Loppuraportti

Käyttöliittymien ohjelmointi

Kari Vesamäki

Miska Sainkangas

Ville Rantala

Harjoitustyö

Huhtikuu 2016

Tieto- ja viestintätekniikan koulutusohjelma

Tekniikan- ja liikenteen ala

Sisällysluettelo

[1 Lopullinen rakennekaavio 2](#_Toc449543972)

[2 Luokkarakenne 2](#_Toc449543973)

[3 Työaikaraportti 5](#_Toc449543974)

[4 Näyttökaappaukset 6](#_Toc449543975)

[5 Keskeisimmät ongelmat ja niiden ratkaisut 9](#_Toc449543976)

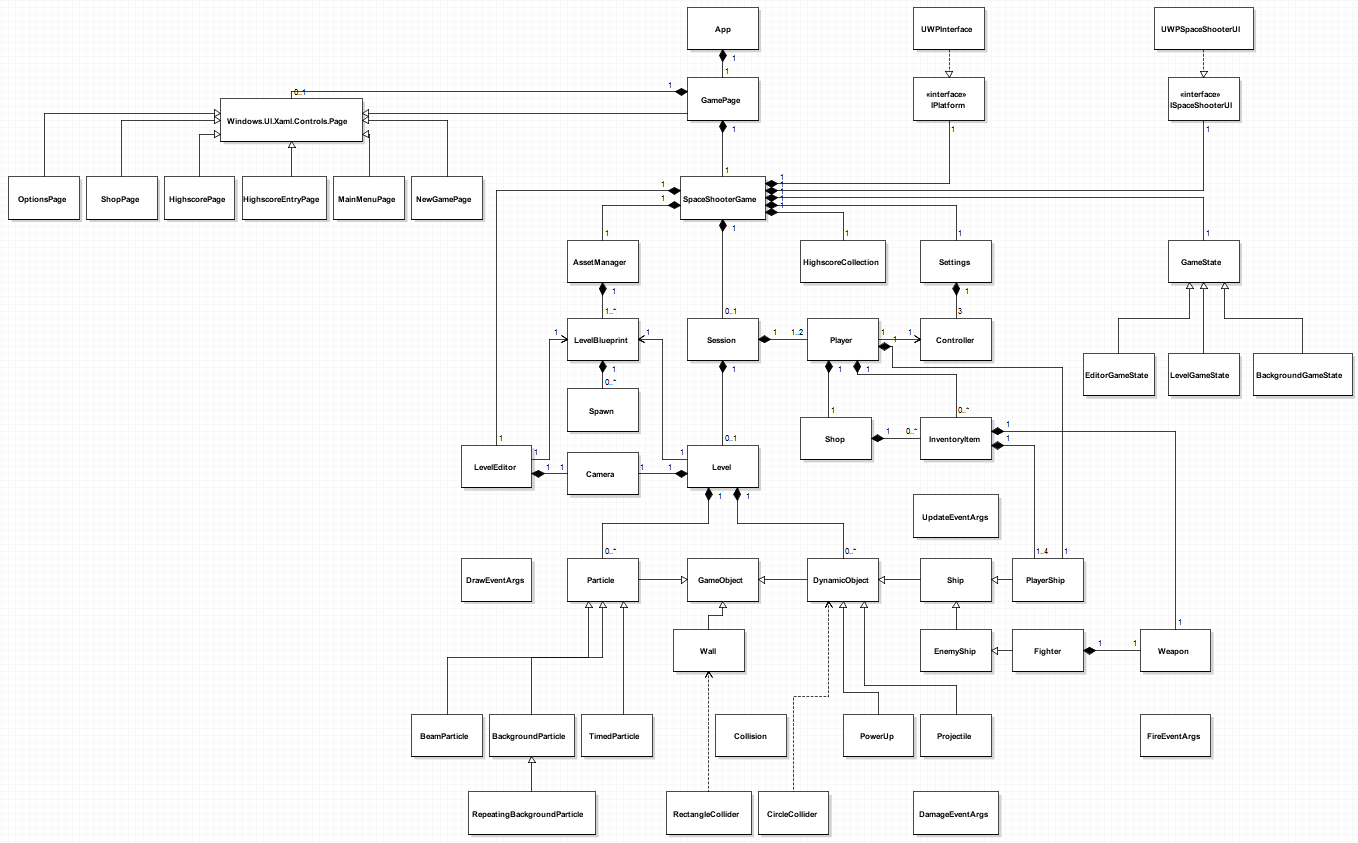
[6 Testaus 10](#_Toc449543977)

