JELLY WARS

Cross-Platform Multiplayer Online Mobile Game

Sami Kauppinen

Opinnäytetyö

Toukokuu 2013

Ohjelmistotekniikka

Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä(t)
KAUPPINEN, Sami

Julkaisun laji
Opinnäytetyö

Päivämäärä
10.5.2013

Sivumäärä
22

Julkaisun kieli
Suomi

Verkkojulkaisulupa myönnetty
(X)

Työn nimi
Jelly Wars - Cross-Platform Multiplayer Online Mobile Game

Koulutusohjelma
Ohjelmistotekniikan koulutusohjelma

Työn ohjaaja(t)
LAPPALAINEN-KAJAN, Tarja

Toimeksiantaja(t)
Star Arcade OY

Tiivistelmä

Projektin tarkoituksena oli kehittää alustariippumaton mobiilimoninpeli jyväskyläläiselle pelitalolle Star Arcadelle.

Pelissä pelaaja kontrolloi oman heimonsa jäsentä ja kohtaa kilpailevan heimon edustajan taistelukentällä. Tavoitteena on kukistaa toinen heimo ja saada haltuunsa vuori, jonka huipulla taivaalta valopallona alas tullut aarre odottaa. Vuoropohjainen taistelu motivoi strategiseen kamppailuun, ja fysiikkamallinnus mahdollistaa monipuolisen aseiden ja ympäristön käyttämisen omaksi edukseen. Pelaajilla on myös käytössään ostettavia lisäaseita ja varusteita. Taistelun keskellä pelaajat voivat myös keskustella keskenään kirjoittamalla viestejä toisilleen.

Työ toteutettiin C++-kielellä käyttäen Microsoft Visual Studio 2010 -sovelluskehitintä. Pelin pohjalla toimi Star Arcaden DieselEngine-pelimoottori ja Star Arcade Client -moninpelikehys.

Jelly Wars julkaistiin 23.8.2011, ja uuden version kehitys on meneillään.

Avainsanat (asiasanat)

Jelly Wars, Peliohjelmointi, C++, Mobiilipeli

Muut tiedot

Author(s)
KAUPPINEN, Sami

Type of publication
Bachelor´s Thesis

Päivämäärä
10.5.2013

Pages
22

Language
Finnish

Permission for web publication
(X)

Title
Jelly Wars - Cross-Platform Multiplayer Online Mobile Game

Degree programme
Software Engineering

Tutor(s)
LAPPALAINEN-KAJAN, Tarja

Assigned by
Star Arcade OY

Abstract

The purpose of this Bachelor's thesis was to develop a mobile multiplayer game and it was assigned by a game development company, Star Arcade in Jyväskylä.

In the game players control members of their tribe and face a rival tribe member in the battlefield. The objective is to defeat the opposing tribe and gain control over the mountain that has a treasure fallen from the skies on top of it. Turn-based action motivates players to engage in strategic battles and the physics simulation enables different ways to use weapons and the environment to gain the upper hand. Players can also purchase additional weapons and items to help them survive. During the battle players can also chat with each other by sending written messages.

The game was written with C++ and using Microsoft Visual Studio 2010 integrated development environment. The game itself was built on top of DieselEngine -game engine used by Star Arcade and Star Arcade Client -social multiplayer framework.

Jelly Wars was released on August 23, 2011 and the sequel is under development.

Keywords
Jelly Wars, Game Programming, C++, Mobile game

Miscellaneous

SISÄLTÖ

KUVIOT

TERMIT JA KÄSITTEET

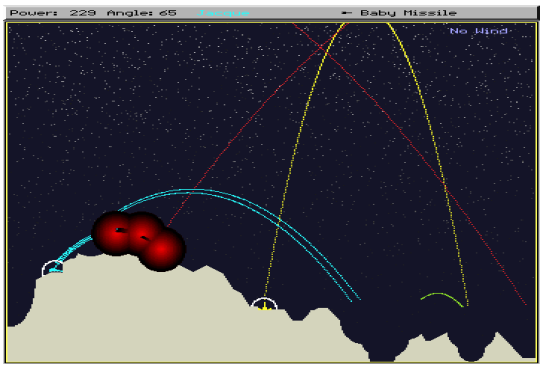
AABB	Axis Aligned Bounding Box eli koordinaatiston akseleiden suuntainen suorakulmio
Angry Birds	Rovion julkaisema suosittu fysiikkapohjainen peli
Botti	Tietokoneen kontrolloima pelaaja
Bugi	Ohjelman sisältämä ongelma
CPU	Central Processing Unit, laitteen suoritin, joka suorittaa laskutoimitukset ja muistiosoitukset
DirectX	Microsoftin grafiikanpiirtorajapinta
Fixed pipeline	Grafiikan piirtoon tarkoitettu yksinkertaistettu käskykanta
GPU	Graphics Processing Unit, näytönohjaimen oma grafiikan piirtoon tarkoitettu suoritin
IDE	Integrated Development Environment, käyttöliittymä koodin kirjoittamiseen, kääntämiseen ja hallintaan
Natiivikoodi	Laitteen ymmärtämä koodimuoto
OpenGL	Open Graphics Library, Khronos Groupin kehittämä laitteistoriippumaton grafiikanpiirto rajapinta
Programmable pipeline	Grafiikan piirron ohjelmoitava käskykanta

Shader	Ohjelmoitavan käskykannan täysin muokattavissa oleva piirron osa
Pikseli	Yksi piste ruudulla
Virtuaalikone	Ohjelma ns. näennäiskone, jonka sisällä voidaan ajaa muita ohjelmia
Worms	Vuoropohjainen Team17-yhtiön luoma toiminta-strategiapeli vuodelta 1994

