Reima Nikola

Ohjelmistokehityspaketin dokumentointi selkeäksi ja myyväksi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

5.6.2025

Tiivistelmä

Tekijä:	Reima Nikola
Otsikko:	Ohjelmistokehityspaketin dokumentointi selkeäksi ja myyväksi
Sivumäärä:	35 sivua
Aika:	5.6.2025
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Pelisovellukset
Ohjaajat:	Lehtori Antti Laiho Teknologiajohtaja Tero Paavolainen

---

Tämän insinöörityön tarkoitus oli laatia Playnest-nimisen mobiilipelialustan ohjelmistokehityspaketille tekninen dokumentaatio. Tavoitteena oli tehdä dokumentaatiosta selkeä ja helppo käyttää. Lisäksi dokumentaation tuli olla myyvän näköinen.

Osa aineistosta dokumentaatiota varten saatiin olemassaolevista dokumenteista ja loput tuotettiin tutkimalla koodikantaa ja kirjoittamalla uutta tekstiä. Dokumentaation tueksi kehitettiin Playnest-alustalla toimiva esimerkkipeli. Esimerkkipeli jaetaan kehittäjille ohjelmistokehityspaketin mukana.

Dokumentaatiota varten rakennettiin ja julkaistiin verkkosivu Notion-projektinhallintatyökalulla. Dokumentaation versionhallinta oli sisäänrakennettu Notion-työkaluun. Esimerkkipeli toteutettiin Unity-pelimoottorilla hyödyntäen Playnest SDK-ohjelmistokehityspakettia. Pelin versionhallintaan käytettiin GitHub-palvelua ja omaa Git-haaraa työn tilaajan kuvauskannassa.

Insinöörityö valmistui tilaajan asettaman aikataulun puitteissa ja täytti vaaditut kriteerit. Työtä varten tutkittiin hyvän teknisen dokumentaation periaatteita. Samoja periaatteita voidaan hyödyntää myös muissa vastaavissa dokumentaatioissa. Valmis työ on luettavissa Playnestin verkkosivuilla osoitteessa https://playnest.notion.site/playnest-docs.

Avainsanat:	dokumentaatio, Notion, Unity, pelikehitys, ohjelmistokehityspaketti

---

Abstract

Author:	Reima Nikola
Title:	Creating a clear and commercially appealing software development kit documentation
Number of Pages:	35 pages
Date:	5 June 2025
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Game Applications
Supervisors:	Antti Laiho, Senior Lecturer Tero Paavolainen, Chief Technology Officer

---

This final year project was a documentation project for a mobile game development platform called Playnest. Playnest needed technical documentation for their software development kit. Making the documentation easy to use and sell was the primary objective.

The documentation was partially based on pre-existing documents and partially produced by studying the codebase and writing new text. To support the documentation, a sample game that integrated into the Playnest platform was also developed. The complete example game project is distributed to external game developers along with the software development kit.

In addition, a website for the documentation was built and published using Notion which is a project management tool. Notion has a built-in version control. The sample game was made using the Unity game engine and the Playnest SDK software development kit. Version control for the sample game was managed using GitHub services and a separate Git branch within the client's repository.

The project met all the criteria and was finished within the schedule set by the client. Numerous good practices for making technical documentation were studied for this project. Most of these practices can be utilized in other documentation projects as well. The complete documentation is available on Playnest's website at https://playnest.notion.site/playnest-docs.

Keywords: documentation, Notion, Unity, game development, software development kit

Sisällys

Lyhenteet ja termit

Idiomi: Tietylle kielelle tyypillinen vakiintunut ilmaus. Niiden sanojen merkityksistä, joista idiomi koostuu ei voi päätellä koko ilmauksen merkitystä. Suomen kielessä esimerkki idiomista on lause "Lyödään viisaat päämme yhteen." Lause tarkoittaa jonkin asian miettimistä yhdessä.

Kognitiiviset taidot:

Aivojen kyky suoriutua ajattelua vaativista toiminnoista. Tarvitaan esimerkiksi luetun tekstin ymmärtämiseen ja opitun tiedon yhdistämiseen osaksi ohjelmistokokonaisuutta.

WCAG: *Web Content Accessibility Guidelines*. Joint Technical Committee ISO/IEC JTC1:n laatimat verkkosisällön saavutettavuuden ohjesäännöt verkkosisällön tuottajille ja kehittäjille.

SDK: *Software Development Kit*. Ohjelmistokehityspaketti. Kokoelma ohjelmistokehityksessä tarvittavia työkaluja.

Kuvauskanta:

Hakemisto, jonka tiedostomuutokset versioidaan ja joka on tavallisesti saatavilla ohjelmistokehitystiimin jäsenille verkon välityksellä.

Git-haara: Git-pohjaisen kuvauskannan tietystä versiosta haarautuva kehityspolku, jolle luodaan oma versiohistoria.

1 Johdanto

Ohjelmistoja luotaessa syntyy monia kooditiedostoja. On tärkeää tehdä koodista helposti luettavaa ja lisätä kommentteja tarpeen mukaan. Pelkkä koodin luettavuus ei kuitenkaan riitä, sillä suurissa ohjelmistoissa kooditiedostoja on vähintään satoja, jopa tuhansia. Kun projektin koko kasvaa, jopa ohjelmiston kehityksessä alusta asti mukana olleet ohjelmoijat tarvitsevat oikeiden tiedostojen löytämiseen ja ohjelmiston rakenteen ymmärtämiseen kattavaa ja selkeää dokumentaatiota.

Tämän insinöörityön tarkoituksena on laatia sellainen dokumentaatio, josta tilaajaorganisaation ulkopuoliset pelikehittäjät voivat löytää tarvitsemansa tiedot helposti ja päästä nopeasti omassa pelikehitystyössään eteenpäin. Työssä keskitytään erityisesti ohjelmistoalan dokumentaation luomiseen, mutta useista työn aikana ilmi tulevista hyvistä käytänteistä ja toimintamalleista voi ottaa ajatuksia muidenkin alojen dokumentaatioiden suunnitteluun ja toteutukseen. Dokumentaation tueksi kehitetään esimerkkipeli. Esimerkkipeliin toteutetaan pelialustan tärkeimmät ominaisuudet. Pelikehittäjät voivat tutkia esimerkkiprojektia ja jopa kopioida suoraan osia koodista. Työn tilaaja on Playnest Oy.

Insinöörityön toisessa luvussa perehdytään siihen, mistä hyvätasoinen ohjelmistoalan tekninen dokumentaatio koostuu. Luvussa tutkitaan millaista tietoa dokumentaation tulisi sisältää ja missä muodossa tieto esitetään kohdeyleisölle. Lisäksi selvitetään, miten tiedot olisi hyvä jäsennellä, jotta lukija suoriutuisi tiedonhausta mahdollisimman tehokkaasti.

Työn käsittelyosuudessa suunnitellaan dokumentaatiota, asetetaan työn tavoitteet ja valitaan työskentelymenetelmät. Lisäksi kartoitetaan työn mahdollisia ongelmakohtia ja riskejä ennen niiden ilmenemistä, jotta niitä voitaisiin välttää. Lopuksi tarkastellaan tehdyn työn tuloksia ja pohditaan työn jatkokehitystä ja ylläpitoa.

2 Ohjelmiston tekninen dokumentaatio

Ohjelmistokehityksessä tekniseksi dokumentaatioksi kutsutaan sellaista tieto- ja ohjekokoelmaa, joka keskittyy yksityiskohtaisesti ohjelmiston tekniseen puoleen. Usein tekninen dokumentaatio on suunnattu aiheen hyvin tunteville tai alan ammattilaisille. Tekninen dokumentaatio eroaa ohjelmiston loppukäyttäjille tehdystä käyttäjädokumentaatiosta menemällä syvemmälle ohjelmiston rakenteeseen ja sisältämällä teknisempää kieltä. (1; 2, 1:00–3:35.)

2.1 Dokumentaation sisältö ja rakenne

Tekninen dokumentaatio sisältää yleensä tietoa ohjelmiston kyvyistä, ominaisuuksista, käytetyistä tietorakenteista ja algoritmeista (1). Tiedon välittämisessä käytetään paljon kirjoitettua tekstiä. Kirjoitetun tekstin lisäksi dokumentaatioon kannattaa sisällyttää havainnollistavia kuvia, koodiesimerkkejä, videoita ja kaavioita. Ne auttavat lukijaa ymmärtämään, oppimaan ja muistamaan tietoa. Tietoa etsiessä havainnollistukset myös kiinnittävät lukijan huomion kirjoitettua tekstiä paremmin ja pitävät lukijan mielenkiinnon yllä. (3; 4, s. 58–64.) Havainnollistavat elementit, kuten kuvat ja kaaviot, on tärkeää sijoitella asiaa käsittelevän tekstin välittömään yhteyteen, jotta kirjoitetun tekstin ja havainnollistuksen yhteys on selkeä.

Dokumentaation aihealue on aina rajattu, joten artikkeleiden teksteissä tarvitaan viittauksia muihin dokumentaatioihin tai julkaisuihin. Ohjelmistoalan kaikki dokumentaatiot ovat nykyään sähköisiä, joten hyperlinkit tekstin lomassa ovat yleinen tapa viitata muihin teksteihin.

Dokumentaation rakenteen on oltava yhtenäinen (1). Lukija oppii dokumentaatiota lukiessaan tietyn jäsentelytavan tiedolle. Jos rakennetta rikotaan, se turhauttaa ja hämmentää lukijaa. (5.)

Selkeyden vuoksi ohjelmiston korkean tason käsitteitä ja matalan tason tietoa käsittelevät artikkelit on syytä jakaa eri osioihin. Korkean tason käsitteitä ovat esimerkiksi käyttöliittymä ja ohjelmiston kyvyt. Koodin luokat ja algoritmit kuuluvat matalan tason käsitteisiin. Esimerkiksi Unity-pelimoottorin julkinen tekninen dokumentaatio on jaettu edellä mainitulla tavalla Manual-osioon, joka keskittyy graafiseen käyttöliittymään ja Scripting API -osioon, joka keskittyy pelimoottorin koodikantaan ja sen luokkien rajapintojen hyödyntämiseen (6).

2.2 Teknisen dokumentaation tekstin tyyli

Kirjoitetun tekstin on oltava asiatyylistä ja varsinkin kansainväliseen käyttöön tulevissa teksteissä on selkeyden takaamiseksi vältettävä puhekielisiä ilmaisuja tai idiomeja (1). Idiomi on tietylle kielelle tyypillinen vakiintunut ilmaus. Niiden sanojen merkityksistä, joista idiomi koostuu, ei voi päätellä koko ilmauksen merkitystä. Suomen kielessä esimerkki idiomista on lause "Lyödään viisaat päämme yhteen." Lause tarkoittaa jonkin asian miettimistä yhdessä. (7.) Lauseiden kannattaa muutenkin olla lyhyitä ja ytimekkäitä, sillä pitkiä ja monimutkaisia lauserakenteita on lyhyitä raskaampaa lukea ja hitaampaa tulkita (3).

