Näyttötyö: ***Ohjelmistosuunnittelu ja ohjelmistokehitysprojektin hallinta,***

SPACEWAR-pelin refaktorointi client-server -arkkitehtuuriin

Tekijä: *Lasse Simonen*

Laatija: Lasse Simonen

Luottamuksellisuus: julkinen

Hyväksynyt: _____

Versio: 0.0.3

Sisällys

Muutoshistoria

Henkilö	Päiväys	Versio	Kommentti
Lasse Simonen	10.6.2025	0.0.1	Dokumentti luotu, johdanto
Lasse Simonen	11.6.2025	0.0.2	Dokumenttia muokattu: johdanto, määritelmät, kohderyhmä, tavoitteet, tehtäväluettelo, organisaatio, resurssit, kustannusarvio, luottamuksellisuus, riskit, laatu
Lasse Simonen	12.6.2025	0.0.3	Dokumenttia muokattu: johdanto, tarkennettu määritelmät, lisätty tehtäväluettelon esittely, tarkennettu ohjelmistoresurssit ja -menetelmät, tarkennettu aikatauluriskit, yleinen dokumentin muotoilu ja muotoiluasetukset. Lisätty Hyväksymismenetelmä.
Lasse Simonen (Uusi asiakirjakokonaisuus Projektisuunnitelma -asiakirjan pohjalta. Säilytetty vanha muutoshistoria tähän uuteen dokumenttiin	13.6.2025	0.0.1	Luotu Projektisuunnitelma -asiakirjasta laajempi kokonaisuus: Näyttötyödokumentti. Laskettu vanhat otsikkotasot yhdellä alaspäin, muokattu Heading 1-tasoa (-> OSA A: Projektisuunnitelma). Esitelty Projektisuunnitelma -osio sekä Johdanto.
Lasse Simonen	17.06.2025	0.0.2	OSA B: Järjestelmän arkkitehtuuri kaavioineen, käyttötapauskaavio ja esittely, käyttöliittymäsuunnitelma wireframe-kuvina, luokkakaavio ja esittely, sekvenssikaavio ja esittely, aktiviteettikaavio ja esittely
Lasse Simonen	18.06.2025	0.0.3	OSA B: Päivitetty luokkakaavio (lisätty luokka Game).
Lasse Simonen	18.07.2025	0.0.4	OSA B: Päivitetty OSA B vastaamaan refaktoroidun ohjelman todellisuutta. Muokattu: Kaaviot ja sanalliset selitykset.
Lasse Simonen	21.07.2025	0.0.5	OSA C: Lisätty osa C.

Taulukko 1. Muutoshistoriataulukko

OSA A: Projektisuunnitelma

Tämä dokumentin ensimmäinen osa muodostaa projektin virallisen projektisuunnitelman. Projektisuunnitelma toimii hankkeen perustana ja ohjaavana karttana, joka määrittelee työn tavoitteet, laajuuden, aikataulun, resurssit sekä tunnistetut riskit ja niiden hallintakeinot. Huolellinen suunnittelu on onnistuneen teknisen toteutuksen edellytys.

1 Johdanto

Tämä johdantoluku asettaa projektin kontekstiin. Aluksi määritellään tämän suunnitelmadokumentin tarkoitus, minkä jälkeen kuvataan projektin tausta, inspiraation lähteet ja ne syyt, jotka ovat johtaneet sen käynnistämiseen. Luvun tarkoituksena on antaa lukijalle kattava yleiskuva projektin perusteista.

1.1 Dokumentin tarkoitus ja sisältö

Tämän dokumentin tarkoituksena on määritellä SpaceWar -strategiapelin tavoitteet, laajuus, vaiheet, aikataulu ja lopputulokset. Suunnitelma toimii projektin ohjaavana dokumenttina ja on tarkoitettu projektin toteuttajalle, oppilaitoksen ohjaavalle opettajalle sekä näyttötyön arvioijille.

Projekti perustuu olemassa olevaan peliprototyyppiin nimeltä SpaceWar – a tiny 4x strategygame (versio 0.20). Prototyyppi on täysin asiakaspohjainen (client-side), yksittäisenä HTML-tiedostona itch.io-alustaa varten toteutettu selainpeli, joka on toiminnallisesti valmis, mutta ei täytä modernin verkkosovelluksen arkkitehtuurivaatimuksia.

Tämän projektisuunnitelman tavoitteena on ohjata prosessia, jossa tämä prototyyppi refaktoroidaan kestävälle client-server-arkkitehtuurille, toteutetaan uusia peliominaisuuksia ja tuotetaan ammattimainen projektidokumentaatio osana Ohjelmistosuunnittelu ja ohjelmistokehitysprojektin hallinta -tutkinnon osan suorittamista. Projektin keskeisenä tavoitteena ei ole ainoastaan tekinen toteutus vaan myös huolellinen suunnittelu, testaus ja dokumentointi.

1.2 Projektin tausta

Projektin inspiraatio ja konseptuaalinen lähtökohta juontaa juurensa aiemmin toteutetusta teknisestä kokeilusta, jossa kehitin sovelluksen visualisoimaan Wikipedian artikkelien välisiä yhteyksiä [https://www.lassesimonen.fi/wikiExplorer]. Tässä ”wikipelissä” noudettiin Wikimedia Commonsin avoimen API:n kautta wikipedia-artikkelien sisältämiä linkkejä, joista muodostettiin audiovisuaalinen solmujen ja yhteyksien verkosto. Tämän verkkorakenteen visuaalinen esitystapa ja sen interaktiivisuus herättivät inspiraation soveltaa samantyyppistä periaatetta pelillisempään ympäristöön. Samalla ”wikipeli” synnytti assosiaation klassisiin reaaliaikaisiin strategiapeleihin, erityisesti Galcon-sarjaan 2000-luvun alkupuolelta. Tavoitteeksi asettui luoda vastaavalla ydinidealla toimiva, mutta teknisesti modernimpi ja ominaisuuksiltaan laajempi avaruusstrategiapeli.

Projektin ensisijainen tarve ei ole kaupallinen, vaan se toimii näyttötyönä ”Ohjelmistosuunnittelu ja ohjelmistokehitysprojektin hallinta” -tutkinnon osan suorittamiseksi. Näin ollen projektin olemassaolo perustellaan tarpeella demonstroida kykyä suunnitella, toteuttaa ja hallita moderni web-sovellus, joka noudattaa ammattimaisia ohjelmistokehityksen käytäntöjä. Projektin määrittely tarkentuu siis prototyypin kehittämisestä täysimittaiseksi client-server-arkkitehtuurin sovellukseksi, joka sisältää huolellisen suunnittelun, testauksen ja dokumentaation.

Taustaselvityksenä projektia varten on analysoitu olemassa olevia 4X- ja reaaliaikastrategiapelejä genren keskeisten pelimekaniikkojen ymmärtämiseksi, sekä tehty teknologista selvitystyötä soveltuvien tekniikoiden valitsemiseksi esimerkiksi pelin tekoälyn kehittämisessä.

Tämän teknologisen selvitystyön tehostamiseksi, erityisesti tekoälyarkkitehtuurin osalta, on hyödynnetty laajakielimalleja (kuten Gemini ja ChatGPT) tiedon synteesissä ja eri ratkaisumallien vertailussa. Tämä moderni lähestymistapa mahdollistaa laajemman tutkimuspohjan ja nopeuttaa uusien ratkaisujen kokeilua ja jatkuvaa parantamista, kehittäjän toimiessa aina lopullisena arkkitehtina ja päätöksentekijänä.

2 Määritelmät ja termien selitykset

Seuraavaksi määritellään projektisuunnitelmassa ja muussa dokumentaatiossa käytettävät keskeiset tekniset ja käsitteelliset termit. Määrittelyjen tarkoituksena on varmistaa yhtenäinen ymmärrys projektin arkkitehtuurista ja toteutuksessa käytettävistä teknologioista.

Selkeyden vuoksi termit on jaoteltu arkkitehtuuriin, frontendiin ja backendiin liittyviin kokonaisuuksiin.

2.1 Arkkitehtuurin ja projektin käsitteet

Termi	Kuvaus
Client-Server-arkkitehtuuri	Ohjelmistoarkkitehtuuri, jossa järjestelmä on jaettu kahteen osaan: asiakkaaseen (Client) ja palvelimeen (Server). Tässä projektissa selainpohjainen peli on asiakas, joka lähettää pyyntöjä, ja Node.js-sovellus on palvelin, joka hallinnoi pelin tilaa ja logiikkaa.
Refaktorointi (Refactoring)	Olemassa olevan koodin uudelleenjärjestely ja parantaminen ilman, että sen ulkoinen toiminta muuttuu. Tässä projektissa se tarkoittaa prototyypin yksittäisen tiedoston purkamista ja siirtämistä Client-Server-arkkitehtuuriin.
API (Ohjelmistorajapinta)	Määrittely, jonka avulla järjestelmän eri osat (tässä frontend ja backend) keskustelevat keskenään. Koostuu REST-komennoista ja WebSocket-tapahtumista.
Pelin tila (Game State)	Kattava tietorakenne, joka sisältää kaiken tiedon pelin senhetkisestä tilanteesta (pelaajat, resurssit, alusten sijainnit jne.). Projektin lopputuloksessa Pelin tilaa hallinnoidaan yksinomaan backendissä.
Entiteetti (Entity)	Yksittäinen pelin kohde tai olio, kuten tähti, alus tai pelaaja, jolla on omat ominaisuutensa ja tietonsa.

Taulukko 2. Arkkitehtuurin ja projektin käsitteet -taulukko

2.2 Frontend-teknologiat

Termi	Kuvaus

Frontend	Järjestelmän asiakas-osa, joka suoritetaan käyttäjän selaimessa. Vastaa pelin visuaalisesta esittämisestä, käyttöliittymästä ja käyttäjän komentojen lähettämisestä backendille.
HTML5	Verkkosivujen rakenteen määrittelykieli. Tässä projektissa se luo peli-ikkunan ja käyttöliittymän elementtien perustan.
CSS3	Tyylitiedostokieli, jolla määritellään verkkosivujen ja käyttöliittymän ulkoasu.
JavaScript (ESM)	Pelin frontendin pääasiallinen ohjelmointikieli. Projektissa hyödynnetään moderneja ES-moduuleja (ECMAScript Modules), jotka parantavat koodin järjestystä ja ylläpidettävyyttä sallimalla koodin jakamisen itsenäisiin osiin import- ja export-komennoilla.
Three.js	JavaScript-kirjasto, jota käytetään 3D-grafiikan, kuten tähtien, planeettojen ja alusten, renderöimiseen selaimessa. Se on pelin visuaalinen moottori.
Tone.js	JavaScript-kirjasto, jota käytetään pelin ääniefektien ja mahdollisesti musiikin tuottamiseen ja hallinnointiin.
Tween.js	JavaScript-kirjasto, jolla luodaan pehmeitä animaatioita ja siirtymiä, kuten kameran liikkeitä kohteesta toiseen.

Taulukko 3. Frontend-teknologiat -taulukko

2.3 Backend-teknologiat

Termi	Kuvaus
Backend	Järjestelmän palvelinpuolen sovellus (Node.js), joka suoritetaan erillisessä palvelinympäristössä ja joka hallinnoi pelin logiikkaa ja dataa.
Node.js	JavaScript-pohjainen palvelinteknologia, jota käytetään tämän projektin backend-sovelluksen toteuttamiseen.
Express.js	Minimalistinen Node.js-sovelluskehys (framework), jota käytetään backendissä API-rajapinnan rakentamiseen.
MongoDB	Dokumentti-orientoitunut NoSQL-tietokanta, johon kaikki pysyvä pelidata (pelaajat, tähdet, alukset) tallennetaan.
Mongoose	Kirjasto, joka helpottaa MongoDB-tietokannan käyttöä Node.js-sovelluksessa tarjoamalla tietorakenteiden mallinnuksen (schemas) ja validointityökalut.
Socket.IO	JavaScript-kirjasto, jolla toteutetaan reaaliaikainen, kaksisuuntainen viestintä frontendin ja backendin välillä. Välttämätön pelitilan välittömään päivittämiseen.

Taulukko 4. Backend-teknologiat -taulukko

3 Kohderyhmä ja hyödynsaajat

3.1 Projektin Asiakas

Vaikka kyseessä ei ole kaupallinen tilaustyö, tämän näyttötyöprojektin asiakkaana voidaan pitää oppilaitosta sekä tutkinnon osan arvioijia (opettaja ja työelämän edustaja).

Asiakkaan roolissa he asettavat projektille selkeät tavoitteet ja laatuvaatimukset (eperusteet, näyttötyön ohjeistus) ja arvioivat lopullisen tuotoksen sekä dokumentaation näiden vaatimusten täyttymisen perusteella. Projektin onnistuminen mitataan asiakkaan asettamien kriteerien kautta.

3.2 Pelin Kohderyhmä

Itse valmiin pelisovelluksen kohderyhmää ovat strategiapeleistä – erityisesti 4X- (eXplore, eXpand, eXploit, eXterminate) ja reaaliaikastrategiapeleistä – kiinnostuneet harrastajat. Peli julkaistaan itch.io-alustalla, joka on tunnettu indie-pelien yhteisö, joten kohderyhmä on tottunut kokeilemaan uusia ja kokeellisia pelikonsepteja. Pelin mekaniikat ja käyttöliittymä suunnitellaan niin, että ne ovat omaksuttavissa genreen perehtyneille pelaajille.

3.3 Projektin Hyödynsaajat

Projektista hyötyvät useat eri osapuolet:

Kehittäjä itse: Merkittävin hyödynsaaja on projektin toteuttaja itse. Projekti syventää osaamista keskeisissä web-teknologioissa (Node.js, Three.js, MongoDB), tarjoaa laajan ja näyttävän portfolio-työn tulevaisuuden työnhakua varten ja on edellytys tutkinnon osan hyväksytylle suorittamiselle.

Oppilaitos ja arvioijat: He saavat konkreettisen ja laadukkaan näytön opiskelijan ammattitaidosta, mikä todentaa opetuksen onnistumisen ja varmistaa, että opiskelija on saavuttanut tutkinnon osalle asetetut osaamistavoitteet.

Pelaajayhteisö: itch.io-alustan käyttäjät ja muut strategiapelien ystävät saavat pelattavakseen uuden, ilmaisen ja avoimesti saatavilla olevan pelin.

Potentiaaliset työnantajat: Projekti toimii julkisena työnäytteenä, josta tulevat työnantajat voivat arvioida kehittäjän teknistä osaamista, projektinhallintataitoja ja kykyä viedä monimutkainen hanke suunnitelmasta valmiiksi tuotteeksi.

