Elias Valle

Asynkronisen keräilykorttipelin suunnittelu ja toteutus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

7.11.2024

Tiivistelmä

Tekijä:	Elias Valle
Otsikko:	Asynkronisen keräilykorttipelin suunnittelu ja toteutus
Sivumäärä:	35 sivua
Aika:	7.11.2024
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Pelisuunnittelu
Ohjaajat:	Lehtori Antti Laiho

Insinöörityöksi suunniteltiin ja toteutettiin verkossa asynkronisesti pelattava mobiilipeli. Peli on auto battler -genren ja keräilykorttipelin yhdistelmä. Projektilla ei ollut ulkoista tilaajaa, vaan se sai alkunsa Metropolia Ammattikorkeakoulun innovaatioprojektikurssilla syntyneestä peli-ideasta. Peli kehitettiin kahden Metropolia Ammattikorkeakoulusta jo valmistuneen opiskelijan kanssa. Pelimoottorina käytettiin Unity 3D -pelimoottoria, ja taustajärjestelmät toteutettiin Unity Gaming Services -palvelulla.

Raportissa esitellään käytettyjä ohjelmistoja, palveluja ja työkaluja. Unity 3D -pelimoottorin ja Unity Gaming Services -palvelun lisäksi käytettyjä ohjelmistoja olivat muun muassa GitHub -versionhallintapalvelu ja Google Play Store -sovelluskauppa. Raportissa mainitaan myös joitain vaihtoehtoisia työkaluja, joita ei lopulta päädytty käyttämään. Pelisuunnitteluun ja mekaniikkoihin kiinnitetään paljon huomiota, ja pelin toteutuksessa käytettyjä käytänteitä esitellään yksityiskohtaisesti. Työssä tuodaan esille kohdattuja ongelmia, ja niitä analysoidaan. Ongelmia kohdattiin eniten taustajärjestelmien toteutuksissa ja pelin jakamisessa.

Projektin tarkoituksena oli luoda esimerkkiprojekti pienellä kehitysryhmällä ja resursseilla, joita mahdollisimman monet voivat hyödyntää. Projektin aikana saatiin aikaiseksi toimiva demoversio pelistä. Demoversiolla on kaikki ydintoiminnallisuudet, joita peliä suunnitellessa määritettiin, mutta lopputulos on vielä hieman hiomaton, ja peliä jatkokehitetään. Peli ei ole raportin kirjoittamisen aikana julkisesti jaossa pelattavaksi, mutta peli pyritään viemään jollekin jakoalustalle tulevaisuudessa.

Avainsanat:	pelisuunnittelu, pelinkehitys, pelitasapaino, keräilykorttipeli, verkkopeli

---

Abstract

Author:	Elias Valle
Title:	Design and development of an asynchronous online collection card game
Number of Pages:	35 pages
Date:	7 November 2024
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Game Applications
Supervisors:	Antti Laiho, Senior Lecturer

An asynchronous online collection card game was designed and developed in the final year project. The game is a combination of auto battler and online collection card game genres. The project had no outside commissioner. Instead, the project originated from a game idea on an innovation project course at Metropolia University of Applied Science. The game was developed with two Metropolia University of Applied Science graduates. Unity 3D game engine was used for the frontend development and the backend systems were implemented via Unity Gaming Services.

The used software, services and tools are introduced in the thesis. This includes the Unity 3D game engine, Unity Gaming Services, GitHub version control software and the Google Play Store application shop. Some alternative tools that were ultimately not used in the project are mentioned as well. A major focus of the thesis is on game design and mechanics as well as detailed practices that were utilized in the development. Additionally, the report describes encountered problems and their analysis. Most problems were related to backend implementation and the distribution of the application.

The goal of the project was to create an example game with a small development team and resources that are widely available. As an outcome of the project a functional demo version of the game was produced. The demo version has all the core functionalities that were defined during the development, but the result is still unrefined and to be further developed. At the time of writing the thesis the game is not publicly available, but the plan is to publish a game distribution platform in the future.

Keywords: game design, game development, game balance, online collection card game

Sisällys

Lyhenteet

SDK: *Software Development Kit*. Ladattava paketti kirjastoja ja funktioita, joita voidaan hyödyntää ohjelmistoissa kohdennettujen toiminallisuuksien toteuttamiseen.

JSON: *JavaScript Object Notation.* Tiedostomuoto, joka käyttää avain-arvo-pareja.

1 Johdanto

Insinöörityön aiheena on verkossa pelattavan asynkronisen keräilykorttipelin suunnittelu ja toteuttaminen ideasta pelattavaksi peliksi. Peliä kehitetään kahden Metropolia Ammattikorkeakoulusta jo valmistuneen tieto- ja viestintätekniikan opiskelijan ja yhden tieto- ja viestintätekniikan opintojen loppuvaiheessa olevan opiskelijan kesken. Jokainen kehitysryhmäläinen erikoistui opinnoissaan pelinkehitykseen.

Työn tavoitteena on luoda pelattava peli projektikurssilla syntyneen idean pohjalta, käydä läpi prosessin vaiheita ja antaa yleinen kokonaiskuva kehityksen kannalta oleellisista osioista. Työ yritetään toteuttaa mahdollisimman pienin kustannuksin. Insinöörityössä tuodaan esille edustan ja taustajärjestelmien ratkaisuja kaikista pelinkehitysprosessin vaiheista.

Työssä kiinnitetään erityisen paljon huomiota mekaniikkojen suunnitteluun ja toteutukseen sekä taustajärjestelmien toimintaan ja käyttämiseen. Lisäksi työssä pyritään antamaan lukijalle esimerkkejä käytetyistä käytänteistä ja palveluista, jotta niitä voidaan hyödyntää samankaltaisissa projekteissa.

Asynkronisten pelien suunnittelussa ja toteutuksessa on omanlaisensa vaatimukset verrattuna reaaliaikaisiin moninpeleihin. Asynkronisuudella tarkoitetaan reaaliaikaisuuden vastakohtaa. Insinöörityön pelin kaltaisissa verkkopeleissä asynkronisuus tarkoittaa, että kaikki pelaajaosapuolet eivät ole pelissä reaaliaikaisesti. Pelin asynkronisuus tulee työssä esille erityisesti taustajärjestelmäratkaisuissa ja pelin mekaniikoissa, sillä asynkronisissa peleissä ei tarvita suuria kutsumääriä taustajärjestelmien ja päätelaitteiden välille, ja pelaajien toimet pelien aikana ovat rajoittuneita. Pelin mekaniikoissa ja ansaintamalleissa puolestaan paneudutaan tarkemmin keräilykorttipelien ominaisuuksiin ja vaatimuksiin.

2 Asynkronisen keräilykorttipelin suunnitteluvaihe

Tavanomaisesti pelinkehitys alkaa ideasta. Kun kehittäjillä on selkeä idea, se yritetään toteuttaa jonkinlaiseksi testattavaksi prototyypiksi. Prototyyppiä sen jälkeen muutetaan ja parannellaan testauksen perusteella, kunnes ollaan tyytyväisiä tuloksiin. (1.) Insinöörityöksi kehitettävän pelin perimmäinen idea on luoda yhdistelmä virtuaalista keräilykorttipeliä ja asynkronisesti pelattavaa auto battler -peliä. Auto battler on peligenre, jonka pelit ovat automaattisesti ohjelmoitujen sääntöjen mukaisesti eteneviä taistelupelejä. Insinöörityön peli on verkkopeli ja kutsumanimeltään DACAB. Kuvassa 1 näkyy pelin logo.

Kuva 1. Pelin logo.

Kuvassa 1 esiintyy yksi pelin hahmoista ja kuvan kortit kuvastavat pelin mekaniikkaa, jossa pelaaja valitsee nostamansa kortit pelin antamista vaihtoehdoista. Korttien eriväriset hehkut kuvastavat pelin erilaisia korttityyppejä. Pelissä on maaginen fantasiateema. Grafiikat ovat tyyliltään kuvan 1 kaltaisesti sarjakuvamaisia ja värikkäitä.

2.1 Pelin toiminnallisuuden lähtökohdat

Yksi onnistuneen pelisuunnittelun merkkejä on se, että peli voi tarjota vaihtelevia ja mielekkäitä kokemuksia yksinkertaistenkin mekaniikkojen pohjalta (2). Tämä, joustavat pelikertojen kestot ja keräilykorttipelien tarjoama strateginen luovuus toimivat ohjaavina tekijöinä pelin toiminnallisuudelle.

DACAB on auto battler, joka tarkoittaa peligenreä, jossa pelaajat luovat ryhmän hahmoja ja ne taistelevat automaattisesti muiden pelaajien ryhmiä vastaan. Hahmoilla on omia kykyjä ja ominaisuuksia, jotka vaikuttavat otteluiden etenemiseen. (3.) Insinöörityön pelissä kerätään virtuaalisia kortteja, joista muodostetaan pakkoja. Näin luoduista pakoista nostetaan ja pelataan kortteja, jotka luovat tai muokkaavat hahmoja, joista puolestaan muodostuu pelaajan joukot. Tiivistettynä pelin ydintoiminnallisuus koostuu pakanrakennuksesta, korttien pelaamisesta kentälle, taisteluista ja korttien keräämisestä. Pelin asynkronisuus tarkoittaa, ettei vastakkain ottelevien pelaajien tarvitse olla samanaikaisesti pelissä otellakseen keskenään. Tämä ominaisuus poistaa monille muille verkkopeleille ominaiset jonotusajat ja virtaviivaistaa pelikokemusta.

2.2 Pelin kehityksen ja jakamisen alustat

Hyvän idean lisäksi pelinkehityksen ensimmäisiin vaiheisiin kuuluu pelin alustan valitseminen. Yleisimmät pelialustat ovat mobiili, tietokone ja konsolit. Jokaisella pelialustalla on hyviä ja huonoja puolia, ja erilaiset pelit sopivat joillekin alustoille paremmin kuin toisille (taulukko 1).

Taulukko 1. Pelialustojen hyvät ja huonot puolet.

Pelialusta	Hyvät puolet	Huonot puolet
Tietokone	Tietokonepelit ovat halpoja, grafiikoiltaan laadukkaita ja niillä on hyvät optimointimahdollisuudet.	Tietokonepelit vaativat usein laitteistopäivityksiä ja tietokonepelien markkina on kylläinen.
Mobiili	Mobiilipelit ovat helppoja kehittää, julkaista ja mainostaa. Mobiilipeleillä on suuri käyttäjäkunta.	Mobiilipelimarkkina on kylläinen. Kehitettäessä peliä mobiililaitteille tulee ottaa laitteiston rajoitukset, kuten ruudunkoko ja akunkesto, huomioon.
Konsolit	Konsolipeleillä on innokas käyttäjäkunta ja vähemmän kilpailua markkinalla. Konsoleiden hinnat ovat käyttäjille halvempia kuin tietokonepeleissä.	Konsolipelejä on haastavaa kehittää ja mainostaa. Pienten pelistudioiden on vaikea saada kehityslupia ja olla kilpailukykyisiä suurten studioiden kanssa.

