Olio-ohjelmointi 2 harjoitustyö loppuraportti

Ryhmä

Tuomas Kyttä H3408

Arttu Heinonen H3425

Joonas Katainen H3248

Joulukuu 2014

Ohjelmistotekniikan koulutusohjelma

Tekniikan ja liikenteen ala

Sisältö

[1 Johdanto 2](#_Toc405902776)

[2 Suunnitelma 2](#_Toc405902777)

[2.1 Lähtö suunnitelma 2](#_Toc405902778)

[2.2 Työnjako suunnitelma ja tavoitteet 3](#_Toc405902779)

[2.3 Toteutuksen suunnitelma 3](#_Toc405902780)

[3 Toteutus 4](#_Toc405902781)

[3.1 Toteutus tilakoneesta ja pelimoottorista 4](#_Toc405902782)

[3.2 Toteutus kartasta 8](#_Toc405902783)

[3.3 Toteutus hahmoista ja vihollisista 8](#_Toc405902784)

[4 Aikataulutus ja työn jako 8](#_Toc405902785)

[4.1 Tuomas Kyttä 8](#_Toc405902786)

[4.2 Joonas Katainen 8](#_Toc405902787)

[4.3 Arttu Heinonen 8](#_Toc405902788)

[5 Yhteenveto ja itsearviointi 9](#_Toc405902789)

[6 Viitteet 9](#_Toc405902790)

# Johdanto

Tämän kurssin harjoitustyönä toteutimme ja suunnittelimme 2 ulotteisen tasohyppelyvideopelin. Ryhmässä toimi kolme jäsentä, joille kullekin oli asetettu tehtäväksi ohjelmoida oma osuus peliä. Nämä pelin osuudet sitten koottiin yhdeksi kokonaisuudeksi, jonka kehitystä jatkettiin ryhmässä. Toteutuksessa käytimme SFML multimedia kirjastoa ja Githubin projektihallinta työkalua.

# Suunnitelma

## Lähtö suunnitelma

Aloitimme suunnittelun jo varhain syksyllä ja aloimme keräämään ideoita projektin toteuttamiseen. Melkein välittömästi kaikki ryhmän jäsenet tulivat tulokseen, että tekisimme harjoitustyö projektiksi pelin. Jatkoimme pohdintaa millaista peliä alkaisimme kehittämään ja kokosimme ajatuksia videopelistä, mikä olisi suhteellisen mahdollista toteuttaa annetuilla resursseilla ja mikä meitä ryhmänä kiinnostaisi tehdä. Päädyimme lopulta ideaan 2D tasoloikka pelistä, jossa maasto ja viholliset luotaisiin satunnaisesti. Pelaajalla olisi myös mahdollisuus ampua vihollisia ja kerätä satunnaisesti putoavia voimavaroja, joilla pelaajasta tulisi tehokkaampi. Kokonaisen valmiin pelin saaminen valmiiksi kuulosti toki isolta ja työläältä urakalta, mutta halusimme kokeilla rajojamme ohjelmisto kehittäjinä ja testata kuinka pitkälle pystyisimme kehittämään tätä projektia. Pitkin syksyä heittelimme toisillemme ideoita pelimekaanikoista ja teknisen puolen toteutuksista, joista pikku hiljaa alkoi hahmottua selkeämpi ja selkeämpi kuva siitä mitä meidän oli aikomus tehdä.

## Työnjako suunnitelma ja tavoitteet

Kaikki olivat yhtä mieltä siitä, että olisi kaikille mukavampaa, jos jokainen saisi oman itsenäisen osan ohjelmasta, jota voisi kehittää. Että jokaisella olisi oma pieni ”nurkka” koodista jonka parissa kukin pystyisi työskentelemään, kunnes lopulta osat kasattaisiin yhteen ja niistä luotaisiin peli kokonaisuus. Työnjaossa sovittiin, että Tuomas Kyttä tekisi karttojen luomisen, tallentamisen ja satunnaisen generoinnin pelimaailmaan pelaajan ja vihollisten ympärille. Arttu Heinonen tekisi pelin rungon, päivityksen, piirron ja pelimoottoria. Joonas Kataisen tehtävänä oli luoda peliin sisältöä, kuten vihollisia ja tavaroita joita poimia. Tarvittaessa muut auttaisivat myös muita. Sovittiin, että kukin sai aloittaa milloin itselle parhaaksi sopi, kunhan vain työt eivät estäisi muiden töitä.

