|  |  |  |
| --- | --- | --- |
| pmf-grb-lat-1in | UNIVERZITET U NOVOM SADU  PRIRODNO-MATEMATIČKI FAKULTET  DEPARTMAN ZA  MATEMATIKU I INFORMATIKU | uns-grb-1in |

Aleksandar Petrović

Seminarski rad

Crtanje grafika realne funkcije jedne realne promenljive

Novi Sad, *2014.*

# Sadržaj

1. Uvod 3

2. Rešenje zadatka 4

2.1. Tabeliranje funkcije 4

2.2. Konverzije kordinata 5

2.2.1. Konverzija iz Dekartovog kordinatnog sistema u WPF kordinatni sistem................. 5

2.2.2. Konverzija iz WPF kordinatnog sistema u Dekartov kordinatni sistem......................6

2.3. Implementacija 7

2.3.1. Tabeliranje funkcije 7

2.3.2. Crtanje 9

2.3.3. Detaljan opis implementacije koda 10

2.3.4. Korisnički interfejs 19

4. Zaključak 21

5. Literatura 22

6. Kratka biografija 23**1. Uvod**

Zadatak je da se napravi aplikacija koja će za proizvoljnu funkciju jedne realne promenljive iscrtati grafik te funkcije na intervalu [*x1*, *x2*] pri čemu se ymin i ymax automatski određuju.

Za pravljenju aplikacije koristi se Microsoft Visual Studio – WPF (“Windows Presentation Foundation”).

Koristi se elementarna matematika za realizaciju projekta, Dekartov kordinatni sistem, odnos dve tačke u ravni, duž između dve tačke, kao i transformacije izmedju dva kordinatna sistema – ekranskog i matematičkog.

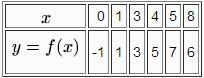
# 2. Rešenje zadatka

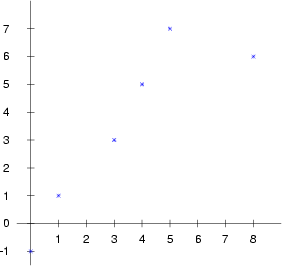
**2.1. Tabeliranje funkcije**

Kako bi uopšte i uspeli da icrtamo grafik funkcije u bilo kom programskom jeziku, potrebno je izraćunati tebelu vrednosti funkcije na posmatranom intervalu, time se dobija niz tačaka. Kada dobijemo niz tačaka, svake dve susedne spojimo jednom linijom i dobijamo ceo grafik funkcije. Da bi smo dobili tačke potrebno je da se krećemo na zadatom intervalu [*x1, x2*] po nekom veoma malom koraku (kako bi iscrtana funkcija bila što preciznija) i ubacimo svako *x[x1, x2]* u zadatu funkciju i dobijemo kordinatu *y.* Kordinate x i y predstavljaju jednu tačku *(x, y)* u Dekartovom kordinatnom sistemu.

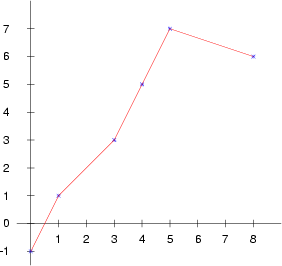
U nekim funkcijama je moguće imati neodređene tačke, tj tačke kod kojih je poznata x kordinata, a y je neodredjena ili je beskonačna. Kada se desi da je y beskonačnost (*infinity*) ili nije broj (*NaN – Not a Number*) u toj tački se grafik prekida i nastaju dva dela putanje grafika.

Sledi primer tabličnog crtanja u matematici.





Slika1. Tablično zadana funkcija

****

Slika 2. Povezane tačke

**2.2. Konverzije kordinata**

**2.2.1. Konverzija iz Dekartovog kordinatnog sistema u WPF kordinatni sistem**

Prozor u wpf aplikaciji ima svojstva Height i Width koja predstavljaju visinu i širinu prozora. Kao što je prikazano na slici dole, gde je *y* height, a *x* width. Međutim povećanje y vrednosti u wpf aplikaciji odgovara smanjenju y vrednosti u Dekartovom kordinatnom sistemu i obrnuto. Da bi se to rešilo uvedena je promenljiva *Ysredina* gore pomenuta koja vrši pretvaranje iz jednog kordinatnog sistema u drugi.

Primetimo da se *x* kordinata isto ponaša u oba sistema pa za nju nije potrebna konverzija.

x

y

x

y

Slika 3. Slika 4.

Prikaz kordinatnog sistema Prikaz Dekartovog

u wpf aplikaciji kordinatnog sistema

Pretvaranje se vrši na sledeći način:

Neka je P1 tačka u wpf kordinatnom sistemu sa kordinatama (*x1, y1*) i neka je *Ysredina* prava *y=0*. Neka je P2 tačka u Dekartovom kordinatnom sistemu sa kordinatama (*x2, y2*). Jednostavnim oduzimanjem *y-y1* dobijamo kordinatu *y2*, tj *y2=y-y1* i *x2=x1* što smo utvrdili da je isto pa dobijamo tačku P2(*x2, y2*) u Dekartovom kordinatnom sistemu.

**.** *P1(x1, y1)*

**.** *P2(x2, y2)*

Slika 5.Slika 6.