Taulukossa 1 listattujen tekijöiden, kohdeyleisön ja pelin toiminnallisuuksien perusteella voidaan valita pelille sopiva kohdealusta. Insinöörityön pelin alustaksi valittiin mobiililaitteet. Mobiililaitteet mahdollistavat pelille tärkeät joustavat pelikerrat pelattavaksi missä vain ja lyhyilläkin ajanjaksoilla. Lisäksi peli ei vaadi tehokasta päätelaitetta ja kaikki pelin tärkeimmät toiminnallisuudet ovat hyvin toteutettavissa mobiililaitteille. (4.)

Pelinkehittämiseen käytetään yleensä valmiita kehitysympäristöjä eli pelimoottoreita. Pelimoottorin voi myös itse ohjelmoida, mutta se on työlästä ja suurimmat pelimoottorit kuten Unity 3D ja Unreal Engine tarjoavat kattavat ja monipuoliset työkalut erilaisten pelien kehittämiseen. Unity 3D -pelimoottoria suositellaan aloitteleville ohjelmoijille useammin, kun taas Unreal Engine -pelimoottorilla on mahdollista saavuttaa grafiikoiltaan äärimmäisen vaativia pelejä (5). DACAB

päädyttiin kehittämään Unity 3D -pelimoottorilla, koska pelimoottori on tuttu kehitysryhmälle eikä peli vaadi erityisen edistyksellistä grafiikkateknologiaa.

Mobiilipelien jakamiseen on kymmeniä alustoja, joista Google Play Store ja Apple App Store ovat selvästi suosituimpia. Vuonna 2017 Google Play Store tarjosi yli 3 miljoonaa sovellusta ja Apple App Store yli 2 miljoonaa sovellusta (6; 7). Kehitysryhmällä on Android-käyttöjärjestelmää käyttävät mobiililaitteet, joten insinöörityön pelin jakelualustaksi valittiin mobiililaitteiden natiivi sovelluskauppa Google Play Store pelitestaamisen helpottamiseksi. Google Play Store -sovelluskaupan julkaisuvaatimukset ovat lisäksi hieman löyhemmät kuin Apple App Store -sovelluskaupan (8; 9).

2.3 Mobiilipelin ansaintamallin määrittely ja valitseminen

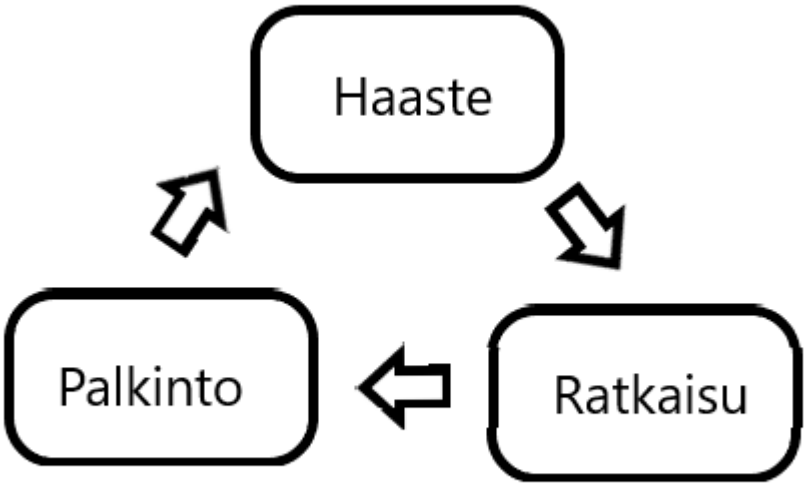
Mobiilipelit ovat joko ilmaiseksi tai mobiilipelikaupassa tehtävää kertaostosta vastaan ladattavissa. Kun ensimmäiset mobiililaitteet alkoivat tarjota sovelluskaupoissa pelejä kuluttajille, oli ominaista, että peli myytiin kertaostettavana tuotteena (10, s. 139–141). Tämä alkuperäinen malli ei kuitenkaan toimi nykyään kovin hyvin keskiverrolle mobiilipelille. Pelien tarjonta alkoi kasvaa hyvin nopeasti ja joukosta alkoi olla vaikea erottua. Suurin osa peleistä on nykyään ilmaiseksi ladattavissa sovelluskaupoista, ja pelien kehittäjät tekevät mahdolliset tuotot pelien sisäisillä ostoilla ja mainoksilla. (11.) Mobiilipelien markkinoinnin ja ansaintamallien tutkija Andrea Knezovic (12) kertoo kirjoittamassaan verkkoartikkelissa, että 74 % yhdysvaltalaisista mobiilipelien pelaajista katsoisi mainosvideoita saadakseen sovelluksen sisäistä sisältöä vastineeksi ja 82 % mobiilipelien pelaajista pitää enemmän ilmaisista mobiilipeleistä kuin maksullisista mobiilipeleistä.

DACAB-pelin ansaintamalliksi valittiin ilmaispeli pelin sisäisillä mainoksilla, joita katsomalla pelaaja saa pelinsisäistä hyötyä. Mainostamiseen käytetään ironSource-palvelua, joka tarjoaa toiminnallisuudet eri mainosverkostojen mainoksien näyttämiseen koodissa ajettavan rajapinnan avulla.

3 Asynkronisen keräilykorttipelin mekaniikkojen kehittäminen

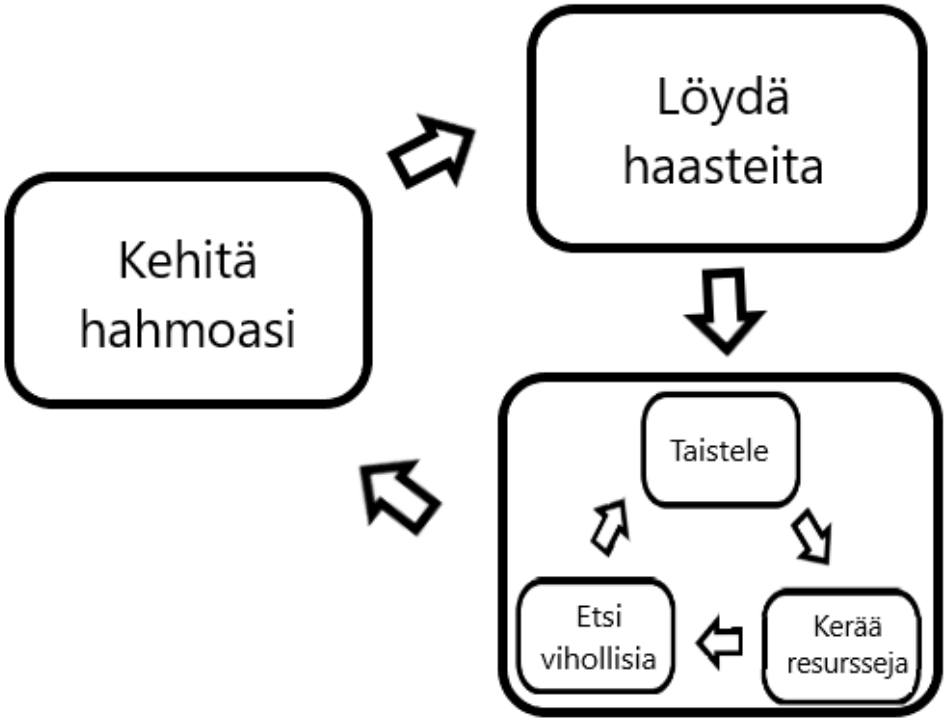
3.1 Pelisilmukan suunnittelu pelin ytimeksi

Pelisuunnittelun yksi oleellisimmista osa-alueista on hyvien pelisilmukoiden suunnittelu. Pelisilmukalla tarkoitetaan vaiheistettua prosessia, jota pelin pelaajat toistavat pelatessaan peliä. Niillä pyritään esittämään ja havainnollistamaan pelin tarjoamaa pelikokemusta. Kuvassa 2 on esitetty esimerkki pelisilmukasta, joka on yksinkertainen ja esiintyy peleissä yleisesti.



Kuva 2. Yksinkertainen pelisilmukka.

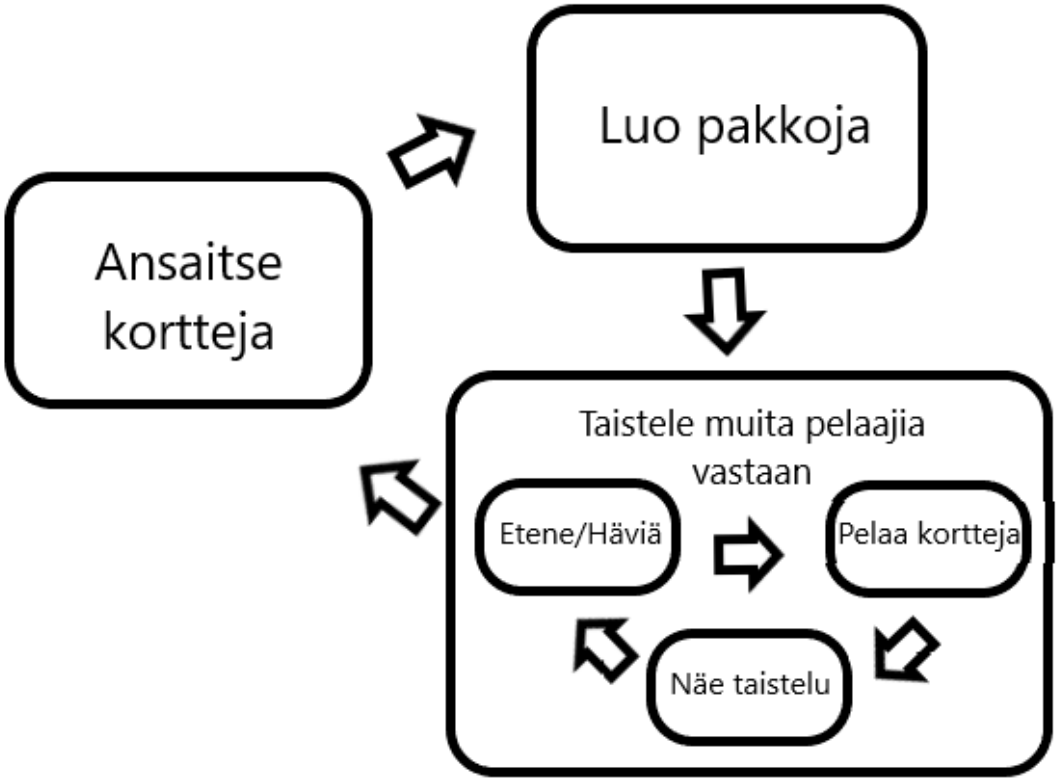
Yleensä pelisilmukoita on kuitenkin peleissä enemmän, ja ne esiintyvät muiden pelisilmukoiden sisällä. Toiset pelisilmukat sisältävät prosessin, joka voi toistua jatkuvasti peliä pelatessa, kun taas toiset saattavat olla korkean tason silmukoita, jotka toistuvat useiden pelikertojen kuluessa. (13.) Kuvassa 3 on esitetty, miten pelisilmukoissa voi esiintyä sisäkkäisyyttä.