4 Projektin tavoitteet

4.1 Tavoitteet ja priorisointi

Projektin päätavoite on refaktoroida olemassa oleva peliprototyyppi ammattimaiseksi client-server-sovellukseksi ja samalla tuottaa kaikki "Ohjelmistosuunnittelu ja ohjelmistokehitysprojektin hallinta" -tutkinnon osan suorittamiseen vaadittava materiaali.

Tavoitteet on priorisoitu Must, Should, Could -menetelmän mukaisesti.

Prioriteetti 1: Must (Pakolliset, ehdottomat tavoitteet)

Nämä tavoitteet on saavutettava, jotta projekti katsotaan onnistuneeksi ja tutkinnon osa hyväksytyksi.

- **Toiminnallinen tavoite:** Client-server-arkkitehtuuri on toteutettu ja toiminnassa. Pelaajan frontendissä tekemät toiminnot (esim. rakennuskäsky) välittyvät reaaliaikaisesti backend-palvelimelle, joka käsittelee ne ja päivittää pelin tilan onnistuneesti. Tämän toimivuus todennetaan esittelytilaisuudessa.
- **Suorituskykytavoite (määrällinen):** Toteutettu arkkitehtuuri täyttää seuraavat suorituskykykriteerit:

    - **Frontend (FPS):** Pelin renderöintinopeus pysyy vähintään 30 FPS (Frames Per Second) -tasolla tyypillisessä pelitilanteessa (esim. ~100 alusta ruudulla).
    - **Backend & Verkkoyhteys (Viive):** Pelaajan komennon lähettämisen ja sen aiheuttaman visuaalisen muutoksen alkamisen välillä on enintään 200 ms (millisekunnin) viive paikallisessa kehitysympäristössä ajettaessa.

- **Määrällinen:** Kaikki tutkinnon osan vaatimat dokumentit (projektisuunnitelma, vaatimusmäärittely, kaaviot, testausraportti, käyttöohje) on laadittu ja palautettu.
- **Määrällinen:** Backend-logiikalle on toteutettu vähintään viisi (5) merkityksellistä yksikkötestiä Jest-kirjastolla.
- **Laadullinen:** Lopputulos on esiteltävissä arviointitilaisuudessa toimivana ja vakaana kokonaisuutena.

Prioriteetti 2: Should (Tärkeät tavoitteet)

Nämä tavoitteet tekevät lopputuloksesta laadukkaan ja täysipainoisen.

- **Määrällinen:** Pelin kaikki prototyypissä olleet ydinmekaniikat (resurssien keräys, rakentaminen, alusten liikkuminen, perusteellinen taistelulogiikka ja tekoälyn toimintavuoro) on siirretty onnistuneesti backend-logiikaksi.
- **Laadullinen:** Koodi on selkeästi jäsennelty, kommentoitu ja noudattaa hyviä käytäntöjä, mikä tekee siitä ylläpidettävän.

Prioriteetti 3: Could (Toissijaiset, "jos aikaa jää" -tavoitteet)

Nämä tavoitteet parantavat peliä entisestään, mutta niiden poisjäänti ei estä projektin päämäärien saavuttamista.

- **Määrällinen:** Peliin on toteutettu vähintään yksi (1) kokonaan uusi ominaisuus, jota prototyypissä ei ollut (esim. uusi rakennus, erikoisalus tai teknologiapuu).
- **Laadullinen:** Pelin käyttöliittymään on lisätty visuaalisia tehosteita ja ääniefektejä parantamaan pelikokemusta.

4.2 Projektin ulkopuolelle jäävät asiat (Rajaaminen)

Selkeä rajaus on onnistuneen projektin edellytys. Tässä projektissa ei tulla toteuttamaan seuraavia ominaisuuksia, vaikka arkkitehtuuri ne mahdollisestikin sallisi tulevaisuudessa:

- **Moninpeli:** Vaikka client-server-arkkitehtuuri on moninpelin perusedellytys, tämän projektin puitteissa toteutetaan ainoastaan yksinpeli tekoälyvastustajia vastaan.
- **Käyttäjähallinta ja pelitilanteiden tallennus:** Projekti ei sisällä käyttäjätilien luontia, sisäänkirjautumista tai useiden eri pelien tallentamista ja lataamista. Peli alkaa aina alusta.
- **Edistynyt tekoäly:** Tekoälyn tavoite on olla toimiva ja haastava vastustaja. Sen ei kuitenkaan odoteta sisältävän monimutkaista diplomatiaa, oppivia algoritmeja tai pitkän tähtäimen strategista ennakointia.
- **Kattava mobiilioptimointi:** Peli suunnitellaan ensisijaisesti työpöytäselaimelle. Vaikka ulkoasu pyritään pitämään responsiivisena, erillistä käyttöliittymää tai ohjausmekaniikkaa kosketusnäytöille ei toteuteta.

4.3 **Tavoitteiden** mittausympäristö

Projektin määrälliset suorituskykytavoitteet (FPS, viive) mitataan ja todennetaan seuraavassa ympäristössä:

Ensisijainen testausympäristö (Kehittäjän laitteisto):

- **Prosessori:** AMD Ryzen 9 7900X 12-Core Processor
- **Keskusmuisti:** 32 Gt DDR5 (6000 MT/s)
- **Näytönohjain:** NVIDIA GeForce RTX 4090 24GB
- **Selain:** Brave, Firefox, Google Chrome

Tavoiteltu minimivaatimus (arvio):

Vaikka testausta ei suoriteta laajasti eri laitteistoilla, projektin tavoitteena on olla pelattava (väh. 30 FPS) myös heikommalla, yleisellä pelikoneella. Tällaisena referenssinä pidetään laitteistoa, joka vastaa noin seuraavia komponentteja:

Prosessori: 4-ytiminen Intel Core i5 tai AMD Ryzen 3 (n. vuodelta 2018 tai uudempi)

Keskusmuisti: 8 GB

5 Tehtäväluettelo, aikataulu, vastuuhenkilöt

Projektin virallinen palautuspäivämäärä on elokuun loppuun mennessä 2025. Tehokkaan ajankäytön ja riskienhallinnan vuoksi projektille on kuitenkin asetettu tiiviimpi sisäinen tavoiteaikataulu, jonka mukaisesti valmis työ pyritään saamaan aikaan **10.8.2025 mennessä (viikko 32)**. Tämä johtuu siitä, että kehittäjän käytettävissä oleva aika vähenee merkittävästi tämän ajankohdan jälkeen.

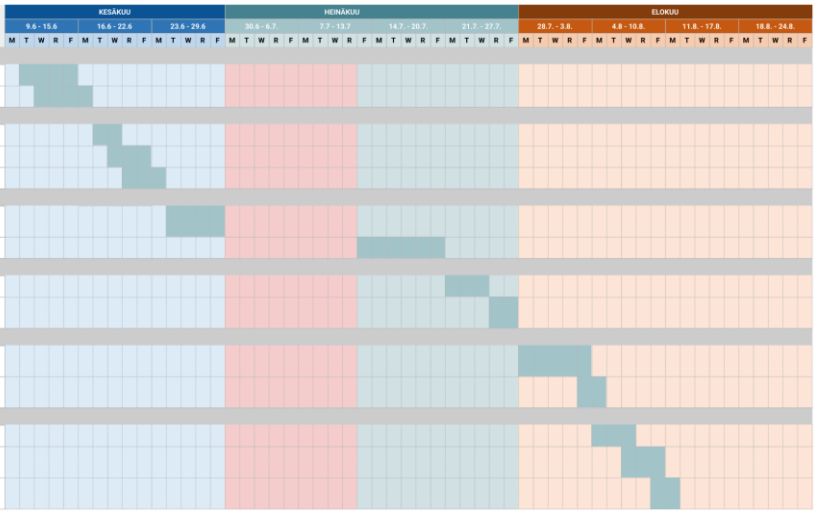
Tässä suunnitelmassa esitetty Gantt-kaavio noudattaa tätä 7 tehokkaan työviikon tavoiteaikataulua (9 kalenteriviikkoa, joista 2 viikkoa poissaoloa). Tavoiteaikataulun ja lopullisen deadlinen välinen noin kahden viikon jakso on varattu puskuriajaksi, jonka aikana voidaan korjata mahdollisia puutteita, viimeistellä dokumentaatiota ja hoitaa odottamattomia haasteita.

Kaikista tehtävistä vastaa projektin toteuttaja Lasse Simonen.

AJANKÄYTTÖSUUNNITELMA

PROJEKTIN NIMI	SpaceWar - a tiny 4X strategygame	AIKAVÄLI	11.06.2025 - 10.08.2025
VASTUUHENKILÖ	Lasse Simonen		

PROJEKTIN OSA	TEHTÄVÄ	VASTUU HENKILÖ	ALOITUSPVM	LOPETUSPVM	KESTO
1	**Suunnittelu & Määrittely**				
1,1	Projektisuunnitelman viimeistely	Lasse S.	10.6.2025	13.6.2025	3
1.1.1	Arkkitehtuurikaavioiden luonnostelu	Lasse S.	11.6.2025	16.6.2025	5
2	**Perusarkkitehtuurin Pystytys**				
2,1	Projektin alustus (kansiorakenne, Git)	Lasse S.	17.6.2025	18.6.2025	1
2,2	Backend-ympäristön ja tietokantayhteyden luonti	Lasse S.	18.6.2025	20.6.2025	2
2,3	Toimivan WebSocket-yhteyden varmistaminen	Lasse S.	21.6.2025	23.6.2025	2
3	**Ydinlogiikan Refaktorointi**				
3,1	Pelitilan ja resurssi/rakennuslogiikan siirto	Lasse S.	24.6.2025	28.6.2025	4
3,2	Liikkumis-, taistelu- ja tekoälylogiikan siirto	Lasse S.	14.7.2025	20.7.2025	6
4	**Uudet Ominaisuudet & Testauksen alustus**				
4,1	Uuden "Tähtiportti"-ominaisuuden toteutus	Lasse S.	21.7.2025	23.7.2025	2
4,2	Yksikkötestauksen alustaminen ja ensimmäisten testien kirjoittaminen	Lasse S.	24.7.2025	27.7.2025	3
5	**Testaus & Suorituskyky**				
4,1	Järjestelmällinen manuaalinen testaus ja virheiden korjaus	Lasse S.	28.7.2025	3.8.2025	5
4,2	Suorituskykymittausten (FPS, viive) suorittaminen ja dokumentointi	Lasse S.	1.8.2025	3.8.2025	2
6	**Viimeistely, Dokumentaatio & Julkaisu**				
4,1	Koodin lopullinen siivous ja kommentointi	Lasse S.	4.8.2025	6.8.2025	2
4,2	Projektidokumenttien viimeistely (testausraportti, käyttöohje)	Lasse S.	6.8.2025	8.8.2025	2
4,3	Julkaisu (Backend pilveen, Frontend itch.io:hon) ja projektin palautus	Lasse S.	9.8.2025	10.8.2025	1



Taulukko 5. Ajankäyttösuunnitelma

6 Projektiorganisaatio

Projektin toteutuksesta vastaa pääasiallisesti yksi henkilö, joka kantaa vastuun useasta roolista. Projektin ohjauksesta ja arvioinnista vastaa oppilaitoksen ja työelämän edustajista koostuva ohjausryhmä. Kehitystyön tukena hyödynnetään digitaalisia asiantuntijajärjestelmiä sekä kolmansien osapuolien palveluita.

6.1 Projektiryhmä

Projektiryhmä on yksijäseninen, ja sama henkilö vastaa kaikista projektin toteutukseen liittyvistä tehtävistä.

- **Nimi:** Lasse Simonen
- **Roolit projektissa:**
    - **Projektipäällikkö:** Vastaa aikataulutuksesta, suunnittelusta ja projektin kokonaishallinnasta.
    - **Pääkehittäjä:** Vastaa sekä frontend- että backend-koodin arkkitehtuurista ja toteutuksesta.

    - **UI/UX-suunnittelija:** Vastaa käyttöliittymän ja pelikokemuksen suunnittelusta.
    - **Testaaja:** Vastaa sovelluksen testauksesta ja laadunvarmistuksesta.

- **Yhteystiedot:** simonenlasse@gmail.com

6.2 Projektin ohjausryhmä

Projektin ohjausryhmän muodostavat näyttötyön arviointiin osallistuvat tahot. Heidän tehtävänään on ohjata projektia asettamalla sille tavoitteet ja vaatimukset (eperusteiden mukaisesti) sekä arvioida lopullinen tuotos.

Jäsen 1:

- **Nimi:** Turpeinen Jarkko
- **Rooli:** Ohjaava opettaja, arvioija

Jäsen 2:

- **Nimi:** Leivo Jari
- **Rooli:** Ohjaava opettaja, arvioija

Jäsen 3:

- **Nimi:** Nimetään myöhemmin
- **Rooli:** Työelämän edustaja, arvioija

6.3 Projektin asiantuntijajäsenet

Projektin kehitysprosessissa hyödynnetään digitaalisia asiantuntijajärjestelmiä tiedonhaun, analyysin ja teknisen sparrauksen tukena.

Asiantuntija 1: Google Gemini (Laajakielimalli)

- **Rooli:** Toimii asiantuntijana ja sparrauskumppanina mm. arkkitehtuurin suunnittelussa, tekoälytutkimuksessa ja monimutkaisten ongelmien ratkaisumallien ideoinnissa.

Asiantuntija 2: OpenAI ChatGPT (Laajakielimalli)

- **Rooli:** Toimii avustajana koodin tuottamisessa, virheiden etsinnässä, refaktoroinnissa ja dokumentaation kirjoittamisessa.

6.4 Kolmannet osapuolet

Projekti hyödyntää ulkopuolisten toimijoiden tarjoamia palveluita, teknologioita ja alustoja.

- **itch.io:** Toimii pelin frontend-sovelluksen julkaisu- ja jakelualustana loppukäyttäjille.
- **Hosting-palvelu [oletettavasti Render.com]:** Toimii projektin backend-sovelluksen ja tietokannan hostaus- eli käyttöympäristönä.
- **GitHub:** Toimii projektin lähdekoodin versionhallinnan etävarastona (remote repository) ja mahdollistaa työn varmuuskopioinnin ja jakelun.
- **NPM (Node Package Manager) -rekisteri:** Keskeinen osa Node.js-ekosysteemiä, josta projektin backendin riippuvuudet (esim. Express, Mongoose, Socket.IO) ladataan. Projekti on riippuvainen tämän kolmannen osapuolen ylläpitämän pakettirekisterin saatavuudesta.
- **Google:** Projektin suunnittelussa ja kehityksessä käytettävien työkalujen (Google Sheets Gantt-kaavio, Gemini-kielimalli) tarjoaja.
- **OpenAI:** Projektin kehityksessä käytettävien työkalujen (ChatGPT-kielimalli) tarjoaja.