Myös tavalla, jolla lukijaa puhutellaan, on merkitystä. Tekstissä kannattaa suosia lukijan puhuttelua aktiivisella sijamuodolla passiivin käyttämisen sijaan (5). Kieltosanoja on myös syytä välttää. Esimerkkinä voidaan tarkastella seuraavia virkkeitä:

- Kun luku 4 on luettu, voidaan siirtyä suunnittelemaan luokan rakennetta.
- Et voi suunnitella luokan rakennetta ennen kuin olet lukenut luvun 4.
- Lue luku 4. Siirry sitten suunnittelemaan luokan rakennetta.

Ensimmäinen virke passivoi lukijaa ja saa prosessin kuulostamaan työläältä ja tylsältä. Toinen virke aiheuttaa negatiivisen tunteen lukijalle. Kolmannessa virkkeessä lukijaa kehotetaan toimintaan, ja prosessin eri vaiheet ovat selkeässä järjestyksessä.

2.3 Dokumentaation saavutettavuus

Saavutettavuus ei ole itsestäänselvyys. Dokumentaatioita laatiessa saattaa unohtua, että monilla lukijoilla on haasteita joko näkökyvyn, värien hahmottamisen tai kuulon kanssa. Lukijoiden kognitiivisissa taidoissa on myös väistämättä eroavaisuuksia. Kognitiivisilla taidoilla tarkoitetaan tässä yhteydessä lukijan kykyä ymmärtää luettu teksti ja yhdistää opittu tieto osaksi ohjelmiston kokonaisuutta. Jotta teksti olisi mahdollisimman saavutettava, edellä mainittujen seikkojen huomioon ottamiseen on syytä käyttää aikaa. Muun muassa videoissa tulisi olla tekstitykset kuulorajoitteisia varten.

Näkörajoitteisille on tärkeää, että kirjoitetun tekstin ja taustan välillä on hyvä kontrasti (1). Puna-viherheikkonäköisyys on yleisin värinäön poikkeama (8). Tekstin lukeminen ja kaavioiden hahmottaminen voi hankaloitua huomattavasti, jos punaista ja vihreää väriä käytetään paljon lähekkäin. Ruudunlukulaitteita käyttäville kuvien vaihtoehtoiset tekstit ovat välttämättömiä. Tekstien puuttuessa sisältö jää täysin saavuttamattomiin.

Jokaisen lukijan on tehtävä ajatustyötä selatessaan dokumentaatiota. Ajatustyö on osalle lukijoista raskaampaa kuin muille. Dokumentaation kirjoittajan on mahdollista keventää ajatustyön määrää kirjoittamalla lyhyitä lauseita, käyttämällä selkeitä lauserakenteita ja selittämällä, miten kirjoitettu teksti liittyy muuhun kokonaisuuteen. On mahdollista kirjoittaa dokumentaatio kokonaisuudessaan edellä mainitulla tavalla, jolloin teksti olisi selkokieltä. Usein riittää, kun artikkelin ymmärtämiseen vaadittavat tiedot on selitetty ja tekstiin on lisätty linkkejä muihin teksteihin. (3; 5.)

WCAG-standardi kehittäjän apuna

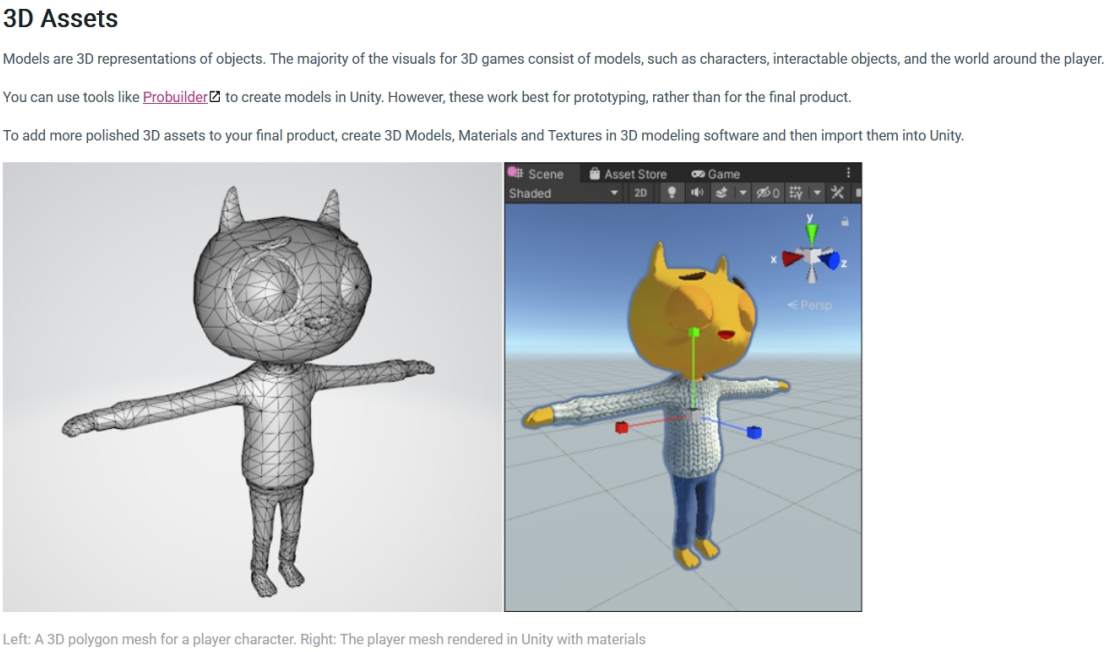
Saavutettavuuden takaamiseksi on laadittu standardi nimeltä WCAG, joka on lyhenne sanoista Web Content Accessibility Guidelines. Suomeksi puhutaan verkkosisällön saavutettavuuden ohjesäännöistä. Standardia seuraamalla verkkosivujen, kuten teknisten dokumentaatioiden, suunnittelijat voivat tehdä verkkosivuistaan maailmanlaajuisesti hyväksyttyjen saavutettavuuskäytänteiden mukaiset. (9.)

WCAG:n ensimmäinen versio julkaistiin vuonna 1999 (10). Toinen versio on ollut käytössä vuodesta 2008. Kirjoitushetkellä uusin versio 2.2 on julkaistu vuonna 2023 ja päivitetty joulukuussa 2024. (9.) WCAG:n kolmas versio on jo kehitteillä, mutta prosessi on kirjoitushetkellä vasta alussa (11).

2.4 Esimerkkejä dokumentaatioista

Ennen dokumentaation luomista on syytä tutustua muihin kyseisen alan dokumentaatioihin. Tässä luvussa tarkastellaan kolmea otosta kolmen eri pelialan yritysten dokumentaatioista. Otokset ovat yksittäisiä artikkeleita teknisistä dokumentaatioista. Otosten kieltä ja rakennetta tarkastellaan kriittisesti.

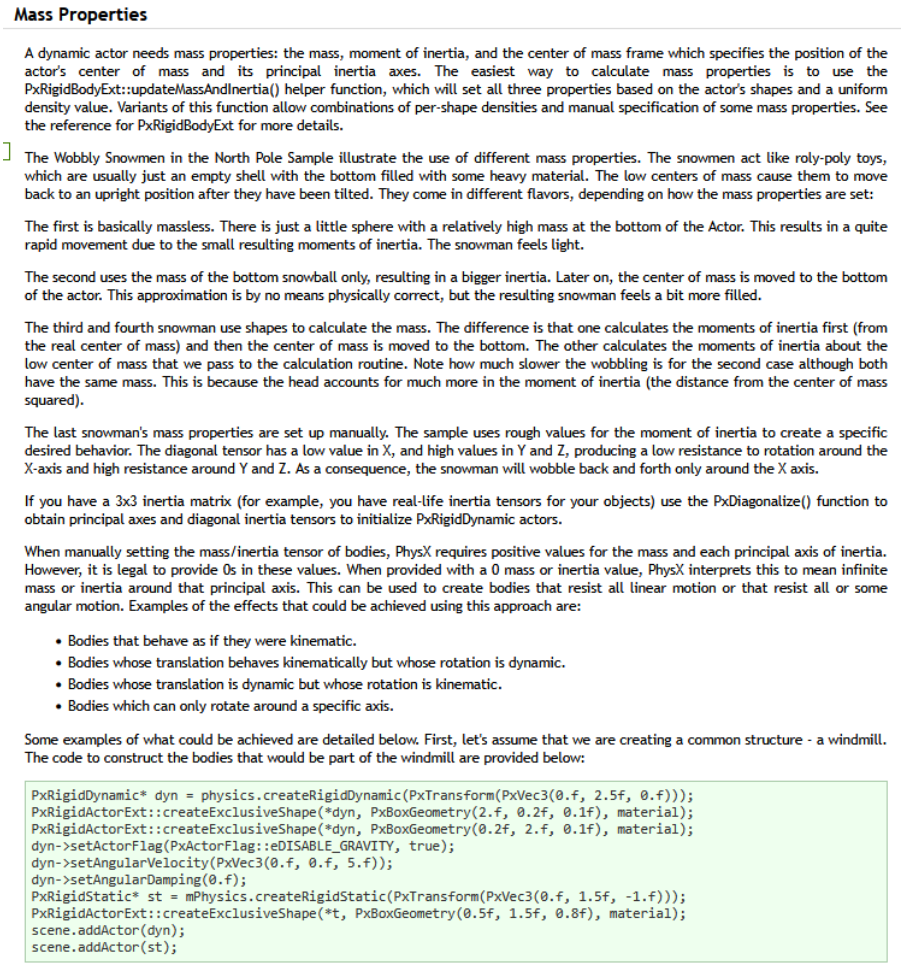
Ensimmäinen otos on Unity Manual -dokumentaatiosta. Unity Manual on korkean tason tekninen dokumentaatio Unity-pelimoottorin käyttäjälle. Unity Manual on selvästi suunnattu suurelle yleisölle. Dokumentaatiosta löytyy pääluvussa 2 mainittuja hyvän dokumentaation ominaisuuksia. Tekstistä pystyy ymmärtämään suurimman osan ilman mittavaa teknistä osaamista. Kuva 1 on kuvakaappaus dokumentaation artikkelista, käsittelee 3D-mallin lisäämistä Unity-projektiin.