1 LÄHTÖKOHDAT

1.1 Tavoite

Projektin alussa saatiin tehtäväksi tehdä Scorched Earth -tyylinen taktinen kaksintaistelupeli ja päivittää se helposti lähestyttävämpään modernimpaan muotoon. Pelaajat, jotka pelasivat alkuperäistä peliä, nauttivat sen tarjoamasta taitoa vaativasta pelimekaniikasta, jossa yhdistyivät tarkasti ammutut panokset ja oikeiden aseiden käyttö. Peli oli hyvin karkealla grafiikalla varustettu kankea tykistöammuskelu. Karut maastot, yksinkertainen grafiikka, kankea käyttöliittymä ja hyvin ankea aihepiiri tekivät alkuperäisestä pelistä pienen pelaajakunnan pelin (ks. kuvio 1).



Kuvio 1. Alkuperäinen peli

Tästä tunnelmasta haluttiin nopeasti eroon, jotta kuka tahansa, joka näkee pelistä kuvankaappauksen tai pelivideon, näkee siinä saman tien jotain kiinnostavaa ja miellyttävää. Myös käyttöliittymä oli suunniteltava kosketusnäyttöjä ajatellen ja helppokäyttöisyys olikin yksi suunnittelua ohjaavia asioita. Nämä asiat mielessä alettiin suunnitella uutta peliä.

1.2 Star Arcade Oy

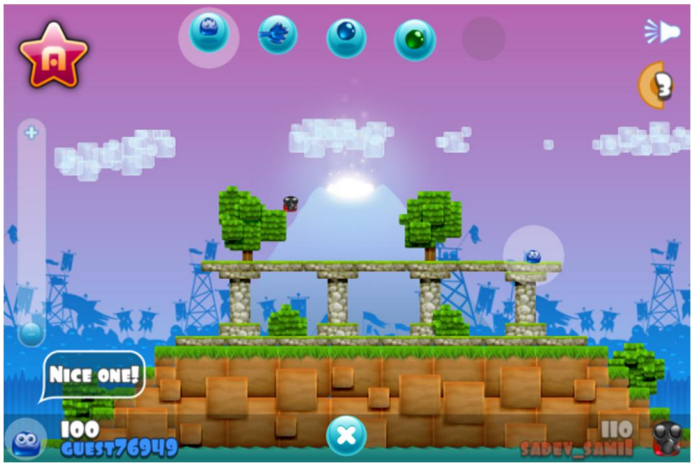
Star Arcade Oy kehittää mobiilimoninpelejä eri alustoille kuten Android, iOS, Bada,Meego/Maemo, Symbian ja WP7 (ks. kuvio 2). Yritys perustettiin vuonna 2010, ja sen tavoite on alusta lähtien ollut luoda pelejä, joita voi pelata muiden pelaajien kans-

sa riippumatta laitteiden käyttöjärjestelmistä ja valmistajista. Käyttäjiä on jo yli 180 maasta, ja tuettuja kieliä noin 20. Tällä hetkellä yrityksessä on noin 25 työntekijää, joista noin puolet on ohjelmoijia. Aiemmin julkaistut pelit olivat ilmaisia mainosten kanssa, ja halutessaan käyttäjät pystyivät ostamaan täyden version ilman mainoksia. Tulevat pelit julkaistaan ilmaisina eivätkä ne sisällä mainoksia, ja tuotto on tarkoitus saada mikromaksuilla pelien sisältä. (Company Presentation 2012.)

Kuvio 2. Star Arcaden logo

1.3 Jelly Wars maailma

Jelly Wars on fysiikkapohjainen kaksinpeli, jossa kaksi eri Jelly-heimon jäsentä kamppailee toisiaan vastaan. Maailmaa asuttavat heimot ovat olleet riidoissa keskenään sen jälkeen, kun kauan aikaa sitten taivaalta tullut valopallo törmäsi läheisen vuoren huipulle. Kumpikaan heimoista ei antanut toisen heimon viedä aarretta, minkä maaginen valopallo vuoren huipulla paljasti. Pelaaja viedään taistelemaan tätä taistelua ja ehkä jopa viimein ratkaisemaan ikiaikainen sota (ks. kuvio 3).



Kuvio 3. Pelitila

Pelimekaniikka on sekoitus Worms-pelisarjan taktista asemasotaa ja Angry Birdsistä tuttua sormen avulla ammuskelua. Pelaajalla on käytössään erilaisia aseita ja mahdollisuus liikkua pitkin taistelukenttää. Voittaja ratkeaa pudottamalla vastustaja veteen tai viemällä tämän energia aseilla. Ottelun voittaa pelaaja, joka osaa liikkua parempaan paikkaan ja ampua tarkasti.

2 ALUSTARIPPUMATON PELINKEHITYS

2.1 Kehitystavat

Jos tarkoitus on tehdä peli ja julkaista se eri alustoille, on päätettävä millä tavalla tätä lähdetään tekemään. Vaihtoehtoja on muutama: alustakohtainen kehitys, tulkattavan kielen käyttö ja alustariippumattoman ohjelmistokehyksen luonti.

