Karelia-ammattikorkeakoulu
Tietojenkäsittely

Dialogijärjestelmät ja pelaajakäyttäytymisen analytiikka videopeleissä

Älykkäät vuorovaikutusmallit ja pelaajien toiminnan ymmärtäminen

Matias Niinivirta

Opinnäytetyö, Tammikuu 2026

www.karelia.fi

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Helmikuu 2026
Tietojenkäsittely

Tikkarinne 9
80200 JOENSUU
+358 13 260 600 (vaihde)

Tekijä(t)
Matias Niinivirta

Nimeke
Dialogue Systems and Player Behaviour in Video Games

Toimeksiantaja

Olipa Games Oy

Tiivistelmä

Opinnäytetyön tavoitteena oli kehittää analytiikkalokittaja, joka kerää pelaajien tietoja ja tallentaa ne tietokantaan. Opinnäytetyötä varten kerätty data perustuu pelaajien tekemiin dialogivalintoihin ja sen avulla pyrittiin parantamaan pelikehitystä, erityisesti tarinankerronnan, dialogin ja pelaajakokemuksen kannalta.

Työn teoriapohjassa tarkasteltiin tarinankerrontaa osana pelikokemusta ja sen merkitystä. Lisäksi tarkasteltiin erilaisia dialogirakenteita tarinankerronnallisena elementtinä ja analytiikan roolia pelisuunnittelussa. Analytiikkalokittaja rakennettiin Unity-ympäristössä ja liitettiin tietokantaan, jonka käyttöliittymästä data oli helposti ladattavissa analyysia varten. Kerättyä dataa analysoitiin ja tulkittiin pelaajakäyttäytymisen näkökulmista.

Työn tuloksena saatiin selville, että pelaajakäyttäytymistä seuraamalla saadaan arvokasta tietoa pelaajien mieltymyksistä ja pelin toimivuudesta. Analytiikka tukee parhaimmillaan päätöksentekoa, ohjaa tarinallista suunnittelua ja auttaa kehittämään vaikuttavampia pelikokemuksia. Työ osoitti myös, että vaikka analytiikka ei yksin ratkaise kaikkia pelisuunnittelun haasteita, se tarjoaa perustan kehitystyölle.

Kieli
suomi

Sivuja 64
Liitteet 0
Liitesivumäärä 0

Asiasanat
analytiikkalokittaja, dialogi, pelaajakäyttäytyminen, videopelit, analytiikka

THESIS
February 2025
Business Information Technology

Tikkarinne 9
FI 80200 JOENSUU
FINLAND
Tel. +350 13 260 600

Author (s)
Matias Niinivirta

Title

Dialogue Systems and Player Behavior Analytics in Video Games: Intelligent Interaction Models and the Analysis of Player Behavior

Commissioned by[i]
Olipa Games

Abstract

The objective of this thesis was to develop an analytics logger that collects player data and stores it in a database. The data collected for this study is based on players' dialogue choices, and it was used to improve game development, particularly in the terms of storytelling, dialogue and overall player experience.

The theoretical foundation of the works examined storytelling as a component of the gaming experience, and it is significance. Additionally, different dialogue structures were analysed as a narrative element, along with the role of analytics in game design. The analytics logger was implemented in Unity environment and connected to a database, whose interface allowed an easy data extraction for analysis. The collected data was analysed and interpreted from the perspective of player behaviour

As a result, it was found that monitoring player behaviour provides valuable insights into player preferences and the functionality of the game. Analytics can support decision-making, guide narrative design, and help develop more engaging and meaningful gaming experiences. The study also demonstrated that, while analytics cannot solve all the challenges in game development, it provides a solid foundation for development.

Language
Finnish

Pages 64
Appendices 0
Pages of Appendices 0

Keywords
analytics logger, dialogue, player behavior, videogames, analytics

Sisältö

1 Johdanto

1.1 Tausta ja työn tavoitteet

Videopelit ovat graafisia ja animaatioihin perustuvia elektronisia pelejä, joita ohjataan käyttöliittymän kautta näyttölaitteella (Sanastokeskus 2013). Nykyaikana videopelit ovat saavuttaneet laajan suosion viihteenä. Useat pelit tarjoavat mahdollisuuden vuorovaikutteiseen tarinankerrontaan ja niillä on monia samankaltaisia tarinankerronnallisia elementtejä kuin kirjat ja elokuvat. Samoin kuin kirjoissa ja elokuvissa, tarina, hahmot ja hahmojen välinen dialogi muodostavat merkittävän osan kokemuksesta ja vaikuttavat siihen, miten yleisö eläytyy siihen. Peli kuitenkin eroaa elokuvista ja kirjoista niin, että pelaaja voi tehdä valintoja dialogissa, mitkä vaikuttavat tarinan kulkuun. Pelaajien tekemistä valinnoista on myös mahdollista kerätä dataa, jota voi käyttää esimerkiksi pelien jatkokehittämisessä, käyttäjäkokemuksessa ja jopa tieteellisessä tutkimuksessa.

Tässä työssä on tavoitteena tutkia, miten tarinankerronta peleissä muodostuu ja mikä johtaa paremmin pelaajien immersioon erityisesti dialogin kannalta. Pelaajan valintojen merkitystä tarkastellaan, jotta voidaan hahmottaa ja luoda pohjaa sille millaisia analytiikkamahdollisuuksia niiden pohjalta syntyy.

Työssä tarkastellaan myös analytiikan merkitystä peliteollisuudessa ja sitä, mitä analytiikalla tässä kontekstissa voidaan saavuttaa. Työssä käydään läpi tapoja, miten on mahdollista mitata pelaajakäyttäytymistä dialogissa ja miten valintoja kerätään ja seurataan. Lisäksi työssä vertaillaan dialogijärjestelmiä eri peleissä ja arvioimme kuinka ihanteellisia ne ovat datan keruun kannalta.

Toiminnallisessa osuudessa kehitetään analytiikkalokittaja. Analytiikkalokittaja on työkalu, joka kerää ja tallentaa tietoa käyttäjien toimista (Google Gemini 3 2026a).

Olipa Games on Helsingissä toimiva peliyritys, joka tuottaa pääsääntöisesti indie-pelejä. Työkalu rakennetaan toimeksiantajayrityksen, Olipa Gamesin kehitteillä olevaan peliin, jonka nimi on Hostile Docking.
Työn päätavoitteena on yhdistää dialogijärjestelmien suunnittelu ja pelaajakäyttäytymisen analytiikka siten, että niiden avulla voidaan lisätä ymmärrystä

pelaajan toimijuudesta ja immersiosta tarinallisissa peleissä. Tavoitteena on tuottaa sekä teoreettinen katsaus roolipelien dialogijärjestelmiin että käytännön toteutus, joka osoittaa, miten pelaajadataa voidaan kerätä ja hyödyntää pelisuunnittelun tukena. Lopuksi työssä arvioidaan kehitetyn järjestelmän vahvuuksia ja rajoituksia sekä pohditaan, millaisia mahdollisuuksia analytiikan ja narratiivisten rakenteiden yhdistäminen tarjoaa tulevaisuuden pelikehitykselle.

1.2 Rajaus

Monissa peleissä on mukana dialogijärjestelmä, muttei mahdollista tehdä valintoja sen sisällä. Tämä työ rajataan koskettamaan vain pelejä, joissa on mahdollista tehdä valintoja dialogin sisällä, koska tietojen kerääminen on turhaa, ellei mahdollisuutta valintaan ole. Mahdollisuus valintaan voi olla esimerkiksi erilaisissa roolipeleissä, kuten Fallout 4 (2015) tai Witcher 3 (2015): Wild Hunt. On myös olemassa niin sanottuja visuaalisia novelleja, jotka keskittyvät lähes kokonaan tarinan kerrontaan, jolloin varsinaiset pelilliset elementit jäävät toissijaisiksi. Näissä peleissä dialogi on keskeisessä roolissa, ja pelaajan tehtävänä on edetä tarinassa tekemällä valintoja keskusteluissa tai vuorovaikutuksessa muiden hahmojen kanssa (Stegner 2021). Tämä työ perustuu ehkä enemmän roolipeleihin kuin visuaalisiin novelleihin, mutta molemmat on hyvä tuoda ilmi.

Analytiikkaa tarkastellaan myös enemmän spekulatiivisesta näkökulmasta kuin konkreettisemmasta näkökulmasta. Tärkeämpää on käsitellä, mihin ja miten pelaajan valinnoista kerättyä tietoa on mahdollista käyttää kuin mitä nämä valinnat ovat olleet konkreettisesti. Toiminnallisessa osuudessa käydään läpi kyllä testituloksia, mutta nämä testitulokset tuotetaan keinotekoisesti vain, jotta prototyyppiä voidaan testata. Painopiste pysyy kuitenkin siinä, mihin tarkoituksiin analytiikkalokittajaa pystyy käyttämään.

1.3 Työn rakenne

Työ koostuu teoriaosuudesta, jossa käsitellään dialogin merkitystä tarinankerronnallisena välineenä ja pelaajan valintojen merkitystä suhteessa pelikokemukseen. Käydään myös läpi analytiikan merkitystä ja mitä pelaajakäyttäytymistä voidaan mitata konkreettisesti. Lisäksi arvioidaan mahdollisia analytiikan hyötyjä

koskien peliteollisuutta ja lopuksi vertailemme erilaisia tapoja toteuttaa dialogeja videopeleissä.

Lopputyö koostuu toiminnallisesta osuudesta, jossa luodaan analytiikkalokittaja toimeksiantajayrityksen (Olipa Games) videopeliin ja testataan sitä käytännössä, kuten aiemmin on mainittu. Analytiikkalokittajan testitulokset luodaan keinotekoisesti, mutta tarkoituksena on enemmän luoda prototyyppi ja spekuloida sille erilaisia käyttötarkoituksia kuin mitata aitoa dataa.

1.4 Tekoälyn käyttö opinnäytetyössä

Tekoälyä on käytetty opinnäytetyössä hiomaan kielioppia. Raaka teksti on aina kuitenkin kirjoitettu itse, mutta tekoälyltä on joskus pyydetty korjaamaan kielioppivirheet tai kysytty, miten sanoa tietty asia tehokkaammin.

Tekoälyä on myös käytetty selittämään tiettyjä sanoja, kuten ”analytiikkalokittaja”. Tämä ainoastaan sellaisissa tilanteissa, missä kyseiset sanan määritelmää ei löydy mistään muusta lähteestä, kuten sanakirjasta. Näissä tilanteissa tekoälyn lähteen on merkitty.

1.5 Lähteiden luotettavuus

Opinnäytetyön suunnitelmassa lähteet valittiin sillä periaatteella, että jokaisella aihealueella olisi ainakin yksi akateeminen, vertausarvioitu lähde. Tätä täydennettiin lähteillä, jotka syvensivät ja tukivat käsiteltävää.

Opinnäytetyön toteutusvaiheessa lähteitä käytettiin usein selittämään sanojen merkityksiä tai hahmottamaan erilaisia vaihtoehtoja analytiikkalokittajan toteutukselle. Tarkoituksena oli löytää sanoille merkitys mahdollisimman luotettavista lähteistä, kuten tieteellisistä termipankeista, mutta jos se ei ollut mahdollista, etsittiin merkitystä muista lähteistä, esimerkiksi tekoälyn avulla. Jos löytyi muita tapoja toteuttaa analytiikkalokittaja, esimerkiksi jokin kolmannen osapuolen työkalu. Lähteet haettiin suoraan tämän työkalun nettisivuilta.

2 Tarinankerronta ja immersio peleissä

Tarinankerronta on muinainen taiteenmuoto, joka tarkoittaa kertomusten esittämistä eri muodoissa. Sen tarkoitus on välittää tietoa, viihdyttää tai toimia opetuksen välineenä. (National Geographic Society 2025.) Videopeleissä tarinankerronta yhdistyy vuorovaikutukseen, jolloin pelaaja ei ainoastaan seuraa tarinaa vaan osallistuu sen kulkuun valintojensa kautta. Videopelit tunnetaan pääosin viihteenä, mutta usein viihteelliseen tarinankerrontaan on sekoitettu opettavaistakin tietoa. Esimerkiksi Red Dead Redemption 2 (2018) tunnetaan siitä, että se on onnistunut luomaan syvästi immersiivisen ja vaikuttavan maailman, joka heijastaa 1800-luvun Amerikan todellisia ja historiallisia koettelemuksia (Slyter 2025).

Tarinankerronta on tärkeä osa videopelejä, koska se tarjoaa kontekstin, merkityksen ja tunteen itse pelaamiselle (Solod 2024). Hyvä narratiivi yhdistää pelimekaniikat, haasteet, seuraukset, voitot ja häviöt keskenään ja antaa pelikokemukselle merkityksen. Hyvä tarinankerronta antaa siis pelaajalle syyn välittää suorittamistaan valinnoista ja tekemistään tehtävistä pelikokemuksen aikana (Solod 2024). Ilman vahvaa tarinankerrontaa pelaajan toiminta voi tuntua merkityksettömältä, jolloin myös pelikokemus ja immersio heikkenee.

Immersio tarkoittaa tässä kontekstissa fiktion kykyä viedä lukija mukanaan tarinan maailmaan (Brown 2023). Parhaimmillaan hyvä tarinankerronta tekee pelimaailmasta uskottavan, syvän ja elävän (Solod 2024). Yleisesti immersion luominen videopeliin tai mihinkään muuhun fiktioon ei ole yksinkertainen asia. Immersiivinen tarinankerronta yleensä sisältää selkeästi jäsennellyn juonen, samaistuttavia hahmoja, mukaansa tempaavan dialogin, emotionaalisesti vaikuttavia tapahtumia sekä uskottavan ja yhtenäisen maailman, joka tukee pelaajan uppoutumista tarinaan (Juegoadmin 2024). Videopelit tarjoavat myös muita välineitä immersioon. Yksi suosittu väline on **ei-lineaarinen tarinankerronta**, jossa tapahtumat eivät etene tiukasti aikajärjestyksessä, vaan pelaaja voi kokea ne eri järjestyksessä tai tehdä valintoja, jotka vaikuttavat juonen kulkuun, esimerkiksi valitsemalla toisen tehtävän toisen sijasta (Dey 2024).

Myös **ympäristöllinen tarinankerronta** tukee immersiota, koska sen kautta pelaaja voi ymmärtää narratiivia ilman suoraa kerrontaa. Esimerkiksi äänet,

visuaaliset vihjeet tai ympäristön muuttuminen vievät juonta eteenpäin ja voivat paljastaa asioita ympäröivästä maailmasta, sen historiasta ja hahmojen välisistä suhteista. (Dey 2024.) On olemassa myös pelejä, jotka nojaavat melkein pelkästään ympäristölliseen tarinankerrontaan, kuten Sons of the Forest (2024).

Dialogit ovat keskeinen elementti useimmissa tarinankerrontamuodoissa, mutta ne eroavat videopeleissä siten, että dialogit ovat usein interaktiivisia. **Interaktiivinen dialogi** tarkoittaa sitä, että pelaajalla on mahdollisuus tehdä valintoja dialogin kautta, eli hän valitsee mitä oma pelihahmo sanoo tai tekee. Tämä tuo tunteen seurauksista ja voi vaikuttaa kovallakin kädellä juonen kulkuun. (Dey 2024.)

Ei-lineaarinen ja ympäristöllinen tarinankerronta yhdistettynä interaktiiviseen dialogiin luo pelaajakeskeisen narratiivin, jossa tarina etenee pelaajan omien valintojen ja tekojen kautta. Tällaista tarinankerrontaa kutsutaan **emergentiksi tarinankerronnaksi** (Suter, Bauer & Kocher 2021, 71). Pelaaja rakentaa tarinansa ja kokemuksensa omien päätöstensä pohjalta, eikä ole sidottu ennalta määrättyyn juoneen. Tämä on muodostaa immersiivisen kokemuksen ytimen. Tällainen lähestymistapa korostaa pelaajan aktiivista roolia tarinan muokkaajana ja lisää sitoutumista pelimaailmaan.

Kuvassa 1 on kaavio, joka auttaa hahmottamaan videopeleissä esiintyviä tarinankerronnan muotoja. Kaavio auttaa myös hahmottamaan näiden tarinankerronnan muotojen välisiä suhteita ja eroja.

Tarinankerronnan keino	Kuvaus	Vaikutus immersioon	Lähde
Ei-lineaarinen tarinankerronta	Tapahtumat eivät etene aikajärjestyksessä, vaan muotoutuu pelaajan tekemien valintojen perusteella.	Luo tunteen vaikutusmahdollisuuksista ja sitouttaa tarinaan.	Dey 2024
Ympäristöllinen tarinankerronta	Narratiivi välittyy ympäristön, Äänten ja visuaalisten vihjeiden kautta ilman suoraa kerrontaa.	Perustuu kertomisen sijaan näyttämiseen ympäristön ja vihjeiden kautta.	Dey 2024
Interaktiivinen dialogi	Pelaaja valitsee, mitä hahmo sanoo tai tekee dialogissa.	Luo tunteen seurauksista ja vahvistaa pelaajan roolia tarinassa.	Dey 2024
Emergentti tarinankerronta	Pelaajan valinnoista ja teoista syntyvä, ei ennalta määrätty tarina.	Muodostaa immersiivisen kokemuksen ytimen ja lisää sitoutumista. Yhdistelee muita tarinankerronnan keinoja.	Suter, Bauer & Kocher 2021

Kuva 1. Tarinankerronnan keinot.

3 Dialogijärjestelmät ja pelaajan valinnat

3.1 Dialogijärjestelmien luokittelu ja esimerkit

Dialogi tarkoittaa kirjaimellisesti kaksinpuhelua tai vuoropuhelua. Se on yleinen tapa draaman, proosan tai lyriikan kerrontamuotona (Tieteen termipankki 2015). Dialogijärjestelmät videopelissä tarkoittavat kirjaimellisesti järjestelmää toteuttaa dialogi videopelissä.

Eri peleissä on kehitetty erilaisia toteutustapoja toteuttaa dialogia. Yksinkertaisimmillaan dialogijärjestelmä eri peleissä voi olla täysin **lineaarinen**, jolloin dialogissa ei ole mahdollisuutta tehdä minkäänlaista valintaa. Esimerkkejä tällaisista peleistä on vaikka God of War (2005) ja The Last of Us (2013). Tällaisia pelejä ei kuitenkaan ole mielekästä käsitellä tässä tutkimuksessa, koska puuttuvien valintojen pohjalta ei voi kerätä mitään tietoa.

Monissa peleissä on taas mahdollista valita mitä pelattava hahmo sanoo. Yleensä kyseiset valinnat tehdään systeemin kautta, jota kutsutaan **dialogipuuksi**. Dialogipuussa pelaajan ja muiden pelihahmojen välinen keskustelu etenee haarautuvien valintojen kautta. Pelaajan tekemä valinta johtaa uuteen puheenvuoron haaraan, joka voi avata erilaisia vastauksia, tapahtumia ja lopputuloksia. (Vitez 2025.) Käytännössä dialogipuu muodostuu listasta

tekstiprompteja (hahmon puheenvuoroja), josta pelaaja valitsee mieluisensa. Peleissä, joissa käytetään tätä mallia, dialogi on usein **vuoropohjaista**, eli keskustelu etenee yksi repliikki kerrallaan hahmojen vaihdellessa puheenvuoroja (Hughes 2022). Pelaajan ja NPC-hahmojen välinen kommunikointi tapahtuu siis vuorotellen, ja pelaaja saa aikaa pohtia valintojaan ennen seuraavaa reaktiota. Juuri tästä syystä vuoropohjainen dialogi on ihanteellinen vaihtoehto analytiikan kannalta.

Kyseinen dialogijärjestelmä on myös yleensä rauhallisempi ja sopii hyvin peleihin, joissa halutaan antaa pelaajalle mahdollisuus keskittyä tarinaan ja hahmojen välisiin suhteisiin. Ja jos peli sisältää paljon toimintaa, vuoropohjainen dialogi voi myös tarjota pelaajalle hengähdystauon**.** Lisäksi tällainen järjestelmä voi sisältää humoristisia tai yllättäviä vastausvaihtoehtoja, jotka lisäävät pelin viihdearvoa ja syventävät pelaajan immersiota pelimaailmaan. (Hughes 2022.)

Vuoropohjainen dialogijärjestelmä sopii erityisesti roolipeleihin. Esimerkkejä tällaisista peleistä ovat edellä mainitut *Fallout* 4 ja *The Witcher 3: Wild Hunt*, joissa vuoropohjainen dialogi muodostaa keskeisen osan pelin tarinankerrontaa ja päätöksenteon rakennetta.

Vaihtoehto täysin vuoropohjaiselle järjestelmälle on **dynaaminen dialogijärjestelmä**. Dynaamisessa dialogissa keskeistä on, että keskustelun eteneminen ja tarjottavat vaihtoehdot muovautuvat pelin senhetkisen tilan perusteella. Toisin sanoen pelaajan aiemmat valinnat, saavutukset ja vuorovaikutukset vaikuttavat siihen, mitä vuorosanoja pelihahmot käyttävät. (Siegel 2009.) Täysin dynaamisessa dialogijärjestelmässä ei ole mukana dialogipuuta, eikä pelaaja voi tehdä valintoja dialogissa. Pelaaja tekee kuitenkin valintoja pelin edetessä erilaisten tehtävien ja tilanteiden aikana, ei niinkään varsinaisten dialogivalikoiden kautta. Hyvä esimerkki tällaisesta pelistä on Red Dead Redemption 2, jossa pelaajan tekoja arvioidaan niin sanotun kunniajärjestelmän kautta. Tämä järjestelmä seuraa pelaajan moraalisia valintoja, esimerkiksi auttaako hän muita vai toimiiko väkivaltaisesti ja vaikuttaa siihen, miten muut hahmot suhtautuvat häneen sekä millaisia dialogeja ja loppuratkaisuja peli tarjoaa. Olennaista kuitenkin on se, että dynaaminen dialogijärjestelmä on väline luoda haarautuvia narratiiveja, vaikkei pelaaja voi välttämättä valita mitä pelihahmo sanoo (Hughes 2022). Tarinan haarautuvuus määräytyy siis pelaajan tekojen, eikä sanojen mukaan.

Dynaaminen dialogijärjestelmä ei ole välttämättä ihanteellinen analytiikan kannalta, jos pelaajan valintoja halutaan mitata itse dialogissa. Tällöin vaikka varsinaisten dialogivalintojen seuranta on haastavaa, pelaajan toimia voidaan silti mitata esimerkiksi tehtävien ja pelimaailman muiden tapahtumien yhteydessä sijoittamalla analytiikkalokittaja näihin kohtiin.

On kuitenkin yleistä, että etenkin roolipelit yhdistelevät vuoropohjaista dialogijärjestelmää ja dynaamista dialogijärjestelmää. Tämä tarkoittaa sitä, että pelissä esiintyy dialogipuu, jonka avulla pelaaja voi valita hahmonsa vuorosanat, mutta saatavalla olevat vuorosanat määräytyvät pelimaailman tilan mukaan, kuten dynaamisessa dialogijärjestelmässä. Usein näihin peleihin liittyy jonkinlainen kunniajärjestelmä. Esimerkkejä tällaisista peleistä ovat uudemmat Fallout-sarjan pelit ja Cyberpunk 2077 (2020).

