LAB University of Applied Sciences

Kovapintaisten videopeliympäristöjen 3D-mallintaminen

LAB-ammattikorkeakoulu
Insinööri (AMK), Tieto- ja viestintätekniikka
2025
Jeremia Huhtinen

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Jeremia Huhtinen	Opinnäytetyö, AMK	2025
	Sivumäärä 37	

Työn nimi

Kovapintaisten videopeliympäristöjen 3D-mallintaminen

Tutkinto ja koulutusala

Insinööri (AMK), Tieto- ja viestintätekniikka

Toimeksiantajaorganisaatio (jos opinnäytetyöllä on toimeksiantaja)

Varattu Valo Games Oy

Tiivistelmä

Opinnäytetyössä tarkasteltiin kovan pinnan 3D-ympäristömallinnusta videopeleissä keskittyen modulaaristen resurssien luontiin ja optimointiin reaaliaikaista renderöintiä varten Unreal Engine 5 -pelimoottorissa. Tarkoituksena oli luoda modulaarinen sarja kovapintaisia 3D-malleja, joiden avulla voidaan rakentaa erilaisia tutkimusasemia peliympäristöksi. Mallit luotiin Thalassophobia-nimiseen peliin, joka on yrityksen Varattu Valo Games kehityksessä oleva kauhupeli.

Työ suoritettiin käyttämällä Blender-sovellusta 3D-mallinnukseen, Adobe Substance 3D Painter -sovellusta teksturointiin ja Unreal Engine 5 -pelimoottoria testaukseen, arviointiin ja resurssien integrointiin. Tutkimuksessa käytettiin erilaisia mallinnus- ja teksturointitekniikoita, kuten modulaaristen resurssien luomista, trimmausarkkeja ja tekstuurikokoelmia, samalla kun käsiteltiin suorituskyvyn optimointistrategioita, kuten piirtokutsujen vähentämistä.

Tulokset osoittavat, että hyvin jäsennelty modulaarinen työnkulku parantaa merkittävästi resurssien uudelleenkäytettävyyttä ja vähentää kokonaismuistia ja laskentakustannuksia. Tulokset näyttävät myös, että tekniikat, kuten instanssit ja tekstuurien pakkaaminen, auttavat minimoimaan piirtokutsuja, mikä parantaa reaaliaikaisen renderöinnin suorituskykyä. Tulosten perusteella voidaan päätellä, että huolellinen suunnittelu modulaarisessa työnkulussa on olennaista peliympäristöjen mallintamisessa ja optimoinnissa.

Asiasanat

3D-mallinnus, pelimallinnus, pelikehitys, Blender, Unreal Engine

Abstract

Author(s)	Type of Publication	Published
Jeremia Huhtinen	Thesis, UAS	2025
	Number of Pages 37	

Title of Publication

3D modeling of hard-surface video game environments

Degree, Field of Study

Engineer (UAS), Information and Communications Technology

Organization of the client (if the thesis work is commissioned by another party)

Varattu Valo Games Oy

Abstract

This thesis examined hard-surface 3D environment modeling for video games, with a focus on optimizing modular assets for real-time rendering in Unreal Engine 5. The goal was to create a modular set of 3D hard-surface models that can be used to build different research stations as game environments. These models were created for a game called Thalassophobia, which is a horror game in development by Varattu Valo Games.

The study was conducted using Blender for 3D modeling, Adobe Substance 3D Painter for texturing, and Unreal Engine 5 for testing, evaluation and asset integration. The study utilized various modeling and texturing techniques, such as modular asset creation, trim sheets and texture atlases while also addressing performance optimization strategies, such as reducing draw calls.

The results show that a well-structured modular workflow significantly improves asset reusability and reduces overall memory and computational costs. The results also demonstrate that techniques such as instanced static meshes and texture packing help minimize draw calls, which improves real-time rendering performance. Based on the results, it can be concluded that careful planning in modular workflow and material usage is essential in modeling and optimizing game environments.

Keywords

3D modeling, game modeling, game development, Blender, Unreal Engine

Sisällys

1 Johdanto

Opinnäytetyön tarkoituksena on suunnitella ja toteuttaa modulaarinen kovapintainen 3D-ympäristö, joka on optimoitu suorituskykyyn ja uudelleenkäytettävyyteen. Projekti keskittyy hylättyyn tutkimusasemaan ja pyrkii luomaan joukon modulaarisia resursseja, joista voidaan rakentaa useita erilaisia rakennuksia peliympäristöön. Modulaarinen lähestymistapa mahdollistaa joustavien ja skaalautuvien ympäristöjen luomisen. Työssä käytetyt ensisijaiset ohjelmistotyökalut ovat Blender 3D-mallinnukseen, Adobe Substance 3D Painter teksturointiin ja Unreal Engine 5 peliin integrointiin.

Työn toimeksiantaja on indie-pelinkehitysstudio Varattu Valo Games Oy. Työ toteutetaan yrityksen uudelle pelille nimeltä Thalassophobia. Thalassophobia on vedenalainen sukelluskauhupeli, jota voidaan pelata enintään neljän pelaajan kesken.

3D-mallinnus on tekniikka, jota käytetään monilla eri aloilla, mukaan lukien elokuva-, arkkitehtuuri-, virtuaalitodellisuus- ja ennen kaikkea videopelien kehittämisessä. Se sisältää kolmiulotteisten objektien luomisen käyttämällä erikoisohjelmistoa objektin muodon, tekstuurin ja muiden ominaisuuksien määrittämiseen. (Gorman & Vakulich 2021.) Videopelien 3D-mallinnus eroaa kuitenkin perinteisestä 3D-mallinnuksesta useilla keskeisillä tavoilla. Pelimallit on optimoitava reaaliaikaista renderöintiä varten, koska niiden on toimittava tehokkaasti interaktiivisissa ympäristöissä. Toisin kuin elokuva- tai animaatiomalleissa, jotka voivat olla erittäin yksityiskohtaisia ja valmiiksi renderöityjä, pelimallit on optimoitava huolellisesti suorituskykyä varten eri alustoille.

Opinnäytetyön painopiste on kovan pinnan mallintaminen, joka käsittelee jäykkiä, muotoutumattomia esineitä, kuten koneita, ajoneuvoja ja arkkitehtonisia elementtejä, jotka ovat peliympäristöjen kehittämisen avainkomponentteja. Opinnäytetyössä tutkitaan videopelien 3D-mallinnuksen taustalla olevaa teoriaa, tekniikoita ja työnkulkua, joiden avulla luodaan realistisia, reaaliaikaiseen renderöintiin optimoituja 3D-malleja. Lisäksi työ kattaa näiden mallien integroinnin Unreal Engine 5 -pelimoottoriin, sekä tekstuurien, materiaalien ja suorituskyvyn optimointistrategioiden soveltamisen kehitysprosessissa. Opinnäytetyön tavoitteena on antaa kattava käsitys 3D-mallinnuksen keskeisestä roolista pelikehityksessä.

