KYMENLAAKSON AMMATTIKORKEAKOULU

Ohjelmistotekniikan koulutusohjelma

Niko Arvilommi

AWAKENING-PELIPROJEKTI

Opinnäytetyö 28.5.2013

TIIVISTELMÄ

KYMENLAAKSON AMMATTIKORKEAKOULU

Tietotekniikka

NIKO ARVILOMMI	Awakening- peliprojekti
Opinnäytetyö	32 sivua
Työn ohjaaja	Yliopettaja Paula Posio,
Toimeksiantaja	Kymenlaakson AMK
Maaliskuu 28.5.2013	
Avainsanat	C#, peliohjelmointi, Unity3d, projektinhallinta

Mobiililaitteiden suosion sekä pelialan kasvusuhdanteiden kasvaessa myös pelien tekeminen kyseisille laitteille kiinnostaa entistä enemmän. Mobiililaitteiden ansiosta niin sanotut autotalliyritykset, jossa on vain muutama henkilö, ovat taas nosteessa. Tämän myötä myös pelien kustannukset ovat laskeneet ja tuotot kasvaneet.

Opinnäytetyön aiheena on suunnitella 2.5D tasoloikka-pulmanratkonta peli. Lähtökohtana oli, että loisimme jotain erilaista, sellaista mitä mobiililaitteilla ei vielä ole, ja jota haluaisimme myös itse pelata. Tämän lisäksi työssä tutustutaan projektinhallintaohjelmistoihin ja niiden asennukseen sekä Scrum-kehykseen.

Peli on toteutettu Unity3d-pelimoottorilla käyttäen monodevelop C#-ohjelmointikieltä. Tämän lisäksi työssä on valmistettu oma GUI-editori ja GUI-rakenne, jota hyödynnetään pelissä. Peli on toteutettu yhteistyössä Kouvolan media-alan opiskelijoiden kanssa

ABSTRACT

KYMENLAAKSON AMMATTIKORKEAKOULU

University of Applied Sciences

Information Technology

ARVILOMMI, NIKO	Awakening game project
Bachelor's Thesis	32 pages
Supervisor	Paula Posio, Senior Lecturer
Commissioned by	Kymenlaakso University of Applied Sciences
March 5/28/13	
Keywords	C#, game programming, Unity3d, project management

The increasing popularity of mobile devices and growth of Finnish game industry, also the development for these devices raised more interest. Because of the mobile devices, so called Garage companies have had a positive impact. Garage companies are founded and owned only by a few people. In addition, because of the digital distribution and the expectation of players are different than of players in other gaming platforms, games can be smaller in content size.Therefor the costs of making these games are lower and the revenues are higher.

The purpose of this study was to design a 2.5D platformer puzzle game. One starting point was to create something different than other developers are creating. Another objective was to explore the project management software and how to install those, also the Scrum framework was reviewed. The game was produced with Unity 3D – engine using the monodevelop C# programming language. A GUI editor and GUI structure, which were used in the game, were also included. The project was made in collaboration with Kouvola media students.

TERMIT JA LYHENTEET

C#	Ohjelmointikieli.
iPad	Apple-yhtiön valmistama mobiililaite.
Android	Googlen valmistama mobiililaitteiden käyttöjärjestelmä.
WYSIWYG	What-you-see-is-what-you-get
GUI	Graphical User interface, graafinen käyttöliittymä.
MESH	3d-objektin rautalankamalli tai -verkko.
FSM	Finite state machine. Tilakonemalli.
SVN	Subversion, versionhallintajärjestelmä.
IK-Solver	Inverse kinematic solver, ohjaa hahmon luusegmenttejä haluttuun asentoon.

SISÄLLYS

Awakening-peliprojekti

1 JOHDANTO

Opinnäytetyö on toteutettu yhteistyössä Kouvolan media-alan opiskelijoiden kanssa. Projektin suunnittelu ja taustatyö alkoi tammikuussa 2012 KyAMKin ja Cursor Oyn yhteishankkeen yhteydessä, jossa opiskelijoita kutsuttiin pelisuunnittelu-kurssitukseen. Työn tulos on tarkoitus olla prototyyppi, josta voitaisiin tehdä lopputuote digitaalisiin pelikauppoihin ja myydä mahdollisille julkaisijoille.

Työssä on tarkoitus tutkia pelisuunnittelua, projektinhallintaa ja niihin vaadittavia ohjelmistoja. Lisäksi on tarkoitus tutkia edellä mainittujen ohjelmistojen asennuksia, sekä mahdollisuuksia ja mobiililaitteiden kasvavaa suosiota pelialustana. Varsinainen peli on toteutettu Unity3D-pelimoottorilla ja C#-ohjelmointikielellä.

2 MOBIILILAITTEET PELIALUSTANA

Mobiililaitteiden suosio ovat olleet kovassa kasvussa viimeisen vuoden aikana, kiitos Steve Jobsin pioneerityön. iPad-laitetta ei alun perin suunniteltu pelikäyttöön laisinkaan, vaan eräänlaiseksi niin sanotuksi lisäkädeksi toimistoihin ja internetin selaukseen kotisohvalle. Peliala on saavuttanut mobiililaitteet ja nostanut ne aivan eri tasolle. Kuitenkaan pelialalla ei monikaan yritys ole onnistunut luomaan onnistuneita tuotteita kosketusnäyttölaitteille. Mobiililaitteiden moninaiset rajoitukset antavat erittäin mielenkiintoisia haasteita pelisuunnittelun näkökulmasta. Miten toteuttaa pelaajahahmon ohjaus? Millä? Kuinka monimutkainen tuote voi olla? Miten tuote voidaan myydä? Toisaalta, peliala on joutunut suunnittelumalleissaan aloittamaan puhtailta pöydiltä.

2.1 Mobiililaitteiden tekniset haasteet

Mobiililaitteiden prosessoriteho sekä muistin määrä tuovat erittäin suuren haasteen, sillä ohjelmiston ja pelien kehittäjät ovat tottuneet ajatukseen kasvavasta tehosta ja muistinmäärästä. Lisäksi Android-laitteiden suuri hajonta, eli valmistajien eri mallien määrä on niin suuri, että kehittäjien on hankala tehdä optimoituja pelejä tai edes varmistaa, että pelit toimivat kaikilla Android-pohjaisilla laitteilla. Kaikki valmistajat

eivät edes täytä Googlen antamia vaatimusmäärittelyjä, joten tekniset haasteet, etenkin Android-puolella, ovat erittäin moninaiset. Jossa kehittäjän on kuitenkin syytä huomioida vain suurimmat nimikkeet, joita ovat Samsung HTC, Asus ja Google-Nexus. Vaikka hän keskittyisi vain kyseisiin laitteisiin, on hajonta silti satoja eri Android-malleja.

Applen leirissä hajonta on jo huomattavasti pienempi. Siellä kehittäjän tarvitsee ottaa huomioon vain ja ainoastaan aikaisemmat versiot.

2.2 Mobiililaitteiden tulevaisuus

Mobiililaitteiden kehityskaari pelialustana on vasta alussa, mutta mobiilipelit ovat pystyneet määrittämään tai muuttaneet koko peliteollisuuden ansainta-logiikan. Aikaisemmi pelit myytiin kappaleittain, josta määräytyi tietty määrä euroja julkaisukanavaan. Varsinainen pelintekijän oli tyytyminen julkaisijan maksamaan kehitysrahaan. Tästä johtuen pelintekijän varsinainen palkkio jäi hyvin pieneksi. Mobiilipelaamisen aikakaudella pelit ovat muuttuneet maksullisista kertaostoista ilmaisiin tuotteisiin(free-to-play), joissa tarjotaan vapaaehtoisia sisäisiä digitaalituotteita. Nämä ostetaan joko oikealla tai pelinsisäisellä valuutalla. Mobiilipelien tuottaminen on yleisesti halpaa, riippuen tietenkin peliprojektin vaatimusmäärittelyistä ja laajuudesta, joten pelintekijän ei välttämättä ole pakko käyttää julkaisijakanavaa saadakseen pelinsä markkinoille.

