OULUN AMMATTIKORKEAKOULU

Joni Anttila

PELIDEMO UNITY-PELIMOOTTORILLA YHDISTÄEN 2D- JA 3D-ELEMENTTEJÄ

PELIDEMO UNITY-PELIMOOTTORILLA YHDISTÄEN 2D- JA 3D-ELEMENTTEJÄ

Joni Anttila
Opinnäytetyö
Kevät 2024
Tietojenkäsittely
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn tradenomi

Tekijä: Joni Anttila
Opinnäytetyön nimi: Pelidemo Unity-pelimoottorilla yhdistäen 2D- ja 3D -elementtejä
Työn ohjaaja: Matti Viitala
Työn valmistumislukukausi ja -vuosi: kevät 2024 Sivumäärä: 35

Opinnäytetyössä käsiteltiin pelidemon kehitystä Unity-pelimoottorissa. Työn tarkoituksena oli oppia uutta kehittäessä demoa yhdistämällä 2D- ja 3D-elementtejä. Käsittelyssä olivat vuoropohjainen taistelumekaniikka, kokonaisvaltaisen inventaariosysteemin implementointi, alustava keskustelusysteemi ja kolmiulotteisen maailman luominen. Pelisuunnittelun ja koodauksen ohella hyödynnettiin omaa graafista tuntemusta tekstuurien sekä vihollisten luomiseen tavoitellen kokonaisvaltaista pelinteon prosessin harjoittelua.

Demon tekeminen kuvattiin prosessin suunnitteluvaiheesta lopputulokseen. Tavoitteet määriteltiin suunnitteluosuudessa ja saavutettua lopputulosta analysoitiin eri kompromissein ja huomioien suunnitelmien muutokset. Samalla paneuduttiin erilaisiin opinnäytetyössä käytettyihin työskentelymenetelmiin ja teknologioihin. Lopulta valittiin oikeat teknologiaratkaisut samalla pitäen mielessä tekemisen rajoitettu aikataulu, resurssimäärä ja omat graafisen osaamisen taidot.

Opinnäytetyöraportti toteutettiin perinteisen opinnäytetyön rakenteen mukaan. Työssä tietoperusta käsitellään ensin, jonka jälkeen tulevat työn toiminnallinen osuus eli pelidemonkehityksen kuvaaminen, demon suunnittelu, demon toteuttaminen ja pohdinta. Opinnäytetyön tietoperustan lähteinä hyödynnettiin alan verkkolähteitä, joihin lukeutuivat verkossa julkaistut artikkelit, ohjedokumentaatiot ja videot.

Demoon saatiin implementoitua suunnitellun mukaisesti halutut perusominaisuudet. Ainoastaan muutamaa ominaisuutta lukuun ottamatta demon runko on lähes valmis ja seuraava askel projektissa olisi olemassa olevien ominaisuuksien yhteennitominen entistä paremmin. Demoa voi tulevaisuudessa kehittää entisestään ja mahdollisesti siirtyä demovaiheesta pelattavaan kokonaisuuteen.

Asiasanat: peliala, pelinkehitys, peligrafiikka

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Bachelor of Business Information Technology

---

Author: Joni Anttila
Title of thesis: Game demo with the Unity game engine combining 2D and 3D elements
Supervisor: Matti Viitala
Term and year when the thesis was submitted: Spring 2024
Number of pages: 35

The thesis involved the development of a game demo in the Unity game engine. The purpose of the thesis was to learn something new while developing a demo by combining 2D and 3D elements. The turn-based combat mechanics, the implementation of a comprehensive inventory system, a preliminary conversation system and the creation of a three-dimensional world were the main topics. Along with game design and coding, own graphical knowledge to create textures and enemies was utilized, aiming for comprehensive practice of the game making process.

Making the demo was described from the planning stage of the process to the end result. The goals were defined in the planning part and the final result achieved was analyzed with different compromises and taking into account the changes in the plans. At the same time, attention was paid to the different working methods and technologies used in the thesis. Ultimately the right technological solutions were finally chosen, while keeping in mind the limited schedule, amount of resources and own graphical skills.

The thesis report was implemented according to the structure of a traditional thesis. In the work, the database is discussed first, followed by the functional part of the work, i.e. describing the development of the game demo, planning the demo, implementing the demo and reflection. The sources of the database of the thesis were the online sources of the field, which included articles published online, instructional documentation and videos.

The desired basic features could be implemented in the demo as planned. With the exception of only a few features, the body of the demo is almost complete and the next step in the project would be to stitch together the existing features even better. The demo can be further developed in the future and possibly move from the demo phase to a playable gaming experience.

Keywords: gaming industry, game development, game graphics

SISÄLLYS

1 JOHDANTO

Tässä opinnäytetyössä käsitellään 3D pelidemon kehitystä Unity-pelimoottorissa. Projektin tarkoituksena on oppia uutta kehittäessä dungeoncrawler-genren peliä yhdistäen 2D- ja 3D- elementtejä. Dungeoncrawler -peleissä kuljetaan sokkelomaisissa suljetuissa peliympäristöissä, joissa taistellaan vihollisia vastaan, kerätään esineitä ja valuuttaa, ratkotaan erilaisia pulmia sekä avataan polkuja (Stuart 2021). Pelidemossa käsittelyssä ovat vuorotaistelu-, inventaario- ja ryöstösaalis (loot)-systeemien implementointi ja kolmiulotteisen maailman luonti retrovivahteisella grafiikalla. Pelisuunnittelun ja koodauksen ohella hyödynnetään omaa graafista tuntemusta ja piirustustaitoa tekstuurien sekä vihollisten luomiseen tavoitellen kokonaisvaltaista pelinteon prosessin harjoittelua.

Pelin tekeminen kuvataan prosessin suunnitteluvaiheesta lopputulokseen. Tavoitteet määritellään suunnitteluosuudessa ja saavutettua lopputulosta analysoidaan eri kompromissein ja suunnitelmien muutoksineen. Lopuksi arvioidaan peliprojektin jatkotoimenpiteitä mahdollista pelijulkaisua varten.

Samalla paneudutaan erilaisiin opinnäytetyössä käytettyihin työskentelymenetelmiin ja teknologioihin ja käydään läpi eri vaihtoehtojen arvot. Analyysin kautta lopulta valitaan oikeat teknologiaratkaisut samalla pitäen mielessä tekemisen rajoitettu aikataulu, resurssimäärä ja omat graafisen osaamisen taidot.

Työssä tarkastellaan Unity-pelimoottorin perusteiden lisäksi niitä ominaisuuksia, jotka ovat pelidemossa olennaisia. Lisäksi käsitellään hyödynnettyjä grafiikkateknologioita, jotka ovat keskeisiä pelin visuaalisen ilmeen kannalta. Ensisijaisena kuvitustyökaluna toimii PaintTool SAI 2, ja apuna animaatioprosessissa on käytössä ilmainen kuvankäsittely- ja animaatio-ohjelma Krita.

Väliaikaisena ratkaisuna hyödynnetään Unityn Asset Store -kaupan ilmaisia graafisia paketteja, jotta saadaan konkreettista visuaalisuutta peliin, kunnes omia ratkaisuja saadaan niiden tilalle peliprojektin edetessä. Lisäksi hyödynnetään valmista koodia nopeuttaakseen tekoprosessia.

Opinnäytetyöraportti noudattaa perinteisen opinnäytetyön rakennetta eli tietoperusta käsitellään ensin, minkä jälkeen tulevat työn toiminnallinen osuus eli pelikehityksen kuvaaminen, johtopäätökset ja pohdinta (Oulun ammattikorkeakoulu 2024). Opinnäytetyön tietoperustaan kuuluvat verkkolähteet, joihin lukeutuvat verkossa julkaistut artikkelit, dokumentaatiot ja videot.

2 PELIKEHITYSTYÖKALUT

Pelinkehitys tapahtuu kehitystyökalujen avulla, joihin lukeutuvat pelimoottorit, ohjelmointikielet, grafiikka- ja ääniohjelmistot sekä testaustyökalut. Pelimoottori ohjelmistoalustana tarjoaa pelien ydintoiminnallisuudet ja ominaisuudet pelien kehittämiseen ja pelaamiseen. Se käsittelee eri tehtäviä, kuten pelin renderöintiä ja fysiikkaa. (LinkedIn 2024.) Pelimoottorit ovat olennainen osa nykyaikaista pelikehitystä, sillä ne tehostavat kehitysprosessia monin tavoin. Ne muun muassa mahdollistavat nopean prototyyppien tekemisen, jossa ideoita ja käsitteitä voidaan testata välittömästi. Tämä auttaa kehittäjiä hahmottamaan pelin mekaniikat ja visiot nopeasti ja tehokkaasti. (Bajaj 2023.) Markkinoiden käytetyimpiä pelimoottoreita ovat esimerkiksi Unity ja Unreal Engine (LinkedIn 2024).