2 Kovan pinnan mallinnuksen perusteet

2.1 3D-mallinnus

3D-mallinnus on prosessi, jossa luodaan, esitetään ja manipuloidaan kolmiulotteisia objekteja tai pintoja tietokoneympäristössä käyttämällä erikoisohjelmistoja. Mallinnetut objektit määritellään niiden koon, muodon ja tekstuurin mukaan, jotka kuvataan kolmiulotteisen avaruuden koordinaattien avulla. (Gorman & Vakulich 2021.) 3D-mallit muodostuvat erikokoisista monikulmioista eli polygoneista. Nämä monikulmiot ovat tyypillisesti litteitä, kolmisivuisia tai nelisivuisia muotoja, ja niiden järjestely määrittää mallin yleisen geometrian. Näitä geometrisia muotoja voidaan edelleen jakaa kärkipisteiksi ja reunoiksi x-, y- ja z-koordinaateissa sekä -linjoissa. (Awati 2024.) Mitä enemmän pisteitä ja reunoja mallissa on, sitä yksityiskohtaisempi ja monimutkaisempi mallista voi tulla.

Mallin topologia, tai sen monikulmioiden järjestely ja virtaus, on ratkaisevan tärkeää oikean muodonmuutoksen varmistamiseksi animaation aikana ja optimaalisen suorituskyvyn varmistamiseksi, kun sitä käytetään reaaliaikaisissa ympäristöissä, kuten videopeleissä. Hyvin topologisoidussa mallissa polygonit jakautuvat tasaisesti, pinta on sujuva, ja reunavirta luo jatkavuuden, joka tukee mallin muotoa ja mahdollisia liikkeitä. (GarageFarm.)

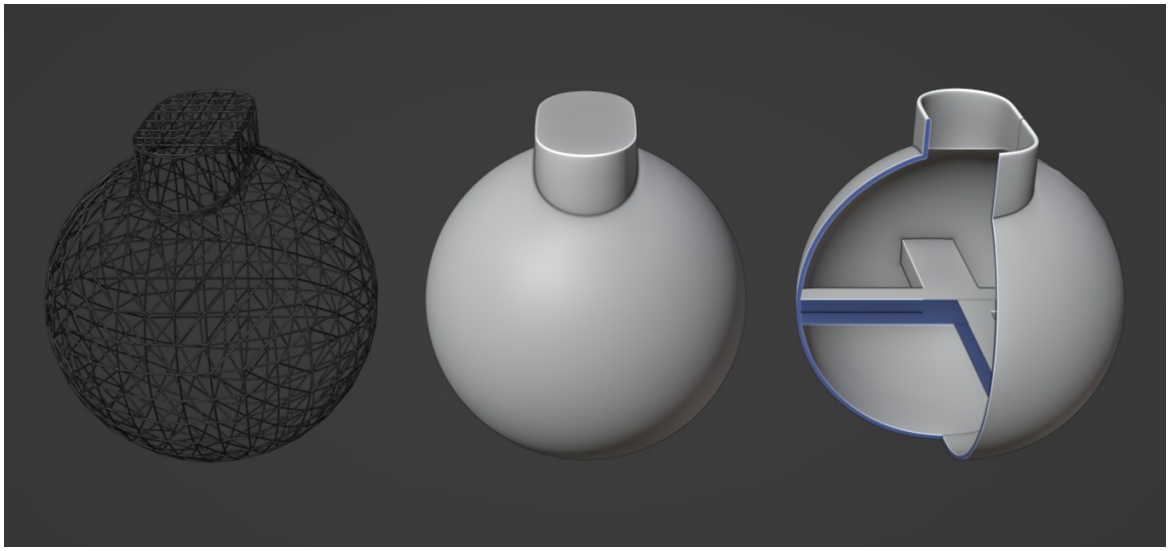
2.1.1 Mallinnusohjelmat

Usein käytettyjä ja tunnettuja mallinnusohjelmia ovat esimerkiksi Maya, Houdini, Blender, 3ds Max, Cinema4D ja ZBrush. Eri mallinnusohjelmat usein erikoistuvat johonkin tiettyyn tarkoitukseen, kuten erikoistehoisteisiin, animointiin, veistämiseen, liikegrafiikkaan tai peligrafiikkaan ja tarjoavat erilaisia erikoistyökaluja. (Katatikarn 2024.) Vain Blender on näistä ilmainen, jonka takia se on suosittu aloittelijoiden ja pienten studioiden keskuudessa. Blenderin suosiota lisää myös sen monipuoliset ominaisuudet ja avoin lähdekoodi, joka takaa sen jatkuvan kehityksen. Pelialalla tunnetuimmat ja laajasti käytössä olevat mallinnusohjelmat Blenderin lisäksi ovat Maya, 3ds Max ja ZBrush (Koneteo 2024). Nämä ovat laajassa käytössä erityisesti suuremmissa AAA-pelistudioissa.

Tuotantoputkissa on yleistä käyttää useita ohjelmia yhdessä, kuten ZBrush korkearesoluutioiseen kuvanveistoon, Maya luurangon luontiin ja animaatioihin ja Substance 3D Painter teksturointiin (Koneteo 2024). Jotkin ohjelmat, kuten Houdini, erottuvat joukosta solmupohjaisella prosessinkulullaan, joka on hyödyllinen laajamittaisten resurssien ja simulaatioiden, kuten maasto- tai tuhovaikutusten, luomiseen (Magee 2022). Mallinnusohjelmiston valinta riippuu projektin erityistarpeista, tiimin koosta, budjetista ja halutusta työnkulun tehokkuudesta.

2.1.2 Mallinnustekniikat

Yleiset mallinnustekniikat voidaan jakaa kolmeen osaan (Kuva 1). Pintamallinnus on tekniikka, jossa kolmiulotteisilla objekteilla on kiinteä pinta. Kiinteän pinnan vuoksi objekti on katsottavissa mistä tahansa kulmasta, koska se keskittyy vain mallin ulko-osaan. Pintamallinnuksessa mallin mittatarkkuudella ei ole väliä. Tekniikkaa, jossa simuloidaan sekä mallin sisä- että ulkopuolta, kutsutaan tilavuusmallinnukseksi. Toisin kuin pintamallinnuksessa, tilavuusmallinnusta soveltavien objektien täytyy olla todella mittatarkkoja. Kolmatta mallinnustekniikkaa kutsutaan rautalankamallinnukseksi. Rautalankamallinnuksessa pintoja ei käytetä ollenkaan, vaan mallin muodon määrää vain sen reunat ja kärkipisteet, jotka ovat usein kolmion muodossa. (Gorman & Vakulich 2021.)



