OULUN SEUDUN
AMMATTIKORKEAKOULU

Janne Köngäs

PELIKEHITYS UNITY3D-YMPÄRISTÖSSÄ

PELIKEHITYS UNITY3D-YMPÄRISTÖSSÄ

Janne Köngäs
Opinnäytetyö
Kevät 2013
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehitys

Tekijä: Janne Köngäs
Opinnäytetyön nimi: Pelikehitys Unity3D-ympäristössä
Työn ohjaaja: Tuomo Tikkanen
Työn valmistumislukukausi ja -vuosi: Kevät 2013 Sivumäärä: 41

Työn tavoitteena oli tutustua pelikehitykseen Unity3D-ympäristössä ja toteuttaa toimiva 3D-pelin pohja, jota voidaan halutessa jatkokehittää. Päätavoite oli oppia pelikehityksessä käytettävien työkalujen käyttö.

Pelin toteutuksessa käytettiin Unity3D-pelinkehitystyökalua. 3D-mallit ja animaatiot tehtiin Autodesk 3ds Max -mallinnusohjelmalla. 3D-mallien ja pelitekstuureiden tekoon käytettiin useita eri kuvankäsittelyohjelmia. Pelissä käytettiin sekä valmiita verkosta ladattuja 3D-malleja, joita muokattiin peliin ja Unityyn sopiviksi, että itse tehtyjä malleja.

Työn tavoitteisiin päästiin ja tuloksena saatiin aikaan toimiva ensimmäisen persoonan 3D-jousiammuntapelin pohja, jota on helppo jatkokehittää. Pohja sisältää pelin perusominaisuudet, maaston ja harjoitusradan. Työn aikana opittiin Unity3D-pelikehitystyökalun käyttö sekä 3D-mallinnus, pelihahmojen riggaus ja animointi Autodesk 3ds Max -mallinnusohjelmalla.

Asiasanat: Unity3D, peliohjelmointi, pelikehitys, 3D-mallinnus

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Software development

Author: Janne Köngäs
Title of thesis: Game development in Unity3D
Supervisor: Tuomo Tikkanen
Term and year when the thesis was submitted: Spring 2013 Pages: 41

The objective of this thesis was to explore the Unity3D game development environment. The aim on the outcome of the thesis was to create a functional base for a 3D game, which could be further developed, if so desired. The main objective of the project was to learn to use the tools used in game development.

The game was created with Unity3D game development tool. 3D models and animations were made in Autodesk 3ds Max modeling program. The textures used by the 3D models and the game itself were made using a variety of different image editing programs. The game uses 3D models downloaded from internet, which were modified to fit into the game and Unity, as well as self-made models.

The target goals of the thesis were achieved and resulted in a base of a functional first-person 3D archery game, which is easy to further develop. The base of the game includes the basic features used, the terrain of the game and a training field. During the making of the thesis, the use of the Unity3D game development tool, as well as 3D modeling, animating and rigging of the character in Autodesk 3ds Max modeling program were learned.

Keywords: Unity3D, game programming, game development, 3D-modeling

SISÄLLYS

TIIVISTELMÄ 3
ABSTRACT 4
SISÄLLYS 5
1 JOHDANTO 6
2 KÄYTETYT OHJELMISTOT 7
2.1 Unity3D 7
2.2 Autodesk 3ds Max 2013 9
2.3 Autodesk FBX Converter 2013 10
2.4 GIMP 11
2.5 Paint.NET 12
2.6 Image Magick 14
3 PROJEKTIN ETENEMINEN 15
3.1 Projektin aloitus 15
3.2 Maasto 15
3.2.1 Heightmap 16
3.2.2 Maaston korkeuserojen viimeistely 17
3.2.3 Vesitaso 19
3.2.4 Maaston tekstuurit 20
3.2.5 Puut, kasvit ja muut yksityiskohdat 21
3.3 Hahmo 23
3.3.1 Hahmomalli 23
3.3.2 3D-mallin riggaus 24
3.3.3 3D-mallin animointi 27
3.3.4 Hahmon ja animaatioiden tuonti Unityyn 28
3.3.5 Hahmon liikkuminen pelissä 30
3.4 Harjoitusrata 33
3.4.1 Maalitaulut 33
3.4.2 Alueskripti 35
3.5 Päävalikko 37
4 YHTEENVETO 39
LÄHTEET 40

1 JOHDANTO

Pelaaminen on ollut osa elämääni pienestä pitäen. Opiskellessani ohjelmistokehitystä lähes kaikki koulussa tekemäni projektit ovat olleet jonkinlaisia pelejä. Mietin viimeisiä harjoitusprojektejani tehdessä, että voisin tehdä omalla ajallani jonkinlaisen 3D-pelin oppiakseni pelikehitystä ja 3D-mallien luomista, mutta aika ei koskaan riittänyt sille. Opinnäytetyön aloittamisen ajankohdan tultua kohdalle minulla ei ollut mitään opinnäytetyöprojektia tähtäimessä. Projektin aihetta etsiessä aika alkoi käydä vähiin, joten ehdotin 3D-peliä opinnäytetyöksi. Opinnäytetyön aihe hyväksyttiin.

Alun perin projektin ideana oli tehdä jonkinlainen ensimmäisen persoonan eli pelihahmon silmin kuvattu jousiammuntapeli. Peli olisi voinut olla joko selviytymispelityyppinen tai verkossa pelattava taistelupeli jousilla. Projektista olisi kuitenkin tullut liian laaja, joten päädyin tekemään pelille vain pohjan, jota olisi helppo jatkokehittää. Pohja sisältää pelin perusominaisuudet, maaston sekä harjoitusradan, jossa voi harjoitella jousella ampumista.

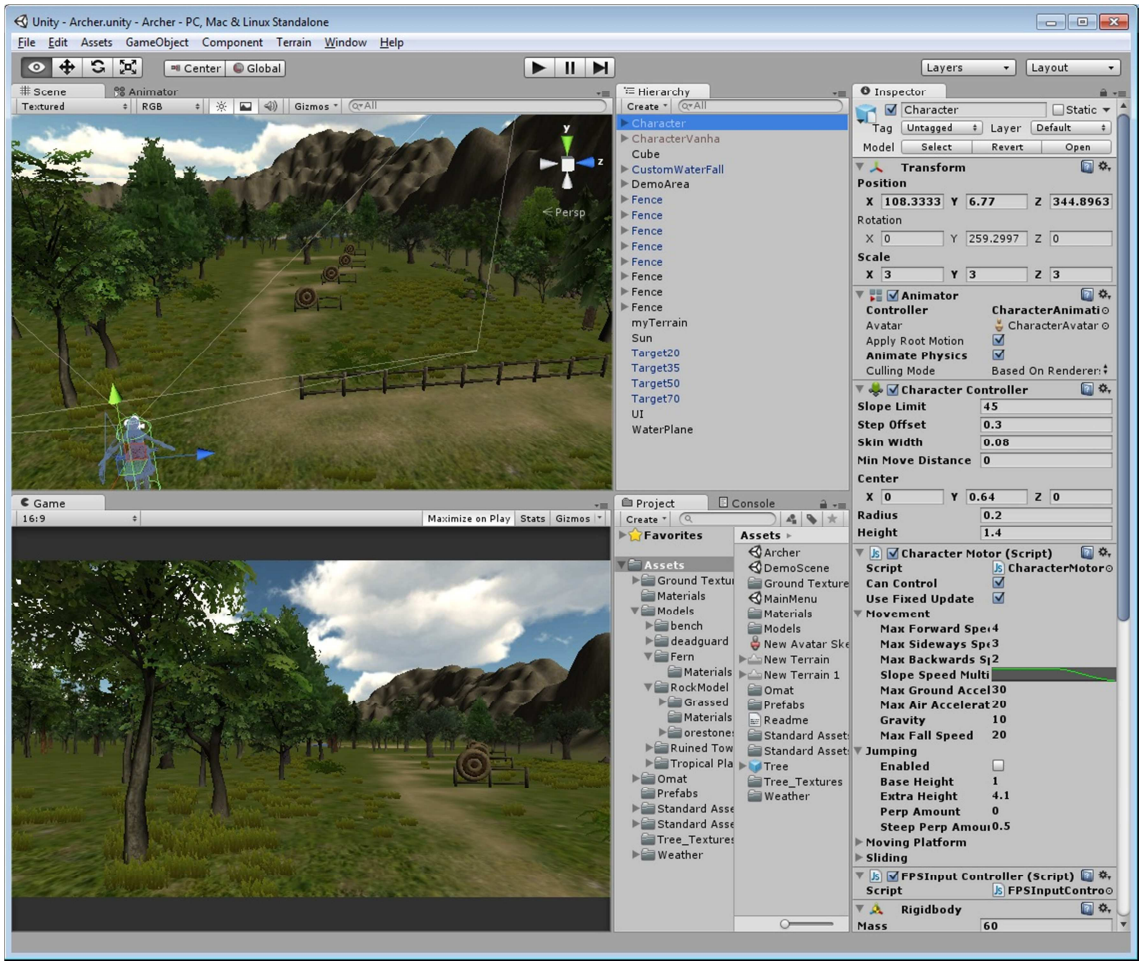
Työssä tutustuin peliohjelmointiin Unity3D-pelinkehitystyökalulla ja 3D-mallinnukseen Autodesk 3ds Max 2013 -mallinnustyökalulla. Käytin työssä myös useita eri 2D-kuvankäsittelyohjelmistoja.

2 KÄYTETYT OHJELMISTOT

2.1 Unity3D

Unity3D on alustariippumaton 3D-pelimoottori, jolla valmis projekti voidaan kääntää useille alustoille. Tämänhetkiset Unity3D:n tukemat alustat ovat iOS, Android, Windows, OS X, Linux, Web-selaimet, Flash, PlayStation 3, Xbox 360 ja Wii U. Unity on pääosittain tarkoitettu mobiili- ja selainpelien kehitykseen, mutta sitä käytetään paljon myös konsoli- ja PC-pelien kehityksessä. Unity-pelimoottori on kehitetty C/C++-kielellä. Pelilogiikka perustuu avoimen lähdekoodin .NET-alustaan nimeltä Mono, jonka tuetut ohjelmointikielet ovat C#, JavaScript ja Boo. Unitystä on olemassa sekä ilmainen että maksullinen Pro-versio. Unity Pron lisenssi maksaa 1140 € ja tarjoaa useita lisäominaisuuksia, jotka puuttuvat ilmaisesta versiosta. Myös iOS-, Android- ja Flash Player -tuet ovat erikseen ostettavia. Työssä käytettiin Unity3D-ilmaisversiota. (1.)

Unity3D on visuaalinen työkalu, jossa on useita erilaisia näkymiä pelin luomista varten. Osaa eri näkymistä ei juuri käytetä, mutta viisi eri näkymää on käytössä käytännössä lähes koko ajan. Projektiselaimessa näkyvät kaikki projektiin ladatut assetit eli resurssit, kuten 3D-mallit, tekstuurit ja koodit. Inspector-ikkunassa voi tarkkailla ja säätää scenessä olevien peliobjektien sekä projektiin lisättyjen assettien ominaisuuksia. Pelinäkymä toimii esikatseluikkunana ja näyttää, miten peli toimii laitteelle käännettynä (WYSIWYG – what you see is what you get). Scene-ikkuna on hiekkalaatikko eli alue, jonka kaikki objektit voivat olla yhteydessä toisten hiekkalaatikossa olevien objektien kanssa, mutta eivät kuitenkaan voi ottaa yhteyttä hiekkalaatikon ulkopuolisiin objekteihin. Tätä ikkunaa käytetään pelin luomiseen. Hierarkia-ikkuna näyttää kaikki scenen käyttämät peliobjektit ja niiden hierarkian. (Kuva 1.) (1.)