[7 Itsearviointi 10](#_Toc449543978)

[8 Kommentit kurssista 11](#_Toc449543979)

Harjoitustyön GitHub-osoite: <https://github.com/VilRan/SpaceShooter>

# Lopullinen rakennekaavio



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 voi sisältää myös jonkin muista pelin sivuista.

SpaceShooterGame

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

DynamicObject

Abstrakti-luokka, jossa määritellään muun muassa törmäyksen tarkistukset, törmäyksistä aiheutuvat vahingot, lähimmän pelaajan ja objektin etsintä jne.

PowerUp

Abstrakti-luokka, joka periytyy DynamicObject-luokasta. Määritellään Powerup:ien ominaisuuksia esimerkiksi kenen kanssa ne voivat törmätä.

Projectile

Abstrakti-luokka, joka periytyy DynamicObject-luokasta. Määritellään ammusten ominaisuuksia esimerkiksi seiniin törmäämisen seuraukset.

Ship

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

GameObject

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

Level

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

Camera

Käytetään ruudun vierittämiseen Levelin laidalta toiseen. Levelin etenemistahtia voidaan muokata muokkaamalla kameran etenemisnopeutta.

Spawn

Abstrakti-luokka, jossa määritellään muun muassa vihollisten esiintymistiheys kentässä.

LevelBlueprint

Ohjaa Level-luokan toimintaa tiedoilla, jotka eivät muutu pelin aikana.

Particle

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

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 tallennetaan, kun peli tallentuu.

Weapon

Abstrakti-luokka, joka määrittää pelaajan ja vihollisen aseille yhteisen käyttäytymisen.

AssetManager

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

Controller

Controller-luokka pitää sisällään muun muassa pelaajan liikkumiseen tarvittavat näppäimistötiedot.

GameState

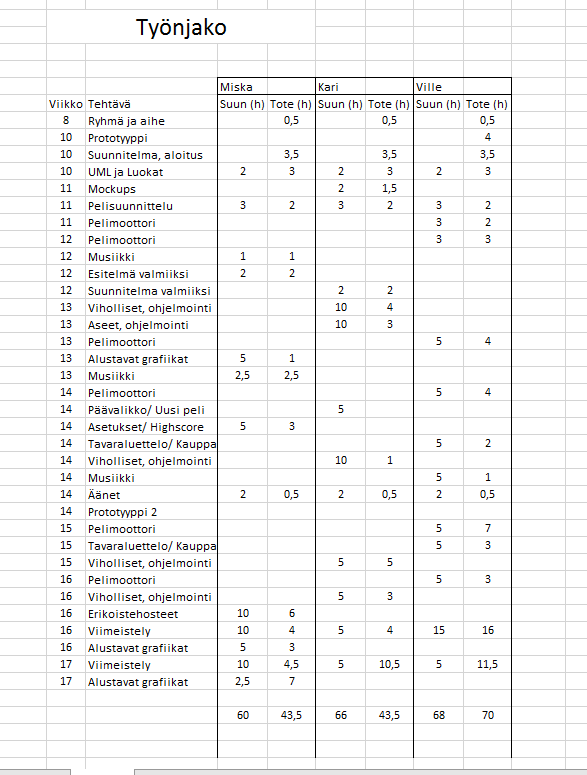
HighscoreCollection

Sisältää pelin pistetilastojen tallentamiseen tarvittavia tietoja.