3.2 Pelaajan valintojen merkitys

Valintojen tarkoitus videopeleissä on luoda pelaajalle vastuun ja vaikutusmahdollisuuden tunne, joka vahvistaa pelikokemuksen emotionaalista ulottuvuutta (Ulanoff 2024). Usein valinnoissa on mukana moraalinen ulottuvuus, jonka tarkoitus on syventää immersiota asettamalla pelaaja moraaliseksi toimijaksi pelimaailmassa (Duigou 2025). Pelaajalla on siis mahdollisuus olla oman pelinsä sankari tai roisto. Tämä aspekti on erityisen näkyvä aiemmin mainituissa Fallout-peleissä sekä pelissä Red Dead Redemption 2.

Edellä mainituissa peleissä valinnat voivat vaikuttaa pelaajan kokemukseen muokkaamalla tarinaa ja ympäröivien hahmojen suhtautumista pelaajaan. On kuitenkin yleistä, että videopeleissä tarinankerronta ei ole täysin emergenttiä, vaan pelaajan tekemät valinnat vaikuttavat lähinnä pelin lopputulokseen. Esimerkiksi The Witcher 3: Wild Huntissa pelaajan tekemät valinnat vaikuttavat pääasiassa pelin lopputulokseen. Erityisesti siihen onko loppuratkaisu positiivinen, negatiivinen vai neutraali. Toinen vastaava esimerkki on Dishonored (2012), jossa pelaajan aiheuttama kaaos vaikuttaa siihen onko loppu toiveikas vai synkkä.

Peleissä voi olla mukana myös **illuusio valinnasta** (Doigou 2025). Pelaajan eteen siis annetaan valinta, jolla ei ole tosiasiassa minkäänlaista vaikutusta. Dialogissa valinta saattaa hetkellisesti johtaa eri reitteihin, mutta lopulta keskustelu palaa aina samaan pisteeseen (Hagenlocher 2023). Tämän tarkoitus on lisätä immersiota ja tapa on yleinen etenkin lineaarisissa peleissä, kuten Deus Ex: Mankind Divided (2016) tai Assassin's Creed: Odyssey (2018).

Analytiikan kannalta sillä ei ole kuitenkaan minkäänlaista merkitystä, onko pelaajan valinta merkityksellinen itse tarinan haarautuvuuden kannalta. Kuitenkin, jos valinta ei vaikuta mitenkään merkitykselliseltä pelaajan kannalta, hän voi tehdä valinnan sattumanvaraisesti, jolloin analytiikalla ei ole enää suurta arvoa. Tämän vuoksi tunne valinnan merkityksellisyydestä on tärkeä.

Tietyt kuviot voivat myös paljastaa, kuinka merkityksellisiä eri vaihtoehdot ovat. Esimerkiksi jos suurin osa pelaajista valitsee aina saman vaihtoehdon, se voi viitata siihen, että muut vaihtoehdot koetaan vähemmän houkutteleviksi tai merkityksellisiksi.

4 Data-analytiikka pelikehityksessä

4.1 Analytiikan merkitys

Monet yritykset keräävät jatkuvasti paljon raakaa dataa, muttei siitä ole paljoa hyötyä, jos sitä ei osata tulkita. Esimerkiksi jos analytiikkalokittaja kerää raakadataa pelin dialogista, jossa annetaan kolme valintaa, tuloksista voi ilmetä, että 60 % pelaajista valitsee vaihtoehdon A, 35 % valitsee valinnan B ja vain 5 % valitsee vaihtoehdon C. Tässä vaiheessa nämä tulokset eivät vielä merkitse mitään, ne ovat vain tyhjiä prosenttilukuja. Data-analytiikaksi voidaan kutsua niitä prosesseja, jotka yrittävät tulkita tätä kerättyä dataa tehden siitä merkityksellistä (Stevens 2023).

On olemassa neljä erilaista pääasiallista tapaa tulkita dataa (Cote 2021). Tässä luvussa tarkastelemme, miten näitä kyseisiä tapoja voitaisiin hyödyntää pelikehityksen viitekehyksessä.

4.2 Deskriptiivinen analytiikka

Deskriptiivinen analytiikka tarkoittaa kirjaimellisesti kuvainnollista analytiikkaa. Tästä näkökulmasta tarkasteltuna vastataan erityisesti siihen ’mitä’ on tapahtunut tai mitä tällä hetkellä tapahtuu (Cote 2021). Jos edellä mainittua esimerkkiä tarkastelee deskriptiivisestä näkökulmasta voi huomata, että vaihtoehto A on suosituin ja vain harva valitsee vaihtoehdon C. Deskriptiivinen analytiikka ei kuitenkaan vastaa siihen, että miksi näin on.

4.3 Diagnostinen analytiikka

Kysymykseen miksi vaihtoehto A on ylivoimaisesti suosituin, vastataan diagnostisessa analytiikassa. Tässä analyytikko yrittää vertailla löydettyä havaintoa olemassa olevaan yleiseen tietoon, muuhun löydettyyn dataan tai trendeihin, jotta hän voi löytää syyseuraussuhteen (Cote 2021). Syy miksi vaihtoehto A on ylivoimaisesti suosituin videopelin dialogissa voi olla esimerkiksi se, että se johtaa pelin ’hyvään’ lopputulokseen, jonka pelaajat suuremmalla todennäköisyydellä haluavat nähdä.

4.4 Prediktiivinen analytiikka

Prediktiivinen analytiikka pyrkii ennustamaan tulevaisuutta aikaisemman datan ja trendien perusteella. Se vastaa siis kysymykseen ”mitä saattaa tapahtua tulevaisuudessa.” (Cote 2021). Käytännössä voitaisiin siis päätellä, että koska vaihtoehto A oli selvästi suosituin pelaajien valinta, se säilyy sellaisena myös jatkossa. Erityisesti silloin, jos sama kuvio toistuu myös muissa kyselyissä. Prediktiivinen analytiikka ei kuitenkaan kerro täysin varmasti mitä tulee tapahtumaan, vaan nojaa olettamukseen aikaisempien tulosten pohjalta.

4.5 Preskriptiivinen analytiikka

Tämä analytiikan muoto ottaa huomioon kaikki muuttujat sekä edellä mainitut analytiikan muodot ja sitten pyrkii vastaamaan kysymykseen ”mitä pitäisi tehdä?” (Cote 2021). Preskriptiivinen analytiikka on siis analytiikan kulminaatio ja pyrkii toimintaan edellisten havaintojen pohjalta. Lyhyesti voidaan hahmottaa, että deskriptiivinen analytiikka toteaa, että vain pieni osa valitsi vaihtoehdon C.

Diagnostinen analytiikka toteaa, että vain pieni osa valitsi vaihtoehdon C, koska se ei anna pelaajille toivottua lopputulosta. Pediktiivinen analytiikka vastaa, että näin tapahtuu myös jatkossa, jos kehitetään samanlainen vaihtoehto seuraavissa peleissä ja näiden havaintojen pohjalta preskriptiivinen analytiikka ehdottaa, että olisiko parempi tulevaisuudessa tehdä sellainen vaihtoehto, joka olisi yhtä houkutteleva kuin vaihtoehto A, jotta pelikokemuksesta saadaan immersiivisempi.

4.6 Pelaaja käyttäytymisen analytiikka dialogissa ja sen hyödyt

Analytiikka tarjoaa tärkeitä työkaluja, joiden avulla voi paljastaa tietoa pelaajan mieltymyksistä ja toiminnasta. Näin kehittäjät voivat ymmärtää asiakkaitaan. Pelaajilta kerätty data voi paljastaa suosittuja piirteitä videopeleistä tai niitä piirteitä, jotka kaipaavat korjausta. (Mizina 2025). Ennen data-analytiikkaa kehittäjät rakensivat pelit kokemuksen, intuition ja oman luovuutensa pohjalta. Analytiikka tuo tähän mukaan **konkreettiset todisteet** pelaajan mieltymyksistä. (Argentics 2023).

Etenkin dialogissa voidaan huomata analytiikan avulla, mitkä valinnat ovat suosittuja ja mitkä ovat vähemmän suosittuja. Myös aika, jonka pelaaja käyttää dialogissa, voi toimia mittarina sen kiinnostavuudelle. Jos pelaaja jatkuvasti ohittaa dialogin ja valitsee aina ensimmäisen vaihtoehdon, voidaan päätellä, ettei tarina ole immersiivinen. (Argentics 2023).

Kun pelaajan mieltymyksiä on kartoitettu ja niiden taustalla olevat tekijät ymmärretty, voidaan näitä mieltymyksiä alkaa painottaa ja hyödyntää preskriptiivisen analytiikan periaatteiden mukaisesti. Lisäksi vähemmän suosittuja ratkaisuja peliyhtiö voi korjata päivityksissä tai jättää samankaltaiset ratkaisut pois tulevaisuuden kehityksessä. Jos kerättyä dataa ymmärretään oikein diagnostisen analytiikan hengessä, voidaan ennustaa pelaajakäyttäytymistä ja rakentaa sen pohjalta merkityksellisempää tarinankerrontaa.

Laadukas tarinankerronta koetaan yleisesti tärkeäksi pelaajien kesken, ja se lisää pelaajien sitoutumista ja pelissä vietettyä aikaa. Tutkimusten mukaan pelit, joissa on hyvin kirjoitetut hahmot ja hyvin rakennettu narratiivi myyvät jopa 50% enemmän kuin pelit, joissa ei ole näitä piirteitä. Pelaajat myös ehdottavat peliä

todennäköisemmin muille, jos siinä on hyvin rakennettu tarina. (Andersen 2024).

5 Eri tavat toteuttaa analytiikkalokittaja videopelissä

5.1 Analytiikkalokittajan toteutusstrategiat

Kuten aiemmin on määritelty, alytiikkalokittaja on työkalu, joka kerää ja tallentaa tietoa käyttäjien toimista (Google Gemini 3 2026a). Tässä kontekstissa datan tarkoitus on ymmärtää pelaajan käyttäytymistä ja löytää sitä kautta pelaajan mieltymyksiä. Tässä tilanteessa analytiikkalokittajan avulla analysoidaan pelaajien valintoja dialogeissa. Kuten edellä on mainittu, pelaajan mieltymyksiä arvioimalla voidaan selvittää suosittuja tai vähemmän suosittuja elementtejä pelistä. Valinnan vaikeustasoa on myös mahdollista arvioida mittaamalla siihen käytettyä aikaa, mutta myös koko pelin vaikeustasoa voi mitata dialogista, jos pelaajalla kestää kauan siirtyä seuraavaan kohtaan, jossa toinen dialogi esiintyy. Dataa voi kerätä myös pelin käyttöliittymän toimivuudesta.

Analytiikkalokittajan varsinaiseen toteutukseen on monia eri tapoja. Usein nämä toteutukset voidaan jakaa omaan räätälöityyn ratkaisuun, jossa itse kehitetään tarpeenmukainen analytiikkatyökalu tai sitten yritys valitsee käyttävän kolmannen osapuolen analytiikkatyökaluja. Myös hybridivaihtoehdot ovat mahdollisia, tässä tilanteessa esiintyy elementtejä molemmista vaihtoehdoista.

Yleisesti parhaan mahdollisen tavan päätökseen vaikuttavat yrityksen käytettävissä olevat resurssit, kuten kehitysaika, tekninen osaaminen sekä taloudelliset resurssit. Jos puhutaan kolmannen osapuolen anaalytiikkatyökaluista, valinta voi riippua enemmän peliyrityksen käyttämästä teknologiasta, etenkin kehitystyössä käytetystä pelimoottorista tai vastaavasta kehitysalustasta (Çoğalan 2023).

Seuraavissa alaluvuissa tarkastellaan tarkemmin eri omaan toteutukseen perustuvia analytiikkaratkaisuja sekä kolmannen osapuolen analytiikkatyökaluja ja niiden hyötyjä ja haittoja.

5.2 Omaan toteutukseen liittyvät edut ja haitat

Mikäli yritys valitsee kehittää analytiikkaratkaisunsa itse, tällöin vastuu sovelluksen suunnittelusta, kehityksestä ja ylläpidosta on kokonaan yrityksellä tai sopimuksesta riippuen, sillä kehitystiimillä, joka sovellusta kehittää.

Tässä tilanteessa suurin etu on se, että yritys saa täysin räätälöidyn ratkaisun, joka vastaa tarkasti sen liiketoiminnallisia ja teknisiä tarpeita. Erityisesti räätälöityyn ratkaisuun voidaan sisällyttää jo olemassa olevat pelimoottorit, tietokannat ja muut järjestelmät. Lisäksi tällainen ratkaisu tarjoaa joustavuutta yrityksen kehittyviin tarpeisiin. Kyseessä on myös suhteellisen varma ja tietoturvallinen ratkaisu, koska on aina olemassa riski, että kolmannen osapuolen palvelut voivat kaatua tai loppua, jolloin kerätty data menetetään. Kolmannen osapuolen analytiikkapalveluiden käyttöön liittyy myös kohonnut tietoturvariski, sillä peliyrityksen keräämää dataa siirretään ja käsitellään ulkopuolisen palveluntarjoajan järjestelmissä. (Yatin 2025.)

Kuten edellä on viitattu, on myös mahdollista, että peliyrityksen oma tiimi ei osallistu välttämättä lokittajan kehitykseen, vaan yritys ostaa kehityspalveluita toiselta yritykseltä, jolloin mukana on kolmannen osapuolen elementti. Kuitenkin tässä tilanteessa omistusoikeus ja päätösvalta säilyvät peliyrityksellä, jos näin on toimeksiantosopimuksessa sovittu. (Kallio 2026.)

Suurin haitta puolestaan tässä tilanteessa on kustannukset. Hinnoittelumalli voi olla erilainen riippuen yrityksestä tai sopimuksesta (Kallio 2026). Kuitenkin keskimääräinen hintahaarukka pienen sovelluksen kehittämiselle on 5000$ - 50,000$. Tuo kuitenkin kattaa vain kehittämisen kulut. Ylläpitämisen kulut ovat suurella todennäköisyydellä noin 15%-20% kehittämisen kuluista. Tähän sisältyy esimerkiksi bugien korjaus, tietoturvan uudistukset, suorituskyvyn monitorointi ja servereiden ylläpito. (Dugdiev 2025.)

Peliyrityksen on mahdollista myös tuottaa tarvittava analytiikkalokittaja itse. Tämä kuitenkin sitoo yrityksen omaa työvoimaa sovelluksen kehittämiseen ja kehityksen hinta määräytyy tällöin työntekijöiden palkan mukaan. Voi olla tilanteita, joissa hinta voi koitua kalliimmaksi, jos sovellus kehitetään talon sisällä, koska keskimääräinen sovelluskehittäjän palkka pelialalla on noin 4000 euroa suomessa (Ronkainen 2025). Tämä voi tapahtua erityisesti silloin, jos lähetään

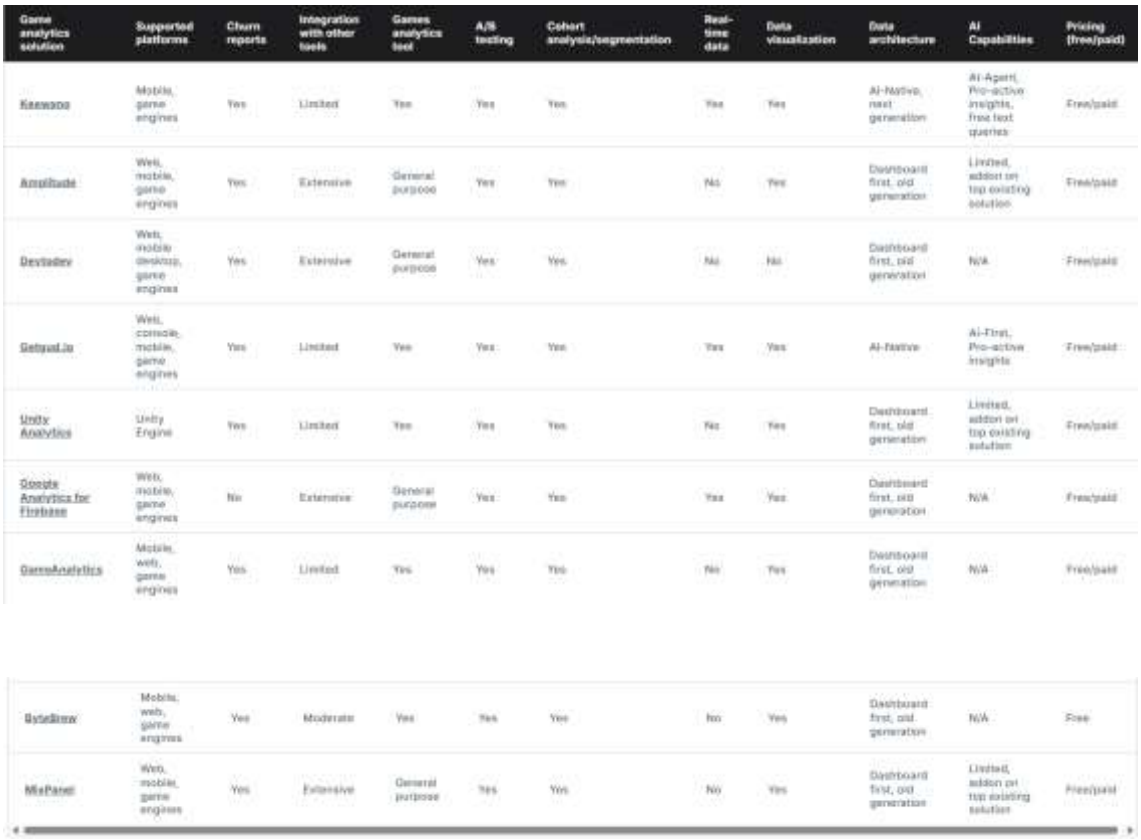
kehittämään hyvin monimutkaista analytiikkalokittajaa, jossa kestää kauan. Analytiikkalokittajat ovat kuitenkin yleensä hyvin yksinkertaisia sovelluksia, joten tämän ei pitäisi olla yleinen ongelma, vaan enemmänkin mahdollisuus, joka voi tapahtua silloin, jos asioita ei mietitä loppuun asti. Kehittäminen on myös pois tavanomaisesta pelikehityksestä ja projektit voivat hidastua, jos samat kehittäjät osallistuvat tavanomaiseen peliohjelmointiin.

Peliohjelmointiin keskittyvillä kehittäjillä ei myöskään ole välttämättä kehittynyttä erityisosaamista koskien analytiikkaa ja siihen tarvittavia työkaluja (Yatin 2026). Tämän vuoksi analytiikkatyökalun kehittäminen voisi olla järkevää jättää ammattilaisille. Tämä kuitenkaan ei koske kaikkia tilanteita, koska kaikki peliyhtiöt ovat kuitenkin erilaisia ja on mahdollista, että jollain kehittäjällä on syvä ymmärrys analytiikasta.

Peliyhtiö voi myös joutua sijoittamaan oman voi datainfrastruktuurin ostoon, ellei sitä jo ole hankittu. Pitäähän data myös tallentaa johonkin tietokantaan tarkastelua varten. Tällöin mukaan tulee myös oman infrastruktuurin ylläpitoon liittyvät kustannukset. Yrityksen on itse arvioitava tilanteen mukaan, onko järkevämpää tuottaa sovellus itse vai ostaa tuotettava palvelu muilta.

5.3 Kolmannen osapuolen analytiikkatyökalut

Kolmannen osapuolen analytiikkatyökaluilla tarkoitetaan valmiita työkaluja, jotka ovat jonkun ulkopuolisen yrityksen kehittämä ja omistama. Tässä mallissa peliyritys ei kehitä analytiikkajärjestelmää itse, vaan hyödyntää kolmannen osapuolen tarjoamaa palvelua lisenssi- tai palvelusopimuksen kautta, jolloin itse työkalu ja sen infrastruktuuri pysyvät palveluntarjoajan hallinnassa. Valmiita analytiikkatyökaluja pelikehitykseen on paljon, esimerkiksi Keewano, Google Analytics ja GameAnalytics. Nämä eroavat toisistaan ominaisuuksillaan, esimerkiksi Keewano tarjoaa mahdollisuuden, kerätä reaaliajassa dataa, mutta GameAnalytics ei (Plotnek 2024). Kuvassa 2 esitetään vertailu kahdeksasta kolmannen osapuolen analytiikkatyökalusta ja niiden ominaisuuksista.



Game analytics solution	Supported platforms	Churn reports	Integration with other tools	Games analytics tool	A/B testing	Cohort analysis/segmentation	Real-time data	Data visualization	Data architecture	AI Capabilities	Pricing (free/paid)
Keewano	Mobile, game engines	Yes	Limited	Yes	Yes	Yes	Yes	Yes	AI-Native, next generation	AI-Agent, Pro-active insights, free text queries	Free/paid
Amplitude	Web, mobile, game engines	Yes	Extensive	General purpose	Yes	Yes	No	Yes	Dashboard first, old generation	Limited, addon on top existing solution	Free/paid
Devtodev	Web, mobile desktop, game engines	Yes	Extensive	General purpose	Yes	Yes	No	No	Dashboard first, old generation	N/A	Free/paid
Gehyad.io	Web, console, mobile, game engines	Yes	Limited	Yes	Yes	Yes	Yes	Yes	AI-Native	AI-First, Pro-active insights	Free/paid
Unity Analytics	Unity Engine	Yes	Limited	Yes	Yes	Yes	No	Yes	Dashboard first, old generation	Limited, addon on top existing solution	Free/paid
Google Analytics for Firebase	Web, mobile, game engines	No	Extensive	General purpose	Yes	Yes	Yes	Yes	Dashboard first, old generation	N/A	Free/paid
GameAnalytics	Mobile, web, game engines	Yes	Limited	Yes	Yes	Yes	No	Yes	Dashboard first, old generation	N/A	Free/paid
ByteBrew	Mobile, web, game engines	Yes	Moderate	Yes	Yes	Yes	No	Yes	Dashboard first, old generation	N/A	Free
MixPanel	Web, mobile, game engines	Yes	Extensive	General purpose	Yes	Yes	No	Yes	Dashboard first, old generation	Limited, addon on top existing solution	Free/paid

Kuva 2. The 8 Best Game Analytics Solutions in 2025 (Plotnek 2024).

