SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

Milan Đurić

Diplomski rad

IZRADA INTERAKTIVNE ANIMACIJE U JAVI ZA SIMULIRANJE GIBANJA I SUPERPONIRANJA VALNIH PAKETA

Zagreb,2006

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

SMJER: PROF. FIZIKE I INFORMATIKE

Milan Đurić

Diplomski rad

IZRADA INTERAKTIVNE ANIMACIJE U JAVI ZA SIMULIRANJE GIBANJA I SUPERPONIRANJA VALNIH PAKETA

Voditelj diplomskog rada: Doc.dr.sc Mirko Planinić

Ocjena diplomskog rada: _____

Povjerenstvo:

1. _____
2. _____
3. _____

Datum polaganja: _____

Zagreb,2006

Uvod 6

1. Valovi 8

1.1 Uvod 8

1.2 Harmonijski oscilator 9

1.3 Valovi i oscilacije 11

1.4 Vrste valova 12

1.5 Valna funkcija 14

1.6 Valna jednadžba 17

1.7 Superpozicija valova 20

1.8 Valni paketi 23

1.8.1 Matematički opis valnih paketa 24

1.8.1.1 Gaussijanski valni paketi 24

1.8.1.2 Superpozicija Gaussijanskih valnih paketa 28

1.8.1.3 Trokutasti valni paketi 33

1.8.1.4 Superpozicija trokutastih valnih paketa 38

2. Programski jezik Java 40

2.1 Java 40

2.2 Karakteristike Jave 40

2.3 Izvršavanje programa 42

2.4 Java development Kit (JDK) 44

2.5 Objektno-orijentirano programiranje 45

2.6 Java komponente 45

2.6.1 AWT(Abstract Window Toolkit) 45

2.6.2 AWT kontejneri 45

2.6.2.1 Panel 46

2.6.3 AWT komponente 47

2.6.3.1 Gumbi(Buttons) 47

2.6.3.2 Labela(Label) 47

2.6.3.3 Izbor(Choice) 48

2.6.4 Događaji 49

2.7 Grafika u Javi 50

2.7.1 Crtanje grafičkih oblika 51

2.7.2 Crtanje teksta 51

2.7.3 Klasa Font 51

2.7.4 Korištenje slika u Javi 53

2.8 Java applet 54

2.8.1 Prevođenje java appleta 54

2.8.2 Pokretanje Java appleta 54

2.8.3 Životni ciklus appleta 55

2.8.3.1 Metoda init() 55

2.8.3.2 Metoda start() 55

2.8.3.3 Metoda stop() 56

2.8.3.4 Metoda destroy() 56
2.8.4 Animacijska dretva(nit) 56
2.8.5 Eliminacija bljeskanja kod animacija 58
2.8.6 Trokutasti valni paketi u Javi 59
2.9 Upute za korištenje appleta 60
3. Implementacija appleta u nastavi fizike 64
Korištene aplikacije 69
Zaključak 70
Literatura 71
Prilog 72

Zahvaljujem svom mentoru Mirku Planiniću na pomoći oko izbora teme, fizikalnog sadržaja i izrade appleta.
Zahvaljujem kolegama Marijanu Marciušu i Marinu Kosoviću na savjetima i pomoći oko izrade appleta.
Hvala roditeljima koji su mi omogućili ovo školovanje i uvijek bili potpora u životu.

Uvod

Fizika je oduvijek bila moj izbor. Zavolio sam je još u osnovnoj školi. Informatici tada nisam pridavao toliko pažnje. Još sam se više udaljio od informatike i posvetio fizici kada sam nakon osnovne škole upisao opću gimanziju. Upisavši ovaj smjer shvatio sam koje su sve prednosti fizike i informatike kada se udruže. Zaista fenomenalan spoj. Na prvoj godini sam se dosta namučio sa programskim jezicima C-om i C++-om i nikad nisam mislio da će takvo nešto biti tema za moj diplomski rad. Sve dok nisam došao na 4 godinu studija gdje smo u sklopu kolegija Korisnička sučelja izradili aplikaciju koja je simulirala rezonanciju u RLC titrajnom krugu. Tada sam shvatio koje su prednosti objektno orijentiranog programiranja, fascinirao me taj jezik svojom jednostavnošću. Nisam mogao vjerovati kako se u vrlo kratkom vremenu može izraditi kvalitetna aplikacija. A sad se postavlja pitanje: Zašto ipak Java?

Tokom školovanja u sklopu informatičkih kolegija neprestano sam se susretao na Internetu sa aplikacijama u Javi koje simuliraju neki fizikalni problem. Primijetio sam da se neki fizikalni problemi mogu puno bolje razjasniti interaktivnom animacijom nego eksperimentalno. Isprva mi je bila želja da napišem aplikaciju u Pythonu koja će simulirati neki fizikalni problem, no već prilikom izrade aplikacije za RLC-krug naišli smo na problem koji tada nismo znali riješiti, definiranje animacijske petlje u Pythonu. U Javi se vrlo jednostavno može definirati animacijska petlja , pa sam zato odlučio za Javu. Nisam radio sa grafikom u Pythonu, pa ne mogu prosuditi, ali grafika u Javi mi se učinila dosta jednostavnom.

Slijedi odabir teme…Mentor mi je ponudio temu o superponiranju valnih paketa. Prihvatio sam izazov iako tada nisam imao nikakvog znanja o Javi, oslonio sam se na znanja iz C,C++ i Pythona nadajući se sličnosti u sintaksi.

O temi…Odabrali smo baš tu temu zato što kod studenata postoji miskoncepcija o superponiranju valnih paketa. Naime kod njih je postojalo shvaćanje da će žica ostati mirovati u trenutku kada se valni paketi susretnu tj. da će superponiranjem nastati destruktivna interferencija. Ta tvrdnja je točna ako promatramo superpoziciju dvaju valnih paketa opisanih Gauss funkcijom.

Ako pak imamo dva trokutasta valna paketa, koji su zrcalno simetrični oko y-osi i koji su još k tome invertirani, postavlja se pitanje što će se dogoditi prilikom njihova susreta? Da li će žica ostat mirovati u tom trenutku ili neće?
Kod mnogo studenata odgovor je bio da će prilikom susreta žica ostat mirovati, tj. da će se superponiranjem valni paketi poništiti, a da i sam priznam krivicu ja sam isto tako mislio. Nadam se da će ovaj diplomski rad razjasniti sve nedoumice oko ove tematike.

Diplomski rad se sastoji od 3 glavna poglavlja. U prvom poglavlju opisane su osnove fizike valova, od valne funkcije, valne jednadžbe, pa do valnih paketa. Drugo poglavlje opisuje grafičke komponente potrebne za izradu appleta, zatim su objašnjenje metode koje se koriste prilikom crtanja geometrijskih oblika…U posljednjem poglavlju prikazana je implementacija appleta u nastavi fizike.

1. Valovi

1.1 Uvod

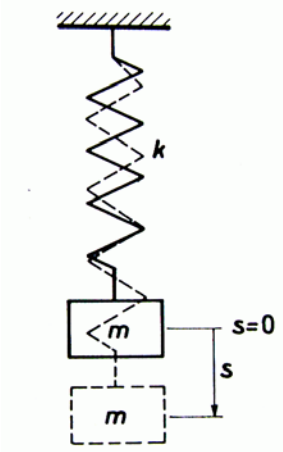
U svakodnevnom životu susrećemo se sa različitim primjerima valova. Npr. bacimo li kamen u vodu, na površini vode će nastati poremećaj(u obliku koncentričnih krugova) koji se širi od mjesta gdje je kamen pao(Slika 1.1). Kada govorimo, kroz okolni prostor se šire zvučni valovi. Zvučni valovi nastaju i kada titra membrana zvučnika ili kada zatitramo žicu nekog muzičkog instrumenta. Radio valove emitiraju antene, u njima pak elektroni prisilno titraju. Sve su ovo primjeri valnog gibanja, općenito možemo reći da valovi koje susrećemo u prirodi nastaju tako da njihov izvor titra. Val je na neki način poremećaj sredstva koji se određenom brzinom širi kroz prostor. Rekli smo da val nastaje kada izvor titra ili oscilira, pogledajmo sad u sljedećem poglavlju što to znači kada nešto harmonijski titra.



Slika 1.1 Valovi na vodi

1.2 Harmonijski oscilator

Da bismo shvatili što znači kada tijelo harmonijski titra proučavat ćemo gibanje tijela mase *m* učvršćenog na kraj horizontalne opruge(Slika 1.2).



Slika 1.2 Titranje tijela na opruzi

Kada otklonimo sistem(tijelo + opruga) iz položaja ravnoteže(s = 0) i pustimo, sistem slobodno titra pod utjecajem elastične sile opruge $F = -ks$. Gdje je k konstanta elastičnosti opruge koja ovisi o svojstvima opruge(njezinu materijalu i obliku). Što je neka opruga rastegljivija to je manja vrijednost njezine konstante. Jednadžbu gibanja za harmonijski oscilator dobijemo pomoću drugog Newtonovog zakona. Prema tome jednadžba gibanja tijela mase m dana je sljedećim izrazom:

$$m\frac{d^2s}{dt^2} = -ks \tag{1}$$

odnosno

$$\frac{d^2s}{dt^2} + \frac{k}{m}s = 0 \tag{2}$$

Izraz (2) predstavlja homogenu linearnu diferencijalnu jednadžbu drugog reda. Pretpostavimo rješenje oblika $s(t) = A\sin(\omega_0 t + \varphi)$ (3). Provjerimo da li naše rješenje zadovoljava izraz (2). Dakle uvrstimo izraz (3) u izraz (2):

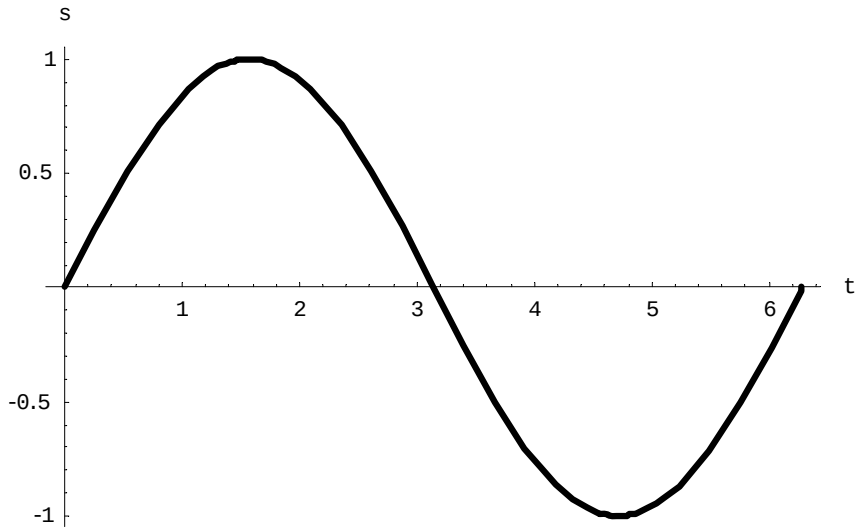
$$-A\omega^2 \sin(\omega_0 t + \varphi) + \frac{k}{m}A\sin(\omega_0 t + \varphi) = 0 \tag{4}$$

dobijemo $\omega_0 = \sqrt{\frac{k}{m}}$ (5) to se naziva **vlastita frekvencija harmonijskog oscilatora**. Dakle rješenje jednadžbe (1) je sinusna harmonijska funkcija $s(t) = A\sin(\omega_0 t + \varphi)$ koja opisuje promjenu elongacije (pomaka) u vremenu.

U funkciji $s(t) = A\sin(\omega_0 t + \varphi)$, A predstavlja amplitudu titranja ili maksimalan otklon iz položaja ravnoteže za dano titranje. Veličina $\omega_0 t + \varphi$ je faza titranja , a φ je početna faza u trenutku t = 0.

Promotrimo dva posebna slučaja:

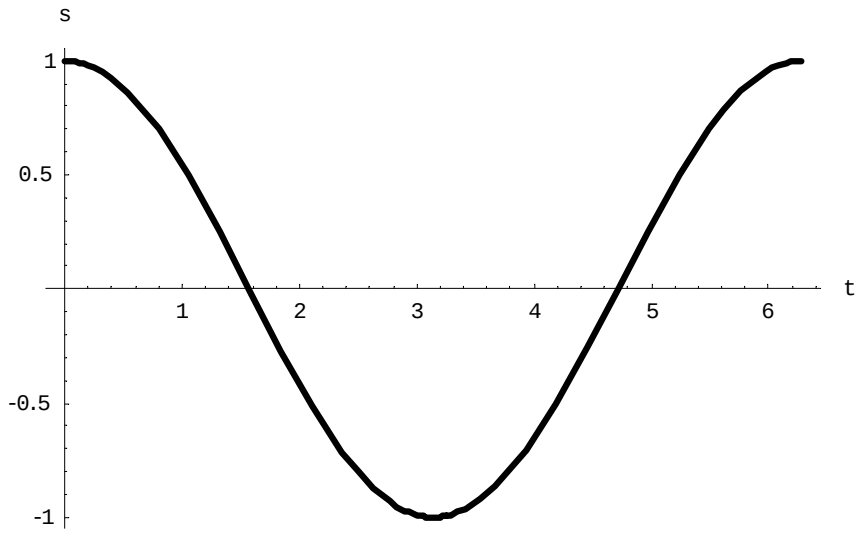
1) $\varphi = 0 \Rightarrow s(t) = A\sin(\omega_0 t)$ *(Slika 1.3)*



Slika 1.3 $s(t) = A\sin(\omega_0 t)$

Titranje s $\varphi = 0$ ostvarujemo tako da u početnom trenutku t=0 imamo oscilator u položaju ravnoteže $s(t=0) = A\sin(0^\circ) = 0$, ali mu naglim udarcem damo početnu brzinu $v(t=0) = \frac{ds(t)}{dt} = \omega_0 A$.

2) $\varphi = \frac{\pi}{2} \Rightarrow s(t) = A\sin(\omega_0 t + \frac{\pi}{2}) = A\cos(\omega_0 t)$ *(Slika 1.4)*



Slika 1.4 $s(t) = A\sin(\omega_0 t + \frac{\pi}{2}) = A\cos(\omega_0 t)$

Titranje s $\varphi = \frac{\pi}{2}$ ostvarujemo tako da u t = 0 pomaknemo oscilator u položaj ravnoteže za iznos A tj. $s(t=0) = A\cos(0^\circ) = A$ i pustimo ga(bez udarca tj. $v(t=0) = \frac{ds(t)}{dt} = -\omega_0 A\sin(\omega_0 t)\big|_{t=0} = 0$).

1.3 Valovi i oscilacije

Kao što smo vidjeli u prethodnom poglavlju, kod oscilacija imamo neku veličinu koja se periodički mijenja. Tako npr. kod harmonijskog oscilatora periodički se mijenja pomak *s,* a kod matematičkog njihala taj pomak se mjeri kutom. Veličine se mijenjaju u vremenu oko nekog stalnog ravnotežnog položaja. Pa za pomak *s* vrijedi kao što smo pokazali u prethodnom poglavlju $s(t) = A\sin(\omega_0 t + \varphi)$.

Promatrajući val, naše oko opaža da se val širi, putuje kroz sredstvo. Razjasnimo malo ovu pojavu. Širenje valova predstavlja širenje poremećaja kroz sredstvo, a ne putovanje samog sredstva. Dakle, *pri širenju vala čestice sredstva ostaju na svojim mjestima i titraju oko svog ravnotežnog položaja*, *a širi se samo stanje titranja*. Ono što oko opaža je zapravo pomicanje forme vala kroz sredstvo koje nastaje zbog vremenskog kašnjenja u titranjima uzastopnih čestica.

Napomenimo još da kod funkcije koja opisuje oscilacije imamo samo jednu nezavisnu varijablu - vrijeme, a kod funkcije koja opisuje valno gibanje imamo dvije nezavisne varijable - vrijeme i koordinatu. Znači valna funkcija nam daje položaj neke čestice u sredstvu u nekom trenutku *t*. O valnoj funkciji ćemo malo kasnije.

1.4 Vrste valova

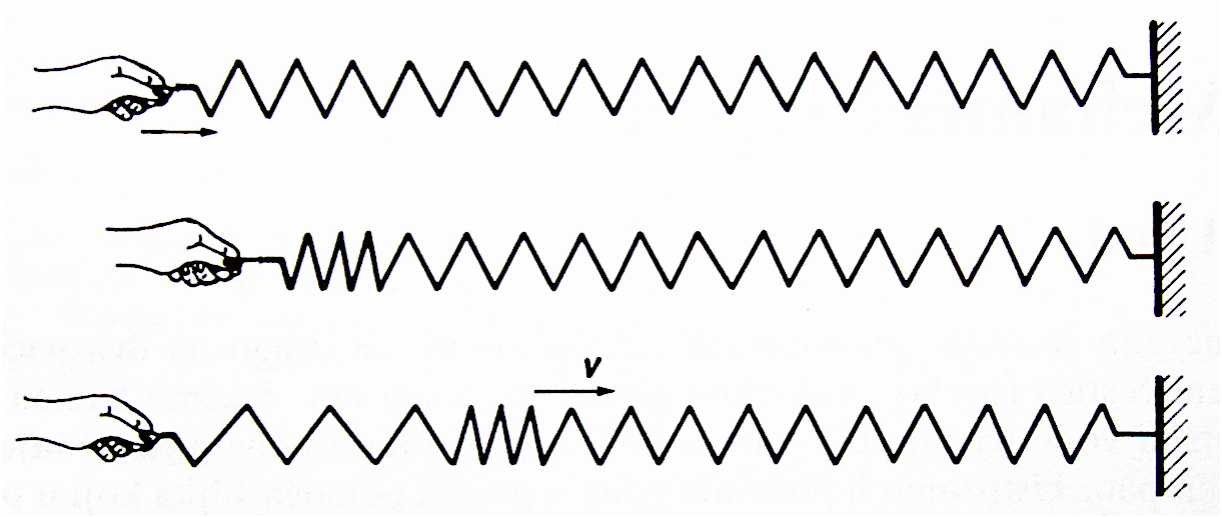
Valove dijelimo na elektromagnetske i mehaničke valove. Elektromagnetski valovi nastaju zbog titranja električnog i magnetskog polja u anteni, dok su mehanički valovi oni koji se gibaju kroz elastična sredstva[1]. Npr. valovi na vodi, zvučni valovi, valovi na žici…

S obzirom kako čestice sredstva titraju dijelimo ih na longitudinalne i transverzalne valove. Transverzalni valovi(Slika 1.5) su oni pri kojima čestice sredstva titraju okomito na smjer širenja vala. To su npr. valovi na užetu, žici…



Slika 1.5 Transverzalni valovi

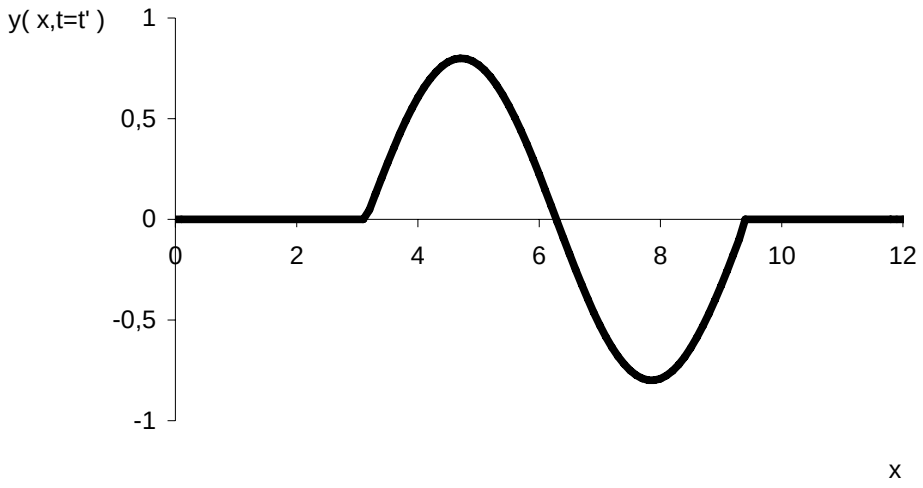
Longitudinalni valovi(Slika 1.6) su oni pri kojima čestice sredstva titraju u smjeru širenja vala. Npr. zvučni valovi, kod njih se lokalno zgušnjenje i razrjeđenje prenosi u smjeru širenja vala. Na Slici 1.6 možemo vidjeti kako eksperimentalno pokazati longitudinalne valove. Na početku par zavoja opruge stisnemo zajedno(lokalno zgušnjenje) kada pustimo vidjet ćemo kako se taj poremećaj(zgušnjenje) širi duž opruge.



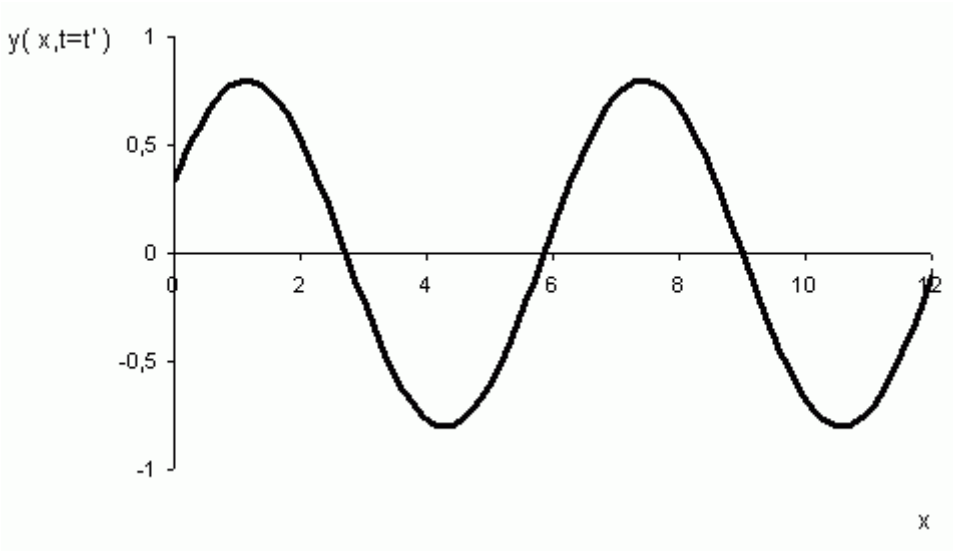
1.6 Longitudinalni valovi

[1] U elastičnim sredstvima čestice su međusobno povezane elastičnim silama te pomak jedne čestice uzrokuje pomak susjednih čestica.

Valovi se još dijele na progresivne ili putujuće i stojne valove. Kod stojnog vala neke čestice titraju, a neke miruju. Kod progresivnog vala slika vala se mijenja u vremenu. Npr. imamo uže, ako početak užeta pomaknemo gore-dolje kroz sredstvo(uže) će se širiti transverzalni valni poremećaj koji nazivamo valni puls ili valni paket(Slika 1.7). Ako nastavimo pomicati uže gore – dolje, kroz sredstvo se širi kontinuirani transverzalni val(Slika 1.8). Kasnije ćemo nešto detaljnije reći o valnim pulsevima odnosno valnim paketima.



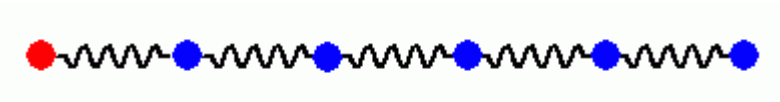
Slika 1.7 Harmonički valni puls



Slika 1.8 Kontinuirani transverzalni val

1.5 Valna funkcija

Uzimo jedno elastično sredstvo kao što je uže. Uže je jednodimenzionalno elastično sredstvo, koje se sastoji od čestica koje su međusobno povezane elastičnim silama. Uže možemo zamisliti kao horizontalni sustav jednakih čestica međusobno povezanih oprugama koji se proteže u beskonačnost(Slika 1.9). Svaka opruga ima svoju konstantu opruge *k*, koja određuje elastičnu silu između čestica.



