

Ohjelmitava sadetin

Optimaalisen kastelulaitteen suunnittelu ja
analysointi

Eric Malmi
Valkeakosken lukio
Ympäristötekniikka

30.11.2006

Tiivistelmä

Vesi on nykyään hyvin kriittinen luonnonvara, minkä vuoksi tehokaiden ja vettä säästävien kastelujärjestelmien suunnittelu on tärkeää. Tutkielman aiheena on ohjelmoitava sadetin, jolla voidaan kastella halutun muotoisia alueita. Sadettimen kastelukuviosta saadaan halutunlainen säätämällä sadettimen kastelusädetä sadettimen pyöriessä. Säteeeseen vaikutetaan muuttamalla veden lähtökulmaa tai virtausnopeutta.

Ohjelmoitavasta sadettimesta tehtiin malli, jota vertailtiin perinteisiin sadettiin luomalla tietokoneohjelma. Tietokoneohjelma tutki veden säästöä, veden osumista kasteltavalle alueella ja kastelun tasaisuutta kolmessa esimerkkitapauksessa: pellolla, kotipihalla sekä golfkentällä. Ohjelmalla saatujen tulosten pohjalta arvioitiin ohjelmoitavalla sadettimella saavutettavia taloudellisia säästöjä.

Tuloksena saatiin, että ohjelmoitavista sadettimista koostuvalla kastelujärjestelmällä voitaisiin saavuttaa hyvinkin merkittäviä parannuksia veden käytössä, veden osumisissa ja kastelun tasaisuudessa. Esimerkiksi golfkentällä saavutettiin 31% veden säästö. Esimerkkigolfkentällä (Aura Golf) ohjelmoitavilla sadettimilla voitaisiin siten saavuttaa jopa 10 000 euron vuosittainen säästö kastelukustannuksissa.

Sisältö

Tiivistelmä	2
Sisältö	3
1 Johdanto	4
2 Käytössä olevat sadettimet	5
3 Ohjelmoitavan sadettimen perusidea	6
3.1 Kastelukuvion säätäminen	6
3.2 Kastelukuvion tasaisuus	6
4 Malli	7
4.1 Oletukset	7
4.2 Veden jakaantuminen kastelukuviolle	8
4.3 Perinteinen ja ohjelmoitava sadetin	8
5 Ohjelma	10
5.1 Yleistä	10
5.2 Mitattavat suureet	10
6 Esimerkkitapaukset	11
6.1 Tapaus 1: Pelto	12
6.2 Tapaus 2: Kotipiha	14
6.3 Tapaus 3: Golfkenttä	16
7 Tulokset	18
7.1 Veden käyttö	18
7.2 Taloudellinen vaikutus	18
8 Jatkokehitys	19
Lähdeluettelo	20
A Lähdekoodi	a
A.1 Luokka Sadetin	a
A.2 Luokka KenttaSovellus	c
A.3 Luokka KastelunArviointi	g

1 Johdanto

Vesi on yksi ihmiskunnan tärkeimmistä elinehdoista. Peltojen, kotipihojen ja muiden nurmialueiden kasteluun kuluu huomattavia määriä vettä – etenkin kuivilla seuduilla. Tämän vuoksi kastelulaitteiden optimointi on tärkeää veden säästämiseksi.

Kastelun suunnittelussa pyritään saavuttamaan mahdollisimman tasainen kastelu ja veden tulee osua mahdollisimman hyvin kasteltavalle alueelle [10]. Tasaisuuteen ja veden osumiseen vaikuttavat kastelulaitteiden sijoittelu sekä niiden kastelukuviot.

Kiinnostuksen aiheeseen sain tämän vuoden alussa, kun osallistuimme kahden luokkatoverini kanssa kansainväliseen korkeakoulutasoiseen matemaattiseen mallinnuskilpailuun MCM:ään (The Mathematical Contest in Modeling). Kilpailussa oli tehtävänä tutkia käsin siirrettävän kastelujärjestelmän asetelua ja siirtämistä $80m \times 30m$ pellolla (”Positioning and Moving Sprinkler Systems for Irrigation”) [3]. Kilpailututkimusta tehdessä huomasin, kuinka hankalaa ympyräkuviollisilla sadettimilla on saavuttaa tasainen kastelu suorakaiteen muotoiselle kentälle.

Tämän tutkielman tarkoituksena on suunnitella ja luoda malli ohjelmoitavasta sadettimesta, jolle voidaan määritellä kastelukuvion muoto, ja jonka pyörimisnopeutta voidaan säätää. Tätä mallia vertaillaan perinteisiin ympyräkuviollisiin sadettimiin luomalla tietokoneohjelma, joka simuloi kumpaakin sadetinta. Ohjelma vertailee hukkaan menneen veden määrää, veden osumista kasteltavalle alueelle sekä kastelun tasaisuutta kolmessa esimerkkitapauksessa. Tulosten pohjalta arvioidaan myös ohjelmoitavalla sadettimella saavutettavia taloudellisia säästöjä.

Markkinoilla on ainakin yksi sadetinvalmistaja, Gilmour, joka valmistaa sadettimia, joille voidaan määritellä kastelukuvion muoto. Valmistajan kotisivuilta [5] löytyy kuitenkin huonosti tietoja näiden sadettimien ominaisuuksista. Gilmourin sadettimet poikkeavat suunnittelemani sadettimista ainakin siten, että Gilmourin sadettimet eivät sisällä lainkaan elektroniikkaa.

Elektroniikan lisäämisellä taataan portaaton kastelukuvion muokkaus ja mahdollistetaan sadettimen pyörimisnopeuden vaihtelu. Sekä kastelukuvion muotoa ja pyörimisnopeutta voidaan myös säätää kauko-ohjatusti ja sadettimen toimiessa. Tämä mahdollistaa esimerkiksi sen, että **sadetin voidaan ohjelmoida sopeutumaan ympäristöolojen muutoksiin, kuten tuulen vaihteluun**. Elektroniikka tuo lisäkustannuksia, mutta elektroniikka on kuitenkin selvästi lisääntymässä ja näin ollen halpenemassa kaikissa ihmisten apuvälineissä.

2 Käytössä olevat sadettimet

Yleisimmät sadetintyyppit ovat: ympärilleen suihkuttava sadetin (spray type sprinkler), heiluva sadetin (oscillating sprinkler) ja pyörivä sadetin (rotor sprinkler).

Ympärilleen suihkuttavissa sadettimissa vesi suihkuu samanaikaisesti jokapuolelle ympäri sadetinta. Nämä sadettimet ovat rakenteeltaan hyvin yksinkertaisia, mutta niiden kastelualueet ovat melko pieniä.



Kuva 1: Ympärilleen suihkuttava sadetin. [7]

Heiluvassa sadettimessa suihku menee edestakaisin, ja näin saavutetaan suorakaiteen muotoinen kastelukuvio.



Kuva 2: Heiluva sadetin. [8]

Pyörivissä sadettimissa vesisuihku lentää yhdestä kohtaa suutinta suuttimen pyöriessä. Näin muodostuu ympyrän muotoinen kastelukuvio. Pyörivät sadettimet ovat yleisimpiä edellä mainituista tyypeistä, ja niitä käytetään sekä kotipihojen että suurempien alueiden, kuten peltojen ja golfkenttien kastelussa. Pyörivien sadettimien suurin etu on se, että yhdellä sadettimella voidaan kastella suuriakin alueita, ja näin ollen laajoillakaan kastelualueilla ei tarvita niin useita sadettimia.

Suurimman ongelman muodostaa kuitenkin pyörivän sadettimen ympyrän muotoinen kastelukuvio. Esimerkiksi golfkenttää kasteltaessa ympyrän muotoinen kastelukuvio aiheuttaa ongelman, kun sadettimia sijoitellaan hiekkasteiden ympärille; vettä ei saisi mennä hiekalle, koska silloin vettä menee

hukkaan ja hiekasta tulee mutaista; toisaalta hiekkaestettä ympäröivän nurmikon pitäisi saada tasaisesti vettä eikä minnekään saisi jäädä kuivia kohtia. Myös kotipihat ovat yleensä muun kuin ympyrän muotoisia, ja siksi niissäkin vettä joko menee pihan ulkopuolelle tai kaikki pihan osat eivät saa tasaisesti vettä.

3 Ohjelmoitavan sadettimen perusidea

Tutkielman tarkoituksena on luoda malli ohjelmoitavasta sadettimesta, jonka idea perustuu pyörivään sadetimeen. Ohjelmoitavalle sadettimelle voidaan määritellä kastelukuvion muoto, jolloin pystytään kastelemaan tasaisesti hyvinkin epäsäännöllisen muotoisia alueita.

3.1 Kastelukuvion säätäminen

Kastelukuviota säädetään muuttamalla sadettimen kastelusädettä samaan aikaan, kun suutin pyörii. Kastelusädettä voidaan muuttaa ainakin kahdella eri tavalla:

- Muuttamalla veden lähtökulmaa
- Muuttamalla veden virtausnopeutta

Vesi lentää sitä pidemmälle, mitä suurempi veden lähtökulma on, kunhan kulmaa ei nosteta yli 45 asteen ja kun tuulen vaikutusta ei huomioida.

Kastelusädettä voidaan kasvattaa myös nostamalla veden virtausnopeutta. Virtausnopeutta saadaan muutettua säätämällä sadettimen suuttimen halkaisijan kokoa. Virtauman tulee pysyä vakiona.