KUVA 1. Unityn käyttöliittymä

Unity tukee useita eri tiedostoformaatteja (taulukko 1) ja antaa täten mahdollisuuden käyttää useita tuotannon työkaluja. Eri tiedostoformaatit tukevat eri malliominaisuuksia. Tiedostojen lisääminen projektiin on helppoa ja vaivatonta, tiedosto vedetään Unityn projekti-ikkunaan tai tallennetaan projektikansioon ja Unity tuo sen heti käytettäväksi projektissa. Tiedostoja voi muokata milloin haluaa ja muutokset päivittyvät suoraan peliin. Vaikka työstettävä projekti olisi kuinka suuri tahansa ja projektiin liitettyjä tiedostoja olisi erittäin paljon, on niiden selaaminen ja etsiminen sekä esikatselu helppoa projektitiedostojen selaimella. Unity tarjoaa myös maksullisia ja ilmaisia assetteja Unity Asset Storessa. (1.)

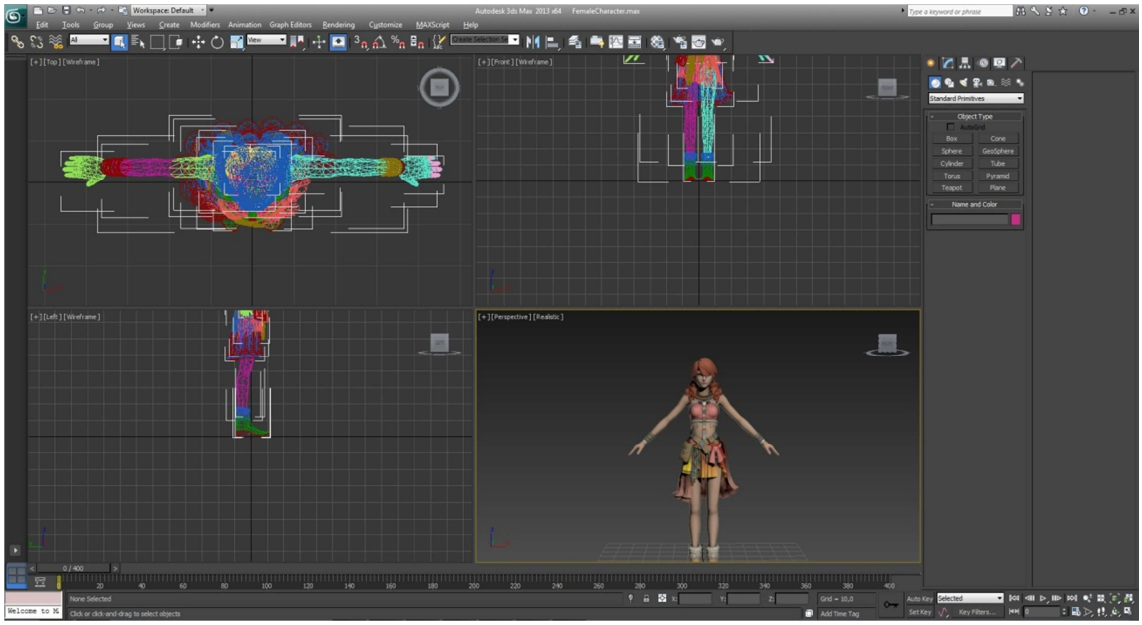
TAULUKKO 1. Tuetut 3D-paketit (1)

3D Package Support	Meshes	Textures	Anims	Bones
Maya .mb & .ma	x	x	x	x
3D Studio Max .max	x	x	x	x
Cheetah 3D .jas	x	x	x	x
Cinema 4D .c4d	x	x	x	x
Blender .blend	x	x	x	x
modo .lxo[2]	x	x	x	
Autodesk FBX	x	x	x	x
COLLADA	x	x	x	x
Carrara	x	x	x	x
Lightwave	x	x	x	x
XSI 5.x	x	x	x	x
SketchUp Pro	x	x		
Wings 3D	x	x		
3D Studio .3ds	x			
Wavefront .obj	x			
Drawing Interchange Files .dxf	x			

Yhdistettäessä Unityn scene-ikkunan toiminnot ja reaaliaikainen pelitila, Unity tarjoaa helposti käytettävän alustan, jolla voi muokata peliin liitettyjen komponenttien arvoja ja toimintoja sekä nähdä niiden vaikutuksen suoraan pelitilassa. Scene-ikkunassa voi helposti lisätä, muokata, siirrellä, skaalata ja kopioida kaikkia peliin liitettyjä objekteja. Unityssä voi myös käyttää valmiita muokattavia prefabejä eli objektipaketteja, joita voi sijoittaa suoraan peliin ja samalla poistaa tarpeen muokata kaikkia peliobjekteja erikseen. (1.)

2.2 Autodesk 3ds Max 2013

Autodesk 3ds Max -ohjelmisto tarjoaa tehokkaat ja helppokäyttöiset 3D-mallinnus-, animaatio- ja renderöintityökalut (kuva 2). Ohjelmisto on tarkoitettu pelikehittäjille, visuaalisten efektien ja liikkuvan grafiikan asiantuntijoille sekä muille media-alan ammattilaisille, arkkitehdeille, suunnittelijoille, insinööreille ja visualisoinnin ammattilaisille. 3ds Max on maksullinen ohjelma ja sen lisenssi maksaa 3900 €. 3ds Maxista on ilmestynyt 2014-versio, mutta käytin itse 3ds Max 2013 -testiversiota. (2.)

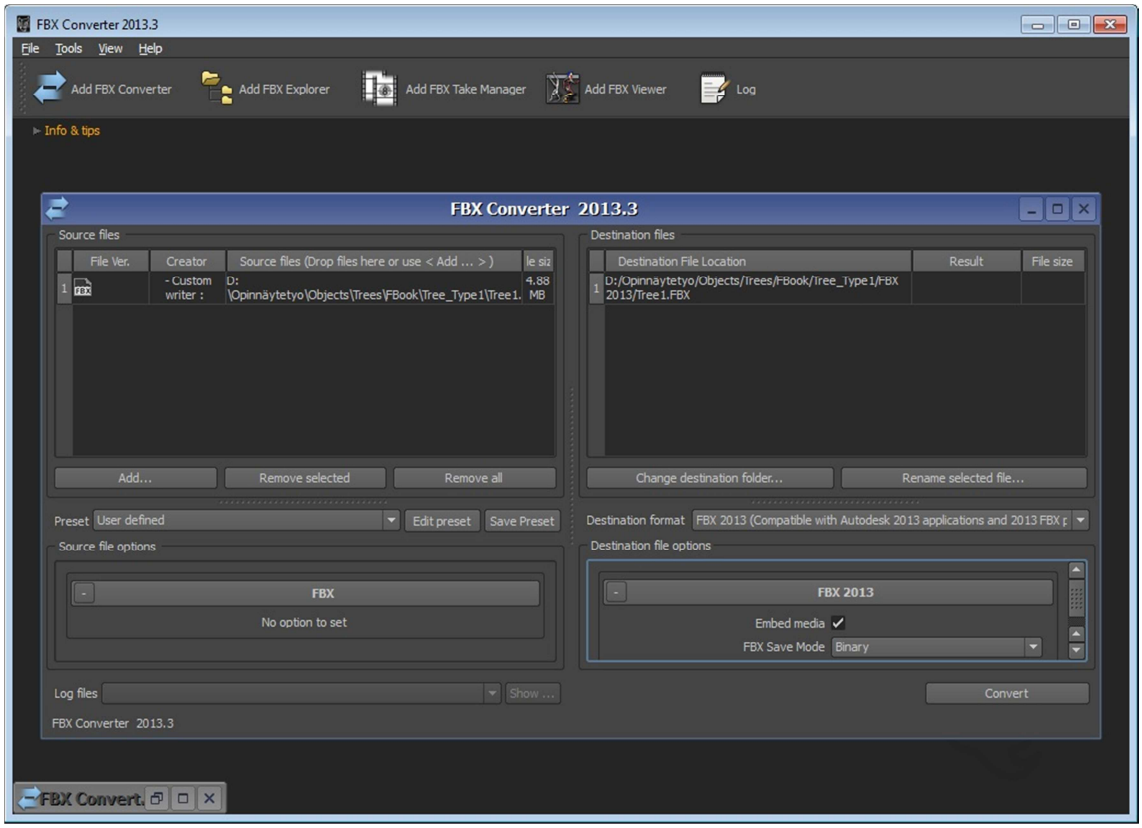


KUVA 2. 3ds Max 2013 -käyttöliittymä

2.3 Autodesk FBX Converter 2013

Autodesk FBX Converter muuttaa 3D-tiedoston tiedostoformaatin helposti toiseen. Ohjelmalla voi muuttaa OBJ-, DXF-, DAE- ja 3DS-tiedostoformaatit FBX-tiedostoformaattiin tai toisinpäin (kuva 3). FBX Viewerillä voi esikatsella FBX-tiedostoja ja niiden sisältämiä animaatioita. FBX Explorerilla voi tutkia FBX-tiedoston sisältöä. (3.)