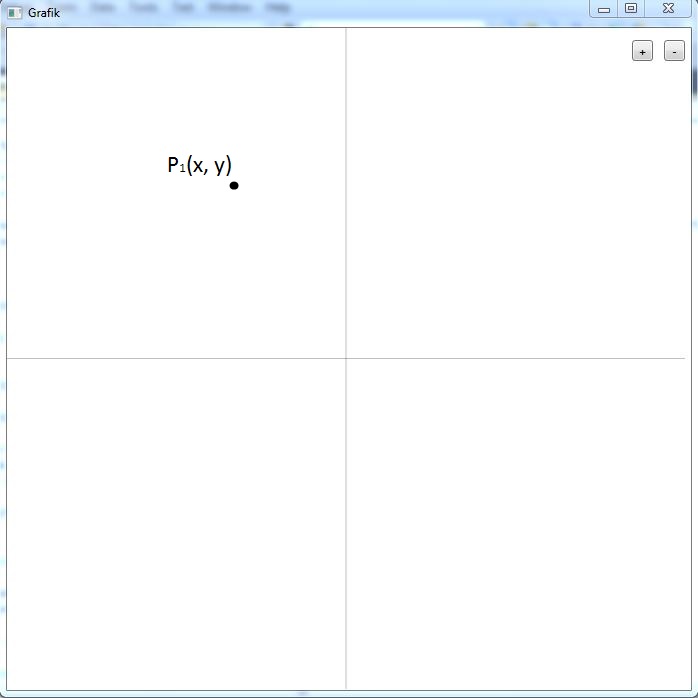
Funkcija preslikavanja tačke *P1(x, y)* iz Dekartovog kordinatnog sistema u wpf kordinatni sistem je sledeća.

*f(P1(x, y)) P2(x, Ysredina-y)*

Gde je *Ysredina* promenljiva koja predstavlja nulu u Dekartovom kordinatnom sistemu a u wpf kordinatnom sistemu predstavlja sredinu prozora, tj visinu/2. Kako je u aplikaciji omogućeno menjanje opsega na kome se posmatra grafik u odnosu na y osu pa se samim tim i Ysredina menja.

**2.2.2. Konverzija iz WPF kordinatnog sistema u Dekartov kordinatni sistem**

Funkcija koja pretvara iz wpf u Dekartov kordinatni sistem je ustvari inverzna funkcija od funkcije koja pretvara iz Dekartov u wpf kordinatni sistem



Slika 7. Prikaz tačke

Neka je P1(x, y) tačka u wpf kordinatnom sistemu, tada funkcija za preslikavanje u Dekartovom kordinatnom sistemu je.

*f(P1(x, y)) P2(X, Y)*

*X=X1+x\*((X2-X1)/canvas.Width*

*Y=(Ysredina-p.Y)\*(Y2 – Y1) /canvas.Height*

Gde su *X1* i *X2*granične vrednosti intervala na kome se crta grafik u odnosu na x-osu, a *Y1*i *Y2* granične vrednosti za y-osu. Vrednosti canvas.Width i canvas.Heigth predstavljaju širinu i visinu prozora.

**2.3. Implementacija**

Sada kada smo odredili matematički deo problema treba još samo to implementirati u programski jezik. Kao što sam već ranije napomenuo tehnologiju koju ću koristiti je *WPF.* Pokazala se kao veoma dobra pri predstavljanju grafičkih objekata u *windows* aplikacijama. Klase korišćene u implementaciji programa su: *Canvas, Point, Line, PathFigureCollection, PathSegmentCollection, LineSegment, PathFigure, PathGeometry, Path.*

*Canvas* predstavlja platno za crtanje, tj na njemu su predstavljeni grafićki elementi.

*Point* je klasa pomoću koje se kreiraju tačke u dvodimenzionalnom kordinatnom sistemu.

*Line* predstavlja liniju koja ima dve tačke, početak i kraj.

*PathFigure* je klasa pomoću koje se kreira objekat figure, tj niz tačaka koji spajanjem predstavlja jedan grafik u ravni.

*LineSegment* kreira liniju izmedju dve tačke u *PathFigure.*

*PathSegmentCollection* predstavlja kolekciju *LineSegment* objekata.

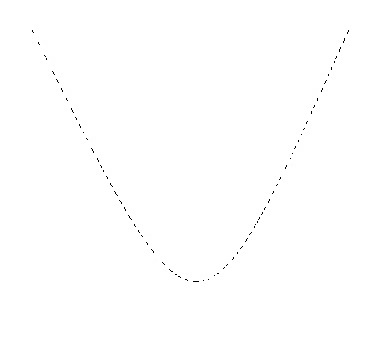
*PathFigureCollection* je klasa koja predstavlja kolekciju *PathFigure* objekatakoji svi zajedno čine jedan *PathGeometry.*

*PathGeometry* predstavlja komleksan objekat sastavljen od elipsi, krugova, pravougaonika,...

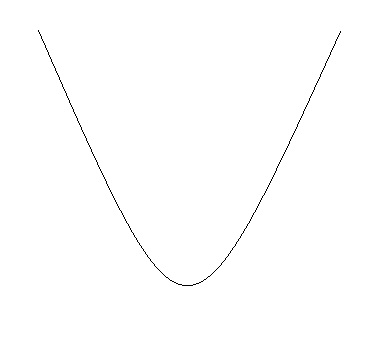
**2.3.1. Tabeliranje funkcije**

Program se izvršava tako što se odrede sve tačke na intervalu [*Xmin, Xmax*] odredjene funkcije sa veoma malim razmakom izmedju njih da bi se dobilo što više tačaka koje bi se spojile, i tako dobijemo precizniju sliku.

Sve te tačke se ubace u niz tačaka koji se kasnije koristi tako što se te tačke ubacuju u posebnu klasu koja sve te tačke spaja.

****

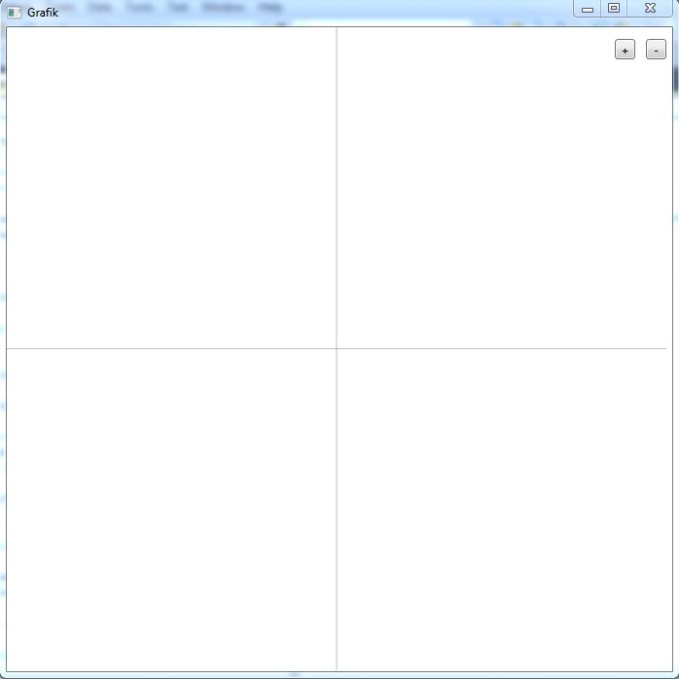
Slika 8. Prikaz tačaka



Slika 9. Prikaz grafika

Slike pokazuju transformaciju, na prvoj slici (Slika 8.) se nalaze tačke koje su na drugoj slici (Slika 9.) spojene linijama i dobija se ceo grafik funkcije *f(x)=*