Kuva 1. Kuvakaappaus Unityn dokumentaatiosta (12; 13).

Artikkelissa on käytetty lyhyitä lauseita. Tekstiin on lisätty korostettu hyperlinkki toiseen tiedonlähteeseen, joka helpottaa lukijan tiedonetsintää. Artikkeliin on myös lisätty havainnekuva selitteellä.

Toinen otos on NVIDIA:n PhysX-ohjelmistokehityspaketin dokumentaatiosta. PhysX on fysiikkamoottori. Ohjelmistokehityspaketilla tarkoitetaan kokoelmaa ohjelmistokehityksessä tarvittavia työkaluja. Toisen otoksen dokumentaatio on matalan tason tekninen dokumentaatio, jota lukevat sellaiset ohjelmoijat, jotka käyttävät PhysX-fysiikkamoottoria pelissä tai pelimoottorissa. Dokumentaatio on suunnattu IT-alan insinööreille tai hyvin syvällisesti ohjelmointiin perehtyneille. Sen täydellinen ymmärtäminen vaatii vuosien kokemusta ja tietämystä ohjelmoinnista, fysiikasta sekä matematiikasta. Kuva 2 on kuvakaappaus dokumentaation artikkelista, joka käsittelee fysiikkamoottorin jäykkien kappaleiden dynamiikan simulointia.



Mass Properties

A dynamic actor needs mass properties: the mass, moment of inertia, and the center of mass frame which specifies the position of the actor's center of mass and its principal inertia axes. The easiest way to calculate mass properties is to use the PxRigidBodyExt::updateMassAndInertia() helper function, which will set all three properties based on the actor's shapes and a uniform density value. Variants of this function allow combinations of per-shape densities and manual specification of some mass properties. See the reference for PxRigidBodyExt for more details.

The Wobbly Snowmen in the North Pole Sample illustrate the use of different mass properties. The snowmen act like roly-poly toys, which are usually just an empty shell with the bottom filled with some heavy material. The low centers of mass cause them to move back to an upright position after they have been tilted. They come in different flavors, depending on how the mass properties are set:

The first is basically massless. There is just a little sphere with a relatively high mass at the bottom of the Actor. This results in a quite rapid movement due to the small resulting moments of inertia. The snowman feels light.

The second uses the mass of the bottom snowball only, resulting in a bigger inertia. Later on, the center of mass is moved to the bottom of the actor. This approximation is by no means physically correct, but the resulting snowman feels a bit more filled.

The third and fourth snowman use shapes to calculate the mass. The difference is that one calculates the moments of inertia first (from the real center of mass) and then the center of mass is moved to the bottom. The other calculates the moments of inertia about the low center of mass that we pass to the calculation routine. Note how much slower the wobbling is for the second case although both have the same mass. This is because the head accounts for much more in the moment of inertia (the distance from the center of mass squared).

The last snowman's mass properties are set up manually. The sample uses rough values for the moment of inertia to create a specific desired behavior. The diagonal tensor has a low value in X, and high values in Y and Z, producing a low resistance to rotation around the X-axis and high resistance around Y and Z. As a consequence, the snowman will wobble back and forth only around the X axis.

If you have a 3x3 inertia matrix (for example, you have real-life inertia tensors for your objects) use the PxDiagonalize() function to obtain principal axes and diagonal inertia tensors to initialize PxRigidDynamic actors.

When manually setting the mass/inertia tensor of bodies, PhysX requires positive values for the mass and each principal axis of inertia. However, it is legal to provide 0s in these values. When provided with a 0 mass or inertia value, PhysX interprets this to mean infinite mass or inertia around that principal axis. This can be used to create bodies that resist all linear motion or that resist all or some angular motion. Examples of the effects that could be achieved using this approach are:

- Bodies that behave as if they were kinematic.
- Bodies whose translation behaves kinematically but whose rotation is dynamic.
- Bodies whose translation is dynamic but whose rotation is kinematic.
- Bodies which can only rotate around a specific axis.

Some examples of what could be achieved are detailed below. First, let's assume that we are creating a common structure - a windmill. The code to construct the bodies that would be part of the windmill are provided below:

```
PxRigidDynamic* dyn = physics.createRigidDynamic(PxTransform(PxVec3(0.f, 2.5f, 0.f)));
PxRigidActorExt::createExclusiveShape(*dyn, PxBoxGeometry(2.f, 0.2f, 0.1f), material);
PxRigidActorExt::createExclusiveShape(*dyn, PxBoxGeometry(0.2f, 2.f, 0.1f), material);
dyn->setActorFlag(PxActorFlag::eDISABLE_GRAVITY, true);
dyn->setAngularVelocity(PxVec3(0.f, 0.f, 5.f));
dyn->setAngularDamping(0.f);
PxRigidStatic* st = mPhysics.createRigidStatic(PxTransform(PxVec3(0.f, 1.5f, -1.f)));
PxRigidActorExt::createExclusiveShape(*t, PxBoxGeometry(0.5f, 1.5f, 0.8f), material);
scene.addActor(dyn);
scene.addActor(st);
```

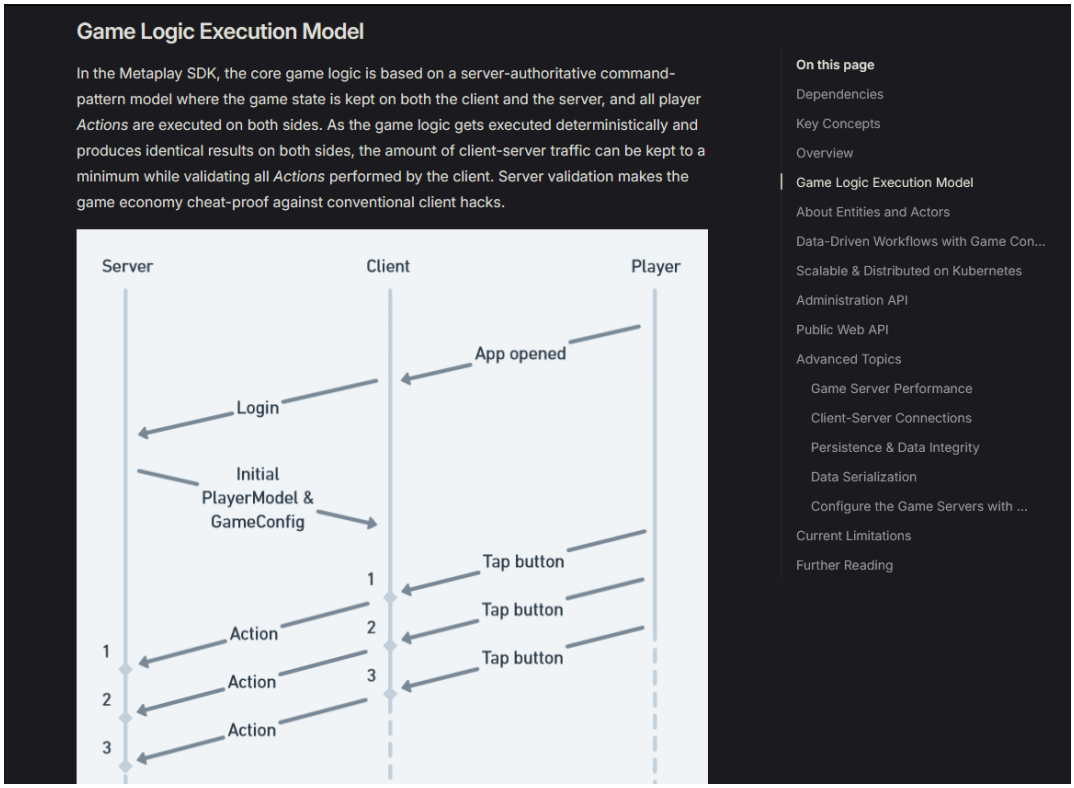
Kuva 2. Kuvakaappaus NVIDIA:n dokumentaatiosta (14).

Artikkelin kirjoitettu teksti on täynnä teknisiä termejä ja sisältää pitkiä, monimutkaisia lauserakenteita. Tekstin tueksi ei ole liitetty kuvia, jotka helpottaisivat lukukokemusta. Artikkelin tekstissä on lopuksi koodiesimerkki, joka on luettavassa muodossa. Esimerkki on pitkä ja täynnä numeroita. Artikkelin tekstissä viitataan visuaalisiin esimerkkeihin, joihin ei ole linkkiä. Visuaalisten esimerkkien löytäminen vie lukijalta aikaa varsinkin, jos sivuston oma hakutoiminto on epäkunnossa. Visuaalisten esimerkkien tutkiminen vaatii PhysX-ohjelmistokehityspaketin asentamista.

Kolmas otos on Metaplayn teknisestä dokumentaatiosta. Metaplay on Unity-pelimoottorin kanssa toimiva ohjelmistokehitypaketti, jolla voidaan toteuttaa taustajärjestelmiä moninpeleihin. Metaplayn dokumentaatio on toisen otoksen lailla suunnattu IT-alan ammattilaisille, mutta siinä on otettu lukija aiempaa paremmin huomioon.

Dokumentaatiossa on sekä korkean että matalan tason artikkeleita. Korkean tason artikkelit selittävät kuvien ja kuvaajien avulla ohjelmiston toimintaa. Korkean tason artikkeleiden ymmärtämiseen ei pääsääntöisesti vaadita ohjelmointikokemusta. Matalan tason artikkelit sisältävät enemmän koodiesimerkkejä ja lyhyitä, toimintaan ohjaavia lauseita. Matalan tason artikkeleiden ymmärtäminen vaatii aiheesta syvällistä tietämystä.