Ohjelmointikieliä tarvitaan pelimoottorin komentojen suorittamiseksi. Riippuen pelimoottorista voi hyödyntää yhtä tai useampaa ohjelmointikieltä pelin logiikan, käyttöliittymän ja mukautettujen ominaisuuksien luomiseksi. Yleisimpiä ohjelmointikieliä pelikehityksessä ovat C#, C++, Java, Python sekä JavaScript. (LinkedIn 2024.) Ohjelmointikielissä voi tulla vastaan erilaisia haasteita. Pelien ohjelmointi voi olla monimutkaista ja aikaa vievää. Samalla tulisi pitää huoli suorituskyvyn optimoinnista eli varmistaa, että pelit toimivat sujuvasti eri laitteistoilla. Yksi keino välttää ongelmatilanteita on esimerkiksi valmiiden koodikirjastojen ja kehitysratkaisujen käyttäminen. (Bajaj 2023.)

Grafiikkaohjelmistojen avulla luodaan ja muokataan kuvia, animaatioita, malleja ja tekstuureja. Niitä käytetään myös hahmojen, ympäristöjen, käyttöliittymän elementtien ja tehosteiden suunnitteluun. Yleisimpiä grafiikkatyökaluja pelikehityksessä ovat Photoshop, Blender ja Maya. Luodut tekstuurit sekä peliobjektit tulee optimoida pitäen silmällä pelin suorituskykyä ja laatua. Myös tiedostomuodon on oltava yhteensopiva suhteessa pelimoottoriin. (LinkedIn 2024.)

Audio-ohjelmiston avulla luodaan ja muokataan ääniä ja musiikkia peleihin. Ohjelmistoa voi käyttää äänitetiedostojen tallentamiseen, miksaamiseen, muokkaamiseen ja viemiseen eri tiedostomuodoissa ja eri laatuisena. Tavanomaisimpia pelinkehityksessä käytössä olevia ääniohjelmistoja ovat muun muassa Audacity ja FL Studio. (LinkedIn 2024.)

Testaustyökalut auttavat koodivirheiden löytämisessä ja korjaamisessa (LinkedIn 2024). Vianetsinnässä koodausvirheiden tunnistaminen ja korjaaminen voi olla pikkutarkka prosessi. Testaustyökalut ovat myös oiva keino pelin suorituskyvyn analysointiin ja optimointiin (Bajaj 2023).

Testaustyökaluilla voidaan tarkistaa koodia, pelattavuutta, grafiikkaa, ääntä sekä yhteensopivuutta eri alustojen ja laitteiden välillä. Pelikehittäjien käyttämiä testausvälineitä ovat esimerkiksi Visual Studio, Xcode, Android Studio, Unity Test Runner ja Unreal Engine Debugger. Pelien testauksessa hyödynnetään lisäksi käyttäjien analytiikkaa ja palautetta. (LinkedIn 2024.) Alla käydään läpi tarkemmin pelidemossa käytettyjä työkaluja lukuun ottamatta ääni- ja testaustyökaluja.

2.1 Ohjelmointi Unityn kehitysympäristössä

Unity on ollut alalla varteenotettava pelimoottori jo 20 vuoden ajan (Drake 2023). Unity on tunnettu saavutettavuudestaan ja laajasta alustatuestaan, ja juuri siksi se on monien indie-kehittäjien ja opiskelijoiden valinta. Se tukee yli 25 alustaa, mukaan lukien Windowsia, MacOSia, Androidia ja iOSia. Vuoteen 2021 mennessä esimerkiksi yli 60 prosenttia kaikista AR/VR-sisällöistä on tehty Unityllä, mikä vahvistaa sen kuvaa saavutettavuudesta ja monipuolisuudesta. (Aladdin 2023.)

Unityn reaaliaikainen 3D-kehitysmoottori mahdollistaa pelinkehityksen konseptivaiheesta suunnitteluvaiheeseen ja lopulta itse pelin kehityksen ytimeen (Unity 2024a). Toisaalta Unityä on kritisoitu sen graafisista kyvyistä erityisesti verrattuna Epic Games -yrityksen Unreal Engine -pelimoottoriin. High Definition Render Pipeline (HDRP) -järjestelmän käyttöönoton myötä Unity on ottanut askeleita kohti tämän eron kaventamista. (Aladdin 2023).

Unityllä kehitettyjä huomattavia pelejä ovat muun muassa Ori and the Blind Forest, Hearthstone ja Pokémon Go. Tulevaisuuteen katsottaessa Unityn Data-Oriented Tech Stack (DOTS) ja sen keskittyminen pilvipelaamiseen takaa lupaavan tulevaisuuden. (Aladdin 2023). Unityn monialustatuki varmistaa, että kaiken taitotason kehittäjät voivat käyttää kattavia työkaluja ja resursseja tavoittaakseen suosituimmat työpöytäjärjestelmät (Unity 2024b).

Unity Asset Store on kokoelma ilmaisia ja kaupallisia resursseja, jotka ovat luoneet sekä Unity että sen käyttäjät. Resursseja on saatavilla laajasti, ja ne kattavat kaikkea tekstuurien, mallien ja animaatioiden osista kokonaisiin projektiesimerkkeihin, opetusohjelmiin ja laajennusresursseihin.

Unity Asset Store -kaupassa tarjolla oleva sisältö auttaa parantamaan projektia, peliä tai sovellusta sekä vähentämään tarvittavaa työmäärää työkalujen tai mallien luomiseen alusta asti. Assetit ostetaan kaupan verkkosivustolta, ja ne voidaan lisätä projekteihin paketinhallinnan avulla, joka löytyy editorin sisältä. (Unity Support 2024.)

Unityllä tehdyt pelit koostuvat peliobjekteista (gameobject), joihin liitetyt koodi ja komponentit ohjaavat niiden käytöstä ja vuorovaikutusta keskenään luoden näin pelin pelattavuuden ja pelikokemuksen. Peliobjektit ovat Unityn perusobjekteja, jotka edustavat pelin hahmoja, esineitä ja maisemaa. Ne eivät saavuta paljoa itsessään, mutta toimivat komponenttien alustana, jotka toteuttavat todellisen toiminnallisuuden (Unity Documentation 2024a). Koodaaminen Unityssä eroaa tavallisesta ohjelmoinnista siinä, että käyttäjän ei tarvitse itse tehdä koodia, joka suorittaa sovelluksen, vaan Unity tekee sen puolesta. Käyttäjä sen sijaan keskittyy ohjelmoinnissa pelikokemukseen. (Unity 2024c.)

Unity lukee kaiken datan, kuten valot, mesh-malliverkostot ja käytökset, luuppina, ja prosessoi nämä kaikki tiedot kehittäjän puolesta jokaisen piirretyn kuvakehyksen (frame) yhteydessä. Kuvakehyksellä tarkoitetaan käyttölaitteen näytölle piirrettyä kuvaa, joka päivittyy ruudunpäivityksen yhteydessä joka sekunti ruudunpäivitysnopeuden mukaan. Unity näin suorittaa koodia yksittäisten kuvakehysten päivitysten yhteydessä toinen toisensa jälkeen niin nopeasti kuin mahdollista. (Unity 2024c.)

Unityssä käytetty kieli on nimeltään C# (C-sharp). Kieli, jota Unity käyttää, on oliopohjainen komentosarjakieli. Kuten millä tahansa kielellä, skriptikielet sisältävät syntaksin, ja sen tärkeimmät osat ovat muuttujat, funktiot ja luokat. (Unity 2024c.)

Unityssä koodi alkaa asettamalla tarvittavat muuttujat yläriveille. Määritetyt muuttujat näkyvät etuosan avainsanalla julkinen (public) tai yksityinen (private), jonka jälkeen tulevat muuttujan tyyppi ja nimi. Muuttujia määriteltäessä on useita näkyvyystyyppejä, mutta kaksi tärkeintä ovat julkinen ja yksityinen. (Unity 2024c.)

Kun koodin liittää peliobjektiin, näkee sen Unityn inspector -näkymässä komponenttina. Julkiseksi määritetty muuttuja on editorissa nähtävissä, ja sen arvoa voi muokata inspector-näkymässä.

