MurderKilllBurn

Käyttöliittymäohjelmoinnin harjoitustyö

Timo Hakamäki

Jarko Miettinen

Teemu Sormunen

Harjoitustyö

Huhtikuu 2017

Insinööri (AMK)

Tieto- ja viestintätekniikka

Sisällysluettelo

[1 Johdanto 2](#_Toc480454290)

[2 Rakennekaavio ja muutokset 2](#_Toc480454291)

[2.1 Toteutetut luokat 2](#_Toc480454292)

[3 Työaikaraportti 3](#_Toc480454293)

[4 Ongelmat ja ratkaisut 5](#_Toc480454294)

[4.1 Hahmon liikkuminen ja kääntyminen 5](#_Toc480454295)

[4.2 Ampuminen 6](#_Toc480454296)

[4.3 Viholliset 7](#_Toc480454297)

[5 Testaus 7](#_Toc480454298)

[6 Lopullinen versio 8](#_Toc480454299)

[7 Itsearviointi 9](#_Toc480454300)

[8 Kommentit kurssista 9](#_Toc480454301)

# Johdanto

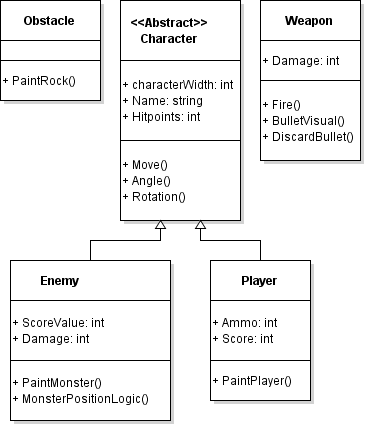
Tämän dokumentin tarkoitus on esitellä käyttöliittymäohjelmointi-kurssin harjoitustyön lopullista versiota.. Raportissa esitellään työn vaiheet, vastaan tulleet ongelmat, muutokset alkuperäisiin suunnitelmiin sekä työn lopputulos.

Harjoitustyö löytyy osoitteesta <https://github.com/K8161/MurderKillBurn>

# Rakennekaavio ja muutokset

Luokkarakenne itsessään pysyi lähestulkoon samana, mutta luokkien metodeja jäi pois ja uusia tuli tilalle. Luokista pois jäi pelkästään Map, sillä kanvaasi luotiin XAMLin puolella ja sen muuttaminen luokkaan perustuvaksi tuntui tarpeettoman työläältä hyötyihin nähden.

## Toteutetut luokat



Kuvio Lopullinen luokkakaavio

Lopulliseen versioon tuli viisi eri luokkaa. Obstacle-luokka toimii kivien pohjana, jossa kivien kaikki ominaisuudet määritellään ellipsin kautta, jolla ne myös piirretään.

Character-luokka toimii pohjana Enemy- ja Player-luokille, jotka perivät yhteiset ominaisuutensä ja metodit ensin mainitulta luokalta. Enemy-luokalla on ominaisuuksinaan ScoreValue ja Damage, jotka kertovat kuinka paljon pisteitä vihollisen tappamisesta saa, sekä kuinka paljon vihollinen tekee vahinkoa pelaajalle. Metodia PaintMonster käytetään vihollisten piirtämiseen ja MonsterPositionLogic hoitaa laskentalogiikan vihollisten liikkumiselle. Player-luokalla taas on ominaisuuksina Ammo sekä Score, jotka määrittävät pelaajan ammus- ja pistetilanteet. PaintPlayer-metodi hoitaa pelaajan piirtämisen. PaintPlayer ja PaintMonster on toteutettu erikseen yhteisen PaintCharacter-metodin sijaan siksi, että viholliset ja pelihahmot hyödyntävät eri

Weapon-luokalla on pelkkä Damage-ominaisuus, jonka avulla oli tarkoitus määritellä aseen tekemä vahinko. Tämä ominaisuus on tällä hetkellä käyttämättä, sillä pelillisesti tuntui paremmalta, että ammus tappaa vihollisen kerrasta. Damage-ominaisuudelle tulisi käyttöä, mikäli peliin lisättäisiin useita erilaisia ja eri vahvuisia vihollisia. Metodeina Weapon-luokalla on Fire, BulletVisual ja DiscardBullet. Ensin mainittua käytetään panoksen liikuttamiseen haluttuun kohteeseen, toista ammuksen piirtämiseen ja kolmas poistaa ammuksen sen osuessa joko viholliseen tai kiveen.