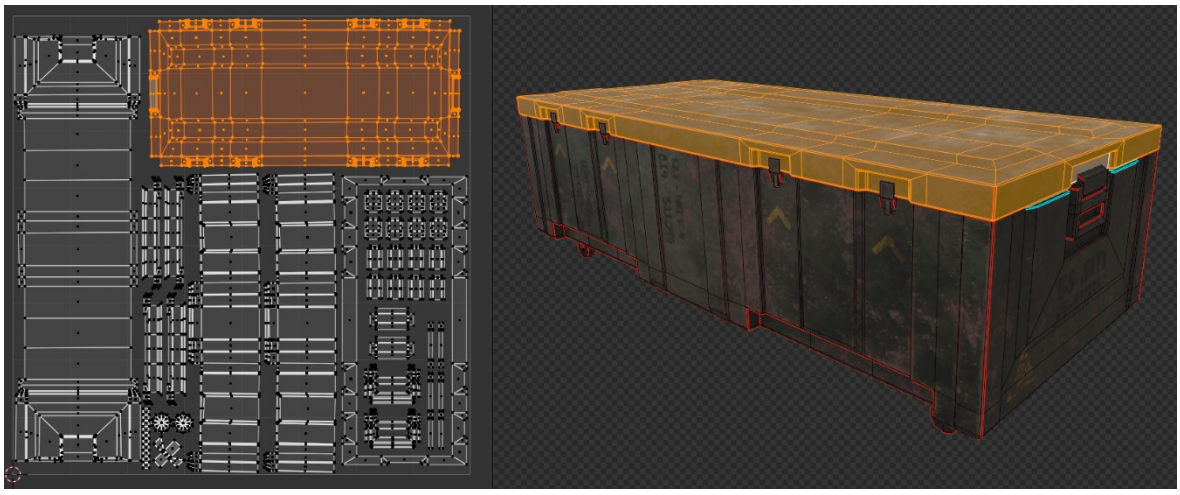
Kuva 1. Rautalankamallinnus, pintamallinnus ja tilavuusmallinnus

Näitä mallinnustekniikoita käytetään eri toimialoilla niiden erityistarpeen mukaan. Pintamallinnusta käytetään yleisimmin aloilla, joissa visuaalinen ilme ja ulkoiset yksityiskohdat ovat etusijalla sisäisten rakenteiden sijaan, kuten animaatio-, elokuva- ja videopelialoilla. Tilavuusmallinnusta käytetään usein lääketieteellisessä kuvantamisessa, suunnittelusimulaatioissa, tieteellisessä visualisoinnissa sekä tuote- ja mekaniikkasuunnittelussa, joissa sisäinen tilavuus ja tarkat fysikaaliset ominaisuudet ovat olennaisia. Rautalankamallinnusta käytetään usein suunnittelun ja prototyyppien valmistuksen alkuvaiheissa, erityisesti arkkitehtuurissa ja tuotesuunnittelussa, jossa painopiste on muodon ja rakenteen visualisoinnissa. (Gorman & Vakulich 2021.)

2.1.3 UV-kartat, UV-kartoitus ja UV-saaret

UV-kartta on 2D-esitys 3D-mallin pinnasta. U ja V ovat kaksiulotteisia tekstuurikoordinaatteja, jotka vastaavat 3D-mallin geometrian kärkipisteitä, kun ne levitetään kaksiulotteiselle alueelle (Pluralsight 2022). Tällä kartoitusprosessilla varmistetaan, että tekstuurit kietoutuvat oikein mallin pintaan ilma vääristymisiä. Oikea UV-kartoitus estää tekstuurien venymistä, parantaa tekstuurin resoluutiota ja optimoi visuaalisen laadun 3D-ympäristöissä. Tämä on erityisen tärkeää pelikehityksessä, jossa suorituskyky ja tehokkuus ovat ratkaisevia.

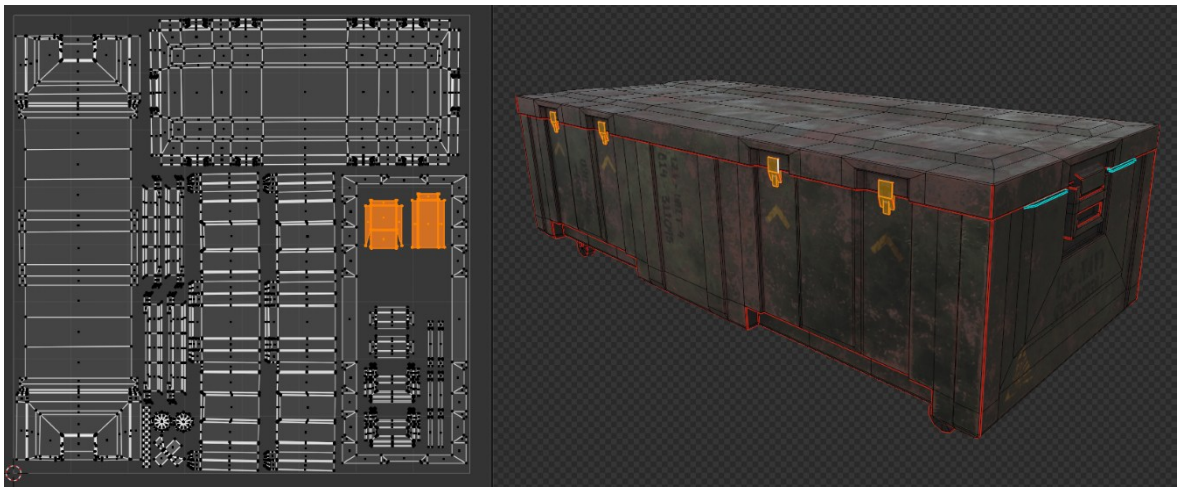
UV-kartoitus aloitetaan, kun 3D-malli on valmis. 3D-malli puretaan sijoittamalla saumoja strategisesti mallin geometriaan, jotta mallin pinnasta voidaan luoda 2D-esitys. Tämä prosessi tunnetaan nimellä UV-kartoitus. UV-kartoitus edellyttää tekstuurin kuvittelemista kääpaperina mallin ympärille ja sen määrittämisen, missä luonnollisia saumoja esiintyisi. (Pluralsight 2022.) Saumat tulee sijoittaa vähemmän näkyville alueille, kuten leuan tai käsivarsien alle orgaanisissa humanoidimalleissa. Kovapintaisissa mekaanisissa malleissa hyvä sauman sijoitus seuraa kohteen luonnollisia paneeliviivoja, reunoja tai piilotettuja alueita, kuten mekaanisten osien alaosaa. Tavoitteena on tasapainottaa tekstuurin laatua pitäen saumat piilossa tai luonnollisesti integroituna mallin suunnitteluun. Kuvassa 2 havainnollistetaan 3D-mallin UV-karttaa. Saumat on merkattu punaisilla viivoilla.