Yksityistä muuttujaa ei inspector-näkymässä näe, vaan kyseistä näkyvyystyypin muuttujaa voi käyttää vain siinä tietyssä C# koodissa. Julkiset muuttujat ovat lisäksi vapaasti käytettävissä muille koodeille ja luokille. Näkyvyystyypin valitseminen siis riippuu tilanteesta. (Unity 2024c.)

Yksityiset muuttujat tekevät koodin ulkoasusta siistimpää, koska niiden arvoja voi muuttaa vain kyseisen luokan sisällä. Tämä tekee virheidenkorjauksesta ja koodin ylläpidosta helpompaa. (Unity 2024c.)

Luokat käsittelevät muuttujia funktioiden avulla. Luokat ovat muuttujien ja funktioiden kokoelmia. Kuvassa 1 on havainnollistettu erilaisten muuttujien ja funktioiden esimerkkejä. Unityssä on useita elinkaarifunktioita, jotka suoritetaan automaattisesti. Esimerkiksi Awake -funktio suoritetaan vain kerran, kun peliobjekti, jossa funktio ilmenee, ilmestyy pelimaailmaan. Start -funktio suoritetaan niin ikään vain kerran, jos peliobjekti on aktiivinen, mutta vain jos komponentti on käytössä. Update -funktio kutsutaan kerran joka ruudunpäivityksellä. Tässä funktiossa määritellään logiikkaa, jota suoritetaan jatkuvasti, kuten animaatioita, tekoälyä ja muita pelin osia, joita tulee päivittää jatkuvasti. (Unity 2024c.)

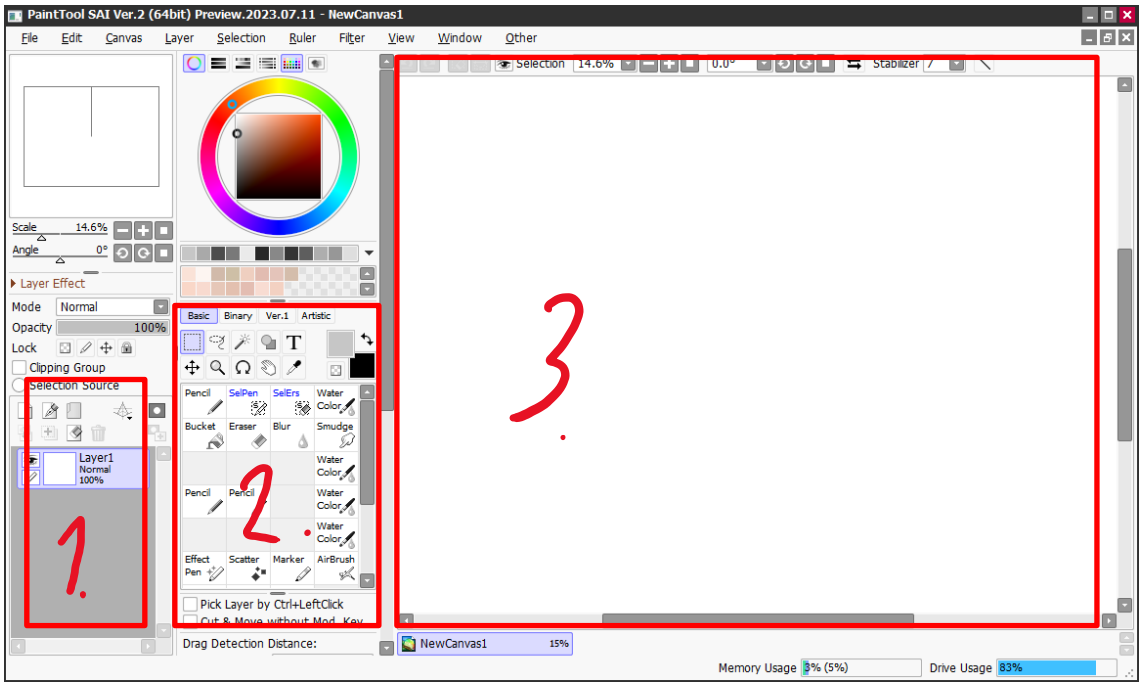
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Esimerkkiluokka : MonoBehaviour
{
    [SerializeField] private InventoryItemData itemData; // Referenssi toiseen luokkaan.

    public GameObject playerPrefab; // Referenssi pelaaja peliobjektiin.

    public int StackSize; // Esimerkkimuuttuja inventaariojärjestelmän koodista.