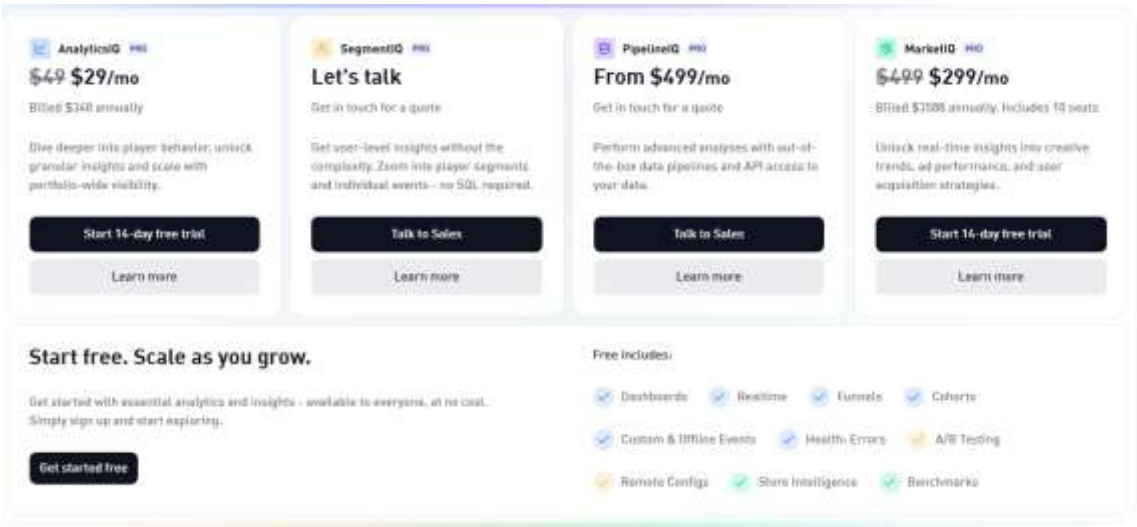
Eri pelikehitys infrastruktuureille, kuten pelimoottoreille on myös olemassa omia analytiikkatyökaluja, esimerkiksi Unitylle on olemassa Unity Analytics. Hinnat voivat nousta tuhansiin euroihin omassa toteutuksessa, mutta Unity Analytics on ilmainen siihen asti, kunnes palvelussa on 50 000 aktiivista käyttäjää. Tämän jälkeen palvelua on jatkettava maksullisella tilauksella (Unity Documentation 2026). Tämän jälkeen Unity Analyticsin hinnoittelu perustuu käyttäjämääriin, ja palvelusta veloitetaan kuukausittain aktiivisten käyttäjien määrän perusteella. Kuvassa 3 on kuvattu paljon Unity Analytics maksaa käyttäjämäärien perusteella.

Number of users	Cost per MAU
0 - 50,000	$0.00 (free tier)
50,001 - 150,000	$0.00360
150,001 - 500,000	$0.00315
500,001 - 1,000,000	$0.00293
1,000,001 - 5,000,000	$0.00225
5,000,001 and higher	Contact Unity Support to learn about more pricing options.

Kuva 3. Pricing (Unity Documentation 2026).

Tämä on vieläkin hyvin halpaa ainakin verrattuna siihen, jos oma toteutus teetetään ulkopuolisen yrityksen kautta. Esimerkiksi noin 80 000 kuukausittaisen aktiivisen käyttäjän pelissä kustannus lasketaan seuraavan mallin mukaan: 80 000 – 50 000 = 30 000 maksullista kuukausittaista käyttäjää. 30 000 * $0.00360 = $108 / kk. Ylläpitokustannukset halvimman mahdollisen analytiikkalokittajan teettämiseen voivat teoriassa nousta jopa 750$ kuussa, koska 5000 * 0.15 = 750. Tämä kuitenkin riippuu sopimuksesta, eikä ylläpitokustannuksia voi siksi pitää kiveen hakattuina totuuksina. Omassa toteutuksessa voi kuitenkin olla tilanteita, ettei sopimukseen edes sisälly minkäänlainen ylläpitäminen ja ostaja yritys hoitaa sen itse. Hinta ylläpidolle voi kuitenkin kiinteämpi itse teetetyssä sovelluksessa kuin esimerkiksi Unity Analyticsin käytössä, jossa se määräytyy käyttäjämäärän mukaan.

Hinnoittelu eroaa kuitenkin palvelua tarjoavan yrityksen mukaan, esimerkiksi GameAnalytics tarjoaa monia eri tilaussuunnitelmia, jotka eroavat toisistaan käytettävissä olevien toimintojen mukaan. Hinta voi olla kiinteä maksu kuussa, joka voi olla viidestäkymmenestä dollarista viiteen sataan dollariin. Esimerkiksi vain pelianalytiikkaan keskittyvä AnalyticsIQ-suunnitelma maksaa noin viisikymmentä dollaria kuussa, mutta jos haluaa keskittyä pelien markkinointiin, on valittavat eri suunnitelma. Tässä tapauksessa MarketIQ-suunnitelman kustannukset voivat nousta viiteen sataan dollariin. (GameAnalytics 2026.) Edellä mainitussa tilanteessa kuukausittaiset kulut voivat olla jo huomattavia. Kuvassa 4 on näytetty eri maksusuunnitelmat GameAnalytics-palvelulle.



Kuva 4. Pricing (GameAnalytics 2026.)

Jotkut kolmannen osapuolen analytiikkapalvelut tarjoavat myös mahdollisuutta käyttää analytiikkatyökalun lisäksi myös asiantuntijapalveluita, kuten data-analyytikon tukea tai konsultointia (Yatin 2025). Tällaisia pelianalytiikka palveluita tarjoaa esimerkiksi ediiie (ediiie 2026).

On varmasti olemassa tilanteita, jossa kolmannen osapuolen analytiikkatyökalun käyttäminen sopii yrityksen tarpeisiin täydellisesti, mutta suurin ongelma niiden käytössä on räätälöidyn kokemuksen puuttuminen. Tästä seuraa se, että yrityksen mahdollisuudet vaikuttaa datan keruuseen tai sen käsittelyyn voi olla rajalliset. (Yatin 2026.) Räätälöitymättömän ratkaisun ongelma on se, että erilaisia toimintoja on joko liikaa tai liian vähän. Jos erilaisia toimintoja on liikaa ja suuri osa niistä on tarpeettomia asiakasyritykselle, voi palvelun käyttäminen olla liian monimutkaista ja hankalaa. Jos analytiikkajärjestelmän toiminnallisuudet ovat rajalliset, voidaan joutua tinkimään siitä, mitä pelin osa-alueita tai tapahtumia analytiikassa seurataan. Pahimmassa tilanteessa käyttö on liian monimutkaista ja silti joudutaan tinkiä analytiikasta. Tästä syystä toteutuksena oman ratkaisun luominen on usein parempi.

Kolmannen osapuolen analytiikkapalveluiden käyttöön liittyy myös tietoturvaan liittyviä haasteita, koska kerättyä dataa käsitellään ja säilytetään ulkopuolisen palveluntarjoajan järjestelmissä, mikä voi lisätä tietoturvariskien mahdollisuutta. Tämän vuoksi tiukat sopimukset ja käytännöt liittyen tietoturvaan on tärkeitä. (Yatin 2026.)

On myös mahdollista, että yritys, jolta analytiikkapalveluita ostetaan, menee konkurssiin, jolloin kerätty data voidaan menettää ikuisiksi ajoiksi tai kyseinen analytiikkapalvelu poistuu myynnistä. Unity Technologies on yrityksenä erityisen tunnettu siitä, että sillä on taipumus poistaa omia palveluitaan. On myös tavanomaista, että käytettyihin palveluihin tulee ongelmia esimerkiksi päivityksien myötä.

Kolmannen osapuolen analytiikkapalveluihin liittyvät rajoitteet huomioon ottaen voidaan ymmärtää, että tällainen ratkaisu voi soveltua parhaiten lyhyen aikavälin tarpeisiin. Kolmannen osapuolen analytiikkatyökalut ovat tyypillisesti nopeampia ottaa käyttöön, sillä ne eivät edellytä omaa kehitystyötä, mikä tekee niistä houkuttelevan vaihtoehdon erityisesti projektin alkuvaiheessa. Ne sopivat myös pidemmällä aikavälille, jos yritys välttämättä tarvitsee ammattimaista konsultointia.

5.4 Hybridivaihtoehdot

Hybridivaihtoehdot ovat sekoitus omaa toteutusta ja kolmannen osapuolen palveluiden käyttämistä. Tämä yleensä voi näyttää esimerkiksi siltä, että itse analytiikkatyökalu toteutetaan itse, mutta tiedot tallennetaan johonkin pilvipalvelun tietokantaan, kuten johonkin Azuren- tai AWS:n tietokantaan. Datan keräys työkalu tehdään siis itse, mutta kyseistä dataa voidaan myös käsitellä ja mallintaa, jollain kolmannen osapuolen sovelluksella, esimerkiksi Azure Data Explorerilla tai Power BI:llä.

Hybridivaihtoehto voi tarjota tasapainoa ratkaisun joustavuuden ja kustannusten välillä. Yritys voi itse kustomoida analytiikkalokittajansa tarpeidensa mukaan, mutta samaan aikaan se voi hyödyntää valmiiden pilvipalvelujen analytiikkatyökalujen skaalautuvuutta, suorituskykyä ja valmiita mahdollisuuksia.

Tällä tavoin on mahdollista luoda yksinkertainen oma toteutus, jota voidaan laajentaa monipuolisemmaksi toteutukseksi ajan myötä. Tässä tilanteessa on kuitenkin samoja tietoturvariskien elementtejä kuin täysin kolmannen osapuolen ratkaisun käytössä, mutta pilvipalveluja pidetään yleisesti arvostettuina ja luotettavina kumppaneina, jotka ovat erikoistuneet alaansa. Pitää kuitenkin muistaa, ettei omassa tietovarastossa datan säilyttäminen ole myöskään täysin

riskitöntä, koska se edellyttää riittäviä tietoturvakäytäntöjä esimerkiksi varmuuskopioinnin ja mahdollisten tietomurtojen varalta. Puutteet näissä voivat altistaa järjestelmän tietoturvaloukkauksille aivan kuin ulkoisten palveluidenkin käytössä.

5.5 Lopputulos

Tässä opinnäytetyössä toteutetaan analytiikkalokittaja itse, koska toimeksiantaja yritys (Olipa Games) haluaa itselleen yksinkertaisen, mutta laadukkaan ratkaisun. Tämä siksi, koska yksinkertaista ja itse toteutettua ratkaisua on helpompi käyttää kuin täysin kolmannen osapuolen omistamaa anaalytiikkalokittajaa ja sitä on myös helpompi kustomoida tuleviin tarpeisiin. Toimeksiantaja yrityksellä on myös oma tietovarasto, johon kerätyt tiedot halutaan tallentaa. Oman toteutuksen voi rakentaa täysin vastaamaan kyseisen infrastruktuurin tarpeita.

Tässä ratkaisussa analytiikkalokittaja hyödyntää kuitenkin joitain kolmannen osapuolen sovelluksia, koska data on tarkoitus saada CSV-tiedostoiksi ja sitä on tarkoitus mallintaa Power BI:llä. Power BI on Microsoftin liiketoiminta analytiikkaan perustuva visualisointityökalu, jonka avulla raa’an datan voi muuttaa käytännöllisiksi oivalluksiksi (Microsoft Learn 2025). Tässä toteutuksessa myös hyödynnetään ulkoiselta palveluntarjoajalta hankittua MySQL-tietokantaa.

6 Analytiikkalokittajan tekninen arkkitehtuuri ja komponentit

Tarkoituksena on määritellä toteutettavalle analytiikkalokittajalle keskeiset ominaisuudet ja tekniset vaatimukset. Tarkastelu keskittyy erityisesti siihen, miten analytiikkalokittaja toteutetaan käytännössä, mutta myös tarkastellaan, mitkä ovat niitä alustoja ja työkaluja, joiden puitteissa toteutus tehdään. Tarkoitus on muodostaa mahdollisimman tarkka käsitys sovelluksen toteutuksen edellytyksistä.

6.1 Pelimoottori

Pelimoottori tarkoittaa sovelluskehitysalustaa joka on pääsääntöisesti optimoitu videopelien kehitykseen. Pelimoottorit tarjoavat tyypillisesti tuen eri ohjelmointikielillä sekä sisältävät 2D- ja 3D renderöintimoottorin grafiikan käsittelyyn. Näihin on usein sisällytetty myös fysiikkamoottori, jonka avulla voidaan simuloida reaalimaailman fysikaalisia ilmiöitä. (Arm 2025.) Analytiikkalokittaja rakennetaan osaksi pelimoottoria pelimoottorin sisäisenä skriptipohjaisena komponenttina.

Pelimoottori, jota tässä toteutuksessa käytetään, on Unity Engine. Unity engine on ollut ainakin vuosina 2023 ja 2024 maailman käytetyin pelimoottori (GameFromScratch 2024). Unity on valittu toteutukseen siitä syystä, että se on toimeksiantaja yrityksen yleisesti käyttämä pelimoottori pelikehityksessä.

6.2 Ohjelmointikielet

Ohjelmointikieli on joukko ohjeita, jota ohjelmoijat käyttävät tietokoneiden kanssa tehtävien suorittamiseen ja tietokoneiden kanssa kommunikoimiseen. Yleensä ohjelmointikielillä luodaan erilaisia sovelluksia, kuten verkkosivuja ja työpöytäsovelluksia. Eri ohjelmointikieliä on olemassa esimerkiksi C#, Python, Java ja C++. (Tieturi 2026.) Analytiikkalokittajan toteutukseen on valittu **C#**, koska se on Unity Enginen pääsääntöisesti käyttämä ohjelmointikieli.

Jotta pelimoottorista saadaan turvallinen yhteys tietokantaan, tarvitaan kyseistä yhteyttä käsittelevä palvelinpuoli. Tässä palvelinpuolella käytetään **PHP**-ohjelmointikieltä, koska PHP ei tarvitse erillistä tukea selaimelta, kuten erillistä palvelinasennusta ja sillä voi käsitellä helposti palvelinpuolen tiedostoja (Laaksonen 2011). Tietokantana käytetään MySql-tietokantaa, joten kyselykielenä käytetään luonnollisesti **SQL**-kieltä.

6.3 Ohjelmistoympäristö

Ohjelmistoympäristöllä tarkoitetaan ohjelmistokehityksessä käytettävää kehitysalustaa, joka tarjoaa tarvittavat työkalut ohjelmistokehitykseen, kuten

koodieditorin, kääntäjän sekä muita apuvälineitä kehitykseen Google Gemini 2026). Tässä toteutuksessa käytetään Microsoft Visual Studio 20219, koska se on yleisesti käytetty ja vakiintunut ohjelmistoympäristö Unityn kanssa.

Palvelinpuolen ohjelmointiin käytetään Visual Studio Codea, koska sen yksinkertaisuus ja kevyt rakenne sopivat hyvin palvelinpuolen kehitykseen.

6.4 Peli

Analytiikkalokittaja on erillinen scriptikomponentti, joka toteutetaan ja ajetaan osana pelin järjestelmää.

Toteutuksena käytettävä peli on nimeltään Hostile Docking. Hostile Docking on Olipa Gamesin tuleva peli, joka sijoittuu neonvalaistuun cyberpunk-tyyliseen kaupunkiin. Pelaaja johtaa taktista ryhmää tutkimaan hiljaiseksi jäänyttä rahtialusta. Aluksi yksinkertaiselta vaikuttava pelastustehtävä muuttuu nopeasti selviytymistaisteluksi, kun paljastuu, että vaikutusvaltainen yhtiö on kaapannut aluksen varastaakseen laittomia neuroaseita. Pelaajan valinnat vaikuttavat tarinan kulkuun ja lopputulokseen, määrittäen ketkä selviytyvät, keihin voi luottaa, tuleeko totuus julki vai jääkö tapahtumat ikuisesti varjoihin. (Olipa Games 2025.)

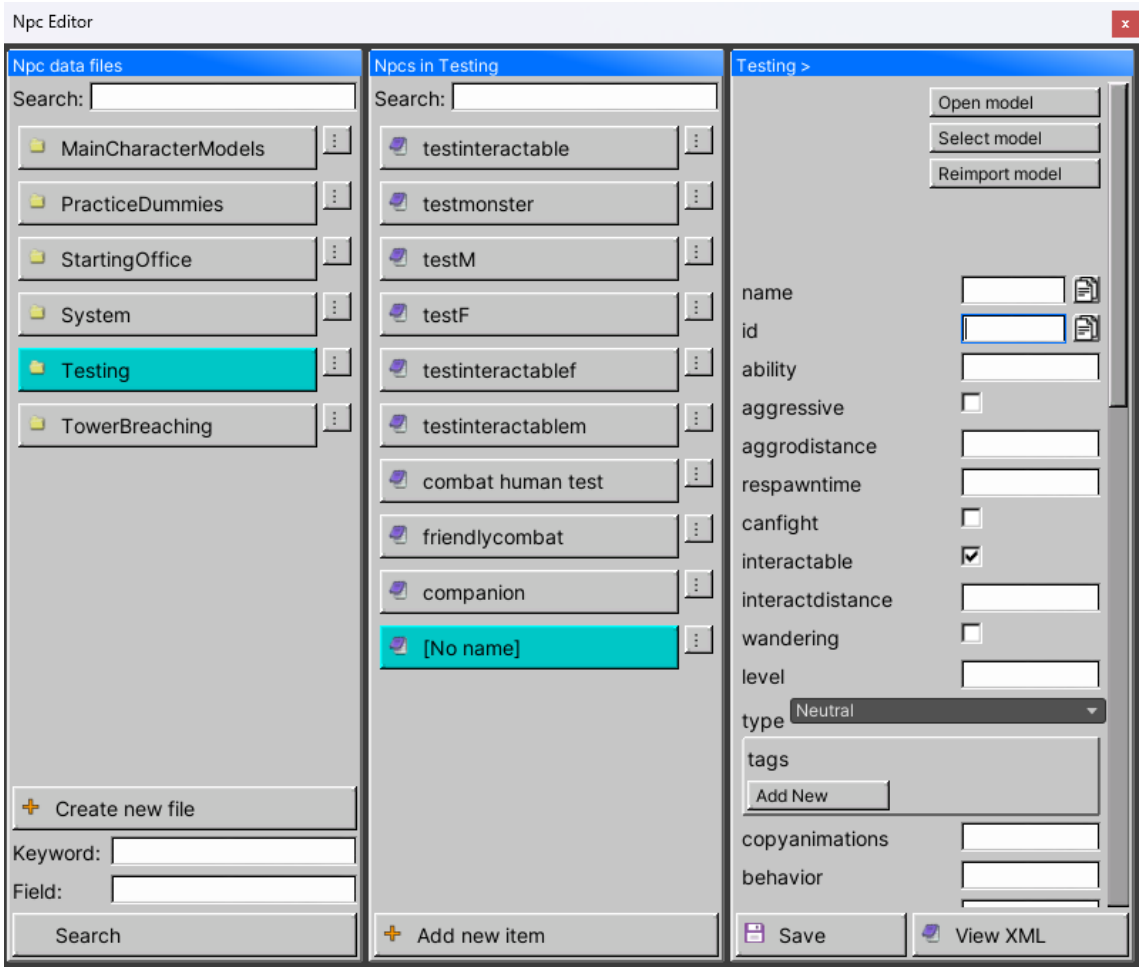
Peli on ytimeltään yksinpelattava taktinen ammuntapeli, jossa pelaaja etenee tehtävissä tulitaisteluiden ja hiiviskelyn avulla. Pelaajan kehittyminen perustuu alueita tutkimalla löydettyihin tai taisteluista saatuihin varusteisiin, kuten aseisiin ja suojuksiin. Pelissä ei ole siis perinteistä kokemukseen perustuvaa järjestelmää. Kuten aiemmin on mainittu, pelin keskiössä on haarautuva tarina, jossa pelaajan valinnat vaikuttavat merkittävästi tarinan kulkuun ja pelin lopulliseen päätepisteeseen. (Olipa Games 2025.) Peliä voisi luonnehtia emergentiksi tarinankerronnaksi, jossa pelaaja rakentaa tarinansa ja kokemuksensa omien päätöstensä pohjalta, eikä ole sidottu ennalta määrättyyn juoneen. Lisäksi siinä on vuoropohjainen dialogipuu. Tämän vuoksi peli sopii täydellisesti tähän opinnäytetyöhön. Kuvassa 5 esitetään kuvankaappaus kyseisestä pelistä, jotta lukija hahmottaisi paremmin millaisesta pelistä on kyse.

Kuva 5. Hostile Docking (Olipa Games 2025.)

6.5 Hahmonluontityökalu

Ennen dialogin luomista, jonka kautta pelaaja tekee valinnan ja tämän kautta pelaajan valintoja analysoidaan, luodaan ei-pelattava hahmo, jonka kanssa keskustelu käydään.

Hahmon luomiseen on käyttetty kahta eri Olipa Gamesin kehittämää hahmonluontityökalua, Npc editoria ja NPCMakerLiteGreeniä. Npc editorissa luodaan kansio, johon luotava hahmotiedosto tallennetaan. Siinä myös päätetään luotavan hahmon ominaisuudet, kuten se, onko hahmo aggressiivinen vai ei, mikä hahmon nimi on ja voiko hahmon kanssa vuorovaikuttaa ylipäätänsä. Lisäksi tässä määritellään jokaiselle luotavalle ei-pelattavalle hahmolle oma id-luokittelu, jonka avulla hahmoon voi viitata ja sitä kautta sille voi asettaa erilaisia tapahtumia ja määrätä esimerkiksi, missä tilanteissa hahmolle asetettu dialogi lähtee käyntiin. Kuvassa 6 esitetään, miltä näyttää Npc Editorin käyttöliittymä.



Kuva 6. Npc editor.

Hahmon ulkonäkö ja aseet päätetään NPCMakerissa. Näillä asioilla ei kuitenkaan ole väliä analytiikkalokittajan toteutuksen kannalta. Kuvassa 7 esitetään NPCMakerLiteGreenin käyttöliittymä.

Kuva 7. NPCMakerLiteGreen.

6.6 Dialogin rakennustyökalu

Dialogi toteutetaan Olipa Gamesin itse kehittämällä solmupohjaisella dialogin rakennustyökalulla, DialogueMaster toolilla. DialogueMaster tool toimii Unityssa. Pääpiirteittäin dialogi rakennetaan työkalussa niin, että ensin luodaan dialogi-datatiedosto ja sen sisälle keskeinen yksittäinen dialogi. Keskusteluun määritellään siihen osallistuvat hahmot ja sitten varsinainen dialogi muodostetaan lisäämällä solmuja, jotka sisältävät hahmojen mahdolliset repliikit ja vastausvaihtoehdot. (OlipaOS Documentation 2025.)

Dialogin etenee solmujen välisten yhteyksien mukaan ja yksi solmu merkitään keskustelun aloituskohdaksi. Vastausvaihtoehtoihin voi liittää ehtoja (checks) ja toimintoja (actions). Eri haarautuvuudet toteutuvat ehtojen pohjalta ja actions tekee, jonkun suoritettavan toiminnon, esimerkiksi antaa toiselle hahmolle tietyn määrän rahaa. Dialogeihin voi myös liittää animaatioita, jotka voi näyttää keskustelun yhteydessä. (OlipaOS Documentation 2025.) Kuvassa 8 esitetään, miltä näyttää DialogueMasterin käyttöliittymä.

Kuva 8. DialogueMaster. (OlipaOS Documentation 2025.)

6.7 Event Editor

Event editor on Olipa Gamesin kehittämä tapahtumanhallintatyökalu, joka käsittelee vuorovaikutuksia pelaajan, ei-pelattavien hahmojen ja objektien välillä (OlipaOS Documentation 2025). Tällä työkalulla voi liittää rakennettuja dialogeja hahmoihin, joille ne kuuluvat ja sitä kautta dialogi lähtee toimimaan.