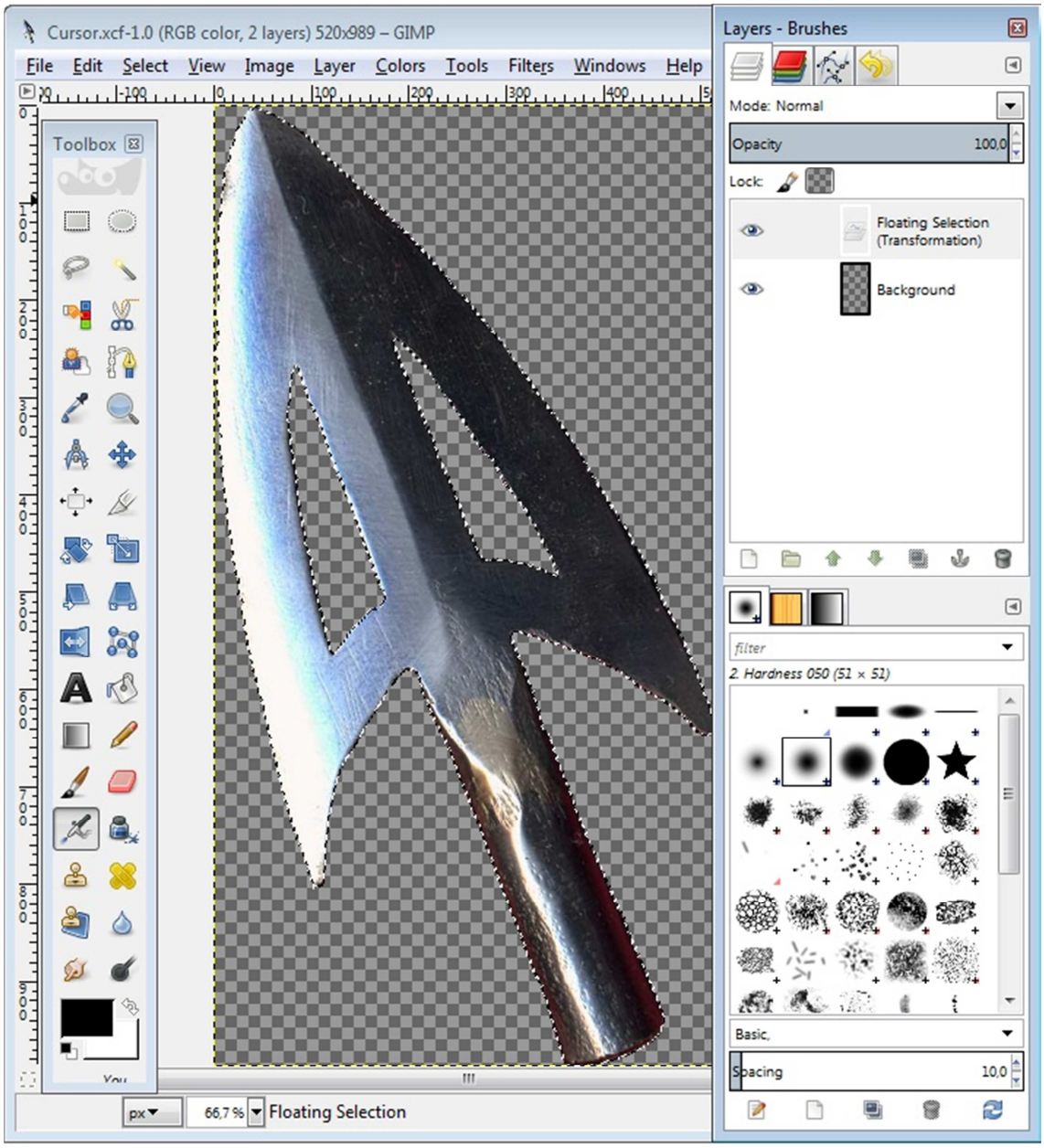
Projektissa ohjelmistoa käytettiin enimmäkseen tekstuureiden ja 3D-mallien yhdistämiseen yhteen tiedostoon, jotta se olisi helpompi viedä Unityyn. Ohjelma toimi myös hyvin animointien tarkastuksessa ennen Unityyn siirtoa. (3.)



KUVA 3. FBX Converter 2013

2.4 GIMP

Gimp (GNU Image Manipulation Program) (kuva 4) on avoimeen lähdekoodiin perustuva monipuolinen ja kevyt kuvankäsittelyohjelma. Ohjelmaa voi käyttää useilla eri käyttöjärjestelmillä. Gimpillä voi muokata ja yhdistellä eri kuvia ja tallentaa ne haluttuun tiedostoformaattiin. Gimpin vahvuutena on se, että sillä voi avata helposti useita eri kuvia yhtä aikaa omiin ikkunoihinsa ja siten kuvien yhdistäminen on erittäin kätevää. Gimp on kuitenkin tarkoitettu enemmän kuvien muokkaukseen kuin itse niiden luomiseen ja piirtäminen ohjelmalla onkin vaikeaa. (4.)

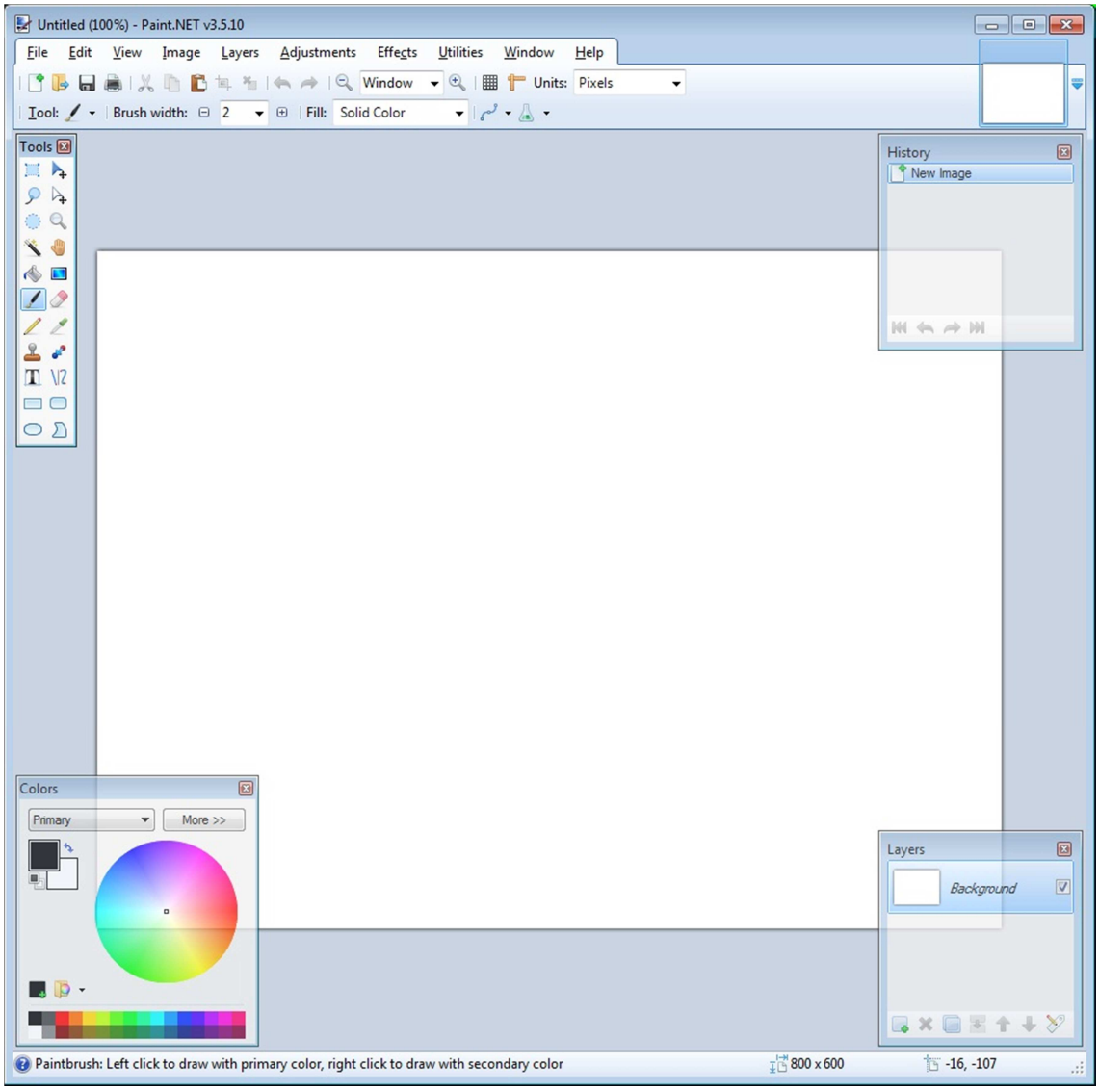


KUVA 4. GIMP käyttöliittymä

2.5 Paint.NET

Paint.NET on ilmainen Windows-alustalla toimiva kuvankäsittelyohjelma. Ohjelma sisältää intuitiivisen ja innovatiivisen käyttöliittymän (kuva 5), erilaisia taso-ominaisuuksia, loputon määrä kumoamisia ja laajan määrän tehokkaita kuvanmuokkaus- ja luontityökaluja. (5.)

Paint.NETin kehitys alkoi tietotekniikan kouluprojektista Microsoftille. Alun perin ohjelma oli tarkoitettu ilmaiseksi korvikkeeksi Microsoft Paint -ohjelmalle, joka tulee kaikkien Windows asennusten mukana. Ohjelma on kehittynyt vahvaksi mutta helpoksi kuvankäsittelytyökaluksi ja sitä on verrattu muihin kuvankäsittelyohjelmiin, kuten Adobe Photoshopiin, Corel Paint Shop Prohon, ja GIMPiin. (5.)



KUVA 5. Paint.NETin käyttöliittymä

2.6 Image Magick

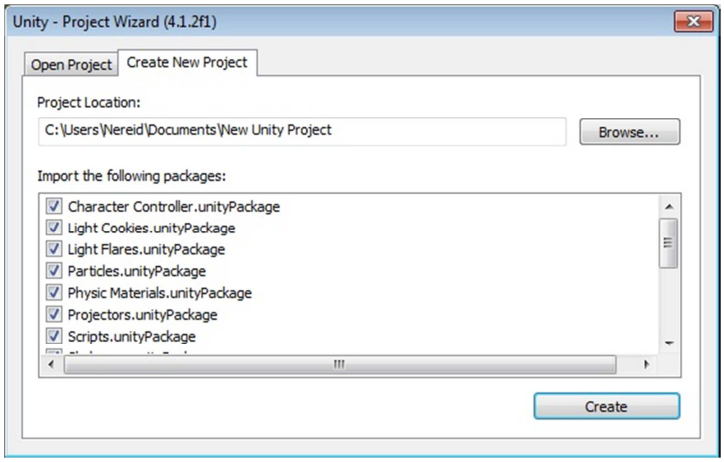
Image Magick on ohjelmisto, jolla voi luoda, muokata, sommitella ja konvertoida bittikarttakuvia. Ohjelma tukee yli sataa eri kuvatiedostoformaattia. Ohjelman ominaisuuksia käytetään yleisesti komentoriviltä tai erillisistä visuaalisista työkaluista. Ohjelma on ilmainen ja sen lähdekoodi on saatavissa ja muokattavissa omien tarpeiden mukaan. (6.)

3 PROJEKTIN ETENEMINEN

3.1 Projektin aloitus

Projektia aloittaessani en ollut koskaan käyttänyt Unityä tai tehnyt minkäänlaisia 3D-mallinnuksia. Aloitin projektin selaamalla verkosta erilaisia ohjevideoita ja materiaaleja Unityn käytöstä. Aluksi tarkastelin ensimmäisen persoonan hahmon ominaisuuksia seuraten opetusvideota YouTubesta (7). Samalla opin, miten Unityssä tehdään uusi projekti ja miten peliin lisätään uusia peliobjekteja. Katsoin myös maastonmuokkauksen ohjevideon (8).

Opittuani Unityn perusominaisuudet aloitin projektin pelin tekemisen luomalla uuden Unity-projektin. Projektia luodessa voidaan valita valmiit asset-paketit, jotka liittyvät projektiin (kuva 6) ja ovat sitten suoraan käytettävissä. Projektin luonti vie paljon prosessointi aikaa, sillä Unity lataa kaikki valitut asset-paketit projektiin.