    // Elinkaari funktio void Start()
    void Start()
    {

    }

    // Elinkaari funktio void Update()
    void Update()
    {

    }
```

KUVA 1. C# koodinäyte

2.2 Grafiikka

Työssä käytetään vihollisten, taustojen ja peliobjektien tekemiseen piirrostyökalua PaintTool SAI. Se on kevyt maalausohjelmisto, joka on erikoistunut digitaaliseen piirtämiseen. Ohjelmisto mahdollistaa maalauksen reunanpehmennyksen ja tarjoaa käyttäjäystävällistä käyttökokemusta. (SYSTEMAX Software Development 2024a).

Vaikka PaintTool SAI on pääosin piirtotyökalu, muistuttaa se ominaisuuksiltaan myös markkinoilla olevia vastaavia kuvanmuokkausohjelmia työkaluillaan, joihin kuuluvat esimerkiksi erilaiset valintatyökalut, lasso sekä taikasauva -ominaisuus. (SYSTEMAX Software Development 2024b). Eri ominaisuudet, kuten siveltimet, edellä mainitut työkalut ovat nähtävissä kuvassa 2 alueella 2. Aluella 3 on piirtoalue ja alueelta 1 löytyy tasot-paneeli.

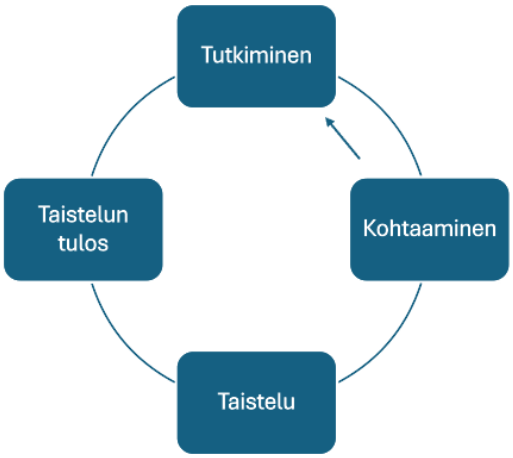


KUVA 2. PaintTool SAI:n työpöytänäkymä

3 PELIN SUUNNITTELU

3.1 Pelin idea

Pelissä yhdistyvät 2D-vuoropohjainen taistelumekaniikka ja 3D-maailma. Pelaaja tutkii ympäristöä, kerää aarteita ja varusteita, taistelee vihollisia vastaan kohtaamisissa sekä käy keskusteluja pelin hahmojen kanssa. Peli on tarkoitettu yksinpeliksi. Pelimaailma koostuu luolista ja maanalaisista linnoituksista, joiden pimeydessä pelaaja kulkee soihdun kanssa. Kuviossa 1 on havainnollistettu demon pelisilmukka (gameplay loop).



KUVIO 1. Demon pelattavuus (gameplay loop)

Kun peli alkaa, herää pelaaja yksin pimeydessä. Lattialla on soihtu, joka on ainoa valonlähde lähiympäristössä. Noustuaan ylös pelaaja löytää soihdun avulla ympäriltään vain niukat varusteet selviytymisen edellytyksenä. Pelaajan on päästävä turvaan mahdollisesti vaarallisesta ympäristöstä. Tavoitteena on saapua ensimmäiselle turvahuoneelle. Pelaaja joko selviytyy matkasta tai kuolee matkan aikana. Jos pelaaja kuolee matkan aikana, herää hän turvahuoneessa, jonne hänet on ilmeisesti tuonut hyväntahtoinen hahmo, jota ei pelata (non-player-character NPC).

Pelaaja etenee 3D-maailmassa, kohdaten Pokemon-pelin kaltaisesti satunnaisia vihollistaisteluita. Pokemon-pelissä ruutu ajoittain muuttuu mustaksi ja vaihtuu taistelunäkymäksi, jossa pelaaja on

vastakkain villin Pokemonin kanssa. Mekaniikka on yleinen myös muissa vuoropohjaisissa roolipeleissä. Reitillä kohti turvahuonetta pelaaja kohtaa korkeintaan 3-4 vihollista. Jos pelaaja selviää matkasta, ottaa hänet vastaan NPC-hahmo. Jos pelaaja kuolee matkalla, kyseinen hahmo tuo pelaajan samaan päämäärään. Keskustelemalla NPC:n kanssa pelaaja saa tietää tilanteestaan ja pelin maailmasta.

Pelin tasot rakentuvat keskeisen turvahuoneen ympärille. Huoneesta erkanevat reitit vievät tuntemattomille alueille, joista osa on suljettu. Turvahuoneet (engl. safezone) ovat tärkeä osa pelin kulkua. Ne tarjoavat rauhallisen ja turvallisen vastapainon pelin jännittäville ja vaarallisille alueille. Pelaaja voi lisäksi tallentaa edistyksensä pelissä, kerätä voimiaan ja valmistautua tuleviin koitoksiin näissä huoneissa. Turvahuoneita ovat esimerkiksi soturin leirintäpaikka tai maagin kenttälaboratorio, joita ylläpitävät NPC:t. Nämä hahmot ovat myös uskaltautuneet tyrmän uumeniin ja ovat keskeisiä pelin tarinalle.

Taistelut hidastavat tutkimista ja kuluttavat resursseja, joten pelaaja joutuu tasapainoilemaan etenemisen ja selviytymisen välillä. Tutkimalla ympäristöä pelaaja voi löytää lisävarusteita ja resursseja, jotka auttavat selviytymään haasteista. Pelaajalla on myös aina vaihtoehtona palata turvahuoneeseen tai puskea eteenpäin kuluneilla resursseilla, mahdollisesti päästen välipysäkille tai oikoreitille takaisin turvahuoneeseen, näin antaen pelaajalle tilaa hengähtää. Pelissä hyödynnetään yhteen kietoutuvaa tasonsuunnittelua, joka tarjoaa pelaajalle monia mahdollisuuksia löytää salaisuuksia ja oikoteitä. Suljetut takaovet ja murentuneet seinämät tarjoavat Metroidvania-tyylisen tason edistymisen, jossa pelaaja voi löytää uusia reittejä pelin olosuhteiden ja edistymisen myötä.

NPC-hahmoilta pelaaja voi ostaa perustarvikkeita ja varusteita. Eri hahmojen kautta pelaaja voi myös saada tärkeää tietoa ja tehtäviä, jotka ohjaavat hänen edistymistään pelissä eteenpäin. Pääkeskusalue (engl. hub area), kuten ylämaailmassa lähettyvillä oleva kylä, toimii keskuksena, jonne pelaaja voi palata myöhemmin pelin aikana. Siellä pelaaja voi saada tehtäviä, tavata muita hahmoja ja hankkia apuvoimia pelin edetessä. Kaupat ja seppä tarjoavat pelaajalle mahdollisuuden parantaa varusteitaan ja valmistautua vaativampiin kohtaamisiin.

Kuvissa 3 ja 4 esitellään peliin suunniteltuja eri hamoja, kuten aikaisemmin käsitellyt maagi ja soturi, jotka ovat luonnosvaiheessa. Demon graafisen ulkoasun on tarkoitus olla mustavalkoinen ja piirretyllä tyylillä toteutettu.

KUVA 3. NPC-hahmoluonnos

KUVA 4. Soturihahmoluonnos

Pelaaja tarvitsee valonlähteitä edetäkseen tehokkaasti pimeydessä. Soihdun käyttö voi kuitenkin vaikuttaa vihollisten käyttäytymiseen. Soihtu voi houkutella tai estää tiettyjä vihollisia lähestymästä riippuen niiden ominaisuuksista ja käyttäytymisestä. Valon läsnäolo voi siis olla pelaajalle haitta, tai hyöty tilanteesta riippuen. Pelaajan tuottama ääni ja toiminta voivat lisäksi houkutella vihollisia.

Kun taisteluita on käyty tietty määrä tai kun pelaaja on saanut tuhottua tason päävastustajan, viholliskohtaamiset menevät rauhoitusajalle määrätyksi ajanjaksoksi. Pelaajalla on mahdollisuus edetä pidempiä matkoja ja halutessaan tutkia tyrmää kulmasta kulmaan.

Kolikonheittomekaniikka päättää, tuleeko pelin aikana taistelua vai ei. Kolikon heittäminen tarjoaa pelaajalle satunnaisia tilanteita ja päätöksiä, jotka voivat vaikuttaa seikkailun kulkuun. Tämä lisää pelin uudelleenpelattavuutta ja antaa pelaajalle mahdollisuuden kokea erilaisia haasteita ja palkintoja jokaisella pelikerralla.

Peli on tarkoitettu nuorille ja aikuisille PC-pelaajille. Peli yhdistää uusia ja vanhoja elementtejä synkän fantasian tyylissä. Peli ottaa vaikutteita FromSoftware-pelistudion filosofiasta. Pelistudion toimintaroolipeleissä keskitytään genren viihteen kulmakiviin, joita ovat jännityspohjainen tutkiminen, löytöjen tekeminen pelimaailmassa ja pelko hirviöiden kohtaamisesta. Pelaaja saa suuren saavutuksen tunteen vastoinkäymisistä selvitessään ja edetessään pelissä (FromSoftware 2024). Taistelut ottavat vaikutteita peleistä Fear and Hunger ja muista vuoropohjaisista roolipeleistä. Graafinen tyyli on omaperäinen ja värimaailmaltaan mustavalkoinen omalla piirros- ja pikseligrafiikan yhdistelmällä. Pelimaailma on kolmiulotteinen, mutta taistelut ovat kaksiulotteisia vuoropohjaisia kohtaamisia, joissa viholliset ovat piirretty kaksiulotteisella grafiikalla.

3.2 Pelin käyttöliittymä ja inventaario

Inventaario ja käyttöliittymä ovat olennainen osa pelaajakokemusta ja ne tarjoavat pelaajalle selkeän ja käyttäjäystävällisen tavan hallita varusteita ja esineitä. Inventaarion hoitaminen ja käsittely on tarkoitus olla keskeinen osa pelattavuusluuppia (gameplay loop). Pelin edetessä pelaaja voi kerätä ja hyödyntää inventaarionsa resursseja. Lisäksi käyttöliittymä ohjeistaa pelaajaa ja on tärkeä komponentti vuorovaikutuksessa pelaajan ja pelin välillä.