Kuva 3. Sisäkkäinen pelisilmukka.

Hyvä pelisilmukka on selkeä, sisältää tavoitteita pelaajalle ja vastakaikua pelaajan toimille. Kaikkien pelien ydinpelisilmukassa tulee esille haaste, toimet ja palkinnot. (13.)

DACAB-pelissä pelaajat keräävät kortteja, luovat niistä pakkoja ja taistelevat pakoista pelattujen korttien luomista hahmoista koostuvilla joukkueilla muiden pelaajien joukkueita vastaan. Kuvassa 4 on esitetty insinöörityön pelin pelisilmukka.



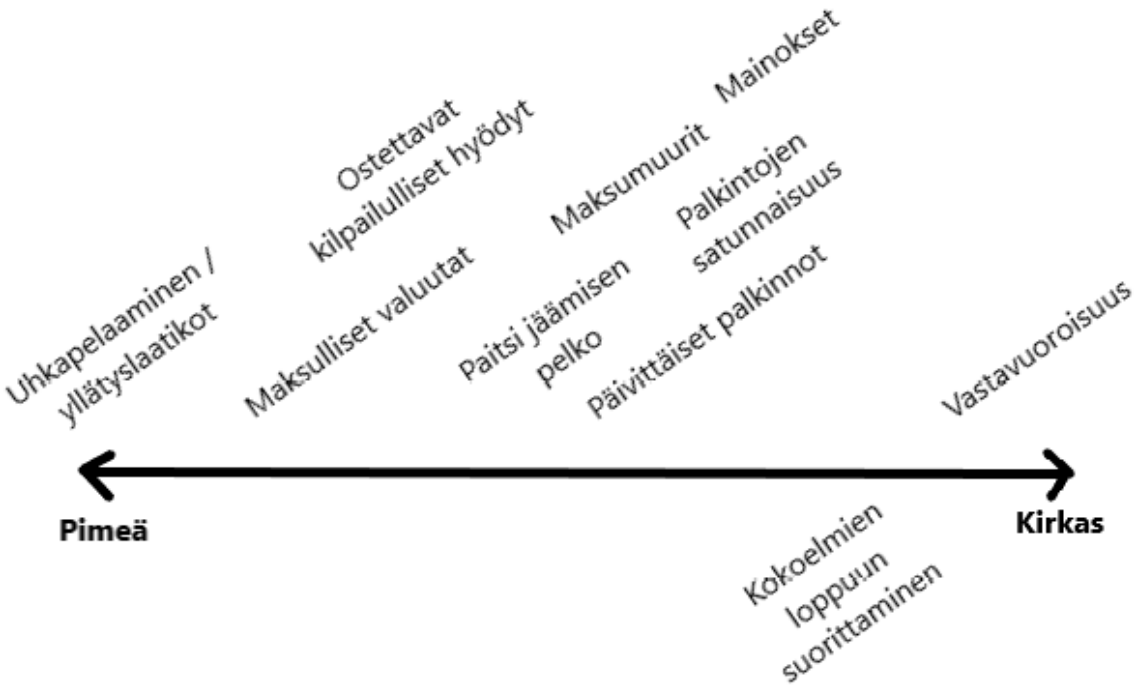
Kuva 4. Insinöörityön pelin pelisilmukka.

Kuvassa 4 esitetyssä pelisilmukassa esitellään insinöörityön pelin tarjoama pelikokemus vaiheistettuna kolmeen päävaiheeseen, jotka ovat korttien kerääminen, pakkojen luominen ja itse pakoilla pelaaminen. Pakoilla pelaaminen on vielä jaettu alavaiheisiin. Motivoivana tekijänä kuvan 4 silmukassa toimii korttien ansaitseminen ja pelaajaa palkitaan hyvästä pakanrakennuksesta ja korttien pelaamisesta sekä yksittäisillä voitoilla että korttipalkinnoilla.

Keräilykorttipeleissä palkitsemisjärjestelmät ovat suuressa osassa pelisilmukkaa. Joillekin pelaajille itse korttien pelaaminen on vähemmän tärkeää kuin korttien kerääminen, pakkojen rakentaminen ja strategioiden kehittäminen. Palkitsemisjärjestelmät ja usein myös korttien pelaaminen sisältävät paljon satunnaisuutta. Jokainen nostettu kortti on tavanomaisesti pakkapohjaisissa keräilykorttipeleissä satunnaisesti valikoitu pakasta, ja korttien kerääminen tapahtuu yleensä joko jälkimarkkinoilta yksittäisten korttien ostamisella tai ansaitsemalla pieniä satunnaisia kortteja sisältäviä korttipaketteja. Sen lisäksi näitä

korttipaketteja voi ostaa joko pelinsisäisellä valuutalla tai oikealla rahalla. Koska pelaajat voivat kuluttaa korttipaketteihin suuria määriä rahaa saaden satunnaisen arvoisia vastineita käyttämälleen rahalleen, korttipelejä on rinnastettu uhkapelaamiseen. (14.)

Aagaardin ym. (15) konferenssijulkaisussa esitellään ja analysoidaan pimeitä malleja pelinkehityksessä. Pimeillä malleilla tarkoitetaan manipuloivia ja käyttäjän terveyden tai talouden vaarantavia pelinkehitykseen käytettyjä suunnittelumalleja. Kuvassa 5 näkyy näitä suunnittelumalleja esitettynä janalla niiden vahingollisuuden perusteella. Mitä lähempänä käsite on pimeää, sitä vahingollisempana sitä pidetään.



Kuva 5. Pelinkehityksen suunnittelumalleja janalla pimeistä kirkkaisiin (15).

Kuten kuvasta 5 ilmenee, monet keräilykorttipelien palkintomallit ovat pelaajille mahdollisesti haitallisia. Palkintojen satunnaisuus ja kokoelmien kerääminen ovat keräilykorttipeleille melkeinpä välttämättömiä, mutta oikealla rahalla

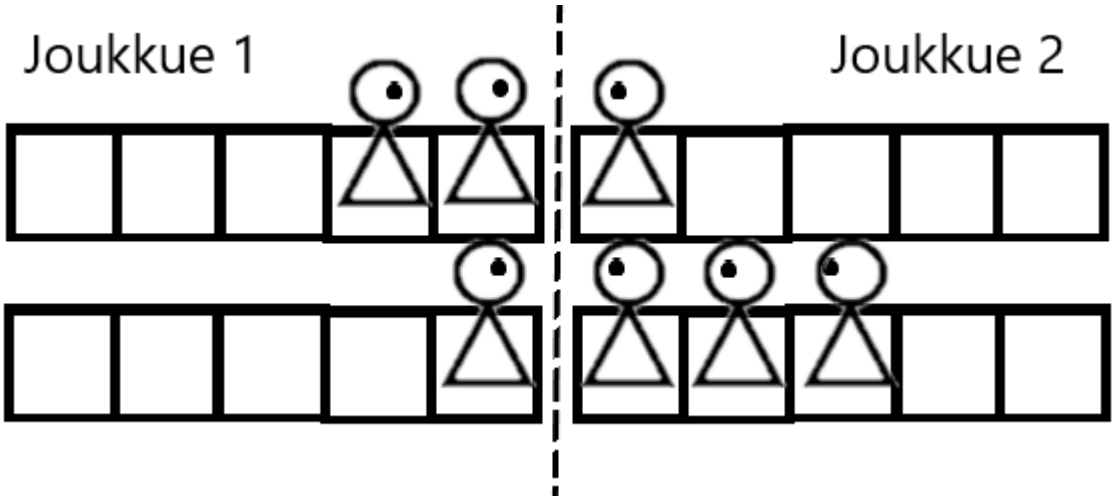
ostettavat yllätyslaatikot ja maksulliset valuutat ovat tarpeettomia pelattavuuden kannalta.

DACAB-pelin korttien kerääminen toteutetaan vain pelien loppuessa saaduilla satunnaispalkinnoilla. Pelaaja ei voi ostaa satunnaisia korttipaketteja, jotta pelaaja ei voi käyttää suuria määriä rahaa korttien hankkimiseen. Pelien lopuksi pelaaja saa palloja pelin aikana kertyneiden voittojen määrän perusteella ja pudottaa pallot kimmokkeiden läpi palkintokoloihin. Palkintokoloissa olevat palkinnot ovat satunnaisia, mutta riippuvaisia pelaajan voittojen määrästä. Palkintojärjestelmä saa vaikutteita Plinko-peleistä, jotka ovat kilpailuohjelmissa ja uhkapeleissä käytetty pelimuoto. Tätä palkintojärjestelmää ei kuitenkaan myydä pelaajalle eettisistä syistä, vaan sillä pyritään vain lisäämään jännitystä ja mielenkiintoa palkintojen ansaitsemisen yhteydessä.

3.2 Taisteluiden mekaniikat

Pelaajalla voi olla enintään viisi samanaikaista peliä käynnissä. Jos pelaajalla on vähemmän pelejä, voi hän aloittaa uuden pelin tyhjässä pelikohdassa. Pelin alussa pelaaja valitsee, millä rakentamallaan tai valmiilla pakalla hän pelaa ja rakentaa joukkueelleen nimen pelin tarjoamista nimen osista. Sen jälkeen pelaaja nostaa valitusta pakasta viisi korttia, joista kolme voidaan pelata ensimmäisessä korttien pelaamisen vaiheessa. Ensimmäisen korttien pelaamisen vaiheen jälkeen pelaaja vuorottelee taisteluiden ja korttien pelaamisen välillä. Taistelut tapahtuvat automaattisesti omien ja muiden pelaajien joukkueiden kesken. Taisteluiden välissä pelaaja nostaa kortin ja käyttää kädessä olevia korttejaan joukkueen kehittämiseksi. Pelit päättyvät, kun joukkue on hävinnyt kolme ottelua yhteensä tai pelaaja luovuttaa pelin.

Pelien joukkueet koostuvat hahmokorteilla luoduista hahmoista, joita voidaan kehittää muilla korteilla. Joukkueessa voi olla enintään yhteensä kymmenen hahmoa kahdessa jonossa, joita kutsutaan mekaniikkojen yhteydessä linjoiksi. Kuvassa 6 on hahmotelma taistelutilanteesta.



Kuva 6. Hahmotelma taistelun joukkueiden asettelusta.

Hahmoilla on jokaisella omat elämäpisteet ja hyökkäysvoima. Joillain hahmoilla on tämän lisäksi erikoiskykyjä. Taistelut etenevät automaattisesti ohjelmoitujen vaiheiden mukaisesti. Taisteluissa on kierroksia, joista jokainen on jaettu kolmeen vaiheeseen, jotka ovat kierroksen alku, taistelu ja kierroksen loppu.

