<a>`ili`<p>`. Ipak, razlika je u tome što se za iscrtavanje grafičkih elemenata na njemu koristi*JavaScript*jezik. *Canvas* element se poziva na sledeći način:

```
<canvasid="canvas"width="1280"height="800">
</canvas>
```

Ovim se definiše oblast unutar stranice, sa zadatom širinom (`width` atribut) i visinom (`height` atribut).

U slučaju da internet pregledač ne podržava ovaj element, moguće je postaviti odgovarajuću poruku jednostavnim upisivanjem teksta unutar tag-a:

```
<canvasid="canvas"width="1280"height="800">
Vaš internet pregledač ne podržava Canvas tag!
</canvas>
```

Ova poruka bi bila prikazana na mestu gde bi se nalazio sam element, dok bi ostatak stranice bio interpretiran na uobičajen način.

2.1. Kontekst

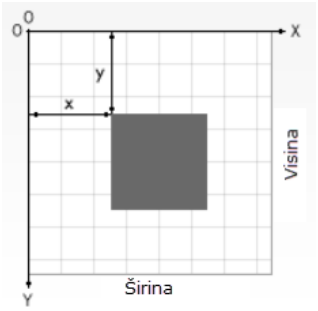
Pozivanjem *Canvas* elementa, kreira se površina zadatih dimenzija, na kojoj je moguće iscrtati različite grafičke elemente. Da bi se to postiglo,je pristupiti tzv kontekstu u kome su definisanje same naredbe za iscrtavanje grafičkih elemenata. U ovom radu će biti opisan i korišćen “2d” kontekst. Pored njega, postoji i “webgl” (3d) kontekst, koji je baziran na *OpenGL ES* tehnologiji.

Kontekst se, koristeći *JavaScript*, poziva na sledeći način:

```
var canvas = document.getElementById('canvas');
var kontekst = canvas.getContext('2d');
```

2.2. Koordinatni sistem

Canvas element je određen dvodimenzionalnim koordinatnim sistemom, sa početkom u gornjem levom uglu, čije su koordinate (0,0). Pozitivan smer x-ose je udesno, a y-ose na dole. Ovo je prikazano na slici 2.2.1.



Slika 2.2.1 Koordinatni sistem

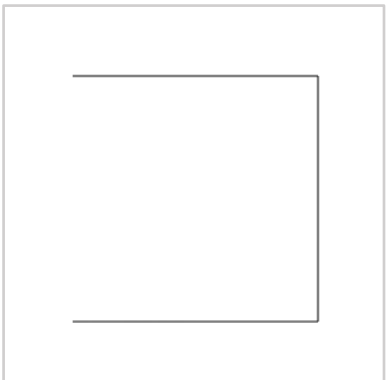
2.3. Linije i putanje

Za iscrtavanje linije, koriste se dve metode 2d konteksta:

- `moveTo(x1,y1)` – kojom se postavlja početna tačka iscrtavanja linije
- `LineTo(x2,y2)` – kojom se postavlja krajnja tačka linije

Da bi linija bila i iscrtana na ekranu, potrebno je pozvati i treću metodu - `stroke()`. Pozivanje motode `LineTo()`još jednom bi definisalo krajnju tačku nove linije,dok bi njena početna tačka bila krajnja prethodne linije. Na taj način se formiraju tzv. putanje.Jedan primer crtanja putanje je dat na slici 2.3.1.

```
<!DOCTYPE html>
<html>
<body>
<canvasid="canvas"width="300"height="300">
</canvas>
<script>
var canvas = document.getElementById('canvas');
var kontekst = canvas.getContext('2d');
kontekst.moveTo(50,50);
kontekst.lineTo(250,50);
kontekst.lineTo(250,250);
kontekst.lineTo(50,250);
kontekst.stroke();
</script>
</body>
</html>
```



Slika 2.3.1. Primer iscrtavanja putanje

Putanja može započeti metodom `beginPath()`.U ovom slucaju, metoda `stroke()`će iscrtati samo one grafičke elemente pozvane nakon metode `beginPath()`. Ovo je ilustrovano na slici 2.3.2

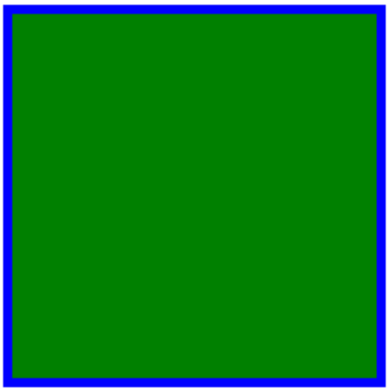
```
<!DOCTYPE html>
<html>
<body>
<canvasid="canvas"width="300"height="300">
</canvas>
<script>
var canvas = document.getElementById('canvas');
var kontekst = canvas.getContext('2d');
kontekst.moveTo(50,50);
kontekst.lineTo(250,50);
kontekst.beginPath();
kontekst.lineTo(250,250);
kontekst.lineTo(50,250);
kontekst.stroke();
</script>
</body>
</html>
```


Slika 2.3.2 Upotreba `beginPath()` metode

Putanju je opciono moguće zatvoriti metodom `closePath()`, čime se pravom linijom spajaju početna i krajnja tačka. Ovako zatvorena putanja se može obojiti metodom `fill()`.Sama boja se može izabrati definisanjem osobine `fillStyle`. Takođe, boja linije se može izabrati definisanjem osobine`strokeStyle` a njena debljina osobinom `lineWidth` , kao što je prikazano na slici 2.3.3.

```
<!DOCTYPE html>
<html>
<body>
<canvasid="canvas"width="300"height="300">
</canvas>
<script>
var canvas = document.getElementById('canvas');
var kontekst = canvas.getContext('2d');
kontekst.beginPath();
kontekst.moveTo(50,50);
kontekst.lineTo(250,50);
kontekst.lineTo(250,250);
kontekst.lineTo(50,250);
kontekst.closePath();
kontekst.strokeStyle="blue";
kontekst.lineWidth=10;
```

```
kontekst.stroke();
kontekst.fillStyle="green";
kontekst.fill();
</script>
</body>
</html>
```



Slika 2.3.3. Iscrtavanje i ispunjavanje zatvorene putanje

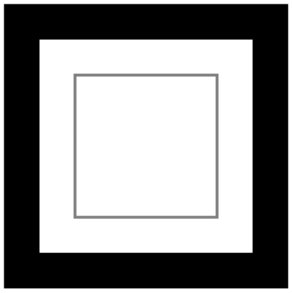
2.4. Pravougaonici

Za iscrtavanje pravougaonika, koriste se tri metode:

- `StrokeRect(x,y,širina,visina)` – kojom se iscrtava okvir pravougaonika
- `fillRect(x,y,širina,visina)` – kojom se ispunjava površina pravougaonika
- `clearRect(x,y,širina,visina)` – kojom se briše definisana pravougaona oblast

Pritom, x i y definišu gornju levu tačku pravougaonika. Upotreba ovih metoda ilustrovana je na sledećem primeru:

```
<!DOCTYPE html>
<html>
<body>
<canvasid="canvas"width="300"height="300">
</canvas>
<script>
var canvas = document.getElementById('canvas');
var kontekst = canvas.getContext('2d');
kontekst.fillRect(50,50,200,200);
kontekst.clearRect(75,75,150,150);
kontekst.strokeRect(100,100,100,100);
</script>
</body>
</html>
```



Slika 2.4.1 Primer upotrebe metoda za iscrtavanje pravougaonika

Dakle, metodom `fillRect(50,50,200,200)`, ispunjena je kvadratna površina dimenzija 200x200. Zatim je metodom `clearRect(75,75,150,150)` obrisan deo njene unutrašnjosti, te unutar nje metodom `strokeRect(100,100,100,100)` ucrtan okvir manjeg kvadrata, dimenzija 100x100.

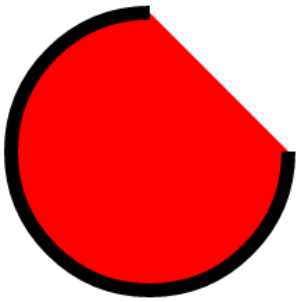
2.5. Krugovi

Krug se iscrtava metodom **`arc()` na sledeći način:**

`arc(x,y,poluprečnik,početniUgao,krajnjiUgao,`*`anticlockwise`*`)`

Definisanjem ugaonog opsega manjeg od 360 stepeni može se iscrtati samo deo kružnice. U ovom slučaju, parametar ***`anticlockwise`***određuje da se kružnica iscrtava u smeru suprotnom od smera kazaljke na satu. Na slica 2.5.1. dat je primer iscrtavanja dela kružnice.

```
<!DOCTYPE html>
<html>
<body>
<canvasid="canvas"width="300"height="300">
</canvas>
<script>
var canvas = document.getElementById('canvas');
var kontekst = canvas.getContext('2d');
kontekst.arc(150,150,100,0,3/2*Math.PI);
kontekst.fillStyle="red";
kontekst.lineWidth=10;
kontekst.fill();
kontekst.stroke();
</script>
</body>
</html>
```



Slika 2.5.1. Iscrtavanje (dela) kružnice

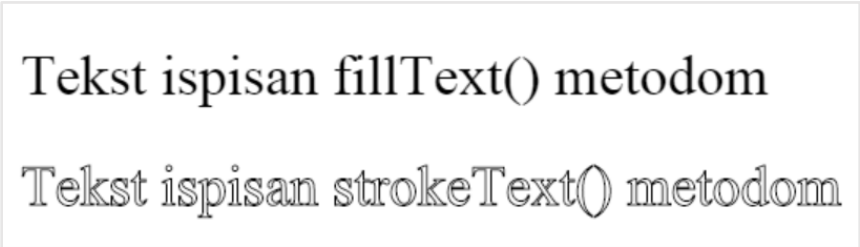
2.6. Tekst

Tekst se može ispisati pomoću dve metode:

- **`fillText(tekst,x,y)`** – ispis punih slova
- **`strokeText(tekst,x,y)`** – ispis samo kontura slova

Primer je dat na slici 2.6.1.

```
<!DOCTYPE html>
<html>
<body>
<canvasid="canvas"width="600"height="300">
</canvas>
<script>
var canvas = document.getElementById('canvas');
var kontekst = canvas.getContext('2d');
kontekst.font ="36px serif";
kontekst.fillText("Tekst ispisan fillText() metodom",10,100);
kontekst.strokeText("Tekst ispisan strokeText() metodom",10,170);
</script>
</body>
</html>
```



Slika 3.6.1. Ispis teksta unutar *Cnnvas* elementa

U prethodnom primeru, tekst je postavljen na veličinu nešto veću od podrazumevane (36px), definisanjem atributa `font`. Ovaj attribut se koristi na isti način kao *CSS Font* osobina.

Dužina teksta, u pikselima, se može izmeriti metodom `measureText()`. Ova metoda kao rezultat vraća objekat, čija osobina `width` sadrži vrednost dužine teksta. Izvršavanjem sledećeg koda

```
<!DOCTYPE html>
<html>
<body>
<canvasid="canvas"width="600"height="300">
</canvas>
<script>
var canvas = document.getElementById('canvas');
var kontekst = canvas.getContext('2d');
kontekst.font ="36px serif";
var tekst = kontekst.measureText("Izmeri dužinu teksta");
alert(tekst.width);
</script>
</body>
</html>
```

dobija se vrednost dužine navedenog teksta:

Slika 2.6.2. Primer korišćenja `measureText()` metode

2.7. Transformacije

2.7.1. Translacija

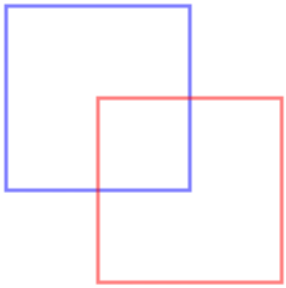
Ovom metodom vrši se pomeranje centra koordinatnog sistema u izabranu tačku. Metoda se koristi na sledeći način:

```
translate(x,y)
```

gde su x i y koordinate na koje se pomera koordinatni početak. Upotreba ove metode prikazana je na sledećem primeru:

```
<!DOCTYPE html>
<html>
<body>
<canvasid="canvas"width="300"height="300">
</canvas>
<script>
var canvas = document.getElementById('canvas');
var kontekst = canvas.getContext('2d');
function nacrtajKvadrat(boja){
 kontekst.strokeStyle=boja;
```

```
        kontekst.strokeRect(50,50,100,100);
}
nacrtajKvadrat("blue");
kontekst.translate(50,50);
nacrtajKvadrat("red");
</script>
</body>
</html>
```



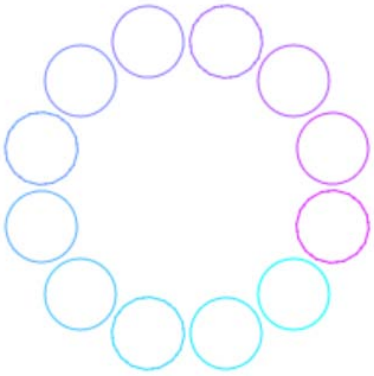
Slika 2.7.1.1. Primer `translate()` metode

Oba kvadrata se iscrtavaju sa početnom tačkom u (50,50). Međutim, pre iscrtavanja crvenog kvadrata, koordinatni početak je pomeren na (50,50), te je on za te vrednosti koordinata pomeren u odnosu na plavi kvadrat.

2.7.2. Rotacija

Metodom `rotate()` vrši se rotiranje koordinatnog sistema oko koordinatnog početka u smeru kazaljke na satu, za ugao dat u radijanima. Naredni primer ilustruje korišćenje ove metode:

```
<!DOCTYPE html>
<html>
<body>
<canvasid="canvas"width="300"height="300">
</canvas>
<script>
var canvas = document.getElementById('canvas');
var kontekst = canvas.getContext('2d');
kontekst.translate(150,150);
for(var i=0;i<12;i++){
 kontekst.beginPath();
 kontekst.strokeStyle="rgb("+(20*i)+","+(255-20*i)+",255)";
 kontekst.arc(60,60,20,0,2*Math.PI);
 kontekst.rotate(Math.PI/6);
 kontekst.stroke();
}
</script>
</body>
</html>
```



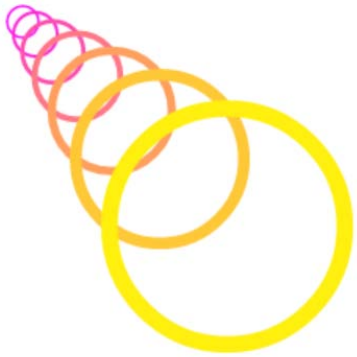
Slika 2.7.2.1. Primer `rotate()` metode

Svi krugovi se iscrtavaju na istoj poziciji, (60,60), ali se svaki sledeći rotira za π/6 radijana. Pritom, koordinatni početak je prethodno transliran na centar *Canvas* elementa.

2.7.3. Skaliranje

Metodom `scale(x,y)`vrši se promena dimenzija osnovnih jedinica *Canvas* elementa. Naime, podrazumevana jedinica veličine je jedan piksel. Međutim, korišćenjem ove metode, ta jedinica veličine može biti manja ili veća. Upotreba metode biće objašnjena na sledećem primeru.

```
<!DOCTYPE html>
<html>
<body>
<canvasid="canvas"width="300"height="300">
</canvas>
<script>
var canvas = document.getElementById('canvas');
var kontekst = canvas.getContext('2d');
kontekst.translate(10,10);
for(var i=0;i<7;i++){
      kontekst.beginPath();
      kontekst.strokeStyle="rgb(255,"+(40*i)+","+(255-40*i)+")";
      kontekst.arc(20,20,10,0,2*Math.PI);
      kontekst.scale(1.4,1.4);
      kontekst.stroke();
}
</script>
</body>
</html>
```



Slika 2.7.3.1. Primer korišćenja `scale()` metode

Svi krugovi se iscrtavaju sa centrom u (20,20) i poluprečnikom 10. Ipak, svaki sledeći se iscrtava u koordinatnom sistemu skaliranom faktorom 1.4. Stoga, ovo skaliranje utiče na svaki aspekt iscrtanih krugova – koordinate centra, poluprečnik, ali i debljinu kružnice.