Peli sisältää yhtenäisen inventaariojärjestelmä, joka toimii sekä 3D- että 2D-tiloissa. Inventaario ruudut sisältävät erilaisia esineitä kuten rohdot, avaimet, aseet ja muut varusteet, joita pelaaja voi kerätä ja käyttää pelin edetessä. Jokaisella esineellä on oma kuvakkeensa ja kuvaus. Kuvaus kertoo niiden ominaisuuksista ja käyttötarkoituksesta. Se listaa myös esineiden mahdollisia efektejä ja esineiden arvon pelimaailmassa. Efektit kuvaavat, mitä hyötyä tai haittaa esineestä voi olla pelaajalle.

Tavaroilla on eri maksimimäärä, mitä inventaariossa voi olla samanaikaisesti. Jos tavaroita haluaa enemmän mitä maksimäärä sallii, vievät ne enemmän tilaa inventaariossa. Inventaarion tila on rajoitettu, mikä voi aiheuttaa pelaajalle haasteita esineiden hallinnassa ja varastoinnissa. Mikäli tila loppuu, voi pelaaja siirtää tavaroita turva-alueella olevaan laatikkoon tai valikoidusti pudottaa tavaroita.

Pelaajalla on käytössään pikavalikko, johon hän voi asettaa joko neljä käyttöesinettä tai taitoa taistelua varten. Pikavalikko antaa pelaajalle nopean ja helpon tavan käyttää tärkeitä esineitä ja taitoja. Lisäksi pelaajalla on mahdollisuus varustaa erilaisia suojavarusteita ja vaatteita eri kehonosiin, kuten päähän, rintakehään, käsiin ja jalkoihin, mikä antaa lisäsuojaa ja mahdollisuuksia muokata hahmon ulkonäköä ja ominaisuuksia. Varusteet ja vaatteet valitaan erillisessä varustevalikossa. Kaikki tämä tehdään taistelun ulkopuolella ja niitä ei voi muokata sen aikana. Aseet ja varusteet sijoitetaan omiin aseruutuihin, jotka pelaaja voi varustaa oman mieltymyksensä mukaan. Pelaajan hahmon räätälöinti ja pelityylin valitseminen mahdollistaa strategisen valmistelun ennen taistelua.

3.3 Taistelut

Maailman tutkimisen lisäksi pelaaja taistelee eri vihollisia vastaan satunnais- taistelukohtaamisissa. Pelin taistelut tapahtuvat vuoropohjaisesti, eli pelaaja ja vihollinen vuorottelevat eri toiminnoin, kunnes kohtaaminen päättyy. Vuoropohjainen taistelu tarjoaa mahdollisuuden tehdä strategisia päätöksiä ja hyökkäyksiä vihollisia vastaan, kun taas vihollisen vuoro tuo lisää haasteita ja vaaroja pelaajalle. Voittaessaan taistelun, pelaaja voi jatkaa seikkailuaan 3D-maailmassa ja, jos pelaaja kuolee, hän palaa takaisin viimeisimmälle tallennuspisteelle.

Taistelut tapahtuvat erillisellä taisteluruudulla, johon ajoittain vaihdetaan tutkimisruudusta. Taisteluruutu toteutetaan 2D-pohjaisella grafiikalla. Tausta on maalaustyylinen vastike pelaajan sen hetkiselle tutkimisruudun sijainnille. Piirretyt vihollliset omaavat vain minimaaliset animaatiot eri hyökkäyksille, kuten välähtävä miekan viilto tai räjähtävä taika.

Pelaajan ja vihollisen tiedot, kuten vihollisen taso (level, LVL), elinvoima (health point, HP) ja mana (MP), näytetään selkeästi näytöllä, jotta pelaaja voi seurata taistelun etenemistä ja tehdä suunni-

telmallisia päätöksiä. Pelaajan toimintavaihtoehdot perustuvat pelaajan hahmon taitoihin ja varusteisiin, jotka sisältävät erilaisia hyökkäyksiä, taikoja ja puolustusvaihtoehtoja. Pelaajahahmon toimintavaihtoehdot ottavat vaikutteita Baldur's Gate -pelisarjan monipuolisista pelaajan toimintavaihtoehdoista. Kuvassa 5 on kyseisen pelisarjan toimintovalikko. Demon taistelunäkymän olisi tarkoitus sisältää pikavalikko pelaajan käyttötavaroille (hotbar-alue) sekä käyttövarusteille (attack actions –alue), kuten miekalle ja kilvelle.

KUVA 5. Pelaajan toimintavaihtoehdot Baldur's Gate 3 -pelissä (Larian Studios 2020.)

Taistelussa on käytettävissään erilaisia esineitä, jotka voivat auttaa kääntämään taistelun kulun. Esineisiin kuuluvat elinvoimaesineet, kuten elinvoimaa ja manaa palauttavat esineet, tehoste-esineet, jotka parantavat pelaajan kykyjä tilapäisesti, sekä hyökkäysesineet, jotka voivat heikentää vihollisia tai vihollisten kykyjä. Tavaroiden käyttö voi olla ratkaisevaa taistelun voiton kannalta ja pelaajan on oltava suunnitelmallinen niiden käytössä.

3.4 Vihollistyypit

Tyrmää hallitsevat monenlaiset pelaajaa vastustavat vihollistyypit. Luurankojen hallitsijana toimii nekromantikkojen herra, joka paimentaa alemman luokan eläviä kuolleita. Kuvasta 6 näkee hahmotelman nekromantikosta. Syvemmällä tyrmän uumenissa asustaa joukko olentoja, joiden pelimaailman ihmiskunta kuvaisi omaavan paholaismaisia piirteitä. Olentoja löytyy kaiken kokoisia, mutta heitä yhdistävät tietyt piirteet.

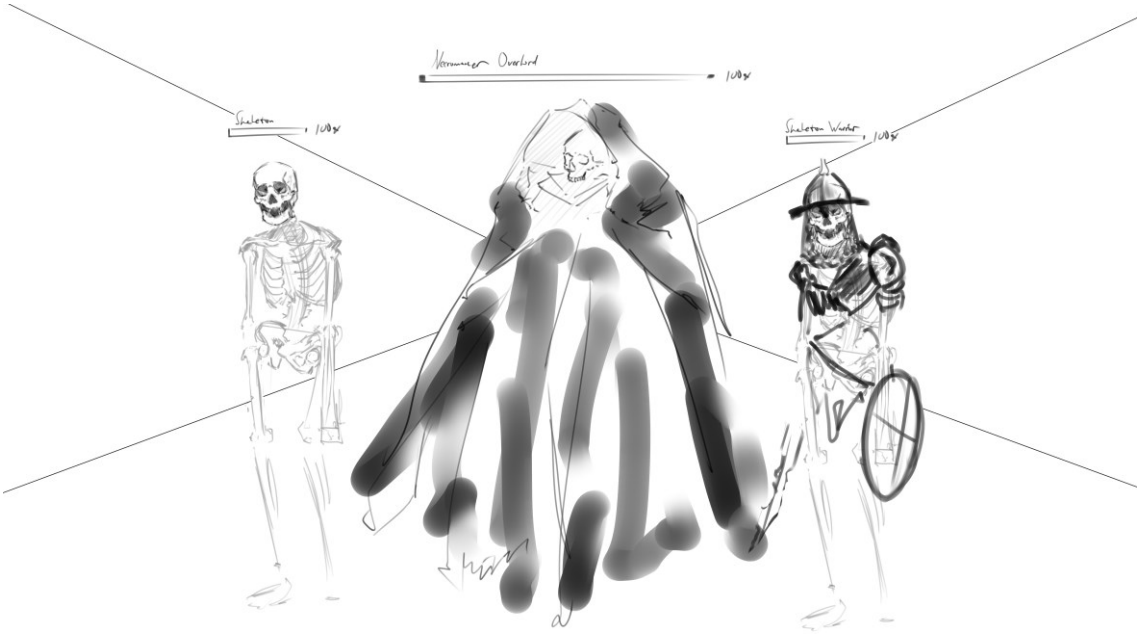
KUVA 6. Ensiluonnos nekromantikosta.

Alimman tason luuranko ei omista paljoa varustusta eikä vihollisena ole kovin vaikea vastus pelaajalle. Nämä luurangot eivät yksinään tuota suurta uhkaa pelaajalle, paitsi yhdessä toisten vihollistyyppien kanssa tai kukistaessaan suurella lukumäärällään huonosti varautuneen pelaajan. Luurankojen kukistamisesta pelaaja ei saa suurta palkkiota, vaan pelkästään pienen määrän pelin valuuttaa tai kehnoja ja lahoja varusteita luurangon aiemmasta elämästä.

Vaikka luurankosotilaalla on miekka, kilpi ja raskaampi haarniska yllään, ne ovat yleisen seikkailijan tai sotilaan varusteisiin verrattuna heiveröisempiä. Pelaaja kestää ongelmitta yhtä epäkuollutta sotilasta vastaan, mutta jo toinen lisää vaikeustasoa huomattavasti. Luurangoista koostuva vihollisyksikkö, mukaan lukien molemmat luurankolajit, on yleinen kohtaaminen pelin alkuvaiheen jälkeen. Kuvassa 7 on hahmoteltu luurankoviholistyypit.

KUVA 7. Luonnos luurankoviholistyypeistä

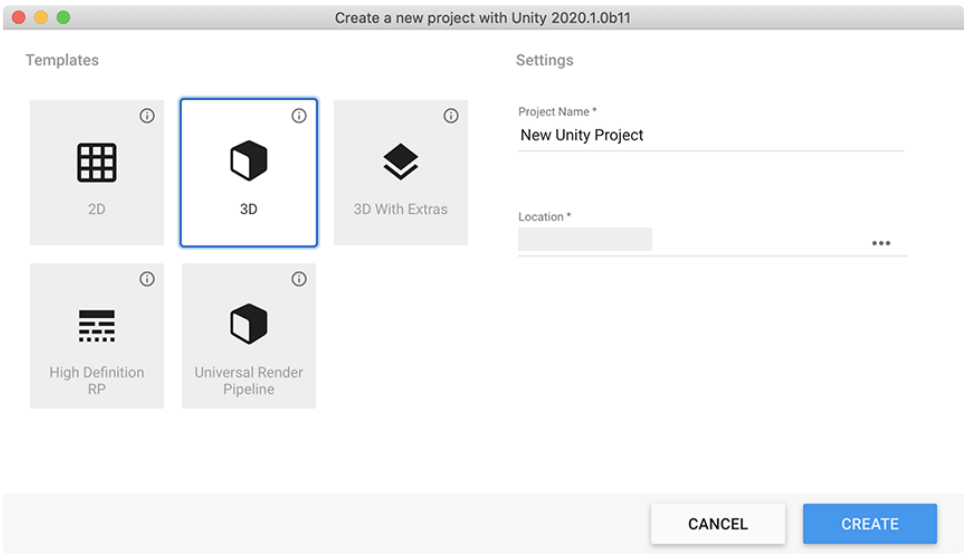
Luurankojen hallitsija toimii päävastuksena ensimmäisellä alueella. Taistelun edetessä nekromantikko herättää eläviä kuolleita avukseen, näin vaikeuttaen taistelua, kuten kuvasta 8 nähdään.



KUVA 8. Luonnos päävastuksen kohtaamisesta

4 PELIN TOTEUTUS

Pelin kehitys alkoi valitsemalla Unityn 3D mallipohja ja tutustumalla Unityn Asset storeen ja sen tarjontaan. Projektia aloittaessa Unity tarjoaa useaa mallipohjaa, joista käyttäjä voi valita sopivan vaihtoehdon omaan käyttötarkoitukseensa. Projektipohjat tarjoavat ennalta valittuja asetuksia yleisimpien parhaiden käytäntöjen perusteella erityyppisille peliprojekteille. Asetukset on optimoitu eri projekteille, kuten 2D- ja 3D-peliprojekteille. Mallit nopeuttavat pelinteon alkuvalmisteluprosessia ja kohdistuvat pelityyppiin tai visuaalisen laadun tasoon. (Unity Documentation 2024b.) Kuvassa 9 on käyttäjän näkymä Unityn tarjoamista projektimalleista.



KUVA 9. Unity Hub -projektialoitusnäkymä (Unity Documentation 2024b.)

Unity Asset Store sivua selatessa listasin assetteja, tekstuureja ja valmista koodia käyttöä varten. Tutustuin muun muassa Unityn ensimmäisen persoonan liikkumisen aloittelijapakettiin ja otin sen käyttöön projektissa. Paketin mukana oli myös tekstuureja tasojen luomiseen. Aloin implementoimaan alustavasti pelimaailmaan geometriaa ja asetin niille tekstuureja siellä.

4.1 3D- ja 2D -maailmoiden vuorovaikutus

Siirtyminen 3D-maailman tutkimisen ja 2D-taistelun välillä toimii samalla Stealth-pelimekaniikkana. Kun hahmo on liikkeessä ja tuottaa ääntä, pelinsisäinen ajastin laskee sekunteja kohti nollaa, jolloin algoritmi valitsee satunnaisen numeron, kuten kuvasta 10 rivillä kolme on nähtävissä. Jos kyseinen

numero osuu algoritmin taistelukynnykseen (KUVA 10 rivi 4), taistelu alkaa. Jos pelaaja väistää kohtaamisen, kynnys kasvaa ja taistelu on todennäköisempi seuraavalla kerralla (KUVA 10 rivi 16). Lisäksi ajastin on aikamäärältään lyhyempi. Kun kohdalle sattuu taistelu, vaihtuu ruutu taistelunäkymään (KUVA 11) ja kynnys sekä ajastin palautuvat alkuperäisiin arvoihinsa (KUVA 10 rivit 7,8).

