EVTEK-ammattikorkeakoulu
Muotoiluinstituutti
Viestintä
3D-Multimediasuunnittelu

Peligrafiikan tuottaminen mobiililaitteille

Opinnäytetyö

Oskari Raunio

Oskari Raunio
3D-Multimediasuunnittelu
Opinnäytetyö
Huhtikuu 2008

EVTEK-ammattikorkeakoulu
Muotoiluinstituutti
Viestintä
3D-Multimediasuunnittelu

TIIVISTELMÄ

Tekijä: Kai Oskari Raunio

Opinnäytetyön nimi: Peligrafiikan tuottaminen mobiililaitteille

Vuosi: 2008 Sivumäärä:46

Opinnäytetyön tavoitteena oli tutkia tehokkaita tapoja tuottaa laadukasta grafiikkaa mobiilipeleihin. Mobiililaitteiden laaja valikoima vaatii pelin optimointia useisiin laitteisiin. Tämä on oleellinen osa mobiilipelituotantoa ja voi olla hyvin työlästä. Laitteiden teknologian ja rajoitusten ymmärtäminen on tärkeää suunniteltaessa pelin kehitystä ja grafiikan valmistamista. Koska kehitystiimin yhteistoiminta on tärkeää, tämä opinnäytetyö kuvaa grafiikan teknisen tuotannon lisäksi myös tiimin toimintaa ja yleistä kehitysprosessia.

Opinnäytetyön käytännön osuudessa tuotettiin peligrafiikkaa ”War Diary: Torpedo” peliin toimimalla osana kehitystiimiä. Pelin kehittäjänä ja julkaisijana toimi Rovio Mobile Oy, jonka palveluksessa toimin tuotannon aikana. Kehitysprosessin yhteydessä selvitettiin sopivia lähestymistapoja projektissa ilmeneviin haasteisiin. Näiden ongelmien ratkaisussa tutkittiin kirjallisuutta, keskusteltiin asiantuntijoiden kanssa ja tutustuttiin muissa peleissä käytettyihinratkaisuiihin.

Opinnäytetyö toi peliprojektin osalta esiin uusia ongelmia ja ratkaisuja ja yksityiskohtaisten ongelmien lisäksi myös näkökulmia yleiseen pelinkehitysprosessiin. Yksittäisten ongelmien ratkaisuja pohditaan tarkemmin ja vertaillaan yleiseen mobiilipelien kehitysprosessiin. Näihin ongelmiin ollaan esitetty ratkaisumalleja ja nämä voidaan ottaa tarkempaan harkintaan tulevissa projekteissa.

Asiasanat: Peli, Mobiilipeli, Pelituotanto, Pikseligrafiikka, Peligrafiikka

EVTEK Institute of Art and Designg
Degree Programme:Media
Major:3D Animation and Visualisation

ABSTRACT IN ENGLISH

Author: Kai Oskari Raunio

Bachelor's Thesis: Creating game graphics for mobile devices

Year: 2008 Pages:46

The objective of the study was to explore effective ways to produce high quality mobile game graphics. The target devices were of a wide range of mobile J2ME devices that the current mobile game industry supports. For the very limited devices, the technical understanding of used technology and resource optimizing is vital, so much of the study is concentrated overcoming these limitations. As professional mobile game production depends also greatly of the co-operation of team-members, the study looks not just individual graphics, but also the grater relation of the whole project.

The study was done by producing game graphics for a mobile game "War Diary 3: Torpedo" produced and published by Rovio Mobile ltd. and by studying different aspects of game industry. During the process different approaches for individual problems were examined and tried to find out best strategies for different situations. Valuable knowledge to deepen the understanding of each problem was gathered by looking into literature, discussion with professionals, and by trying out current and older products.

The study yielded a more comprehensive understanding of the aspects that are affecting game graphics production in mobile devices. The individual challenges of the project were solved and are discussed in detail, and compared to the common process of mobile game making. Many problems were found to be solved by using more sophisticated production tools to organize project and to automatize tasks. As many problems and new approaches were uncovered during the process, they can be taken into a greater consideration for later projects and in developing production process and tools.

key words: mobile game, game graphics, game production, pixel art, game art

Sisällys

KÄSITTEET JA LYHENTEET

LÄHDELUETTELO

LIITTEET (10 kpl)

1 JOHDANTO

1.1 Esittely

Peliteollisuudesta on tullut suurimmaksi viihdeteollisuuden alaksi muutamassa vuodessa. Lyhyessä ajassa matkapuhelimien ja muiden mukana kuljetettavien multimedialaitteiden yleistyessä ja kehittyessä, on mobiilipelien tuotanto lähtenyt nopeaan kasvuun, ja alalle ollaan asetettu suuria tulevaisuuden odotuksia. Verrattuna massivisiin konsolipeliprojekteihin, mobiilipelit ovat vielä suhteellisen kevyitä projekteja. Tämä on houkutellut monet pelialalle haluavista pienistä tiimeistä juuri tekemään mobiilipelejä. Alan suuret odotukset ovat puolestaan houkutelleet monet isot pelialan yritykset panostamaan mobiilipeleihin. Mobiiliympäristöllä on omien erityismahdollisuuksiensa lisäksi myös monia erityishaasteita, jotka eivät välttämättä ole ilmeisiä. Laitteiden nopea kehitys luo myös alalle tarpeen kehittyä mukana. Tämä lopputyö käsittelee peligrafiikan tuottamista ladattaviin JAVA-peleihin graafikon näkökulmasta ja alan erikoishaasteisiin vastaamista.

1.2 Tutkimuksen tarve

Kasvava mobiilipeliala on jo nyt merkittävä ala, ja sen odotetaan jatkavan kymmenien prosenttien vuosikasvua ja laiteympäristön monipuolistuvan. Pelinkehityksestä on yleisesti ja erikseen mobiilipuolella julkaistu tutkimuksia ja kirjoituksia, mutta grafiikantuotannon kohdalla julkaisut ovat lähes olemattomia, tai niukkoja. Tässä tutkimuksessa on tarkoitus selvittää useimpia graafikon kohtaamia kehityshaasteita, ja tarjota niihin ratkaisuja. Lisäksi kehittyvä ympäristö luo uusia haasteita, joten tulevaisuuteen varautuminen ja muutosten ennakointi on yksi osa tutkimusta.

1.3 Tutkimusalue

Tutkimuksen kohteena oli selvittää mitä erikoishaasteita mobiilipeligrafiikan tuotannossa esiintyy. Aihetta tarkastellaan erityisesti teknisestä näkökulmasta joka ottaa huomioon laitteiden erityispiirteet jotka vaihtelevat huomattavasti sadoilla kohdelaitteilla. Tämänlaisia rajoituksia ovat erityisesti laitteiden vaihtelevat muistin määrä, suoritusteho ja näytön koko. Tässä tutkimuksessa keskitytään ladattaviin peleihin J2ME mobiililaitteilla, jotka ovat pääsääntöisesti matkapuhelimia. Tuotannon erityispiirteet ja markkinaympäristö vaativat graafikoilta sopivat kehitysmetodit, joita myös käsitellään.

1.4 Käytännön osuus

Osana työtä toteutettiin peligrafiikkaa peliin ”War Diary 3: Torpedo”, jonka kehitti ja julkaisi Rovio Mobile Oy. Tämän toteutin yhteistyössä toisen graafikon, Juha Impivaaran kanssa. Grafiikoiden tavoitteena oli saada näyttävää grafiikkaa aikataulun puitteissa pelin lukuisille eri laiteprofiileille. Tässä työssä ei kuitenkaan riittänyt pelkkä grafiikoiden valmistaminen, vaan grafiikoiden olisi oltava myös sellaisia, että projektin sisällä niitä voidaan käyttää tehokkaasti. Erityisen tärkeää prosessissa on, että ohjelmoija saa tarvittavat grafiikat käyttöönsä nopeasti ja pelien käännösvaiheessa eri grafiikkaprofiilit tuovat mahdollisimman vähän lisätyötä ja riskejä.

1.5 Opinnäytetyön kuvaus

Opinnäytetyössä kuvataan mobiilipeliteollisuuden toimintaympäristöä lyhyesti. Mobiililaitteiden teknisiä ominaisuuksia ja rajoituksia kuvataan yksityiskohtaisesti, sen mukaan mitä rajoituksia ne luovat graafikolle. Samalla näihin ongelmiin esitetään yleisiä ratkaisumalleja, mitä alalla käytetään. Graafikon työskentelymetodit ovat hyvin läheisesti liitoksissa muuhun projektiin ja sen vuoksi työskentelymetodien lisäksi kuvataan laajemmin pelin tuotantoa ja peligrafiikoiden suhdetta muuhun projektiin ja graafikon yhteistyötä kehitystiimin kanssa. Niin teknisellä tasolla, kuin työskentelymetodeissa tarkastellaan aihetta tehokkaan tuotannon näkökulmasta. Grafiikoiden olisi oltava näyttävää vähillä resursseilla ja tuotannon tehokasta pienellä tiimillä. Käytännönosuudessa kuvataan käytännön toteutus ja verrataan sitä teoriaan. Uniikkeja ratkaisumalleja selostetaan tarkemmin. Lopuksi arvioidaan työn onnistumista, siitä saatuja kokemuksia, kehitysmahdollisuuksia ja alan tulevaisuutta yleisemmin.

2 TAUSTAA

2.1 Mobiilialusta

Mobiilialustan perimmäinen piirre on, että se on tarpeeksi pienikokoinen, ja sitä voidaan kuljettaa mukana ja olla helposti käytettävä. Tästä on seurauksena, että huomattavaa osaa laitteen ominaisuuksia on jouduttu karsimaan. Laitteet ovat usein tarkoitettu ensisijaisesti toimimaan puhelimena ja tekstiviestien vaihtamiseen. Näin ollen pelin on hankalampi hyödyntää laitetta, joka ei ole suunniteltu pelikäyttöön. Laitteiden kehittyessä ja pelimarkkinoiden kasvaessa, ovat laitteet kuitenkin ottaneet entistä paremmin huomioon myös pelikäytön

2.2 Mobiilipelien historia

Jon Franzas kertoo, että ensimmäiset mukana kuljetettavien laitteiden mukana tulevat pelit ilmestyivät jo 1970-luvun lopussa taskulaskimiin lisätuotteina.[1] Pian tämän jälkeen ilmestyivät Nintendon julkaisemat pelkästään peliköyttään tarkoitetut pelilaitteet. Näissä laitteissa grafiikka perustui nestekidenäytöllä yksinkertaisesti liikkuviin hahmoihin. Vuonna 1989 Nintendo otti uuden askeleen pelialalla julkaisemalla Gameboy laitteen, jossa grafiikka perustui jo mustavalkoisiin pikseleihin. Myöhemmin julkaistussa Gameboy Colorissa oli jo värinäyttö, ja mukana kannettavat pelilaitteet ovat jatkaneet kehitystä nykyaikaisiin kosketusnäytöllisiin multimedialaitteisiin. Matkapuhelimiin pelit tulivat vuonna 1997, kun Nokia julkaisi 5100 ja 6100 puhelinmallien mukana yksinkertaiset mato-, muisti-, ja logiikkapelit. Puhelimilla oli pienet mustavalkoiset näytöt, joissa oli huono valoteho. Myöhemmin puhelimiin pystyi myös lataamaan SMS-, WAP-, ja Java-pelejä. Myös puhelimet kehittyivät ja Nokia julkaisi jopa pelikäyttöön tarkoitettu N-Gage puhelimen 2003. Käytettävyystutkimuksessa N-gage esitti kuitenkin myös suuria puutteita pelattavuudellaan.[2] Lisäksi ohjelmien lataaminen ei ole toiminut asiakasta tyydyttävällää tavalla ja vielä vuonna 2008 joka kolmas ladatuista mobiilipeleistä ei toiminut tyydyttävästi, tai ollenkaan. [3]

2.3 Mobiilipeliala

Mobiilipeliala nuorena alana on päässyt jo pioneerivaiheen yli, ja alkanut vakiintumaan ja kypsymään. Mobiilipelien vuosikasvu on kuitenkin ollut kymmeniä prosentteja, ja nopean kasvun uskotaan vielä jatkuvan vielä vuosia. Juniper Research-tutkimusyhtiö arvioi mobiilipelimarkkinoiden kasvavan lähes 10 000 000 000 dollariin vuoteen 2009 mennessä.[4]

3 MOBIILIPELIN TUOTANTO

3.1 Työvälineet

Kuten ohjelmistotutoannossa yleensä, ovat mobiilipelituotannon kustannukset pitkälti henkilökustannuksia. Laajan kohdelaitekokoelman vuoksi on moni työssä paljon toistuvia työvaiheita projektien sisällä ja projektien välillä. Tehokkailla työvälineillä voidaan näitä prosesseja automatisoida ja tehostaa osallistujien tuottavuutta. Projektit kuitenkin poikkeavat toisistaan monin osin ja asettavat arvaamattomia haasteita, joten työkalujen on pystyttävä muuntautumaan vaihteleviin projekteihin. Tehokkailla työvälineillä voidaan myös hyödyntää paremmin monien laitteiden rajattuja resursseja tehokkaammin, jolloin voidaan tuottaa teknisesti laadukkaampia tuotteita. Varsinkin grafiikan tuotannossa korostuu työkalujen merkitys. Sopivilla työvälineillä alennetut henkilöstökustannukset ja paremmalla teknisellä laadulla voi olla yritykselle merkittävä kilpailuetu.

3.1.1 Grafiikkaohjelmistot

Grafiikantuotannossa käytetään yleisiä grafiikkaohjelmia. Yleisin grafiikkaohjelma on standardiksi muodostunut Adobe Photoshop-kuvankäsittelyohjelma, jolla pystyy tekemään monipuolista kuvankäsittelyä ja grafiikantuotantoa. Tämän lisäksi voidaan käyttää ohjelmia, jotka ovat erikoistuneet tiettyyn aiheeseen. Pikselitason grafiikkaa, jolle mobiilipelit pitkälti perustuvat, toteutetaan yleensä niille tarkoitetuilla ohjelmilla. Varsinkin animaatioiden teossa on hyvä olla animaatioiden tekoon tarkoitettu pikselitason piirto-ohjelma. Lisäksi apuna voidaan käyttää vektoripohjaisia piirto-ohjelmia, tai 3d ohjelmistoja. Grafiikkatiedostojen käsittelyyn ja pakkaamiseen tarvitaan myös omat ohjelmat, koska PNG-kuvaa voidaan pakata tiiviimmäksi, kuin yleiset kuvankäsittelyohjelmat pystyvät.

3.1.2 Versionhallinta

Kun projektiin osallistuu useampi henkilö eri osa-alueilla, heidän on kehitettävä projektia jatkuvassa yhteistyössä. Ohjelmaa voi olla kirjoittamassa samaan aikaan useampi ohjelmoija, grafiikka tuottamassa useampi graafikko, pelisuunnittelija valmistaa pelikenttiä ja lisäksi tuottaja haluaa kokeilla uusinta versiota. Tällöin kaikkien on oltava yhteydessä versionhallintaohjelmalla palvelimelle, josta kaikki saavat haettua uusimmat peliin kuuluvat tiedostot. Suosituimpia ovat CVS tai SVN versionhallinta-ohjelmistot, joita voidaan käyttää monen ohjelmistoympäristön tai muun ohjelman kautta suoraan. Projekti on kuitenkin organisoitava siten, etteivät useat ihmiset muokkaa samoja tiedostoja yhtä aikaa, eikä sisäisiä konflikteja pääse syntymään. Konfliktien selvittely voi viedä verrattain paljon aikaa. Jos projekti muuttuu epävakaaksi, tai havaitaan että jossain kehitysvaiheessa on tehty valintoja, joita ei voida suoraan korjata, voidaan versionhallinnasta palauttaa vanhempi versio projektista tai sen osasta.

3.1.3 Kommunikaatio

Mobiilipelikehityksen luonteeseen sopivat niin sanotun ketterän kehitysideologian mukainen vapaamuotoinen spontaani kommunikaatio kasvokkain. Tämä tapahtuu luontevasti suullisesti, mikäli kaikki osalliset ovat fyysisesti läsnä. Mikäli tiimin jäseniä työskentelee muualla, on heidän

kanssaan yhteydenpito sulavaa erilaisilla pikaviestintäohjelmilla, kuten Skypellä. Näillä voidaan myös siirtää helposti tekstiä ja tiedostoja työpisteiden välillä, ja voidaan pitää ryhmäkeskusteluja häiritsemättä muita. Tärkeämpien viestien lähettämiseen soveltuu perinteinen sähköposti. Dokumentit ovat usein palvelimella saatavana, ja bugien raportoimiseen käytetään siihen kehiteltyä ohjelmaa, kuten *Bugzillaa*. Yhä enemmän alalla on siirrytty käyttämään ohjelmakehitykseen tarkoitettuja yrityswikejä joilla tietojen tavoittamista ja muokkaamista on pyritty helpottamaan. Näillä ohjelmilla on lisäksi pyritty yhdistämään versionhallinta, projektinhallinta ja viestinnän osa-alueet jouhevasti yhdessä toimiviksi kokonaisuuksiksi.

3.1.4 Tuotantoalusta

Tuotantoalusta (prduction platform) on mobiilipelituotannon keskeisimpiä työvälineitä. Hyvä tuotantoalusta tarjoaa rajapinnan päätelaitteiden ja ohjelmakoodin välillä, jolloin koodista saadaan päätelaitteista riippumatonta, ja ohjelman muokkausta erilaisille päätelaitteille saadaan nopeutettua, kun uutta koodia ei tarvitse kirjoittaa eri laitteille. J2ME ympäristöön ohjelmia tuotettaessa tuotanto-alustan on mahdollistettava kevyen ja tehokkaan ohjelmakoodin ja sen käyttämän datan nopean ja luotettavan kehitystyön ja useille laitteille kääntämisen. Hyvä tuotantoalusta vastaa siitä, että peli saadaan toimitettua myös porttausvaiheen läpi tehokkaasti, ilman että varsinaiseen ohjelmakoodiin tarvitsee enää koskea.

3.1.5 Kenttäeditori

Pelisuunnittelija luo peliin pelikentät käyttäen tätä varten suunniteltua kenttäeditoria. Editori mahdollistaa pelisuunnittelijan vaivattoman työskentelyn pelikenttiä luodessa ja mahdollisuuden kokeilla kenttää pelissä välittömästi. Käytännössä editorilla rakennetaan kenttä siihen ladatuista spriteistä ja kenttään sijoitetaan pelihahmoja, esineitä, merkityksellisiä alueita, tapahtumankulun määrittelyjä ja muita pelikentässä tarvittavia elementtejä. Kentän muistinkulutuksen seuraaminen on myös tärkeää, ja mahdollisuus määritellä erikseen korkean ja matalan profiilin laitteille yksityiskohtaisemmat, tai karsitummat kentät. Editori ohjelmoidaan itse pelinkehityksen yhteydessä, monesti vanhan editorin pohjalta, jolloin uudelleen kirjoitettavaa koodia tulee mahdollisimman vähän. Peliohjelman ja editorin kirjoittajat ja pelisuunnittelija joutuvat tekemään yhteistyötä, jotta suunnittelija pystyy tuottamaan editorilla peliohjelmalle sopivaa dataa ilman, että varsinaiseen peliohjelmaan joudutaan lisäämään kenttädataan kuuluvaa tietoa (”hard coding”). Hard coding toteutusta pitäisi välttää mahdollisimman pitkälle, koska silloin suunnittelija ei itse pysty muokkaamaan kenttää tältä osin ja vaikeuttaa ohjelman muokkaamista ja uudelleen käyttämistä. Mikäli editoria ei saada tarpeeksi ajoissa kenttäsuunnittelijan käyttöön toimivana, hukkaantuu suunnittelijan työresursseja ja aikataulu voi ylittyä. Pelikehittelyn edetessä pelisuunnitelma muuttua usein joiltain osin, jolloin editoriin ja peliohjelmaan joudutaan tekemään muutoksia. Mikäli nämä muutokset eivät ole suuria ja ne huomataan aikaisessa vaiheessa, ei suurempia ongelmia pääse syntymään.