## Toteutuksen suunnitelma

Koska peliin oli suunniteltu monta erilaista tilaa, niin pelin runkoon kuului tehdä toimiva tilakone, joka pitää kaikki pelitilat saman katon alla. Kun ohjelman aluksi käynnistäisi, menisi peli päävalikkotilaan, josta pystyisi vaihtamaan tilaa kartanluontiin, pelitilanteeseen ja takaisin päävalikkoon. Päävalikon alle oli myös suunniteltu asetusvalikko, josta pystyisi vaihtamaan erinäisiä peliin liittyviä asetuksia. Asetukset tallennettaisiin tiedostoon ja luettaisiin aina pelin avautuessa uudelleen.

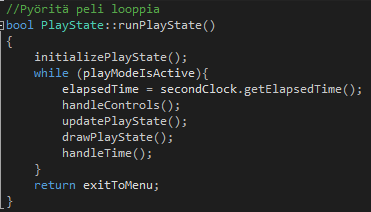
Kartanluontiin suunniteltiin käytettäväksi satunnaista kartanluontia ennalta määrätyistä karttapaloista. Nämä karttapalat piti erikseen rakentaa ja siksi sen helpottamiseksi tehtiinkin karttapalaeditori. Karttapalaeditorissa pystyisi lisäämään karttapaloihin ennalta määrättyjä karttaobjekteja.

Vihollisiin ja hahmoihin suunniteltiin, että luodaan pelaajalle ja vihollisille yhteinen kantaluokka, jonka molemmat perisivät. Hahmot päivitettäisiin ja piirrettäisiin kaikki erillisesti ja ne tarkastettaisiin törmäiltävän maaston kanssa. Pelaajalle uniikkeja ominaisuuksia olisi mm. pelaajan ohjaaminen näppäimistöllä. Vihollislitania tallennettaisiin linkitettyyn listaan ja käsiteltäisiin sieltä.

# Toteutus

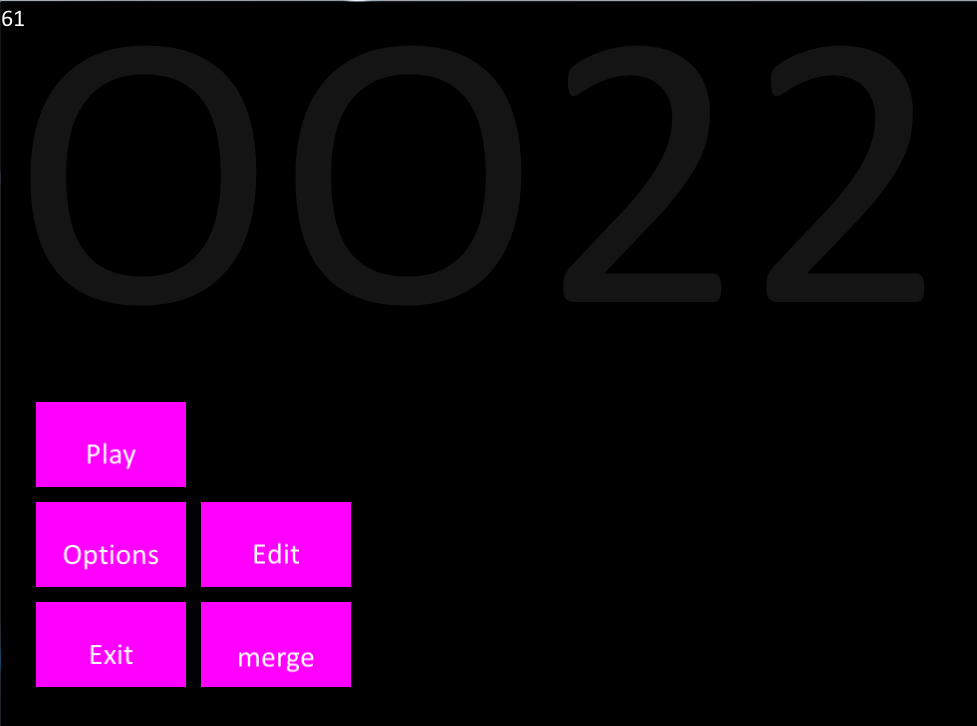
## Toteutus tilakoneesta ja pelimoottorista

