Loppuraportti

Käyttöliittymien ohjelmointi

Kari Vesamäki

Miska Sainkangas

Ville Rantala

Harjoitustyö

Huhtikuu 2016

Tieto- ja viestintätekniikan koulutusohjelma

Tekniikan- ja liikenteen ala

Sisällysluettelo

[1 GitHub-sivu 2](#_Toc449640519)

[2 Lopullinen rakennekaavio 2](#_Toc449640520)

[3 Luokkarakenne 3](#_Toc449640521)

[4 Työaikaraportti 8](#_Toc449640522)

[5 Näyttökaappaukset 9](#_Toc449640523)

[6 Vakionäppäimet 13](#_Toc449640524)

[7 Keskeisimmät ongelmat ja niiden ratkaisut 14](#_Toc449640525)

[8 Testaus 15](#_Toc449640526)

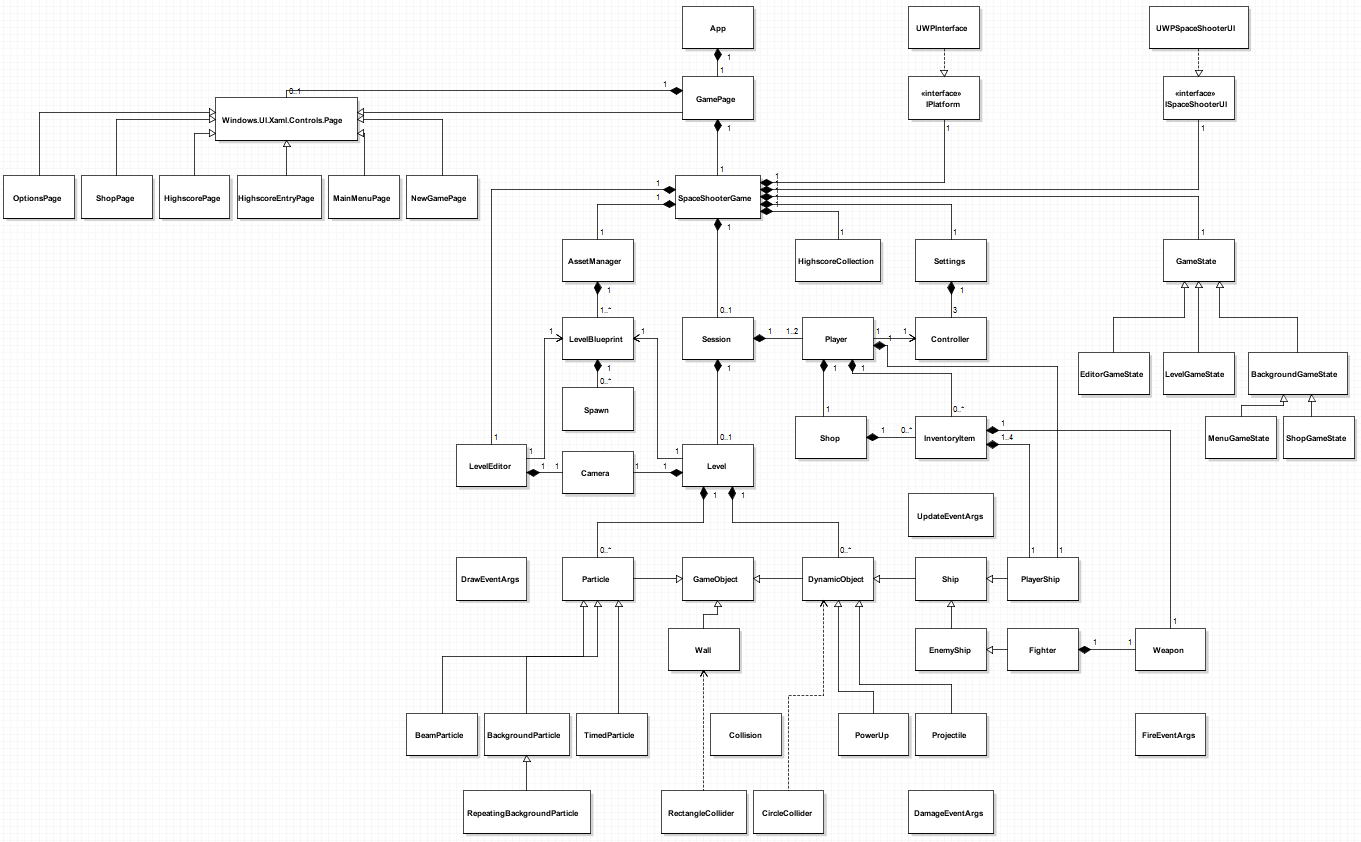
[9 Itsearviointi 16](#_Toc449640527)

[10 Kommentit kurssista 16](#_Toc449640528)

# GitHub-sivu

<https://github.com/VilRan/SpaceShooter>

# Lopullinen rakennekaavio



Kuva

Linkki GitHubista löytyvään isompaan kuvaan <https://raw.githubusercontent.com/VilRan/SpaceShooter/master/Documentation/ClassDiagram%20v0.2.PNG>

Tämä on raskaalla kädellä karsittu versio, joka jättää ulos suurimman osan konkreettisista abstraktien-luokkien toteutuksista. Kokonainen versio sekoittaisi enemmän kuin selventäisi.

# Luokkarakenne

GamePage

GamePage sisältää itse pelin ja SwapChainPanel-elementin, jolle se piirretään. Lisäksi sivu voi pitää sisällään toisen sivun, jolloin alisivu piirretään pelin päälle.

MainMenuPage

XAML-sivu, joka sisältää päävalikkoon liittyvät elementit ja niiden toiminnollisuuden.