7 Resurssit

Tässä luvussa arvioidaan projektin toteuttamisen kannalta oleelliset resurssit.

7.1 Henkilöresurssit

Projektin ainoa henkilöresurssi on projektin toteuttaja Lasse Simonen, joka toimii samalla projektipäällikön, kehittäjän ja testaajan rooleissa.

Projektin kokonaistyömääräksi arvioidaan noin 175 tuntia 7 viikon tehollisen työjakson aikana (noin 25 tuntia/viikko).

7.2 Laitteistoresurssit

Projektin kehitys- ja testausympäristönä toimii seuraava laitteisto:

- **Prosessori:** AMD Ryzen 9 7900X 12-Core Processor
- **Keskusmuisti:** 32 Gt DDR5 (6000 MT/s)
- **Näytönohjain:** NVIDIA GeForce RTX 4090 24GB
- **Käyttöjärjestelmä:** Windows 11

7.3 Ohjelmistoresurssit ja -menetelmät

Projektissa hyödynnetään seuraavia ohjelmistoja, kirjastoja, menetelmiä ja palveluita.

Frontend-teknologiat

- **HTML5 & CSS3:** Verkkosivun rakenteen ja ulkoasun määrittelykielet.
- **JavaScript (ESM):** Selaimessa suoritettava pelilogiikka, joka on jäsennelty moderneilla ES-moduuleilla.
- **Three.js:** WebGL-pohjainen 3D-grafiikkakirjasto pelin visuaaliseen esittämiseen. Se on pelin visuaalinen moottori.
- **Tween.js:** JavaScript-kirjasto, jolla luodaan pehmeitä, ajassa tapahtuvia animaatioita ja siirtymiä (esim. kameran sulavat liikkeet kohteesta toiseen).

- **Tone.js:** JavaScript-kirjasto, jota käytetään pelin ääniefektien ja käyttöliittymän äänipalautteen tuottamiseen ja hallinnointiin.

Backend-teknologiat

- **Node.js:** Palvelinpuolen JavaScript-ajoympäristö, joka suorittaa backend-logiikan.
- **Express.js:** Web-sovelluskehys Node.js:lle, jota käytetään API-rajapintojen ja reitityksen toteuttamiseen.
- **Socket.IO:** Kirjasto reaaliaikaisen, kaksisuuntaisen kommunikaation mahdollistamiseen selaimen ja palvelimen välillä.

Tietokanta

- **MongoDB:** NoSQL-tietokanta, jota käytetään pelin tilan, käyttäjätietojen ja muiden pysyvien tietojen tallentamiseen.
- **Mongoose:** Object Data Modeling (ODM) -kirjasto, joka helpottaa MongoDB-tietokannan käsittelyä Node.js-ympäristössä.

Testaus

- **Jest:** JavaScript-testauskehys yksikkötestien kirjoittamiseen ja suorittamiseen.
- **Yksikkötestaus (Unit Testing):** Menetelmä, jolla varmistetaan yksittäisten koodin osien (funktioiden) toimivuus eristyksessä.

Kehitysympäristö ja työkalut

- **Visual Studio Code:** Koodieditori.
- **Git:** Versionhallintajärjestelmä.
- **Google Chrome (Kehittäjän työkalut):** Selain ja sen työkalut frontendin testaamiseen ja virheenjäljitykseen.

Projektinhallinta ja dokumentaatio

- **Microsoft Word:** Projektisuunnitelman ja dokumentaation kirjoittaminen.

- **Google Sheets:** Projektin aikataulun (Gantt-kaavio) luominen ja hallinta.
- **MoSCoW-menetelmä:** Priorisointikehys (Must, Should, Could, Won't) projektin vaatimusten ja ominaisuuksien järjestämiseen.

Tekoälyavustajat

- **Google Gemini & OpenAI ChatGPT:** Hyödynnetään ideointiin, koodin generointiin ja ongelmanratkaisuun sekä dokumentaation tuottamiseen.

7.4 Taloudelliset resurssit

Vaikka projektiin ei ole varattu erillistä, uutta budjettia, sen toteutuksessa hyödynnetään olemassa olevia, maksullisia tilauspalveluita. Nämä palvelut ovat keskeisiä projektin laadukkaalle ja tehokkaalle toteutukselle.

- **OpenAI ChatGPT Plus:**
    - *Rooli projektissa:* Tehokas koodin generointi, refaktorointi, virheenjäljitys ja dokumentaation tuottaminen. ***Kuukausittainen kustannus:*** 23 €/ kk.
- **Google One (sis. Gemini Advanced):**
    - *Rooli projektissa:* Syväluotaava tutkimus, eri ratkaisumallien analyysi ja arkkitehtuurin sparraus. ***Kuukausittainen kustannus:*** 22,99 € / kk

Projektin kesto (kesäkuu-elokuu) kattaa kolme kuukautta, joten työkalujen kokonaiskustannukseksi projektin ajalta arvioidaan noin 138 €.

Muilta osin projekti ei aiheuta suoria rahallisia kustannuksia, sillä muut käytettävät ohjelmistot ja palvelut (esim. sovelluksen hostaus) perustuvat ilmaisiin versioihin.

8 Projektin kustannusarvio

Projektin kustannusarvio koostuu kahdesta pääosasta: arvioidusta työmäärästä (henkilöresurssit) ja projektin toteutukseen vaadittavista suorista kuluista.

8.1 Työkustannukset (Henkilöresurssit)

Kuten luvussa 8.1 on arvioitu, projektin kokonaistyömäärä on 175 tuntia.

Kustannusarviossa käytettävä tuntihinta perustuu Suomen markkinoilla toimivan junior-tason freelancer-ohjelmistokehittäjän tyypilliseen hintahaarukkaan (40-80 €/h). Projektissa käytetään konservatiivista arviota haarukan alapäästä, 45 €/tunti (alv 0%), mikä antaa realistisen kuvan projektin arvosta, jos se ostettaisiin ulkopuolisena palveluna.

- Laskelma: 175 tuntia * 45 €/tunti = 7875 €

8.2 Suorat kulut (Ohjelmistot ja palvelut)

Kuten luvussa 8.4 on eritelty, projektin suorat kulut muodostuvat maksullisista tilauspalveluista, joiden kokonaiskustannus projektin ajalta on noin 138 €.

8.3 Kokonaiskustannusarvio

Yhdistämällä työkustannukset ja suorat kulut saadaan projektin kokonaiskustannusarvioksi:

- Työkustannukset: 7875 €
- Suorat kulut: 138 €
- Yhteensä (alv 0%): 8013 €

On huomioitava, että tämä on arvio, ja koska projekti toteutetaan näyttötyönä ilman ulkopuolista rahoitusta, todellisia laskutettavia kustannuksia ei synny. Arvio antaa kuitenkin tärkeän viitekehyksen projektin laajuudesta ja arvosta.

9 Luottamuksellisuus, salassapito ja oikeudet työn tuloksiin

Tässä luvussa määritellään projektin tiedonhallinnan periaatteet sekä oikeudet projektin aikana tuotettuihin tuloksiin.

Vaikka projekti toteutetaan julkisena työnäytteenä ilman kaupallisia salaisuuksia, on oleellista linjata selkeästi projektin aineiston julkisuusaste, tekijänoikeuksien omistajuus ja lähdekoodin lisensointi. Nämä määritykset varmistavat läpinäkyvyyden ja selkeyttävät, miten projektin tuloksia voidaan hyödyntää tulevaisuudessa.

9.1 Luottamuksellisuus ja salassapito

Projekti ei käsittele luottamuksellista tai salassa pidettävää tietoa, kuten arkaluontoisia henkilötietoja tai liikesalaisuuksia.

Sekä projektin dokumentaatio että sen lopputuloksena syntyvä lähdekoodi ja sovellus ovat luonteeltaan julkisia. Ne toimivat tekijän julkisena työnäytteenä ja portfoliona, ja niiden on tarkoitus olla avoimesti tarkasteltavissa.

Projektin sisäisessä viestinnässä (esim. sähköpostit ohjaavan opettajan kanssa) noudatetaan normaalia ammatillista luottamuksellisuutta.

9.2 Oikeudet työn tuloksiin

Tämä kohta määrittelee, kuka omistaa projektin tulokset ja millä ehdoin muut voivat niitä hyödyntää.

Omistajuus: Tekijänoikeuslakiin perustuen kaikki oikeudet projektin aikana tuotettuun alkuperäiseen materiaaliin, mukaan lukien lähdekoodi, suunnitteludokumentit ja pelin graafinen ilme, kuuluvat projektin tekijälle, Lasse Simoselle. Oppilaitoksella on oikeus käyttää projektia ja sen tuloksia osana opetusta ja arviointia.

Lisensointi (Käyttöoikeudet muille): Jotta projektia voidaan hyödyntää avoimesti työnäytteenä ja mahdollisesti muiden opittavana materiaalina, sen osat lisensoidaan seuraavasti:

- **Lähdekoodi:** Projektin lähdekoodi julkaistaan GitHub-palvelussa avoimella MIT-lisenssillä. Tämä erittäin salliva lisenssi antaa kenelle tahansa oikeuden tarkastella, kopioida, muokata ja hyödyntää koodia sekä ei-kaupallisissa että kaupallisissa projekteissa, edellyttäen alkuperäisen tekijänoikeusmaininnan säilyttämistä.
- **Pelisovellus (itch.io):** Itch.io-alustalla jaettava suoritettava pelisovellus julkaistaan perinteisellä tekijänoikeudella (Copyright © 2025 Lasse Simonen, Kaikki oikeudet pidätetään). Käyttäjillä on oikeus pelata peliä vapaasti alustan kautta, mutta sen luvaton kopioiminen, muokkaaminen tai levittäminen on kielletty.

10 Riskit ja niiden hallinta

Projektin mahdolliset riskit on tunnistettu ja niiden hallintaan on laadittu suunnitelma. Alla on erittely keskeisimmistä riskeistä ja niiden hallintakeinoista.

10.1 Aikatauluriski

- **Riskin kuvaus:** Projekti viivästyy. Erittäin tiukan 7 viikon tehollisen työajan vuoksi jopa pieni, odottamaton haaste tai tehtävän aliarviointi voi vaarantaa koko aikataulun.
- **Hallintakeino:** 1) Tiukka priorisointi: Noudatetaan MoSCoW-mallia. "Could"- ja tarvittaessa "Should"-tason tavoitteita karsitaan, jos aikataulu alkaa venyä. 2) Säännöllinen seuranta: Edistymistä seurataan viikoittain Gantt-kaaviota vasten. 3) Pilkkominen: Suuret tehtävät on pilkottu pienempiin, hallittaviin osiin.

  Projektin tavoiteaikataulu päättyy 10.8., mutta lopullinen palautuspäivä on vasta elokuun lopussa. Tämä noin kahden viikon jakso on varattu puskuriksi, jonka aikana voidaan viimeistellä työtä ja hoitaa mahdollisia viivästyksiä ilman, että lopullinen deadline vaarantuu.

10.2 Henkilöriskit

- **Riskin kuvaus:** Projekti on riippuvainen yhdestä henkilöstä. Sairastuminen, motivaation lasku tai muu odottamaton este pysäyttää projektin kokonaan.
- **Hallintakeino:** 1) Säännöllinen versionhallinta: Koodi tallennetaan päivittäin GitHubiin, jolloin työhön on helppo palata tauonkin jälkeen. 2) Realistinen työtahti: Vältetään ylirasitusta ja pidetään kiinni suunnitellusta lomajaksosta burnoutin ehkäisemiseksi.

10.3 Taloudelliset riskit

- **Riskin kuvaus:** Projekti on riippuvainen maksullisista tilauspalveluista (ChatGPT, Google One). Hinnankorotus tai palvelun muuttuminen maksulliseksi voisi vaikuttaa budjettiin.
- **Hallintakeino:** Riski on arvioitu pieneksi. Kustannukset ovat tiedossa ja suhteellisen matalat. Projektissa voidaan tarvittaessa siirtyä käyttämään työkalujen ilmaisversioita, mikä voi hidastaa kehitystä mutta ei estä sitä.

10.4 Omaisuusvahinkoriskit

- **Riskin kuvaus:** Kehityslaitteistona toimiva tietokone rikkoutuu tai varastetaan. Tämä voisi johtaa kaiken tehdyn työn menettämiseen.
- **Hallintakeino:** Kaikki projektin lähdekoodi ja dokumentaatio tallennetaan säännöllisesti ja järjestelmällisesti ulkoisiin pilvipalveluihin (GitHub ja Google Drive). Laiterikko ei täten tuhoa itse projektia.

10.5 Tietoriskit

- **Riskin kuvaus:** Riski liittyy pääasiassa tiedon menettämiseen laiterikon tai inhimillisen virheen vuoksi. Tietovuotoriski on olematon, koska projekti ei käsittele arkaluontoista dataa.

- **Hallintakeino:** Päivittäinen ja johdonmukainen Git-versionhallinnan käyttö (git commit, git push). Tämä minimoi tiedonmenetyksen riskin.

10.6 Toiminnan vastuuriskit

- **Riskin kuvaus:** Projekti rikkoo tietämättään kolmannen osapuolen tekijänoikeuksia käyttämällä lisensoimatonta koodia, grafiikkaa tai muuta materiaalia.
- **Hallintakeino:** 1) Lisensointi: Käytetään ainoastaan avoimen lähdekoodin kirjastoja, joiden lisenssit (esim. MIT) sallivat käytön. 2) Vastuullinen tekoälyn käyttö: Projektissa hyödynnetään tekoälytyökaluja (Gemini, ChatGPT) koodin tuottamisessa. Tekoälyn generoimaa koodia ei kopioida sokeasti, vaan sitä käytetään apuna ja referenssinä. Kehittäjä vastaa aina lopullisen koodin ymmärtämisestä, testaamisesta ja muokkaamisesta omiin tarpeisiin sopivaksi, kantaen vastuun sen toiminnasta ja alkuperäisyydestä.

10.7 Uuden teknologian riskit

- **Riskin kuvaus:** Projektin keskeiset osa-alueet (backend-kehitys Node.js:llä, yksikkötestaus Jestillä, palvelinympäristön pystytys) ovat tekijälle uusia. Oppimiskäyrä voi olla jyrkempi kuin arvioitu, mikä aiheuttaa viivästyksiä.
- **Hallintakeino:** 1) Aikataulutettu opettelu: Gantt-kaaviossa on varattu erikseen aikaa uusien teknologioiden opetteluun. 2) Iteratiivinen lähestyminen: Edetään pienin, yksinkertaisin askelin ("Hello World" ensin). 3) Resurssien hyödyntäminen: Käytetään aktiivisesti dokumentaatiota, tutoriaaleja ja tekoälyavustajia ongelmien ratkaisuun.