6.8 Tietokanta

Tietokanta tarkoittaa tietojen keräämiseen ja järjestämiseen tarvittavaa työkalua (Microsoft 2026). Olipa Games käyttää MySQL tietokantaa, joka on hankittu Cpanel InMotion Hosting - palvelutarjoajan kautta. Tähän työhön on valittu kyseinen tietokanta vain siksi, koska toimeksiantajayritys maksaa siitä jo. Olisi kuitenkin teoriassa mahdollista käyttää muitakin tietokantoja. Tiedot lähetetään tietokantaan välittäjäohjelman kautta, joka on kirjoitettu PHP-kielellä.

Tietokannan käyttöjärjestelmänä toimii PhpMyAdmin, joka on integroitu Cpanelin palvelupaketin mukaan. PhpMyAdminilla voi muokata ja ylläpitää olemassa olevia tietokantoja mielekkäästi.

Toinen mahdollinen toteutustapa olisi tallentaa kerätty data Olipa Gamesin omalle palvelimelle rajapintakutsujen avulla JSON-muotoisena. Tämä ei kuitenkaan olisi kestävä ratkaisu, sillä se ei sallisi skaalautuvuutta. Siitä syystä, että JSON-tiedoston muistikapasiteetti on niin paljon pienempi kuin tietokannan. Suuri tietomäärä voisi teoriassa tukkia Olipa Gamesin oman palvelimen kokonaan.

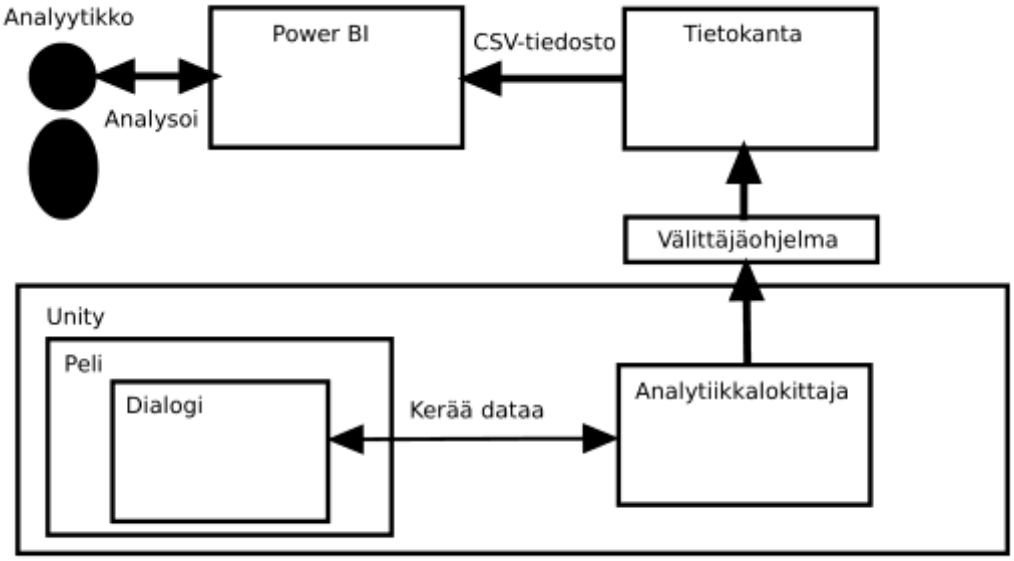
6.9 Ulkopuoliset sovellukset

Power BI on Microsoftin liiketoiminta analytiikkaan perustuva visualisointityökalu, jonka avulla raa’an datan voi muuttaa käytännöllisiksi oivalluksiksi (Microsoft Learn 2025). Power BI tukee erityisesti CSV-tiedostoja ja siksi data on tärkeää saada CSV-tiedostoiksi. Power BI:tä ei liitetä suoraan analytiikkalokittajaan tai tietovarastoihin, vaan kerättyä dataa on tarkoitus visualisoida sillä erikseen ja sen kautta tehdä päätöksiä.

6.10 Synteesi

Tässä luvussa on kuvattu analytiikkalokittajan toteutukselle tärkeät työkalut ja tekninen arkkitehtuuri pääpiirteittäin. Yhdessä ne muodostavat kokonaisuuden, joka alkaa siitä, että peliä kehitetään Unityssä. Kyseisessä ympäristössä peliin luodaan testidialogi käyttäen Olipa Gamesin omaa dialogityökalua,

DialogueMaster toolia. Analytiikkalokittaja rakennetaan skriptipohjaisena komponenttina Unityssä ja se liitetään keräämään dataa testidialogissa. Data varastoidaan tietokantaan välittäjäohjelman kautta, jotta tietoturvariskit minimoitaisiin. Sitten data haetaan tietokannasta CSV-tiedostoksi ja sitä visualisoidaan Power BI:n avulla. Visualisoinnin pohjalta tarvittavat analyysit ja johtopäätökset tehdään analytiikan neljän peruslähtökohdan mukaisesti. Kuvassa 9 oleva kaavio kuvaa toteutuksen kokonaisuutta.



Kuva 9. Kaavio toteutuksen arkkitehtuurista

7 Analytiikkalokittajaa koskevat sovelluskehityksen periaatteet

Analytiikkalokittajan suunnittelussa on tärkeää kartoittaa toivotut periaatteet koskien sovelluskehitystä, jotta saadaan aikaiseksi paras mahdollinen lopputulos. Sovelluskehityksen periaatteilla tarkoitetaan käytäntöjä ja filosofioita, joiden avulla lähestytään kehitystyötä. Tarkoituksena on saada aikaan mahdollisimman yksinkertainen ja toimiva ratkaisu, joka on skaalautuva ja sen voi käyttää uudelleen mahdollisimman monissa tilanteissa.

7.1 Yksinkertaisuus

Analytiikkalokittajan on oltava mahdollisimman yksinkertainen toteutukseltaan, koska, monimutkaisuus voi myös vaikeuttaa ohjelman uudelleen käytettävyyttä muissa tilanteissa, kuten toisissa peleissä. Liian sekavaa koodia on vaikeampi muokata käytettäväksi muissa peleissä ja joissakin tilanteissa ohjelma joudutaan tekemään kokonaan uudestaan vastaamaan kyseisiä tarpeita.

Lisäksi monimutkainen koodi on raskaampi ajaa ja vie tietokoneen suorituskyvyltä enemmän resursseja kuin yksinkertainen ohjelma. Hyvin monimutkainen ja raskas analytiikkalokittaja taustalla voi haitata pelin suorituskykyä ja tällöin analytiikkalokittajasta on enemmän haittaa kuin hyötyä. Tämä johtuu siitä, että videopelien suorituskyky on keskeisessä roolissa: huonosti toimiva ja heikosti pyörivä peli voi johtaa negatiivisiin arvosteluihin ja sitä kautta heikkoon myyntiin.

Liian monimutkaista analytiikkalokittajaa voi olla myös vaikea käyttää, koska erilaisia ominaisuuksia on liikaa. Nämä ominaisuudet voivat olla myös turhia tai ne tarjoavat vain vähän hyötyä. Tällaisia ominaisuuksia voisi teoriassa olla esimerkiksi erilaisten metatietorakenteiden tai muiden tietojen keruu, joille ei ole selkeää käyttötarkoitusta. Turha tiedonkeruu ja liian suuri tiedonkeruu kasvattaa vain suorituskuormaa ilman selviä hyötyjä. Tästä voi olla itseasiassa vain haittaa, koska turhia tietoja voidaan joutua poistamaan jälkeenpäin tietokannasta. Liika turha tieto, voi myös vaikeuttaa hyödyllisen tiedon löytämistä tietokannasta. Tästä syystä analytiikkalokittaja rakennetaan siten, että sillä voi kerätä vain tarpeellista tietoa.

7.2 Uudelleenkäytettävyys

On Parempi tuottaa sellainen sovellus, jota voi käyttää uudelleen monissa eri tilanteissa kuin sovellus, jota voi käyttää vain yhdessä tilanteessa. Tällä tavoin maksimoidaan tehdyn työn hyöty. Vain yhteen tilanteeseen sopiva sovellus vie tulevaisuudessa kehitysaikaa, koska tulevaisuudessa voidaan joutua tuottamaan samankaltainen sovellus toiselle tilanteelle.

Mahdollisimman korkea uudelleenkäytettävyys saadaan aikaa sillä tavalla, että sovellus suunnitellaan modulaariseksi ja joustavaksi. Tämä tarkoittaa sitä, että sovelluksen eri toiminnot, kuten tiedonkeruu, käsittely ja tallennus erotetaan toisistaan selkeiksi kokonaisuuksiksi. Tällöin sovelluksen eri ominaisuudet eivät ole sotkeutuneet toisiinsa ja, jos yhtä joudutaan muokkaamaan jatkossa, se ei tarkoita sitä, että koko sovellus joudutaan tehdä uusiksi. Niin kuin aikaisemmin on myös mainittu, mahdollisimman yksinkertainen sovellus tukee koodin uudelleen käytettävyyttä.

7.3 Skaalautuvuus

Sovelluksesta on saatava mahdollisimman skaalautuva, koska se mahdollistaa sovelluksen kestävyyden pienien ja suurien datamäärien edessä. Skaalautuva arkkitehtuuri mahdollistaa sen, että suorituskyky ja luotettavuus pysyvät kunnossa, vaikka datamäärien kerääminen nousisikin dramaattisesti.

Skaalautuvuus saavutetaan erityisesti siten, että käytetään skaalautuvaa tietokantaa, joka sallii suuret tietomäärät. Esimerkiksi pelkälle JSON-tiedostolle suuren määrän datan tallentaminen voi johtaa ohjelman tukkeutumiseen. Toinen tapa tehdä mahdollisimman skaalautuva ratkaisu on myös tukeutua modulaarisuuteen, koska tällä tavoin yksittäisiä komponentteja voidaan skaalata ja päivittää itsenäisesti ilman, että koko sovellus hajoaa.

8 Analytiikkalokittajan konkreettiset ominaisuudet ja moduulit.

Analytiikkalokittajan konkreettisilla ominaisuuksilla tarkoitetaan sovelluksen teknisiä piirteitä, jotka toteutetaan käytännössä. Kyseiset piirteet jaetaan moduuleiksi, koska tällä tavoin voidaan maksimoida uudelleenkäytettävyys ja skaalautuvuus. Tässä luvussa käsitellään, mitä ominaisuuksia tulee mukaan toteutukseen.

8.1 Tiedonkeräys

Analytiikkalokittajan keskeisin tarkoitus on kerätä tietoa. Paras mahdollinen tapa toteuttaa kyseinen toiminto on niin, että lokittaja aktivoituu, kun pelaaja tekee valinnan. Jos analytiikkalokittaja kerää jatkuvasti dataa reilaajassa, syntyy paljon turhaa dataa, jolla ei tee mitään. Tämä voi hidastaa pelin suorituskykyä ja palvelin voi tukkeutua. Kattavan määrän hyödyllistä dataa saadaan myös, jos analytiikkalokittaja toteutetaan globaalisti, jolloin se kerää dataa pelaajien kaikista valinnoista.

Kerätyt tietueet on myös oltava tunnistettavia pelaajaan tallennustiedostoon kytkeytyvän tunnisteen kautta. Tämän täytyy olla mahdollista siksi, jotta voidaan seurata, mitä valintoja pelaajat ovat tehneet tietyllä läpipeluukerralla ja miten valinnat eroavat seuraavilla läpipeluukerroilla.

Pelaajan tunnisteen lisäksi analytiikkalokittajan tulee tallentaa dialogin tunniste, josta selviää, että mistä dialogista tiedot on kerätty. Sen jälkeen pitää selvitä, missä solmussa valinta on tehty, koska dialogissa voi olla monia eri kohtia, jossa on mahdollisuus valintaan. Sitten on tallennettava, minkä valinnan pelaaja on valinnut. Myös aika, jolloin valinta on tehty ja kesto, kauanko pelaaja on viettänyt dialogissa. Ajalla voi yrittää arvioida valinnan vaikeutta ja se keretään sekunteina, koska API-yhteys toimii int-tyyppisillä muuttujilla parhaiten.

8.2 Tallennus

Toinen tärkeä ominaisuus on tietojen tallentaminen tietokantaan. Tietojen tallentaminen tietokantaan toteutetaan API-kutsujen kautta. API tarkoittaa joukkoa ennalta määrättyjä sääntöjä, joiden avulla kaksi tai useampi sovellus kommunikoi keskenään (Goodwin 2025). Analytiikkalokittajan on muodostettava yhteys Unitystä tietokantaan, jotta se voi lähettää sinne kerätyt tiedot POST-pyyntöjen avulla. POST on juuri se protokolla, jonka avulla lähetetään tietoja eteenpäin.

Tiedot on siirrettävä kuitenkin turvallisesti pelimoottorista tietokantaan. Tämän vuoksi tarvitaan koodi, joka toimii välittäjänä pelimoottorin ja tietokannan välillä. Tämä koodi on tarpeellinen siksi, koska sen avulla tiedot voidaan siirtää

tietokantaan turvallisesti. Suora yhteys tietokantaan voisi altistaa sen mahdollisille tietoturvariskeille.

8.3 Ajastin

Jotta on mahdollista mitata aikaa, jonka pelaaja käyttää dialogissa tehdessään valinnan, on luotava ajastin. Ajastin luodaan omaksi luokakseen, koska se parantaa koodin uudelleenkäytettävyyttä ja ylläpidettävyyttä. Lisäksi erillinen luokka mahdollistaa ajastimen käytön myös pelin muissa osissa ilman, että logiikkaa tarvitsee kirjoittaa uudelleen.

Ajastimessa on oltava kolme toimintoa: ajastimen aloitus, ajan palauttaminen valinnasta sekä ajastimen lopetus. Myös analytiikkalokittajaan rakennetaan metodit, jotka kutsuvat näitä ajastimen metodeja, koska näin ajastimen käyttö voidaan keskittää analytiikkajärjestelmään, mikä varmistaa sen, että mittaustiedot tallennetaan yhdenmukaisesti.

8.4 Yhteenveto

Tässä vaiheessa voidaan huomata, ettei analytiikkalokittaja ole vain yksi sovellus, mutta se enemmän monen eri komponentin kokonaisuus, jotka yhdessä vastaavat tiedot keräämisestä, käsittelystä ja tallentamisesta.

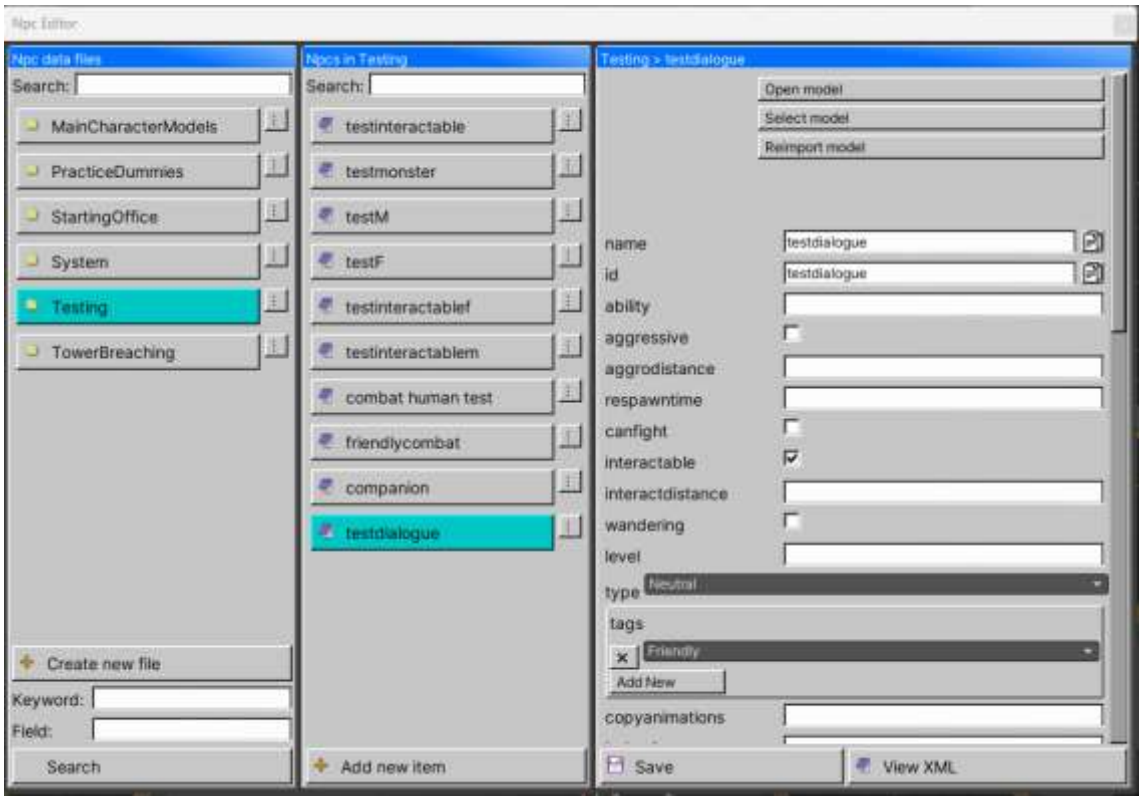
9 Toteutus

9.1 Ei pelattavan hahmon luonti dialogia varten

Tässä toteutuksessa tehdään testihahmo, joka ei tule oikeaan julkaistavaan peliin. Testihahmolle luodaan testidialogi, jolle rakennetaan analytiikkalokittaja. Hahmo rakennetaan Olipa Gamesin hahmonluontityökaluilla.

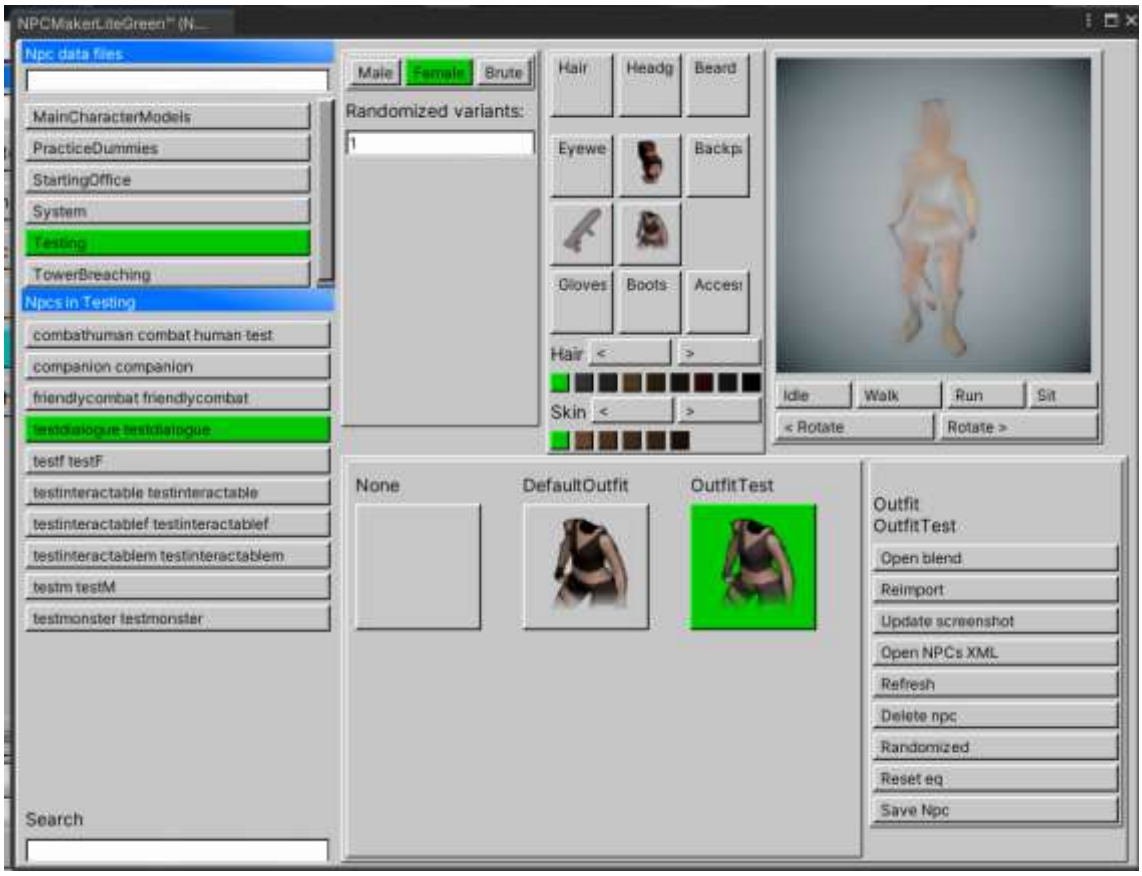
Aluksi luodaan tarvittava kansio, johon hahmon xml-tiedosto sijoitetaan. Kansio nimetään ”testdialogue”-nimiseksi ja kansio tulee ”Testing”-kansion alle. Id:ksi tulee myös ”testdialogue”, koska hahmon id on helppo muistaa, jos se on sama kuin hahmon nimi. Hahmosta tehdään myös ”interactable”, jotta sen kanssa voi

käydä dialogia myöhemmin. Kuvassa 10 määritellään konkreettisesti hahmon ominaisuudet ja kansio, johon hahmon tiedosto menee Npc Editorissa.



Kuva 10. Hahmon ominaisuuksien määrittely Npc Editorissa.

Tämän jälkeen hahmolle annetaan ulkomuoto NPCMakerLiteGreenissä. Hostile Docking pelissä on tällä hetkellä vain muutamaa erilaista asuvaihtoehtoja, joten valitsen mieluiset niistä, mitä on. Hahmon ulkonäöllä ei kuitenkaan ole mitään väliä tässä toteutuksessa. Kuvassa 11 hahmon ulkonäkö määritellään NPCMakerLiteGreenissä.



Kuva 11. Hahmon ulkomuodon määrittely NPCMakerLiteGreenissä.

Ei-pelattava hahmo lisätään peliin npcSpawnerilla, jotta hahmon voi kohdata testitilanteessa. Hahmo esiintyy näkymässä keltaisena hahmona, jos peli ei ole päällä. Kuvassa 12 esitetään hahmo, joka on lisätty npcSpawner-työkalun avulla. Kuvassa 13 puolestaan esitetään testaus, toimiiko hahmo pelin käynnistyksen yhteydessä.

Kuva 12. Hahmon lisääminen näkymään.

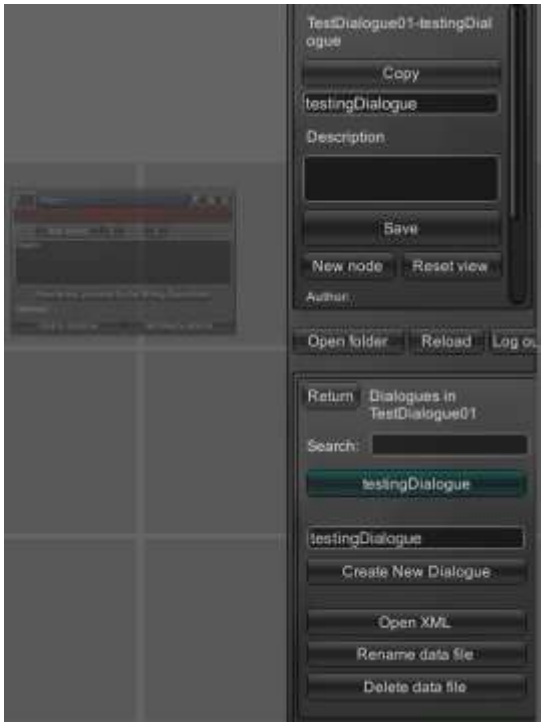
Kuva 13. Hahmo näkymässä pelin käynnistyessä.