3. OPIS *JAVASCRIPT* BIBLIOTEKE ZA ISCRTAVANJE GRAFIKA

Koristeći *Canvas* element opisan u prethodnom poglavlju, u ovom će biti opisana realizacija biblioteke, pomoću koje će na jednostavan način moći da se iscrtavaju grafici matematičkih funkcija unutar HTML stranice.

Osnovne funkcionalnosti koje će biblioteka imati su iscrtavanje 3D grafika (plot3), 2D grafika (plot), kao i grafičko prikazivanje diskretnih sekvenci podataka (stem metoda).

3.1. Pregled 3D matrica transformacija

Kao što je već napomenuto u poglavlju 2, za realizaciju iscrtavanja grafika biće korišćen 2d kontekst *Canvas* elementa. Stoga, neophodno je definisati i realizovati matematički model kojim će biti omogućeno prikaz trodimenzionalnih objekata na dvodimenzionalnoj površini.

U tu svrhu, tačke Euklidskog koordinatnog sistema biće predstavljene tzv. homogenim koordinatama. Naime, svaka tačka (x,y,z) Euklidskog 3D prostora može se predstaviti u homogenim koordinatama kao (x1,y1,z1,w), tako da je x=x1/w, y=y1/w i z=z1/w. Tako, za realizacuju transformacija tačaka u 3D prostoru (translacije, rotacije i skairanja), moguće je koristiti množenje matrica.

Matrica translacije u homogenim koordinatama se može predstaviti kao:

$$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translacija jedne tačke se sada može dobiti množenjem te tačke sa transformacijom:

$$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ z + t_z \\ 1 \end{bmatrix}$$

Rotacija tačke se može definisati kao rotacija oko svake ose pojedinačno. Matrica rotacije oko x-ose je data kao:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & -sin\theta & 0 \\ 0 & sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

oko y-ose kao:

$$\begin{bmatrix} cos\phi & sin\phi & 0 & 0 \\ -sin\phi & cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

i konačno oko z-ose:

$$\begin{bmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Skaliranje se vrši množenjem tačke sa matricom skaliranja:

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Na kraju, neophodno je definisati i mehanizam kojim bi trodimenzionalni objekti bili prikazani na dvodimenzionalnoj ravni. Ovo se postiže tzv. perspektivnom projekcijom, kojom se vrši projekcija tačaka iz 3D prostora na 2D ravan. Ovom projekcijom, tačke se projektuju duž linija koje proističi iz jedne tačke, nazvanom tačka projekcije. To znači da će objekti biti manji ukoliko se projektuju dalje od centra projekcije i obrnuto.

Ovaj tip projekcije može se definisati sledećom matricom:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

3.2. Definisanje koordinatnog sistema

Sama biblioteka će biti realizovana kao objekat čijim instanciranjem, definisanjem osobina i pozivanjem metoda se mogu koristiti sve njene funkcionalnosti, na sledeći način:

```
window.Grafik =function(canvas){
```

```
. . .
}
```

Nakon toga, definiše se kontekst *Canvas* elementa:

```
var kontekst = canvas.getContext("2d");
```

Radi jednostavnijeg definisanja koordinata, koordinatni sistem *Canvas* elementa će biti modifikovan tako da se koordinatni početak translira na njegov centar, a opseg vrednosti x i y koordinata bude između -1 i 1, nezavisno od definisanih dimenzija tog elementa:

```
kontekst.translate(sc/2, vc/2)
kontekst.scale(sc/2,-vc/2)
kontekst.lineWidth=2/vc;
```

gde su sc i vc širina i visina *Canvas* elementa:

```
var sc = canvas.width;
var vc = canvas.height;;
```

3.3. Implementacija 3D transformacija

Svaka transformaciona matrica definisana u poglavlju 3.1. biće definisana zasebnom funkcijom.

Translacija je implementirana na sledeći način:

```
function transliraj(x, y, z){
return[[1,0,0, x],[0,1,0, y],[0,0,1, z]]
}
```

Rotiranje oko x, y, odnosno z-ose respektivno, implementirano je kao:

```
// rotiranje po x-osi, vraća prve tri vrste matrice
function rotirajOkoX(theta){
theta=theta/180*Math.PI;
var s = Math.sin(theta)
var c = Math.cos(theta)
return[[1,0,0,0],[0, c,-s,0],[0, s, c,0]]
}
// rotiranje po y-osi, vraća prve tri vrste matrice
function rotirajOkoY(theta){
theta=theta/180*Math.PI;
var s = Math.sin(theta)
var c = Math.cos(theta)
return[[c,0,-s,0],[0,1,0,0],[s,0, c,0]]
}

// rotiranje po z-osi, vraća prve tri vrste matrice
function rotirajOkoZ(theta){
theta=theta/180*Math.PI;
var s = Math.sin(theta)
```

```
    var c = Math.cos(theta)
    return[[c,-s,0,0],[s, c,0,0],[0,0,1,0]]
    }
```

Funkcije vraćaju samo prve tri vrste matrice, jer je četvrta uvek `[0,0,0,1].`

Skaliranje kao posebnu funkciju nije bilo potrebno implementirati. Naime, pomeranjem centra projekcije dalje od, ili bliže ka 3D objektu dobija se isti efekat, te je ova funklcionalnost tako i realizovana.

Dalje, potrebno je realizovato operacije množenja matrice i tačke, množenje niza tačaka s matricom, kao i množenja dve i više matrica.

Množenje tačke i transformacije je prikazano sledećim kodom:

```
    function transormacijaXTacka(transform, p){
    var rezultat=[]
    for(var ii =0; ii <3; ii++){
    var rv = transform[ii][3]
    for(var jj =0; jj <3; jj++) rv += transform[ii][jj]* p[jj]
        rezult.push(rv)
    }
    return rezultat
    }
```

Ulazni parametri ove funkcije su matrica transformacije i tacka kao niz [x,y,z]. dok je rezultat transformisana tačka [x1,y1,z1].

Koristeći ovu funkciju, realizovana je naredna koja množi niz tačaka s transformacijom:

```
    function transormacijaXNizTacaka(xf, tacke){
    var xp =[]
    for(var ii =0; ii < tacke.length; ii++){
        xp.push(transormacijaXTacka(xf, tacke[ii]))
    }
    return xp
}
```

Rezultat je transformisani niz tačaka.

Množenje dve matrice transformacija implementirano je ugnježđivanjem više **for** petlji:

```
    function mnozenjeMatrica(x1, x2){
var rv =[[0,0,0,0],[0,0,0,0],[0,0,0,0]]
for(var ii =0; ii <3; ii++){
rv[ii][3]= x2[ii][3]
for(var jj =0; jj <3; jj++){
      rv[ii][3]+= x1[jj][3]* x2[ii][jj]
for(var kk =0; kk <3; kk++){
        rv[ii][jj]+= x1[kk][jj]* x2[ii][kk]
}
}
}
return rv
}
```

Kao što je već napomenuto rezultat su prve tri vrste matrice rezultujuće transformacije.

Koristeći funkciju množenja dve matrice, realizovana je funkcija množenja proizvoljnog broja matrica na sledeći način:

```
function mnozenjeMatrica_n(nizMatrica){
var rv = jedinicnaMatrica();
for(var ii =0; ii < nizMatrica.length; ii++) rv =
mnozenjeMatrica(rv,nizMatrica[ii])
return rv
}
```

Konačno, perspektivna projekcija se izvršava sledećom funkcijom:

```
function persp(p){
      return[(p[0]-dx)/(p[1]-dy)*6,(p[2]-dz)/(p[1]-dy)*6]
}
```

Promenjive dx, dy i dz su koordinate centra projekcije, odnosno pomeraj centra projekcije u odnosu na koordinatni početak.

3.4. Iscrtavanje 3D gradika – plot3

Sve tri glavne funkcionalnosti biblioteke (plot3, plot i stem) biće realizvane kao metode objekta Graph. Svi parametri, uključujući i samu funkciju koja se iscrtava, domen, uglovi rotacije, zum, biće osobine instance tog objekta. Na taj način, svi oni će moći biti lako pozivani i menjani iz glavnog programa.

Metoda plot3 se definiše na sledeći način:

```
this.plot3 =function(){
var grafik=this;
...
}
```

3.4.1. Rotacija i zumiranje

Osnovni parametri rotacije grafika biće uglovi rotacije oko x, odnosno z-ose. Ove vrednosti se upisuju `grafik.rotX`, odnosno `grafik.rotZ`osobine. Ove osobine je moguće promeniti iz glavnog programa i tako rotirati grafik. Slicno je i sa osobinom `grafik.zoom` , kojom se može postaviti vrednost zumiranja grafika (podrazumevana vrednost je 100).

Medjutim, takođe će biti implementirana i rotacija grafika klikom i povlačenjem miša, kao i zumiranje okretanjem skrol dugmeta na mišu.

Da bi se ovo postiglo, najpre je potrebno postaviti tzv. *EventListener*-enakog čega će sedetektovati određene akcije miša nad *Canvas* elementom. Ovo se postiže `addEventListener()` metodom na sledeći način:

```
canvas.addEventListener("mousemove",function(e){
                        nadjiXY('move', e)
```

```
                        },false);
canvas.addEventListener("touchmove",function(e){
                        nadjiXY('tmove', e)
                        },false);
canvas.addEventListener("mousedown",function(e){
                        nadjiXY('down', e)
                        },false);
canvas.addEventListener("mouseup",function(e){
                        nadjiXY('up', e)
                        },false);
canvas.addEventListener("mouseout",function(e){
                        nadjiXY('out', e)
                        },false);
canvas.addEventListener("click",function(e){
                        nadjiXY('click', e)
                        },false);
canvas.addEventListener("dblclick",function(e){
                        nadjiXY('dblclick', e)
                        },false);

// IE9+, Chrome, Safari, Opera
canvas.addEventListener("mousewheel", skrolMisFukcija,false);
// Firefox
canvas.addEventListener("DOMMouseScroll",skrolMisFukcija,false);
```

Za rotaciju, sledeći korak je realizovati da se grafik rotira kada je pritisnuto levo dugme miša koji se pomera. Stoga, kada je dugme pritisnuto, promenjiva flag se postavlja na *true*. Nakon toga se proverava da li se miš pomera i ukoliko da, poziva se funkcija rotacije:

```
function nadjiXY(res, e){
if(res =='down'){
        prethodniX = trenutniX;
        prethodniY = trenutniY;
        trenutniX = e.clientX - canvas.offsetLeft;
        trenutniY = e.clientY - canvas.offsetTop;

        flag =true;

}

if(res =='up'|| res =="out"){
        flag =false;
}
if(res =='move'){
if(flag){
            prethodniX = trenutniX;
            prethodniY = trenutniY;
            trenutniX = e.clientX - canvas.offsetLeft;
            trenutniY = e.clientY - canvas.offsetTop;
                  rot();
}
      }
```

Funkcija rotacije`rot()` u odnosu na pomeraj miša, modifikuje osobine grafik.rotX i grafik.rotY, nakon čega poziva iscrtavanje određenog tipa grafika, koji će biti opisani u daljem tekstu.

```
function rot(){
```

```
var rx1=(trenutniY-prethodniY)/2
var rz1=(trenutniX-prethodniX)/2;
grafik.rotX+=rx1;
grafik.rotZ+=rz1;
if(grafik.type=='plot'){grafik.plot()}
elseif(grafik.type=='plot3'){grafik.plot3()}
elseif(grafik.type=='stem'){grafik.stem()}
}
```

Slično kai i za rotaciju, sledeća funkcija vrši promenu parametra grafik.zoom prilikom okretanja skrol dugmeta, nakon čega poziva ponovno iscrtavanje funkcije:

```
function skrolMisFukcija(e){
var e = window.event || e;
if(e.preventDefault) e.preventDefault();
var delta = Math.max(-1, Math.min(1,(e.wheelDelta ||-e.detail)));

grafik.zoom=Math.round(Math.min(300,Math.max(10,grafik.zoom+grafik.zoom*0.
1*delta)));
if(grafik.type=='plot'){grafik.plot()}
elseif(grafik.type=='plot3'){grafik.plot3()}
elseif(grafik.type=='stem'){grafik.stem()}

returnfalse;
}
```

3.4.2. Izračunavanje vrednosti funkcija i iscrtavanje njihovih grafika

Nezavisno promenljiva x se unosi preko tri osobine:

- xmin –donja granica opsega promenjive x
- xmax – gornja granica opsega promenjive x
- korak – raspon izmedju vrednosti x za koje se računaju i iscratavaju vrednosti samih funkcija

Drugi način unošenja vrednosti je preko osobine x, u formatu “xmin:korak:xmax”. Funkcija koja proverava na koji način su unete vrednosti i vrši njihovo parsiranje u odgovarajući format je prikazana ispod:

```
if(typeof grafik.xmin=="undefined"||
      typeof grafik.xmax=="undefined"||
      typeof grafik.korak=="undefined"){
      var x_niz=grafik.x.split(":");
      xmin=parseFloat(x_niz[0]);
      korak_x=parseFloat(x_niz[1]);
      xmax=parseFloat(x_niz[2]);
      grafik.xmin=xmin;
      grafik.xmax=xmax;
      grafik.korak=korak_x;
}else{
      if(xmin!=grafik.xmin||
            xmax!=grafik.xmax||
            korak_x!=grafik.korak){
            grafik.selX="";
```

```
            grafik.selY="";
            grafik.selZ="";
            nt=false}
      xmin=parseFloat(grafik.xmin);
      xmax=parseFloat(grafik.xmax);
      korak_x=parseFloat(grafik.korak);}
```

Dakle, program dalje upotrebljava promenljive xmin, xmax i korak_x. Osobine selX, selY, selZ, kao i promenljiva nt se postavljaju kao indikatori neophodni za pozivanje funkcije izbora tačke na grafiku, koja će biti opisana kasnije.

Same funkcije čiji grafik se iscrtava se upisuju u *JavaScript* formatu u osobine x i y instanciranog objekta, što su u ovom slučaju grafik.y i grafik.z. Te funkcije biblioteka evaluira na sledeći način:

```
function funkcijaY(x){
  return eval(grafik.y);
};
```

Da bi se izvršilo skaliranje vrednosti funkcija u odnosu na veličinu grafika, potrebno je izračunati maksimum i minimum svake funkcije.

Funkcija koja računa maksimu implementirana je po *bubble sort* algoritmu i data je ispod:

```
function maksFje(mxfja){

      var x=xmin;
      var ymaxset=false;

      while(x<=xmax){

      var y=mxfja(x);

      if(isFinite(y)){
            if(ymaxset){
                  if(y>ymax){ymax=y}
                  }else{
                              var ymax=y;
                              ymaxset=true;
                  }
            }
      x+=korak_x;
      }

      return ymax;
}
```

Ulazni parametar je funkcija čiji se maksimum traži.

