Samuel Rajamäki

FYSIIKAN SIMULAATIOITA JAVALLA

Tietotekniikan koulutusohjelma
Ohjelmistotuotannon suuntautumisvaihtoehto
2009

FYSIIKAN SIMULAATIOITA JAVALLA

Rajamäki, Samuel
Satakunnan ammattikorkeakoulu
Tekniikka ja merenkulku Rauma
Tietotekniikan koulutusohjelma
Toukokuu 2009
Ohjaaja: Mikko Javanainen
UDK: 681.3.06
Sivumäärä: 46
Avainsanat: ohjelmointi, fysiikka, Java

Tämän opinnäytetyön tavoitteena oli suunnitella ja toteuttaa tietokoneella toimivia fysiikan simulaatioita opetuskäyttöön. Tavoitteena oli suunnitella simulaatioista mahdollisimman helppokäyttöisiä ja simulaatioiden tulisi olla myös helposti saatavilla. Toinen tärkeä tavoite oli luoda edellytykset uusien simulaatioiden helpolle luomiselle.

Simulaatiot toteutettiin internetselaimessa toimivina Java-appletteina. Appleteille luotiin alusta, jonka päällä simulaatiot ajetaan. Simulaatiota toteutettiin kolme kappaletta: vino heittoliike, siniaaltojen summa ja liike koordinaatistossa. Simulaatioiden toiminta kuvataan opinnäytetyön raportissa. Raporttiin on liitetty myös ohjeet uuden simulaation luomisen sekä luotujen luokkien käyttöohjeet.

Simulaatiot toteutettiin Javalla sen web-ohjelmointimahdollisuuden vuoksi. Muita tärkeitä valintakriteereitä olivat Javan ympäristöriippumattomuus, oliopohjaisuus ja valmiit piirtorutiinit. Javalle löytyy ilmaisia kehitysympäristöjä, joista käyttöön valittiin Sun Microsystemsin ylläpitämä avoimeen lähdekoodiin perustuva NetBeans.

PHYSICS SIMULATIONS WITH JAVA

Rajamäki ,Samuel
Satakunta University of Applied Sciences
Technology and Maritime Management Rauma
Degree Programme in Information Technology
May 2009
Tutor: Mikko Javanainen
UDC: 681.3.06
Number of Pages: 46
Keywords: programming, physics, Java

The purpose of this Bachelor's thesis was to design and implement computer simulations for the teaching purposes of physics. The aim was to design simulations, which are easy to use and acces. Another important aim was to create conditions for easy creating of new simulations.

The simulations were implemented as Java applets, which operate with a Web browser. The platform for the simulations was created during the implementation. Three simulations were implemented: diagonal throw motion, the sum of sine curves and motion in a coordinate grid. The functionality of the simulations is described in the thesis. Instructions for creating new simulations and using the created classes are included in the appendix of the report.

The simulations were implemented with the Java programming language, because Java supports Web programming. Other important reasons for using Java for this project were cross-platform programming, supports for object-oriented programming, and a good set of drawing routines. There are several free development environments for Java. One of them is Netbeans, which is developed by Sun Microsystems and the program used in this project.

SISÄLLYS

TERMILUETTELO

Vapaan lähdekoodin ohjelmisto	Tietokoneohjelma, jota käyttäjät ja kehittäjät voivat muokata ja levittää vapaasti.
IDE	Tietokoneohjelma joka tarjoaa perustyökalut ohjelmistojen kehittämiseen.
Olio-ohjelmointi	Ohjelmoinnin lähestymistapa, jossa ohjelmointiongelmien ratkaisut jäsennetään olioiden yhteistoimintana.
Luokka	Ohjelmakokonaisuus joka sisältää metodeja ja kenttiä.
Olio	Luokan ilmentymä, olio sisältää kaikki luokan sisältämät metodit ja kentät.
Paketti	Kokoelma samankaltaisia luokkia.
Metodi	Luokan sisältämä toiminto.
Kenttä	Luokan sisältämä tietorakenne.
Perintä	Luokka saa luokan yliluokkiin liitetyt piirteet ilman erillistä määrittelyä.
Aliluokka	Luokka, joka periytymishierarkiassa on jonkun muun luokan alapuolella.
Yliluokka	Luokka, joka on perintähierarkiassa toisen luokan yläpuolella.
Konstruktori	Luokan sisältämä metodi joka suoritetaan luokan luonnin yhteydessä
AWT	Abstract Windowing Toolkit. Javan luokkajoukko käyttöliittymien rakentamista varten
Swing	Laajennos AWT-pakettiin.
Säije	Moniajon toteutusväline Javassa.

(Kosonen & Peltomäki & Silander 2005, 247)
(Helsingin yliopisto 2000.)

1 JOHDANTO

Tietokonesimulaatiota käytetään opetuksen tukena laajalti, aina monimutkaisista lentosimulaattoreista yksinkertaisiin fysiikan simulaatioihin. Simulaatiot mallintavat jonkin reaalimaailman asian tai ilmiön. Simulaatio antaa oppijalle mahdollisuuden olla vuorovaikutuksessa tarkasteltavan sisällön kanssa. Simulaatioiden avulla pystytään luomaan tilanteita joita ei muuten olisi mahdollista järjestää esimerkiksi niiden harvinaisuuden, kalleuden, vaarallisuuden tai monimutkaisuuden takia. (Kuopion yliopisto 2009.)

Simulaatiot soveltuvat erityisen hyvin fysiikan opetukseen, sillä niiden avulla havainnollistetaan fysiikan ilmiöitä, joita ei pystytä näkemään, tai jos koetilannetta olisi hankala järjestää kouluolosuhteissa. Esimerkkeinä tällaisista käyvät tässä opinnäytetyössä toteutetut simulaatiot: lentoradan simulointi tyhjiössä, seisovan aaltoliikkeen muodostuminen kahdesta siniaallosta ja tasaisesti kiihtyvän liikkeen havainnollistaminen koordinaatistojen avulla.

Tämän opinnäytetyön aikana toteutettavat simulaatiot ovat hyvin yksinkertaisia, ja vuorovaikutteisuus oppijan kanssa muodostuu lähinnä alkuparametrien syötöstä ohjelmalle. Tästä syystä voimme käyttää simulaatiolle myös nimitystä animaatio. Toteutettavien simulaatioiden yleinen toimintakaava on seuraava: simulaatiolle syötetään alkuarvot, ilmiötä havainnollistetaan animaation avulla, ja lopuksi simulaation pysähtyessä tai kun käyttäjä pysäyttää simulaation, näytetään loppuarvot.

Ennen simulaatioiden toteuttamista etsittiin vastaavanlaisia animaatiota internetistä. Sivustoja löytyi useita erilaisia (katso liite 1). Lisäksi löytyi valmiita ympäristöjä simulaatioiden toteuttamiseen. Valmiin ympäristön käyttö simulaatiota toteutettaessa olisi ollut tehokkaampaa ja haasteet olisivat painottuneet enemmän fysiikan kaavojen

soveltamiseen. Koska tämä projekti on tietotekniikan koulutusohjelmaan liittyvä opinnäytetyö, päätettiin simulaatiot tehdä alusta loppuun itse, jolloin haasteet painottuvat enemmän ohjelmistoteknisiin ongelmiin.

2 TYÖKALUT

Tässä luvussa käydään läpi projektissa käytetyt työkalut. Kaikki käytetyt ohjelmistot ovat vapaasti saatavilla ja ne perustuvat ainakin osittain vapaaseen lähdekoodiin.

2.1 Java–toteutuksessa käytetty ohjelmointikieli

Java on Sun Microsystemsin kehittämä tietoverkkoihin suunnattu järjestelmäriippumaton olio-ohjelmointi kieli, joka on kehitetty C++ pohjalta. Javan kehitys aloitettiin vuonna 1991, ja se yleistyi internetin suosion kasvun myötä. Javan avulla pystyttiin luomaan dynaamista sisältöä web-sivuille, mikä Javan alkuaikoina nähtiin sen tärkeimmäksi käyttöalueeksi. Nykyään Javaa käytetään moniin eri tarkoituksiin, aina pienistä kännykässä toimivista sovelmista mittaviin palvelinohjelmistoihin. Javan tyypillisiä piirteitä ovat seuraavat:

- puhdas oliopohjaisuus
- vahva tyypitys
- C++:aa muistuttava syntaksi
- automaattinen roskien keruu
- WWW-ohjelmointi tuki
- tulkattavuus.

(Vesterholm & Kyppö 2001, 17 ; Deitel & Deitel, 2005, 9.)

Puhdas oliopohjaisuus tarkoittaa käytännössä sitä, että kaikki suoritettava ohjelmakoodi sijaitsee luokissa eikä perinteinen hierarkkinen ohjelmointi ole mahdollista. Aloittelevasta ohjelmoijasta tämä saattaa tuntua aluksi hankalalta, mutta vähänkään laajempien kokonaisuuksien hallinnassa oliopohjaisuudesta on huomattava etu. Java sisältää hyvin paljon laajoja valmiita luokkakirjastoja. Tämä oli yksi kriteeri ohjelmointikieltä valittaessa tähän projektiin, sillä Javassa on simulaatioiden tekoon tarvittavat tehokkaat laskenta- ja piirto-kirjastot valmiina. (Vesterholm & Kyppö 2001, 19.)

Vahva tyypitys on ohjelmointikielissä käytettävä ominaisuus, joka määrittelee, että jokaisella muuttujalla on tyyppi ja muuttujaan voi sijoittaa ainoastaan sen tyypin mukaisia arvoja. Vahva tyypitys aiheuttaa tyyppivirheiden löytymisen jo käännöksen aikana.

Automaattinen roskien keruu helpottaa ohjelmointia, kun ohjelmoijan ei tarvitse huolehtia olioiden varaaman muistin vapauttamisesta. Automaattisen roskienkeruun avulla kierretään monet muistinhallintaan liittyvät sudenkuopat.

WWW-ohjelmointituki on toteutettu Javassa WWW-selaimeen asennettavalla liitännäisellä, joka mahdollistaa applettien ajon suoraan WWW-selaimella. Liitännäinen on saatavilla useimpiin selaimiin ilmaseksi Sun Microsystemsin internetsivuilta. Javalla tehtyjä sovelluksia, jotka toimivat WWW-sivulla HTML-objektina, kutsutaan appleteiksi tai sovelmiksi. Appletit soveltuvat erityisen hyvin projektissa tehtävien simulaatiotyyppisten ohjelmien toteuttamiseen, koska niiden tulisi olla helppokäyttöisiä ja helposti saatavilla. Käyttäjän ei tarvitse ladata tai asentaa mitään koneellensa, vaan appletteja voi ajaa missä tahansa tietokoneessa, jossa on internetselain Java-tuella. Appletit olivat yksi tärkeä kriteeri valittaessa projektiin ohjelmointikieltä.

2.1 JDK–aplettien kääntö ja virheiden etsintä

JDK (Java Development Kit) on Javan kehitysympäristö, joka sisältää kääntäjän, ajoympäristön (JVM) ja paljon muita ohjelmistokehitystä helpottavia työkaluja. JDK:sta on saatavilla kolme eri versiota: J2EE, J2SE, J2ME. J2EE on tarkoitettu laajojen yrityssovellusten tekemiseen, jotka toimivat usein palvelin-koneilla. J2SE:llä kehitetään lähinnä yksittäisille tietokoneille tarkoitettuja ohjelmia ja J2ME ympäristö on tarkoitettu kannettavien laitteiden ohjelmistojen kehittämiseen. Internetselaimessa toimivien sovelmien tekoon käytetään JDK:n J2SE versiota. JDK:n sisältämät ohjelmat toimivat komentoriviltä, mutta nykyään niitä käytetään harvoin ilman graafista kehitystyökalua. Tässä projektissa käytetään NetBeans nimistä ilmaista graafista kehitysympäristöä, se esitellään myöhemmässä luvussa.

2.2 JVM–applettien suorittaminen

JVM on osa Javan ajonaikaista ympäristöä, se ajaa Java ohjelman tavukoodin, joka on käännetty Java-lähdekoodista. JVM on ratkaiseva komponentti Java-ympäristössä, koska Java-ohjelmia on mahdollista ajaa niillä alustoilla, joille JVM on saatavilla. Yleisin käytössä oleva Javan virtuaalikone on Sun Microsystemsin kehittämä, mutta muiltakin julkaisijoilta on saatavilla vastaavia komponentteja. JVM-liitännäinen on saatavilla yleisimpiin WWW-selaimiin.