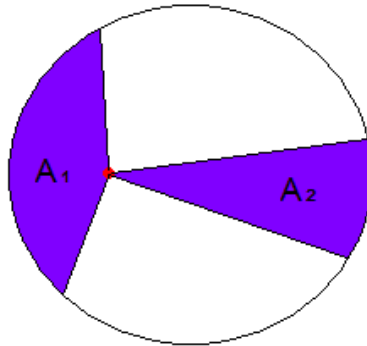
Vettä voidaan siis teoriassa lennättää välille $u = [0, \text{maksimisäde}] \subseteq \mathbb{R}$. Ohjelmoitavalla sadettimella voidaan näin muodostaa esimerkiksi tähden muotoinen kastelukuvio nostamalla ja laskemalla suuttimen kulmaa tai pienentämällä ja suurentamalla suuttimen halkaisijaa vuoronperään.

Perinteisissäkin sadettimissa virtausnopeutta pystytään säätämään vaihtamalla sadetimeen erilainen suutin. Ohjelmoitavassa sadettimessa virtausnopeutta tulee pystyä säätämään portaattomasti, joten suuttimen vaihto ei onnistu, vaan suuttimessa tulee olla jokin automaattinen mekanismi, joka säätelee suuttimen halkaisijaa. Lähtökulman muuttaminen saattaakin olla teknisesti helpompi ratkaisu. Teoriassa kummallakin ratkaisutavalla saavutetaan kuitenkin samanlainen kastelu. Tutkielma keskittyy vertaamaan tätä kastelua perinteisten sadettimien kasteluun, eikä niinkään paneudu ohjelmoitavan sadettimen tekniseen toteutukseen.

3.2 Kastelukuvion tasaisuus

Mitä pidemmälle vettä suihkutetaan, sitä suuremmalle alueelle se jakaantuu. Jos vettä kuitenkin virtaa suuttimesta vakiomäärä aikayksikköä kohti, niin

sadettimen pyörimisnopeutta (kulmanopeutta) täytyy muuttaa, jotta vesi jakaantuisi tasaisesti. Kuva 3 havainnollistaa asiaa:



Kuva 3: Esimerkki ohjelmoitavan sadettimen kastelukuviosta, jossa kastelusäde vaihtelee.

Kuvassa 3 tummennetut pinta-alat A_1 ja A_2 ovat yhtä suuret, mutta niitä vastaavat keskuskulmat ovat erisuuret. Jotta kummallekin alueelle kasaantuisi yhtä paljon vettä, täytyy sadettimen pyörähtää alueiden yli yhtä nopeasti, kun oletetaan, että veden virtaama pysyy vakiona. A_1 -sektorin keskimääräisen kulmanopeuden täytyy siis olla suurempi kuin A_2 -sektorin, koska A_1 -sektoria vastaava keskuskulma on suurempi.

Kulmanopeuden muuttuminen vastaa Keplerin toista lakia, jonka mukaan planeetat pyyhkäisevät yhtä suuren sektoripinta-alan aurinkoon nähden yhtä pitkässä ajanjaksossa. Keplerin toinen laki koskee vain elliptisiä ratoja, mutta perusajatus siinä ja ohjelmoitavan sadettimen pyörimisnopeuden muuttumisessa on sama.

4 Malli

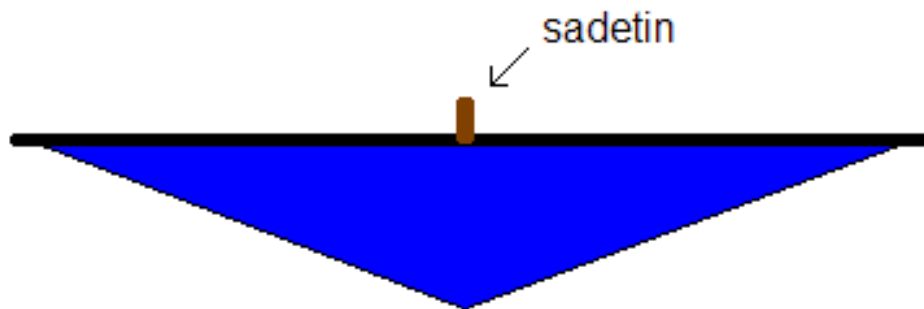
4.1 Oletukset

Jotta tietokoneohjelmaa varten luotava malli pysyisi riittävän yksinkertaisena, seuraavat asiat oletetaan:

- Mahdollinen tuuli ei vaikuta veden jakaantumiseen. (Todellisuudessa tuulella saattaa olla hyvinkin merkittävä vaikutus. Mielenkiintoinen jatkotutkimuksen kohde olisikin sellaisen ohjelman tekeminen, joka ottaa tuulen vaikutuksen huomioon.)
- Kasteltava alue on tasainen, eli mäkiä ja kuoppia ei huomioida.
- Vettä kasaantuu pinta-alayksikköä kohden sitä enemmän, mitä lähempänä sadetinta ollaan (kts. kohta 4.2)
- Kaikki kasteltavan alueen osat vaativat yhtä paljon vettä. (Käytännössä siis alueella ei ole esimerkiksi kohtia, johon paistaa voimakkaasti aurinko, ja jotka näin ollen vaatisivat enemmän kastelua.)

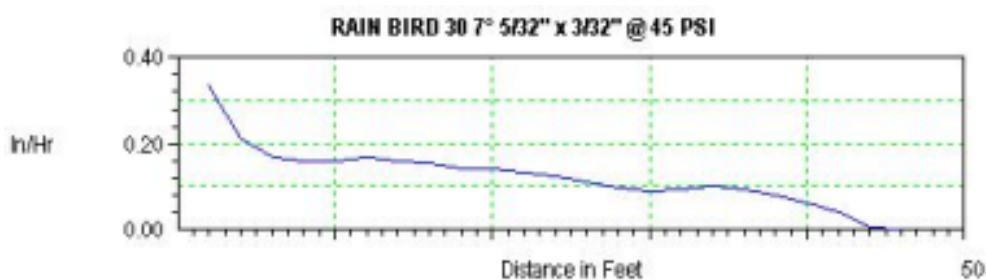
4.2 Veden jakaantuminen kastelukuvioille

Mallissa oletetaan, että vettä kasaantuu pinta-alayksikköä kohden sitä enemmän, mitä lähempänä sadetinta ollaan. Muun muassa Yhdistyneiden kansakuntien elintarvike- ja maatalousjärjestön FAO:n julkaisemassa artikkelissa "Irrigation methods" [4] esitetään pyöriville sadettimille tällainen veden jakaantuminen. Kuva 4 havainnollistaa jakaantumista.



Kuva 4: Kastelukuvio sivustapäin katsottuna.

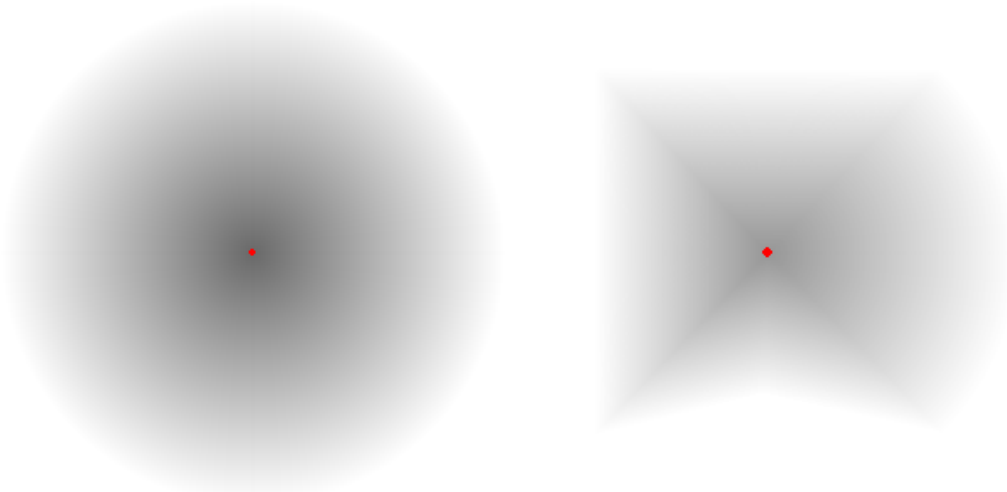
Oletusta tukee myös CIT:n (Center for Irrigation Technology) tekemä testaus [2], jossa kartoitetaan kaupallisten sadettimien tarkkoja kasteluprofileita. Kuvassa 5 on esimerkkinä Rain Birdin "30 7" -mallin kastelukuvio, joka vastaa likimäärin oletusta.



Kuva 5: Rain Birdin "30 7" -mallin kastelukuvio sivustapäin. [9]

4.3 Perinteinen ja ohjelmoitava sadetin

Perinteisillä sadettimilla kastelukuvio on ympyrän muotoinen, kun taas ohjelmoitavilla sadettimella se voidaan määrittellä.



Kuva 6: Esimerkki yksittäisen perinteisen ja yksittäisen ohjelmoitavan sadettimen kastelukuviosta. Mitä tummempi alue sitä enemmän vettä.

Yllä olevan kaltainen veden jakaantumista kuvaava kastelukuvio muodostetaan seuraavasti:

- Jaetaan kasteltava alue ruutuihin.
- Lasketaan jokaiselle ruudulle numeroarvo, joka ilmoittaa ruutuun kertyvän veden määrän.
- Väritetään sadettimen kastelukuvion sisällä olevat ruudut siten, että mitä suurempi numeroarvo ruudussa on sitä tummempi väri ruutuun tulee.