Na sličan način je implementirana i funkcija za računanje minimuma matematičke funkcije koja se iscrtava:

```
function minFje(mnfja){

var x=xmin;
var yminset=false;
while(x<=xmax){

      var y=mnfja(x);
```

```
            if(isFinite(y)){
                  if(yminset){
                        if(y<ymin){ymin=y}
                  }else{
                        var ymin=y;
                        yminset=true;
                  }
            }
      x+=korak_x;
      }

      return ymin;
}
```

Dalje, za svako x, vrši se računanje vrednosti funkcija y i z:

```
            while(x<=xmax){
            var y=funkcijaY(x);
            var z=funkcijaZ(x);
```

Zatim se vrši skaliranje vrednosti x, y i z. Koordinatni sistem koji će biti iscrtan, ima centar u tački (0,0,0) u 3D prostoru, a grafici se skaliraju tako da budu u opsegu od 0 do 1 po sve tri ose.Potom se nad njima, ukoliko je potrebno, vrše dodatne transformacije (transliranje i rotacija). Na kraju, vrši se projekcija na 2D ravan *Canvas* elementa. U nastavku je data implementacija navedenih koraka.

```
      // Upisivanje originalnih tačaka u niz
      koordGraf3d.push([x,y,z]);
      // Skaliranje vrednosti u opseg od 0 do 1
      grafTacka[0]=(x-xmin)/(xmax-xmin),
      grafTacka[1]=(y-ymin)/(ymax-ymin),
      grafTacka[2]=(z-zmin)/(zmax-zmin);
      // transformacije nad tačkama grafika
      grafTacka=transormacijaXTacka(mnozenjeMatrica_n(transformacije),
      grafTacka);
      // Perspektivna projekcija
      var xy2d=persp([grafTacka[0],grafTacka[1],grafTacka[2]]);
```

Pritom, sledeći kod određuje navedene transformacije:

```
      var transformacije=[transliraj(-0.5,-0.5,-0.5),
                  rotirajOkoZ(0+grafik.rotZ),
                        rotirajOkoX(0+grafik.rotX),
                        transliraj(0.5,0.5,0.5)];
```

gde su grafik.rotX, odnosno grafik.rotZ osobine koje u koje je upisan trenutni ugao rotacije grafika oko odgovarajuće ose.

Nakon toga, dolazi do iscrtavanja grafika sledećim kodom:

```
      //    Pozicioniranje za početak iscrtavanja grafika
      if(pocetak){
            if(isFinite(y)&&!isNaN(y)&&isFinite(z)&&!isNaN(z)){
                  kontekst.moveTo(xy2d[0],xy2d[1]);

                  pocetak=false;
```

```
        }
//      Ukoliko je vrednost funkcije beskonačna,
//      sprečava se iscrtavanje asimptote
}elseif(!pocetak&&(!isFinite(y)||!isFinite(z))){
        asimptota=true;
}else{
        if(asimptota){
        kontekst.moveTo(xy2d[0],xy2d[1]);
        asimptota=false;
        }else{
        //    Iscrtavanje grafika
        kontekst.strokeStyle ='red';
        kontekst.lineTo(xy2d[0],xy2d[1]);

        }
}

        x+=korak_x;
```

U ovom delu koda vrši se provera da li je vrednost neke od funkcije beskonačna, te se u tom slučaju iscrtavanje te tačke potpuno preskače. Iscrtavanje se vrši od tačke do tačke, pravom linijom.

Konačno, posle izlaska iz **while**petlje, samo iscrtavanje se vrši komandom

```
kontekst.stroke();
```

3.4.3. Iscrtavanje koordinatnog sistema

Koordinatni sistem će biti definisan u 3D prostoru u opsegu od 0 do 1, po sve tri ose, u formi kvadrata. Za početak, neophodno je definisati sve koordinate temena ovog kvardata:

```
koordTacke =[[0,0,0],[1,0,0],[1,1,0],[0,1,0],
            [0,0,1],[1,0,1],[1,1,1],[0,1,1]];
```

Zatim, mapiraju se linije kao niz od kojih će svaka biti određena sa dve tačke prethodnog niza:

```
mapaLinija =[[0,1],[4,5],[6,7],[2,3],
            [0,3],[1,2],[5,6],[4,7],
            [0,4],[1,5],[2,6],[3,7]];
```

Pre iscrtavanja linija, potrebno je definisati da li će se iscrtavati puna ili crtičasta linija. Naime, pune linije će biti samo one na kojima će biti ispisani podeoci (dakle, ukupno 3), dok će ostale biti crtičaste. Naredne funkcije realizuju ove funkcionalnosti:

```
function crtice(){
        if(kontekst.setLineDash){
                kontekst.setLineDash([10/vc,20/vc])
        }elseif(typeof kontekst.mozDash!="undefined"){
                kontekst.mozDash=[10/vc,20/vc];
        }
}
```

```
function punaLinija(){
	if(kontekst.setLineDash){
		kontekst.setLineDash([null])
	}elseif(typeof kontekst.mozDash!="undefined"){
		kontekst.mozDash=[];
	}
}
```

Kao što se vidi iz priloženog koda, metoda `setLineDash()`, ukoliko joj se navedu parametri koji odredjuju dimenzije crtica, aktivira ovu funkcionalnost. Izuzetak je jedino *Firefox* pregledač gde istu funkcionalnost obavlja `mozDash()` metoda.

Još jedna funkcionalnost koju je potrebno implementirati je promena linija na kojima se nalaze podeoci, a samim tim i njihovog tipa, u zavisnosti od ugla rotacije. Sledeći kod prikazuje realizaciju ove funkcionalnosti:

```
var w=[];
w[2]=3-Math.floor(grafik.rotZ/90);
w[0]=Math.floor(grafik.rotX/90);
w[1]=Math.floor(grafik.rotZ/180);
if(grafik.rotZ>90&&grafik.rotZ<=270){
	if(w[0]==0) w[0]=3;
}
if(grafik.rotX>90&&grafik.rotX<=180){
	if(w[1]==0) w[1]=3;
	if(w[1]==1) w[1]=2;
}
```

Ovim algoritmom se postavljaju određene vrednosti niza `w` koje se pri iscrtavanju linija koordinatnog sistema koriste na način prikazan u daljem tekstu.

Radi jednostavnijeg iscrtavanja, implementirana je i funkcija `crtLin()` koja kombinuje `moveTo()` `ilineTo()` metode:

```
function crtLin(a,b,c,d){
kontekst.moveTo(a,b);
kontekst.lineTo(c,d);
}
```

Konačno, linije koordinatnoh sistema se iscrtavaju na sledeći način:

```
for(var i=0;i<12;i++){

if(i==w[0]||i-4==w[1]||i-8==w[2]){
	kontekst.stroke();
	kontekst.beginPath();
	punaLinija();
}

crtLin(persp(koordTacke[mapaLinija[i][0]])[0],
	persp(koordTacke[mapaLinija[i][0]])[1],
	persp(koordTacke[mapaLinija[i][1]])[0],
	persp(koordTacke[mapaLinija[i][1]])[1]);

kontekst.stroke();
```

```
kontekst.beginPath();
crtice();
}
```

3.4.4. Podela osa na podeoke

Pri određivanju bvrednosti podeoka na osama, osnovni zahtev je da te vrednosti budu „pogodno“ izabrane. Na primer, nije prihvatljivo da vrednosti budu (2,9,16,23...), ali jeste da budu (0,5,10,15,20...). Pritombroj podeoka koji se prikazuje mora biti tačno određen. Ova dva zahteva nije uvek moguće zajedno ispuniti. Stoga u narednom kodu biće prikazana implementacija algoritma koji pokušava ispuni oba zahteva, kad god je to moguće.

```
// Određivanje podeoka na osama
function podeli_ose(min, max, broj_podeoka, xyz){

var opseg = max - min;
      korak =
      Math.pow(10,Math.floor(Math.log(opseg/broj_podeoka)/Math.log(10)));
      var greska = broj_podeoka / opseg * korak;

      // Zaokruživanje podeoka na pogodnije vrednosti
      if(greska <=.15) korak *=10;
      elseif(greska <=.35) korak *=5;
      elseif(greska <=.75) korak *=2;

      // Zaokruživanje početne i krajnje vrednosti
      var  start = Math.ceil(min / korak)* korak,
            stop = Math.floor(max / korak)* korak + korak *0.5,
            podeoci =[];

      if(xyz=="y") korakY=korak;
      if(xyz=="z") korakZ=korak;
      korakF=korak;
      // Upisivanje podeoka u niz
      for(var i=start; i < stop; i +=korak){

i=zaok(korak,i);
            podeoci.push(i);
      }
      return podeoci;
}     //podeli_ose
```

Kao što se vidi iz koda, ova funkcija kao parametre uzima minimalnu i maksimanu vrednost opsega, zadati broj podeoka, i parametar xyz, koji nije neophodan samom algoritmu, već u svrhu ispisivanja vrednosti podeoka na y ili z osi, o cemu ce biti reci kasnije. Algoritam računa razmak između podeoka po formuli datoj u kodu (promenljiva korak), zatim grešku nastalu određivanjem koraka po ovoj formuli i konačno uobličavanje vrednosti podeoka na osnovu ova dva parametra. Takođe, određuju se i početna i krajnja vrednost koje će biti na grafiku. Na kraju, funkcija vraća vrednosti ovako određenih podeoka u obliku niza..

Algoritam uvek određuje pogodne vrednosti za ispisavanje na osama. Cena ovoga je da ne vrati uvek traženi broj podeoka (unet kao ulazni parametar), već se taj broj može razlikovati za 1 ili 2 od njega.

3.4.5. Određivanje koordinata i iscrtavanje podeoka

Koristeći funkciju opisanu u prethodnom poglavlju, najpre se odredjuju podeoci za sve tri ose:

```
var podeociX=podeli_ose(xmin,xmax,7);
var podeociY=podeli_ose(ymin,ymax,7,"y");
var podeociZ=podeli_ose(zmin,zmax,7,"z");
```

Sada se koordinate ovih podeoka određuju i iscrtavaju u formi crtica sledećim kodom:

```
// Određivanje koordinata podeoka za x-osu
for(var i=0;i<podeociX.length;i++){
      var tpt=(podeociX[i]-xmin)/(xmax-xmin);
      var koordTemp=koordTacke;
      defKoordTacke();
      vartp3d=[[tpt,koordTacke[mapaLinija[w[0]][0]][1],koordTacke[mapaLini
      ja[w[0]][0]][2]],
      [tpt,koordTacke[mapaLinija[w[0]][0]][1]-
      0.01,koordTacke[mapaLinija[w[0]][0]][2]-0.01]];
      koordTacke=koordTemp;
      tp3d=transormacijaXNizTacaka(mnozenjeMatrica_n(transformacije),
      tp3d);
      var tp2d1=persp(tp3d[0]);
      tp2d2[i]=persp(tp3d[1]);
      kontekst.moveTo(tp2d1[0],tp2d1[1]);
      kontekst.lineTo(tp2d2[i][0],tp2d2[i][1]);
}
// Određivanje koordinata podeoka za y-osu
for(var i=0;i<podeociY.length;i++){
      var tpt=(podeociY[i]-ymin)/(ymax-ymin);
      var koordTemp=koordTacke;
      defKoordTacke();
      var tp3d=[
            [koordTacke[mapaLinija[w[1]+4][0]][0],tpt,koordTacke[mapaLinij
      a[w[1]+4][0]][2]],
      [koordTacke[mapaLinija[w[1]+4][0]][0]-
      0.01,tpt,koordTacke[mapaLinija[w[1]+4][0]][2]-0.01]];
      koordTacke=koordTemp;
      tp3d=transormacijaXNizTacaka(mnozenjeMatrica_n(transformacije),
      tp3d);
      var tp2d1=persp(tp3d[0]);
      tp2d2Y[i]=persp(tp3d[1]);
      kontekst.moveTo(tp2d1[0],tp2d1[1]);
      kontekst.lineTo(tp2d2Y[i][0],tp2d2Y[i][1]);
      }
// Određivanje koordinata podeoka za z-osu
var kp =[[-1,-1],[1,-1],[1,1],[-1,1]];
for(var i=0;i<podeociZ.length;i++){
      var tpt=(podeociZ[i]-zmin)/(zmax-zmin);
      var koordTemp=koordTacke;
      defKoordTacke();
      var tp3d=[  [koordTacke[w[2]][0],koordTacke[w[2]][1],tpt],
      [koordTacke[w[2]][0]+kp[w[2]][0]*0.01,koordTacke[w[2]][1]+kp[w[2]][1
      ]*0.01,tpt]];
      koordTacke=koordTemp;
      tp3d=transormacijaXNizTacaka(mnozenjeMatrica_n(transformacije),
      tp3d);
      var tp2d1=persp(tp3d[0]);
      tp2d2Z[i]=persp(tp3d[1]);
```

```
        kontekst.moveTo(tp2d1[0],tp2d1[1]);
        kontekst.lineTo(tp2d2Z[i][0],tp2d2Z[i][1]);
    }
    kontekst.stroke();
```

Na primeru odredjivanja podeoka za x-osu, najpre se vrši skaliranje vrednosti podeoka preko promenljive tpt. Ova vrednost se uzima kao x koordinata početne i krajnje vrednosti crtice koju je potrebno iscrtati. Vrednosti te dve tačke se upisuju u promenljivu tp3d, nad kojom se potom vrši isti set transformacioja kao nad ostalim tačkama grafika i koordinatnog sistema. Na kraju, vrši se perspektivna projekcija čime se dobijaju tačke koje se koriste za iscrtavanje crtica. Na sličan način se određuju koordinate crtica i za y i za z osu.

3.4.6. Ispisivanje vrednosti podeoka

U prethodnom poglavlju su određene koordinate podeoka, gde su i iscrtane crtice. Sada je ispod, odnosno pored njih, potrebno i ispisati same vrednosti tih podeoka.

Ove vrednosti se ispisuju sledećim kodom:

```
kontekst.scale(2/sc,-2/vc);
var velFonta=vc/50-dy/-4.5;
kontekst.font=velFonta+"px Arial";
kontekst.textBaseline="top";
kontekst.fillStyle="black";
for(var i=0;i<podeociX.length;i++){
        var podX=tp2d2[i][0]*sc/2-kontekst.measureText(podeociX[i]).width/2;

        var podY=-tp2d2[i][1]*vc/2+velFonta*0.15;
        kontekst.fillText(podeociX[i],podX,podY);
}

for(var i=0;i<podeociY.length;i++){
        var podX=tp2d2Y[i][0]*sc/2-
        kontekst.measureText(zaok(korakY,podeociY[i])).width/2;

        var podY=-tp2d2Y[i][1]*vc/2+velFonta*0.15;
        kontekst.fillText(zaok(korakY,podeociY[i]),podX,podY);
}

for(var i=0;i<podeociZ.length;i++){
        var podX=tp2d2Z[i][0]*sc/2-
        kontekst.measureText(zaok(korakZ,podeociZ[i])).width/2;

        var podY=-tp2d2Z[i][1]*vc/2+velFonta*0.15;
        kontekst.fillText(zaok(korakZ,podeociZ[i]),podX,podY);
}
```

Najpre je potrebno odrediti veličinu fonta, tako da je ona u skladu sa skaliranim*Canvas-om.* Zatim se vrednosti ispisuju na već određenim koordinatama, modifikovanim za veličinu samihbrojeva. Naime, ukoliko su na primer brojevi ispisuju sa leve strane crtice, oni moraju biti dodatno pomereni u odnosu na svoju dužinu da se s njom ne bi preklapali. U ovu svrhu se u kodu iznad koristi metoda `measureText()`.

3.5. Iscrtavanje 2D grafika – plot

Funkcionalnost iscrtavanja 2D grafika će takođe biti realizovana u 3D prostoru (baš kao i funkcija stem, koja će kasnije biti opisana). s tim da će jedna od koordinata uvek biti nula. Ovim se omogućava da neke od funkcionalnosti 3D grafika, kao što su rotacija i zumiranje, budu dostupne i u ovom slučaju.