11 Laatu

Tässä luvussa eritellään ne toimintamallit ja menetelmät, joilla varmistetaan projektin laadukas toteutus ja lopputulos.

Laadunhallinta kattaa projektin kaikki vaiheet suunnittelusta ja seurannasta aina tekniseen toteutukseen, testaukseen ja dokumentointiin asti. Tavoitteena on tuottaa paitsi toimiva sovellus, myös ammattimaisesti hallittu ja dokumentoitu projektikokonaisuus.

11.1 Seuranta ja ohjaus

Koska projektin pääasiallinen toteutusaika sijoittuu kesälomakaudelle, säännöllistä ohjausta opettajan kanssa ei ole saatavilla. Tämän vuoksi projektin seuranta ja ohjaus perustuvat vahvasti itsekuriin, ennalta laadittuun suunnitelmaan ja huolelliseen dokumentointiin.

- **Itseohjautuva seuranta:** Kehittäjä seuraa edistymistään viikoittain vertaamalla toteutuneita tehtäviä luvussa 6 esitettyyn Gantt-kaavioon. Mahdolliset poikkeamat aikataulusta analysoidaan ja niiden vaikutus arvioidaan välittömästi.

- **Projektipäiväkirja:** Projektin etenemisestä, teknisistä haasteista, tehdyistä ratkaisuista ja perustelluista poikkeamista alkuperäisestä suunnitelmasta pidetään projektipäiväkirjaa. Päiväkirja toimii todisteena järjestelmällisestä työskentelystä ja päätöksenteosta itsenäisen työskentelyjakson aikana.

- **Katselmointi:** Projekti sisältää kaksi pääkatselmointia:

    1. **Alkukatselmointi:** Tämä projektisuunnitelma hyväksytetään ohjaavalla opettajalla ennen itsenäisen työskentelyjakson alkua.

    2. **Loppukatselmointi (arviointitilaisuus):** Valmis tuote, lähdekoodi ja kaikki dokumentaatio (mukaan lukien projektipäiväkirja) esitellään ohjausryhmälle projektin päätyttyä.

11.2 Muutosten hallinta

Koska projekti toteutetaan yhden henkilön toimesta, muutoshallintaprosessi on kevyt mutta järjestelmällinen. Suurimmat muutospyynnöt tulevat todennäköisesti kehittäjältä itseltään (ns. feature creep).

Kaikki merkittävät muutosehdotukset, jotka vaikuttavat projektin laajuuteen, aikatauluun tai tavoitteisiin, käsitellään seuraavasti:

1. Muutosehdotus kirjataan ylös projektipäiväkirjaan, jossa perustellaan sen tarve ja arvioidaan vaikutukset suhteessa projektisuunnitelmaan.

2. Merkittävät, projektin laajuuteen tai päämääriin vaikuttavat muutokset siirretään odottamaan ja käsitellään yhdessä ohjaavan opettajan kanssa lomakauden jälkeen.

3. Pienet, välttämättömät tekniset muutokset, jotka eivät vaaranna aikataulua tai tavoitteita, toteutetaan ja dokumentoidaan itsenäisesti projektipäiväkirjaan.

11.3 Tiedottaminen

Koska projekti toteutetaan pääosin itsenäisenä kesätyönä, säännöllistä raportointia ei tapahdu. Tiedotus hoidetaan seuraavasti:

- **Sisäinen tiedotus:** Projektin eteneminen, haasteet ja tehdyt päätökset dokumentoidaan järjestelmällisesti projektipäiväkirjaan. Ohjaavalle opettajalle toimitetaan tilannekatsaus ennen lomakauden alkua ja välittömästi sen päätyttyä.

- **Ulkoinen tiedotus:** Projektin päätyttyä valmis pelisovellus julkaistaan itch.io-alustalla ja lähdekoodi GitHubissa. Tämä toimii julkisena tiedotteena projektin valmistumisesta ja sen tuloksista.

11.4 Menetelmät ja työkalut

Projekti noudattaa hybridimallia, jossa yhdistyvät vesiputousmallin ja ketterien menetelmien piirteet. Kokonaisuus etenee vesiputousmaisesti (Suunnittelu -> Toteutus -> Testaus -> Julkaisu),

mutta teknisessä toteutuksessa edetään ketterästi pala kerrallaan: uusia ominaisuuksia rakennetaan ja testataan pienissä, hallittavissa osissa.

Tarkempi lista käytettävistä työkaluista on eritelty luvussa 8.3 (Ohjelmistoresurssit).

11.5 Dokumentointi

Kaikki projektin dokumentit tuottaa projektin toteuttaja Lasse Simonen. Dokumentit laaditaan suomeksi ja ne tallennetaan ja jaetaan oppilaitoksen tarjoaman Microsoft OneDrive -pilvipalvelun kautta. Lopulliset versiot palautetaan oppilaitoksen määrittelemään järjestelmään PDF-muodossa.

Projektissa tuotettavat keskeiset dokumentit ovat:

- **Projektisuunnitelma (tämä dokumentti):** Määrittelee projektin puitteet, tavoitteet, aikataulun ja resurssit.
- **Vaatimusmäärittely:** Erittelee yksityiskohtaisesti sovelluksen toiminnalliset ja ei-toiminnalliset vaatimukset.
- **Suunnitteludokumentaatio (UML):** Sisältää mm. käyttötapaus-, sekvenssi- ja luokkakaaviot, jotka kuvaavat järjestelmän rakenteen ja toiminnan.
- **Testausraportti:** Dokumentoi suoritetut testitapaukset, niiden tulokset ja löydetyt virheet.
- **Käyttöohje:** Ohjeistaa loppukäyttäjää pelin pelaamisessa ja toiminnoissa.
- **Lähdekoodin dokumentaatio:** Koodi kommentoidaan selkeästi suoraan tiedostoihin (JSDoc-tyylisesti) kuvaamaan funktioiden ja monimutkaisten osien toimintaa.

11.6 Laadunvarmistus

Projektin laadunvarmistus perustuu jatkuvaan prosessiin, joka sisältää seuraavat menetelmät:

- **Koodin katselmointi:** Kehittäjä suorittaa jatkuvaa itsekatselmointia. Koodin laatua ja yhtenäistä tyyliä varmistetaan automaattisella ESLint-työkalulla.
- **Testaus:** Laadunvarmistus nojaa vahvasti testaukseen:
    - **Yksikkötestaus:** Backendin kriittisille logiikan osille kirjoitetaan yksikkötestejä Jest-kirjastolla.
    - **Manuaalinen järjestelmätestaus:** Pelin toiminnallisuutta testataan kokonaisuutena pelaamalla sitä läpi eri skenaarioissa.
- **Virheiden raportointi:** Löydetyt virheet ja bugit kirjataan ja niiden tilaa seurataan GitHub Issues -työkalulla.
- **Laatukriteerit:** Projektin laadun lopullinen mittari on luvussa 5.1 määriteltyjen tavoitteiden täyttyminen, mukaan lukien toiminnalliset vaatimukset ja asetetut suorituskykytavoitteet (FPS, viive).

12 Hyväksymismenettely

Tässä luvussa kuvataan ne periaatteet, ehdot ja kriteerit, joiden täyttyessä SpaceWar-projekti katsotaan hyväksytysti toimitetuksi ja valmiiksi. Hyväksymismenettely varmistaa, että projektin lopputulos vastaa sille asetettuja tavoitteita.

12.1 Hyväksymisen edellytykset

Ennen kuin projekti voidaan siirtää lopulliseen arviointiin (loppukatselmointiin), seuraavien edellytysten on täytyttävä:

- Kaikki luvussa 12.5 mainitut dokumentit (projektisuunnitelma, vaatimusmäärittely, suunnitteludokumentaatio, testausraportti, käyttöohje) on viimeistelty ja palautettu oppilaitoksen määrittelemään järjestelmään.
- Pelin toimiva versio on julkaistu ja käytettävissä julkisesti itch.io-alustalla.

- Projektin koko lähdekoodi, mukaan lukien JSDoc-kommentit, on saatavilla GitHub-palvelussa.
- Projektipäiväkirja on täytetty ja valmis esiteltäväksi.

12.2 Hyväksymiskriteerit

Järjestelmä ja projekti katsotaan hyväksytysti toimitetuiksi, kun projektisuunnitelman luvussa 5.1.1 määritellyt pakolliset **"Must"-tason tavoitteet** on todennetusti saavutettu. Nämä kriteerit ovat:

1. **Toiminnallinen kriteeri:** Toimiva Client-Server-arkkitehtuuri on toteutettu. Pelaajan toiminnot välittyvät reaaliaikaisesti frontendistä backendiin, joka päivittää pelin tilan onnistuneesti.
2. **Suorituskykykriteeri:** Sovellus täyttää asetetut suorituskykytavoitteet:
    - Renderöintinopeus pysyy vähintään **30 FPS** -tasolla.
    - Käyttäjän komennon ja visuaalisen muutoksen välinen viive on enintään **200 ms**.
3. **Laadullinen kriteeri:** Sovellus on vakaa ja esiteltävissä toimivana kokonaisuutena loppukatselmoinnissa.
4. **Testauskriteeri:** Backend-logiikalle on toteutettu vähintään **viisi (5) merkityksellistä yksikkötestiä** Jest-kirjastolla.
5. **Dokumentaatiokriteeri:** Kaikki vaaditut projektidokumentit on laadittu ja palautettu.

12.3 Hyväksymistilaisuus ja päätös

Projektin virallinen hyväksyminen tapahtuu arviointitilaisuudessa. Tässä tilaisuudessa projektin toteuttaja esittelee valmiin sovelluksen ja dokumentaation, ja demonstroi, että edellä mainitut hyväksymiskriteerit täyttyvät.

Ohjausryhmä (opettajat ja työelämän edustaja) tekee esityksen ja toimitettujen materiaalien perusteella päätöksen projektin hyväksymisestä.

Hyväksytty päätös päättää projektin ja vastaa periaatteeltaan "lupaa lähettää projektin viimeinen lasku", johtaen tutkinnon osan hyväksyttyyn suoritusmerkintään.

OSA B: Tekninen toteutus ja suunnittelu

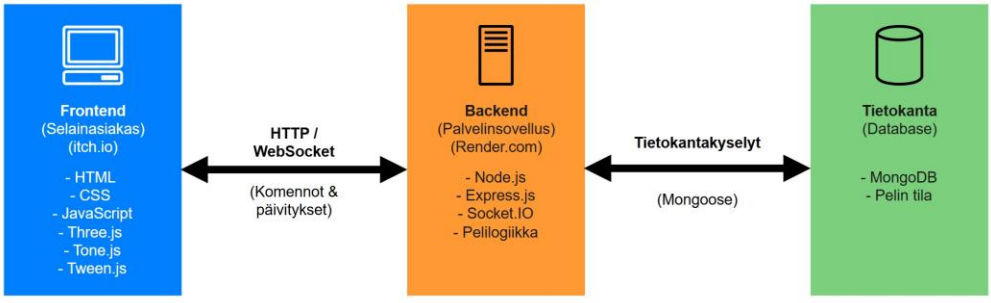
Tämä osa toimii projektin teknisenä selkärankana, joka kuvaa, miten vaatimusmäärittelyssä esitetyt tavoitteet toteutetaan käytännössä. Luvussa esitellään järjestelmän arkkitehtuuri ja pureudutaan sen rakenteeseen sekä dynaamiseen toimintaan standardisoitujen UML-kaavioiden avulla. Nämä kaaviot toimivat tulevan koodaustyön "rakennuspiirustuksina".

1 Järjestelmän arkkitehtuuri

Järjestelmän arkkitehtuuri määrittelee sen rakenteellisen perustan ja pääkomponenttien välisen työnjaon. Tässä luvussa kuvataan SpaceWar-pelin noudattama client-server-malli, joka jakaa sovelluksen selkeästi kahteen osaan: selaimessa suoritettavaan frontend-asiakkaaseen ja keskitettyä pelilogiikkaa hallinnoivaan backend-palvelimeen. Lisäksi kuvataan tietokannan rooli osana kokonaisuutta.

Seuraava alaluku esittää tämän rakenteen visuaalisesti korkean tason arkkitehtuurikaaviona, joka havainnollistaa komponenttien väliset viestintäyhteydet.

1.1 Korkean tason arkkitehtuurikaavio



Kaavion sanallinen selitys:

Yllä oleva kaavio esittää SpaceWar-pelin toteutetun client-server-arkkitehtuurin ja sen pääkomponenttien välisen vastuunjaon sekä tiedonkulun. Järjestelmä on jaettu kolmeen selkeään kerrokseen: esityskerrokseen (Frontend), sovelluslogiikan kerrokseen (Backend) ja datakerrokseen (Tietokanta).

- **Frontend (Selainasiakas):** Kaavion vasemmassa laidassa on itch.io-alustalla julkaistu selainsovellus, joka toimii pelin esityskeroksena. Sen rakenne määritellään index.html-tiedostossa, ja sen toiminnallisuus on jaettu modulaarisesti:
    - **Renderöinti (scene.js):** Hyödyntää Three.js-kirjastoa 3D-pelimaailman visualisointiin, Tone.js-kirjastoa ääniefekteihin ja Tween.js-kirjastoa animaatioihin . Tämä moduuli on puhtaasti esitystä varten eikä sisällä pelilogiikkaa.
    - **Käyttöliittymän hallinta ja sovelluslogiikka (client.js):** Vastaa kaikkien index.html-tiedostossa määriteltyjen käyttöliittymäelementtien (valikot, paneelit, napit) hallinnasta ja dynaamisesta päivittämisestä. Se käsittelee pelaajan syötteet, hallinnoi kommunikaatiota palvelimen kanssa ja toimii siltana, joka välittää pelaajan komennot eteenpäin ja käskee scene.js-moduulia päivittämään visuaalista näkymää palvelimelta saatujen tietojen perusteella.
- **Backend (Palvelinsovellus):** Kaavion keskellä on Render.com-pilvipalvelussa ajettava Node.js-sovellus, joka toimii järjestelmän aivoina ja päällikkönä. Sen pääkomponentit ovat:
    - **server.js:** Sovelluksen käynnistyspiste, joka hyödyntää Express.js-kehystä REST API -rajapinnan luomiseen (pääasiassa uusien pelien aloittamiseen) ja alustaa Socket.IO-palvelimen reaaliaikaista viestintää varten. Se hallinnoi aktiivisten GameManager-instanssien elinkaarta.
    - **GameManager.js:** Yksittäisen pelisession sydän. Jokaisella aktiivisella pelillä on oma GameManager-instanssi, joka pitää pelin tilaa muistissa ja suorittaa pelisilmukkaa (tick). Se käsittelee kaiken pelilogiikan, kuten resurssien laskennan, rakentamisen, taistelut ja tekoälyn toimintojen suorittamisen.