Ne mobiilipelit, jotka ovat menestyneet hyvin, ovat pääsääntöisesti ilmaisia ja varsinainen tuotto tulee mainosten ja pelin sisällä tarjottavilla lisätuotteilla. Mobiililaitteet pelialustana eivät tule tulevaisuudessa häviämään mihinkään, vaan ne ottavat paikkansa joko konsolien rinnalla niin sanottunalisälaitteena tai yksinäisenä lisälaitteena, jolla voidaan pelata. Hyvin käytetty mobiilipelitekniikka osaisi tulevaisuudessa hyödyntää konsolilla tai tietokoneella pelaamisen, jossa yhdistettäisiin mobiililaite esimerkiksi niin sanottuna lisänäyttönä taikka lisäohjaimena. Tämä kirjoittajan ennustus todennäköisesti tulee toteutumaan seuraavan 2 - 3 vuoden aikana, sillä mobiililaitteiden suosio kasvaa jatkuvasti. Monet toimistot ja koulut ovat ottaneet tabletit jokapäiväiseen käyttöön. Konsolivalmistaja Nintendo on tehnyt oman tuotteensa, jolla tämä on jo mahdollista, kuitenkaan kyseinen laite ei ole mobiililaite, vaan johdannainen mobiililaitteista.

3 PROJEKTINHALLINTA

Projektinhallinta tarkoittaa käytännössä resurssien organisointia ja hallintaa sellaisella tavalla, että projekti voidaan päättää suunnitellun sisältöisenä ja laatuisena sekä siinä aikataulussa, joka sille on määritelty. Projektin resursseihin voidaan lukea mm. raha, työvoima, raaka-aineet jne(6).

Projektiin nimitetään projektipäällikkö. Projektipäällikkö ei yleensä osallistu itse tuotantoon vaan keskittyy projektin etenemisen sekä muiden osapuolten yhteistyön varmistamiseen. Toisin sanoen projektipäällikön toimenkuva voidaan mieltää niin, että hän toimii asiakkaan edustajana(6).

Projektinhallintaa varten valitut menetelmät ovat yleensä ohjelmistoprosessissa vaikeinta osa-aluetta, etenkin työtapojen ylläpito. Ohjelmistoprojekteissa on monia erilaisia projektinhallinta menetelmiä. Yleisimpiä ohjelmistoprojektin hallinta menetelmiä ovat Agile, RUP ja vesiputousmalli, joka on yleisen tason projektinhallintamenetelmä.

3.1 Projektinhallintaohjelmistot

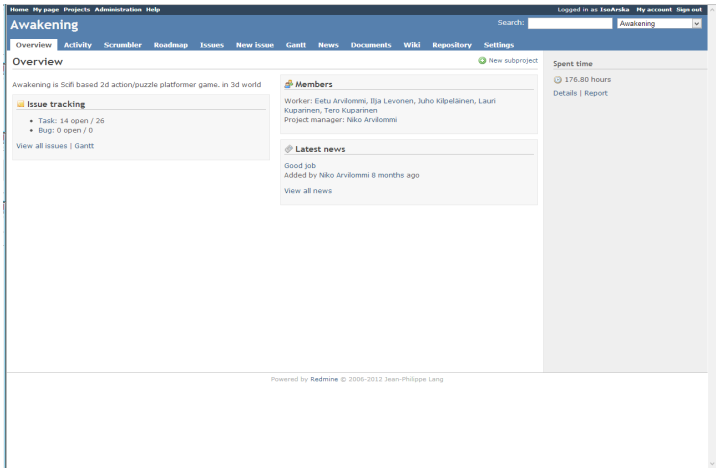
Projektinhallinta ilman pätevää ohjelmistoa on lähestulkoon mahdotonta, koska projektit ovat yleensä niin laajoja, että pelkillä paperilapuilla ei selvitä. Parhaimmissa projektinhallintaohjelmistoissa voidaan seurata projektin etenemistä sekä mahdollisia sprinttien onnistumisia, että tarkastella ja luoda aikataulutusta. Yleensä projektinhallintaohjelmiston valinta kannattaa tehdä suoraan niihin menetelmiin, joita omassa toimessaan käyttää. Tässä huomioidaan muutamia, jotka kykenivät toteuttamaan Scrummin antamat ehdot.

Useimmat ohjelmistot ovat pelkistettyjä aikataulutusohjelmia, joilla ei varsinaista projektinhallintaa voida viedä pitkälle. Ne perustuvat Gantt-kaavioihin ja vesiputousmalleihin. Hyvänä huonona esimerkkinä voidaan pitää Gantt-project-ohjelmaa, joka toimii paikallisesti ja tehtävien luominen ja ylläpitäminen on todella hankalaa.

3.2 Redmine ohjelmisto

Redmine on projektinhallintaohjelmisto, joka on laajennettavissa moneen käyttöön kolmannen osapuolen liitännäisillä. Redmine on kirjoitettu Ruby- ohjelmointikielellä ja on open-source-pohjainen. Pääsääntöisesti Redmine-ohjelmaa käytetään tehtävien seurantaan, eli ohjelmistossa on projektien backlogit sekä tulevat sprintit.

Redmine-ohjelmiston asentaminen ei ollut aivan yksinkertainen prosessi ja niiden käsitteleminen tässä dokumentissa ei ole relevanttia, vaan seuraamalla viitettä (2) selviää kyseisen ohjelmiston käyttöönottoprosessi. Lopputulos on näkyvissä kuva 1:ssä.



Kuva 1: Redmine-ohjelmiston perusnäkymä

Redmine-ohjelmistossa olevassa projektissa voi olla erilaisia tehtävälistoja. Kyseistä ohjelmaa voidaan käyttää projektin varsinaisena tehtävälistana, sekä bugien eli virheiden tehtävälistana. Eli ohjelmistontestaus vaiheessa löytyneitä virheitä on listattu tehtävälistaan ja luotu niistä tehtäviä. Löytyneiden virheiden listaaminen ja virheisiin reagoiminen on nopeaa, ja niiden etenemisen seuranta onnistuu kätevästi yhdestä paikasta.

Tämän lisäksi ohjelmistoa voidaan laajentaa erilaisilla liitännäisillä. Kirjoittamishetkellä liitännäisten saatavuus on todella hyvä. Redminen käyttö varsinaisessa sprinttikäytössä jäi vähäiseksi, sillä projektin edetessä sprinttien suunnittelusta tuli lähes mahdotonta, koska tiimin jokainen jäsen tuotti materiaalia omalla vapaa-ajallaan, joten tämä työkalu jäi lähinnä dokumentaatioalustaksi.

3.3 Version hallinta

Versionhallinnan tarkoitus on mahdollistaa ohjelmistokomponenttien ristiriidaton kehitys. Tämä käytännössä tarkoittaa sitä, että kahdesta useampaa ohjelmoijaa tai muuta projektin parissa työskentelevää henkilöä voivat jakaa ja muokata samoja projektitiedostoja samanaikaisesti, välittämättä toisesta henkilön toimista.

Versionhallinta mahdollistaa myös ohjelmiston seurannan, kehityssuuntien pilkkomisen erisuuntiin. Tämä saattaa olla välttämätöntä ohjelmiston kehittämistyössä eri alustoille tai asiakkaille. Versionhallinta mahdollistaa kehitysversioiden vertailun sekä kehitysversion palauttamisen aikaisempaan versioon.

4 AGILE / KETTERÄ OHJELMAN KEHITYS

Agile- ohjelmistonkehitys on joukko erilaisia menetelmiä. Ensisijaisesti Agile-ohjelmistokehityksessä on suora viestintä sekä nopea reagointi muutoksiin.

Monet Agile menetelmät pyrkivät minimoimaan riskejä jakamalla kehityksen elinkaaren lyhyisiin iteraatioihin, tyypillisesti 1 – 4 viikkoon. Jokainen iteraatio voidaan ajatella olevan pieni ohjelmistoprojekti, sillä jokainen iteraatio sisältää kaikkien uusien toimintojen julkaisemiseen tarvittavat tehtävät: suunnittelu, analyysit, koodaus ja testaus sekä mahdollinen dokumentointi. Jokainen iteraatio pyrkii periaatteessa julkaisukelpoiseen tuotteeseen.