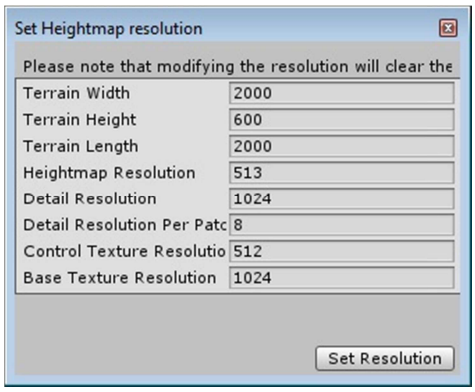


KUVA 6. Unity projektin luonti

3.2 Maasto

Kun Unity-projekti on luotu, pelissä ei ole vielä minkäänlaista alkutasoa, jonka päälle peliä voitaisiin alkaa rakentaa. Aluksi peliin pitää siis luoda pelitaso tai maasto eli terrain. Maaston luonti tapahtuu helposti painamalla Unityn yläpalkista Terrain-alasvetovalikkoa ja sieltä painamalla Create Terrain. Unity luo perus-

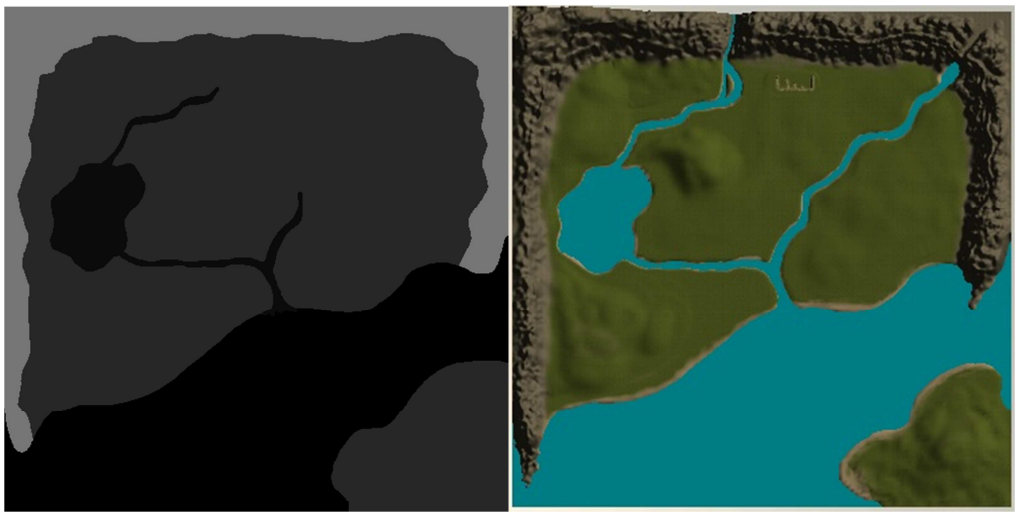
asetuksien mukaisen sileän laatan maastoksi. Maaston asetuksia (kuva 7) kannattaa kuitenkin aluksi muuttaa pelin vaatimuksien mukaiseksi painamalla Set Resolutionia Terrain -alasvetovalikossa.



KUVA 7. Maaston (Terrain) asetukset

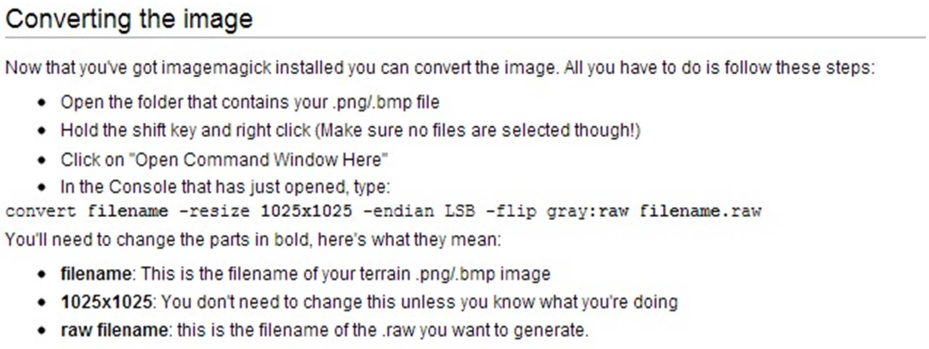
3.2.1 Heightmap

Pelin maaston korkeuserojen tekeminen on mahdollista kokonaan Unityn tarjoamien maastoneditointityökalujen avulla, mutta tähän on helpompikin tapa eli heightmap. Heightmap on kuvatiedosto (kuva 8), joka sisältää maaston korkeuserot mustavalkokuvana. Kuvassa musta väri tarkoittaa kaikista matalinta kohtaa ja valkoinen korkeinta. Kaikki harmaan sävyt mustan ja valkoisen välillä ovat eri korkeuksia. Maaston korkeuseron maksimimäärän mustan ja valkoisen välillä voidaan määrittää maaston asetuksista.



KUVA 8. Heightmap ja pelin lopullinen maasto

Heightmapin voi tehdä millä tahansa kuvankäsittelyohjelmalla, mutta Unity vaatii heightmap tiedoston tiedostoformaatiksi .raw:n. Tähän formaattiin tallennus ei onnistukaan juuri miltään kuvankäsittelyohjelmalta ja siitä aiheutui ongelmia. Tein heightmapin Paint.NETillä ja tallensin sen .png-formaattiin. Tämän jälkeen saadakseni kuvan .raw-formaattiin, minun täytyi käyttää erillistä komentorivipohjasta konvertointiohjelmaa nimeltä Image Magick. Image Magickin käyttöön jouduin etsimään ohjeen verkosta (kuva 9) (9).



Converting the image

Now that you've got imagemagick installed you can convert the image. All you have to do is follow these steps:

- Open the folder that contains your .png/.bmp file
- Hold the shift key and right click (Make sure no files are selected though!)
- Click on "Open Command Window Here"
- In the Console that has just opened, type:

```
convert filename -resize 1025x1025 -endian LSB -flip gray:raw filename.raw
```

You'll need to change the parts in bold, here's what they mean:

- **filename**: This is the filename of your terrain .png/.bmp image
- **1025x1025**: You don't need to change this unless you know what you're doing
- **raw filename**: this is the filename of the .raw you want to generate.

KUVA 9. Image Magick -ohjelman .raw:n konvertointiohje (9)

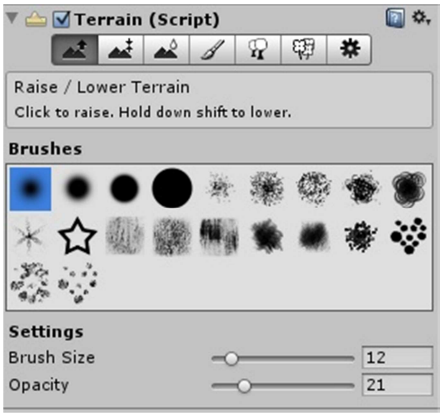
Kun heightmap on .raw formaatissa, sen voi tuoda Unityyn painamalla Import Heightmap – Raw painiketta Terrain -alasvetovalikossa ja valitsemalla luodun tiedoston. Unity lataa heightmapin ja muokkaa maaston sen mukaiseksi.

3.2.2 Maaston korkeuserojen viimeistely

Riippuen siitä kuinka tarkkaa heightmapia käyttää, maaston korkeuserojen muokkaamiseen voi käyttää myös Unityn työkaluja (kuva 11). Jos heightmap on tarkka, näitä korkeuserotyökaluja ei tarvitse käyttää ollenkaan. Tekemäni heightmap ei kuitenkaan ollut tarkka, vaan vain raakavedos siitä minkälainen maasto pelille olisi tarkoitus tulla, joten käytin paljon Unityn tarjoamia maaston-muokkaustyökaluja viimeistelläkseni maaston haluamani näköiseksi (kuva 8).

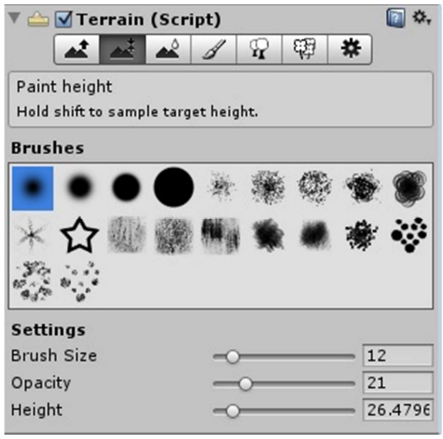
KUVA 10. Unityn terraintyökalut

Maaston nosto- ja laskutyökalulla (kuva 11) voi joko nostaa tai pitämällä vaihtonäppäintä pohjassa laskea maaston korkeutta. Työkalun siveltimen voi valita Unityn sivellinlistasta. Siveltimen kokoa ja vaikutusvoimakkuutta voidaan säätää tarpeen mukaan työkalun asetuksista. Tätä työkalua käytin eniten kaikista maaston korkeustyökaluista. Työkalulla saa aikaan jyrkkiä korkeuseroja ja jos työkalun tehoa laskee, sillä voi myös tehdä pienempiä korkeuseroja kuten mäkiä.



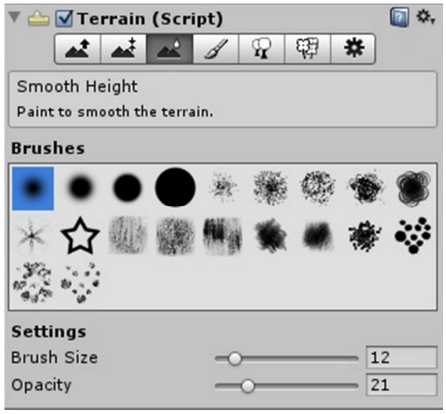
KUVA 11. Unityn maaston nosto- ja laskutyökalu

Maastokorkeuden tasaustyökalu (kuva 12) toimii muuten samalla tavalla kuin nosto- ja laskutyökalu, mutta työkaluun valitaan tietty korkeus, johon työkalu sitten korottaa tai laskee maaston. Korkeuden voi valita helposti painamalla pelialueella jotain kohtaa vaihtonäppäintä pohjassa pitäen tai valitsemalla korkeuden itse työkalun asetuksista. Tätä työkalua käytin eniten tilanteissa, joissa halusin saada jonkin laajemman alueen tietylle tasolle, ennen kuin tein pienialaisempia korkeuseroja nosto- ja laskutyökalulla.



KUVA 12. Unityn maastokorkeuden tasaustyökalu

Maastokorkeuserojen pehmennystyökalua (kuva 13) käytetään korkeuserojen viimeistelyyn. Työkalu pyöristää ja loiventaa korkeuserot siveltimen alueella. Työkalun vahvuutta ja siveltimen kokoa voidaan muuttaa muiden työkalujen tapaan työkalun asetuksista.