Iscrtavanje grafika se poziva metodom plot().. Opseg vrednosti promenjive x se definiše na identičan način kao za plot3, dakle preko osobine grafik.x, ili osobina grafik.xmin, grafik.xmax, odnosno grafik.korak.

Za razliku od plot3 metode, plot omogućava iscrtavanje do 5 različitih grafika funkcija u jednom koordinatnom sistemu. Svaki od ovih grafika će biti isscrtan različitom bojom radi lakšeg raspoznavanja.

Funkcije koje se iscrtavaju se upisuju preko osobina grafik.f1 do grafik.f5. Jednostavnim pozivanjem metode plot svi definisani grafici će biti iscrtani.

3.5.1. Izračunavanje vrednosti funkcija i iscrtavanje grafika

Na početku, vrši se određivanje maksimuma i minimuma svih funkcija:

```
fmm[1][0]=maksFje(funkcijaF1);
fmm[1][1]=minFje(funkcijaF1);
fmm[2][0]=maksFje(funkcijaF2);
fmm[2][1]=minFje(funkcijaF2);
fmm[3][0]=maksFje(funkcijaF3);
fmm[3][1]=minFje(funkcijaF3);
fmm[4][0]=maksFje(funkcijaF4);
fmm[4][1]=minFje(funkcijaF4);
fmm[5][0]=maksFje(funkcijaF5);
fmm[5][1]=minFje(funkcijaF5);
```

Nakon toga, određuje se najveća vrednost od ovih maksimuma, kao i najmanja od minimuma:

```
for(iii=1;iii<6;iii++){
 if(typeof fmm[iii][0]=="undefined"||typeof
fmm[iii][1]=="undefined"){continue;}
 if(minMaxPoc==1){
       var maxMax=fmm[iii][0];
       var minMin=fmm[iii][1];
       minMaxPoc=0;
 }
 maxMax=Math.max(maxMax,fmm[iii][0]);
 minMin=Math.min(minMin,fmm[iii][1]);
 }
```

Te vrednosti se smeštaju u maxMax, odnosno minMin promenljive. One će kasnije biti korišćene u svrhu skaliranja grafika.

Sledeće, vrši se izračunavanje svih tačaka svih funkcija i upisuju u nizove f1 do f5:

```
var f1=IzracunajY(funkcijaF1);
var f2=IzracunajY(funkcijaF2);
var f3=IzracunajY(funkcijaF3);
var f4=IzracunajY(funkcijaF4);
var f5=IzracunajY(funkcijaF5);

function IzracunajY(fjaa){
```

```
var tempXYZ=[];
var x=xmin;
var iii=0;
while(x<=xmax){
            var y=fjaa(x);
            var z=0.5;
            var grafTacka=[];

tempXYZ[iii]=[x,y,z];
iii++;
x+=korak_x;
}
return tempXYZ;
}
```

Za iscrtavanje grafika, implementirana je funkcija koja se poziva i iscrtava svaku definisanu matematičku funkciju:

```
function crtanjeGrafika (tempfja){
kontekst.strokeStyle = mixBoja[grafBojaID];
grafBojaID++;
kontekst.beginPath();
for(var iii=0;iii<tempfja.length;iii++){
koordGraf3d[iii]=tempfja[iii];
var x=tempfja[iii][0];
var y=tempfja[iii][1];
var z=tempfja[iii][2];

 grafTacka[0]=(tempfja[iii][0]-xmin)/(xmax-xmin);
 grafTacka[1]=0.5;
 grafTacka[2]=(tempfja[iii][1]-minMin)/(maxMax-minMin)

 grafTacka=transormacijaXTacka(mnozenjeMatrica_n(transformacije),
                               grafTacka);
var xy2d=persp([grafTacka[0], grafTacka[1],grafTacka[2]]);

koordGrafCanv.push([xy2d[0]*sc/2+sc/2,vc/2-xy2d[1]*vc/2]);

//    Pozicioniranje za početak iscrtavanja grafika

      if(pocetak){
            if(isFinite(y)&&!isNaN(y)&&isFinite(z)&&!isNaN(z)){
                  kontekst.moveTo(xy2d[0],xy2d[1]);

                  pocetak=false;
            }

      //    Ukoliko je vrednost funkije beskonačna,
      //    sprečava se iscrtavanje asimptote
      }elseif(!pocetak&&(!isFinite(y)||!isFinite(z))){
                  asimptota=true;
      }else{
            if(asimptota){
                  kontekst.moveTo(xy2d[0],xy2d[1]);
                  asimptota=false;
            }else{

                  //    Iscrtavanje grafika
```

```
                    kontekst.lineTo(xy2d[0],xy2d[1]);
                }
            }
}

            kontekst.stroke();
```

Ovde postoji tek par razlika u odnosu na plot3 funkcionalnost. Prva je što je vrednost jedne koordinate uvek fiksirana. Druga je što se skaliranje vrši u odnosu nanajveću vrednost maksimuma, odnosno najmanju minimuma svih funkcija. Sve ostalo, uključujući i transformacije i perspektivnu projekciju je identično prethodno opisanoj metodi. Pritom, svaki sledeći grafik se iscrtava u drugoj boji, definisanoj u sledećem nizu:

```
var mixBoja=["red","green","blue","orange","violet"]
```

Iscrtavanje svih grafika se vrši jednostavnim pozivanem funkcije za vrednosti tačaka različitih grafika:

```
crtanjeGrafika(f1);
crtanjeGrafika(f2);
crtanjeGrafika(f3);
crtanjeGrafika(f4);
crtanjeGrafika(f5);
```

3.5.2. Iscrtavanje koordinatnog sistema i podeoka

S obzirom da se za koordinatni sistem iscrtava samo pravougaonik, ovde su koordinate tačaka i mapiranje linija drugačije:

```
koordTacke =[[0,0.5,0],[1,0.5,0],[1,0.5,0],[0,0.5,0],
            [0,0.5,1],[1,0.5,1],[1,0.5,1],[0,0.5,1]];

mapaLinija =[[0,1],[1,5],[5,4],[4,0]];
```

Međutim, samo iscrtavanje sistema, podela na podeoke, iscrtavanje crtica i ispisivanje vrednosti identično je kao za plot3, s tim da se za minimum i maksimum pri izračunavanjima koristi apsolutni minimum i maksimum svih funkcija.

3.6. Iscrtavanje diskretnih sekvenci podataka – stem

Kao i plot metoda, vrednosti parametara stem metode se jako slično definišu. Razlika je što stem metoda iscrtava diskretne vrednosti a ne kontinualne funkcije. Stoga postoji nekoliko bitnih razlika pri implementaciji ove metoda u odnosu na plot.

Kao i za plot, stem metodom je moguće iscrtavati više grafika na jednom koordinatnom sistemu, i ova funkcija je identično realizovana. Ono gde se ove dve metode razlikuju pri implementaciji jeste da je potrebno iscrtati linije od vrednosti funkcije do njene projekcija na x-osu.

Funkcija za crtanje grafika je prikazana ispod:

```
function crtanjeGrafika (tempfja){
kontekst.strokeStyle = mixBoja[grafBojaID];
grafBojaID++;
```

```
kontekst.beginPath();
for(var iii=0;iii<tempfja.length;iii++){

      koordGraf3d[iii]=tempfja[iii];
      var x=tempfja[iii][0];
      var y=tempfja[iii][1];
      var z=tempfja[iii][2];

       grafTacka[0]=(tempfja[iii][0]-xmin)/(xmax-xmin);
       grafTacka[1]=0.5;
       grafTacka[2]=(tempfja[iii][1]-minMin)/(maxMax-minMin)

       grafTackaX[0]=(tempfja[iii][0]-xmin)/(xmax-xmin);
       grafTackaX[1]=0.5;
       grafTackaX[2]=(-minMin)/(maxMax-minMin);

       grafTacka=transormacijaXTacka(mnozenjeMatrica_n(transformacije),
                                     grafTacka);

       grafTackaX=transormacijaXTacka(mnozenjeMatrica_n(transformacije),
                                     grafTackaX);

      var xy2d=persp([grafTacka[0], grafTacka[1],grafTacka[2]]);

      var x2d=persp([grafTackaX[0], grafTackaX[1],grafTackaX[2]]);

      koordGrafCanv.push([xy2d[0]*sc/2+sc/2,vc/2-xy2d[1]*vc/2]);

      //    Pozicioniranje za početak iscrtavanja grafika
      kontekst.beginPath();
      kontekst.arc(xy2d[0],xy2d[1],7/((sc+vc)/2)*grafik.zoom/100,0,2*Math.
      PI);

      kontekst.moveTo(xy2d[0],xy2d[1]);

      //    Iscrtavanje grafika

      kontekst.lineTo(x2d[0],x2d[1]);
      kontekst.stroke();

}

}//crtanjeGrafika
```

Dakle, pored definisanja tačke vrednosti funkcije, potrebno je definisati i tačku koja predstavlja njenu projekciju na x-osu. Nad ovom tačkom se takođe vrši transformacija i perspektivna projekcija. Iscrtavanje grafika se svodi na povačenje linije između ove dve tačke.

Takođe, radi lakšeg uočavanja same vrednosti funkcije, na njenom mestu se ucrtva krug arc() metodom.

Iscrtavanje koordinatnog sistema i određivanje podeoka se vrši na identičan način kao kod plot metode.

4. PRIMERI KORIŠĆENJA BIBLIOTEKE

4.1. Primer korišćenja plot3 metode.

Kao primer korišćenjaove metode biće iscrtan trodimenzionalni grafik matematičkih funkcija. Kod HTML stranice je prikazan ispod:

```
<!DOCTYPE html>
<html>
<head>
<metacharset="utf-8">
<scriptsrc="Grafik.js">
</script>

</head>
<body>
<divstyle="float:left;width:810px;">
<canvasid="canvas"width="700"height="600">Vaš internet pregledač ne podržava
canvas tag!</canvas>
<fieldsetstyle="width: 700px;margin-top:5px;">
xmin:<inputtype="text"id="xmin"size="8"style="margin-right:10px;"value=""/>
xmax:<inputtype="text"id="xmax"size="8"style="margin-right:10px;"value=""/>
korak:<inputtype="text"id="korak"size="8"value=""/>
<br/>
Y funkcija:<inputtype="text"id="fjaY"size="70"value=""/>
<br/>
Z funkcija:<inputtype="text"id="fjaZ"size="70"value=""/>
<br/><br/>
Uglovi rotacije:<br/>
x:<inputtype="text"id="rotX"size="3"value=""style="margin-left:3px;"/>
<br/>
z:<inputtype="text"id="rotZ"size="3"value=""style="margin-left:3px;"/>
<br/>
<br/>
Veličina grafika: <inputtype="text"id="zum"size="3"value=""style="margin-
left:3px;"/>%
<br/>
<inputtype="button"onclick="unesi(test)"value="Unesi promene">
</fieldset>
</div>
```

```
<script>

var canvas = document.getElementById("canvas");

function unesi(test){
	test.y=document.getElementById("fjaY").value;
	test.z=document.getElementById("fjaZ").value;
	test.xmin=document.getElementById("xmin").value;
	test.xmax=document.getElementById("xmax").value;
	test.korak=document.getElementById("korak").value;
	test.rotX=parseFloat(document.getElementById("rotX").value);
	test.rotZ=document.getElementById("rotZ").value;
```

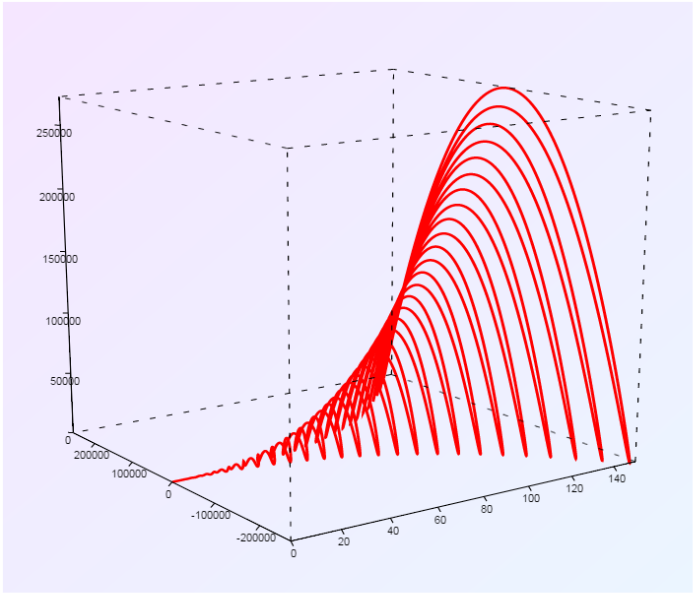
```
	test.zoom=parseFloat(document.getElementById("zum").value);
	test.plot3();
}
var
trenRx,prosRx,trenRz,prosRz,trenX,prosX,trenY,prosY,trenZ,prosZ,prosZum,trenZum;
function upd(gr){
	prosRx=trenRx;
	trenRx=gr.rotX;
	prosRz=trenRz;
	trenRz=gr.rotZ;
	prosX=trenX;
	trenX=gr.selX;
	prosY=trenY;
	trenY=gr.selY;
	prosZ=trenZ;
	trenZ=gr.selZ;
	prosZum=trenZum;
	trenZum=gr.zoom;

	if(trenRx!=prosRx||trenRz!=prosRz){
			document.getElementById("rotX").value=trenRx;
			document.getElementById("rotZ").value=trenRz;

		}
	if(trenZum!=prosZum){
		document.getElementById("zum").value=gr.zoom;
		}
}
var test =new Graph(canvas);
test.x="0:0.02:150.4469";
test.y="Math.pow(x,2.5)*Math.cos(x)";
test.z="Math.pow(x,2.5)*Math.sin(x)*Math.sin(x)";
document.getElementById('fjaY').value = test.y;
document.getElementById('fjaZ').value = test.z;
test.plot3();
document.getElementById('xmin').value = test.xmin;
document.getElementById('xmax').value = test.xmax;
document.getElementById('korak').value = test.korak;
document.getElementById("zum").value=test.zoom;
setInterval(function(){upd(test)},10);
</script>
</body>
</html>
```

Rezultat prikazivanja ove HTML stranice dat je na slici 4.1.1.

Dakle, nakon instanciranja objekta Graph, definisane su sve osobine te instance potrebne za iscrtavanje grafika, što su opseg vrednosti x i funkcije y i z. Pozivom metode plot () grafik je iscrtan.

Da bi osobine koje se mogu promeniti bolje bile ilustrovane, napravljen je korisnički interfejs koji čita vrednosti tih osobine i dozvoljava njihovu promenu.. Na primer, mogu se promeniti funkcije y i z, opseg promenjive x i uglovi rotacije. Ovo je prikazano na slici 4.1.2.

Rotacija se može postići i mišem, jednostavnim klikom i povlačenjem u stranu u koju se želi izvršiti. Slično, zumiranje se može postići promenom osobine *test.zoom,* ili okretanjem skrol dugmeta na mišu.