# Työaikaraportti

Työajan käyttö on selvennetty alla olevassa taulukossa.

|  |  |  |  |
| --- | --- | --- | --- |
| Viikko | Tehtävä | Käytetyt tunnit | Vastuu |
| 10 | Alustava käyttöliittymä valikoille | 4 | Jarko Miettinen |
| 10 | Luokkien luonti | 4+4 | Timo Hakamäki, Teemu Sormunen |
| 12 | Ohjelman perustan luonti | 4\*3 | Kaikki |
| 13 | Hahmon liikkuminen | 8 | Timo Hakamäki |
| 13 | Hahmon kääntyminen | 8+8 | Jarko Miettinen, Teemu Sormunen |
| 14 | Hahmon kääntyminen ja ampuminen kohti hiirtä | 8 | Jarko Miettinen |
| 14 | Hahmon liikkuminen kohti hiirtä | 4 | Teemu Sormunen |
| 14 | Esteet ja törmäystarkastelu | 4 | Timo Hakamäki |
| 15 | Pelaajaa seuraava vihollinen | 4 | Teemu Sormunen |
| 15 | Useamman pelaajaa seuraavan vihollisen luonti | 2 | Jarko Miettinen |
| 15 | Refaktorointi ja koodin siirtäminen luokkiin | 4 | Timo Hakamäki |
| 15 | Raportin pohjan luonti ja aloitus | 3 | Jarko Miettinen |
| 16 | Refaktorointi ja koodin kommentointi | 5 | Timo Hakamäki |
| 16 | Useamman ammuksen luominen | 3 | Jarko Miettinen |
| 16 | Törmäystarkastelujen parantelu | 2 | Jarko Miettinen |
| 16 | Omien grafiikoiden luonti | 5 | Teemu Sormunen |
| 16 | Äänien sisällyttäminen peliin | 3 | Timo Hakamäki |

# Ongelmat ja ratkaisut

Ongelmia projektin edetessä tuli vastaan useita, joitakin osattiin odottaa ja joitakin ei. Ongelmat koostuivat pääasiassa pelin tekijöiden kokemattomuudesta, joten peliä luodessa jokainen projektin jäsen joutui itsenäisesti opettelemaan paljon uusia asioita, sekä hyödyntämään lähes kaikkia opintojaksolla opittuja asioita.

Suurimmat vastaan tulleet ongelmat on lajiteltu omiksi aliotsikoikseen. Jokaisen ongelman kohdalla on myös kerrottu, miten kyseinen ongelma saatiin ratkaistua.

## Hahmon liikkuminen ja kääntyminen

Hahmon haluttiin liikkuvan kentällä niin, että hiiri toimisi tähtäimenä ja pelaajahahmo katsoisi koko ajan tähtäintä päin. Lisäksi liikkumisesta haluttiin sulavaa pelkän x- ja y-akselilla suoraan liikkumisen sijaan. Peliin otettiin pohja tunnilla yhdessä tehdystä matopelistä, jossa mato liikkui kanvaasilla vain neljään eri suuntaan jättäen aina jälkeensä hännän. Ensimmäinen askel oli siis luonnollisesti poistaa häntä, ja liikuttaa pelkkää hahmoa itseään. Tämä onnistui kohtuullisen helposti vain poistamalla hännän piirtäminen ohjelmasta.

Seuraavaksi haluttiin saada mukaan hahmon kääntyminen. Tällaista ei oltu käsitelty kurssilla, joten asiaan piti itse perehtyä internetin avulla. Internetin ohjeiden avulla selvisi, että kääntyminen tapahtuu RotateTransform-ominaisuuden avulla. Käytännössä käännettävälle objektille lasketaan kulma Angle-metodin avulla, jonka mukaan ohjelma kääntää objektia Rotation metodissa. Näiden avulla hahmo saatiin sivulle liikkumisen sijaan kääntymään nuolinäppäimillä haluttuun suuntaan. Tämä ei kuitenkaan ollut vielä pelillisesti kovin hyvä ratkaisu, sillä hahmon haluttiin kääntyvän hiiren osoittimen suuntaan.