Slika 1.9 Jednodimenzionalno elastično sredstvo

Na gornjoj slici čestica crvene boje predstavlja početak vala. Ta čestica je spojena na izvor koji harmonijski titra. Ako postavimo ishodište u sam početak vala, čestica crvene boje ima koordinate (0,0). Prema tome ako izvor harmonijski titra, transverzalno gibanje tj. titranje nulte čestice(čestica crvene boje) opisano je relacijom:

$$y(x=0,t) = A\sin(\omega t) \tag{5}$$

Dakle, izvor vala se ponaša kao harmonijski oscilator(Poglavlje 1.2). Kao što smo već spomenuli u poglavlju 1.3, opažač vidi da se valni poremećaj giba, rekli smo da se to javlja zbog vremenskog kašnjenja u titranjima uzastopnih čestica. Objasnimo to sada malo detaljnije. Naime ako čestica crvene boje počne transverzalno titrati, to uzrokuje pomak susjedne plave čestice, a to pak uzrokuje pomak njoj susjedne čestice i tako redom. Titranje svake čestice kasni u odnosu na česticu u izvoru vala, dakle što je čestica dalje to će ona kasnije zatitrati. Zbog toga oko opaža poremećaj koji se giba kroz valno sredstvo. Dakle možemo zaključiti da postoji neko fazno kašnjenje između čestica.

Nulta čestica(crvena čestica) izvodi gibanje od A do –A i natrag. Njoj susjedna prva čestica izvodi isto gibanje ali s nekim vremenskim kašnjenjem Δt. Prema tome izraz koji opisuje gibanje prve čestice glasi:

$$y_1(t) = A\sin\omega(t-\Delta t) \tag{6}$$

Znamo da je vremensko kašnjenje ekvivalentno faznom kašnjenju $\Delta\varphi = \omega\Delta t$. Uvrstimo li taj izraz u (6) dobijemo sljedeće:

$$y_1(t) = A\sin(\omega t - \Delta\varphi) \tag{7}$$

Kao što smo već rekli sve čestice titraju jednakom amplitudom i frekvencijom ali s vremenskim kašnjenjem uzastopnih čestica. Prema tome gibanje druge čestice dano je izrazom:

$$y_2(t) = A\sin(\omega t - 2\Delta\varphi) \tag{8}$$

a i-te čestice:

$$y_i(t) = A\sin(\omega t - i\Delta\varphi) \tag{9}$$

Fazno kašnjenje i-te čestice u odnosu na izvor je $i\Delta\varphi$. Neka je *a* horizontalni razmak između čestica. Prema tome položaj i-te čestice $x_i = ia$. Dakle njeno fazno kašnjenje u odnosu na izvor vala je:

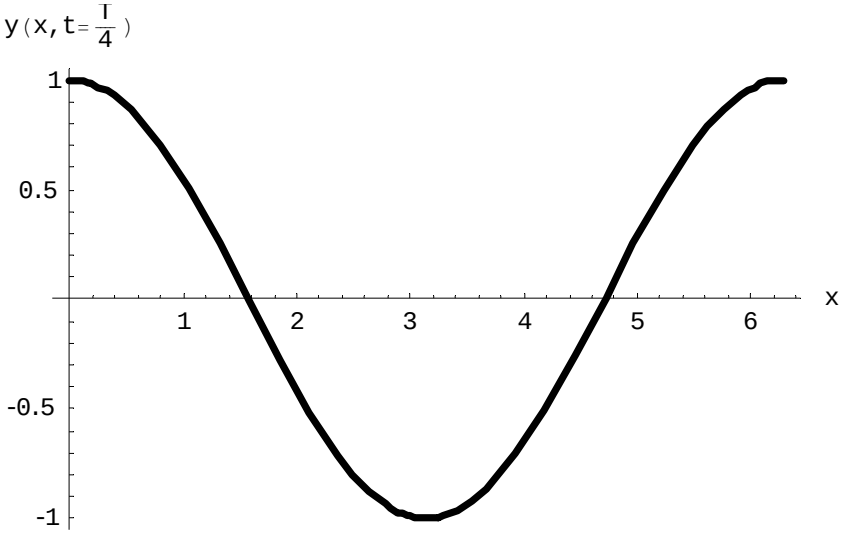
$$i\Delta\varphi = \frac{x_i}{a}\Delta\varphi = kx_i \tag{10}$$

Prema tome možemo reći da je val prikazan funkcijom:

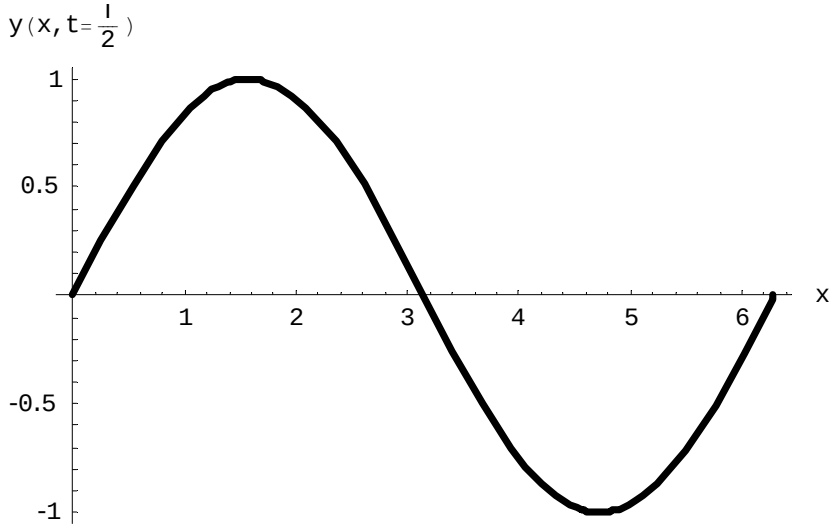
$$y(x,t) = A\sin(\omega t - kx) \tag{11}$$

$$k = \frac{2\pi}{\lambda} \quad \textbf{je kutni valni broj} \tag{12}$$

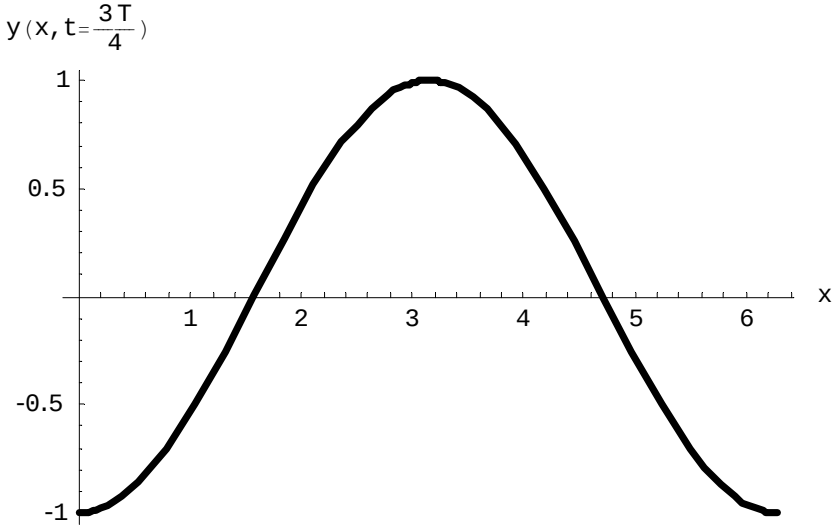
Na donjim slikama je prikazan sinusni val u različitim vremenima t.



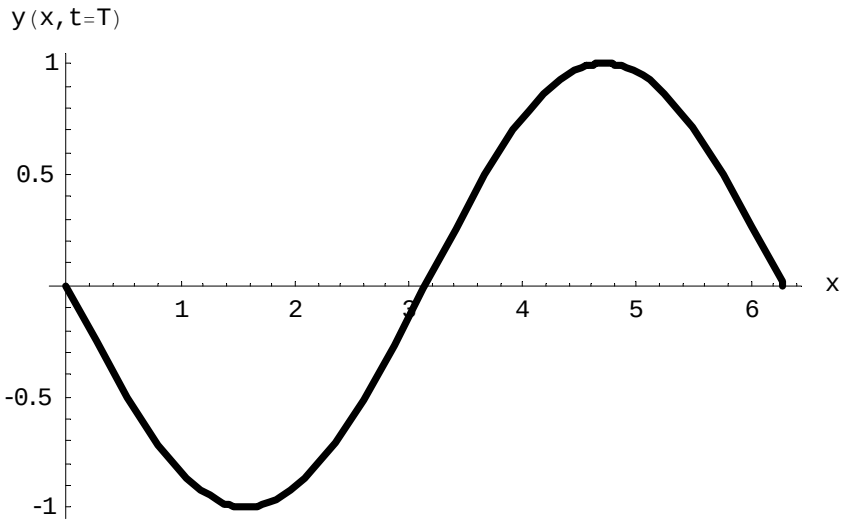
Slika 2.0 Prikaz sinusnog vala u trenutku t= T/4



Slika 2.1 Prikaz sinusnog vala u trenutku t= T/2



Slika 2.2 Prikaz sinusnog vala u trenutku t= 3T/4

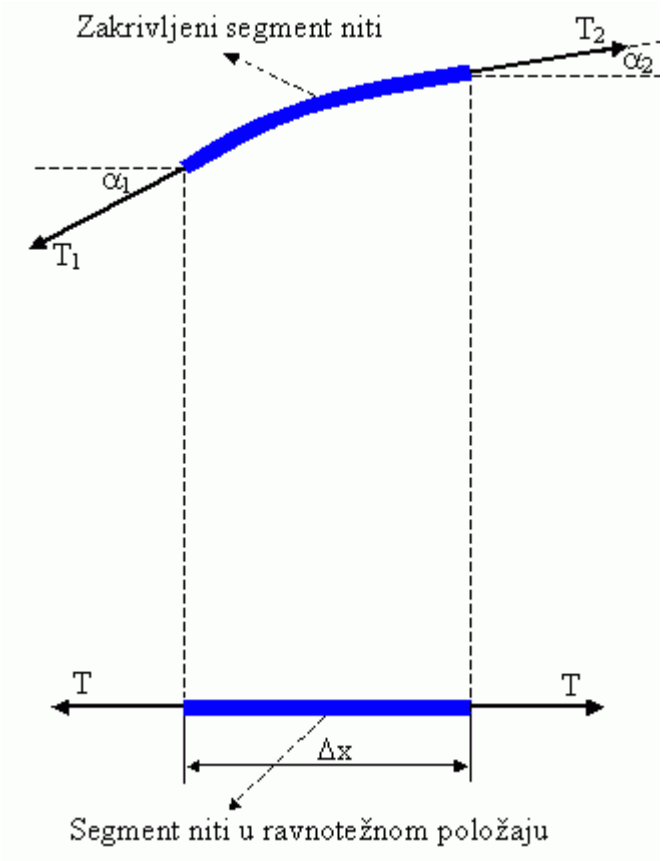


Slika 2.3 Prikaz sinusnog vala u trenutku t= T

Za t = T(Slika 2.3) tj. nakon jednog perioda kroz sredstvo se proširi jedan cijeli val. Udaljenost koju val prevali tokom jednog perioda naziva se valna duljina. Valna duljina je najmanja udaljenost između dviju čestica koje titraju u fazi.

1.6 Valna jednadžba

U ovome poglavlju ćemo izvesti valnu jednadžbu za transverzalni val na žici. Žica ima duljinu *l* i masu *m*. Promatrajmo mali segment te žice Δx, masa segmenta je Δm te je u ravnotežnom položaju napet silom *T* (Slika 2.4-donji dio slike).U jednom trenutku na segment djelujemo silom *F*. Segment više nije ravan kao u ravnotežnom položaju, već poprimi zakrivljen oblik(Slika 2.4-gornji dio slike).



Slika 2.4 Segment niti

Pronađimo koliko iznosi ta sila F. Lijevi kraj segmenta je povučen prema dolje silom $T_1 \sin\alpha_1$, a desni kraj prema gore silom $T_2 \sin\alpha_2$. Prema tome sila koja djeluje na zakrivljeni segment je:

$$F = T_2 \sin\alpha_2 - T_1 \sin\alpha_1 \tag{13}$$

Horizontalne komponente napetosti moraju biti izjednačene(segment se ne pomiče lijevo-desno)

$$T_2 \cos\alpha_2 = T_1 \cos\alpha_1 = T \tag{14}$$

Jednadžba gibanja glasi:

$$m\frac{d^2y}{dt^2} = T_2 \sin\alpha_2 - T_1 \sin\alpha \tag{15}$$

$$\begin{aligned} T_1 \sin\alpha_1 &= \frac{T}{\cos\alpha_1}\sin\alpha_1 = Ttg\alpha_1 = T(\frac{dy}{dx})_1 \\ T_2 \sin\alpha_2 &= \frac{T}{\cos\alpha_2}\sin\alpha_2 = Ttg\alpha_2 = T(\frac{dy}{dx})_2 \end{aligned} \tag{16}$$

Masa segmenta $m=\rho\Delta x$. Izraz $tg\alpha_1$ je koeficijent smjera tangente u točki x_1, a $tg\alpha_2$ je koeficijent smjera tangente u točki x_2. Kada uvrstimo izraze pod (16) u izraz (15) te uzmemo u obzir izraz za masu segmenta dobijemo:

$$\rho\Delta x\frac{d^2y}{dt^2} = T\Delta x\underbrace{\frac{\left(\frac{dy}{dx}\right)_2 - \left(\frac{dy}{dx}\right)_1}{\Delta x}}_{\Delta x \to 0 \;\Rightarrow\; \frac{d^2y}{d^2x}} \tag{17}$$

Prema tome **valna jednadžba** čije rješenje opisuje gibanje transverzalnog vala po žici koja je postavljena duž osi x, čija masa iznosi *m*, a napeta je silom *T* glasi:

$$\frac{\partial^2 y}{\partial t^2} = \frac{T}{\rho}\frac{\partial^2 y}{\partial x^2} \tag{18}$$

Sada ćemo pokazati da je svaka derivabilna funkcija od (x - vt) **rješenje valne jednadžbe**.

$$y(x,t) = f(x - vt) \tag{19}$$

$$\frac{\partial y}{\partial t} = \frac{\partial y}{\partial u} \cdot \frac{\partial u}{\partial t} = \frac{\partial y}{\partial u} \cdot (-v)$$
$$\frac{\partial^2 y}{\partial t^2} = \frac{\partial^2 y}{\partial u^2} \cdot v^2 \tag{20}$$

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u} \cdot \frac{\partial u}{\partial x} = \frac{\partial y}{\partial u}$$
$$\frac{\partial^2 y}{\partial x^2} = \frac{\partial^2 y}{\partial u^2} \tag{21}$$

$$(u = x - vt, \frac{\partial u}{\partial t} = -v, \frac{\partial u}{\partial x} = 1)$$

Kada uvrstimo izraze (21) i (20) u valnu jednadžbu tj. izraz (18) dobijemo da je jednakost ispunjena kada je:

$$v = \sqrt{\frac{T}{\rho}} \tag{22}$$

Izraz (22) se naziva fazna brzina vala tj. brzina širenja transverzalnog vala na žici čija napetost iznosi T ,a linearna gustoća žice iznosi ρ(masa po jedinici duljine). Fazna brzina vala predstavlja brzinu pomicanja forme vala(poremećaja) kroz neko sredstvo. Ovime smo pokazali da je svaka funkcija oblika f(x – vt) rješenje valne jednadžbe. Napomenimo da je val prikazan sinus ili kosinus funkcijom samo jedan od primjera rješenja valne jednadžbe. U poglavlju 1.3 imali smo val koji je opisan sinus funkcijom. Izvor vala harmonijski titra tj. početak žice titra sa elongacijom $y(x = 0, t) = A\sin(\omega t)$. Ostale čestice vremenski kasne u titranju za nultom česticom, pa su njihove elongacije na udaljenosti x od početka žice dane relacijom $y(x,t) = A\sin(\omega t - kx)$. Taj izraz se može napisati i na drukčiji način:

$$y(x,t) = A\sin(\omega t - kx) = A\sin(\frac{2\pi}{T}t - \frac{2\pi}{\lambda}x) = A\sin(\frac{2\pi}{T} \cdot \frac{\lambda}{2\pi}t - x) = A\sin(\frac{\lambda}{T}t - x) \tag{23}$$
$$y(x,t) = A\sin(vt - x) = A\sin\left[-(x - vt)\right] = -A\sin(x - vt)$$

Val prikazan funkcijom $y(x,t) = A\sin(\omega t - kx)$ se gibao udesno(u smjeru pozitivne x-osi) brzinom v. To možemo primijetiti ako pogledamo slike od (2.0 do 2.3) koje prikazuju sinusni val u raznim vremenima t. Brzina širenja tog vala dana je relacijom $v = \frac{\omega}{k}$ (24).Relacija (24) je također izraz za faznu brzinu vala, taj izraz dobijemo ako valnu funkciju $y(x,t) = A\sin(\omega t - kx)$ uvrstimo u valnu jednadžbu . Općenito vrijedi da je val koji se giba u pozitivnom smjeru osi-x opisan funkcijom **y(x,t)=f(x-vt)**, a val koji se giba u negativnom smjeru osi-x opisan funkcijom **y(x,t)=f(x+vt)**.

1.7 Superpozicija valova

Promatrajmo dva vala, koji se gibaju u pozitivnom smjeru osi-x. Neka je valna funkcija prvog definirana kao y_1(x,t)=f(x-vt), a drugog y_2(x,t)=f(x-vt). Tada superpoziciju valova definiramo kao zbroj tih dviju funkcija tj. **$y= y_1(x,t)+ y_2(x,t)$**. Prema tome rezultantna je elongacija jednaka zbroju elongacija individualnih valova.

Uzmimo sad konkretni slučaj. Promatrajmo dva vala koji su opisani sinus funkcijom(izvor vala harmonijski titra –poglavlje 1.5).Oni se gibaju u pozitivnom smjeru osi-x, oba vala imaju jednake amplitude, frekvencije i fazne brzine.
Valne funkcije valova dane su relacijama:

$$y_1(x,t) = A\sin(\omega t - kx) \tag{24}$$

$$y_2(x,t) = A\sin(\omega t - kx + \varphi) \tag{25}$$

Kao što smo rekli zbrojimo elongacije komponentnih valova:

$$\begin{aligned} y_1(x,t) + y_2(x,t) &= A\sin(\omega t - kx) + A\sin(\omega t - kx + \varphi) \\ &= A\left[\sin(\omega t - kx) + \sin(\omega t - kx + \varphi)\right] \end{aligned} \tag{26}$$

Koristeći trigonometrijsku relaciju:

$$\sin\alpha + \sin\beta = 2\sin\frac{\alpha+\beta}{2}\cos\frac{\alpha-\beta}{2} \tag{27}$$

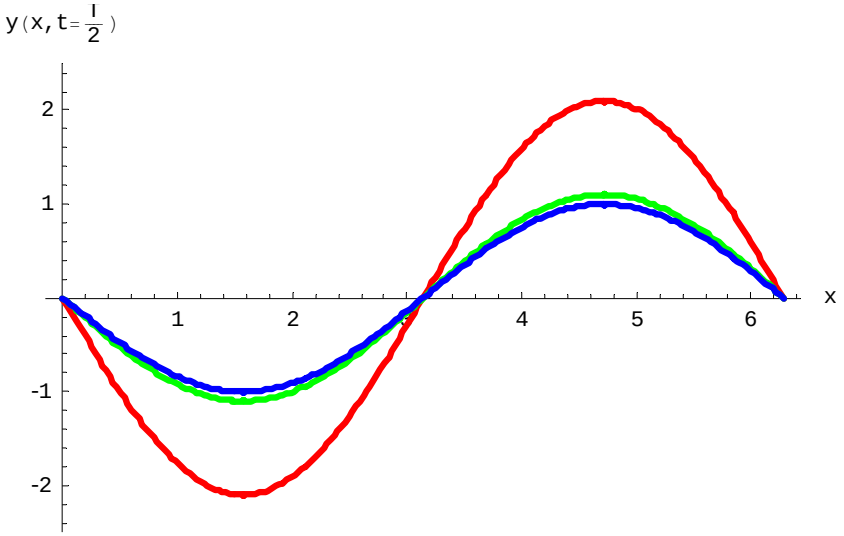
Kada primijenimo izraz (27) u izrazu (26) dobijemo:

$$\begin{aligned} y_1(x,t)+y_2(x,t) &= 2A\sin\frac{(\omega t-kx+\omega t-kx+\varphi)}{2}\cos\frac{(\omega t-kx-\omega t+kx-\varphi)}{2} \\ &= 2A\sin\left(\omega t-kx+\frac{\varphi}{2}\right)\cos\left(-\frac{\varphi}{2}\right) \\ &= 2A\sin\left(\omega t-kx+\frac{\varphi}{2}\right)\cos\left(\frac{\varphi}{2}\right) \end{aligned} \tag{28}$$

Kao rezultat superpozicije valova javlja se interferencija. Konstruktivna interferencija nastaje tamo gdje je amplituda rezultantnog vala maksimalna, a destruktivna interferencija nastaje tamo gdje je amplituda rezultantnog vala nula.

Za naš gornji primjer konstruktivna interferencija(Slika 2.5) nastaje kada je φ=0, tada je rezultantni val dan izrazom:

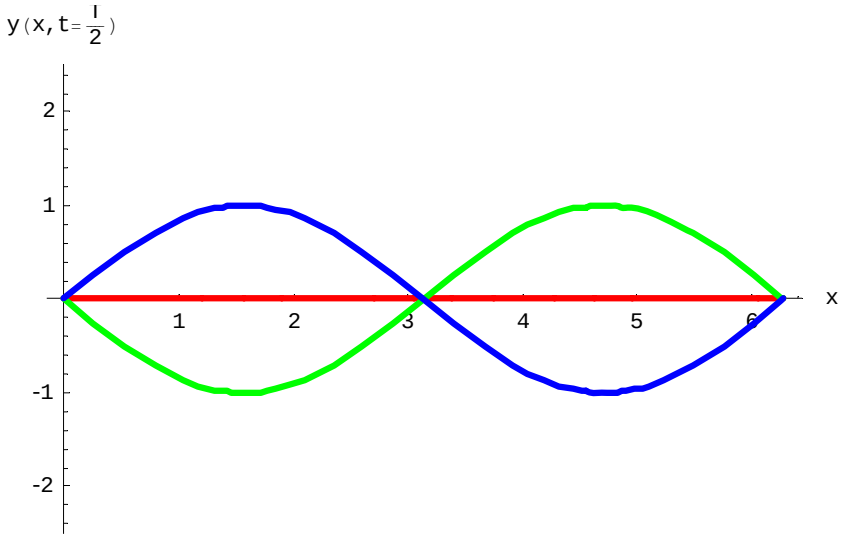
$$\begin{aligned} y_1(x,t)+y_2(x,t) &= 2A\sin(\omega t-kx+0)\cos(0) \\ &= 2A\sin(\omega t-kx) \end{aligned} \tag{30}$$



Slika 2.5 Konstruktivna interferencija

Plavom bojom je predočen val čija valna funkcija glasi $y_1(x,t)=A_1\sin(\omega t-kx)$, zelenom bojom val čija valna funkcija glasi $y_2(x,t)=A_2\sin(\omega t-kx)$. Dok je crvenom bojom prikazan rezultat superpozicije tih dvaju valova.

Destruktivna interferencija nastaje kada je φ=π, tada je amplituda rezultantnog vala nula.(Slika 2.6).



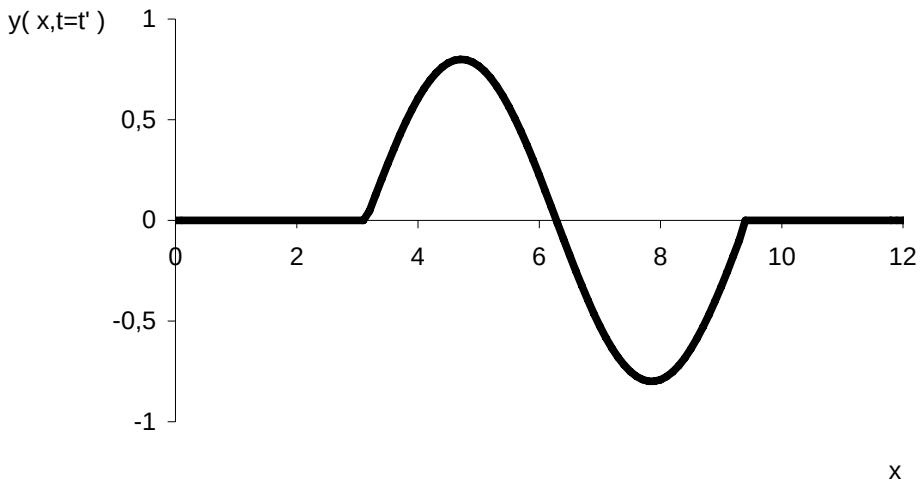
Slika 2.6 Destruktivna interferencija

Plavom bojom je predočen val čija valna funkcija glasi $y_1(x,t) = A\sin(\omega t - kx)$, zelenom bojom val čija valna funkcija glasi $y_2(x,t) = A\sin(\omega t - kx + \varphi)$. Dok je crvenom bojom prikazan rezultat superpozicije tih dvaju valova.