2.3 NetBeans–toteutuksessa käytetty ohjelmointiympäristö

NetBeans on Sun Microsystemsin kehittämä ohjelmointityökalu, jolla voi kirjoittaa, kääntää, etsiä virheitä ja ottaa käyttöön sovelluksia. Ohjelma on kirjoitettu Javalla, mutta se tukee mitä tahansa ohjelmointikieltä. NetBeans on menestyvä ja avoimeen lähdekoodiin perustuva projekti, jolla on erittäin laaja käyttäjäkunta, kasvava yhteisö. Sun Microsystems perusti NetBeans-projektin kesäkuussa 2000 ja toimii edelleen projektin pääsponsorina. (NetBeans 2008.)

3 APPLETIN RAKENNE

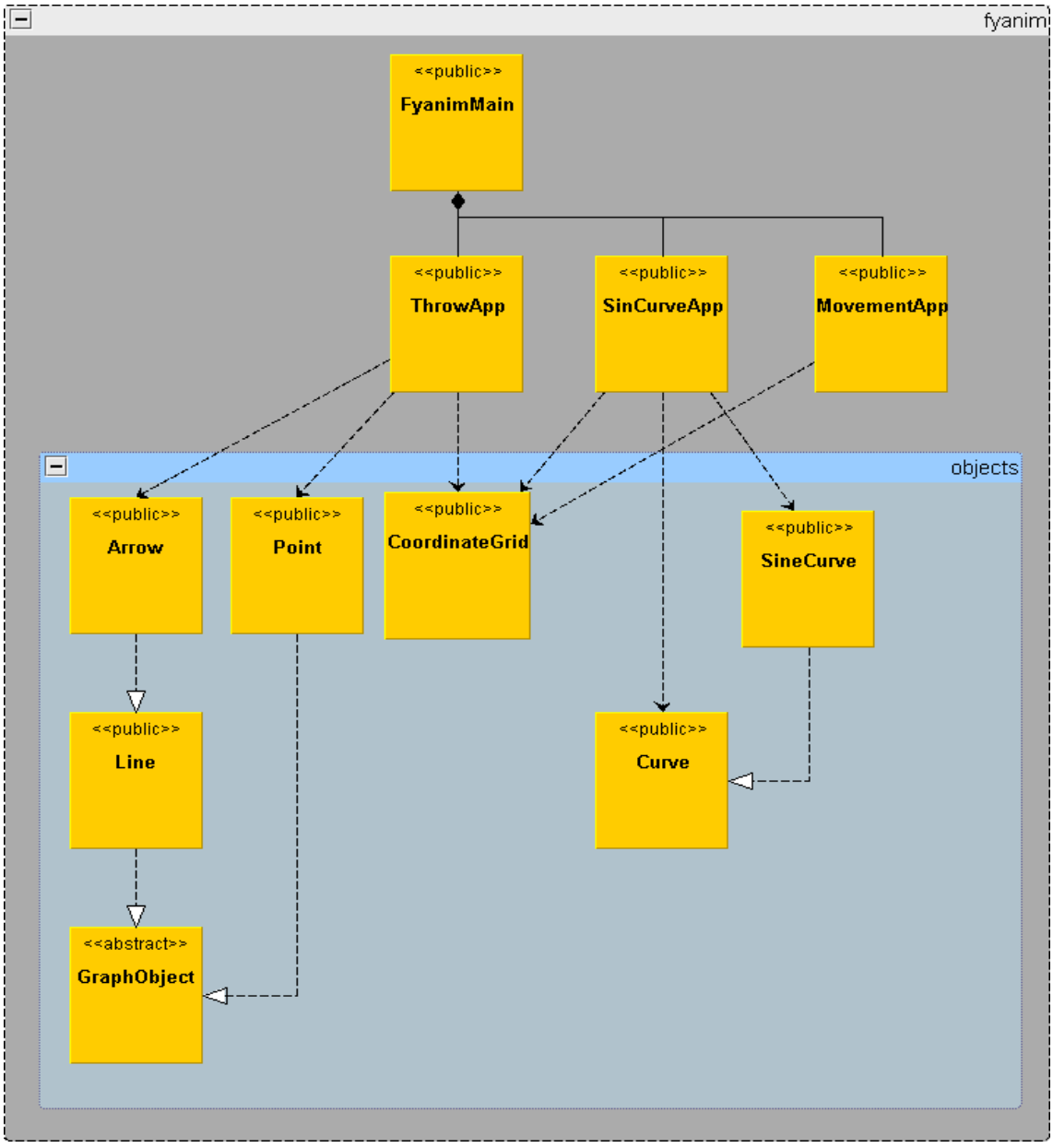
3.1 Yleistä

Simulaatioille tehtiin yksi yhteinen alusta, jossa ne toimivat. Vaihtoehtoinen toteutustapa olisi ollut tehdä jokaisesta simulaatiosta oma appletti. Yhteiseen alustaan päädyttiin mahdollisen jatkokehittämisen helpottamiseksi. On helpompaa luoda paneeli jo valmiina olevaan applettiin kuin luoda jokaista uutta simulaatiota varten oma appletti. Haittapuolena toteutettaessa simulaatiot samaan applettiin on HTML-linkin tekemisen hankaluus tiettyyn simulaation. Tämä ongelma ratkaistiin appletin HTML-parametrien avulla. Appletille toteutettiin parametri, jolla määritellään appletin avautuessa käynnistyvä simulaatio.

Appletin koko selaimen ikkunassa määriteltiin vakioksi käyttöliittymän suunnittelun helpottamiseksi. Simulaatiolle varattu tila määriteltiin sellaiseksi, että simulaatiota voidaan käyttää myös hieman normaalia pienemmillä näyttötarkkuuksilla. Simulaatio-appletin kooksi valittiin 800 x 600 pikseliä, joka mahtuu hyvin vielä 1024 x 768:n resoluutioon pystyvään näyttölaitteeseen. Tavallisesti koko näyttötila on käytettävissä, mutta pystysuunnassa lähes aina käyttettävissä olevasta alueesta tilaa vievät tehtäväpalkki ja internetselaimen valikot.

3.2 Rakenne

Java-ohjelman eri osat jaotellaan paketteihin ja alipaketteihin. Tässä projektissa käytetään yhtä pakettia ja sen alipakettia. Pääpaketin nimi on fyanim, ja se sisältää pääohjelman, simulaatiot ja components -alipaketin, joka sisältää simulaatioiden tarvitsemat komponentit. Kuvassa 1 on appletin luokkakaavio, josta käyvät ilmi appletin sisältämät paketit ja luokat sekä luokkien väliset yhteydet.



Kuva 1. Oliokaavio

3.3 Pääohjelma

Pääohjelma on fyanimMain.java-tiedostossa. Se sisältää luokan, joka on periytetty JApplet luokasta. Sitä käytetään kaikkien swingiä käyttävien applettien pohjana, ja luokka toimii simulaatioiden alustana. Simulaation vaihto tapahtuu valikon avulla. Pääohjelmaan on toteutettu HTML-parametri, jolla määritellään appletin käynnistyessä käynnistyvä simulaatio.

Paneelien vaihto on pääluokan vastuulla. Ennen paneelin lisäämistä on vanha paneeli poistettava ja tämä tapahtuu JApplett-luokasta löytyvän removeAll()-funktion avulla. Koska asettelunhallinta ei ole käytössä, on paneelille ennen sen lisäämistä asetettava koko ja paikka JPanel-aliluokasta löytyvän setBounds()-funktion avulla. Kooksi asetetaan 800 x 600 pikseliä, joka määriteltiin suunnitteluvaiheessa, ja paikaksi appletin vasen yläkulma, jonka koordinaatit ovat 0,0. Paneelin lisäämisen jälkeen se on vielä asetettava näkyväksi ja kutsuttava lisätyn paneelin revalidate()-funktiota, joka piirtää paneelin sisältämät komponentit uudelleen. Jos simulaatiossa on säie, joka on käynnistettävä heti appletin avautuessa. se kannattaa tehdä paneelin lisäämisen jälkeen.

3.4 Simulaatiopaneelit

Simulaatiopaneelin on perittävä JPanel-luokka, jotta se voidaan lisätä applettiin. Simulaation alkuarvot asetetaan luokan konstruktorissa, jossa luodaan myös tarvittavat oliot. Paneelin asetteluhallinnan voi ottaa pois päältä asettamalla se null-arvoon, koska appletti, jossa paneeli sijaitsee, on vakiokokoinen.

Useimmissa tapauksissa simulaatiot tarvitsevat käyttöönsä Runnable-rajapinnan, joka mahdollistaa säikeiden käytön. Säikeet helpottavat animointia. Luokan, joka käyttää Runnable-rajapintaa, on ylikirjoitettava ainakin rajapinnan julkinen funktio run(). Lisäksi rajapinta sisältää muita hyödyllisiä funktiota kuten start() ja stop(), jotka liittyvät säikeen käynnistyessä ja pysähtyessä tehtäviin toimenpiteisiin.

3.5 Komponenttiluokat

Objects-paketti sisältää komponentit, joita käytetään simulaatiossa. Komponentit ovat graafisia objekteja, kuten esimerkiksi, viiva, piste, koordinaatisto ja käyrä. Näistä koordinaatisto on toteutettu omaan paneeliin, ja muut tarvitsevat luontiparametrina piirtopinnan, jolle komponentti sijoitetaan.

4 VINO HEITTOLIIKE

4.1 Määrittely

Simulaation tarkoitus on havainnollistaa lähtönopeuden, lähtökulman ja lähtökorkeuden vaikutusta kappaleen lentorataan vinossa heittoliikkeessä sekä auttaa ymmärtämään kappaleen kulkusuunnan muodostumista nopeuden komponenteista kaksiulotteisessa koordinaatistossa.

Kappaleelle annetaan alkuarvoina lähtökorkeus, lähtökulma, lähtönopeus. Näiden alkuarvojen perusteella simulaatio piirtää lentoradan kaksiulotteiseen koordinaatistoon. Sellaisten alkuarvojen antaminen on estettävä, joilla heitto ylittäisi koordinaatiston. Heitto tapahtuu tyhjiössä, joten ilmanvastusta ei oteta huomioon. Painovoima on vakio, $g = 9.81\ m/s^{2.}$ Heiton aikana piirretään vaaka- ja pystysuuntaiset nopeusvektorit. Simulaatio antaa lopputuloksena seuraavat tiedot: lentoradan korkeimman kohdan, heiton kantaman ja laskeutumiskulman. Simulaation käyttöliittymän olisi hyvä olla yksinkertainen ja selkeä.

4.2 Teoria

Vinosti heitetyn kappaleen lentorata on tyhjiöolosuhteissa alaspäin aukeavan paraabelin kaaren muotoinen. Kun kappale on heitetty ilmaan, siihen vaikuttaa lentoradan aikana vain maan vetovoima, kun ilmanvastusta ei oteta huomioon. Koska maan vetovoima on pystysuora voima, on heittoliikkeessä vain pystysuoraa kiihtyvyyttä g ja vaakasuora liike on tasaista.

Vaakasuoran ja pystysuoran nopeuden komponentit on selvitettävä käyttäjän antamista parametreista. Nopeuden komponentit $v0_x$ ja $v0_y$ ajanhetkenä t saadaan laskettua, kun tiedetään lähtönopeus v0 ja lähtökulma α. Seuraavassa on esitetty nopeuskomponenttien laskentakaavat.

$$v_y = v0_y \sin\alpha - g\,t$$

$$v_x = v0_x \cos\alpha$$

Kun heittokaarta lasketaan animoinnin aikana, on kappaleen paikka x-y koordinaatistossa tiettynä hetkenä selvitettävä. Tämä onnistuu laskettujen nopeuskomponenttien avulla seuraavia kaavoja käyttäen:

$$x = v0_x t$$

$$y = v0_y t - \frac{1}{2} g\,t^2$$

Käyttäjän antaessa parametreja on sellaisten arvojen antaminen estettävä, joilla heitto menisi yli koordinaatistosta. Pystysuunnan raja-arvo h lasketaan heittoliikkeen lakikorkeuden kaavalla.

$$h = \frac{v_0^2 \sin^2\alpha_0}{2\mathrm{g}} + y_0$$

Vaakasuunnan raja-arvo R lasketaan heiton kantaman kaavan avulla.

$$R = \frac{V_x(V_{0\mathrm{y}} + \sqrt{v_{0\mathrm{y}}^2 + 2\mathrm{g}y_{0\mathrm{y}}})}{g}$$

(Hautala & Peltonen 2002, 18,15.; Opetushallitus 2008; Wikipedia 2008a; Nieminen 2008.)