Width

****

H

e

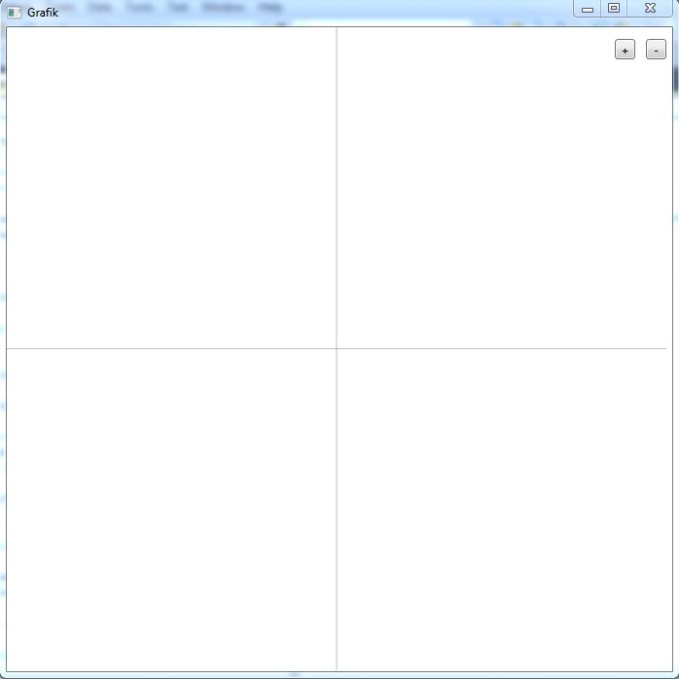
i

g

t

h

Slika 10. Prikaz prozora



Y2

Y1

X1

X2

Slika 11. Prikaz intervala na kome se crta funkcija

Kao što se vidi na slici 10. prozor ima neku svoju širinu i visinu. A mi smo izračunali niz tačaka na intervalu [*x1, x2*], sada bi trebalo pretvoriti da x1 bude početna tačka prozora tj kada je *width=0, a x2* treba da bude width od prozora, što je predstavljeno na slici 11.

Interval [y1, y2] je unapred odredjen i menja se kada se menja i prikaz na kome korisnik želi da vidi datu funkciju.

**2.3.2. Crtanje**

Kao što je već opisano u poglavlju 2.3.1. kako se funkcija tabelira i ubacuje u niz, sada taj niz trebamo iskoristi.

Svaka tačka u nizu je određena kordinatama (*x,y*), međutim svaka kordinata je na intervalu [*x1, x2*]. Da bi ih prebacio u interval [*0, width*] potrebno je kretati se od 0 do width po nekom koraku koji je 1 i ubacujem tačke na poziciji (*dosadašnji korak, y\*odnosY*) gde se *dosadašnji korak* svaki put povečava za *korak*, a *odnosY* predstavlja *heigth(*visina*)/(y2-y1)*. Dodaje se drugoj kordinati još i *Ysredina-y\*odnosY* gde je *Ysredina=height/2* što ujedno predstavlja na grafiku nulu y ose, tj kad je *y=0* u Dekartovom kordinatnom sistemu.

Opisano je već da u nekim situacijama vrednost y kordinate postane (*infinity*) ili (*NaN – Not a Number*) pa se u tim situacija grafik pretvara u dve nezavisne putanje.

Pomoću *for* petljeunutar druge *for* petlje ostvarujem podelu na više putanja.

Prvo se kreira objekat klase *PathFigureCollection* koji je namenjen za čuvanje svih putanja. Ispituje se da li je vrednost niza u trenutnoj iteraciji za *y* kordrinatu  *infinty* ili *NaN,* ako jeste ta tačka se preskače i nastavlja se od sledeće iteracije. Ako nije onda se pravi novi objekat klase *PathSegmentCollection* u koji se novom *for* petljom ubacuju tačke sve dok se ne stigne do kraja niza ili dok se ne naidje na novu nedozvoljenu[[1]](#footnote-1) vrednost. Kada se završi ugnježdena petlja dobija se deo putanje ili cela putanja grafika *funkcije* i ona se ubacuje u novi objekat klase *PathFigure* koji predstavlja tu putanju tj deo celog grafika. Tu putanju ubacujemo u već kreirani objekat kolekcije putanja *PathFigureCollection* i tako sve dok se ne stigne do kraja niza. Kada se završe sve iteracije spoljašnje petlje dobijamo kolekciju putanja koju možemo sa lakoćom iscrtati na platno *canvas.*

**2.2.3. Detaljan opis implementacije koda**

Kada se pokrene aplikacija inicijalizujemo sledeće promenljive koje su predstavljene kao globalne.

double Xmin, Xmax, Ymin, Ymax;

double Ysredina;

Point lokacijaPointera;

bool pomeraj;

bool prvi;

Label xyPokazivac;

Label xyStatusBar;

ContextMenu meni;

InitializeComponent() – ugrađena metoda od strane wpf aplikacije koja inicijalizuje sve komponente potrebne za grafičko prikazivanje prozora na ekranu.