Kierrosten alussa ja lopussa hahmoilla voi olla aktivoituvia erikoiskykyjä. Jokaisen kierroksen taisteluvaiheessa molempien puolien molempien linjojen ensimmäiset, eli lähimpänä kuvan 6 katkoviivaa olevissa ruuduissa olevat hahmot, hyökkäävät toisiaan kohti. Hyökkäykset toimivat siten, että hyökkäävät hahmot vähentävät ensisijaisesti oman linjansa vastustajalta hyökkäysvoimansa verran elämäpisteitä. Jos hyökkäävän hahmon linjalla ei ole vastustajaa etummaisessa ruudussa, vähennetään elämäpisteet toisen linjan etummaisen ruudun vastustajalta. Ensin alemman linjan hahmot hyökkäävät, ja sitten ylemmän linjan hahmot hyökkäävät.

Aina, kun hahmon elämäpisteet vähenevät alle yhteen elämäpisteeseen, hahmo tuhoutuu, ja kun toisen joukkueen kaikki hahmot tuhoutuvat, peli päättyy sen joukkueen voittoon, jolla on hahmoja vielä jäljellä. Kierrosten lopuilla hahmot liikkuvat etummaisiin vapaisiin ruutuihin omilla linjoillaan.

Taisteluissa voi syntyä tilanteita, joissa on jäljellä vain hahmoja, joiden hyökkäysvoima on nolla, tai joidenkin hahmojen erikoiskyvyt aiheuttavat sen, että kummankaan joukkueen kaikki hahmot eivät voi menettää kaikkia elämäpisteitään. Tämän takia taisteluilla on kierrosten enimmäisraja, jonka täyttyessä tai tilanteessa, jossa molempien joukkueiden viimeiset hahmot tuhoutuvat samanaikaisesti, taistelu päättyy tasapeliin.

Jos taistelu ei pääty tasapeliin, pelaaja saa valita korttien nostamisen ja pelaamisen vaiheessa kahdesta kortista toisen, jonka hän lisää käteensä. Molemmat valittavat kortit ovat pelaajan pakasta ja erityyppisiä. Pelaajan valittua käteensä lisättävän kortin hän saa pelata yhden kortin kädestään ennen seuraavaa taistelua. Tässä vaiheessa pelaaja voi myös järjestellä joukkueensa kentälle haluamallaan tavalla.

3.3 Korttien mekaniikat

Strategisten keräilykorttipelien mekaniikkojen syvyyttä voidaan korostaa lisäämällä kortteihin vaihtelua. Yksi menestyksekkäiden strategisten keräilykorttipelien piirre on, että kortit on jaettu eri korttityyppeihin, joiden pelaamisella ja toiminnalla on omanlaisensa säännöt. Wizards of the Coast -yrityksen keräilykorttipelissä Magic: the Gathering on kahdeksan eri korttityyppiä ja Blizzard Entertainment -yrityksen virtuaalisessa keräilykorttipelissä Hearthstone on viisi eri korttityyppiä (16; 17). DACAB-pelissä erilaisia korttityyppejä on neljä, ja ne ovat hahmo, lumous, varuste ja loitsu.

Pakkojaan rakentaessa pelaajan täytyy sisällyttää pakkaan kymmenen hahmokorttia, viisi lumouskorttia ja viisi varustekorttia. Hahmokortit ovat pelin ydin, koska vain niitä voidaan pelata kentälle yksittäisinä erillään muista korteista. Muilla korteilla kehitetään ja muokataan hahmoja.

Kun pelaaja pelaa hahmokortin kentälle, kentälle luodaan hahmokortin määrittelemä hahmo. Hahmoilla on hyökkäysvoiman ja elämäpisteiden lisäksi taso ja mahdolliset erikoiskyvyt. Erikoiskyvyt voivat aktivoitua joko tietyissä vaiheissa

kierroksia tai tapahtumien, kuten hahmojen tuhoutumisen, takia. Jokainen hahmo aloittaa tasolta yksi ja hahmojen tasot kasvavat taisteluiden ja muiden korttien pelaamisen myötä enintään tasolle kymmenen. Hahmon tasojen kasvaessa hahmon hyökkäysvoima ja elämäpisteet tai erikoiskyvyt tavallisesti kehittyvät. Kuvassa 7 on hahmokortti, jolla ei ole erikoiskykyjä.

Kuva 7. Hahmokortti, jolla ei ole erikoiskykyjä tasoilla yksi ja kymmenen.

Hahmokorteilla, kuten kuvassa 7 esitetyssä Barbarian-kortilla, on keltaiset reunukset ja alakulmiin merkityt hyökkäysvoimat ja elämäpisteet. Punaisen miekkakuvakkeen päällä oleva luku on hyökkäysvoima ja vihreän kilpikuvakkeen päällä oleva luku on elämäpisteet. Kaikilla muillakin korteilla tyypistään riippumatta on kortin yläreunassa nimi ja alaosassa kuvauslaatikko. Kuvauslaatikossa on kuvaus kortin yksilöllisestä toiminnasta tai mahdollisista erikoiskyvyistä.

Jokaisella tasolla Barbarian-hahmon hyökkäysvoima ja elämäpisteet kehittyvät tasaisesti tason kymmenen elämäpisteitä ja hyökkäysvoimaa kohti. Koska Barbarian-kortilla ei ole omia erikoiskykyjä, sen hyökkäysvoima ja elämäpisteet

ovat korkeat suhteessa erikoiskyvyllisiin hahmokortteihin. Kuvassa 8 on hahmokortti, jolla on erikoiskyky.

Kuva 8. Hahmokortti, jolla on erikoiskyky tasoilla yksi ja kymmenen.

Kuvassa esitetyllä Field Medic -kortilla on erikoisvoima, joka toimii siten, että hyökkäysten jälkeisessä kierroksen vaiheessa Field Medic -hahmot parantavat satunnaista oman joukkueen hahmoa, jonka elämäpisteet ovat alentuneet. Field Medic -kortilla on kuvan 7 Barbarian-korttiin verrattuna paljon matalammat hyökkäysvoimat ja elämäpisteet samoilla tasoilla, koska kortin erikoisvoima lisää kortin tehokkuutta taisteluissa. Barbarian-hahmon toiminnallisuus perustuu vain hyökkäysten voimakkuuteen ja hahmon kestävyyteen, kun taas Field Medic -hahmo on paljon heikompi hyökätessään, mutta suojattuna parantaa taistelevien hahmojen elämäpisteitä olipa se sijoitettu mihin tahansa kentällä.

Hahmokortteja voi muokata lumous- ja varustekorteilla. Kuvassa 9 näkyy vierekkäin esimerkit lumouskortista ja varustekortista.

Kuva 9. Esimerkit lumous- ja varustekorteista.

Kuvan 9 Stoic-kortti on lumouskortti, ja Extra Knife -kortti on varustekortti. Lumous- ja varustekorttien toiminnallisuudet ovat hyvin samanlaiset pelissä. Kummankin korttityypin kortit pelataan aina liitetyksi yhteen hahmoon ja hahmo saa pelatussa kortissa olevan kyvyn. Jokaisella hahmolla voi olla enintään yksi lumouskortti ja yksi varustekortti. Jos pelaaja yrittää pelata hahmolle toisen lumous- tai varustekortin, aiemmin hahmossa kiinni ollut samantyyppinen kortti korvautuu myöhemmin pelatulla kortilla. Stoic-lumous kaksinkertaistaa käyttävän hahmonsa elämäpisteet taisteluiden alussa. Extra Knife -varuste puolestaan aiheuttaa käyttäjänsä hyökkäysten vahingoittavan myös samassa rivissä olevan vierekkäisen linjan hahmoa.

Neljäs ja viimeinen korttityyppi on loitsukortti. Kuvassa 10 on kaksi esimerkkiä loitsukorteista.

Kuva 10. Kaksi erilaista loitsukorttia.

Loitsukortteja ei sisällytetä pakkoihin, vaan ne esiintyvät korttien nostovaihtoehtoina yhtä todennäköisesti kuin lumous- tai varustekortit. Loitsukorttien joukko on kaikille pelaajille sama pelatusta pakasta riippumatta. Loitsukortit antavat hahmoille kokemuspisteitä, joiden avulla hahmot nousevat tasoissa. Jokaista kymmentä kokemuspistettä kohden hahmon taso kasvaa yhdellä.

Hahmot saavat kokemuspisteitä myös taisteluiden myötä. Taisteluissa hahmon ansaitsemien kokemuspisteiden määrä riippuu siitä, kuinka monta hahmoa pelaajalla on kentällä. Yhteensä joukkueen hahmot saavat yhdestä taistelusta kymmenen kokemuspistettä, ja ne jaetaan mahdollisimman tasaisesti kaikkien kentällä olevien hahmojen kesken. Loitsukorteilla voidaan kiihdyttää ja keskittää kokemuspisteitä strategian kannalta oleellisimpiin hahmoihin.

3.4 Korttien tasapaino ja synergiat

Pelitasapaino on tärkeä pelinkehityksen osa-alue kaikissa peleissä ja erityisen tärkeä moninpeleissä. Pelitasapainolla tarkoitetaan pelaajien tavoitteiden saavuttamiseksi tekemien valintojen ja strategioiden tasapainottamista. Täydellisen tasapainon voi saavuttaa yksinkertaisimmin siten, että kaikki pelaajan tekemät valinnat ovat pelimekaanisesti identtisiä. (18.) Tällaisen pelitasapainon voisi DACAB-pelissä saavuttaa siten, että jokainen kortti olisi toiminnaltaan tismalleen samanlainen, mutta korttien nimet ja taide voisivat vaihdella. Esimerkkinä voidaan käyttää kolmea erikoiskyvytöntä hahmokorttia (taulukko 2).

Taulukko 2. Symmetristen hahmojen ominaisuudet.

Ominaisuus	**Hahmo 1**	**Hahmo 2**	**Hahmo 3**
Hyökkäysvoima	5	5	5
Elämäpisteet	5	5	5
Erikoiskyky	-	-	-

Taulukon 2 mukainen peli olisi teoriassa tasapainoinen, mutta sen täydellisesti pelaamiseen ei tarvittaisi paljoa taitoa. Edellä mainittu pelitasapaino voi toimia esimerkiksi reaaliaikaisissa taistelupeleissä, joissa pelaajien reaktiot ja pelityylien toimet vaihtelevat. Pelitasapainoa voidaan kuitenkin toteuttaa muillakin tavoilla kuten syklisesti (taulukko 3).

Taulukko 3. Syklisesti tasapainotettujen hahmojen ominaisuudet.

Ominaisuus	**Hahmo 1**	**Hahmo 2**	**Hahmo 3**
Hyökkäysvoima	5	1	5
Elämäpisteet	5	10	1
Erikoiskyky	-	-	Hyökkää ennen vastustajaa

Taulukon 3 hahmoista hahmo 1 voittaa hahmon 2, hahmo 2 voittaa hahmon 3 ja hahmo kolme taas voittaa hahmon 1 niiden ollessa ainoat hahmot molempien puolien joukkueissa. Tämänlainen pelitasapaino on rinnastettavissa kivi, paperi ja sakset -peliin ja sitä kutsutaan sykliseksi tasapainoksi. Keräilykorttipeleissä pelaajien luovuus ja taito tulee esille siitä, miten he yhdistelevät erilaisia kortteja tietäen minkälaisia kortteja vastapuoli saattaa käyttää. On tärkeää sisällyttää mielekkäitä vastamekaniikkoja ja synergioita korttien välille ja tasapainottaa niitä datan sekä pelaajien tuntemusten pohjalta. (18; 19.)