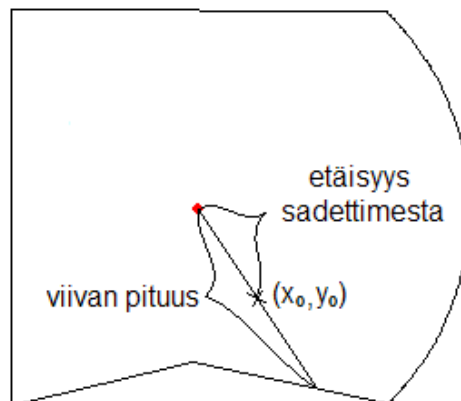
Yksittäisen ruudun (x_0, y_0) numeroarvon laskeminen tapahtuu seuraavasti:

$$arvo = \left(1 - \frac{\text{etäisyys sadettimesta}}{\text{viivan pituus}}\right) \cdot (\text{maksimivesimäärä}), \text{ missä}$$

etäisyys sadettimesta = pisteen (x_0, y_0) etäisyys sadettimesta

viivan pituus = katso kuva 7

maksimivesimäärä = se vesimäärä, joka kertyy aivan sadettimen viereen



Kuva 7: Vesimäärän numeroarvon laskeminen.

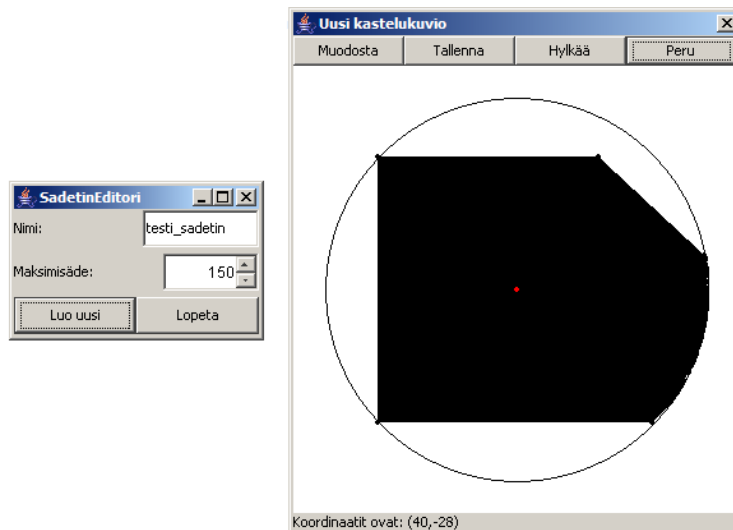
5 Ohjelma

5.1 Yleistä

Ohjelman ideana on pystyä helposti mallintamaan satunnaista kastelualueita ja tutkimaan, kuinka hyvä kastelu tälle alueelle saavutetaan perinteisillä ja ohjelmoitavilla sadettimilla.

Ohjelma on toteutettu Java-ohjelmointikielellä, ja se koostuu kolmesta osasta: SadetinEditori, KenttaEditori ja KenttaSovellus.

SadetinEditorilla ja KenttaEditorilla luodaan käytettävät sadettimet ja käytettävä kenttä. Luominen tapahtuu valitsemalla ensin kentän tai sadettimen koko ja piirtämällä sitten rajat painelemalla tai raahaamalla hiirtä (kuva 8). Luotu sadetin/kenttä voidaan tallentaa, jolloin sitä on helppo käyttää myöhemmin.



Kuva 8: Kuvakaappaus SadetinEditorista.

KenttaSovelluksella suoritetaan varsinainen simulointi. Aluksi ladataan haluttu kenttä ja halutut sadettimet sekä asetellaan sadettimet kentälle omille paikoilleen. Sen jälkeen ohjelma piirtää kuvan veden jakaantumisesta kentälle ja tulostaa arvot kastelun laadusta komentoriville.

5.2 Mitattavat suureet

5.2.1 Tasaisuus

Tasaisuutta mitataan J.E. Christiansenin tasaisuuskertoimella (CU) [12]:

$$CU = \left(1 - \frac{\sum |x_i - x_{\text{keskiarvo}}|}{n \cdot x_{\text{keskiarvo}}}\right) \cdot 100\%, \text{ missä}$$

- x_i = veden määrä yksittäisessä kentän ruudussa
- $x_{\text{keskiarvo}}$ = kentän sisällä olevien ruutujen vesimäärien keskiarvo
- n = kentän sisällä olevien ruutujen lukumäärä

5.2.2 Ulkopuolelle mennyt vesi

Ulkopuolelle menneen veden osuus (x) kertoo, kuinka suuri osa käytetystä vedestä on mennyt hukkaan kentän ulkopuolelle.

$$x = \frac{V_u}{V_u + V_o}, \text{ missä}$$

V_u = ulkopuolelle menneen veden määrä

V_o = osuneen veden kokonaismäärä

Ulkopuolelle mennyt vesi ja osunut vesi saadaan laskettua seuraavalla algoritmilla:

Algoritmi 1 Veden osuminen

$V_u = 0$

$V_o = 0$

for all ruudut **do**

if *ruutu_iOnKentanSisalla* **then**

$V_o = V_o + ruutu_i$

else

$V_u = V_u + ruutu_i$

end if

end for

5.2.3 Veden säästö

Simulaatiota ajetaan kummallakin sadetintyyppillä niin kauan, kunnes kentän jokainen osa on saanut riittävän vesimäärän (V). Sitten lasketaan, kuinka paljon kaiken kaikkiaan vettä tämän vähimmäisvesimäärän saavuttamiseksi on kulunut perinteisiltä ja kuinka paljon ohjelmoitavilta sadettimilta. Veden kokonaismäärä on sitä suurempi mitä enemmän vettä on mennyt turhaan kentän ulkopuolella ja mitä enemmän vettä on kertynyt turhaan jo valmiiksi riittävän kosteille paikoille. Ohjelmoitavilla sadettimilla saavutettava säästetyn veden osuus on:

$$x_{\text{säästöosuus}} = \frac{V_p - V_o}{V_p}$$

V_p = perinteisillä sadettimilla kulunut vesimäärä

V_o = ohjelmoitavilla sadettimilla kulunut vesimäärä

6 Esimerkkitapaukset

Ohjelmoitavaa ja perinteistä sadetinta vertaillaan tietokoneohjelman avulla kolmessa esimerkkitapauksessa: pelto, kotipiha ja golfkenttä.

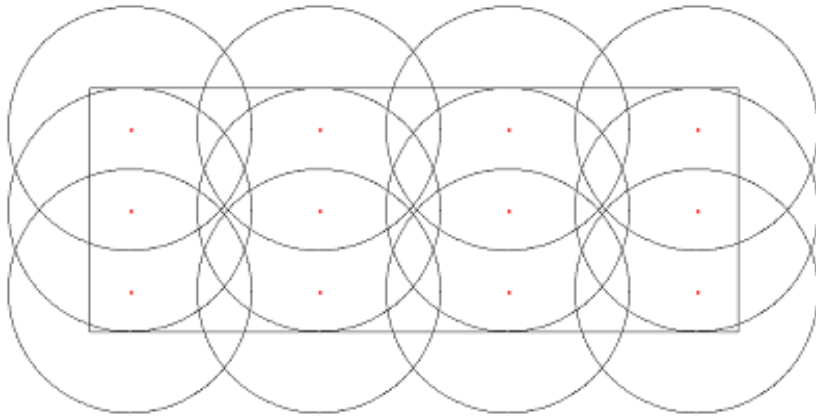
6.1 Tapaus 1: Pelto

MCM-kilpailussa oli tarkoitus tutkia, kuinka ympyräkuviollisia sadettimia kannattaa asettaa pellolle siten, että kastelu olisi mahdollisimman tasainen. Kilpailun tehtävänannossa määriteltiin, että pellon koko on $80m \times 30m$, ja että sadettimet ovat siirrettävässä putkessa kiinni, joten niiden täytyy olla samassa linjassa [3]. Tehtävänannossa annetuista tiedoista saimme sadettimien kastelukuvion säteeksi 30 metriä.

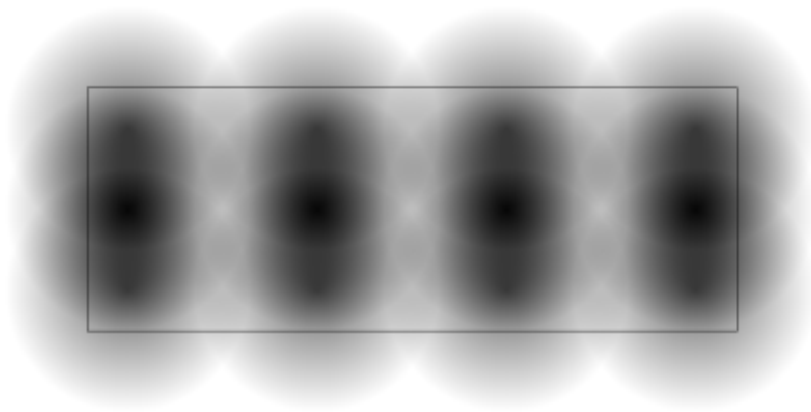
Saimme kaksi mahdollista ratkaisumallia sadettimien asetteluun, joista tässä tutkitaan ensimmäistä [6]. Vaihdataan sadettimet ohjelmoitaviin sadettiin ja määritetään niille käsin mahdollisimman optimaaliset kastelukuviot. Sen jälkeen vertaillaan pellon kastelun onnistumista perinteisillä ja ohjelmoitavilla sadettimilla.