OptionsPage

Sivu, jolla voi säätää pelin asetuksia, kuten äänenvoimakkuudet ja näyttömoodin.

HighscorePage

Näyttää kaikki tallennetut pisteet suurimmasta pienimpään.

HighscoreEntryPage

Ilmestyy, kuten palaaja(t) häviää pelin. Näyttää paljonko pelin aikana pisteitä kertyi ja pyytää syöttämään nimen.

NewGamePage

Sivu, jolla valitaan uuden session vaikeusaste ja pelaajien määrä.

ShopPage

Sivu, jolla pelaajat voivat valita aseistuksen ennen tason alkua.

SpaceShooterGame

Pelin pääluokka, joka sisältää (käyttöliittymää lukuun ottamatta) kaiken peliin kuuluvan sisällään. Se luodaan kerran ohjelman suorituksen alussa ja sitä käytetään sen loppuun asti. Periytyy MonoGamen Game-luokasta.

IPlatform, ISpaceShooterUI

Rajapintoja, joiden kautta peli keskustelee alustan ja käyttöliittymän kanssa. Tällä hetkellä toteutukset löytyy vain UWP:lle.

Settings

Pitää sisällään pelin asetuksiin liittyviä tietoja, kuten äänien voimakkuuden säädöt, näppäimistön määritykset ja näyttömoodin.

AssetManager

Luokka pitää tallessa viittaukset pelin käyttämiin assetteihin, kuten tekstuureihin ja LevelBlueprintteihin. Säilyy koko ohjelman suorituksen ajan.

HighscoreCollection

Kokoelma, joka sisältää pistetilastot ja niiden lataamiseen ja tallentamiseen tarvittavat metodit.

Session

Luodaan kun pelaaja aloittaa uuden pelin. Sisältää kaikki kenttien välillä säilyvät tiedot kuten pistemäärän ja pelaajien tiedot. Sisältää lisäksi aktiivisen kentän. Tämän luokan sisältö tallennettaisiin, kun peli tallentuu, jos tallennus olisi jo toteutettu.

Player

Session-luokan palvelija, joka pitää yllä yksittäisten pelaajien tietoja, kuten rahan määrän ja käytetyn Controllerin.

GameState

Abstrakti luokka, jonka toteutukset ohjaavat pelin toimintaa. Tämä päättää, mitä olioita päivitetään ja piirretään ja mitä musiikkia soitetaan.

Level

Näyttämö, jolla pelin pääesitys tapahtuu. Se sisältää listat esiintyjistä (DynamicObject) sekä lavasteista (Particle.)

LevelBlueprint

Malli, jonka pohjalta tasoja luodaan. Voidaan satunnais-generoida, ladata tiedostosta tai luoda editorilla. Nyt jälkikäteen ajatellen olisi ehkä ollut parempia idea vain kloonata Levelit ja niiden sisällöt.

LevelEditor

Työkalu, jolla voi muokata blueprinttien sisältöä ja tallentaa ja ladata niitä. Pitäisi ehkä muuttaa yhdeksi GameStateista.

Spawn

Abstrakti luokka, jonka velvollisuus on säilyttää tiedot siitä, missä kohtaa ja millä vaikeusasteilla objektien on ilmestyttävä tasolle. Perivissä luokissa voitaisiin lisäksi määritellä muitakin ominaisuuksia.

Camera

Käytetään ruudun vierittämiseen Levelin laidalta toiseen. Levelin etenemistahtia voidaan muokata muokkaamalla kameran etenemisnopeutta. Lisäksi määrittää alueen, jolla objektit ovat aktiivisia.

GameObject

Abstrakti luokka, joka sisältää pelissä näytöllä liikkuvien olioiden perusominaisuuksia, kuten paikka, nopeus ja tekstuuri.

DynamicObject

Abstrakti luokka, joka määrittelee pelimekaanisesti peliin osallistuvien olioiden yhteisen käyttäytymisen. Tähän kuuluu esimerkiksi törmäyksentarkistus muiden objektien kanssa ja toisena esimerkkinä apumetodi, joilla voidaan löytää oliota lähin olio, joka täyttää vaaditut ehdot.

CircleCollider

Rakenne, jolle DynamicObject delegoi törmäyksentarkistuksen. Tällä hetkellä jokaista DynamicObjectia kohden käytetään vain yhtä, mutta tulevaisuudessa voisi olla monimutkaisempia objekteja jotka koostuisivat useammasta Collider-elementistä.

Ship

Abstrakti luokka, joka periytyy DynamicObject-luokasta. Määritellään alusten ominaisuuksia esimerkiksi Healthbar:n piirtämisen ja tuhoutumisen erikoistehosteet.

Projectile

Abstrakti luokka, joka periytyy DynamicObject-luokasta. Määrittää ammuksille yhteisen käyttäytymisen. Käytännössä tämä tarkoittaa, että luokan oliot liikkuvat suhteessa kameraan ja törmäävät seiniin.

PowerUp

Abstrakti luokka, joka periytyy DynamicObject-luokasta. Määrittää Powerup:ien ominaisuuksia, esimerkiksi että ne voivat vain törmätä pelaajan alukseen.

Weapon

Abstrakti luokka, joka määrittää kaikille aseille yhteisen käyttäytymisen.

Controller

Controller-luokka hoitaa näppäinten kartoittamisen pelin sisäisiin toimintoihin.

Wall

Staattinen este. Luokka on toimiva, mutta vielä keskeneräinen.

RectangleCollider

Tällä hetkellä vain staattinen kohde johon CircleCollider:it voivat törmätä. Käytetään Wall:in kanssa.

Particle

Kevyt ja "tyhmä" olio, jota päivitetään vain piirto-syklin aikana. Käytetään erikoistehosteissa eikä vaikuta suoraan pelimekaniikkaan.