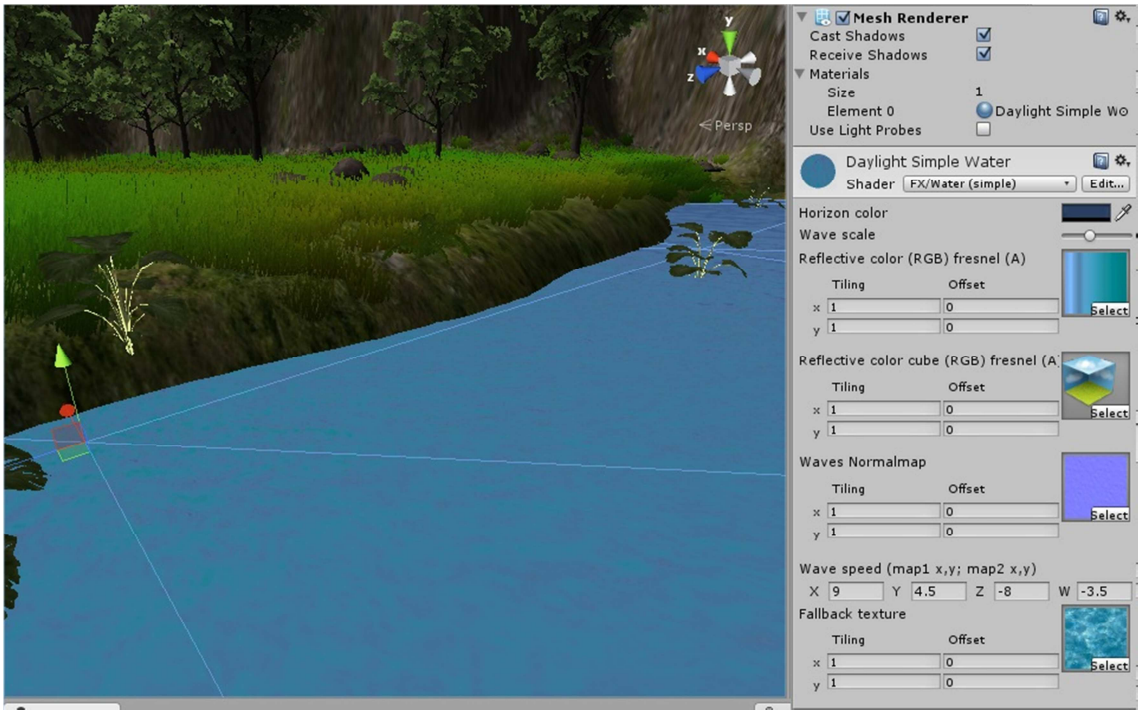


KUVA 13. Unityn maastokorkeuserojen pehmennystyökalu

3.2.3 Vesitaso

Mikäli luotuun maastoon halutaan lisätä vesielementti, on peliin lisättävä uusi taso, vesitaso. Vesitaso lisätään lisäämällä sceneen uusi peliobjekti Plane. Plane on taso, jota voi käyttää muuhunkin tarkoitukseen kuin vain vesitasoksi. Vesitasona taso laitetaan valitulle korkeudelle maastokorkeuteen nähden, jolloin vesi peittää osan matalammasta maastosta. Tasoon lisätään Mesh Render,

johon liitetään haluttu vesielementti (kuva 14). Jos objektissa ei ole Mesh Renderiä, niin objekti ei näy pelaajalle, mutta on silti vielä pelissä. Unity tarjoaa muutamia valmiita vesielementtejä, joita voi käyttää omissa projekteissa. Ilmaiset Unityn tarjoamat vesielementit eivät ole läpinäkyviä, kuten maksullisessa Pro versiossa. Vesielementtejä voi myös luoda itse tai käyttää toisten tekemiä elementtejä.

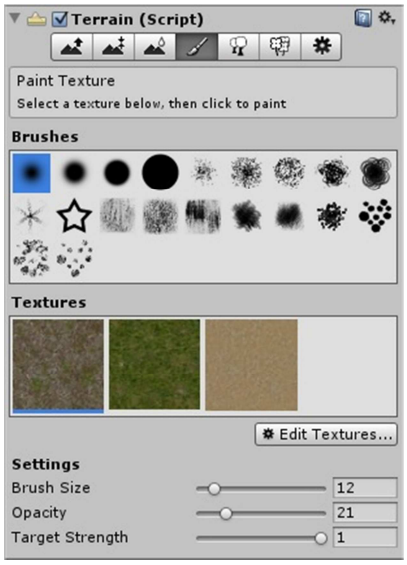


KUVA 14. Unityn vesitaso, jossa käytössä ilmainen Daylight Simple Water-elementti

3.2.4 Maaston tekstuurit

Projektiin lisätty maasto on harmaa, ennen kuin siihen maalataan maaston tekstuurit. Maalaustyökaluun voi lisätä useita eri tekstuureita (kuva 15). Projektissa käytin kolmea eri tekstuuria: kiveä, nurmikkoa ja hiekkaa. Tekstuureita maalatessa voi valita muiden maastotyökalujen tapaan siveltimen asetuksia. Aluksi maalasin koko maaston ruohotekstuurilla, joten käytin mahdollisimman suurta sivellintä. Tämän jälkeen maalasin vuoret ja vuorirajat kivitekstuurilla ja vesistöt hiekkatekstuurilla. Tekstuurinmaalaustyökalun Opacity-arvoa muokkaamalla pystyi maalaamaan myös useita tekstuureja päällekkäin, jolloin sai aikaan luo-

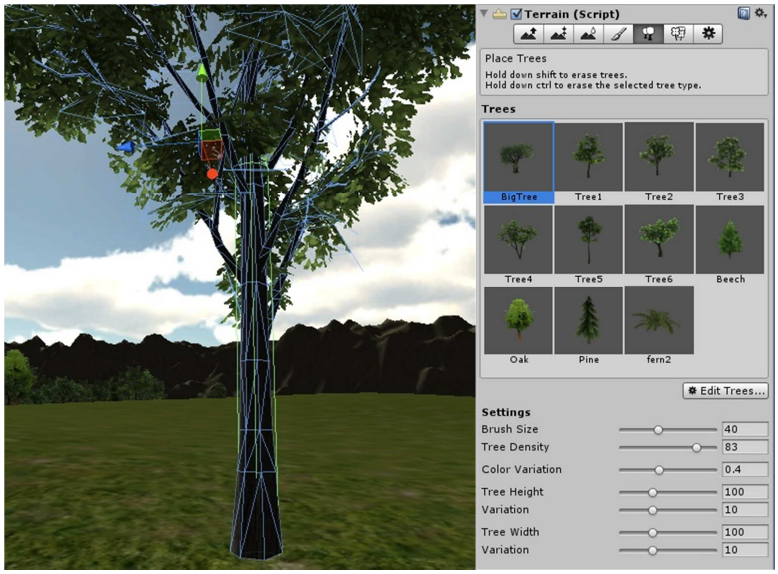
nollisemman näköisen maaston. Kaikki tekstuurista toiseen siirtymät viimeisteltiin sekoittamalla tekstuureita raja-alueella.



KUVA 15. Unity maaston tekstuureiden maalaustyökalu

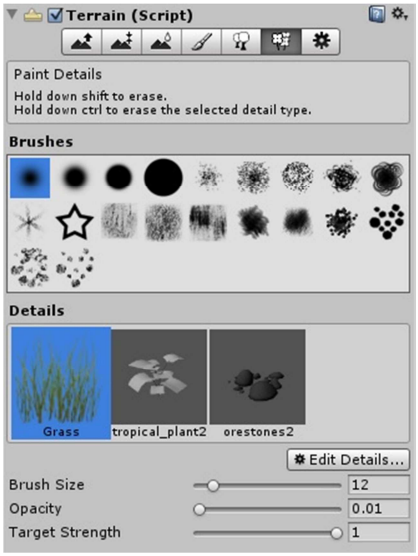
3.2.5 Puut, kasvit ja muut yksityiskohdat

Unityssä on kaksi vaihtoehtoa puiden luomiseen. Puita voi luoda sisäänrakennetulla Tree Creatorilla tai niitä voi tuoda projektiin valmiina 3D-malleina. Käytin projektissa itse valmiita 3D-malleja, joita sai ladattua verkosta ilmaiseksi (10). Verkosta ladatut mallit eivät kuitenkaan toimineet suoraan Unityn puidenlisäystyökalun kanssa (kuva 16), vaan niiden koordinaatisto piti kääntää vastaamaan Unityn koordinaatistoa. Unityn käyttämässä koordinaatistossa Y-akseli on ylöspäin, kun taas 3D-mallinnusohjelmissa Z-akseli on ylöspäin. Ennen puiden lisäämistä maastoon puidenlisäystyökalulla, puista piti tehdä prefabejä eli peliobjekteja, joihin on lisätty valmiiksi tiettyjä ominaisuuksia. Kaikkien puiden prefabeihin liitettiin capsule collider, jotta puiden läpi ei voi kävellä tai ampua. Puista tehdyt prefabit lisättiin puidenlisäystyökaluun, jonka jälkeen niitä pystyi sijoittelemaan maastoon.



KUVA 16. Puu-prefabin capsule collider ja puidenlisäystyökalu

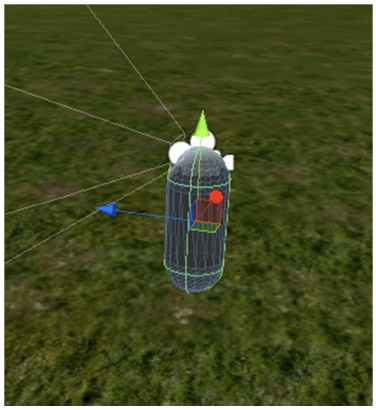
Puiden lisäksi maastoon lisättiin yksityiskohtia, kuten korkeampaa nurmikkoa ja kiviä. Näitä yksityiskohtia lisätään maaston yksityiskohtien maalaustyökalulla (kuva 17). Yksityiskohtien maalaustyökalu toimii samaan tapaan kuin puiden-lisäystyökalu, mutta lisättäessä yksityiskohtia on valittava, onko yksityiskohta nurmikkoa vai jotain muuta. Nurmikkona toimiva yksityiskohta liikkuu tuulessa, joten jos tämän asetuksen unohtaa vaihtaa kivielementeistä, on lopputulos mielenkiintoinen.



KUVA 17. Maaston yksityiskohtien lisäystyökalu

3.3 Hahmo

Unityssä tulee mukana valmis ensimmäisen (pelaajan silmin nähty) ja kolmannen persoonan (hahmon takaa kuvattu) hahmo-ohjain. Ensimmäisen persoonan ohjain on vain kapseli, johon on liitetty kamera (kuva 18). Valmiin ohjaimen hahmo liikkuu pelissä ilman minkäänlaisen koodin lisäämistä. Ohjaimessa on mukana Character Motor -skripti, johon määritetään hahmon nopeudet ja muut ominaisuudet, FPSImput Controller -skripti, joka lähettää ohjauskomennot Character Motorille, sekä Mouse Look -skripti, joka ohjaa hahmon kääntymistä hiirellä. En koskenut itse näiden skriptien koodeihin ollenkaan, mutta tein oman Mouse Look -skriptin hahmon muiden liikkeiden ohjaukseen.



KUVA 18. First Person Controller

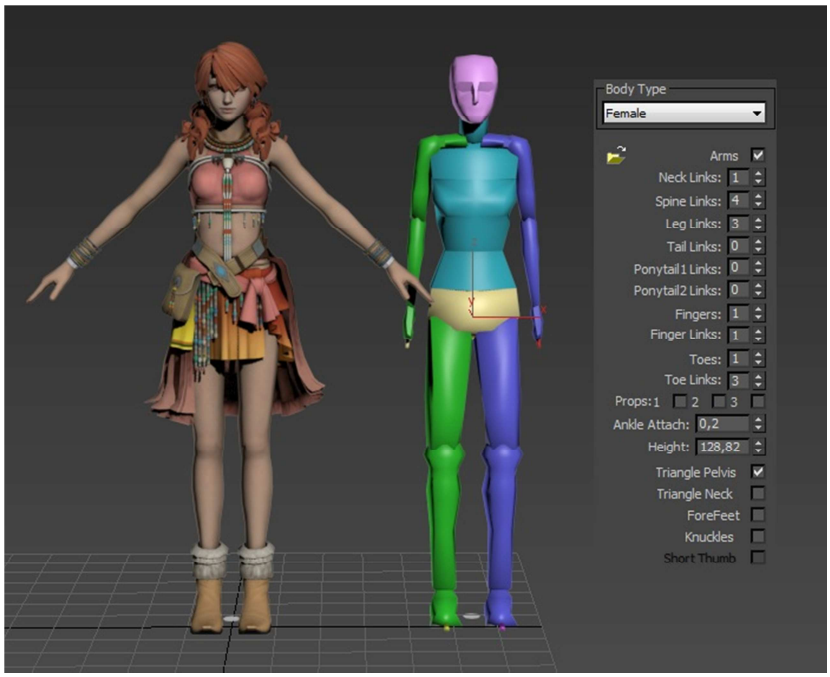
3.3.1 Hahmomalli

Vaikka peli onkin kuvattu ensimmäisen persoonan kameralla, pelin hahmo oli silti parempi muuttaa ihmisen kaltaiseksi hahmoksi, kuin jättää se alkuperäiseksi kapseliksi. Aloitin ensin hahmon luomisen tyhjästä 3D-mallinnusohjelmalla. Huomasin kuitenkin nopeasti ettei pelkkä hahmon luonti riitä siihen, että hahmon saa toimimaan oikein pelissä. Päädyin etsimään verkosta valmista hahmoa peliä varten. Löydettyäni sopivan hahmon, kokeilin sen sopivuutta pelissä korvaamalla valmiin hahmo-ohjaimen kapselin uudella 3D-hahmomallilla. Etsin myös verkosta 3D-mallin jouselle ja nuolille.