3.1.6 Erikoistuneet työkalut

Projekti voi vaatia erityisiä työkaluja, varsinkin, mikäli teknologiaympäristö ei ole päivitetty vastaamaan projektin tarpeita. Tämänlaisia työkaluja voidaan usein joutua rakentamaan vaikka kesken projektin, jos vastaan tulee yllättäviä ongelmia. Pääsääntöisesti kuitenkin työkalut muodostuvat edellisten projektien kokemusten pohjalta, tai sitten projektin alussa on suunniteltu, että projektia varten rakennetaan jokin tarvittava työkaluohjelma.

3.2 Projektin vaiheet

Pelituotanto jakaantuu kolmeen osaan: esituotantoon, tuotantoon ja jälkituotantoon. Mobiilipeliprojektit ovat lyhyitä, ja seuraava projekti alkaa heti edellisen päätyttyä. Yrityksen markkinointi tarvitsee kuitenkin usein pitkälle eteenpäin tiedon tulevista peleistä ja niiden valmistumisaikatauluista. Yrityksen on suunniteltava pitkälle tulevat projektit myös ihmisresurssien käytön kannalta ja jos yksi projekti myöhästyy, vaikuttaa se myös muihin jonossa olevin projekteihin.

Koska projekti jakautuu kolmeen osaan, voidaan projektit ketjuttaa peräkkäin pipeline-ajattelun mukaisesti, jolloin esituotannon päätyttyä esituotantotiimi siirtyy seuraavaan projektiin ja tuotantotiimi siirtyy projektiin edellisestä projektista. Näin henkilöstö on koko ajan tehokkaassa käytössä, eikä työvoima seiso toimettomana.

Todellisuudessa tuotanto ei toimi näin yksinkertaisesti. Esituotannon ja tuotannon hoitaa monesti sama tiimi, mahdollisesti pienillä henkilövaihdoilla. Jälkituotannossa yleensä on siihen erikoistunut tiimi, mutta jälkituotanto voi tarvita jonkin verran myös tuotantotiimin läsnäoloa. Tuotantotiimistä myös vapautuu henkilöitä jo ennen tuotannon loppua, tai heidän ei tarvitse projektissa täysin läsnä. Lisäksi henkilöstö voi olla myös lomalla, tai poissa jonkin odottamattoman syyn, kuten sairauden vuoksi jolloin tämä pitää sovittaa projektin henkilöresursseihin. Jos studiolla on useampi tuotanto rinnan, voidaan resursseja jakaa projektien kesken. Niinsanotut ketterät, eli ”Agile” tuotantomallit tähtäävät myös projektin ja resurssien dynaamiseen vuorovaikutukseen.

3.2.1 Tuotannon aloittaminen

Mobiilipelituotannon aloittaminen vaihtelee yrityksen ja tilanteen mukaan, mutta yleisesti projektin aloittamispäätös tehdään peli-idea pohjalta, joka on laadittu arvioitavaksi. Kun päätös on tehty, määritellään projektin tarvitsemat henkilöstöresurssit ja aikataulu. Tuotanto itsessään jakautuu kolmeen vaiheeseen. Käytännöt voivat vaihdella, esimerkiksi esituotantoa ei välttämättä määritellä erikseen, tai projektin aloittaminen voidaan päättää ”Green light” kokouksessa, joita voidaan pitää useampiakin projektin alussa.

3.2.2 Esituotanto

Esituotannon tärkein rooli on selvittää mitä tuotannossa tullaan tekemään. Esituotantovaiheen merkitys on korostunut varsinkin konsolipelin tuotannossa, joissa riskit ovat suuremmat ja projektit monimutkaisemmat. Varsinkin riskitekijöiden määrittely on tärkeää tehdä tässä vaiheessa, jotta niihin voidaan varautua itse tuotannon alettua.

Esituotannossa konsepti voi vielä muuttua, ennen kuin se viimeistellään. Esituotannossa on hyvä myös tehdä varhaisia prototyyppejä pelimekaniikasta, jotta voidaan havaita toimiiko peli-idea ja algoritmit suunnitellulla tavalla. Esituotannossa voidaan konseptitaidetta jatkaa ja peligraafikko tekee placeholder-grafiikat, joita voidaan käyttää suoraan prototyypeissä ja varsinaisessa pelissä ennen viimeisteltyjen grafiikoiden valmistumista. Pelisuunnittelijoita varten määritellään, millaisia työkaluja tullee tarvitsemaan, mikä tarkoittaa pääasiassa kenttäeditoria. Esituotannon kesto voi olla kahdesta neljään viikkoon. Esituotantoa voidaan kutsua myös suunnittelu-, tai visiointivaiheeksi.

3.2.3 Tuotanto

Tuotannon alkaessa, kaikkien työtehtävät ovat määriteltyinä. Tuotannossa kirjoitettaan peliohjelma valmiiksi; grafiikat tehdään viimeistellyiksi; äänet ja musiikki tuotetaan valmiiksi; Pelikentät suunnitellaan valmiiksi. Työkaluohjelmoijan on saatava kenttäsuunnittelijalle tuotannon varhaisessa vaiheessa alkeellinen kenttäeditori, jotta tämä pystyy aloittamaan työnsä.

Projektin aikana voidaan huomata, että jokin suunnitelma ei täytä asetettuja odotuksia. Tällaiseen huomioon on reagoitava nopeasti, koska mitä myöhemmäksi projektia tämä jää, sitä hankalampi sitä on korjata. Aikaisilla prototyypeillä ja pelattavuustestauksella voidaan nopeuttaa tätä prosessia. Myös laitteiden suoritustehot, tai itse aikataulu voi osoittautua riittämättömäksi joillekin ominaisuuksille, jolloin kyseisestä ominaisuudesta on luovuttava, tai yksinkertaistettava, jotteivät aikataulut ylity. Tuotannon edetessä, kun määriteltyjä ominaisuuksia ollaan saatu valmiiksi, tulee tärkeämmiksi näiden toiminnan testaaminen bugien varalta, jotta ohjelmoija pystyy tehokkaasti korjaamaan bugeja. Projektin tuottajan on pidettävä huolta, että työvoiman resursointi on riittävää kaikille työtehtäville, ja projektin etenemistä suhteessa aikatauluarvioihin seurataan. Projekti voi olla suurissa ongelmissa, mikäli aikatauluarviot osoittautuvat vääriksi ja tilanteeseen reagoidaan liian myöhään. Normaaleilla projekteilla tuotantovaihe kestää kahdeksasta kahteentoista viikkoon.

Koska tuotantovaihe on huomattavasti esituotantoa pidempi, on se jaettu seuraavan jälkituotannon kanssa välitavoitteisiin(”Milestone”'). Tuotannossa on yleensä vähintään viisi välitavoitetta, joiden nimet voivat vaihdella organisaation mukaan:

First Palyable: Sisältää ensimmäisen pelikentän jota voidaan pelata, ja tärkeimmät pelilliset elementit toimivat. Grafiikoiksi riittävät placeholder grafiikat.

Alpha versio: Alpha versiossa pitäisi olla kaikki oleelliset elementit, jota pelissä tulee olemaan. Grafiikka ei ole vielä viimeisessä muodossaan, mutta välittää kuitenkin pelin yleisilmeen.

Beta versio: Beta version pitäisi sisältää kaikki lopullisen version elementit, mutta voi sisältää vielä tunnettuja bugeja.

QA-Release: Peli julkaistaan laadunvalvontaan (Quality Assurance), eikä siinä ole tunnettuja bugeja. Tällöin jälkituotanto alkaa. Löydettäessä bugit korjataan.

Gold-Release: Peli on valmis asiakkaalle toimitettavaksi. Myöhempien porttauksien teko jatkuu.

Välivaiheet kestävät yleisesti muutaman viikon. Mobiilituotannon vaatimat useat eri tason profiilit tarvitsevat omat versionsa. Nämä versiot valmistuvat eri tahdissa, jolloin voidaan määritellä, mikä profiili valmistuu ensiksi, ja missä aikataulussa loput valmistuvat. Tuotannossa on hyvä pitää mielessä kuitenkin muutkin profiilit, jotta voidaan varmistua, että kaikki profiilit voidaan toteuttaa suunnitelman mukaan. Graafikolla pitää olla valmiina muihin profiileihin sopivat placeholder grafiikat, kun ohjelmoija kokeilee koodin toimintaa eri profiileissa. Järkevällä työjärjestyksellä ja työkaluilla voi graafikko tehostaa tuotantoa omalta osaltaan.

3.2.4 Jälkituotanto

Jälkituotannossa mobiilipeleillä tarkoittaa pääasiassa sitä, että pelistä valmistetaan kullekin

kohdelaitteelle sopivat versiot(eli peli ”portataan”) ja niiden toimivuus testataan.[5] Tuotantovaiheen tavoitteena on tehdä jälkituotannolle sellainen tuote, ettei jälkituotannossa tarvitse enää koskea itse peliohjelmaan tai grafiikoihin, vaan peli saadaan käännettyä tehokkaasti sadoille vaadittaville laitteille automatisoidusti. On myös varauduttava myöhemmin ilmestyviin laitteisiin, joita ei ole vielä markkinoilla. Tämä ei aina onnistu, vaan joillekin laitteiden kohdalla joudutaan tekemään pientä säätöä grafiikoiden, tai ohjelman osalta. Jälkituotannossa voidaan joutua myös lisäämään asiakkaan pyynnöstä operaattorikohtaisia lisäyksiä, kuten uusia kieliä tai lisäyksiä laskutusta tai piste-ennätysten keräämistä varten. Operaattori voi myös vaatia grafiikoiden tai muun sisällön muuttamista, varsinkin kun peliä myydään eri kulttuurialueilla. [6]

3.2.5 Projektin jälkeen

Projektin valmistuttua siirrytään ylläpitovaiheeseen, jolloin tarvittaessa tehdään uusia porttauksia, tai lisätään vanhoihin ominaisuuksia operaattoreiden tarpeisiin. Kehitystiimi itse pitää post mortem palaverin, jossa analysoidaan projektin kulkua ja lopputulosta. Tämän tarkoituksena on oppia, mitä asioita tehtiin oikein, mitä väärin ja miten voidaan kehittyä.

3.3 Tuotantoprosessi

Mobiilipelituotannossa voidaan käyttää yleisiä ohjelmistokehitys- ja erityisesti pelikehitysprosessi-malleja. Kasvavaa suosiota ovat saaneet ns. ketterät kehitysmetodit(Agile software development methods), jotka ovat kehitetty haasteisiin, joita varsinkin mobiilipelien kehityksessä ilmenee. Projektista ja kehitysfilosofiasta riippuen projektissa voi olla eri määrä vaiheita, ja niillä voi olla poikkeavat nimet. Nämä voivat vaikka olla *määrittely, implementointi, validointi ja kehittäminen*. [7]

3.3.1 Määrittely

Kehitykseen liittyy yleensä vaatimusten kokoaminen, jossa asiakkaita tai mahdollisia käyttäjiä haastatellaan. Vaatimukset määritellään sitten yksityiskohtaisesti ja laaditaan Pelissuunnittelu dokumentti (GDD). Tästä kerätään pelin ohjelmoinnin vaatimukset yksityiskohtaisesti. Projektin edetessä monet vaatimukset tarkentuvat tai muuttuvat, joten GDD:tä on päivitettävä jatkuvasti ajantasalle.

3.3.2 Implementointi

Implementointi sisältää yleensä tietyn osan suunnittelun ja ohjelmoinnin, tai se voi tarkoittaa myös koko projektin suunnittelua ja valmistamista. Erikseen voidaan puhua arkkitehtuurista, jolla tarkoitetaan tuotteen abstraktimpaa suunnittelua. Implementointi jakautuu projektin koon mukaan vaiheisiin, jossa alisysteemi tai moduuli suunnitellaan, kehitellään ja testataan. Korjattu ja uudelleen testattu osa hyväksytään ennen siirtymistä seuraavaan vaiheeseen. Kehitysprosessia voidaan nähdä sarjana syklejä, joissa tuotantoprosessi tuottaa yksittäisiä komponentteja, mutta myös itse projektin valmistamista. Graafikolla tämä tarkoittaa jonkin graafisen elementin tuottamista ja toiminnan testaamista. Tuotannollisista syistä graafikko ei aina pääse heti testaamaan grafiikan toimivuutta teknisesta tai visuaalisesta näkökulmasta, jolloin hän joutuu työstämään useaa elementtiä yhtä aikaa. Tämä voi aiheuttaa ongelmia myös grafiikoiden yhteensopivuuden kannalta.

3.3.3 Validointi

Tuotetta testataan. Testattavana voi olla koko tuote, tai vain jokin osa sitä. Mobiilipeleissä usein ongelmaksi muodostuu grafiikan yhteen sovittaminen ja oikein piirtäminen, joten suuri osa testaamista tapahtuu peliä pelaamalla ja virheitä havaitsemalla. Tahti kiihtyy projektin lähestyessä loppua.

3.3.4 Kehittäminen

Kun tuote on otettu käyttöön, voivat vaatimukset vielä muuttua ja ohjelmiston on sopeuduttava uusiin tarpeisiin. Myös ilmenneitä bugeja fiksataan.Tietokoneohjelmistoissa päivitykset julkaistaan yleensä patcheina. Mobiileissa ei voida julkaista patcheja, mutta ylläpitoon kuuluu uusien laitteiden tukeminen ja uusien asiakkaisden vaatimusten tukeminen, mikä tuotteen levitystä koitetaan laajentaa mahdollisimman kattavaksi. Uusia ominaisuuksia ei yleensä lisätä, koska harva asiakas on niistä valmis maksamaan.

3.4 Tuotantomallit

3.4.1 Vesiputousmalli

Vesiputousmalliksi kutsutaan erästä perinteistä tapaa tuottaa ohjelmistoja. Tässä mallissa oletetaan, että tuotteen vaatimukset voidaan määritellä ennalta, jolloin voidaan määritellä tuotantoprosessi tarkkaan etukäteen ja edetä vaiheesta toiseen, suorittamalla yksi vaihe kerralla, palaamatta edelliseen vaiheeseen. Tässä mallissa suunnitelmasta on hankala poiketa, ja mikäli tuotteen ja prosessin vaatimuksia ei ole ennakoitu oikein, voi projektissa syntyä suuria vaikeusia.

3.4.2 Ketterät kehitysmallit

Ketterillä metodeilla tarkoitetaan erinäisiä kehitysmalleja, jotka noudattavat ketterän ohjelmistokehityksen manifestia (Manifesto for Agile Software Development)[8]. Tämä malli tähtää kehitysmetodeihin, jotka mahdollistavat epävarmassa ympäristössä tehokkaan tuotannon, joka on sopeutuvainen muutoksiin. Näitä kehitysmalleja on lukuisia, ja niitä voidaan soveltaa tarpeen mukaan. Manifestissa mainitaan arvostukseksi:

Yksilöt ja vuorovaikutus *ennen prosessia ja työkaluja.*
Toimiva ohjelma *ennen kattavaa dokumentaatiota.*
Asiakas yhteistyö *ennen sopimusneuvottelua.*
Muutokseen vastaaminen *ennen suunnitelman noudattamista.*

Mobiilipeliympäristö on erityisesti sellainen, jossa muutokset ovat yleisiä ja aikataulut tiukkoja, joten monet mobiilipelikehittäjät ovat omaksuneet näitä metodeja.

3.5 Grafiikan tuotanto

Grafiikka on pelin myyntivaltti. Sen on oltava samalla mahdollisimman näyttävää, mutta toimittava myös pelattavuuden kannalta. Tässä tutkimuksessa myöhemmin mainitut vähien resurssien rajoitteet tuovat omat haasteet mobiilipeligrafiikalle. Grafiikan on myös monistuttava helposti

moneen eri kokoon, ja suoritustehoon. Grafiikantuotannon on oltava myös yhteydessä markkinointiin, jolla on omat tarpeensa.

3.5.1 2D Grafiikka

2D grafiikka voi olla hyvinkin erilaista pelistä ja pelityypistä riippuen. Peli voi tapahtua yhdessä ruudussa, jolloin grafiikka voi olla muutamasta staattisesta elementistä muodostettu, tai peli voi kattaa suuren alan, josta pelinäkymä näyttää osan kerrallaan. Suuria alueita ei voida tallentaa kokonaisina laitteen muistiin, vaan ne koostetaan toisiinsa sopivilla vierekkäisillä grafiikka laatoilla (tile). Joissain tapauksissa voidaan taustalla käyttää muutamaa isoa ja näytttävää grafiikkaelementtiä, joiden päälle pelitapahtumat piirretään. Joskus, varsinkin matalan suoritustehon laitteilla voidaan tausta piirtää ohjelmallisesti, tai jättää kokonaan piirtämättä. Helpoimmillaan pelialueen geometria koostuu suorassa olevista nelikulmioista. Ylhäältäpäin kuvatuissa peleissä jotkut laatat voivat muodostaa seinämiä, jotka rajoittavat pelaajaan liikkumista. Jos kuvakulmaa hieman lasketaan, voidaan pelaaja päästää kävelemään myös seinämien taakse. Tälläiseten geometrioiden toteuttaminen on hyvin yksinkertaista. Hieman monimutkaisempaa on isometrisen maalman toteuttaminen, jossa pelilaatat ovat näytöllä vinottain. Tämä lisäksi voidaan kehitellä muunkinlaisia aksonometrioita. Tälläisiä kuvakulmia, joissa pelaaja näkee pelihahmon kutsutaan kolmannen persoonan näkymiksi. Voidaan myös käyttää ensimmäisen persoonan näkymää, jossa pelinäkymä piirretään pelaajasta katsottuna.

3.5.2 3D Grafiikka

Teniikan kehittyessä, monet laitteet tukevat 3D-grafiikkaa. 3D grafiikaa käyttäviä pelejä onkin jo plajon markkinoilla, ja monesta pelistä tehdään erikseen 2D ja 3D versiot. 3D peleissä on myös pystyttävä optimoimaan resurssien käyttö. Tähän päästään käyttämällä malleihin mahdollisimman vähän polygoneja, tekstuureja, ja mahdollisesti uudelleenkierrättämällä elementtejä.

3.6 Prosessiin osallistuvien henkilöiden roolit

Projektin tuotantoon kootaan kehitystiimi valitsemalla vapaita henkilöitä heidän ammattitaitonsa ja kokemuksensa mukaan. Projektiin osallistuvat voidaan jakaa viiteen Andrew Rollings'n mukaan[9] jaostoon (division). Nämä ovat "johto ja suunnittelu", "Ohjelmointi", "Taide", "Musiikki ja ääni", "Tuki ja laadunvarmistus". Jokaiselle määritellään projektissa roolit ja suoritettavat tehtävät. Tilanteesta riippuen samaa roolia voi suorittaa useampi henkilö, ja yksi henkilö voi osallistua useamman roolin suorittamiseen. Tämä on varsin yleistä, koska kaikki tehtävät eivät ole selkeästi tietylle roolille määrättyjä. Koska tiimissä on usein useampi ohjelmoija ja useampi graafikko, toimii joku heistä johtavana(Lead), jolloin hän ohjaa oman jaoston toimintaa.

Tuottaja (Producer): Tuottajan tehtävä on varmistaa, että kaikki tietävät tehtävänsä ja projekti etenee sulavasti. Tuottaja pitää kiinni aikataulusta ja priorisoi työtehtävät. Tarvittaessa tuottaja voi uudelleen määritellä tehtäviä ja tavoitteita. Tuottaja katsoo myös että projekti ei lähde tarpeettomasti rönsyilemään vaan pysyy suunnitelmassa. Producerin suurinta huomiota vaaditaan itse tuotantovaiheessa, jolloin projektin tavoitteet ja etenemisaikataulu on selvillä.

Konseptisuunnittelija (Concept Designer)**:** Konseptisuunnittelija suunnittelee pelin konseptin ja miettii tulevaa ulkoasua. Tähän kuuluu myös esimerkkimateriaalin kerääminen muista peleistä arviointia varten. Konseptin suunnittelu alkaa jo ennen varsinaista projektin alkamista ja jatkuu esituotannossa.