TimedParticle, BackgroundParticle, RepeatingBackgroundParticle, BeamParticle

Partikkeleita, joilla on hieman kehittyneempää toiminnollisuutta. TimedParticle on olemassa rajatun ajan kunnes katoaa. BackgroundParticle:lla on etäisyys, joka vaikuttaa sen liikkumisnopeuteen ruudulla. RepeatingBackgroundParticle looppaa ruudun ympäri. BeamParticle piirtää jatkuvan viivan.

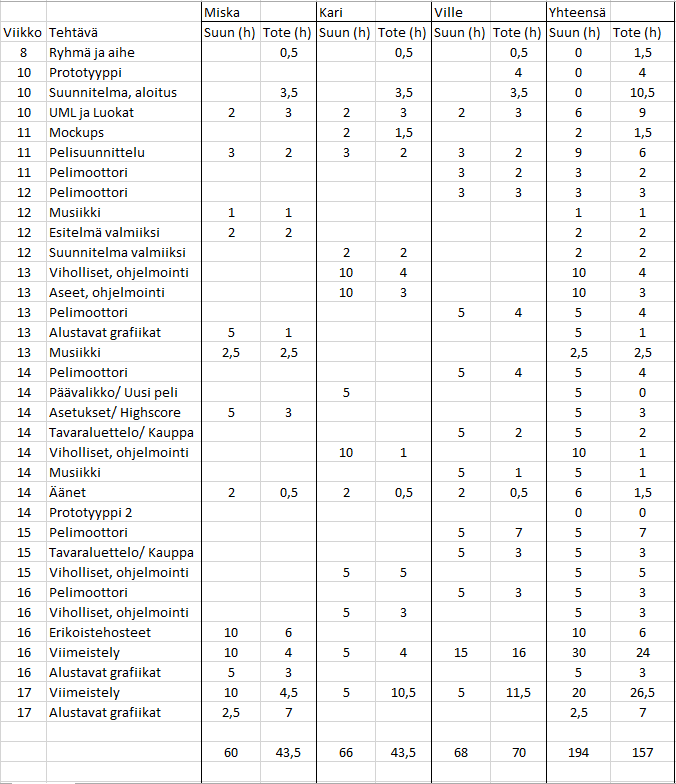
Shop, InventoryItem

Shop on jokaiselle pelaajalle uniikki ja sisältää listan tavaroista, joita pelaaja voi ostaa. InventoryItem on tavara, johon on lisätty hintalappu.

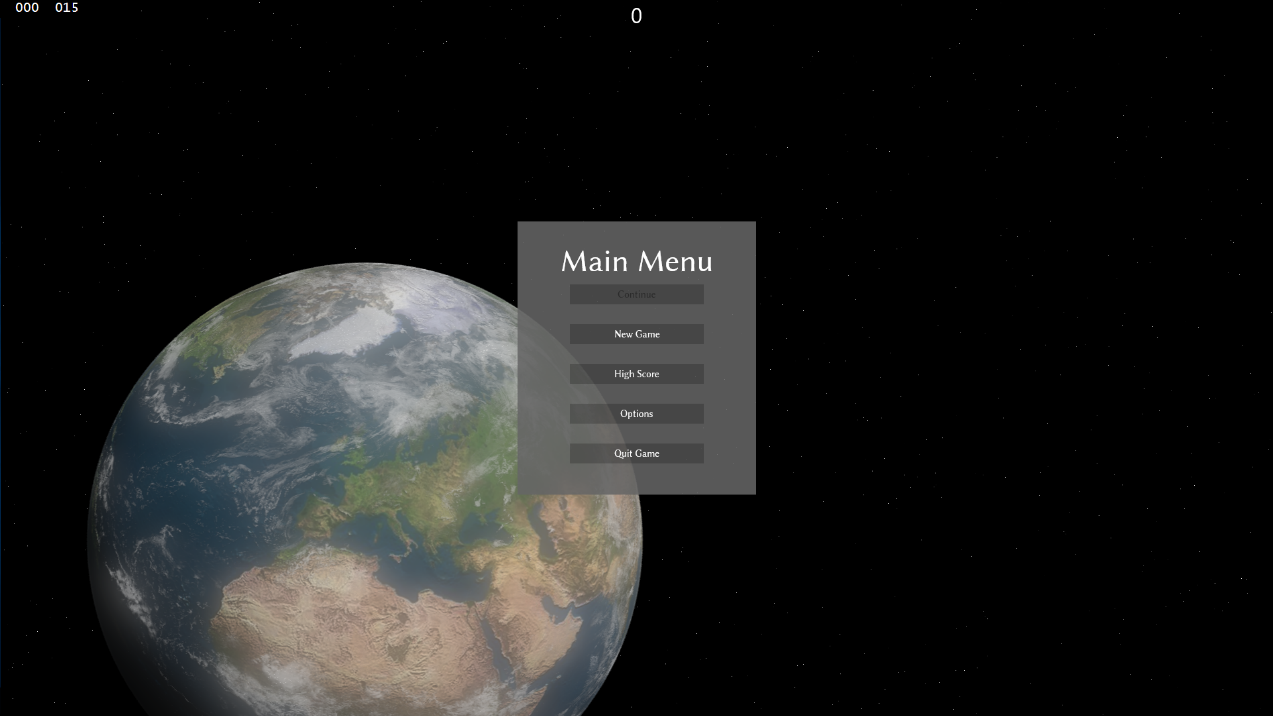
WeightedList

WeightedList on satunnaisgeneroinnin apuna käytetty geneerinen kokoelma. Jokaiseen objektiin listalla on lisätty painoarvo, joka määrittää sen, kuinka suurella mahdollisuudella se valitaan suhteessa toisiin listalla oleviin objekteihin.