6.1.1 Pelto – perinteisillä sadettimilla

Kuvista 9 ja 10 huomataan, että perinteisillä sadettimilla joudutaan suihkuttamaan vettä runsaasti kentän ulkopuolelle, jotta kaikki kentän osat saisivat riittävästi vettä. Sen lisäksi vettä kasaantuu huomattavan paljon sadettimien lähiympäristöön.



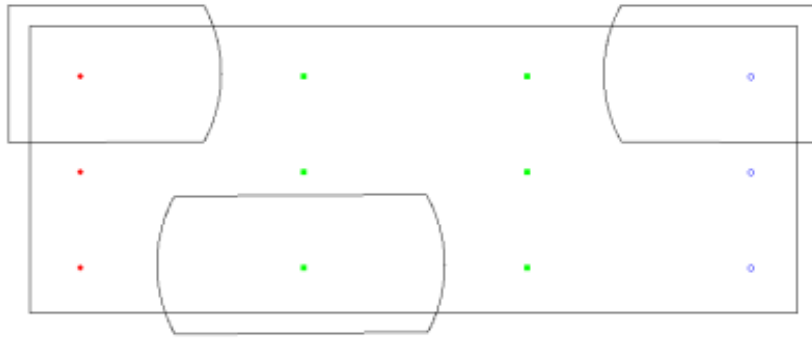
Kuva 9: Perinteisten sadettimien kastelukuviot. Pisteet ilmaisevat sadettimien paikat.



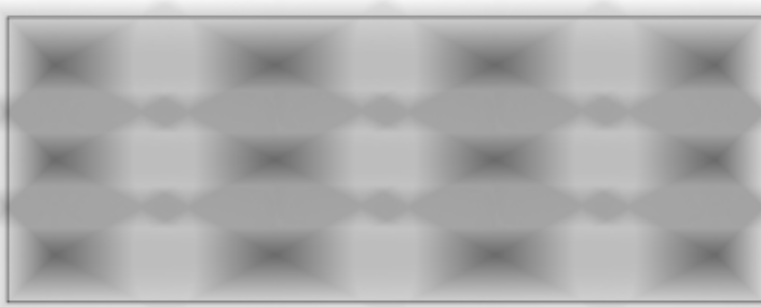
Kuva 10: Veden jakaantuminen pellolle perinteisillä sadettimilla. Mitä tummempi alue sitä enemmän vettä.

6.1.2 Pelto – ohjelmoitavilla sadettimilla

Peltoa varten luodaan ohjelmoitavia sadettimia, jotka käyttävät kolmea erilaista kastelukuviota. Kastelukuviot on määritelty käsin käyttämällä omaa päättelyä ja kokeilemalla eri vaihtoehtoja. Näillä kastelukuvioilla pyritään erityisesti välttämään sitä, ettei vettä menisi turhaan kentän ulkopuolelle, ja ettei vesi kasaantuisi niin merkittävästi sadettimien kohdalle kuin perinteisillä sadettimilla.



Kuva 11: Ohjelmoitavien sadettimien kastelukuviot. Pisteet ilmaisevat sadettimien paikat.



Kuva 12: Veden jakaantuminen pellolle ohjelmoitavilla sadettimilla. Mitä tummempi alue sitä enemmän vettä.

6.1.3 Vertailu

Ohjelmoitavilla sadettimilla saavutettava veden säästö on 49%. Erot perinteisten ja ohjelmoitavien sadettimien välillä kasvavat huomattaviksi, sillä sadettimien säteet ovat niin suuret. Perinteisillä sadettimilla tämä tarkoittaa väistämättä sitä, että vettä joudutaan ampumaan runsaasti kentän ulkopuolelle. Kastelun epätasaisuutta lisää se, että sadettimien täytyy olla tehtävänannon rajoitteiden vuoksi samassa linjassa.

	Perinteinen sadetin	Ohjelmoitava sadetin
Tasaisuus (CU)	67.9%	81.2%
Ulkopuolelle mennyt vesi	23.3%	5.6%

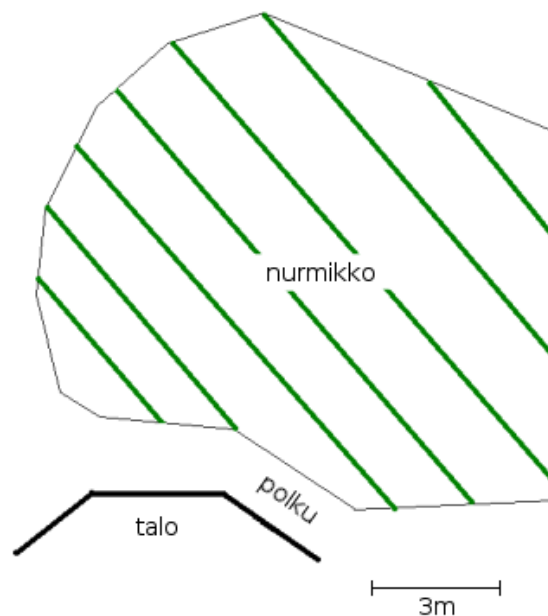
Taulukko 1: Pelto-tulosten vertailu

6.2 Tapaus 2: Kotipiha

Asumme paritalossa, jonka pihalle istutimme viime kesänä uuden nurmikon piharemontin yhteydessä. (Kuva 13)



Kuva 13: Valokuva pihasta.

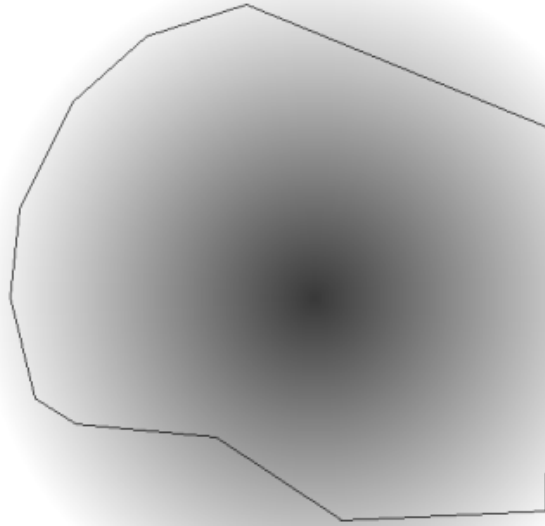


Kuva 14: Kaaviokuva pihasta.

Keskikesällä pihalle ilmestyi kuivuuden vuoksi kellertäviä alueita, joten päätimme hankkia pihalle sadettimen. Sadettimesta oli apua, mutta pihan muodon takia yhdellä sadettimella on erittäin hankalaa saavuttaa tasaista kastelua pihalle. Tämän vuoksi jouduimme siirtämään sadetinta vähän väliä.

6.2.1 Kotipiha – perinteisellä sadettimella

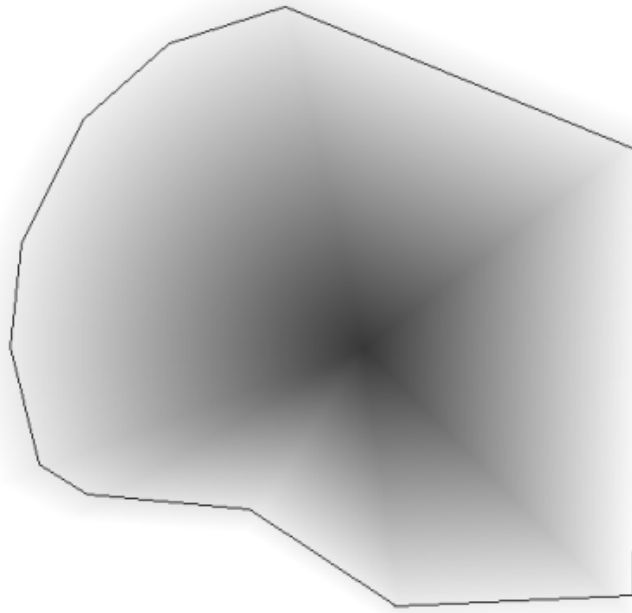
Tietokonemallissa pihalle asetetaan yksi perinteinen sadetin, joka yltää kentän joka puolelle. Todellisuudessa tämä ratkaisu on mahdoton toteuttaa, sillä vesisuihku lentäisi talolle asti. Piha vaatisi siis useampaa sadetinta tai yhden sadettimen siirtelemistä. Mallissa käytetään kasteluun kuitenkin vain yhtä sadetinta, jotta malli pysyisi riittävän yksinkertaisena.



Kuva 15: Veden jakaantuminen kotipihalle perinteisellä sadettimella. Mitä tummempi alue sitä enemmän vettä.

6.2.2 Kotipiha – ohjelmoitavalla sadettimella

Ohjelmoitavan sadettimen kastelukuvio on määritelty pihan rajojen mukaisesti. Kotipihalla ohjelmoitava sadetin toimii erityisen hyvin, sillä piha voidaan kastella kokonaisuudessaan yhdellä sadettimella – ilman että vettä menee merkittävästi pihan ulkopuolelle.



Kuva 16: Veden jakaantuminen kotipihalle ohjelmoitavalla sadettimella.
Mitä tummempi alue sitä enemmän vettä.