    - **AIController.js:** Erillinen moduuli, joka sisältää tekoälyvastustajan puhtaan päätöksentekologiikan. Se ei muokkaa pelin tilaa suoraan, vaan palauttaa GameManagerille listan suoritettavia toimintoja.

- **Tietokanta (Database):** Kaavion oikeassa laidassa on MongoDB-tietokanta, joka toimii järjestelmän pysyvänä muistina pelitilojen tallentamista varten.

    - **Mongoose:** Backend-sovellus kommunikoi tietokannan kanssa Mongoose-kirjaston avulla, joka jäsentää datan ennalta määriteltyjen skeemojen (Star.js, Ship.js, Player.js, Game.js) mukaisesti, varmistaen datan eheyden.

Komponenttien välinen viestintä:

Tiedonkulku on jaettu kahteen kanavaan:

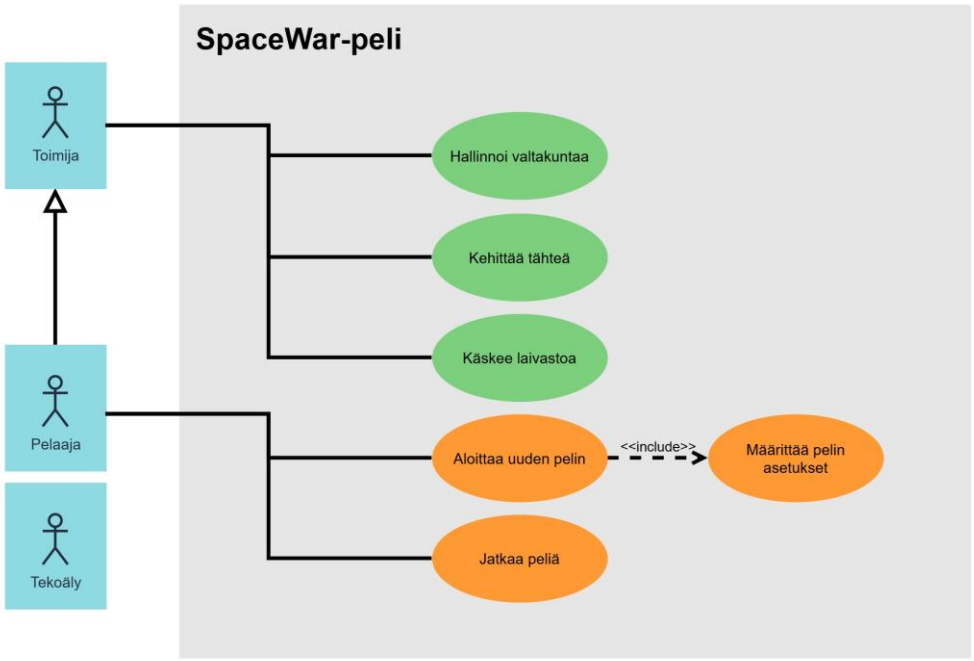
- **REST API (HTTP):** Käytetään ainoastaan uuden pelin luomiseen (POST /api/games/new). Client lähettää pelin asetukset, ja palvelin palauttaa koko pelin alkutilan (initialState).

- **Socket.IO (WebSocket):** Uuden pelin luomisen jälkeen kaikki kommunikaatio siirtyy reaaliaikaiselle WebSocket-yhteydelle.

    - **Client → Server:** Pelaajan komennot (player_command) lähetetään palvelimelle.

    - **Server → Client:** Palvelimen GameManager lähettää pelitilan muutoksista pieniä päivityspaketteja (game_diff), jotka clientin käyttöliittymä ja 3D-näkymä heijastavat välittömästi.

1.2 Käyttötapauskaavio



Kaavion sanallinen selitys:

Kaavio kuvaa SpaceWar-pelin päätoiminnallisuudet korkealla tasolla. Se määrittelee järjestelmän ulkopuoliset toimijat ja ne keskeiset tavoitteet, joita he voivat järjestelmän avulla saavuttaa. Järjestelmän raja (suuri harmaapohjainen suorakulmio) erottaa pelin sisäiset toiminnot ulkoisista toimijoista.

Toimijat (Actors)

Kaaviossa on määritelty kolme toimijaa:

- **Pelaaja:** Ihmiskäyttäjä, joka ohjaa peliä graafisen käyttöliittymän kautta.
- **Tekoäly (AI):** Tietokonevastustaja, joka suorittaa itsenäisesti pelin sisäisiä toimintoja tavoitteenaan voittaa peli.
- **Toimija (yleistetty):** Abstrakti yläluokka, joka edustaa sekä Pelaajaa että Tekoälyä. Tähän aktoriin on yhdistetty kaikki ne toiminnot, jotka ovat yhteisiä molemmille konkreettisille toimijoille.

Käyttötapaukset (Use Cases)

Käyttötapaukset on jaettu yhteisiin ja pelaajakohtaisiin toimintoihin.

Yhteiset käyttötapaukset (liittyvät Toimija-aktoriin):

- **Hallinnoi valtakuntaa:** Kuvaa toimijan kykyä tarkastella pelitilannetta, kuten omia resursseja, planeettojen tilastoja ja laivastojen sijainteja.
- **Kehittää tähteä:** Sisältää kaikki toiminnot, jotka liittyvät yhden planeetan rakentamiseen ja parantamiseen, kuten kaivosten, telakoiden ja puolustuksen rakentamisen.
- **Käske laivastoa:** Kattaa kaikki laivastonhallintaan liittyvät komennot, kuten alusten valitsemisen, siirtämisen tähtien välillä ja hyökkäyskäskyjen antamisen.

Pelaajakohtaiset käyttötapaukset:

- **Aloittaa uuden pelin:** Kuvaa prosessia, jossa pelaaja käynnistää uuden pelisession päävalikosta.
- **Jatkaa peliä:** Kuvaa toimintoa, jossa pelaaja palaa keskeytetystä pelitilanteesta takaisin peliin.
- **Määrittää pelin asetukset:** Kuvaa pelin alkuasetusten (esim. tähtien ja vastustajien määrä) valitsemista.

Suhteet (Relationships)

Kaavion viivat kuvaavat eri elementtien välisiä suhteita:

- **Assosiaatio:** Yhtenäinen viiva toimijan ja käyttötapauksen välillä kuvaa, että toimija osallistuu kyseiseen toimintoon.
- **Yleistäminen (Generalization):** Ontolla kolmiopäällä varustettu nuoli Pelaajalta ja Tekoälyltä kohti Toimija-aktoria. Se tarkoittaa, että Pelaaja ja Tekoäly ovat Toimijan erikoistapauksia ja "perivät" kaikki sen oikeudet ja toiminnot.
- **Sisällyttäminen (Include):** Katkoviiva nuolella ja <<include>>-tekstillä. Se kuvaa, että Aloittaa uuden pelin -toiminto sisältää aina pakollisena osana Määrittää pelin asetukset -toiminnon.

1.3 Käyttöliittymäsuunnitelma

Tässä luvussa esitellään sovelluksen käyttöliittymäsuunnitelma. Projektin tavoitteena ei ole käyttöliittymän uudelleensuunnittelu, vaan olemassa olevan ja toimivaksi todetun prototyypin käyttöliittymän dokumentointi.

Seuraavissa alaluvuissa esitellään pelin keskeisten näkymien **rautalankamallit (wireframes)**. Nämä mallit on luotu vastaamaan nykyistä prototyyppiä ja ne toimivat visuaalisena tukena

vaatimusmäärittelyn käyttötapauksille, havainnollistaen, miten ja missä eri toiminnot suoritetaan.

Päävalikko (startScreen)

Kuvaus:

Päävalikko on sovelluksen ensimmäinen näkymä, joka tervehtii pelaajaa. Se toimii keskitettynä pisteenä, josta uusi peli voidaan aloittaa tai aiemmin keskeytettyä peliä jatkaa. Näkymä on jaettu kolmeen pääpaneeliin, jotka tarjoavat pelaajalle kaiken tarvittavan informaation ja asetukset ennen pelin alkua.

- **Game Legend (vasen paneeli):** Tämä osio esittelee tiivistetysti pelin keskeiset säännöt ja yksiköt. Se antaa pelaajalle nopean yleiskuvan eri alustyypeistä, planeettojen kehitysmahdollisuuksista, rakennuksista ja päivityksistä. Tämän tarkoituksena on madaltaa oppimiskynnystä uusille pelaajille.

- **Game Settings (keskimmäinen paneeli):** Tämä on uuden pelin luomisen ydin. Paneelissa pelaaja voi määritellä pelisession muuttujat:

    - **Tähtien lukumäärä:** Vaikuttaa pelikartan kokoon ja pelin kestoon.

    - **Tekoälyvastustajien lukumäärä:** Määrittää vastustajien määrän (1-4).

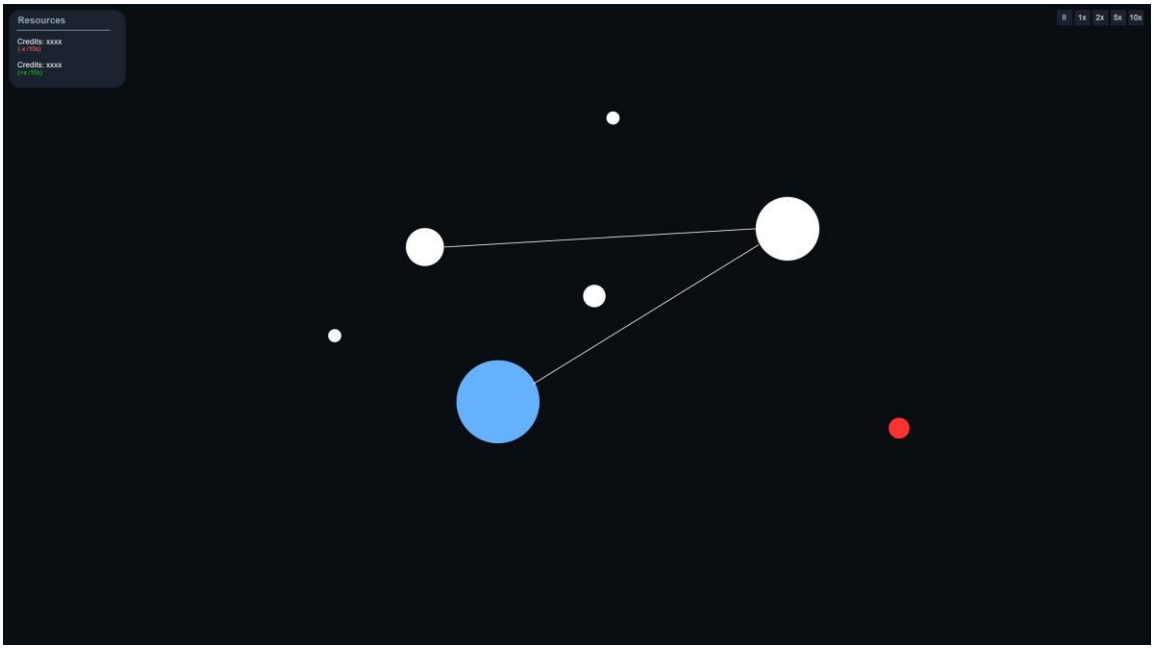
    - **Tekoälyjen värit:** Pelaaja voi kustomoida kunkin tekoälyvastustajan tunnusvärin.

- Paneelin alaosassa ovat pelin käynnistävät päätoimintonapit: **Start Game** (aloittaa uuden pelin annettujen asetusten mukaan) ja **Resume Game** (jatkaa aiemmin keskeytettyä peliä).

- **Controls Guide (oikea paneeli):** Tämä osio tarjoaa tiiviin oppaan pelin ohjaamiseen liittyvistä näppäinkomennoista ja hiiren käytöstä. Opas kattaa kameran liikuttelun, yksiköiden valinnan ja ryhmittelyn pikanäppäimillä.

Yhdessä nämä paneelit antavat pelaajalle kaikki tarvittavat työkalut ja tiedot pelisession aloittamiseen omien mieltymysten mukaisesti.

Pelin päänäkymä ja HUD (Heads-Up Display)



Kuvaus:

Pelin päänäkymä on pelaajan ensisijainen "komentosilta" ja näkymä pelimaailmaan. Se on aktiivinen, kun peli on käynnissä eikä mikään yksittäinen kohde, kuten planeetta, ole valittuna. Näkymä koostuu kolmesta pääosasta, jotka antavat pelaajalle yleiskuvan tilanteesta ja mahdollisuuden hallita pelin kulkua.

- **Pelikartta (keskiosa):** Suurin osa ruudusta on varattu pelikartalle, joka on kolmiulotteinen esitys galaksista. Pelaaja voi vapaasti liikuttaa kameraa (kiertää, panoroida, zoomata) tarkastellakseen tähtiä, laivastojen liikkeitä ja pelitilanteen

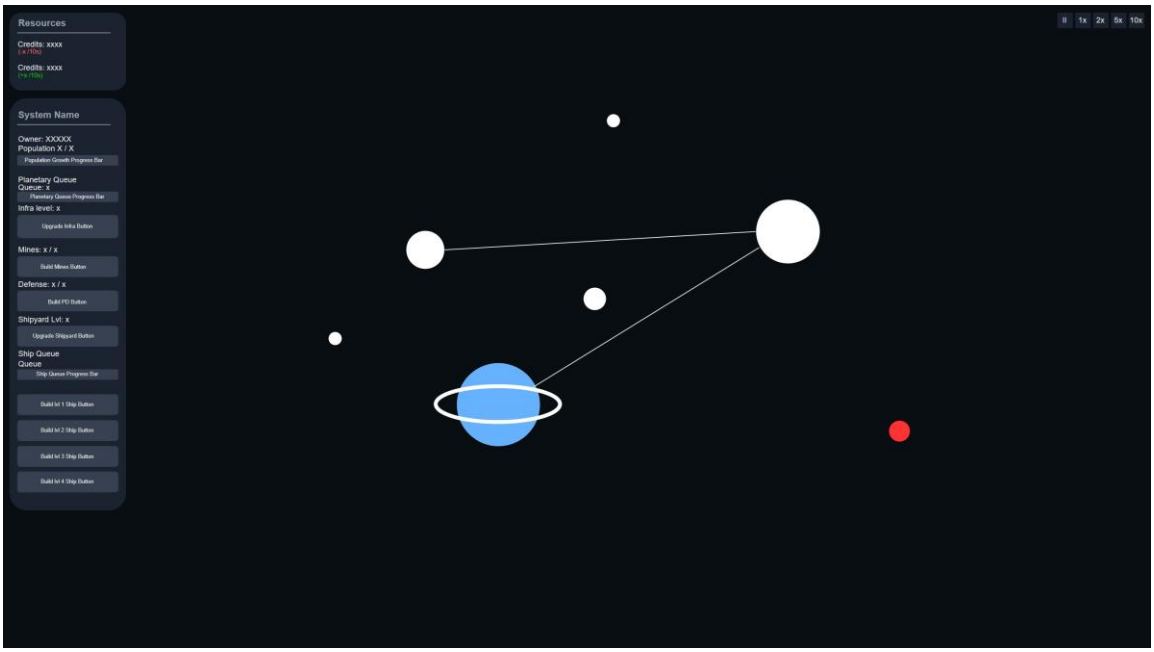
kehittymistä. Hiirellä tapahtuva yksiköiden ja kohteiden valinta tapahtuu tässä näkymässä.