Pelisuunnittelija (Game Designer): Pelisuunnittelija suunnittelee peliä yksityiskohtaisemmin, ja joutuu olemaan läsnä tuotannon ajan ja tarkistamaan, että toteutettu vastaa hänen suunnitelmaansa. [10] Projektin edetessä, joitakin pelisuunnittelun yksityiskohtia voidaan joutua miettimään kokonaan uudestaan, mikäli suunnitelman osa osoittautuu riittämättömäksi. Suurempiin korjauksiin on varaa, mikäli ne tehdään varhaisessa vaiheessa tuotantoa.

Konseptigraafikko (Concept Artist)**:** Konseptisuunnittelija tarvitsee konseptitaiteilijaa visualisoimaan konseptia. Projektin aloittamispäätös tehdään konseptidokumentaation pohjalta, joten konseptitaide tehdään ennen esituotantovaihetta. Konseptitaiteeseen kuuluu mm. luonnokset pelihahmoista, ympäristöistä ja tulevien peliruutujen visualisointia. Konsepti taide näyttää yleensä paremmalta, kuin todellisen pelin taide, johtuen laitteen muistin ja prosessoritehon rajoituksesta. Jos peli myydään ulkopuoliselle asiakkaalle konseptikuvien perusteella, voi asiakas pettyä, mikäli lopullinen peligrafiikka eroaa liiaksi konseptigrafiikasta. Jos konseptigraafikolla on kokemusta peligrafiikan teosta, pystyy hän paremmin arvioimaan miltä lopullinen peli voi näyttää.

Peligraafikko (Game Artist)**:** Peligraafikko on vastuussa kaiken pelissä käytettävän grafiikan tuottamisesta. Peligrafiikka tehdään käyttäen pohjana konseptitaidetta, jota graafikko käyttää apuna. Peligraafikkoa tarvitaan koko esi- ja varsinaisen tuotannon ajan, koska peligrafiikat pitää tehdä moneen eri profiiliin. Joskus graafikkoa voidaan tarvita jälkituotannossa, jos grafiikoissa esiintyy laitekohtaisia bugeja. Monesti sama henkilö toimii konsepti- ja peligraafikkona. Yleensä projektissa on vähintään kaksi graafikkoa, joista toinen toimii johtavan graafikkona(Lead Artist). Hänen tehtävänä on pitää huolta, että grafiikka on yhtenäistä ja neuvottelee johtavan ohjelmoijan kanssa grafiikoiden teknisestä toteutuksesta.

Markkinointigraafikko (Marketing Artist)**:** Markkinointigraafikko tekee näyttäviä kuvia markkinoinnin tarpeisiin, kuten julisteisiin ja web-sivuille. Markkinointigrafiikoihin kuuluu myös kuvaruutukaappaukset itse pelistä, mitä usein pitää jälkikäteen käsin hioa näyttävien pelitilanteiden luomiseksi. Markkinointigraafikko ei kuulu varsinaisesti pelinkehitystiimiin, vaan toimii markkinoinnin alaisuudessa.

Äänisuunnittelija (Sound Designer)**:**Äänisuunnittelija tekee peliin musiikkia ja ääniefektejä. Myös äänisuunnittelijan pitää tulla toimeen laitteiden eri rajoitusten kanssa, ja äänten on kuullostettava hyviltä myös vanhemmilla luureilla, joiden äänentoistokyvyt ovat hyvin rajoittuneet.

Kenttäsuunnittelja (Level Designer)**:**Kenttäsuunnittelijan tehtävä on tehdä peliin pelikenttiä. Hän pääsee tekemään kenttiä yleensä vasta projektin loppupuolella, kun tarvittavat pelin featuret ovat lyöty lukkoon ja kenttien testaamiseen tarvittava koodi ja grafiikka on tarpeeksi pitkälle kehittyneet. Kenttäsuunnittelija tarvitsee yleensä myös pitkälle kehitetyn kenttäeditorin ja muita työkaluja, jotka hän saa työkaluohjelmoijalta. Näiden kaikkien tarjoaminen kenttäsuunnittelijalle mahdollisimman nopeasti on tärkeää, jotta hän pääsee suunnittelemaan kenttiä.

Työkaluohjelmoija (Tool Programmer)**:** Työkaluohjelmoija rakentaa kenttäsuunnittelijan ja ohjelmoijien tarpeiden mukaisen kenttäeditorin, joka sisältää kenttäsuunnittelussa tarvittavat työkalut. Kenttäeditoria suunniteltaessa on taas otettava huomioon mobiililaitteen rajoitukset, jotka vaativat myös pelikenttäkoodilta tiukkaa pakkausta. Kenttädata voi olla hyvinkin monimutkaista, jolloin graafinen editori tarvitaan sen visualisoimiseen. Peliprojekteista riippuen kenttädata voi poiketa toisistaan huomattavasti, jolloin jokaiselle projektilla pitää olla oma editorinsa ja työkaluohjelmoija. Kaupallisessa levityksessä on muutamia kenttäeditoreja, mutta niiden yhteensovittaminen projektin kanssa olisi hyödyttömän työlästä, mutta usein käytetään hyväksi

aiemmissa projekteissa käytettyjä editoreja.[14]

Peliohjelmoija (Game Programmer)**:** Peliohjelmoija tuottaa pelin ohjelmakoodin. Peliohjelmoinnissa on pystyttävä tuottamaan testattavia prototyyppejä nopeasti pelinprojektin alkuvaiheessa, mutta myös ylläpidettävää koodia projektin loppuun saakka. Ohjelmoijan on myös tunnettava laitteet ja niiden rajoitukset hyvin ja pystyttävä tekemään koodia, joka pystyy käyttämään laitteiden resursseja tehokkaasti. Peliohjelmoijan on myös rakennettava peli helposti portattavaan muotoon. Koska mobiilikehitystiimi on pieni, huolehtii johtava peliohjelmoija myös yleisesti projektin arkkitehtuurin suunnittelun, eikä erikoistunutta arkkitehtiä(Architect) tarvita.

Kehitysalustaohjelmoija (Paltform Programmer)**:** Koska eri laitteilla on jokaisella hieman eri toimintatavat, on oltava kirjasto yksinkertaisista rutiineista. Kirjastossa on rutiini resurssien lataamiselle, äänten soittamiselle ja grafiikan piirtämiselle. Kirjastolla on yleinen API(Application Programming Interface, Sovellusohjelmointikäyttöliittymä) peleille ja erilaiset implementoinnit erilaisille laitteille. Tälläisen klirjaston käyttäminen helpottaa huomattavasti jälkituotantovaihetta. Kirjaston kehittäminen on alustaohjelmoijan. Alustaohjelmoija voi tämän lisäksi myös ohjelmoida alustan osana työkaluja, jotka auttavat yleisesti pelin ohjelmoinnissa tai grafiikan tuottamisessa. Erotuksena työkaluohjelmoija tekee työkaluja kyseistä peliä varten.

Peli-insinööri (Game Engineer)**:** Peli-insinööri ei yleensä itse ohjelmoi, vaan kääntää versiot eri laitteille jälkituotannossa peliohjelmoijan tekemistä muutamisea referenssiversioista. Peli-insinööri ei tee uusia ominaisuuksia peliin, mutta voi joutua tekemään asiakkaan vaatimia lisäyksiä.

Testaaja (Tester): Mobiilipelit vaativat paljon teknistä testaamista, koska melkein joka laite vaatii omanlaisensa version. Pelissä voi tulla yllättäviäkin ongelmia vastaan, joten peli on testattava huolella. Pääsääntöisesti testaaja käy läpi tietyt ennalta suunnitellut ongelmakohdat ja varmistaa, että peli toimii virheettä. Tekninen testaaminen kiihtyy tuotantovaiheen loppuvaiheessa, ja jatkuu jälkituotannossa.[6] Mobiilipeleissä ei voida jättää bugeja valmiiseen tuotteeseen, koska ei ole mahdollisuutta lähettää loppukäyttäjälle myöhemmin korjauspaketteja, kuten PC-peleissä.

4 MOBIILIPELIN TEKNINEN TOTEUTUS

4.1 Mobiilialustan tekniset rajoitteet ja eroavaisuudet

Mobiilialustoille tehdyissä sovellutuksissa yksi suurimpia haasteita on mobiililaitteiden laaja kirjo. Mobiililaite on tässä yhteydessä mikä tahansa mukanakulkeva laite, johon voidaan ladata ja ajaa viihdesovellutuksia. Käytännössä nämä laitteet ovat matkapuhelimia, joihin voi tilata sovellutuksia, mutta tulevaisuudessa myös kannettavat pelikonsolit, kämmentietokoneet ja soittimet ovat mukana, joskin näihinkin yhdistyy puhelin ominaisuus, joten raja voi näiden osalta hämärtyä. Mobiililaitteita tuottavat useat valmistajat, jotka julkaisevat vuosittain lukuisia malleja erinäisissä tehokkuus ja hintaryhmissä. Laitteen ominaisuuksia voidaan myös painottaa joltain osalta jonkin toisen kustannuksella oletetun käyttötarkoituksen ja valmistajan mieltymysten mukaan.

4.1.1 Firmware

Yksittäisen mallin sisälläkin voi olla suurta vaihtelua. Laitevalmistajien tuotannossa aikaisemmat

kappaleet saavat usein laiteohjlemiston (”*firmware*”) joka on puutteellinen tai jopa virheellinen. Laiteohjelmistosta tehdään korjattuja versioita jotka ovat myöhemmin ilmestyneissä laitteissa, ja jotka käyttäjä voi itse asennuttaa myöhemmin vanhemman tilalle. Tällöin voi esiintyä riski, että vaikka ohjelmisto toimisi hyvin uudemman firmwaren laitteilla, niin käytössä voi olla paljon vanhemman firmwaren laitteita, joissa sovellus ei toimi. Monet operaattorit myös myyvät itse puhelimen puhelinliittymän mukana, jolloin puhelimeen on asennettu operaattorin oma käyttöjärjestelmä. Päätelaitteesta itsestään riippumattomana erottavana tekijänä toimivat monille kielille käännettävät versiot.

4.1.2 Sovellutusten muoto ja yhteensopivuus

Mobiililaitteille ladattavat sovellutukset voivat olla monessa muodossa. Ne voidaan ohjelmoida vaikka C++ kielellä ja kääntää suoraan laitteen omalle natiivikielelle, jolloin pelistä saadaan tehokkaampi, mutta käyttöympäristö on rajattu vain kyseiseen laitteeseen tai sille yhteensopiviksi suunniteltuihin laitteisiin. Puhelimeen valmiiksi ladatut ohjelmat ja pelit ovat yleisesti tehty laitteen natiivikielellä. Vaihtoehtoina tälle on kehitetty laitteistoriippumattomia ohjelmointikieliä, jotka vaativat yhteensopivalta laitteelta java-virtuaalikoneen. Euroopassa ja Aasiassa matkapuhelinkanta tukee ladattavia Java-sovellutuksia, kun taas Yhdysvalloissa BREW-laitteet ovat yleisempiä. Myös harvinaisempia alustoja on kehitetty, kuten Adoben Flash Lite, joka on Adobe Flash Playeristä mobiililaitteille optimoitu versio.

4.1.3 suoritusteho

Päätelaitteet toisistaan erottavat rajoitukset liittyvät suoritustehoon, muistin kokoon, näytön kokoon, näppäimistöön, äänentoistoon ja laitekohtaisiin bugeihin. Laitteen suoritusteho on pääasiassa seurausta prosessorin tehosta, mutta siihen voi myös vaikuttaa laitteen muut ominaisuudet, kuten muistin määrä, firmwaren ja java-virtuaalikoneen ohjelmoinnissa tehdyt ratkaisut ja mahdolliset bugit. Jotkin laitteet voivat suorittaa tiettyjä toimintoja nopeammin, kuin toiset laitteet, mutta vuorostaan suorittavat hitaammin muita toimintoja. Esimerkkinä alueen peittäminen yhdellä värillä voidaan suorittaa joko käyttämällä piirrossa valmista PNG-kuvaa, tai ohjelmallisesti käyttämällä Javan omaa ”fillrect”-käskyä. Joillakin laitteilla näiden tapojen välille voi muodostua suuriakin eroja. Laitteen suoritusteho vaikuttaa siihen, kuinka monimutkaisia toimituksia peliohjelma voi suorittaa pelin aikana. Mobiilipeleissä itse pelimekaniikka on yleensä suunniteltu niin kevyeksi, että sitä voidaan pyörittää hitaammillakin laitteilla ilman huomattavaa hidastumista. Nopeammilla laitteilla käytetään samaa pelimekaniikkaa, mutta nopeampi laite mahdollistaa monimutkaisemman grafiikan piirron ja lisätyt visuaaliset tehokeinot, joilla pelistä saadaan näyttävämpi. Kun visuaaliset tehokeinot ja grafiikan monimutkaisuus on jaoteltu helposti lisättäviin ja poistettaviin osiin, voidaan näiden määrää säätämällä pelistä saada sulavasti pyörivä, mutta laitteen tehot hyödyntävä kokonaisuus.

4.1.4 Latauspaketit

Laitteen suoritusteho on myös riippuvainen laitteen muistista. Mobiilipeliä kehiteltäessä on otettava huomioon kahdenlaista tilankulutusta, Jar-paketin kokoa ja heap-muistin käyttöä. Java-pelit ladataan käyttäjän laitteeseen pääasiassa tilaamalla asennettava Jar-paketti, joka sisältää tilatun pelin zip-pakattuna. Jar paketissa pelin grafiikat ovat pääasiassa png-kuvatiedostoina, joista käytettävät kuvat ladataan. Jar-paketin koko on puolestaan rajoitettu puhelimen mallin mukaan, jolloin joillekkin puhelimille voidaan toimittaa satojen kilotavujen tiedostoja, kun pienimmillään

toimitettavan paketin koko on rajattu vain 65 kilotavuun. Vaikka paketista karsittaisiin ohjelmakoodia, pelikenttiä, tekstiä, voi grafiikoille jäädä vain muutama kilotavu tilaa paketista. Tällöin pakettiin täytyy jättää vain kaikkein tärkeimmät elementit. Kun elementtien poistaminen ei riitä, voidaan kierrättää samoja graafisia elementtejä useampaan tarkoitukseen. Osan grafiikoista voidaan korvata ohjelmallisesti piirrettävillä geometrisillä elementeillä, kuten pisteillä, viivoilla, neliöillä ja kaarilla. Koska grafiikat tulevat yleisesti PNG tiedostomuotoisena pelipakettiin, niin png-pakkauksen ansiosta vähentämällä kuvasta värejä ja yksityiskohtia, voidaan kuvien tilanvientiä vähentää. Joskus joudutaan tekemään hyvinkin radikaalia karsintaa. Kuva-dataa voidaan joissain tilanteissa hyödyntää pidemmällekin, esimerkiksi kuvaan voidaan sisällyttää useampi värimäärittely, jolloin sama kuva voidaan ladata muistiin eri värisinä.

Suurissa latauspaketeissa muodostuu ongelmaksi myös operaattoreiden laskuttama hinta kuluttajalle, ja hidas tiedonsiirto pakettia ladatessa. Monesti yhteys voi katketa kesken latauksen, mikä syö kuluttajan luottamusta mobiilipeleihin.

4.1.5 Heap-muisti

Toinen muisti, jonka käyttöä on tarkkailtava, on pelin aikana käytettävä heap-muisti, jonne pelikoodin ja muun datan lisäksi ladataan kuvat pakkaamattomina suorittamisen ajaksi. Samoin kuin jar-paketin koko, myös heap-muistin koko asettaa huomattavat rajoitteet lowend-laitteille. Koska heap-muistissa oleva tieto on purettuna, niin png- ja zip-pakkauksilla säästetty tila menettää merkityksensä, ja ladatut kuvat vievät pinta-alansa verran muistia. Värejä ja yksityiskohtia karsimalla ei voida yleisesti säästää heap-muistia. Jos yhteen kuvaan sijoitetaan ladattavaksi useampi sprite, täytyy spritet järjestellä kuvaan niin, ettei kuvaan jää käyttämätöntä tilaa, joka syö heap-muistia. Laajempi heap-muisti myös mahdollistaa lyhyemmät latausajat, kun data voidaan ladata kerralla muistiin, eikä samaa tieto tarvitse jatkuvasti ladata uudestaan.

4.1.6 Näyttö

Peligrafiikkaa esitettäessä on grafiikka sovitettava erikokoisiin näyttöruutuihin. Näyttöjen vaihtelevuus on iso ongelma grafiikalle. Pienimmillään näytön koko vai olla 96*65, kun taas suurimmissa näytöissä koko voi olla vaikka 800*400. Vanhoissa laitteissa näytöt ovat lisäksi naarmuisia, pölyisiä, rasvaisia ja valovoimaltaan heikkoja, joten grafiikan on oltava selkeää, mutta myös viimeisteltyä ja pikkutarkkaa. Huonolaatuisten näyttöjen päivitysnopeus voi olla hyvin hidas, jolloin liikkuvat grafiikat jättävät jälkeensä pitkän haamukuvan. Pienissä näytöissä pelaaja voi saada esimerkiksi pelihahmosta väärän käsityksen, kun suuremmilla resoluutioilla kuvaaminen on helpompaa. Yleisesti näyttöruutu on pystyssä, kuten yleinen 176*208, mutta mallista riippuen kuvasuhteet vaihtelevat. Vaihtelevat näytönkoot vaativat peligrafiikoiden tuottamisen useampaan eri kokoluokkaan, joista voidaan valita kullekin näytölle sopivat grafiikat. Joskus voidaan laitteelle valita kokoluokkaan nähden pienemmät grafiikat, jos muistissa ei ole tilaa kyseiselle koolle tarkoitetuille grafiikoille. Tästä kuitenkin voi seurata ongelma, jos näytettävä pelialue kasvaa, ja pelialueen piirtämisessä joudutaan kutsumaan piirtokomento niin monta kertaa, että pelin ajaminen ei onnistu hidastumatta. Tällöin piirrettävää pelialuetta voidaan lisäksi rajoittaa staattisilla elementeillä, kuten ylä- ja alapaneeleilla. Vanhemmissa malleissa on myös rajoituksia värien kanssa. Vanhat näytöt pystyivät näyttämään vain 12 bittistä väriavaruutta, kun uudet näytöt pystyvät näyttämään täydet 24 bitin väriavaruuden. Tällöin 24bittiseen avaruuteen määritellyt värit joko pyristetään lähimpään suurempaan, tai pienempään 12bit arvoon, tai väriarvo ditheröidään käyttäen lähimpiä värejä.

4.1.7 Grafiikkatiedostot

4.1.7.1 PNG-Formaatti

J2ME:n tukema kuvaformaatti on PNG. Laitteet tukevat PNG-formaattia vaihtelevasti. Esimerkiksi joissain laitteissa voi olla 256 sävyinen alpha-kanava läpinäkyvyydelle, muissa yhden bitin läpinäkyvyys. Alpha-kanavalla voidaan piirtää kuvia, joissa on vaihteleva osittainen läpinäkyvyys, kuten pilviä, varjoja ja heijastuksia. Laitteet eivät kuitrenkaan tuo formaattia aina oikein, ja joillekkin laitteille kuvaa täytyy erikseen muotoilla, että se toimisi ongelmitta. MIDP1.0 ja MIDP2.0 määrittelevät, että laitteiden on tuettava myös alpha kanavaa läpinäkyvyyksille, mutta osittainen läpinäkyvyys on optionaalinen. Läpinäkyvät kuvat ovat muuttumattomia(immutable), eli niiden pikselidataan voida lataamisen jälkeen puuttua.[11,12]

Alimmillaan heap voi olla 64kb, ja koodi 25-40 kb ja muu data 10-20kb, niin grafiikalle jää vähän tilaa. Joissain heap-muistia voi grafiikalle olla vain 128kb, niin purettua grafiikkaa mahtuu 4*screensize. Oikeiden grafiikkaprofiilien valinta vaatii siis laitteiden hyvää tuntemusta. Tähän auttaa laaja tietokanta laitteiden ominaisuuksista.