Agile- menetelmissä jokaisen iteraation sisältö jaetaan tiimeittäin ja tiimit päättävät itsenäisesti, mitä iteraation sisällä on mahdollista toteuttaa. Agile- menetelmät korostavat tai pitävät suoraa viestintää tärkeämpänä kuin itse dokumentteja. Tästä aiheutuukin harhakuvitelma, että Agilen kehitys olisi mielivaltaista tai että suunnittelmia ei tehtäisi ollenkaan. Näin ei kuitenkaan ole, suunnittelua ja siihen dokumentaatiota luodaan ja päivitetään jatkuvasti. Olemassa olevia suunnitelmia ollaan halukkaampia muuttamaan kuin muissa malleissa.

5 SCRUM

Scrum on projektinhallinnan kehys, jota yleensä käytetään ohjelmistokehityksessä. Se on hyvin suosittu tyyli suomalaisissa ohjelmistoyrityksissä. Vaikka Scrum on kehitetty ohjelmiston kehitystä silmälläpitäen, sitä voidaan myös soveltaa yleisesti

projektinhallinnassa. Scrumin sisällä työskennellään iteratiivisesti ja lisäävästi riskien kontrolloimiseksi, sekä ennustettavuuden optimoimiseksi Tavoitteena oleva tuote kehittyy pikkuhiljaa täydellisemmäksi useiden eri iteraatioiden aikana. Yhtä iteraatiota kutsutaan sprintiksi. Sprintin yleinen pituus on 1 – 4 viikkoa, jonka sisällä tuotetaan määritelmän täyttävä ja potentiaalisesti julkaisukelpoinen versio(5).

Jokaisen iteraation sisältö sovitaan sprint- suunnittelupalaverissa ennen sprintin aloitusta. Toteutukseen valitaan sellaisia tehtäviä, joilla on suurin mahdollinen merkitys projektin onnistumiselle. Sprintin lopuksi järjestetään sprint-katselmus eli sprint review, jossa kehitystiimi esittelee konkreettiset saavutukset. Jokaisen sprintin lopuksi ja ennen seuraavan aloittamista pidetään retrospektiivi, jossa tarkastetaan, mikä sprintin aikana sujui hyvin ja mitä voitaisiin vielä parantaa seuraavassa sprintissä(5).

5.1 Scrumin roolit

Scrum-kehyksen sisällä tiimin jäsenille annetaan erilaisia roolituksia. Scrum-tiimi koostuu tuoteomistajasta, scrum-mastereista ja kehitystiimistä. Scrum-tiimi päättää jokaisen sprintin tavoitteet ja tehtävät sekä vastaa siitä, että asetettuihin tavoitteisiin päästään.

Yksi tiimin jäsen valitaan tuoteomistajaksi. Tuoteomistajan pääasiallinen rooli on vastata tuotteen arvosta ja koko kehitystiimin työn ylläpitämisestä. Tämän tuotteen omistaja varmistaa määrittelemällä tuotteen vaatimukset sekä järjestelemällä tuotteen kehitysjonon (product backlog). Tuoteomistaja on yksi oikea henkilö, ei yrityksen hallitus, ryhmä tai komitea. Tuotteen omistajan täytyy ymmärtää tuotteeseen liittyvät liiketoimintamallit(5).

Yksi ehkä tärkeimmistä scrum-tiimeistä on itse kehitystiimi. Kehitystiimi koostuu osaajista, jotka kykenevät muuttamaan sprinttiin valitun kehitysjonon ts. tehtävälistan mahdolliseksi julkaisukelpoiseksi versioksi.

Scrum-master vastaa siitä, että jokainen tiimissä työskentelevä osaa käyttää ja ymmärtää Scrumia. Scrum-master on Scrum-tiimin palveleva johtaja. Scrum-masterilla itsellään ei ole suoraa määräysvaltaa, vaan hän vaikuttaa sprintin kautta. Scrum-master tarkkailee työn etenemistä. Sprintin tavoitteiden alkaessa näyttää epätodennäköisiltä, hän kommunikoi kehitystiimin ja tuoteomistajan kanssa. Mutta

kuitenkin Scrum-masterin tärkeimpiin rooleihin sisältyy tiimin suojaaminen ulkopuoliselta hälyltä, kuten sprintin aikana pyydetyiltä uusilta vaatimuksilta(5).

5.2 Scrumin tapahtumat

Sprintin suunnittelupalaverissa suunnitellaan sprintin aikana tehtävä työ. Tämä suunnitelma luodaan yhteistyössä koko tiimin kanssa.

Päiväpalaveri on enintään 15 minuutin mittainen. Palaverissa kehitystiimi tahdistaa työnsä ja luo suunnitelman kuluvalle työpäivälle.

Tuotteen kehitysjonon työstö tarkoittaa yksityiskohtien ja työmääräarvioiden, sekä työtehtävien järjestyksen tekemistä. Tämä toimenpide on toistuva prosessi. Sprinttikatselmus on sprintin lopussa pidettävä epämuodollinen palaveri, jossa tarkastellaan kehitettyä versiota ja muutetaan kehitysjonoa tarvittaessa.

Lopuksi pidetään vielä niin sanottu retropeksiivinen palaveri, jossa tarkastellaan edellisen sprintin huonoja ja hyviä puolia seuraavaa sprinttiä varten(5).

6 UNITY3D PELIKEHITYKSESSÄ

Unity on mahdollistanut uuden tavan luoda ja innovoida tuotteita nopeasti. Tästä syystä monella kokeneella ohjelmoijalla on suuri kynnys ja ymmärtämisen ongelma, kun Unity- moottoriin ensimmäisen kerran törmätään. Normaalisti ohjelmoinnissa dataa hallitaan luomalla suuria luokkakokonaisuuksia ja lukemalla mahdollista dataa joistakin tiedostoista, mutta Unityn tapa on heittää perinteinen luokkasuunnittelu roskiin. Unityllä luodaan yksittäisiä komponentteja, jotka lisäävät peli-objekteihin lisäominaisuuksia. Tämän lisäksi Unity tarjoaa monipuolisen editorin, jossa toimitaan drag & drop menetelmällä.

Oikein käytettynä Unity- pelimoottori on erittäin tehokas työkalu pelien luomisessa. Suurimpia hidastavia tekijöitä peliprojektissa on odottaminen, jotta toinen tiimin jäsen saa tehtyä jonkin nopean version, esimerkiksi animaatiosta, jota ohjelmoija saattaisi tarvita. Unitylla ohjelmoijakin voi tehdä animaation nopeasti testatakseen omaa komponenttiaan todistaakseen sen, että kyseinen suunnittelutapa on oikea, tai suunnitella animaattorin kanssa uuden. Näinollen ohjelmoija voi testata omat

algoritminsa luoden pienen animaation ja antaa vaatimusmäärittelyn tätä kautta animaattorille ja mallintajalle siitä, mitä muutoksia heidän täytyy tehdä.

Unity osaa kääntää kaikille tärkeimmille alustoille valmiita tuotteita ilman mitään suuria ohjelmistomuutoksia. Tämä on Unityn suurin etu, sillä monet muut pelimoottorit eivät taivu aivan yhtä mutkattomasti, vaikka tuki eri alustoille saattaisi ollakin. Peliohjelmoinnissa tämä tuo nopeutta ja hallintaa. Suurimmat muutokset tulevatkin optimointi vaiheessa, kun tekstuurin kokoja tai algoritmeja täytyy muuttaa, jotta peli pyörisi paremmin.

7 PALVELIMEN ASENNUS

Varsinaisessa tuotekehityksessä tarvittiin versionhallintaohjelmistoja sekä projektinhallintaohjelmistoja. Projektia varten asennettiin palvelin, joka täyttää nämä ehdot.

Paikallinen yritys ITMagick KY lahjoitti yhden palvelinlaitteiston sekä laitepaikan, johon voidaan perustaa vaadittavat palvelut, joita tarvitaan kyseiseen projektiin.

Palvelin käyttöjärjestelmäksi valittiin Centos, joka on Linux-pohjainen luotettava palvelin-käyttöjärjestelmä. Palvelinohjelmistoiksi valittiin Apache www- palvelin, SVN-version hallintapalvelin, Redmine-projektinhallintaohjelmisto sekä Webmin www-pohjainen järjestelmän hallintatyökalu ja USVN, jolla voidaan hallita SVN-palvelinta helposti.