Esimerkkinä DACAB-pelin vastamekaniikoista voidaan käyttää Assassin-hahmokorttia vastamekaniikkana kuvan 8 Field Medic -hahmoon. Assassin-hahmolle on puolestaan vastamekaniikkana toisena esimerkkinä Book Worm -kortti. Kuvassa 11 on vastamekaniikkaesimerkkien kortit Assassin ja Book Worm.

Kuva 11. Hahmokortit Assassin ja Book Worm.

Assassin-hahmo hyökkää aina vastustajan hahmoon, jolla on matalimmat elämäpisteet riippumatta siitä, missä päin kenttää hahmo, johon hyökätään, sijaitsee, joten Assassin-hahmo on tehokas poistamaan eturivien suojaamia hahmoja, joilla on tavallisesti pienemmät elämäpisteet. Book Worm -hahmo antaa muille joukkueensa hahmoille hyökkäysvoimaa ja elämäpisteitä tuhoutuessaan. Koska Book Worm -hahmolla on hyvin matalat elämäpisteet, se on tehokas vetämään puoleensa Assassin-hahmojen hyökkäyksiä ja suojelemaan muita joukkueensa hahmoja, joilla on korkeammat elämäpisteet.

Pelissä esiintyvistä synergioista eli kahden tai useamman kortin kumuloivasta yhteisvaikutuksesta voidaan käyttää esimerkkinä Battle Medic -hahmokortin ja Fervor-lumouskortin yhteisvaikutusta. Kuvassa 12 on synergiaesimerkin kortit Fervor ja Battle Medic.

Kuva 12. Fervor-lumouskortti ja Battle Medic -hahmokortti.

Battle Medic -hahmo palauttaa menetettyjä elämäpisteitä sen edessä olevalle hahmolle aina edessä olevan hahmon hyökätessä. Fervor-lumouskortti antaa lumousta käyttävän hahmon tehdä ylimääräisiä hyökkäyksiä yhden kierroksen aikana. Jos pelaaja on pelannut Fervor-kortin Battle Medic -hahmon edessä olevalle hahmolle, sen lisäksi että hahmo vahingoittaa vastustajan hahmoja useammin, myös Battle Medic -hahmon erikoiskyky parantaa Fervor-kortin käyttäjää enemmän.

Kuten synergia- ja vastamekaniikkaesimerkeistä voidaan huomata, myös yksinkertaisilla ja automaattisilla mekaniikoilla voidaan luoda mielekkäitä valintoja ja vivahteikkaita tasapainoja pelikokemukseen. Yhdistelemällä kortteja eri tavoin kentälle pelaaja voi luoda hyvin erilaisia joukkueita, jotka toimivat selvästi eri tavoilla. Jokainen hahmo voidaan muokata todella monella tavalla, sillä niillä on omien erikoisvoimiensa ja tasojensa lisäksi mikä tahansa yhdistelmä lumous- ja varustekorteista, jotka molemmat vaikuttavat sekä hahmon että toistensa toimintaan.

4 Pelin toteutus

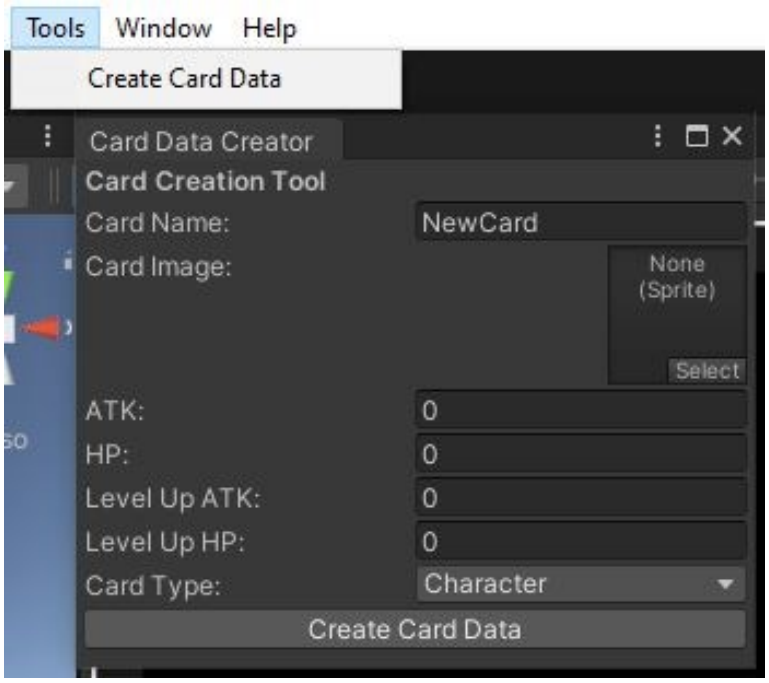
DACAB ohjelmoitiin ja rakennettiin C#-ohjelmointikielellä Unity 3D -pelimoottorissa. Tässä luvussa esitellään pelin toteutuksen ratkaisuja, jotka ovat oleellisia kehitetyn DACAB-pelin kaltaisten pelien näkökulmasta.

Versionhallinta toteutettiin GitHub-palvelun Git-komentoriviohjelmalla. Palvelu mahdollisti usean kehittäjän muokata yhteisiä tiedostoja samanaikaisesti. Kehitysryhmä oli jo aiemmin käyttänyt Git-ohjelmaa pienempien projektien versionhallinnassa, joten DACAB-pelin versionhallinta saatiin sujuvasti järjestettyä. Aluksi luotiin GitHub-palveluun uusi arkisto, jonne kaikki kehittäjät lisättiin muokkaajiksi. Jaettavaan arkistoon siirrettiin kansio, jossa Unity 3D -projektin tiedostot sijaitsivat. Jokaiselle kehittäjälle luotiin oma Branch-haarautuma eli paikallinen versio kansiosta, jossa tiedostot sijaitsevat. Kehittäjät voivat muokata omaa haarautumaansa ja puskea muutokset pilvipalveluun jaettavaksi, kun muutoksia halutaan yhdistää muihin haarautumiin. Kansioon lisättiin .gitignore.txt -tekstitiedosto, joka estää turhien välitiedostojen lähettämisen ja monistumisen haarautumia yhdistettäessä. (20; 21).

4.1 Edustaratkaisut

Jokainen kortti on mahdollista ohjelmoida ja luoda erikseen, mutta se olisi työlästä. Koska kortit toimivat keskenään samankaltaisesti vain pienillä eroilla, ne luotiin Unity 3D -pelimoottorin ScriptableObject-olioina projektiin luodusta luokasta, joka sisältää korttien toiminnan määrittävät muuttujat, kuten hyökkäysvoimat ja elämäpisteet eri tasoilla ja listan korttien erityiskyvyistä. ScriptableObject-oliot ovat tiedonkäsittelyllisiä säiliöitä, joihin voi tallentaa dataa siten, ettei toistuvat informaatiot, kuten insinöörityön tapauksessa korttien tiedot, vie lisää muistia. Unityn EditorWindow-luokka mahdollistaa käyttäjien omien syöttöikkunoiden luomisen muokkausohjelmassa. Uusien korttien lisäämisen helpottamiseksi luotiin oma työkalu kirjoittamalla luokka, joka perii EditorWindow-luokan ominaisuudet ja käyttää pelimoottorin sisäisen AssetDatabase-luokan CreateAsset-funktiota ScriptableObject-olioiden luomiseen. AssetDatabase-luokan CreateAsset-

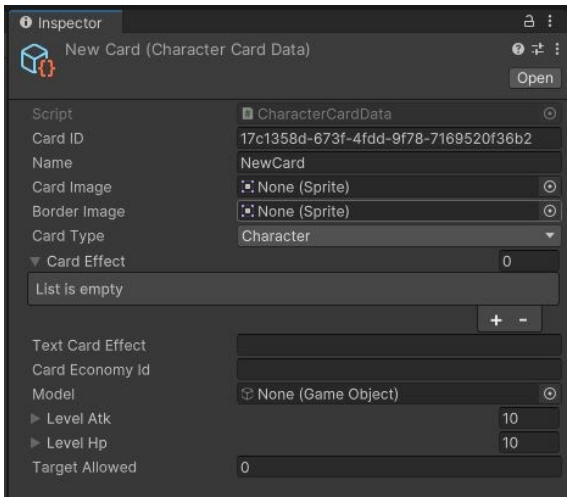
funktiolla voidaan luoda Unity-projekteihin pelimoottorin natiiveja resursseja kuten ScriptableObject-olioita. (22; 23; 24.) Kuvassa 13 näkyy korttien luomisen työkalu Unity-projektin muokkausohjelmassa.



Kuva 13. Korttien luomisen työkaluikkuna.

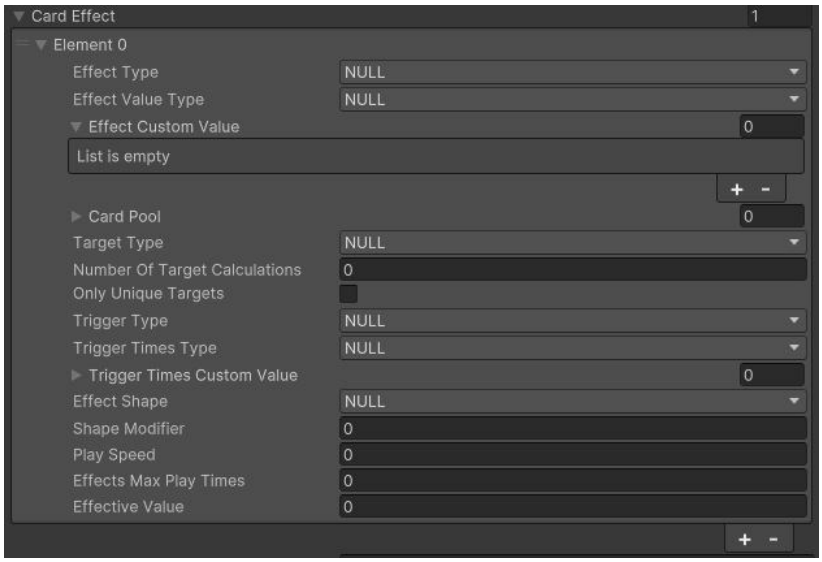
Kuvassa 13 esitettyyn työkaluun syötetään halutut muuttujat, ja painamalla työkalun alareunassa näkyvää nappia kortti luodaan osaksi projektia. Kortit tunnistetaan ohjelmakoodissa niiden luomisen yhteydessä generoidusta yksilöllisestä merkkiyhdistelmästä. Lisäksi näitä merkkiyhdistelmiä hyödynnetään tunnistamaan, millä lumouksella ja varusteella hahmot on varustettu, ja pakanrakennuksessa käsittelemällä merkkiyhdistelmiä merkkijonolistoissa.

Kuvassa 14 näytetään, minkälainen ScriptableObject-olio luodaan, kun kuvan 13 alareunan Create Card Data -nappia painetaan.



Kuva 14. Esimerkki kortin ScriptableObject-olioista.