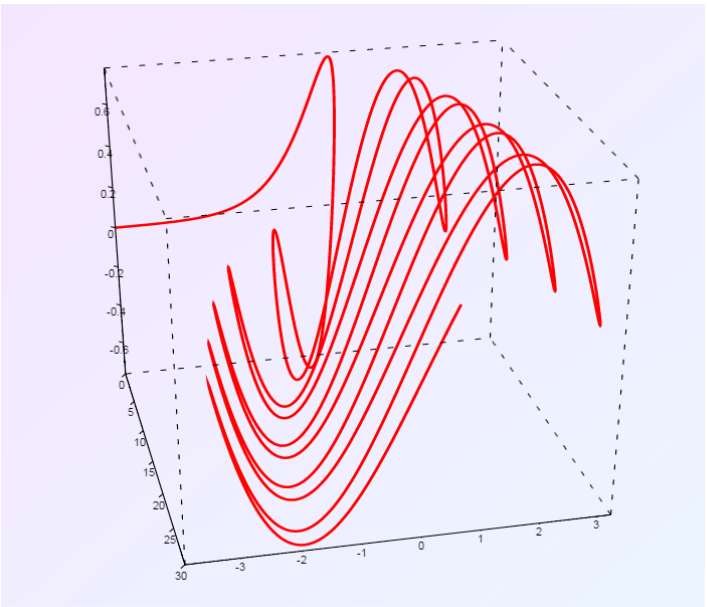


xmin: 0 xmax: 150.4469 korak: 0.02
Y funkcija: Math.pow(x,2.5)*Math.cos(x)
Z funkcija: Math.pow(x,2.5)*Math.sin(x)*Math.sin(x)

Uglovi rotacije:
x: 15
z: 35

Veličina grafika: 100 %
Unesi promene

Slika 4.1.1. Primer iscrtavanje 3D grafika primenom plot3 metode



xmin: 0 xmax: 30 korak: 0.02
Y funkcija: Math.cos(x)*Math.log(x)
Z funkcija: Math.sin(x)*Math.sin(2*x)

Uglovi rotacije:
x: 28.5
z: 283

Veličina grafika: 100 %
Unesi promene

Slika 4.1.2. Promena osobina direktno iz glavnog programa

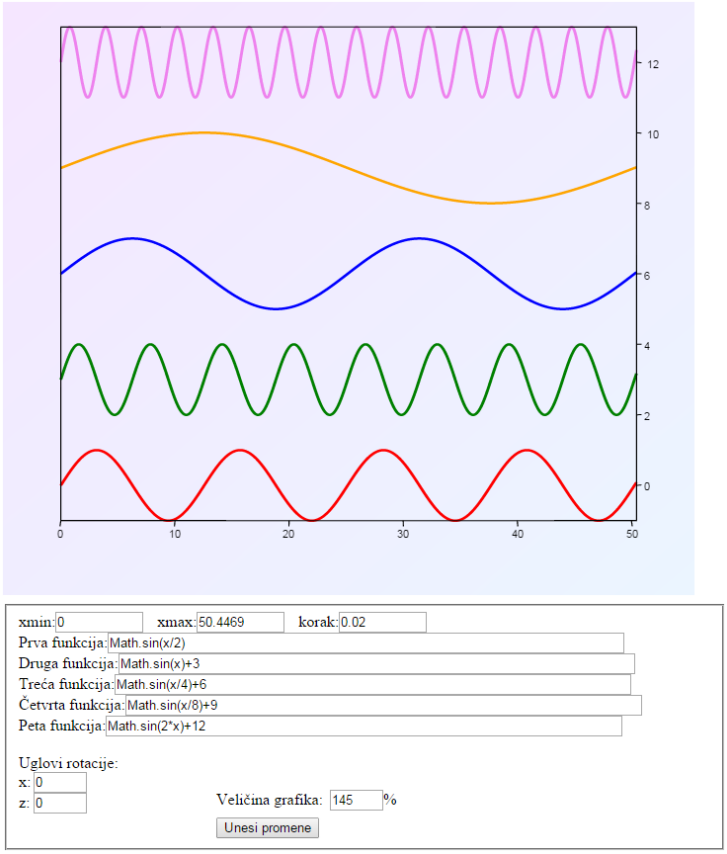
4.2. Primer korišćenja plot metode

U ovom poglavlju će biti dat primer korišćenja plot metode iscrtavanjem pet različitih grafika na jednom koordinatnom sistemu.

```
<!DOCTYPE html>
<html>
<head>
<metacharset="utf-8">
<scriptsrc="Grafik.js">
</script>
</head>
<body>
<divstyle="float:left;width:810px;">
<canvasid="canvas1"width="700"height="600"style="float:left;display:block;">Vaš
internet pregledač ne podržava canvas tag!</canvas>
<br/>
<pstyle="clear:both;">
      <fieldsetstyle="width: 700px;margin-top:25px;">
xmin:<inputtype="text"id="xmin1"size="8"style="margin-right:10px;"value=""/>
xmax:<inputtype="text"id="xmax1"size="8"style="margin-right:10px;"value=""/>
korak:<inputtype="text"id="korak1"size="8"value=""/>
<br/>
Prva funkcija:<inputtype="text"id="fja1_1"size="70"value=""/>
<br/>
Druga funkcija:<inputtype="text"id="fja2_1"size="70"value=""/>
<br/>
Treća funkcija:<inputtype="text"id="fja3_1"size="70"value=""/>
<br/>
Četvrta funkcija:<inputtype="text"id="fja4_1"size="70"value=""/>
<br/>
Peta funkcija:<inputtype="text"id="fja5_1"size="70"value=""/>
<br/><br/>
<divstyle='display:inline-block;float:left;width:200px;'>
Uglovi rotacije:<br/>
x:<inputtype="text"id="rotX1"size="3"value=""style="margin-left:3px;"/>
<br/>
z:<inputtype="text"id="rotZ1"size="3"value=""style="margin-left:3px;"/>
</div>
<br/><br/>
Veličina grafika: <inputtype="text"id="zum1"size="3"value=""style="margin-
left:3px;"/>%
<br/>
<inputtype="button"style="margin-top:7px;"onclick="unesi1(test1)"value="Unesi
promene">
</fieldset>
<pid="paragraf2">
</p>
</p>
</div>
<script>
var canvas1 = document.getElementById("canvas1");

function unesi1(test1){
      test1.f1=document.getElementById("fja1_1").value;
      test1.f2=document.getElementById("fja2_1").value;
      test1.f3=document.getElementById("fja3_1").value;
      test1.f4=document.getElementById("fja4_1").value;
      test1.f5=document.getElementById("fja5_1").value;
```

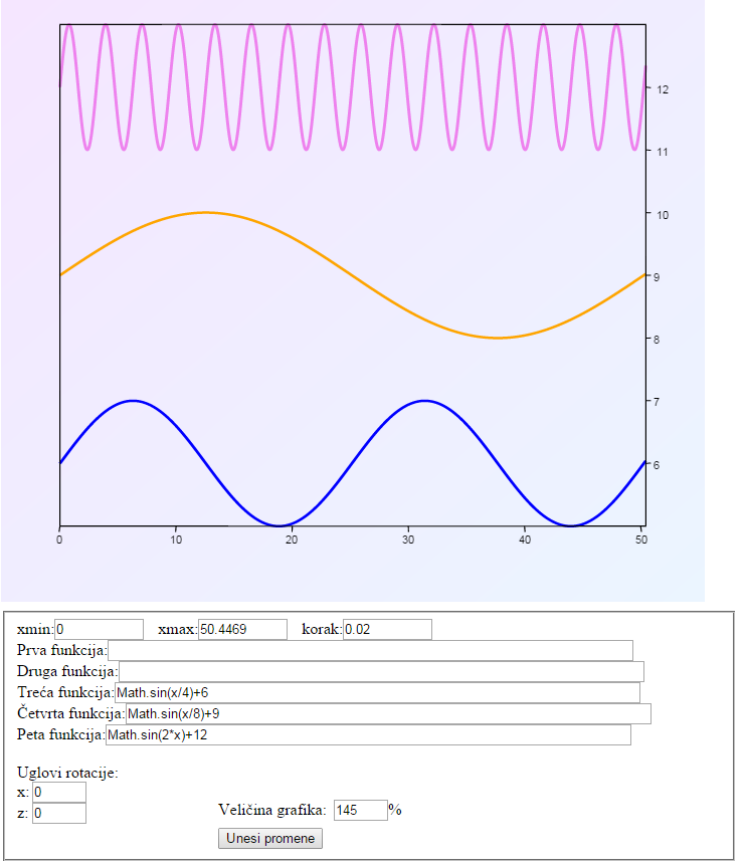
```
	test1.xmin=document.getElementById("xmin1").value;
	test1.xmax=document.getElementById("xmax1").value;
	test1.korak=document.getElementById("korak1").value;
	test1.rotX=parseFloat(document.getElementById("rotX1").value);
	test1.rotZ=document.getElementById("rotZ1").value;
	test1.zoom=parseFloat(document.getElementById("zum1").value);

test1.plot();
}

var
trenRx1,prosRx1,trenRz1,prosRz1,trenX1,prosX1,trenY1,prosY1,trenZ1,prosZ1,prosZu
m1,trenZum1;

function upd(gr1){

	prosRx1=trenRx1;
	trenRx1=gr1.rotX;
	prosRz1=trenRz1;
	trenRz1=gr1.rotZ;
	prosX1=trenX1;
	trenX1=gr1.selX;
	prosY1=trenY1;
	trenY1=gr1.selY;
	prosZ1=trenZ1;
	trenZ1=gr1.selZ;
	prosZum1=trenZum1;
	trenZum1=gr1.zoom;

	if(trenRx1!=prosRx1||trenRz1!=prosRz1){
		document.getElementById("rotX1").value=trenRx1;
		document.getElementById("rotZ1").value=trenRz1;

	}
	if(trenZum1!=prosZum1){
		document.getElementById("zum1").value=gr1.zoom;
		gr1.plot();
		}
}

var test1 =new Graph(canvas1);
test1.x="0:0.02:50.4469";
test1.f1="Math.sin(x/2)";
test1.f2="Math.sin(x)+3";
test1.f3="Math.sin(x/4)+6";
test1.f4="Math.sin(x/8)+9";
test1.f5="Math.sin(2*x)+12";
document.getElementById('fja1_1').value = test1.f1;
document.getElementById('fja2_1').value = test1.f2;
document.getElementById('fja3_1').value = test1.f3;
document.getElementById('fja4_1').value = test1.f4;
document.getElementById('fja5_1').value = test1.f5;
test1.plot();
document.getElementById('xmin1').value = test1.xmin;
document.getElementById('xmax1').value = test1.xmax;
document.getElementById('korak1').value = test1.korak;
document.getElementById("zum1").value=test1.zoom;

setInterval(function(){upd(test1)},10);
```

```
</script>
</body>
</html>
```



Slika 4.2.1. Primer korišćenja plot() metode

Na ovom primeru vide se sinusoide pomerene po y osi kako se ne bi preklapale Veličina se automatski skalira tako da svi grafici mogu biti prikazani. Ukoliko bi se, na primer izostavile prve dve funkcije, rezultat bi bio kao na slici 4.2.2.



Slika 4.2.2. Primer skaliranja veičine grafika

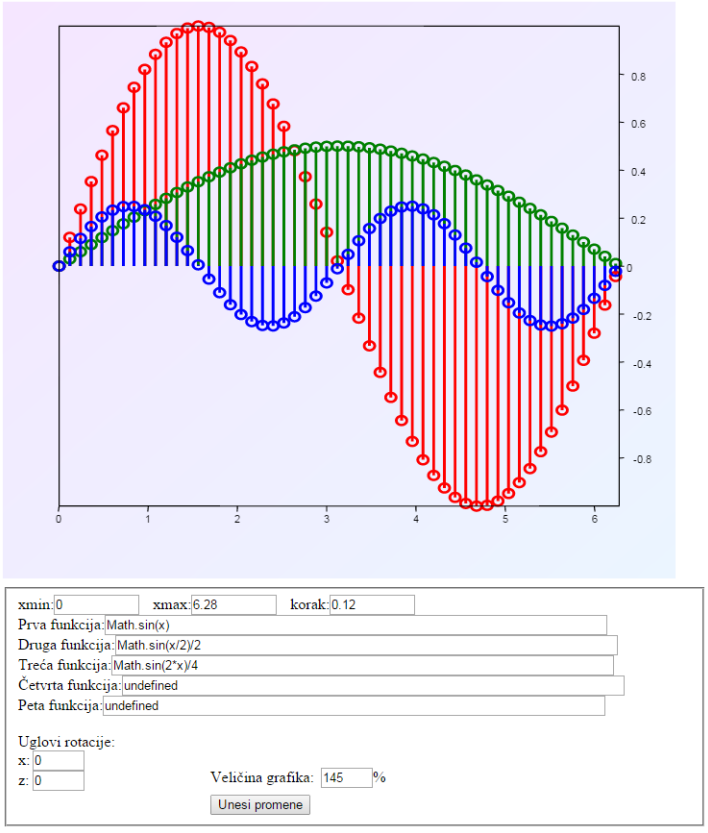
4.3. Primer korišćenja stem metode

Stem metodom se iscrtavaju diskretne vrednosti. Ovo je ilustrovano sledećim kodom:

```
<!DOCTYPE html>
<html>
<head>
<metacharset="utf-8">
<scriptsrc="Grafik.js">
</script>
</head>
<body>
<divstyle="float:left;width:810px;">
<canvasid="canvas2"width="700"height="600"style="float:left;display:block;">Vaš
internet pregledač ne podržava canvas tag!</canvas>
<br/>
<pstyle="clear:both;">
      <fieldsetstyle="width: 700px;margin-top:25px;">
xmin:<inputtype="text"id="xmin2"size="8"style="margin-right:10px;"value=""/>
xmax:<inputtype="text"id="xmax2"size="8"style="margin-right:10px;"value=""/>
korak:<inputtype="text"id="korak2"size="8"value=""/>
<br/>
Prva funkcija:<inputtype="text"id="fja1_2"size="70"value=""/>
<br/>
Druga funkcija:<inputtype="text"id="fja2_2"size="70"value=""/>
<br/>
Treća funkcija:<inputtype="text"id="fja3_2"size="70"value=""/>
```