7.1 Apachen asennus

Apache on open-source pohjainen www-palvelin, joka on erittäin laajassa käytössä ympäri maailmaa. Apache-palvelimelle ei ole Linux-maailmassa korvaajaa, joten valinta oli helppo tehdä. Erillistä asennusta ei tarvittu, vaan ohjelmistopaketti tuli asennuspaketin yhteydessä

7.2 SVN-palvelin asennus ja conffaus

Varsinaisen ohjelmistopaketin asennus on hyvin yksinkertaista, koska Centos-järjestelmässä on YUM-paketinhallintaohjelma. Ohjelma osaa hakea omista tietopankeistaan viimeisimmän version ja asentaa sen oikeaan paikkaan, jonka alla oleva (Listaus 1) esittää.

```
[root@svn ~]#  yum install mod_dav_svn subversion
```

Listaus 1: kyseinen komentosarja annetaan konsolissa tai komentorivillä.

Tämä komento asentaa Apacheen dav-moduulin. Dav-moduulilla svn-repositoryt voidaan jakaa http-protokollan kautta. Tämän jälkeen on luotava kansio turvalliseen hakemistoon vapaavalintaisella nimellä. Tässä tapauksessa on käytetty ohjeiden mukaista kansiota (1). Tämän jälkeen apache-palvelin on configuroitava, jotta luodut repositoryt näkyvät ulkomaailmaan. Toiminnot näkyvät Listaus 2:ssa

```
[root@svn ~]# cd /etc/httpd/conf.d/

[root@svn ~]# vim subversion.conf
```

Listaus 2: ensimmäinen komento avaa kansion conf.d jossa sijaitsee apachen configuraatio tiedot ja seuraava on vim teksti editor jolla avataan kyseinen configuraatio tiedosto.

Listaus 3 esittää kyseisen tiedoston sisällön, joka kertoo Apache-palvelimelle, mikä on repositoryn osoite ja missä se sijaitsee, sekä valitun authentikaatin tyypin.

```
        <Location /repos >
    DAV svn
    SVNPath /var/www/svn/repos
    AuthType Basic
    AuthName ”Awakening Repo”
    AuthUserFile /etc/svn-auth-conf
    Require valid user
</Location>
```

Listaus 3: Location komento syöttää asiakas selaimelle osoitteen esim. http://localhost/repos, jossa kyseiset svn tiedot sijaitsevat

8 AWAKENINGIN TEKNIIKAT

Awakening peliprojekti käynnistyi tammikuussa 2012. Ryhmä opiskelijoita Kouvolasta ja Kotkasta yhdistivät voimansa ja päättivät tehdä mielenkiintoisen ja erilaisen tasohyppelypelin. Pelissä on yhdistetty monien entisten vanhojen pelien

hyviä puolia sekä puzzle-tekniikkaa. Awakening sijoittuu kuvitteelliseen lähitulevaisuuden maahan, johon on iskeytynyt pieni asteroidi tuoden mukanaan uuden mikroskooppisen eliölajin. Laji valtaa ajan kuluessa koko maapallon väestön.

Pelin päähahmona toimii ihmismäinen robotti (Kuva 4.), joka yrittää selvittää, mitä maapallolle oikein tapahtui. Alkuperäiset suunnitelmat kattoivat vain PC-pelaamisen, mutta muokkautuivat myöhemmässä vaiheessa toimintapeliksi tabletille. Peli on toteutettu Unity3D-pelimoottorilla, koska kyseinen tekniikka tarjosi monet vaadittavat tekniikat valmiina sen sijaan, että olisimme joutuneet kasaamaan kokonaisen pelimoottorin itse. Unity sisältää kolme erilaista ohjelmointikieltä Boo, Java-script ja C#. Tähän projektiin valittiin C#-ohjelmointikieli Monodevelop-ympäristössä.

Unityn tarjoama C#-ohjelmointikieli vastaa toiminnaltaan Microsoftin omaa C#-ohjelmointikieltä. Siinä on kuitenkin muutamia pieniä puutteita. Unityssä oleva C#-ohjelmointikieli ei siis ole Microsoftin tukemaa.

Kuva 4: Pelin aloitustilanne.

8.1 Pelaaja

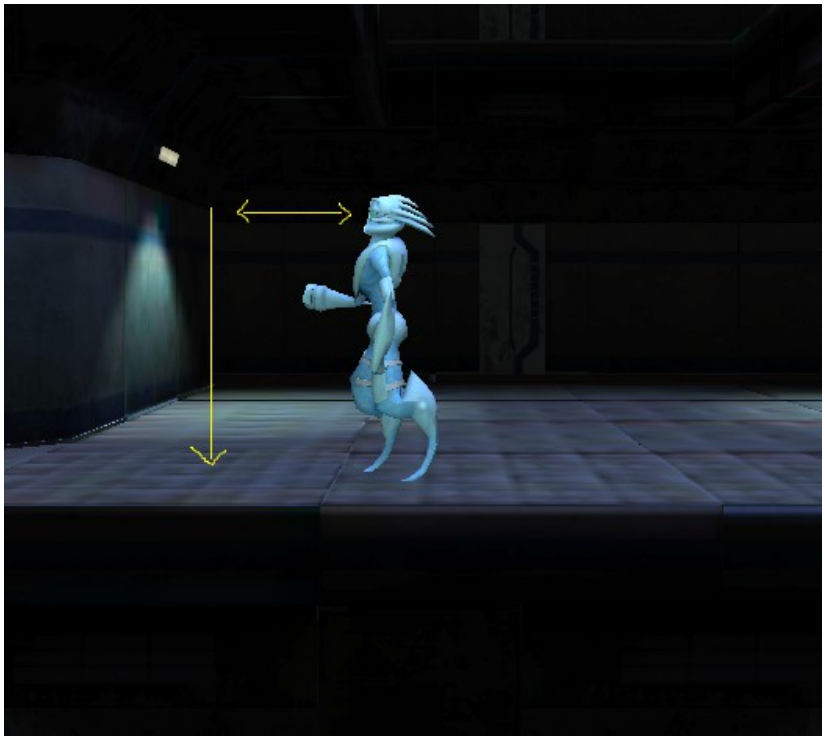
Pelaajahahmona toimii ihmismäinen robotti, joka osaa ampua ja hypätä. Varsinaiset maailma-toiminnot tapahtuvat pelaajan toimesta sormenpainalluksella tai raahamalla. Esimerkkinä pelimaailmassa on peiliobjekti, jota täytyy siirtää oikeaan kohtaan. Tämä

tapahtuu painamalla sormi peiliobjektin kohdalle ja raahamalla oikealle tai vasemmalle.

Pelaajahahmon rakentamiseen ja suunnitteluun käytettiin hyvä tovi. Pelaajahahmolle oli ohjelmoitava luiden ohjausta, eli IK-solver. IK-solverin tehtävä oli siirtää hahmon luu-objekteja katsomaan määrättyä pistettä kohden. Kun pelaaja liikuttaa hiirtä tai sormea, pelaajahahmon on seurattava hiiren kursorin liikettä. Tähän löytyi Unityn assetstoresta ilmainen lisäpaketti, joka sisälsi kaiken tämän. Pienellä muokkauksella onnistuin saamaan siitä tähän projektiin sopivan.

Unityn assetstore on Unityn tarjoama www-palvelu, jossa kehittäjä voi laittaa malleja, koodia tai muuta peleihin ja Unityyn liittyviä laajennuksia myytäväksi ja ilmaiseen jakeluun.

Jotta pelaajahahmo pysyisi objektien, kuten lattian päällä, ja ettei hahmo putoaisi tai kulkisi seinien lävitse, täytyi hahmolle kehittää törmäyksen tunnistus. Onneksi Unity-pelimoottorissa on integroituna Physx-fysiikkamoottori. Pelaaja-hahmon ympärille luotiin fysiikkakapseli, joka reagoi törmäyksiin ja voimiin. Pelaaja-hahmolta oli myös estettävä kaikki mahdollinen kaatuilu ja pyöriminen. Tämän lisäksi hahmolle oli kehitettävä ja suunniteltava liikutus-skripti. Normaalisti pelihahmoja liikutetaan antamalla koodillisesti haluttu suunta ja vauhti, mutta tämä aiheutti ongelmia. Ongelma hahmon kohdalla oli se, että kun liikkuessaan pelaajahahmo sai riittävästi vauhtia ja saavutti riittävän jyrkän alamäen joka oli yli 35 astetta. Hahmo hyppäsi mäestä alas, mikä ei ollut tarkoitus. Ratkaisu hahmon liikkuttamiseen oli seuraavanlainen: Pelaajahahmon pään korkeudelta ammutaan raycast maata kohden. Raycastin aloituskohdan erotusta pelaajahahmoon säädetään antamalla maksimi ja minimietäisyys (Kuva 5).