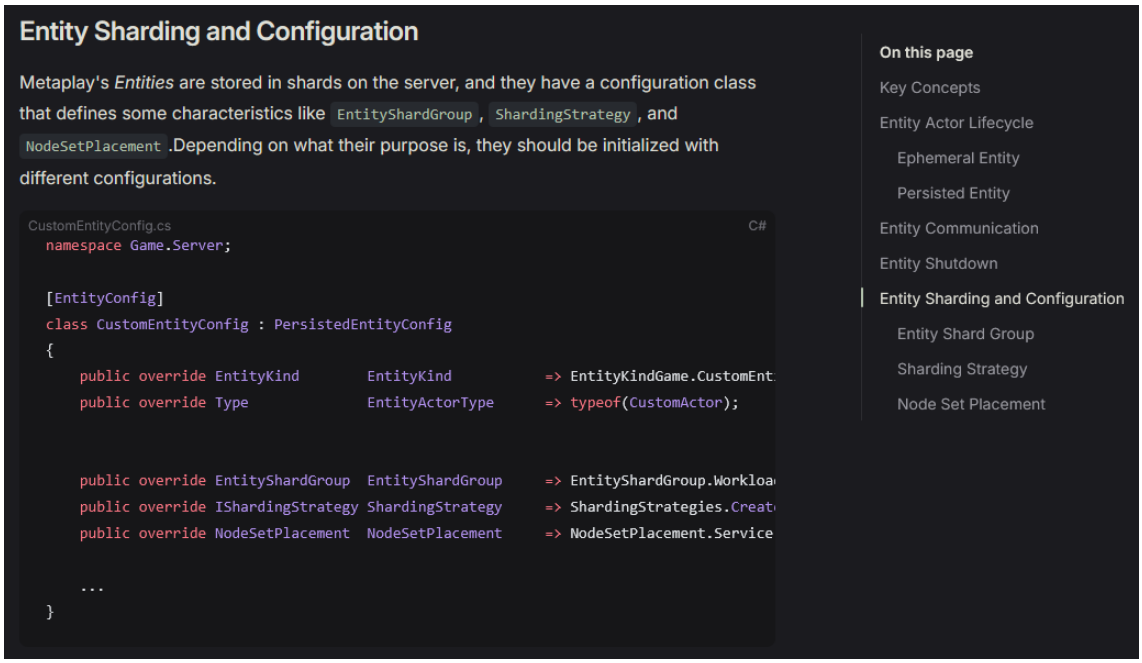
Kuva 3 on kuvakaappaus artikkelista, joka käsittelee pelilogiikan suoritusmallia Metaplayn järjestelmässä.



Kuva 3. Kuvakaappaus Metaplayn korkean tason artikkelista (15; 16).

Artikkelissa käsitellään Metaplayn dokumentaatiolle tyypillisellä tavalla korkean tason käsitettä. Selittävä teksti ja selkeä kuvaaja auttavat lukijaa ymmärtämään käsitettä.

Kuvan 4 artikkelissa käsitellään Metaplayn dokumentaatiolle tyypillisellä tavalla matalan tason käsitettä. Artikkeli käsittelee Metaplayn järjestelmän entiteettien särötystä ja niiden konfigurointia.



Kuva 4. Kuvakaappaus Metaplayn matalan tason artikkelista (17).

Tekstikappale ennen esimerkkiä on pidetty melko lyhyenä ja selkeänä. Ohjelmakoodia tekstin lomassa on korostettu korostusvärillä.

Metaplayn dokumentaatiossa sekä korkean että matalan tason artikkelien vierellä on sisällysluettelo, jossa on korostettuna lukijan ajantasainen sijainti sivulla. Sivustolla on myös hakutoiminto, joka ei näy kuvakaappauksissa.

Havaintoja

Otosten perusteella on havaittavissa, että alan yrityksillä on erilaisia näkemyksiä hyvän teknisen dokumentaation rakenteesta. Unityn pelimoottorin dokumentaation eteen on nähty paljon vaivaa. Korkean ja matalan tason artikkelit on sijoiteltu täysin erillisille dokumentaatiosivustoille (7). Korkean tason dokumentaatiossa kiinnitetään luettavuuteen ja helppokäyttöisyyteen erityistä huomiota.

NVIDIA:n ohjelmistokehityspaketin dokumentaatio keskittyy lähes kokonaan matalan tason käsitteisiin. Korkean tason käsitteitä mainitaan nimeltä, mutta niihin ei tarjota lukijalle linkkejä. Dokumentaatio kaipaisi luettavuuteen ja helppokäyttöisyyteen parannusta.

Metaplayn ohjelmistokehityspaketin dokumentaatio tasapainoilee kahden lähestymistavan välillä. Samasta dokumentaatiosta löytyy sekä korkean että matalan tason artikkeleita. Matalan tason artikkeleissa on usein linkit samaa aihetta käsitteleviin korkean tason artikkeleihin (18). Yksittäisiltä sivuilta saattaa löytyä niin matalan kuin korkeankin tason asioita käsitteleviä tekstejä. Ne etenevät loogisesti korkeammista matalamman tason käsitteisiin (17).

Teknistä dokumentaatiota suunnitellessa on mietittävä tarkkaan dokumentaation kohderyhmään kuuluvien lukijoiden tarpeet ja lähtötaso. Jos resurssit eivät riitä useamman dokumentaation ylläpitoon, dokumentaation rakenteessa on kiinnitettävä erityistä huomiota tiedon jäsentelyyn ja linkittämiseen. Sivuston sujuvan käytön ja hyvän lukukokemuksen saavuttamiseksi matalan tason artikkelin lukijalle on tarjottava polku korkeamman tason tietoon. Toimiva hakutoiminto helpottaa oikeiden artikkelien löytämistä ja saattaa jopa olla tiedonhaun pelastus, jos linkkejä artikkelissa viitattuun tietoon ei ole lisätty.

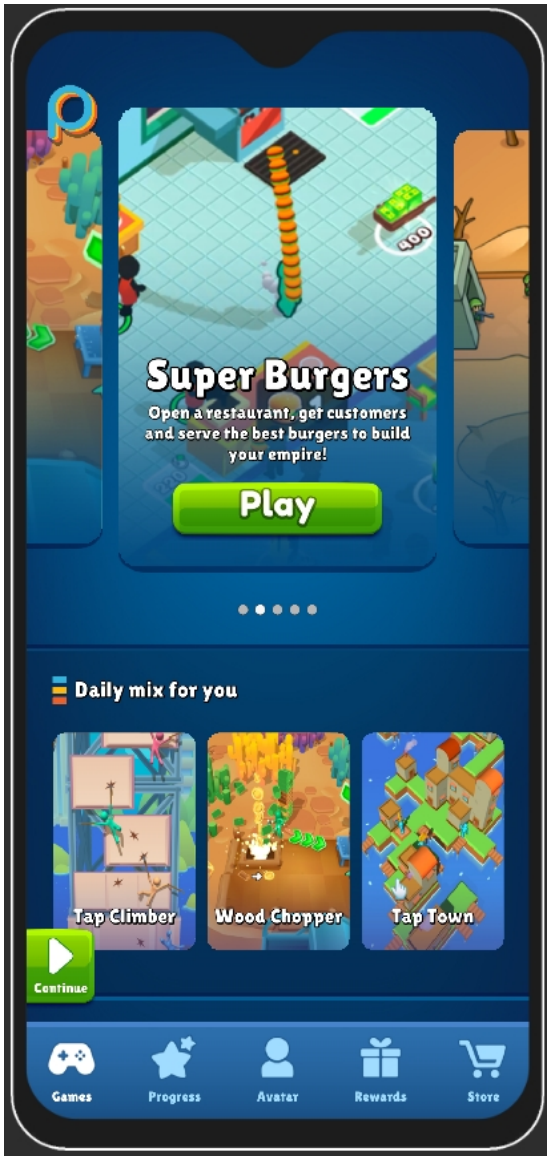
3 Insinöörityön ohjelmisto ja tilaaja

Insinöörityön tilaaja on Playnest Oy. Yritys tunnettiin aiemmin nimellä Wonder Hive Oy. Yhtiö on perustettu vuonna 2022, ja sen päätoimialana on tietokoneohjelmistojen suunnittelu ja valmistus. Kirjoitushetkellä yrityksellä on kuusi vakituista työntekijää. (19.)

Playnest Oy kehittää Playnest-nimistä mobiilipelien kehitys- ja julkaisualustaa. Alustan ydintarkoituksena on luoda yhtenäinen kanava julkaisijan, pelikehittäjien ja pelaajien välille. Playnest-alusta on toteutettu Unity-pelimoottorilla.

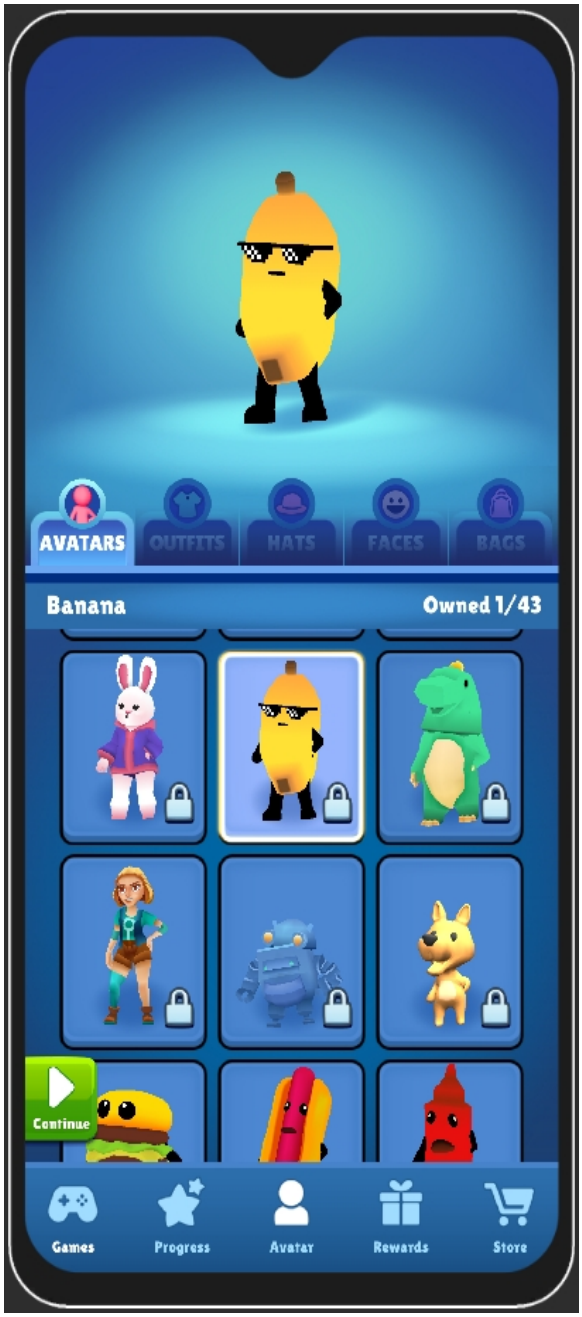
3.1 Playnest-sovellus

Playnest tarjoaa pelaajille yhden ladattavan sovelluksen, josta löytyy monia erilaisia pelejä. Pelaajille suositellaan tiettyjä pelejä alustan analytiikkaan perustuen. Kuvan 5 keskellä näkyvän Daily mix for you -otsikon alla olevat pelien kuvakkeet ovat esimerkki näistä suosituksista.



Kuva 5. Playnest-sovelluksen etusivu.

Playnest-sovelluksessa pelaajalla on oma avatar-hahmo, joka kulkee pelaajan mukana pelien välillä ja edustaa pelaajaa. Kuvassa 6 on avatar-hahmon muokkausnäkymä. Kuvan kaikki hahmot ovat aluksi lukittuina.