Kuva 2. UV-kartta ja saumat

UV-saaret, jotka tunnetaan myös nimellä UV-kuoret Mayassa ja UV-klusterit 3ds Maxissa, ovat yhdistettyjen UV-koordinaattien ryhmiä, jotka muodostavat erillisiä osia 3D-mallin

purettuun UV-karttaan. Nämä saaret määrittävät, miten tekstuuri sijoittuu mallin pinnalle varmistaen oikean kohdistuksen. Jos UV-saaret asetetaan päällekkäin UV-editorissa, tekstuuri saattaa toistua ei-toivotulla tavalla. Tätä tulisi yleensä välttää, ellei päällekkäisyys ole tarkoituksellista. Esimerkiksi joissain videopelimallien tekstuureissa saarien päällekkäisyys ja tekstuurien toistaminen voi olla hyödyllistä ja tehokasta. (Pluralsight 2022.) Kuvassa 3 UV-saaret on korostettu oranssilla ja samanlaiset UV-saaret ovat aseteltu päällekkäin. Tämä mahdollistaa saarten skaalaamista isommaksi, mikä antaa näille korostetuille osille korkeamman tekstuuriresoluution, mutta osat tulevat myös jakamaan saman tekstuurin. Jos kaikille osille halutaan uniikit tekstuurit, ne jaetaan erikseen, kuten kuvassa 2.

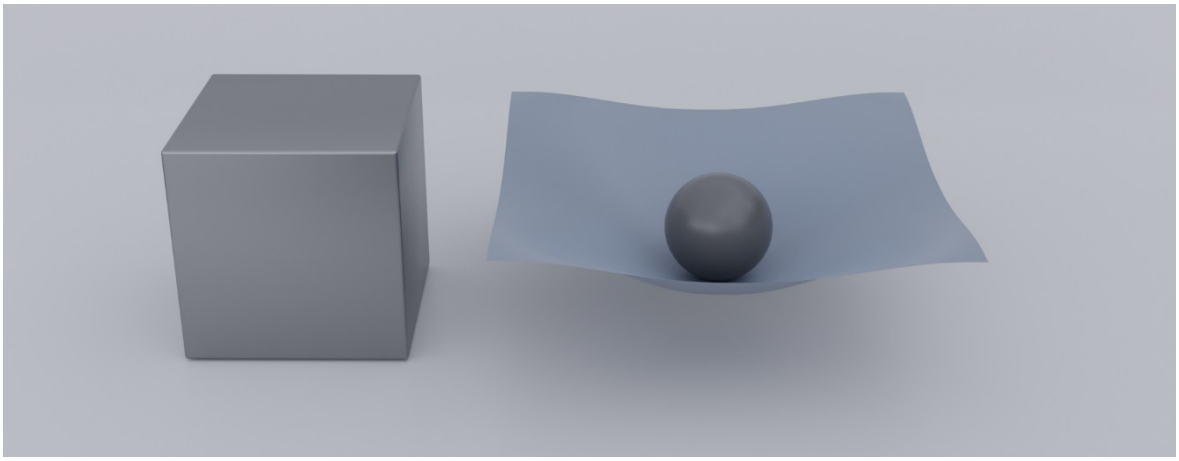


Kuva 3. UV-saaret ja saarten päällekkäisyys

2.1.4 Kovan pinnan mallinnus

Kovan pinnan mallinnus on mallinnustapa, jota käytetään mallintamaan staattisia sileä- tai kovapintaisia elottomia objekteja, kuten huonekaluja, ajoneuvoja, koneita, rakennuksia tai aseita. Useimmat ihmisen tekemät esineet voidaan luokitella kovapintaisiksi esineiksi. (Belec 2022, 1.) Kovan pinnan mallinnus perustuu tekniikoihin, kuten Boolean operaatioihin, tarkkoihin reunasilmukoihin ja ruudukkopohjaisiin lähestymistapoihin tarkkuuden varmistamiseksi. Kovan pinnan tai orgaanisen pinnan määritelmä ei kuitenkaan ole universaali, vaan se riippuu usein mallintajan omasta määritelmästä. Silti jokaisella 3D-mallintajalla olisi hyvä olla yleinen käsitys molemmista mallinnustavoista ja syvällisempi ymmärrys niihin liittyvistä tekniikoista.

Kovapintaisen mallin yleisesti määrittää sen tyypilliset tekniset ominaisuudet, kuten terävät reunat, tasaiset pinnat ja irtonaisten osien erotus. Ne ovat jäykkiä kappaleita tai esineitä, joiden liike on rajoitettu vain tiettyyn mekaaniseen liikkeeseen ja niiden muoto ei muutu. (Belec 2022, 1.) Jos kovapintaiselle esineelle luodaan animaatio, joka muokkaa sen pintojen geometriaa, se muuttuu orgaaniseksi. Kovapintaisten mallien staattisuuden ansiosta niiden topologialla ei ole yhtä suurta merkitystä kuin orgaanisilla malleilla (Wingfox 2022). Kovan pinnan ja orgaanisen pinnan erot nähdään kuvassa 4.



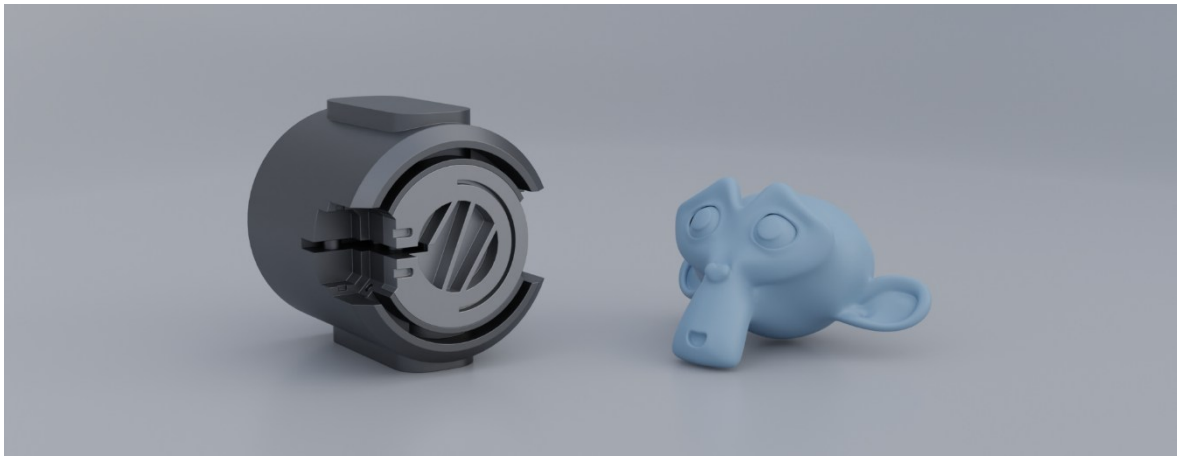
Kuva 4. Kova pinta ja orgaaninen pinta

2.1.5 Orgaanisen pinnan mallinnus

Orgaanisen pinnan mallinnus on mallinnustapa, jota käytetään mallintamaan esimerkiksi hahmoja, eläimiä, vaatteita ja muita eläviä asioita (Belec 2022, 1). Orgaanisille malleille ominaisia piirteitä ovat tasaisesti virtaavat ja taipuvat reunat sekä muotojen sujuva siirtyminen toisiin (Wingfox 2022). Orgaanisilla malleilla ei siis usein ole ollenkaan teräviä reunoja ja ne ovat usein helposti animoitavia. Hyvät animaatiot vaativat hyvää topologiaa. Jos topologia tehdään hyvin, kärkipisteet järjestetään niin, että objekti pystyy taipumaan ja muotoutumaan sujuvasti animaatioita varten. Orgaanisen ja kovapintaisen mallin erot nähdään kuvassa 5.