PNG tiedostoformaatti tukee monenlaisia väriformaatteja. Käytetyin on indeksoitu paletti (indexed palette) jossa kuva muodostuu maksimissaan 256 erivärisestä pikselistä, joiden väriarvo koostuu kolmesta värikanavasta, jotka kukin on merkitty kahdeksalla bitillä, eli voivat saada 256 arvoa. Läpinäkyvyys on yleensä merkitty vain yhdellä bitillä, jolloin pikseli on joko täysin peittävä, tai läpinäkyvä. Läpinäkyvyyskanava voi kuitenkin saada myöskin 8bittisen arvon, jolloin kuvassa voi olla osittaisia läpinäkyvyyksiä. Mikäli kuva on suurempi ja väreiltään monisävyinen, voi 256 väriä olla liian vähän, jolloin voidaan tallentaa suoraan 24-bittiset väriarvot jokaiselle pikselille erikseen. Tällöin kuvan tilanvienti kasvaa, ja monet vanhemmat laitteet eivät pysty käsittelemään tämän kaltaisia kuvia, tai piirtävät ne väärin[13].

4.1.7.2 Jpeg-Formaatti

Jpeg-kuvaformaatti on häviölliseen pakkaukseen soveltuva formaatti. Erityisesti tästä on hyötyä, kun pakataan suuria valokuvamaisia kuvia, joissa on paljon sävyjä ja vaihtelevaa tekstuuria. Jpeg-tukea ei ole kovin kattavasti dokumentoitu, mutta havaintojen mukaan suurin osa laitteista tukee formaattia, mutta vanhemmissa laitteissa tukemisen varaan ei voida laskea. Tämä ei sikäli ole ongelma, koska isommat Jpeg-kuvat soveltuvat muutenkin vain tarpeeksi suurinäyttöisten, ja suurimuististen laitteiden käyttöön.

4.1.7.3 Laitteiden grafiikankäsittely bugit

Laitteiden omat bugit on myös otettava huomioon grafiikassa. Useimmiten ongelmia ilmenee kuvien värien piirrossa, erityisesti läpinäkyvyyksien huomioimisessa. Jotkin laitteista voivat esimerkiksi piirtää kaikki mustat alueet läpinäkyviksi, tai kaikki valkoiset alueet läpinäkyviksi, jolloin kyseisille laitteilla ei voi käyttää niitä värejä. Pahimmissa tapauksissa muistiin ladattu kuva voi vaikuttaa toisen muistissa olevan kuvan piirtoon, tai laite voi kaatua, mikäli muistissa on yli 256 pikseliä leveä tai korkea kuva. Laitteet voivat piirtää väärin myös hyvin yksinkertaisia komentoja, kuten kaarenpiirtoa.[13]

Pelissä tekstien kirjoittamiseen voidaan käyttää joko laitteen omaa fonttia, tai sisällyttää toimituspakettiin peliä varten tehty fontti. Laitteen omaan fonttiin luotettaessa, pitää varautua

siihen, että eri laitteiden fontit voivat poiketa toisistaan paljonkin. Suurin ongelma on fonttien poikkeavat koot leveydessä ja korkeudessa ja mahdollisesti myös luettavuudessa. Tästä johtuen fontin koon vaihtelu on otettava suunnittelussa huomioon. Jos fontti tehdään itse, voidaan saada aikaan kaikilla puhelimilla samannäköistä tekstiä, ja ennen kaikkea näyttävämpää ja pelin ulkoasuun sopivaa tekstiä. Omalla fontilla voidaan paremmilla laitteilla käyttää myös osittaista läpinäkyvyyttä, jolla saadaan moniväriselle fontille antialiasoinnit ja varjot. Oma fontti tietenkin vie tilaa muistissa, joten sitä ei voida käyttää alempitasoisimmissa laitteissa. Kun tekstiä näytetään jo valmiiksi pienellä ruudulla, on otettava huomioon tekstin vaatima alue, jolla tekstin on pysyttävä. Lisäksi peli käännetään useille kielille, joissa sanapituudet vaihtelevat. Ratkaisuna tekstit pyritään pitämään mahdollisimman lyhyinä, ja teksti sovitetaan dynaamisiin tekstilaatikoihin. Itse pelialueella ei kannata käyttää tekstiä, mikäli fontin, tai kielen vaihtuminen aiheuttaa ongelmia. Tällaisissa tilanteissa turvallisempaa on käyttää sanan korvaavia graafisia ikoneita

4.1.8 Ohjaus

Laitteiden näppäimet ovat usein standardissa 3*4 ruudukossa, jonka lisäksi kaksi soft-key nappia. Joukossa on kuitenkin muutamia poikkeuksia, kuten Nokian 3650, jossa näppäimet on sijoitettu rinkiin. Jotkin laitteet voivat olla kokonaan ilman numeronäppäimiä, ja toimia kosketusnäytön kautta, tai niissä voi olla näppäimet jokaiselle kirjaimelle. Tästä syystä kontrollit ovat pidettävä mahdollisimman yksinkertaisina. Näppäimiä ei kaikissa puhelimissa voi myöskään painaa useaa yhtä aikaa, tai näppäimistön viive on niin suuri, että pelaajan on mahdotonta reagoida tarpeeksi ajoissa pelin tapahtumiin.

4.2 Rajoituksiin sopeutuminen

Pelin optimoinnissa on otettava huomioon, että yhden osa-alueen säätäminen vaikuttaa monesti myös toisiin osa-alueisiin. Jos muistia koitetaan säästää piirtämällä pelialue pienemmillä elementeillä ja koostamalla hahmoja useammalla palalle, voi ajon aikaiset piirtokomennot kasvaa jyrkästi, mikä hidastaa pelin toimintaa.

Näiden rajoitteiden ennalta tunteminen ja niihin varautuminen auttavat pelin kääntämisessä tehokkaasti kaikille laitteille ja toiminnan testaamisessa, mitkä ovat yksiä alan suurimpia haasteista. Tämä tuotteen skaalaus on kuitenkin tehtävä laitteiden ominaisuuksia tehokkaasti hyödyntäen, mutta kuitenkin kallisarvoista ihmistyötä säästäen. Käytännössä pelistä tehdään muutamalle referenssilaitteelle versiot, joista käännetään automaattisesti versiot muille laitteille. Tämän prosessin automatisoinniksi tehokkaaksi prosessiksi täytyy hyödyntää tehtävää varten kehitettäviä työkaluja. Lisäksi vanhojen toteutusten uudelleen kierrättäminen säästää työtunteja merkittävästi.

4.2.1 Heap-muistinkäytön optimointi

Peligrafiikat kootaan tiedostokoon säästämiseksi yleisesti muutamaan isompaan PNG-kuvaan, joista voidaan valikoida yksittäisiä grafiikkaelemnttejä ruudulle piirrettäväksi.Tällaisesta kokoelmakuvasta voidaan käyttää nimitystä ”sprite sheet”. Tämä keino säästää png-tiedostojen määrittelytietojen turhalta toistolta. Voidaan myös käyttää monimutkaisimpia kuvanpakkausmenetelmiä. Helpointa spritesheetin muodostaminen on, jos käytetyt grafiikat ovat yhteneviä muodoiltaan, mieluiten nelikulmioita. Näin saadaan koottua grafiikat muotoon, jolloin ne eivät vie turhaa tilaa JAR-latauspaketissa, tai puhelimen heap-muistiin ladattuna. Mikäli grafikkaa on eri kokoisena ja eri muotoisena, on niiden tallentaminen samaan sprite sheet kuvaan hankalaa.

Viimeistään suoritustehoiltaan heikoimmille laitteille luotaessa grafiikkaa parhaiden laitteiden grafiikoiden pohjalta, joudutaan sopeutumaan heap- ja jar-tilan loppumiseen. Graafikko joutuu pienentämään peligrafiikoiden tilanvientiä monin keinoin. Helpoimmat tavat tässä on vähentää grafiikoiden määrää, kokoa, värejä ja yksityiskohtia. Kaikista heikoimmilla laitteilla voi tuloksena olla dramaattinen tasonlaskeminen spritejen osalta ja pitkällinen pikselitason pelkistäminen jotta viimeiset kymmenet tavut saadaan poistettua.

Spritejen pinta-alan pienentäminen vaikuttaa merkittävästi heap-muistin käyttöön, koska PNG-tiedostosta muodostetaan ajon aikana muistiin bitmap kuva. Mikäli spritesheet kuvassa on paljon erikokoisia spritejä, voi niiden asettelu olla hankalaa, ilman että väliin jää tyhjää tilaa. Tämän helpottamiseksi kannattaa olla työkalu, joka asettelee automaattisesti spritet tiiviisti toisiinsa. Spritejen asettelussa voidaan myös miettiä grafiikan lukemista päällekkäisiltä alueilta. Yleensä kappaleiden reunoilla oleva tyhjä alue voidaan sovittaa päällekkäin viereisen tilen kanssa. Joissain maastotileissä voi olla yhtenäisten kuvioiden alueita, jotka sopivat päällekkäin isommaltakin alueelta, ja peligrafiikat voidaan suunnitella tämä mielessä pitäen.

Spritet kannattaa koota isoiksi sheeteiksi, jolla saadaan hieman pienennettyä grafiikan muistinkulutusta. Tästä kuitenkin pitää tehdä poikkeus niiden grafiikoiden osalta, joita käytetään vain yhdessä kentässä. Ei ole tarpeen ladata muistiin sellaista grafiikkaa, jota ei tulla kyseisessä pelikentässä näyttämään. Osa grafiikasta voidaan erottaa omakseen, jos sitä tarvitaan useammassa kentässä, ja monesti kannattaa jakaa omille sheeteille sellainen grafiikka, joka koostuu samoista paletin väreistä. Tällä voidaan säästää pari sataa tavua. Lisäksi omissa sheeteissään olevat loogiset kokonaisuudet ovat profiilien hallinnan kannalta selkeämmät ja helpottavat tilannetta, jossa voidaan säästää muistia vaihtamalla tilalle joitain kevyemmän profiilin sheettejä.

Peligrafiikat voidaan myös piirtää vähemmistä resursseista, kokoamalla spritet pienemmistä spriteistä. Tämä pätee varsinkin pelihahmojen animaatioihin. Yksinkertaista on animoida vain pelkät pelihahmon jalat ja piirtää tämän päälle yksi staattinen ylävartalo. Jos hahmo liikkuu kahdeksaan suuntaan neljän framen syklillä, voidaan tuon hahmon jalat irrottaa torsosta, jolloin torson grafiikoista on säästynyt 7*3, eli 21 torsoa. Näitä elementtejä voidaan käyttää myös muissa hahmoissa, jos pelaajan vastustajatkin käyttävät samoja jalkoja, voidaan säästää myös vastustajan jalkojen grafiikat. Pelispriten optimaalinen muoto on nelikulmio, koska silloin reunoihin ei tuhlaannu turhaa tyhjää tilaa. Jos pelispriten saa jaettua kahteen pienempään spriteen ja säästää näin tyhjältä tilalta, kannattaa se tehdä. Liian pitkälle ei kuitenkaan jakamista kannata viedä, jos siitä ei ole selviä hyötyjä. Liian monen osan piirtäminen ruudulle voi olla laitteelle liian raskasta ja on otettava huomioon, että kuvien generoimiseen tarvittava ohjemalogiikka vaatii oman tilansa. Sekavaa piirtohierarkiaa on vaikeampi ylläpitää tai muuttaa ja on lisäksi hyvin altis bugeille. Lisäksi sen laatiminen on aikaa vievää niin graafikolta, kuin ohjelmoijaltakin. Tarkoitukseen sopivien työkalujen käyttämisellä pystytään torjumaan näitä ongelmia. Tällaista optimointia joutuu tekemään monesti heikomman suoritustason puhelimille, mutta laitekannan uudistuessa, jäävät nämä ongelmat vähemmälle huomiolle. Grafiikoita voidaan myös käyttää usemapaan kertaan samassa kohteessa. Tällä tekniikalla usein piirretään HUDeja ja tekstilaatikoiden reunoja.

Kun JAR paketti on mahdollisimman tehokkaasti pakattu, pitää grafiikan sisältöä rajata, jos muistinkulutuksen rajat tulevat vastaan. Mikäli grafiikoissa on paljon värejä, näiden karsiminen on helppoa ja tehokasta. Tässä hyödyllisin apuväline on Adobe Photoshpin ”*Save for Web*”-toiminto, jolla värien vähentäminen on helppoa. Mikäli väriä vähennetään paljon voidaan joutua grafiikoita korjailemaan käsin. Jos tämä ei riitä, voidaan yksityiskohtia karsia maalaamalla kuviin suuria neliömäisiä värialueita, jotka pakkaantuvat hyvin. Tälläisiä keinoja voivat olla vaikka taivaalta

pilvien poistaminen ja sen värittäminen yksiväriseksi. Mikäli värialueiden välissä on rasterointia, voi senkin poistaa, joskin myös toistuvat kuviot pystyvät pakkaantumaan.

Mikäli tallennettava grafiikka vaatii yli 256, voidaan se tallentaa PNG24-muodossa, jolloin sen tilantarve kasvaa huomattavasti. Jos mahdollista, voidaan kuva jakaa ja tallentaa pienemmissä osissa, jolloin yhdelle osalle riittää 256 väriä. Jos kuvassa on paljon sävyjä, kannattaa se tallentaa Jpeg formaatissa, jolloin kuvan pakkaussuhde paranee huomattavasti ja kuvaan saadaan enemmän sävyä. Yleisesti 176*208 kokoinen Jpeg kuva vie noin puolet vähemmän muistia, kuin samantasoinen PNG-kuva. Jpeg formaatin rajoitus on, ettei se tue läpinäkyvyyttä. Jpeg formaatti ei kuitenkaan kuulu MIDP määrittelyn standardeihin, joten varsinkaan vanhempien laitteiden tukemiseen ei voida luottaa. Mikäli JPG kuvia haluaa käyttää, on pystyttävä määrittelemään helposti laitteille vaihtoehtoinen PNG-kuva, tai sisällytettävä ohjelmakoodiin metodit, joilla JPG kuva voidaan muuttaa PNG kuvaksi ladattaessa. Tällainen monimutkaistaa prosessia, ja ei välttämättä ole saavutetun hyödyn arvoista.

4.2.2 Tiedostojen optimointi

4.2.2.1 JAR-tiedostot optimointi

JAR-latauspaketti käyttää ZIP-pakkausta, jolloin peli voidaan toimittaa pienemmässä muodossa, kuin se vie kännykkään purettuna. ZIP pakkaus toimii tehokkaasti toistuvien merkkijonojen pakkauksessa ja tehokkaammin isoilla tiedostoilla, kuin monella pienellä. Tätä voidaan käyttää hyväksi suunnitellessa pelin data-rakennetta. On kuitenkin pidettävä mielessä, että jos tiedosto saadaan tehokkaasti JAR-pakettiin pakattua, ei se vie välttämättä yhtään vähempää tilaa heap-muistissa. Myös itse ZIP-pakkauksen tehokkuutta voidaan parantaa sopivilla pakkausohjelmilla. On olemassa myös kaupallisia sovellutuksia, jotka suorittavat koko J2ME paketin optimoinnin monilla tasoilla. Suosittuja ohjelmia ovat *mBooster*, *ProGuard*, ja *retroGuard*. Valmista JAR pakettia voidaan myös koittaa pienentää sopivilla ohjelmilla, kuten *Kjar* ja *AdvCOMP*.

4.2.2.2 PNG-tiedostot optimointi

PNG-tiedostomuoto antaa hyvät mahdollisuudet erilaisille väritarkkuuksille ja häviämättömään pakkaukseen.

Paketin kokoa ajatellen suurin hyöty on png-pakkauksen optimoinnista, varsinkin paletti-formaatissa. PNG formaatti käyttää DEFLATE-algoritmia paakauksessa, joka on sama, kuin JAR paketin käyttämä ZIP pakkauksessa. PNG kuva ei siis pienene enää pakattaessa toiseen kertaan latauspakettiin. Tilanteessa jossa PNG:n sisäinen pakkaus suoritetaan jollain kevyellä algoritmilla, kuten Huffman-pakkauksella, kasvaa lopullinen JAR paketti jopa suuremmaksi, kuin jos PNG tiedostot olisi tallennettu kokonaan ilman pakkausta.

Yksi vaihtoehto on suoraan purkaa png-kuva dataksi, joka sitten ajon aikana kootaan laitteen muistiin ehjäksi png-tiedostoksi. Etuna tässä on, että png-kuvaan voidaan luoda grafiikkaa peilaamalla ja kääntämällä tai muuten manipuloimalla grafiikkaa. Tämä on erityisen kätevää pelihahmojen kohdalla, kun puolet hahmoista voidaan luoda peilaamalla. Lisäksi voidaan myös palettidataa käsitellä jolloin samaa kuvaa voidaan käyttää eriväriseną. Tilanteissa joissa datapaketissa on monia yksittäisiä kuvatiedostoja, voidaan saada jopa kymmenien prosenttien pakkaushyöty pelkästään liittämällä tiedostoja yhteen. Tämä selittyy sillä, että kun pakattavana on suurempia tiedostoja, pystyy pakkausalgoritmi löytämään enemmän toistuvia merkkijonoja, ja

pakkaamaan nämä pois. Pienempiä datamääriä low-end laitteissa pakattaessa hyöty voi jäädä alle prosenttiin.

4.2.2.3 JPG-tiedostojen optimointi

Jpeg pakkauksen kohdalla ei eri ohjelmien suhteen ole juuri havaittavaa eroa. Osaltaan tämä johtuu siitä, että Jpeg kuvien laadun arvioiminen silmämääräisesti on hankalampaa, kuin PNG kuvien. Yleisesti ottaen Adobe PhotoShopin ”Save for web” tallentaa hyvin pakattua Jpeg formaattia, parempaa kuin suoraan tiedostoksi tallennettaessa. On olemassa myös erikoistuneita ohjelmia, joilla voidaan kontrolloida tarkemmin pakkauksen laatua ja metodia. Jpeg pakkauksessa kuvan värit jaetaan tummuus- ja punaiseen ja siniseen värikomponenttieihin (YCbCr). Näistä kolmesta komponentista ihmissilmä erottaa tarkimmin tummuusasteen, jonka jälkeen värikomponenteista punaisen paremmin, kuin sinisen. Tätä hyödyntämällä komponentit voidaan pakata eri tarkkuudella (”subsampling”) ja vaikuttaa kuvan visuaaliseen laatuun. Kuvissa, joissa on paljon tummuusvaihtelua pakkaantuvat huomattavasti paremmin subsampling arvoilla 4:2:2, tai 4:1:1. Kuvat, joissa on laajoja yksivärisiä pintoja, tarkkoja yksityiskohtia, kuten tekstiä ja värivaihtelua ilman tummuusvaihtelua saavuttavat paremman visuaalisen tason 4:4:4 subsampling arvolla. Jpeg kuvan yleiseen pakkaantuvuuteen voidaan vaikuttaa myös esikäsittelemällä kuvaa sopivilla filttereillä, kuten pehmennyksellä, kohinanpoistolla ja kehittyneemmillä yksityiskohtien poisto-filttereillä. Jpeg kuvia voidaan myös yhdistää yhdeksi isoksi tiedostoksi, jolloin voidaan saada muutama prosentti säästettyä JAR paketin koossa. Myös JPG kuvia kannattaa koota yhteen isompaan kuvaa, koska tällä voi säästää myös jonkin verran.

4.2.3 Grafiikanpiirron optimointi

Grafiikkapiirrossa nopeimman suorituksen saa, kun piirtää mahdollisimman vähillä piirtokäskyillä. Toinen tapa on piirtää pinta-alaltaan pienenmpiä spritejä ja välttää osittaisia läpinäkyvyyksiä. Puhelimien kesken ilemenee tämän suhteen eroja. Näissä muodostuu ongelmia, jos ne joudutaan piirtämään usein. Staattisia elementtejä,kuten HUD-paneeleja voi piirtää monimutkaisesti, jos sitä ei tarvitse uudelleenpiirtää joka tilanteessa. Itse pelialueella tapahtuva liike joudutaan usein piirtämään kymmeniä kertoja sekunnissa. Tätä voidaan helpottaa erilaisilla puskurointitekniikoilla, joissa pelialue piirretään vain kerran muistiin, josta se luetaan näyttömuistiin. Muita keinoja on kasvattaa tilekokoa, niin että niitä tarvitsee piirtää vähemmän ruudun täytteeksi, tai jättää osan ruutua täyttämättä ja korvata sen levennetyllää HUDilla, tai muulla staattisella elementillä.