- **Resurssipaneeli (vasen yläkulma):** Tämä paneeli näyttää pelaajalle reaaliaikaisen tilannekuvan hänen resursseistaan. Se kertoo nykyisen krediitti- ja mineraalimäärän sekä niiden kertymisnopeuden pelin sisäistä aikayksikköä kohden.

- **Pelin nopeuden hallintapaneeli (oikea yläkulma):** Tämä paneeli sisältää joukon nappeja, joilla pelaaja voi hallita peliajan kulkua. Pelin voi pysäyttää (pause-nappi) tai sen nopeutta voi kiihdyttää (1x, 2x, 5x, 10x), mikä on hyödyllistä esimerkiksi odotellessa rakennusprojektien valmistumista.

Tämä perustila antaa pelaajalle kaikki tarvittavat työkalut pelin yleiseen havainnointiin ja strategiseen suunnitteluun.

Planeettavalikko (Planetary Menu)



Kuvaus:

Kun pelaaja valitsee omistamansa tähden pelikartalta, käyttöliittymä siirtyy tähän näkymään. Se tarjoaa yksityiskohtaista tietoa valitusta kohteesta ja antaa pelaajalle työkalut planeetan kehittämiseen. Muutos pelin perustilaan on kolmiosainen:

1. **Visuaalinen korostus pelikartalla:**

- Valitun tähden ympärille ilmestyy visuaalinen valintaindikaattori ("tähtäin"), joka tekee selväksi, mikä kohde on aktiivisena.

- Lisäksi valitun tähden oma hehku kirkastuu, mikä antaa pelaajalle välitöntä visuaalista palautetta onnistuneesta valinnasta.

2. **Planeettavalikon ilmestyminen:**

    - Ruudun vasempaan laitaan ilmestyy planetMenu-paneeli. Se resurssipaneelin alapuolelle ja sisältää kaiken valittuun tähteen liittyvän tiedon ja toiminnot, jotka on jaoteltu selkeisiin osioihin:

        - **Perustiedot:** Tähden nimi, omistaja ja väestön tila.

        - **Planetaarinen rakennusjono:** Näyttää käynnissä olevat ja jonossa olevat rakennus- ja päivitysprojektit sekä niiden etenemisen.

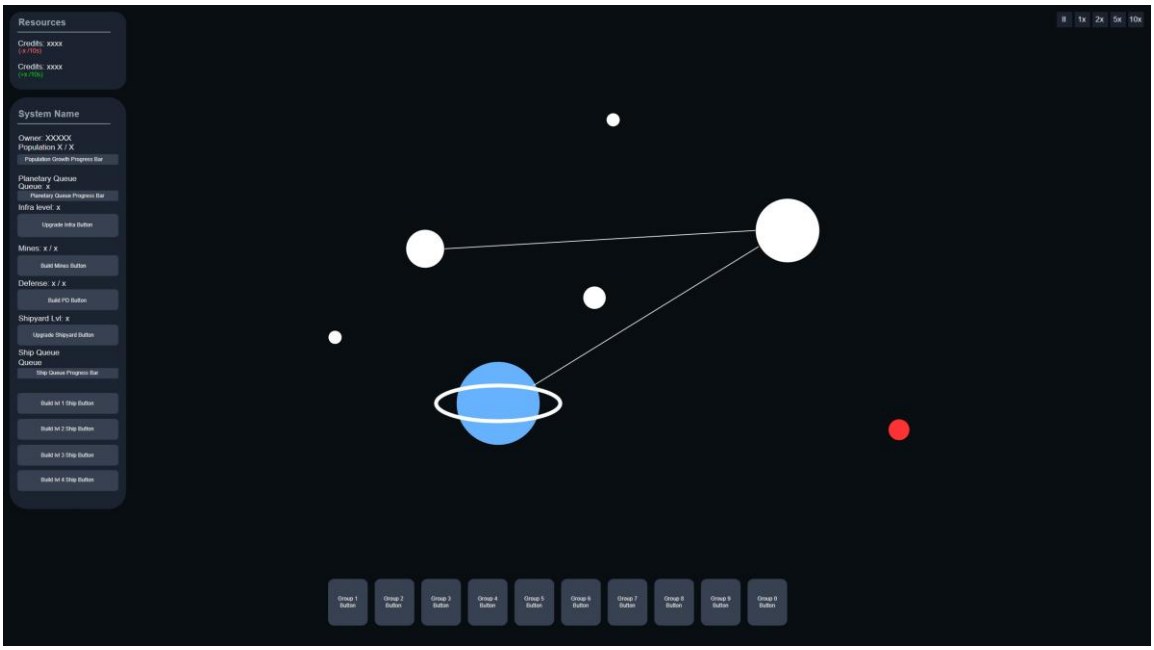
        - **Rakennusvaihtoehdot:** Painikkeet uusien kaivosten, puolustuksen, telakoiden ja infrastruktuurin rakentamiseksi.

        - **Alusten tuotantojono:** Näyttää käynnissä olevat ja jonossa olevat alusten tuotantoprojektit.

        - **Tuotantovaihtoehdot:** Painikkeet erilaisten alustyyppien (esim. Fighter, Destroyer) rakentamiseksi.

Tämä näkymä on keskeinen käyttöliittymä **Kehittää tähteä** -käyttötapaukselle, sillä kaikki siihen liittyvät toiminnot suoritetaan tämän valikon kautta.

Ryhmävalikko (groupsPanel)



Kuvaus:

Helpottaakseen suurten alusmäärien tehokasta hallintaa ja komentamista, peli sisältää kontrolliryhmä- eli "fleet"-toiminnallisuuden. Kun pelaaja luo ensimmäisen ryhmän, pelinäkymän alareunaan ilmestyy groupsPanel-valikko, joka näyttää kaikki luodut ryhmät omina painikkeinaan.

Toiminnallisuus jakautuu seuraaviin osiin:

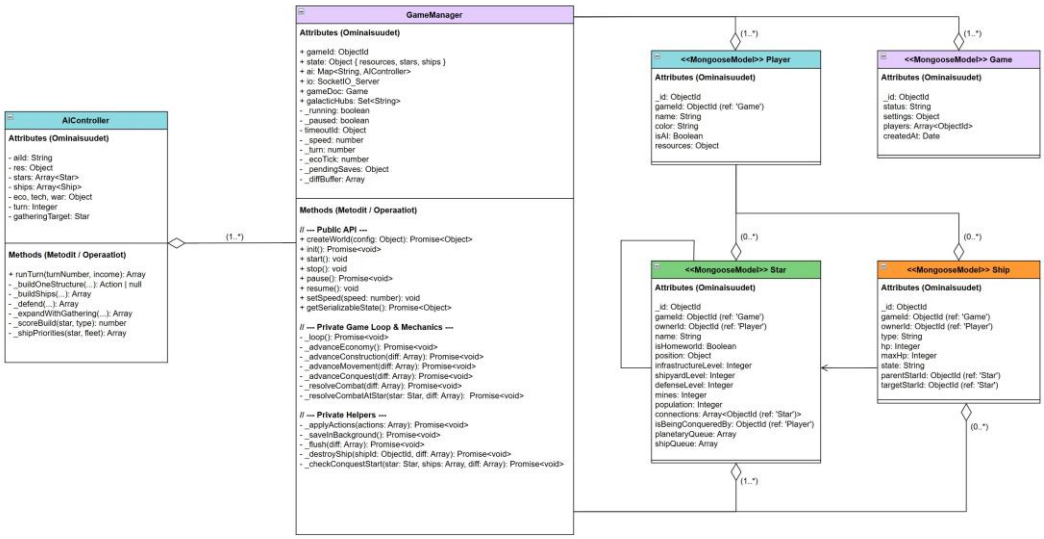
- **Ryhmän luominen ja valinta:** Pelaaja voi valita haluamansa alukset ja määrittää ne numeroryhmään Ctrl + [numero] -pikanäppäimellä. Tämän jälkeen hän voi valita kaikki ryhmään kuuluvat alukset kerralla joko klikkaamalla paneelissa olevaa ryhmäpainiketta tai painamalla vastaavaa numeroa näppäimistöltä.

- **Kameran kohdistaminen:** Nopeaa navigointia varten kamera voidaan kohdistaa välittömästi ryhmän keskimääräiseen sijaintiin. Tämä tapahtuu joko **tuplanäpäyttämällä** ryhmän numeroa näppäimistöllä tai **klikkaamalla** paneelissa olevaa ryhmäpainiketta.

- **Samanaikainen käyttö:** Kuten kuvasta nähdään, groupsPanel voi olla auki samanaikaisesti planetMenu-valikon kanssa. Tämä mahdollistaa tehokkaan pelin hallinnan, jossa pelaaja voi samanaikaisesti johtaa planeetan tuotantoa ja antaa komentoja eri puolilla karttaa oleville laivastoilleen.

Tämä käyttöliittymän osa on keskeinen työkalu **Käske laivastoa** -käyttötapauksen sujuvaan toteuttamiseen.

1.4 Luokkakaavio



Luokkakaavion sanallinen selitys

Tämä luokkakaavio esittää SpaceWar-pelin backend-sovelluksen staattisen rakenteen. Kaavio kuvaa järjestelmän keskeiset komponentit, niiden tärkeimmät ominaisuudet (attribuutit) ja toiminnot (metodit) sekä niiden väliset suhteet. Arkkitehtuuri perustuu selkeään vastuunjakoon datalähtöisten Mongoose-mallien ja ajonaikaista logiikkaa suorittavien ohjainluokkien välillä.

Pääkomponentit

Komponentit on jaettu kahteen kategoriaan: sovelluslogiikkaa sisältäviin luokkiin ja tietokannan rakennetta määritteleviin malleihin.

Loogiset ohjainluokat:

- **GameManager:** Yksittäisen pelisession keskitetty orkestroija ja "sydän". Kutakin aktiivista peliä varten luodaan yksi GameManager-instanssi, joka elää palvelimen muistissa pelin ajan. Sen vastuulla on:
    - Ajonaikaisen pelitilan (tähdet, alukset, resurssit) ylläpito muistissa nopeiden operaatioiden varmistamiseksi.
    - Pelisilmukan (_loop) pyörittäminen, joka päivittää pelin tilaa säännöllisin väliajoin ("tick").
    - Kaikkien pelimekaniikkojen, kuten talouden, rakentamisen, liikkumisen ja taisteluiden, logiikan suorittaminen.
    - Pelaajien ja tekoälyn toimintojen (actions) vastaanottaminen ja validointi.
- **AIController:** Tekoälypelaajan itsenäiset "aivot". Tämä luokka on eristetty pelin tilan suorasta muokkaamisesta, ja sen tehtävänä on:
    - Analysoida sille annettua pelitilan näkymää (gameState).
    - Tehdä strategisia päätöksiä perustuen sisäiseen logiikkaansa ja painoarvoihinsa .
    - Palauttaa GameManagerille lista suoritettavia toiminto-objekteja, jotka GameManager sitten toteuttaa.

Datan mallit (Mongoose Models):

Nämä komponentit eivät ole perinteisiä luokkia, vaan ne määrittelevät Mongoose-skeemojen avulla MongoDB-tietokannan dokumenttien rakenteen ja säännöt. Ne toimivat datan "rakennuspiirustuksina".

- **Game:** Edustaa yhtä pelisessiota ja toimii kaiken datan juuriobjektina. Kaikki muut peliin liittyvät dokumentit sisältävät gameId-viittauksen tähän, linkittäen ne yhteen.
- **Player:** Edustaa yhtä pelin osallistujaa, oli se sitten ihminen tai tekoäly. Tekoäly erotellaan yksinkertaisella isAI: Boolean -lipulla.

- **Star:** Edustaa yksittäistä tähteä tai planeettaa. Sisältää staattista tietoa (sijainti) sekä dynaamista tietoa, kuten omistajan (ownerId) ja rakennusjonot (planetaryQueue, shipQueue).

- **Ship:** Edustaa yksittäistä avaruusalusta. Sisältää aluksen perustiedot (tyyppi, HP) ja sen hetkisen tilan, kuten state ('orbiting', 'moving') ja sijaintiviittaukset (parentStarId, targetStarId).

Suhteiden kuvaus

Kaavion viivat kuvaavat komponenttien välisiä suhteita:

- **Hallinnointi ja koostumus:** GameManager-luokka hallinnoi yhteen Game-dokumenttiin liittyvää peliä. Se pitää pelisession ajan muistissaan kokoelmia Player-, Star- ja Ship-dokumenteista. Tietokannassa Game-dokumentti on looginen "kokonaisuus", joka koostuu muista dokumenteista gameId-viittauksen kautta.

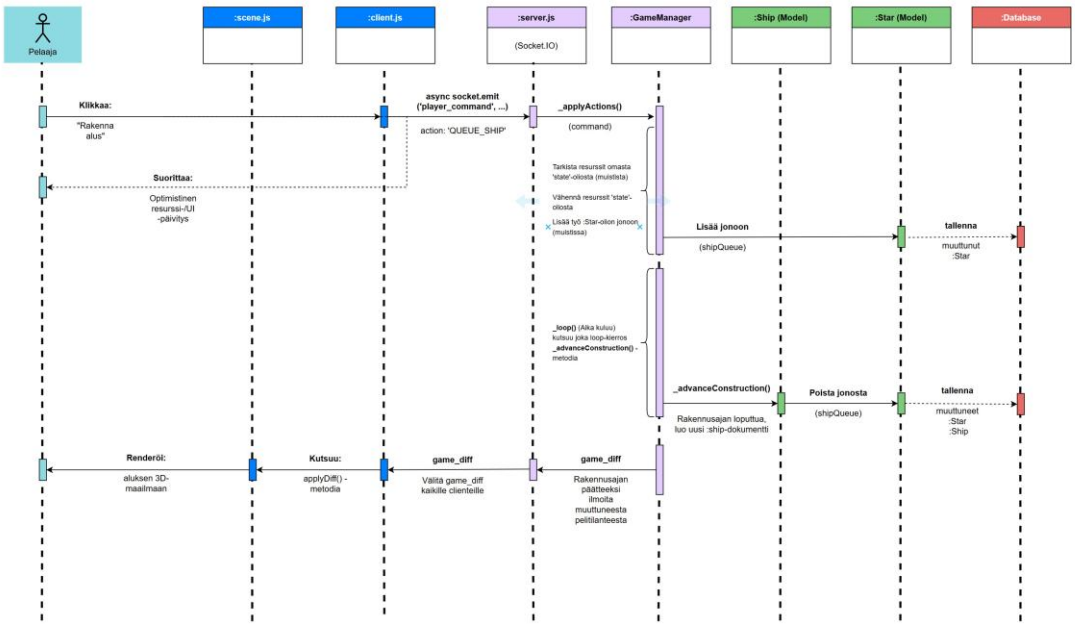
- **Assosiaatio (Käyttösuhde):**

    - GameManager luo ja käyttää yhtä AIController-instanssia kutakin Player-dokumenttia kohden, jolla on isAI: true.