Orgaanisen pinnan mallinnuksessa tärkeintä on aina käyttää nelisivuisia polygoneja, koska se auttaa hyvän topologian saavuttamisessa. Orgaaniset mallit ovat usein haastavampi mallintaa, koska orgaanisen mallinnuksen hallitsemiseksi on tutkittava erilaisten elävien olioiden anatomiaa, jotta voidaan luoda hyvä ja realistinen topologia. Orgaanisessa mallinnuksessa tulee myös esiin monia vaikeita mallinnustekniikoita, kuten veistäminen.

Ammattilaatuista digitaalista veistämistä varten käytetään usein kalliita sille tarkoitettuja laitteistoja ja ohjelmistoja. Tunnetuin veisto-ohjelmisto on ZBrush, joka maksaa jopa 55,35 euroa kuukaudessa tai 441,57 euroa vuosittain (Maxon 2025). Digitaalinen veistäminen onnistuu myös Blender-mallinnusohjelmalla, mutta ZBrush hallitsee ammattimaista 3D-veistoa erikoistyökalujensa, suorituskykynsä ja alan laajan käyttöönoton ansiosta.



Kuva 5. Kovapintainen malli ja orgaaninen malli

2.1.6 Pelimallinnus

Pelimallinnuksessa hyödynnetään sekä kovan pinnan että orgaanisen pinnan mallinnusta. Pelimallinnuksessa luodaan malleja, jotka on suunniteltu toimimaan interaktiivisissa, reaaliaikaisissa ympäristöissä pelimoottorin sisällä. Toisin kuin monissa muissa 3D-mallinnusta hyödyntävissä mediamuodoissa, kuten elokuvatuotannossa tai arkkitehtuurissa, pelimallinnus priorisoi suorituskykyä ja interaktiivisuutta.

Mallit on optimoitava pelaamista varten keskittyen erilaisiin näkökohtiin, kuten reaaliaikaiseen valaistukseen, animaatioiden yhteensopivuuteen ja pelimoottorin suorituskykyyn. Suorituskykyrajoituksien takia mallin geometria ei voi olla yhtä yksityiskohtaista, kuin muussa 3D-mallinnuksessa.

2.2 Suorituskyvyn optimointi pelimallien luomisessa

Pelimallintamisessa korkean yksityiskohtaisuuden saavuttaminen pienellä polygonimäärällä on olennaista suorituskyvyn optimoinnin kannalta. Tämä saavutetaan käyttämällä erilaisia tekniikoita, jotka tasapainottavat visuaalisen laadun ja suorituskyvyn tehokkuuden.

Yksi yleisimmistä menetelmistä on luoda mallista kaksi versiota: high-poly ja low-poly. High-poly-mallissa on tiheä geometria, joka mahdollistaa monimutkaiset ja tarkat yksityiskohdat ja sileät pinnat. Suuren prosessointitehon tarpeen vuoksi se ei kuitenkaan sovellu reaaliaikaiseen renderöintiin. Low-poly-malli puolestaan on optimoitu suorituskykyä varten käyttämällä vähemmän polygoneja objektin muodon määrittämiseen, mikä tekee siitä tehokkaamman reaaliaikaisissa sovelluksissa. (Furneri 2022.)

High-poly-mallin yksityiskohtien säilyttämiseksi polygonien lukumäärää lisäämättä käytetään prosessia, jota kutsutaan nimellä ”Baking”. Tämä tarkoittaa tekstuurikarttojen, kuten normaali-, korkeus- ja ympäristön okkluusiokarttojen (AO) generoimista high-poly-mallista. Näitä karttoja voidaan soveltaa low-poly-mallin teksturoinnissa pinnan yksityiskohtien, varjostuksen ja syvyyden simuloimiseksi, mikä luo illuusion monimutkaisuudesta ilman ylimääräistä geometriaa. (Furneri 2022.)

Toinen yleinen optimointitekniikka on nimeltään Level of Detail (LOD). Sitä käytetään vähentämään 3D-mallien renderöinnin laskentakuormaa. Malleista luodaan useita versioita, joiden yksityiskohtaisuus vaihtelee. Yksityiskohtaisin versio näytetään, kun kohde on lähellä kameraa, kun taas vähemmän yksityiskohtaisempia versioita käytetään, kun mallit ovat kauempana kamerasta.

2.2.1 Teksturointitekniikat kovapintaisille malleille

Kovapintaisten mallien ominaisuudet mahdollistavat useiden eri teksturointitekniikoiden hyödyntämisen, sisältäen tekstuurikokoelmat, trimmausarkit, toistettavat tekstuurit, dekaalit ja yksityiskohtakartat. Teksturointitekniikoiden valitseminen riippuu paljon siitä, millainen 3D-malli on ja mikä on sen käyttötarkoitus. Lähestulkoon kaikki teksturointitekniikat vaativat hyviä UV-karttoja.

Tekstuurikokoelmien (Texture Atlas) käyttö on yksi yleisimmistä teksturointitekniikoista. Tekstuurikokoelmassa yhdistetään useita pienempiä eri mallien tai mallin osien tekstuureja yhdeksi suuremmaksi tekstuuriksi, jolloin ne voidaan renderöidä tehokkaammin (Lark 2024). Vaikka tekstuurikokoelmat ovat yleensä suunniteltu tiettyjen mallien UV-karttojen perusteella, niitä voidaan käyttää uudelleen useissa malleissa, jos niiden UV-asettelu suunnitellaan vastaavasti tai tekstuurikokoelma koostuu monesta tasaisesta tekstuurista, jotka ovat tarkoitettu jaettavaksi. Kuva 6 havainnollistaa erilaisia tekstuurikokoelmia.

Kuva 6. Tekstuurikokoelmat

Trimmausarkit (Trim sheets) ovat uudelleenkäytettäviä jatkuvia tekstuuriarkkeja, jotka sisältävät erilaisia esivalmistettuja yksityiskohtia. Ne säästävät muistia ja lisäävät ympäristöön paljon yksityiskohtia. Niitä käytetään yleisesti malleissa, joissa on paljon toistuvia elementtejä, kuten modulaarisissa malleissa. Yksi tärkeimmistä eroista trimmausarkkien kanssa on se, että 3D-mallin UV-saaret ovat todennäköisesti päällekkäisiä ja ylittävät UV-rajat. Tämä on mahdollista, koska trimmausarkin tekstuurit ovat toistettavia U- ja V-suunnassa. (Burns 2023.) Trimmausarkit (Kuva 7) ovat teollisuusstandardi varsinkin kovapintaisten ympäristöjen teksturoinnissa, sillä ne tarjoavat erinomaisen tasapainon tehokkuuden ja laadun välillä.