9.2 Testidialogin luonti

Aluksi luodaan tarvittava kansio Olipa Gamesin dialogityökalulla, DialogueMaster 3000:lla. Dialogi kirjoitetaan englanniksi ja kansion luominen tapahtuu DialogueMasterin käytöliittymässä kuvan 14 esittämällä tavalla.

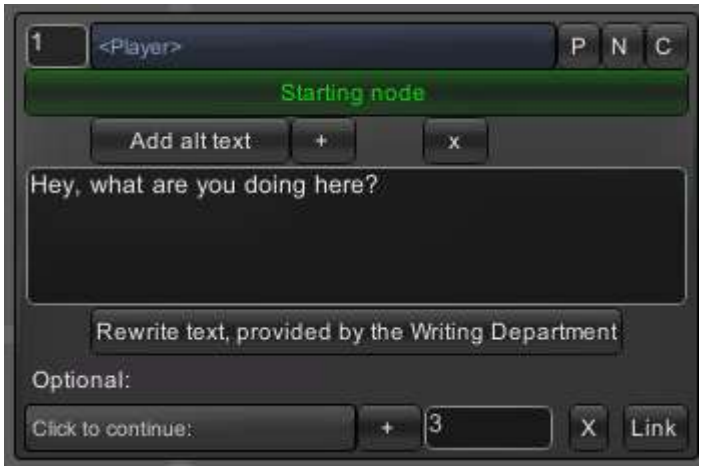
Kuva 14. Dialogi-kansion luominen DialogiMaster 3000 käyttöliittymässä.

Sitten luodaan itse dialogitiedosto ja nimetään se testingDialogueksi. Sen jälkeen voi alkaa luomaan uusia solmuja ”New node”-kohdasta. Solmuihin määritellään, kuka on repliikin puhuja, mihin eri vaihtoehtoihin se johtaa ja mitä eri toimintoja siitä voi seurata. Kuvassa 15 esitetään Dialoguemasterin käyttöliittymä solmuja lisättäessä.



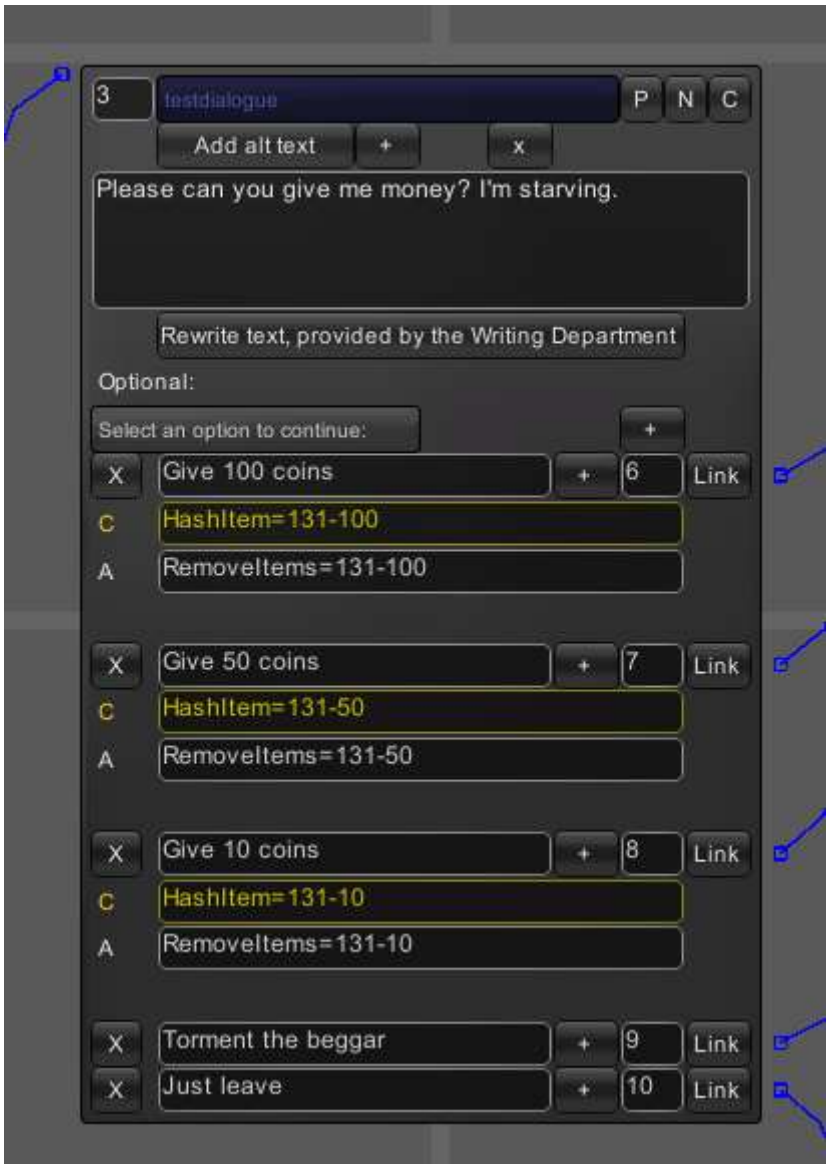
Kuva 15. solmujen luominen DialogueMaster 3000 käyttöliittymässä.

Seuraavaksi määritellään dialogin aloitus kohdaksi pelaajan repliikki, jossa pelaaja kysyy, että mitä hahmo tekee täällä. Kuvassa 16 esitetään solmu, missä kyseinen repliikki on määritelty DialogueMaster 3000 käyttöliittymässä.



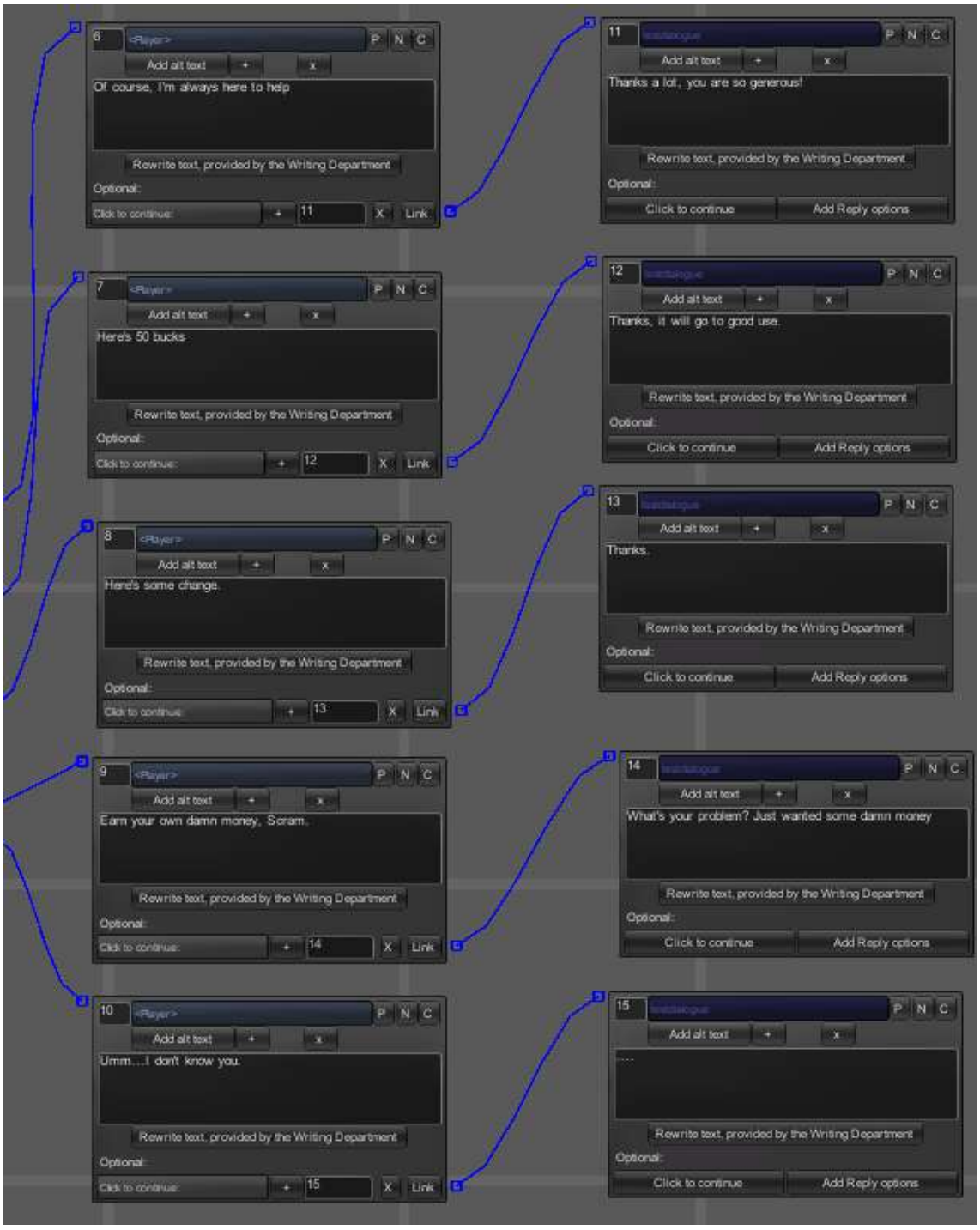
Kuva 16. Ensimmäisen repliikin määrittely.

Sen jälkeen hahmo kysyy pelaajalta rahaa. Pelaajalle annetaan viisi vaihtoehtoja, joista ensimmäinen on antaa hahmolle 100 kolikkoa. Toinen vaihtoehto on antaa hahmolle 50 kolikkoa ja kolmas vaihtoehto on antaa hahmolle vain kymmenen kolikkoa. Neljännessä vaihtoehdossa pelaaja haukkuu hahmon ja viidennessä hän vain poistuu vähin äänin. Tässä vaihtoehtojen skaalassa on tietynlainen moraalinen ulottuvuus, ja tällä tavoin voi teoriassa mitata pelaajien halukkuutta moraalisiin päätöksiin, hänen halukkuuttaan tehdä moraalittomia valintoja tai hänen välinpitämättömyyttään. Tämä solmu on erityisen tärkeä analytiikan kannalta, koska siinä määritellään pelaajan mahdolliset valinnat, joista on tarkoitus kerätä tietoa. Kuvassa 17 esitetään solmu, missä repliikki ja siihen määritellyt valinnat on asetettu DialogueMaster 3000 - käyttöliittymässä.



Kuva 17. Valintojen määrittely dialogissa.

Jokaiselle vaihtoehdolle on olemassa pelaajan oma repliikki ja hahmon uniikki vastaus. Tässä kuvassa näkyy mihin vastauksiin ja lopputulokseen pelaajan vallinnat johtavat. Jos pelaaja valitsee antaa hahmolle sata kolikkoa, pelaajaa vastaa, että ” Of course.. I’m always here to help” ja hahmo kiittää häntä. Kuvassa 18 esitetään solmu, missä, jossa pelaajan valinta ja hahmon siihen antama uniikki vastaus näkyvät selkeästi.

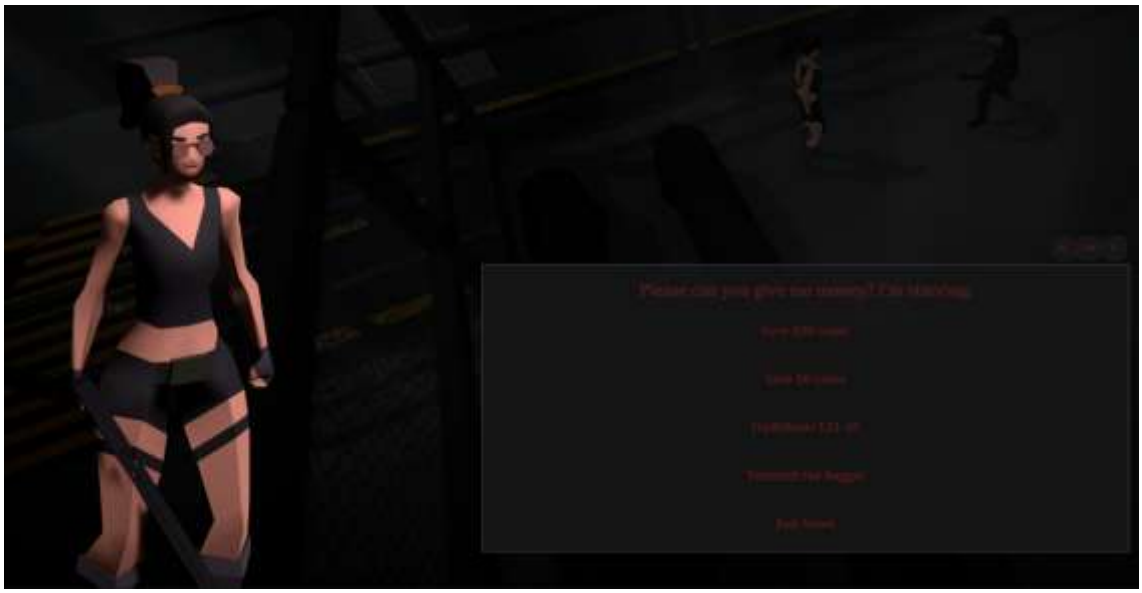


Kuva 18. Valintojen vastaukset ja lopputulokset

Lopuksi vielä testidialogi liitetään äsken luotuun ei-pelattavaan hahmoon Event Editorin kautta. Even editorissa luodaan aluksi kansio, johon kyseinen tapahtuma menee ja sitten määritellään toiminto, luotu dialogi ja kyseisen hahmon id. Kuvassa 19 esitetään, kuinka testidialogi liitetään hahmoon Event-editorin kautta.

Kuva 19. Testidialogin liittäminen hahmoon Event-editorin kautta.

Kuvassa 20 näytetään vielä, miltä dialogin käyttöliittymä näyttää itse pelissä. Pelaaja voi valita viidestä eri vaihtoehdosta.

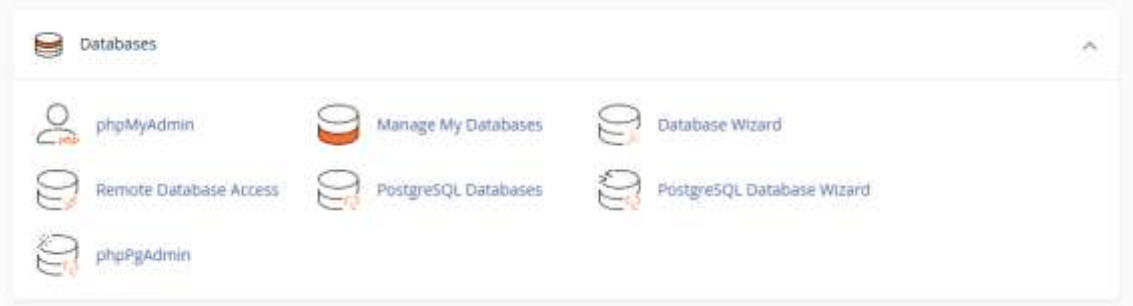


Kuva 20. Dialogin käyttöliittymä pelissä.

9.3 Tietokanta

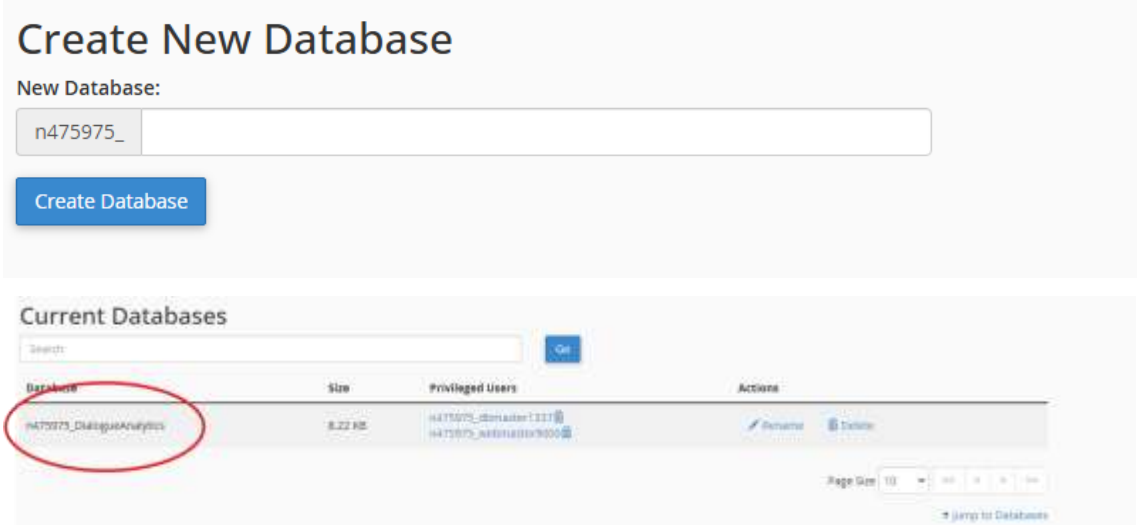
9.3.1 Tietokannan luonti

Tietokanta luodaan niin, että aluksi navigoidaan Cpanelin sivulta ”Manage My Databases”-kohtaan. Myöhemmin luotua tietokantaa voi tarkastella ja muokata phpMyAdminin kautta. Kuvassa 21 esitetään miltä näyttä eri vaihtoehdot, jolla voi hallita tietokantoja Cpanelin sivuilla.



Kuva 21. Tietokantojen hallinta Cpanelin sivulla.

Sitten nimetään uusi MySQL-tietokanta ja luodaan se painamalla ”Create Database”-nappia. Sivulta voi myös asettaa käyttäjiä tietokannoille. Kuvassa 22 esitetään, miten tietokanta luodaan Cpanelin sivuilla.

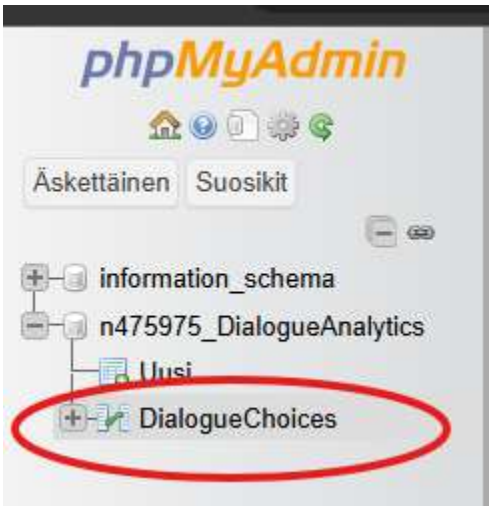


Kuva 22. Tietokannan luonti

Tietokantaan pitää kuitenkin nyt luoda tarvittavat taulut ja sarakkeet. Tämä onnistuu siten, että mennään takaisin Cpanelin etusivulle ja valitaan PhpMyAdmin, joka on tietokantojen käyttöliittymä ja hallintatyökalu.

9.3.2 Taulut ja sarakkeet

PhpMyAdminissa tietokannalle luodaan yksi taulu, joka nimetään DialogueAnalyticsiksi. Taulu on tietokannan perusrakenne, jolla dataa järjestetään riveihin ja sarakkeisiin. Tämä helpottaa tiedon hallintaa ja hakua. (GeeksForGeeks 2024). Kuvassa 23 luodaan tietokantaan taulu PhpMyAdminissa.



Kuva 23. Taulun luonti tietokantaan.

Seuraavaksi taululle määritellään sarakkeet, joita tulee yhteensä seitsemän. Sarakkeet määräävät tietueiden attribuutit. ID on yksittäisen valinnan tunniste, joka erottaa tietueen muista tietueista. ID generoidaan automaattisesti auto-increment-toiminnon perusteella, eikä sitä lisätä manuaalisesti. Player_id on pelaajan henkilökohtainen tunniste, joka on sidottu tallennustiedostoon. Tämän avulla voidaan kartoittaa sitä, että mikä pelaaja on tehnyt valinnan dialogissa. Dialogie_id kertoo taas mikä dialogi on kyseessä ja sinne varastoidaan kyseisen dialogin nimi. Tässä tapauksessa dialogue_id on ”testingDialogue”. Node_id:seen tallennetaan missä solmukohdassa valintaa on tehty. Jos dialogissa haarautuu enemmän kuin yhden kerran ja siinä on mahdollista tehdä monta eri valintaa, dialogue_id pysyy samana, mutta node_id kertoo mikä valinta on ollut kyseessä. Node_id kertoo valinnan solmun numeron. Option_id kertoo, minkä valinnan pelaaja on valinnut siinä solmukohdassa, johon node_id viittaa. Timestamp kertoo kellonajan, jolloin valinta on tehty. Timestampin kellonaika on sillä aikavyöhykkeellä missä tämä serveri on, eli tässä tapauksessa Yhdysvaltain itärannikolla (UTC-04:00). Duration kuvaa sekunteina, kuinka kauan pelaajalla on kulunut aikaa tietyn valinnan tekemiseen. Myöhemmät valinnat näyttävät korkeamman keston, koska luonnollisesti pelaajalla kestää kauemmin päästä sinne, joten kestoa kannattaa lukea node_id:n yhteydessä. Kuvassa 24 näytetään miltä nämä sarakkeet näyttävät tietokannassa.

#	Nimi	Tyyppi	Aakkosjärjestys	Attribuutit	Tyhjä	Oletusarvo	Kommentit	Lisätiedot	Toiminnot
1	id	int(11)			Ei	None		AUTO_INCREMENT	Muokkaa Tuhoa Lisää
2	player_id	varchar(100)	utf8mb3_general_ci		Ei	None			Muokkaa Tuhoa Lisää
3	dialogue_id	varchar(100)	utf8mb3_general_ci		Ei	None			Muokkaa Tuhoa Lisää
4	node_id	int(11)			Ei	0			Muokkaa Tuhoa Lisää
5	option_id	int(11)			Ei	0			Muokkaa Tuhoa Lisää
6	timestamp	datetime			Ei	current_timestamp()			Muokkaa Tuhoa Lisää
7	duration	int(11)			Ei	0			Muokkaa Tuhoa Lisää

Kuva 24. Sarakkeet tietokannassa.

9.3.3 Tietokantaan yhdistäminen

Jotta tarvittavat tiedot pelistä voidaan tallentaa tietokantaan analytiikkalokittajan avulla, luodaan koodi, joka toimii välittäjänä pelimoottorin ja tietokannan välillä. Tämä koodi vastaanottaa aikaisemmin kuvatut tiedot, jotka halutaan lisätä tietokantaan.