Kuva 5: Raycast on virtuaalinen säde, joka ammutaan 3D-maailmaan. Törmätessään raycast palauttaa tietoa törmäyksestä.

Raycastin törmätessä maa-objektiin sitä liikutetaan kyseistä törmäyspistettä kohden.

Pelaaja-hahmon ominaisuuksiin kuuluu myös ledge-grapping, eli pelaaja-hahmon on mahdollista roikkua kulmissa. Tässä käytettiin lähelle samantapaista metodia kuin kävelyssä, mutta tässä tilanteessa käytetään kahta erillistä raycast-toimintoa. Ensimmäiseksi raycast ammutaan pelaaja-hahmon pään korkeudelta suoraan alaspäin, jonka jälkeen tarkastellaan osuuko se sallittuihin objekteihin. Osuman tullessa tarkistetaan pintamateriaalin normaalit ja verrataan sitä ylöspäin olevaan vektoriin. Osuman ollessa annettujen kulma-arvojen välissä (Listaus 4), samanaikaisesti ammutaan pelaajan kulkusuuntan raycast, jolla tarkistetaan onko kosketuspinnan normaalit erisuuntaiset kuin pelaajan kulkusuunta. Kun nämä ehdot toteutuvat, sallitaan tarraus.

```
float fEdge = Vector3.Dot(hit.normal,Vector3.up);
```

Listaus 4: verrataan osuman normaalia ylöspäin olevaan vektoriin.

Dot-metodi on pistetulo, joka palauttaa arvoja -1 ja 1 väliltä. Jos arvo on 1, ovat vektorit samansuuntaiset. Tämän jälkeen tehdään tarkastelu, jos fEdge arvo on arvoltaan suurempi kuin 0.9, niin sallitaan tarraus ja siirrytään ledge-grap-tilaan (Listaus 5).

```
if( fEdge >= 0.9f )
{
 CharacterState_LedgeGrab grab = new CharacterState_LedgeGrab();
}
```

Listaus 5: Siirtyminen ledge-grap-tilaan.

8.2 FSM

Finite State machine on tilakone. Tilakonemallia käytetään monissa tekniikanalan ohjelmointitehtävissä muun muassa digitaalisessa äänentuotossa. Tässä tapauksessa tilakone ohjaa pelaaja-hahmon animaatiotiloja ja sitä, mitä kyseisessä tilassa tapahtuu.

Tilakoneen ”aivoina” toimii globaali tila, joka osaa ohjata tilakoneen johonkin toiseen tilaan.”Aivot” toimivat kuitenkin vain keskusohjaimena, joka osaa sanoa, että haluan seuraavan tilan päälle. Tila joka on päällä, määrää kuitenkin sen, saako tilakone siirtyä seuraavaan tilaan vai ei.

Varsinaisen tilakoneen tarkoitus on päivittää annettua tilaa, sekä vaihtaa tilat aina niin pyydettäessä. Listaus 5. Esimerkkinä tilanvaihtokoodi.

```
public void RequestState( FSMState<T> wantedState )
{
    bool bCanDoTransition  = false;
     foreach( String type in PossibleTransitions )
     {
      if( type.CompareTo(wantedState.GetType().FullName) == 0 )
         {
           bCanDoTransition = true;
         }
     }

    if( bCanDoTransition )
    {
      if(currentState != null )
      {
        if( currentState.GetType() == wantedState.GetType() )
            return;
        currentState.Exit( Owner );
        currentState.SetActive(false,this);
        ClearTransitions();
       prevState = currentState;
      }
      currentState = wantedState;
      currentState.Enter( Owner );
      currentState.SetActive(true,this);
    }
}
```

Listaus 5: Tilakoneen tilanvaihto

Kyseinen tilakoneluokka on rakennettu geneeriseksi, eli tässä tapauksessa geneerinen tietotyyppi on tilakoneen omistaja, joka syötetään attribuuttina jokaiselle tilakoneen tilojen funktioille.

Jokaisen toteuttavan tilan täytyy toteuttaa FSMState-luokka, joka tarjoaa liityntärajapinnan tilakoneeseen (Listaus 6).

```
public abstract class FSMState<T>
{
        private bool m_bIsActivated = false;

        public abstract void Enter( T entity );
        public abstract void Excecute( T entity );
        public abstract void ExcecuteLate( T entity );
        public virtual void FixedUpdate( T entity ){}
        public abstract void Exit( T entity );
        public abstract void MessageFunction( T go , Message animEvent );
        public bool IsActive(){ return m_bIsActivated; }
        public void SetActive( bool bBOOl, FSMStateMachine<T> macine )
        {
                if( macine != null )
                        m_bIsActivated = bBOOl;
        }

}
```

listaus 6: pohjaluokka jokaiselle toteuttavalle tilalle.

Kun FSMState-luokka peritään, voidaan toteuttaa esimerkiksi pelaajahahmon Idle-tila jossa toistetaan idle-animaatiota (Listaus 7).

```
public override void Enter( CharacterEntity entity )
{
 entity.GetStateMachine().AddTransition( typeof(CharacterState_Move).FullName );
 entity.GetStateMachine().AddTransition( typeof(CharacterState_Jump).FullName );
 entity.GetStateMachine().AddTransition( typeof(CharacterState_Drag).FullName );
 entity.GetAnimController().SetState(CharacterAnimationState.CHARACTER_IDLE);
 Debug.Log("Idle");
}
```

Listaus 7: pelaajahahmo idle tilan toiminnot, eli tämä tila saa siirtyä kolmeen eri tilaan ja asettaa idle animaation.

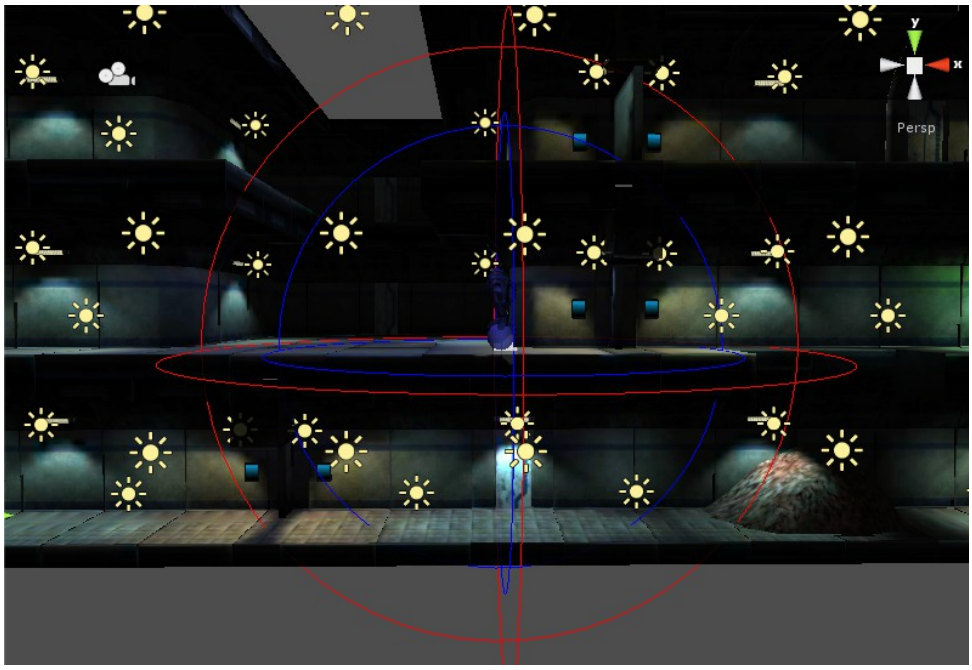
8.3 Tekoäly

Pelin vastustajat olivat luonteeltaan yksinkertaisia. Puolustustykki sekä perinteinen vartija-tyyppinen hahmo, jota kutsuttiin nimellä Sentinel. Molemmille oli kuitenkin rakennettava omanlaisensa tekoäly.