```
TimerOn = false;

int value = Random.Range(0, 100);
if (value < currentEncounterThreshold)
{
    Debug.Log("Encounter started");
    TimeLeft = TimeLeftReset;
    currentEncounterThreshold = DEFAULT_ENCOUNTER_THRESHOLD;
    LoadCoin(1);
    LoadNextLevel();
}
else
{
    Debug.Log("Encounter Skipped");
    TimeLeft = TimeLeftNew;
    currentEncounterThreshold += 15;
    LoadCoin(2);
}
```

KUVA 10. Encounter Manager -luokka

```
IEnumerator LoadLevel()
{
    transition.SetTrigger("Start");
    yield return new WaitForSeconds(transitionTime);
    ScenesManager.Instance.LoadScene(ScenesManager.Scene.Combat);
}
```

KUVA 11. Encounter Manager -luokka ja LoadLevel-funktio

Taistelun todennäköisyys kasvaa ajan myötä, mikä luo jännitystä ja odottamattomuutta pelaajalle. Lisäksi systeemi estää taistelut pelaajan ollessa paikoillaan, jos pelaaja esimerkiksi tarkastelee ympäristöään (KUVA 12). Kuvien 10 ja 12 koodi on void Update -elinkaarifunktion sisällä, joten pelaajan sijainti ja ajastin päivittyvät joka ruudunpäivityksen yhteydessä.

```
if (player.transform.position != lastPos)
{
    TimerOn = true;
}
else
{
    TimerOn = false;
}

lastPos = player.transform.position;
```

KUVA 12. Encounter Manager luokan ajastintoiminto

Kolikonheittomekaniikkaa kuvaa ruudun oikeaan yläreunaan ilmestyvä kolikonheittoanimaatio (KUVA 13). Koodi aktivoi kolikonheittoanimaation ja valitsee kolikon kääntöpuolen, jonka jälkeen ruutu tummenee, peli pysähtyy ja ruudun keskelle ilmestyy joko taistelua tai suojautumista edustava kolikon symboli.

KUVA 13. Kolikonheittotapahtuma

ScenesManager-luokka käsittelee kaikki peli-instanssit, kuten päävalikon ja peliruudut (KUVA 14). Lisäksi se sisältää toiminnallisuuden ja funktiot ruudun vaihtamiseen, joita muut luokat voivat hyödyntää mistä tahansa (KUVAT 14 ja 15). Kaikki scenet pelin sisällä Scene enum -muutujan sisällä (KUVA 14, rivi 10) ja ne ovat samassa järjestyksessä projektin koontiversio asetuksissa. Scene -

näkymien vaihtamisen toiminnallisuus toteutettiin seuraamalla Dani Krossing Youtube-kanavan ohjevideota. Unityn SceneManagement-toimintoa hyödyntämällä saadaan aikaiseksi asianmukainen näkymänvaihtaja -ominaisuus (Krossing 2022).

```
public class ScenesManager : MonoBehaviour
{
    public static ScenesManager Instance;

    private void Awake()
    {
        Instance = this;
    }

    public enum Scene
    {
        MainMenu,
        Level01,
        Combat
    }
```

KUVA 14. SceneManager-luokka

```
public void LoadScene(Scene scene)
{
    SceneManager.LoadScene(scene.ToString());
}

public void LoadNewGame()
{
    SceneManager.LoadScene(Scene.Level01.ToString());
}

public void LoadNextScene()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
}

public void LoadMainMenu()
{
    SceneManager.LoadScene(Scene.MainMenu.ToString());
}
```

KUVA 15. SceneManager-luokka ja ruudunvaihto-funktiot.

4.2 Taistelusysteemi

Taistelusysteemi toteutettiin seuraamalla Brackeys Youtube-kanavan ohjevideota. Videon tavoitteena on havainnollistaa yksinkertaisen, mutta toimivan vuoropohjaisen taistelusysteemin luominen Unityssä (Brackeys 2019).

Taistelut rakentuvat BattleState-muuttujan ympärille. BattleState on Enumerator-muuttuja, jonka avulla voi luoda kokoelman toisiinsa liittyviä vakioita (Unity Learn 2023). Se sisältää kaikki taistelun vaiheet, joita aktivoidaan taistelun tilan kulun mukaan. (KUVA 16)

```
public enum BattleState { START, PLAYERTURN, ENEMYTURN, WON, LOST }
```

KUVA 16. BattleSystem-luokan BattleState-muuttuja

START-vakio kutsutaan void Start-elinkaarifunktiossa heti taistelun alettua. Samalla kutsutaan SetupBattle Coroutine-funktio. Funktio tuo näkyviin ja aktivoi pelaaja sekä vihollishahmot. Peliobjektit rakentuvat valmiista Prefab-elementeistä ja koodi hakee vastaavat komponentit Unityn puolelta. Samalla käyttöliittymässä pelaajaa informoidaan tapahtumaketjusta dialogueText-elementissä. Funktion lopuksi taistelutila vaihdetaan pelaajan käsiin.

Unity suorittaa koodia joka ruudunpäivityksellä, mutta IEnumerator-funktiomuodon käyttö mahdollistaa odottamisen taistelun tilojen välillä, tehden taistelun kulun seuraamisesta selvemmän pelaajalle.