Tämä koodi on tarpeellinen siksi, koska sen avulla tiedot voidaan siirtää tietokantaan turvallisesti. Suora yhteys tietokantaan voisi altistaa sen mahdollisille tietoturvariskeille, kuten SQL-injektoinneille ja tietomurroille. SQL-injektio tarkoittaa kyberhyökkäystä, jonka avulla hakkeri syöttää haitallisen SQL-lauseen tietokantaan päästäkseen siihen käsiksi (Plexicus 2025).

Tämä koodi on kirjoitettu PHP-ohjelmointikielellä, koska PHP ei tarvitse erillistä tukea selaimelta, kuten erillistä palvelinasennusta ja sillä voi käsitellä helposti palvelinpuolen tiedostoja (Laaksonen 2011). Välittäjäohjelman nimeksi tulee analytics.php ja se kehitetään Visual Studio Codessa, ohjelmistoympäristön yksinkertaisuuden vuoksi.

Analytics.php-tiedoston ohjelmakoodi ja sen toiminnot on selitetty tarkemmin koodiin sisältyvissä kommenteissa. Kyseinen ohjelmakoodi esitetään alla olevien kuvien muodossa.

Aluksi määritellään yhteystietokantaa, jotta myöhemmin toteutettava analytiikkalokittaja tietää minne tallentaa kerätyt tiedot. Kuvassa 25 esitetään, miten koodissa on tämän lisäksi määritetty virheenkäsittely, mikäli yhteyden

muodostaminen epäonnistuu. Käyttäjätunnus ja salasana on mustattu pois, koska ne ovat arkaluonteista tietoa.

analytics.php

C: > Users > mttnn > Downloads > analytics.php

```php
<?php
// Ottaa virheiden raportoinnin käyttöön kehitysympäristössä
error_reporting(E_ALL);
ini_set('display_errors', 1);

//Tietokantayhteyden asetukset
$servername = "localhost";
$username = [redacted]; // cPanel MySQL käyttäjä
$password = [redacted]; // cPanel MySQL salasana
$dbname = "n475975_DialogueAnalytics"; // Tietokannan nimi

// Muodostaa yhteyden tietokantaan
$conn = new mysqli($servername, $username, $password, $dbname);

// Virheen käsittely
if ($conn->connect_error) {
    die("Yhteys epäonnistui: " . $conn->connect_error);
}

```

Kuva 25. Tietokantayhteyden määrittely analytics.php-tiedosdossa.

Kuvassa 26 esitetään, miten määritellään POST-kutsut, jotta välittäjä tietäisi mitkä kutsut on lisättävä palvelimelle. Analytiikkalokittajaan on myöhemmin rakennettava systeemi, joka sopii yhteen välittäjäohjelman Post-kutsujen kanssa.

```php
// POST-kutsut tietokantaan lisäämistä varten
$playerId = $_POST['player_id'] ?? '';
$dialogueId = $_POST['dialogue_id'] ?? '';
$node = $_POST['node_id'] ?? 0;
$option = $_POST['option_id'] ?? 0;
$duration = $_POST['duration'] ?? 0;
```

Kuva 26. POST-kutsujen määrittely analytics.php-tiedosdossa.

Kuvassa 27 esitetään, miten koodissa määritellään SQL-lause, jonka avulla tiedot lisätään niille kuuluviin sarakkeisiin tietokannassa.

```
27   // Luo SQL-lauseen, joka lisää uuden rivin DialogueChoices-tauluun.
28   $stmt = $conn->prepare("INSERT INTO DialogueChoices (player_id, dialogue_id,
29
```

Kuva 27. SQL-lauseen luonti, joka lisää uuden rivin tietokantaan.

Kuvassa 28 esitetty osa ohjelmakoodista kuuluu kuvan 27 esittämään lauseeseen, jossa määritellään SQL-lause. Kuvat esitetään kahdessa eri kuvassa yhden sijaan, koska liian pitkäksi venyneestä kuvasta on vaikea saada mitään selvää.

```
, node_id, option_id, timestamp, duration) VALUES (?, ?, ?, ?, NOW(), ?)");
```

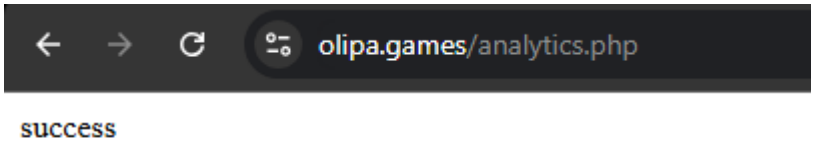
Kuva 28. SQL-lauseen jatko.

Kuvassa 29 määritellään komento, joka sitouttaa parametrit SQL-lauseeseen, joka on esitetty kuvissa 27 ja 28. Kyseisessä lauseessa, sana ”ssiii” on anagrammi, ja sen kirjaimet tarkoittavat tietotyyppejä, joihin tallennetut tietueet kuuluvat. Esimerkiksi kirjain s tarkoittaa tietotyyppiä string ja kirjain i tarkoittaa tietotyyppiä int. Lopuksi yhteys suljetaan, jotta päällekkäisiä yhteyksiä ei syntyisi. Tämä voisi kuormittaa järjestelmää. Lisäksi kuvassa 29 esitetään komennot, joka käsittelevät virheen käsittelyä ja sulkevat yhteyden.

```
30   // Tarkista, onnistuiko valmistelu
31   if (!$stmt) {
32       die("Prepare epäonnistui: " . $conn->error);
33   }
34
35   // Sitouttaa parametrit SQL-lauseeseen
36   $stmt->bind_param("ssiii", $playerId, $dialogueId, $node, $option, $duration);
37
38   // Suorittaa SQL-lauseen ja tarkistaa, onnistuiko se
39   if ($stmt->execute()) {
40       echo "success";
41   } else {
42       echo "SQL error: " . $stmt->error;
43   }
44
45   // Sulkee yhteyden
46   $stmt->close();
47   $conn->close();
48   ?>
```

Kuva 29. SQL-lauseen sitouttaminen ja yhteyden sulku.

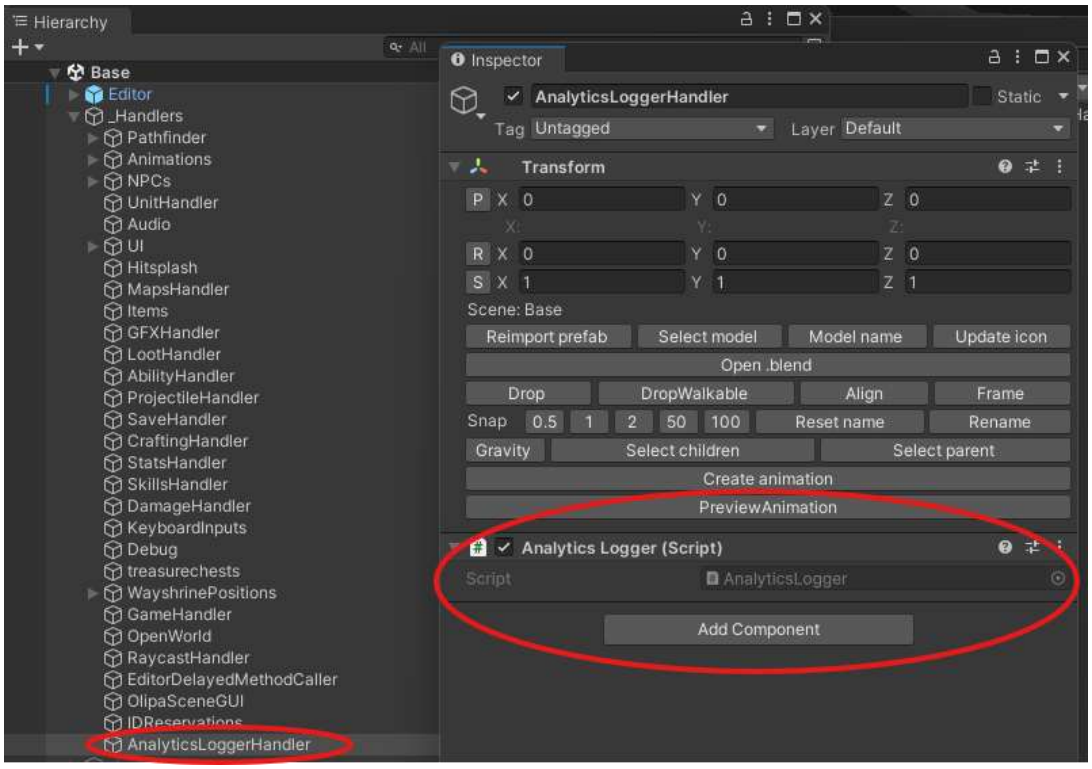
Yhteyden toimintaa testataan vielä selaimessa, jotta voidaan nähdä, että yhteys toimii. Jos yhteys toimii sivulla näkyy vain teksti ”success”, ja jos se ei toimi niin näkyy ”error” sekä siihen liittyvät syyt. Tämä luo tyhjän tietueen tietokantaan, mikäli testaaja on kirjautunut Cpaneliin samalla selaimella. Kuvassa 30 esitetään onnistunut yhteyden muodostaminen selaimen kautta.



Kuva 30. Yhteyden testaaminen selaimessa.

9.4 Analytiikkalokittajan luonti

Unity Enginessä luodaan pelin tiedostoihin GameObject-tyyppinen käsittelijä nimeltä AnalyticsLoggerHandler. Tähän käsittelijään luodaan tiedosto, jonka nimeksi tulee Analytics Logger. Tähän tiedostoon rakennetaan analytiikkalokittajan toiminnot. Itse kooditiedosto luodaan Unityyn kohdasta ”Add Component”, niin kuin kuvasta 31 näkyy.



Kuva 31. Analytiikkalokittaja-tiedoston luominen Unity-pelimoottorissa.

Analytiikkalokittaja ohjelmoidaan käyttäen Visual Studio 2019 ohjelmointiympäristöä, sen Unity-tuen vuoksi ja aluksi on tärkeä määritellä sovelluksen kannalta tärkeät muuttujat. Kuten aiemmin, myös kuvassa 32 ohjelmakoodi selitetään tarkemmin kommenttien avulla. Kuvassa 32 esitetään analytiikkalokittajalle keskeiset muuttujat ja kuinka ne määritellään ohjelmakoodissa.

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.Networking;
5
    Unity Script | 4 references
6   public class AnalyticsLogger : MonoBehaviour
7   {
8       // Tämä muuttuja määrittelee palvelimen osoitteen, johon data lähetetään.
9       private string analyticsUrl = "https://olipa.games/analytics.php";
10      // Näihin muuttujiin tallennetaan pelajaan tunniste
11      private string playerId;
12      private string saveId;
13
14      // Tässä määritellään luokan singleton-malli, johon voi viitata globaalisti.
15      public static AnalyticsLogger instance;
16
17      // Tässä luodaan ilmentymä Timer (ajastin) luokasta, jotta pelaajan aikaa keskustelussa voidaan mitata
18      private Timer timer = new Timer();
```

Kuva 32. Analytiikkalokittajan tärkeät muuttujat.

Tärkeää on luoda luokasta singleton-instanssi, jotta sitä voi kutsua mistä tahansa pelin osasta ilman, että syntyy päällekkäisiä instansseja. Tässä kohtaa

määritellään pelaajan tunniste, jotta pelaajan voi tunnistaa tietokannasta myöhemmin (player_id). Pelaajan tunniste tallennetaan, jotta voitaisiin nähdä, mitä valintoja pelaajat ovat tehneet jollain tietyllä läpipeluukerralla. Tämä tehdään Awake()-metodissa, koska se suorittaa kyseiset toiminnot ennen muita metodeja, jotta voidaan turvallisesti tallentaa pelaajan tiedot ilman virheitä tai päällekkäisiä instansseja. Kuvassa 33 esitetään, kuinka edellä mainitut asiat toteutetaan ohjelmakoodissa.

```
20      // Tämä metodi varmistaa, että pelissä on vain yksi analytiikkalokittaja
        Unity Message | 0 references
21      void Awake()
22      {
23          if (instance == null)
24              instance = this;
25          else
26              Destroy(gameObject); // estää duplikaatit
27          // Alustetaan playerId ja saveId tässä
28          playerId = PlayerPrefs.GetString("CBUID");
29          SaveHandler.onSaveLoaded += OnSaveLoaded;
30      }
31
32      // Tallentaa pelaajan id:n
        1 reference
33      void OnSaveLoaded(SaveHandler.Save s)
34      {
35          saveId = s.uniqueSaveId;
36      }
37
```

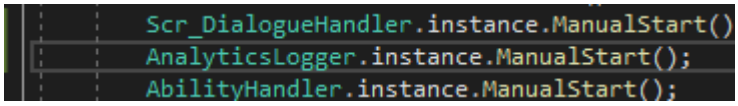
Kuva 33. Pelaajan tunnisteen määrittely analytiikkalokittajan käynnistyksessä.

Analytiikkalokittaja on hyvä käynnistää manuaalisesti, koska jos sen instanssi luodaan ennen Scr_DialogueHandleria, Scr_DialogueHandlerin OnPlayerPickOption()-metodia ei voida kutsua. Tämä metodi on vastuussa pelaajan valintojen havaitsemisesta ja niiden välittämisestä analytiikkaan. Scr_DialogueHandlerin puolestaan vastaa dialogien yleisistä toiminnoista, kuten niiden aloittamisesta ja lopettamisesta. Kuvassa 34 esitetään, kuinka edellä mainitut asiat toteutetaan ohjelmakoodissa.

```
32      // Luodaan manuaalinen aloitusmetodi, jotta voidaan hallita ohjelman alkamista.
33      // Aloitusmetodi liittää tämän kuuntelemaan tapahtumaa, joka laukeaa, kun pelaaja valitsee dialogivaihtoehdon.
        1 reference
34      public void ManualStart()
35      {
36          Scr_DialogueHandler.instance.OnPlayerPickOption += OnPlayerPickOption;
37      }
38
```

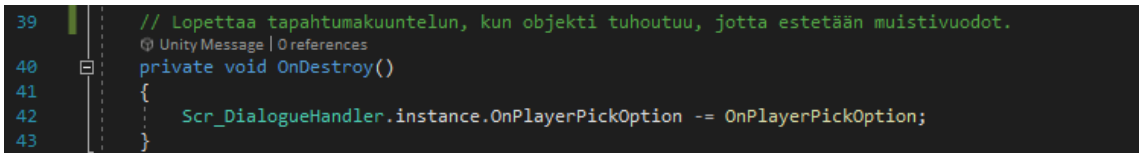
Kuva 34. Analytiikkalokittajan manuaalinen aloitus.

Analytiikkalokittaja aloitetaan GameHandler.cs skriptissä, joka vastaa pelin kaikkien yleisten toimintojen hallinnasta. Järjestys on tärkeää, koska analytiikkalokittaja ei löydä Scr_DialogueHandlerin instanssia, jos tämä käynnistyy ennen Scr_DialogueHandleria. Kuvassa 35 esitetään, kuinka edellä mainitut asiat toteutetaan ohjelmakoodissa.



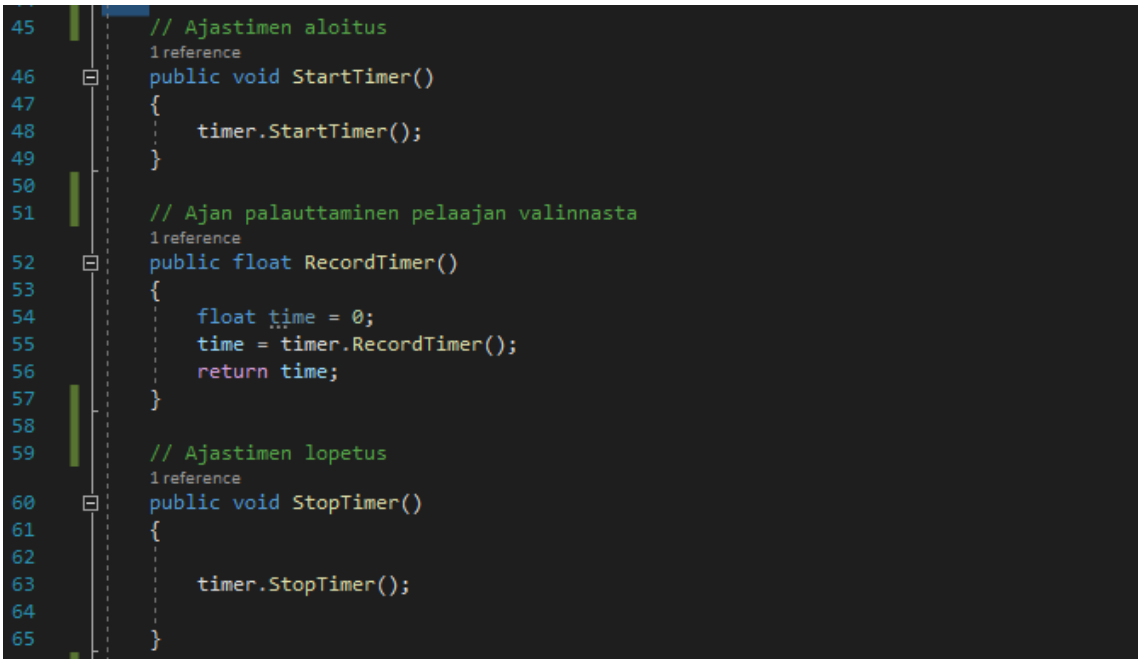
Kuva 35. Analytiikkalokittajan manuaalinen aloittaminen GameHandler.cs-tiedostossa.

Tapahtumakuuntelu on myös hyvä lopettaa silloin kuin AnalyticsLoggerHandler-objekti tuhoutuu, jotta estetään muistivuodot. Tämä erityisen tärkeää järjestelmän kuormittumisen estämisen kannalta. OnDestroy()-metodi määrittelee mitä käy objektin tuhoutuessa. Kuvassa 36 esitetään, kuinka edellä mainitut asiat toteutetaan ohjelmakoodissa.



Kuva 36. Tapahtumakuuntelun lopettaminen.

Analytiikkalokittajaa varten on tehty myös erillinen Timer-luokka, eli ajastin luokka. Erillinen luokka noudattaa modulaarisuuden periaatteita, jotta saavutettaisiin mahdollisimman korkea uudelleenkäytettävyys. Aikaisemmin analytiikkalokittajaan on luotu kyseisen ajastimen ilmentymä ja nyt viitataan ajastimen keskeisiin funktioihin. Kuvassa 37 esitetään, kuinka edellä mainitut asiat toteutetaan ohjelmakoodissa.



```
45      // Ajastimen aloitus
        1 reference
46      public void StartTimer()
47      {
48          timer.StartTimer();
49      }
50
51      // Ajan palauttaminen pelaajan valinnasta
        1 reference
52      public float RecordTimer()
53      {
54          float time = 0;
55          time = timer.RecordTimer();
56          return time;
57      }
58
59      // Ajastimen lopetus
        1 reference
60      public void StopTimer()
61      {
62
63          timer.StopTimer();
64
65      }
```

Kuva 37. Timer-luokkaan liitännäiset metodit.

Tässä rakennettu Timer-luokka (Timer.cs). Periaatteessa tätä ajastinta voisi myös käyttää pelin muissa toiminnoissa, mutta silloin sen toimintoja ei kannata johtaa analytiikkalokittajasta käsin. Ajastimessa on kolme metodia, joista ensimmäinen aloittaa laskurin, toinen palauttaa kuluneen ajan sekunteina ja kolmas lopettaa laskurin. Aikaa ei palauteta lopetuksen yhteydessä sen vuoksi, koska dialogissa voi olla monta kohtaa, jossa valinta on mahdollinen. Tämä tarkoittaisi sitä, että myöhempien valintojen aikoja ei voitaisi tallentaa tietokantaan. Haittana tässä ratkaisussa kuitenkin on se, että myöhempiin valintoihin käytetty aika esiintyy aina korkeampina kuin ensimmäiseen käytetty aika. Taulusta näkee kuitenkin solmun missä valinta on tehty, joten on helppo arvioida mikä on ensimmäinen valinta ja mikä on toinen. Kuvassa 38 esitetään Timer-luokkaa.

```
Unity Script | 2 references
public class Timer : MonoBehaviour
{
    private float startTime;
    private float endTime;
    private bool running = false;

    // Aloittaa laskurin
    1 reference
    public void StartTimer()
    {
        startTime = Time.time;
        running = true;
    }

    // Palauttaa kuluneen ajan sekunteina
    1 reference
    public float RecordTimer()
    {
        if (!running)
            return 0f;

        endTime = Time.time;
        return endTime - startTime;
    }

    // Lopettaa laskurin
    1 reference
    public void StopTimer()
    {
        running = false;
        startTime = 0;
    }

}
```

Kuva 38. Timer-luokka.

Ajastin aloitetaan luomalla analytiikkalokittaja instanssi Src_ DialogueHandlerissa samaan aikaan, kun dialogi alkaa. Ajastin johdetaan analytiikkalokittajasta, koska sen avulla ajastin voidaan liittää tarkasti lokittajan toimintoihin. Kuvassa 39 esitetään, kuinka analytiikkalokittajasta johdettu ajastin aloitetaan Src_ DialogueHandlerissa.

```
public class Scr_DialogueHandler
    public void StartDialogue(string dialogueDataFile, string dialogueId, int nodeId, object focusObject)

else
{ // starting fresh dialogue
    NPCHandler.instance.StopAllNpcs();
    AudioHandler.instance.PlayDialogue(nodeId, 0);
    AnalyticsLogger.instance.StartTimer();
}
```

Kuva 39. Ajastimen aloitus Src_DialogueHandlerissa.

Ajastin myös lopetetaan, kun dialogi loppuu Src_ DialogueHandlerista käsin. Kuvassa 40 esitetään, kuinka analytiikkalokittajasta johdettu ajastin lopetetaan Src_ DialogueHandlerissa.

```
public class Scr_DialogueHandler
    public void StartDialogue(string dialogueDataFile, string dialogueId, int nodeId, object focusObject)
  else
  { // end dialogue
      Debug.Log("ending dialogue ");
      AnalyticsLogger.instance.StopTimer();

      EndDialogue();

  }
```

Kuva 40. Ajastimen lopetus Src_DialogueHandlerissa.

Kuten aiemmin on käsitelty, ajastimen aika palautetaan sillä hetkellä, kun pelaaja tekee valinnan. Tämä metodi lähettää kerätyt tiedot palvelimelle ja se liitetään kuuntelemaan Src_ DialogueHandlerin OnPlayerPickOption-metodia. Tässä metodissa kutsutaan myös SendAnalytics()-metodia, joka lähettää tiedot palvelimelle. Kuvassa 41 esitetään, kuinka aika tallennetaan analytiikkalokittajassa ja miten kerätyt valinnat siirtyvät palvelimelle SendAnalytics()-metodin avulle.

```
// Käynnistää coroutine-funktion, joka lähettää tiedot tietokantaan ja palauttaa ajan siinä kohtaa, kun pelaaja tekee valinnan.
2 references
void OnPlayerPickOption(string dialogueId, int node, int option)
{
    float duration = RecordTimer();

    StartCoroutine(SendAnalytics(playerId, dialogueId, node, option, duration));
}
```

Kuva 41. Ajan palauttaminen analytiikkalokittajassa ja tietojen lähettäminen palvelimelle.

Kuvassa 42 esitetty SendAnalytics()-metodi rakentaa POST-datan PHP-skriptin hyväksymässä muodossa, jotta sen voi siirtää palvelimelle. Tämä käsittää kaikki kerätyt tiedot, jotka siirretään tietokannan mukaisiin sarakkeisiin.