4.3 Käyttöliittymä

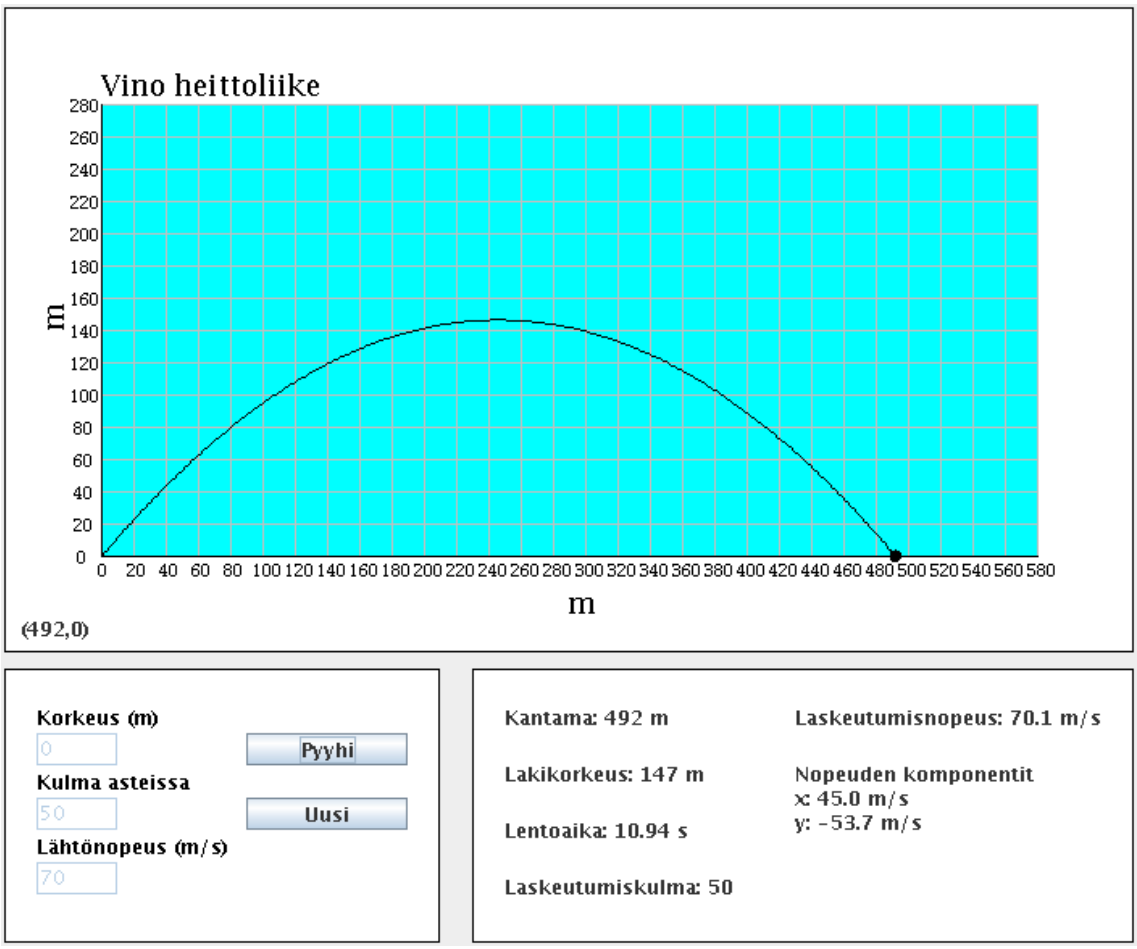
Käyttöliittymän suunnittelu aloitettiin jakamalla simulaatiolle varattu tila kolmeen sopivan kokoiseen osioon: koordinaatisto ja alkutiedot, tulokset. Alkutietojensyöttö päätettiin tehdä mahdolliseksi kahdella eri tavalla, tekstikenttiin arvoja syöttämällä tai hiiren kursorin avulla.

Tietojen syötön tapahtuessa hiiren avulla on eri syöttötilojen toiminta seuraavanlainen. Lähtökorkeuden syöttötilassa valitaan heiton lähtökorkeus osoittamalla hiiren kursorilla haluttua korkeutta. Lähtökulmansyötössä valitun korkeuden ja kursorin paikan välille piirtyy nuoli jonka kulma osoittaa heiton lähtökulman. Nopeudensyöttötilassa ei enää nuolen kulmaa eikä alkukorkeutta ei ole mahdollista muuttaa, sen sijaan nuolen pituus muuttuu hiiren kursoria liikuttamalla vaakasuunnassa tai pystysuunnassa riippuen nuolen kulmasta. Kaikista näistä syöttötiloista seuraavaan tilaan siirtyminen tapahtuu hyväksymällä osoitettu arvo hiiren vasenta näppäintä painamalla. Arvot näkyvät reaaliajassa niille varatuissa tekstikentissä. Kuitenkaan tila ei vaihdu, jos valittu arvo ei ole sallituissa rajoissa. Rajojen ylittymisestä kertoo nuolen värin vaihtuminen punaiseksi.

Kun alkutiedot syötetään tekstikenttiin, tapahtuu arvon hyväksyminen kyseessä olevan tekstikentän viereen ilmestynyttä nuolipainiketta painamalla. Arvojen oikeellisuus tarkistetaan jo niitä syötettäessä, eikä tilasta seuraavaan siirtyminen ole mahdollista ilman hyväksyttäviä arvoja.

Molemmissa syöttötavoissa syöttövuorossa olevan arvon otsikko muuttuu punaiseksi selventämään käyttäjälle, minkä arvon syöttö on kyseessä. Näitä molempia tapoja voi myös käyttää sekaisin, esimerkiksi jos lähtökorkeus syötetään hiiren avulla, voi seuraavat tiedot syöttää siitä huolimatta tekstikenttiin. Jos arvon syötössä tapahtuu virhe, ei tiloissa voi siirtyä taaksepäin, vaan heitto on suoritettava loppuun ja on aloitettava uusi heitto alusta. Tämän simulaation käyttöliittymäkomponentit lisättiin paneelille ilman NetBeans- käyttöliittymäsuunnittelutyökalun apua. Komponenteille

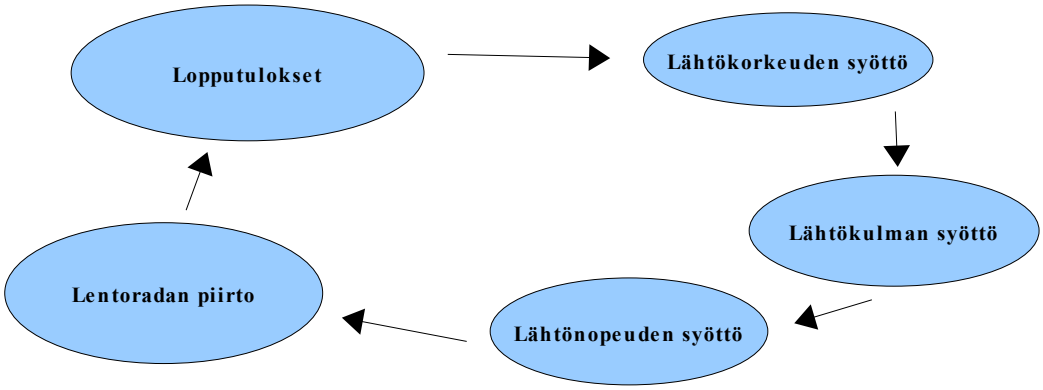
asetettiin paikat ja koot käsin. Kuvassa 2 näkyy vinoheittoliike -simulaation käyttöliittymä.



Kuva 2. Vinon heittoliikkeen simulaation käyttöliittymä

4.4 Toiminta

Ohjelman kulku toteutettiin tilakoneella. Jokaisen alkuarvon syöttöön tehtiin oma tila, ja lisäksi tehtiin tilat lopputuloksen lentoradan animoimiselle ja lopputuloksen näyttämiselle. Tilat näkyvät seuraavassa kaaviossa.



Kuva 3. Vino heittoliike simulaation tilakone

Alkuarvojen syöttötiloista siirrytään seuraavaan tilaan, kun käyttäjä on syöttänyt oikeanlaiset arvot ja hyväksynyt ne. Lentoradan piirtotilasta siirrytään Lopputulokset-tilaan, kun kappale kohtaa y-akselin nollakohdan.

4.5 Testaus

Testejä varten tehtiin taulukkolaskentaohjelmalla taulukko, johon laskettiin tarvittavat lopputulokset annetuilla alkuarvoilla, ja verrattiin niitä simulaation antamiin arvoihin. Testeihin käytetyt arvot näkyvät alla olevasta taulukosta.

Taulukko 1. Vino heittoliike simulaation testaus arvot

	Heitto1	Heitto2	Heitto3	Heitto4	Heitto5	Heitto6
Lähtökorkeus (m)	0	0	60	200	30	5
Lähtökulma	45	70	20	-45	14	15
Lähtönopeus (m/s)	50	30	70	60	54	55
V_{0X}	35,36	10,26	65,78	42,43	52,4	53,13
V_{0Y}	35,36	28,19	23,94	-42,43	13,06	14,24
Lakikorkeus (m)	63,71	40,51	89,21	200	38,7	15,33
Lentoaika (s)	7,21	5,75	6,71	3,39	4,14	3,22
Nousuaika (s)	3,6	2,87	2,44	-4,32	1,33	1,45
Laskuaika (s)	3,6	2,87	4,26	7,71	2,81	1,77
Kantama (m)	254,84	58,97	441,07	143,72	216,95	171
Laskeutumisnopeus (m/s)	50	30	77,96	86,74	59,2	55,88
V_{1X}	35,36	10,26	65,78	42,43	52,4	53,13
V_{1Y}	-35,36	-28,19	-41,84	-75,66	-27,55	-17,34
Laskeutumiskulma	45	70	32,46	60,72	27,74	18,08

Testeissä havaittiin, että laskennalliset ja simulaation antamat arvot vastaavat toisiaan. Näiden testien perusteella todettiin vino heittoliike- simulaatio toimivaksi.

5 SINIAALTOJEN SUMMA

5.1 Määrittely

Simulaation tarkoitus on havainnollistaa kahden summakäyrän muodostumista kahdesta siniaallosta sekä auttaa ymmärtämään seisovan aaltolikkeen muodostumista kahdesta samanlaisesta vastakkaiseen suuntaan liikkuvasta aalloista.

Simulaatio piirtää kaksi siniaaltoa annetuista parametreista samaan koordinaatistoon sekä niiden summan aallon toiseen koordinaatistoon. Sinikäyrien parametreiksi syötetään taajuus, amplitudi, vaihe ja Off-piste. Summakäyrä piirtyy reaaliaikaisesti koordinaatistoon. Sinikäyrät tulee saada liikkeeseen sekä saman- että vastakkaissuuntaisesti käyrien vaiheitta muuttamalla. Koordinaatistojen x- ja y-akselien skaaloja tulee voida muuttaa.

5.2 Teoria

Siniaalto on jaksollinen, sinifunktion muotoinen aaltomuoto. Se on harmonisen värähtelyn perusmuoto. Sinikäyrän matemaattinen muoto ajan funktiona lasketaan seuraavasta kaavasta.

$$y(t) = A\sin(\omega + 0)$$

Aaltofunktion tapauksessa parametri saattaa kuvata aikaa, paikkaa tai niiden yhdistelmää: esimerkiksi sinimuotoisen liikkuvan aallon korkeus y voidaan ilmaista ajan t ja paikan x funktiona.

$$y(x,t) = A\sin(kx - \omega t + 0)$$

missä A on amplitudi, k on kulma-aaltoluku, ω on kulmataajuus ja θ on aallon vaihe pisteessä (x = 0, t = 0). (Wikipedia 2008b.)

Kahden samantaajuisen siniaallon vaihe-ero ilmaistaan joko radiaaneina tai asteina. Esimerkiksi 90 asteen vaihe-ero voidaan ajatella syntyvän siten, että pyörivän osoittimen kärki projisioidaan kahdelle toisiaan vastaan 90 asteen kulmassa olevalle suoralle, kuten koordinaatiston x- ja y-akseleille.

(Wikipedia 2008c.)

Kulmataajuus (tunnus ω) ilmoittaa kuinka suuren kulman jokin sini-signaalin syntymistä kuvaava vektori pyörähtää aikayksikössä. Nimitystä kulmataajuus käytetään erityisesti elektroniikan ja sähkötekniikan signaalien yhteydessä. Fysiikassa ja mekaniikassa käytetään tavallisemmin termiä kulmanopeus. Tällöin sinimuotoisen signaalin ajatellaan muodostuvan pyörivästä vektorista, jonka kulmanopeus ilmaisee vektorin kulman muutoksen aikayksikössä.