Alustakohtaiset versiot

Alustakohtainen kehitys tarkoittaa sitä, että ohjelmointikieli, rajapinnat ja toiminnallisuus voivat olla täysin erilaisia eri alustojen välillä, esim. PC:n DirectX-pelistä tehdyn Android-version on pakko käyttää OpenGL rajapintaa, koska Android ei tue DirectX:ää. Toisaalta Android-versio voi käyttää kallistuksen tunnistusta yms., mitä Windows-versio ei voi. Eri versiot voivat olla myös käyttäjälle täydellisiä kopioita toisistaan, vaikka niiden sisäinen rakenne olisikin aivan erilainen.

Hyvät puolet:
Peli on mahdollista optimoida joka alustalle täydellisesti.

Huonot puolet:
Pelimekaniikan muutos tarkoittaa muutosta jokaiseen versioon. Jokaisella versiolla on omat ohjelmistoversiot, bugit, testaus- ja päivitysjärjestelmät.

Tulkattava kieli

Tulkattavalla kielellä kirjoitettu ohjelma luottaa alustakohtaisen virtuaalikoneen tulkitseman koodin toimivuuteen ja lopputuloksen yhteneväisyyteen. Esimerkiksi Javalla tehty ohjelma toimii monessa laitteessa sellaisenaan, koska laitteen oma tulkki kääntää alustakohtaisen koodin toimivaksi.

Hyvät puolet:
Yksi koodipolku.

Huonot puolet:
Tulkattavan kielen mahdollinen hitaus. Käytettyjen kirjastojen mahdolliset puutteet ja rajoitteet.

Alustariippumaton ohjelmakehys

Alustariippumattoman ohjelmakehyksen käyttö yhdistää tehokkaan koodin suorituksen ja mahdollisuuden käyttää yhtä koodipolkua kaikkien eri versioiden pohjana antaen silti vapaat kädet alustakohtaisille variaatioille. Itse ohjelma käyttää yhtä kieltä, esim. C++:aa, ja se rakennetaan kehyksen päälle käyttäen sen tarjoamia rajapintoja. Ohjelmakehys hallitsee alustakohtaiset toteutukset, tarjoaa yleiset rajapinnat sovellukselle ja tarvittaessa myös alustakohtaiset rajapinnat.

Hyvät puolet:
Käännettävä versio on natiivikoodia eli nopeaa. Yksi koodipolku(itse ohjelmalla) ja yhtenäiset rajapinnat. Täysi kontrolli ohjelman suoritukseen ja koodin sisältöön.

Huonot puolet:
Alustakohtaiset toiminnot on kirjoitettava itse.

2.2 Työkalut

Työpisteellä oli 2 monitoria sekä PC, jossa käyttöjärjestelmänä oli Windows 7. Kehitysympäristönä käytössä oli Visual Studio 2010 ja ohjelmointikielenä C++. Bugzillaa käytettiin bugien hallintaan ja Basecamp-sivustoa työtuntien kirjaamiseen. Testiresurssien luontiin ja käsittelyyn käytettiin ilmaisia Audacity- ja Paint .NET-ohjelmia.

Itse pelin ohjelmointi tapahtuu käyttäen kahta rajapintaa. Ensimmäinen on Diesel Engine, jonka avulla voidaan piirtää ruutuun grafiikkaa, käsitellä käyttäjän toimintoja, soittaa ääniä, hallita alustakohtaisia tapahtumia sekä paljon muuta. Toinen on Star Arcade Client, joka tarjoaa kehyksen mm. verkkopelien hallintaan, vastustajien etsimisen ja virtuaaliostojen teon.

Projektin alkaessa käytössä oli CPU-pohjainen piirto, joka tarkoittaa sitä, että jokainen piste piirretään käyttäen laitteen prosessoria. Kaikki pelissä laskettavat asiat käyttävät jo muutenkin prosessoria, joten sen käyttäminen myös piirtoon hidasti peliä vakavasti. Jonkin ajan kuluttua Diesel Engineen tuli OpenGL ES 2.0-piirtopinta, joka vapautti prosessorin piirrosta, valjasti käyttöön GPU:n ja paransi suorituskykyä huomattavasti.

2.3 OpenGL ES 2.0

OpenGL ES 2.0 on sulautettuihin järjestelmiin suunniteltu version OpenGL-rajapinnasta, jota tukevat käytännössä kaikki älypuhelimiksi kategorioidut puhelimet ja lukemattomat muut nykylaitteet. Tässä versiossa poistuivat vanhat (fixed pipeline) käskyt ja tilalle tulivat uudet (programmable pipeline) kutsut, jotka käyttävät ohjelmoitavia piirtosuorituksia (shader). Tämä antaa ohjelmoijalle täyden vapauden käyttää grafiikkakiihdytystä haluamallaan tavalla ja suorittaa käskytystä mahdollisimman tehokkaasti.

2.4 Diesel Engine

Diesel Engine on alustariippumaton ohjelmistokehys. Sen rajapintoja käyttävä peli voidaan kääntää kaikille tuetuille alustoille natiivikoodiksi, jolloin siitä saadaan mahdollisimman nopeasti toimiva ja samalla lailla toimiva joka alustalla. Diesel Engine tarjoaa ohjelmalle rajapinnat kaikkeen, mitä ohjelmat tarvitsevat, ja alustat mahdollistavat mm. grafiikan piirron, äänentoiston, musiikin soiton, ajastimet, käyttäjän tekemien liikkeiden ja kosketusten hallinnan. Näiden lisäksi se pitää sisällään myös merkkijonot, kokoelmat (jonot, kartat yms.) ja lukemattoman määrän muita käytännöllisiä komponentteja joita jokainen ohjelma tarvitsee.