|  |  |
| --- | --- |
| *Xmin, Xmax* | označavaju interval na kome se crta funkcija po x-osi |
| *Ymin, Ymax* | označavaju interval na kome se crta funkcija po y-osi |
| *Ysredina* | označava y kordinatu u kordinatnom sistemu kad je y=0 |
| *lokacijaPointera* | označava trenutnu lokaciju u wpf kordinatnom sistemu |
| *pomeraj* | kada se drži levi klik miša i pomera miš ostvaruje se pomeranje intervala na kome se posmatra grafik funkcije, pa se pomoču ove promenljive saznaje da li je desni klik miša pritisnut |
| *prvi* | označava promenljivu pomoću koje se reguliše *Ysredina* u događaju *Window\_SizeChanged,* jer dolazi do pogrešne dodele vrednosti za *Ysredina* kada se pokrene program |
| *xyPokazivac* | predstavlja *Label* u kome je upisana trenutna lokacija pokazivača miša u Dekartovom kordinatnom sistemu, njeno prikazivanje u aplikaciji se ostvaruje pritiskom na dugme tastature „P“ |
| *xyStatusBar* | predstavlja isto što i *xyPokazivac* stim što je ta labela u gornjem levom uglu sve dok je pokazivač miša u okviru aplikacije |
| *meni* | označava meni koji se pokreće desnim klikom miša, on sadrži stavku „Save to file“ pomoću koje se pretvara trenutni grafik funkcije u sliku i smešta u fajl |

public MainWindow()

{

InitializeComponent();

Xmin = Ymin = -5;

Xmax = Ymax = 5;

Ysredina = canvas.Height / (double)2;

prvi = true;

xyStatusBar = new Label();

canvas.Children.Add(xyStatusBar);

ContextMenu meni = new ContextMenu();

MenuItem mi = new MenuItem();

mi.Header = "File";

meni.Items.Add(mi);

canvas.ContextMenu = meni;

inicMeni();

CrtajFunkciju();

}

Inicijalizuju se sve globalne promenljive, kreira se padajući meni na desni klik miša.

void inicMeni()

{

meni = new ContextMenu();

MenuItem item = new MenuItem();

item.Header = "Save to file";

meni.Items.Add(item);

item.Click += new RoutedEventHandler(item\_Click);

canvas.ContextMenu = meni;

}

I na kraju main metoda se poziva metod CrtajFunkciju()

Ima ulogu crtanja funkcije na platno(*Canvas*).

void BrisiSveLinije()

{

Type put = typeof(Path);

Type linija = typeof(Line);

Type label = typeof(Label);

for (int i = canvas.Children.Count - 1; i >= 0; i--)

{

if (canvas.Children[i].GetType() == put || canvas.Children[i].GetType() == linija)

canvas.Children.RemoveAt(i);

}

}

Prvo se očisti ekran od svih prethodni iscrtanih linija pomoću metoda *BrisiSveLinije*(), zatim se u *niz* ubacuju y kordinate izračunate u metodi *funkcija*() za svako x na intervalu [*Xmin, Xmax*] po nekom malom koraku, tj koraku koji predstavlja *(Xmax-Xmin)/sirina canvasa*

double funkcija(double x)

{

return x\*x;

}

Koriste se gotove klase za crtanje na canvasu

* *LineSegment*
* *PathSegmentCollection*
* *PathFigure*
* *PathFigureCollection*
* *PathGeometry*
* *Path*

Prolazim kroz niz gore napravljenih kordinata i postavljam tačke u *LineSegment* objekat koji predstavlja tačku sa kordinatama (x, y). *LineSegment* objekati se postavljaju u *PathSegmentCollection* koji predstavlja kolekciju više tačaka. *PathFigure* predstavlja skup linija koje spajaju tačke iz *PathSegmentCollection* i tako se dobija deo grafika, neisprekidana linija skupa tačaka. Svaka ta figura se ubacuje u *PathFigureCollection* koji predstavlja kolekciju više figura. Kolekcija tih figura se ubacuje u *PathGeometry* koja predstavlja celokupan grafik prikazan u aplikaciji. I na kraju se kolekcija *PathGeometry* ubacuje u *Path* koja ima polja kojima može da se podesi debljina linije, boja linije...

*Path* se ubacuje u *canvas koji to iscrtava.*

Metoda *CrtajKordinatni*() iscrtava *x=0 i y=0*, tj crta kordinatni sistem.

void CrtajKordinatni()

{

if (0 > Xmin && 0 < Xmax)

{

double podela = canvas.Width / (Xmax - Xmin);

Line vertikalna = new Line();

vertikalna.X1 = podela \* (0 - Xmin);

vertikalna.X2 = podela \* (0 - Xmin);

vertikalna.Y1 = 0;

vertikalna.Y2 = canvas.Height;

vertikalna.Stroke = System.Windows.Media.Brushes.Black;

vertikalna.StrokeThickness = 0.3;

canvas.Children.Add(vertikalna);

}

Line horizontala = new Line();

horizontala.X1 = 0;

horizontala.X2 = canvas.Width;

horizontala.Y1 = Ysredina;

horizontala.Y2 = Ysredina;

horizontala.Stroke = System.Windows.Media.Brushes.Black;

horizontala.StrokeThickness = 0.3;

canvas.Children.Add(horizontala);

}

private void Window\_SizeChanged(object sender, SizeChangedEventArgs e)

{

canvas.Width = this.Width - 22;

canvas.Height = this.Height - 39;

if (prvi)

{

Ysredina = canvas.Height / (double)2;

prvi = false;

}

else

{

Point p = (Point)e.PreviousSize;

Point n = (Point)e.NewSize;

Ysredina += n.Y - p.Y;

}

CrtajFunkciju();

}

Na događaj, kada se promeni velićina prozora, izvršava se prethodni kod. Gde promenljiva *prva* tipa bool predstavlja da li se veličina prozora prvi put promenila jer kad bi bilo bez tog dela koda dolazilo bi do greške, tj Ysredina bi dobila neku lošu vrednost jer kad se pokrene program tada se vrši promena veličine prozora programski u onom delu koda *InitializeComponent*() i tada se isto poziva ovaj događaj *Window*\_*SizeChanged* koji sam ja redefinisao*.* Pa sam zato uveo promenljivu *prvi* koja registruje prvi put pozivanje tog događaja.