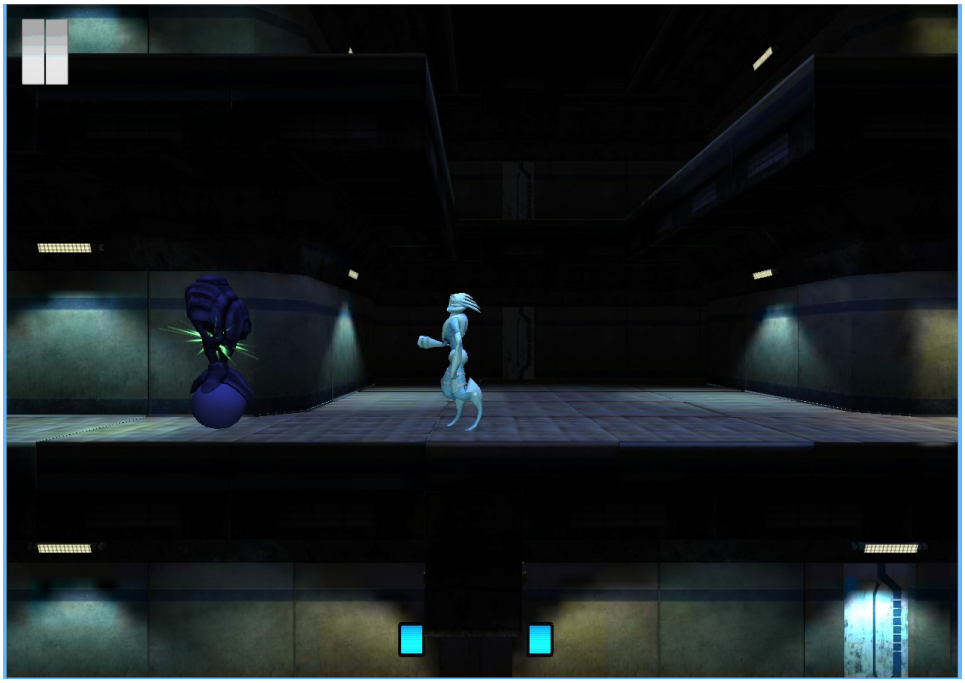
Puolustustykin tekoäly oli yksinkertainen. Tykistä lähtee lasersäde (Kuva 6) pelaaja-hahmon törmätessä tähän säteeseen, jolloin tykki ampuu niin kauan kunnes pelaaja ei ole enää näkökentässä, jonka jälkeen tykki jatkaa alueen tutkimista. Puolustustykki pyörii oman paikallisen x-akselin ympäri ylös ja alas, kunnes huomaa pelaaja-hahmon.

Kuva 6: puolustus tykki on huomannut pelaajan ja aloittaa ampumisen.

Sentinel-hahmon ohjelmoiminen vastasi pitkälti pelaaja-hahmon ohjelmointia. Sille oli rakennettava FSM-tilat. Sentinel-hahmolla on näkökenttä, joka on rajattu kahteen erilliseen osaan. Näkökentän kaukaisin osa, siniset renkaat (kuva 7) tarkoittavat näkökentän loppumista. Jos pelaaja päätyy sinisen ja punaisen alueen väliin, huomaa tekoäly tässä tilanteessa jotakin tapahtuvan ja lähtee kyseistä pistettä kohti. Jos taas pelaaja päätyy punaisten renkaiden sisään, aloittaa tekoäly taistelumoodin (kuva 8).



Kuva 7: editorissa näkyvät hahmon näkökenttä rajat, joita voidaan reaaliaikaisesti säätää. Tämä helpottaa suunnittelua.



Kuva 8: Sentinel on siirtynyt taistelumoodiin, ja lataa asettaan.

8.4 Kontrollit

Mobiili-laitteiden luonteesta johtuen, oli todella hankalaa luoda painikejärjestelmä joka tuntui sopivalta tähän projektiin. Monista iteraatioista päädyimme kuitenkin pyyhkäisytekniikkaan johon yhdistimme näkymättömiä virtuaalipainikkeita.

Hahmon liikkuminen on toteutettu niin, että kun käyttäjä pyyhkäisee näytöllä oikealle tai vasemmalle, riippuen pyyhkäisyn nopeudesta ja pituudesta, aloittaa hahmo kävelemisen tai juoksemisen. Ylös pyyhkäisystä pelaaja-hahmo hyppää ja alas pyyhkäisy pysäyttää hahmon.

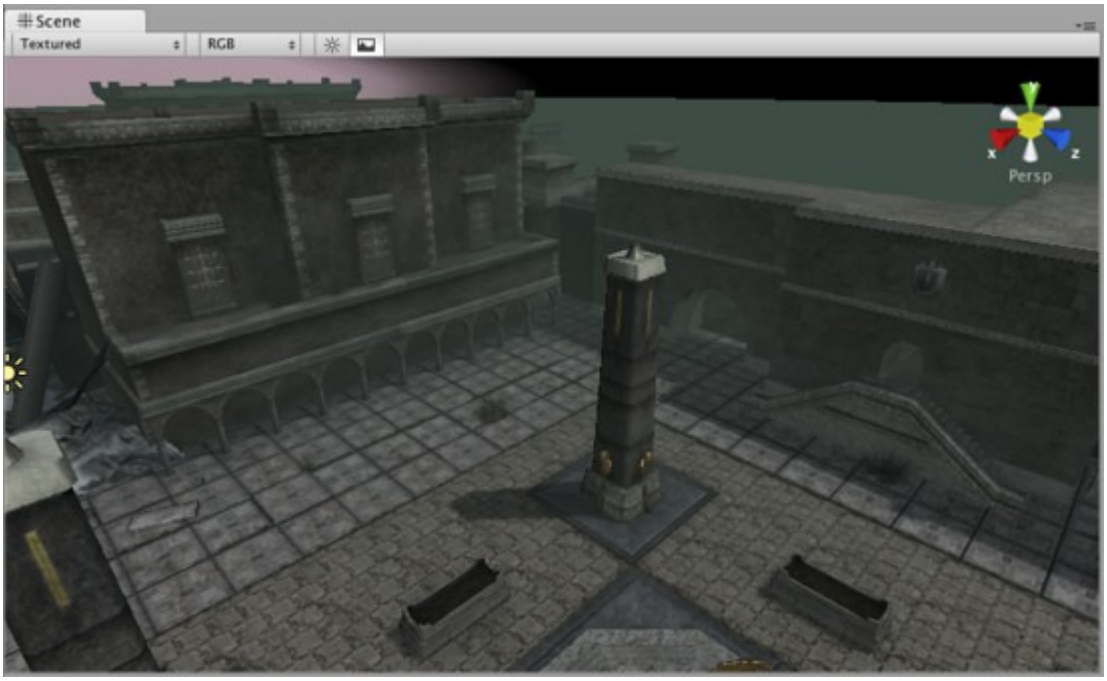
Hahmon ampuminen on toteutettu niin, että käyttäjä asettaa kaksi sormea näytölle, jolloin näyttö jaetaan kahteen osaan. Tällöin ruudun oikea puoli liikuttaa hahmon tähtäystä ja vasen puoli liikuttaa pelaaja-hahmoa.

8.5 GUI-editori

Unity tarjoaa omaa GUI-järjestelmää, mutta sen rakentamiseen ei ole ohjelmaa ja sen suorituskyky on todella huono. Päätin rakentaa oman GUI-rakenteen ja GUI-editorin, jota voisin jatkossa käyttää muihin projekteihini.

GUI-järjestelmä oli suunniteltava tukemaan sekä mobiililaitteita ja PC- laitteita. Editorin oli oltava helppokäyttöinen ja reaaliaikainen, jotta suunnittelija näkee samantien oman kädenjälkensä, eli WYSIWYG editori. Laajennettavuus eli skriptien lisäämisen oli myös oltava yksinkertaista.