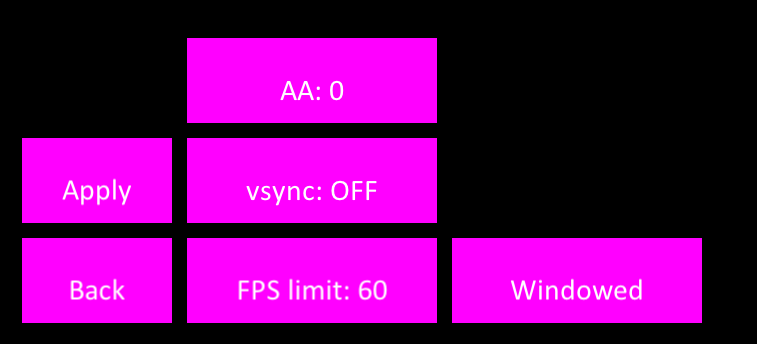
Pelimoottorin ydin on tilakone. Tilakone hallitsee mitä tilaa pelissä pyöritetään ja vaihtaa sitä switch case:lla, kunnes tulee tila käsky lopettaa ohjelma. Tiloista itsessään olisi voinut tehdä virtuaalisen luokan, josta olisi voitu luoda muut tilat, mutta ajan puutteen ja käytännöllisen hyödyn olemattomuus jätti sen tekemisen pois. Ohjelman tilojen pääidea on kiteytettynä seuraavanlainen:



Eli, tila alustetaan, jonka jälkeen siirrytään tilan pyöritys looppiin. Loopissa katsotaan kauan viime looppiin meni aikaa, käsitellään näppäimet, päivitetään tila, piirretään tila ja käsitellään aika.

Päävalikkotila on ensimmäinen tilakoneen tuottama tila, joka aukeaa käynnistäessä. Päävalikon alussa alustetaan valikkojen napit ja ladataan käyttäjän asetukset asetustietueeseen. Jos asetuksia ei ole, ne luodaan. Asetusten määritykset vaikuttavat pelin suoritukseen. Esimerkiksi anti aliasing vaikuttaa kuinka pehmeältä kuvat näyttävät. Päävalikon päivityksessä kuunnellaan lähinnä hiirtä, ja sitä painetaanko ruudulla näkyviä nappeja. Jos Esc painiketta tai ruudun raksia painaa, niin tilakone siirtyy ”exit” tilaan, jolla ohjelma sulkeutuu. Päivityksessä säädetään myös otsikon näkyvyyttä hieman näkyvämmäksi, kunnes se on täysin näkyvä.

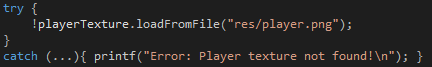




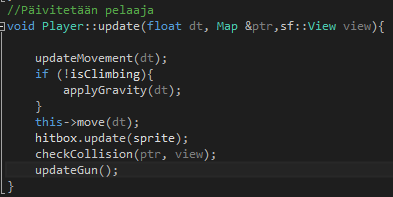
Oikeassa yläkulmassa näkyy fps lukema, jonka saa näkyviin/piiloon F1 näppäimellä. Siirtyessä Play, Edit tai Merge tilaan ladataan kartta tiedostosta, jossa kestää tovi. Tätä varten ruudulle piirretään latausruutu.

Pelitilassa ladataan kartta, pelaaja ja generoidaan viholliset. Vihollisten generointi toimii satunnaisesti kartan koon mukaan. Viholliset generoidaan linkitettyyn listaan, josta ne myös päivitetään ja piirretään.

Pelaajan luonnin yhteydessä ladataan sprite muuttujaan Content luokasta tekstuuri. Jos pelaajan tekstuuria ei syystä tai toisesta löydy, niin heitetään konsoliin virheilmoitus. Pelaajan tekstuurin puuttuminen itsessään ei ole kriittistä ohjelman suorittamiselle, mutta pelaajan tilalle tulee vain valkoinen suorakulmio.