private void Window\_MouseWheel(object sender, MouseWheelEventArgs e)

{

if (Keyboard.IsKeyDown(Key.LeftCtrl))

{

if (e.Delta > 0)

{

double smanji = ((Xmax - Xmin) / (double)2) / (double)2;

Xmax -= smanji;

Xmin += smanji;

CrtajFunkciju();

}

else if (e.Delta < 0)

{

double povecaj = ((Xmax - Xmin) / (double)2);

Xmax += povecaj;

Xmin -= povecaj;

CrtajFunkciju();

}

}

else if (Keyboard.IsKeyDown(Key.LeftAlt))

{

if (e.Delta > 0)

{

double smanji = ((Ymax - Ymin) / (double)2) / (double)2;

Ymax -= smanji;

Ymin += smanji;

CrtajFunkciju();

}

else if (e.Delta < 0)

{

double povecaj = ((Ymax - Ymin) / (double)2);

Ymax += povecaj;

Ymin -= povecaj;

CrtajFunkciju();

}

}

else

{

if (e.Delta > 0)

{

double smanji = ((Xmax - Xmin) / (double)2) / (double)2;

Xmax -= smanji;

Xmin += smanji;

smanji = ((Ymax - Ymin) / (double)2) / (double)2;

Ymax -= smanji;

Ymin += smanji;

CrtajFunkciju();

}

else if (e.Delta < 0)

{

double povecaj = ((Xmax - Xmin) / (double)2);

Xmax += povecaj;

Xmin -= povecaj;

povecaj = ((Ymax - Ymin) / (double)2);

Ymax += povecaj;

Ymin -= povecaj;

CrtajFunkciju();

}

}

}

Događaj *Window*\_*MouseWheel* detektuje pomeranje točkića miša. Kada se to dogodi u promenljivu *e* se upiše vrednost koja je >0 ili <0 tada je korisnik okrenuo točkić za napred ili nazad, kada je *e=0* tada se točkić nije okrenuo. U tom događaju se izvršava promena amplitude na klik tastera levi ctrl+okretanje točkića i u odnosu na *e* se povećava ili smanjuje amplituda. Tu se takođe izvršava i promena periode na levi alt+okretanje točkića i u odnosu na *e* se smanjuje ili povećava perioda.

Kada se ne drži pritisnutim levi ctrl ili levi alt, a okreće se točkić miša onda se menja interval na kome se posmatra funkcija tj smanjuje se ili se povećava interval u odnosu na promenljivu *e.* Kada se zumira onda se *x1* i *x2* smanjuju za po četvrtinu od njihove razdaljine *x2-x1*, a kada se odzumira onda se *x1 i x2* povećavaju za po polovinu od njihove razdaljine *x2-x1*.

Klikom na dugmiće + i – na prozoru se postize isti efekat kao i okretanjem točkića miša.

Događaji *Window*\_*MouseLeftButtonDown*, *Window*\_*MouseLeftButtonUp*, *Window*\_*MouseMove*, *Window*\_*MouseLeave* omogućuju kada korisnik klikne na prozor i pomera pokazivač miša na prozoru da menja interval u kome želi da vidi funkciju.

Dogadjaj *Window*\_*MouseLeftButtonDown* omogučuje registrovanje da li je korisnik pritisnuo levi taster miša, to omogučava pomeranje intervala na kome se posmatra funkcija.

Kada korisnik prestane da drži levi klik miša prestaje i menjanje intervala na kome se posmatra funkcija, to se postiže registrovanje dogadjaja *Window\_MouseLeftButtonUp.*

Kako smo postigli osluškivanje da li je korisnik pritisnuo levi taster miša sada to možemo koristiti u dogadjaju *Window\_MouseMove* koji registruje da li se pokazivač miša pomera. Kada se pomera pokazivač miša i još i kada je pritisnut levi taster miša menja se interval na kome se posmatra funkcija.

private void Window\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

lokacijaPointera = e.GetPosition(this);

pomeraj = true;

}

private void Window\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

pomeraj = false;

}

private void Window\_MouseMove(object sender, MouseEventArgs e)

{

if (pomeraj && (e.GetPosition(this).X - lokacijaPointera.X != 0 || e.GetPosition(this).Y - lokacijaPointera.Y != 0))

{

double x = (e.GetPosition(this).X - lokacijaPointera.X) / canvas.Width \* (Xmax - Xmin);

Xmin -= x;

Xmax -= x;

Ysredina += e.GetPosition(this).Y - lokacijaPointera.Y;

CrtajFunkciju();

}

lokacijaPointera = e.GetPosition(this);

Point m = Mouse.GetPosition(this);

xyStatusBar.Margin = new Thickness(0, 0, 0, 0);

double w = Math.Round(Xmin + m.X \* ((Xmax - Xmin) / canvas.Width), 4);

double h = Math.Round((Ysredina - m.Y) \* (Ymax - Ymin) / canvas.Height, 4);

xyStatusBar.Content = "(" + w + ", " + h + ")";

}

private void Window\_MouseLeave(object sender, MouseEventArgs e)

{

pomeraj = false;

}

U dogadjaju *Window\_MouseMove* se još i menja vrednost labele *xyStatusBar* u kojoj se upisuje trenutna lokacija pokazivača miša, tj vrednost *x, y* kordinate na grafiku.

Omogućeno je da se trenutna kordinata prikaže pored pokazivača miša u labeli. Klikom na taster „P“ tastature pojavljuje se labela sa trenutnim kordinatama pokazivača miša.

private void prozor\_KeyDown(object sender, KeyEventArgs e)

{

if(Keyboard.IsKeyDown(Key.P))

prikaziKordinate(Mouse.GetPosition(this));

if (Mouse.LeftButton == MouseButtonState.Released)

pomeraj = false;

}

Pozivom metode *prikaziKordinate()* se izvršava sledeći programski kod koji kao argument dobija trenutnu lokaciju pokazivača miša. Metodom se kreira labela u kojoj se funkcijom (slika 12) upisuje vrednost kordinate

*(x, y),* tj trenutna lokacija pokazivača na grafiku funkcije.