```
// Tämä metodi rakentaa POST-datan PHP-skriptin hyväksymässä muodossa ja lähettää sen palvelimelle.
1 reference
IEnumerator SendAnalytics(string playerId, string dialogueId, int node, int option, float duration)
{
    WWWForm form = new WWWForm();
    form.AddField("player_id", playerId);
    form.AddField("dialogue_id", dialogueId);
    form.AddField("node_id", node);
    form.AddField("option_id", option);
    form.AddField("duration", Mathf.CeilToInt(duration));

    using (UnityWebRequest www = UnityWebRequest.Post(analyticsUrl, form))
    {
        yield return www.SendWebRequest();

        if (www.result != UnityWebRequest.Result.Success)
        {
            Debug.LogError("Analytics failed: " + www.error);
        }
        else
        {
            Debug.Log("Analytics sent successfully.");
        }
    }
}
```

Kuva 42. POST-datan rakentaminen.

10 Datan käyttö

10.1 Datan hakeminen

Kerätyn datan löytää helposti PhpMyAdminin kautta ja sitä voi hakea erilaisilla SQL-kyselyillä. Esimerkiksi, kaikki tietueet, jotka sisältävät pelaajien dataa, voidaan hakea seuraavalla kyselyllä: SELECT * FROM ’DaialogueChoices’ WHERE player_id != ””;. Tämä siksi, koska mukaan ei haluta tietueita, jotka ovat syntyneet yhteyden testauksen tuloksena. Nämä tietueet eivät sisällä mitään dataa.

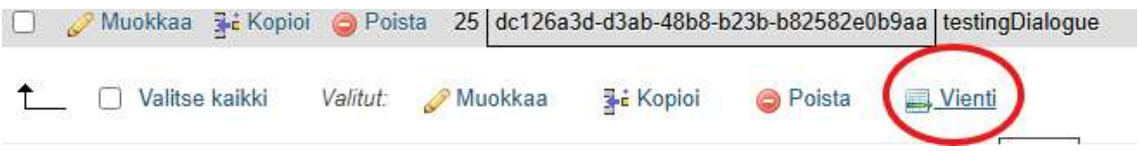
Kuvassa 43 on esitetty on kaikki pelaajien valintoihin liittyvät tietueet, jotka on haettu PhpMyAdminin kautta. Näissä tietueissa on myös mukana muita dialogeja tällä hetkellä, kuin vain testDialogue.

	id	player_id	dialogue_id	node_id	option_id	timestamp	duration
Muokkaa Kopioi Poista	3	dc126a3d-d3ab-48b8-b23b-b82582e0b9aa	testingDialogue	3	1	2026-01-21 08:04:22	0
Muokkaa Kopioi Poista	4	dc126a3d-d3ab-48b8-b23b-b82582e0b9aa	testingDialogue	3	3	2026-01-21 08:22:11	0
Muokkaa Kopioi Poista	5	dc126a3d-d3ab-48b8-b23b-b82582e0b9aa	testingDialogue	3	4	2026-01-21 11:10:51	5
Muokkaa Kopioi Poista	6	dc126a3d-d3ab-48b8-b23b-b82582e0b9aa	testingDialogue	3	4	2026-01-21 11:13:04	7
Muokkaa Kopioi Poista	7	dc126a3d-d3ab-48b8-b23b-b82582e0b9aa	testingDialogue	3	0	2026-01-21 11:13:55	13
Muokkaa Kopioi Poista	8	dc126a3d-d3ab-48b8-b23b-b82582e0b9aa	testingDialogue	3	0	2026-01-21 11:14:56	4
Muokkaa Kopioi Poista	12	dc126a3d-d3ab-48b8-b23b-b82582e0b9aa	testingDialogue	3	3	2026-01-22 08:39:33	5
Muokkaa Kopioi Poista	13	ed613474-2b47-45db-ac14-985efa7207ea	maintenanceworkertest	2	2	2026-01-22 13:37:58	558
Muokkaa Kopioi Poista	14	ed613474-2b47-45db-ac14-985efa7207ea	maintenanceworkertest	2	2	2026-01-22 13:40:45	17
Muokkaa Kopioi Poista	15	ed613474-2b47-45db-ac14-985efa7207ea	maintenanceworkertest	2	1	2026-01-22 13:43:44	67
Muokkaa Kopioi Poista	16	ed613474-2b47-45db-ac14-985efa7207ea	maintenanceworkertest	2	0	2026-01-22 13:43:48	71
Muokkaa Kopioi Poista	17	ed613474-2b47-45db-ac14-985efa7207ea	maintenanceworkertest	2	2	2026-01-22 13:43:51	74
Muokkaa Kopioi Poista	18	ed613474-2b47-45db-ac14-985efa7207ea	maintenanceworkertest	2	2	2026-01-22 14:09:44	2
Muokkaa Kopioi Poista	19	ed613474-2b47-45db-ac14-985efa7207ea	maintenanceworkertest	2	1	2026-01-22 14:17:37	1
Muokkaa Kopioi Poista	20	ed613474-2b47-45db-ac14-985efa7207ea	maintenanceworkertest	2	2	2026-01-22 14:17:39	3
Muokkaa Kopioi Poista	21	ed613474-2b47-45db-ac14-985efa7207ea	maintenanceworkertest	2	2	2026-01-22 14:22:26	68
Muokkaa Kopioi Poista	22	ed613474-2b47-45db-ac14-985efa7207ea	IntroGuy	3	2	2026-01-22 14:49:20	6
Muokkaa Kopioi Poista	23	ed613474-2b47-45db-ac14-985efa7207ea	IntroGuy	3	1	2026-01-22 14:49:30	3
Muokkaa Kopioi Poista	24	ed613474-2b47-45db-ac14-985efa7207ea	IntroGuy	10	2	2026-01-22 14:49:33	6
Muokkaa Kopioi Poista	25	dc126a3d-d3ab-48b8-b23b-b82582e0b9aa	testingDialogue	3	3	2026-01-23 03:13:07	4

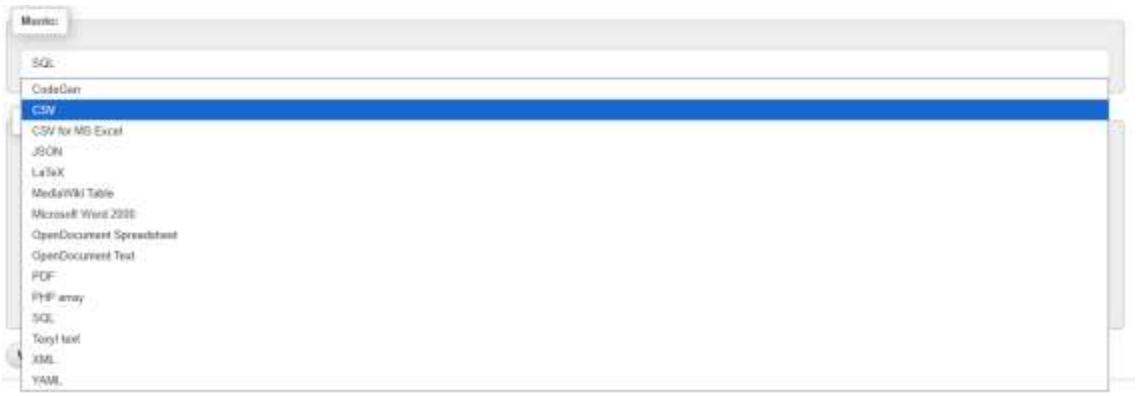
Kuva 43. tietueet taulussa.

10.2 Datan vienti

Jotta dataa voidaan mallintaa esimerkiksi Power Bi:llä, se on saatava ulos tietokannasta ja datan vienti onnistuu PhpMyAdminissa helposti sisäänrakennetun vientitoiminnon ansiosta. Datan voi myös viedä useisiin eri formaatteihin ja tässä tilanteessa valitaan CSV-tiedosto. Vienti on kyselykohtainen. Kuvassa 44 esitetään, kuinka data viedään CSV-tiedostoksi PhpMyAdminissa.



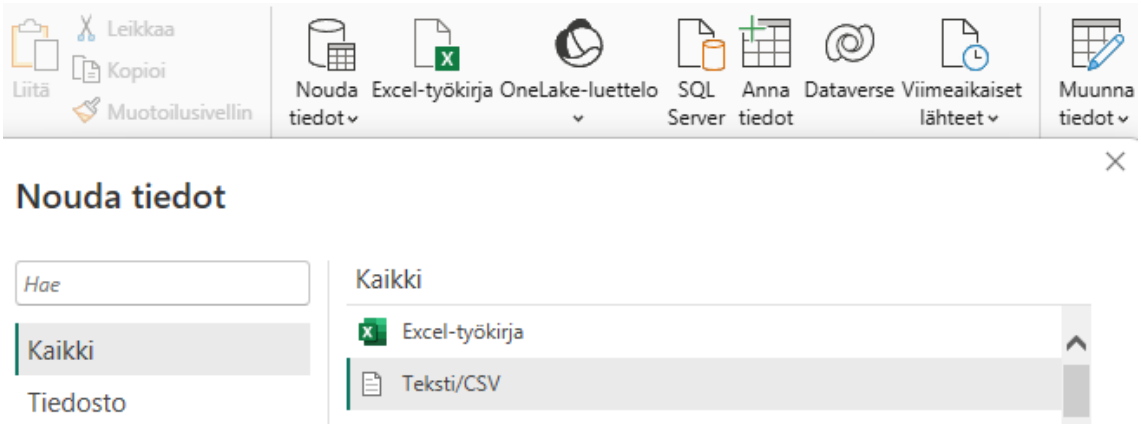
Kuva 44. Kyselyn vienti CSV-tiedostoksi.



Kuva 45. Tiedostoformaatin valinta.

10.3 Datan visualisointi ja suositut valinnat

Kuvassa 46 esitetään, kuinka avataan Microsoft PowerBI ja valitaan Nouda tiedot -valikosta Teksti/CSV, minkä jälkeen tiedot tuodaan sovellukseen, jotta tietoja voisi analysoida.

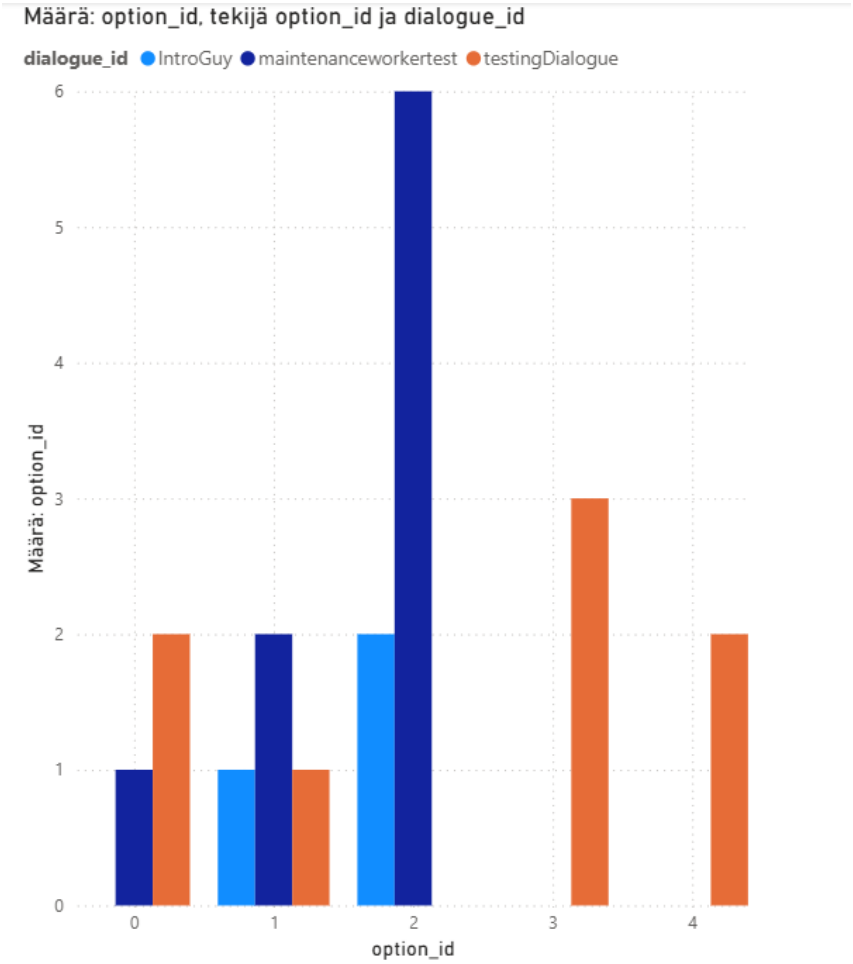


Kuva 46. Datan syöttäminen Microsoft PowerBi:hin.

Tässä esimerkissä analysoidaan, että mikä on ollut suosituin valinta missäkin dialogissa ja miksi näin teoriassa? Tällä tavoin voidaan arvioida, mitkä valinnat ovat suosittuja ja tämä on tärkeää, jotta voitaisiin kartoittaa pelaajien mieltymyksiä ja näin rakentaa parempaa tarinankerrontaa tulevaisuudessa.

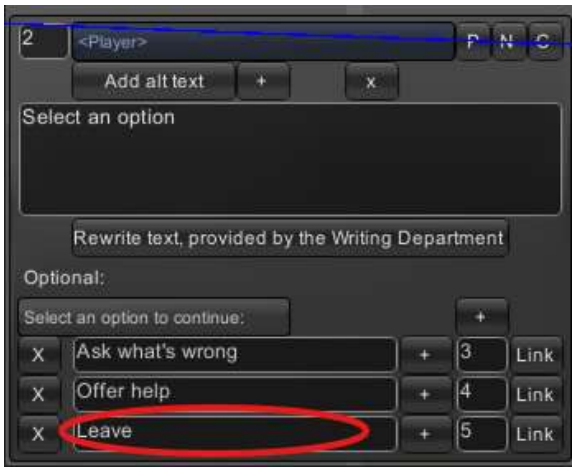
Aluksi valitaan tarvittavat sarakkeet, joilla dataa halutaan visualisoida. Tässä tilanteessa valitaan x-akseliksi option_id ja x-akseliksi option_id:n määrä. X-akseli kuvaa PowerBI:ssa eri valintavaihtoehtoja ja y-akseli kuvaa eri valintojen

määrää. Selitteeksi valitaan dialogue_id, jolloin eri pylväät kuvaavat sitä dialogia, jossa valinta on tehty. Tällä tavoin saadaan aikaan visuaalinen kaavio, jolla voi kuvata valintojen suosiota (Kuva 47).



Kuva 47. Valintojen (option_id) suosio dialogikohtaisesti (dialogue_id), valintojen määrän mukaan (Määrä optio_id).

Tässä kohtaa pitää muistaa, että testatessani analytiikkalokittajaa, valitsin testingDialoguessa valintoja täysin satunnaisesti. Tämän kaavion mukaan pelaajat valitsisivat useimmiten mahdollisimman epäkohteliaan ja loukkaavan vaihtoehdot, mikä ei välttämättä toistu todellisessa tilanteessa. Kiinnostavaa tässä on kuitenkin maintenanceworkertestin tulos, jossa vaihtoehtoa kaksi on valittu selvästi enemmän kuin muita vaihtoehtoja. Kuvassa 48 esitetään kyseisessä dialogissa esiintyvä valinta suoraan DialogueMaster 3000:ssa.



Kuva 48. Maintenanceworkertest-dialogissa esiintyvä valinta.

Tämä voisi tarkoittaa, että useimmat pelaajat ovat valinneet lähteä keskustelusta sen sijaan, että he olisivat auttaneet hahmoa tai kysyneet tältä lisätietoja. Tästä on helppo päätellä, ettei pelaajia kiinnosta mahdollinen avautuva tehtävä, jonka he voisivat pelata läpi. Mahdollisia syitä tälle on se, että tehtävä vaikuttaa niin ikävystyttävältä, ettei sitä edes harkita tai, että se tulee vain väärään aikaan, missä pelaaja haluaa tehdä mieluummin jotain muuta. Diagnostisen analytiikan hengessä pitäisi mennä tehtävän luonteen juurisyihin ja arvioida rehellisesti, mikä tässä on vikana. Sitten preskriptiivisen analytiikan hengessä tulisi tehdä tarvittavat muutokset, joka voi tarkoittaa tehtävän siirtämistä tai muuttamista. Tässä tilanteessa on todennäköisesti testattu dialogin toimivuutta kehittäjien kesken, mutta teoriassa, jos tällainen kuvio esiintyy, on se selvä merkki siitä, ettei pelaaja kiinnosta.

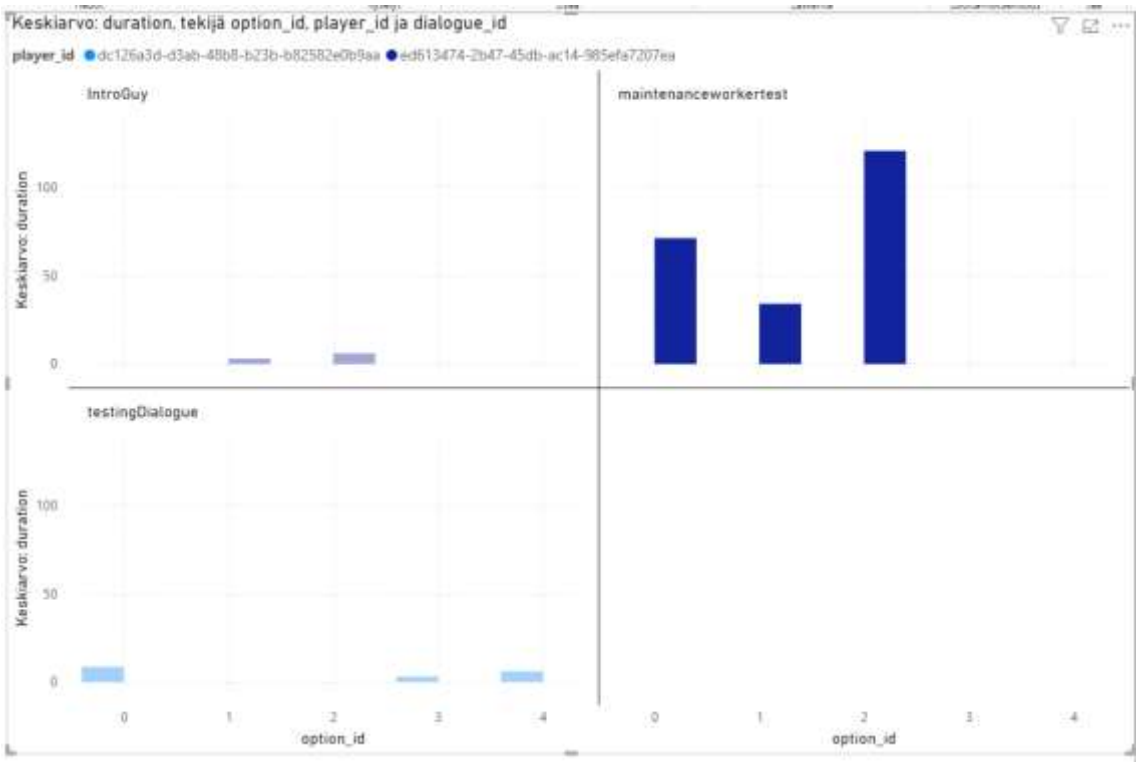
IntroGuy-dialogissa on nähtävissä samanlainen kuvio, koska suurin osa on valinnut lähteä dialogista, eikä kukaan ole kysynyt lisätietoja tehtävästä. Jos pelaaja on valinnut mennä eteenpäin tehtävän kannalta, he ovat halunneet tehdä sen suoraan. Kuvassa 49 esitetään IntroGuy-dialogissa esiintyvä valinta.

Kuva 49. IntroGuy-dialogissa esiintyvä valinta.

10.4 Datan visualisointi ja valintaan käytetty aika

Jos halutaan arvioida valinnan vaikeutta, voidaan mitata aikaa, jonka pelaaja on käyttänyt dialogissa. Voidaan mitata keskimääräistä aikaa, jonka pelaajat ovat kuluttaneet tiettyjen dialogien kohdalla. Tällä tavalla voidaan kartoittaa, miten haastava valinta on, jos siihen käytetään paljon aikaa. Jos aika on keskimääräistä lyhyempi, se voi kertoa siitä, että pelaajat eivät käytä aikaa dialogien lukemiseen ja valitsevat täysin satunnaisesti. Tätä havaintoa tukee myös se, että tietty pelaaja valitsee aina saman valinnan järjestyksen, ensimmäisen, toisen tai kolmannen tapojensa mukaan.

Kuten aikaisemminkin tähän tilanteeseen sopii parhaiten ja x-akseliksi valitaan pelaajan valinta (option_id) ja y-akseliksi keston (duration) keskiarvo. Selitteeksi valitaan pelaajan tunniste (player_id) ja kerrannaisiksi dialogin tunniste (dialogue_id). Tuloksena on diagrammi, joka kuvaa dialogikohtaisesti valintoihin käytettyjen aikojen keskiarvot (Kuva 50).

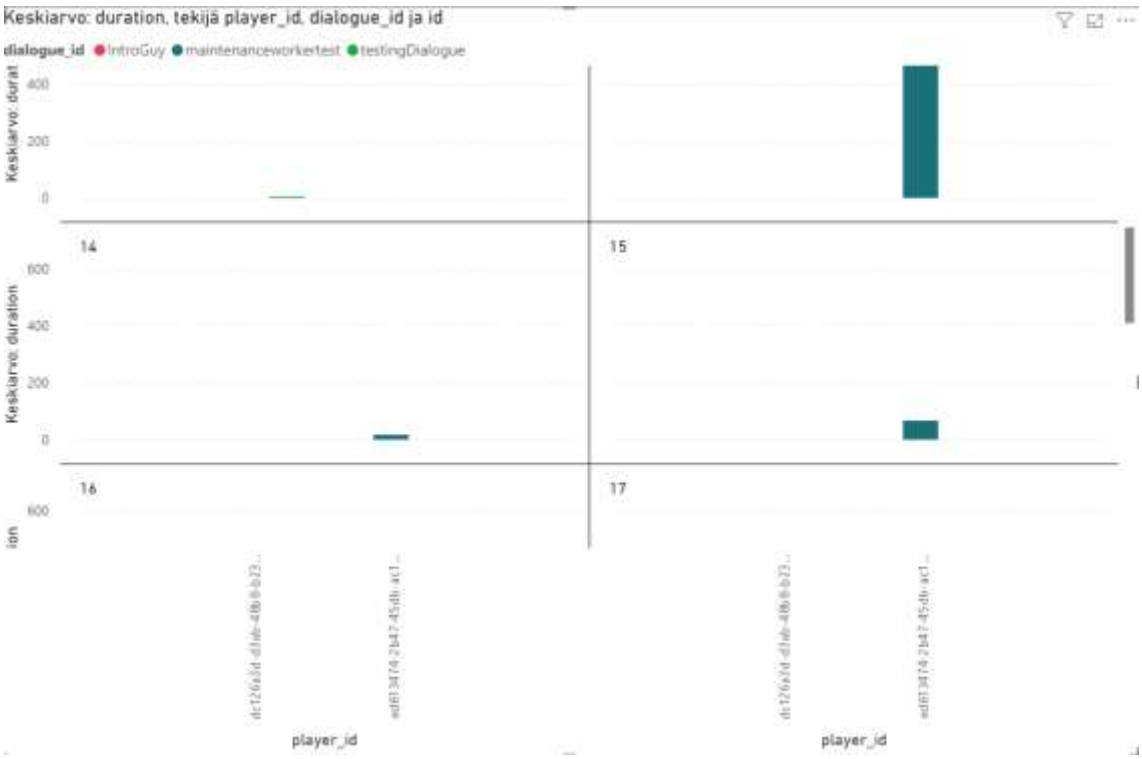


Kuva 50. Valintoihin (option_id) käytettyjen aikojen keskiarvot (Keskiarvot: duration) dialogikohtaisesti.

Pelkkien keskiarvojen lukemisen kautta voidaan havaita, että maintenanceworktest-dialogissa on käytetty eniten aikaa päätöksen tekoon. Tällä hetkellä tästä voidaan päätyä kahteen eri johtopäätökseen, joko kyseinen dialogi on paljon kiinnostavampi ja intensiivisempi kuin muut tai se on paljon vaikeampi, tai molemmat samaan aikaan.

Kuitenkin, jos tarkastelee yksittäisiä valintoja erikseen, voidaan huomata, että yhdessä tietyssä tilanteessa dialogissa vietetty aika on huomattavasti suurempi kuin muut. Tämä ei todennäköisesti riipu valinnan vaikeudesta, vaan siitä, että pelaaja on jättänyt pelin päälle samaan aikaan, kun hän on poistunut koneelta. Tämän vuoksi keskiarvot voivat vääristyä, eikä johtopäätöksiin kannata hypätä heti, vaan dataa kannattaa tarkastella monesta eri näkökulmasta. Toisaalta tässä kyseisessä dialogissa on myös muihin vaihtoehtoihin käytetty huomattavasti enemmän aikaa kuin toisissa dialogeissa, mutta kaikki nämä valinnat ovat samalta pelaajalta. Tämä voi myös yksinkertaisesti kertoa ainoastaan sen, että tämä kyseinen pelaaja käyttää paljon aikaa tehdessään valintaa, tai ei pelaa niin keskittyneesti. Dataa tarvittaisiin paljon enemmän, että sen pohjalta voisi tehdä järkeviä havaintoa ja johtopäätöksiä.

Aikaiseksi saadaan kuitenkin diagrammi, joka kuvaa kaikkien valintojen keston (Kuva 51). Tulokset jatkuvat alaspäin, mutta tarkastellaan vain tuloksia, jotka selvästi erottuvat muista. Tästä syystä kuvassa 51 on esitetty maintenanceworkertest-dialogi.

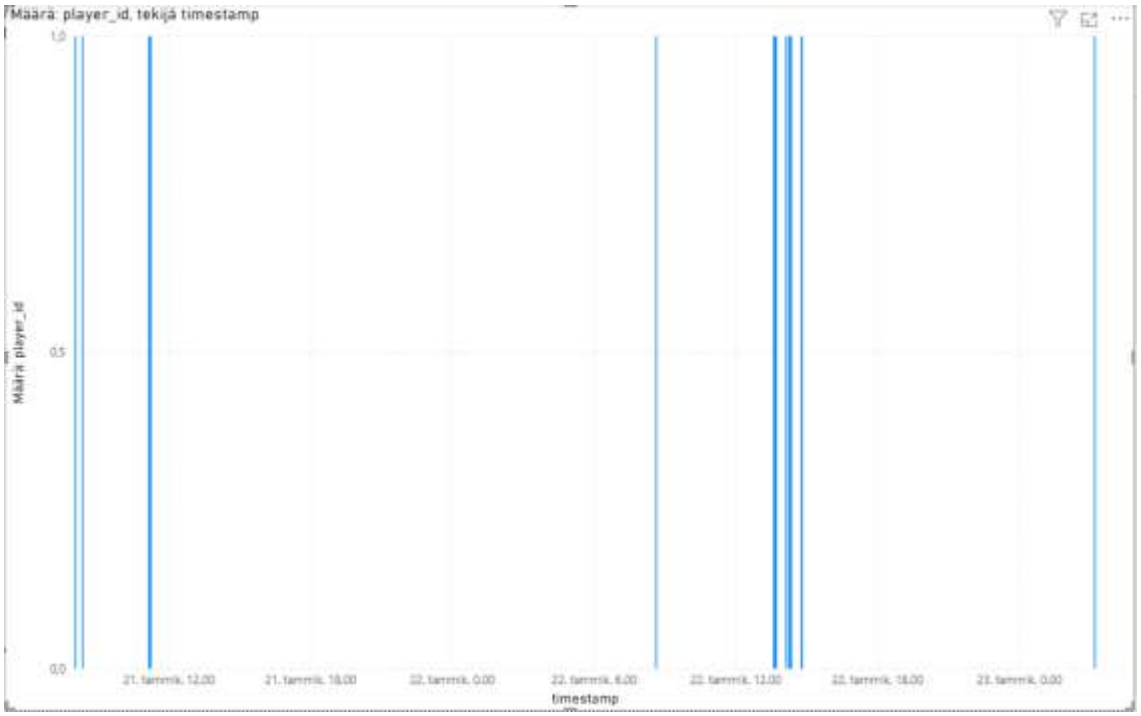


Kuva 51. Yksittäiset valintoihin käytetyt ajat dialogikohtaisesti.

10.5 Datan visualisointi ja aika

Pelaajien valintojen perusteella voidaan analysoida, milloin valinnat on tehty. Tässä on kuitenkin tärkeää huomioida aikavyöhykkeet, sillä tiedot on tallennettu tietokantaan Pohjois-Amerikan itärannikon aikavyöhykkeellä, kuten aiemmin on käsitelty. Joka tapauksessa tämän kaavion avulla voi yrittää hahmottaa mihin aikaan pelaajat usein pelaavat peliä. Tämän avulla tulevaa lisäsisältöä voidaan ajoittaa paremmin, esimerkiksi julkaisemalla päivityksiä ja tapahtumia silloin, kun pelaajat ovat todennäköisesti aktiivisia.

Tämä saadaan aikaiseksi, jos x-akseliksi laittaa aikaleiman (timestamp) ja y-akseliksi laittaa pelaajan tunnisteen (player_id). Määränä samat muuttujat kuin x-akselissa ja y-akselissa. Kuvassa 51 esitetään diagrammi, joka näyttää mihin aikoihin valintoja on tehty pelaajan tunnisteen (player_id) mukaan.



Kuva 52. Yksittäiset valintoihin käytetyt ajat pelaajan tunnisteen (player_id) mukaan.

Suurin osa pelaajista näyttäisi olevan aktiivisia noin kahdentoista aikaan suomen ajassa, joten mahdolliset päivitykset ja tapahtumat kannattaisi julkaista silloin.

11 Yhteenveto

Peliyritykset myyvät kokemusta ja kyseinen kokemus koostuu pääosin tarinasta ja toiminnoista. Hyvät tarinat luovat ikimuistoisia kokemuksia ja dialogi on olennainen osa sitä. Parempien kokemusten rakentamisen vuoksi, on otettava selvää heidän mieltymyksistään ja tämä onnistuu parhaiten keräämällä pelaajista tietoja. Työkalu tähän tarkoitukseen on analytiikkalokittaja, joka ei ole yksittäinen sovellus, vaan useiden eri komponenttien muodostama kokonaisuus.

Tämä kokonaisuus rakennettiin alusta alkaen aloittamalla testidialogista ja lopettaen itse analytiikkalokittajan koodiin. Tämän jälkeen data analysoitiin, ja tulkittiin monilla eri tavoilla. Näiden tulkintojen tarkoituksena oli, että peliyritysten olisi mahdollista luoda parempia ja merkityksellisempiä pelikokemuksia.