Peli itse kirjoitetaan C++-kielellä, mutta käyttöjärjestelmä ja IDE on vapaasti valittavissa. Milloin tahansa koodista voi tehdä laiteversion halutulle alustalle ja testata sitä laitteessa. Tämä nopeuttaa laitekohtaisten ongelmien etsimistä ja suorituskykytestausta.

2.5 Star Arcade Client

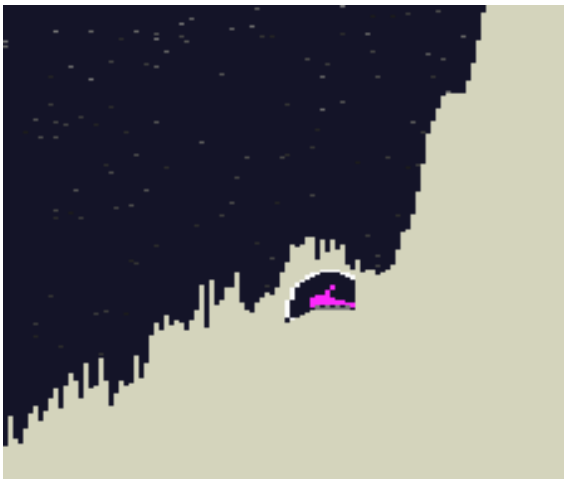
Star Arcade Client on Diesel Enginen päälle rakennettu sosiaalinen moninpelikerros. Sen avulla peliin voi lisätä sosiaalisen verkkokerroksen, joka mahdollistaa mm. pelaamisen muiden käyttäjien kanssa, käyttäjäkohtaiset asetukset ja tilastot, keskustelun vastustajien kanssa ja virtuaaliostosten hallinnan. Tämä kerros pitää huolta kirjautumisesta, rekisteröinnistä ja kertoo pelille tapahtumista ja mahdollisista virheistä.

3 SUUNNITTELU

Alkuperäinen peli ei olisi sellaisenaan kovin miellyttävä nykypelaajan makuun ja käyttöliittymä ei soveltuisi kosketusnäytöille. Tästä syystä yritettiin keksiä uusia ideoita pelimaailman rakenteeseen, miten tehdä tästä kaikille pelaajille helposti lähestyttävä ja kuinka pelimekaniikka saadaan ensikertalaisellekin helppokäyttöiseksi.

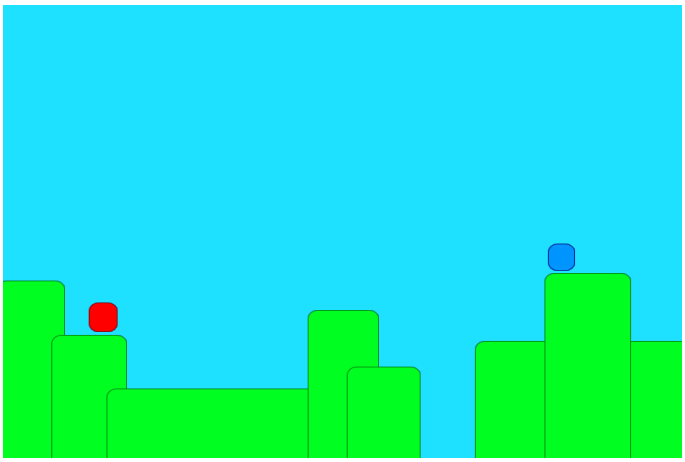
3.1 Taistelukenttä

Ensimmäiseksi keskityttiin itse taistelukenttään eli maastoon. Scorched Earth -pelissä oli käytössä pikselipohjainen maasto, josta tuli muutaman ammuksen jälkeen rosoinen, mutta koska pelissä ei juurikaan liikuttu, niin se ei myöskään haitannut (ks. kuvio 4).



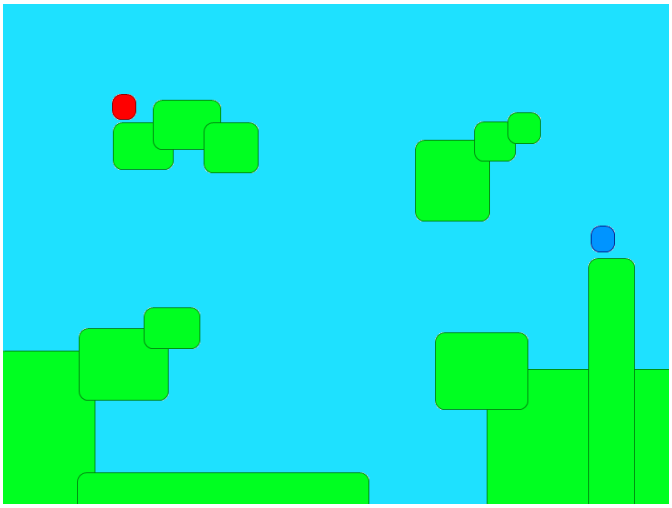
Kuvio 4. Epätasainen maasto alkuperäisessä pelissä

Peliin haluttiin lisää liikkuvuutta ja strategiamahdollisuuksia, joten maaston oli mahdollistettava se. Alettiin miettiä erilaisia vaihtoehtoja kuten käsin piirrettyä tai palkeista koostuvaa maailmaa. Aluksi päädyttiin pystypalkeista koostuvaan maastoon (ks. kuvio 5).