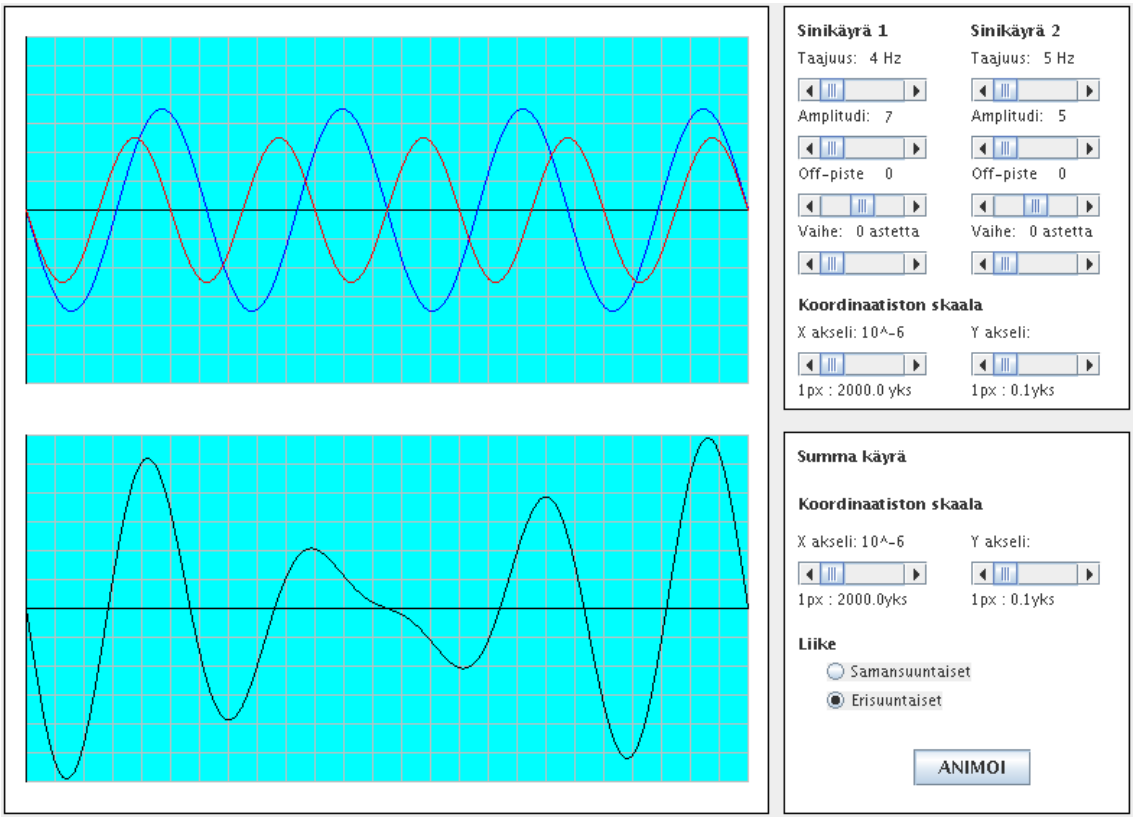
Samantapaisia suureita kuin kulmataajuus ovat taajuus ja kierrosnopeus, jotka kertovat värähtelyn jaksojen tai kierrosten lukumäärän aikayksikössä. Täysi kierros on 2π rad, joten taajuutta 1 Hz tai kierrosnopeutta 1 kierr/s vastaa kulmataajuus tai kulmanopeus 2π rad/s.

(Wikipedia 2008d.)

5.3 Käyttöliittymä

Käyttöliittymä jaettiin kolmeen eri paneeliin. Koordinaatistot sijaitsevat omassa paneelissaan, sinikäyrien parametrien syöttö ja koordinaatiston skaalaus omassaan sekä summakäyrän koordinaatiston ja animaation käynnistyspainike omassa. Kuvassa 4

näkyy siniaaltojen summasimulaation käyttöliittymä.



Kuva 4. Siniaaltojensumma simulaation käyttöliittymä

5.3.1 Koordinaatistot ja käyrät

Koordinaatistoina käytettiin aiemmin luotua CoordinateGrid-luokkaa. Koordinaatistoille ei asetettu otsikoita eikä numeroita ruudukkoon, koska ne ovat epäolennaisia tietoja tässä simulaatiossa. Luokkaan lisättiin mahdollisuus muuttaa koordinaatiston akselien skaalaa luonnin jälkeen. CoordinateGrid-luokan esitetään luvussa 8.1.

Sinikäyrät piirretään samaan koordinaatistoon eri väreillä, summakäyrää varten on oma koordinaatisto. Käyrien piirtoa varten luotiin uusi luokka Curve, joka ylläpitää ja käsittelee vektoria käyrän pisteistä. Luokka hoitaa myös käyrän piirtämisen annettuun grafiikka objektiin. Curve-luokan toiminta käsitellään kohdassa 8.4.

Sinikäyrien piirtoa varten luokka Curve laajennettiin luokalla SineCurve. SineCurve-luokka sisältää sinikäyrälle tyypilliset piirteet kuten amplitudin, taajuuden, Off-pisteen, skaalan ja vaiheen. Luokassa on myös metodi calculatePoints(), joka laskee sinikäyrän kaikkien pisteiden koordinaatit.

5.3.2 Parametrien syöttö ja animoinnin käynnistäminen

Sinikäyrien parametrien asetus ja koordinaatiston skaalaus tapahtuu JscrollBar-tyyppisillä liukuvalitsimilla joiden ylä ja ala rajat näkyvät taulukosta 2.

Taulukko 2. Sinikäyrien parametrien raja-arvot

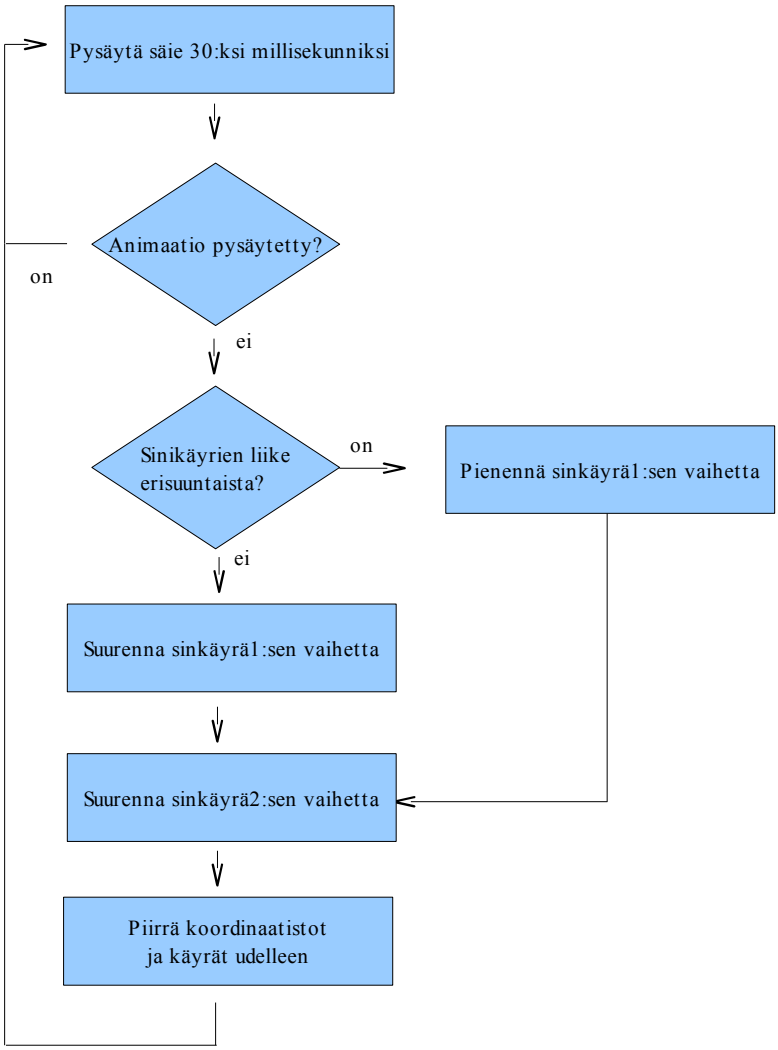
Liukuvalitsin	**Minimi**	**Maksimi**
Taajuus	1 Hz	1000 Hz
Amplitudi	1	1000
Off-piste	-1000	1000
Vaihe	0	360
X akselin skaalaus	1 pikseli : 2000.0 yksikkö * 10 ^ -6	1 pikseli : 10.0 yksikkö * 10 ^ -6
Y akselin skaalaus	1 pikseli : 0.1 yksikköä	1 pikseli : 10.0 yksikköä

Syötetyt arvot näkyvät tekstikentissä liukuvalitsimien yläpuolella, ja arvoihin tehdyt muutokset näkyvät reaaliaikaisesti sinikäyrissä ja summakäyrissä.

Animaatio käynnistetään JToggleButton-tyyppisellä painikkeella, joka jää pohjaan painettaessa. Kun painike on pohjassa, animaatio on käynnissä ja vastaavasti, kun painike on ylhäällä, animaatio on pysähdyksissä. Animaation ollessa käynnissä muutetaan sinikäyrien vaiheita tarvittaessa. Sinikäyrien vaiheen muutossuunta valitaan radiovalitsimilla, joista vain toinen voi olla aktiivinen. Kun animaatio pysäytetään, siirtyvät liukukytkimet, joilla valitaan sinikäyrien vaihe siihen arvoon, mihin animaatio pysäytettiin.

5.4 Toiminta

Simulaatio toimii tapahtumapohjaisesti. Kun käyttäjä muuttaa sinikäyrien parametreja tai koordinaatiston skaalausta, piirtää appletti käyrät ja koordinaatistot uudelleen uusilla parametreilla. Summakäyrä lasketaan joka kerta uudelleen, kun jommankumman sinikäyrän parametrit muuttuvat. Summakäyrä muodostuu sinikäyrien pisteiden summista. Animoitaessa sinikäyrien vaiheita muutetaan 30 millisekunnin välein vastakkaiseen tai samaan suuntaan. Kuvassa 5 on kuvattu vuokaaviolla animaation toiminta.



Kuva 5. Siniaaltojen liikeen animointi

5.5 Testaus

Testauksella varmistettiin, että koordinaatistojen skaalaus toimii aiotulla tavalla ja että käyrät muuttuvat skaalauksen muutosten mukaisesti. Testejä varten tehtiin taulukkolaskentaohjelmalla taulukko, johon laskettiin tarvittavat lopputulokset annetuilla alkuarvoilla ja verrattiin niitä simulaation antamiin arvoihin. Testeihin käytetyt arvot näkyvät taulukosta 3.

Taulukko 3. Siniaaltojensumma simulaation testaustaulukko

	Tapaus1	Tapaus2	Tapaus3
Sinikäyrä1			
Taajuus (Hz)	100	100	100
Amplitudi	100	100	100
Off-piste	0	0	20
Vaihe (astetta)	0	180	0
Y(250µ)	100	-100	120
Y(75µ)	45,38	-45,38	65,38
Y(100µ)	58,75	-58,75	78,75
Sinikäyrä2			
Taajuus (Hz)	150	150	150
Amplitudi	50	50	50
Off-piste	0	0	-20
Vaihe (astetta)	0	90	0
Y(250µ)	35,4	-35,31	15,4
Y(75µ)	32,46	38,03	12,46
Y(100µ)	40,44	29,41	20,44
Summakäyrä			
Y(250µ)	135,4	-135,31	135,4
Y(75µ)	77,84	-7,35	77,84
Y(100µ)	99,19	-29,34	99,19

Vertailtaessa simulaation muodostamia sinikäyriä ja niiden summakäyrää todettiin, että ne muodostuvat oikein koordinaatistoihin. Lisäksi varmistettiin, että animoitaessa sinikäyrien liikettä niiden vaiheet muuttuvat oikealla tavalla. Näiden testejen perusteella todettiin simulaatio toimivaksi.

6 LIIKE KOORDINAATISTOSSA

6.1 Määrittely

Simulaation tarkoitus on havainnollistaa tasaisesti kiihtyvää liikettä animaation ja kaksiulotteiseen koordinaatistoon piirrettävien käyrien avulla. Kappaleen nopeutta, paikkaa ja kiihtyvyyttä tarkastellaan ajan suhteen.