1.8 Valni paketi

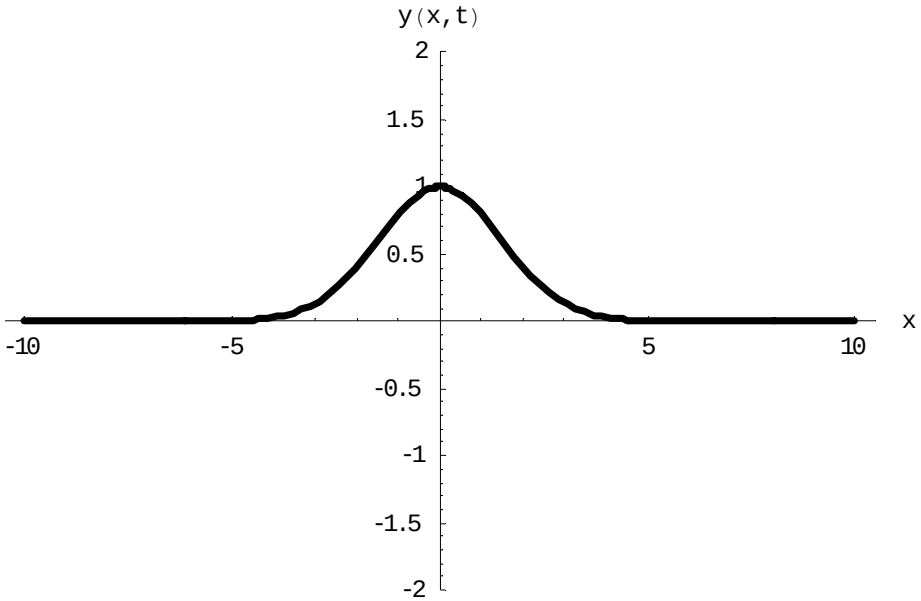
U ovom poglavlju ćemo promatrati širenje i superpoziciju valnih pulseva na 1-D elastičnom sredstvu tj. užetu. Valne pulseve možemo zvat i valni paketi. Pogledajmo kako eksperimentalno dobiti valne pulseve.

Ako početak užeta pomaknemo jednom gore-dolje nastane transverzalni valni poremećaj. Taj transverzalni valni poremećaj se naziva valni puls i on se širi kroz sredstvo(uže). Takav oblik valnog pulsa se naziva harmonički valni puls(Slika 2.7).



Slika 2.7 Harmonički valni puls

Postoji još i neharmonički valni puls. Jedan takav neharmonički valni puls je prikazan na slici(Slika 2.8). Za valne pulseve odnosno valne pakete je karakteristično to da je valno gibanje lokalizirano tj. ograničeno na određeni dio prostora Δx, a izvan tog prostora nema valnog gibanja tj. čestice elastičnog sredstva miruju. Suprotno valnom paketu koji je ograničen u prostoru, kontinuirani transverzalni val(izvor vala harmonijski titra) nije ograničen u prostoru.



Slika 2.8 Neharmonički valni puls

1.8.1 Matematički opis valnih paketa

1.8.1.1 Gaussijanski valni paketi

U poglavlju 1.5 smo dokazali da svaka derivabilna funkcija oblika f(x-vt) je rješenje valne jednadžbe. Tj. svaka funkcija f(x-vt) opisuje valni poremećaj koji se širi u pozitivnom smjeru osi –x. Uzmimo sada Gauss funkciju(Slika 2.8):

$$y(x) = Ae^{-\frac{x^2}{2\sigma^2}} \tag{31}$$

Pretpostavimo da je valna funkcija koja opisuje gaussijanski valni paket koji se kreće u pozitivnom smjeru osi –x dana s relacijom:

$$y(x,t) = Ae^{-\frac{(x-vt)^2}{2\sigma^2}} \tag{32}$$

Provjerimo da li je naša pretpostavka bila točna, izraz (32) mora zadovoljit valnu jednadžbu tj. izraz(18):

$$\frac{\partial^2 y}{\partial t^2} = \frac{T}{\rho}\frac{\partial^2 y}{\partial x^2}$$

$$\frac{\partial y}{\partial t} = e^{-\frac{(x-vt)^2}{2\sigma^2}} \cdot \frac{-1}{2\sigma^2} \cdot 2(x-vt)(-v) = \frac{v}{\sigma^2} \cdot e^{-\frac{(x-vt)^2}{2\sigma^2}} \cdot (x-vt)$$

$$\frac{\partial^2 y}{\partial t^2} = \frac{\partial}{\partial t}\left(\frac{\partial y}{\partial t}\right)$$

$$= \frac{v}{\sigma^2} \cdot \left(\frac{v}{\sigma^2} \cdot e^{-\frac{(x-vt)^2}{2\sigma^2}} \cdot (x-vt) \cdot (x-vt) + e^{-\frac{(x-vt)^2}{2\sigma^2}} \cdot (-v) \right)$$

$$= \frac{v^2}{\sigma^4} \cdot e^{-\frac{(x-vt)^2}{2\sigma^2}} \cdot (x-vt)^2 - \frac{v^2}{\sigma^2} \cdot e^{-\frac{(x-vt)^2}{2\sigma^2}}$$

$$= v^2 \cdot \left(\frac{1}{\sigma^4} \cdot e^{-\frac{(x-vt)^2}{2\sigma^2}} \cdot (x-vt)^2 - \frac{1}{\sigma^2} \cdot e^{-\frac{(x-vt)^2}{2\sigma^2}} \right)$$

$$\frac{\partial y}{\partial x} = e^{-\frac{(x-vt)^2}{2\sigma^2}} \cdot \frac{-1}{2\sigma^2} \cdot 2(x-vt) \cdot (1) = -\frac{1}{\sigma^2} \cdot e^{-\frac{(x-vt)^2}{2\sigma^2}} \cdot (x-vt)$$

$$\frac{\partial^2 y}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial y}{\partial x}\right)$$

$$= -\frac{1}{\sigma^2} \cdot \left(-\frac{1}{\sigma^2} \cdot e^{-\frac{(x-vt)^2}{2\sigma^2}} \cdot (x-vt) \cdot (x-vt) + e^{-\frac{(x-vt)^2}{2\sigma^2}} \cdot 1 \right)$$

$$= 1 \cdot \left(\frac{1}{\sigma^4} \cdot e^{-\frac{(x-vt)^2}{2\sigma^2}} \cdot (x-vt)^2 - \frac{1}{\sigma^2} \cdot e^{-\frac{(x-vt)^2}{2\sigma^2}} \cdot 1 \right)$$

$$v^2 \cdot \left(\frac{1}{\sigma^4} \cdot e^{-\frac{(x-vt)^2}{2\sigma^2}} \cdot (x-vt)^2 - \frac{1}{\sigma^2} \cdot e^{-\frac{(x-vt)^2}{2\sigma^2}} \right) = \frac{T}{\rho} \cdot \left(\frac{1}{\sigma^4} \cdot e^{-\frac{(x-vt)^2}{2\sigma^2}} \cdot (x-vt)^2 - \frac{1}{\sigma^2} \cdot e^{-\frac{(x-vt)^2}{2\sigma^2}} \right)$$

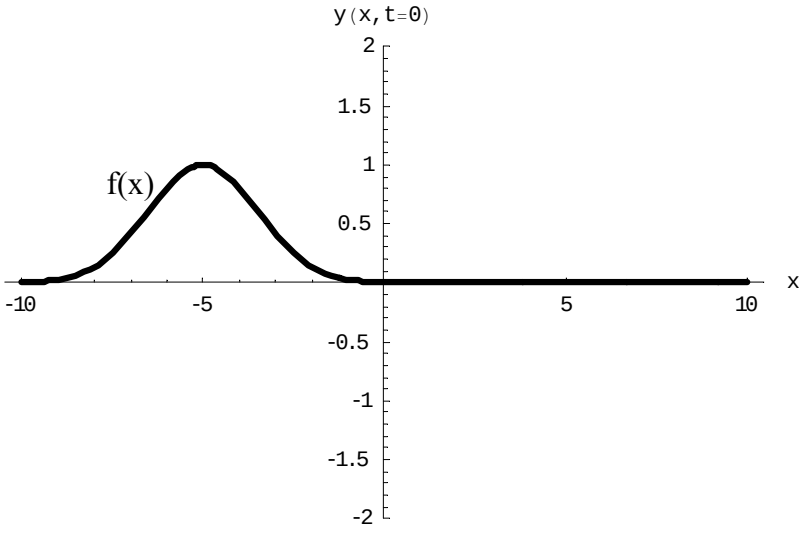
$$v = \sqrt{\frac{T}{\rho}}$$

U gornjem smo izvodu dokazali da je izraz (32) valna funkcija koja opisuje gaussijanski valni paket koji se giba brzinom v u pozitivnom smjeru osi -x. Na kraju izvoda smo dobili isti izraz za faznu brzinu kao i u poglavlju 1.6 znači da je funkcija iz izraza (32) derivabilna funkcija oblika f(x-vt) koja opisuje valno gibanje.

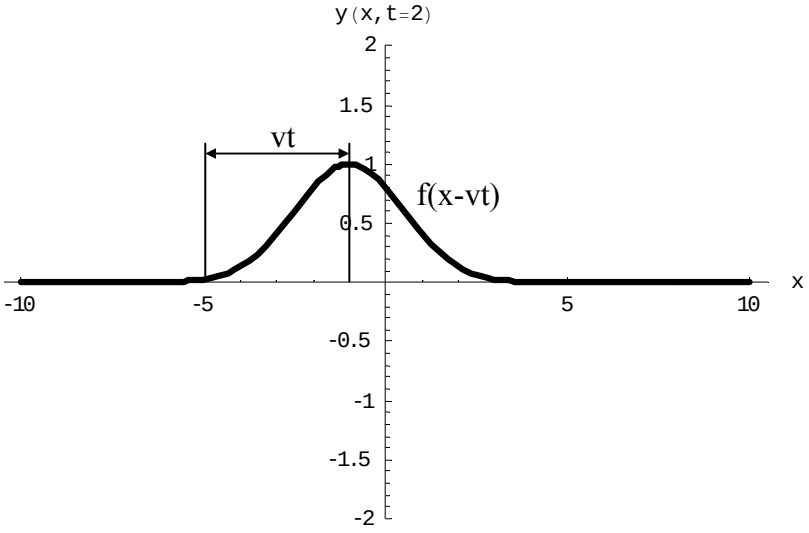
Slika 2.8 pokazuje gaussijanski valni paket koji se nalazi u ishodištu koordinatnog sustava u trenutku t=0. Ukoliko želimo da nam u trenutku t=0 valni paket ne kreće iz ishodišta definirat ćemo novi parametar *B* unutar valne funkcije(izraza 32). Parametar *B* će pomaknuti gaussijanski valni paket iz ishodišta ulijevo ili udesno.

$$y(x,t) = Ae^{-\frac{(x-vt-B)^2}{2\sigma^2}} \tag{33}$$

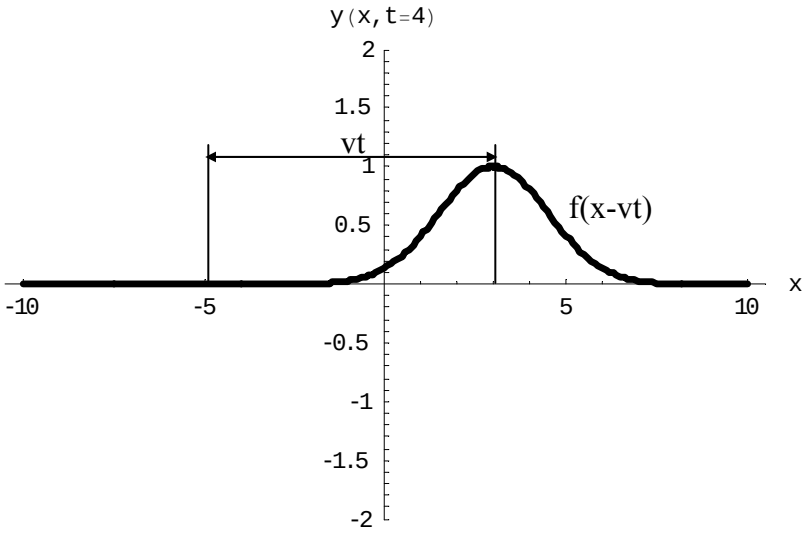
Ostali parametri koji su još važni za gaussijanski valni paket su brzina kretanja valnog pulsa *v*, zatim parametar koji definira širinu valnog paketa σ i naravno parametar *B* koji smo već prije spomenuli. Pogledajmo sada na slikama 2.9 do 3.1 kako izgleda naš valni paket u različitim vremenima t.



Slika 2.9 Valni paket u trenutku t=0



Slika 3.0 Valni paket u trenutku t=2



Slika 3.1 Valni paket u trenutku t=4

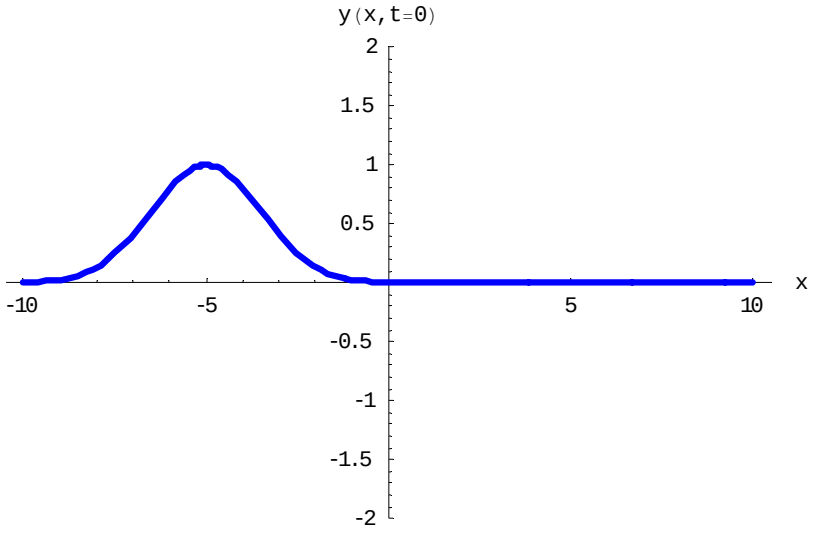
1.8.1.2 Superpozicija Gaussijanskih valnih paketa

Promatrat ćemo superpoziciju dvaju gaussijanskih valnih paketa koji se kreću u različitim smjerovima. Iznosi vektora faznih brzina su im jednaki, no različiti su im smjerovi vektora. Širina paketa će im također biti jednaka dok će im položaji u početnom trenutku biti različiti. Jedan valni paket će se nalazit u prvom kvadrantu, a drugi valni paket u drugom kvadrantu.

Valna funkcija za prvi valni paket koji se u trenutku t=0 nalazi u drugom kvadrantu glasi:

$$y_1(x,t) = Ae^{-\frac{(x-v_1t-B_1)^2}{2\sigma^2}} \tag{34}$$

Da bi se valni paket nalazio u drugom kvadrantu i gibao udesno nekom brzinom v_1, mora vrijediti $B_1<0$ i $v_1>0$(Slika 3.2).

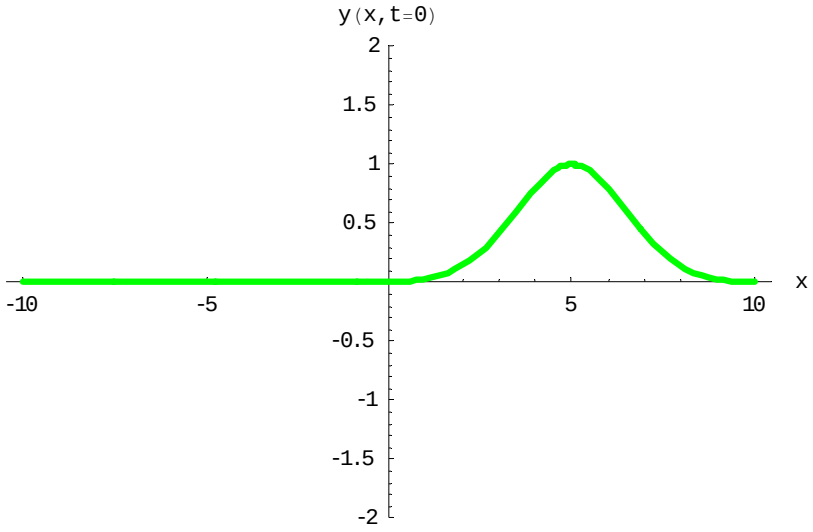


Slika 3.2 Valni paket u drugom kvadrantu

Valna funkcija za drugi valni paket koji se u trenutku t=0 nalazi u prvom kvadrantu glasi:

$$y_2(x,t) = Ae^{-\frac{(x-v_2t-B_2)^2}{2\sigma^2}} \tag{35}$$

Da bi se valni paket nalazio u prvom kvadrantu i gibao ulijevo nekom brzinom v_2, mora vrijediti $B_2>0$ i $v_2<0$(Slika 3.3).



Slika 3.3 Valni paket u prvom kvadrantu

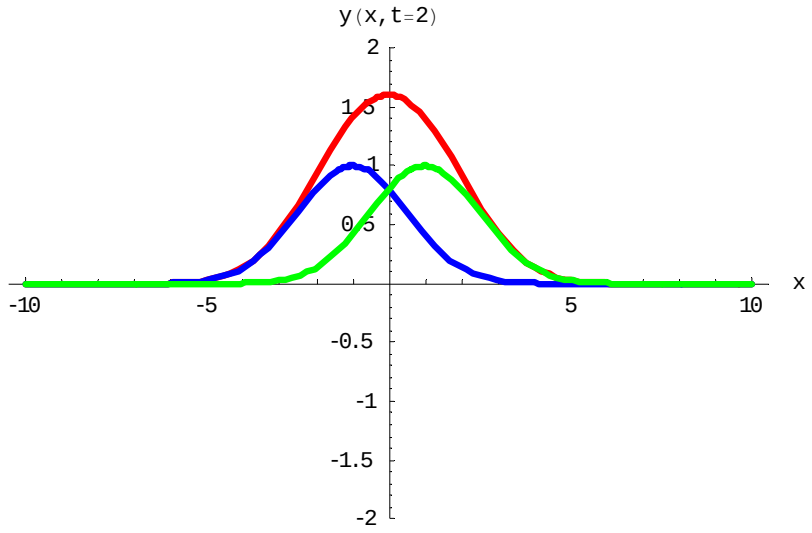
Superpozicija (Slika 3.4-3.6) valnih paketa prikazanih na slikama 3.2 i 3.3 je dana relacijom:

$$y_1(x,t)+y_2(x,t)=Ae^{-\frac{(x-v_1t-B_1)^2}{2\sigma^2}}+Ae^{-\frac{(x-v_2t-B_2)^2}{2\sigma^2}} \tag{36}$$

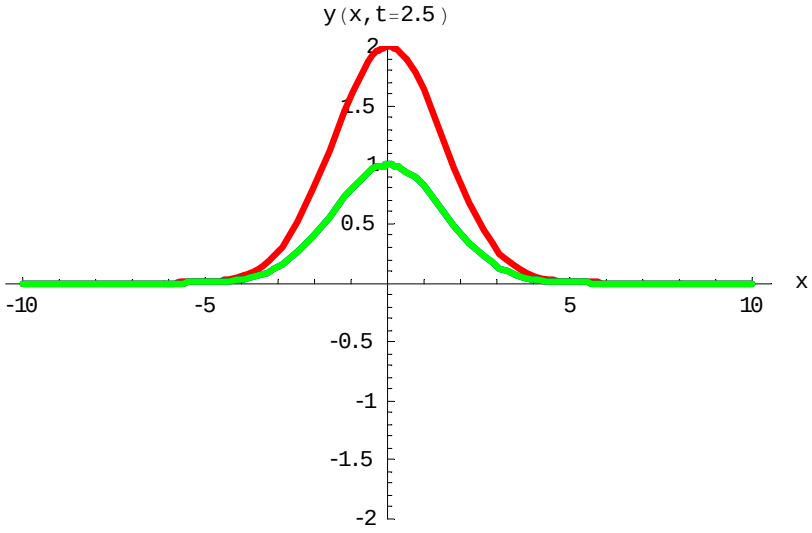
Ako valni paketi imaju jednake brzine po iznosu te su za jednaki iznos pomaknuti u početnom položaju ulijevo odnosno udesno. Tj. $v_1= v_2=v$, a $B_1= B_2=B$, tada izraz (36) izgleda:

$$y_1(x,t)+y_2(x,t)=Ae^{-\frac{(x-vt+B)^2}{2\sigma^2}}+Ae^{-\frac{(x+vt-B)^2}{2\sigma^2}} \tag{37}$$

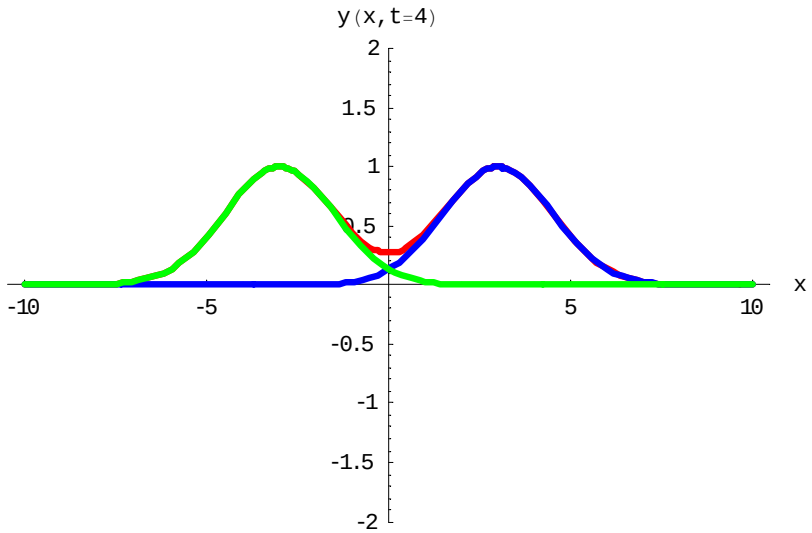
Na donjim slikama je prikazana superpozicija gaussijanskih valnih paketa u raznim vremenima *t*. Krivulja obojena crveno prikazuje superpoziciju u danome trenutku.