4.2.4 Kenttädatan optimointi

Leveldata voi pelistä riippuen viedä merkittävän osan heap-muistista ja JAR-paketista. Joissain tilanteissa tätä tilanvientiä voidaan oleellisesti pienentää järkevällä data-rakenteella, joka pakkaantuu hyvin JAR-paketissa. Kovin monimutkaisia datan järjestelyillä leveldatan purkaminen voi olla laitteelle liian raskasta, tai liian työlästä toteuttaa. Monimutkainen leveldata voi myös aiheuttaa ongelmia eri laiteprofiilien kanssa, jos heikompitehoisille laitteille käytetään karsittua kenttädataa. Monesti kuitenkin voidaan luottaa JAR-paketin pakkaustehoon, koska vain vähän pakattu leveldata voi viedä enemmän JAR-tilaa, kuin täysin pakkaamaton data. Esimerkiksi deltakoodattu leveldata yleisesti monesti kasvattaa datan kokoa. [14] Tämä pätee myös PNG-kuvien kohdalla.

Pelikentät muodostetaan monesti useammasta päällekkäisestä kerroksesta(layer). Näillä datan luonne voi olla erilainen. Alemmassa voi olla esimerkiksi maastotyyppi ja päällimmäisessä varjoja

ja lisäksi vielä kerros, jolle on aseteltu objekteja, laukaisimia ja muuta dataa. Joskus kerroksia voidaan piirtää päällekkäin ja vierittää eri nopeuksilla (parallax scrolling), jolloin ne näyttävät myös käyttäjälle olevan eri syvyystasoilla.

Karttadataan tallennettavien tilejen määrää voidaan pienentää käyttämällä metatilejä. Metatilet ovat useammasta tilestä muodostettuja kokonaisuuksia, joita voidaan käsitellä yksittäin. Kartta voidaan jakaa esimerkiksi 3*3 ruudun alueisiin, jotka tallennetaan omaksi metatilekseen, ja jota kutsumalla saadaan yhden tilen muistinkulutuksella tallennettua 9 tileä. Metatilestä pitää tietenkin tallentaa yksi kokonainen malli, mutta seuraavat voidaan piirtää 1/9 muistinkulutuksella.

4.2.5 Parametrinen ja simuloitu sisällöngenerointi

Peliin voidaan generoida monella eri tasolla sisältöä. Parametrinen generointi tarkoittaa sitä, että jokin asia ei ole kehitystiimin itsensä tarkkaan muotoilema, vaan ohjelma generoi sen itse annettujen parametrien mukaan, tai jonkin simulaation tulosta. 3D-ympäristössä ovat monet tällaiset tekniikat käytössä. Varsinkin demoskene on käyttänyt pitkään proseduraalista generointia, saadakseen pieneen tilaan mahtumaan paljon sisältöä. Parametriset materiatekstuurit eivät käytä valmista kuvaa, vaan pintatekstuuri mallinnetaan tietyn kaavan mukaan, jolle on annettu suunnittelijan määrittelemät parametrit. Vastaavasti monet 3d-maastot ovat generoituja suunnittelijan antamien suuntaviivojen mukaan, ja näyttävät hyvin realistisilta. Peleissä voidaan nähdä sisällön automaattista generoinnin mahdollisuuksia hyvin laajasti.

Parametrisen generoinnin etuja ovat sen monistettavuus laajoille alueille ja muuttuviin tilanteisiin. Tällöin sisältä myös muistuttaa luonnollista ja on ihmismieleen vetoavampi. Monessa pelissä voidaan uusia pelimaastoja generoida lähes loputtomasti, jolloin pelin uudelleenpelattavuus säilyy. Haittapuolena voidaan nähdä vaikeus säätää parametreja ja generointilogiikkaa pelisuunnittelijan mieleiseksi. Proseduraalista generointia hyväksikäyttäen voidaan näyttäviä 3d pelejä pakata hyvin pieneen tilaan. Hyvänä esimerkkinä Brakpoint 2004 96kt pelikilpailun voittanut .kkrieger, joka vei vain 97,280 tavua. Vastaavan pelin kokoaminen normaaleilla menetelmillä olisi vienyt 200-300

Kuva 1: Wolfmoon pelissä muodostettiin puita alkeellisista osista, ja luotiin maastoja proseduraalsisesti.© Rovio Mobile ltd.

4.2.5.1 Tekstuurit

Monissa 3d- ohjelmissa on tekstuureiden generointi ominaisuus, ja valmiiksi tehdyt tekstuurit tehdään usein grafiikkaohjelman työkaluja osin hyväksikäyttäen. Tällaisia ovat monet kohina, plasma ja pilvi työkalut. Myös J2ME ympäristössä voidaan generoida grafiikkaobjekteja proseduraalisesti, mutta se voi osoittautua työlääksi ja hitaaksi metodiksi, eikä sitä juuri käytetä.

4.2.5.2 Objektit

Parametrisistä objekteista voidaan pitää esimerkkeinä vaikka kirjahyllyjä, joille generoidaan kirjat sopivaan epäjärjestykseen. Monet luonnon kasvit muuttuvat hyvin loogisesti parametrisiksi objekteiksi. Puut voidaan helposti rakentaa yksilöllisiksi ja luonnollisen näköisiksi maastoparametrin ja alkusiemenluvun(seed) avulla. [kuva 1]

4.2.5.3 Maastot

Pelikenttiä ja maastoja voidaan muistin säästämiseksi generoida laajoja alueita (Parametric terrain generation)[14]. Haittapuolena on tähän tarvittavan koodin kirjoittaminen, koodin muistinkulutus ja kenttien generoimiseen tarvittava prosessoriteho. Varsinkin maastojen fraktaaliominaisuudet ovat omiaan laajojen alueiden luonnolliseen rakentamiseen[15]. Monissa strategiapelissä kuten Civilizationissa ja The Settlersissä pelin kiinnostavuus on jatkuva, koska uusia pelattavia maailmoja on lähes loputtomasti. The Sentinel(1980) sisälsi 10 000 pelikenttää proseduraalisesti generoituna. Pelissä Just Cause (2006) luotiin todella iso pelialue (1024km^2) yhdistelemällä perinteistä ja proseduraalista generointia ympäristössä. Pelissä myös simuloidaan säätilan muutokset, kuten pilvien mudostuminen ja sade syntyminen automaattisesti. [15] Surffauspelissä Surf's UP pelintekijät kokeilivat rakentaa fyysisesti realistisen aaltosimulaation, jossa palaaja toimisi. Säätäminen oli kuitenkin niin työlästä, että ajatuksesta luovuttiin, ja siirryttiin enemmän valmiiksi generoitujen aaltojen käyttöön[17].

4.2.5.4 Maailmat

Vanhassa klassikkopeli *Elitessä* tutkittiin kahdeksaa galaksia, joissa jokaisessa oli 256 planeettaa. [18] Jokaisen planeetan arvot, kuten nimi, sijainti, kauppahinnat ja kuvaus generoitiin yksilöllisiksi käyttämällä seed-arvon perusteella. Pelisää olisi ollut mahdollista generoida yli 281 biljoonaa galaksia, mutta tekijät päättivät luopua tästä pelimaailman uskottavuuden säilymiseksi.

4.2.5.5 Tarina, juoni ja pelihahmot

Elitessä oli myös jokaiselle maailmalle oma nimi, kuvaus ja parametrit, jotka saatiin aikaan generoimalla planeetan siemenestä. Monessa pelissä on yleisesti vastustajat generoitu pelitilanteen ja paikan mukaan vaihtuvilla ominaisuuksilla.

Joissakin peleissä liikkuvat vastustajat pelisuunnittelijan ennalta määräämiä reittejä, jolloin pelaajan piti pärjätäkseen opittava nämä reitit ulkoa. Joissakin peleissä on jopa itse pelaajan liikkeet ennalta määrätty, jolloin pelaajan tehtäväksi jää vain vihollisten ammuskelu. Pelaaja tuntee pelin hyvin rajoittuneeksi ja rajoittavaksi. Tekoälyn kehittyessä, viholliset liikkuvat kentällä omien mielihalujensa mukaan ja reagoivat omalla tavallaan ympäristöön, toisiin pelihahmoihin ja pelaajan toimintaan. Tekoälyn ei tarvitse olla edes kovin monimutkainen, jotta se olisi mielenkiintoinen. *Lemmings* pelin suuri suosio perustui sopulilauman minimalistisen älyn kanssa toimimiseen. Monessa pelissä on myös käytetty tehtävängenerointia.

4.2.5.6 Animaatiot

Spore pelissä pystyi luomaan oman eläinlajin evoluution myötä. Pelaaja pystyi asettelemaan valitsemalleen torsolle vaihtelevan määrän jalkoja ja muita elimiä haluamaansa järjestykseen. Tällaisien eläinten animaatiota ei tietenkään voi valmiiksi mallintaa, vaan hahmoilla on oltava tarpeeksi hyvä liikkumisalgoritmi sopeutuakseen muuttuviin olomuotoihin. Monessa pelissä on yksinkertaisempia animaatioita pitkään generoitu. Esimerkkinä monet partikkeli-ruiskut, tulen simulointi tai veden aallokon simulointi.

5 PELIPROJEKTI: WAR DIARY 3:TORPEDO

5.1 Projektin taustat

Pelikonsepti oli Rovio Mobilen Oy:n *War Diary- p*elisarjan kolmas osa nimeltä *Torpedo*. Pelin Tapahtumapaikkana oli Andamanin meri Intian valtameren itäosassa, Andamanien saarten ja Thaimaan välissä. Pelissä komennettaisiin Iso-Britannian merivoimien sukellusvenettä Japanin merivoimia vastaan. Pelin tulisi olla mahdollisuuksien puitteissa luonteeltaan realistinen ja historiallisesti todenmukainen.

Tuottajana toimi Niklas Hed, pelisuunnittelijana Ville Vuorela, johtavan ohjelmoijana Markus Eräpolku, ohjelmoijana Xiang Liu, graafikkoina minä ja Juha Impivaara, muusikona Ilmari Hakkola ja testaajina Aleksei Semenov, Kiarii Ngua ja Irina Smirinova.

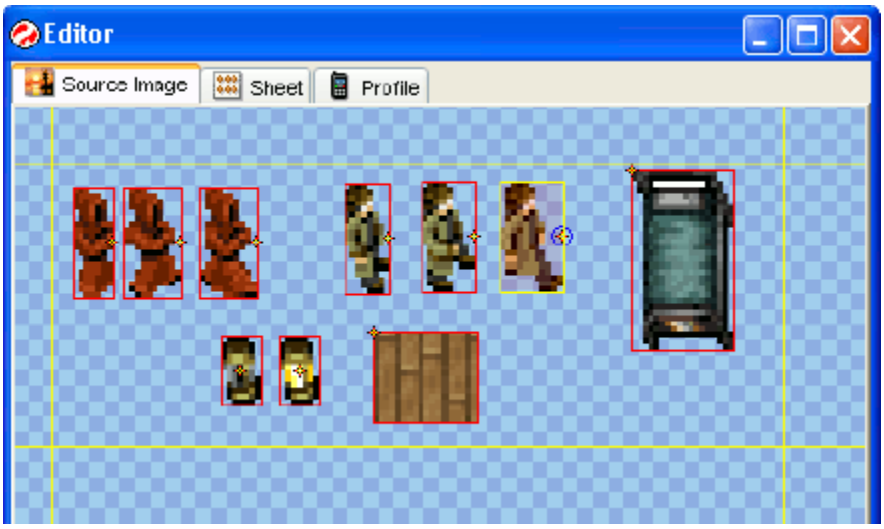
Projektin aihepiiriin tutustuminen ja visuaalisen lähdemateriaalin hankinta kävi helposti Internetin kuvahaulla, karttapalveluilla, ja usealla aiheesta kertovan video-dokumentin avulla. Teknisten ongelmakohtien ratkaisussa käytin pitkälti omaa kokemusta edellisistä projekteista, tutustumalla muissa peleissä tehtyihin ratkaisuihin, kirjallisuuteen tutustumalla, vertailemalla useita vaihtoehtoja käytännössä ja ohjelmoijien kanssa keskustelemalla.

5.2 Työvälineet

Graafikoiden apuna on osana teknologiaympäristöä oleva grafiikanhallintatyökalu (Graphics Management Tool), jota olen kehittänyt teknologia-tiimin kanssa ja muiden graafikoiden avustuksella sen ensimmäisestä versiosta eteenpäin. Työkalu sisältää kaikki kriittiset toiminnat, joilla graafikon tekemä grafiikka saadaan ohjelmoijien helposti käytettäväksi kokonaisuudeksi. Koska useimmiten pelin grafiikka kerätään muutamaan isompaan PNG-kuvaan, joista ne luetaan peliin, on näiden kuvien määrittelyssä paljon työkalulla automatisoitua työtä. Työkalun pääasiallinen toiminta on määritellä spriten nimi, spritet lukemiseen tarvittavat koordinaatit ja spriten keskipiste. Lisänä työkalulla voidaan tehdä lukuisia muitakin graafisia määrittelyjä. Useammasta spritestä voidaan koostaa suurempia kokonaisuuksia ja animoitujen spritejen kuvien yhteensovittaminen on helpompaa.

Laiteprofiilien kasvaessa tärkeä ominaisuus on useiden grafiikkaprofiilien hallitseminen ja tuottaminen. Tällä voidaan seurata eri profiileiden muistinkulutusta ja tuotannon edistymistä. Profiilinhallinnalla voidaan myös helposti muuttaa jonkin profiilin määrittelyjä ja vaihtaa profiilissa käytettävää grafiikkaa, jolloin erilaisten ratkaisujen käytännössä kokeileminen on helppoa.

Profiileja yleisesti syntyy kolme eri kokoluokkaa ja mahdollisesti jokaiseen kokoluokkaan erikseen normaali, väreiltään karsittu ja osittaisia läpinäkyvyyksiä tukevat versiot. Mahdollisesti voidaan joutua tekemään useampiakin profiileja ja joissain osa-alueissa voidaan käyttää samaa grafiikkaa. Joissain laitteessa voidaan käyttää osia useammasta profiilista. Hallintatyökalu pystyy myös automaattisesti järjestelemään erikokoisia spritejä kuvaan, niin ettei graafikon tarvitse käyttää siihen aikaa. Pienten PNG tiedostojen saamiseksi, automaattinen pakkausoptimointi voidaan tehdä PNGOut-ohjelmalla, jolla tuotetuista PNG-tiedostoista saadaan entistä pienempiä. Vaikka kuvien määrittely itsessään on yksinkertaista, voi pelissä olla satoja spritejä ja mahdollisesti toistakymmentä profiilia, jolloin kokonaistyömäärä kasvaa hyvinkin suureksi. Työkalu pystyy tätä työmäärää helpottamaan huomattavasti, samalla parantaen optimoidun grafiikan laatua. Työkaluun ollaan myös kehitetty niin, että sillä voidaan määritellä monimutkaisempaa grafiikan piirtoa, kuten usean yhteen sovitetun Spriten samanaikaisia animaatioita, joiden toteutus pelkästään ohjelmoinnilla olisi liian hankalaa. Kuten muidenkin työkalujen kohdalla, myös grafiikanhallintatyökalu kehitttyy projektien esiintuomien tarpeiden mukaan.[7]



Kuva 2: Työvälineillä grafiikoiden käsittely on helppoa

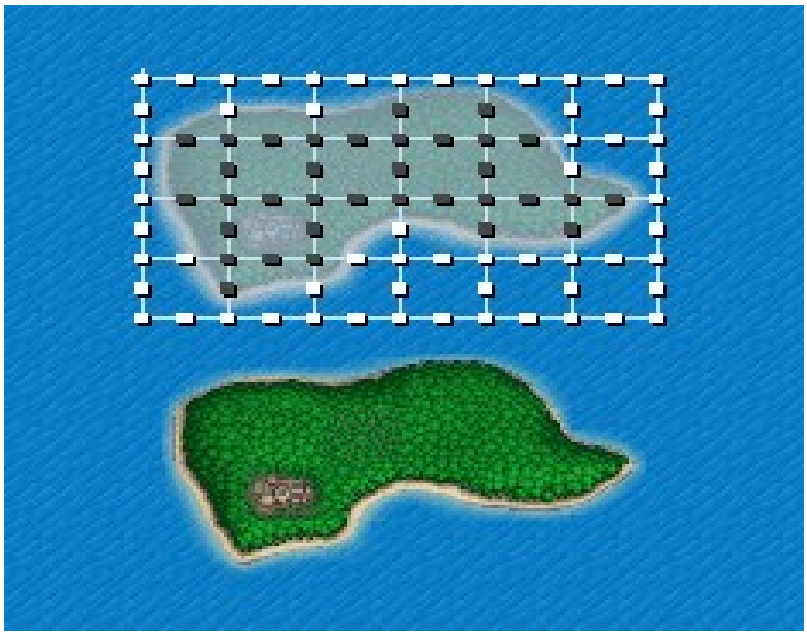
Itse grafiikkaa tuotetaan sopivilla piirto- ja kuvankäsittelyohjelmilla. Tarkkuutta vaativat yksityiskohtaiset pikseligrafiikat tuotetaan siihen sopivilla ohjelmilla. Itse olen käyttänyt pikselöintiin tarkoitettuja PersonalPaint- ja ProMotion-ohjelmia, koska niissä on animaatiotuki ja sujuva värienhallinta. PersonalPaint on oma suosikkini helppokäyttöisyydellään, mutta sille ei ole Windows-versiota, joten joudun käyttämään sitä WinUEA-emulaattorin välityksellä. ProMotionin etu on, että siitä voi suoraan kopioida kuvia leikepöydälle, ja siten kätevästi liikuttaa kuvia sen ja Adobe PhotoShopin välillä. Myös muita pikseligrafiikkaan soveltuvia ohjelmia on kehitetty. Suurpiirteisempään grafiikkaan ja yleiseen kuvankäsittelyyn suosittu ohjelma on Adobe PhotoShop. Joskus apuna voidaan käyttää muitakin ohjelmia, kuten ilmaista GIMP-kuvankäsittelyohjelmaa. Adobe PhotoShop ja muut vastaavat kuvankäsittelyyn tarkoitetut ohjelmat eivät kuitenkaan sovellu kovin hyvin esimerkiksi spriteanimaatioon.

Ohjelmointiympäristönä käytin tarvittaessa Eclipseä, jonka kautta voidaan tekstitiedostoja tarvittaessa editoida, peli kääntää ja synkronoida projekti versionhallintaan. Eclipse itsessään voi olla ajoittain raskas ja muuttua epävakaaksi, mutta soveltuu tehtäväänsä muuten hyvin.

Lisäksi käytössä on perinteiset kynä ja paperi, jolle voi luonnostella ja piirtää skannattavaksi kuvia, tai laskea geometriaa.

5.3 Pelikenttägrafiikat

Koska suuren kartan toteuttaminen vähällä muistinkulutuksella oli yksi odotettavissa olevista ongelmista, täyttyisi pelin käyttää pelimaaston piirtoon soveltuvaa toteutusmallia. Pelimaasto oli kolmen syvyisen veden ja yhden tyypin maan alueesta koostuva maisema, jonka piirrossa rantaviivan piirtotarkkuus ei olisi kaikkialla merkittävä, mutta olisi kuitenkin näytettävä yksityiskohtaiselta. Lisäksi kartalla olisi laajoja yhden tyypin alueita, kuten ulapalla olisi laaja syvän meren alue, jonka tallennuksessa voisi säästää muistia. Näitä tavoitteita vastaan ehdotin piirtotpaa, joka perustuisi nelihaaraiseen (quadratree) puurakenteeseen, jolla olisi mahdollisesti fraktaalipiirteitä. Tällä tietorakenteella olisi maasto mahdollista pakata pieneen tilaan niin latauspaketissa, kuin käyttömuistissakin. Fraktaalisuudella voitaisiin generoida rantaviivaan yksityiskohtia periaatteessa äärettömiin Benoit Mandelbrotin oppien mukaan, kasvattamatta itse karttamäärittelyn viemää muistia. Karttadataa voisi myös optimoida määrittämällä tarvittaessa rantalinjaa tarkemmin, esim. Taistelupaikkana toimivan sataman ympäristössä, mutta taas pitkiä matkoja geneeristä rantakaistaletta voitaisiin määritellä kevyemmin ja antaa satunnaisalgoritmin luoda rantaviivaa. Toinen etu kyseisessä rakenteessa olisi, että karttaa olisi helppo zoomata vain vaihtamalla lukusyvyyttä puusta. Zoomaustasoja, tai laajaa pelialuetta ei kuitenkaan katsottu pelissä tarvittavan, ja päädyimme tekemään staattisen pelinäkymän.