Simulaatiolle annetaan alkuarvoina kiihtyvyys, lähtönopeus ja paikka. Simulaatio piirtää nopeuden ja paikan ajan funktioina eri koordinaatistoihin. Koordinaatistot skaalautuvat automaattisesti. Kiihtyvää liikettä kuvaamaan piirretään animaatio, jossa kappale etenee kiihtyvällä nopeudella janalla, joka kertoo etäisyyden janan lähtökohtaan. Kulunut aika, kappaleen nopeus ja kappaleen kulkema matka sekä kiihtyvyys näytetään tekstikentissä. Animaatiolle tulee pystyä määrittelemään tietty aika, nopeus tai paikka jossa animaatio pysäytetään. Animaation voidaan pysäyttää tai resetoida myös painikkeiden avulla. Resetointi aloittaa käyrien ja animaation piirron alusta valituilla alkuarvoilla.

6.2 Teoria

Animoitaessa kappaleen liikettä sekä piirrettäessä kappaleen nopeus ja sijaintikäyriä koordinaatistoon tarvitaan laskukaavat, joiden avulla lasketaan kappaleen sijainti ja nopeus tiettynä ajanhetkenä. Kappaleen nopeus v ja sen kulkema matka s tasaisesti kiihtyvässä liikkeessä ajanhetkenä t saadaan seuraavista kaavoista:

$$s = s_0(v_0 t + 1/2\mathrm{at}^2)$$

$$v = v_0 + at$$

, missä v0 on alkunopeus ja s0 on paikka alkutilanteessa sekä a on kiihtyvyys.

(Rovaniemen ammattikorkeakoulu 2009.)

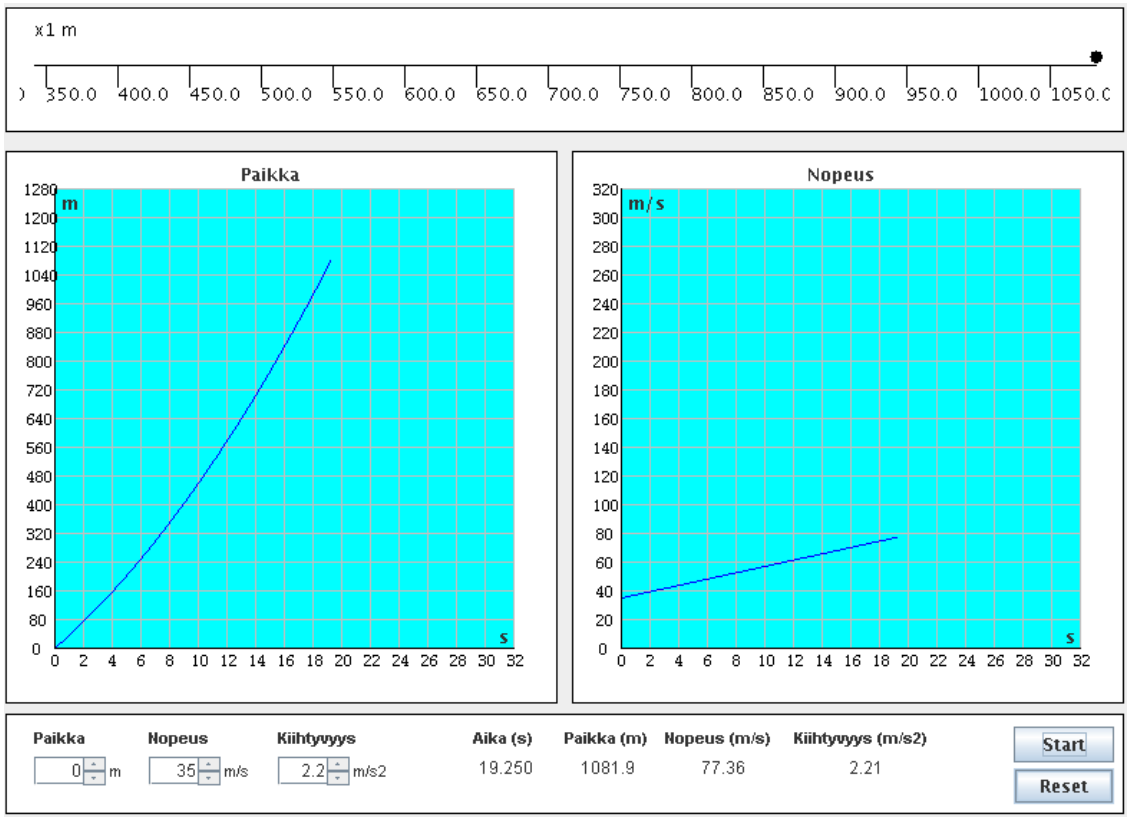
Testeissä tarvitaan kaavoja, joilla on mahdollista laskea kulunut aika sijainnnin, kiihtyvyyden ja lähtönopeuden avulla, sekä kaava, jolla voidaan laskea sama asia lähtönopeuden, nopeuden ja kiihtyvyyden avulla. Kaavat johdettiin edellä olevista kaavoista. Alla johdetut kaavat.

$$t = \frac{-v_0 + \sqrt{v_0^2 + 2a(s - s_0)}}{a}$$

$$t = \frac{v - v_0}{a}$$

6.3 Käyttöliittymä

Käyttöliittymässä on neljä eri paneelia, yksi kappaleen kiihtyvyysjanaa varten, molemmille koordinaatistoille oma sekä arvojen syöttöön varattu paneeli. Kuvassa 6 on kuvankaappaus liikekoordinaatistossa simulaation käyttöliittymästä.



Kuva 6. Liikekoordinaatistossa simulaation käyttöliittymä

6.3.1 Arvojen syöttö

Alkuarvojen ja pysäytysarvojen syöttö tapahtuu JSpinner-tyyppisillä liukuvalitsimilla. JSpinner on Javan Swing-käyttöliittymäkirjaston komponentti. Liukuvalitsimeen voi syöttää arvoja näppäimistöä tai hiirtä käyttäen. JSpinner-komponentille asetetaan minimi, maksimi arvot sekä tarkkuus. Ohjelmassa arvot ovat rajoitettu taulukon 4 mukaisesti.

Taulukko 4. Liike koordinaatistossa simulaation alkuarvojen raja-arvot

Liukuvalitsin	**Minimi**	**Maksimi**	**Tarkkuus**
Lähtöpaikka	0	1000	1
Lähtönopeus	0	1000	1
Kiihtyvyys	0	100	0,1
Pysäytysaika	0	3600	0,1
Pysäytysnopeus	0	10000	1
Pysäytyspaikka	0	10000	1

Alkuarvoja ei ole mahdollista muuttaa, kun animointi on käynnissä. Pysäytysarvojen syöttöliukuvalitsimien vieressä on valitsin, jolla määritellään onko, kyseinen pysäytysarvo käytössä.

6.3.2 Reset- ja Start- painikkeet

Reset-painike palauttaa animaation alkutilaan. Kun Reset-painiketta painetaan, siirtyy kappale liukuvassa koordinaatistossa kohtaan, jonka lähtöpaikka määrittelee ja käyrät koordinaatistoissa pyyhitään. Kappaleelle haetaan uudet alkuarvot kiihtyvyys, nopeus ja paikka liukuvalitsimista. Animaation pysäytyspainike on kaksitoiminen: kun animaatio on käynnissä, animaatio pysäytetään painikkeesta, kun taas animaatio on pysähdyksissä, käynnistetään se kyseisellä painikkeella. Painikkeen teksti vaihtelee toiminnon mukaan.

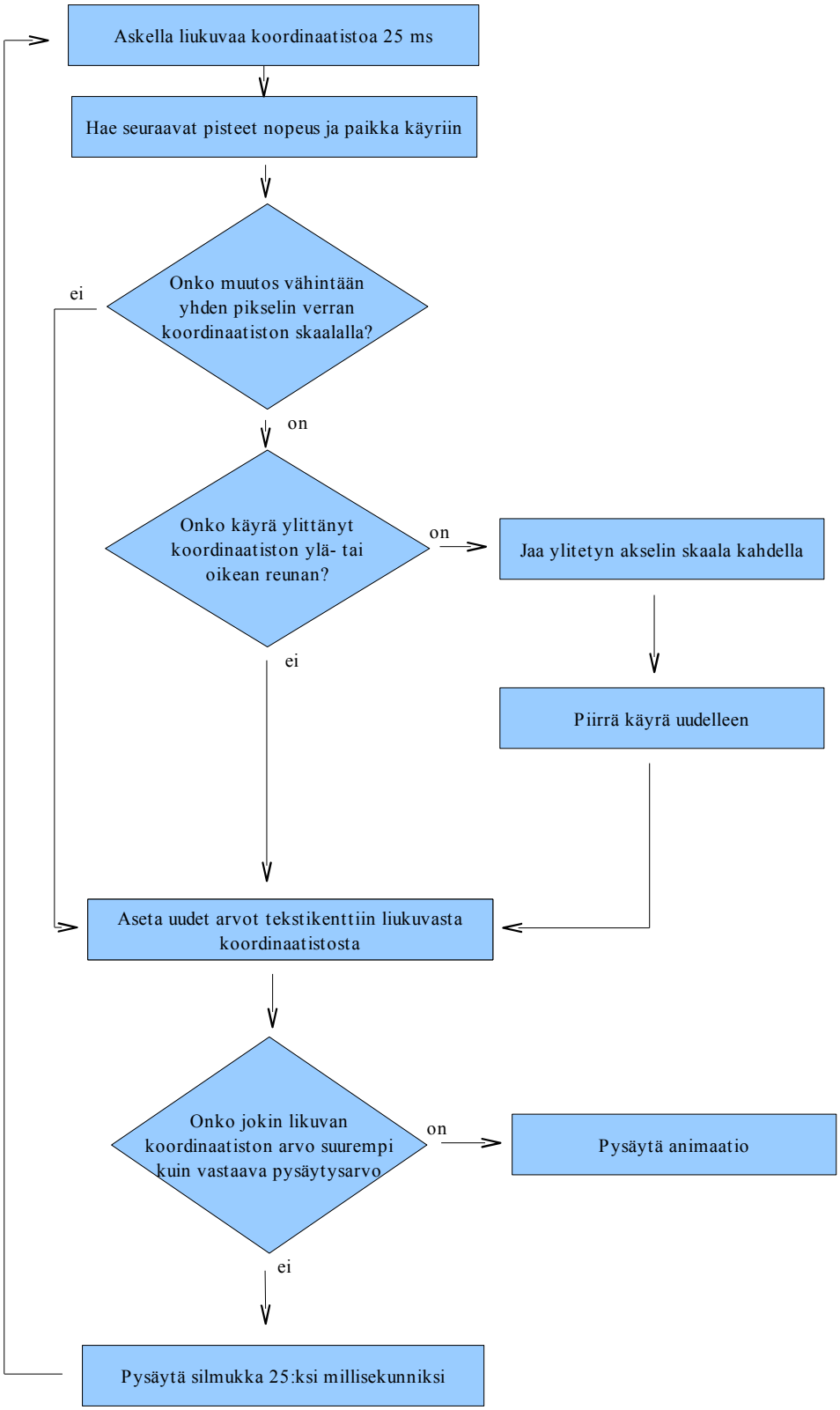
6.3.3 Koordinaatistot ja käyrät

Paikka- ja nopeuskoordinaatistoina käytetään aiemmin luotua CoordinateGrid-luokkaa. Luokan toiminta käsitellään kohdassa 8.1. Ohjelman pääsilmukassa toteutettiin koordinaatistoon automaattinen skaalaus. Koordinaatiston akselien skaala jaetaan kahdella aina, kun käyrä on ylittämässä koordinaatiston ylä reunan tai oikean reunan. Käyrien piirtämisessä käytettiin hyväksi aiemmin luotua Curve -luokkaa. Sen toiminta käsitellään kohdassa 8.6. Koordinaatistojen skaalan vaihtuessa on myös käyrät piirrettävä uudelleen uudella skaalalla.

Kiihtyvän kappaleen animointia varten suunniteltiin luokka, joka piirtää kappaleen yksiulotteiseen koordinaatistoon ja liuttaa koordinaatistoa, kun kappale saavuttaa sen oikean reunan. Koordinaatiston ainoalla akselilla on kuljettu matka metreinä. Koordinaatistolle annetaan alkuarvoina paikka, koko ja taustaväri luonnin yhteydessä. Koordinaatisto ja kappale piirretään paneelille, joten luokka perii JPanel -luokan. Luokan toiminta perustuu metodiin, joka laskee annetusta ajasta kappaleen kulkeman matkan ja nopeuden ja siirtää kappaletta eteenpäin koordinaatistossa lasketun matkan verran.

6.4 Toiminta

Simulaation toiminta perustuu liukuvaan koordinaatistoon, jota askelletaan 25 ms:n välein silmukassa. Arvot paikka- ja nopeuskäyrille haetaan liukuvan koordinaatiston arvoista. Pääsilmukassa tarkkaillaan myös, etteivät käyrät ylitä koordinaatistoa sekä sitä, että animaatio pysäytetään, jos jokin pysäytysarvo ylittyy. Ohjelman pääsilmukka pyörii omassa säikeessä, ja sen toiminta on kuvattu kuvassa 7.