Slika 3.4 Superpozicija valnih paketa u t=2



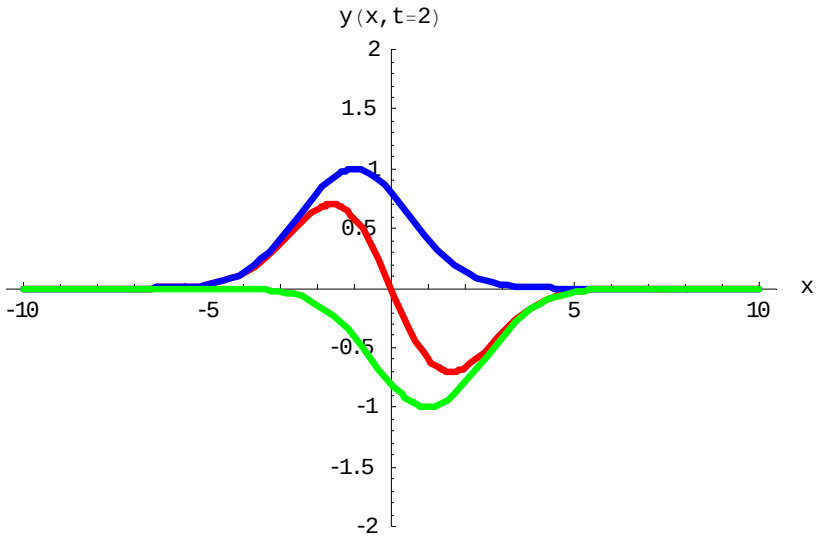
Slika 3.5 Superpozicija valnih paketa u t=2.5



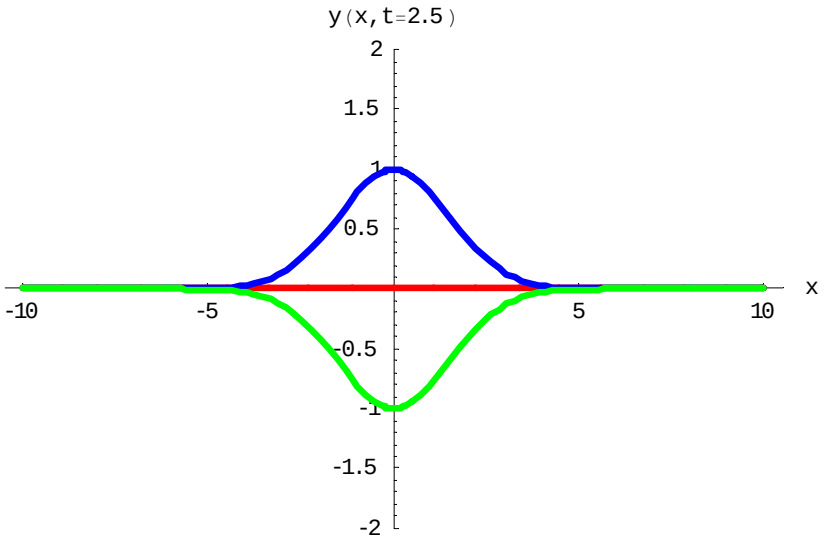
Slika 3.6 Superpozicija valnih paketa u t=4

Promatrajmo sada superpoziciju invertiranih valnih paketa(Slika 3.7-3.9)(izraz 38). Amplituda jednog valnog paketa će biti A, a drugog –A.

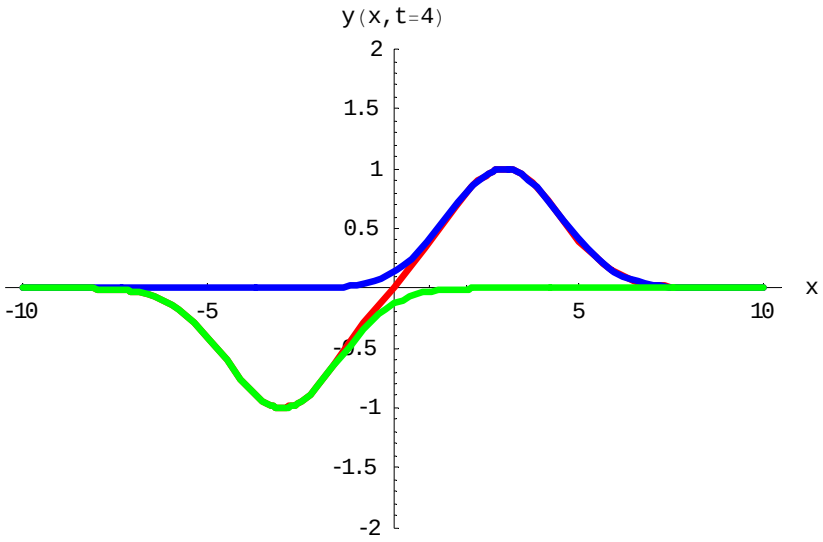
$$y_1(x,t)+y_2(x,t)=Ae^{-\frac{(x-vt+B)^2}{2\sigma^2}}-Ae^{-\frac{(x+vt-B)^2}{2\sigma^2}} \tag{38}$$



Slika 3.7 Superpozicija invertiranih valnih paketa u t=2



Slika 3.8 Superpozicija invertiranih valnih paketa u t=2.5

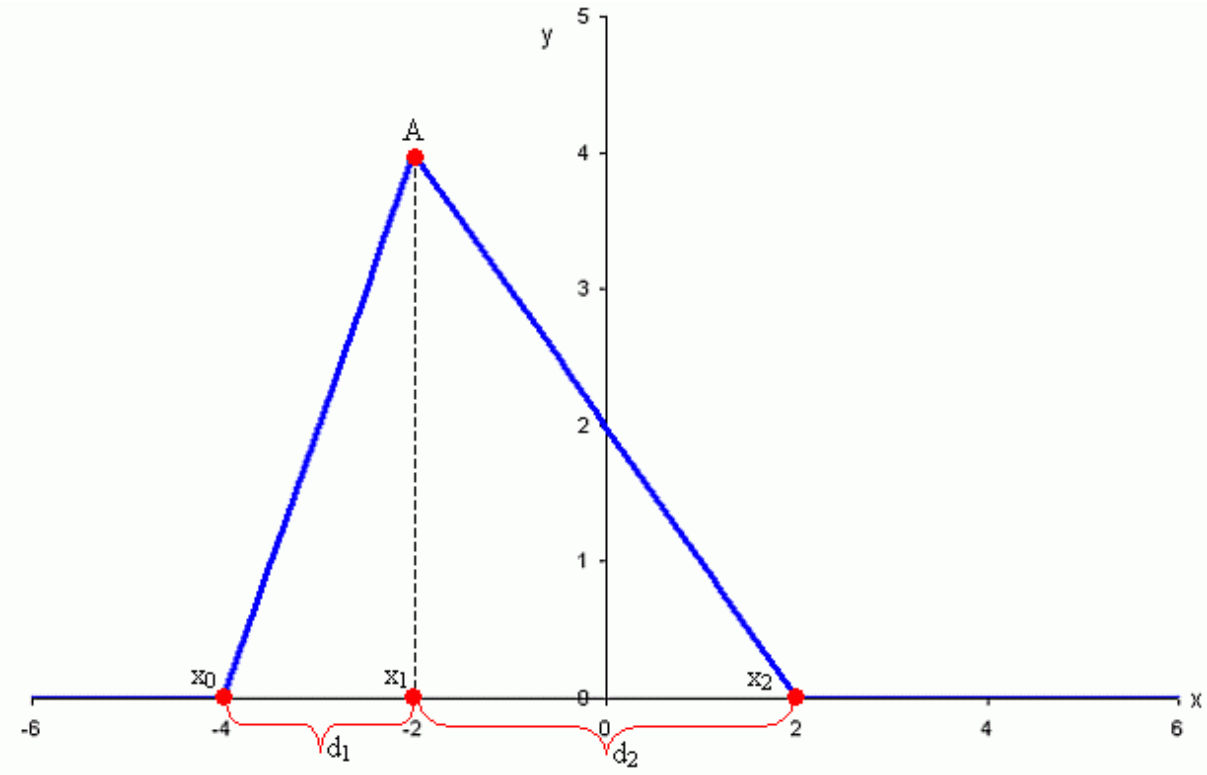


Slika 3.9 Superpozicija invertiranih valnih paketa u t=4

Na slici 3.5 kada su se susrela dva valna paketa koja imaju jednaku amplitudu, u trenutku t=2.5 imamo konstruktivnu interferenciju tj. novonastali val ima dvostruko veću amplitudu od svakog pojedinačnog vala. Dok na slici 3.8 kada se susretnu dva invertirana valna paketa u trenutku t=2.5 imamo destruktivnu interferenciju, tj. elongacija je nula.

1.8.1.3 Trokutasti valni paketi

Prvo ćemo pokazati kako se **matematički definira trokut**(Slika 4.0).



Slika 4.0 Matematički opis trokuta

Uvedimo neke parametre:

a…koeficijent smjera pravca koji je određen točkama $(x_0,0)$ i $A(x_1,y)$.

b… koeficijent smjera pravca koji je određen točkama $(x_2,0)$ i $A(x_1,y)$.

Definirajmo trokut promatrajući gornju sliku:

$$y = \begin{cases} 0, & x < x_0 \\ 0, & x > x_2 \\ ax, & x_0 < x < x_1 \\ bx, & x_1 < x < x_2 \end{cases}$$

Područje unutar trokuta je definirano na sljedeći način(Slika 4.0 i Slika 4.1):

$$x > x_0 \Leftrightarrow \mathbf{x - x_0 > 0} \tag{39}$$

$$x < x_2 \Leftrightarrow x - x_2 < 0 \Leftrightarrow x - (x_0 + d_1 + d_2) < 0 \Leftrightarrow \mathbf{x - x_0 < d_1 + d_2} \tag{40}$$

Područje unutar trokuta ali lijevo od točke x_1 je definirano (Slika 4.0 i Slika 4.1):

$$x < x_1 \Leftrightarrow x < x_0 + d_1 \Leftrightarrow \mathbf{x - x_0 < d_1} \tag{41}$$

Ako smo unutar trokuta i u području lijevo od točke x_1 Slika 4.0 i Slika 4.1):

$$\mathbf{y = (x - x_0)a} \tag{42}$$

Ako smo unutar trokuta i u području desno od točke x_1 (Slika 4.0 i Slika 4.1):

$$\mathbf{y = ad_1 + (x - x_0 - d_1)b} \tag{43}$$

Još nam ostaje da računski odredimo parametar d_2. Promotrimo točku A(Slika 4.0 i Slika 4.1), njene koordinate su (x_1, ad_1) odnosno (x_2, bd_2). Prema tome možemo pisati:

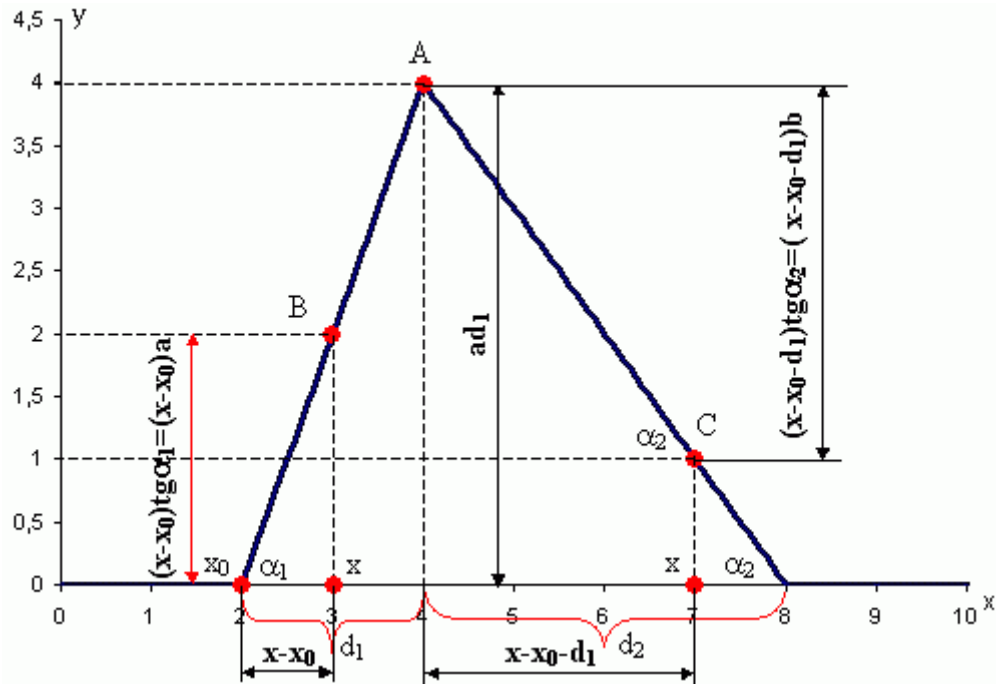
$$\mathbf{ad_1 = bd_2}$$

$$d_2 = \frac{a}{b} \cdot d_1$$

Odnosno ako uzmemo u obzir da koeficijenti smjera pravaca mogu biti negativni, uvedimo negativan predznak u gornji izraz:

$$d_2 = -\frac{a}{b} \cdot d_1 \tag{44}$$

Na slici 4.1 je prikazano kako smo došli do izraza (42) i izraza (43).



Slika 4.1 Matematički opis trokuta

Radi provjere izračunat ćemo koordinate točaka A, B i C te provjeriti dal se koordinate poklapaju s onima na slici. Koeficijenti smjera pravaca iznose redom a = 2 i b = -1, dok parametar d_1=2.

Prvo ćemo izračunat parametar d_2, koristeći izraz (44) dobijemo da on iznosi 4. Sa slike očitamo koordinatu x_0, te ona iznosi 2. Krenimo sada redom provjeriti koordinate točaka.

Točka A

Koordinate očitane sa slike glase (4,**4**).

Računski: $y = (x-x_0)a=(4-2)*2=ad_1=$**4**

Točka B

Koordinate očitane sa slike glase (3,**2**).

Računski: $y = (x-x_0)a=(3-2)*2=$**2**

Točka C

Koordinate očitane sa slike glase (7,**1**).

Računski: $y = ad_1+(x - x_0 - d_1)b=2*2+(7-2-2)*(-1)=4+3*(-1)=4-3=$**1**

Time smo dokazali da je naš matematički opis trokuta ispravan, pa možemo u daljnjem kontekstu **definirati trokutasti valni paket koji se giba u pozitivnom smjeru osi x**.
On je definiran na sljedeći način:

$$y(x,t)=\begin{cases} 0, & (x - vt) < x_0 \\ 0, & (x - vt) > x_2 \\ a(x-vt), & x_0 < (x-vt) < x_1 \\ b(x-vt), & x_1 < (x - vt) < x_2 \end{cases}$$

Područje unutar trokuta je definirano na sljedeći način(Slika 4.0 i Slika 4.1):

$$\mathbf{(x - vt) - x_0 > 0} \tag{45}$$

$$\mathbf{(x - vt) - x_0 < d_1 + d_2} \tag{46}$$

Područje unutar trokuta ali lijevo od točke x_1 je definirano (Slika 4.0 i Slika 4.1):

$$\mathbf{(x - vt) - x_0 < d_1} \tag{47}$$

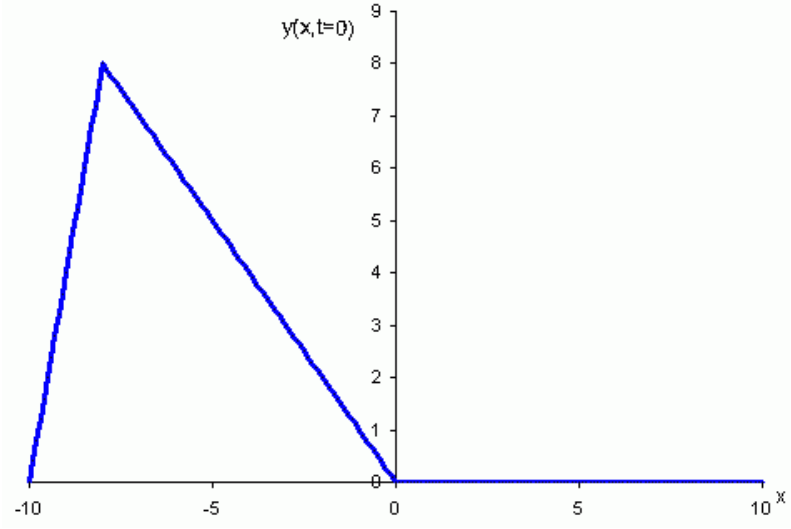
Ako smo unutar trokuta i u području lijevo od točke x_1 (Slika 4.0 i Slika 4.1):

$$\mathbf{y = [(x - vt) - x_0]a} \tag{48}$$

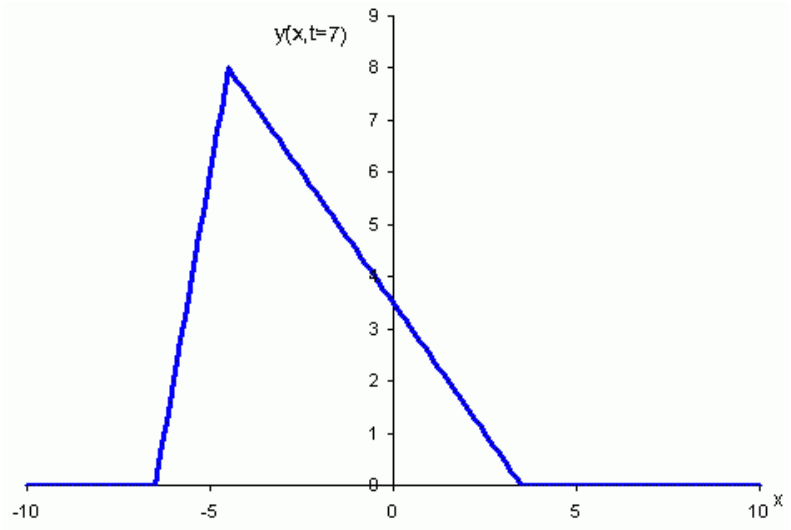
Ako smo unutar trokuta i u području desno od točke x_1 (Slika 4.0 i Slika 4.1):

$$\mathbf{y = ad_1+[(x-vt) - x_0 - d_1]b} \tag{49}$$

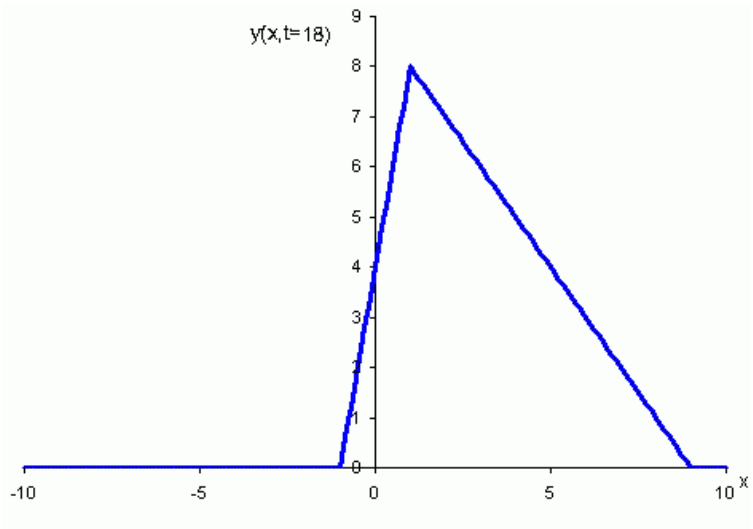
Na slikama 4.2-4.4 je prikazan trokutasti valni paket u raznim trenucima t.



Slika 4.2 Trokutasti valni paket u trenutku t=0



Slika 4.3 Trokutasti valni paket u trenutku t=7



Slika 4.4 Trokutasti valni paket u trenutku t=18

1.8.1.4 Superpozicija trokutastih valnih paketa

Promatrajmo superpoziciju dva trokutasta valna paketa koji su zrcalno simetrični oko y-osi i koji su još k tome invertirani(Slika 4.5-4.7).Na slici 4.6 je prikazano kako dolazimo do superpozicije, dakle zbroj elongacije individualnih valnih paketa(zelena i plava kota) nam daje elongaciju rezultantnog valnog paketa(crvena kota).

Relacije (45)-(49) za **trokutasti valni paket(plavi valni paket –Slika 4.5) koji se giba udesno brzinom v_1** i čiji su parametri a_1, b_1, d_{11} ,d_{12} *i početak u* x_{01} izgledaju:
Područje unutar trokuta je definirano na sljedeći način(Slika 4.0 i Slika 4.1):

$$\mathbf{(x - v_1t) - x_{01} > 0} \tag{50}$$

$$\mathbf{(x - v_1t) - x_{01} < d_{11} + d_{12}} \tag{51}$$

Područje unutar trokuta ali lijevo od točke x_1 je definirano (Slika 4.0 i Slika 4.1):

$$\mathbf{(x - v_1t) - x_{01} < d_{11}} \tag{52}$$

Ako smo unutar trokuta i u području lijevo od točke x_1 (Slika 4.0 i Slika 4.1):

$$\mathbf{y_1 = [(x - v_1t) - x_{01}]\, a_1} \tag{53}$$

Ako smo unutar trokuta i u području desno od točke x_1 (Slika 4.0 i Slika 4.1):

$$\mathbf{y_1 = a_1d_{11} + [(x - v_1t) - x_{01} - d_{11}]\, b_1} \tag{54}$$

Relacije (45)-(49) za **trokutasti valni paket(zeleni valni paket –Slika 4.5) koji se giba ulijevo brzinom v_2** i čiji su parametri a_2, b_2, d_{21}, d_{22} *i početak u* x_{02} izgledaju:
Područje unutar trokuta je definirano na sljedeći način(Slika 4.0 i Slika 4.1):

$$\mathbf{(x - v_2t) - x_{02} > 0} \tag{55}$$

$$\mathbf{(x - v_2t) - x_{02} < d_{21} + d_{22}} \tag{56}$$

Područje unutar trokuta ali lijevo od točke x_1 je definirano (Slika 4.0 i Slika 4.1):

$$\mathbf{(x - v_2t) - x_{02} < d_{21}} \tag{57}$$

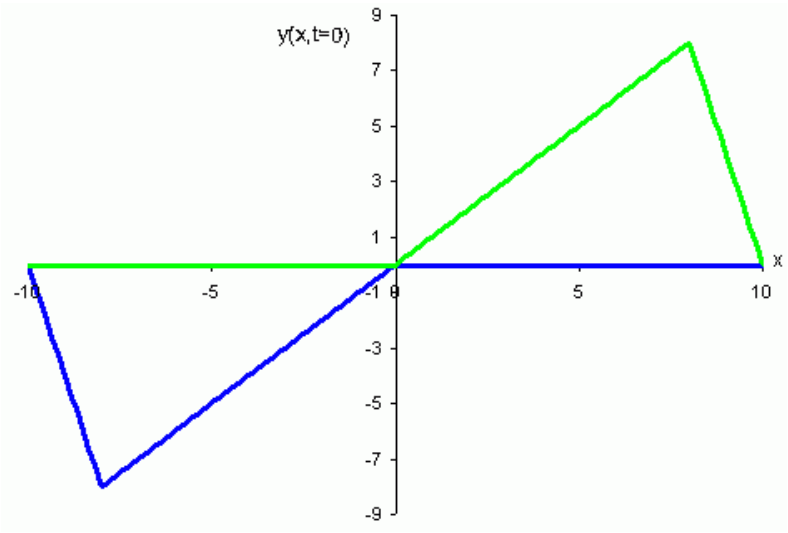
Ako smo unutar trokuta i u području lijevo od točke x_1 (Slika 4.0 i Slika 4.1):

$$\mathbf{y_2 = [(x - v_2t) - x_{02}]\, a_2} \tag{58}$$

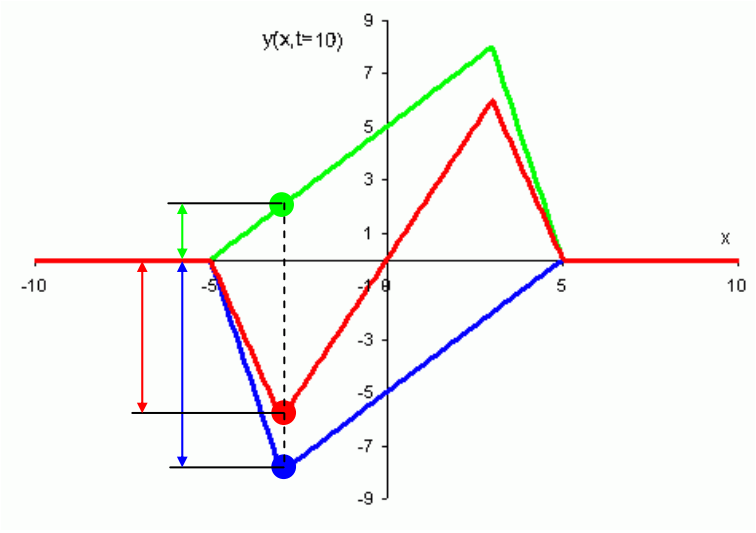
Ako smo unutar trokuta i u području desno od točke x_1 (Slika 4.0 i Slika 4.1):

$$\mathbf{y_2 = a_2d_{21} + [(x - v_2t) - x_{02} - d_{21}]\, b_2} \tag{59}$$

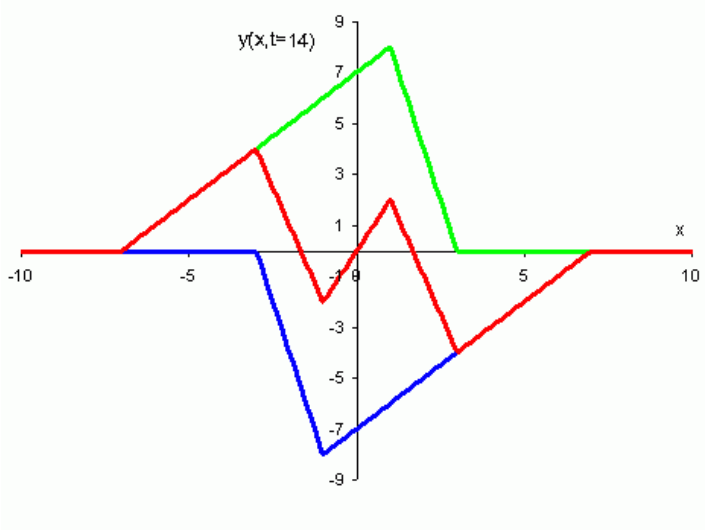
Krivulja na slici 4.6 označena crvenom bojom predstavlja superpoziciju($y_1(x,t)+y_2(x,t)$) gore definiranih valnih paketa.