Kuva 6. Avatar-hahmon muokkausnäkymä.

Avatar-hahmon ulkoasu on siis muokattavissa, mutta pelaajan on avattava jokainen muokkausvaihtoehto erikseen voidakseen käyttää sitä. Muokkausvaihtoehtoja voi avata suorittamalla tehtäviä, ansaitsemalla pisteitä peleissä ja ostamalla muokkausvaihtoehtoja pelialustan omalla valuutalla. Pelialustan omaa valuuttaa voi ostaa lisää oikealla rahalla.

3.2 Playnest SDK

Playnest SDK on kevään 2025 aikana rajoitetusti julkaistava Unity-pelimoottoriin asennettava ohjelmistokehityspaketti, jonka avulla pelikehittäjät voivat toteuttaa ja julkaista oman mobiilipelinsä Playnest-alustalle. Playnest SDK:n tarkoitus on tehdä pelin yhtenäistäminen osaksi Playnest-alustaa mahdollisimman helpoksi.

Ohjelmistokehityspaketti sisältää valmiita toteutuksia monille ominaisuuksille, joita mobiilipeleissä usein käytetään. Näihin ominaisuuksiin lukeutuvat esimerkiksi valmiit rajapinnat pelitilan tallentamiseen, pelikohtaisten tehtävien luomiseen ja pelaajaa palkitsevien mainosten näyttämiseen.

Ohjelmistokehityspaketin yleisimpiä käyttäjiä tulevat olemaan pienet peliyhtiöt ja yksin toimivat pelikehittäjät. Vastineeksi oman pelin julkaisemisesta ja ylläpitämisestä Playnest-alustalla kehittäjät saavat julkaisijalta esimerkiksi näkyvyyttä, asiakashankintaa ja reilun osan mainostuloista sekä pelin sisäisten ostosten tuotosta.

4 Insinöörityön suunnittelu

Tilaaja oli dokumentoinut joitakin merkittävimpiä sovelluksen ja ohjelmistokehityspaketin ominaisuuksia. Dokumentit olivat enimmäkseen toisistaan irrallisia. Niiden välille ei siis ollut luotu ajatuksellista yhteyttä eikä hyperlinkkejä. Vain harvoja dokumentteja oli aktiivisesti päivitetty, joten tieto oli osin vanhentunutta.

Dokumentit sijaitsivat eri sovelluksissa. Osa sijaitsi Notion-projektinhallintatyökalun sivuilla, osa GitHub-versionhallintapalvelun kuvauskannassa. Kuvauskanta on hakemisto, jonka tiedostomuutokset versioidaan. Kuvauskanta on yleensä saatavilla ohjelmistokehitystiimin kaikille jäsenille verkon välityksellä.

4.1 Työn tavoite ja aikataulu

Toimeksiantona on toteuttaa ulkoasun ja sisällön osalta yhtenäinen dokumentaatio, josta ohjelmistokehityspaketin käyttäjät pystyvät hakemaan nopeasti tietoa. Dokumentaation tulee olla yksinkertainen ymmärtää ja myyvä, jotta pelistudiot haluaisivat tehdä yhteistyötä Playnestin kanssa.

Dokumentaation tueksi kehitetään Playnest-alustalle yhtenäistetty esimerkkipeli, jossa toteutetaan kaikki alustan tärkeimmät ominaisuudet. Esimerkkiprojekti tiedostoineen ja lähdekoodeineen jaetaan ohjelmistokehityspaketin mukana kehittäjille. Projektista voi kopioida suoraan koodia sekä katsoa mallia. Esimerkkipeliä on aloitettu tekemään jo vuonna 2024, mutta työ on jäänyt edelliseltä tekijältä kesken ja luodut kooditiedostot sisältävät paljon vanhentunutta koodia.

Työkalut ja -menetelmät

Dokumentaatio tullaan totetuttamaan Notion-projektinhallintatyökalulla. Notionin ominaisuuksiin kuuluu verkkosivujen julkaiseminen ja ylläpito ilman lisäkustannuksia sisällöntuottajalle. Insinöörityön tilaajalla on jo Plus-tilaus Notionista, joten kaikilla luoduilla sivuilla on 30 päivän versiohistoria ja monia muokkausvaihtoehtoja. (20.) Kuvakaappauksiin käytetään Microsoft Windowsin Snipping Tool -ohjelmaa ja kuvien editointiin GIMP-kuvankäsittelyohjelmaa.

Osa dokumentaation sisällöstä haetaan ja päivitetään vanhoista dokumenteista. Paljon sisältöä on kirjoitettava alusta alkaen muita työntekijöitä konsultoimalla ja lukemalla koodia.

Esimerkkiprojektia tullaan kehittämään Unity-pelimoottorilla ja Microsoft Visual Studio 2022 -koodieditorilla. Versionhallintaan käytetään GitHub-versionhallintapalvelua. Tarkkaan ottaen versionhallinnassa käytetään tilaajan kuvauskantaa ja siellä omaa Git-haaraa. Git-haara on Git-pohjaisen kuvauskannan tietystä versiosta haarautuva kehityspolku, jolle luodaan oma versiohistoria (21).

Dokumentaatio pyritään julkaisemaan neljän viikon sisällä aloituspäivämäärästä. Dokumentaation ei tarvitse silloin olla täysin valmis, mutta tiedot tärkeimmistä ominaisuuksista ja ohjeet ohjelmistokehityspaketin käyttöönottoon tulee olla sisällytettynä dokumentaation sivuille.

Dokumentaation rakenne

Dokumentaatio kannattaa pilkkoa pieniin osioihin, jotta siitä tulee selkeä käyttää. Osiot tulee kartoittaa, jotta päästään alkuun. Seuraavaksi aineistot kootaan dokumentaatiosivustolle. Uutta sisältöä ei kannata kirjoittaa, ennen kuin olemassaoleva aineisto on saatu kokoon. Vanhaa aineistoa läpi käydessä ja muokatessa dokumentaation kirjoittaja oppii koodikannasta ja vallitsevista käytännöistä, jolloin uuden aineiston tuottaminen on tehokkaampaa.

Ennen dokumentointityön aloittamista yrityksen teknisen johdon kanssa pidettiin kaksi suunnittelupalaveria. Palavereiden tuloksena dokumentaatio tullaan jakamaan seuraaviin sivuihin:

- alustan esittely
- ohjelmistokehityspaketin ominaisuudet
- yksityiskohtainen opas pelin luomiseen alustalle
- muut oppaat ja käytänteet.

Työn edetessä on tärkeää saada palautetta dokumentaation asettelusta, kielestä ja käytettävyydestä. Esihenkilön kanssa pidetään vähintään kerran viikossa palaveri, jossa työn etenemistä seurataan. Myös muiden työntekijöiden palaute on tervetullutta ja otetaan huomioon.

4.2 Työn ongelmakohdat ja riskit

Notionissa on paljon valmiita elementtejä. Se helpottaa dokumentaation julkaisua ja ylläpitoa. Siinä on myös rajoitteita. Työkalulla ei esimerkiksi pysty itse luomaan mitään kelluvia ikkunoita tai sivupalkkeja navigointia varten. Sivusto itsessään käyttää monia kelluvia elementtejä ja sivupalkkeja, joten niiden käytön evääminen maksavilta asiakkailta on harmillista.

Ohjelmistokehityspaketin tekninen dokumentaatio on Playnestin ensimmäinen julkinen dokumentaatio, joten kohderyhmän rajaaminen on haaste. Teksti on kirjoitettava niin, että sitä pystyy lukemaan ohjelmoijan lisäksi myös graafinen suunnittelija tai pelisuunnittelija.

Riskejä nopealla aikataululla tehdessä on useita. Ohjelmistokehityspakettia muokataan dokumentaation kirjoitushetkellä vielä julkaistavaan kuntoon. Sen vuoksi kaikkia esimerkkikoodin osia ei välttämättä ehditä tai muisteta testata ennen julkaisua. Virheellinen esimerkkikoodi johtaa lukijan pettymykseen ja luottamuksen menetykseen dokumentaatiota kohtaan. Kaikista ohjelmistokehityspakettiin kytkeytyvistä koodikannan muutoksista on tiedotettava dokumentaation kirjoittajaa, jotta tiedot pysyvät ajantasaisina.

Muutenkin kirjoitus- ja asiavirheiden minimointi on tärkeää, jotta Playnestin maine ammattimaisena yhteistyökumppanina säilyy. Notionissa on tukityökalu, jossa on englanninkielen sanakirja. Työkalu ehdottaa tunnistamattoman sanan tilalle korjausta. Selkeät kirjoitusvirheet pystytään siis korjaamaan nopeasti.

5 Insinöörityön toteutus

Ensimmäinä päivinä tutustuttiin tuontantotiimiin, luotiin käyttäjätunnuksia eri palveluihin, luettiin olemassaolevia dokumentteja ja opeteltiin Notion-työkalun käyttöä. Erityisesti kontaktien luominen muihin työntekijöihin on tärkeää, jotta heitä on jatkossa luontevaa lähestyä ongelmilla ja kysymyksillä.

5.1 Olemassaolevan aineiston kokoaminen

Olemassaoleviin dokumentteihin perehdyttyä selvisi, että GitHub-kuvauskannan dokumentit olivat tekstipohjaisia selostuksia muutamilla koodiesimerkeillä. Vain pieni osa näistä dokumenteista käsitteli aiheita, jotka kuuluvat ohjelmistokehityspakettiin. Notion-työkalun dokumentit olivat pääasiassa kirjoitusasultaan ohjeita ja suosituksia. Monissa dokumenteissa oli havainnekuvia ja -koodia, mutta useimpia ei voitu enää käyttää, sillä sovelluksen grafiikka ja käyttöliittymä oli muuttunut.

Koska osa dokumentaatiosta oli vanhentunutta, täytyi tekstin paikkansapitävyys ja esimerkkikoodin toiminta varmistaa samalla, kun dokumentaatiota kirjoitettiin. Varmistaminen onnistui tutkimalla ja vertailemalla pelialustalla julkaistujen pelien toimivia lähdekoodeja olemassaoleviin dokumentteihin. Joissain tapauksissa pelien välillä oli eroja samojen asioiden toteutuksissa, jolloin muilta työntekijöiltä oli kysyttävä tarkennusta.