Kuva 7. Liike koordinaatistossa -simulaation pääsilmukan toiminta

6.5 Testaus

Testejä varten tehtiin taulukkolaskentaohjelmalla taulukko, johon laskettiin tarvittavat lopputulokset annetuilla alkuarvoilla ja verrattiin niitä simulaation antamiin arvoihin. Testeihin käytetyt arvot näkyvät taulukosta 3.

Taulukko 5. Tasaisesti kiihtyvän liikkeen simulaation testaustaulukko

	Tapaus1	Tapaus2	Tapaus3	Tapaus4
Lähtönopeus (m/s)	0	10	0	10
Lähtöpaikka (m)	0	0	50	50
Kiihtyvyys (m/s2)	5	10	5	10
Paikka 10s päästä (m)	250	600	300	650
Aika 1000m päässä (s)	20	13,18	19,49	12,82
Paikka kun nopeus on 28 (m/s)	5,6	1,8	5,6	1,8

Testauksen aikana havaittiin ongelma animaation pysäytyksessä: suurilla kiihtyvyyden arvoilla animaatio ei pysähdy tarkalleen määriteltyyn nopeuden tai sijainnin arvoon. Tämä johtuu siitä, että animaatiota askelletaan 25 millisekunnin askelilla, ja yhden askeleen aikana animaatio ehtii hypätä annetun pysäytysarvon ylitse. Vikaa ei korjattu, koska pysäytysarvojen ylitykset normaaleilla kiihtyvyyksillä ovat mitättömät.

7 YHTEISET KOMPONENTIT

7.1 Yleistä

Tässä luvussa käydään läpi komponentit, jotka kuuluvat useampaan simulaatioon. Komponentit sijaitsevat Components-paketissa, joka on Fyanim-pääpaketin alipaketti. Luotaessa uusia animaatioita ohjelmaan tullaan tässä luvussa kuvattuja komponentteja tarvitsemaan. Keskeisin ja suuritöisin komponentti on koordinaatisto, ja muut komponentit ovat graafinen objekti, viiva, nuoli ja käyrä. Components-paketti sisältää myös muita komponentteja, jotka liittyvät yksittäisiin simulaatioihin.

7.2 Koordinaatisto – CoordinateGrid.java

CoordinateGrid.java -tiedosto sisältää koordinaatistoluokan toteutuksen. CoordinateGrid on components-paketin tärkein luokka, ja sitä käytetään kaikissa toteutettavissa simulaatiossa. Koska luokasta pyrittiin tekemään mahdollisimman yleiskäyttöinen, tuli luokasta suhteellisen monimutkainen. Koordinaatistoon lisättiin uusia ominaisuuksia projektin kuluessa. Luokkaa jatkokehitettäessä on varmistettava, että yhteensopivuus jo tehtyjen simulaatioiden kanssa säilyy. Luokan olemassa olevia julkisia funktiota ei tulisi muuttaa, tai jos muutos joudutaan tekemään, on se toteutettava myös kaikkiin jo tehtyihin simulaatioihin. Parempi tapa kuin muuttaa luokan rajapintaa, on tehdä uusi luokka, joka perii koordinaatistoluokan ominaisuudet. Koordinaatistoon toteutettiin seuraavat ominaisuudet:

- Koon määrittely luonnin yhteydessä
- Apuruudukon tiheyden määrittely
- Pääotsikon sekä X ja Y akseleiden otsikot
- Mahdollisuuden määritellä marginaalit
- Skaalan määritteleminen
- Mahdollisuus muuttaa skaalaa luonnin jälkeen
- Mahdollisuus apuruudukon numeroiden piirtämättä jättämiseen

Kun koordinaatisto luodaan, sille välitetään luontiparametreina alkutietoja. Koordinaatiston luontiparametrit ja niiden selitykset löytyvät taulukosta 6.

Taulukko 6. CoordinateGrid olion luontiparametrit

Parametri	**Käyttö**
int widthPx Int heightPx	Koordinaatiston leveys ja korkeus pikseleinä sisältäen marginaalit.
int marginPx	Koordinaatiston vasen, oikea, ylä ja ala marginaalit pikseleinä.
int gridCellWidth int gridCellHeight	Y ja X akselien apuviivojen väli pikseleinä.
double oneUnitInPxX Double oneUnitInPxY	Yksi pikseli yksikköinä X ja Y akseleilla. Esimerkiksi jos haluttaisiin että yksi pikseli näytöllä vastaisi puolta yksikköä X akselilla, annettaisiin OneUnitInPxX parametrille arvo 0,5.
int xStart Int yStart	Kertoo kunka monta yksikkö näytetään Y ja X akselien negatiiviselta puolelta. Nämä parametrit ovat aina negatiivisia. Esimerkiksi, jos haluttaisiin luoda koordinaatisto joka Y akselin negatiivisella alueella olisi 120 yksikön verran annettaisiin yStart parametrin arvoksi -120.
String caption	Koordinaatiston pääotsikko.
String xCaption String yCaption	Koordinaatiston X ja Y akselien otsikot.

Luonnin yhteydessä koordinaatisto piirretään annettujen parametrien avulla muistissa olevaan kuvaan createCoordinateGridPicture()-metodissa. Seuraavaksi piirretään muistissa oleva kuva muistissa olevalle piirtopinnalle clear()-metodissa. Vielä pitää siirtää muistissa oleva piirtopinta näytölle, mikä tapahtuu metodissa paintComponent(Graphics g), joka ylikirjoitettaan koordinaatisto luokan JPanel yliluokasta. PaintComponent -metodia ei saa kutsua suoraan vaan se on tehtävä repaint()- metodin kautta.

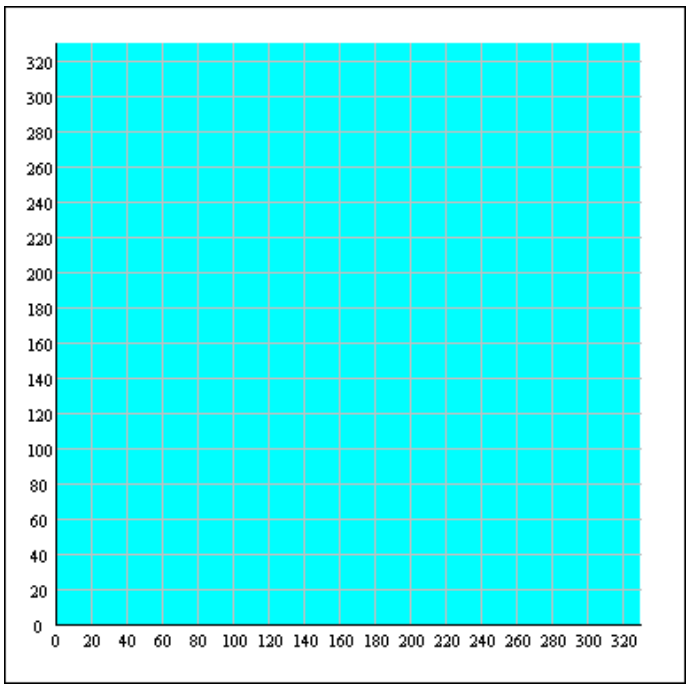
Tämäntyyppistä piirtämistä kutsutaan kaksoispuskuroinniksi. Kaksoispuskuroinnilla tarkoitetaan sitä, että kaikki piirrettävä piirretään ensin muistissa sijaitsevalle piirtopinnalle, ja kun piirtäminen on kokonaisuudessaan suoritettu, siirretään valmis kuva näytölle. Kaksoispuskuroinnin etuna on se, että se vähentää välkkymistä ja muita haitallisia ilmiötä, jotka saattaisivat ilmetä, jos kuva piirrettäisiin suoraan näytölle. (Cadenhead 2001, 291.)

Koordinaatiston parametreja voidaan muutta metodien setYscale(double onePx), setXscale (double onePx)- ja setCaptions (String xCaption, String yCaption) avulla.

Muutosten jälkeen on aina kutsuttava clear()- ja repaint()-metodeja.

Haluttaessa piirtää koordinaatiston pinnalle toisesta luokasta, on koordinaatiston piirtopinta haettava getbuffer()-metodilla. Metodi palauttaa osoitteen koordinaatiston piirtopintaan. Piirtopinta on Graphics2D-tyyppinen olio, joka tukee Javan kehittyneitä piirtorutiineita. Piirtämisen jälkeen on koordinaatiston repaint()-metodia kutsuttava, jotta piirretyt kuviot tulisivat näkyviin. Piirrettäessä koordinaatistoon on otettava huomioon, että piirtopinnan 0,0 koordinaatti on pinnan vasemmassa yläreunassa eikä oikeassa niin kuin koordinaatiston. Lisäksi piirtäminen tapahtuu pikseleittäin, eikä koordinaatiston skaalaa oteta huomioon. Skaala on siis laskettava piirtävässä luokassa. Tämä on yksi luokan kehityskohteista.

Kuvassa 8 on esimerkkikoordinaatisto, joka on luotu ilman otsikoita 20 pikselin marginaaleilla ja 20 pikselin apuviivojen välillä. Yksi pikseli on määritelty yhden yksikön pituiseksi, ja x- ja y-akseleiden negatiivisiskialueiksi on asetettu nollat.



Kuva 8. Esimerkkikoordinaatisto

7.1 Graafinen objekti – GraphicObject.java

GraphObject.java tiedosto sisältää GraphObject-luokan toteutuksen. Luokkaa käytetään apuna graafisten objektien yliluokkana. GraphObject on abstrakti luokka, josta ei voi luoda ilmentymiä, vaan se on tarkoitettu perittäväksi. Luokka sisältää koordinaatit sekä desimaalisina että kokonaislukuina, ja lisäksi luokassa on määritelty muuttujat pystysuuntaiselle ja vaakasuuntaiselle liikkeelle. GraphObject toimii Line, Arrow- ja Point-luokkien yliluokkana.

7.2 Viiva – Line.java

Line.java tiedosto sisältää Line-luokan toteutuksen. Luokkaa käytetään viivan piirtoon. Se perii GraphObject-luokan ominaisuudet. Luokka sisältää perittyjen kenttien lisäksi viivan loppupään koordinaatit sekä kulmakertoimen. Yliluokasta on ylikirjoitettu draw- ja setLocations-metodit, joiden toiminta on muutettu viivan siirtoon ja piirtoon sopiviksi. Lisäksi luokka sisältää metodeita viivan kulmakertoimen laskemiseen sekä viivan pituuden ja loppupään koordinaattien muokkaamiseen.

7.3 Nuoli – Arrow.java

Arrow.java- tiedosto sisältää Arrow-luokan toteutuksen. Luokkaa käytetään nuolen piirtoon ja se perii Line-luokan ominaisuudet. Arrow -luokka siis lisää Line-luokkaan ominaisuuden, joka piirtää viivan nuolenpään. Nuolenpää koostuu kahdesta toisiinsa nähden 45 asteen kulmassa olevasta viivasta, jotka pitenevät nuolen pidetessä. Arrow-luokka sisältää muuttujat nuolenpään viivojen koordinaateille sekä nuolenpään laskemiseen tarvittavat metodit.

7.4 Käyrä – Curve.java

Curve.java- tiedosto sisältää Curve-luokan toteutuksen. Luokkaa käytetään käyrien piirtoon. Curve-luokka pitää yllä kaksiulotteista taulukkoa, joka sisältää käyrän kaikki pisteet. Luokka sisältää käyrän pyyhkimiseen, piirtoon ja pisteiden lisäämiseen

tarvittavat metodit. Käyrälle voi asettaa myös skaalaan. Käyrälle määritellään sen luonnin yhteydessä taulukon 7 parametrit.

Taulukko 7. Curve olion luontiparametrit

Parametri	**Käyttö**
Graphics2D drawingSurface	Piirtopinta jolle käyrä piirretään
int startX	Alun X koordinaatti
int startY	Alun Y koordinaatti
int width	Pituus, määrittelee myös käyrän pisteiden lukumäärän
int top	Suurin korkeus
int bottom	Pienin korkeus
Color c	Käyrän väri

8 JATKOKEHITTÄMINEN JA YHTEENVETO

Projektin alussa etsittiin sivustoja, joilla on toteutettu samankaltaisia fysiikan simulaatiota, joita oli suunnitteilla. Lisäksi mietittiin valmiiden ympäristöjen käyttöä simulaatioiden toteutusalustaksi. Simulaatiot päädyttiin toteuttamaan alusta asti itse Java-kielellä. Perusteena tälle oli haasteiden pitäminen ohjelmistoteknisinä ja valmiiden alustojen puutteellisuus. Nämä olivat ehkä osittain vääriä perusteita, sillä projektin loppupuolella tutustuin muutamaan erittäin mielenkiintoiseen ja joustavaan ympäristöön, jotka olisivat tehostaneet ja parantaneet simulaatioiden kehittämistä Javalla. Väärien olettamusten syntyminen perustui muutamaan alustaan, joihin tutustuin. Näissä alustoissa simulaatiot muodostettiin graafisen käyttöliittymän avulla ilman ohjelmointia, ja näin ollen ne eivät olleet kovin joustavia ja monipuolisia. Voidaan siis jälkiviisaana todeta, että alussa tehtyyn tutkimustyöhön olisi voinut käyttää enemmän aikaa.