    - Player-, Star- ja Ship-mallit ovat vahvasti sidoksissa toisiinsa viittausten kautta (esim. Ship-dokumentti sisältää ownerId-viittauksen Player-dokumenttiin).

- **Tekoälyn toteutus:** Perinteisen periytymismallin sijaan arkkitehtuuri erottaa tekoälyn datan ja logiikan. Player-malli isAI-lipulla kuvaa tekoälyn olemassaolon ja tilan (data), kun taas erillinen AIController-luokka toteuttaa sen käyttäytymisen (logiikka). Tämä tekee rakenteesta joustavamman ja ylläpidettävämmän.

1.5 Sekvenssikaavio: Aluksen rakentaminen



Sekvenssikaavion sanallinen selitys:

Tämä sekvenssikaavio kuvaa dynaamisesti ne tapahtumat ja **asynkroniset viestit**, jotka välittyvät järjestelmän eri osien välillä, kun pelaaja antaa käskyn rakentaa uuden aluksen. Kaavio havainnollistaa modernin client-server-arkkitehtuurin tapahtumapohjaista luonnetta, jossa client suorittaa **optimistisen päivityksen** ja palvelin toimii lopullisena auktoriteettina pelin tilan muutoksille.

Osallistujat (Lifelines)

Kaavion pystysuorat elämänviivat edustavat seuraavia prosessiin osallistuvia toimijoita ja järjestelmän komponentteja:

- **Pelaaja:** Toimija, joka käynnistää koko prosessin käyttöliittymän kautta.
- **:scene.js:** Selainpohjainen renderöintimoottori, joka vastaa lopullisen tuloksen, eli uuden 3D-aluksen, visualisoinnista pelimaailmaan.

- **:client.js:** Selainpohjainen käyttöliittymän hallintalogiikka, joka vastaanottaa pelaajan syötteen, suorittaa optimistisen päivityksen ja lähettää komennon palvelimelle.
- **:server.js (Socket.IO):** Toimii viestinvälittäjänä, joka ottaa vastaan komennon clientiltä ja reitittää sen oikealle GameManagerille.
- **:GameManager:** Backendin keskitetty ohjainolio, joka orkestroi ja hallinnoi koko rakennusprosessia.
- **:Ship (Model) & :Star (Model):** Backendin muistissa olevat Mongoose-dokumentit, jotka edustavat pelin tilaa.
- **:Database:** Tietokantajärjestelmä, joka vastaa tietojen pysyväistallennuksesta.

Tapahtumien kulku (Viestit)

Kaavion vaakasuorat nuolet kuvaavat viestien kulkua osallistujien välillä seuraavassa järjestyksessä:

1. **Pelaaja** klikkaa käyttöliittymästä "Rakenna alus" -nappia.
2. **:client.js** suorittaa välittömästi optimistisen päivityksen: se vähentää pelaajan resursseja paikallisesti käyttöliittymästä ja lisää rakennuskohteen jonoon visuaalisesti, antaen pelaajalle välitöntä palautetta.
3. **:client.js** lähettää asynkronisen socket.emit('player_command', ...) -viestin, joka sisältää action: 'QUEUE_SHIP' -komennon.
4. **:server.js** vastaanottaa komennon ja välittää sen oikealle :GameManager-instanssille _applyActions()-kutsulla.
5. **:GameManager** suorittaa auktoritatiivisen validoinnin: se tarkistaa resurssien riittävyyden omasta muistissa olevasta state-oliostaan.
6. **Hyväksynnän jälkeen :GameManager** lisää aluksen rakennustyön :Star-mallin shipQueue-jonoon ja merkitsee tähden tallennettavaksi :Database:en.
7. **Pelisilmukka (_loop)** etenee taustalla kutsuen _advanceConstruction()-metodia joka kierros.

8. **Kun riittävästi aikaa on kulunut, _advanceConstruction()-metodi** toteaa työn valmistuneen, luo uuden :Ship-dokumentin, tallentaa sen tietokantaan ja poistaa työn :Star-mallin jonosta.

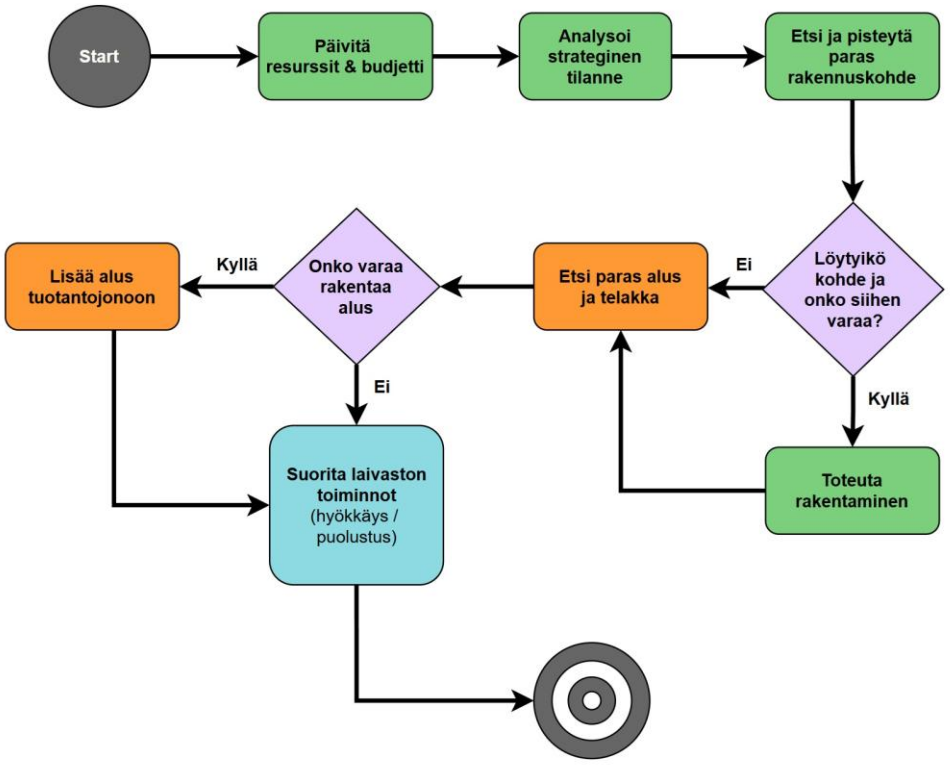
9. **:GameManager** luo game_diff-päivitysviestin, joka ilmoittaa uuden aluksen syntymisestä (SHIP_SPAWNED).

10. **:server.js** välittää game_diff-viestin kaikille pelihuoneessa oleville clienteille.

11. **:client.js** vastaanottaa game_diff-viestin ja kutsuu :scene.js:n applyDiff()-metodia.

12. **:scene.js** renderöi uuden aluksen 3D-maailmaan, jolloin Pelaaja näkee valmiin aluksen ruudulla.

1.6 Aktiviteettikaavio: tekoälyn toimintavuoro



Yleiskuvaus

Tämä aktiviteettikaavio kuvaa vuokaaviona yhden monimutkaisen prosessin: tekoälypelaajan toimintavuoron alusta loppuun. Kaavio havainnollistaa ne loogiset vaiheet ja päätöspisteet, jotka **AIController**-luokan runTurn()-metodi käy läpi joka kierroksella. Prosessin lopputuloksena on lista toiminto-objekteja, jotka palautetaan GameManagerille suoritettavaksi.

Kaavion elementit

- **Aloitus- ja Lopetussymbolit:** Musta ympyrä kuvaa prosessin alkua ja häränsilmä-symboli sen loppua.
- **Aktiviteetit (Pyöristetyt suorakulmiot):** Kuvaavat yksittäistä, suoritettavaa tehtävää, kuten "Analysoi strateginen tilanne".
- **Päätökset (Vinoneliöt/Timantit):** Kuvaavat prosessin haarautumiskohtia, joissa on tyypillisesti "Kyllä"- ja "Ei"-vaihtoehdot.
- **Siirtymät (Nuolet):** Osoittavat prosessin kulkusuunnan aktiviteetista tai päätöksestä toiseen.

Prosessin kulku

Tekoälyn toimintavuoro etenee kaavion mukaisesti seuraavassa järjestyksessä:

1. Prosessi käynnistyy, kun GameManager kutsuu **AIControllerin** runTurn()-metodia.
2. Ensimmäiseksi tekoäly päivittää resurssinsa ja jakaa edellisen kierroksen tulot eri tarkoituksiin varattuihin budjettilompakoihin (talous, teknologia, sota).
3. Tämän jälkeen se analysoi pelin kokonaistilanteen arvioiden omia ja vastustajien voimavaroja, laivastojen koostumusta ja strategisia asemia.
4. Seuraavaksi tekoäly suorittaa sarjan rinnakkaisia arviointeja kerätäkseen listan toiminnoista tälle vuorolle:

- **Planetaarinen rakentaminen:** Se etsii ja pisteyttää parhaan mahdollisen ei-sotilaallisen rakennuskohteen (_buildOneStructure). Jos strategisesti arvokas ja budjettiin sopiva kohde löytyy, se lisätään toimintolistalle.

- **Alusten tuotanto:** Riippumatta edellisestä, se arvioi laivaston rakennustarpeet kaikilla telakoillaan (_buildShips). Jos aluksia tarvitaan ja sotabudjetissa on varaa, se lisää yhden tai useamman QUEUE_SHIP-toiminnon listalle.

- **Laivaston toiminnot:** Lopuksi se arvioi laivastojensa sijainnit. Se tunnistaa uhat ja voi lisätä puolustuksellisia siirtokäskyjä (_defend) sekä etsii laajentumismahdollisuuksia ja voi lisätä hyökkäyskäskyjä (_expandWithGathering) toimintolistalle.

5. Vuoron lopuksi AIController palauttaa täydellisen listan kaikista kerätyistä toiminnoista GameManagerille, joka huolehtii niiden suorittamisesta. Prosessi päättyy End-solmuun.

OSA C: Testaus, tulokset ja loppuarviointi

Tämä dokumentin viimeinen osa esittelee projektin laadunvarmistuksen tulokset. Osiossa käydään läpi suoritetut testit, analysoidaan niiden tulokset – erityisesti suorituskyvyn osalta – ja tehdään johtopäätökset projektin onnistumisesta suhteessa projektisuunnitelmassa asetettuihin tavoitteisiin.

1 Testauksen yhteenveto

Projektin laadunvarmistus suoritettiin kattavasti yhdistämällä manuaalista järjestelmätestausta, automatisoitua yksikkötestausta ja suorituskykytestausta. Testauksen tavoitteena oli varmistaa sovelluksen toiminnallinen oikeellisuus, vakaus ja suorituskyky projektisuunnitelmassa määriteltyjen kriteerien mukaisesti.

Kaikki ydintoiminnallisuudet todettiin toimiviksi, ja asetetut suorituskykytavoitteet saavutettiin suunnitellussa "normaalissa" peliympäristössä. Suurimmilla mahdollisilla asetuksilla tehdyissä stressitesteissä havaittiin palvelinympäristöön liittyvä suorituskyvyn pullonkaula, joka on analysoitu tarkemmin luvussa 4.

2 Manuaalisen testauksen tulokset (Käyttötapaukset)

Sovelluksen käyttötapauspohjainen testaus suoritettiin manuaalisesti testaussuunnitelman (Osa A) mukaisesti. Testauksen keskiössä oli varmistaa sovelluksen ydintoimintojen oikeellisuus. Erityistä huomiota kiinnitettiin **raja-arvotestaukseen**, jolla todennettiin järjestelmän vakaus ja sääntöjenmukainen toiminta syötteiden äärirajoilla, kuten resurssien riittävyyden ja rakennusrajoitusten osalta.

Raja-arvotestaus: Pelin luominen

Testattava toiminto: Uuden pelin aloittaminen päävalikosta

Riippuvuus: Tähtien ja tekoälyvastustajien määrän valinnat

Tavoite: Varmistaa, että peli luodaan oikein sekä pienimmillä että suurimmilla sallituilla asetuksilla.

Syöte (Tähdet / Tekoälyt)	*Kuvaus*	*Odotettu tulos*
75 / 1	Raja-arvo (Alarajat)	Peli käynnistyy onnistuneesti pienimmillä mahdollisilla asetuksilla. Kartalla on 75 tähteä ja vastustajia on yksi.
500 / 4	Raja-arvo (Ylärajat)	Peli käynnistyy onnistuneesti suurimmilla mahdollisilla asetuksilla. Kartalla 500 tähteä ja vastustajia on neljä.
(Ei mahdollista UI:ssa)	Raja-arvo (Epävalidi)	Käyttöliittymän pudotusvalikot estävät epävalidien arvojen (esim. 0 tai 5 tekoälyä) syöttämisen.

Raja-arvotestaus: Resurssien käyttö

Testattava toiminto: Fighter-aluksen rakentaminen (hinta 50 krediittiä, 25 mineraalia)

Tavoite: Varmistaa, että rakentaminen on mahdollista vain, kun resurssit ovat täsmälleen riittävät tai enemmän.

Syöte (Krediitit / Mineraalit)	*Kuvaus*	*Odotettu tulos*
50 / 25	Raja-arvo (Tasan)	Rakentaminen onnistuu. Resurssit vähenevät nollaan.
49 / 25	Raja-arvo (Alle, -1)	Rakentaminen epäonnistuu. "Build"-nappi on pois käytöstä. Resurssit eivät muutu.
50 / 24	Raja-arvo (Alle, -1)	Rakentaminen epäonnistuu. "Build"-nappi on pois käytöstä. Resurssit eivät muutu.
51 / 26	Valid (Yli)	Rakentaminen onnistuu. Resurssit vähenevät (saldo: 1C, 1M).