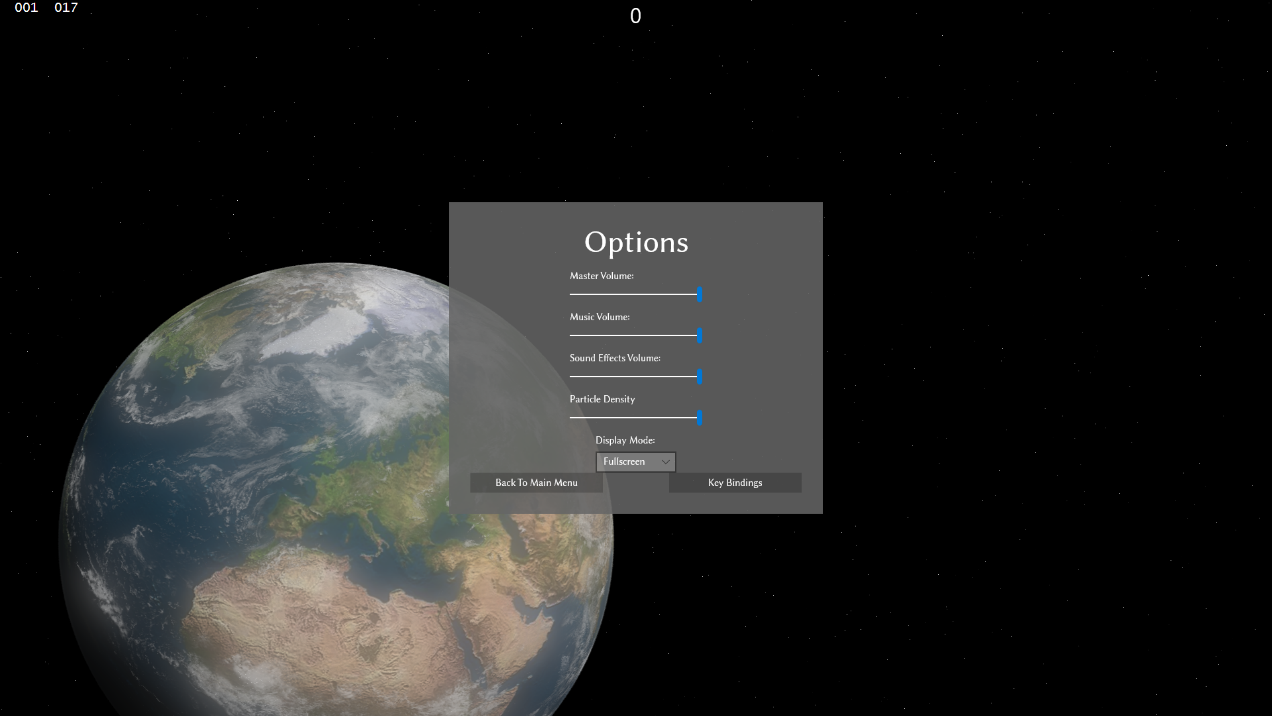
# Työaikaraportti



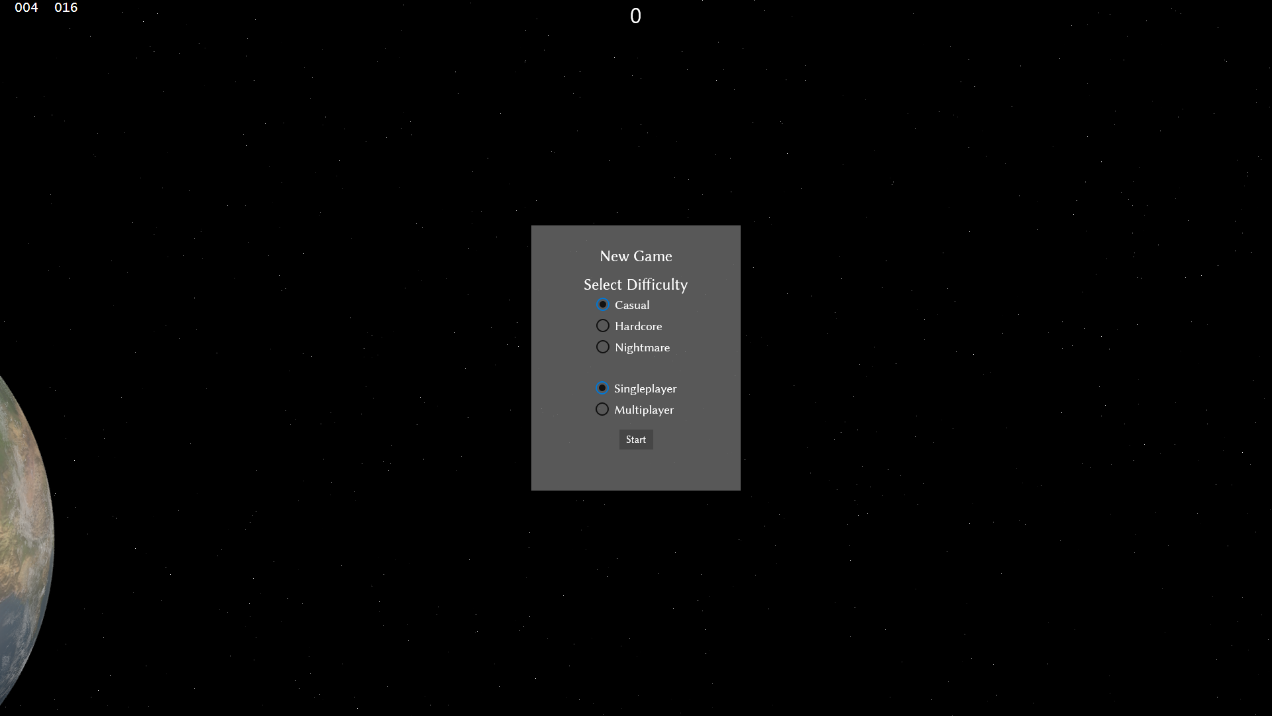
# Näyttökaappaukset



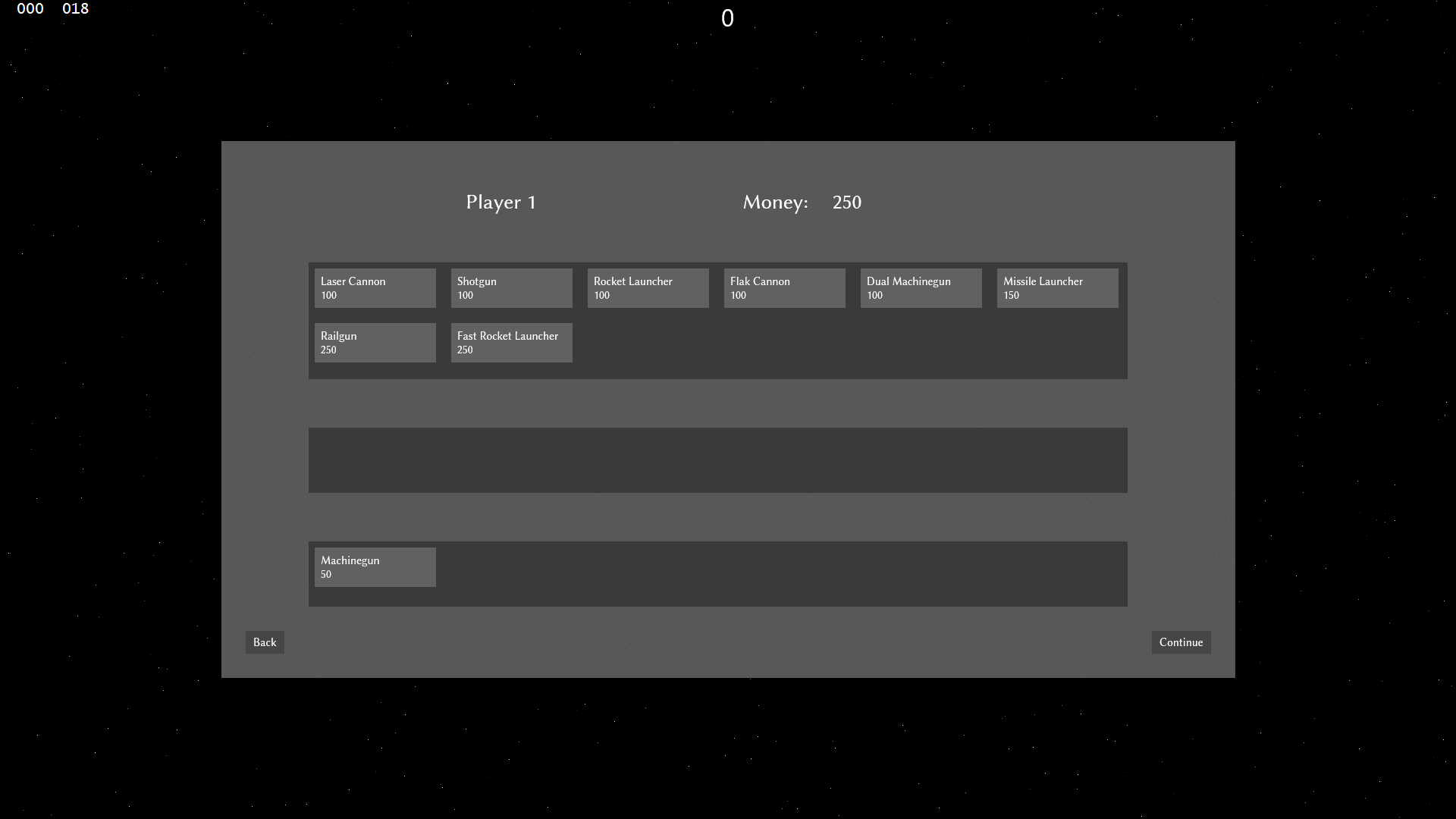
Kuva 2 Päävalikko



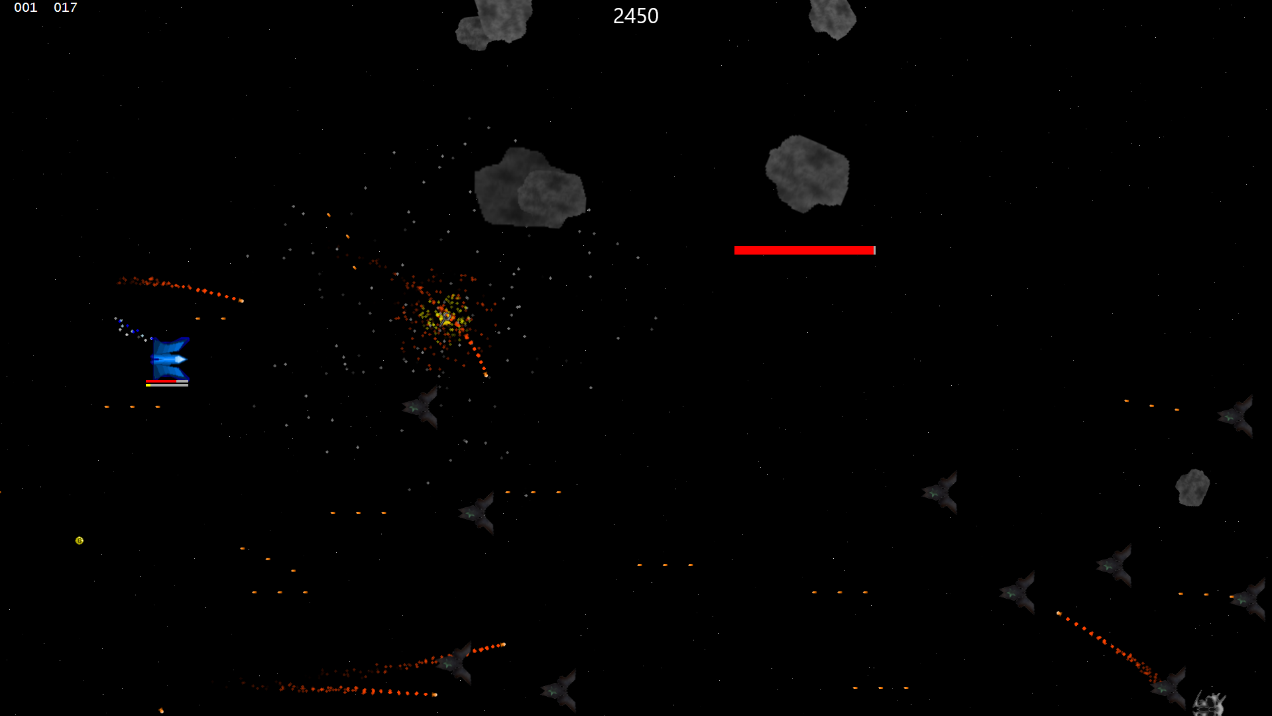
Kuva 3 Asetus-valikko