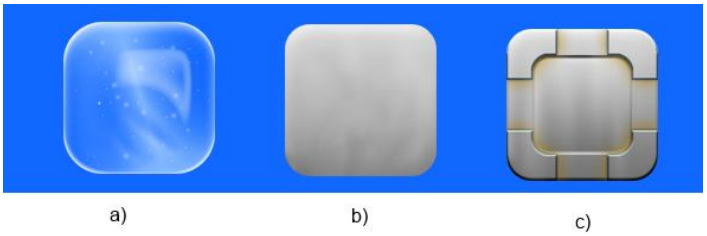
Kuvio 5. Pystypalkeista koostuva maasto

Tässä testiversiossa maasto generoitiin satunnaisesti ja pelaajien alkupaikat arvottiin rajatuin parametrein (kuinka kaukana reunoista ja toisistaan pelaajat olisivat kentällä). Hyvä puoli tässä systeemissä oli se, että kentät oli helppo luoda tyhjästä ja jokainen kenttä oli erilainen. Huono puoli oli se, että kentät eivät olleet tarpeeksi erilaisia. Jokainen kenttä antoi mahdollisuuden puolikaaren muotoiseen heittoon suunnilleen yhtä kauaksi joka kerta, koska pelaajat piti luoda tarpeeksi etäälle toisistaan ja kentän koko on rajattava järkevään pituuteen. Seuraavassa versiossa peliin lisättiin editori, jonka avulla palkkien sijaintia ja kokoa voitiin muokata. Tästä eteenpäin palkit saivat olla missä päin kenttää tahansa ja minkä muotoisia tahansa. Maaston hallintajärjestelmä oli löytynyt (ks. Kuvio 6).



Kuvio 6. Vapaasti muotoiltava maasto

Pelikenttään haluttiin lisätä variaatiota, joten palikat jaettiin eri ryhmiin niiden kestävyyden mukaan (ks. kuvio 7). Tällä tavoin pelaajan tulee ottaa huomioon liikkuessaan myös maaston kestävyys.



Kuvio 7. Palkkien eri ryhmät. a) heikko(jää, pilvi) b) normaali (maa, hiekka) c) kestävä (metalli, rauta)

3.2 Taistelutapa

Kun maaston rakenne oli saatu halutuksi, oli aika keksiä, millä tavalla taistelut käytäisiin ja miten ne vaikuttaisivat maastoon. Alkuperäisessä pelissä valittiin ase, asetettiin piipun kulma asteina ja voima, jolla ammus lähtee piipusta. Näin kankea tapa ei nykypelaajaa houkuta, joten piti keksiä jotain muuta. Angry Birdsistä tuttu "vedä ja vapauta"-tyylinen kontrolli vakiintui välittömästi halutuksi tavaksi. Tarvittiin luonteva tapa tuoda tämän tyyppinen kontrolli peliin. Päädyttiin kätevään ratkaisuun: kaikki tapahtuu vetämällä itse pelaajaa. Jos käytössä on ase, tapahtuu ammunta kyseisellä aseella, ja jos ei ole, liikutaan.

Pelaajilla on myös käytössään erikoisaseita, kuten jääpommi ja rypälepommi. Jääpommi muuttaa räjähtäessään kaiken maaston ympäriltään heikoksi jääksi. Rypälepommista lentää kolme uutta pienempitehoista pommia eri suuntiin sen räjähtäessä. Molemmat näistä erikoispommeista jäävät kiinni ensimmäiseen osumaansa maastopalikkaan heiton jälkeen.

3.3 Maailma

Maasto ja pelikontrollit olivat valmiit, mutta maailma oli tyhjä, koska pelissä oli vain laatikoita ilmassa eikä mitään pelihahmoja tai tunnelmaa oikeasta pelimaailmasta. Alkoi ideointi maailman ulkoasusta, hahmoista ja aseistuksesta. Ensimmäisiä ideoita oli ottaa vanha jäniksen ja kilpikonnan kilpajuoksu ja ratkaista kilpa lopullisesti taistelukentällä. Tämä hylättiin, koska se ei vain tuntunut hyvältä konseptilta.

Ratkaisu tuli lopulta sattumalta Googlen kuvahaun tulosten seasta. Etsittiin satunnaisia kuvia hahmoista ja otuksista, ja vastaan tuli pallero, jolla oli silmät. Kuva otettiin ja pistettiin pelihahmon tilalle. Tämän jälkeen palaset alkoivat loksahdella paikalleen. Jos pelaaja on hyllyvä kuminen hyytelöpallo, hän voi venyttää itseään ja ampua joko itsensä tai ammukset linkoamalla hahmonsa haluttuun suuntaan (ks. kuvio 8).

Kuvio 8. Pelaajan tapa ampua venyttämällä

Lopullinen tarina heimoista ja ikiaikaisesta taistelusta syntyi pikkuhiljaa graafikon mielikuvituksessa. Heimojen välinen sota vaikutti maailman ulkoasuun tehden siitä vähän turnajaisia muistuttavan areenan heimolaisten katsellessa taustalla (ks. kuvio 9).

Kuvio 9. Ensimmäisen julkaisun ulkoasu

4 TOTEUTUS

4.1 Työskentely

Työskentely tapahtui pääosin graafikon kanssa suunnitellen ja itsenäisesti ohjelmoiden. Peli koostuu useammasta eri osasta, joita käyttämällä saadaan toimiva kokonaisuus.