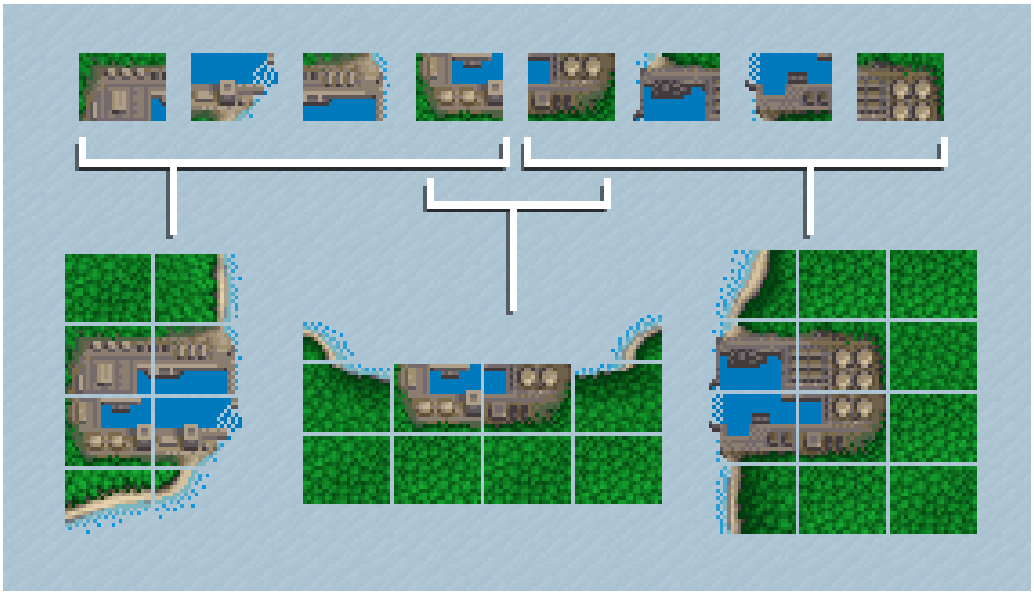
Kuva 3: Pelikenttä koostettiin karttadaataan sopivista tileistä

Vesitilejä oli suunnittelija määritellyt kolme erilaista sen mukaan, kuinka syvällä sukellusvene voi kulkea: pinta-ajo, periskooppisyvyys ja täysi sukellus. Kyseiset tilet väritettiin, niin että syvin olisi tummin. Matalin vesi olisi taas vaaleampi hohtaen matalalla vaalempaa merenpohjaa. Koska syvyyden hyväksikäyttö olisi pelaamisen kannalta kriittistä, näiden kolmen alueen tummuuseron tuli olla tarpeeksi vahva erottuakseen selvästi myös huonopinäyttöisimmissä puhelimissa. Kahden vierekkäisen erivärisen tilen raja saatiin sulavaksi piirtämällä osin läpinäkyväksi liukuvaa tileä toisen päälle. Koska Tilet piirrettäisiin ruudulle aina tietyssä järjestyksessä, gradienttikuvion olisi oltavava oikeanlainen toimiakseen kaikissa tilanteissa, joissa kaikki kolmekin vesialuetta olisivat keskenään kosketuksissa. Tähän tarkoitukseen käytin aimpaan ”Darkest Fear” peliin kehittämääni gradienttiläpinäkyvyystileä, jota hieman muovasin terävämmäksi. Vesitilet päällystettiin vielä yhtenäisellä auringonvaloa heijastavalla aallokko tekstuurilla. Aluksi kokeilimme myös erikseen tilen päälle piirettävällä osin läpiväkyvällä animoidulla aallokolla joka jopa voisi iskeytyä rantaa

vasten muodostaen ranta aallokko kuvioinnin, mutta sen piirtäminen osoittautui liian raskaaksi. Tämän sijasta merta elävöitettiin yksittäisillä satunnaisilla vaahtopäillä.

Rannan piirtoon päädyin käyttämään yhdeksän noden alueelle levittäytyvää tileä, joka jakaisi reunimmaiset nodet naapureidensa kanssa. Nämä nodet kertovat onko kohdassa maata vai vettä, jonka mukaan alueelle valitaan siihen sopiva rantapala. Koska muistinsäästö on aina ongelma, kaikkia variaatioita ei tehty, mutta riittävän iso tilekoko takasi tarpeeksi paljon variaatiota rantaviivaan, näyttävämpää grafiikkaa ja nopeamman piirron ruudulle yhtäaikaisesti piirrettävien tilejen määrän laskiessa. Koska sisämaan viidakko olisi yhdellä metsätilellä piirrettynä liian monotonisen näköistä, tein muutaman erilaisen tilen, joilla generoitiin vaihtelevaa maastoa sisämaahan.

Lisäksi tarvittiin muutama tile lisää edustamaan satamia. Pelisuunnittelijan alkuperäinen tarkoitus oli tehdä kolme satamaa. Satamat olisivat sijoittuneet maantieteellisesti niin, että vesi ja viidakko ympäröivät niitä eri puolilta.Tällasessä tilanteessa, jos sama satama piirrettäisiin ympäristöönsä sulautmattomasti olisi lopputulos visuaalisesti tökerö, varsinkin kun peli muuten koostuisi vedestä ja viidakosta. Suunnittelija kuitenkin karsi kolmannen sataman pois, koska näin käytettävät resurssit voitaisiin keskittää kahteen satamaan. Sain kuitenkin suunniteltua kaksi satamaa, niin, että niiden osista voitaisiin koostaa kolmaskin satama, eikä pelisuunnittelua tarvinnut karsia tältä osin.



*Kuva 4: **Y**hdistelemällä pienempiä elementtejä, voidaan rakentaa suurempia kokonaisuuksia.*

5.4 Laivat

Pelissä piirrettäviä laivoja oli kolme sukellusvenettä, kolme rahtilaivaa ja kolme sota-alusta. Sukellusveneistä pelaaja sai komennettavakseen suurempia edetessään tehtävässä ja vastassa oli vihollisen kuusi laivatyyppiä. Osa laivoista esiintyi myös brittien komennuksessa. Sukellusveneen ja muiden laivojen toteuttamisessa katsottiin haasteena olevan useiden laivojen piirtäminen mahdollisimman moneen suuntaan. Vaihtoehtoina mietittiin joko monen spriten piirtämistä eri suuntiin, ohjelmallisesti vektoreita piirtäen, tai pelkästään symbolisesti. Kaikki vaihtoehdot tuntuivat aika huonoilta, joten päätimme pilkkoa aluksia osiin ja uudelleenkäyttää näitä osia. Pääsimme Juhan kanssa hyvään tulokseen piirtämällä erilevyisiä runko- ja keulapalasia kahdeksaan suuntaan, ja kokoamalla laiva järjestämällä lähimmäksi sopivat palaset oikeaan suuntaan.

Peruspalojen päälle piirrettiin suunnasta riippumattomia kansirakenteita. Näin alukset pystyivät kääntymään asteettomasti pikselin tarkkuudella suhteellisen pienellä grafiikalla, mutta kuitenkin ulkoasun visuaalisesti yhtään kärsimättä.

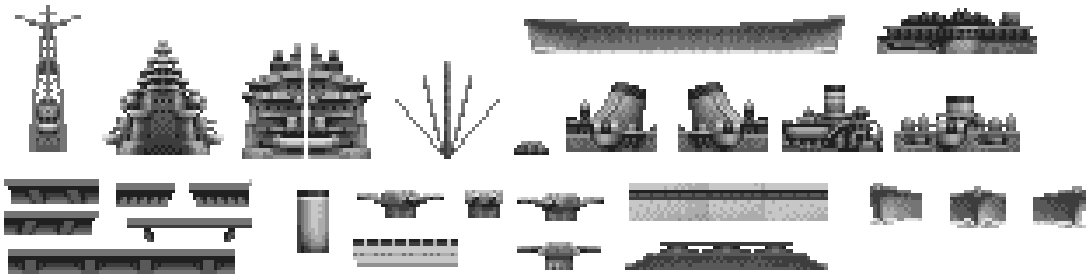
5.5 Partikkelit

Partikkeleilla ilmaistiin myös pelillisesti tärkeitä laivojen jättövanoja, joiden avulla pelaajan on helpompi arvioida alusten suunta ja nopeus. Toinen käyttökohde oli laivojen haavoittumista ilmaiseva savuvana. Lisämausteena oli Paid To Kill-peliin tehty sade-efekti, jolla ei itse pelin kannalta ollut mitään merkitystä, vaan oli puhtaasti tuomassa visuaalista ilmettä, muuten yksitoikkoiselle avomerelle.

5.6 3D näkymä

Olimme jättäneet avoimeksi olisiko pelissä lisänä periskooppinäkymää, ja millainen siitä tulisi. Tulimme siihen tulokseen että se on visuaalisesti niin tärkeä elementti, että se kannattaa tehdä ruudun alareunaan näyttämään taistelukenttää realistisesti 3d näkymänä.

Tässä laivat tulisivat näkyviin jopa hyvin läheltäkin, joten ne olisi toteutettava realistisesti ja pikkutarkasti eri etäisyyksiltä ja oltava historiallisesti uskottavia. Tutkin lähdemateriaalia, selvittääkseni japanilaisten laivatyyppien ulkomuotoa tuona ajankohtana. Lopulta olin saanut määriteltyä sopivat mittasuhteet ja ominaispiirteet senaikaisille laivatyypeille. Määrittelin laivojen rungon mittasuhteet ohjelmoijalle, jotta pelinäkymään voitaisiin laskea piirrettävän laivan koko. Tämän päälle piirrettäisiin myös keula, mikäli näkyvissä. Rungon yläpuolelle taas määrittelin sijainnit, mihin piirrettäisiin kansirakenteet käyttäen etäisyyteen ja katsomiskulmaan parhaiten sopivaa spriteä. Periskooppinäkymässä näkyvät mantereet piirrettiin sopivista rantaelementeistä näyttämään yhtenäistä viidakkoa eri etäisyyksiltä. Taivas ja Meri piirrettiin ohjelmallisesti gradienttia käyttäen. Mereen lisättiin aaltoja ja torpedon ollessa liikkeellä torpedon vana. Katselukorkeutta myös vaihdettiin sen mukaan, oltiinko pinnalla vai periskooppisyvyydellä. Syvemmällä sukellettaessa näkymä täyttyi vedestä, ja ruudun ohitse nousi ajoittain kuplia.



Kuva 5: 3D Laivat koostettiin yksinkertaisista elementeistä.

5.7 Valikot

5.7.1 Pelinaikaiset valikot

Pelin aikaiset valikot näyttävät pelaajalle tämän mahdollisesti kaipaamia tietoja, tärkeimpänä kartta koko alueesta ja kohteen sijainnista. Lisäksi valikosta pystyi tarkistamaan oman aluksen tarkemmat tiedot, tehtävänannon ja oman pelistatuksen. Nämä toteutettiin näyttämällä yksittäisiä isoja taustakuvia ja koostamalla pienemmistä elementeistä tarvittavat näkymät. Toteutuksesta vastasi Juha

5.7.2 Päävalikko

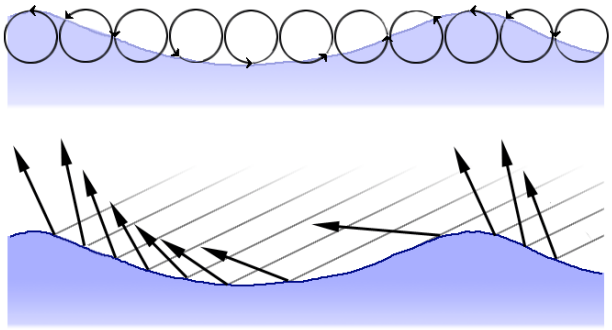
Pelin näyttävyytden kannalta ovat valikoiden taustat usein tärkeässä osassa. Käytettävä kehitysalusta oli tuolloin valikoiden osalta jokseenkin rajoittava, joten valmiiseen valikkomalliin oli tyydyttävä. Tämä rajoitti käytettävän valikkotekstin fontin puhelimen omaan perusfonttiin. Päätelaitteiden laaja valikoima oli myös tässäkin haasteena päävalikon ulkoasua suunniteltaessa.

Päädyin lähestymismalliin, jossa tausta generoitaisiin ohjelmallisesti mahdollisimman pitkälle skaalatutuvaksi. Pelin aiheeseen sopiva teema olisi animoitu merinäkymä, jolle toisi lisäilmettä tuhoutuneesta tankkerista nouseva tulipatsas. Taivas oli helppo muodostaa piirtämällä vierekkäin sopivalla gradientilla väritettyä tileä, joita piirrettiin ruudun vasemmasta laidasta oikeaan laitaan. Horisonttiin piirrettäisiin tuhoutunut öljytankkeri kahdesta spritestä. Näiden spritejen välistä nousisi sitten kahdessa virrassa tulipatsaan muodostavia animoituja pilvimäisiä spritejä. Tämä toteutettiin valmiilla partikkelirutiinilla, jollepystyi antamaan parametreiksi sopivan suunnan, tuulen ja ikääntymisnopeuden pienillä satunnaisvaihteluilla.

Horisontin yläpuolelle piirretty näkymä luettiin pikselirivi kerrallaan ja piirrettiin vastakkaisessa järjestyksessä horisontin alapuolelle, saaden aikaan vettä esittävä peilattu näkymä. Tasaisesti peilaava taso ei vielä itsessään muistuta vettä, varsinkaan avomerellä, jossa käy jatkuvasti aallokko. Koska ohjelmoijalla oli projektin kannalta kiireellisempää tekemistä, jatkoin itse ohjelman muokkaamista. Aallokon veteen sain muuttamalla luettavaa kohtaa taustassa liu'uttamalla sopivaa aaltomuotoa ajan kuluessa kohti katsojaa.

Tällainen aallokko onkin monesti käytetty ohjelmoinnin visuaalinen tehokeino, mutta näyttää usein myös aika mauttomalta. Aallokon realistisemmaksi olisi itse aalto oltava jotain muuta, kuin sini-aaltoa, ja tasaista aaltorintamaa rikottava. Aallokot koostuvat myös erikokoisista ja pituisista aalloista. Sopivan aaltomuodon likiarvon laskin paperille hahmottelemalla sini-aalto-funktion mukaan pyörivän pintaveden muodostaman vedelle tyypillisen muodon, jonka heijastuskulmat sovitin katselukulmaan sopiviksi.(Kuva 6)Tällaisesta olisi ollut turha kirjoittaa omaa prosessoritehoa vievää ohjelmaa, joten sijoitin sopivat heijastuskulmat suoraan luettavaan taulukkoon. Näin saatiin vedelle tyypilliseksi tunnistettavan pinta-aallokon muoto nopeasti määriteltyä.

Aaltorintaman rikkomiseksi jaoin aallon leveyssuunnassa pienempiin osiin, jotka piirrettäisiin hieman eri vaiheessa, jolloin aaltorintama näyttäisi tulevan hieman viistosti. Poikkiviivan leikkauskohdan jaoin tasaisesti sivusuunnassa eri riveille, ettei leikkauskohta muodostaisi näkyvää pystylinjaa. Saatuun aallokkoon lisäsin toisesta suunnasta tulevan ristiaallokon, jolloin aallot muodostaisivat yksilöitä, eivätkä yhteistä rintamaa. Tähän olisi voinut lisätä myös pienempiä ja suurempia aallokkoja, jolloin meren elävyys olisi lisääntynyt entisestään. Käytännössä kuitenkin monimutkaisempi meri ei näyttänyt juurikaan paremmalta, mutta olisi tarvinnut enemmän suorituskykyä puhelimelta, joten tyydyin malliin jossa aaltorintama tulee kohti etuviistosta.



*Kuva 6: **Y**hdistelemällä pienempiä elementtejä, voidaan rakentaa suurempia kokonaisuuksia.*

Kyseinen malli toimi hyvin peittääkseen koko ruudun alan ilman raskaita grafiikoita, vaikkakin itse näkymän animointi voisi tuottaa ongelmia heikoimmille puhelimille tehden liikkeestä nykivän. Javassa piirtokäskyjen kutsuminen on hyvin hidasta verrattuna yksinkertaisiin operaatioihin. Niinpä, kun vaakaviivat jaetaan useisiin piirrettäviin osiin, moninkertaistuu lähes samassa suhteessa piirtoon käytetty aika. Jotta hitaammatkin puhelimet pystyisivät pyörittämään tausta-animaatiota riittävän sulavasti, lisäsin ohjelmaan piirtonopeuden tarkistuksen, jonka avulla ohjelma itse sääti aaltopiirron sivusuuntaista tarkkuutta päävalikkoon siirryttäessä ja jatkuvasti taustaa piirrettäessä. Mikäli puhelin ei kyennyt piirtämään viistottaista aallokkoa kyllin sulavasti, alkoi kohtisuoran aallokon piirto välittömästi. Kaikista matalimman profiilin puhelimille ei voinut taustaa piirtää, joten tausta jätettiin pois.

Kuva 7: Päävalikon animoitu tausta

Lopputuloksena päävalikon ilmeestä saatiin yksinkertaisella ohjelmoinnilla näyttävä, mutta kuitenkin hyvin skaalautuva eri laitteille. Testaus- ja porttausvaiheissakaan ei ongelmia havaittu, vaan idea toimi ilahduttavasti suunnitelman mukaan.

5.8 HUD elementit

HUD(Head-Up Display) sisälsi määritelmän mukaan laivan nopeusindikaattorin, info-ikkunan, torpedojen määrän ja laivan vauriotason. Keskella HUDia jätettiin tilaa periskooppinäkymää varten, joka tulisi High-end laitteisiin. Info-ikkunassa taas näkyä laivan syvyys ulkoa päin kuvattuna (pinnalla, periskooppi syvyydessä, täydessä sukelluksessa)High-end laitteilla myös syvyyttä muutettaessa ja torpedoja ammuttaessa, näkyisi info-ikkunassa vastaava animaatio. Tämän toteuttaminen oli suhteellisen yksinkertaista. Lisäksi näytettäessä pelaajalle uusia raportteja, tekstilaatikkossa näytetään tekstin lisäksi myös laivan perämies. Jaoimme HUD elementit niin, että itse tein pelkästään info-ikkunan animaatiot.

5.9 Välikuvat

Monissa pelissä saadaan lisää näyttävyyttä pienten pelispritejen lisäksi käyttämällä tehtävien välissä näyttäviä välikuvia. Tämä osaltaan motivoi pelaajaa pelaamaan peliä eteenpäin, kuvien toimiessa palkintoina. Tässä pelissä yksinkertaiset ja aiheeseen sopivat kuvat olivat Sanomalehtisivut, joissa oli uutinen ja mustavalkoinen uutiskuva. Kuvien toteutuksesta ja suunnittelusta vastasi Juha.

5.10 Projektin valmistuminen

Kuva 8: Valmis peli erikokoisissa näytöissä

Lopullinen tuote vastasi aika hyvin alkuperäistä tavoitetta. Myös grafiikoiden osalta saimme aika hyvin alkuperäisen suunnitelman toteutumaan ja joissain kohdin jopa ylitimme alkuperäiset visiot. Arvostettu saksalainen Airgamer mobiilipelisivusto arvosteli pelin arvosanalla 86% ja Airgamer award-erikoismaininnalla.[19]

6 YHTEENVETO

6.1 Projektin onnistuminen

Peliprojektin onnistuminen sujui kohtuullisesti. Projektin alkuperäinen aikatauluarvio osoittautui liian optimistiseksi, joten loppupäästä jouduttiin projektille budjetoimaan lisää resursseja. Lopputulokseksi saatiin pelattava peli, pystyttiin kehittämään uutta pelimekaniikkaa, joita on voitu käyttää myöhemmin uudestaan.