Hiiren osoittimen mukaisesti kääntyminen osoittautui yllättävän hankalaksi. Vaikka hiiren sijainnin kuunteleminen ei ollut hankalaa, saatiin hahmo aluksi lähinnä pyörimään joko myötä tai vastapäivään hiiren sijainnista riippuen. Tämä ratkaistiin ottamalla hiiren sijainti talteen pointtina, jota sitten hyödynnettiin aiemmin mainitussa rotation-metodissa.

## Ampuminen

Ampuminen oli luultavasti koko projektin haastavin osuus toteutuksensa kannalta. Itse ammus saatiin luotua samalla tavalla kuin pelihahmokin, mutta yleensä ammus jäi joko paikoilleen pelaajan kohdalle, tai hiiren kohdalla olevan tähtäimen kohdalle. Ampumisen ongelman kanssa painittiin monta tuntia tietoa etsien ja eri ratkaisuja kokeillen.

Monen tunnin tuskailun jälkeen ampuminen saatiin kuitenkin toimimaan vektorien avulla. Käytännössä ammus ottaa pelaajan sen hetkisen sijainnin ja luo tähän ammuksen. Seuraavaksi ammus ottaa hiiren sijainnin, josta tulee suunta ammuksella. Tämän jälkeen ohjelma pyörittää laskukaavaa laskien vektorien erotuksen, ja siirtää ammusta jatkuvasti haluttuun suuntaan. Tällä tavalla ammuksia voi tosin olla pelissä vain yksi kerrallaan. Lisäksi peli antoi joka kerta virheilmoituksen, mikäli uutta ammusta yritettiin laukaista liian aikaisin.

Useamman yhtäaikaisen ammuksen ongelma saatiin ratkaistua tekemällä ammuksista olioita ja luomalla näille lista, johon ne talletetaan. Lisäksi listaan tallennetaan myös jokaiselle ammukselle oma kohde, johon sen on tarkoitus matkata. Tämän jälkeen laskennat piti muuttaa niin, että yksittäisen ammuksen sijaan peli käy aina ammuslistaa läpi tarkkaillen jokaisen listassa olevan ammuksen sijaintia.

## Viholliset

Aluksi peliin saatiin luotua melko helposti staattisia vihollisia, jotka pysyivät vain paikallaan. Näistä vihollisista oli helppo katsoa sijainti, ja verrata sitä ammuksen sijaintiin. Aina jos ammuksen ja vihollisen sijainnit kohtasivat, annettiin pelaajalle pisteitä. Paikallaan pysyvät viholliset eivät kuitenkaan tarjoa juurikaan haastetta, joten oli oleellista saada viholliset liikkumaan. Vihollisen liikkumaan saaminen pelaajaa kohti oli sekin vaikeampaa kuin olisi osannut olettaa. Vihollisen sijainti piti määritellä vektorien avulla, joista taas laskettiin erotus pelaajan sijaintiin ja siirrettiin vihollista pelaajaa kohti. Sijaintien kohtaamiseen oli kohtuullisen helppo sijoittaa pelaajan kuolema eli pelin päättyminen.

Useamman vihollisen kohdalla oli sama ongelma kuin ammusten kanssa. Yksittäinen vihollinen toimi hyvin, mutta monta samaan aikaan ei läheskään niin hyvin. Aluksi tämä ratkaistiin tekemällä vihollisista olioita, joille annettiin jokaiselle oma sijaintivektori ominaisuutena ja laitettiin nämä viholliset taulukkoon. Peli sitten kävi tätä taulukkoa läpi siirtäen jokaista vihollista tarpeen mukaan kohti pelaajaa.

Taulukko ei kuitenkaan vastannut odotuksia siinä mielessä, että siihen ei voinut lisätä enempää vihollisia pelaajien edetessä. Niinpä taulukko täytyikin muuttaa listaksi ja peliin lisättiin ominaisuus, että aina yhtä ammuttua vihollista kohden peliin syntyy kaksi lisää tilalle.