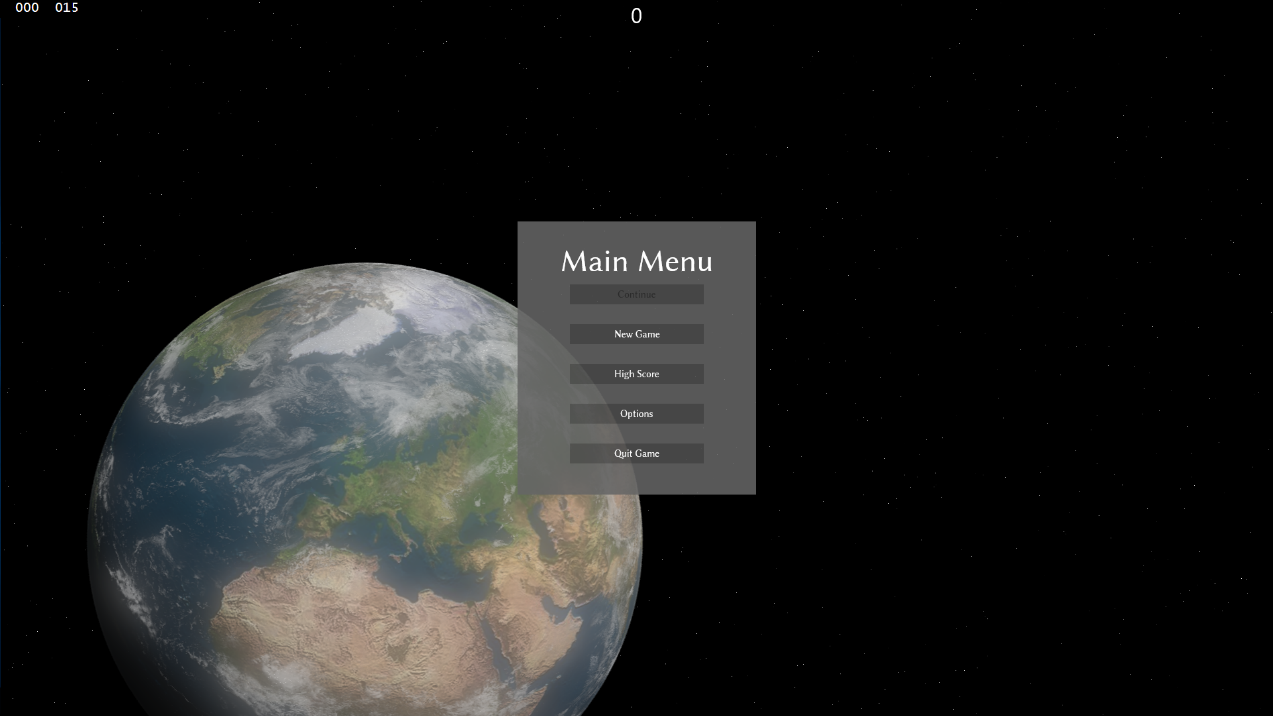
Settings

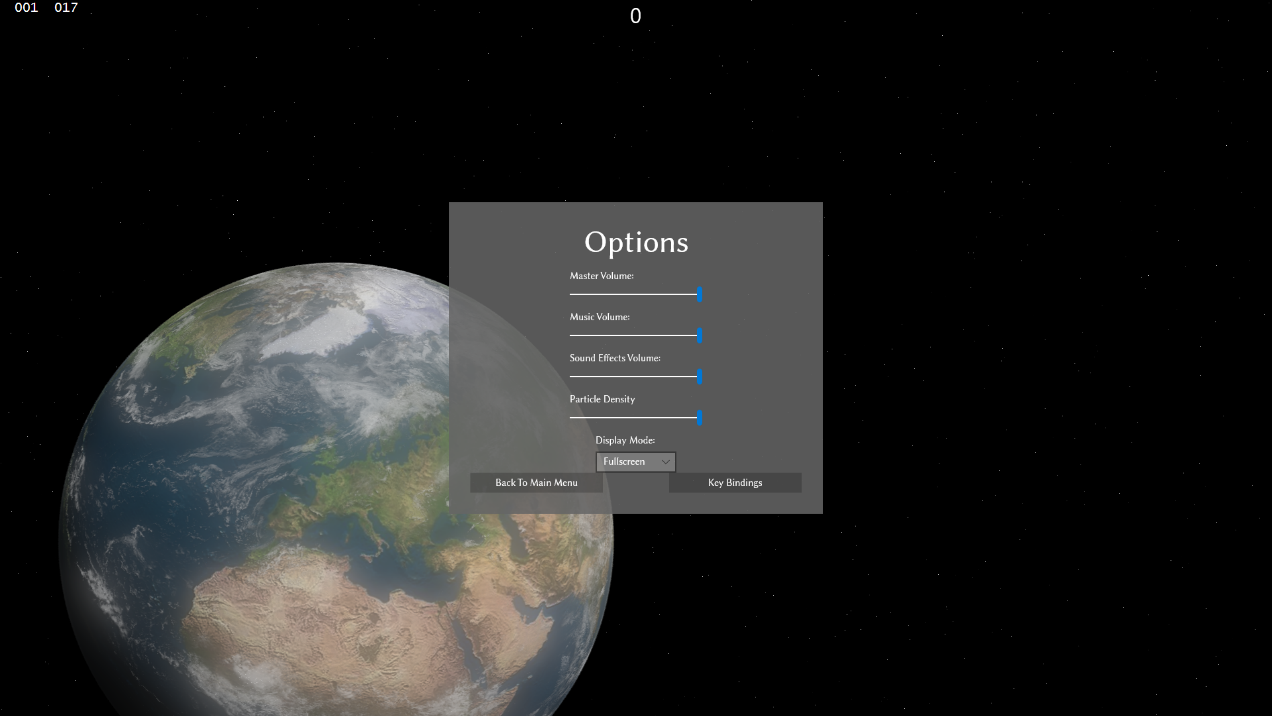
Settings-luokassa on pelin asetuksiin liittyviä tietoja, kuten äänien voimakkuuden säädöt, näppäimistön määritykset ja näytön koko.