```
void Start()
{
    state = BattleState.START;
    StartCoroutine(SetupBattle());
}

IEnumerator SetupBattle()
{
    GameObject playerGO = Instantiate(playerPrefab, playerBattleStation);
    playerUnit = playerGO.GetComponent<Unit>();

    GameObject enemyGO = Instantiate(enemyPrefab, enemyBattleStation);
    enemyUnit = enemyGO.GetComponent<Unit>();

    dialogueText.text = "A " + enemyUnit.unitName + " approaches...";

    playerHUD.SetHUD(playerUnit);
    enemyHUD.SetHUD(enemyUnit);

    yield return new WaitForSeconds(1f);

    state = BattleState.PLAYERTURN;
    PlayerTurn();
}
```

KUVA 17. BattleSystem-luokka

PLAYERTURN-vakion kohdalla on pelaajan vuoro, jossa pelaaja tekee valinnan, joka on joko hyökkäävä tai puolustava tekniikka, tavara/hyödyke/potion toiminto, taistelusta pakeneminen tai odottaminen, joka ohittaa pelaajavuoron. ENEMYTURN-vakiossa vihollinen tekee yhden valmiiksi koodatun vaihtoehdon muutamasta vihollistoiminnosta, kuten esimerkiksi hyökkäys, erikoishyökkäys, puolustus tai parantaminen.

WON-vakiossa pelaaja on voittanut taistelun ja käyttöliittymä siirtää pelaajan seuraavaan ruutuun takaisin pelimaailmaan. LOST-vakiossa vihollinen päihittää pelaajan, joka palaa edelliselle tallennuspisteelle.

Unit -koodi sisältää vihollisyksikön tiedot, kuten sen nimi (unitName), taso (unitLevel), sen tekemä vahinko (damage), maksimi elinvoima (maxHP), senhetkinen elivoima (currentHP) ja puolustus arvo (defence). Pelaajan hyökätessä koodi laskee pelaajahahmon tekemän vahingon ja vähentää sen vihollisen HP:sta. Jos pelaaja tai vihollinen puolustaa, vähennetään tehdystä vahingosta yksikön puolustus arvo.

```
public class Unit : MonoBehaviour
{
    public string unitName;
    public int unitLevel;

    public int damage;