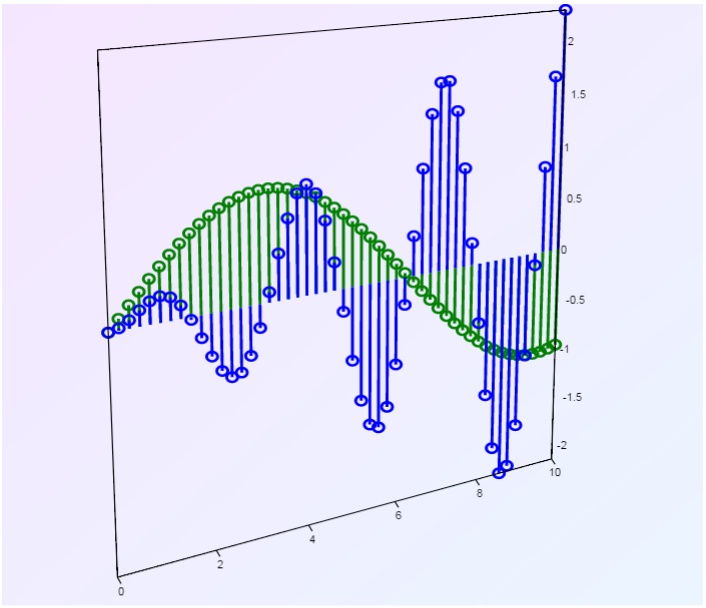
```
<br/>
Četvrta funkcija:<inputtype="text"id="fja4_2"size="70"value=""/>
<br/>
Peta funkcija:<inputtype="text"id="fja5_2"size="70"value=""/>
<br/><br/>
<divstyle='display:inline-block;float:left;width:200px;'>
Uglovi rotacije:<br/>
x:<inputtype="text"id="rotX2"size="3"value=""style="margin-left:3px;"/>
<br/>
z:<inputtype="text"id="rotZ2"size="3"value=""style="margin-left:3px;"/>
</div>
<br/><br/>
Veličina grafika: <inputtype="text"id="zum2"size="3"value=""style="margin-
left:3px;"/>%
<br/>
<inputtype="button"style="margin-top:7px;"onclick="unesi2(test2)"value="Unesi
promene">
</fieldset>
<pid="paragraf2">
</p>
</p>
</div>
<script>
var canvas2 = document.getElementById("canvas2");
function unesi2(test2){
      test2.f1=document.getElementById("fja1_2").value;
      test2.f2=document.getElementById("fja2_2").value;
      test2.f3=document.getElementById("fja3_2").value;
      test2.f4=document.getElementById("fja4_2").value;
      test2.f5=document.getElementById("fja5_2").value;
      test2.xmin=document.getElementById("xmin2").value;
      test2.xmax=document.getElementById("xmax2").value;
      test2.korak=document.getElementById("korak2").value;
      test2.rotX=parseFloat(document.getElementById("rotX2").value);
      test2.rotZ=document.getElementById("rotZ2").value;
      test2.zoom=parseFloat(document.getElementById("zum2").value);
test2.stem();
}
var
trenRx2,prosRx2,trenRz2,prosRz2,trenX2,prosX2,trenY2,prosY2,trenZ2,prosZ2,prosZu
m2,trenZum2;
function upd(gr2){
      prosRx2=trenRx2;
      trenRx2=gr2.rotX;
      prosRz2=trenRz2;
      trenRz2=gr2.rotZ;
      prosX2=trenX2;
      trenX2=gr2.selX;
      prosY2=trenY2;
      trenY2=gr2.selY;
      prosZ2=trenZ2;
      trenZ2=gr2.selZ;
      prosZum2=trenZum2;
      trenZum2=gr2.zoom;
      if(trenRx2!=prosRx2||trenRz2!=prosRz2){
            document.getElementById("rotX2").value=trenRx2;
            document.getElementById("rotZ2").value=trenRz2;

      }
```

```
        if(trenZum2!=prosZum2){
              document.getElementById("zum2").value=gr2.zoom;
              gr2.stem();}

}
var test2 =new Graph(canvas2);
test2.x="0:0.12:6.28";
test2.f1="Math.sin(x)";
test2.f2="Math.sin(x/2)/2";
test2.f3="Math.sin(2*x)/4";
//test2.f4="Math.sin(x/8)";
//test2.f5="Math.sin(2*x)";
document.getElementById('fja1_2').value = test2.f1;
document.getElementById('fja2_2').value = test2.f2;
document.getElementById('fja3_2').value = test2.f3;
document.getElementById('fja4_2').value = test2.f4;
document.getElementById('fja5_2').value = test2.f5;
test2.stem();
document.getElementById('xmin2').value = test2.xmin;
document.getElementById('xmax2').value = test2.xmax;
document.getElementById('korak2').value = test2.korak;
document.getElementById("zum2").value=test2.zoom;
setInterval(function(){upd(test2)},10);
</script>
</body>
</html>
```



Slika 4.3.1. Primer korišćenja stem metode

Kao što je već rečeno i na plot i na stem graficima moguće je koristiti rotaciju i zumiranje. Na slici 4.3.2. dat je primer promene više parametara iz prethodnog primera.



xmin: 0 xmax: 10 korak: 0.2
Prva funkcija:
Druga funkcija: Math.sin(x/2)
Treća funkcija: Math.sin(2*x)/4*x
Četvrta funkcija: undefined
Peta funkcija: undefined

Uglovi rotacije:
x: 15.5
z: 40

Veličina grafika: 145 %

Unesi promene

Slika 4.3.2. Primer rotacije i promene parametara grafika

5. ZAKLJUČAK

Internet pregledači se, pored računara, danas nalaze na velikom broju uređaja, kao što su npr, pametni telefoni ili čak i televizori.. Takođe nalaze se na svim operativnim sistemima. To ih čini moćnom platformom za aplikacije napisane u HTML5 i *JavaScript-u*. Tako, biblioteka opisana u ovom radu moći će da se koristi na svim pomenutim uređajima, u svim operativnim sistemima.

Biblioteka se, na primer, može koristiti u edukativne svrhe, jer se integracijom u sajt postiže veliki stepen interakivnosti i korisniku dozvoljava da sam menja parametre funkcija u realnom vremenu, umesto da samo posmatra sliku neke matematičke funkcije.

Budući razvoj biblioteke bi mogao ići u pravcu generisanja sve većeg broja tipova grafika, dodavanja više funkcionalnosti postojećim graficima, kao i postizanje određenog stepena integracije sa drugim kompleksnijim alatima, kao što je na primer *Matlab*.