*f(P1(x, y)) P2(X, Y)*

*X=X1+x\*((X2-X1)/canvas.Width*

*Y=(Ysredina-p.Y)\*(Y2 – Y1) /canvas.Height*

Slika 12. Funkcija za pretvaranje ekranskih (wpf) kordinata u Dekartove kordinate

void prikaziKordinate(Point p)

{

canvas.Children.Remove(xyPokazivac);

xyPokazivac = new Label();

double x = Math.Round(Xmin + p.X \* ((Xmax - Xmin) / canvas.Width), 4);

double y = Math.Round((Ysredina - p.Y) \* (Ymax - Ymin) / canvas.Height, 4);

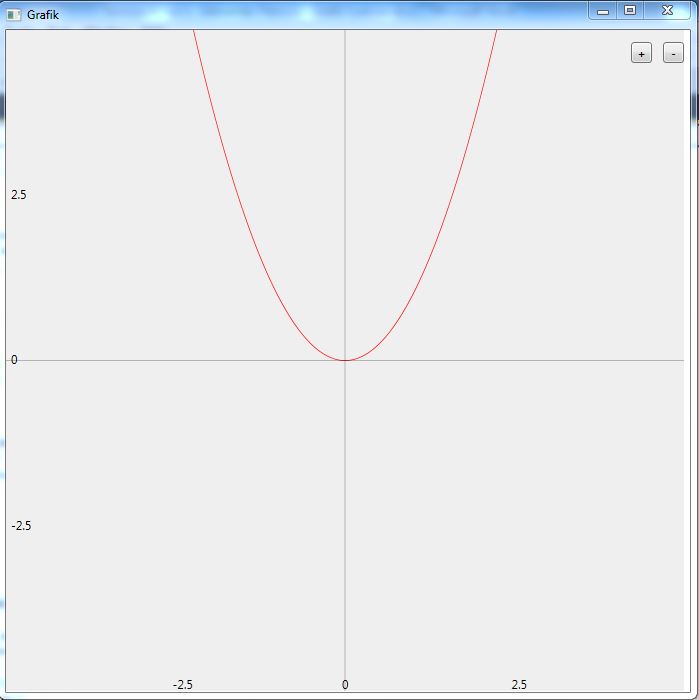
xyPokazivac.Content = "("+x+", "+ y+")";

xyPokazivac.Margin = new Thickness(p.X, p.Y - 20, 0, 0);

canvas.Children.Add(xyPokazivac);

}

Na slici 13 su prikazane kordinate koje omogućavaju korisniku lakše posmatranje grafika na odredjenom intervalu.



Slika 13. Prikaz funkcije f(x)=x2 i kordinate

na levoj i donjoj ivici prozora

Metoda koja se poziva za crtanje kordinata na levoj i donjoj strani prozora je *crtajKordinate().*

void crtajKordinate()

{

for (int i = canvas.Children.Count - 1; i >= 0; i--)

{

if (canvas.Children[i].GetType() == typeof(Label) && canvas.Children[i]!=xyStatusBar)

canvas.Children.RemoveAt(i);

}

double j = Xmin + (Xmax - Xmin) / 4;

double odnosx = (Xmax - Xmin) / 4;

double xosa = (canvas.Width) / 4;

for (int i = 0; i < 3; i++)

{

Label l = new Label();

l.Margin = new Thickness(xosa-8, canvas.Height-20, 0, 0);

xosa += (canvas.Width) / 4;

l.Content = Math.Round(j, 4);

j += odnosx;

canvas.Children.Add(l);

}

j = canvas.Height / 4;

double odnosy = (Ymax - Ymin) / 4;

double yosa = (canvas.Height) / 4;

for (int i = 0; i < 3; i++)

{

Label l = new Label();

l.Margin = new Thickness(0, yosa-14, 0, 0);

yosa += (canvas.Height) / 4;

l.Content = Math.Round((Ysredina - j) \* (Ymax - Ymin) / canvas.Height, 4);

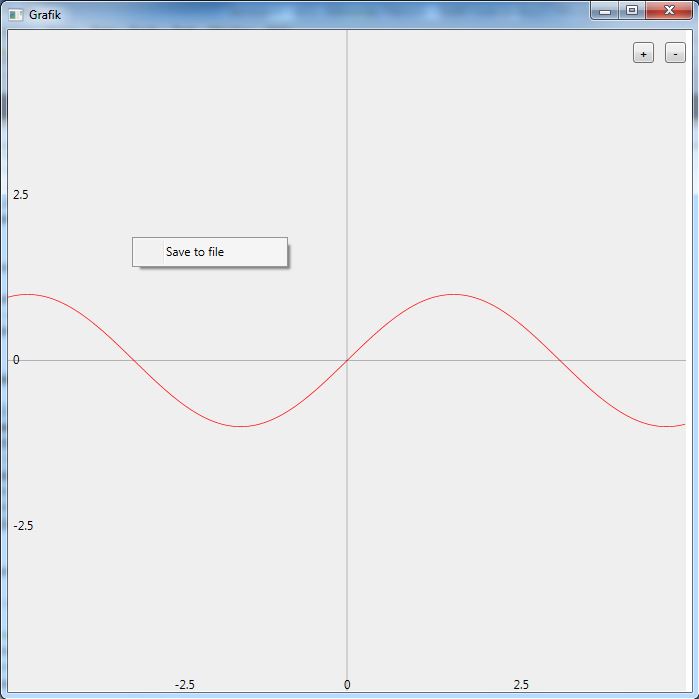
j += canvas.Height / 4;

canvas.Children.Add(l);

}

}

Klikom na desni taster miša otvara se padajući meni kao na slici 14.