6.2 Prosessin kehittäminen

Projektissa kokeiltiin monia uusia tekniikoita, jotka saatiin toimimaan yllättäen hyvin, lisäksi peli pystyttiin pitämään pelattavana, ennen kuin kaikki ominaisuudet olivat lisätty, jolloin ongelmien ilmetessä olisi lisäominaisuuksista voitu luopua. Nämä ominaisuudet kuitenkin pystyttiin lisäämään, koska niillä katsottiin olevan niin paljon lisäarvoa. Pelissä käytetyt tekniikat olisivat kuitenkin voineet muodostua riskeiksi, joten aiemmista prototyypeistä näiden tekniikoiden kohdalta olisi olut hyötyä riskinhallinnan kannalta. Katson, että aikaisiin prototyyppeihin ja niiden syvempään arviointiin panostamalla voitaisiin nopeuttaa varsinaista kehitystyötä, parantaa laatua ja pienentää riskejä. Myös muiden ketterien metodien enempi hyödyntäminen voisi parantaa tuotantoa.

War Diary3: Torpedo oli tuotannossa sinä ajankohtana, kun sisäisiä grafiikanhallintatyökaluja oltiin kehittämässä seuraavalle asteelle. Pelikehityksen aikana tuli vastaan monia kehitysongelmia, joihin kehitystyökaluja tarvitaan lisäksi, ja avasi uusia visioita, mihin voidaan tulevaisuudessa pyrkiä. Torpedo toi esiin uusia vaatimuksia myös kenttäeditorille, jota kehitettiin eteenpäin.

Projektin aikana monin paikoin tuli muutamia viivästyksiä, koska tiedostonhallinta ei pysynyt projektin kanssa ajan tasalla. Osa ongelmista johtui versionhallinta ohjelmien bugeista, ja osa johtui keskeneräisen tiedostojärjestelmän ongelmista, mikä oli altis bugeille. Vastauksena tiedostojärjestelmää on kehitetty eteenpäin entistä vakaammaksi. Mahdollisesti tiedostojärjestelmää pystytään kehittämään vielä yksinkertaisemmaksi ja integroiduksemmaksi. Yhtenä vaihtoehtona erilaiset wiki-ympäristöt.

Graafikot voisivat myös ottaa ketterän metodien periaatteita mukaan omaan työskentelyyn. Monesti graafikko tekee sellaista työtä, joka joudutaan muuttamaan myöhäisessä vaiheessa. Tällöin voi päivien työpanos mennä hukkaan. Suositeltavaa olisi että kehityksessä selviäisi mahdollisimman ajoissa tarvittavat grafiikkaratkaisut, ja ne pystyttäisi testaamaan. Graafikot voisivat miettiä parempia tapoja arvioida aikaisia testikuvia visuaalisen ilmeen ja toteutettavuuden kannalta. Monesti useampi kriittinen arvioija pystyy esittämään enemmän huomionarvoisia kommentteja.

6.3 Mobiilipelien tulevaisuus

Mobiililaitekanta kasvaa yhä edelleen ja varsinkin Kiinassa ja Lähi-idässä on kasvun olevan suurta. Laitteiden kehittyessä myös pelien hankinta tulee houkuttelevammaksi kuluttajille. Kehittäjille tämä merkitsee entistä enemmän laitteita, joita on voitava tukea. Toisaalta monet järjestelmät koettavat luoda parempaa yhteensopivuutta laitteiden välillä, kuin J2ME laitteissa on toteutunut. Varsinkin 3D-pelien merkitys tulee kasvamaan laitteiden kehittyessä. Myös tuotantometodien on seurattava laitteiden monipuolistumista ja kilpailun ollessa kireä, tehokkaasti laadukkaita pelejä tuottava yhtiö on vahvoilla.

KÄSITTEET JA LYHENTEET

Agile metodit	Ns. ketterät kehitysmetodit projektintuotannossa. Tähtäävät sopeutumiseen yllättäviin muutoksin kehitystavoitteissa
Bug	Toiminnallinen virhe, myös grafiikassa voi olla väärin piirtyviä osia.
BREW	**Binary Runtime Environment for Wireless**. Langattomien laitteiden sovellutuskehitysalusta.
CVS	**Concurrent Versions System.** Versionhallinta järjestelmä
Heap	Kekomuisti. Dynaamisesti allokoituva muisti, jota sovellutus käyttää ajon aikana
J2ME	**Java 2 Micro Edition**
JAR	**Java Archive**
JPEG	**Joint Photographic Experts Group**
PNG	**Portaple Network Graphics**
Lossy	Häviöllinen tiedonpakkausmenetelmä
Lossless	Häviötön tiedonpakkausmenetelmä
Low-End, High-End	Laiteprofiili. matalampi ja korkeampi suoritusteho.
MIDP	**Mobile Information Device Profile**
QVGA	**Quarter Video Graphics Array.** Näyttötila, 320*240 pikseliä
Quadtree	Nelihaarainen puumuotoinen tietotaulukko
Sprite	Piirrettävä graafinen objekti
Symbian	vähäresurssisille laitteille kehitetty köyttöjärjestelmä
SVN	**Subversion**. versionhallinta-systeemi
Tile	Neliön muotoinen grafiikka-"laatta", joita voidaan piirtää vierekkäin
YCbCr	Väriavaruus, jossa tummuus, punainen ja sininen värikomponentti ovat erotettu omiksi kanavikseen
ZIP	Tiedostojen pakkausformaatti
Wiki	Kokoelma verkkosivuja, joita kävijä voi helposti muokata. Käytetään yhteistyötä ja tiedon jakamista helpottamaan

LÄHDELUETTELO

[1] FRANZAS, JON. J2ME Moninpelattava mobiilipeli. Diplomityö, Helsingin Yliopisto

[2] 2005.KANKAANRANTA, MARJA; NEITTAANMAA, PEKKA; HÄKKINEN, PÄIVI. Digitaalisten pelien maailmoja. Jyväskylän Yliopisto 2004

[3] Digitoday. 2008, Joka kolmannes mobiilipeleistä ei toimi. http://www.digitoday.fi/viihde/2008/02/29/Sovellusyhti%F6%3A+kolmannes+mobiilipeleist%E4+ei+toimi/20086188/66

[4] JUNIPER RESEARCH: Press Release: Juniper Research Predicts Mobile Games Market to Reach $10bn by 2009, Driven by Emerging Markets and Casual Gamers http://www.juniperresearch.com/shop/viewpressrelease.php?id=95&pr=63

[5] HEPO-OJA, MARKUS. Developing Porting Process for Mobile Games.Thesis, HELIA, 2005.

[6] NGUA, KIARII. Java 2nd Micro Edition game testing. Bachelor's Thesis. EVTEK, 2006

[7] KARLSSON, KIMMO. Using Custom Tools in Mobile Game Development. Master's Thesis, Helsinki University Of Technology 2007.

[8] ROLLINGS, ANDREW: Game Architecture and Design. 2000

[9] VUORELA, VILLE. Pelintekijän käsikirja. 2007

[10] Sun Microsystems, Inc.,2000: J2ME(TM) MIDP Specification 1.0a

[11] Sun Microsystems, Inc. and Motorola, Inc.,2002:Mobile Information Device Profile Specification (“Specification”)Version: 2.0W3C:

[12] Portable Network Graphics (PNG) Specification (Second Edition) http://www.w3.org/TR/REC-png.html.

[13] LEHTINEN, TUOMO. Developing Tools for Mobile Games. Master's Thesis, Helsinki University Of Technology 2006.

[14] MANDELBROT, BENOÎT B. How long is the coastline of Britain. Science 1957

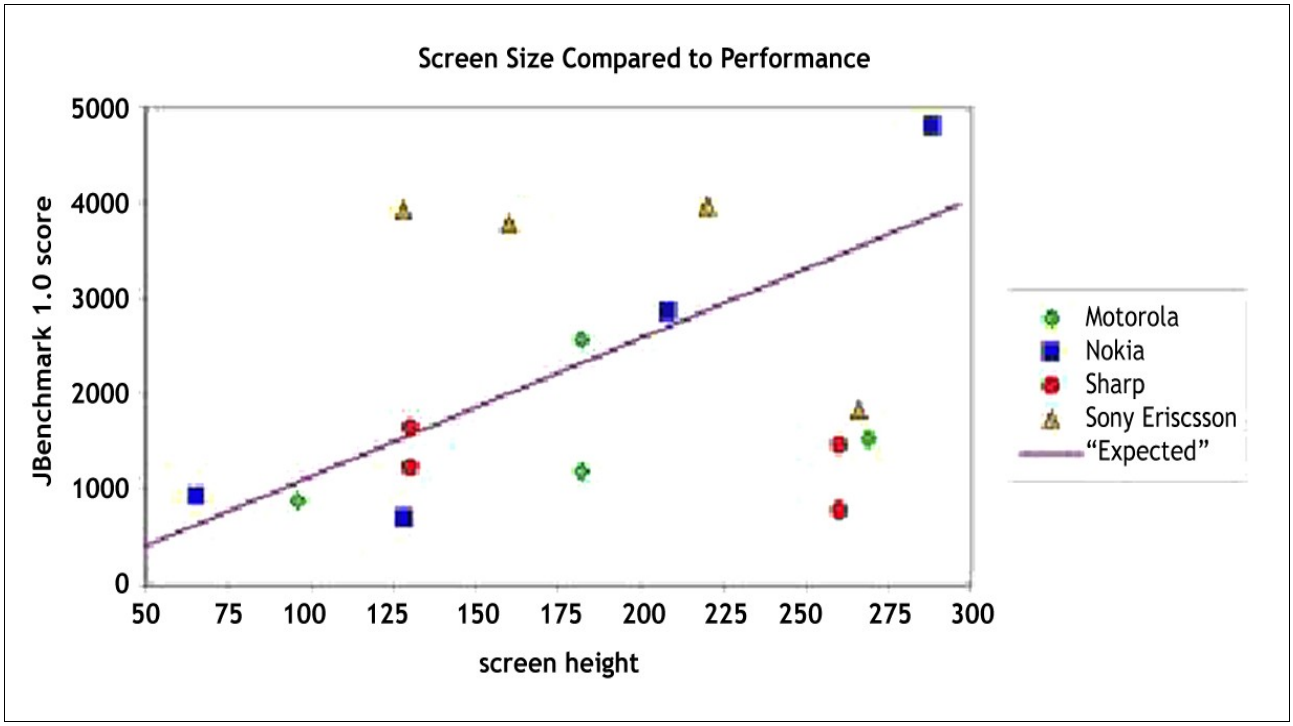
[15] GAME VIDEOS. http://www.gamevideos.com/video/id/5973

[16] BAR-LEV, ADI. Game Developer February 2008 Volume 15, Big WavesThe Guardian. Masters of their universe. October 18, 2003 http://books.guardian.co.uk/extracts/story/0,,1065455,00.htmlAIRGAMER. 2006. http://www.airgamer.de/handyspiele/test/zeige/war-diary-torpedo.html

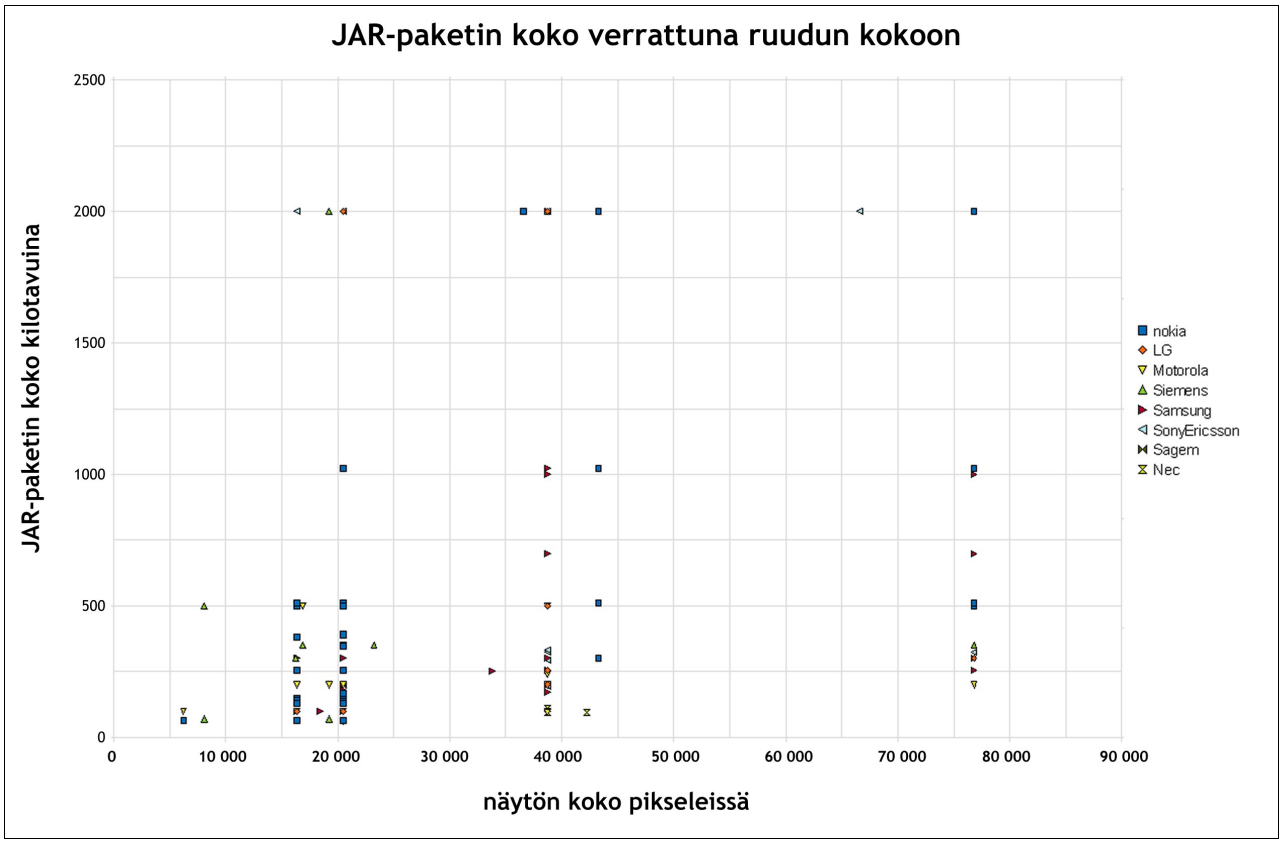
[17] The Guardian. Masters of their universe. October 18, 2003 http://books.guardian.co.uk/extracts/story/0,,1065455,00.html

[18] AIRGAMER. 2006. http://www.airgamer.de/handyspiele/test/zeige/war-diary-torpedo.html

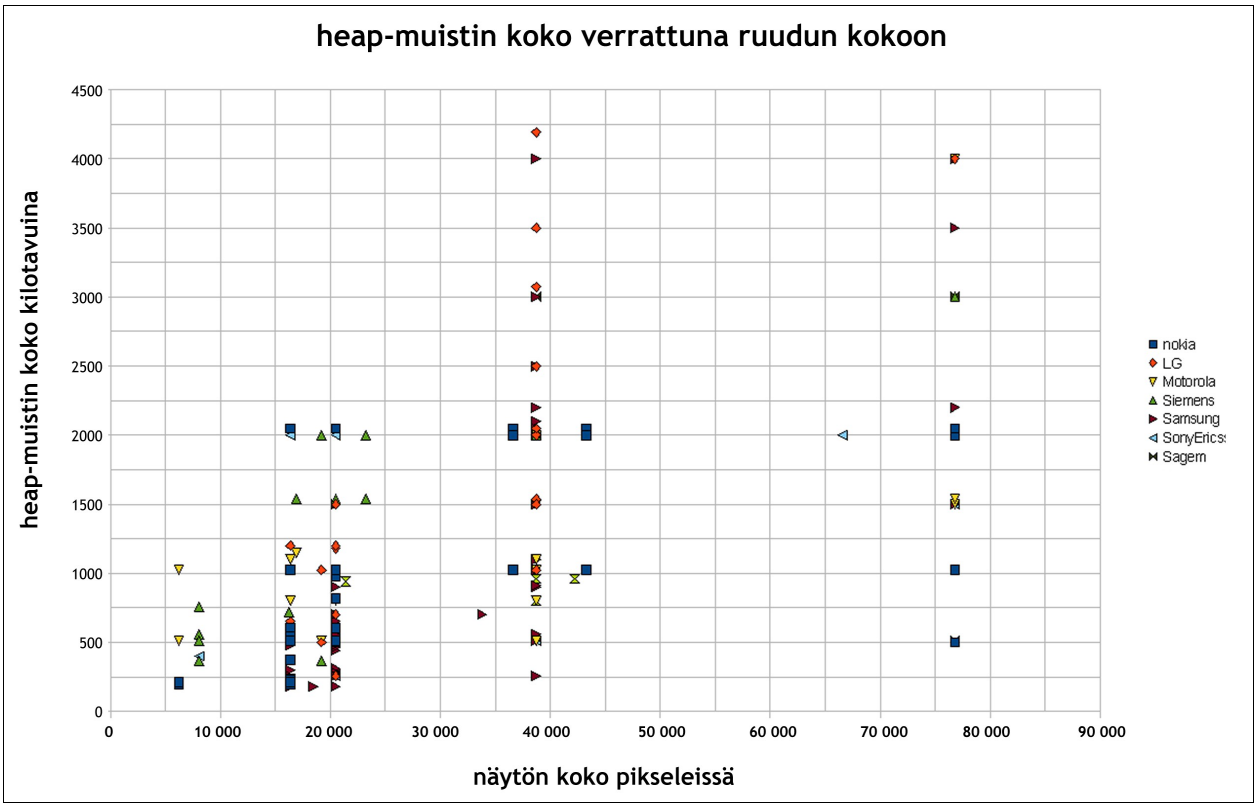
LIITTEET (10kpl)



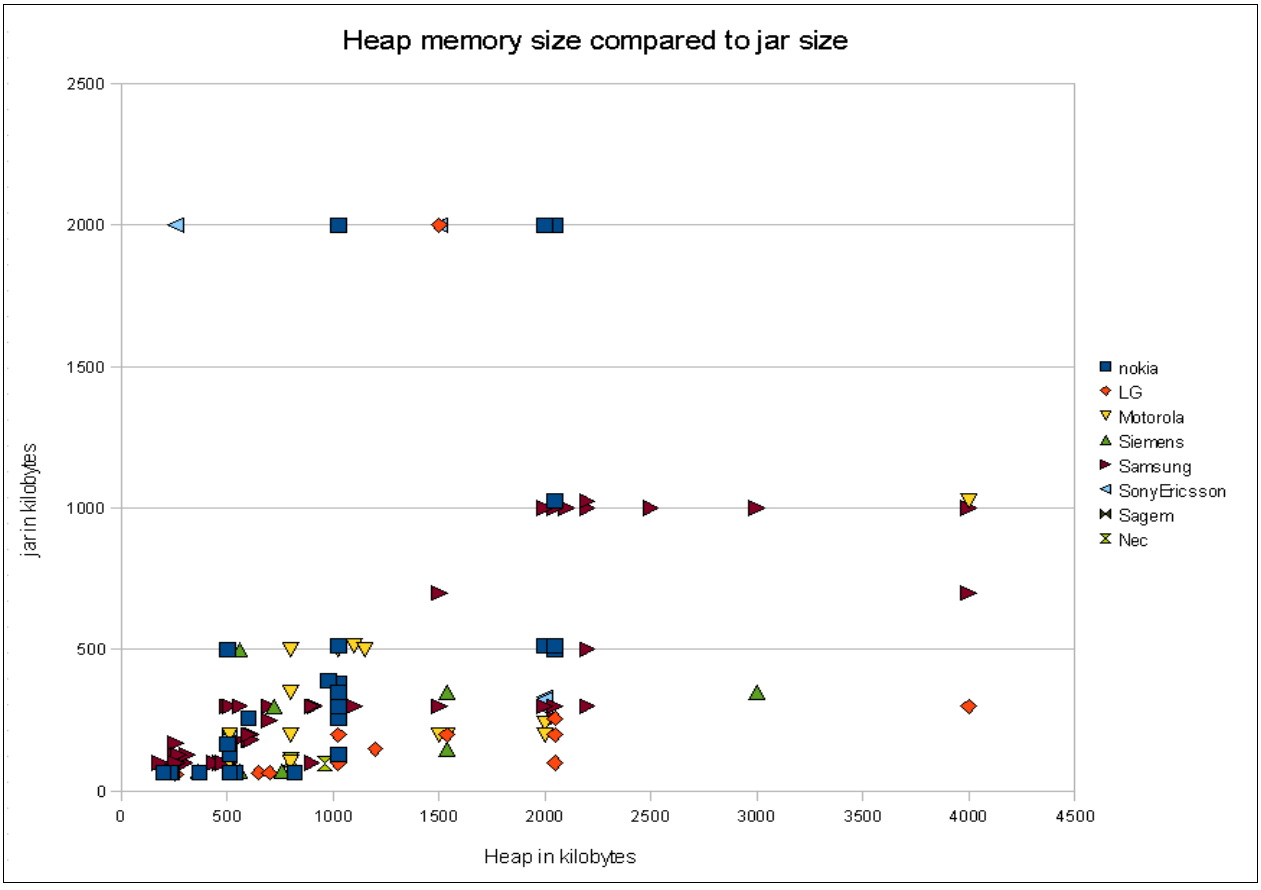
Liite 1: Kaavio laitteiden suoritustehoista verrattuna ruudun kokoon. Kaavion laati Kimmo Karlsson.