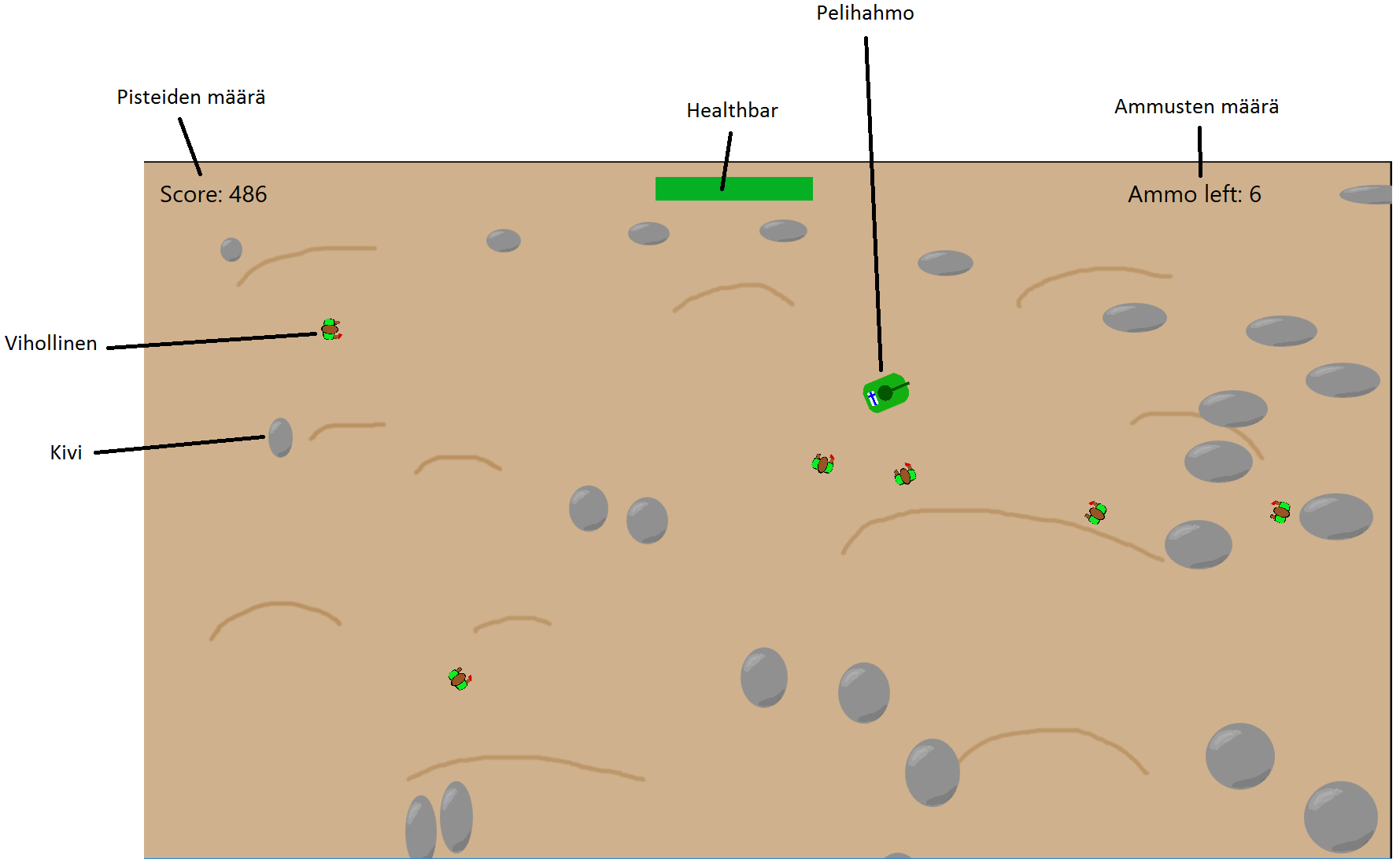
# Testaus

Peliä testattiin jatkuvasti samalla, kun sitä tehtiin. Esimerkiksi jokaista uutta ominaisuutta testattiin koko tekijätiimin voimin ja jokainen antoi mielipiteensä. Tekijöiden lisäksi peliä ja sen koodia kävivät katsomassa muut luokkalaiset ja opettaja, jotka antoivat yleensä palautetta, mitä tulisi kehittää.