Slika 4.5 Dva invertirana valna paketa u trenutku t=0



Slika 4.6 Superpozicija invertiranih valnih paketa u trenutku t=10



Slika 4.7 Superpozicija invertiranih valnih paketa u trenutku t=14

2. Programski jezik Java

2.1 Java

Zamislite da ste programer aplikacija. Programski jezik u kojem programirate je C ili C++. U posljednjih nekoliko godina u svijetu računalstva se pojavi mnoštvo inkompatibilnih računalskih arhitektura i još više inkompatibilnih operativnih sistema. Zadaća vam je da razvijete aplikaciju koja će se izvršavati u svim tim okruženjima. Shvatili ste da programski jezici C i C++ koje imate na raspolaganju i nisu baš od neke koristi u ovoj situaciji. Kažete sami sebi:″ Postoji li bolji način?″ I zaista postoji bolji način - to je Java.

Krajem 1990. skupina programera iz tvrtke Sun Microsystem pod vodstvom Jamesa Goslinga i Billa Joya krenula je u kreiranje jedinstvenog programskog jezika. Cilj im je bio napraviti programski jezik čije će se aplikacije moći izvršavati na računalima različitih proizvođača i pod različitim operativnim sistemima. To znači da će se aplikacija napisana u tom programskom jeziku moći izvršavati na PC-u, Mac-u ali i pod različitim operativnim sistemima kao što su Unix,Linux,Windows…Nakon 5 godina uspjeli su ostvariti svoj cilj, kreiran je novi programski jezik pod imenom Java. I sami smo svjedoci da im je u tome uspjelo. Java aplikacije se danas izvršavaju na mobilnim uređajima, Internetu…

2.2 Karakteristike Jave

Kako bismo se upoznali sa mogućnostima Jave nabrojimo neke od njenih glavnih karakteristika koje će pokazati da Java zaista jest ″bolji način″. Pa krenimo redom:

- **Sličnost i jednostavnost**

  Ove dvije karakteristike možete i osobno primijetiti ako se bez ikakvog predznanja upustite programirat u Javi. Ono što ćete odmah zaključiti da je Java vrlo jednostavna, ukoliko ste prije programirali u C i C++, uočit ćete sličnost u sintaksi Jave i C odnosno C++.

- **Robusnost**

  Osobina koja čini Javu robusnom je nepostojanje pokazivača, na taj način je onemogućeno pretrpavanje memorije ili slučajno ulaženje u tuđi memorijski prostor. Naime izvedbeni dio Jave se brine o memoriji npr. oslobađanje memorije se vrši automatski, ta tehnika se naziva ″automatic garbage collection″.

- **Nezavisnost o strojnoj platformi , prenosivost**

  Aplikacija napisana u Javi prenosiva je među raznim strojnim platformama. Dovoljno je jednom napisati programski kod aplikacije i možete ga izvoditi bez ikakvih izmjena u kodu na raznim strojnim platformama i pod raznim operativnim sustavima.

- **Performanse i višenitnost**

  Java podržava višenitnost i time omogućava brže izvođenje paralelnih aktivnosti u aplikaciji(npr. pomicanje nekog objekta i istovremeno sviranje glazbe u web pregledniku). Također postoji mogućnost pisanja dijelova aplikacije u C-u čime se dobiva na brzini. Performansama pridonosi i već prije navedena tehnika ″automatic garbage collection″, koja omogućava upravljanje memorijom kada je to aplikaciji potrebno.

- **Sigurnost**

  Java nudi sigurnost zato jer ima ugrađene mehanizme za zaštitu od virusa i drugih bugova. Neki od mehanizma su:

  -računalna memorija nije izravno dostupna ni jednom Java programu

  -provjera bajt-kodova da bi se ustanovile neželjene izmjene

U tablici[2] je usporedba Jave i ostalih programskih jezika s obzirom na gore navedene karakteristike:

	Java	**SmallTalk**	**TCL**	**Perl**	**Shells**	**C**	**C++**
Jednostavnost	●	●	●	◐	◐	◐	○
Robusnost	●	●	●	●	●	○	○
Sigurnost	●	◐	◐	●	◐	○	○
Višenitnost	●	○	○	●	○	○	○
Prenosivost	●	◐	●	●	◐	◐	◐
Objektno orijentiran	●	●	○	●	○	○	◐
garbage collection	●	●	○	○	○	○	○
Performanse	Visoko	Srednje	Nisko	Srednje	Nisko	Visoko	Visoko

● Osobina dostupna ◐ Osobina nije u potpunosti dostupna ○ Osobina nije dostupna

Iz gornje tabele vidimo da Java nudi najviše osobina koje su potrebne za razvoj aplikacija. Uz Javu napomenimo još da Pearl i SmallTalk također nude dovoljno mogućnosti te su dobri za razvoj aplikacija.

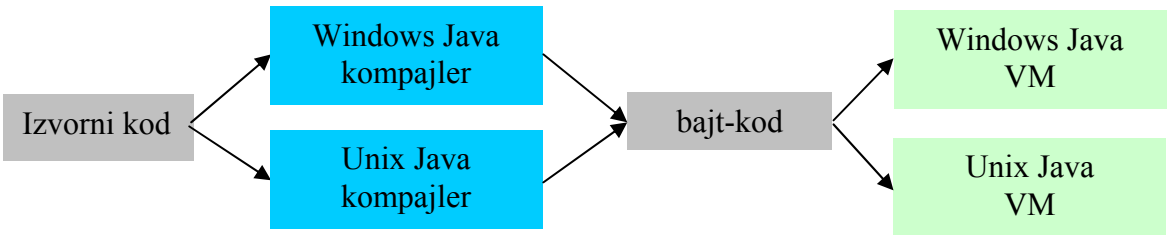
2.3 Izvršavanje programa

Već smo prije napomenuli da su programi napisani u Javi prenosivi među raznim strojnim platformama. To je ostvarivo zato što se Java programi prevode pa interpretiraju.

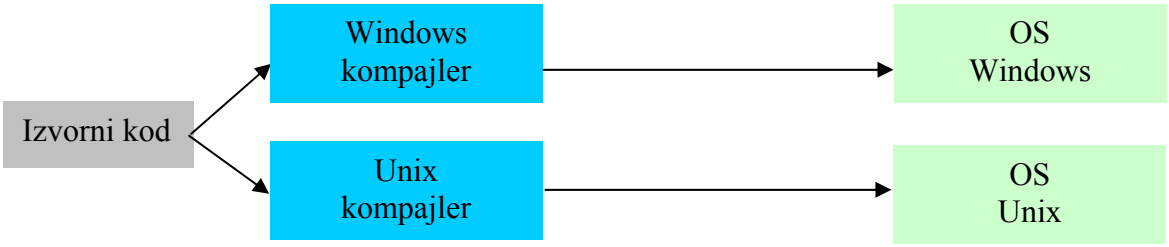
Objasnimo sada postupak prevođenje-interpretiranje u Javi. Izvorni kod(source code) se prvo prevodi(kompajlira) u tzv. **bajt-kod(bytecode)** zatim slijedi interpretacija koju provodi Java virtualna mašina(**Java VM**). Java VM se nalazi u sklopu **Java Runtime Enviromenta**(Java izvedbena okolina-**JRE**). Za svaki operativni sistem postoji različita Java izvedbena okolina(npr. za Linux, Windows, Solaris..). Java izvedbena okolina provjerava primljeni bajt-kod te ga zatim Java VM interpretira. Takav princip (prevođenje-interpretiranje) omogućava Java aplikacijama neovisnost o strojnoj platformi . Tj. izvorni kod u Javi možemo prevesti u bajt-kod npr. pod Unixom, a zatim taj bajt-kod interpretirati pod Windowsima(Slika 4.8).

[2] Usporedba programskih jezika po James Goslingu i Henry McGiltonu.

Kod ostalih jezika npr. C, C++ izvorni kod se direktno prevodi u strojni jezik karakterističan za dani operativni sistem(Slika 4.9). Izvorni kod napisan u C ili C++ je djelomično prenosiv, pogotovo ako se u kodu programiraju mrežni servisi. Dok su java izvorni kodovi potpuno prenosivi, dakle: ″Napiši jednom vrti svugdje″.



Slika 4.8 Prevođenje java programa na različitim operativnim sustavima



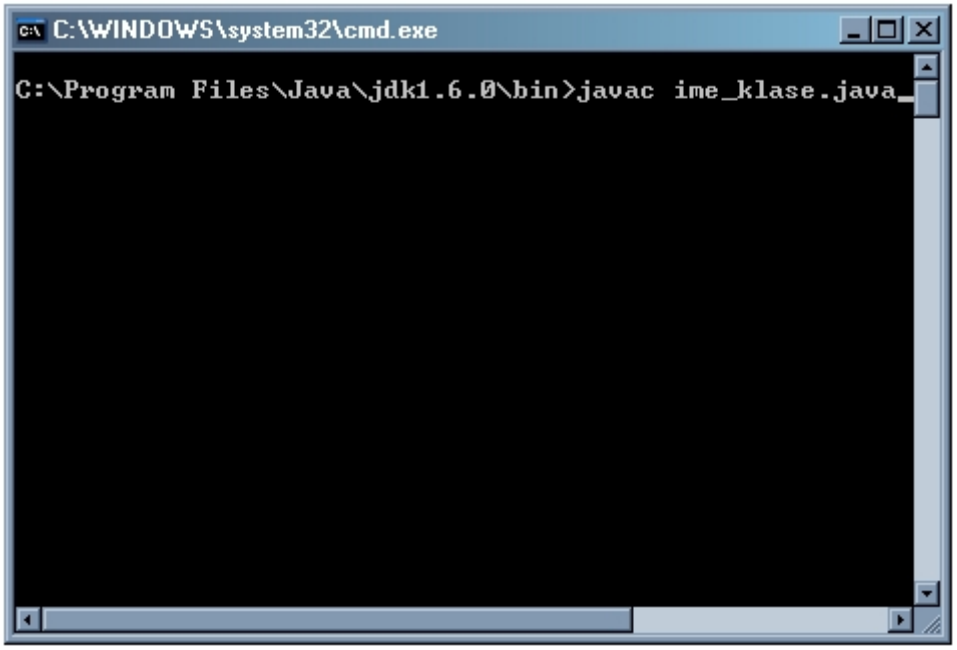
Slika 4.9 Prevođenje programa napisanog u C ili C++ na različitim operativnim sustavima

2.4 Java development Kit (JDK)

Dakle da bismo mogli pokrenuti pod našim operativnim sustavom aplikaciju napisanu u Javi, moramo imati instalirani JRE(Java Runtime Enviroment). JRE u sebi sadrži Java VM koja omogućava interpretaciju bajt-koda.

Ukoliko pak želimo prevoditi izvorne kodove napisane u Javi, potrebno je da instaliramo JDK. JDK je razvojno okruženje koje u sebi uključuje alate koji nam omogućavaju razvoj, prevođenje i interpretaciju(Java VM) Java aplikacija. Razvojni alati se nalaze u poddirektoriju bin i pokreću se iz komandne linije(komadnog prompta)(Slika 5.0). Nabrojimo neke od njih:

- **javac.exe** –služi za prevođenje izvornog koda u Javi u bajt-kod. Izvorni kod u Javi je oblika ime_klase.java, a kad se prevede u bajt-kod dobiva ekstenziju class dakle ime_klase.class.



Slika 5.0 Prevođenje Java programa u komadnom promptu

- **java.exe** –služi za interpretiranje Java bajt-koda, izvršava bajt-kod aplikacije.
- **appletviewer.exe**[3] –služi za izvršavanje bajt-koda apleta.

[3] Da bismo koristili razvojni alat appletviewer prethodno moramo bajt-kod apleta implementirat unutar web-stranice(Poglavlje 2.8.2).

2.5 Objektno-orijentirano programiranje

Java je objektno orijentiran programski jezik. Temelj objektno-orijentiranog programiranja su **klase i objekti**. Svugdje oko nas se nalaze objekti npr. automobili, zgrade, drveća… Ako promatramo automobile, misleći pritom na sve automobile , a ne na neki konkretni automobil, tada nam ti automobili predstavljaju **klasu**. Suzimo sada naš izbor na plavu Hondu Accord. Ako saznamo sve detalje o tom automobili (godina proizvodnje, jačina motora, vrsta motora, vlasnika) znamo o kojem se konkretnom vozilu radi te nam to vozilo tada predstavlja **objekt. Dakle objekt nam predstavlja konkretizaciju klase, a klasa je skup mnogo objekata iste vrste.** Klase mogu imati neka **svojstva** npr. kod klase automobil svojstva bi bila: marka, tip, jačina motora, max. brzina…Klase također mogu imati i **metode**, tj. radnje koje će se nad njima izvršavati npr. za klasu automobil: dodaj gas, promijeni brzinu, promijeni ulje…Sljedeći važan pojam unutar klase je **konstruktor.** Konstruktor je metoda unutar klase koja ima isto ime kao i klasa. Zadaća mu je incijaliziranje svojstva objekata. Na kraju ovog poglavlja navedimo još jednu važnu karakteristiku, a to je **nasljeđivanje klasa**. Nasljeđivanje klasa se koristi kako bi se izbjeglo višestruko pisanje istih dijelova koda.

2.6 Java komponente

2.6.1 AWT(Abstract Window Toolkit)

AWT paket sadrži u sebi mnoštvo gotovih klasa koje nam služe za generiranje elemenata grafičkog sučelja. Korištenjem AWT klasa ne dolazite u dodir s detaljima grafičkog sučelja, već koristimo gotove klase i metode.Ukoliko jedna klasa nasljeđuje drugu, ona nasljeđuje i metode te klase. AWT paket pozvat ćemo naredbom:

```
import java.awt.*
```

Klase unutar AWT paket se dijele na sljedeće klase:

- AWT kontejneri
- AWT komponente
- Ostale AWT klase

2.6.2 AWT kontejneri

AWT kontejneri mogu sadržavati AWT komponente i druge kontejnere. Pod kontejnere spadaju okviri (Frames), dijalogi (Dialogs),prozori (Windows) i paneli(Panels).

2.6.2.1 Panel

Panel je najjednostavnija klasa kontejnera. Klasa sadrži definiciju jednog konstruktora, a sve ostale metode nasljeđuje iz drugih klasa.

Konstruktor:

public Panel();

- kreira novi panel

Metode:

public void setSize(int arg0, int arg1);

- definira veličinu panela

public void setLocation(int arg0, int arg1);

- definira mjesto unutar Containera na kojem će se nalaziti Panel

public void setBackground(Color arg0);

- definira boju Panela

Primjer kreiranja Panela trokut:

```
Panel trokut;
trokut = new Panel();
trokut.setSize(600,100);
trokut.setLocation(0,300);
trokut.setBackground(Color.BLACK);
add(trokut);
```

Da bismo dodali Panel na kontejner koristimo metodu **add** klase Container. Opći oblik metode add je:

public void add(Component comp);

2.6.3 AWT komponente

Komponente su grafički elementi korisničkog sučelja koje se koriste za interakciju s korisnikom. U ovome ćemo poglavlju objasniti samo one koje sam koristio u appletu. To su: Gumb(Button), Labela(Label) i Izbor(Choice).

2.6.3.1 Gumbi(Buttons)

Gumb je jednostavna komponenta koja generira događaj kada pritisnemo na nju. Izgled gumba ovisi o operativnom sistemu pod kojim se aplikacija izvršava. Gumb se kreira pomoću klase Button.

Konstruktor:

public Button(String label);

- konstruira gumb s tekstom label.

Metode:

public void setSize(int arg0, int arg1);

- definira veličinu gumba

public void setLocation(int arg0, int arg1);

- definira mjesto unutar Containera na kojem će se nalaziti gumb

Primjer kreiranja gumba bstart:

```
Button bstart;
bstart = new Button("Start");
bstart.setSize(90,20);
bstart.setLocation(130,15);
bstart.addActionListener(this);
glavni.add(bstart);
```

U zadnjoj liniji primjera smo opet korištenjem metode **add** dodali objekt bstart na panel glavni.

2.6.3.2 Labela(Label)

Labela služi za ispis određenog teksta. Kao i kod svakog drugog teksta moguće je odrediti veličinu slova, boju, vrstu fonta. Labela se kreira pomoću klase Label.

Konstruktor:

public Label(String label);

- konstruira labelu s tekstom label.

Metode:

public void setSize(int arg0, int arg1);

- definira veličinu labele

public void setLocation(int arg0, int arg1);

- definira mjesto unutar Containera na kojem će se nalaziti labela

setForeground(Color arg0);

- definira boju teksta

setFont(Font arg0);

- definira vrstu fonta, veličinu fonta…

Primjer kreiranja labele ime te njezino dodavanje na panel glavni:

```
Label ime;
ime = new Label("M.Đurić,2006");
ime.setForeground(Color.BLACK);
ime.setFont(new Font("Arial",Font.BOLD,10));
ime.setSize(70,10);
ime.setLocation(520,38);
glavni.add(ime);
```

2.6.3.3 Izbor(Choice)

Izbor je komponenta koja omogućava korisniku izbor između više ponuđenih opcija. Izbor se kreira pomoću klase Choice.

Konstruktor:

public Choice();

- konstruira novi izbor.

Metode iz klase Choice:

public void addItem(String item);

- dodaje novi element u izbor

public String getSelectedItem();

- vraća izabrani element izbora

Metode koje nasljeđuje iz drugih klasa:

public void setSize(int arg0, int arg1);

- definira veličinu izbora

public void setLocation(int arg0, int arg1);

- definira mjesto unutar Containera na kojem će se nalaziti izbor

Primjer kreiranja izbora a_drugi,dodavanje na panel trokut:

```
Choice a_drugi;
a_drugi = new Choice();
a_drugi.setLocation(480,30);
a_drugi.setSize(70,20);
trokut.add(a_drugi);
```

Primjer dodavanja novih elemenata u izbor a_drugi:

```
a_drugi.addItem("-3");
a_drugi.addItem("2");
a_drugi.addItem("1");
```

Primjer vraćanja izabranog elementa iz izbora a_prvi:

```
string=a_prvi.getSelectedItem();
```

2.6.4 Događaji

Kao što smo već rekli pritiskom na gumb generira se neki događaj. Rad s događajima omogućava nam klasa **ActionListener** koja sadrži samo zaglavlje metode **actionPreformed**. Unutar tijela metode actionPreformed definirat ćemo kod koji će se izvršavati kao reakcija na neki događaj.

Da bismo mogli koristiti klasu ActionListener moramo je najaviti na sljedeći način:

```
public class superpozicija extends Applet implements ActionListener
```

Također moramo uključiti paket u kojem se nalazi klasa ActionListener:

```
import java.awt.event;
```

Sljedeći korak je registrirati elemente na kojima želimo pratiti događaje:

```
ime_objekta.addActionListener(this);
```

Primjer:

```
import java.awt.event;
.
.
.
public class superpozicija extends Applet implements ActionListener{
.
.
.
bstart = new Button("Start");
bstart.setSize(90,20);
bstart.setLocation(130,15);
bstart.addActionListener(this);
glavni.add(bstart);
.
.
.
public void actionPerformed(ActionEvent e)
        {

                if(e.getSource() == bstart)
                {
                        racunaj();
                }
        }

}
```

Parametar **e** metode actionPerformed ima sve informacije o događajima, pa zna nad kojim elementom je događaj izvršen. Metoda **getSource()** vraća ime objekta nad kojim je izvršen događaj, nju pozivamo nad objektom tipa **ActionEvent**.

2.7 Grafika u Javi

Metoda koja nam omogućuje rad s grafikom u Javi je public void paint(Graphics g). Metoda kao parametar prima grafički objekt tipa Graphics. Grafički objekt predstavlja područje unutar kojeg je dozvoljeno crtanje. Ako želimo koristiti napredene mogućnosti 2D grafike, potrebno je napraviti pretvorbu grafičkog objekta unutar metode paint():

```
Graphics offGraphics;
Graphics2D offGraphics2D;
offGraphics2D = (Graphics2D) offGraphics;
```

2.7.1 Crtanje grafičkih oblika

Metode koje se koriste za crtanje grafičkih oblika se nalaze u klasi Graphics. Navest ću metode koje sam koristio u appletu:

Metode:

```
public abstract void drawLine(int x1, int y1, int x2, int y2);
```

- crta liniju od točke (x1,y1) do točke (x2,y2)

```
public void drawRect(int x,int y, int width, int height);
```

- crta pravokutnik širine width i visine height s ishodištem u točki (x1,y1)

```
public abstract void fillRect(int x,int y, int width, int height);
```

- crta ispunjeni pravokutnik širine width i visine height s ishodištem u točki (x1,y1), boje ispunjenja je definirana trenutnom bojom grafičkog objekta.

2.7.2 Crtanje teksta

Osim crtanje osnovnih grafičkih oblika, grafički objekt omogućava i crtanje teksta. Za crtanje teksta koriste se klase Graphics i Font.

Metode za crtanje definirane su unutar klase Graphics:

```
public abstract void drawString(String str, int x, int y);
```

- crta znakovni niz str na poziciji definiranoj koordinatama x,y.

2.7.3 Klasa Font

U klasi Font definirane su metode za rad s fontovima. Klasa sadrži jedan konstruktor.

Konstruktor:

```
public Font(String name, int style, int size);
```

- konstruira font s nazivom name, stilom style i veličinom size.