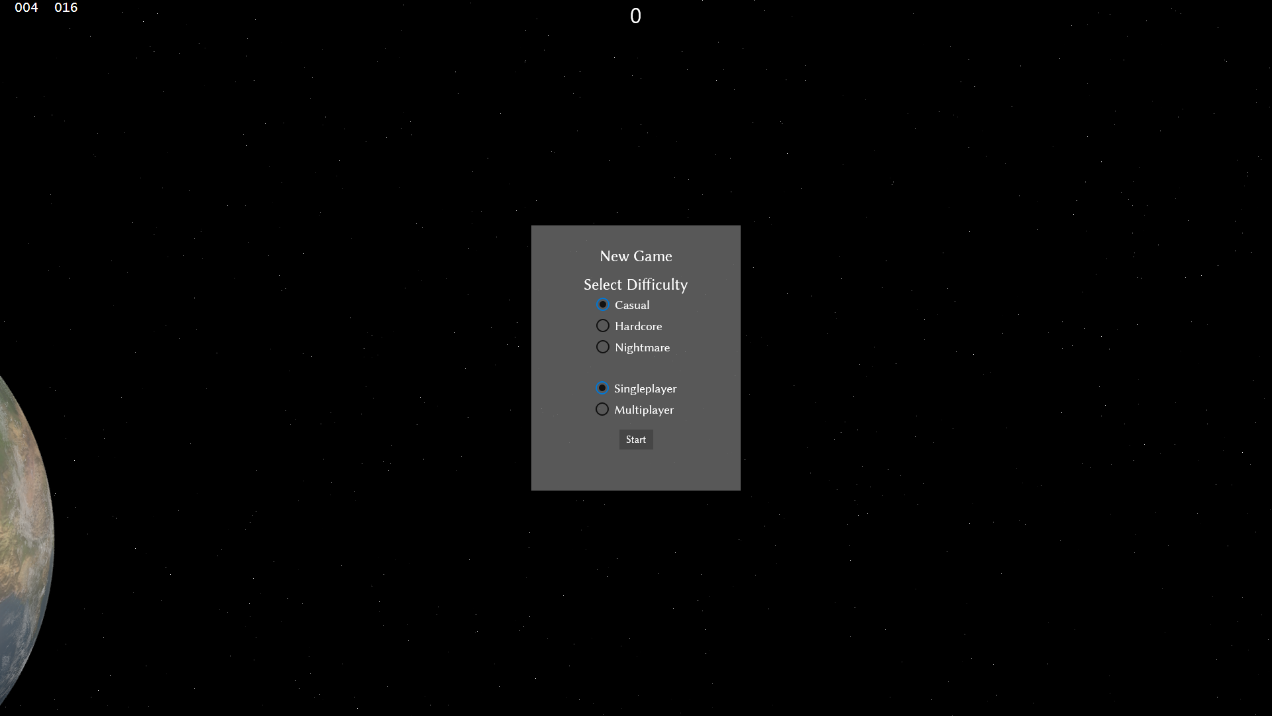
# Työaikaraportti

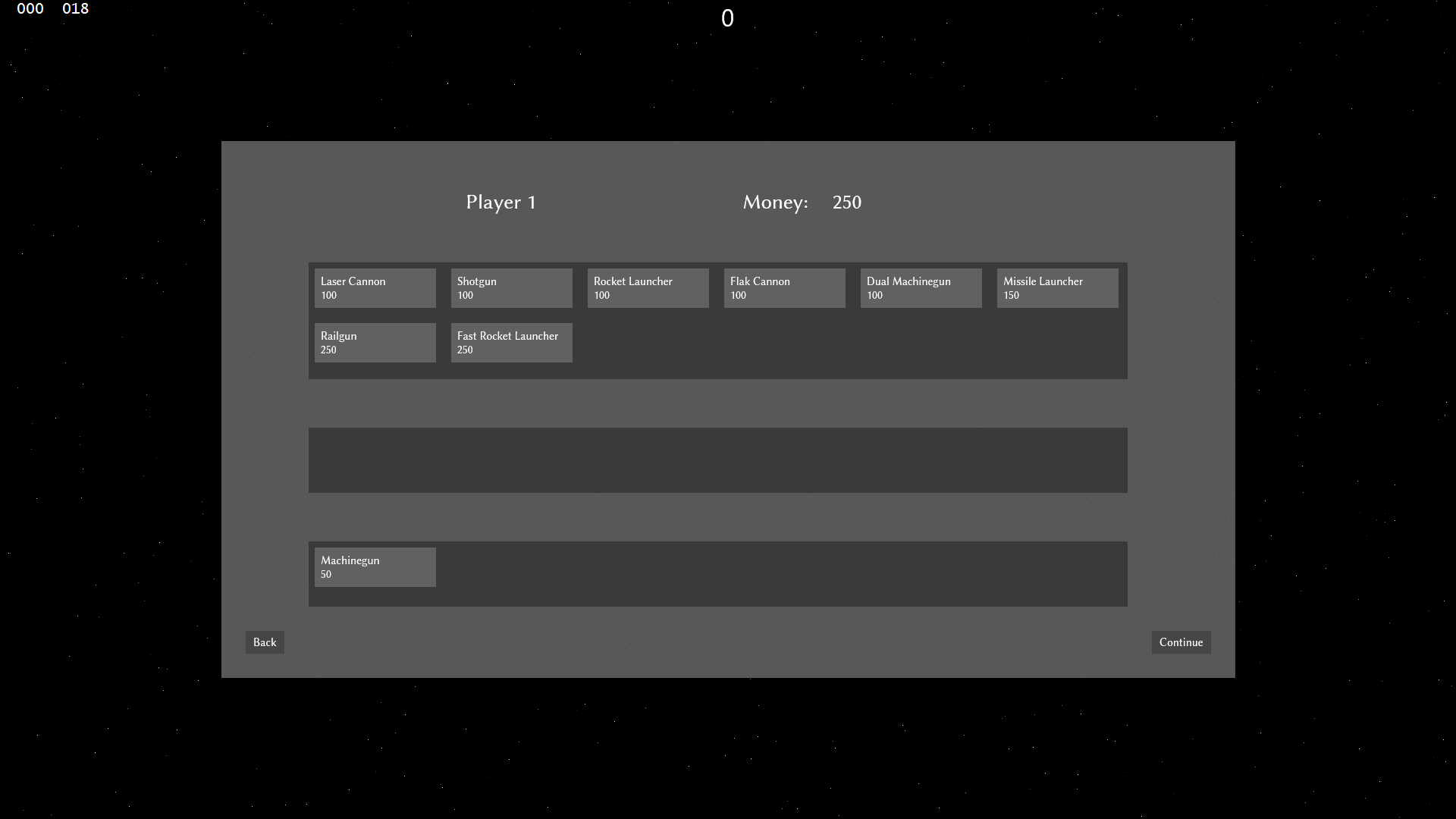


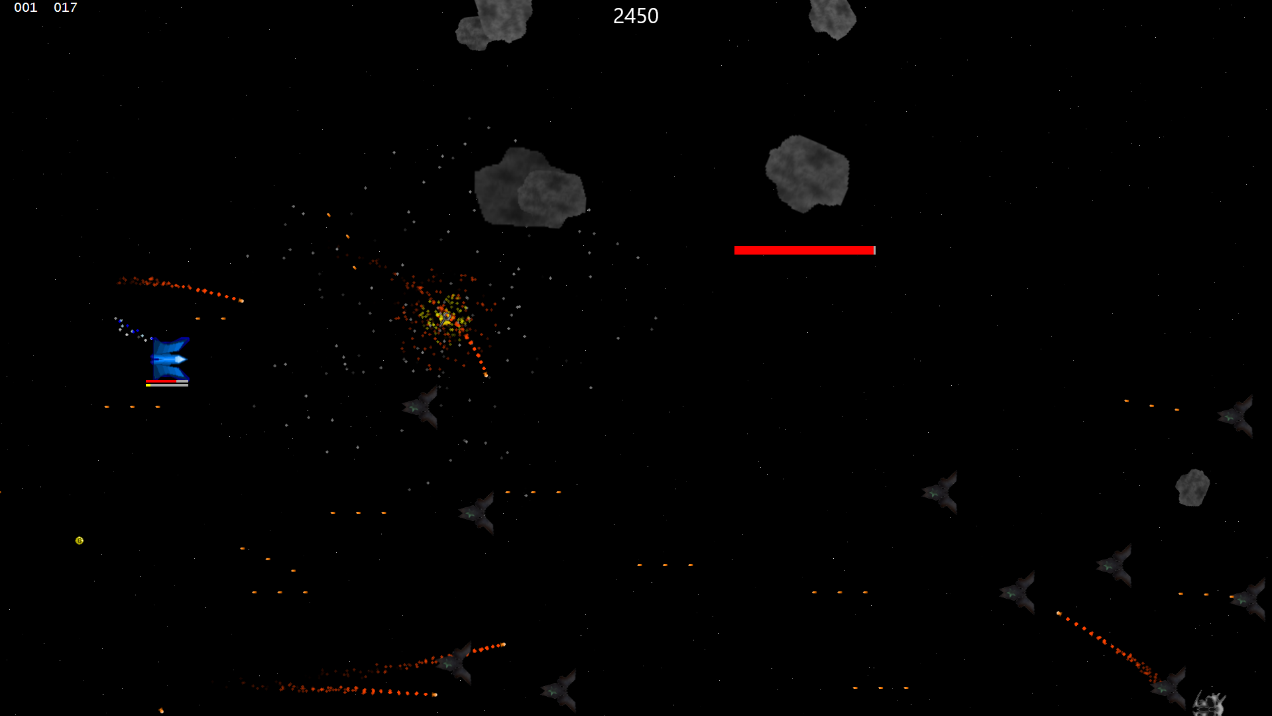