6.2.3 Vertailu

Ohjelmoitavilla sadettimilla saavutettava veden säästö on 27%. Vesi osuu huomattavasti paremmin nurmikolle, mikä on merkittävää taloudellisen säästön vuoksi, ja koska viereiselle kävelytielle joutuessaan vesi saattaa tehdä tiestä mutaisen ja huonon kävellä sekä häiritä ohikulkijoita. Esimerkkitapauksessa perinteisen sadettimen ongelmana on myös se, että talorakennus on hyvin lähellä nurmikkoa. Kun käytetään vain yhtä suurisäteistä sadetinta, vesi lentää osittain talon seinille.

	Perinteinen sadetin	Ohjelmoitava sadetin
Ulkopuolelle mennyt vesi	15.4%	2.3%

Taulukko 2: Kotipiha-tulosten vertailu

Tasaisuutta ei ole tutkittu, koska käytössä on vain yksi sadetin.

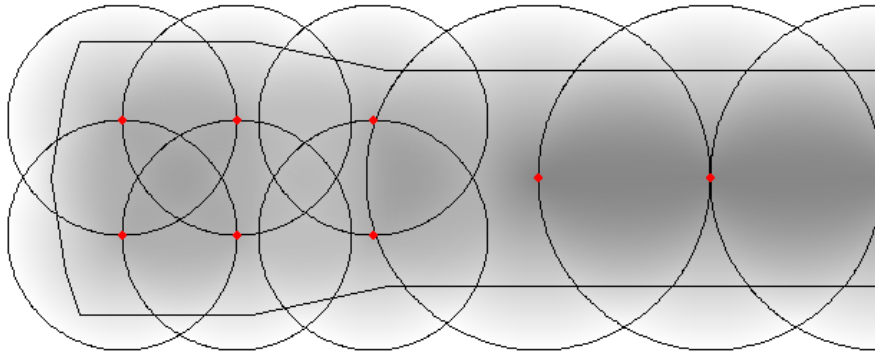
6.3 Tapaus 3: Golfkenttä

Golfkentillä hyvä kastelu on erittäin tärkeää, sillä kastelu määrää, miltä kenttä näyttää. Hyvin hoidetulta näyttävä kenttä houkuttelee asiakkaita ja tekee pelaamisesta miellyttävämpää.

6.3.1 Golfkenttä – perinteisillä sadettimilla

Richard Choaten ja Jim Watkinsin teos ”Turf Irrigation Manual” (1994) [1] antaa esimerkin yleisestä golfkentän yksittäisestä väylästä ja havainnollistaa, miten sadettimet tulisi asetella väylälle. Käytetään tätä mallia ja tutkitaan,

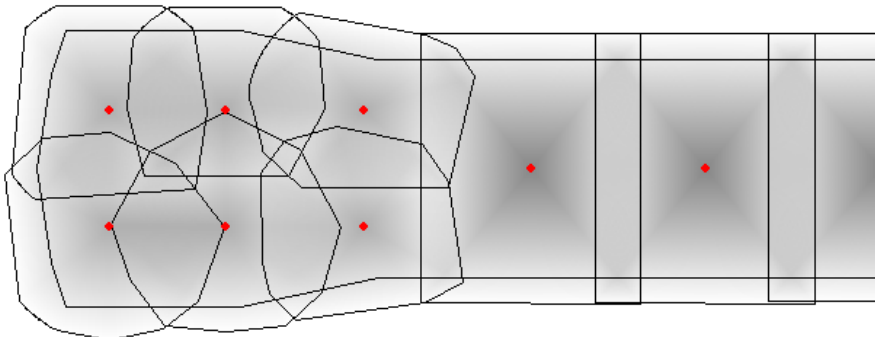
millainen kastelu sillä saavutetaan. Kuvassa 17 on havainnollistettuna kirjan esittämä sadettimien asettelu. Kuvassa näkyy myös ohjelman mallintama veden jakaantuminen.



Kuva 17: Veden jakaantuminen golfkentälle perinteisillä sadettimilla. Mitä tummempi alue sitä enemmän vettä.

6.3.2 Golfkenttä – ohjelmoitavilla sadettimilla

Asetellaan ohjelmoitavat sadettimet väylälle samalla tavoin kuin perinteiset sadettimet. Ohjelmoitaville sadettimille yritetään löytää mahdollisimman optimaaliset kastelukuviot määrittelemällä kastelukuviot käsin käyttämällä omaa päättelyä. Kastelukuvioiden määrittelyssä pyritään ensisijaisesti saavuttamaan mahdollisimman vähäinen vedenkulutus.



Kuva 18: Veden jakaantuminen golfkentälle ohjelmoitavilla sadettimilla. Mitä tummempi alue sitä enemmän vettä.

6.3.3 Vertailu

Ohjelmoitavilla sadettimilla saavutettava veden säästö on 31%. Tämä on hyvin merkittävä säästö, sillä golfkentällä kasteluvedenkulutus on valtava. Tulokset-kappaleen lopussa arvioidaan, millaisia taloudellisia säästöjä tällaisella vedensäästöllä saavutetaan.

Tasaisuudessa ei näy merkittävää eroa perinteisten ja ohjelmoitavien sadettimien välillä, mikä johtuu siitä, että ohjelmoitavien sadettimien kastelukuvioiden määrittämisessä keskityttiin vedenkulutukseen laskemiseen. Veden osuminen kentälle on parantunut kuitenkin selvästi.

	Perinteinen sadetin	Ohjelmoitava sadetin
Tasaisuus (CU)	75.3%	75.5%
Ulkopuolelle mennyt vesi	10.5%	5.9%

Taulukko 3: Golfkenttä-tulosten vertailu

7 Tulokset

7.1 Veden käyttö

Ohjelmoitavaa sadetinta verrattiin perinteiseen sadetimeen kolmessa esimerkkitaapauksessa: pellolla, kotipihalla ja golfkentällä. Tietokonesimulaatio loi mallit näistä tapauksista ja vertaili käytetyn veden määrää, veden jakaantumisen tasaisuutta sekä veden osumista kentälle.

1. Pellolla ohjelmoitavalla sadettimella saavutettiin 49%:n veden säästö.

Veden jakaantumisen tasaisuus (CU) oli perinteisellä sadettimella 67.9%, kun taas ohjelmoitavalla sadettimella se oli 81.2%.

Perinteisillä sadettimilla vedestä lensi pellon ulkopuolelle 23.3%, kun taas ohjelmoitavilla sadettimilla ulkopuolelle lensi 5.6%.

2. Kotipihalla ohjelmoitavalla sadettimella saavutettiin 27%:n veden säästö.

Perinteisellä sadettimella vedestä lensi pihan ulkopuolelle 15.4%, kun taas ohjelmoitavalla sadettimella ulkopuolelle lensi 2.3%.

3. Golfkentällä ohjelmoitavalla sadettimella saavutettiin 31%:n veden säästö.

Perinteisillä sadettimilla vedestä lensi kentän ulkopuolelle 10.5%, kun taas ohjelmoitavilla sadettimilla ulkopuolelle lensi 5.9%.

7.2 Taloudellinen vaikutus

Säästetyllä veden määrällä on suora taloudellinen vaikutus kasteltavan alueen ylläpitokustannuksiin. Golfkentillä kasteluun kuluu huomattavia määriä vettä, joten siellä vesisäästö saattaa olla taloudellisesti hyvinkin merkittävä. Yleensä golfkentät saavat kasteluvetensä läheisestä järvestä, jolloin veden käyttö ei tuota niin merkittäviä kustannuksia. Kuitenkin esimerkiksi kotiseudullani Turussa sijaitseva Aura Golf joutuu käyttämään vesijohtovettä, sillä kentän lähistöltä ei löydy sopivaa vesistöä. Merivesi ei sovellu kasteluun liian suuren suolapitoisuutensa vuoksi.

Aura Golfin kenttämestarin Matti Höglundin mukaan Aura Golf käytti vuonna 2006 kenttien kasteluun noin 30 000 m^3 vettä. Vedestä ei tarvitse maksaa jätevesimaksua, sillä vesi menee maahan, mutta vesimaksu yksinään on Turussa 1.07 euroa/ m^3 ilman arvonlisäveroa ([11]). Tämä tarkoittaa, että

vuonna 2006 Aura Golfilta kului kasteluun noin 32 100 euroa.

Luvun (6.3) golfkenttäesimerkissä saavutettiin 31%:n veden säästö. Jos oletetaan, että ohjelmoitavista sadettimista rakennettua kastelujärjestelmää pystytettäisiin soveltamaan Aura Golfissa esimerkin tavoin, voidaan laskea ohjelmoitavilla sadettimilla saavutettava taloudellinen säästö:

$$\text{säästö} = 32100 \text{ euroa} \cdot 0.31 = 9951 \text{ euroa} \approx 10000 \text{ euroa}$$

Kymmentuhannen euron säästö on jo hyvin merkittävä yksittäisen kentän budjetissa. Taloudelliset säästöt nousevat kuitenkin vieläkin merkittävimiksi maissa, joissa kuivuus on ongelma, kuten esimerkiksi Yhdysvalloissa. Rahan lisäksi paremmalla kastelulla säästetään myös luontoa.

8 Jatkokehitys

Ohjelmoitava sadetin on osoittautunut erittäin mielenkiintoiseksi tutkimuksen kohteeksi tutkielmaa tehdessä, ja sen ympäriltä löytyy paljon jatkotutkittavaa. Tietokonemallin toimivuuden testaaminen olisi merkittävä saavutus, ja sen toteuttamiseksi ohjelmoitavasta sadettimesta tulisi rakentaa prototyyppi. Tällaisen prototyypin rakentaminen onnistuisi luultavasti ilman merkittävää tekniikan alan tuntemusta.