Raja-arvotestaus: Rakennusrajoitukset

Testattava toiminto: Kaivoksen rakentaminen

Riippuvuus: Infrastruktuurin taso. Taso 1 sallii 5 kaivosta.

Tavoite: Varmistaa, että rakennusrajoitukset toimivat oikein.

Syöte (Nykyinen kaivosten määrä)	Kuvaus	**Odotettu tulos**
4	Valid (Alle rajan)	"Build Mine" -nappi on aktiivinen. Rakentaminen onnistuu.
5	Raja-arvo (Tasan rajalla)	"Build Mine" -nappi on pois käytöstä. Rakentaminen ei mahdollista.
0	Raja-arvo (Alaraja)	"Build Mine" -nappi on aktiivinen. Rakentaminen onnistuu.

2.1 Yleiset testitapaukset ja havaitut virheet

Yleisessä testaamisessa pelattiin läpi useita pelisessioita eri asetuksilla. Ydintoiminnallisuudet, kuten pelin aloitus, alusten komentaminen ja planeettojen valloitus, toimivat odotetusti.

Testauksessa havaittiin seuraavat matalan prioriteetin virheet:

Bugi 1

Virhe: Alusten valinta "jumiutuu" satunnaisesti Firefox-selaimella.

Kuvaus: Testauksen aikana havaittiin selainkohtainen virhe, jossa alusten visuaalinen valinta ei poistunut tyhjää avaruutta klikatessa, kuten sen kuuluisi. Virhe ilmeni satunnaisesti Firefox-selaimella, mutta sitä ei onnistuttu toistamaan Chrome-pohjaisilla selaimilla (Brave, Edge). Kyseessä on kosmeettinen haitta, joka ei estä pelin pelaamista.

Tila: Tiedossa.

Bugi 2

Virhe: Valloitettujen planeettojen kaivosindikaattorit eivät poistu visuaalisesti.

Kuvaus: Kun pelaaja valloittaa vihollisen planeetan, jolla on kaivoksia, vanhan omistajan kaivosindikaattorit jäävät näkyviin, vaikka kaivokset loogisesti tuhoutuvat. Virhe korjaantuu, kun planeetalle rakennetaan uusi kaivos. Populaatioindikaattorit poistuvat oikein.

Tila: Tiedossa.

3 Selainyhteensopivuustestaus

Testauksen aikana havaittiin vähäisiä eroja sovelluksen toiminnassa eri selainten välillä. Sovellus toimi vakaasti ja odotetusti Chrome-pohjaisilla selaimilla (Google Chrome, Brave, Microsoft Edge). Firefox-selaimella havaittiin seuraavat eroavaisuudet:

- **Suorituskyky:** Tutoriaali-ikkunan tekstin animointi (animateText-funktio) oli havaittavasti hitaampaa Firefoxilla, mikä johtuu todennäköisesti selainten JavaScript-moottorien välisistä eroista setInterval-ajastimen käsittelyssä.
- **Äänentoisto:** Pelin äänet toistuivat Firefoxilla hieman eri tavalla. Taustalla soiva drone-ääni oli matalampi ja bassovoittoisempi, ja lennätinääni puolestaan vaimeampi. Nämä erot johtuvat selainten Web Audio API -toteutusten pienistä eroista, eivätkä ne heikennä pelikokemusta.

Vaikka nämä eroavaisuudet on tunnistettu, ne eivät estä pelin pelaamista ja sovellus katsotaan yhteensopivaksi yleisimpien selainten kanssa.

4 Yksikkötestauksen tulokset

Projektisuunnitelman mukaisesti sovelluksen backend-logiikalle toteutettiin viisi (5) merkityksellistä yksikkötestiä käyttäen Jest-testauskehystä. Testit kattoivat sekä

AIController- että GameManager-moduulien kriittisiä, puhtaita funktioita, joilla varmistettiin laskennallisen logiikan oikeellisuus eristyksissä muusta järjestelmästä.

Testatut yksiköt ja niiden päätavoitteet olivat:

- **Tekoälyn apufunktiot:** Testattiin peruslaskentaa suorittavien funktioiden (distance3D ja shipPower) matemaattinen oikeellisuus. Nämä ovat tekoälyn jatkuvasti käyttämiä peruspilareita.

- **Tekoälyn taistelusimulaatio:** Varmistettiin starThreatScore-funktion toimivuus. Testillä todennettiin tekoälyn kyky arvioida oikein planetaarisen puolustuksen aiheuttamia tappioita hyökkäävälle laivastolle, mikä on keskeistä sen strategisille päätöksille.

- **Tekoälyn talouslogiikka:** Varmistettiin tekoälyn perustoiminto, eli kyky tarkistaa resurssien riittävyys ennen rakennuspäätöstä.

- **Pelin kinematiikka:** Testattiin GameManager-luokan _interpolatePosition-metodin matemaattinen oikeellisuus. Tämä funktio on pelin alusten sulavan liikkumisen ytimessä.

Kaikki viisi yksikkötestiä läpäisivät onnistuneesti, mikä antaa vahvan luottamuksen backend-sovelluksen ydinlogiikan vakaudelle ja oikeellisuudelle.

```
F:\Opinnot_ohjelmointi\SpaceWar_Refactored\backend>npm test

> backend@1.0.0 test
> jest

 PASS  gameLogic/test.js
  AIController Apufunktiot
    √ 1. distance3D laskee etäisyyden oikein (1 ms)
    √ 2. shipPower palauttaa oikeat voima-arvot
  AIController Taistelulogiikka
    √ 3. starThreatScore laskee tappiot oikein (1 ms)
  AIController Talouslogiikka
    √ 4. AI tunnistaa, onko sillä varaa rakentaa kaivos
  GameManager Pelimekaniikka
    √ 5. _interpolatePosition laskee sijainnin oikein

Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        0.552 s, estimated 1 s
Ran all test suites.
```

5 Suorituskykytestauksen tulokset ja analyysi

Sovelluksen suorituskykyä mitattiin sekä frontendin (renderöintinopeus, FPS) että backendin (vasteaika, latenssi) osalta.

5.1 Frontend-suorituskyky (FPS)

Pelin renderöintinopeus ylitti asetetun **30 FPS tavoitteen** kaikissa testiolosuhteissa. Raskaassa stressitestissä (500 tähteä, 4 tekoälyä, useita satoja aluksia ruudulla) suorituskyky pysyi vakaasti **59-60 FPS:ssä**. Tämä osoittaa, että Three.js-pohjainen renderöintimoottori on tehokas ja client-sovellus on hyvin optimoitu.

5.2 Backend- ja verkkosuorituskyky (Latenssi)

Sovellukseen implementoitiin reaaliaikainen ping-mittari, joka mittaa edestakaista viivettä (RTT) clientin ja Render.com-palvelimen välillä.

- **Normaali peli (150 tähteä):** Normaalilla karttakoolla pelatessa keskimääräinen vasteaika asettui **noin 100 millisekuntiin**. Tämä alittaa selvästi projektisuunnitelmassa asetetun **200 ms tavoitteen**.
- **Stressitesti (500 tähteä):** Suurimmalla, 500 tähden kartalla, havaittiin merkittäviä ja ajoittain useita sekunteja kestäviä latenssipiikkejä (esim. 380 ms, 650 ms, 1100 ms).
- **Juurisyy-analyysi:** Analyysi osoitti, että latenssipiikit korreloivat suoraan palvelimen suoritinkäytön kanssa. Render.comin tarjoamat metriikat vahvistivat, että palvelimen jaettu **0.5x suoritin oli 500 tähden kartalla jatkuvasti 100% kuormituksessa**. Tämä johtuu erityisesti neljän tekoälypelaajan vaativasta laskennasta, joka suoritetaan jokaisella pelitickillä. Client-sovelluksen suorituskyky (FPS) pysyi vakaana näiden palvelinpuolen piikkien aikana, mikä vahvistaa ongelman olevan palvelimen resurssien riittämättömyys äärimmäisessä kuormituksessa.

Johtopäätös: Sovellus saavutti ja ylitti projektisuunnitelmassa asetetut suorituskykytavoitteet tyypillisissä peliolosuhteissa (150 tähteä). Äärimmäisessä stressitestissä (500 tähteä, 4 tekoälyä) havaittiin kuitenkin merkittävä skaalautuvuuden haaste, joka ilmeni korkeina latenssipiikkeinä.

Vaikka sovellusta voi aina optimoida, analyysi osoitti, että kyseessä on ensisijaisesti **infrastruktuurin pullonkaula** eikä niinkään algoritminen virhe sovelluksessa. Palvelimen jaettu 0.5x suoritin ei kyennyt vastaamaan usean tekoäly-instanssin samanaikaiseen, raskaaseen laskentakuormaan suurimmalla kartalla.

Näin ollen voidaan todeta, että sovellus itsessään on onnistunut ja täyttää sille asetetut vaatimukset, mutta sen skaalautuvuus vaatisi tehokkaamman palvelinympäristön.

6 Yhteenveto ja johtopäätökset

Projekti oli kokonaisvaltainen menestys, ja se saavutti kaikki projektisuunnitelmassa määritellyt tavoitteet – ylittäen ne monin osin.

Kaikki pakolliset **("Must")** -tason tavoitteet täyttyivät: olemassa oleva prototyyppi refaktoroitiin onnistuneesti toimivaksi client-server-arkkitehtuuriksi, vaaditut dokumentit tuotettiin ja laadunvarmistus suoritettiin suunnitelman mukaisesti. Myös tärkeät **("Should")** -tason tavoitteet saavutettiin täysin. Kaikki alkuperäisen prototyypin ydinmekaniikat siirrettiin onnistuneesti backend-logiikaksi, ja tekoälyä kehitettiin merkittävästi analysoimaan aktiivisesti vihollislaivastoja ja rakentamaan niille strategisia vastayksiköitä. Koodin laatuun kiinnitettiin erityistä huomiota, ja koko koodikanta on selkeästi jäsennelty ja kattavasti kommentoitu.

Erityisen onnistunut projekti oli toissijaisten **("Could")** -tavoitteiden osalta, jotka ylitettiin huomattavasti. Vaaditun yhden uuden ominaisuuden sijaan peliin toteutettiin suunnitelman mukaisen "Galactic Hub" -rakennuksen lisäksi laaja, tapahtumapohjainen tutoriaalijärjestelmä. Tämä järjestelmä parantaa pelin immersiota ja opastaa pelaajaa dynaamisesti reagoivien avustajahahmojen kautta, rikastaen pelikokemusta visuaalisilla ja äänitehosteilla.

Suorituskykytestaus osoitti, että sovellus on pelattava ja toimii tavoitteiden mukaisesti suunnitellussa käyttöympäristössä. Samalla testaus tuotti arvokasta tietoa järjestelmän

skaalautuvuudesta ja tunnisti palvelinresurssien merkityksen pelikokemukselle äärimmäisissä olosuhteissa.

Vaikka sovellus täyttää asetetut suorituskykytavoitteet yhden pelaajan osalta normaalilla karttakoolla, testauksen ja analyysin syventyessä ilmeni merkittävä skaalautuvuuden haaste nykyisessä palvelinympäristössä. Arkkitehtuuri, jossa jokainen pelisessio luo oman laskentaintensiivisen `GameManager`-instanssin, kuormittaa jaettua 0.5x suoritinta huomattavasti. Jo kaksi samanaikaista, suurilla asetuksilla (500 tähteä, 4 tekoälyä) pelattavaa peliä ylittää palvelimen kapasiteetin, mikä johtaa merkittäviin latenssiongelmiin.

Tämä havainto vahvistaa, että moninpelio minaisuuksien lisääminen tai useiden samanaikaisten yksinpelien tukeminen vaatisi ehdottomasti siirtymistä tehokkaampaan palvelinarkkitehtuuriin, kuten "Kehitysideoita tulevaisuuteen" -luvussa esitettyyn dedikoituun tai itsehostattuun palvelimeen, joka takaisi sovellukselle tarvittavat suorituskykyresurssit.

7 Kehitysideoita tulevaisuuteen

Testauksen ja analyysin perusteella tunnistettiin seuraavat jatkokehityspolut sovelluksen skaalautuvuuden ja pitkän tähtäimen elinkaaren varmistamiseksi:

1. Infrastruktuurin skaalaus (Lyhyen tähtäimen ratkaisu): Helpoin välitön ratkaisu testauksessa havaittuun suorituskyvyn pullonkaulaan olisi päivittää Render.com-palvelin tehokkaampaan versioon (esim. 1x tai 2x CPU). Tämä todennäköisesti poistaisi latenssipiikit suurilla kartoilla ja parantaisi pelikokemusta. Palvelimien ylläpidon kustannukset pelin suosion kasvaessa saattavat kuitenkin johtaa siihen, ettei ratkaisu ole kestävä pitkällä aikavälillä.

2. Koodin optimointi: Backend-logiikkaa, erityisesti tekoälyn AIController-luokan silmukoita ja taistelunratkaisun _resolveCombat-metodia, voitaisiin optimoida entisestään vähentämään suoritinkuormaa per pelitick. Tämä parantaisi sovelluksen tehokkuutta ja voisi mahdollistaa suurempien pelimaailmojen sujuvan toiminnan myös edullisemmalla palvelinratkaisulla.

3. Pelaajaisännöity P2P-arkkitehtuuri (Pitkän tähtäimen visio): Kestävän ja kustannustehokkaan moninpelin mahdollistamiseksi tulevaisuudessa tulisi harkita

siirtymistä arkkitehtuuriin, jossa yksi pelaajista isännöi (hostaa) pelipalvelinta omalla koneellaan.

- Toteutus: Nykyinen Node.js-backend-sovellus voitaisiin paketoida itsenäiseksi, ladattavaksi ohjelmaksi (esim. Electron- tai pkg-työkaluilla). Pelaajat voisivat ladata tämän palvelinsovelluksen ja käynnistää omia pelejään.

- Hyödyt: Tämä malli ratkaisisi suoraan suorituskyvyn ja kustannusten haasteet, sillä raskas pelilogiikan laskenta siirtyisi kehittäjän keskitetyltä palvelimelta pelaajien omille, tehokkaammille tietokoneille. Vastuu laitteiston riittävyydestä siirtyisi pelaajalle, mikä on yleinen malli PC-peleissä.

- Haasteet: Tämä lähestymistapa vaatii pelaajilta enemmän teknistä osaamista, kuten portinohjausten (port forwarding) tekemistä reitittimeen, jotta muut pelaajat voivat liittyä peliin. Yhteydenmuodostuksen helpottamiseksi tarvittaisiin todennäköisesti edelleen kevyt, keskitetty "matchmaking"-palvelin, joka ainoastaan listaa aktiivisia pelejä.

Tämä pelaajaisännöity malli on strategisesti kestävin polku pelin tulevaisuudelle, erityisesti moninpeliä ajatellen.