# Näyttökaappaukset

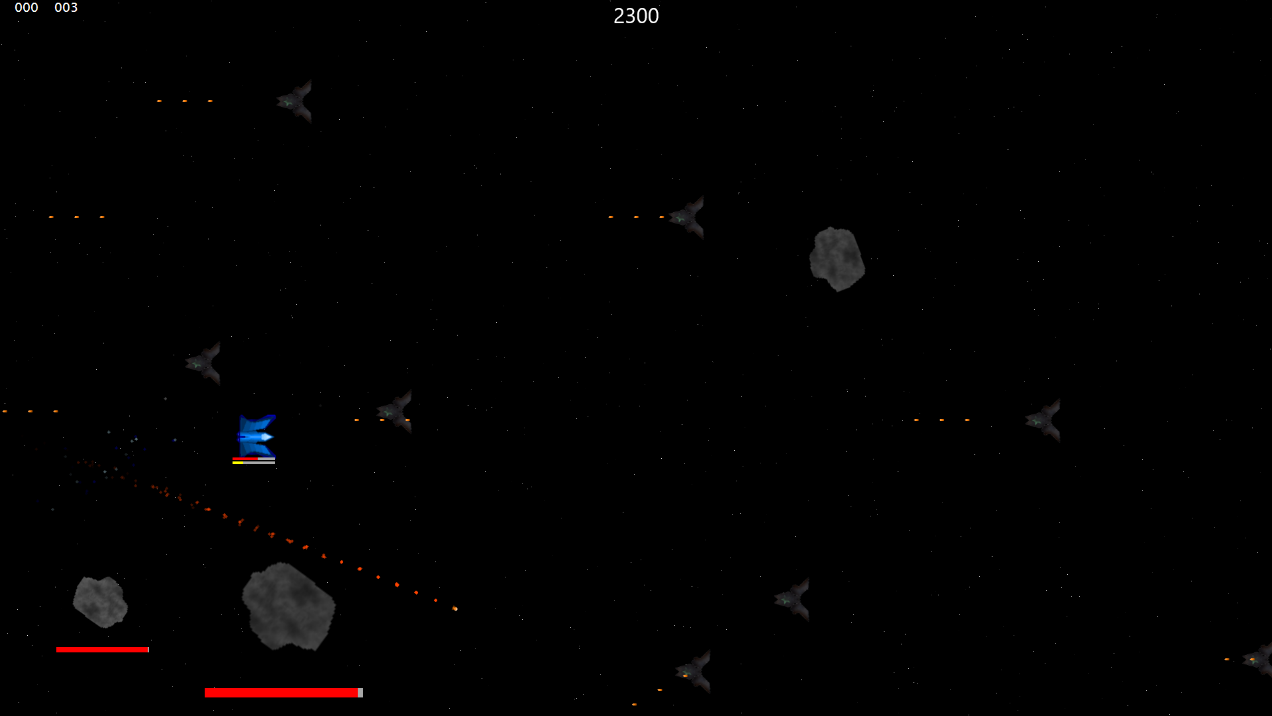


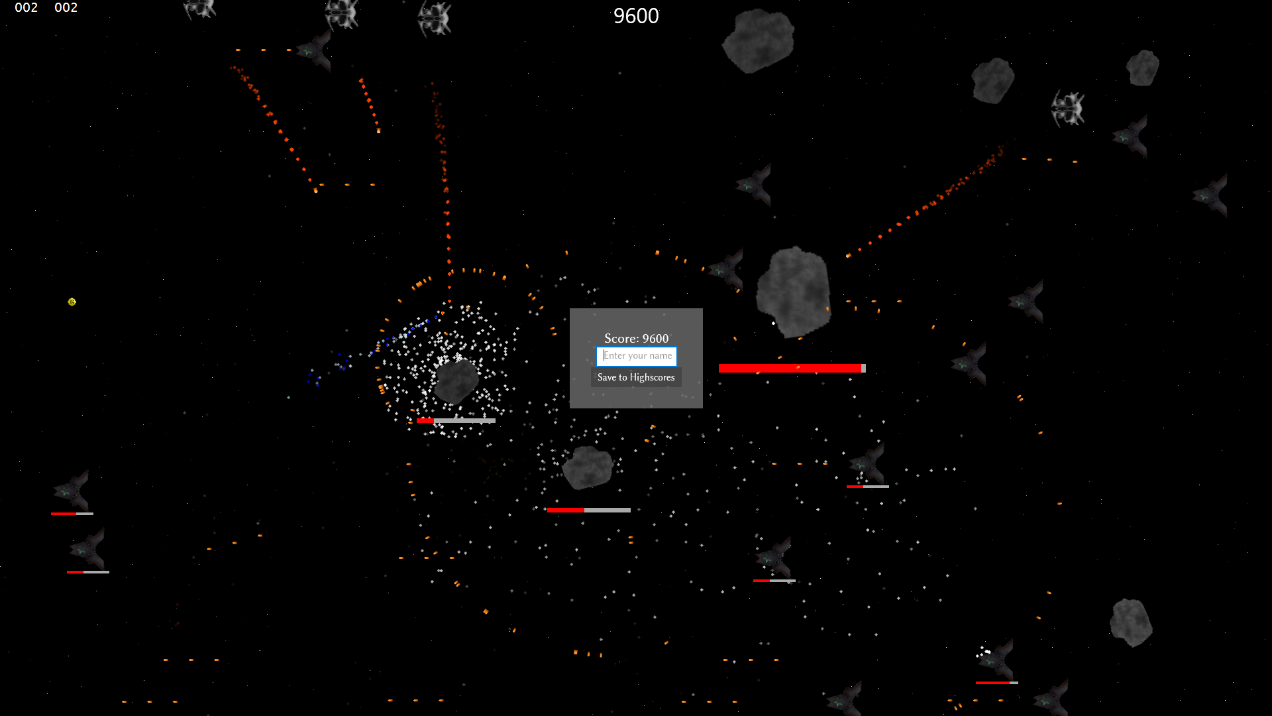