Kun määrittelin ohjelmoitavien sadettimien kastelukuviot kolmessa esimerkitapauksessa, käytin ainoastaan omaa päättelyä ja intuitiota, minkä vuoksi kastelukuviot eivät varmastikaan ole täydellisen optimaalisia. Olisi hyödyllistä luoda tietokoneohjelma, joka laskisi parhaat mahdolliset kastelukuviot sadettajille. Tällöin päästäisiin lähemmäksi sadettimien todellista vedenkulutusta ja kastelun todellista tasaisuutta. Ohjelmaa voitaisiin myöskin hyödyntää, jos jossain otettaisiin käyttöön ohjelmoitavista sadettimista koostuva kastelujärjestelmä. Ohjelman avulla kastelun suunnittelijalta säästyisi huomattavasti aikaa ja vaivaa.

Eräs jatkotutkimuksen kohde on tuulen vaikutus kasteluun. Tuuli on usein sadettimien ongelmana, sillä se vaihtelee melko arvaamattomasti, ja se voi vaikuttaa merkittävästikin veden lentoon. Ohjelmoitavalla sadettimella tuulen muutoksiin voidaan mukautua muuttamalla sadettimen pyörimistä ja kastelukuviota. Ohjelmoitava sadetin voidaan esimerkiksi kytkeä tuulisen-soriin, ja luoda järjestelmä, joka mukauttaa sadettimen toimintaa sitä mukaa, kun sensori antaa tietoa tuulen muutoksista. Kehittämäni simulaatio-ohjelmaa voisi laajentaa siten, että se osaisi mallintaa tuulen vaikutusta veden lentoon. Tällöin voitaisiin arvioida, miten tuulen voimakkuus vaikuttaa ohjelmoitavalla sadettimilla saavutettaviin vesisäästöihin.

Viitteet

- [1] Choate, R. & Watkins, J.
Turf Irrigation Manual (1994)
- [2] CIT (Center for Irrigation Technology)
Sprinkler Profile Index (2005)
Saatavilla: <http://cati.csufresno.edu/cit/good/citprofiles.html> [30.11.2006]
- [3] Comap
Vuoden 2006 MCM-kilpailun (Mathematical Contest in Modeling) A tehtävän tehtävänanto.
Saatavilla: <http://www.comap.com/undergraduate/contests/mcm/contests/2006/problems/> [6.11.2006]
- [4] FAO (Yhdistyneiden kansakuntien elintarvike- ja maatalousjärjestö)
Irrigation Water Management: Irrigation Methods
<http://www.fao.org/docrep/S8684E/s8684e00.HTM#Contents>
- [5] Gilmour Group
PATTERN MASTER SPRINKLERS
Saatavilla: http://www.gilmour.com/Watering_Hose_End/Sprinklers/PatternMaster.asp [27.11.2006]
- [6] Kivelä, A., Malmi, E., Harjumäki, J.
Uniform And Easy-To-Maintain Irrigation (2006)
Saatavilla: http://wiki.paivola.fi/mediawiki/images/e/e6/Mcm_862_06.pdf [14.11.2006]
- [7] Kuva 1
Saatavilla: <http://www.jessstryker.com/includes/irrigationtutorials1.jpg> [30.11.2006]
- [8] Kuva 2
Saatavilla: <http://www.cmsgardens.co.uk/images/hozelock/lawnq.jpg> [30.11.2006]
- [9] Kuva 5
Saatavilla: <http://cati.csufresno.edu/cit/good/readprofile.asp?File=RAINBIRD&Record=18> [30.11.2006]
- [10] Rogers, D., Lamm, F., Alam, M., Trooien, T., Clark, G., Barnes, P. & Kyle
EFFICIENCIES AND WATERLOSSES OF IRRIGATION SYSTEMS
Kansas State University, Manhattan (toukokuu, 1997)
Saatavilla: <http://www.oznet.ksu.edu/library/ageng2/mf2243.pdf> [27.11.2006]

- [11] Turun vesilaitos
Palveluhinnasto 2006 (1.2.2006 alkaen)
Saatavilla: <http://www05.turku.fi/vesilaitos/asiakaspalvelu/palveluhinnasto.html> [30.11.2006]
- [12] Zoldoske, D. and Solomon, K.
Coefficient of Uniformity - What it tells us (tammikuu, 1988)
Saatavilla: <http://cati.csufresno.edu/cit/rese/88/880106/>
[30.11.2006]

A Lähdekoodi

Tietokonesimulaatio on tehty Java-ohjelmointikielen versiolla *5.0 Standard Edition*. Alla on ohjelmiston kolme algoritmillisesti merkittävintä luokkaa: Sadetin, KenttaSovellus ja KastelunArviointi. Ohjelmisto löytyy kokonaisuudessaan osoitteesta: <http://ekku.users.paivola.fi/sadetus/Tutkielma.zip>.

A.1 Luokka Sadetin

```
import java.awt.Polygon;
import java.io.Serializable;

public class Sadetin implements Serializable {

    // kastelukuvion reunat
    public Polygon po;

    // sadettimen keskipisteen x- ja y-koordinaatit
    int kx, ky;

    // säde
    int sade;

    // taulukko, joka sisältää kasauma-arvot
    public int[][] taulu;

    // iteroimisaskel, kun etsitään polygonin reunaa
    double v = 1;
    // iteroimisaskeleen komponentit
    double v_x, v_y;

    // tämä ilmoittaa, kuinka paljon vettä maksimissaan voi ruutuun kertyä
    // yhden sadettimen vaikutuksesta
    int maxVesi;

    // myös tämä vaikuttaa siihen, kuinka paljon yhteen ruutuun kertyy vettä.
    // Tämän avulla kastelukuvioita voidaan skaalata, mikäli arvot uhkaavat
    // mennä yli sallitun värialueen (255,255,255).
    double kerroin = 1.0;

    // Tämä kertoo pyöriikö suutin vakionopeudella, vai muutetaanko
    // pyörimisnopeutta, jotta vettä kertyy tasaisesti. (EI IMPLEMENTOITUNA
    // MALLIIN)
    boolean vakioPyoriminen = false;

    // sijainti pellolla
    public int kenttax = 0;
    public int kenttay = 0;

    // reunaväli editorissa
    int rv = EditoriAlue.rv;
    // reunaväli kentällä
    int rv_kentta;

    // tätä konstruktoria käytetään editorissa,
    // kun halutaan muodostaa kastelutaulukko polygonin avulla
    public Sadetin(Polygon po, int sade) {
        this.po = po;
        this.sade = sade;

        // tätä tulee skaalata
        this.maxVesi = (int)(sade*kerroin);
        this.kx = sade + rv;
        this.ky = sade + rv;
        taulu = new int[2*sade][2*sade];
        kayLapiRuudut();
    }

    // tätä konstruktoria käytetään editorissa,
    // kun halutaan muodostaa kastelutaulukko vakioyörimisellä
    // (EI IMPLEMENTOITUNA MALLIIN)
    public Sadetin(Polygon po, int sade, boolean vakioPyoriminen) {
        this.vakioPyoriminen = vakioPyoriminen;
        this.po = po;
        this.sade = sade;

        // tätä tulee skaalata
        this.maxVesi = (int)(sade*kerroin);
        this.kx = sade + rv;
        this.ky = sade + rv;
        taulu = new int[2*sade][2*sade];
        kayLapiRuudut();
    }

    // tätä konstruktoria käytetään Kentta-luokassa, kun
    // ladataan tallennetut polygoni ja taulukko
```

```

public Sadetin(Polygon po, int[] taulu) {
    this.po = po;
    this.taulu = taulu;
    this.sade = (taulu.length/2);
    this.kx = sade + rv;
    this.ky = sade + rv;
}

/**
 * Metodi käy läpi kaikki sadettimen ympärillä olevat ruudut ja laskee
 * niille kertyvät kasaumat
 */
void kayLapiRuudut() {
    for(int i = rv; i < 2*sade+rv; i++) {
        for(int j = rv; j < 2*sade+rv; j++) {
            // tarkastetaan, että piste on kastelukuvion sisällä
            if(po.contains(i, j)) {
                taulu[i-rv][j-rv] = laskeKasautuma(i, j);
            }
        }
    }
}

/**
 * Laskee, kuinka paljon kyseiseen ruutuun tulee vettä. Tarkastelee
 * kuviota neljässä osassa - ympyrän neljänneksissä.
 *
 * @return kasautuma
 */
public int laskeKasautuma(int x, int y) {

    double kulma = laskeKulma(x, y);
    v_x = Math.cos(kulma)*v;
    v_y = Math.sin(kulma)*v;

    // keskipisteen ja tarkasteltavan pisteen välinen etäisyys
    double etaisyyys = Math.sqrt(Math.pow(kx-x, 2)+Math.pow(ky-y, 2));

    // sen janan pituus, joka kulkee keskipisteestä reunaan tarkasteltavan
    // pisteen (x, y) kautta. Pituus on luonnollisesti vähintään
    // keskipisteen ja tarkasteltavan pisteen välinen etäisyys.
    double pituus = etaisyyys;

    int askel = 1;

    // ympyrän 1. neljännes (tässä koordinaatistossa se on tosin oikeassa
    // alakulmassa eikä yläkulmassa, sillä piste (0,0) on ylhäällä)
    if(x > kx && y >= ky) {
        while(po.contains(x+askel*v_x, y+askel*v_y)) {
            askel++;
        }
    } else if(x <= kx && y > ky) {
        while(po.contains(x-askel*v_x, y-askel*v_y)) {
            askel++;
        }
    } else if(x < kx && y <= ky) {
        while(po.contains(x-askel*v_x, y-askel*v_y)) {
            askel++;
        }
    } else if(x >= kx && y < ky) {
        while(po.contains(x+askel*v_x, y+askel*v_y)) {
            askel++;
        }
    } else if(x == kx && y == ky) {
        askel = sade;
    }

    pituus += askel*v;

    int kasautuma;
    if(vakioPyoriminen) {
        // (EI IMPLEMENTOITUNA MALLIIN)
        maxVesi = (int)(sade / pituus * sade *kerroin);
        kasautuma = (int)((1-etaisyyys/pituus)*maxVesi);
    } else {
        kasautuma = (int)((1-etaisyyys/pituus)*maxVesi);
    }

    if(kasautuma > 254) {
        // debuggausta
        System.out.println(etaisyyys);
        System.out.println(pituus);
        System.out.println(kasautuma);
        System.out.println(kulma*180/Math.PI);
        System.out.println(v_x);
        System.out.println(v_y);
    }

    return kasautuma;
}

/**
 * Laskee annetun pisteen (x, y) kulman
 * suhteessa sadettimen keskipisteeseen
 *
 * @param x
 * @param y
 * @return kulma radiaaneina
 */