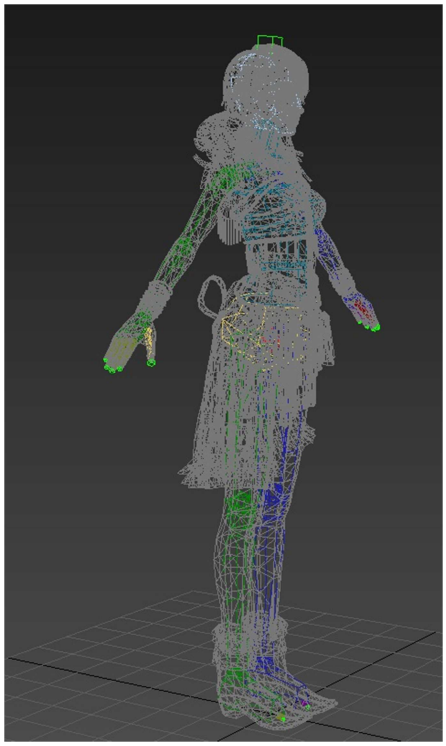
3.3.2 3D-mallin riggaus

Jotta 3D-mallilla saataisiin aikaan ihmismäisiä liikkeitä, 3D-malli pitää rigata. Riggaus tarkoittaa käytännössä luiden lisäämistä hahmomallin sisälle ja niiden liittämistä ympäröivään hahmoon siten, että kun luuta liikuttaa, hahmon raajat liikkuvat luonnonmukaisesti.

Riggauksessa päädyin käyttämään 3ds Maxin biped-mallia (kuva 19). Biped-malli on vaihtoehtoinen tapa luoda hahmon luusto kuin lisäämällä itse yksi luu kerrallaan. Biped-malli on hahmon näköinen malli, jonka nivelten määriä voi muokata mallin luontivaiheessa. Kun biped-mallia aletaan sijoittaa hahmomallin sisään, kannattaa biped-malli piirtää suunnilleen siten, että hahmon ja biped-mallin lantio ovat samalla korkeudella. Tämän jälkeen biped-malli siirretään hahmon sisään (kuva 20). Biped-malli on saatava mahdollisimman samankokoiseksi joka kohdasta kuin itse hahmo. Aloitin biped-mallin muokkauksen vasemmalta puolelta hahmoa muuttamalla sen raajojen kokoa ja asentoa vastaamaan hahmon vastaavia osia. Kun vasen puoli oli saatu vastaamaan hahmon mallia, kopioin sen peilikuvana työkalun avulla myös oikealle puolelle.



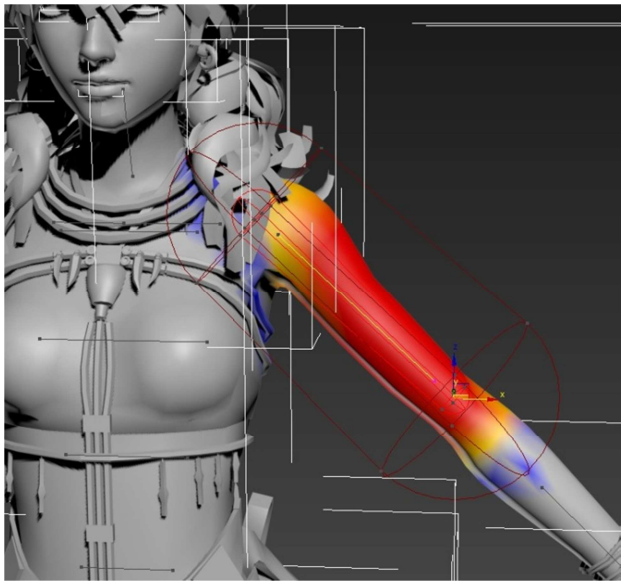
KUVA 19. Bibed-malli ja sen asetukset sekä hahmo malli



KUVA 20. Biped-malli hahmomallin sisällä.

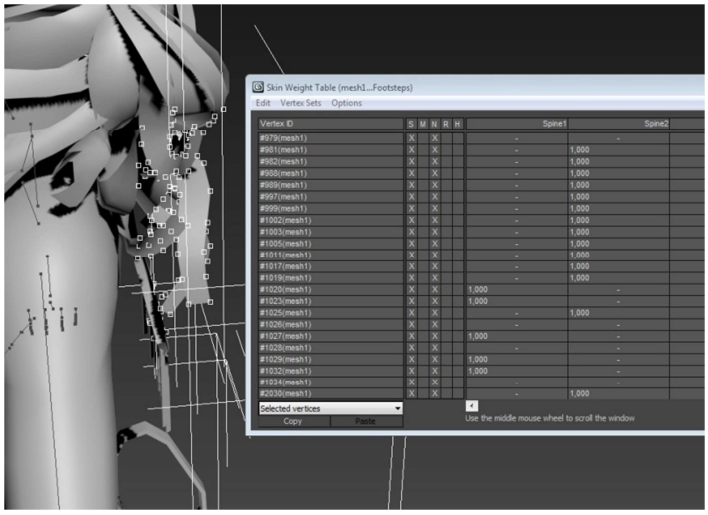
Saatuani biped-mallin vastaamaan hahmomallin asentoa (kuva 20) testasin sen toimintaa ja liikkeitä animoimalla 3D-mallinnusohjelmassa biped-mallille ihmismäisiä liikkeitä ja tutkimalla, liikkuivatko kaikki hahmon osat, kuten niiden kuuluisi. Huomasin nopeasti, että jotkin kohdat hahmosta liikkuivat väärin, sillä osa luiden liikkeistä vaikutti myös paikkoihin, joihin niiden ei olisi pitänyt vaikuttaa. Tämä ei kuitenkaan tullut yllätyksenä, koska en ollut vielä muokannut luiden vaikutusalueita ja niiden vahvuuksia envelopeilla.

Jokaista luuta kohden on olemassa yksi envelope. Envelope koostuu kahdesta sisäkkäisestä kapselista (kuva 21). Sisäisen kapselin sisältö liikkuu aina, kun sitä vastaava luu liikkuu. Ulkoinen kapseli rajaa vaikutusalueen ulkoreunan, mutta tämä alue voi olla myös toisen luun vaikutusalueella. Kapseleiden kokoa muuttamalla hahmon pinnan väri työkalussa muuttuu sen mukaiseksi, kuinka vahva luun vaikutus on tällä alueella. Punainen väri tarkoittaa, että se alue liikkuu vain ja ainoastaan kyseisen luun vaikutuksella, kun taas värin muuttuessa keltaiseksi ja siitä siniseksi alueisiin vaikuttavat myös muut luut.



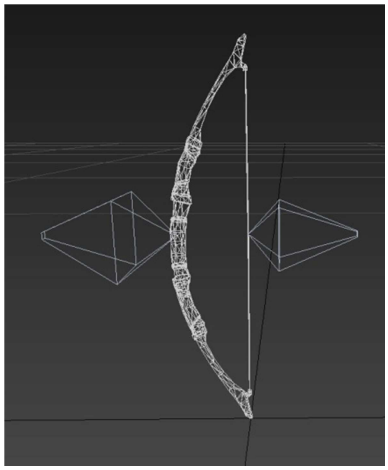
KUVA 21. Olkavarren luun vaikutus hahmomalliin

Envelopejen säädöllä sai suurimman osan luiden vaikutuksista toimimaan oikein. Kuitenkin jotkin kohdat mallista, kuten hiukset ja hahmon hame vaativat täsmällisempää hienosäätöä. Tätä varten käyttöön piti ottaa yksittäiset veticesit eli pisteet mallin pinnalla. Jokaisen mallin pisteen painoarvoa pystyi muokkaamaan Skin Weight Tablesta (kuva 22). Taulukkoon pystyi syöttämään pisteelle painoarvon jokaista luuta kohti. Näitä painoarvoja säätämällä pystyi viimeistelemään kaikki hahmon luiden vaikutusalueet.



KUVA 22. Vertices ja Skin Weight Table

Hahmolla olevan jousen malli piti myös rigata, jotta jousta pystyy venyttämään pelissä. Tätä varten jouselle täytyi tehdä kaksi luuta, toinen jousen varrelle ja toinen jänteelle (kuva 23). Näin jousta pystyy venyttämään Unityssä siirtämällä jousen jänteen luuta poispäin jousen varren luusta.



KUVA 23. Jousen riggaus

3.3.3 3D-mallin animointi

Pelin hahmon animointi tehtiin 3ds Maxissa. Animointi onnistui helposti piilottamalla itse hahmon mallin ja näyttämällä vain biped-mallin. 3ds Maxin alalaidassa olevasta aikajanasta valittiin haluttu kohta ja aktivoitiin Auto Key -ominaisuus päälle (kuva 24). Kun Auto Key -ominaisuus on päällä ja hahmon biped-mallin asentoa muokkaa, ohjelma lisää automaattisesti valitulle aikajanan kohdalle avaimen, joka sisältää senhetkisen hahmon asennon.