Pelin hahmojen hyökkäykset, tuhoutuminen, liikkuminen ja pelien kulun logiikat toimisivat ilmankin kuvan 14 Card Effect -listan alkioita, mutta suurin osa pelin toiminnallisuudesta, kuten hahmojen erikoiskyvyt sekä loitsu-, varuste- ja lumouskorttien toiminnallisuus, toteutettiin omassa korttien vaikutuksia käsittelevässä luokassa. Kuvassa 15 näkyy kuvan 14 Card Effect -listaan lisätty alkio.



Kuva 15. Korttien vaikutuksia käsittelevän luokan ominaisuudet ScriptableObject-olion tarkasteluikkunan Card Effect -listassa.

Kaikkien korttien toiminnallisuudet toteutettiin muokkaamalla kuvan 15 muuttujia luotujen korttien ScriptableObject-olioissa. Jos peliin luodaan uudenlainen toiminnallisuus, saatetaan joutua muokkaamaan korttien vaikutusta käsittelevää luokkaa, mutta kuvan 15 muuttujilla voidaan jo sellaisenaan luoda uudenlaisia kortteja, jotka tekevät erilaisia toiminnallisuuksia pelin eri vaiheissa.

Taistelut lasketaan ohjelmakoodissa niiden kulkua käsittelevässä luokassa, jossa taisteluiden vaiheet on listattu, ja niissä suoritettavaa koodia suoritetaan kutsumalla vaiheiden suorittamiseen luotua funktiota, kunnes saavutetaan jokin taisteluiden loppumisehdoista. Vaiheiden suorittamisen funktiosta kutsutaan taistelun vaiheen tai taistelun tapahtumien mukaan hahmojen, lumousten ja varusteiden erikoiskykyjä käyttäen toimintojen suorittavaa funktiota. Funktio on määritelty korttien vaikutusten luokassa ja esimerkkinä tapahtumasta, joka aiheuttaa funktion kutsumisen, on hahmojen tuhoutuminen. Funktion toiminnallisuus riippuu kuvassa 15 listatuista ominaisuuksista. Jotta pelaajat eivät voi yrittää manipuloida pelien kulkua, taistelu lasketaan ensin ja tapahtumat tallennetaan tapahtumalistaksi, joka visualisoidaan pelaajalle vasta laskennan jälkeen.

Pelin grafiikoissa hyödynnettiin testaamisen nopeuttamiseksi aluksi tekoälytyökalua nimeltä Midjourney. Tekoälygrafiikoiden tarkoitus oli kuitenkin vain testaamisen nopeuttaminen ja tekoälygrafiikoita korvattiin kehitysryhmän itse luomalla grafiikalla, jota näkyy kuvien 7–12 korttien kuvissa. Korttien grafiikka luotiin Procreate-ohjelmalla.

4.2 Taustajärjestelmien ratkaisut

Auto battler -genren pelit eivät vaadi korkeita kutsumääriä palvelimen ja päätelaitteiden välille. Pelaajat eivät voi tehdä toimia reaaliajassa, vaan pelaaja pelaa kortit paikallisesti ja lähettää palvelimelle kutsun, kun hän on tyytyväinen joukkoonsa ja tahtoo alkaa etsiä vastustajaa. Pienen kutsumäärän ja kehitysryhmän heikon taustajärjestelmäosaamisen takia päätettiin, että taustajärjestelmiä ei ohjelmoitaisi ja ylläpidettäisi itse, vaan valittiin sopiva valmis ja mahdollisimman yksinkertainen taustajärjestelmäpalvelu pelille.

Aluksi tutkittiin mahdollisuutta käyttää Microsoft-yhtiön Azure PlayFab -palvelua, mutta huomattiin nopeasti, ettei se vaikuttanut helposti sovellettavalta projektissa. Ilmaiseksi tarjoamillaan ominaisuuksilla Azure PlayFab -palvelulla ei pystytty tallentamaan riittävästi informaatiota pelaajaprofiilien yhteyteen, jotta pelin toiminnallisuuksia olisi voitu toteuttaa.

Lopulta DACAB-pelissä päädyttiin käyttämään Unity 3D -pelimoottorin omaa taustajärjestelmäpalvelua Unity Gaming Services. Unity Gaming Services tarjoaa pelaajien tunnistautumisen, mahdollisuuden liittää pelaajiin dataa, taustajärjestelmän puolella suoritettavia ohjelmakoodeja, pelille luotavan talousjärjestelmän ja kattavat dokumentaatiot. Sen lisäksi projektiin voitiin joustavasti liittää vain tarvittavat taustajärjestelmäpalvelut. Unity Gaming Services on maksuttomasti käytettävissä määritetyillä rajoituksilla, joiden ylityttyä kehittäjiä veloitetaan käytön mukaisesti (25). DACAB pystyttiin kehittämään ja testaamaan ilmaiseksi.

Unity Gaming Services -palvelun käyttöönottamiseksi piti luoda projekti pelille Unity Cloud Dashboard -palvelussa ja yhdistää se Unity-projektiin. Sen jälkeen asennettiin pelin toiminnallisuuden kannalta tarpeelliset paketit Unity Editor -ohjelman sisältä löytyvästä pakettien hallinnan työkalusta. (26.) DACAB hyödyntää Unity Gaming Services -paketteja Authentication, Cloud Save, Cloud Code, Economy ja Leaderboards.

Viimeinen vaihe pakettien käyttöönottamiselle oli suorittaa alustuskoodi pelin ohjelmakoodissa. Esimerkkikoodissa 1 esitellään, miten Unity Gaming Services -palvelu alustetaan koodissa.

```
using System;
using Unity.Services.Core;
using UnityEngine;

public class InitializationExample : MonoBehaviour
{
     async void Awake()
     {
          await UnityServices.InitializeAsync();
     }
}
```

Esimerkkikoodi 1. Esimerkki C#-luokasta, jolla Unity Gaming Services -palvelut voidaan alustaa Unity-projektissa (25).

Pelaajien tunnistautumiseen käytettiin Authentication-pakettia. Aluksi testaamiseen käytettiin paketin tarjoamaa nimetöntä tunnistautumismahdollisuutta, jolloin pelaajan tiedot ja pelaajadata oli yhdistetty laitteeseen, jolla nimetön tunnistautuminen suoritettiin. Pelaajat kuitenkin menettäisivät keräämänsä kortit, luomansa pakat ja peliensä tilanteet, jos mobiililaitteella ei voitaisi enää pelata, joten myöhemmin tunnistautuminen vaihdettiin käyttäjänimellä ja salasanalla toteutettavaan tunnistautumiseen. Esimerkkikoodissa 2 esitellään, miten ohjelmakoodissa voidaan suorittaa nimetön tai käyttäjätunnuksilla tehtävä tunnistautuminen.

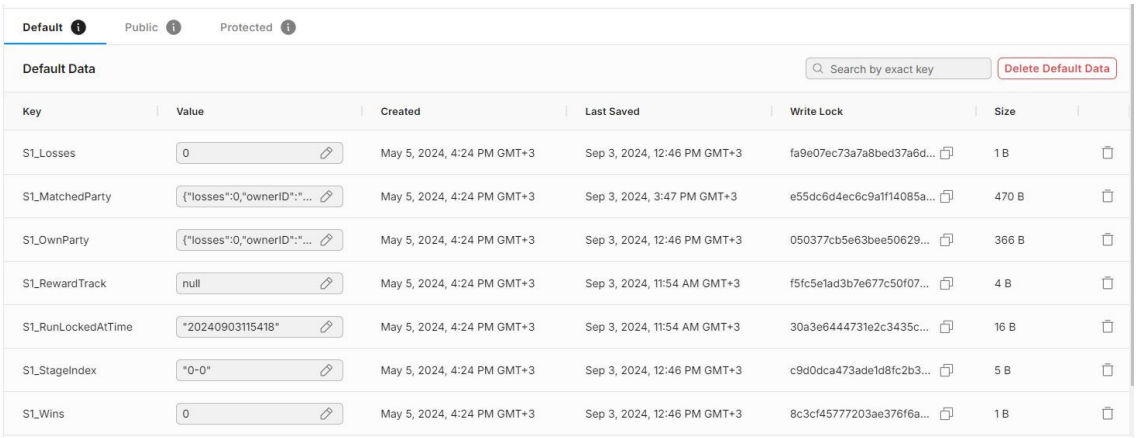
```
async Task SignInAnonymouslyAsync()
{
     await AuthenticationService.Instance.SignInAnonymouslyAsync();
}

async Task SignUpWithUsernamePasswordAsync(string username, string
password)
{
     await AuthenticationService.Instance.SignUpWithUsernamePass-
     wordAsync(username, password);
}
async Task SignInWithUsernamePasswordAsync(string username, string
password)
{
     await AuthenticationService.Instance.SignInWithUsernamePass-
     wordAsync(username, password);
}
```

Esimerkkikoodi 2. Funktiot Unity Gaming Services -palvelussa tunnistautumiseen.

Esimerkkikoodin 2 ylimmässä funktiossa esitellään, miten Authentication-pakettia voidaan yksinkertaisimmillaan käyttää pelaajan nimettömään

tunnistautumiseen. Keskimmäinen esimerkkifunktio rekisteröi pelaajan palveluun käyttäen parametreihin username ja password syötettyä käyttäjänimeä ja salasanaa, minkä jälkeen rekisteröintifunktiossa käytettyjä tunnuksia voi käyttää alimmalla esimerkkifunktiolla tunnistautumiseen. Tunnistautuminen voidaan suorittaa muillakin tavoilla, joita on listattu Unity-ohjelmiston dokumentaatiossa. (27.)

Cloud Save -paketti mahdollistaa informaation tallentamisen tunnistautuneisiin pelaajakäyttäjiin. Pakettia käytettiin myös pelaajien datan hakemiseen palvelimelta. Kuvassa 16 esitellään miltä pelaajien data näyttää Unity Cloud Dashboard -palvelussa.



Kuva 16. Pelaajan dataa Unity Cloud Dashboard -palvelussa.

Kehittäjät voivat tarkastella kuvan 16 mukaisessa käyttöliittymässä pelaajien tietoja, joita koodista pystytään muokkaamaan tai lukemaan esimerkkikoodin 3 mukaisesti.

```
public async void SaveData()
{
     var playerData = new Dictionary<string, object>{{"S1_Losses", 1}};
     await CloudSaveService.Instance.Data.Player.SaveAsync(playerData);
}

public async void LoadData()
{
     var playerData = await CloudSaveService.In-
     stance.Data.Player.LoadAsync(new HashSet<string> {"S1_Losses"});

     if (playerData.TryGetValue("S1_Losses", out var lossesFetched)) {
          Debug.Log("Losses in S1:{loss-
          esFetched.Value.GetAs<string>()}");
      }
}
```

Esimerkkikoodi 3. Pelaajaan yhdistettävän datan tallentamisen ja hakemisen esimerkkikoodit.