Meni

Slika 14. Prikaz padajućeg menija

Detektovanje da li je korisnik pritisnuo desni taster miša vrši se sledećom metodom

private void prozor\_MouseRightButtonDown(object sender, MouseButtonEventArgs e)

{

meni.PlacementTarget = this;

meni.IsOpen = true;

}

Kada je otvoren padajući meni ćeka se sledeća akcija korisnika, tj da korisnik klikne levim tasterom miša na stavku Save to file. Kada je korisnik kliknuo na Save to file stavku izvršava se sledeći programski kod.

private void item\_Click(object sender, RoutedEventArgs e)

{

SaveFileDialog save = new SaveFileDialog();

save.FileName = "Slika";

save.DefaultExt = ".png";

save.Filter = "Bitmap|\*.png";

save.ShowDialog();

String str = @save.FileName;

Uri adresa = new Uri(str);

ExportToPng(adresa, canvas);

}

Metoda *ExportToPng* kreira sliku grafika funkcije u odredjenom fajlu izvršavanjem sledećeg koda

public void ExportToPng(Uri path, Canvas surface)

{

if (path == null) return;

Transform transform = surface.LayoutTransform;

surface.LayoutTransform = null;

Size size = new Size(surface.Width, surface.Height);

surface.Measure(size);

surface.Arrange(new Rect(size));

RenderTargetBitmap renderBitmap =

new RenderTargetBitmap(

(int)size.Width,

(int)size.Height,

96d,

96d,

PixelFormats.Pbgra32);

renderBitmap.Render(surface);

using (System.IO.FileStream outStream = new System.IO.FileStream(path.LocalPath, System.IO.FileMode.Create))

{

PngBitmapEncoder encoder = new PngBitmapEncoder();

encoder.Frames.Add(BitmapFrame.Create(renderBitmap));

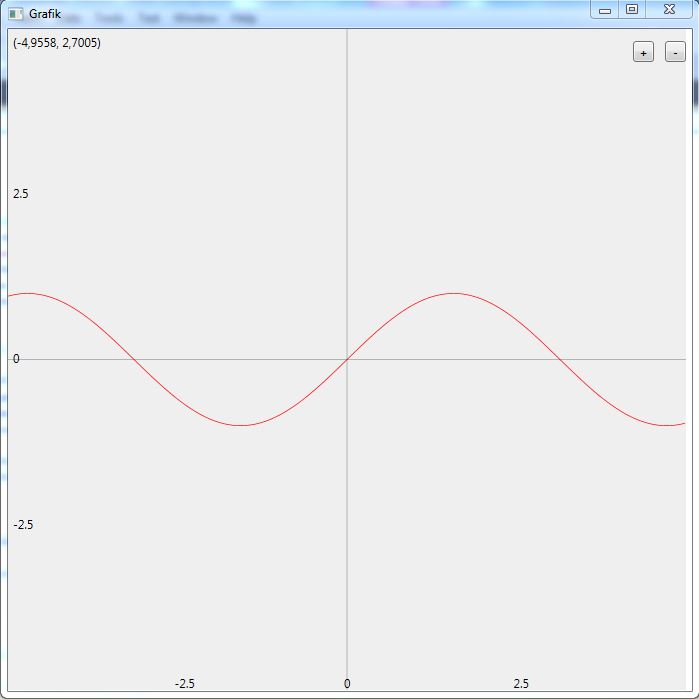
encoder.Save(outStream);

}

surface.LayoutTransform = transform;

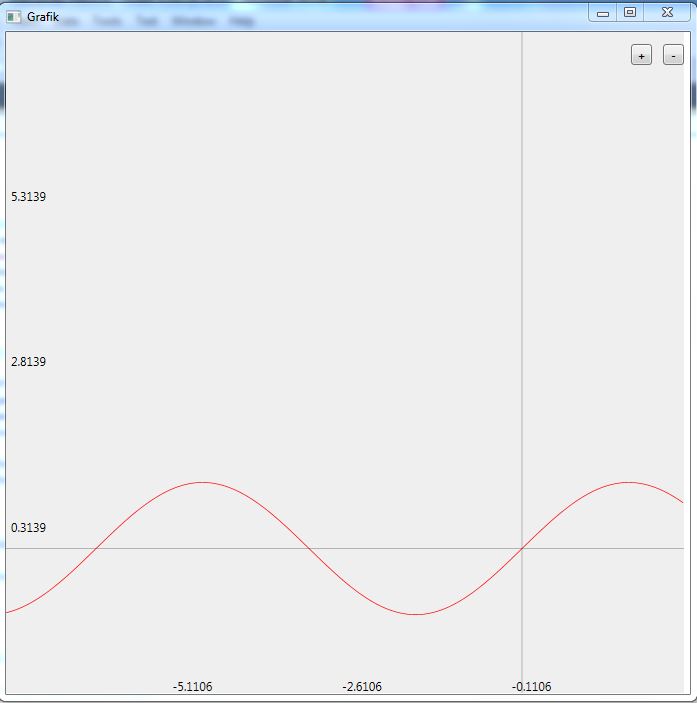
}

**Primer funkcije *f(x)=sinx***



Slika 14. Pre pomeranja

Levim klikom na prozoru aplikacije i držanjem istog, zatim pomeranjem pokazivača miša na dole-desno dijagonalno dobija se sledeća slika.