5.2 Sisällön tuottaminen

Olemassaolevien aineistojen parissa työskentelyn tuloksena ymmärrys yrityksen koodaustyyliin ja -käytäntöihin kasvoi. Sen ansiosta esimerkkipelin kehitys voitiin aloittaa. Esimerkkipeliin päätettiin toteuttaa kosketuksella ohjattava avatar-hahmo. Hahmon lisäksi peliin toteutettiin jalkapallo ja -maali sekä yksinker-

tainen pistelaskujärjestelmä. Esimerkkiprojektin edetessä ilmeni, että dokumentaatiosta puuttuvien osuuksien kirjoittaminen ja esimerkkiprojektin kehittäminen kulkivat miltei käsi kädessä. Esimerkkipeliin lisätyn ominaisuuden pystyi kirjoittamaan lähes sellaisenaan dokumentaation ohjeeksi. Taustoituksia ja selvennyksiä koodille sekä funktioiden parametreille oli usein tarpeellista lisätä.

Esimerkkipelin pistelaskurissa pystyttiin hyödyntämään kahta alustan eri ominaisuutta yhden funktion sisällä. Esimerkkikoodissa 1 on esitetty pistelaskurin AddScore-funktio. Funktion ensimmäisellä rivillä muokataan pelin tallennustiedostoa alustan tallennusominaisuutta käyttäen. Seuraavaksi lisätään 1 piste pistetaulun _score-muuttujaan. Lopuksi pyydetään alustan tehtäväjärjestelmää edistämään goals-scored -nimistä tehtävää yhdellä askeleella.

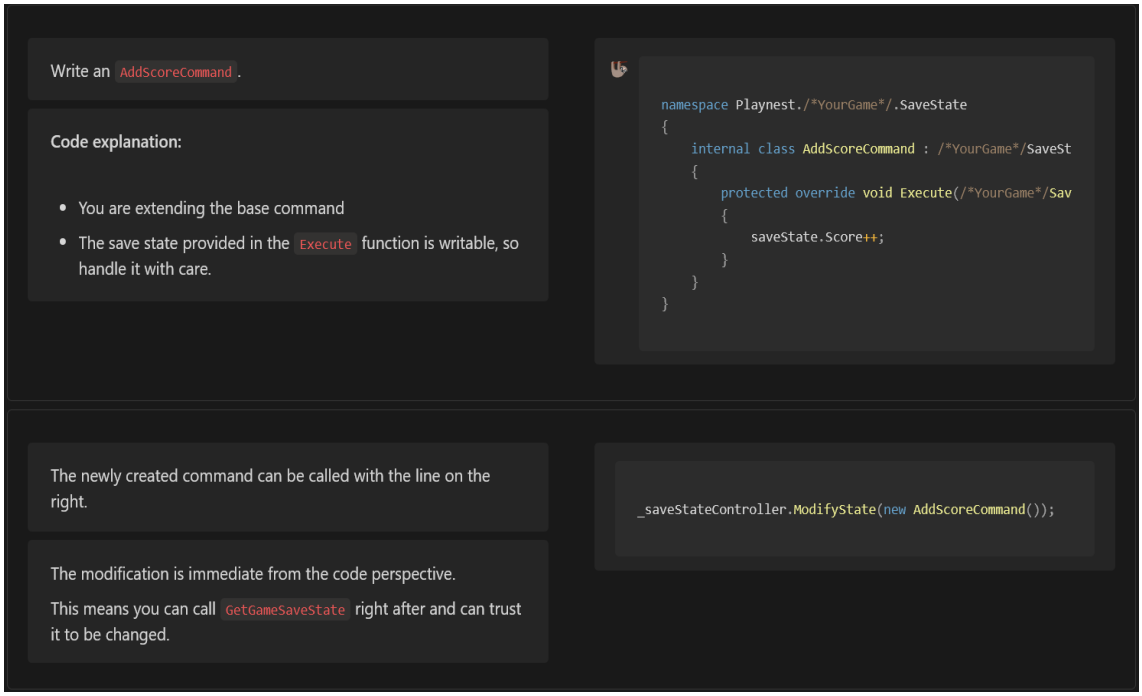
```
public void AddScore()
{
     _saveStateController.ModifyState(new AddScoreCommand());

     _score.Value++;

     // Advance the mission
     _missionSystemController.AdvanceMission(
          missionTypeId: new("goals-scored"),
          amount: new(1),
          shouldSerialize: true);
}
```

Esimerkkikoodi 1. Esimerkkipelin pistelaskurin AddScore-funktio.

Dokumentaatiossa Playnest-alustan tallennusjärjestelmää käsittelevässä artikkelissa käsitellään tallennuskomennon luontia. Esimerkkinä siinä esiintyy ainoastaan esimerkkikoodi 1:n ylin rivi (kuva 7).

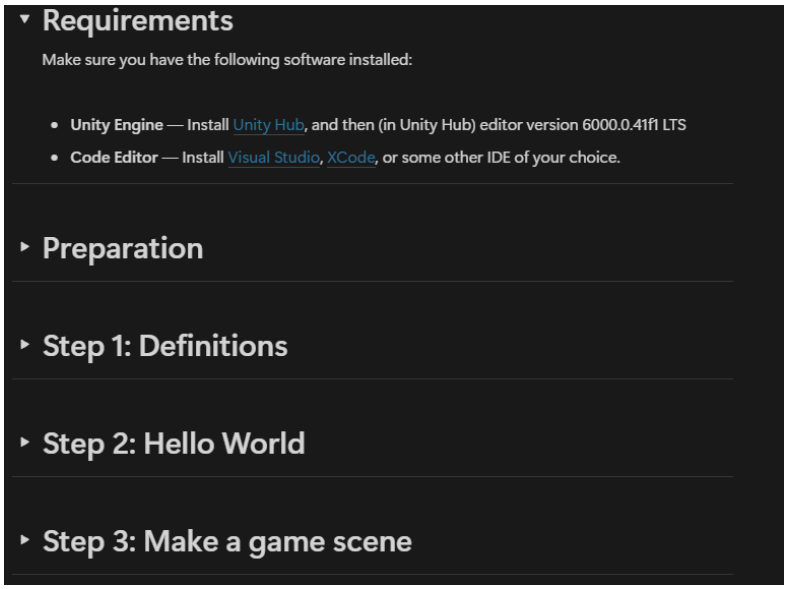


Kuva 7. Kuvakaappaus dokumentaation tallennusjärjestelmän artikkelista.

Esimerkkiprojektia kehitettäessä oli jatkuvasti otettava huomioon, että pelkkä koodin toimivuus ei riitä. Koodin oli oltava riisuttua kaikesta tarpeettomasta ja mahdollisimman helppoa luettavaa.

5.3 Dokumentaation rakenteen ja ulkoasun kehitys

Ensimmäinen versio sisälsi enimmäkseen otsikoita, joiden alainen teksti oli piilotettuna. Sisältö avautui ainoastaan napsauttamalla otsikon viereistä kolmikulmaista symbolia (kuva 8).



Kuva 8: Ensimmäisen version otsikkorakenne.

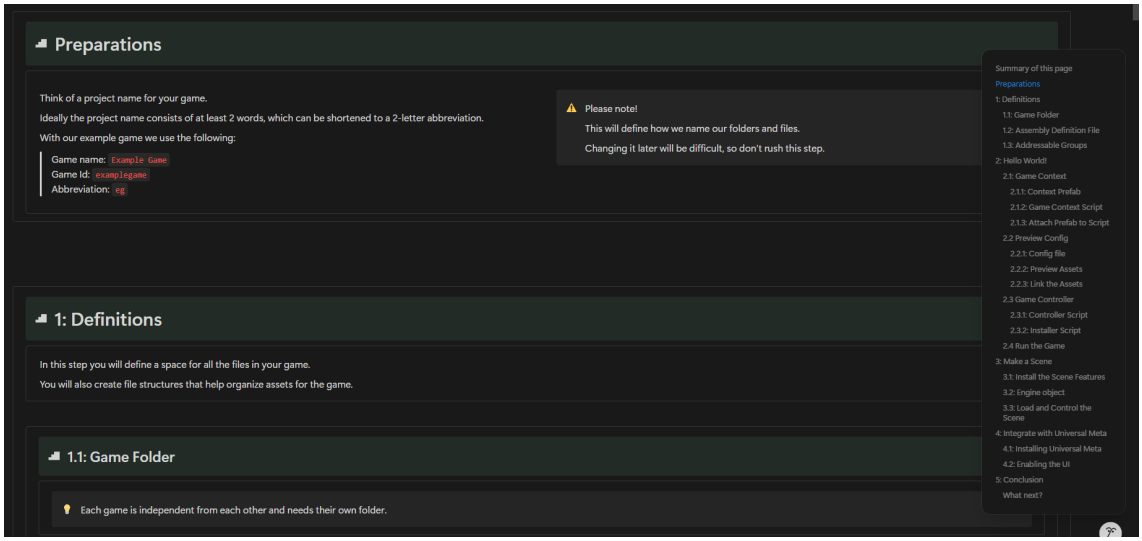
Piilotettavien otsikoiden käyttö teki dokumentaatiosta kompaktin näköisen, mutta sisälsi ainakin kolme huonoa puolta. Visuaalinen ilme oli melko tylsä, lukijan oli jatkuvasti pakko painella otsikoita auki ja Notionin oma automaattisesti päivittyvä sisällysluetteloelementti ei tunnistanut mitään piilotettavien otsikoiden alaisia alaotsikoita. Kuvassa 9 näkyvässä ensimmäisen version sisällysluettelossa pitäisi olla useita alaotsikoita.

Kuva 9. Ensimmäisen version sisällysluettelo.

Kuten artikkelien otsikot, myös navigointielementti oli aluksi piilotettava otsikko (kuva 10). Se vaati yhtälailla ylimääräisen napsauttamisen avautuakseen. Elementti ei kertonut lukijalle, millä sivulla tämä on.

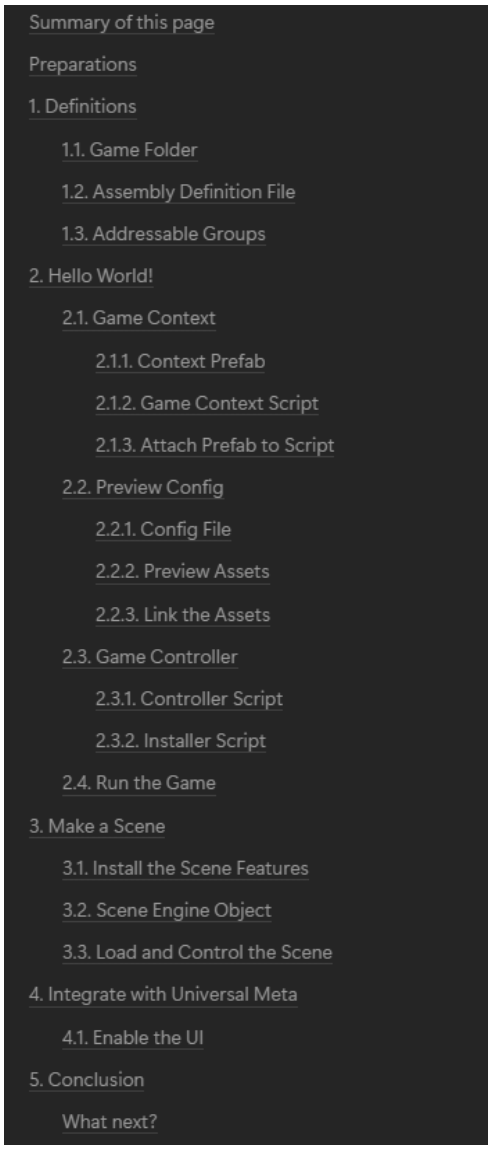
Kuva 10. Ensimmäisen version navigointielementti.