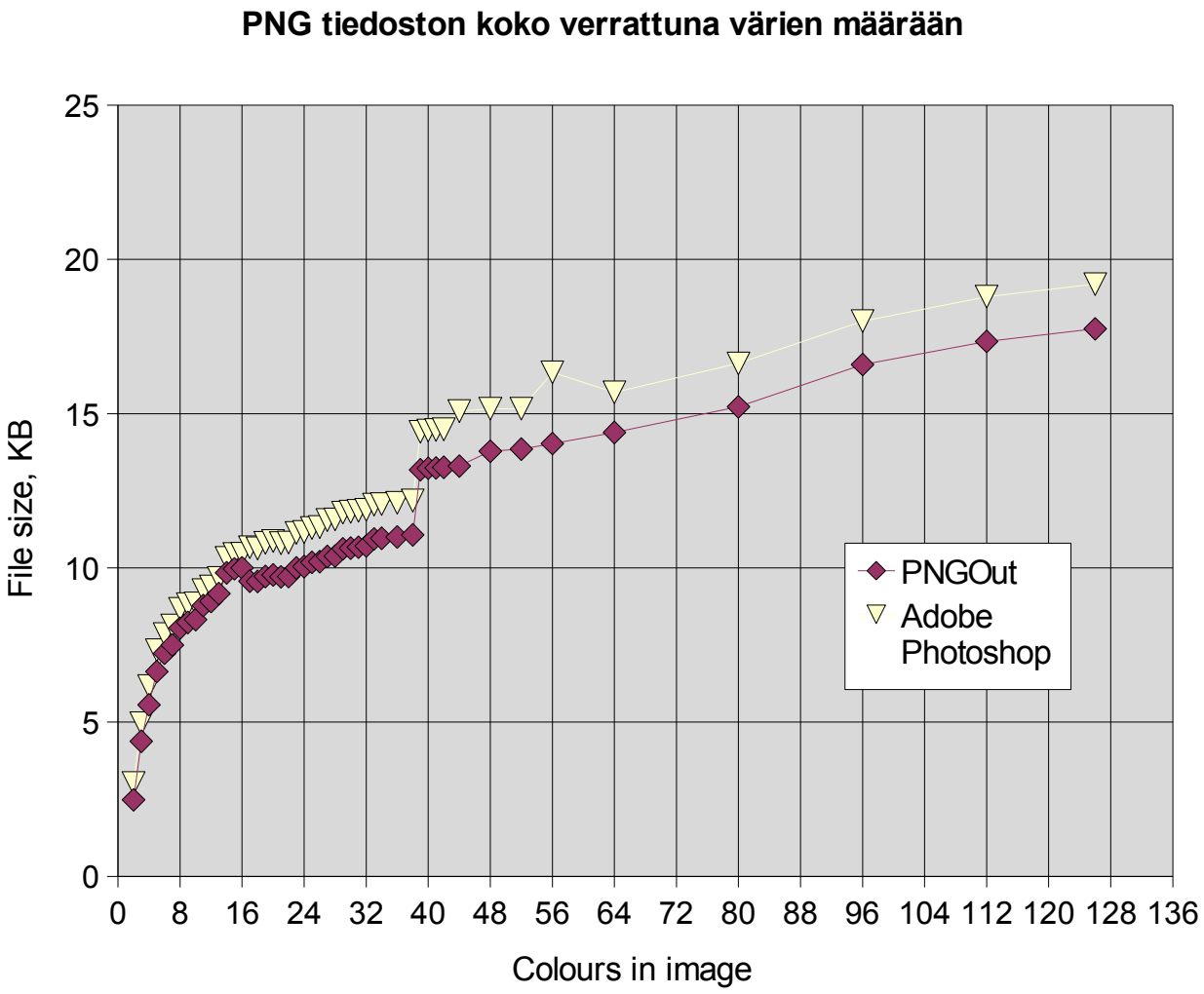
Liite 2: Latauspakettien maksimikoko verrattuna laitteiden näytön kokoon. Kaaviosta huomataan, että QVGA koon näytöillä varustetuilla laitteilla voi maksimi latauspaketti olla alle 250 kilotavua. Data kerätty Rovion tietokannasta.



Liite 3: Heap-muistin koko verrattuna laitteiden näytön kokoon. Kaaviosta huomataan, että QVGA koon näytöillä varustetuilla laitteilla voi heap muisti olla vain 500 kilotavua. Data kerätty Rovion tietokannasta.

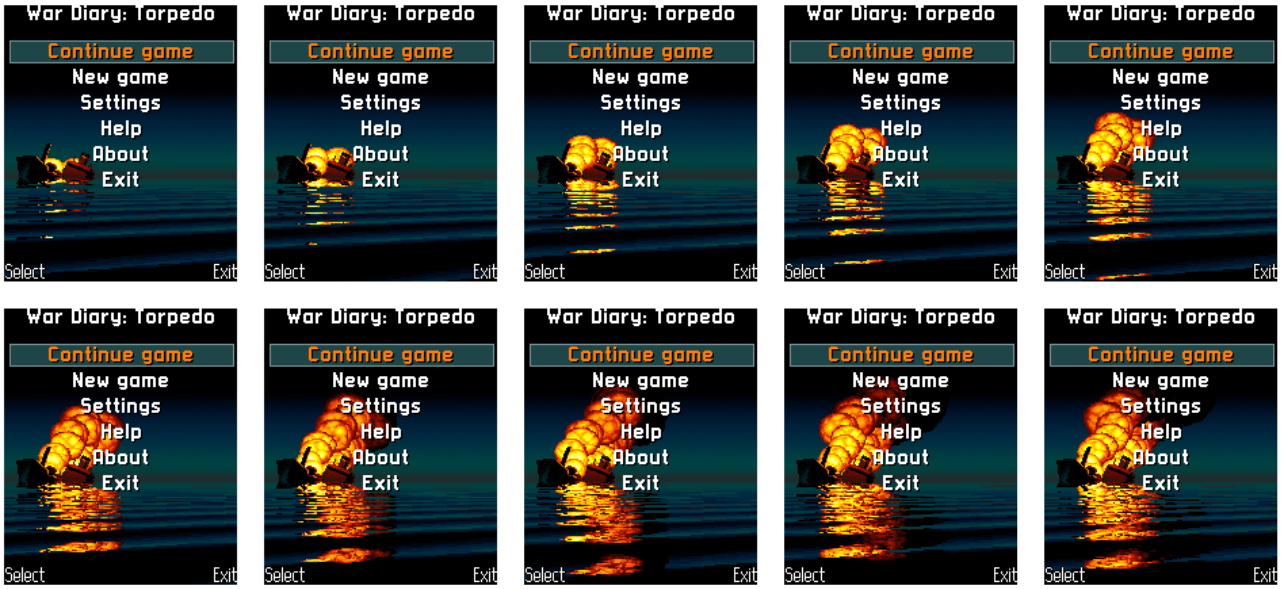


Liite 4: Latauspakettien maksimikoko verrattuna laitteiden heap-muistin määrään. Kaaviosta huomataan, että näiden suhde voi poiketa huomattavasti. Data kerätty Rovion tietokannasta.

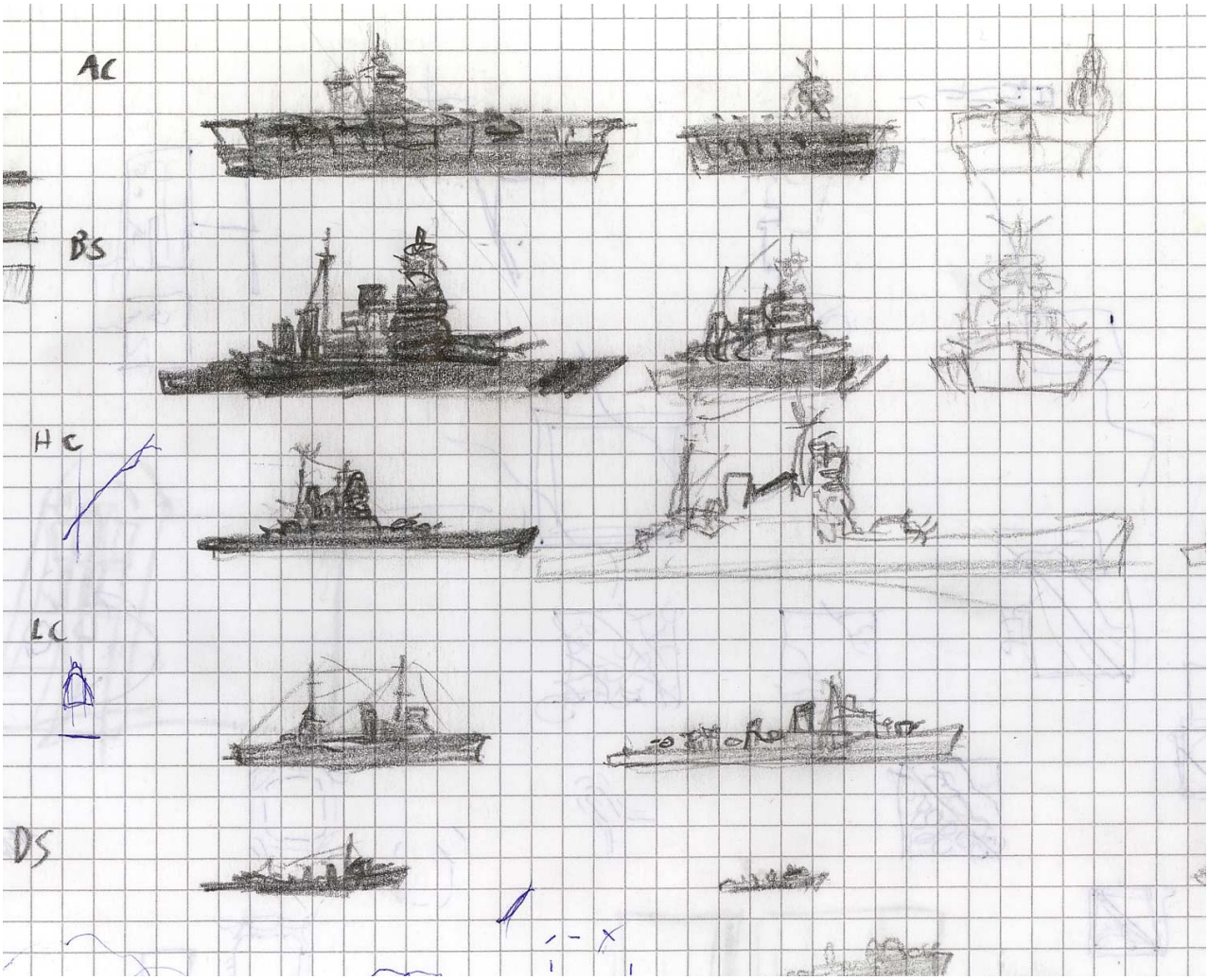


*Liite 5: 176*208 kokoisen png-kuvan koko eri värimäärillä. Vertailussa PNGOut ohjelmalla pakattu ja pelkästäänAdobe Photoshopilla tallennettu kuva.*

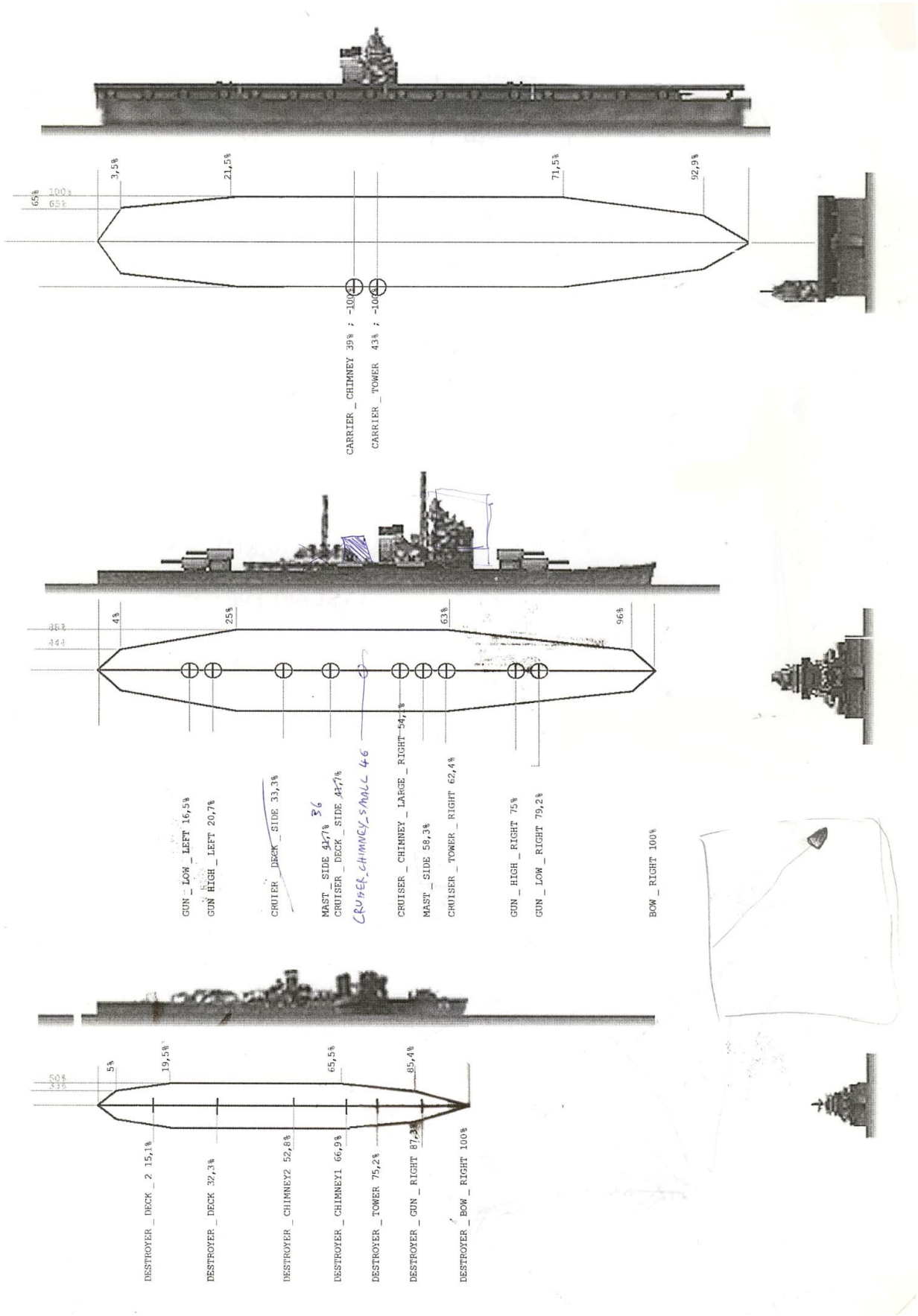
*Liite 6: 176*208 koon kuvakaappauksia pelistä.*



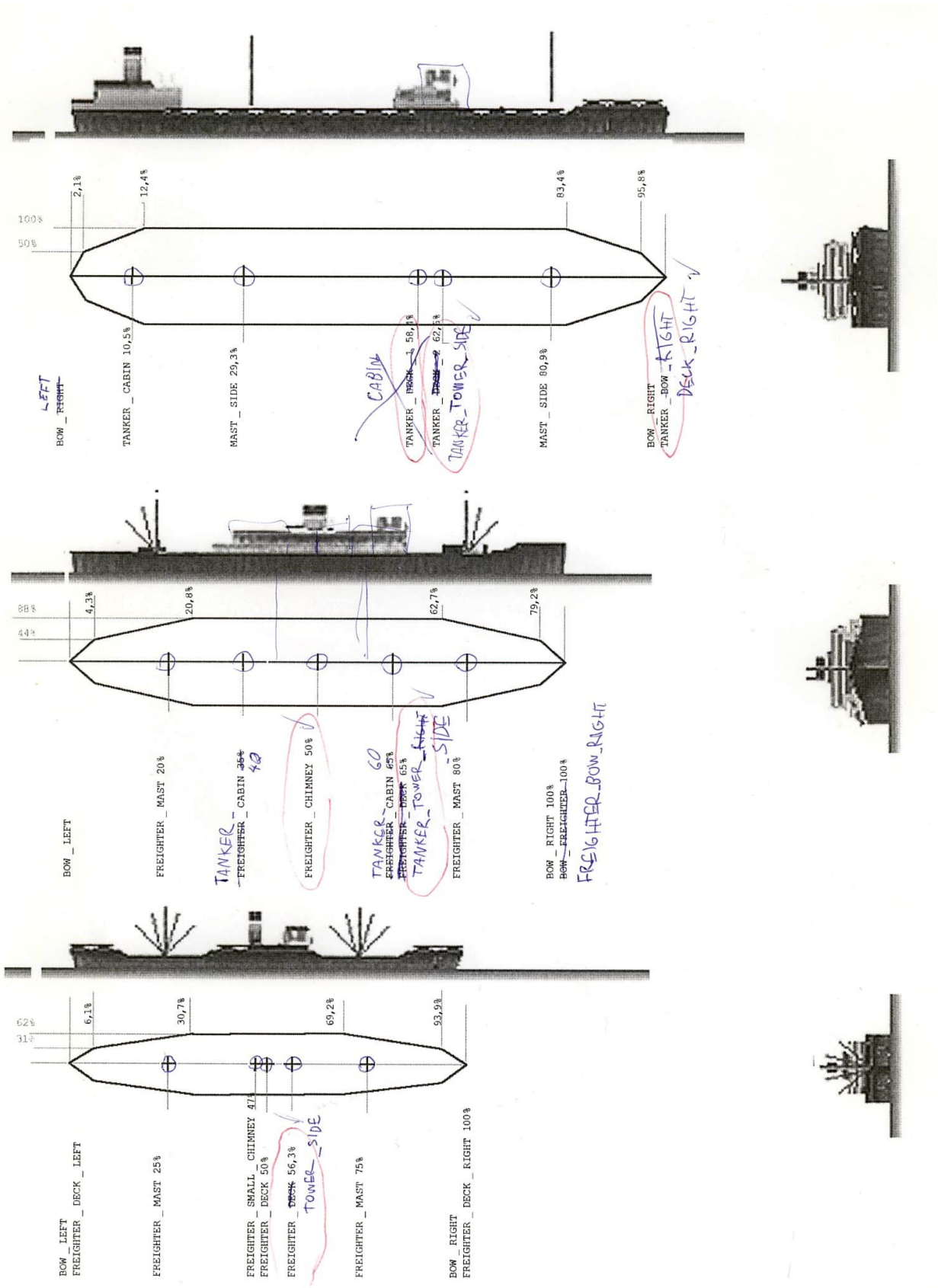
*Liite 7: 176*208 koon päävalikon tausta-animaatio.*



Liite 8: Japanilaisten sota-alusten hahmottelua paperille.



Liite 9: 3D Laivojen elementtien suunnittelua.



Liite 10: 3D Laivojen elementtien suunnittelua.