4.2 Fysiikkamoottori

Fysiikkamoottori on yksi pelin tärkeimpiä elementtejä. Se ratkaisee, miltä itse liikkuminen ja taistelu tuntuu ja kuinka luonnolliselta maailma pelaajalle vaikuttaa. Peli tulee rakentumaan tämän osan päälle, eli jos fysiikkamoottori epäonnistuu, silloin epäonnistuu koko peli. Alussa toivottiin että olisi ollut mahdollisuus käyttää jotain valmista fysiikkamoottoria, kuten Box2D:ta, mutta tämä ei ollut projektin alkaessa mahdollista. Kyseiseen moottori oli kuitenkin entuudestaan tuttu ja päätettiin kirjoittaa yksinkertaisempi versio sen rakenteen pohjalta. Kirjoitettiin alusta alkaen moottori, joka perustuu AABB -matematiikkaan, mikä on huomattava ero alkuperäisen moottorin rakenteeseen. (Box2D User Manual 2011, 32-35.)

Kaikki pelimaailman objektit ovat suorakulmioita, joilla on paino, nopeus, kitka ja kimmoisuus (ks. kuvio 10). Suorakulmioiden paras puoli on matematiikan yksinkertaisuus, joka sallii suuren osan laskuista suoritettavan vertailuilla ja yksinkertaisilla laskutoimituksilla.

```
class BodyDesc
{
public:
    BodyDesc()
    {
        Position.Set(0,0);
        Size.Set(0,0);
        BodyType    = BODY_STATIC;
        Radius      = 0.0f;
        Mass        = 0.0f;
        Restitution = 0.5f;
        Friction    = 0.5f;
    }
    CDieselVector2  Position;
    CDieselVector2  Size;
    BODYTYPE        BodyType;
    DE_FLOAT32      Radius;
    DE_FLOAT32      Mass;
    DE_FLOAT32      Restitution;
    DE_FLOAT32      Friction;
};
```

Kuvio 10. Fysikaalisen objektin rakenne

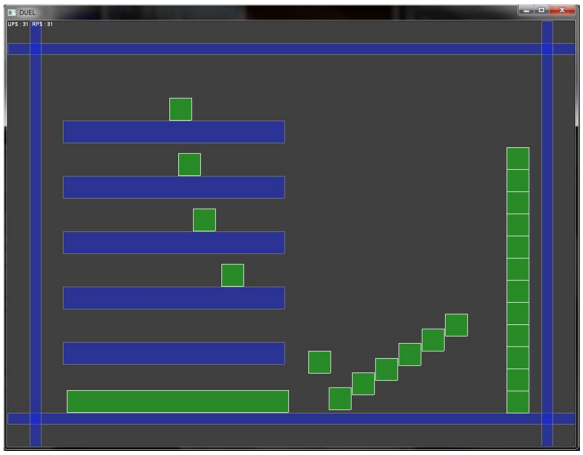
Suurin osa pelin tapahtumista johtuu siitä, että fysiikkamoottori on havainnut törmäyksen tai vastaavasti joku maailman objekteista on pysähtynyt. Nämä tapahtumat ovat elintärkeitä ja näiden tarkkailuun käytetäänkin tarkkailija-mallia (Gamma, Heim, Johnson & Vlissides 2001, 293-303). Pelimaailma asettaa itsensä tarkkailijaksi, jolloin se saa tarvittavan tiedon fysiikkamoottorin laskemista tapahtumista (ks. kuvio 11).

```
struct CollideData
{
    CBody*          pBody[2];
    CDieselVector2  vCollisionPosition;
    CDieselVector2  vCollisionNormal;
    DE_FLOAT32      fCollisionMagnitude;
};

class CSamPhysicsObserver
{
public:
    virtual void BodyCollide(CollideData pData) = 0;
};
```

Kuvio 11. Törmäyksentarkkailija rajapinta

Mutta kuten minkä tahansa fysiikkamoottorin sen on oltava luotettava ja käsiteltävä lepotilat, pinoaminen, törmäysten hallinta yms. Tämän kaiken tekemiseen meni paljon aikaa, vaivaa ja monet hermojen menetykset, mutta lopulta siitä tuli todella tehokas pieni fysiikkamoottori (ks. kuvio 12).



Kuvio 12. Fysiikkamoottori. Testissä kitkat, kimmoisuudet ja pinoaminen

4.3 Pelin kulku

Pelaajalle pelikokemus alkaa joko painamalla Play-nappia, jolloin peli etsii vastustajan automaattisesti tai sitten haastamalla vastustaja pelaajalistasta. Kun vastustaja on löydetty, peli siirtyy taistelukentälle, jossa aloittajaksi arvottu pelaaja tekee oman siirtonsa. Tämän jälkeen vuoro vaihtuu, ja tätä jatketaan, kunnes vain toinen pelaaja on hengissä tai molemmat ovat kuolleet, jolloin tulee tasapeli. Tasapelin tapauksessa uusi taistelukenttä arvotaan ja ottelu alkaa alusta.

4.4 Pelin rakenne

Koko pelitila on rakennettu tapahtumien (CSamEvent) avulla. Jokainen pelin tapahtuma lisätään tapahtumalistaan, josta tapahtumia käsitellään ja lähetetään eteenpäin tai luodaan uusia (McShaffry & Graham 2012, 307-326) . Tapahtuma voi olla joko ulkoinen tai sisäinen. Ulkoiset viestit lähetetään vastustajalle, mutta sisäisiä ei (ks. kuvio 13).

```
void CSamPlayState::HandleGameEvents()
{
    while (m_pGameEvents->HasNextEvent())
    {
        CSamEvent *e = (CSamEvent *)m_pGameEvents->PopEvent();
        switch (e->m_iEventType)
        {
            case EVENT_PHYSICS_STATE: EventPhysicsState(e); break;
            case EVENT_SYNC_PLAYERS:  EventSyncPlayers(e); break;
            case EVENT_NEXT_PLAYER:   EventNextPlayer(e); break;
            .
            .
            .
        }

        if (e->m_bBroadcast)
        {
            m_pNetworkEvents->AddEvent( e->m_iEventType,
                                        e->m_pData,
                                        e->m_iLength,
                                        DE_FALSE);
        }
        DE_SAFE_DELETE(e);
    }
}
```

Kuvio 13. Tapahtumien käsittely

4.5 Verkkoviestintä

Pelivuoron vaihto oli asia, joka vaihtui moneen kertaan. Ensikertalaisena verkkopelin tekijänä tuli tehtyä monia virheitä.