Jatkuvan testaamisen ansiosta peli kehittyikin melko hyvällä tahdilla ja uusia ominaisuuksia saatiin lisättyä lähes jatkuvasti.

# Lopullinen versio

Alla kuva lopullisen pelin pelinäkymästä (ks. Kuvio 1).



Kuvio 2 pelinäkymä 19.4.2017 v 0.9

Pelin lopullisesta versiosta tuli yllättävänkin toimiva peli. Kaikki suunnittelut tärkeimmät ominaisuudet saatiin peliin, sekä osa lisäominaisuuksistakin. Lopullisessa versiossa viholliset liikkuvat pelaajaa kohti yrittäen vahingoittaa tätä. Mikäli viholliset saavuttavat pelaajan, pelaaja menettää energiaa, joka on kuvattuna peliruudun ylälaidassa. Energian loppuessa myös peli päättyy. Pelaaja voi puolustautua vihollisia vastaan ampumalla. Ampuminen on toteutettu niin, että hiiri toimii tähtäimenä ja laukaisu tapahtuu hiiren oikeasta painikkeesta. Mikäli pelaaja osuu viholliseen, vihollinen kuolee ja tästä kertyy pisteitä vasempaan ylänurkkaan. Ammusten määrä näkyy oikeassa ylänurkassa, ja ammusten loppuessa ase täytyy ladata.

Vihollisten ja pelaajan lisäksi pelialueelle on luotu myös esteenä toimivia kiviä törmäystarkastelu seuraa pelaajan ja vihollisten liikkeitä, ja mikäli pelaaja tai vihollinen osuu kiveen, vaikeutuu liikkuminen. Pelialueen reunojen yli liikkuminen on estetty pelaajalta, mutta vihollisilta ei.

Törmäystarkastelut ja liikkumiset pelissä on toteutettu vektoreiden avulla. Käytännössä peli vertailee jatkuvasti taustalla pelaajan, vihollisten, ammusten ja kivien sijaintia ja suorittaa eri metodeja mikäli esimerkiksi ammus kohtaa vihollisen.

# Itsearviointi

Projekti onnistui yllättävänkin hyvin. Kuten aiemmin mainittiin, peliin saatiin mukaan oikeastaan kaikki halutut ominaisuudet tyrmääviä grafiikoita ja ääniä myöten. Pelin aikana jouduttiin opettelemaan itsenäisesti paljon uusia asioita kuten vektoreiden ja äänien sekä kuvien hyödyntäminen liikkuvana grafiikkana. Uutta oli myös hiiren liikkeen tapahtumakäsittelijöiden käyttö.

Uusien asioiden lisäksi jouduttiin perehtymään tarkemmin kurssin aikana läpikäytyihin asioihin. Esimerkkinä luokkiin jakaminen, metodien siirtäminen luokkiin, tiedon sidonta (”terveyspalkki”) sekä refaktorointi. Varsinkin refaktorointia jouduttiin tekemään useaan kertaan, sillä koodia kertyi todella paljon uusia ominaisuuksia lisättäessä ja testattaessa.

Työtä projektin eteen tehtiin tasaisesti projektin alusta loppuun saakka ja työt saatiin jaettua tasaisesti koko ryhmän kesken. Ainoaksi ongelmaksi meinasi muodostua aika, sillä jokaisella tuli jatkuvasti mieleen uusia ominaisuuksia, joita peliin oltaisiin haluttu lisätä. Loppuvaiheessa koodia oltaisiin haluttu parantaa esimerkiksi törmäystarkastelujen osalta, mutta puutteellisen ajan vuoksi koko pelimekaniikkaa ei uskallettu laittaa uusiksi.

# Kommentit kurssista

Kurssi oli mielenkiintoinen, sillä oli hyvä päästä välillä tekemään graafisia sovelluksia pelkkien konsolisovellusten sijaan.