Ensimmäisen version esittelyn jälkeen tilaajan Notion-dokumentaatioista vastaava työntekijän opastuksella syntyi uusi visuaalinen ja looginen rakenne, joka sai koko tiimin hyväksynnän lopulliseen dokumentaatioon. Myös Notionin oman sivupalkin kautta navigoinnista tuli varteenotettava tapa selata dokumentaatiota, kun alaotsikot toimivat oikein. Kuvassa 11 on esitelty dokumentaation lopullista visuaalista ilmettä ja rakennetta. Oikealla näkyy piilotettava sivupalkki.



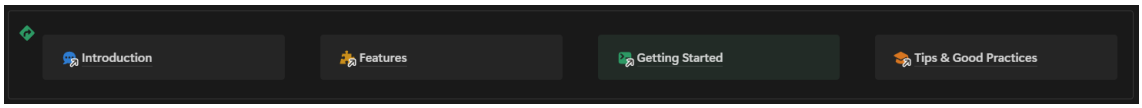
Kuva 11. Lopullisen version ulkoasu.

Otsikoista poistettiin tässä kohtaa turhat työvaihetta tarkoittavat Step-sanat ja ne korvattiin kuvassa 11 näkyvillä porras-symboleilla. Sisällysluettelo toimi otsikoiden rakennemuutoksen jälkeen oikein (kuva 12).



Kuva 12. Lopullisen version sisällysluettelo.

Lopullisen version navigointielementti on jatkuvasti esillä. Elementti korostaa tummennetulla värillä sen sivun painiketta, jolla lukija on. Myös pieni nuoli-symboli elementin vasemmassa yläkulmassa vaihtaa väriä sen sivun ikonin mukaan, jota lukija on sillä hetkellä selaamassa.



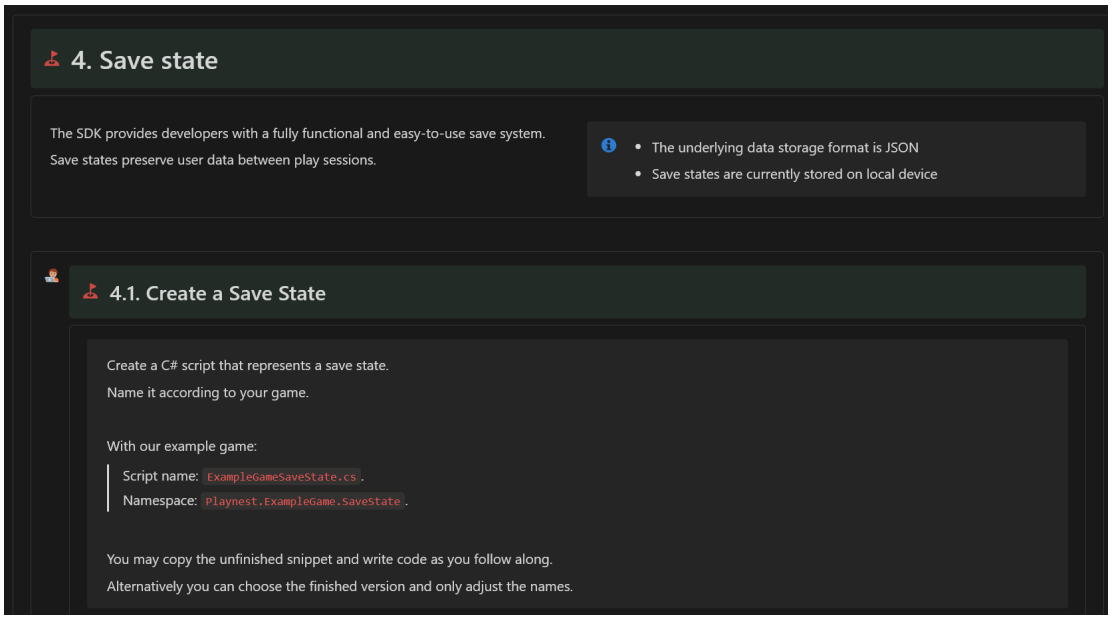
Kuva 13. Lopullisen version navigointielementti.

Lukija saa selkeää visuaalista palautetta lopullisen version navigointielementiltä. Palaute ehkäisee hämmennystä esimerkiksi tilanteessa, jossa lukija painaa eri sivulle vievää linkkiä, mutta toinen sivu ei lataudukaan jostain syystä. Lukija huomaa jonkin olevan vialla todennäköisesti nopeammin, kun korostusväri ei pian vaihdu toisen painikkeen kohdalle.

5.4 Työn tulokset

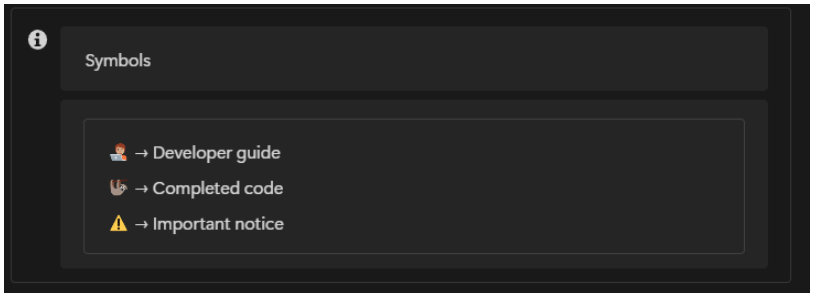
Dokumentaatio oli julkaisuvalmis aikataulun mukaisesti. Oppaat joidenkin alustan ominaisuuksien käyttöönottamiseksi eivät olleet valmiita julkaisupäivänä, mutta ne lisättiin julkaisuviikolla. Lopullista ulkoasua on esitelty luvun 5.3 kuvissa 10–13.

Dokumentaatiossa on korkean ja matalan tason artikkeleita. Matalan tason artikkelit on eroteltu pienellä sisennyksellä ja ohjelmistokehittäjä-symbolilla. Kuvasta 14 voi nähdä tämän rakenteen.



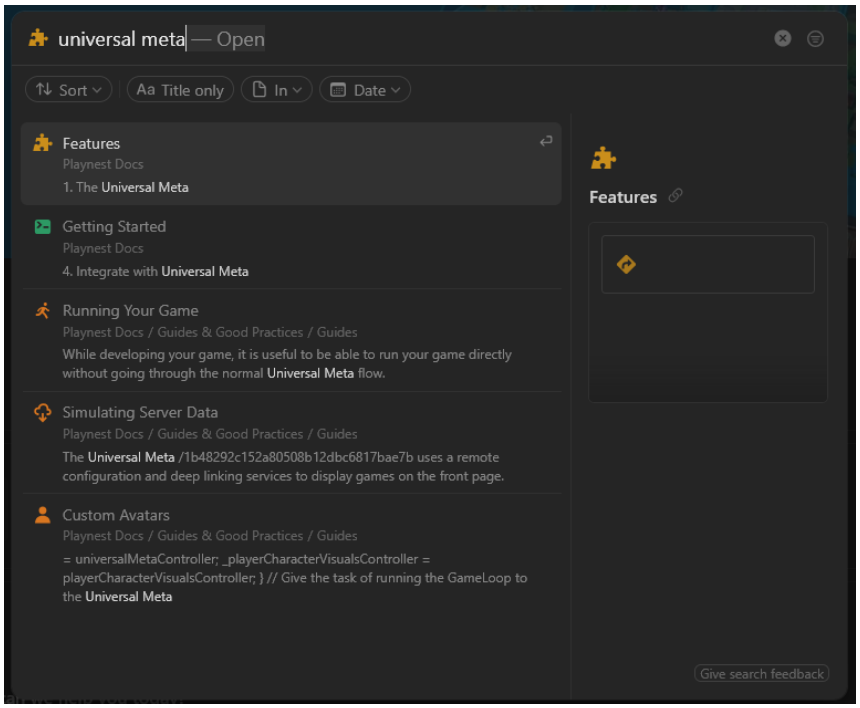
Kuva 14. Dokumentaation lopullisen rakenteen artikkeleiden erottelu.

On hyvä varmistaa symbolien merkitys lukijoille. Sivuilla käytettäville symboleille on aina selitteet sivun alussa. Jokaisesta kuvassa 15 näkyvästä pikkukuvasta lähtee nuoli selitetekstiin.



Kuva 15. Dokumentaation symboleiden selite-elementti.

Sivustolla on monipuolinen hakutoiminto, jonka Notion-työkalu tarjoaa. Hakutoiminto on pienen suurennuslasikuvakkeen takana sivuston yläreunassa. Kuvassa 16 nähdään hakutoiminto käytännössä. Toiminnon avulla voi lajitella ja rajata hakutuloksia monella eri tavalla. Hakutoiminto löytää tehokkaasti kaikki maininnat sivuston jokaisesta artikkelista, myös esimerkkikoodista.



Kuva 16. Kuvakaappaus dokumentaation hakutoiminnosta.

Kokonaisvaltaisen käsityksen dokumentaatiosta saa tietenkin lukemalla sitä. Koko dokumentaatio on kirjoitushetkellä luettavissa Playnestin verkkosivuilla osoitteessa https://playnest.notion.site/playnest-docs.

Esimerkkipeli

Esimerkkipeli on vielä kesken. Siitä puuttuu joitakin ominaisuuksia, kuten palkitsevien mainosten ja pelin sisäisten ostosten esittelyt. Monet ominaisuudet, kuten käyttöliittymä, pelitilan tallennus, avatar-hahmo ja sen liikuttaminen ovat toiminnassa. Kuvassa 17 näkyy pelin yleiskuva tällä hetkellä.

Kuva 17. Kuvakaappaus esimerkkipelistä.