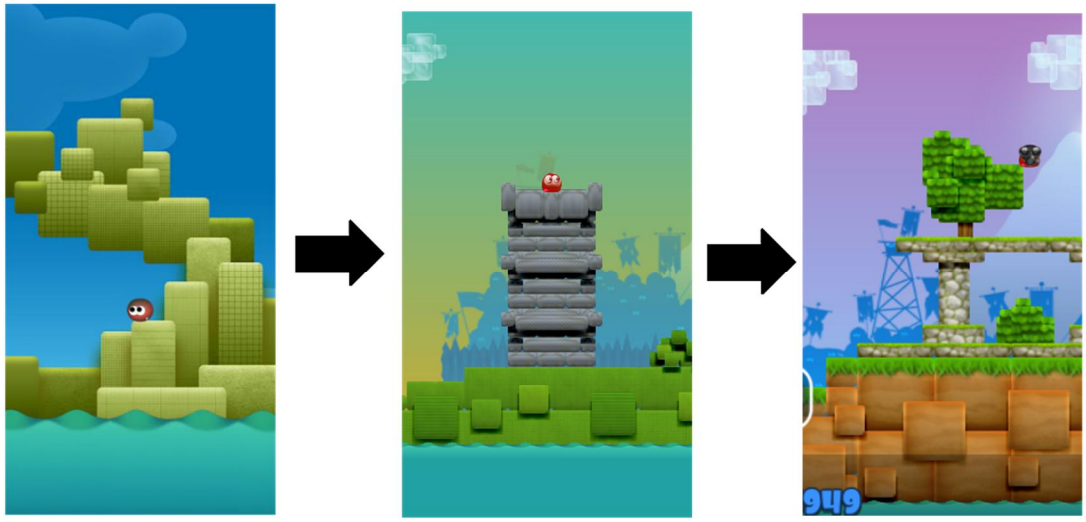
Alunperin jokainen tapahtuma, joka maailmassa tapahtui, välitettiin toiselle pelaajalle välittömästi, ja toinen pelaaja viestin saadessaan suoritti tapahtuman. Verkkoviiveiden vaihtelun vuoksi tämä tapa ei toimi ollenkaan. Toinen pelaaja voi saada ammuksen heitto -viestin viiveellä ja vastaavasti ammuksen räjähdys -viestin nopeasti, jolloin vastustaja ei välttämättä näe koko ammusta vaan pelkän räjähdyksen jossakin.

Toinen versio oli tallettaa kaikki tapahtumat, ja kun maailma on pysähtynyt, lähettää lista toiselle pelaajalle. Tämä tapa toimii, mutta toinen pelaaja joutuu odottamaan ensin, että pelaaja tekee mitään, sitten koko simuloinnin ajan odottaa pakettia ja sitten vasta näyttää simulointi. Tämä tapa on käytössä esimerkiksi Worms-pelisarjan peleissä.

Kolmas ja julkaistussa pelissä käytössä oleva versio ottaa pelaajan antaman käskyn (hyppy, ampuminen yms.) ja simuloi koko vuoron välittömästi (noin 1s aikana) taustalla, väliaikaisessa kopiossa maailmasta, tallettaa tapahtumat ja lähettää ne vastustajalle ja sitten näyttää ne omalla ruudulla. Tämä toimii todella hyvin, ja kohtuullisen nopealla yhteydellä vastustaja näkee melkein reaaliajassa toisen tekemän vuoron, jolloin vuoronvaihdotkin nopeutuvat ja pelaamisesta tulee sujuvampaa.

4.6 Graafinen ulkoasu

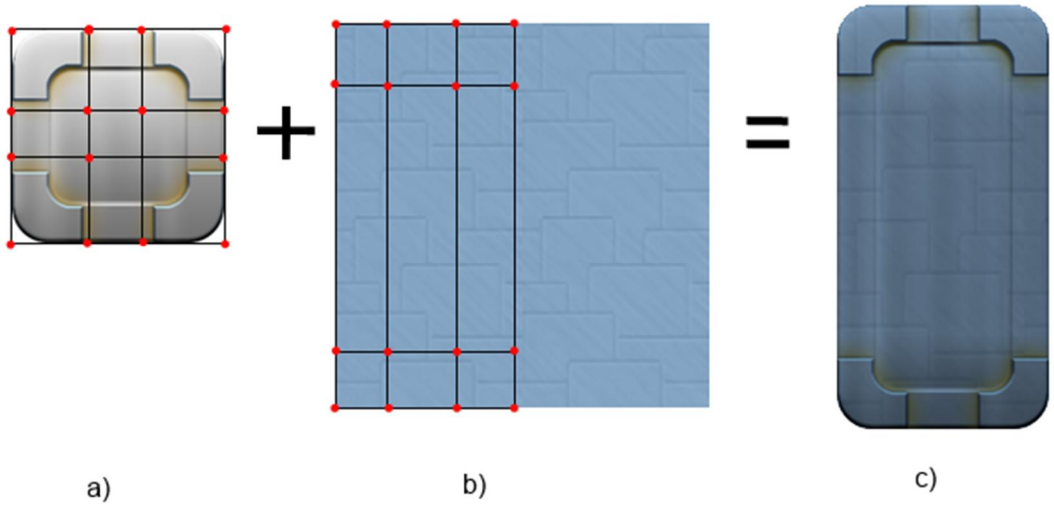
Pelin grafiikka muotoutui testigrafiikoista ensimmäisen julkaisun grafiikoihin ja myöhemmin isomman päivityksen mukana nykyiseen ulkoasuunsa. Maaston editoinnin myötä kenttäsuunnittelija alkoi luoda taistelukentistä tosimaailmaa kuvaavia miniatyyrejä, minkä myötä myös palikoiden tekstuurit alkoivat muuttua erilaisten materiaalien näköisiksi (ks. kuvio 14).