Esimerkkikoodin 3 SaveData-funktiolla voidaan tallentaa arvo 1 kuvan 16 avaimeen S1_Losses. Koodiesimerkin funktio LoadData puolestaan hakee arvon avaimesta S1_Losses ja tulostaa haetun arvon Unity-projektin muokkausohjelman konsoliin. Pelien tilanne, pelaajien rakentamat pakat ja pelaajien peliversiotunnisteet ovat tallennettuina pelaajadataan.

Unity Gaming Services -palvelun Cloud Code -paketti mahdollistaa palvelimien puolella ajettavat ohjelmakoodit. Palvelimen puolella ajettava koodi kirjoitetaan esimerkkikoodin 4 mukaisesti JavaScript-ohjelmointikielellä.

```
const _ = require("lodash-4.17");
const axios = require("axios-0.21");
const { DataApi } = require("@unity-services/cloud-save-1.4");

module.exports = async ({ params, context, logger }) => {
     const { projectId, playerId } = context;
     const config = { headers: { 'Content-Type': 'application/json',
Authorization: `Bearer ${context.serviceToken}` } };
     const api = new DataApi(context);
     const setID = await api.setItem(projectId, playerId, { key: "Id",
value: playerId });

     for(let i = 0; i < 5; i++){
          const slot = i + 1;
          const stageIndexKey = "StageIndex" + slot;
          const isSearchableKey = "IsSearchable" + slot;
          const winsKey = "Wins" + slot;
          const lossesKey = "Losses" + slot;
          const matchedPartyKey = "MatchedParty" + slot;
          const ownPartyKey = "OwnParty" + slot;

          const stageIndexValue = await api.setItem(projectId, play-
erId, { key: stageIndexKey, value: " " });
          const isSearchableValue = await api.setItem(projectId, play-
erId, { key: isSearchableKey, value: false });
          const winsValue = await api.setItem(projectId, playerId, {
key: winsKey, value: 0 });
          const lossesValue = await api.setItem(projectId, playerId, {
key: lossesKey, value: 0 });
          const matchedPartyValue = await api.setItem(projectId, play-
erId, { key: matchedPartyKey, value: null });
          const ownPartyValue = await api.setItem(projectId, playerId,
{ key: ownPartyKey, value: null });
     }
     return {responseKey: "Success"};
}
```

Esimerkkikoodi 4. Unity Cloud Dashboard -palvelussa sijaitseva palvelimen puolella ajettava ohjelmakoodi ClearGamesData.

Esimerkkikoodissa 4 esitetty ohjelmakoodi nollaa pelaajan peleihin liittyvät tiedot palvelimella. Palvelimen puolella ajettavaa ohjelmakoodia voidaan kutsua pelistä esimerkkikoodin 5 esittämällä tavalla.

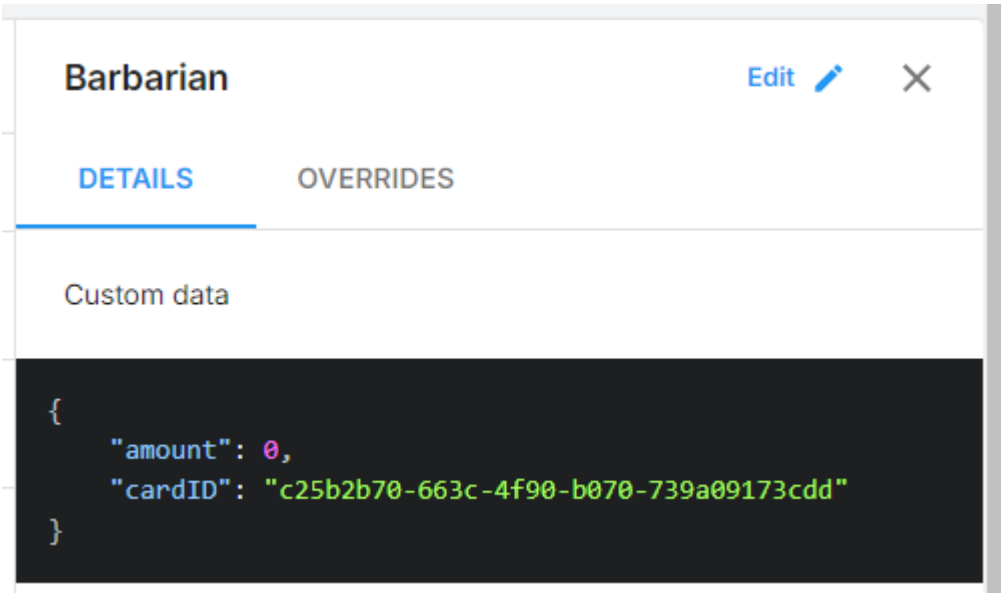
```
public class CloudClearPlayersGamesData : MonoBehaviour
{
     class CloudCodeResponseCatcher
     {
          public string responseKey;
     }

     public async Task CallClearGamesData()
     {
          CloudCodeResponseCatcher response = await CloudCodeServ-
ice.Instance.CallEndpointAsync<ClearCloudSaveResponse>("ClearGames-
Data");
          Debug.Log("Response: " + response.responseKey);
     }
}
```

Esimerkkikoodi 5. C#-luokka, jonka funktiolla CallClearGamesData voidaan kutsua esimerkkikoodin 4 palvelimen puolella ajettavaa ohjelmakoodia.

Esimerkkikoodilla 5 voidaan ajaa esimerkkikoodissa 4 esitetty palvelimen puolen ohjelmakoodi, kun palvelimen puolella suoritettava ohjelma on nimetty ClearGamesData-nimiseksi.

Palvelimen puolella suoritettavia ohjelmia käytettiin DACAB-pelissä pelitilanteiden datan muokkaamiseen, otteluiden vastustajien muodostamiseen ja tulostaulukkojen päivittämiseen. Cloud Code -ohjelmakoodeja käytettiin myös pelin version tarkistamiseen.

Economy-paketti mahdollistaa pelin resurssijärjestelmien hallinnan. Peliin voidaan lisätä Ecomony-paketin avulla pelinsisäisiä valuuttoja ja pelaajien virtuaalisesti omistamia tavaroita. Paketti soveltuu hyvin keräilykorttipeleihin, koska omistettaviin tavaroihin voidaan sisällyttää mukautettua dataa. Pelissä omistettujen korttien hallinta toteutettiin Economy-paketin tavaroilla, joihin liitettiin mukautettuna datana omistettujen korttien määrä ja yksilöivä merkkijono, jonka avulla pelin ohjelmakoodissa tunnistetaan, mitä kortteja pelaaja omistaa. Kuvassa 17 on omistettavan kortin, Barbarian, määrittely Unity Cloud Dashboard -palvelussa. (29.)



Kuva 17. Barbarian-kortin talousjärjestelmän tavaran määrittely Unity Cloud Dashboard -palvelussa.

Kun pelaaja luo käyttäjänsä, jokaista pelin korttia vastaava tavara annetaan pelaajalle, mutta pelaaja ei voi käyttää niitä pakanrakennuksessa, jos niiden sisältävän datan määrän arvo on nolla. Esimerkkikoodissa 6 esitellään, miten pelin ohjelmakoodissa voidaan muokaten tavaran dataa antaa tai poistaa pelaajalta kortteja.

```
public async Task GivePlayerACard(string _cardToGiveEconomyItemId)
{
     int newAmount = oldAmount + 1;
     Dictionary<string, object> instanceData = new Dictionary<string,
object> { { "amount", newAmount } };

     PlayersInventoryItem playersInventoryItem = await Econo-
myService.Instance.PlayerInventory.UpdatePlayersInventory-
ItemAsync(cardToGiveEconomyItemId, instanceData);
}
```

Esimerkkikoodi 6. Funktio, jolla voidaan lisätä pelaajalle kortti muokkaamalla pelaajan omistamaa talousjärjestelmän tavaraa.

Esimerkkikoodin 6 parametri ei ole sama kuin kortille luonnin yhteydessä määrittynyt yksilöivä merkkijono, vaan se on talousjärjestelmän tavaran luonnin yhteydessä määritetty yksilöivä merkkijono. DACAB-pelissä talousjärjestelmän tavaroiden yksilöivistä merkkijonoista tehtiin kortin nimen mukaisia, jotta niiden

käyttäminen olisi mahdollisimman helppoa. Esimerkkikoodin muuttujaan newAmount voidaan sijoittaa mikä vain määrä, ja se korvaa pelaajan korttia vastaavassa tavarassa aiemmin olleen omistuksien määrän.

Unity Gaming Services tarjoaa oman mainostusjärjestelmän Unity-projekteille, mutta peliin päädyttiin silti toteuttamaan mainokset ironSource-palvelun avulla. IronSource mahdollistaa sellaisten mainosvideoiden näyttämisen, joiden katsomisesta voidaan palkita pelaajaa. Kun DACAB-pelissä yhdelle joukkueelle kertyy kolme häviötä, peli päättyy ja pelaaja saa palkinnot voittojen määrän perusteella. Sen jälkeen uutta peliä ei voi aloittaa samassa paikassa määrätyn ajan sisällä paitsi katsomalla mainosvideon.

IronSource-mainosten käyttöönottamiseksi piti ensin luoda ironSource-käyttäjä ja asentaa ironSource SDK -ohjelmistokehityspaketti ja LevelPlay SDK -ohjelmistopaketti ja lisätä ne Unity-projektiin. IronSource-palvelussa määritettiin, millaisia mainoksia pelissä halutaan näyttää ja lisättiin mainostusverkkoja, joiden mainoksia näytetään. (30.) Mainoksia oli helppo kutsua pelin ohjelmakoodista, mutta mainosverkkojen ja tarvittavien lisäosien määrittäminen oli aikaa vievää.

Analytiikat yritettiin toteuttaa Googlen Firebase -analytiikkapalvelulla. Analytiikkojen alustamiseksi luotiin Firebase-projekti, rekisteröitiin sovellus Firebase-palvelussa, lisättiin tarvittavat määrittelytiedostot Unity-projektiin ja asennettiin Firebase Unity SDK -paketit, jotka myös lisättiin Unity-projektiin (31). Firebase-palvelun avulla voidaan kerätä analytiikkoja pelaajien toiminnoista joko valmiiden tapahtumien avulla tai ohjelmakoodissa voidaan luoda mukautettuja analytiikkatapahtumia. Kaikki tapahtumat näkyvät kehittäjille Firebase-palvelussa.

4.3 Kohdatut ongelmat

Pelin toteutuksessa kohdattiin useita ongelmia, jotka hidastivat kehitystyötä, mutta ne pystyttiin lopulta ratkaisemaan. Hidasteita kohdattiin eniten taustajärjestelmiä toteutettaessa ja pelin jakamisessa.

Yksi ongelmista oli pakata pelaajien joukkueet JSON-tiedostomuotoihin ja lähettää pakattuja JSON-tiedostoja palvelimen ja pelin välillä. Koska koko prosessi ei tapahdu vain yhdessä ohjelmassa, tilanteiden virheiden korjaus oli vaikeaa ja virheviestit epäselviä. DACAB-pelin kaltaisten pelien taustajärjestelmäratkaisuista ei löytynyt monia esimerkkejä eikä kehitysryhmällä ollut paljoa kokemusta JSON-tiedostomuotojen käsittelystä. JSON-tiedostot olivat kuitenkin yksinkertaisin tapa tallentaa pelien tietoja palvelimelle ja saatiin lopulta toimimaan noin kahden viikon virheentunnistustyön jälkeen.