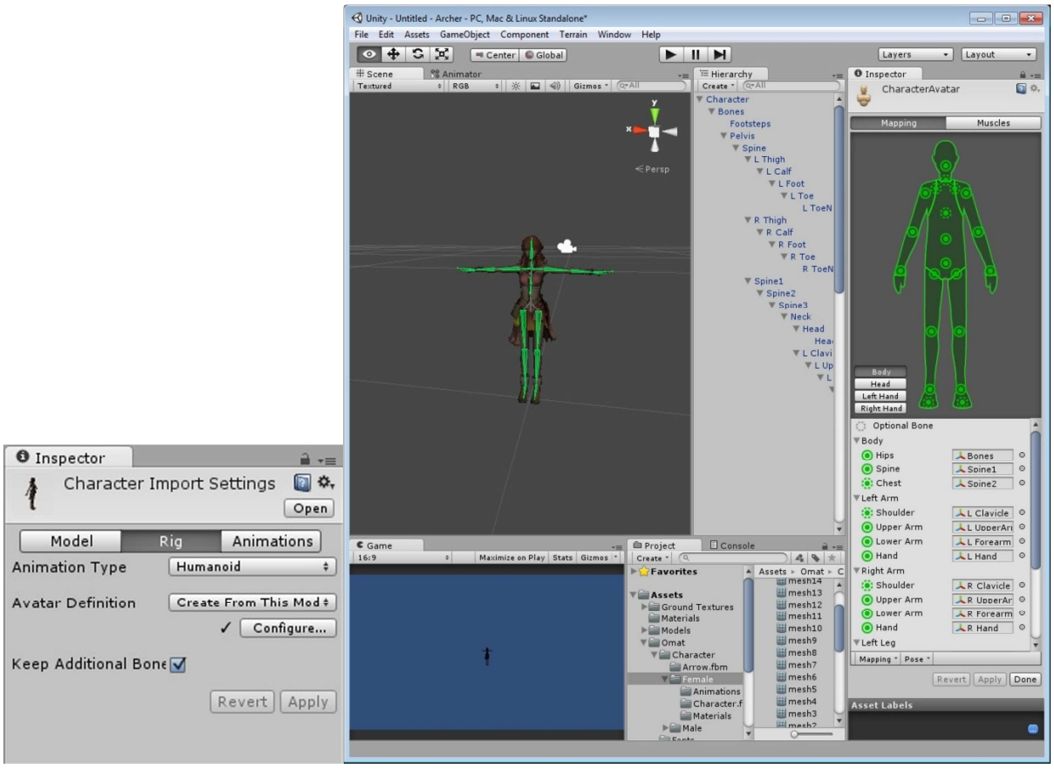
Ohjelma luo liikkeen eri avainten välisten asentojen välillä automaattisesti. Esimerkiksi tähtäysanimaatiossa aikajanan kohdassa 60 hahmo on nostanut kädet ylös ja valmistautuu jännittämään jousta. Tähän tilanteeseen siirtävä animaatio on automaattisesti ohjelman luoma. Hahmon kävelyanimaation jalkojen liikkeiden oikeudellisuuden saavuttamiseksi kävelin itse edestakaisin ja kopioin liikkeen mallille. Hahmon seistessä paikallaan hahmo huojuu hieman, kuten aito ihminenkin. Jousen käsittelyanimaatioiden mallina käytin YouTube-videota, jossa opetettiin jousiammunnan perusasennot (11). Pysäytin videon ja kopioin videon osoittaman asennon mallille. Jos automaattisesti luotu animaatio päättyen

tähän loppuasentoon ei ollut ilman väliavaimia miellyttävä, lisäsin aikajanalle uuden väliavaimen ja asennon.

KUVA 24. Hahmon tähtäysanimaatio ja avaimet aikajanalla

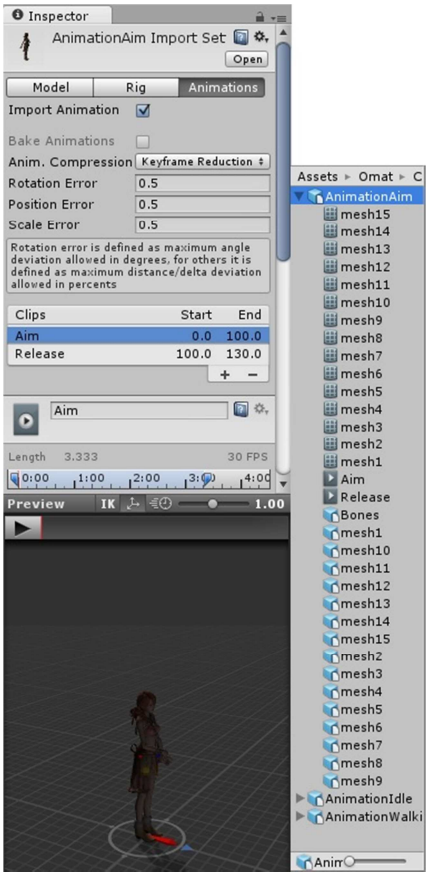
3.3.4 Hahmon ja animaatioiden tuonti Unityyn

Hahmon animaatiot tallennetaan 3D-mallin tiedostoon. Tein jokaisesta animaatiosta ja hahmon 3D-mallista oman tiedostonsa, jotta pienten animaatiomuutosten tuonti Unityyn helpottuisi. Animaatiot sisältävät 3D-mallit tuodaan Unityyn samaan tapaan kuin 3D-malli, joka ei sisällä animaatiota. Kun malli on tuotu Unityyn, tarkistetaan aluksi että Unity on tunnistanut mallin luut oikein (kuva 25). Mallin asetuksista valitaan animaatiotyypiksi humanoidi. Tämän jälkeen mallille luodaan uusi avatar mallin mukaan. Sitten mallin luita pääsee tarkistelemaan painamalla Configure-näppäintä. En muuttanut luiden nimiä rigatessani mallia, vaan käytin automaattista nimeämistä, ja ehkä siksi Unity sekoitti mallin peukalon ja etusormen luut keskenään. Luiden määritykset pystyi kuitenkin muuttamaan Unityssä mallin luiden asetuksista.



KUVA 25. Hahmon mallin asetukset Unityssä ja luuasetusikkuna

Käytin kaikissa muissa Unityyn tuomissani hahmomalleissa, jotka sisältivät animaatioita, samaa avataria. Pystyin tekemään näin, koska tiesin kaikkien 3D-mallien luustojen olevan tismalleen samanlaisia, joten asetukset saivat kopioitua. Jotta animaatioita sisältävien mallien animaatiot saatiin käytettäväksi, piti ne määritellä Unityssä (kuva 26). Määrittely tapahtui mallin asetuksista Animations-välilehdellä. Täältä valittiin Import Animation, jolloin hahmon animaatiot tulivat käytettäväksi asetuksiin. Jotta animaatioita voitiin käyttää, ne piti valita aikajanalta ja nimetä, jonka jälkeen animaatiot näkyivät mallin alaobjekteina.



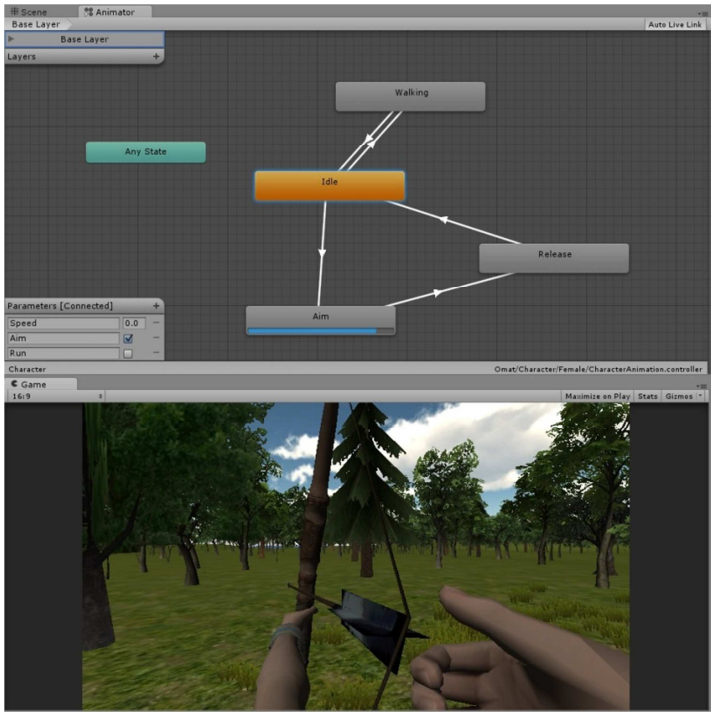
KUVA 26. Hahmon animaatiomallin asetukset ja animaatiot objekteina

3.3.5 Hahmon liikkuminen pelissä

Saadakseni hahmon liikkumaan pelissä tein ensimmäisen persoonan ohjaimen mukana tulevien ohjausskriptien lisäksi hahmolle CharacterControl-skriptin. CharacterControl-skripti ohjaa hahmon animaatioita Animator-kontrollerin kanssa. CharacterControl-skripti luo ja ohjaa myös nuolen liikkeet nuolen osumahetkeen saakka. Hahmon liikkeitä ohjaavat myös hahmon päässä ja selkärangassa sijaitsevat MouseLook-skriptit.

Animator-komponentti toistaa pelin pyöriessä oletustilan animaatiota kunnes jokin oletustilasta toiseen tilaan ohjaavan tilasiirron ehto toteutuu (kuva 27). Kun hahmon nopeus nousee, siirtyy komponentti toistamaan kävelyanimaatiota. Pelaajan painaessa ampumisnäppäintä, CharacterControl-skripti muuttaa Animator-komponentin Aim-float-arvon ja täten toteuttaa tilasiirron tilaan Aim, jossa

toistetaan hahmon tähtäysanimaatio. Kun painike päästetään, siirrytään Release-tilaan, josta suoraan tietyn ajan kuluessa takaisin Idle-tilaan.



KUVA 27. Hahmon animaatiotilat

Ammuttava nuoli on projektiin tehty prefab. Pelaaja painaessa ampumisnäppäintä Animator toistaa tähtäysanimaatiota samaan aikaan, kun CharacterControl-skripti tekee kloonin nuolen prefabistä (kuva 28) ja sijoittaa sen jouselle. Nuolen klooni asetetaan Rigidbodyksi, jolloin siihen vaikuttaa suoraan painovoima. Painovoiman vaikutus nollataan nuolen luontivaiheessa, jotta nuoli ei tipahtaisi suoraan pois jouselta ennen sen ampumista. Samalla poistetaan käytöstä hahmon päähän vaikuttava MouseLook-skripti, joka kääntää päätä idle-tilassa hiiren mukana, ja otetaan käyttöön kaksi selkärangan vastaavaa skriptiä, joilla saa ohjattua tähtäyksen korkeutta.

```
//Aim
if(Input.GetButton("Fire2") && CameraLock == false)
{
    Head.GetComponent<MouseLook>().enabled = false;
    Spine.GetComponent<MouseLook>().enabled = true;
    Spine.parent.GetComponent<MouseLook>().enabled = true;
    anim.SetBool("Aim", true);
    if(currentBaseState.nameHash != releaseState)
    {
        if(arrowShot == true)
        {
            print("Arrow clone created");
            clone = Instantiate(Arrow) as Rigidbody;
            clone.useGravity = false;
            clone.isKinematic = false;
            arrowShot = false;
            firstArrowCloned = true;
            arrowVelocitySet = false;
        }
        clone.transform.position = BowString.position;
        clone.transform.LookAt(GameObject.Find("BowCenter").transform.position);
    }
}
else
{
    anim.SetBool("Aim", false);
}
```

KUVA 28. Tähtäysskripti ja nuolen kloonin luonti

Kun pelaaja on tähdännyt ja päästää ampumispainikkeen, siirtyy hahmo release tilaan. Tässä tilassa kontrolleriskripti antaa nuolen kloonille eteenpäin suuntaavan voiman ja ottaa painovoiman vaikutuksen takaisin käyttöön (kuva 29). En tehnyt jouselle erillistä animaatiota jousen jännityksen näyttämiseksi, vaan siirsin sen paikkaa suoraan koodista. Kun nuoli ammutaan, jousen jännitys palautuu takaisin oletuskohtaan. Ilman erillistä koodia nuoli lentäisi samassa asennossa koko lentomatkansa ajan, mutta lisäämällä LookRotation-komennon nuoli kääntyy nopeuden osoittamaan suuntaan.

```
if(currentBaseState.nameHash == releaseState)
{
    BowString.transform.position = StringDefault.position;
    arrowShot = true;
    if(!arrowVelocitySet)
    {
        areaEvent.ArrowShot();
        clone.velocity = clone.transform.TransformDirection(new Vector3(0,0,30f));
        arrowVelocitySet = true;
        print("velocity set");
    }
    clone.useGravity = true;
}
else
{
    BowString.transform.position = Hand.position + new Vector3(0f,0.03f,0f);
}

if(firstArrowCloned)
{
    if(clone.velocity != Vector3.zero)
    {
        clone.rotation = Quaternion.LookRotation(clone.velocity);
    }
}
```