Kuvio 14. Graafisen ulkoasun kehitys. a) testaus b) 1. julkaisu c) päivitys

Grafiikankiihdytys rajapinnat piirtävät ruutuun kolmioita, joita yhdistelemällä voidaan piirtää melkein mitä tahansa. Yksittäinen palikka piirretään käyttäen useampaa näitä kolmiota ja luoden niistä ruudukko. Tämä ruudukon sisältö lasketaan palikan koon pe-

rusteella ja sille luodaan pintamateriaali koordinaatit. Piirrettäessä käytetään kahta materiaalia, maskia ja pintakuviota. Maskin avulla pyöristetään kulmat ja lisätään pohjakuviointi kyseiselle palikalle. Pintakuvio antaa palikalle tarkemman ja yksilöllisemmän ulkoasun. (ks. Kuvio 15). Yhdistelemällä eri pintakuvioita ja maskeja saadaan luotua erilaisia ryhmiä palikoista. Pelimaailman palikoilla on 3 eri ryhmää ja jokainen ryhmä käyttää samaa maskia, mutta niiden pintakuvioita vaihdellaan. Tällä tavoin pelaajat voivat erottaa samaan ryhmään kuuluvat palikat niiden ulkoasun perusteella.



Kuvio 15. Yhden palikan piirto. a) maski materiaali ja koordinaatit b) pintakuvio ja koordinaatit c) lopputulos

4.7 AI - Keinoäly

Peliin tehtiin myös automaattisia pelaajia, joita ohjaa tekoäly. Näitä käytetään pelitestauksessa ja tilanteissa joissa ihmisvastustajia ei löydy.

Keinoäly rakennettiin tarkoituksella aika vaatimattomaksi. Se laskee alustavan suunnan ja voiman pelaajien etäisyyden perusteella. Näihin arvoihin lisätään satunnaisuutta ja arvotaan käytettävä ase ja liikkuminen. Näillä eväillä saatiin luotua melko koomisen oloinen ja näin ollen ensipelaajamainen vastustaja, jota vastaan aloittelevien pelaajien on mukava pelata.

5 TYÖN ARVIOINTI

5.1 Ongelmat

Ongelmia ratkottiin muiden kanssa ja palavereita pidettiin tarpeen mukaan projektiryhmän kesken. Viikoittaiset palaverit pitivät yllä jatkuvaa kehitystä ja antoivat mahdollisuuden nopeille muutoksille ja uusien ideoiden esittämiselle. Mielestäni parasta tämmöisessä pienessä projektiryhmässä oli mahdollisuus työskennellä luovien ihmisten kanssa, ideoida uusia asioita ja ratkoa ongelmia.

Suurimpia ongelmia oli saada verkkoviestintä toimimaan. Samaan aikaan oli kehityksessä moninpelikerros, uusi serveri ja peli. Varsinkin verkkoviesteihin liittyvien vikojen kanssa ei voinut tietää mistä päin vikaa lähtisi edes etsimään. Sen jälkeen koko verkkoviestien rakenne on yksinkertaistunut ja toimii luotettavasti.

Julkaisun jälkeen uusia pelaajia tuli niin paljon että serverit menivät tukkoon ja pelaajat eivät päässeet kirjautumaan peliin. Serveriä ei ollut ikinä testattu niin suurella käyttäjämäärällä. Nykyään tehdään kuormitustestejä ulkopuolisilta koneilta automaattisilla pelaajilla.

5.2 Onnistumiset

Kokonaisuudessaan peli onnistui. Sitä pelataan jatkuvasti ja pelaajat haluavat lisää uutta sisältöä peliin. Kirjoitushetkellä peliä on pelattu noin. 4 miljoonaa kertaa. Pelimaailma on miellyttävä ja pelattavuus on kohdallaan. Parannettavaa tietysti on aina ja uusi versio on kehitteillä.

Suurimpana onnistumisena voi pitää pelinkehityksestä kertynyttä kokemusta. Projektin aikana tuli eteen mitä ihmeellisimpiä ongelmia ja kaikki niistä saatiin kuitenkin ratkaistua.

LÄHTEET

Box2D User Manual. 2011. Viitattu 29.3.2013. http://box2d.org/manual.pdf

Company presentation. 2012. Esite. Star Arcaden markkinointimateriaali.

Gamma, E., Heim, R., Johnson, R. & Vlissides, J. 2001. Olio-ohjelmointi - Suunnittelumallit. Helsinki: IT Press.

McShaffry, M. & Graham, D. 2012. Game Coding Complete. 4. uud. p. Boston: Course Technology PTR.