Kuva 7. Trimmausarkki

Toistettavat tekstuurit (Tileable textures) ovat yksinkertaisia saumattomia tekstuureja, joita voidaan toistaa tasaisella pinnalla ilman näkyviä saumoja. Ne ovat myös suunniteltu uudelleenkäytettäviksi, mutta toisin kuin trimmausarkit, ne koostuvat vain yhdestä saumattomasta tekstuurista. Tämän takia niitä sovelletaan lähinnä trimmausarkkien tekemisessä tai tasaisten pintojen, kuten lattioiden ja seinien teksturoinnissa.

Decalit (Decals) ovat pieniä tekstuuriyksityiskohtia, jotka sovitetaan erikseen päätekstuurista. Niitä käytetään lisäyksityiskohtien, kuten lian, naarmujen, etikettien tai vaurioiden lisäämiseen. Dekaalit mahdollistavat visuaalisen monimuotoisuuden ilman, että perustekstuuria muokataan. Niitä voidaan myös käyttää uudelleen usein, koska ne eivät ole osa päätekstuuria. (A23D 2023.)

Yksityiskohtakarttojen (Detail maps) avulla voidaan päällystää toinen joukko tekstuureita päätekstuurin päälle. Tyypillisesti yksityiskohtakartta skaalautuu useita kertoja kohteen pinnan poikki lisätäkseen materiaaliin pieniä yksityiskohtia. (Unity.)

2.2.2 Tekstuurien optimointi ja materiaalitehokkuus

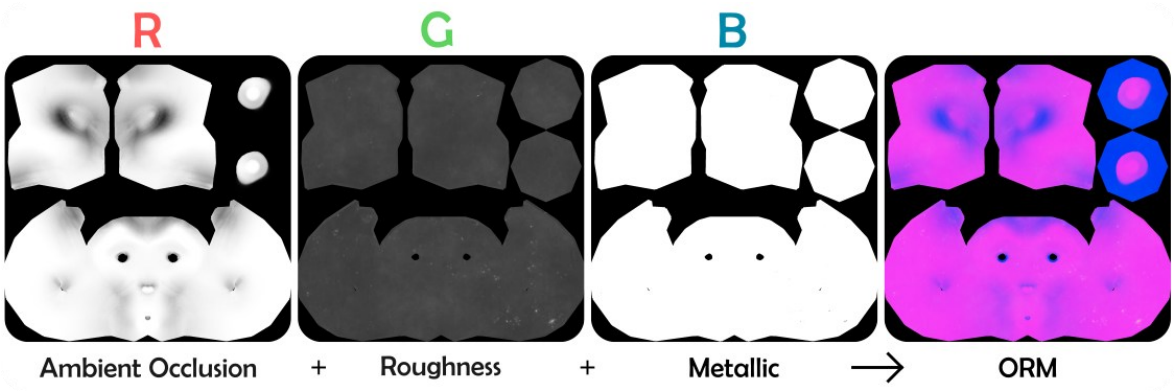
Materiaalit määrittävät, miltä objekti näyttää 3D-näkymässsä, mukaan lukien sen värit, pinnan ominaisuudet ja miten se on vuorovaikutuksessa valon kanssa. Toisin kuin tekstuurit, jotka ovat 2D-kuvia, materiaalit käyttävät useita tekstuurikarttoja, kuten diffuusi-, normaali- ja karheuskarttoja, luodakseen realistisia pintoja. Fyysisesti pohjautuvassa renderöinnissä (PBR) materiaalit hyödyntävät varjostimia simuloidakseen oikean maailman materiaaleja. (Blender 2025.) Tekstuureja käytetään siis materiaalien varjostin-parametreinä.

Tekstuurien optimointi on tärkeää suorituskyvyn ja muistin säästämisen kannalta. Tekstuurien optimointi aloitetaan jo UV-kartoitusvaiheessa. Pääsääntöisesti UV-kartat järjestetään niin, että UV-alueelle jää mahdollisimman vähän tyhjää tilaa. Kaikki tyhjä tila on hukkaan menevää muistia, koska sitä voitaisiin käyttää teksturoinnissa. Tietyissä teksturointitekniikoissa, kuten trimmausarkeissa, tämä ei ole tarpeellista, jos itse tekstuurit kattavat lähes koko UV-alueen ja UV-saaret asetellaan tekstuurin päälle vasta teksturoinnin jälkeen.

Tekstuurikokoelmia käyttäessä UV- ja materiaalitehokkuutta voidaan parantaa yhdistämällä monien eri mallien UV-karttoja yhdeksi UV-kartaksi. (Yang 2025.) Kaikki mallit, jotka jakavat UV-tilaa, voivat käyttää samaa materiaalia, joka edistää materiaalitehokkuutta. Haittapuoli tekstuurien ja UV-karttojen yhdistämisessä on se, että jos yhdistetyistä tekstuureista tehtyjä materiaaleja lähdetään muokkaamaan, muutokset otetaan käyttöön jokaisen mallin

pinnassa, joka jakaa samaa materiaalia. Tätä haittapuolta voidaan minimoida pelimoottorin puolella tekemällä materiaali-instansseja.

UV-karttojen pakkaamisen lisäksi voidaan myös pakata tekstuurikarttoja. Tätä kutsutaan kanavapakkaukseksi (Channel packing). Kanavapakkaus on tekniikka, jossa useita tekstuuritietotyyppejä tallennetaan yhteen tekstuuritiedostoon. Tämä on erityisesti hyödyllistä harmaasävyisten tekstuuridatan, kuten pinnan karheuden, metallisuuden ja ympäristön okkluusion tallentamisessa. Harmaasävyiset tekstuurit pakataan käyttämällä tekstuurin RGB-kanavia. Ympäristön okkluusio tallennetaan punaiseen kanavaan, pinnan karheus vihreään kanavaan ja metallisuus siniseen kanavaan. (Chadwick 2025.) Tällä tavoin voidaan pakata kolme eri tekstuuria yhteen, mikä vähentää tekstuurien määrää materiaalissa ja parantaa muistinhallintaa. Lopullista yhdistettyä tekstuuria kutsutaan nimellä ORM-kartta (Occlusion, roughness, metallic). Kuva 8 havainnollistaa ORM-kartan jaetut osat.



Kuva 8. ORM-kartta

Yksittäisistä tekstuureista voidaan myös luoda matalamman resoluution tekstuuriversioita, joita käytetään automaattisesti sen mukaan, kuinka kaukana kamera on objektista. Tätä tekniikkaa kutsutaan nimellä ”mipmaps” ja se toimii lähestulkoon samalla tavalla, kuin LOD-optimointitekniikka mallintaessa. Useat pelimoottorit luovat mipmapit automaattisesti tekstuurin tuonnin yhteydessä, joten niitä ei yleensä tarvitse tehdä manuaalisesti. (Unity.)