```

```

*/
public double laskeKulma(int x, int y) {
    if(kx-x != 0) {
        return Math.atan((double)(ky-y)/(kx-x));
    } else {
        return (-Math.PI/2);
    }
}

/**
 * Lisää kentälle tämän sadettimen vaikutuksen.
 *
 * @param kentta
 * @return
 */
public int[][] kastele(int[][] kentta) {
    for(int i = 0; i < 2*sade; i++) {
        for(int j = 0; j < 2*sade; j++) {
            kentta[kenttax-sade+i][kenty-sade+j] += taulu[i][j];
        }
    }
    return kentta;
}

/**
 * Sijoittaa sadettimen kentälle haluttuun paikkaan ja palauttaa sen.
 *
 * @param x
 * @param y
 */
public Sadetin sijoita(int x, int y, int rv_kentta) {
    this.rv_kentta = rv_kentta;
    this.kenttax = x+rv_kentta;
    this.kenty = y+rv_kentta;
    siirraPolygon();
    return this;
}

/**
 * Muuttaa polygonin arvoja, jotta se piirtyy oikealle paikalleen kentällä.
 */
public void siirraPolygon() {
    int siirtox = kenttax - kx;
    int siirtoy = kenty - ky;

    for(int i = 0; i < po.npoints; i++) {
        po.xpoints[i] += siirtox;
        po.ypoints[i] += siirtoy;
    }
}
}
}

```

A.2 Luokka KenttaSovellus

```

import java.awt.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.util.Vector;

import javax.swing.JComponent;

import editori.Sadetin;

/**
 * Tässä luokassa ladataan sadettimet ja kenttä sekä asetellaan
 * sadettimet kentälle. Sen jälkeen suoritetaan kastelu, piirretään
 * havainnollistava kuva ja tulostetaan kastelun arvot komentoriville.
 *
 * @author Eric
 */
public class KenttaSovellus extends JComponent {

    // määrittää, mitä kenttää simuloidaan
    int kentta = 1;

    // määrää kasteleeko kenttä perinteisillä
    // vai ohjelmoitavilla sadettimilla
    boolean perinteinen = false;

    // määrää käytetäänkö vakionopeudella pyöriviä ohjelmoitavia sadettimia
    // (EI IMPLEMENTOITUNA MALLIIN)
    boolean vakioPyoriminen = false;

    // kenttä jaettuna taulukkoon
    int[][] kenttaTaulu;

    int kentanLeveys = 0;
    int kentanKorkeus = 0;

    // reunoihin tuleva väli
    static int rv = 100;
}

```

```

// tämän avulla voidaan skaalata värejä, jos ne uhkaavat
// mennä ulos RGB-alueelta
double variKerroin = 1.0;

Polygon kentanReunat;

// tähän vektoriin lisätään kaikki kentälle laitettavat sadettimet
Vector<Sadetin> sadettimet = new Vector<Sadetin>();

// määrittää, merkataanko kenttään kohta, johon
// on tullut vähiten vettä
boolean varitaMinimi = false;
// veden määrä minimikohdassa
static int minx = -1;
static int miny = -1;

// seuraavilla määritetään, mitkä osat halutaan piirtää
boolean piirraSadettimienReunat = !false;
boolean piirraKasaantuminen = !true;
boolean piirraKentanReunat = !true;
// sadettimien keskipisteet
boolean piirraPisteet = !false;

public KenttaSovellus() {
    kentanReunat = maaritaKenttaAalue(kentta);
    luoSadettimet(kentta, perinteinen);
    suoritaKastelu();
    repaint();
}

public void suoritaKastelu() {

    // laitetaan jokainen sadetin kastelemaan
    for(Sadetin s : sadettimet) {
        this.kenttaTaulu = s.kastele(kenttaTaulu);
    }

    // lasketaan, millainen kastelu saavutettiin
    // ja tulostetaan arvot komentoriville

    System.out.println("Keskiarvo: "
        + KastelunArviointi.keskiarvo(kentanReunat, kenttaTaulu
        , kentanLeveys, kentanKorkeus, rv));

    System.out.println("CU: "
        + KastelunArviointi.laskeCU(kentanReunat, kenttaTaulu
        , kentanLeveys, kentanKorkeus, rv));

    System.out.println("Kuivin arvo: "
        + KastelunArviointi.minimi(kentanReunat, kenttaTaulu
        , kentanLeveys, kentanKorkeus, rv));

    System.out.println("Kostein arvo: "
        + KastelunArviointi.maksimi(kentanReunat, kenttaTaulu
        , kentanLeveys, kentanKorkeus, rv));

    System.out.println("Hukkaan menneen veden osuus: "
        + KastelunArviointi.haaskatunVedenOsuus(kentanReunat
        , kenttaTaulu, kentanLeveys, kentanKorkeus, rv));
}

public void paint(Graphics g) {

    System.out.println("Piiirretään...");

    // väritetään pohja valkoiseksi
    g.setColor(Color.white);
    g.fillRect(0, 0, this.getWidth(), this.getHeight());

    int variArvo;
    // piirretään yksittäisen ruudun veden kasaantuminen
    if(piirraKasaantuminen) {
        for(int i = 0; i < kentanLeveys*2*rv; i++) {
            for(int j = 0; j < kentanKorkeus*2*rv; j++) {
                variArvo = (int) (kenttaTaulu[i][j] * variKerroin);
                if(variArvo > 255) {
                    g.setColor(new Color(255, 0, 0));
                } else {
                    g.setColor(new Color(255-variArvo
                        , 255-variArvo
                        , 255-variArvo));
                }
                g.drawLine(i,j,i,j);
            }
        }
    }

    if(piirraSadettimienReunat) {
        g.setColor(Color.black);
        for(Sadetin s : sadettimet) {
            g.drawPolygon(s.po);
        }
    }

    if(piirraKentanReunat) {
        g.setColor(Color.black);
        g.drawPolygon(kentanReunat);
    }
}

```

```

    if(piiirraPisteet) {
        g.setColor(Color.red);
        for(Sadetin s : sadettimet) {
            g.fillOval(s.kenttax-3, s.kenttay-3, 6, 6);
        }
    }

    if(varitaMinimi) {
        g.setColor(Color.red);
        g.drawLine(minx, miny, minx+5, miny+5);
    }
}

public Polygon maaritaKenttaAlue(int tyyppi) {
    Polygon ret = null;

    if(tyyppi == 1) {
        // määritellään MCM-pellon muotoinen polygoni
        int[] xpoints = new int[4];
        int[] ypoints = new int[4];

        xpoints[0] = 0+rv;
        ypoints[0] = 0+rv;

        xpoints[1] = 800+rv;
        ypoints[1] = 0+rv;

        xpoints[2] = 800+rv;
        ypoints[2] = 300+rv;

        xpoints[3] = 0+rv;
        ypoints[3] = 300+rv;

        ret = new Polygon(xpoints, ypoints, 4);
    } else if(tyyppi == 2) {
        // ladataan kotipiha-polygoni
        ret = lataaKentta("kotipiha");
    } else if(tyyppi == 3) {
        // ladataan golfkenttä-polygoni
        ret = lataaKentta("auraGolf");
    } else if(tyyppi == 4) {
        // alue testausta varten
        int[] xpoints = new int[4];
        int[] ypoints = new int[4];

        xpoints[0] = 0+rv;
        ypoints[0] = 0+rv;

        xpoints[1] = 800+rv;
        ypoints[1] = 0+rv;

        xpoints[2] = 800+rv;
        ypoints[2] = 300+rv;

        xpoints[3] = 0+rv;
        ypoints[3] = 300+rv;

        ret = new Polygon(xpoints, ypoints, 4);
    }

    this.kentanLeveys = ret.getBounds().width;
    this.kentanKorkeus = ret.getBounds().height;

    kenttaTaulu = new int[kentanLeveys+2*rv][kentanKorkeus+2*rv];

    // tulostetaan mitat
    System.out.println("Taulun koko on: (" + (kentanLeveys+2*rv) + "x"
        + (kentanKorkeus+2*rv) + ")");

    return ret;
}

/**
 * Määritellään mitkä sadettimet asetellaan minne milläkin kenttätyypillä.
 * @param tyyppi
 * @param perinteinen
 */
public void luoSadettimet(int tyyppi, boolean perinteinen) {
    if(tyyppi == 1) {
        // luodaan sadettimet MCM-peltoa varten
        if(perinteinen) {
            for(int k = 0; k < 4; k++) {
                for(int i = -1; i < 2; i++) {
                    sadettimet.addElement(lataaSadetin("ympyra")
                        .sijoita(50 + k*233, 150 + i*100, rv));
                }
            }
        }
    } else {
        // ohjelmoitavien sadettimien sijoittelu pellolle
    }
}