Opinnäytetyön ensimmäinen tavoite täyttyi hyvin. Kolme hyvin toimivaa ja havainnollista simulaatiota toteutettiin web-sivuilla toimiviksi. Projektin alussa asetettiin toiseksi tavoitteeksi uusien simuulaatioiden luomisen helpottaminen. Tämä

tavoite ei ehkä toteutunut siinä laajuudessaan kuin oli suunniteltu. Kattavan ja hyvin dokumentoidun ympäristön rakentaminen olisi vaatinut liikaa työtä, eikä se ollut mahdollista tämän opinnäytetyön puitteissa. Jatkokehittämistä ajatellen tehtiin projektista kuitenkin helposti laajennettava ja muutamia käyttökelpoisia luokkia syntyi simulaatioita luotaessa. Kuitenkin ehkä tärkein asia, mitä tämä opinnäytetyö antaa mahdolliselle jatkokehittäjälle, on uusien simulaatioiden luomisen malli. Luotujen simulaatioiden lähdekoodit toimivat ohjeena uusia simulaatioita kehitettäessä. Esimerkin vuoksi vino heittoliike -simulaation käyttöliittymä on koostettu käsin ilman NetBeans- käyttöliittymäsuunnittelutyökalun apua. Lisäksi projektin aikana koottiin ohjeita keskeisistä aiheista simulaatiota suunniteltaessa Java-kielellä. Ohjeet on koottu liitteeseen 2.

Projektin aikataulu ylittyi, koska projektin toteuttaja työllistyi. Projektia ei ollut mahdollista tehdä töiden ohella suunnitellussa aikataulussa. Projekti oli myös pitkän aikaan katkolla vaiheessa, jossa simulaatiot olivat viimeistelyä vaille valmiit. Katkos johtui sisäisen motivaation puutteesta eikä ulkoinenkaan motivointi ollut ollut tarpeeksi vahvaa. ”Usein ohjelmoijan innostus lopahtaa helposti vaiheessa, jossa perusongelmat on saatu ratkaistua”, toteaa Linus Torvaldskin kirjassaan Just For Fun.

LÄHTEET

Julkaisut

Cadenhead, R. 1999. Java 2 Trainer. Helsinki: IT Press.

Deitel, H. & Deitel, J. 2005. Java how to program. New Jersey: Deitel.

Hautala, M. & Peltonen, H. 2002. Insinöörin (AMK) fysiikka. Osa 1. Jyväskylä: Lahden Teho-Opetus Oy.

Koskinen, K. 2000. Oliokirja. Jyväskylä: Satku -Kauppakaari.

Kosonen, P., Peltomäki J. & Silander, S. 2005. Java 2 Ohjelmoinnin peruskirja. Porvoo: Docendo Finland Oy.

Vesterholm, M. & Kyppö, J. 2001. Java-ohjelmointi PRO TRAINING. Helsinki: Talentum Media Oy.

Verkkodokumentit

Helsingin yliopisto 2008. Oliosanasto [www-dokumentti]. [Viitattu 16.10.2008]. Saatavissa: http://www.cs.helsinki.fi/u/laine/oliosanasto/

Kuopion yliopisto 2008. Verkko-opetuksen tietotekniikkaa – Simulaatio opetuksessa [www-dokumentti]. [Viitattu 16.10.2008].
Saatavissa: http://www.cs.uku.fi/tutkimus/publications/reports/B-2004-3.pdf

NetBeans 2008. Ohjelmiston esittely [www-dokumentti]. [Viitattu 17.10.2008].
Saatavissa: http://www.netbeans.org/index_fi.html

Nieminen 2008. Gravitaation alainen liike [www-dokumentti]. [Viitattu 17.10.2008]
Saatavissa: http://www.kotiposti.net/ajnieminen/graval.pdf

Opetushallitus 2008. Fysiikan oppimateriaali, gravitaatio [www-dokumentti]. [Viitattu. 17.10.2008]
Saatavissa: http://www.oph.fi/etalukio/opiskelumodulit/fysiikka/mekaniikka/gravitaatio/heittoliike/index.html

Rovaniemen ammattikorkeakoulu 2009. Fysiikan harjoituksia [www-dokumentti]. [Viitattu 12.03.2009] Saatavissa: http://ta.ramk.fi/~jouko.teeriaho/fys1s2002_2.doc

Wikipedia 2008. OpenOffice.org [www-dokumentti]. [Viitattu 17.10.2008].
Saatavissa: http://fi.wikipedia.org/wiki/OpenOffice.org

Wikipedia 2008a. Vino heittoliike [www-dokumentti]. [Viitattu 17.10.2008]. Saatavissa: http://fi.wikipedia.org/wiki/Heittoliike

Wikipedia 2008b. Siniaalto [www-dokumentti]. [Viitattu 30.10.2008]. Saatavissa: http://fi.wikipedia.org/wiki/Siniaalto

Wikipedia 2008d. Kulmataajuus [www-dokumentti]. [Viitattu 13.03.2009]. Saatavissa: http://fi.wikipedia.org/wiki/Kulmataajuus

Wikipedia 2008c. Vaihe [www-dokumentti]. [Viitattu 13.03.2009]. Saatavissa: http://fi.wikipedia.org/wiki/Vaihe

LIITE 1 – Fysiikan simulaatiot internetissä

Tähän liitteeseen on kerätty internetsivustoja, joilta löytyy vastaavanlaisia simulaatioita, joita tämän opinnäytetyön aikana toteutettiin. Lisäksi liite sisältää lyhyen kuvauksen fysiikan simulaatioympäristöistä, joihin tutustuttiin opinnäytetyötä suunniteltaessa.

Simulaatiot

http://physics.uwstout.edu/PhysApplets/
Winsconsin yliopiston laaja kokoelma fysiikan simulaatio appletteja.

http://www.falstad.com/mathphysics.html
Useita fysiikan appletteja liittyen mm aaltoliikkeeseen, akustiikkaan ja magnetismiin.

http://www.walter-fendt.de/ph14e/
Laaja valikoima erilaisia fysiikan simulaatioita.

http://jersey.uoregon.edu/
Oregonin yliopiston fysiikan laitoksen oppimisympäristö joka sisältää useita fysiikan simulaatioita.

http://sun.ylojarvi.fi/java/pos/
Ylöjärven lukion oppilaan hyvin toteuttamia fysiikan appletteja.

http://www.myphysicslab.com/
Sivusto, jolle on koottu fysiikan simulaatioita, animaatiot ovat hyvin toteutettuja ja toimivia.

http://www.physics.purdue.edu/academic_programs/courses/applets.shtml
Kokoelma fysiikan simulaatioita.

Simulaatioympäristöt

Easy Java simulations

http://www.um.es/fem/EjsWiki/

Ohjelmisto, jolla voidaan luoda simulaatioita graafisesti. Ohjelmiston käyttö ei vaadi ohjelmointitaitoa. Tehdyistä simulaatioista generoidaan Java-appletteja. Ohjelmisto on tarkoitettu yksinkertaisten simulaatioiden luontiin ja ehkä enemmän opetuskäyttöön sellaisenaan kuin uusien simulaatioapplettejen toteuttamiseen. Ohjelmisto todettiin liian rajoittuneeksi tämän opinnäytetyön toteuttamiseen.

Feynman

http://feynman.sourceforge.net/

Java apuluokkia simulaatioiden toteuttamiseen. Ei kovin monipuolinen ja viimeinen versio vuodelta 2002. Tämä kirjasto ei olisi auttanut paljon opinnäytetyön simulaatioiden suunnittelussa.

Phys2d

http://www.cokeandcode.com/phys2d/

Javalla toteutettu 2D fysiikan mallinnusmoottori pelikehitystä varten. Sivuilta löytyy esimerkkejä, joissa moottoria on käytetty. Mielenkiintoinen projekti ja olisi voinut sopia opinnäytetyön alustaksi.

Open Source Physics

http://www.compadre.org/OSP/

Kokoelma erilaisia kirjastoja, ohjelmia ja ohjeita fysiikan simulaatioiden laatimista varten. Todella laaja valikoima toteutettuja simulaatioita. Tämä olisi ollut varteenotettava vaihtoehto simulaatioiden toteuttamisalustaksi.

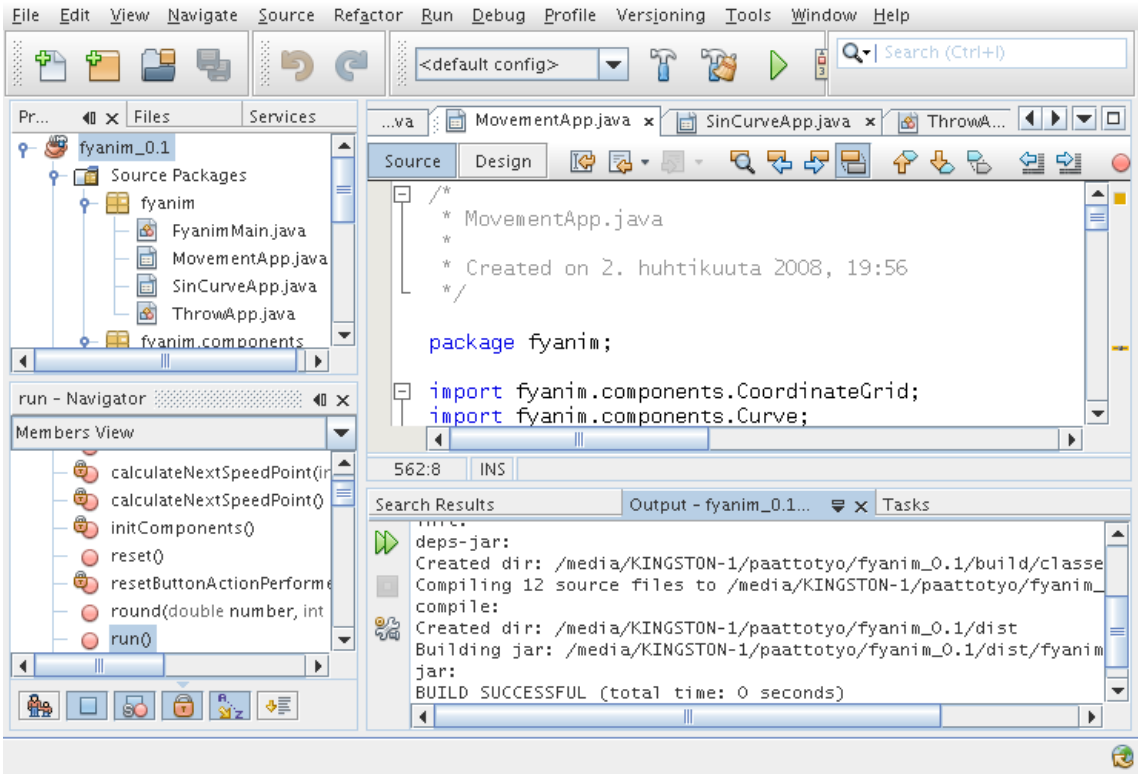
LIITE 2 – Ohjeita jatkokehittäjälle

Tässä luvussa on ohjeita jatkokehittäjälle. Ohjeissa käydään läpi uuden simulaation lisääminen Fyanim aplettiin ja tärkeimpien valmiiden komponenttien käyttöohjeet. Luvussa on myös hieman yleisiä ohjeita animaation ohjelmoimiseen, kuten säikeiden ja piirtorutiineiden käyttöä. Applettia on kehitetty NetBeans IDE:llä, joten myös sen käyttöä käydään pintapuolisesti lävitse.

JPanel luokan luominen projektiin uutta simulaatiota varten

Projektin avaamiseen tarvitaan Netbeans 5.5.1 tai uudempi versio, ohjelman voi ladata NetBeans-projektin kotisivuilta. Projektin kansio löytyy tämän opinnäytetyön mukana olevalta CD:ltä.