Sada ćemo navesti primjer koji demonstrira crtanje grafičkih elemenata u Javi:

```
import java.applet.Applet;
import java.awt.*;

public class primjer extends Applet {
public void paint(Graphics g){
        Graphics2D g2D;
        g2D = (Graphics2D) g;

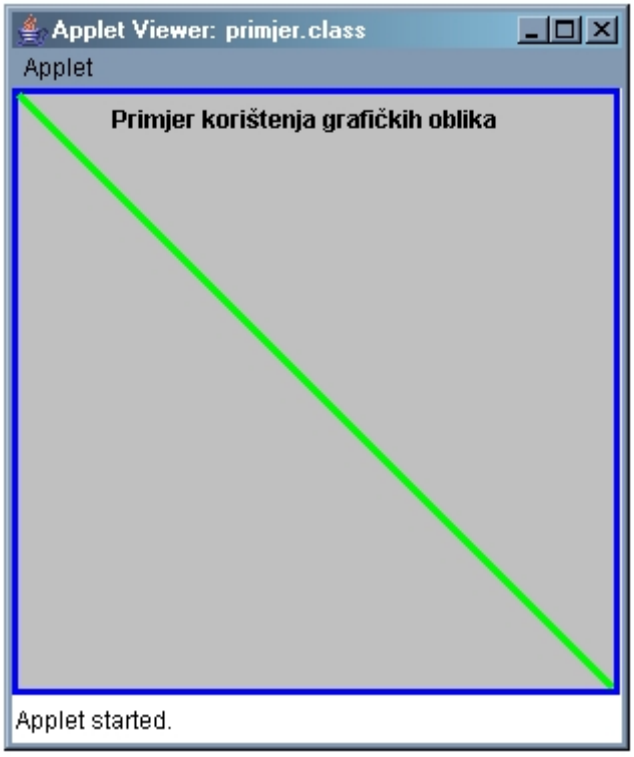
        g2D.setStroke(new BasicStroke(5));
        g2D.setColor(Color.blue);
        g2D.drawRect(0,0,300,300);

        g2D.setColor(Color.lightGray);
        g2D.fillRect(3,3,297,297);

        g2D.setStroke(new BasicStroke(3));
        g2D.setColor(Color.green);
        g2D.drawLine(4,4,298,298);

        g2D.setColor(Color.BLACK);
        g2D.setFont(new Font("Arial",Font.BOLD,12));
        g2D.drawString("Primjer korištenja grafičkih oblika",50,20);
}
}
```

Nakon što izvršimo gornji primjer dobijemo:



Slika 5.1 Grafički elementi u Javi

2.7.4 Korištenje slika u Javi

Unutar paket java.awt, java.awt.image su definirane metode za rad sa slikama. Za učitavanje slika koristi se metoda **getImage()** koja je definirana unutar klasa Applet i AWT-a. Prije nego što učitamo sliku, potrebno je definirati objekt tipa Image.

Metode koje koristimo *za učitavanje slika* su:

public URL getCodeBase();

- vraća lokaciju gdje se slika nalazi

public Image getImage(URL url,String name);

- učitava sliku name sa lokacije url

Metode za učitavanje slika su definirane u klasi Applet unutar java.applet paketa.

Za prikazivanje učitanih slika se koriste metode **drawImage()** koje su definirane u klasi Graphics.

Metode koje koristimo za *prikaz učitanih slika*:

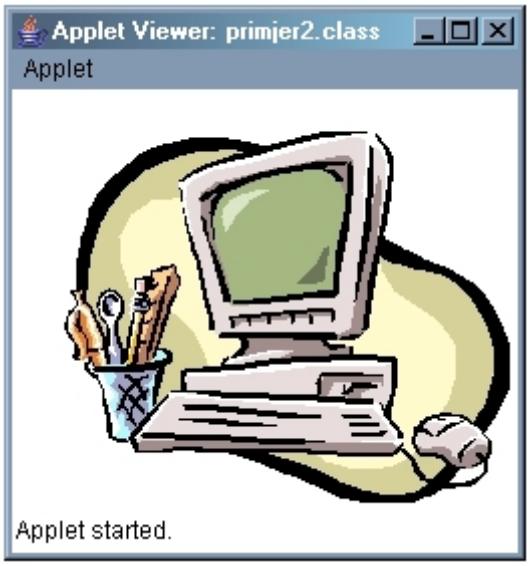
public abstract boolean drawImage(Image img, int x, int y, this)

- prikazuje sliku img na poziciji s koordinatama (x,y)

Primjer koda u kojem se koriste metode za učitavanje i prikazivanje slika:

```
import java.applet.Applet;
import java.awt.*;
public class primjer2 extends Applet{
Image slika;
public void init(){
        slika=getImage(getCodeBase(),"pc.jpg");
        }
public void paint(Graphics g){
        g.drawImage(slika,0,0,this);
        }
}
```

Nakon što izvršimo gornji kod dobijemo:



Slika 5.2 Izgled primjer2 appleta

2.8 Java applet

Java applet je java program koji se izvršava unutar Internet preglednika. Naravno da bi mogli izvršiti applet na računalu mora biti instalirana JRE koji u sebi sadrži Java VM-interpreter koji služi za interpretaciju bajt-koda(vidi poglavlja 2.3 i 2.4).

2.8.1 Prevođenje java appleta

Prevođenje java appleta se vrši kao i prevođenje drugih java programa. Na računalu mora biti instaliran JDK koji u sebi sadrži razvojne alate za izradu java programa(detaljnije vidi poglavlje 2.4).Da bismo preveli neki izvorni kod u Javi potrebno je u komadnom promptu napisat naredbu:

```
javac ime_appleta.java
```

Javac. exe je java prevodilac koji prevodi izvorni kod u Javi u bajt-kod. Prevođenjem izvornog koda ime_appleta.java dobijemo bajt-kod ime_appleta.class.

2.8.2 Pokretanje Java appleta

Da bismo mogli pokrenuti applet potrebno je napisati HTML datoteku iz koje se pokreće Java applet. Primjer HTML datoteke sa Java appletom:

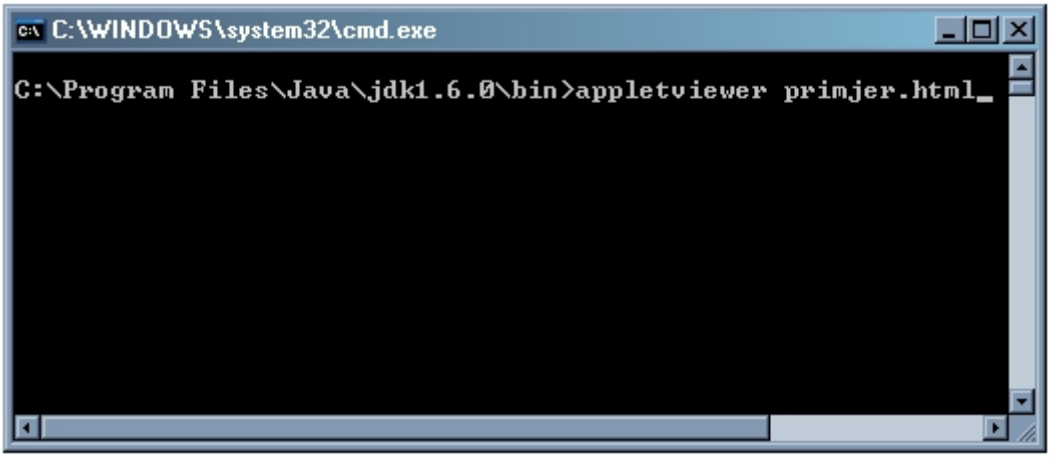
```
<html>
<body>
<applet code=ime_appleta.class width="304" height="304" >
</applet>
</body>
</html>
```

Spremimo sada HTML datoteku po imenom primjer.html. Pokretanje appleta se može vršiti na dva načina:

- pomoću Internet preglednika(Slika 5.3).
- korištenjem **appletviewer.exe** koji se nalazi u sklopu JDK razvojnih alata(Slika 5.4).



Slika 5.3 Pokretanje appleta iz Internet preglednika



Slika 5.4 Pokretanje Java appleta iz komadnog prompta

2.8.3 Životni ciklus appleta

Svaki applet ima svoj životni ciklus. Stanja kroz koja applet prolazi tijekom svog životnog ciklusa su:

- inicijalizacija
- pokretanja
- zaustavljanje
- oslobađanje resursa

Prilikom učitavanja HTML datoteke koja sadrži applet, applet se prvo incijalizira, zatim pokrene. Ako napustimo Internet stranicu na kojoj se nalazi applet npr.(minimiziramo preglednika) applet se zaustavi, ponovnim prikazivanjem preglednika applet se opet pokrene. Ako pak zatvorimo preglednik, applet se završava i vrši se oslobađanje sistemskih resursa.

Metode koje nam omogućavaju realizaciju stanja životnog ciklusa appleta su:

- init()
- start()
- stop()
- destroy()

2.8.3.1 Metoda init()

Ova metoda se izvodi svaki put kada učitamo applet. Izvodi se prva po redu, vrijeme izvođenje je vrlo kratko. U njoj se obično vrši učitavanje slika te inicijalizacija varijabli.

2.8.3.2 Metoda start()

Metoda koja se izvršava nakon metode init(). U njoj se obično definira animacijska dretva(nit). O tome ćemo malo kasnije.

2.8.3.3 Metoda stop()

Metoda stop() se koristi za zaustavljanje izvođenje applet kako se ne bi trošili sistemski resursi. Ako napustimo Internet stranicu na kojoj se nalazi applet npr.(minimiziramo preglednika)pokreće se metoda stop() koja zaustavlja izvođenje appleta.

2.8.3.4 Metoda destroy()

Metoda koja se izvršava kada ugasimo Internet preglednik. Služi za oslobađanje sistemskih resursa, koje je applet koristio.

2.8.4 Animacijska dretva(nit)

Efekt animacije se postiže slijednim prikazivanjem slika velikom brzinom. Da bi se postigla razina realne animacije, brzina izmjena slika mora biti 24 u sekundi.

Kod animacije je najvažnije pravilno definirati animacijsku petlju. Animacijska petlja se izvodi unutar animacijske dretve(niti) te se brine o pravilnom izmjenjivanju slika i o redovitom osvježavanju ekrana. Definiranje dretve omogućuje nam klasa Thread i sučelje Runnable.

Prvo je potrebno deklarirat klasu koja implementira Runnable sučelje. Runnable je klasa koja u sebi sadrži samo zaglavlje run() metode. Sve klase koje u sebi sadrže samo zaglavlja metoda zovu se sučelja(interface). Dakle sada je potrebno metodu run() implementirati unutar klase te definirati njezino tijelo. Kreiranje i pokretanje dretve se vrši u metodi start():

```
Thread animatorThread;
animatorThread=new Thread(this);
animatorThread.start();
```

Prvo je potrebno definirati objekt animatorThread koji je tipa klase Thread. Zatim pozivanjem konstruktora klase Thread definiramo objekt. Te na kraju pokrećemo taj objekt metodom start(). Objekt animatorThread nam predstavlja animacijsku dretvu unutar koje će se izvoditi animacijska petlja.

Objasnimo sada pojedine dijelove našeg programa kako bismo shvatili način izvođenje animacije.

```
int delay,fps;
fps=24;
delay=1000/fps;
```

fps –varijabla pomoću koje definiramo brzinu animacije koju želimo, tj. koliko slika želimo prikazati u sekundi

delay –varijabla koja se dobije računski tj. 1000/fps. Ona definira broj milisekundi između prikazivanja dvije uzastopne slike.

Pogledajmo sad run() metodu:

```
public void run()
        {

                Thread.currentThread().setPriority(Thread.MIN_PRIORITY);

                long startTime = System.currentTimeMillis();

                while(Thread.currentThread() = = animatorThread)
                {
                        repaint();
                        try
                        {
                                startTime=startTime+delay;
                        Thread.sleep(Math.max(0,startTime-System.currentTimeMillis( )));
                } catch (InterruptedException e1){stop();}

                }

        }
```

1. poziva se repaint() metoda koja zahtijeva iscrtavanje sljedeće slike.
2. čeka se određeno vrijeme definirano vrijednošću varijable delay.

Da bismo dobili pravilan vremenski interval između slika, tj. periodičnost, potrebno je dretvu staviti u stanje spavanja. To se postiže metodom public vodi sleep(long ms) klase Thread. Vrijeme koje dretve mora spavati određeno je varijablom delay.

2.8.5 Eliminacija bljeskanja kod animacija

Prilikom izrade appleta često se nailazi na problem bljeskanja koji se javlja kod animacija. Pogotovo ako se za prikaz slike koriste složena računanja. Događa se da je vrijeme računanja koordinata valnih paketa i prikazivanja na ekranu dulje od vremena osvježavanja ekrana. Konkretno to znači da se npr. prva polovica slike iscrta prilikom prvog osvježenja ekrana, a druga polovica prilikom drugog i zbog toga se javlja bljeskanje. Da bismo doskočili tom problemu koristi se tehnika :

dvostrukog spremnika(double buffering)

Ta tehnika se koristi neprikazanom grafikom(offGraphics) na kojoj se obavlja crtanje i tek kada je slika u potpunosti gotova ona se prikaže ne ekranu. U daljnjem kontekstu ću navest dio koda iz programa u kojem se koristi tehnika dvostrukih spremnika:

```
if ( (offGraphics == null) || (d.width != offDimension.width) || (d.height != offDimension.height) )
                {
                offDimension = d;
                offImage = createImage(d.width, d.height);
                offGraphics = offImage.getGraphics();
                }
```

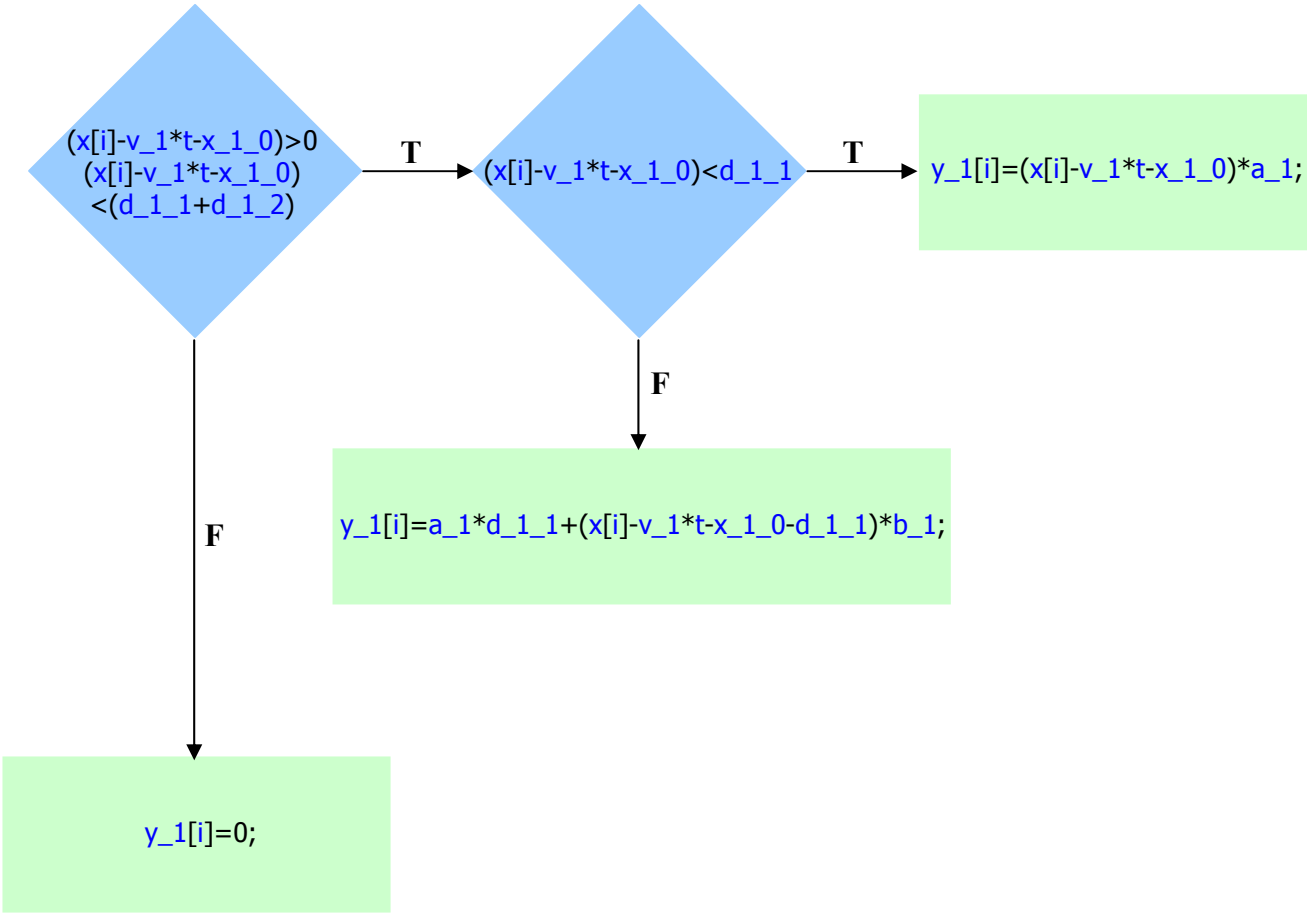
Grafički objekt offGraphics tipa Graphics nam definira područje za neprikazanu grafiku unutar kojeg je dozvoljeno crtanje. Nakon toga potrebno je definirati sliku odgovarajuće veličine(offImage) u koju će se učitat grafika kada završimo s crtanjem. Te na kraju prikažemo sliku(offImage).

2.8.6 Trokutasti valni paketi u Javi

U prethodnim poglavljima objasnili smo način korištenja i implementiranja komponenti koje smo koristili u appletu. U daljnjem kontekstu ću pomoću dijagrama toka objasniti način na koji sam definirao trokutasti valni paket u Javi. Matematički opis trokutastog valnog paketa koji se giba udesno brzinom v_1 dan je u poglavlju 1.8.1.3. Kod u Javi koji predstavlja programski zapis matematičkog opisa izgleda:

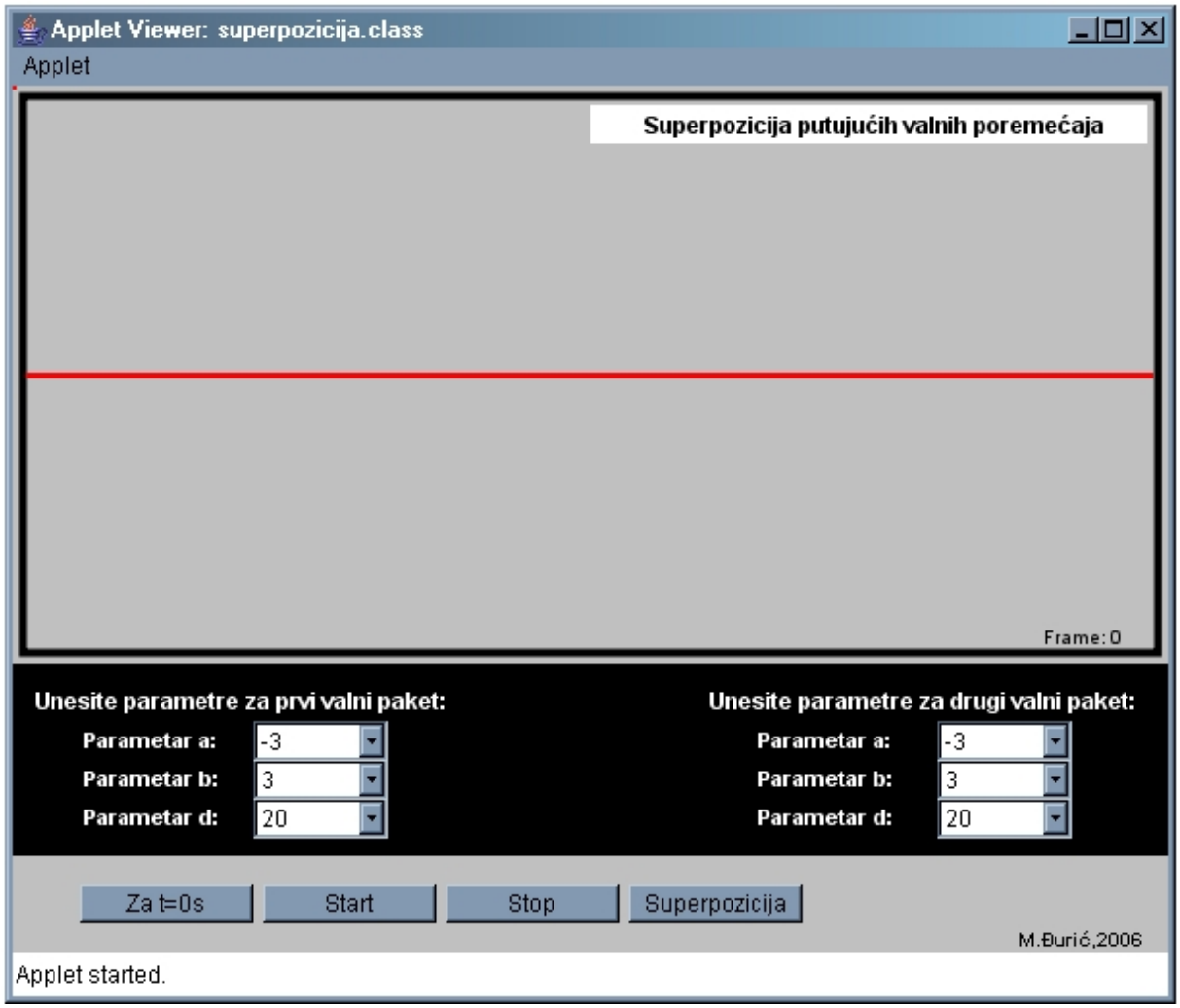
```
for(i=0;i<601;i++)
        {
                if(((x[i]-v_1*t-x_1_0)>0)&&((x[i]-v_1*t-x_1_0)<(d_1_1+d_1_2)))
                        if((x[i]-v_1*t-x_1_0)<d_1_1)
                                y_1[i]=(x[i]-v_1*t-x_1_0)*a_1;
                        else
                                y_1[i]=a_1*d_1_1+(x[i]-v_1*t-x_1_0-d_1_1)*b_1;
                else
                        y_1[i]=0;
        }
```

Dijagrama toka za gore navedeni programski kod izgleda:



2.9 Upute za korištenje appleta

U daljnjem kontekstu govorit ćemo o načinu korištenja appleta. Prvo ćemo podijeliti applet na tri djela(Slika 5.5).Prvi dio predstavlja područje na kojem se odvija simulacija, zatim slijedi područje gdje biramo parametre koji definiraju valne paket te područje sa kontrolama koje omogućavaju interakciju appleta i korisnika.



Slika 5.5 Dijelovi appleta

Na appletu se mogu odabirom pravilnih parametara simulirati superpozicija tri vrste valnih paketa:

- superpozicija trokutastih valnih paketa koji su zrcalno simetrični s obzirom na y-os
- superpozicija trokutastih valnih paketa koji su zrcalno simetrični s obzirom na x-os
- superpozicija trokutastih valnih paketa koji su zrcalno simetrični s obzirom na y-os i još k tome invertirani

Superpozicija trokutastih valnih paketa koji su zrcalno simetrični s obzirom na y-os

Da bismo ostvarili simulaciju takvih vrsta paketa potrebno je parametre odabrati na način kako je to prikazano na slici 5.6.



Slika 5.6 Odabrani parametri za takvu vrstu valnih paketa

Superpozicija trokutastih valnih paketa koji su zrcalno simetrični s obzirom na x-os

Ukoliko pak želimo dobiti prikaz simulacije invertiranih valnih paketa parametre odabiremo na način kao što je to prikazano na slici 5.7.