3 Videopeliympäristöt pelimoottorissa

3.1 Johdatus pelimoottoreihin

Pelimoottori on erikoistunut ohjelmistokehys, joka on suunniteltu ensisijaisesti videopelien kehittämiseen. Ajan myötä pelimoottorit ovat laajentuneet pelaamisen ulkopuolelle, ja niitä käytetään nykyään laajalti eri teollisuudenaloilla sovelluksiin, kuten visualisointiin, simulaatioihin ja virtuaaliseen yhteistyöhön. (Perforce 2023.) Yksi pelimoottoreiden perusominaisuuksista on käyttäjien vuorovaikutus, jonka avulla pelaajat tai kehittäjät voivat olla yhteydessä virtuaaliympäristöön. Pelimoottorin sovellusliittymä usein koostuu käyttöliittymästä, joka mahdollistaa pääsyn sen erilaisiin ominaisuuksiin ja asetuksiin ja 2D- tai 3D-näkymästä, jossa kehittäjät voivat manipuloida ympäristöä ja objekteja reaaliajassa.

Pelimoottorit tarjoavat kattavan valikoiman työkaluja ja toimintoja, mukaan lukien renderöintimoottorin reaaliaikaista 2D- ja 3D-grafiikkaa varten, valaistusjärjestelmät dynaamiseen ja staattiseen valaistukseen, fysiikka- ja törmäysjärjestelmät realististen vuorovaikutusten simulointiin, animaatiotyökalut hahmojen liikkeisiin ja muihin animaatioihin sekä tekoälyjärjestelmät NPC-käyttäytymiseen ja polunhakuun. Lisäksi pelimoottoreissa on äänimoottori äänitehosteiden ja tilaäänen hallintaan sekä komentosarja- ja logiikkajärjestelmä, joiden avulla kehittäjät voivat toteuttaa pelimekaniikkaa käyttämällä tuttuja ohjelmointikieliä. (Epic Games 2025.)

Pelikehitysalalla on sekä julkisesti saatavilla olevia ja yritysten sisäisesti luomia pelimoottoreita. Eri pelimoottorit tarjoavat tiettyjä ominaisuuksia erilaisiin tarpeisiin indie-projekteista AAA-peleihin. Julkisesti saatavista pelimoottoreista tunnetuimmat ovat Unity ja Unreal Engine niiden joustavuuden, tehokkaiden ominaisuuksien, laajan tuen ja käyttäjäyhteisöjen ansiosta. (Mitra ym. 2025.) Toisin kuin julkiset pelimoottorit, yritysten sisäiset moottori eivät ole ulkopuolisten kehittäjien saatavilla, vaan ne tarjoavat studioille täyden hallinnan teknologiaansa, mutta vaativat merkittäviä resursseja kehittämiseen ja ylläpitoon. Vehkalan (2023) mukaan pelimoottorin valinnassa kannattaa pohtia seuraavia asioita: kuinka hyvin tietty pelimoottori sopii pelinkehitystiimin vahvuuksiin, mikä on tiimin koko ja kokoonpano, mikä on pelin visio ja sen ydinpilarit ja mille pääalustoille peli julkaistaan.

3.2 Renderöinti pelimoottoreissa

Renderöinti on prosessi, jossa luodaan ja esitetään lopullinen visuaalinen tulos 2D-kuvana 3D-mallien, tekstuurien, valaistuksen ja erilaisten graafisten tehosteiden perusteella. Nykyaikaiset pelimoottorit käyttävät yhdistelmiä erilaisia renderöintitekniikoita reaaliaikaisen suorituskyvyn ja visuaalisen tarkkuuden saavuttamiseksi. Pelimoottorit ovat suunniteltu

reaaliaikaiseen renderöintiin ja sen optimointiin, jotta pelit toimivat mahdollisimman sujuvasti. Toisin kuin animaatioelokuvan esirenderöidyt grafiikat, joissa yhden ruudun renderöinti voi kestää tunteja, reaaliaikaisen renderöinnin on tuotettava useita kuvia sekunnissa, usein 60 tai enemmän. Monia pelimoottorin sisäisiä optimointimenetelmiä, kuten LOD-malleja, okkluusiokarsintaa ja piirtokutsujen vähennys tekniikoita, käytetään sujuvan kuvanopeuden saavuttamiseksi. Nykyaikaiset pelit käyttävät myös tekoälyyn perustuvia resoluution skaalausmenetelmiä nopeamman kuvanopeuden saavuttamiseksi renderöinnin jälkeen.

3.2.1 Renderöintiputket

Renderöintiputki on jäsennelty prosessi, joka määrää, kuinka grafiikkaa piirretään näytölle. Renderöintiputki voidaan yleensä jakaa kolmeen päävaiheeseen: sovellus-, geometria- ja rasterointivaiheeseen. Jokaisella vaiheella on tärkeä rooli 3D-datan muuntamisessa lopulliseksi renderöidyksi kuvaksi. (Agladze 2024.)

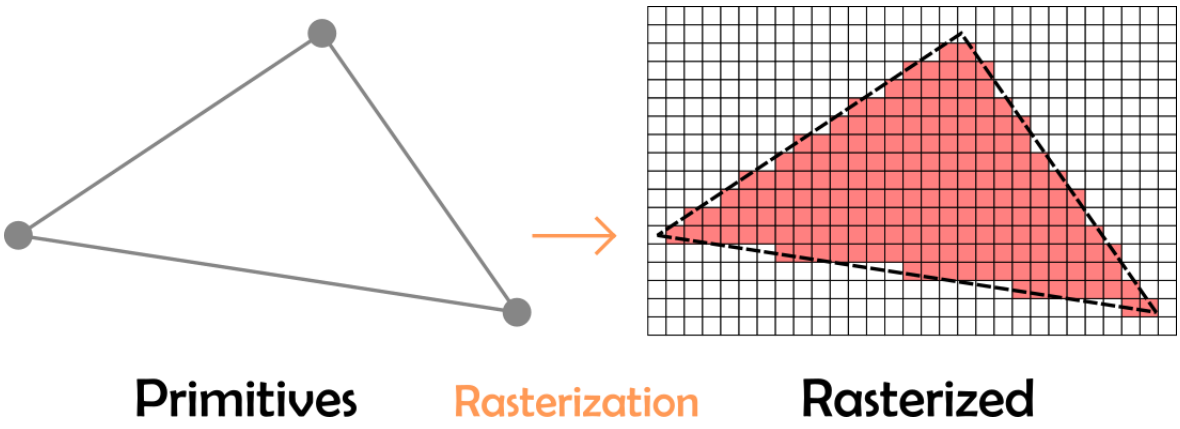
Sovellusvaihe on renderöintiputken ensimmäinen vaihe, jossa sijaitsee pelilogiikka ja sen hoitaa prosessori (CPU). Tämä vaihe sisältää syötteiden käsittelyn, pelin tilojen päivittämisen ja tietojen valmistelun renderöintiä varten. Sovellusvaiheen aikana moottori laskee kaikkien näkymien kohteiden sijainnit, suunnat ja animaatiot. Nämä tiedot järjestetään sitten muotoon, joka voidaan käsitellä tehokkaasti renderöintiputken myöhemmissä vaiheissa. Sovellusvaihe vaihtelee suuresti riippuen käytetystä pelistä ja pelimoottorista, koska siihen liittyy huomattava määrä korkean tason logiikkaa ja päätöksentekoa. (Agladze 2024.)