Slika 15. Nakon pomeranja

**2.** **Korisnički intefejs**

1

# Grafikprimer

2

# Slika 16. Prikaz aplikacije

Na slici iznad je prikazan prozor aplikacije u koji je iscrtana funkcija**,** koja služi samo kao primer kako je aplikacija prikazana grafički

1. Dugme koje služi za smanjenje trenutnog intervala na kom se funkcija crta, takozvani (Zoom-In)

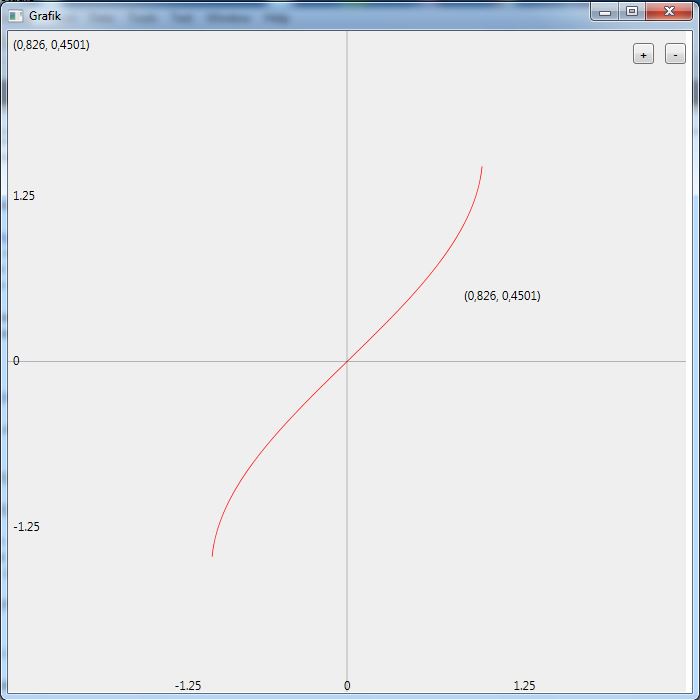
2. Dugme koje služi za od uvećanje trenutnog intervala na kom se funkcija crta, takozvani (Zoom-Out)

# Postoje i skraćenice sa tastature, takozvani (Hotkeys) pomoću kojih se ostvaruje isti efekat. Okretanjem točkića miša ostvaruje se zumiranje, isti efekat kao i dugmićima + i – prikazanim gore na slici 16.

# Skraćenica sa tastature pomoću koje se menja amplituda je levi alt+točkić miša, moguće je menjati i periodu levi ctrl+točkić miša.

Levim klikom na prozor i pomeranjem pokazivača miša dobija se promena intervala na kome se grafik funkcije iscrtava.

Pritiskom tastera „P“ na tastaturi ostvaruje se prikaz trenutne lokacije pokazivača miša na grafiku u Dekartovim kordinatama.



Slika 17. Funkcija f(x)=arcsin(x) i prikaz trenutne

lokacije pokazivača miša u odnosu na grafik

# Pozicioniranjem strelice miša na ivicu prozora moguće je povećavati i smanjivati prozor.

**4. Zaključak**

Ovo je prva verzija programa u kojoj je omogučeno crtanje funkcije zadate programski u samom kodu programa, menjanje intervala na kome se posmatra data funkcija, mogućnost promene amplitude i periode.

U narednoj verzija bi trebalo da se promeni unos funkcije, tj da se to menja na samom prozoru pomoću „formula parser“ tj metod koji će moći da iscrta bilo koju funkciju koju korisnik unese u text boxu na ekranu.

# Literatura

1. <http://msdn.microsoft.com/en-us/library/ms754130(v=vs.110).aspx>

Biblioteke svih klasa u wpf-u i primeri rada nekih metoda.

2. Wikipedia

3. Google Search

4. http://stackoverflow.com/

**Kratka biografija**

|  |  |
| --- | --- |
| Lični podaci | Aleksandar Petrović |
|  | |
|  | Petrijevska 75, 11300 Smederevo (Srbija) |
| 0264612183  0606811803 |
| acapetsd@hotmail.com |
| Pol Muški | Datum rođenja 10/05/1993 | Državljanstvo Srpsko |

Obrazovanje:

* Prirodno-matematički fakultet, Novi Sad(u toku, II godina)
  + Stručni naziv: Diplomirani informatičar
  + 2012. –
* Tehnička škola Smederevo, Smederevo
  + Smer: Elektrotehničar računara
  + 2008. – 2012.

Novi Sad, *2014.* Aleksandar Petrović

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

UNIVERZITET U NOVOM SADU

PRIRODNO - MATEMATIČKI FAKULTET

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj: RBR

Identifikacioni broj:

IBR

Tip dokumentacije: Monografska dokumentacija

TD

Tip zapisa: Tekstualni Stampani materijal

**TZ**

Vrsta rada:

**VR**

Autor:

AU

Mentor:

MN

Naslov rada:

MR

Jezik publikacije: *Srpski (latinica)*

**JP**

Jezik izvoda: *s / e*

JI

Zemlja publikovanja: SR Jugoslavija

ZP

Uže geografsko područje: Vojvodina

**UGP**

Godina:

**GO**

Izdavač: Autorski reprint

IZ

Mesto i adresa:

MA

Fizički opis rada: (*broj poglavlja/ broj strana/ broj lit. citata/ broj tabela/ broj slika/ broj grafika broj priloga*)

**FO**

Naučna oblast:

**NO**

Naučna disciplina:

**ND**

Ključne reči:

PO

**UDK:**

Čuva se:

ČU

Važna napomena:

VN

Izvod:

IZ

Datum prihvatanja teme od strane NN veća:

DP

Datum odbrane:

**DO**

**Članovi komisije:**

KO

Predsednik:

Član:

Član:

UNIVERSITY OF NOVI SAD

FACULTY OF SCIENCE KEY  
WORDS DOCUMENTATION

Accession number:

**ANO**

Identification umber:

**INO**

Document type: Monograph type

DT

Type of record: Printed text

**TR**

Contents Code:

CC

Author:

AU

Mentor:

MN

Title:

XI

Language of text:

LT

Language of abstract:

LA

Country of publication: Yugoslavia

CP

Locality of publication: Vojvodina

LP

Publication year:

**PY**

Publisher: Author's reprint

PU

Publ. place:

PP

Physical description: (*broj poglavlja, broj strana, broj lit. citata, broj tabela, broj slika, broj grafika, broj priloga*)

PD

Scientific field:

SF

Scientific discipline:

Key words:

UC:

Holding data:

HD Note:

Abstract:  
AB

Accepted by the Scientific Board on:

Defended:

Thesis defend board:

Member:

Member:

1. Infinity ili NaN vrednost [↑](#footnote-ref-1)