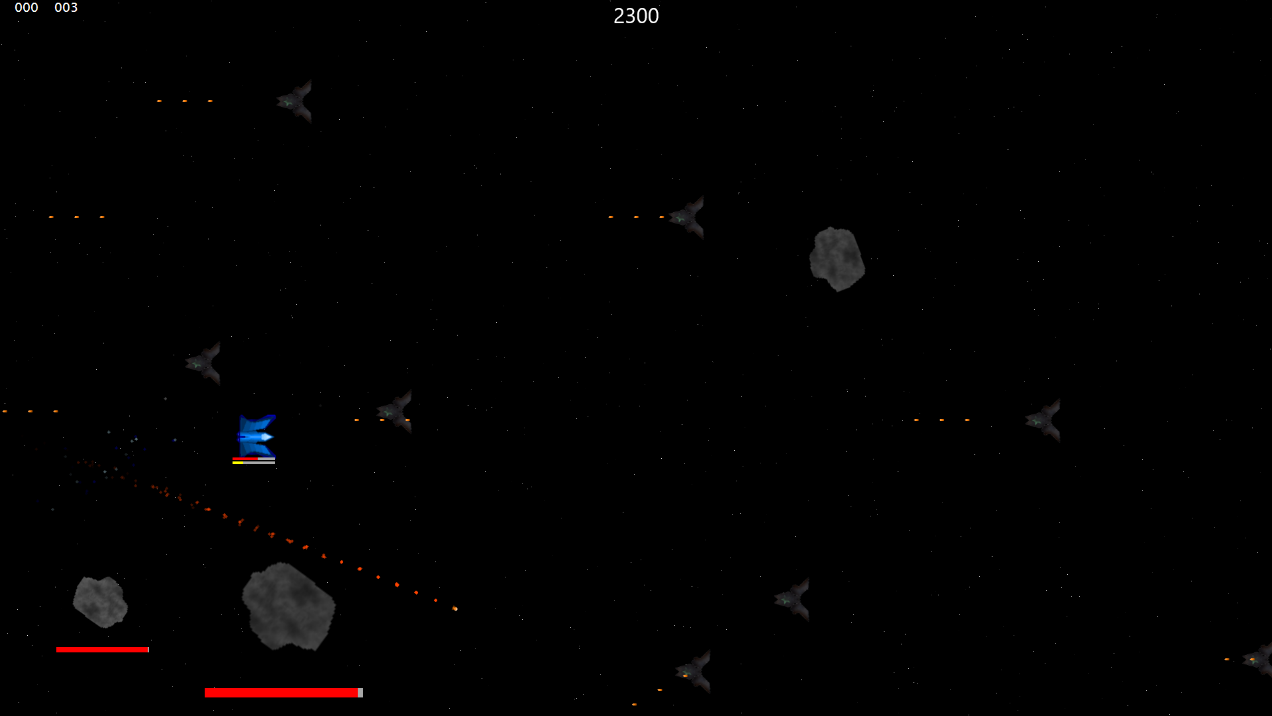
Kuva 4 Uuden pelin aloitusruutu



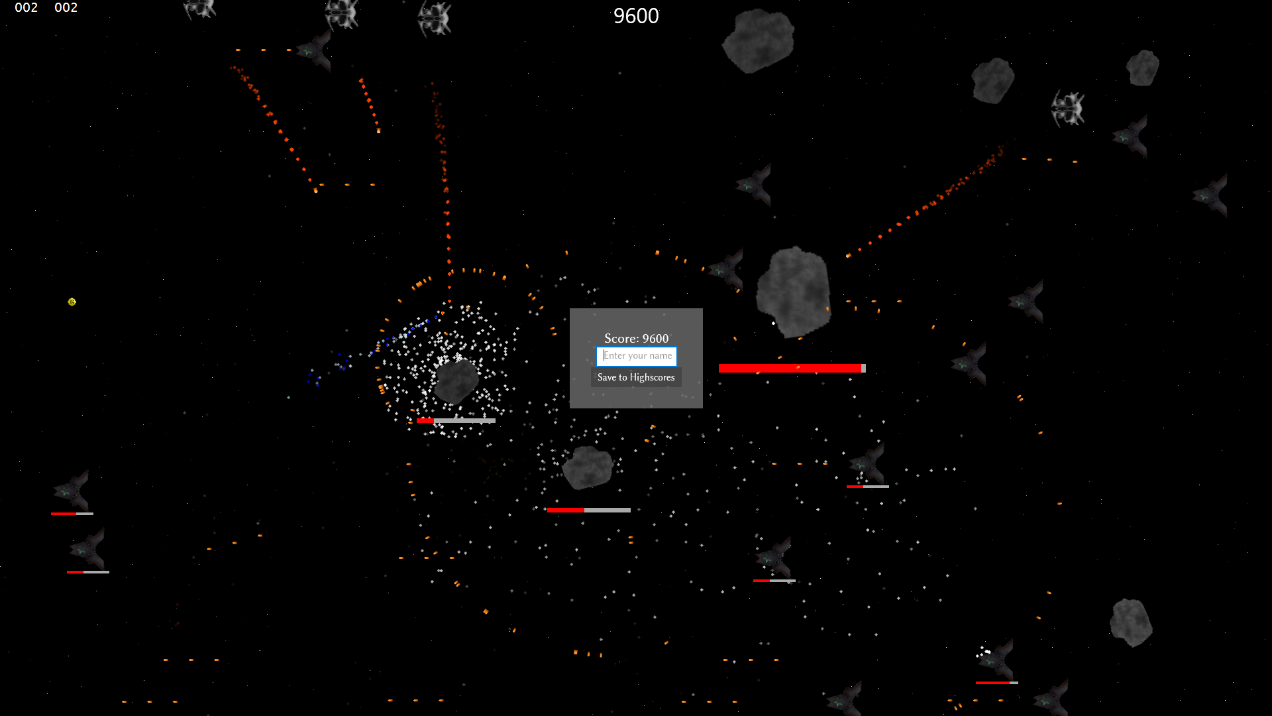
Kuva 5 Kauppa



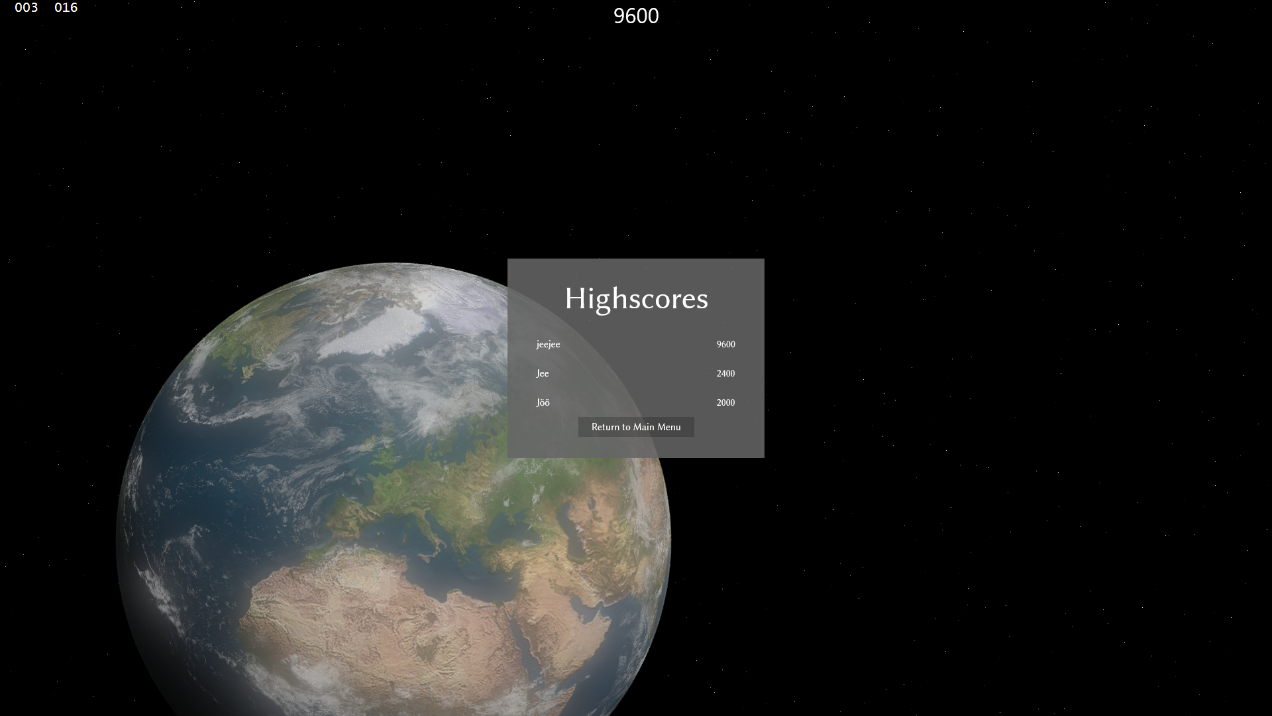
Kuva 6 Pelitilanne



Kuva 7 Toinen pelitilanne



Kuva 8 Peli hävitty, pyytää pelaajan nimeä



Kuva 9 Pisteet

# Vakionäppäimet

* Pelaaja 1
  + Liikkuminen – Nuolinäppäimet, Numpad
  + Ampuminen – Space, Enter
  + Lataaminen – Backspace
  + Aseen vaihto – 1, 2, 3, 4
  + Seuraava ase – Numpad 9
  + Edellinen ase – Numpad 7
* Pelaaja 2
  + Liikkuminen – WASD
  + Ampuminen – Left Ctrl
  + Lataaminen – R
  + Seuraava ase – E
  + Edellinen ase – Q
* Yleiset
  + Päävalikko – Escape
  + Kokoruututila – F11
  + Editori – F8
  + Paussi – P
* Editori (väliaikaisia)
  + Tallennus tiedostoon – F
  + Lataus tiedostosta – O
  + Testaus – T
  + Tyhjennys – C