KUVA 29. Release-tilan koodi ja nuolen lentoradan ohjaus

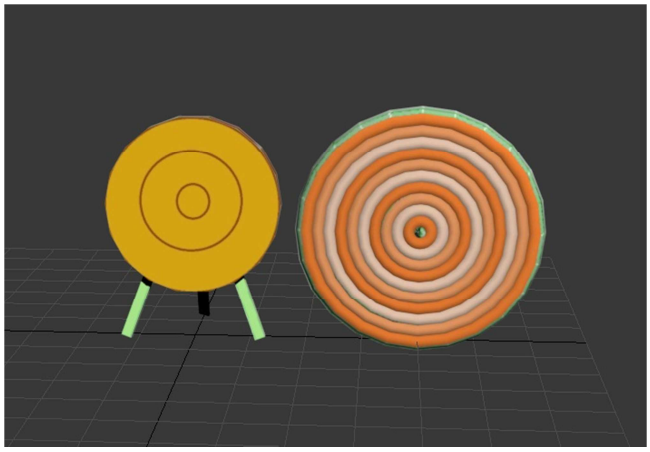
3.4 Harjoitusrata

Harjoitusrata on alue pelikentällä, jossa voi harjoitella jousella ampumista. Radalle käveltäessä peli kysyy haluaako pelaaja aloittaa harjoituksen. Harjoituksen aikana ammutaan viisi nuolta, joiden osumapisteet lasketaan. (Kuva 30.)

KUVA 30. Harjoitusrata

3.4.1 Maalitaulut

Projektissa käytetyt maalitaulut ovat täysin omatekemiä. Aloitin tekemällä yksinkertaisen maalitaulun yhdestä sylinteristä ja lisäämällä siihen jalustimen (kuva 31). Maalitaulu näytti kuitenkin karulta ja päätin tehdä toisen vedoksen, jossa käytin useata pyöreää ympyrää. Siitä tuli kuitenkin huonon näköinen ja päädyin tekemään kolmannen maalitaulun (kuva 32). Kolmannessa maalitaulussa käytin samoja pyöreitä ympyröitä, mutta kasvatin niiden leveyttä. Ulkonäön parantamiseksi käytin maalitaulussa olkitekstuuria pelkkien värien sijaan.



KUVA 31. Maalitaulujen kaksi raakavedosta

KUVA 32. Valmis maalitaulu

Maalitaulujen tekstuureiden liittämiseksi maalitaulujen pinnat piti ensin avata tasoksi (kuva 33). Kun pinnat ovat tasolla, kaikki erilliset pinnat voidaan värjätä. Kun valmis värjätty kuva valitaan mallin tekstuuriksi, värjätyt kohdat näkyvät mallin pinnalla. Tasolla näkyvien pintojen välissä oleva musta alue ei vaikuta itse mallin tekstuureihin, joten niiden päälle voi vapaasti maalata samaa väriä.

KUVA 33. Maalitaulun pinnat tasolla ja värjättynä tekstuureilla

3.4.2 Alueskripti

Harjoitusalueen ampuma-alueella on näkymätön Box Collider. Kun hahmo kävelee siihen sen ja alueen colliderit törmäävät keskenään aiheuttaen OnTrigger-Enter-eventin. OnTriggerEnter-eventissä alueskripti tulostaa näytölle valikon (kuva 34), joka kysyy haluaako pelaaja aloittaa harjoituksen. Kysymyslaatikon ollessa esillä, hiiren kontrolli siirtyy hahmosta kursoriin, jolla voi valita aloittaako harjoituksen vai ei.

KUVA 34. Harjoituksen aloitus

Aloitettaessa harjoituksen alueskripti laskee ammuttujen nuolien määrän. Kun nuolia on ammuttu viisi, harjoitus loppuu ja näytölle tulostetaan saavutettu osumapistemäärä. Osumapistemäärä lasketaan nuolen ja maalitaulun törmäyspisteen mukaan. Kun nuoli osuu johonkin, sen kärjessä oleva collideri aiheuttaa OnCollisionEnter-eventin, joka käsitellään nuoleen liitetyssä ArrowCollision-skriptissä (kuva 35). Skripti pysäyttää nuolen osumapisteeseen ja poistaa siihen kohdistuvan painovoiman vaikutuksen. Skripti lähettää törmäystapahtuman tiedot AreaEvent-skriptin metodiin AddScrore (kuva 36), jossa tutkitaan onko nuoli osunut johonkin alueella olevista maalitauluista. Jos maalitauluun on osuttu, skripti tutkii ScoreByDistance-metodissa osumapisteen etäisyyden taulun keskikohdasta ja palauttaa tämän mukaisen pistemäärän, joka kerrotaan osutun maalitaulun etäisyyskertoimella.

```
using UnityEngine;
using System.Collections;

public class ArrowCollision : MonoBehaviour {

    void OnCollisionEnter(Collision collision) {
        print("Collision with " + collision.transform.name);
        rigidbody.velocity = Vector3.zero;
        rigidbody.angularVelocity = Vector3.zero;
        rigidbody.isKinematic = true;
        rigidbody.useGravity = false;
        GameObject.Find("Character").GetComponent<AreaEvent>().AddScore(collision);

        //Destroy the arrow 2 minutes after the collision
        //Destroy(gameObject, 120);
    }
}
```

KUVA 35. Nuolen törmäysskripti

```
public void AddScore(Collision collision)
{
    ContactPoint contact = collision.contacts[0];
    Vector3 collisionPointOnTarget = collision.transform.InverseTransformPoint(contact.point);
    float distance = Vector3.Distance(collisionPointOnTarget, new Vector3(0f,0f,0f));

    if(collision.transform.name == "Target20")
    {
        score += ScoreByDistance(distance) * 1;
    }
    else if(collision.transform.name == "Target35")
    {
        score += ScoreByDistance(distance) * 2;
    }
    else if(collision.transform.name == "Target50")
    {
        score += ScoreByDistance(distance) * 3;
    }
    else if(collision.transform.name == "Target70")
    {
        score += ScoreByDistance(distance) * 4;
    }
}

private int ScoreByDistance(float distance)
{
    if (distance < 0.044f)
    {
        return 70;
    }
    else if (distance < 0.11f && distance > 0.044f)
    {
        return 50;
    }
    else if(distance < 0.17f && distance > 0.11f)
    {
        return 30;
    }
    else
        return 10;
}
```

KUVA 36. AreaEvent-skriptin osumapisteiden laskukoodi

3.5 Päävalikko

Viimeiseksi tein projektille aloitusikkunan. Aloitusikkuna on toinen 3D-maasto-alue. Maastossa on kolme 3D-tekstiobjektia, yksi pelin otsikolle ja kaksi pelin aloitus- ja lopetuspainikkeille (kuva 37). Painikkeisiin on liitetty skriptit, jotka tutkivat, osuuko hiiri tekstialueelle. Hiiren osuessa tekstialueelle tekstin väri muuttuu. Skripti tutkii myös, klikataanko tekstiä hiirellä, ja ajaa sen osoittaman tapahtuman koodin, joilla joko sammutetaan peli tai ladataan itse pelikentän scene.

KUVA 37. Päävalikko

4 YHTEENVETO

Työn tavoitteena oli tutustua pelikehitykseen ja 3D-mallinnukseen Unity3D-pelikehitysympäristössä sekä saada aikaan pohja ensimmäisen persoonan 3D-pelille. Tavoitteeseen päästiin, vaikka projektin aikana tulikin vastaan useita haasteita. Eniten haasteita toi 3D-mallinnus, joka vei projektissa uskomattoman paljon aikaa. Unity3D:llä työskentely oli kuitenkin helpompaa kuin olisi voinut aluksi uskoa. Unity tarjosi todella helpon käyttöliittymän pelien luontiin. Pelin toiminnallisuuden ohjelmointi oli helppoa, sillä C#-kieli oli ennestään hyvin hallussa koulun kurssien ja harjoitteluprojektien ajalta.

Projektin aikana huomasin, kuinka paljon työtä pelikehitys vaatii. Varsinkin peligrafiikoiden ja -mallien tekemiseen kuluva aika yllätti. Normaalisti pelikehityksessä käytetään kuitenkin yleensä tiimejä, joissa tehtävät on jaettu tiettyjen osa-alueiden ammattilaisille, mikä nopeuttaa kehitysprosessia.

Opin projektin aikana Unity3D-pelikehitystyökalun käytön lähes kaikkine osa-alueineen. Opin myös 3D-mallinnuksen perustat ja pelihahmojen valmistelun peliympäristöön vientiä varten. Varsinkin hahmojen riggauksen ja animoinnin pienet kikat tulivat hyvin tutuksi. Haluaisin jatkossa myös mahdollisuuden kokeilla motion capture -animointia. Pelitoiminnallisuuden ohjelmointi lisäsi hyvin C#-kielen osaamista varsinkin pelikehitysympäristössä.

Projektin aikana syntynyt peli ei tullut valmiiksi, eikä se ollut opinnäytetyön puitteissa tarkoituskaan. Työtä tehdessä kiinnostus pelikehitystä kohtaan kuitenkin kasvoi ja tarkoituksena on, että jatkan pelin kehityksen loppuun. Toiveena on myös, että voisin tulevaisuudessa työskennellä pelikehitystiimissä.

LÄHTEET

1. Unity Technologies 2013. Saatavissa: http://unity3d.com/. Hakupäivä 18.4.2013.

2. Autodesk 3DS Max 2013. Saatavissa: http://www.autodesk.com/products/autodesk-3ds-max/overview. Hakupäivä 22.4.2013.

3. Autodesk FBX Converter 2013. Saatavissa: http://www.autodesk.mx/adsk/servlet/pc/item?siteID=123112&id=10775855. Hakupäivä 22.4.2013.

4. GIMP. The GIMP Team 2013. Saatavissa: http://www.gimp.org/. Hakupäivä 23.4.2013.

5. dotPDN LLC – Rick Brewster 2013. Paint.NET. Saatavissa: http://www.getpaint.net/. Hakupäivä 23.4.2013.

6. Image Magick. ImageMagick Studio LLC 2013. Saatavissa: http://www.imagemagick.org/script/index.php. Hakupäivä 23.4.2013.

7. MRxTeabag 2013. Unity 3d FPS Tutorial | In Depth from Scratch. Saatavissa: http://youtu.be/UZWwDLS1j58. Hakupäivä 23.4.2012

8. Grid Lab 2010. Unity 3D Terrain. Saatavissa: http://vimeo.com/7884429. Hakupäivä 24.4.2013.

9. Ring of Rods 2013. Creating a raw file. Saatavissa: http://www.rigsofrods.com/wiki/pages/Creating_a_raw_file. Hakupäivä 23.4.2013.

10. Unify community 2013. Free Game Content. Saatavissa: http://wiki.unity3d.com/index.php?title=Free_Game_Content. Hakupäivä 24.4.2013.

11. Howcast 2012. Pulling the Bow Back Properly | Archery and Bow Hunting. Saatavissa: http://www.youtube.com/watch?v=GIC7OD792iQ. Hakupäivä 24.4.2013.