LITERATURA

[1] Bruce Lawson, *Introducing HTML5*, New Riders,2011

[2] Jeremy Keith, *HTML5 For Web Designers*, A Book Apart Jeffrey Zeldmann, 2010

[3] Eric Rowell, *HTML5 Canvas Cookbook*, Packt Publishing, 2011

[4] Steve Fulton, Jeff Fulton, *HTML5 Canvas*, O'Reilly Media, 2013

[5] Cody Lindley, *Javascript Enlightenment*, O'Reilly Media, 2012

Prilog A

Kod biblioteke Grafik.js

```
window.Graph =function(canvas){

var kontekst = canvas.getContext("2d");
var sc = canvas.width;
var vc = canvas.height;

var grafik=this;

var nt=false;
var ind;
var korakY,korakZ,korakF;
grafik.yz='y';

// Podešavanje centra koordinatnog sistema u centru canvasa
// sa osama u opsegu (-1,1)
kontekst.save();
kontekst.translate(sc/2, vc/2);
kontekst.scale(sc/2,-vc/2);
kontekst.lineWidth=2/vc;

var flag =false,
    prethodniX =0,
    trenutniX =0,
    prethodniY =0,
    trenutniY =0;

var koordGraf3d=[];
var koordGrafCanv=[];
var koordGrafCanvNorm=[];

canvas.addEventListener("mousemove",function(e){
      nadjiXY('move', e)
},false);
canvas.addEventListener("touchmove",function(e){
      nadjiXY('tmove', e)
},false);
canvas.addEventListener("mousedown",function(e){
      nadjiXY('down', e)
},false);
canvas.addEventListener("mouseup",function(e){
      nadjiXY('up', e)
},false);
canvas.addEventListener("mouseout",function(e){
      nadjiXY('out', e)
},false);
canvas.addEventListener("click",function(e){
      nadjiXY('click', e)
},false);
canvas.addEventListener("dblclick",function(e){
      nadjiXY('dblclick', e)
},false);
```

```
if(canvas.addEventListener){
// IE9+, Chrome, Safari, Opera
canvas.addEventListener("mousewheel", skrolMisFukcija,false);
// Firefox
canvas.addEventListener("DOMMouseScroll",skrolMisFukcija,false);
}



function skrolMisFukcija(e){
      var e = window.event || e;
      if(e.preventDefault) e.preventDefault();
      var delta = Math.max(-1, Math.min(1,(e.wheelDelta ||-e.detail)));

      grafik.zoom=Math.round(Math.min(300,Math.max(10,grafik.zoom+grafik.zoom*0.
1*delta)));
      if(grafik.type=='plot'){grafik.plot()}
      elseif(grafik.type=='plot3'){grafik.plot3()}
      elseif(grafik.type=='stem'){grafik.stem()}

      returnfalse;
}

function nadjiXY(res, e){
if(res =='down'){
        prethodniX = trenutniX;
        prethodniY = trenutniY;
        trenutniX = e.clientX - canvas.offsetLeft;
        trenutniY = e.clientY - canvas.offsetTop;

        flag =true;

}

if(res =='up'|| res =="out"){
        flag =false;
}
if(res =='move'){
if(flag){
            prethodniX = trenutniX;
            prethodniY = trenutniY;
            trenutniX = e.clientX - canvas.offsetLeft;
            trenutniY = e.clientY - canvas.offsetTop;
                  rot();
}
}

      if(res =='tmove'){
      e.preventDefault();
var touchobj = e.changedTouches[0];
            prethodniX = trenutniX;
            prethodniY = trenutniY;
            trenutniX = parseInt(touchobj.clientX)- canvas.offsetLeft;
            trenutniY = parseInt(touchobj.clientY)- canvas.offsetTop;
                  rot();

}

      if(res =='click'){
```

```
        prethodniX = trenutniX;
        prethodniY = trenutniY;
        trenutniX = e.clientX - canvas.offsetLeft;
        trenutniY = e.clientY - canvas.offsetTop;

            for(var i=0;i<koordGrafCanv.length;i++){
                  if(Math.abs(trenutniX-koordGrafCanv[i][0])<5
                        &&Math.abs(trenutniY-koordGrafCanv[i][1])<5){
                              nt=true;
                              ind=i;
                  if(grafik.type=='plot'){grafik.plot()}
                  elseif(grafik.type=='plot3'){grafik.plot3()}
                  elseif(grafik.type=='stem'){grafik.stem()}

                              break;
            }

      }

      }


      if(res =='dblclick'){
            nt=false;
            grafik.selX="";
            grafik.selY="";
            grafik.selZ="";
if(grafik.type=='plot'){grafik.plot()}
elseif(grafik.type=='plot3'){grafik.plot3()}
elseif(grafik.type=='stem'){grafik.stem()}
}

}

function nacrtajTacku(koord, r){
            kontekst.save();
            kontekst.setTransform(1,0,0,1,0,0);
            kontekst.beginPath();
            kontekst.arc(koord[0],koord[1],r,0,2*Math.PI);
            kontekst.fillStyle="blue";
            kontekst.fill();
            kontekst.restore();

}

function rot(){
var rx1=(trenutniY-prethodniY)/2
var rz1=(trenutniX-prethodniX)/2;
grafik.rotX+=rx1;
grafik.rotZ+=rz1;
if(grafik.type=='plot'){grafik.plot()}
elseif(grafik.type=='plot3'){grafik.plot3()}
elseif(grafik.type=='stem'){grafik.stem()}
}



// dx,dy,dz-koordinate pogleda
var dy=-4.5;
```

```
var dx=0.5;
var dz=0.45;
var koordTacke =[];

function defKoordTacke(){

      koordTacke =[     [0,0,0],[1,0,0],[1,1,0],[0,1,0],
                              [0,0,1],[1,0,1],[1,1,1],[0,1,1]];


      mapaLinija =[     [0,1],[4,5],[6,7],[2,3],
                              [0,3],[1,2],[5,6],[4,7],
                              [0,4],[1,5],[2,6],[3,7] ];


}

function defKoordTacke2d(){

      koordTacke =[     [0,0.5,0],[1,0.5,0],[1,0.5,0],[0,0.5,0],
                              [0,0.5,1],[1,0.5,1],[1,0.5,1],[0,0.5,1]];

      mapaLinija =[     [0,1],[1,5],[5,4],[4,0]];


}

function persp(p){
      return[(p[0]-dx)/(p[1]-dy)*6,(p[2]-dz)/(p[1]-dy)*6]
}
function transliraj(x, y, z){
return[[1,0,0, x],[0,1,0, y],[0,0,1, z]]
}
// jedinicna matrica, preko translate f-je
function jedinicnaMatrica(){return transliraj(0,0,0)}

// rotiranje po x-osi, vraća prve tri vrste matrice
function rotirajOkoX(theta){
theta=theta/180*Math.PI;
var s = Math.sin(theta)
var c = Math.cos(theta)
return[[1,0,0,0],[0, c,-s,0],[0, s, c,0]]
}
// rotiranje po y-osi, vraća prve tri vrste matrice
function rotirajOkoY(theta){
theta=theta/180*Math.PI;
var s = Math.sin(theta)
var c = Math.cos(theta)
return[[c,0,-s,0],[0,1,0,0],[s,0, c,0]]
}

// rotiranje po z-osi, vraća prve tri vrste matrice
function rotirajOkoZ(theta){
theta=theta/180*Math.PI;
var s = Math.sin(theta)
var c = Math.cos(theta)
return[[c,-s,0,0],[s, c,0,0],[0,0,1,0]]
}
```

```
// mnozenje 2 matrice, obrnuto
function mnozenjeMatrica(x1, x2){
var rv =[[0,0,0,0],[0,0,0,0],[0,0,0,0]]
for(var ii =0; ii <3; ii++){
    rv[ii][3]= x2[ii][3]
for(var jj =0; jj <3; jj++){
      rv[ii][3]+= x1[jj][3]* x2[ii][jj]
for(var kk =0; kk <3; kk++){
       rv[ii][jj]+= x1[kk][jj]* x2[ii][kk]
}
}
}
return rv
}
// mnozenje vise matrica pomocu mnozenja 2, obrnuto
function mnozenjeMatrica_n(nizMatrica){
var rv = jedinicnaMatrica()
for(var ii =0; ii < nizMatrica.length; ii++) rv = mnozenjeMatrica(rv,
nizMatrica[ii])
return rv
}
// mnozenje transformacije i tacke
function transormacijaXTacka(transform, p){
var rezultat =[]
for(var ii =0; ii <3; ii++){
var rv = transform[ii][3]
for(var jj =0; jj <3; jj++) rv += transform[ii][jj]* p[jj]
    rezultat.push(rv)
}
return rezultat
}
// mnozenje vise tacaka s transformacijom,
// promenjiva "tacke" je niz tacaka
function transormacijaXNizTacaka(xf, tacke){
var xp =[]
for(var ii =0; ii < tacke.length; ii++){
    xp.push(transormacijaXTacka(xf, tacke[ii]))
}
return xp
}



// crtanje linije od početne do krajnje tacke
function crtLin(a,b,c,d){
      kontekst.moveTo(a,b);
      kontekst.lineTo(c,d);
}


var xmin,xmax,korak_x;

function funkcijaY(x){
      return eval(grafik.y);
}
function funkcijaZ(x){
```

```
            return eval(grafik.z);
}


function funkcijaF1(x){

            return eval(grafik.f1);
}

function funkcijaF2(x){

            return eval(grafik.f2);
}

function funkcijaF3(x){

            return eval(grafik.f3);
}

function funkcijaF4(x){

            return eval(grafik.f4);
}

function funkcijaF5(x){

            return eval(grafik.f5);
}

function maksFje(mxfja){

      var x=xmin;
      var ymaxset=false;

      while(x<=xmax){

            var y=mxfja(x);

            if(isFinite(y)){
                  if(ymaxset){
                              if(y>ymax){ymax=y}
                  }else{
                              var ymax=y;
                              ymaxset=true;
                  }
            }
      x+=korak_x;
      }

      return ymax;
}
function minFje(mnfja){

      var x=xmin;
      var yminset=false;
      while(x<=xmax){

            var y=mnfja(x);
```

```
            if(isFinite(y)){
                  if(yminset){
                        if(y<ymin){ymin=y}
                  }else{
                        var ymin=y;
                        yminset=true;
                  }
            }
      x+=korak_x;
      }

      return ymin;
}

function zaok(temp,x)
      {
            var dec=(temp.toString().split('.')[1]||[]).length;
            if(dec>0){
                  x=Math.round(x*Math.pow(10,dec))/Math.pow(10,dec);
            }
            return x;
      }

function back(){
var grd=kontekst.createLinearGradient(-1,1,1,-1);
grd.addColorStop(0,"#F5E6FF");
grd.addColorStop(1,"#EBF5FF");
kontekst.fillStyle=grd;
kontekst.rect(-1,-1,2,2);
kontekst.fill();
}

function ugao360(ugao){
      ugao=ugao-360*Math.floor(ugao/360);
      return ugao;
}


this.plot3 =function(){
if(typeof grafik.rotX=="undefined"||
      typeof grafik.rotZ=="undefined"){
grafik.rotX=15;
grafik.rotZ=35;
}
if(typeof grafik.zoom=="undefined"){
grafik.zoom=100;}
grafik.rotX=ugao360(grafik.rotX);
grafik.rotZ=ugao360(grafik.rotZ);
grafik.type='plot3';
dy=-4.5*100/grafik.zoom;

defKoordTacke();


if(typeof grafik.xmin=="undefined"||
      typeof grafik.xmax=="undefined"||
      typeof grafik.korak=="undefined"){
            var x_niz=grafik.x.split(":");
            xmin=parseFloat(x_niz[0]);
```

```
            korak_x=parseFloat(x_niz[1]);
            xmax=parseFloat(x_niz[2]);
      grafik.xmin=xmin;
      grafik.xmax=xmax;
      grafik.korak=korak_x;
}else{

      if(xmin!=grafik.xmin||
            xmax!=grafik.xmax||
            korak_x!=grafik.korak){
            grafik.selX="";
            grafik.selY="";
            grafik.selZ="";
            nt=false}

      xmin=parseFloat(grafik.xmin);
      xmax=parseFloat(grafik.xmax);
      korak_x=parseFloat(grafik.korak);

}
var pocetak=true;
var asimptota=false;
var x=xmin;

var ymax=maksFje(funkcijaY),
      ymin=minFje(funkcijaY),
      zmax=maksFje(funkcijaZ),
      zmin=minFje(funkcijaZ);

      kontekst.clearRect(-1,-1,2,2);
      back();
      kontekst.beginPath();
      kontekst.lineWidth=2/vc*2.5;


var transformacije=[transliraj(-0.5,-0.5,-0.5),
                              rotirajOkoZ(0+grafik.rotZ),
                              rotirajOkoX(0+grafik.rotX),

                              transliraj(0.5,0.5,0.5)];

koordGraf3d=[];
koordGrafCanvNorm=[];
koordGrafCanv=[];
while(x<=xmax){
            var y=funkcijaY(x);
            var z=funkcijaZ(x);
            var grafTacka=[];

            // Upisivanje originalnih tačaka u niz
            koordGraf3d.push([x,y,z]);
            // Skaliranje vrednosti u opseg od -1 do 1
            grafTacka[0]=(x-xmin)/(xmax-xmin),
            grafTacka[1]=(y-ymin)/(ymax-ymin),
            grafTacka[2]=(z-zmin)/(zmax-zmin);
            // transformacije nad tačkama grafika
            grafTacka=transormacijaXTacka(mnozenjeMatrica_n(transformacije),
                              grafTacka);
            // Perspektivna projekcija
```

```
            var xy2d=persp([grafTacka[0],
                                    grafTacka[1],
                                    grafTacka[2]]);

            koordGrafCanv.push([xy2d[0]*sc/2+sc/2,vc/2-xy2d[1]*vc/2]);

            //    Pozicioniranje za početak iscrtavanja grafika
            if(pocetak){
                  if(isFinite(y)&&!isNaN(y)&&isFinite(z)&&!isNaN(z)){
                        kontekst.moveTo(xy2d[0],xy2d[1]);

                        pocetak=false;
                  }

            //    Ukoliko je vrednost funkcije beskonačna,
            //    sprečava se iscrtavanje asimptote
            }elseif(!pocetak&&(!isFinite(y)||!isFinite(z))){
                        asimptota=true;
            }else{
                  if(asimptota){
                        kontekst.moveTo(xy2d[0],xy2d[1]);
                        asimptota=false;
                  }else{
                        //    Iscrtavanje grafika
                        kontekst.strokeStyle ='red';
                        kontekst.lineTo(xy2d[0],xy2d[1]);

                  }
            }

            x+=korak_x;

      }
kontekst.stroke();



      // 3d transformacija koordinatnog sistema
      koordTacke=transormacijaXNizTacaka(mnozenjeMatrica_n(transformacije),
                              koordTacke);


      function crtice(){
            if(kontekst.setLineDash){
                        kontekst.setLineDash([10/vc,20/vc])
            }elseif(typeof kontekst.mozDash!="undefined"){
                        kontekst.mozDash=[10/vc,20/vc];
            }
      }

      function punaLinija(){
            if(kontekst.setLineDash){
                  kontekst.setLineDash([null])
            }elseif(typeof kontekst.mozDash!="undefined"){
                  kontekst.mozDash=[];
            }
      }
      kontekst.beginPath();
      kontekst.lineWidth=2/vc;
```

```
	kontekst.strokeStyle ='black';
	crtice();


	var w=[];
	w[2]=3-Math.floor(grafik.rotZ/90);
	w[0]=Math.floor(grafik.rotX/90);
	w[1]=Math.floor(grafik.rotZ/180);
	if(grafik.rotZ>90&&grafik.rotZ<=270){
		if(w[0]==0) w[0]=3;
	}
	if(grafik.rotX>90&&grafik.rotX<=180){
		if(w[1]==0) w[1]=3;
		if(w[1]==1) w[1]=2;
	}



	// iscrtavanje linija koordinatnog sistema
	for(var i=0;i<12;i++){

		if(i==w[0]||i-4==w[1]||i-8==w[2]){
			kontekst.stroke();
			kontekst.beginPath();
			punaLinija();
		}

		crtLin(
	persp(koordTacke[mapaLinija[i][0]])[0],persp(koordTacke[mapaLinija[i][0]])
[1],

	persp(koordTacke[mapaLinija[i][1]])[0],persp(koordTacke[mapaLinija[i][1]])
[1]);

		kontekst.stroke();
		kontekst.beginPath();
		crtice();
	}

	punaLinija();

	//podela na 5 podeoka od 0 do 5
	var podeociX=podeli_ose(xmin,xmax,7);
	var podeociY=podeli_ose(ymin,ymax,7,"y");
	var podeociZ=podeli_ose(zmin,zmax,7,"z");
	kontekst.beginPath();
	var tp2d2=[];
	var tp2d2Y=[];
	var tp2d2Z=[];
	// Određivanje koordinata podeoka za x-osu
	for(var i=0;i<podeociX.length;i++){
		var tpt=(podeociX[i]-xmin)/(xmax-xmin);
		var koordTemp=koordTacke;
		defKoordTacke();
		var tp3d=[
	[tpt,koordTacke[mapaLinija[w[0]][0]][1],koordTacke[mapaLinija[w[0]][0]][2]
],
				[tpt,koordTacke[mapaLinija[w[0]][0]][1]-
0.01,koordTacke[mapaLinija[w[0]][0]][2]-0.01]];
```

```
            koordTacke=koordTemp;
            tp3d=transormacijaXNizTacaka(mnozenjeMatrica_n(transformacije),
                                    tp3d);
            var tp2d1=persp(tp3d[0]);
            tp2d2[i]=persp(tp3d[1]);
            kontekst.moveTo(tp2d1[0],tp2d1[1]);
            kontekst.lineTo(tp2d2[i][0],tp2d2[i][1]);
      }
      // Određivanje koordinata podeoka za Y-osu
      for(var i=0;i<podeociY.length;i++){
            var tpt=(podeociY[i]-ymin)/(ymax-ymin);

            var koordTemp=koordTacke;
            defKoordTacke();
            var tp3d=[
      [koordTacke[mapaLinija[w[1]+4][0]][0],tpt,koordTacke[mapaLinija[w[1]+4][0]
][2]],
                              [koordTacke[mapaLinija[w[1]+4][0]][0]-
0.01,tpt,koordTacke[mapaLinija[w[1]+4][0]][2]-0.01]];
            koordTacke=koordTemp;
            tp3d=transormacijaXNizTacaka(mnozenjeMatrica_n(transformacije),
                                    tp3d);
            var tp2d1=persp(tp3d[0]);
            tp2d2Y[i]=persp(tp3d[1]);
            kontekst.moveTo(tp2d1[0],tp2d1[1]);
            kontekst.lineTo(tp2d2Y[i][0],tp2d2Y[i][1]);
      }
            // Određivanje koordinata podeoka za Z-osu
            var kp =[[-1,-1],[1,-1],[1,1],[-1,1]];
            for(var i=0;i<podeociZ.length;i++){
            var tpt=(podeociZ[i]-zmin)/(zmax-zmin);
            var koordTemp=koordTacke;
            defKoordTacke();
            var tp3d=[  [koordTacke[w[2]][0],koordTacke[w[2]][1],tpt],

      [koordTacke[w[2]][0]+kp[w[2]][0]*0.01,koordTacke[w[2]][1]+kp[w[2]][1]*0.01
,tpt]];
            koordTacke=koordTemp;
            tp3d=transormacijaXNizTacaka(mnozenjeMatrica_n(transformacije),
                                    tp3d);
            var tp2d1=persp(tp3d[0]);
            tp2d2Z[i]=persp(tp3d[1]);
            kontekst.moveTo(tp2d1[0],tp2d1[1]);
            kontekst.lineTo(tp2d2Z[i][0],tp2d2Z[i][1]);
      }

      kontekst.stroke();


      kontekst.scale(2/sc,-2/vc);
      var velFonta=vc/50-dy/-4.5;
      kontekst.font=velFonta+"px Arial";
      kontekst.textBaseline="top";
      kontekst.fillStyle="black";