# Keskeisimmät ongelmat ja niiden ratkaisut

Karille vastaan tulleita ongelmia:

* MachinegunDrone:n liikkuminen
  + Tarkoitus oli saada Drone kiertämään pelaajaa ympäri ja ampumaan vihollisaluksia. Tämä toiminta oli tarkoitus toteuttaa matriisien avulla. Ongelmia aiheutti Dronen kiertoliike. Se kyllä seurasi pelaajaa hyvin, mutta kierto pelaajan ympäri puuttui. Ongelma vei turhan paljon aikaa ja kulutti hermoja, varsinkin kun ratkaisu oli vain yhden muuttujan alustaminen väärässä paikassa…
* AceFighter:n liikkuminen
  + Tarkoitus oli saada AceFighter liikkumaan jaksoissa nii, että ensin se jahtaisi pelaajaa ja tietyn ajan jälkeen siirtyisi jonnekin päin ruutua ja pysähtyisi paikoilleen ja ampuisi pelaajaa kohti raketeilla. Ongelmia aiheutti liikkeen pysäyttäminen. Se meni kyllä tiettyyn pisteeseen ruudulla, mutta liike jatkui senkin jälkeen. Ratkaisu oli kameran liikkeen huomioiminen, kun AceFighterin oli tarkoitus pysähtyä.
* AdvancedFighter:n toiminta
  + Tarkoitus oli saada Fighter jahtaamaan pelaajaa ja tarpeeksi lähelle päästyään pysähtymään pelaajan eteen ampumaan. Ongelmia aiheutti liikkeen pysäytys. Fighter ei tuntunut ikinä pääsevän perille pysäytyspisteeseen, koska pelissä kaikki liikkuu koko ajan. Ratkaisu oli ”toleranssin” lisääminen Fighterin ja pysäytyspisteen etäisyyteen.

Villen kohtaamat ongelmat:

* Navigaatio
  + Jo projektin alkuvaiheessa tuli selväksi, että UWP:n standardi navigaatio ei toimisi tässä projektissa. Aina, kun yritti navigoida GamePage:lle, ohjelma latasi pelin uudelleen ja kaikki tilatiedot menetettiin. Aluksi yritin ottaa pelin talteen App-luokkaan ja kiinnittää sitä uudestaan SwapChainPanel-elementtiin, mutta tämä osoittautui liian mutkikkaaksi. Ongelma ratkesi väliaikaisesti ottamalla GamePage talteen ja asettamalla se suoraan Window.Current.Content:iin. Myöhemmin asetin GamePage:n pysyvästi taustalle ja loin muut sivut tarvittaessa sen päälle.
* XAMLin käyttäminen pelin HUDissa
  + Alun perin pelin HUD oli tarkoitus toteuttaa XAML elementteinä, mutta projektin edetessä huomasin, että peliä testatessa näppäimistö meni välillä mystisesti lukkoon. Ammuslaskuria - joka joutui päivittämään itseään hyvin tiheään - tehdessäni sain viimein vihjeen siihen, mistä ongelma johtui. Koko XAML HUD idea meni roskakoriin, ja vielä melko myöhäisessä vaiheessa. Tilalle kyhäsin pikaisesti pelin sisäiset yksinkertaiset palkit.
* Tiedostojen hallinta
  + UWP:n tiedostonhallinta aiheutti aluksi hieman hämmennystä, mutta hetken selailun ja parin yrityksen jälkeen siitäkin päästiin yli. Järjestelmän perinteisestä poikkeava malli kuitenkin sai minut luomaan rajapinnan sen ja pelin välille, jotta peli voitaisiin saada toimimaan muissakin järjestelmissä.
* Aika
  + Ideoita parannuksiin olisi riittänyt vaikka kuinka, mutta aika ei vaan riittänyt kaiken haluamani toteutukseen.

# Testaus

Varsinaista testaussuunnitelmaa ei ollut, vaan koodia testattiin sitä mukaa kun sitä tuotettiin. Mikäli virheitä havaittiin ja niiden syy ei ollut heti selvä, asiaa tutkittiin VisualStudion debuggerin avulla. Tämä yleensä riitti vikojen löytämiseen ja korjaamiseen.

# Itsearviointi

Karin itsearviointi:

Projekti oli kohtuullisen haastava, koska aikaisempaa koodaustaustaa ei juurikaan ollut, mutta pelin tekeminen oli ihan hauskaa puuhaa. Jos projektin alussa koodaustaidot olivat 1-2:n luokkaa, niin ehkä projektin myötä ne nousivat sinne kolmoseen.

Miskan itsearviointi:

Harjoitustyö oli haastava, koska olin tutustunut ennen koulua vain vähän Python koodauskieleen. Pelin tekeminen C# -kielellä oli mielenkiintoista ja se innosti tekemään entistä parempaa koodia jatkossa. Koodaustaitoni kehittyivät mielestäni 2 arvosanasta arvosanaan 3 harjoitustyön tekemisen aikana.

Villen itsearviointi:

Tunnen hieman epäonnistuneeni tässä projektissa. En koodini kannalta, vaikkei se olekaan niin täydellistä kuin haluaisin, vaan kaiken muun kannalta. Valitsin ehkä tiimilleni liian laajan ja hankalan aiheen. Aikaa ja taitoa ei kaikilta riittänyt kaiken tekemiseen ja sen seurauksena kärsivät projektin kaikki osa-alueet jotka eivät liittyneet pelimekaniikkaan. Tavoitteena oli 5, mutta antaisin itselleni enintään 4.

# Kommentit kurssista

Käytännönläheisyys ja vapaa työskentely maistui. Ei valittamista.