Ympäristön grafiikoiden on tarkoitus jäädä hyvin yksinkertaisiksi, jotta pelialustan omat ominaisuudet korostuvat. Visuaalista ilmettä tärkeämpiä ovat ominaisuuksien koodipuolen toteutukset.

Ongelmakohtien pohdinta

Notionin työkalujen ja elementtien rajoittuneisuus ja dokumentin kirjoittajan kokemattomuus niiden käytössä hidastivat hieman prosessia, mutta eivät estäneet saavuttamasta selkeää ja visuaalisesti miellyttävää lopputulosta.

Tämän raportin kirjoitushetkellä vasta muutama tilaajaorganisaation ulkopuolinen ohjelmoija on tutustunut dokumentaatioon. Näiltä ohjelmoijilta ei toistaiseksi ole tullut negatiivista palautetta dokumentaation luettavuudesta.

Ohjelmistokehityspaketin julkaisuaikataulu venyi suunnitellusta. Vaikka se ei ollut hyvä asia, se mahdollisti dokumentaation esimerkkikoodien toiminnan varmistamisen ennen ohjelmistokehityspaketin julkaisua.

5.5 Työn jatkokehitys

Mikä tahansa dokumentaatio, joka käsittelee jatkuvasti kehitettävää ohjelmistoa, ei ole koskaan valmis. Lähitulevaisuudessa Playnest SDK:n teknisen dokumentaation päivittämisestä ja uusien materiaalien lisäämisestä vastaa ensisijaisesti dokumentaation kirjoittaja yhteistyössä tilaajan teknisen johdon kanssa. Jos organisaation koko kasvaa, dokumentaation ylläpitäjäksi saatetaan palkata eri ammattilainen.

Esimerkkipelin kehittämistä aiotaan jatkaa. Dokumentaation lailla myös esimerkkipeli on keskeneräinen niin kauan, kun pelialustaan lisätään ominaisuuksia.

Kehityskohteita

Dokumentaatio sisältää tällä hetkellä ainoastaan ohjeita ja koodiesimerkkejä. Usein ohjelmoijia kiinnostaa tietää myös ohjelmakoodin luokista tarkemmin, jotta niitä osataan käyttää tehokkaasti. Muun muassa Unityn Scripting API -dokumentaatiossa ja Unreal Enginen C++ API Reference -dokumentaatiossa on taulukoituna kaikki luokkien ominaisuudet (22; 23). Vastaavia kokoelmia varten dokumentaatioon voisi luoda uuden sivun.

Dokumentaatiossa on tällä hetkellä vain kuvia, kirjoitettua tekstiä ja koodia. Kiireen vuoksi kaikilta kuvilta puuttuu vaihtoehtoiset tekstit. Ne on lisättävä ensi tilassa. Tekstin tueksi voisi liittää myös grafiikkaa, joka esittäisi yksinkertaistettuna eri ominaisuuksien käyttöönoton vaiheita. Grafiikka tai jopa animoidut kuvaajat helpottaisivat lukijaa hahmottamaan työvaiheita.

Korkean tason käsitteitä, kuten tähtipassia, selittäviä artikkeleita olisi hyvä luoda lisää. Tähtipassi on palkintojärjestelmä, jossa pelaaja saa sitä parempia palkintoja, mitä pitempään hän pelaa. Kehittäjien pitää ymmärtää alustan eri ominaisuuksien merkitys, jotta niiden tukeminen omissa peleissä nähtäisiin hyödyllisenä.

Esimerkkipelistä voisi tehdä mielenkiintoa herättävämmän, kuin se kirjoitushetkellä on. Siihen voisi toteuttaa useita erilaisia minipelejä, joissa avatar-hahmoa liikutetaan eri tavoin ja sitä esitetään eri kuvakulmista. Minipelit saattaisivat innoittaa kehittäjiä lähestymään pelihahmon hyödyntämistä uusilla tavoilla ja mahdollisesti luomaan alustalle erityyppisiä pelejä.

Mobiilipelaajat pitävät hyvin erilaisista peleistä. Playnest-alustalle pystyy luomaan esimerkiksi tasohyppelypelejä, ammuntapelejä, älypelejä, urheilupelejä tai seikkailupelejä. Mitä laajempi pelien kirjo on, sitä houkuttelevammaksi pelialusta pelaajille tulee.

6 Yhteenveto

Insinöörityönä toteutettavan dokumentaation päätavoitteet olivat selkeä luettavuus, tiedonhaun helppous ja ammattimainen sekä kiinnostava visuaalinen ilme. Niiden avulla dokumentaatio olisi myös myyvä.

Insinöörityötä varten tutustuttiin hyvän teknisen dokumentaation perusteisiin. Teorian sisäistämisen jälkeen kartoitettiin Playnest SDK:n dokumentaation lähtötilanne ja tehtiin suunnitelma. Mahdolliset riskit ja ongelmakohdat yritettiin tunnistaa ennalta niiden välttämiseksi.

Toteutusvaiheen alussa tutustuttiin tilaajaorganisaation henkilöstöön, tavoitteisiin ja ohjelmiston rakenteeseen. Sen jälkeen koottiin valmista aineistoa ja tuotettiin puuttuvaa sisältöä. Visuaalisesta ilmeestä ja rakenteesta ei tullut ensimmäisellä yrittämällä hyviä. Tilaajan palautteen ja alan ammattilaisen neuvojen jälkeen sivustosta saatiin näyttävä ja sujuva selata.

Tässä insinöörityössä käsiteltyjä hyvän teknisen dokumentaation perusteita voidaan soveltaa myös muihin olemassaoleviin tai tuleviin ohjelmistoalan dokumentaatioihin.

Insinöörityö opetti tekijälleen yhteistyö-, suunnittelu-, ja projektinhallintataitoja. Toteutusvaiheen sujuvuus lisäsi uskoa tekijän omia ammattivalmiuksia ja -taitoja kohtaan. Tulevaisuutta ei voi ennustaa, mutta on mahdollista, että tekijä jatkaa saman alustan kehittämistä vielä insinöörityön jälkeenkin.

Lähteet

1 Oragui, David. 2021. Software Documentation Best Practices [With Examples]. Verkkoaineisto. Helpjuice. <https://helpjuice.com/blog/software-documentation>. Päivitetty 20.3.2024. Luettu 15.3.2025.

2 What is User Documentation? 2021. Verkkoaineisto. Technical Writer HQ. <https://www.youtube.com/watch?v=bYxbwsGG3Zo>. Katsottu 16.3.2025.

3 Pavlenko, Maria. 2023. Technical Documentation in Software Development: Types, Best Practices, and Tools. Verkkoaineisto. Altexsoft. <https://www.altexsoft.com/blog/technical-documentation-in-software-development-types-best-practices-and-tools/>. Päivitetty 6.7.2024. Luettu 18.3.2025.

4 Poikela, Esa. 2008. Miten informaatio muuttuu osaamiseksi?. Teoksessa Sormunen, Eero & Poikela, Esa (toim.). Informaatio, informaatiolukutaito ja oppiminen. E-Kirja. Tampere University Press.

5 Malamed, Connie. 2010. The Small Print: Writing UI Instructions. Verkkoaineisto. UX Magazine. <https://uxmag.com/articles/the-small-print-writing-ui-instructions>. 17.5.2010. Luettu 13.4.2025

6 Docs and guides to work with the Unity ecosystem. Verkkoaineisto. Unity. <https://docs.unity.com/>. Luettu 15.4.2025.

7 idiomi. Verkkoaineisto. Tieteen termipankki. <https://tieteentermipankki.fi/wiki/Kielitiede:idiomi>. Luettu 20.5.2025.

8 Saarelma, Osmo. 2021. Värisokeus ja poikkeava näkö. Verkkoaineisto. Duodecim terveyskirjasto. <https://www.terveyskirjasto.fi/dlk00347>. 14.6.2021. Luettu 18.4.2025.

9 ISO/IEC 40500:2012. Information technology — W3C Web Content Accessibility Guidelines (WCAG) 2.0. Joint Technical Committee ISO/IEC JTC1.

10 Jewett, Peter. 2020. WCAG Version History. Verkkoaineisto. Accessible Web. <https://accessibleweb.com/wcag/wcag-version-history/>. 12.10.2020. Luettu 1.6.2025.

11 WCAG 2 Overview. Verkkoaineisto. W3C.
<https://www.w3.org/WAI/standards-guidelines/wcag/>. Luettu 26.4.2025.

12 Creating a 3D Game. Verkkoaineisto. Unity.
<https://docs.unity3d.com/Manual/Quickstart3DCreate.html>. Luettu 15.4.2025.

13 3D-mallin kuva. Verkkoaineisto. Unity.
<https://docs.unity3d.com/uploads/Main/quickstart-3D-Graphics.png>. Luettu 15.4.2025.

14 Rigid Body Dynamics. Verkkoaineisto. NVIDIA.
<https://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/guide/Manual/RigidBodyDynamics.html>. Luettu 17.4.2025.

15 Game Server Architecture. Verkkoaineisto. Metaplay.
<https://docs.metaplay.io/game-server-programming/introduction-to-the-game-server-architecture.html#game-logic-execution-model>. Luettu 27.4.2025.

16 Pelilogiikan suoritusmallin kuvaaja. Verkkoaineisto. Metaplay.
<https://docs.metaplay.io/assets/execution-sequence-diagram.ZPEOGafC.png>. Luettu 27.4.2025.

17 Introduction to Entities and Actors. Verkkoaineisto. Metaplay.
<https://docs.metaplay.io/game-server-programming/introduction-to-entities-and-actors.html#entity-sharding-and-configuration>. Luettu 27.4.2025.

18 Custom Server Entities. Verkkoaineisto. Metaplay.
<https://docs.metaplay.io/game-server-programming/how-to-guides/custom-server-entities.html#dependencies>. Luettu 27.4.2025.

19 Yrityshaku. Verkoaineisto. Asiakastieto.
<https://www.asiakastieto.fi/yritykset/fi/playnest-oy/33163367/yleiskuva>. Luettu 16.4.2025.

20 Publish anything, fast. Verkkoaineisto. Notion.
<https://www.notion.com/product/sites>. Luettu 8.3.2025.

21 Git Branch. Verkkoaineisto. W3Schools.
<https://www.w3schools.com/git/git_branch.asp>. Luettu 20.4.2025.

22 Collision2D. Verkkoaineisto. Unity. <https://docs.unity3d.com/ScriptReference/Collision2D.html>. Luettu 19.4.2025.

23 AGameState. Verkkoaineisto. Epic Games. <https://dev.epicgames.com/documentation/en-us/unreal-engine/API/Runtime/Engine/GameFramework/AGameState>. Luettu 19.4.2025.