Pelaajalle myös luodaan gun olio, joka hoitaa pelaajan ampumiskäden. Nykyisessä versiossa sen käyttö on tosin jouduttu aika syistä karsimaan. Pelaaja myös periytyy Character luokasta, jossa hoidetaan liikkuminen, painovoima ja piirtäminen.

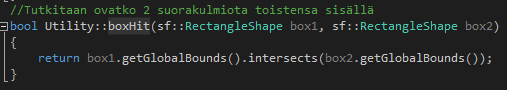


Pelaajan päivityksessä hoidetaan sen liikkuminen, painovoima, Hitboxien seuraaminen pelaajaa, törmäystarkastelu ja aseen erillinen päivitys. Liukuluku arvo dt edustaa delta-aikaa. Aikaa, kuinka suuri eri ruudunpäivitystahdilla on. Ilman tätä arvoa ruudunpiirtonopeus vaikuttaisi liikkumisnopeuteen ja sitä emme halua. Metodeissa kummittelee myös Map tyyppinen pointteri kartan paloihin. Tätä tietoa tarvitaan, kun tutkitaan osuuko pelaaja kartanpaloissa oleviin objekteihin.

Pelaajan törmäystarkastelu onkin hieman hankalampi ja olisi varmasti pystytty toteuttamaan viisaammin. Periaate on seuraavanlainen:

* Asetetaan törmäysarvot epätosiksi
* Tutkitaan pelaajan nykyinen karttasektori
* Loopataan pelaajan ympärillä olevat karttasektorit, jos sektori on sallittujen rajojen sisällä
* Käydään läpi sektorin jokainen pala
* Jos kartan objektiin voi törmätä, niin tarkistetaan törmättävät objektit pelaajan kaikkien Hitboxien kanssa
* Jos kyseessä ei ole törmättävä objekti, vaan esm. tikkaat ja pelaaja koskee siihen, annetaan mahdollisuus kiipeämiseen
* Kun kaikki objektit on käyty, niin korjataan arvoja riippuen mihin törmättiin

Törmäystä tarkastellaan Utility luokassa. Metodiin annetaan kaksi suorakulmaista laatikkoa ja metodissa katsotaan koskevatko laatikot toisiinsa.



Viholliset käyttävät suurin piirtein samaa logiikkaa.

## Toteutus kartasta

## Toteutus hahmoista ja vihollisista

# Aikataulutus ja työn jako

<https://github.com/JamkOlioOhjelmointi2HarkkaTiimiHKK/oo22/commits/master?page=1>

Oheisessa linkissä näkyy kunkin työnjako ja päivityshistoria projektille.

Projekti tehtiin GitHubia käyttäen, jonka ansiosta kaikki projektiin liittyvät lisäykset ja päivitykset näkyvät helposti koottuna commit-historiana. Listasta näkee milloin, mitä ja kuka teki mitäkin. Projektin virallinen työstäminen aloitettiin 23.10.2014. ja lopetettiin 9.12.2014.

## Tuomas Kyttä

## Joonas Katainen

## Arttu Heinonen

# Yhteenveto ja itsearviointi

Loppujen lopuksi pelinkehitys jäi erittäin kesken, mutta saimme kuitenkin aikaan paljon ja pelinkehitys onnistui yllättävän pitkälle. Toiveet pelinlopputilasta oli kaikilla hieman saavutettua korkeammat, mutta voimme kaikki olla ylpeitä saavutuksestamme ja opimme projektin aikana paljon asioita. GitHubin käyttäminen auttoi projektin etenemistä merkittävästi ja ilman sitä emme varmaan olisikaan päässeet näin pitkälle. Siitä huolimatta koodin yhdistämisessä tuli moneen otteeseen hankaluuksia ja aiheutti ongelmia usealla taholla.

Projektia olisi auttanut selkeämpi suunnitelma ja paremmat rajapinnat koodeille, joita muut olisivat voineet käyttää.

# Viitteet

<https://github.com/JamkOlioOhjelmointi2HarkkaTiimiHKK/oo22>

<http://sfml-dev.org/tutorials/2.1/>