      for(var i=0;i<podeociX.length;i++){
            var podX=tp2d2[i][0]*sc/2-kontekst.measureText(podeociX[i]).width/2;

            var podY=-tp2d2[i][1]*vc/2+velFonta*0.15;
            kontekst.fillText(podeociX[i],podX,podY);
```

```
	}

	for(var i=0;i<podeociY.length;i++){
		var podX=tp2d2Y[i][0]*sc/2-
kontekst.measureText(zaok(korakY,podeociY[i])).width/2;

		var podY=-tp2d2Y[i][1]*vc/2+velFonta*0.15;
		kontekst.fillText(zaok(korakY,podeociY[i]),podX,podY);
	}

	for(var i=0;i<podeociZ.length;i++){
		var podX=tp2d2Z[i][0]*sc/2-
kontekst.measureText(zaok(korakZ,podeociZ[i])).width/2;

		var podY=-tp2d2Z[i][1]*vc/2+velFonta*0.15;
		kontekst.fillText(zaok(korakZ,podeociZ[i]),podX,podY);
	}




	kontekst.scale(sc/2,-vc/2);


	if(nt){
		grafik.selX=zaok(korak_x,koordGraf3d[ind][0]);
		grafik.selY=koordGraf3d[ind][1];
		grafik.selZ=koordGraf3d[ind][2];
		nacrtajTacku(koordGrafCanv[ind],4-(4-dy)/14);

		}

	}

this.plot =function(){

if(typeof grafik.rotX=="undefined"||
	typeof grafik.rotZ=="undefined"){
grafik.rotX=0;
grafik.rotZ=0;
}
if(typeof grafik.zoom=="undefined"){
grafik.zoom=145;}
grafik.rotX=ugao360(grafik.rotX);
grafik.rotZ=ugao360(grafik.rotZ);

grafik.type='plot';
dy=-4.5*100/grafik.zoom;

defKoordTacke2d()

if(typeof grafik.xmin=="undefined"||
	typeof grafik.xmax=="undefined"||
	typeof grafik.korak=="undefined"){
		var x_niz=grafik.x.split(":");
		xmin=parseFloat(x_niz[0]);
		korak_x=parseFloat(x_niz[1]);
		xmax=parseFloat(x_niz[2]);
```

```
	grafik.xmin=xmin;
	grafik.xmax=xmax;
	grafik.korak=korak_x;
}else{



	if(xmin!=grafik.xmin||
		xmax!=grafik.xmax||
		korak_x!=grafik.korak){
		grafik.selX="";
		grafik.selY="";
		grafik.selZ="";
		nt=false}

	xmin=parseFloat(grafik.xmin);
	xmax=parseFloat(grafik.xmax);
	korak_x=parseFloat(grafik.korak);

}

var pocetak=true;
var asimptota=false;
var x=xmin;
var fmm =[[],[],[],[],[],[]];
fmm[1][0]=maksFje(funkcijaF1);
fmm[1][1]=minFje(funkcijaF1);
fmm[2][0]=maksFje(funkcijaF2);
fmm[2][1]=minFje(funkcijaF2);
fmm[3][0]=maksFje(funkcijaF3);
fmm[3][1]=minFje(funkcijaF3);
fmm[4][0]=maksFje(funkcijaF4);
fmm[4][1]=minFje(funkcijaF4);
fmm[5][0]=maksFje(funkcijaF5);
fmm[5][1]=minFje(funkcijaF5);

	var minMaxPoc=1;
//	alert(maxMax);
for(iii=1;iii<6;iii++){
	if(typeof fmm[iii][0]=="undefined"||typeof
fmm[iii][1]=="undefined"){continue;}
	if(minMaxPoc==1){
		var maxMax=fmm[iii][0];
		var minMin=fmm[iii][1];
		minMaxPoc=0;
	}
	maxMax=Math.max(maxMax,fmm[iii][0]);
	minMin=Math.min(minMin,fmm[iii][1]);
}



	kontekst.clearRect(-1,-1,2,2);
	back();
	kontekst.beginPath();
	kontekst.lineWidth=2/vc*2.5;

var transformacije=[transliraj(-0.5,-0.5,-0.5),
				rotirajOkoZ(0+grafik.rotZ),
```

```
                              rotirajOkoX(0+grafik.rotX),

                              transliraj(0.5,0.5,0.5)];

                              koordGraf3d=[];
koordGrafCanvNorm=[];
koordGrafCanv=[];
function IzracunajY(fjaa){
      var tempXYZ=[];
      var x=xmin;
      var iii=0;
      while(x<=xmax){
                  var y=fjaa(x);
                  var z=0.5;
                  var grafTacka=[];


      tempXYZ[iii]=[x,y,z];
      iii++;
      x+=korak_x;
      }
return tempXYZ;
}

var f1=IzracunajY(funkcijaF1);
var f2=IzracunajY(funkcijaF2);
var f3=IzracunajY(funkcijaF3);
var f4=IzracunajY(funkcijaF4);
var f5=IzracunajY(funkcijaF5);
var grafTacka=[];

var mixBoja=["red","green","blue","orange","violet"]
var grafBojaID=0;
function crtanjeGrafika (tempfja){
kontekst.strokeStyle = mixBoja[grafBojaID];
grafBojaID++;
 kontekst.beginPath();
for(var iii=0;iii<tempfja.length;iii++){

koordGraf3d[iii]=tempfja[iii];
var x=tempfja[iii][0];
var y=tempfja[iii][1];
var z=tempfja[iii][2];

 grafTacka[0]=(tempfja[iii][0]-xmin)/(xmax-xmin);
 grafTacka[1]=0.5;
 grafTacka[2]=(tempfja[iii][1]-minMin)/(maxMax-minMin)


 grafTacka=transormacijaXTacka(mnozenjeMatrica_n(transformacije),
                              grafTacka);

            var xy2d=persp([grafTacka[0],
                                    grafTacka[1],
                                    grafTacka[2]]);

                                    koordGrafCanv.push([xy2d[0]*sc/2+sc/2,vc/2-
xy2d[1]*vc/2]);
```

```
            //    Pozicioniranje za početak iscrtavanja grafika

            if(pocetak){
                  if(isFinite(y)&&!isNaN(y)&&isFinite(z)&&!isNaN(z)){
                        kontekst.moveTo(xy2d[0],xy2d[1]);

                        pocetak=false;
                  }


            //    Ukoliko je vrednost funkije beskonačna,
            //    sprečava se iscrtavanje asimptote
            }elseif(!pocetak&&(!isFinite(y)||!isFinite(z))){
                        asimptota=true;
            }else{
                  if(asimptota){
                        kontekst.moveTo(xy2d[0],xy2d[1]);
                        asimptota=false;
                  }else{

                        //    Iscrtavanje grafika

                        kontekst.lineTo(xy2d[0],xy2d[1]);

                  }
            }

}

 kontekst.stroke();

}//crtanjeGrafika

crtanjeGrafika(f1);
crtanjeGrafika(f2);
crtanjeGrafika(f3);
crtanjeGrafika(f4);
crtanjeGrafika(f5);



      // 3d transformacija koordinatnog sistema
      koordTacke=transormacijaXNizTacaka(mnozenjeMatrica_n(transformacije),
                              koordTacke);

      kontekst.lineWidth=2/vc;
      kontekst.strokeStyle ='black';

      // iscrtavanje linija koordinatnog sistema
      for(var i=0;i<4;i++){


                  kontekst.beginPath();

            crtLin(
      persp(koordTacke[mapaLinija[i][0]])[0],persp(koordTacke[mapaLinija[i][0]])
[1],
```

```
      persp(koordTacke[mapaLinija[i][1]])[0],persp(koordTacke[mapaLinija[i][1]])
[1]);

            kontekst.stroke();
            kontekst.beginPath();

      }

      //podela na 5 podeoka od 0 do 5
      var podeociX=podeli_ose(xmin,xmax,7);
      var podeociF=podeli_ose(minMin,maxMax,7);
      kontekst.beginPath();
      var tp2d2=[];
      var tp2d2Y=[];
      var tp2d2Z=[];
      for(var i=0;i<podeociX.length;i++){
            var tpt=(podeociX[i]-xmin)/(xmax-xmin);

            var koordTemp=koordTacke;
            defKoordTacke2d();
            var tp3d=[
      [tpt,koordTacke[mapaLinija[0][0]][1],koordTacke[mapaLinija[0][0]][2]],
                              [tpt,koordTacke[mapaLinija[0][0]][1]-
0.01,koordTacke[mapaLinija[0][0]][2]-0.01]];
            koordTacke=koordTemp;
            tp3d=transormacijaXNizTacaka(mnozenjeMatrica_n(transformacije),
                                    tp3d);
            var tp2d1=persp(tp3d[0]);
            tp2d2[i]=persp(tp3d[1]);
            kontekst.moveTo(tp2d1[0],tp2d1[1]);
            kontekst.lineTo(tp2d2[i][0],tp2d2[i][1]);
      }

            var kp =[[-1,-1],[1,-1],[1,1],[-1,1]];
            for(var i=0;i<podeociF.length;i++){
            var tpt=(podeociF[i]-minMin)/(maxMax-minMin);
            var koordTemp=koordTacke;
            defKoordTacke2d();
            var tp3d=[  [koordTacke[2][0],koordTacke[2][1],tpt],

      [koordTacke[2][0]+kp[2][0]*0.01,koordTacke[2][1]+kp[2][1]*0.01,tpt]];
            koordTacke=koordTemp;
            tp3d=transormacijaXNizTacaka(mnozenjeMatrica_n(transformacije),
                                    tp3d);

            var tp2d1=persp(tp3d[0]);
            tp2d2Z[i]=persp(tp3d[1]);
            kontekst.moveTo(tp2d1[0],tp2d1[1]);
            kontekst.lineTo(tp2d2Z[i][0],tp2d2Z[i][1]);
      }

      kontekst.stroke();


      kontekst.scale(2/sc,-2/vc);
      var velFonta=vc/50-dy/-4.5;
      kontekst.font=velFonta+"px Arial";
      kontekst.textBaseline="top";
```

```
	kontekst.fillStyle="black";
	for(var i=0;i<podeociX.length;i++){
		var podX=tp2d2[i][0]*sc/2-kontekst.measureText(podeociX[i]).width/2;

		var podF=-tp2d2[i][1]*vc/2+velFonta*0.15;
		kontekst.fillText(podeociX[i],podX,podF);
	}

	for(var i=0;i<podeociF.length;i++){
		kontekst.textBaseline="top";
		var
podX=tp2d2Z[i][0]*sc/2+kontekst.measureText(zaok(korakF,podeociF[i])).width/2;
		kontekst.textBaseline="middle";
		var podF=-tp2d2Z[i][1]*vc/2+velFonta*0.15;
		kontekst.fillText(zaok(korakF,podeociF[i]),podX,podF);
	}

	kontekst.scale(sc/2,-vc/2);


	if(nt){
		grafik.selX=zaok(korak_x,koordGraf3d[ind][0]);
		grafik.selY=koordGraf3d[ind][1];
		grafik.selZ=koordGraf3d[ind][2];
		nacrtajTacku(koordGrafCanv[ind],4-(4-dy)/14);

		}

}//plot


this.stem =function(){

if(typeof grafik.rotX=="undefined"||
	typeof grafik.rotZ=="undefined"){
grafik.rotX=0;
grafik.rotZ=0;
}
if(typeof grafik.zoom=="undefined"){
grafik.zoom=145;}
grafik.rotX=ugao360(grafik.rotX);
grafik.rotZ=ugao360(grafik.rotZ);

grafik.type='stem';
dy=-4.5*100/grafik.zoom;

defKoordTacke2d()

if(typeof grafik.xmin=="undefined"||
	typeof grafik.xmax=="undefined"||
	typeof grafik.korak=="undefined"){
		var x_niz=grafik.x.split(":");
		xmin=parseFloat(x_niz[0]);
		korak_x=parseFloat(x_niz[1]);
		xmax=parseFloat(x_niz[2]);
	grafik.xmin=xmin;
	grafik.xmax=xmax;
	grafik.korak=korak_x;
}else{
```

```
      if(xmin!=grafik.xmin||
            xmax!=grafik.xmax||
            korak_x!=grafik.korak){
            grafik.selX="";
            grafik.selY="";
            grafik.selZ="";
            nt=false}

      xmin=parseFloat(grafik.xmin);
      xmax=parseFloat(grafik.xmax);
      korak_x=parseFloat(grafik.korak);

}

var pocetak=true;
var asimptota=false;
var x=xmin;
var fmm =[[],[],[],[],[],[]];
fmm[1][0]=maksFje(funkcijaF1);
fmm[1][1]=minFje(funkcijaF1);
fmm[2][0]=maksFje(funkcijaF2);
fmm[2][1]=minFje(funkcijaF2);
fmm[3][0]=maksFje(funkcijaF3);
fmm[3][1]=minFje(funkcijaF3);
fmm[4][0]=maksFje(funkcijaF4);
fmm[4][1]=minFje(funkcijaF4);
fmm[5][0]=maksFje(funkcijaF5);
fmm[5][1]=minFje(funkcijaF5);

      var minMaxPoc=1;
//    alert(maxMax);
for(iii=1;iii<6;iii++){
      if(typeof fmm[iii][0]=="undefined"||typeof
fmm[iii][1]=="undefined"){continue;}
      if(minMaxPoc==1){
            var maxMax=fmm[iii][0];
            var minMin=fmm[iii][1];
            minMaxPoc=0;
      }
      maxMax=Math.max(maxMax,fmm[iii][0]);
      minMin=Math.min(minMin,fmm[iii][1]);
}


      kontekst.clearRect(-1,-1,2,2);
      back();
      kontekst.beginPath();
      kontekst.lineWidth=2/vc*2.5;

var transformacije=[transliraj(-0.5,-0.5,-0.5),
                              rotirajOkoZ(0+grafik.rotZ),
                              rotirajOkoX(0+grafik.rotX),

                              transliraj(0.5,0.5,0.5)];

                              koordGraf3d=[];
```

```
koordGrafCanvNorm=[];
koordGrafCanv=[];
function IzracunajY(fjaa){
      var tempXYZ=[];
      var x=xmin;
      var iii=0;
      while(x<=xmax){
                  var y=fjaa(x);
                  var z=0.5;
                  var grafTacka=[];


      tempXYZ[iii]=[x,y,z];
      iii++;
      x+=korak_x;
      }
return tempXYZ;
}

var f1=IzracunajY(funkcijaF1);
var f2=IzracunajY(funkcijaF2);
var f3=IzracunajY(funkcijaF3);
var f4=IzracunajY(funkcijaF4);
var f5=IzracunajY(funkcijaF5);
var grafTacka=[];
var grafTackaX=[];

var mixBoja=["red","green","blue","orange","violet"]
var grafBojaID=0;
function crtanjeGrafika (tempfja){
kontekst.strokeStyle = mixBoja[grafBojaID];
grafBojaID++;
 kontekst.beginPath();
for(var iii=0;iii<tempfja.length;iii++){

koordGraf3d[iii]=tempfja[iii];
var x=tempfja[iii][0];
var y=tempfja[iii][1];
var z=tempfja[iii][2];

 grafTacka[0]=(tempfja[iii][0]-xmin)/(xmax-xmin);
 grafTacka[1]=0.5;
 grafTacka[2]=(tempfja[iii][1]-minMin)/(maxMax-minMin)


 grafTackaX[0]=(tempfja[iii][0]-xmin)/(xmax-xmin);
 grafTackaX[1]=0.5;
 grafTackaX[2]=(-minMin)/(maxMax-minMin);


 grafTacka=transormacijaXTacka(mnozenjeMatrica_n(transformacije),
                              grafTacka);


 grafTackaX=transormacijaXTacka(mnozenjeMatrica_n(transformacije),
                              grafTackaX);

            var xy2d=persp([grafTacka[0],
                                    grafTacka[1],
```

```
                                    grafTacka[2]]);

                        var x2d=persp([grafTackaX[0],
                                    grafTackaX[1],
                                    grafTackaX[2]]);

                                    koordGrafCanv.push([xy2d[0]*sc/2+sc/2,vc/2-
xy2d[1]*vc/2]);


      //    Pozicioniranje za početak iscrtavanja grafika
                        kontekst.beginPath();

      kontekst.arc(xy2d[0],xy2d[1],7/((sc+vc)/2)*grafik.zoom/100,0,2*Math.PI);

                        kontekst.moveTo(xy2d[0],xy2d[1]);

                        //    Iscrtavanje grafika

                        kontekst.lineTo(x2d[0],x2d[1]);
                        kontekst.stroke();


}



}//crtanjeGrafika

crtanjeGrafika(f1);
crtanjeGrafika(f2);
crtanjeGrafika(f3);
crtanjeGrafika(f4);
crtanjeGrafika(f5);



      // 3d transformacija koordinatnog sistema
      koordTacke=transormacijaXNizTacaka(mnozenjeMatrica_n(transformacije),
                              koordTacke);

      kontekst.lineWidth=2/vc;
      kontekst.strokeStyle ='black';

      // iscrtavanje linija koordinatnog sistema
      for(var i=0;i<4;i++){


                  kontekst.beginPath();

            crtLin(
      persp(koordTacke[mapaLinija[i][0]])[0],persp(koordTacke[mapaLinija[i][0]])
[1],

      persp(koordTacke[mapaLinija[i][1]])[0],persp(koordTacke[mapaLinija[i][1]])
[1]);

            kontekst.stroke();
            kontekst.beginPath();
```

```
	}

	//podela na 5 podeoka od 0 do 5
	var podeociX=podeli_ose(xmin,xmax,7);
	var podeociF=podeli_ose(minMin,maxMax,7);
	kontekst.beginPath();
	var tp2d2=[];
	var tp2d2Y=[];
	var tp2d2Z=[];
	for(var i=0;i<podeociX.length;i++){
		var tpt=(podeociX[i]-xmin)/(xmax-xmin);

		var koordTemp=koordTacke;
		defKoordTacke2d();
		var tp3d=[
	[tpt,koordTacke[mapaLinija[0][0]][1],koordTacke[mapaLinija[0][0]][2]],
				[tpt,koordTacke[mapaLinija[0][0]][1]-
0.01,koordTacke[mapaLinija[0][0]][2]-0.01]];
		koordTacke=koordTemp;
		tp3d=transormacijaXNizTacaka(mnozenjeMatrica_n(transformacije),
					tp3d);
		var tp2d1=persp(tp3d[0]);
		tp2d2[i]=persp(tp3d[1]);
		kontekst.moveTo(tp2d1[0],tp2d1[1]);
		kontekst.lineTo(tp2d2[i][0],tp2d2[i][1]);
	}

		var kp =[[-1,-1],[1,-1],[1,1],[-1,1]];
		for(var i=0;i<podeociF.length;i++){
		var tpt=(podeociF[i]-minMin)/(maxMax-minMin);
		var koordTemp=koordTacke;
		defKoordTacke2d();
		var tp3d=[  [koordTacke[2][0],koordTacke[2][1],tpt],

	[koordTacke[2][0]+kp[2][0]*0.01,koordTacke[2][1]+kp[2][1]*0.01,tpt]];
		koordTacke=koordTemp;
		tp3d=transormacijaXNizTacaka(mnozenjeMatrica_n(transformacije),
					tp3d);

		var tp2d1=persp(tp3d[0]);
		tp2d2Z[i]=persp(tp3d[1]);
		kontekst.moveTo(tp2d1[0],tp2d1[1]);
		kontekst.lineTo(tp2d2Z[i][0],tp2d2Z[i][1]);
	}

	kontekst.stroke();


	kontekst.scale(2/sc,-2/vc);
	var velFonta=vc/50-dy/-4.5;
	kontekst.font=velFonta+"px Arial";
	kontekst.textBaseline="top";
	kontekst.fillStyle="black";
	for(var i=0;i<podeociX.length;i++){
		var podX=tp2d2[i][0]*sc/2-kontekst.measureText(podeociX[i]).width/2;

		var podF=-tp2d2[i][1]*vc/2+velFonta*0.15;
		kontekst.fillText(podeociX[i],podX,podF);
```

```
	}

	for(var i=0;i<podeociF.length;i++){
		kontekst.textBaseline="top";
		var
podX=tp2d2Z[i][0]*sc/2+kontekst.measureText(zaok(korakF,podeociF[i])).width/2;
		kontekst.textBaseline="middle";
		var podF=-tp2d2Z[i][1]*vc/2+velFonta*0.15;
		kontekst.fillText(zaok(korakF,podeociF[i]),podX,podF);
	}

	kontekst.scale(sc/2,-vc/2);


	if(nt){
		grafik.selX=zaok(korak_x,koordGraf3d[ind][0]);
		grafik.selY=koordGraf3d[ind][1];
		grafik.selZ=koordGraf3d[ind][2];
		nacrtajTacku(koordGrafCanv[ind],4-(4-dy)/14);

		}

}//stem



var hh=0;
var korak;



	// Određivanje podeoka na osama
	function podeli_ose(min, max, broj_podeoka, xyz){

		var opseg = max - min;
		korak = Math.pow(10, Math.floor(Math.log(opseg / broj_podeoka)/
Math.log(10)));
		var greska = broj_podeoka / opseg * korak;

		// Zaokruživanje podeoka na pogodnije vrednosti
		if(greska <=.15) korak *=10;
		elseif(greska <=.35) korak *=5;
		elseif(greska <=.75) korak *=2;

		// Zaokruživanje početne i krajnje vrednosti
		var  start = Math.ceil(min / korak)* korak,
		      stop = Math.floor(max / korak)* korak + korak *0.5,
		      podeoci =[];
		//if (Math.abs(min/(start-korak))>0.999) {start=start-korak};

		if(xyz=="y") korakY=korak;
		if(xyz=="z") korakZ=korak;
		korakF=korak;
		// Upisivanje podeoka u niz
		for(var i=start; i < stop; i +=korak){
		      //if (i+korak>stopMath.abs((i+korak)/max)>0.999)
stop=(i+korak)*1.5;
		      //alert(Math.abs((i+korak)/max));
		      i=zaok(korak,i);
```

```
                podeoci.push(i);
            }
            return podeoci;
        }     //podeli_ose


}// Grafik
```