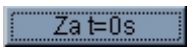
Slika 5.7 Odabrani parametri za takvu vrstu valnih paketa

Superpozicija trokutastih valnih paketa koji su zrcalno simetrični s obzirom na y-os i još k tome invertirani

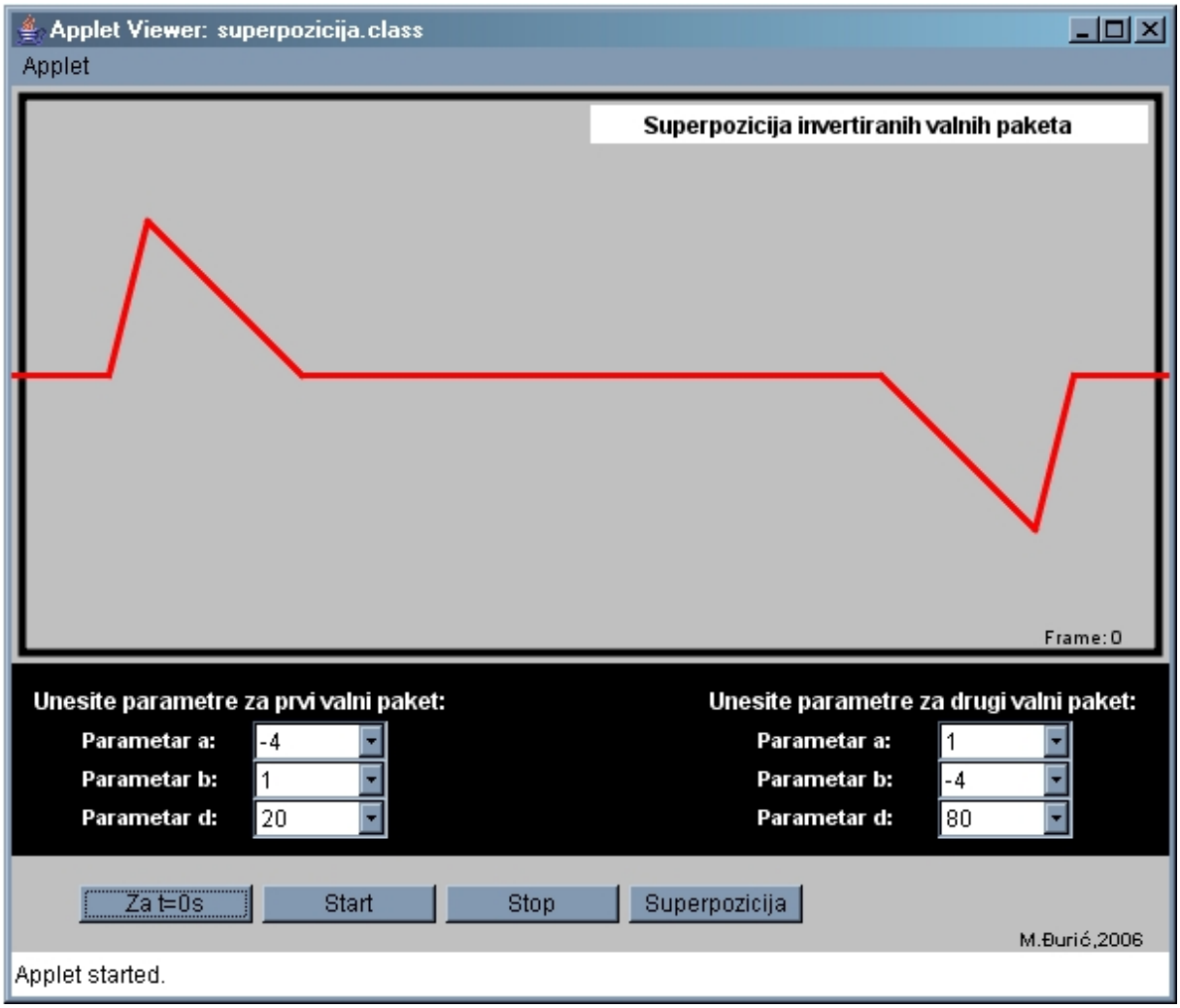
Odabrani parametri za prikaz simulacije takve vrste paketa su prikazani na slici 5.8.



Slika 5.8 Odabrani parametri za takvu vrstu valnih paketa

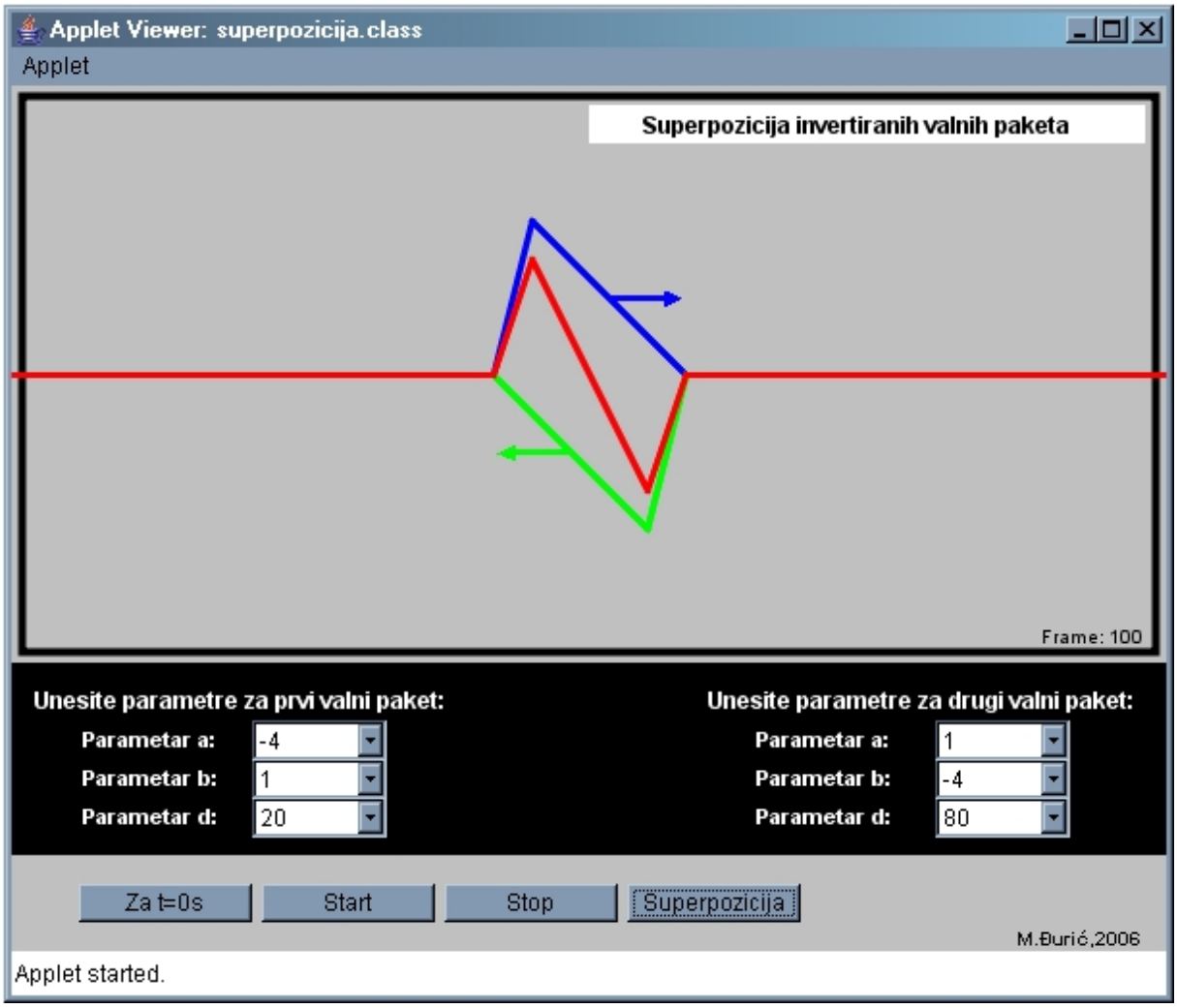
Pretpostavimo da smo odabrali parametre koji definiraju valne pakete koji su zrcalno simetrični s obzirom na y-os i invertirani, pokažimo sada kako se pokreće simulacija. Preporučuje se sljedeći slijed odabira gumbi. Prvo gumb 

koji će prikazat položaj valnih paketa u trenutku t=0 (Slika5.9).



Slika 5.9 Prikaz položaja valnih paketa u trenutku t=0

Zatim odaberite gumb Start kojim se pokreće simulacija, gumbom Stop se naravno zaustavlja simulacija. Pritiskom na gumb Superpozicija dobijemo odmah prikaz superpozicije odabranih valnih paketa(Slika 6.0).



Slika 6.0 Superpozicija valnih paketa

3. Implementacija appleta u nastavi fizike

Škola: Gimnazija

Razred: 3

Nastavni predmet: Fizika

Nastavna cjelina: Valovi

Nastavna jedinica: Superpozicija i putovanje valnih pulseva

Trajanje: 1 školski sat

Mjesto održavanja nastave: Računalna učionica, hodnik škole

Cilj: Uočiti rezultate superpozicije za različitu vrstu valnih pulseva

Nastava pomagala: Računalo sa instaliranom Javom, projektor, ploča, dugo elastično uže.

Nastavna sredstva: Java applet, Powerpoint prezentacija, Mathematica 5.0.

Nastavne metode: Rad na računalu. Po dva učenika za jednim računalom. Metoda razgovora(diskusija).Eksperiment.

Nastavni proces:

Nastavna jedinica Superpozicija i putovanje valnih pulseva je izvedena iz nastavne cjeline Valovi. Podrazumijeva se da su učenici shvatili što je zapravo val i na koji način možemo eksperimentalno ostvariti valno gibanje. Također se podrazumijeva da su učenici upoznati sa pojmom superpozicija valova.

Eksperimentalni prikaz superponiranja i gibanja valnih pulseva

Prvo ćemo eksperimentalno uvesti pojam valnog pulsa.

Postupak izvođenja eksperimenta:

Za izvođenje eksperimenta potrebno nam je dugo elastično uže. Ukoliko je učionica premala elastično uže raširimo po dužini hodnika. Prihvatimo uže rukom za jedan kraj.

Pitanje koje postavimo učenicima:

- Što moramo raditi s užetom da bi proizveli valno gibanje?

Očekivani odgovor od učenika:

- Početak užeta trebamo pomicati gore-dolje.

Uže pomičemo gore-dolje i kroz valno sredstvo(uže) se širi val. Također demonstriramo val koji nastaje kada samo jednom pomaknemo uže gore-dolje. Nakon što smo izveli eksperiment tražiti od učenika da zapišu svoja zapažanja. Dakle da zapišu koja je razlika između ta dva valna gibanja. Kad su zapisali, slijedi rasprava o njihovim zapažanjima.

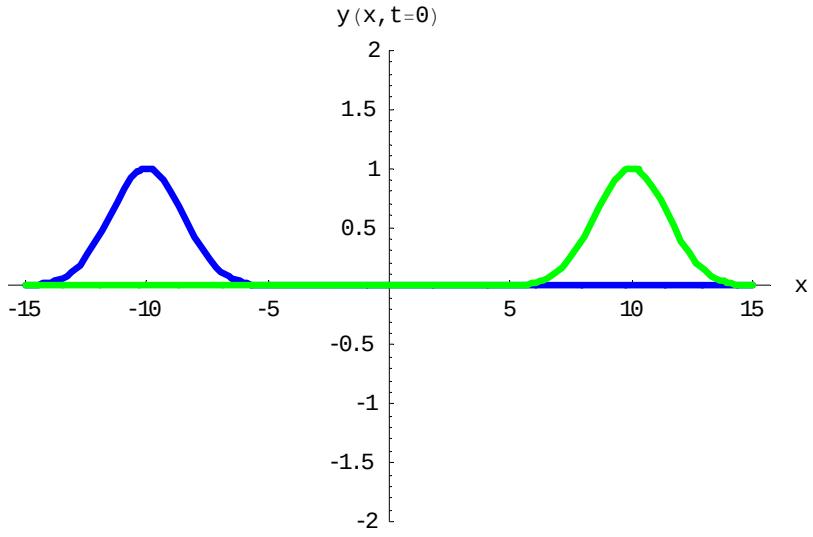
Očekivano zapažanje kod učenika:

- Kada neprestano pomičemo uže gore dolje učenici zamjećuju da se cijelo uže neprestano pomiče, a kada samo jednom pomaknemo uže gore-dolje postoje dijelovi užeta koji miruju i dijelovi koji se gibaju.

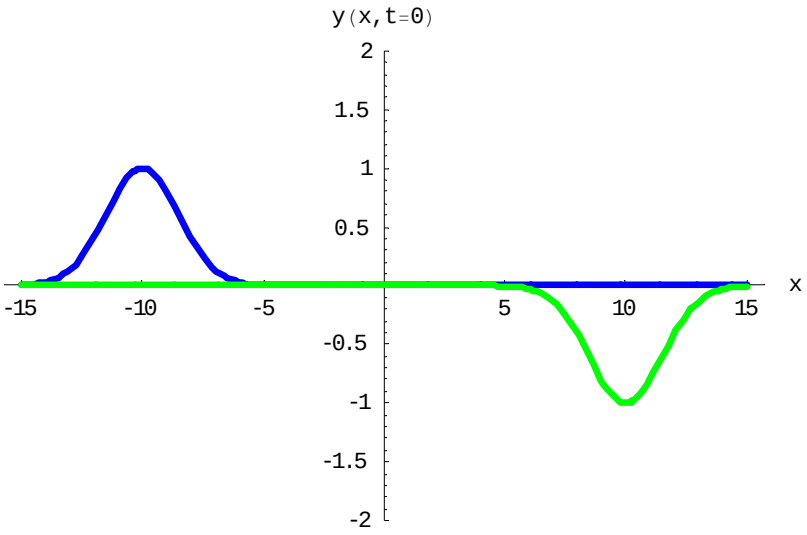
Zatim uvedemo pojam valnog pulsa kao valno gibanje koje nastaje ako jednom uže pomaknemo gore-dolje. Onda odaberemo jednog učenika iz razreda koji će proizvesti valni puls s jedne strane, a profesor valni puls s druge strane užeta. Tražiti od učenika da se okupe oko užeta i da promatraju što će se dogoditi kad se ta dva valna pulsa susretnu. Upozoriti ih da obrate pažnju što će dogoditi prilikom njihova susreta. Tj. da li će se valni pulsevi poništiti ili neće? Otvoriti raspravu o njihovim zapažanjima. Podsjetiti ih da se takav fenomen kada se dva valna pulsa susretnu zove superpozicija. Ovim eksperimentom smo uveli pojam valnog pulsa i podsjetili učenike na superpoziciju valova.

Primjeri valnih pulseva opisanih Gaus funkcijom

Pomoću projektora projiciramo na platno dvije slike sa računala. Prva slika neka pokazuje položaje dvaju simetrična valnih paketa koji nisu invertirani u trenutku t=0(Slika 6.1), a druga položaje dvaju valnih paketa koji jesu invertirani isto u trenutku t=0(Slika 6.2).



Slika 6.1 Dva simetrična valna paketa u trenutku t=0



Slika 6.2 Dva invertirana simetrična valna paketa u trenutku t=0

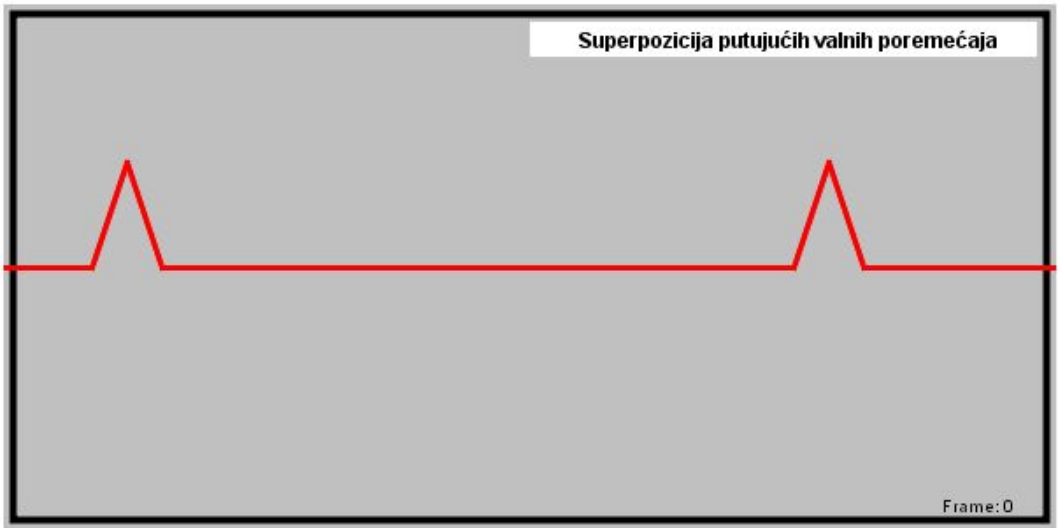
Učenici neka iznesu svoje pretpostavke o tome što će se dogoditi kada se ta dva valna paketa nađu u istoj točki prostora, tj. kako će izgledati rezultat superpozicije u prvom slučaju, a kako u drugom. Napomenut učenike da smo takve valne pulseve maloprije pokušali proizvesti na užetu.

Nakon toga pokazati učenicima u programu Mathematici simulaciju koja pokazuje superponiranje takvih valnih pulseva na taj će način učenici provjeriti dali su njihove pretpostavke bile točne. Promatrajući simulaciju upozorimo učenike neka obrate pozornost na oblik, brzinu i veličinu individualnih valnih pulseva prije i nakon susreta. Otvorimo raspravu o njihovim zapažanjima. Program u Mathematica koji nam omogućava prikaz simulacije simetričnih valnih paketa je priložen na CD-u na kraju diplomskog.

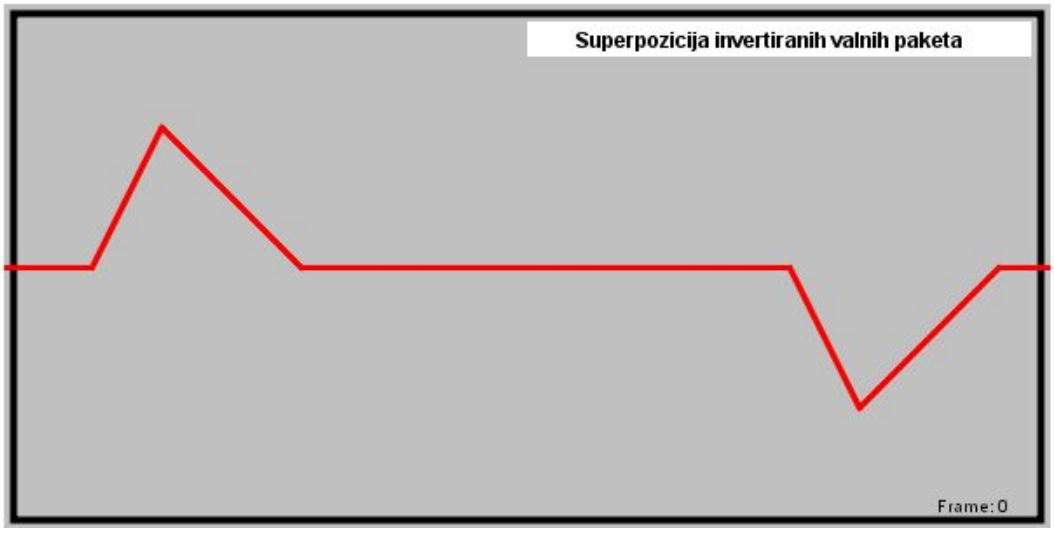
Primjeri trokutastih valnih pulseva

Na isti način kao što smo uveli valne pakete opisane Gaus funkcijom, na taj način uvedemo i trokutaste valne pakete. Ovaj put imamo tri slike. Prva neka demonstrira dva trokutasta valna pulsa koji su zrcalno simetrični s obzirom na y-os u trenutku t=0(Slika 6.3), druga dva trokutasta valna puls koji su zrcalno simetrični s obzirom na x-os isto u trenutku t=0(Slika 6.4), a treća dva trokutasta valna pulsa koji su zrcalno simetrični s obzirom na y-os i još k tome invertirani u t=0(Slika 6.5).

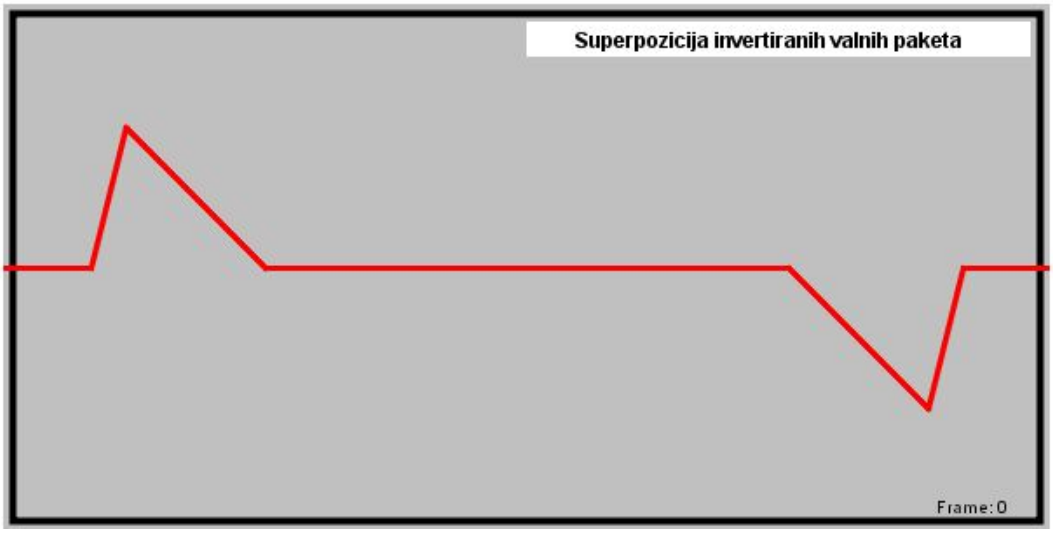
Tražimo od njih neka prvo promatrajući uoče koji su trokuti simetrični, a koji nisu. Nakon toga tražiti od njih da zapišu svoje pretpostavke što će se dogoditi prilikom superponiranja valova na slikama. Tj. gdje će se superponiranjem valovi poništiti, a gdje neće? Nakon toga otvoriti raspravu o njihovim pretpostavkama.



Slika 6.3 Dva trokutasta valna pulsa koji su zrcalno simetrični s obzirom na y-os



Slika 6.4 Dva trokutasta valna pulsa koji su zrcalno simetrični s obzirom na x-os



Slika 6.5 Dva trokutasta valna pulsa koji su zrcalno simetrični s obzirom na y-os i još k tome invertirani

Provjera pretpostavki korištenjem appleta

Na svakom računalu se nalazi applet koji učenici trebaju učitati. Objasniti učenicima kako se pokreće simulacija, kako ostvariti prikaz u trenutku t=0 i gdje se unose parametri za valne pakete.

Zadaci za učenike:

Svaki od učenika neka pokuša unosom parametra dobiti valne pakete kao na prvoj slici. Nakon što je uspio dobiti položaj valnih paketa koji odgovara onima na slici neka pokrene simulaciju.

Pitanja do čijih odgovora učenici trebaju doći korištenjem appleta:

- Što se dešava sa amplitudom valnih pulseva kada se susretnu?
- Opišite kako izgleda rezultat superpozicije.
- Kakav je oblik individualnog valnog pulsa prije i nakon susreta?
- Da li se mijenja brzina individualnog valnog pulsa nakon susreta?

Odgovore na gore navedena pitanja učenici trebaju zapisati u bilježnicu.

Očekivani odgovori na gornja pitanja:

- Amplituda rezultantnog vala se udvostruči ako promatramo superopoziciju dvaju valnih paketa koji su simetrični i koji nisu invertirani.
- Oblik individualnog valnog pulsa se ne mijenja prije i nakon susreta.
- Brzina individualnog valnog pulsa se ne mijenja.

Gore navedeni postupak učenici trebaju ponoviti i za ostale dvije slike. Na taj će način učenici korištenjem appleta sami doći do zaključka što se dešava prilikom susreta različitih vrsta valnih pulseva.

Korištene aplikacije

Pri izradi ovog diplomskog koristio sam sljedeći software:

- MS WORD 2000 -tekstualni dio i slike
- IrfanView 3.98 -obrada slika
- JDK 5.0 i JRE 5.0
- Eclipse SDK 3.1.2 -razvojno okruženje za Java aplikacije
- MS Excel 2000 -obrada i crtanje grafova
- Mathematica 5.0 -obrada i crtanje grafova, provjera ispravnosti nacrtanih grafova

Svi gore navedeni programi su izvršavani pod operativnim sustavom Windows XP Proffesional Version 2002 SP2.

Zaključak

Kada sam započeo s izradom diplomskog rada nisam bio posve siguran da li će se superponiranjem invertirani valni paketi poništiti. Korištenjem appleta koji simulira superponiranje i gibanje valnih paketa uvjerio sam se u neke činjenice. Rezultat superpozicije dvaju trokutastih valnih paketa koji su zrcalno simetrični s obzirom na y-os i k tome još invertirani neće biti destruktivna interferencija. Trokutasti valni paketi će se poništiti samo u slučaju ako promatramo superpoziciju valnih paketa koji su zrcalno simetrični s obzirom na x-os. Također sa appleta se može zaključiti ponešto i o samim valnim paketima, vidimo da se individualni valni paketi ponašaju neovisno jedan o drugome, a nakon susreta, brzina, oblik i veličina valnog pulsa se ne mijenja.