Peliä jaettiin testattavaksi Google Play Store -mobiilisovelluskaupassa. Mobiilisovelluskauppa nosti pelin alimman kohdistettavan ohjelmistorajapinnan tason vaatimusta pelin ollessa avoimessa testauksessa. Siitä aiheutuen monet pelin käyttämät ulkoiset työkalut, kuten ironSource- ja FireBase-palveluiden vaatimat SDK-paketit, aiheuttivat paljon yhteensopimattomuusongelmia. Pelistä ei pystytty enää viemään versioita Unity Editor -ohjelmiston ulkopuolelle ilman suurta päivitys- ja virheenkorjaustyötä. Kun virheet lopulta saatiin korjattua ja uusin versio lähetettiin tarkastettavaksi Google Play Store -palveluun, automaattinen tarkastus hylkäsi version ja jäädytti kehitysryhmän Google-kehittäjäkäyttäjällä sovelluksen. Syiksi listattiin haittaohjelmistot, sovelluksen toiminnan selkeyden puute ja haitalliseksi luokiteltava mobiilisovellus. Vuorokauden sisään sovelluksen jäädyttämisestä kehitysryhmän Google-kehittäjäkäyttäjä poistettiin. Kehitysryhmän kesken tutustuttiin tarkemmin listattuihin käytänteisiin eikä ymmärretty, miten sovellus rikkoisi niitä, joten asiakastukeen otettiin yhteyttä. Vastauksissa asiakaspalvelusta todettiin, että Google Play Team pysyisi päätöksessään ja kehitysryhmältä pyydettiin, ettei uusia kehittäjäkäyttäjiä luotaisi. Peliä yritetään kuitenkin siirtää uusille jakoalustoille tulevaisuudessa.

5 Yhteenveto

Työn tavoitteena oli suunnitella ja luoda toimiva asynkroninen auto battler -genren keräilykorttipeli. DACAB-pelistä saatiin aikaiseksi hyvin toimiva demoversio, jossa toteutuu pelille suunnitellut tärkeimmät ominaisuudet. Pelissä toteutuvia tärkeimpiä ominaisuuksia ovat pelaajan tunnistautuminen, korttien kerääminen ja pelien edistyminen. Mekaniikoista saatiin toimivia ja suhteellisen helposti muokattavia ja lisättäviä. Kerättäviä kortteja toteutettiin kymmeniä ja suurin osa korteista toimii pelissä mielekkäästi. DACAB-pelin kokonaisvaltaisesta pelattavuudesta saatiin tavoitteiden mukaisesti joustava ja helppo, mutta kuitenkin mielekäs.

Peliin ei lisätty vielä ääniä, ja pelin graafinen ilme on hieman keskeneräinen. Pelien taisteluissa käytetty satunnaisuus lasketaan päätelaitteella, joka voi johtaa samojen taisteluiden eri lopputulemiin osapuolien kesken. Lisäksi käyttäjäkokemus on yleisesti hiomaton ja erityisesti pakkojen rakennuksen ja kokoelman selaamisen osalta huono. Vaikka DACAB oli kuukausia Google Play Store -mobiilisovelluskaupan sisäisessä testauksessa ja lyhyesti myös avoimessa testauksessa, peliä ei pystytä kätevästi jakamaan ennen kuin löydetään uusi jakoalusta, koska kehitysryhmän Google-kehittäjätili on poistettu.

Pelillä on paljon jatkokehitysmahdollisuuksia. Puuttuvat äänielementit ja grafiikat voidaan lisätä ja käyttäjäkokemussuunnittelua tarvitaan paljon. Lisäksi kerättäviä ja pelattavia kortteja tarvitaan vielä lisää, jotta pakkoihin ja strategioihin saadaan vaihtelua ja pelin keräilyelementistä tulee oleellisempi pelikokemuksen kannalta. DACAB tarvitsee toimivan analytiikkapalvelun, jotta jatkokehityksessä ja korttien tasapainottamisessa voidaan hyödyntää kerättyä dataa. Firebase-palvelu saatiin toimimaan osittain, mutta muokattavat analytiikkatapahtumat jäivät puuttumaan ja ne ovat jatkokehityksen kannalta tärkeitä.

Työllä todistettiin, että verkossa pelattava mobiilipeli voidaan suunnitella ja toteuttaa pienellä budjetilla ja pienellä kehitysryhmällä hyödyntäen ilmaisia, kohtuuhintaisia ja joustavasti hinnoiteltuja palveluja. Työn aikana huomattiin, että

valittaessa palveluja ja työkaluja kannattaa olla kuitenkin tarkka, koska jotkin palvelut ja työkalut soveltuvat joillekin peleille paremmin kuin toiset. DACAB-pelin taustajärjestelmiin suunniteltiin käytettävän Azure PlayFab -palvelua, mutta löydettiin parempi vaihtoehto Unity Gaming Services. Etenkin pienten kehitysryhmien pitää olla tarkkoja palveluiden käytänteiden noudattamisessa.

Lähteet

1 Schell, Jesse. 2019. The Art of Game Design, 3rd Edition. E-Kirja. A K Peters/ CRC Press.

2 Hämäläinen, Perttu &Takatalo, Jari. 2017. Millainen peli koukuttaa ja tuottaa mielihyvää? Verkkoaineisto. Aikakausikirja Duodecim. <https://www.duodecimlehti.fi/duo14046>. Päivitetty 2017. Luettu 17.9.2024.

3 Xu, Jiayu ym. 2020. Lineup Mining and Balance Analysis of Auto Battler. Verkkoaineisto. <https://doi.org/10.1109/ICCC51575.2020.9345074> Luettu 7.11.2024

4 Mobile Games Vs. PC Vs. Console Games: What Market is the Best Bet?. Verkkoaineisto. Starloop Studios. <https://starloopstudios.com/mobile-games-vs-pc-vs-console-games-what-market-is-the-best-bet/>. Luettu 18.9.2024.

5 Unreal vs. Unity 3D: Choosing the Best Engine for Your Game. Verkkoaineisto. Starloop Studios. <https://starloopstudios.com/unreal-vs-unity-3d-choosing-the-best-engine-for-your-game/>. Luettu 18.9.2024.

6 Number of available applications in Google Play Store from March 2017 to June 2024. Verkkoaineisto. Statista. <https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/>. Luettu 18.9.2024.

7 App Store Sets New Records With $240M in Sales on New Year's Day, $20B Paid to Developers in 2016. Verkkoaineisto. MacRumors. <https://www.macrumors.com/2017/01/05/apple-busiest-day-app-store-imessage/>. Luettu 18.9.2024.

8 App Review Guidelines. Verkkoaineisto. <https://developer.apple.com/app-store/review/guidelines/>. Luettu 18.9.2024.

9 Prepare your app for review. Verkkoaineisto. <https://support.google.com/googleplay/android-developer/answer/9859455?hl=en>. Luettu 18.9.2024.

10 Cotton, Brandon & Fields, Tim. 2014. Mobile & Social Game Design, 2nd Edition. E-Kirja. A K Peters/ CRC Press.

11 Clark, Oscar. 2014. Games as a service: how free2play design can make better games. E-Kirja. Focal Press.

12 Knezovic, Andrea. 2024. 200+ Mobile Games Statistics: Market & Revenue Report [2024]. Verkkoaineisto. <https://www.blog.udonis.co/mobile-marketing/mobile-games/mobile-gaming-statistics>. Luettu 19.9.2024.

13 Brazie, Alexander. Designing The Core Gameplay Loop: A Beginner's Guide]. Verkkoaineisto. <https://gamedesignskills.com/game-design/core-loops-in-gameplay/>. Luettu 19.9.2024.

14 Mattinen, Topias ym. 2023. A Ruse by Any Other Name: Comparing Loot Boxes and Collectible Card Games Using Magic Arena. Artikkeli. Proceedings of the ACM on human-computer interaction, 7(CHI PLAY), s. 721–47.

15 Aagaard, Jacob ym. 2022. A Game of Dark Patterns: Designing Healthy, Highly-Engaging Mobile Games. Verkkoaineisto. <https://doi.org/10.1145/3491101.3519837>. Luettu 7.11.2024.

16 HOW TO PLAY MAGIC: THE GATHERING. Verkkoaineisto. <https://magic.wizards.com/en/how-to-play#key-card-types>. Luettu 23.9.2024.

17 Card. Verkkoaineisto. <https://hearthstone.wiki.gg/wiki/Card#Collectible_cards>. Luettu 23.9.2024.

18 Habgood, Jacob & Overmars, Mark. 2006. The Gamer Maker's Apprentice: Game Development for Beginners. E-kirja. Apress.

19 Map Balance. Verkkoaineisto. The Level Design Book. <https://book.leveldesignbook.com/process/combat/balance>. Luettu 25.9.2024.

20 An Intro to using GitHub with Unity for Beginners #56071. Verkkoaineisto. <https://github.com/orgs/community/discussions/56071>. Luettu 30.9.2024.

21 Unity.gitignore. Verkkoaineisto. <https://github.com/github/gitignore/blob/main/Unity.gitignore>. Luettu 30.9.2024.

22 ScriptableObject. Verkkoaineisto. Unity Documentation. <https://docs.unity3d.com/Manual/class-ScriptableObject.html>. Luettu 30.9.2024.

23 EditorWindow. Verkkoaineisto. Unity Documentation. <https://docs.unity3d.com/ScriptReference/EditorWindow.html>. Luettu 30.9.2024.

24 AssetDatabase.CreateAsset. Verkkoaineisto. Unity Documentation. <https://docs.unity3d.com/ScriptReference/AssetDatabase.CreateAsset.html>. Luettu 30.9.2024.

25 Pricing. Verkkoaineisto. <https://unity.com/products/gaming-services/pricing>. Luettu 30.9.2024.

26 Get Started With UGS. Verkkoaineisto. Unity Documentation. <https://docs.unity.com/ugs/manual/overview/manual/getting-started>. Luettu 30.9.2024.

27 Approaches to authentication. Verkkoaineisto. Unity Documentation. <https://docs.unity.com/ugs/en-us/manual/authentication/manual/approaches-to-authentication>. Luettu 30.9.2024.

28 Unity SDK tutorial. Verkkoaineisto. Unity Documentation. <https://docs.unity.com/ugs/en-us/manual/cloud-save/manual/tutorials/unity-sdk>. Luettu 30.9.2024.

29 Economy. Verkkoaineisto. Unity Documentation. <https://docs.unity.com/ugs/en-us/manual/economy/manual>. Luettu 1.10.2024.

30 Getting started with Unity. Verkkoaineisto. <https://developers.is.com/ironsource-mobile/unity/levelplay-starter-kit/#step-1>. Luettu 1.10.2024.

31 Add Firebase to your Unity project. Verkkoaineisto. <https://firebase.google.com/docs/unity/setup>. Luettu 1.10.2024.