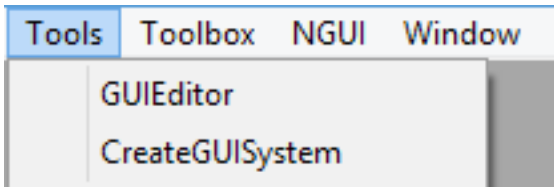
Normaalisti GUI-objektit piirretään 2D-rajapinnan kautta tietokoneen ruudulle, mutta tässä tapauksessa päätin käyttää 3D-maailman objekteja, Mesh tietotyyppiä. Mesh on yksinkertaisin 3D-maailman objekti, sillä se piirretään kahdesta kolmiosta, joista muodostuu neliö. GUI-Mesh voidaan piirtää kahdella tavalla, joko perspektiivissä tai ortografisesti. Normaalisti kaikki GUI-elementit piirretään ortografisesti, koska tässä kameratyypissä (kuva 9) ei ole syvyysvaikutelmaa. Tätä kameraa käyttäessä GUI-elementit voidaan renderöidä niin, että ne vaikuttavat olevan 2D-elementtejä. Kun taas perspektiivi kamerassa (kuva 8) on syvyysvaikutelma, eli elementti pienentyy, mitä kauemmas se siirtyy kamerasta.



Kuva 8: Perspektiivinen kamera(3).

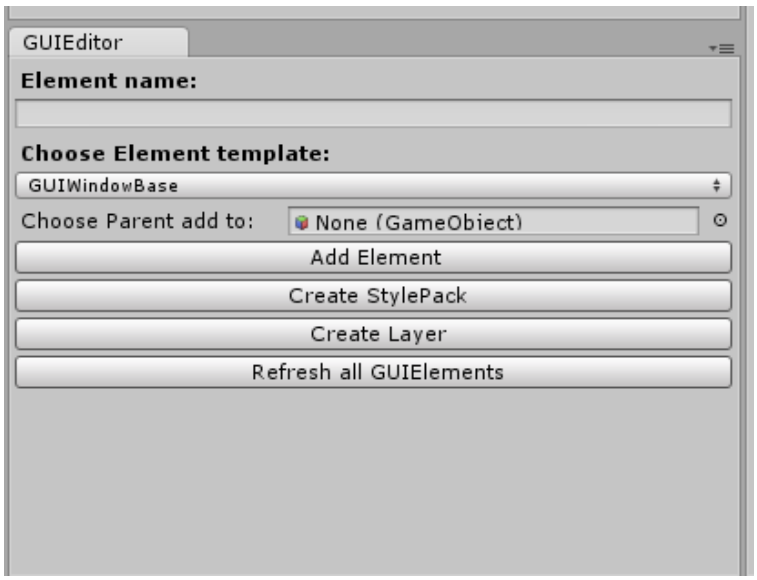
Kuva 9: Sama kohde orthograafisella kameralla(3).

GUI-editori on upotettu Unity-editoriin, sillä unity on helposti laajennettavissa. Editoriin kuuluu toolbar painikkeet (kuva10).



Kuva 10: unity editorin ylälaidassa olevat painikkeet.

GUIEditor - painike avaa toolbox painikeryhmän josta voidaan lisätä nappuloita, ikkunoita ja muita elementtejä (Kuva 11).



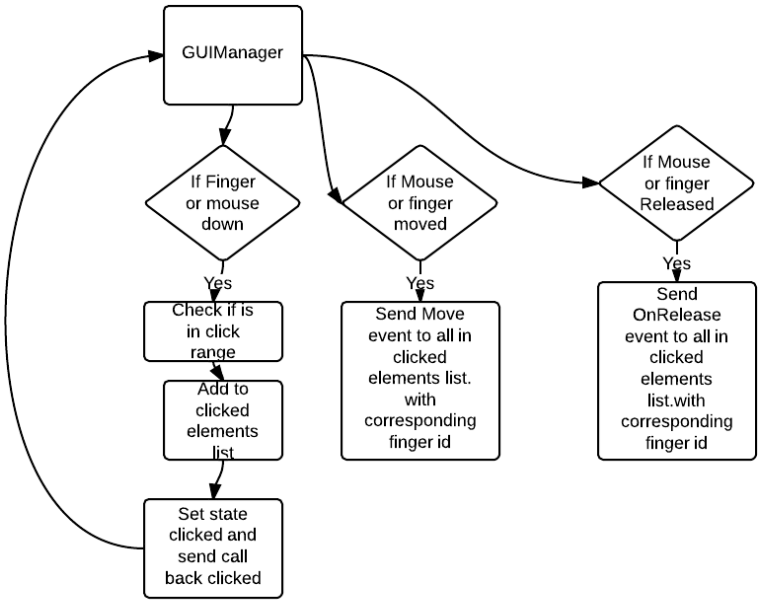
Kuva 11: Varsinainen elementti manageri. Eli toolbox.

Toolbox avautuu omaan ikkunaansa, joka voidaan siirtää, liittää tai upottaa mihin tahansa unity-editorin sisällä. Toolbox-ikkunassa annetaan elementille vapaavalintainen nimi ja valitaan valmiiksi räätälöity luokkamalli. Tämän jälkeen valitaan isäntä-objekti. Vaihtoehtoisesti editorissa voidaan myöd luoda layer-objekti ja tyylipaketteja.

Layer-objektit tehtävä on pitää huolta siitä, että objektien skaalaus pysyy käyttäjän antamissa mitoissa. Käyttäjän painaessa Add Element painiketta GUI-editori luo tarvittaessa skripti-tiedoston annetulla nimellä. Tämä tapahtuu vain ja ainoastaan luotaessa ikkuna-elementtiä, koska varsinainen ikkuna - ja nappula toiminnot kirjoitetaan luokkaan, joka peritään GUIWindowBase -luokasta.

GUI-järjestelmä vaatii oman kameransa, joka voidaan luoda tools valikosta, jossa on alivalintana CreateGUISystem (Kuva 10). Kyseinen painike luo automaattisesti yhden

kameran ja antaa kameralle GUIManager-skriptin, jonka tarkoitus on valvoa kaikkia ikkunoita ja elementtejä. Sekä sitä onko niitä painettu, hiirtä tai sormea liikutettu ja lähettää callback-eventtejä oikeille elementeille. (Kuva 12) esittää karkeasti GUI-Managerin toimintaa ajatustasolla.

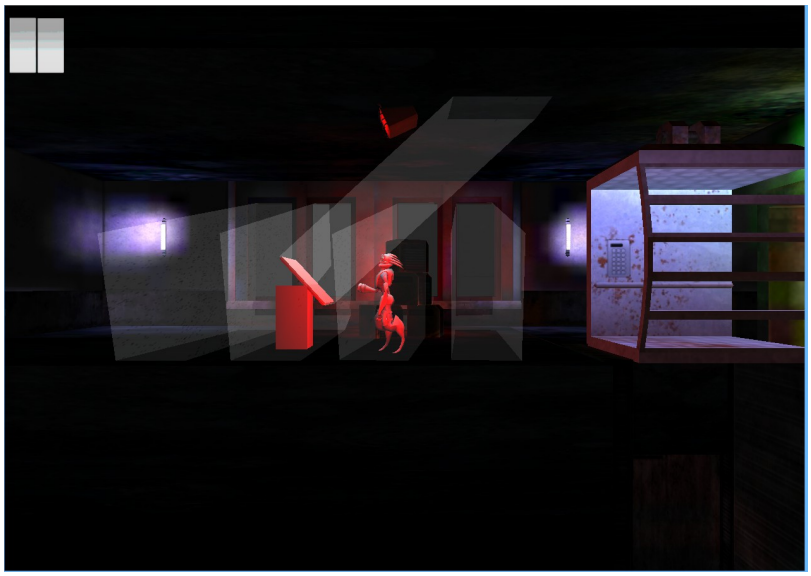


Kuva 12: GUIManager teoreettinen toiminta.

Kaikki GUI-elementit perustuvat samaan pohjaluokkaan, jossa on virtuaalisia funktiota ja joita laajentamalla saadaan luotua täysin uusia GUI kokonaisuuksia. Kuten nappula-objekti, jota hyödynnetään pelin puzzle-osuudessa.