Erityisesti kiinnitin huomiota siihen, että pelaajakäyttäytymistä koskevien datatulkintojen avulla pyrittiin arvioimaan, onko peli kiinnostava vai ei. Kiinnostavuus on jokaisen tarinan merkityksellisin komponentti ja tarinoihin liittyy monia eri piirteitä, jotka tekevät niistä kiinnostavia, näistä esimerkkejä ovat samaistuttavat hahmot ja hyvin rakennettu dialogi. Kerätyn datan avulla on helppoa arvioida, että onko pelissä ylipäätään sellaisia kohteita, joita tarvitsee kehittää kriittisesti. Nämä kohteet selviävät esimerkiksi sellaisista tilanteista, joissa pelaajat valitsevat mieluummin vaihtoehdot, jotka eivät johda tarinan syvällisempään ymmärtämiseen tai edes tehtävän suorittamiseen. Kuitenkin analytiikan avulla on vaikea arvioida, että millä tavalla tarinaa voisi kehittää paremmaksi. Pelaajakäyttäytymistä seuraamalla voidaan löytää kehityskohteita, muttei suoraan ratkaisua niille.

Pelaajakäyttäytymistä olisi voinut seurata muualtakin kuin dialogista, esimerkiksi tehtävissä olevista valinnoista. Tämä muistuttaisi hyvin paljon dialogin valintapuussa tapahtuvia valintoja, mutta valinnat otettaisiin suoraan toiminnasta tehtävän suorittamisen yhteydessä. Analytiikkalokittaja voisi tässä tilanteessa olla hyvin samanlainen, mutta se metodi, joka tunnistaa valinnan dialogista, pitäisi vaihtaa metodiin, joka tunnistaa valinnan suoraan pelaajan toiminnasta. Myös kerättävät tietueet olisivat erilaisia, eikä välttämättä sopisi samaan tietokantaa, kuin siinä, että valinnat kerätään dialogeista. Tällä voisi myös kartoittaa tehtävien vaikeutta, mittaamalla tehtävässä suoritettua aikaa. Opinnäytetyössä kehitetty analytiikkalokittaja ei sovi muuhun tarkoitukseen sellaisenaan, kuin valintojen mittaamiseen dialogissa. Tämä analytiikkalokittaja on kuitenkin toteutettu rakenteeltaan yksinkertaiseksi, mikä mahdollistaa sen helpon muokattavuuden ja uudelleenkäytettävyyden. Näitä periaatteita noudattaen työ on helposti toistettavissa ja muokattavissa myös muihin tarkoituksiin. Erilaisilla, mutta samankaltaisilla analytiikkalokittajilla voisi myös kartoittaa sitä, kuinka usein pelaaja kuolee tai viettää aikaa erilaisissa valikoissa. Tällä voisi myös mitata pelin kiinnostavuutta tai vaikeutta.

Moninpeleissä olisi myös teoriassa mahdollista seurata myös, miten pelaajat toimivat yhdessä muiden pelaajien kanssa. Esimerkiksi voidaan seurata, miten pelaajat kommunikoivat, kilpailevat tai tekevät yhteistyötä keskenään. Monissa monipeleissä on mukana valikoita, jonka kautta he voivat vuoro vaikuttaa toistensa kanssa erilaisten hymiöiden avulla. Tämän kaltaista analytiikkalokittajaa

voisi teoriassa käyttää myös tällaisessa pelissä. Tässä tilanteessa havaintojen luonne ei kuitenkaan välttämättä kerro tarinan kehityksestä mitään, koska harva moninpeli keskittyy vahvan narratiivin ympärille. Se voisi ainakin kertoa, että mitkä hymiöt ovat pelissä suosittuja ja missä tilanteissa pelaajat haluavat niitä käyttää.

Tässä opinnäytetyössä keskityttiin kuitenkin enemmän valintoihin dialogissa, koska valintojen seuraaminen dialogissa on suhteellisen helppoa ja suoraviivaista. Kiinnostavuutta voi yrittää arvioida mittaamalla pelaajien valintojen jakautumia tai kuluvaa aikaa, jonka pelaajat ovat käyttäneet valinnassa. On myös osoitettu, että dialogista voi seurata muutakin kuin valintojen mielekkyyteen liittyviä asioita, kuten sitä, minä aikoina pelaajat yleisesti pelaajat peliä. Tämän avulla voidaan ajoittaa mahdollisia tulevia päivityksiä tai tapahtumia.

Tämän opinnäytetyön aikana tehtyjen havaintojen perusteella voidaan kuitenkin todeta, että pelkästään kerätyllä datalla ei voi tehdä mitään yksinään. Data tarvitsee tulkintaa, jotta sen pohjalta voidaan tehdä merkityksellisiä havaintoja, jotka johtavat toimintaan. Data analytiikka on taiteenlaji, joka tarvitsee pätevän data-analyytikon tulkitsemaan kerättyä dataa. Periaatteessa dataa voisi tulkita myös tekoälyn avulla, mutta tekoälyn tuottamat analyysit eivät aina vastaa pätevän analyytikon tulkinnan laatuun, koska tekoäly perustaa johtopäätöksensä sille syötetyn aineiston perusteella. Myös inhimillinen tulkinta on altis virheille, eikä täysin luotettavaa. Vaikka ei voi olettaa, että tulkinnat olisivat erehtymättömän luotettavia, kerätty data kuitenkin auttaa tukemaan päätöksen tekoa, mutta päätöksiä ei kannata tehdä sokeasti.

Kuitenkin voidaan todeta, että analytiikkalokittajan avulla saadaan konkreettisempaa näyttöä pelaajan mieltymyksistä sen sijaan, että niitä arvioitaisiin pelkästään intuitioon perustuen. Kerätty ja analysoitu data tukee parhaimmillaan päätöksentekoa, ohjaa tarinallista suunnittelua ja auttaa kehittämään pelikokemusta entistä vaikuttavammaksi. Vaikka analytiikka ei yksin ratkaise kaikkia haasteita, se kuitenkin tarjoaa vankan pohjan sille.

Lähteet

Andersen, G. 2024. The Importance of Storytelling in Video Game Design: Engaging players through narratives. https://moldstud.com/articles/p-the-importance-of-storytelling-in-video-game-design-engaging-players-through-narratives?utm_source=chatgpt.com. MoldStud.9.2.2026

Argentics. 2023. Leveraging Data Analytics for Game Development. Medium. 2023. https://medium.com/@argentics/leveraging-data-analytics-for-game-development-afb5b0f46c6b.24.10.2025.

Arm. 2025. Gaming Engines. https://www.arm.com/glossary/gaming-engines. 13.1.2026.

Assassin's Creed Odyssey.2018. Ubisoft Quebec.

Brown, D. 2023. Two Pillars of Storytelling.https://darlingaxe.com/blogs/news/two-pillars-of-storytelling?srsltid=AfmBOoqCgIYou7pKhDQ9IAiixDxbitcEH1upED7izGW1APS60G6YUEHC.7.10.2025.

Çoğalan, A. 2023. Telemetry In-Game Industries. https://medium.com/@anilcogalan/telemetry-in-game-industries-2540c3cf98d9.6.1.2026

Cote, C. 2021.4 Types of Data Analytics to Improve Decision-Making. HBS Online.2021. Blog. 2021.https://online.hbs.edu/blog/post/types-of-data-analysis.22.10.2025.

Cyberpunk 2077.2020. CD Projekt RED.

Kallio, M. 2026. Ohjelmistokehityksen ulkoistaminen – kumppanin valinta pk-yritykselle. https://datastorm.fi/oppaat/ohjelmistokehitys/ohjelmistokehityksen-ulkoistaminen/?utm_source=chatgpt.com.6.1.2026.

Deus Ex: Mankind Divided.2016. Eidos-Montréal.

Dey, K. 2024. Creating Immersive Storytelling in Games: Techniques and Tools.https://www.redappletech.com/blog/creating-immersive-storytelling-in-games-techniques-and-tools/.7.10.2025.

Dishonored.2012.Arkane Studios.

Dishonored 2.2016.Arkane Lyon.

Dugdiev, A. 2025. App Development Cost (2025). https://www.businessofapps.com/app-developers/research/app-development-cost/.6.1.2026.

Duigou, B.2025. A reflection on the concept of choice in video games with Baldur’s Gate 3. https://www.pointnthink.fr/en/une-reflexion-sur-la-notion-de-choix-dans-les-jeux-video-en-compagnie-de-baldurs-gate-3/.16.20.2025.

Ediiie.2026. https://www.ediiie.com/game-analytics-implementation/.8.1.2026.

Fallout 3.2008.Bethesda Game Studios.

Fallout 4.2015.Bethesda Game Studios.

Fallout 76.2018.Bethesda Game Studios.

GameAnalytics. 2026. Pricing.https://www.gameanalytics.com/pricing. 8.1.2026

GameFromScratch. 2024. Game Engine Popularity in 2024. https://gamefromscratch.com/game-engine-popularity-in-2024/.13.1.2026.

GeeksForGeeks. 2024. Components of Table in Database. https://www.geeksforgeeks.org/sql/components-of-table-in-database/.21.1.2026

Goodwin, M. 2025. What is an API?. IBM Think. https://www.ibm.com/think/topics/api.23.1.2026.

Google Gemini 3. 2026a. Mikä on analytiikkslokittaja. https://gemini.google.com/app/b4225a296667aa4d.14.1.2026.

Google Gemini 3. 2026b. mikä on ohjelmointiympäristö. https://gemini.google.com/app/b4225a296667aa4d.14.1.2026.
Hagenlocher, P.2023.Video Game Dialogues and Graph Theory. 09.07.2023. Hallöchen!. https://philipphagenlocher.de/post/video-game-dialogues-and-graph-theory/.15.10.2025.
Hughes, K. 2022. RPGs and their Dialogue Systems.14.12.2022. https://medium.com/@kaptainviciorious/rpgs-and-their-dialogue-systems-73307caa2b81.15.10.2025.
Juegoadmin. 2024. Game Storytelling Secrets: Creating Engaging Narratives.https://www.juegostudio.com/blog/mastering-game-storytelling-crafting-compelling-narratives.7.10.2025.
Laaksonen, A. 2011. PHP-ohjelmointi: Osa 1 – Johdanto. Ohjelmointiputka. https://www.ohjelmointiputka.net/oppaat/opas.php?tunnus=php_0.22.1.2026.
Macrosoft. 2024. Microsoft Visual Studio: mikä se on ja mihin sitä tarvitaan. https://macrosoft.store/fi/blog/post/29-mihin-microsoft-visual-studioak%C3%A4ytet%C3%A4%C3%A4n.20.1.2026.
Microsoft. 2026. Perustiedot tietokannasta https://support.microsoft.com/fi-fi/topic/perustiedot-tietokannasta-a849ac16-07c7-4a31-9948-3c8c94a7c204.13.1.2026.
Microsoft Learn. 2025. What is Power BI. https://learn.microsoft.com/en-us/power-bi/fundamentals/power-bi-overview.13.1.2026
Mizina, A.2025. What is Data Analytics in the Gaming Industry? A Comprehensive Overview. TrapPlan Blog. 2021.https://www.trapplan.com/blog/what-is-data-analytics-in-the-gaming-industry-a-comprehensive-overview.23.10.2025.
National Geographic Society. 2025.Storytelling. https://education.nationalgeographic.org/resource/storytelling-x/.7.10.2025.
OlipaOS Documentation. 2025. DialogueMaster.https://docs.google.com/document/d/1fEoPw8f0h70rrxWc6eof-KXlPKZhl_EIipcNMj8oS1A/edit?tab=t.hvenseyqfhql#heading=h.d3g4l1th1j64.13.1.2026.
Olipa Games. 2025. Hostile Docking. https://docs.google.com/document/d/1YBfmC9Z5asZv7Tjfh1i0CYdu1YxvxuYkJdCd6JSEajQ/edit?tab=t.2ghff8eh2jhc#heading=h.qmothzb7ujzy.13.1.2026.
Plexicus. 2025. Mikä on SQL-injektio (SQLi)? https://www.plexicus.ai/fi/glossary/sql-injection/.22.1.2026.
Plotnek, J. 2024. The 8 Best Game Analytics Solutions in 2025. https://keewano.com/blog/best-game-analytics-solutions/.7.10.2025
Sanastokeskus, TEPA-Termipankki. 2013.Tietotekniikan termitalkoot. https://termipankki.fi/tepa/fi/haku/videopeli.7.10.2025.
Red Dead Redemption 2.2018.Rockstar Games.
Ronkainen, N. 2025. Pelialan palkat suomalaisille: mitä eri rooleissa ansaitaan?. https://keskipalkka.com/pelialan-palkat-suomalaisille-mita-eri-rooleissa-ansaitaan/.6.1.2026.
The God of War. 2005. SCEA.
The Last of Us.2013. Naughty Dog.
The Witcher 3: WildHunt.2015.CD Projekt RED, CD Projekt.
Siegel, J. & Szafron, D. 2009. Dialogue patterns—A visual language for dynamic dialogue. Journal of Visual Languages & Computing, 20(3), 196–220.

https://www.researchgate.net/publication/222422920_Dialogue_patterns-A_visual_language_for_dynamic_dialogue.16.10.2025.
Slyter, C.2025.A Historian’s 10 Favorite Video Games Set in the Past. https://medium.com/@coreyslyter/a-historians-10-favorite-video-games-set-in-the-past-b56af1b48644.7.10.2025.
Solod, T. 2024. The importance of story and character development in games. https://pinglestudio.com/blog/the-importance-of-story-and-character-development-in-games.7.10.2025.
Sons of the Forest.2024.Endnight Games
Stegner, B. 2021.What Is a Visual Novel Video Game? https://www.makeuseof.com/what-is-visual-novel-video-game/.5.10.2025.
Stevens, E. 2023. What is Data Analytics? A Complete Guide for Beginners. 1. What is data analytics?. 2023.https://career-foundry.com/en/blog/data-analytics/what-is-data-analytics/.23.10.2025.22.10.2025.
Suter, B. Bauer, R. Kocher, M.2021.Narrative Mechanics: Narrative Patterns in Video Games.Research-Gate.7.10.2025.
Tieteen termipankki.2015. Kirjallisuudentutkimus: dialogi. https://tieteentermipankki.fi/wiki/Kirjallisuudentutkimus:dialogi.17.10.2025.
Tieturi. 2026. Ohjelmointikielet. https://www.tieturi.fi/koulutusala/ohjelmistokehitys/ohjelmointikielet/.13.1.2026.
Ulanoff, J. 2024. What Makes Choices Matter In Video Games? https://www.cbr.com/what-makes-choices-matter-video-games/.16.10.2025.
Unity Documentation. 2026. Pricing. https://docs.unity.com/ugs/en-us/manual/analytics/manual/billing.7.1.2026
Vitez, M. 2025.Algebraic Dialogue Trees.2025.vitez.me.https://vitez.me/dialogue-trees.15.10.2025.
Yatin, S. 2025. In-House vs Outsourced Data Analytics Services: Which Is Right for You? https://www.hashstudioz.com/blog/in-house-vs-outsourced-data-analytics-services-which-is-right-for-you/.8.1.2026.