    public int maxHP;
    public int currentHP;
    public int defence;
```

KUVA 18. Vihollisyksikön tiedot Unit-luokassa

Ohjevideon pohjalta sovelsin toimintojen lisäksi pelaajalle puolustus, odotus ja pakenemistoiminnot (KUVA 19). Taisteluruutu on eritelty osiin. Yläruudussa näkyvät vihollisen statistiikka ja keskelle ruutua ilmestyvät viholliset. Alaruudun vasemmalla puolella on pelaajan statistiikka ja sen oikealla puolella ovat pelaajan vaihtoedot. Pelaajan tietojen yläpuolella sijaitseva tekstielementti päivittää pelaajaa taistelun kulusta (KUVA 19)

KUVA 19. Taistelunäkymä

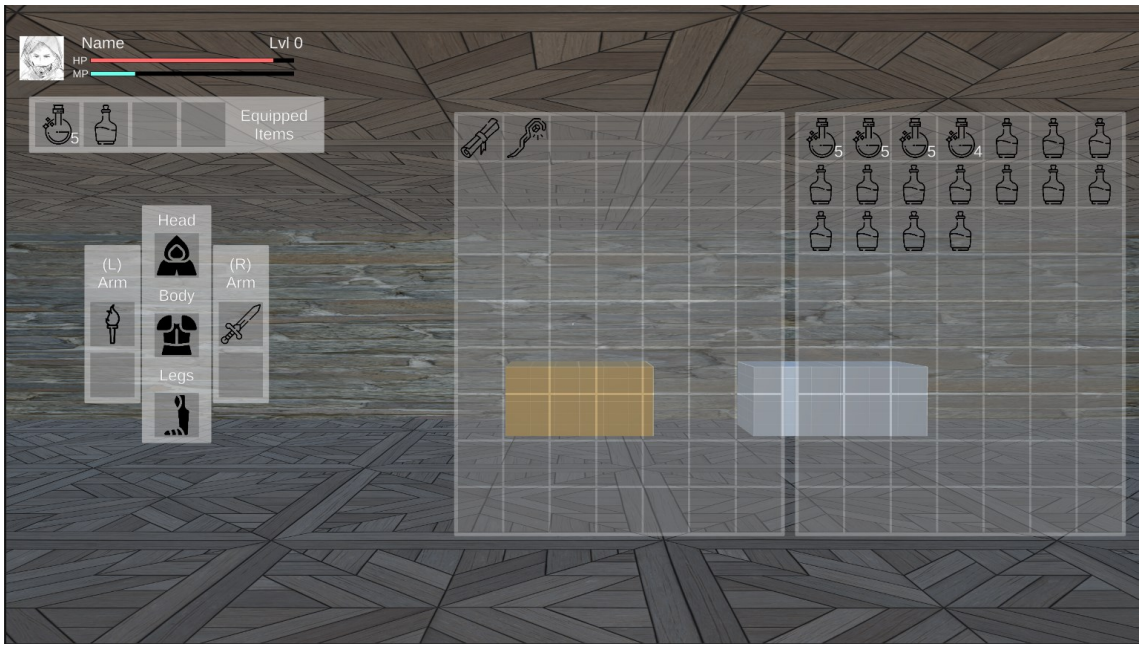
4.3 Inventaario ja käyttöliittymä

Demon inventaario järjestelmä toteutettiin seuraamalla kanavan Dan Pos ohjevideosarjaa liittyen inventaario järjestelmän toteuttamiseen hyödyntäen Unityn ScriptableObject-toimintoa. Ohjevideosarja käsittelee ruudukkopohjaisen inventaariojärjestelmän luomista Unityssä. Siihen sisältyy esimerkiksi pinottavat tavarat rajoitetulla maksimimäärällä. Pinoja voi jakaa ja yhdistää halutulla tavalla sekä poistaa kokonaan inventaariosta. Kyseinen inventaario on roolipeleihin sopiva. (Postlethwaite 2021.)

Pelaajan oma inventaario on isoin ja sinne voi järjestellä paremmin tavaroita ja sen ideana on toimia enemmän varasto valikkona pelissä. Sinne voi sijoittaa ei niin tärkeät tavarat, joita ei välttämättä heti tarvitse käyttöön. Kuten kuvasta näkee pelaajan inventaarionäkymä avautuu ruudun oikeaan laitaan ja ruudun keskelle avautuu tilanteesta riippuen esimerkiksi sillä hetkellä aktivoitu arkku pelissä. Pelaajanäkymän vasemmassa reunassa näkyvät pikavalikko sekä käytössä olevat taistelu-välineet, kuten lyömäaseet, taiat ja puolustusvälineet.

Koska inventaario on ruudukko pohjainen siihen implementointiin raahaus ominaisuus, joka mahdollistaa tavaroiden siirtelyn ja pinoamisen eri varastojen välillä.

Kuvassa 20 näkyy, miten eri inventaariojärjestelmät hahmottuivat demoon arkkujen, pelaajan oman inventaarion sekä pikavalikon osalta. Ensimmäisen persoonan tutkimisruudussa näkyy vasemmassa yläreunassa pelaajastatistiikan alapuolella pikavalikko, johon voi valita tärkeimmät käyttötavarat, jotka ovat myös käytössä taistelussa. Käyttötavarat tulee valita ennen taistelua, sillä niitä ei voi enää vaihtaa, kun taistelu on alkanut.



KUVA 20. Eri inventaarioruudukot

Keskusteluominaisuus implementoitiin BMo Youtube-kanavan ohjevideon mukaan. Kyseisen dialogijärjestelmän ei ole tarkoitus tukea mitään laajamittaista systeemiä, mutta se on oiva lähtökohta saada alustava vuorovaikutus toimimaan NPC-hahmojen kanssa (BMo 2021).

Keskusteluruutu toimii keskustelun välineenä NPC-hahmojen kanssa. Ruutu koostuu tekstilaatikosta, joka sisältää dialogin. Keskustelunäkymässä pelaaja saa yleistietoa, vihjeitä tarinan tapahtumista, sekä mahdollisia vinkkejä pelin eri ominaisuuksista.

NPC-hahmo sijaitsee ruudun oikealla puolella. Pelaaja hahmo taas vasemmalla. Hahmot vuorottelevat ruudulla keskustelun mukaan. Hahmon dialogi näkyy tekstilaatikossa ruudun alhaalla.

Tekstilaatikosta löytyy keskusteltavan hahmon nimi. Pelaajan valinnat keskusteltaessa ovat esimerkiksi kyllä tai ei -vaihtoehdot. Kun alkukeskustelu käyty läpi, tulle valikko, jossa on listattuna vaihtoehdot keskustele, kauppa tai poistu keskustelusta. Edellä läpikäydyt ominaisuudet ovat nähtävissä kuvasta 21.

KUVA 21. Keskustelukäyttöliittymä

Koodi käy tekstimuuttujat läpi kirjain kirjaimelta ja ne ilmestyvät halutulla nopeudella teksti laatikoon näin tehden lukemisesta luontevampaa. Klikkaamalla pelaaja etenee keskustelussa ja käy läpi dialogin. Lauseet pilkotaan osiksi, jotka For-silmukka käy läpi halutulla nopeudella. Lauseet ovat määritelty listana muuttujia.

5 POHDINTA

Demon työstäminen on ollut opettavainen kokemus ylipäätään pelinkehitystyökalujen käyttämisen näkökulmasta. Unityn käyttäminen on opettanut lisää pelien tekemisen eri vaiheista, vaikkakin kyseessä on ollut rajatun demon laatiminen. Grafiikan laatiminen on ollut itse piirtämisen takia pitkä prosessi. Toisaalta projektin luonteen myötä grafiikka ei ole vielä tässä vaiheessa niin tärkeässä roolissa, kuin itse pelin peruspilarit eli pelattava sisältö.

Demossa toteutettujen perusominaisuuksien jälkeen on mahdollisuus lähteä rakentamaan lisää sisältöä ja syvyyttä demon rungon ympärille, jotta demovaiheesta voitaisiin siirtyä tulevaisuudessa toimivaksi peliksi. Inventaariojärjestelmä, taistelumekaniikka ja 3D- ja 2D -alueiden välillä liikkuminen ovat toiminnallisuudeltaan jo kunnossa ja näiden ominaisuuksien toimivuus luo hyvän pohjan pelin jatkokehitykselle ja laajentamiselle.

Luodut edellä mainitut niin sanotut demon moduulit ovat nyt valmiiksi opiskeltu työn laatimisen myötä ja seuraavana askeleena olisi niiden parempi sitominen yhteen yhteneväisemmän pelikokemus luomiseksi. Luotujen systeemien laajentaminen olisi tarkoitus tehdä tulevaisuudessa ja lopputavoitteena olisi pelattava pelikokemus. Tämä vaatisi jo luotujen ominaisuuksien hiomista ja kehittämistä ja samalla peliin lisättäisiin lisäsisältöä. Mahdollinen kaupankäynnin implementointi ei-pelaaja-hahmojen kanssa, jossa voisi hyödyntää jo laadittuja inventaario- ja dialogijärjestelmiä olisi yksi potentiaalinen jatkokehityksen kohde.

Työn aikana tärkein resurssi on aika ja niinpä kaikkia ominaisuuksia ei ole mahdollista toteuttaa rajallisen ajan takia. Muun muassa soihtumekaniikan toteuttaminen peliin juuri sellaiseksi kuin alkuperäinen hahmotelma siitä on. Se vaatisi vielä tarkempaa tarkastelua tulevaisuudessa, jotta se toimisi eri tavalla pelimaailmassa, esimerkiksi lähtökohtaisesti valaistuksen lähteenä pelimaailmassa mutta myös mahdollisesti pelaajan apuna muissakin tilanteissa. Myös kolikonheittoon liittyvän mekaniikan monipuolistaminen vaatii jatkuvaa kehitystä ja hienosäätöä. Alkuperäisen kolikonheitto-mekaniikan päälle voidaan lisätä erilaisia muuttujia, kuten pelaajan tekemät valinnat, päävihollisten vaikutukset tai pelimaailman tilanteeseen liittyvät tekijät. Näiden muuttujien integrointi kolikonheittoon lisäisi entisestään pelin syvyyttä ja pelaajan vaikutusmahdollisuuksia.

Työssä esitelty tämänhetkinen osuus demosta antaa maistiaisen siitä, millainen lopullinen peliprojekti voisi olla. Se toimii esimerkkinä pelin mekaniikoista, visuaalisesta ilmeestä ja pelattavuudesta. Demo sisältää yhden pääpelialueen, joka antaa pelaajalle käsityksen pelin maailmasta ja tarinasta. Tämä on tärkeä vaihe projektin kehityksessä, jossa testataan pelin perusideoita ja tätä olisi tavoitteena jalostaa ennen pitkää laajemmaksi kokonaisuudeksi.

NPC-dialogin kehittäminen pelissä on mahdollinen kehittämisen kohde. Jatkossa pelaaja voisi kommunikoida enemmän muiden hahmojen kanssa, saada tehtäviä ja oppia lisää pelimaailmasta. NPC-dialogin edistäminen voi sisältää monimutkaisten dialogipuiden luomista, erilaisten vastausvaihtoehtojen toteuttamista ja hahmojen omaperäisempien ilmaisujen luomista. Tämä osa-alue vaatii sekä kirjoittamista että ohjelmointia.

LÄHTEET

Aladdin, Jamal 2023. The Future of Game Development: Unity or Unreal Engine? Hakupäivä 3.3.2024. https://medium.com/@Jamal_Aladdin/the-future-of-game-development-unity-or-unreal-engine-a0ea1d6f984d.'.

Bajaj, Niharika 2023. Best Resources and Tools for Game Development: A Complete Guide. Hakupäivä 24.3.2024. https://blog.searchmyexpert.com/game-dev-tools-resources/.

BMo 2021. 5 Minute Dialogue System in Unity Tutorial. Hakupäivä 4.5.2024. https://www.youtube.com/watch?v=8oTYabhj248&pp=ygUVdW5pdHkgZGlhbG9ndWUgc3lzdGVt.

Brackeys 2019. Turn-Based Combat in Unity. Hakupäivä 27.4.2024. https://www.youtube.com/watch?v=_1pz_ohupPs&ab_channel=Brackeys.

Drake, Jeff 2023. TheGamer. 24 Great Games That Use The Unity Game Engine. Hakupäivä 3.3.2024. https://www.thegamer.com/unity-game-engine-great-games/#subnautica.

FromSoftware 2024. Products. Dark Souls. Hakupäivä 21.4.2024. https://www.fromsoftware.jp/ww/detail.html?csm=086.

Krossing, Dani 2022. CHANGE SCENE WITH BUTTON IN UNITY. Scene Manager in Unity. Learn Unity. Hakupäivä 28.4.2024. https://www.youtube.com/watch?v=jrPTpD2eAMw&ab_channel=DaniKrossing.

Larian Studios 2020. Artikkelissa Jeffrey Parking & Jeff Ramos. Baldur's Gate 3 guide: Understanding the interface, HUD, icons, and minimap. Hakupäivä 20.3.2024. https://www.polygon.com/baldurs-gate-3-guide-walkthrough/21504872/hud-screen-interface-buttons-layout-menus-turn-order-map-actions-bonus-attack-spells-items-reaction.

LinkedIn 2024. Which technologies and tools do you use for game development? Hakupäivä 3.3.2024. https://www.linkedin.com/advice/0/which-technologies-tools-do-you-use-game-development.

Oulun ammattikorkeakoulu 2024. Opinnäytetyö. Erilaisia raportointimuotoja. Perinteinen raportointi. Hakupäivä 24.2.2024. https://www.oamk.fi/opinto-opas/opintojen-sisalto/opinnaytetyo.

Postlethwaite, Daniel 2021. Inventory Tutorial Series. Hakupäivä 27.4.2024. https://www.youtube.com/playlist?list=PL-hj540P5Q1hLK7NS5fTSNYoNJpPWSL24.

Stuart, Keith 2021. Dungeon crawler or looter shooter? Nine video game genres explained. Hakupäivä 26.4.2024. https://www.theguardian.com/games/2021/oct/11/modern-video-game-genres-explained-metroidvania-dungeon-crawler.

SYSTEMAX Software Development 2024a. PaintTool SAI. Hakupäivä 24.2.2024. https://www.systemax.jp/en/sai/.

SYSTEMAX Software Development 2024b. PaintTool SAI Development Room. Hakupäivä 24.2.2024. https://www.systemax.jp/en/sai/devdept.html.

Unity 2024a. Products. Unity Engine. Hakupäivä 24.2.2024. https://unity.com/products/unity-engine.

Unity 2024b. Develop games for desktop platforms. Hakupäivä 24.2.2024. https://unity.com/solutions/desktop-games.

Unity 2024c. Coding in C# in Unity for beginners. Hakupäivä 10.3.2024. https://unity.com/how-to/learning-c-sharp-unity-beginners.

Unity Documentation 2024a. Gameobject. Hakupäivä 10.3.2024. https://docs.unity3d.com/560/Documentation/Manual/class-GameObject.html.

Unity Documentation 2024b. Project Templates. Hakupäivä 14.4.2024. https://docs.unity3d.com/2020.1/Documentation/Manual/ProjectTemplates.html.

Unity Learn 2023. Enumerations. Hakupäivä 28.4.2024. https://learn.unity.com/tutorial/enumerations.

Unity Support 2024. What is the Unity Asset Store and how do I purchase Assets? Hakupäivä 24.2.2024. https://support.unity.com/hc/en-us/articles/210142503-What-is-the-Unity-Asset-Store-and-how-do-I-purchase-Assets.