8.6 Puzzlet

Peliin pääsi lopulta yksi puzzle (kuva 13), jossa tarkoituksena oli siirtää peilipaneelia pelaajan omalla sormella raahamalla oikealle tai vasemmalle oikean valopisteen väliin. Kun paneeli on oikean valopisteen välissä, aktivointilamppu aktivoi sillan, jossa on hissitoiminto.



Kuva 13: Valopaneeli ja valonsäteet puzzle.

Paneeliosaan on luotu oma Mirrorentity tyyppinen -skripti, joka lisää alustus kohdassa itselleen GUIButton-komponentin.

```
void Start ()
	{
		m_vReflectPosition = transform.position;
		m_vReflectDir = transform.forward;


		GameObject m = new GameObject( gameObject.name + "_Button");
		m_Button = m.AddComponent<GUIButton>();
		m_Button.elementSize = new Vector3(0.05f,0.2f,1);
		m_Button.OnClickEvent += StartMoving;
		m_Button.OnReleaseEvent += StopMoving;
		mirrorLightObject.active = false;
		//m_Button.transform.parent = m_Button.uiCamera.transform;
	}
```

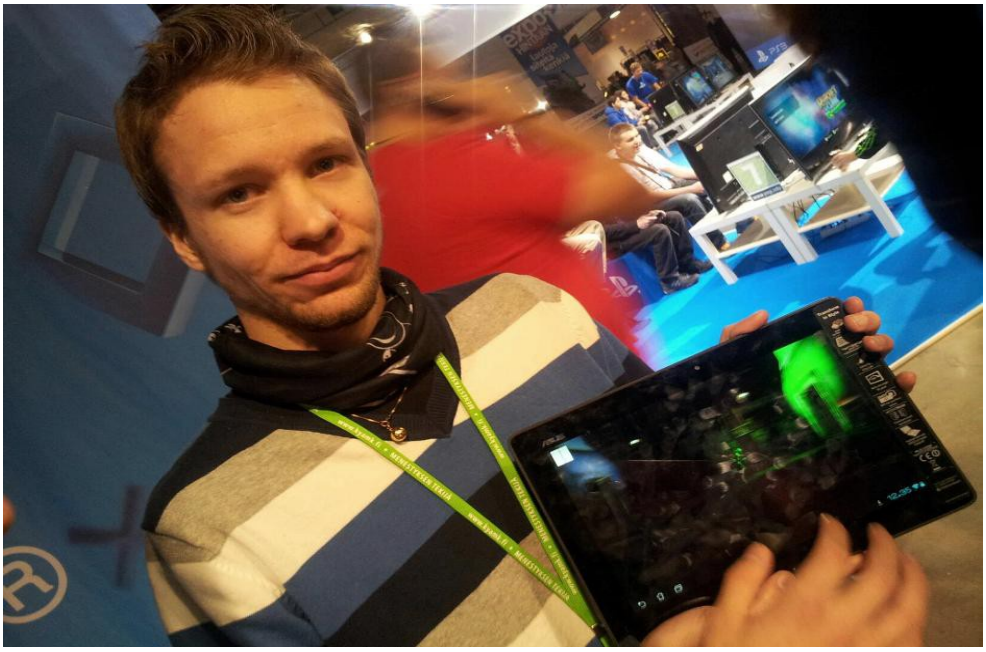
Listaus 8 : Mirror entityn alustus, jossa annettaan nappula ominaisuus.

9 DIGI-EXPO

Loppuvuodesta 2012 pääsimme osallistumaan DigiExpo messuille Kyamkin esittelypisteen yhteyteen, jossa Kymenlaakson ammattikorkeakoulut mainostivat omaa koulutustarjontaansa. Samassa yhteydessä pääsimme esittelemään Kouvolan ja Kotkan koulujen yhteistyöprojektia eli Awakening peliä.

Kymmenet kiinnostuneet henkilöt pysähtyivät esittelypisteelle pelailemaan tuotostamme ja osoittivat mielenkiintoa tuotetta kohtaan. Samalla pääsimme testaamaan, onnistuimmeko kontrollivalinnoissa ja miten peli vaikuttaa pelaajiin. Tästä tilaisuudesta selvisi se, että peliin valitut kontrollit eivät olleet kaikille helposti lähestyttäviä, osa pelaajista ei löytänyt tapoja, kuinka hahmoa liikutetaan ja miten hahmo ampuu. 10–15 vuotiaat pojat löysivät kuitenkin kontrollit hyvinkin nopeasti. He juuri ovatkin kohderyhmämme, joten voisin kutsua kontrollivalintaa onnistuneeksi.

Yksi ryhmämme jäsenistä pääsi YleX-haastateltavaksi (kuva 14), jonka voi käydä lukemassa YleX sivuilta (4). Digi-expo oli äärimmäisen hyvä tapahtuma luoda yhteyksiä muihin pelialan yrityksiin ja alasta kiinnostuneisiin ihmisiin.



Kuva 14: Kuva lainattu YleX sivuilta(4). Kuvassa Lauri Kuparinen esittelemässä Awakening peliä.

10 LOPPUSANAT

Opinnäytetyön valinta tuli lähestulkoon itsestään selvyytenä, sillä projektin mittakaava ja sen antamat oppimismahdollisuudet olivat suuret, siitä huolimatta, että olen ollut lähes kaksi vuotta pelialan yrityksessä töissä. Projektin aloitusvaiheessa en tiennyt Unity- pelimoottorista juuri mitään. Projektin edetessä Unityn oppiminen helpottui ja kehitin uuden UI–systeemin ja siihen liittyvän editorin. Editorin olisi voinut jopa eritellä omaksi projektikseen.

Omasta mielestäni saimme sen kaiken ruudulle jonka halusimmekin näyttää. Projektin hyvästä etenemisestä huolimatta emme aio jatkokehittää peliä eteenpäin, sillä vastaavia pelejä on maailma pullollaan. Lisäksi tuotteen monetisointi on hankalaa.

Opinnäytetyössä törmättiin moniin erilaisiin haastaviin ongelmiin, mutta päättäväisellä tiimitoiminnalla pääsimme nopeasti hankaloiden tilanteiden ohitse. Suurinta kiitosta annan Kouvolan media-alan opiskelijoille kärsivällisyydestä ja heidän innovatiivisista toimintatavoistaan ja halusta oppia aina jotain uutta. Tästä johtuen projektin läpivieminen ei tuntunut missään vaiheessa työltä.

Suosittelen vahvasti vastaavia projekteja muillekin oppilaille ja koulujen väliselle yhteistyölle. Kyseinen toiminta kasvattaa opiskelijoita parempaan toimintaan ja opettaa löytämään vastauksia ongelmiin, sekä antaa mahdollisuuden ottaa uusia työhaasteita erilaisissa ympäristöissä.

11 LÄHTEET

1. Subversion on centos. Centos käyttöjärjestelmän wiki-sivusto. Saatavissa: http://wiki.centos.org/HowTos/Subversion [viitattu: 6.2.2013]

2. Redmine on CentOS installation HOWTO. Redmine-ohjelmiston wikisivusto. Saatavissa: http://www.redmine.org/projects/redmine/wiki/HowTo_install_Redmine_on_CentOS_5 [viitattu: 6.2.2013]

3. Unity manual rendering components camera. Unity manuali. Saatavissa: http://docs.unity3d.com/Documentation/Components/class-Camera.html[viitattu:7.3.2013]

4. Suomalaiset oppilaitokset heräävät kasvavan pelialan tarpeisiin,YleX.fi 2012. julkaistu 3.11.2012. Uutinen. Saatavissa: http://ylex.yle.fi/uutiset/popuutiset/suomalaiset-oppilaitokset-heraavat-kasvavan-pelialan-tarpeisiin-hautuuko-taalla-s. [Viitattu :10.3.2013]

5. Schwaber,K. & Sutherland,J.2012. The Scrum Guide 2012. Lokakuu 2011. Saatavissa: http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum%20Guide%20-%20FI.pdf[Viitattu: 10.3.2013]

6. Projekti-instituutti, Projektijohtamisen sanastoa. Sanasto. saatavissa: http://www.projekti-instituutti.fi/sanasto% [Viitattu 10.3.2013]