Takve činjenice bi bilo vrlo teško uočiti okom ukoliko bi pokušali eksperimentalno prikazati ovaj fizikalni problem. Tu bi još naišli na problem kako eksperimentalno na užetu ili žici ostvariti prikaz trokutastog valnog pulsa, gotovo nemoguće. Sve su to nedostaci eksperimentalnog prikaza našeg fizikalnog problema. Izradom appleta možemo vrlo lako doskočiti tim nedostacima i na taj način spriječiti da učenici steknu kriva uvjerenja o ovoj tematici.

Iako na Internetu ima podosta primjera interaktivnih animacija koje objašnjavaju fizikalne probleme one se premalo koriste u školama. Stoga ovim diplomskim radom želim ukazati na to da korištenjem interaktivne animacije za prikaz fizikalnog problema tamo gdje je eksperimentalno to nemoguće, uvelike pomažemo učenicima da steknu pravilna konceptualna shvaćanja o nekoj tematici iz fizike.

Literatura

Fizika

1. Waves berkley physics course –volume 3, Frank S.Crawford, Jr.,Berkley, California, 1968.
2. Fizika odabrana poglavlja, Horvat Dubravko, Hinus, Zagreb, 1999.
3. Valovi i optika, Henč-Bartolić Višnja, Kulišić Petar, Školska knjiga, Zagreb, 1991.
4. Predavanje iz Osnova Fizike 3, Dulčić Antonije, Zagreb, 2003/2004.

Informatika

5. Learning Java Second Edition, Patrick Niemeyer, Jonathan Knudsen, O'Reilly, Sebastopol, 2002.
6. The Java Tutorial Second Edition, Mary Campione, Kathy Walrath, Sun Microsystem, Palo Alto,1998.
7. JAVA, Tihomir Čukman, Alfej, Zagreb, 1997.
8. Izrada interaktivne animacije u Java-i za simuliranje gibanja nabijene čestice u homogenom magnetskom polju, Marin Kosović, Zagreb, 2005.
9. Računarski praktikum 2, Goran Muić, PMF –Matematički odsjek, Zagreb, 2005., http://www.math.hr/nastava/rp2/
10. Java i objektno orijentirano programiranje, Predrag Brođanac, Zagreb, 2005.
11. Jack's Page : Basic Physics and Optics, Jack Ord, http://www.kw.igs.net/~jackord/
12. NTNU Java-Superpositon of pulses, Fu-Kwun Hwang http://www.phy.ntnu.edu.tw/~hwang/wave/impulse.html
13. Java Applets on Physics, Walter Fendt http://www.walter-fendt.de/ph14e
14. The Language Environment white paper, James Gosling, Henry McGilton,1996., http://java.sun.com/docs/white/langenv/

Prilog

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.applet.*;

public class superpozicija extends Applet implements Runnable,ActionListener{


Dimension offDimension;
Image offImage;
Image imagez;
Image imagep;
Graphics offGraphics;

Label tekst1,tekst2,tekst_a,tekst_b,tekst_d;
Label time1,time2,time3,ime,superpo;
Panel trokut,glavni;
Button bstart,bstop,bzat0,bsuperpo;

Choice a_prvi;
Choice b_prvi,d_prvi;
Choice a_drugi,b_drugi,d_drugi;

Thread animatorThread;

int i,j;
int pom;

int kordxp1,kordxp2,kordxp3;
int kordxz1,kordxz2,kordxz3;

int kordxp11,kordxp22,kordxp33;
int kordxz11,kordxz22,kordxz33;

int [] x=new int[601];
int [] y=new int[601];

int [] x_pom=new int[601];
int [] y_pom=new int[601];

int [] x_1=new int[601];
int [] y_1=new int[601];

int [] x_2=new int[601];
int [] y_2=new int[601];
```

```
int t=0;

int x_1_0;
int x_2_0;

int v_1;
int v_2;

int a_1,b_1,d_1_1,d_1_2,a_2,b_2,d_2_1,d_2_2;
int delay,fps;

boolean frozen = false;
boolean pocetni = false;
boolean setFrameZero = false;
boolean superpozicija = false;

String str;

public void init()
{

//System.out.println("init()");
fps=24;
delay=1000/fps;
x_1_0=50;
v_1=2;

x_2_0=450;
v_2=-2;

kordxp1 = 272;
kordxz1 = 290;

kordxp2 = 292;
kordxz2 = 270;

kordxp3 = 242;
kordxz3 = 260;

kordxp11 = 312;
kordxz11 = 250;

kordxp22 = 332;
kordxz22 = 230;

kordxp33 = 282;
kordxz33 = 220;

/******Ucitavanje slika*****/
```

```
imagez = getImage(getCodeBase(),"zelena.gif");
imagep = getImage(getCodeBase(),"plava.gif");
/******Ucitavanje slika ukoliko se applet nalazi na mreži, posluzitelj dominis.phy.hr*****/
/*try {
imagez = getImage(new URL("http://dominis.phy.hr/~miland/diplomski/zelena.gif"));
} catch (MalformedURLException e) {
e.printStackTrace();}
try {
imagep = getImage(new URL("http://dominis.phy.hr/~miland/diplomski/plava.gif"));
} catch (MalformedURLException e) {
e.printStackTrace();
}*/
/******Definiranje razmjesataja za awt kontejnere*********/
setLayout(null);
/******Definiranje panela trokut*********/
trokut = new Panel();
trokut.setSize(600,100);
trokut.setLocation(0,300);
trokut.setBackground(Color.BLACK);
add(trokut);
/******Definiranje panela glavni*********/
glavni = new Panel();
glavni.setSize(600,50);
glavni.setLocation(0,400);
glavni.setBackground(Color.LIGHT_GRAY);
add(glavni);


/******Definiranje razmjestaja za awt komponente na panelu trokut*******/
trokut.setLayout(null);

//Definiranje i razmjestaj komponenti Label za prvi valni paket
tekst1 = new Label("Unesite parametre za prvi valni paket:");
tekst1.setForeground(Color.WHITE);
tekst1.setFont(new Font("Arial",Font.BOLD,12));
tekst1.setLocation(10,10);
tekst1.setSize(250,20);
trokut.add(tekst1);

tekst_a = new Label("Parametar a:");
tekst_a.setForeground(Color.WHITE);
tekst_a.setFont(new Font("Arial",Font.BOLD,11));
tekst_a.setLocation(35,30);
tekst_a.setSize(90,20);
trokut.add(tekst_a);

tekst_b = new Label("Parametar b:");
tekst_b.setForeground(Color.WHITE);
tekst_b.setFont(new Font("Arial",Font.BOLD,11));
tekst_b.setLocation(35,50);
```

```
tekst_b.setSize(90,20);
trokut.add(tekst_b);

tekst_d = new Label("Parametar d:");
tekst_d.setForeground(Color.WHITE);
tekst_d.setFont(new Font("Arial",Font.BOLD,11));
tekst_d.setLocation(35,70);
tekst_d.setSize(90,20);
trokut.add(tekst_d);

//Definiranje i razmjestaj komponenti Label za drugi valni paket
tekst2 = new Label("Unesite parametre za drugi valni paket:");
tekst2.setForeground(Color.WHITE);
tekst2.setFont(new Font("Arial",Font.BOLD,12));
tekst2.setLocation(360,10);
tekst2.setSize(250,20);
trokut.add(tekst2);

tekst_a = new Label("Parametar a:");
tekst_a.setForeground(Color.WHITE);
tekst_a.setFont(new Font("Arial",Font.BOLD,11));
tekst_a.setLocation(385,30);
tekst_a.setSize(90,20);
trokut.add(tekst_a);

tekst_b = new Label("Parametar b:");
tekst_b.setForeground(Color.WHITE);
tekst_b.setFont(new Font("Arial",Font.BOLD,11));
tekst_b.setLocation(385,50);
tekst_b.setSize(90,20);
trokut.add(tekst_b);

tekst_d = new Label("Parametar d:");
tekst_d.setForeground(Color.WHITE);
tekst_d.setFont(new Font("Arial",Font.BOLD,11));
tekst_d.setLocation(385,70);
tekst_d.setSize(90,20);
trokut.add(tekst_d);

//Definiranje i razmjestaj komponenti Choice za prvi valni paket
a_prvi = new Choice();
a_prvi.setLocation(125,30);
a_prvi.setSize(70,20);
trokut.add(a_prvi);

a_prvi.addItem("-3");
a_prvi.addItem("-2");
a_prvi.addItem("-4");

b_prvi = new Choice();
```

```
b_prvi.setLocation(125,50);
b_prvi.setSize(70,20);
trokut.add(b_prvi);

b_prvi.addItem("3");
b_prvi.addItem("1");
b_prvi.addItem("1");

d_prvi = new Choice();
d_prvi.setLocation(125,70);
d_prvi.setSize(70,20);
trokut.add(d_prvi);

d_prvi.addItem("20");
d_prvi.addItem("40");
d_prvi.addItem("20");


//Definiranje i razmjestaj komponenti Choice za drugi valni paket
a_drugi = new Choice();
a_drugi.setLocation(480,30);
a_drugi.setSize(70,20);
trokut.add(a_drugi);

a_drugi.addItem("-3");
a_drugi.addItem("2");
a_drugi.addItem("1");


b_drugi = new Choice();
b_drugi.setLocation(480,50);
b_drugi.setSize(70,20);
trokut.add(b_drugi);

b_drugi.addItem("3");
b_drugi.addItem("-1");
b_drugi.addItem("-4");


d_drugi = new Choice();
d_drugi.setLocation(480,70);
d_drugi.setSize(70,20);
trokut.add(d_drugi);

d_drugi.addItem("20");
d_drugi.addItem("40");
d_drugi.addItem("80");


/******Definiranje razmjestaja za awt komponente na panelu glavni*******/
```

```
glavni.setLayout(null);

bzat0 = new Button("Za t=0s");
bzat0.setSize(90,20);
bzat0.setLocation(35,15);
bzat0.addActionListener(this);
glavni.add(bzat0);

bstart = new Button("Start");
bstart.setSize(90,20);
bstart.setLocation(130,15);
bstart.addActionListener(this);
glavni.add(bstart);

bstop = new Button("Stop");
bstop.setSize(90,20);
bstop.setLocation(225,15);
bstop.addActionListener(this);
glavni.add(bstop);

bsuperpo = new Button("Superpozicija");
bsuperpo.setSize(90,20);
bsuperpo.setLocation(320,15);
bsuperpo.addActionListener(this);
glavni.add(bsuperpo);


ime = new Label("M.Đurić,2006");
ime.setForeground(Color.BLACK);
ime.setFont(new Font("Arial",Font.BOLD,10));
ime.setSize(70,10);
ime.setLocation(520,38);
glavni.add(ime);
}

public void start() {
/****Kreiranje dretve pozivom konstruktora klase Thread i startanje animacijske
dretve*****/
animatorThread=new Thread(this);
animatorThread.start();
}

public void run()
{

Thread.currentThread().setPriority(Thread.MIN_PRIORITY);

long startTime = System.currentTimeMillis();

while(Thread.currentThread() == animatorThread)
```

```
{

//System.out.println("run()");
repaint();
try
{
startTime=startTime+delay;
Thread.sleep(Math.max(0,startTime-System.currentTimeMillis()));

} catch (InterruptedException e1){stop();}

}

}
/*****interface klase ActionListener*****/
public void actionPerformed(ActionEvent e)
{
/*****Parametar e metode actionPerformed ima sve informacije o dogadajima****/
/*****Parametar e zna nad kojim elementom je događaj izvršen******/
/*****Metoda getSource() vraca ime objekta nad kojim je izvrsena radnja****/
if(e.getSource() == bsuperpo)
{
pocetni=true;
frozen = false;
superpozicija = true;
t=100;
pozovi_vrijednosti();
racunaj();
repaint();
kordxp11 = 312;
kordxz11 = 250;

kordxp22 = 332;
kordxz22 = 230;

kordxp33 = 282;
kordxz33 = 220;

}

if(e.getSource() == bzat0)
{
pocetni=true;
frozen = false;
setFrameZero = true;
superpozicija = false;
t=0;
pozovi_vrijednosti();
racunaj();
repaint();
```

```
/***********************/
kordxp1 = 272;
kordxz1 = 290;

kordxp2 = 292;
kordxz2 = 270;

kordxp3 = 242;
kordxz3 = 260;

/***********************/
kordxp11 = 312;
kordxz11 = 250;

kordxp22 = 332;
kordxz22 = 230;

kordxp33 = 282;
kordxz33 = 220;

}

if(e.getSource() == bstart)
{
pocetni=true;
frozen = true;
superpozicija = false;
}

if(e.getSource() == bstop)
{
pocetni=true;
frozen = false;
superpozicija = false;
}

}
/*****Metoda koja sadrzi metodu getSelectedItem() klase Choice*****/
/*****Metoda getSelectedItem() iz izbora vraca selektiranu vrijednost kao string****/
public void pozovi_vrijednosti()
{
//System.out.println("pozovi vrijednost");
a_1= Integer.parseInt((String) a_prvi.getSelectedItem());
//System.out.println(a_1);
b_1=  Integer.parseInt((String) b_prvi.getSelectedItem());
//System.out.println(b_1);
d_1_1= Integer.parseInt((String) d_prvi.getSelectedItem());
//System.out.println(d_1_1);

a_2= Integer.parseInt((String) a_drugi.getSelectedItem());
```

```
//System.out.println(a_2);
b_2= Integer.parseInt((String) b_drugi.getSelectedItem());
//System.out.println(b_2);
d_2_1= Integer.parseInt((String) d_drugi.getSelectedItem());
//System.out.println(d_2_1);
}
/*****Metoda za izracun koordinata trokutastih valnih paketa*****/
private void racunaj()
{
//System.out.println("racunaj");
//System.out.println(t);
//System.out.println(frozen);
d_1_2=-(a_1/b_1)*d_1_1;
d_2_2=-(a_2/b_2)*d_2_1;
//System.out.println(d_1_2);

if(d_2_2 == 0)
d_2_2=d_1_1;

//System.out.println(d_2_2);

pom = 0;

for(i=0;i<601;i++)
{
x[i]=pom;
pom=pom+1;
}
/****Glavni dio programa, izracun koordinata valnih paketa*****/
for(i=0;i<601;i++)
{
if(((x[i]-v_1*t-x_1_0)>0)&&((x[i]-v_1*t-x_1_0)<(d_1_1+d_1_2)))
if((x[i]-v_1*t-x_1_0)<d_1_1)
y_1[i]=(x[i]-v_1*t-x_1_0)*a_1;
else
y_1[i]=a_1*d_1_1+(x[i]-v_1*t-x_1_0-d_1_1)*b_1;
else
y_1[i]=0;
}

for(i=0;i<601;i++)
{
if(((x[i]-v_2*t-x_2_0)>0)&&((x[i]-v_2*t-x_2_0)<(d_2_1+d_2_2)))
if((x[i]-v_2*t-x_2_0)<d_2_1)
y_2[i]=(x[i]-v_2*t-x_2_0)*a_2;
else
y_2[i]=a_2*d_2_1+(x[i]-v_2*t-x_2_0-d_2_1)*b_2;
else
y_2[i]=0;
}
```

```
/****Izracun superpozicije valnih paketa(zbroj individulanih koordinata svakog paketa)*****/
for(i=0;i<601;i++)
{
y[i]=y_1[i]+y_2[i];
//System.out.println(y[i]);
}
/*****Pomak koordinata za 150 prema dolje zbog koordinatnog sustava u Javi******/
for(i=0;i<600;i++)
{
y[i]=y[i]+150;
//System.out.println(y[i]);
}

for(i=0;i<600;i++)
{
y_1[i]=y_1[i]+150;
y_2[i]=y_2[i]+150;
//System.out.println(y_1[i]);
//System.out.println(y_2[i]);
}
}

public void paint(Graphics g)
{
update(g);
}

public void update(Graphics g)
{
Dimension d = getSize();
Graphics2D offGraphics2D;


/******Kreiranje grafickog objekta offGraphics tipa Graphics******/
/******offGraphics definira podrucje za crtanje za neprikazanu grafiku*******/
/******offImage objekt klase Image, slika u koju se učitava neprikazana grafika*******/
/******offDimnesion definira dimenzije slike offImage*******/
if ( (offGraphics == null) || (d.width != offDimension.width) || (d.height != offDimension.height) )
{
offDimension = d;
offImage = createImage(d.width, d.height);
offGraphics = offImage.getGraphics();
}
/******Pretvorba grafickog objekta offGraphics u 2D grafiku********/
/******Dalje je nas graficki objekt offGraphics2D*******/
offGraphics2D = (Graphics2D) offGraphics ;
offGraphics2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING,RenderingHints.VALUE_ANTIALIAS_ON);
```

```
if(frozen==true)
{
t=t+1;
pozovi_vrijednosti();
racunaj();
//System.out.println("crtam");
}

//System.out.println("paint()");

/*******Zapocinje se sa crtanjem grafike na podrucje definirano grafickim objektom
offGraphics2D*******/
offGraphics2D.setColor(Color.LIGHT_GRAY);
offGraphics2D.fillRect(0, 0, 600,300);

offGraphics2D.setColor(Color.BLACK);
offGraphics2D.setStroke(new BasicStroke(4));
offGraphics2D.drawRect(5, 5, 589,289);

offGraphics2D.setColor(Color.WHITE);
offGraphics2D.fillRect(300, 10, 289,20);

offGraphics2D.setColor(Color.BLACK);
offGraphics2D.setFont(new Font("Arial",Font.BOLD,12));
offGraphics2D.drawString("Superpozicija putujućih valnih poremećaja",328,25);


if(a_1 == -2 && b_1 == 1 && a_2 == 2 && b_2 == -1 && d_1_1 == 40 && d_2_1 == 40)
{
offGraphics2D.setColor(Color.WHITE);
offGraphics2D.fillRect(300, 10, 289,20);

offGraphics2D.setColor(Color.BLACK);
offGraphics2D.setFont(new Font("Arial",Font.BOLD,12));
offGraphics2D.drawString("Superpozicija invertiranih valnih paketa",328,25);
}

if(a_1 == -4 && b_1 == 1 && a_2 == 1 && b_2 == -4 && d_1_1 == 20 && d_2_1 == 80)
{
offGraphics2D.setColor(Color.WHITE);
offGraphics2D.fillRect(300, 10, 289,20);

offGraphics2D.setColor(Color.BLACK);
offGraphics2D.setFont(new Font("Arial",Font.BOLD,12));
offGraphics2D.drawString("Superpozicija invertiranih valnih paketa",328,25);
}

if(pocetni==false)
{
```

```
offGraphics2D.setColor(Color.red);
offGraphics2D.setStroke(new BasicStroke(3));
offGraphics2D.drawLine(8,150,590,150);
}

if(setFrameZero == true)
{
str ="0";

offGraphics2D.setColor(Color.LIGHT_GRAY);
offGraphics2D.fillRect(500,275,92,17);

offGraphics2D.setColor(Color.BLACK);
offGraphics2D.setFont(new Font("Arial",Font.PLAIN,10));
offGraphics2D.drawString("Frame:",535,290);

offGraphics2D.setColor(Color.BLACK);
offGraphics2D.setFont(new Font("Arial",Font.PLAIN,10));
offGraphics2D.drawString(str,570,290);
}


str = Integer.toString(t);

offGraphics2D.setColor(Color.LIGHT_GRAY);
offGraphics2D.fillRect(500,275,92,17);

offGraphics2D.setColor(Color.BLACK);
offGraphics2D.setFont(new Font("Arial",Font.PLAIN,10));
offGraphics2D.drawString("Frame:",535,290);

offGraphics2D.setColor(Color.BLACK);
offGraphics2D.setFont(new Font("Arial",Font.PLAIN,10));
offGraphics2D.drawString(str,570,290);

if(t > 70 && t < 130)
{
offGraphics2D.setStroke(new BasicStroke(3));
offGraphics2D.setColor(Color.blue);

for(i=0;i<599;i++)
offGraphics2D.drawLine(x[i],y_1[i],x[i+1],y_1[i+1]);

offGraphics2D.setStroke(new BasicStroke(3));
offGraphics2D.setColor(Color.green);

for(i=0;i<599;i++)
offGraphics2D.drawLine(x[i],y_2[i],x[i+1],y_2[i+1]);

for(i=0;i<599;i++)
```

```
offGraphics2D.drawLine(x[i],y_2[i],x[i+1],y_2[i+1]);
}


if(superpozicija == true && t == 100 && a_1 == -4 && b_1 == 1 && a_2 == 1 && b_2 == -
4 && d_1_1 == 20 && d_2_1 == 80)
{

offGraphics2D.drawImage(imagez,250,185,this);
offGraphics2D.drawImage(imagep,310,105,this);
}

if(superpozicija == true && t == 100 && a_1 == -2 && b_1 == 1 && a_2 == 2 && b_2 == -
1 && d_1_1 == 40 && d_2_1 == 40)
{
offGraphics2D.drawImage(imagez,230,185,this);
offGraphics2D.drawImage(imagep,330,105,this);
}

if(superpozicija == true && t == 100 && a_1 == -3 && b_1 == 3 && a_2 == -3 && b_2 ==
3 && d_1_1 == 20 && d_2_1 == 20)
{
offGraphics2D.drawImage(imagez,220,115,this);
offGraphics2D.drawImage(imagep,280,115,this);
}


if(frozen == true && superpozicija == false && t > 80 && t< 100 && a_1 == -4 && b_1 ==
1 && a_2 == 1 && b_2 == -4 && d_1_1 == 20 && d_2_1 == 80)
{

offGraphics2D.drawImage(imagez,kordxz1,185,this);
kordxz1=kordxz1-2;
offGraphics2D.drawImage(imagep,kordxp1,105,this);
kordxp1=kordxp1+2;
}

if(frozen == true && superpozicija == false && t > 80 && t< 100 && a_1 == -2 && b_1 ==
1 && a_2 == 2 && b_2 == -1 && d_1_1 == 40 && d_2_1 == 40)
{

offGraphics2D.drawImage(imagez,kordxz2,185,this);
kordxz2=kordxz2-2;
offGraphics2D.drawImage(imagep,kordxp2,105,this);
kordxp2=kordxp2+2;
}

if(frozen == true && superpozicija == false && t > 80 && t< 100 && a_1 == -3 && b_1 ==
3 && a_2 == -3 && b_2 == 3 && d_1_1 == 20 && d_2_1 == 20)
{
```

```
offGraphics2D.drawImage(imagez,kordxz3,115,this);
kordxz3=kordxz3-2;
offGraphics2D.drawImage(imagep,kordxp3,115,this);
kordxp3=kordxp3+2;
}

if(frozen == true && superpozicija == false && t > 100 && t < 130 && a_1 == -4 && b_1
== 1 && a_2 == 1 && b_2 == -4 && d_1_1 == 20 && d_2_1 == 80)
{

offGraphics2D.drawImage(imagez,kordxz11,185,this);
kordxz11=kordxz11-2;
offGraphics2D.drawImage(imagep,kordxp11,105,this);
kordxp11=kordxp11+2;
}

if(frozen == true && superpozicija == false && t > 100 && t < 130 && a_1 == -2 && b_1
== 1 && a_2 == 2 && b_2 == -1 && d_1_1 == 40 && d_2_1 == 40)
{

offGraphics2D.drawImage(imagez,kordxz22,185,this);
kordxz22=kordxz22-2;
offGraphics2D.drawImage(imagep,kordxp22,105,this);
kordxp22=kordxp22+2;
}

if(frozen == true && superpozicija == false && t > 100 && t < 130 && a_1 == -3 && b_1
== 3 && a_2 == -3 && b_2 == 3 && d_1_1 == 20 && d_2_1 == 20)
{

offGraphics2D.drawImage(imagez,kordxz33,115,this);
kordxz33=kordxz33-2;
offGraphics2D.drawImage(imagep,kordxp33,115,this);
kordxp33=kordxp33+2;

}


offGraphics2D.setStroke(new BasicStroke(3));
offGraphics2D.setColor(Color.red);
/*****Iscrtava rezultat superpozicije na podrucje definirano grafickim objektom
offGraphics2D*****/
for(i=0;i<599;i++)
offGraphics2D.drawLine(x[i],y[i],x[i+1],y[i+1]);
/****Prikaz slike nakon sto je sav izracun i crtanje obavljeno****/
g.drawImage(offImage,0,0,this);

}
```

```
public void stop()
{
//System.out.println("stop()");
animatorThread=null;
offImage=null;
offGraphics=null;
}


}
```