NetBeanssissa valitse File valikosta ”New File”-valinta ja avautuvasta ikkunasta Java Gui Forms-kansiosta löytyvä Jpanel Form-valinta. Seuraavaksi kysytään luokan nimeä. Javassa yleinen käytäntö on että luokan nimi alkaa isolla alkukirjaimella ja jos nimi sisältää useampia sanoja ne erotetaan tosistaan aloittamalla jokainen sana isolla kirjaimella. Fyanim-projektissa kaikki simulaatiopaneeleiden nimen loppuun on lisätty -App-liite. Esim. NewJavaAnimationApp. Ennen Finnish-painikkeen painamista tarkista vielä, että Package kohdassa on fyanim-paketti valittuna. NetBeans avaa luodun paneelin käyttöliittymänsuunnittelu näkymään, josta voi siirtyä lähdekoodi näkymään yläpaneelissa olevien painikkeiden avulla. Käyttöliittymänsuunnittelu näkymässä voit piirtää simulaatiolle käyttöliittymän, lähdekoodi näkymässä muokata lähdekoodia. Kuvassa 1 on Fyanim projektin MovementApp auki lähdekoodinäkymässä.



Kuva 9: NetBeansin lähdekoodinäkymä

Simulaation lisääminen valikkoon

Lisättäessä uutta simulaatiota applettiin on sille luotava valikkoelementti simulaatiot valikkoon. Uuden valikkoelementin tapahtumakuuntelu lisätään actionPerformed-metodiin ja StartSimulation-metodiin lisätään luodun simulaatio käynnistystoimenpiteet. Uudesta simulaatioluokasta on luotava ilmentymä, annettava sille koordinaatit , lisättävä se pääluokan säiliöön ja asettaa lisätty paneeli näkyväksi. Pääluokan lähdekoodissa on kommentit paikoissa joihin tulee muutoksia uutta luokkaa lisättäessä.

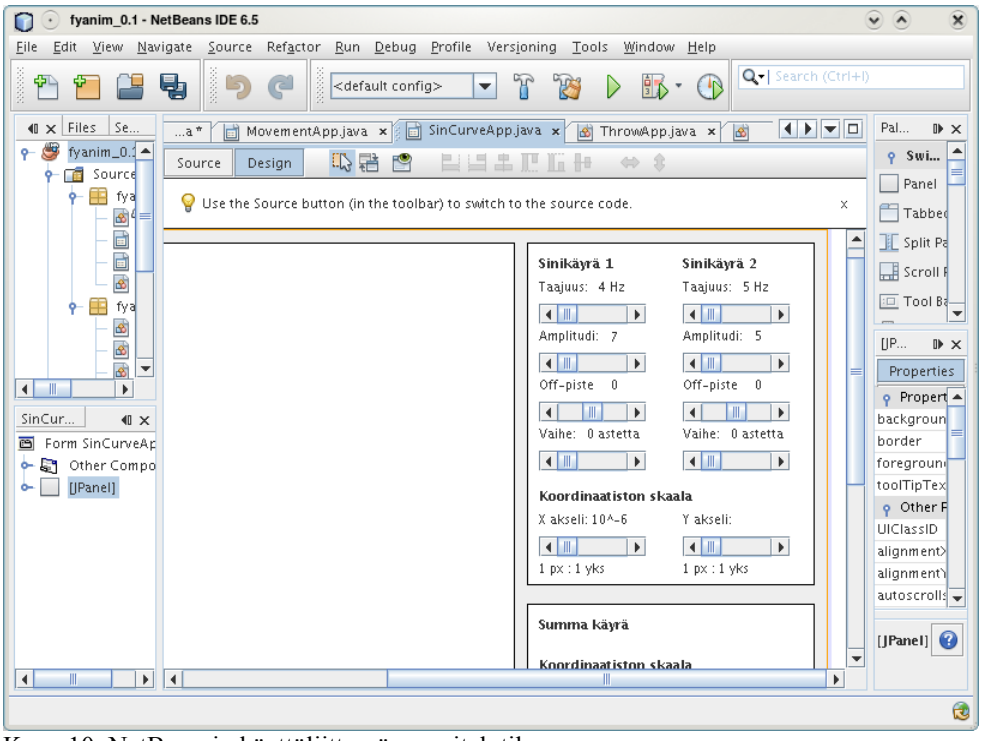
Käyttöliittymän suunnittelu desing tilassa

Käyttöliittymänsuunnittelu-näkymässä onnistuu käyttöliittymän piirtäminen ja

suunnittelu. Komponenttien lisääminen ja muokkaaminen tapahtuu valikoista, NetBeans generoi komponentin lisäämiseen ja muokkaamiseen tarvittavan koodin, koodia voi lukea, mutta ei muokata. Myös omien komponenttien lisääminen onnistuu graafisen käyttöliittymän kautta. Oman komponentin voi lisätä komponentti palettiin, jos sen periytymispuussa on JComponent-luokka.

Ensimmäiseksi simulaation pääpaneelin preferredSize, maximumSize ja minimumSize arvot. Simulaatioiden leveys on 800 pikseliä ja korkeus 600 pikseliä. Pääpaneeli on hyvä jaotella alipaneeleihin. Koska appletin koko ei muutu, voi paneelien asettelunhallinnan kytkeä pois päältä, tämä tapahtuu asettamalla asettelunhallinnan arvoksi null. Kun asettelunhallinta ei ole päällä asetetaan paneelien komponenteille koordinaatit sekä koot.

Komponentit, joita joudutaan käsittelemään koodista, on hyvä nimetä uudelleen. Komponenttien nimeämisessä kannattaa noudattaa käytäntöä jossa komponentin nimen alussa on lyhenne komponentin tyypistä. Esimerkiksi tekstikenttäkomponentit voisi aloittaa ”txt” lyhenteellä. Kuvassa 2 on NetBeans käyttöliittymänsuunnittelu tilassa.



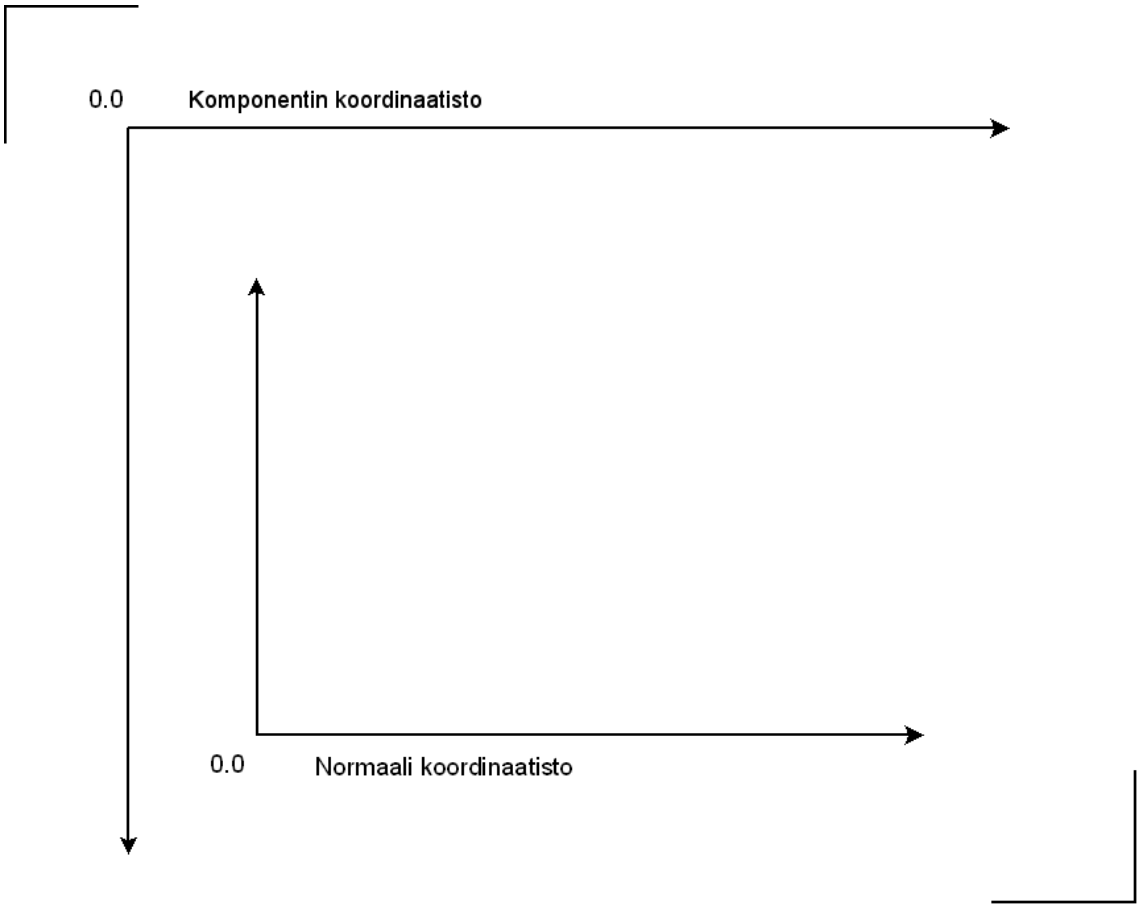
Kuva 10: NetBeansin käyttöliittymäsuunnitelutila

Komponentin pinnalle piirtäminen

Simulaatioita tehdessä on usein tarpeellista päästä piirtämään komponenttien pinnalle, kuten esimerkiksi paneelien pinnalle. Swing-komponenttien pinnalle piirtäminen on mahdollista ylikirjoittamalla komponentin paintComponent-metodi. PaintComponent-metodissa käytetään parametrina tullutta Graphics-oliota, jonka voi laajentaa tyyppimuunnoksella Graphigs2D tyyppiseksi. Graphics2D-luokka sisältää Javan kehittyneemmät piirto-ominaisuudet. Kun komponentin piirtopinta pitää päivittää kutsutaan komponentin repaint()-metodia. PaintComponent-metodia ei ole hyvä kutsua suoraan. Alla olevassa listauksessa on esimerkkitoteus JPanel-komponentin pinnalle piirtämisestä.

```
public class TestiPaneeli extends javax.swing.JPanel{

   private BufferedImage buffer;
   private Graphics2D osg;

   public MotionLine(){
      buffer = new BufferedImage(width, height,
                              BufferedImage.TYPE_INT_RGB);

      osg = buffer.createGraphics();
   }

   public void paintComponent(Graphics g){

      super.paintComponent(g);
      g.drawImage(buffer, 0,0,this);
   }
```

Piirtäessä komponentin pinnalle on otettava huomioon että piste (0,0) on komponentin oikeassa yläreunassa eikä oikeassa alareunassa niin kuin normaalissa koordinaatistossa. Kuva 3 havainnollistaa komponentin ja normaali koordinaatiston eroa.



Kuva 11: Komponentin koordinaatisto

Runnable rajapinta

Useimmissa simulaatioissa on jonkinlaista animointia. Animaatio on hyvä toteuttaa omassa säikeessä jotta se ei häiritse muun ohjelman toimintaa. Säiettä käyttävä luokka toteutetaan Runnable-rajapinnalla. Rajapinta otetaan käyttöön implements avainsanalla luokan määrittelyn yhteydessä.

Runnable rajapinnassa on yksi abstrakti metodi, joka on pakko toteuttaa. Metodin nimi on run. Run-metodissa tapahtuu säikeen toiminta. Tyypillisesti run-metodi sisältää silmukan, joka sisältää animaation askelluksen. Säikeen pysäytys voidaan hoitaa muuttamalla silmukan ehto epätodeksi. Alla olevassa listauksessa on esimerkki luokasta, joka käyttää säiettä ja on tyypillinen animaatiosilmukka.

```
public class MovementApp extends javax.swing.JPanel
                         implements Runnable{

   private boolean running;
   private Thread main;

   public MovementApp() {
      running = true;

      main = new Thread(this);
      main.start();
   }

   public void run() {

      while(running){
         //Askella animaatiota

         //Päivitä piirtopinta
         repaint();

         if(/*ehto jolloin animaatio lopetetaan*/){
            running = false;
         }

         //Animaation päivitysnopeus
         try {
            main.sleep(25);
         } catch (InterruptedException ex) {
            ex.printStackTrace();
         }
      }
   }
}
```

Esimerkissä säie pysähtyy jonkin silmukassa tapahtuvan ehdon tullessa voimaan. Usein säie on tarpeellista pysäyttää myös jonkin tapahtuman kautta, kuten esimerkiksi painikkeen painalluksesta. Tämä onnistuu muuttamalla painikkeen tapahtumakuuntelijassa running muuttujan arvoksi false, se on mahdollista koska muuttuja on luokkamuuttuja.