```

```

for(int k = 1; k < 3; k++) {
    for(int i = -1; i < 2; i++) {
        sadettimet.addElement(lataaSadetin("keski")
            .sijoita(50 + k*233, 150 + i*100, rv));
    }
}

sadettimet.addElement(lataaSadetin("av")
    .sijoita(50, 50, rv));
sadettimet.addElement(lataaSadetin("av")
    .sijoita(50, 150, rv));
sadettimet.addElement(lataaSadetin("av")
    .sijoita(50, 250, rv));

sadettimet.addElement(lataaSadetin("41")
    .sijoita(749, 50, rv));
sadettimet.addElement(lataaSadetin("41")
    .sijoita(749, 150, rv));
sadettimet.addElement(lataaSadetin("41")
    .sijoita(749, 250, rv));
}

} else if(tyyppi == 2) {
    // luodaan sadettimet kotipihaa varten
    if(perinteinen) {
        sadettimet.addElement(lataaSadetin("kotiSadetinPer")
            .sijoita(130, 130, rv));
    } else {
        sadettimet.addElement(lataaSadetin("pihaSadetin2")
            .sijoita(130, 130, rv));
    }
} else if(tyyppi == 3) {
    // luodaan sadettimet golfkenttää varten
    if(perinteinen) {
        sadettimet.addElement(lataaSadetin("aura_80")
            .sijoita(170 + 20, 155, 0));
        sadettimet.addElement(lataaSadetin("aura_80")
            .sijoita(250 + 20, 155, 0));
        sadettimet.addElement(lataaSadetin("aura_80")
            .sijoita(170 + 20, 235, 0));
        sadettimet.addElement(lataaSadetin("aura_80")
            .sijoita(250 + 20, 235, 0));
        sadettimet.addElement(lataaSadetin("aura_80")
            .sijoita(345 + 20, 155, 0));
        sadettimet.addElement(lataaSadetin("aura_80")
            .sijoita(345 + 20, 235, 0));
        sadettimet.addElement(lataaSadetin("aura_120")
            .sijoita(460 + 20, 195, 0));
        sadettimet.addElement(lataaSadetin("aura_120")
            .sijoita(580 + 20, 195, 0));
        sadettimet.addElement(lataaSadetin("aura_120_puol")
            .sijoita(720, 195, 0));
    } else {
        sadettimet.addElement(lataaSadetin("aura_11")
            .sijoita(170 + 20, 155, 0));
        sadettimet.addElement(lataaSadetin("aura_22")
            .sijoita(250 + 20, 155, 0));
        sadettimet.addElement(lataaSadetin("aura_12")
            .sijoita(170 + 20, 235, 0));
        sadettimet.addElement(lataaSadetin("aura_21")
            .sijoita(250 + 20, 235, 0));
        sadettimet.addElement(lataaSadetin("aura_31b")
            .sijoita(345 + 20, 155, 0));
        sadettimet.addElement(lataaSadetin("aura_32")
            .sijoita(345 + 20, 235, 0));
        sadettimet.addElement(lataaSadetin("aura_120_ohj")
            .sijoita(460 + 20, 195, 0));
        sadettimet.addElement(lataaSadetin("aura_120_ohj")
            .sijoita(580 + 20, 195, 0));
        sadettimet.addElement(lataaSadetin("aura_120_puol_ohj")
            .sijoita(720, 195, 0));
    }
} else if(tyyppi == 4) {
    // testaus sadettimet
    sadettimet.addElement(lataaSadetin("ohj_malli2")
        .sijoita(100, 100, rv));
}
}

/**
 * Lataa annetun nimisen sadettimen tiedostosta.
 * @param nimi
 * @return
 */
public Sadetin lataaSadetin(String nimi) {
    Sadetin ret = null;

    if(!perinteinen) {
        if(vakioPyoriminen) {
            nimi = nimi + "_vakio";
        }
    }

    try{
        // ladataan polygoni reunojen piirtämistä varten
        FileInputStream polyIn = new FileInputStream("tallennetut"

```

```

        + File.separator + nimi + "_p.dat");
// ladataan taulukko väritymistä varten
FileInputStream tauluIn = new FileInputStream("tallennetut"
        + File.separator + nimi + "_t.dat");
ObjectInputStream polyObjIn = new ObjectInputStream(polyIn);
ObjectInputStream tauluObjIn = new ObjectInputStream(tauluIn);
Polygon po = (Polygon)polyObjIn.readObject();
polyObjIn.close();
int[][] taulu = (int[][])tauluObjIn.readObject();
tauluObjIn.close();

ret = new Sadetin(po, taulu);
return ret;

}catch(ClassNotFoundException cnfe){
    System.out.println("Lataaminen epäonnistui!");
    cnfe.printStackTrace();
}catch(IOException ioe){
    System.out.println("IOException: Sadettimen " + nimi + ".dat" +
        " lataaminen epäonnistui");
}
return ret;
}

/**
 * Lataa annetun nimisen kentän tiedostosta.
 * @param nimi
 * @return
 */
public Polygon lataaKentta(String nimi) {
    try{
        // ladataan polygoni reunojen piirtämistä varten
        FileInputStream polyIn = new FileInputStream("tallennetut"
            + File.separator + "kentat"
            + File.separator + nimi + ".dat");
        ObjectInputStream polyObjIn = new ObjectInputStream(polyIn);
        Polygon po = (Polygon)polyObjIn.readObject();
        polyObjIn.close();

        return po;

    }catch(ClassNotFoundException cnfe){
        cnfe.printStackTrace();
        System.out.println("Lataaminen epäonnistui.");
        return null;
    }catch(IOException ioe){
        System.out.println("IOException: Kentän " + nimi + ".dat" +
            " lataaminen epäonnistui");
        return null;
    }
}
}
}

```

A.3 Luokka KastelunArviointi

```

import java.awt.Polygon;

/**
 * Tätä luokkaa käytetään kastelun arviointiin liittyvien arvojen laskemiseen.
 *
 * @author Eric
 *
 */
public class KastelunArviointi {

    public static double keskiarvo(Polygon kentanReunat, int[][] kenttaTaulu
        , int kentanLeveys, int kentanKorkeus, int rv) {
        int summa = 0;
        int n = 0;
        for(int i = 0; i < kentanLeveys+2*rv; i++) {
            for(int j = 0; j < kentanKorkeus+2*rv; j++) {
                if(kentanReunat.contains(i, j)) {
                    n++;
                    summa += kenttaTaulu[i][j];
                }
            }
        }
        return (summa/(n));
    }

    public static double laskeCU(Polygon kentanReunat, int[][] kenttaTaulu
        , int kentanLeveys, int kentanKorkeus, int rv) {
        int summa = 0;
        int n = 0;
        double keskiarvo = keskiarvo(kentanReunat, kenttaTaulu, kentanLeveys
            , kentanKorkeus, rv);

        for(int i = 0; i < kentanLeveys+2*rv; i++) {
            for(int j = 0; j < kentanKorkeus+2*rv; j++) {
                if(kentanReunat.contains(i, j)) {
                    n++;
                    summa += Math.abs(kenttaTaulu[i][j]-keskiarvo);
                }
            }
        }
    }
}

```

```

    }
  }
}
return (100*(1-(summa/n)/keskiarvo));
}

public static int minimi(Polygon kentanReunat, int[][] kenttaTaulu,
int kentanLeveys, int kentanKorkeus, int rv) {

int min = 1000000;
for(int i = 0; i < kentanLeveys+2*rv; i++) {
for(int j = 0; j < kentanKorkeus+2*rv; j++) {
if(kentanReunat.contains(i, j)) {
if(kenttaTaulu[i][j] < min) {
min = kenttaTaulu[i][j];
KenttaSovellus.minx = i;
KenttaSovellus.miny = j;
}
}
}
}
return min;
}

public static int maksimi(Polygon kentanReunat, int[][] kenttaTaulu,
int kentanLeveys, int kentanKorkeus, int rv) {

int max = -1;
for(int i = 0; i < kentanLeveys+2*rv; i++) {
for(int j = 0; j < kentanKorkeus+2*rv; j++) {
if(kentanReunat.contains(i, j)) {
if(kenttaTaulu[i][j] > max) {
max = kenttaTaulu[i][j];
}
}
}
}
return max;
}

public static double haaskatunVedenOsuus(Polygon kentanReunat, int[][] kenttaTaulu,
int kentanLeveys, int kentanKorkeus, int rv) {

int hukka = 0;
int osunut = 0;

for(int i = 0; i < kentanLeveys+2*rv; i++) {
for(int j = 0; j < kentanKorkeus+2*rv; j++) {
if(kentanReunat.contains(i, j)) {
osunut += kenttaTaulu[i][j];
} else {
// ei osunut kentälle
hukka += kenttaTaulu[i][j];
}
}
}
System.out.println("Hukkaan mennyt: " + hukka);
System.out.println("Vettä yhteensä: " + (hukka+osunut));
return ((double)100.0*hukka/(hukka+osunut));
}
}

```