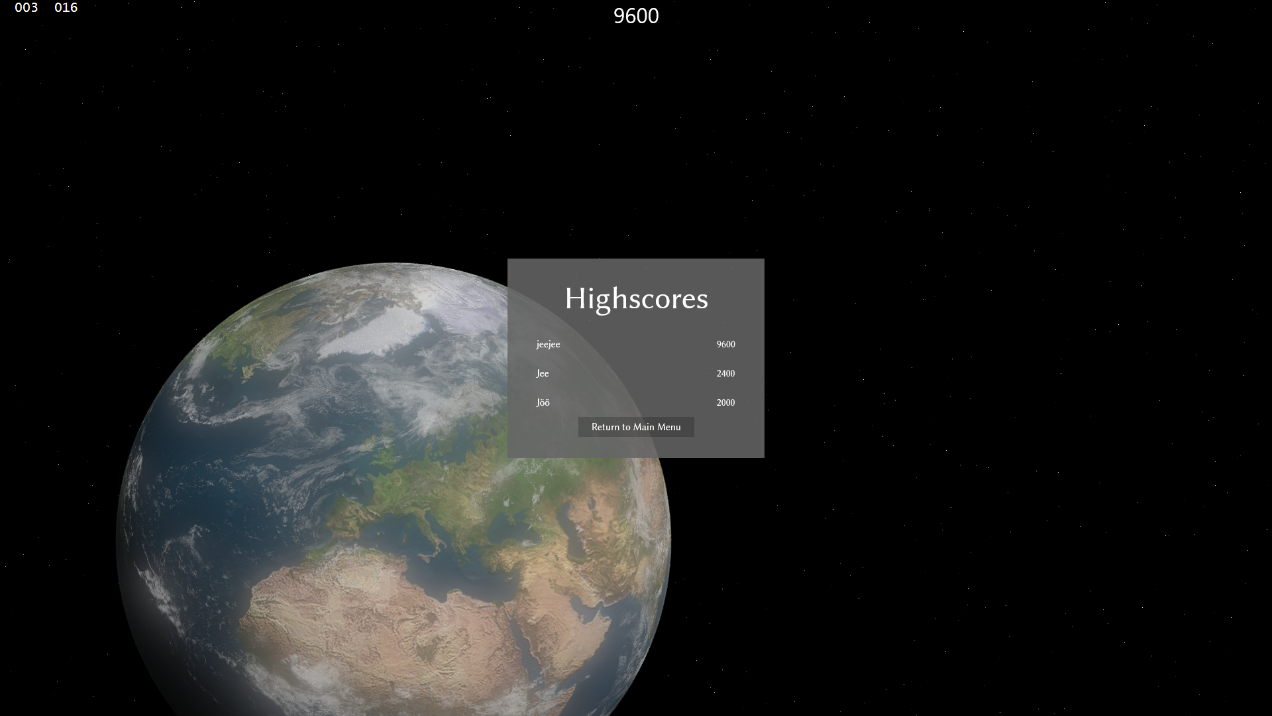












# Keskeisimmät ongelmat ja niiden ratkaisut

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.

# Testaus

Varsinaista testaussuunnitelmaa ei ollut, vaan koodia testattiin sitä mukaa kun sitä tuotettiin. Mikäli virheitä havaittiin, ne debugattiin ja korjattiin VisualStudion työkaluilla.

# 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.

# Kommentit kurssista