Geometriavaiheessa näytönohjaimen prosessori (GPU) alkaa käsitellä 3D-näkymää. Tämä vaihe keskittyy 3D-objektien muuntamiseen ja valaistukseen. Ensisijaisia prosesseja tässä vaiheessa ovat kärkipistekäsittely, primitiivinen kokoonpano ja leikkaus. Kärkipisteiden käsittely muuttaa 3D-mallien kärkipisteet objektiavaruudesta näyttötilaan. Primitiivinen kokoonpano on prosessi, jossa kärkipisteet järjestetään geometrisiksi primitiiveiksi, kuten kolmioiksi, viivoiksi ja pisteiksi. Leikkaus suoritetaan sitten kaikkien näkymän ulkopuolelle jäävien primitiivien poistamiseksi, mikä varmistaa, että vain näkyvät kohteet käsitellään. Geometriavaiheessa tehdään myös valaistuslaskemia, joissa määritetään, miten valo on vuorovaikutuksessa objektien pinnan kanssa. (Agladze 2024.)

3.2.2 Rasterointi ja reaaliaikainen renderöinti

Rasterointi on sekä perusvaihe renderöintiputkessa että yleisin käytössä oleva renderöintimenetelmä reaaliaikaisessa grafiikassa. Se on vastuussa 3D-mallien muuntamisisesta 2D-kuvaksi projisoimalla niiden geometriaa näytölle ja määrittämällä, kuinka ne pitäisi näyttää

pikseleinä. Rasterointivaihe sisältää useita tärkeitä vaiheita, kuten skannausmuunnoksen, varjostuksen ja teksturoinnin. Skannausmuunnos on prosessi, jossa määritetään, mitkä ruudun pikselit vastaavat geometristen primitiivien huippuja. Näihin pikseleihin sovelletaan sitten varjostusta käyttämällä valaistuslaskelmia, jotka lasketaan usein geometriavaiheessa, määrittääkseen kunkin pikselin värin ja intensiteetin. Varjostusmallit, kuten PBR (Physical Based Rendering) vaikuttavat siihen, miten pinnat reagoivat valoon. Teksturoinnissa tekstuurikartat lisätään varjostettuihin pikseleihin yksityiskohtien ja realistisuuden lisäämiseksi renderöityyn kuvaan. Tekstuurien suodatustekniikat, kuten bilineaarinen tai anisotrooppinen suodatus, parantavat kartoitettujen tekstuurien laatua. Rasteroinnin lopputulos on rasterikuva, joka on ruudukko pikseleitä, mikä edustaa lopullista renderöityä kehystä. (Caulfield 2018; Agladze 2024.) Kuvassa 9 havainnollistetaan rasterointia.



Kuva 9. Rasterointi

Renderöintimenetelmänä rasterointi on edelleen alan standardi reaaliaikaisessa sovelluksissa sen tehokkuuden ja soveltuvuuden vuoksi nykyaikaisiin näytönohjaimiin. Vaikka rasterointi on erittäin tehokasta ja optimoitu reaaliaikaisiin sovelluksiin, se ei luonnollisesti simuloi monimutkaisia optisia tehosteita, kuten heijastuksia tai globaalia valaistusta, minkä vuoksi nykyaikaisissa pelimoottoreissa käytetään usein lisäjälkikäsittely- tai hybriditekniikoita, kuten säteenseurantaa (Ray Tracing). (Caulfield 2018.)

3.2.3 Piirtokutsut

Piirtokutsu on prosessorin (CPU) näytönohjaimen prosessorille (GPU) tekemä pyyntö objektin tai objektiryhmän renderöimiseksi. Jokainen piirtokutsu sisältää tietoa siitä, mikä verkko piirretään, mitä materiaalia käytetään ja kuinka valaistusta ja varjostusta tulisi

käsitellä objektia kohtaan. CPU valmistautuu piirtokutsuun määrittämällä resursseja ja muuttamalla näytönohjaimen sisäisiä asetuksia. Näitä asetuksia kutsutaan renderöintitilaksi. Renderöintitilan muutokset, kuten materiaalin vaihto, ovat hyvin resurssi-intensiivisiä näytönohjaimen suorittamia toimintoja.

Vaikka näytönohjaimet ovat optimoitu käsittelemään monia piirtokutsuja, liian monet kutsut voivat johtaa suorituskyvyn pullonkauloihin. Tämän hallitsemiseksi pelimoottorit optimoivat piirtokutsuja ryhmittelyn (Batching), instanssien ja verkkojen yhdistämisen avulla (Unity; Epic Games).

3.3 Suorituskyvyn optimointi pelimoottoreissa

Suorituskyky on tärkeä elementti pelikehityksessä ja pelimoottorin valinnassa. Sillä varmistetaan, että pelit toimivat sujuvasti eri laitteistokokoonpanoissa ja alustoissa. Yksi tärkeimmistä optimointitekniikoista on piirtokutsujen optimointi, koska jokainen piirtokutsu lisää käsittelykuluja, ja liialliset piirtokutsut voivat johtaa suorituskykyongelmiin erityisesti monimutkaisissa näkymissä. Liialliset piirtokutsut voivat ylikuormittaa varsinkin prosessoria, koska prosessorin on käsiteltävä ja lähetettävä jokainen kutsu näytönohjaimelle ja piirtokutsuun valmistautuminen on resurssivaltaisempaa kuin itse piirtokutsu. (Unity.)

Piirtokutsujen optimointiin voidaan käyttää useita eri tekniikoita. Ennen sopivan optimointitekniikan soveltamista kuitenkin on tärkeä tietää, mikä näkymässä aiheuttaa suuren piirtokutsujen määrän. Piirtokutsut koostuvat tyypillisesti kahdesta ensisijaisesta osasta: renderöitävistä 3D-malleista ja materiaaleista. Jokainen ainutlaatuinen 3D-malli ja mallin materiaali vaativat erillisen piirtokutsun.

Ryhmittely (Batching) on prosessi, jossa useita objekteja ryhmitellään yhdeksi piirtokutsuksi. Pelimoottorit käyttävät kahta pääasiallista ryhmittelytapaa: staattinen ja dynaaminen ryhmittely. Staattista ryhmittelyä käytetään objekteille, jotka eivät liiku, pyöri tai skaalaa. Pelimoottori yhdistää useita staattisia verkkoja yhdeksi verkoksi, joka renderöidään yhdellä piirtokutsulla. Dynaamista ryhmittelyä käytetään objekteille, jotka liikkuvat, pyörivät tai skaalautuvat. Tämä tekniikka on rajoitetumpi, koska jokainen muunnos vaatii lisäkäsittelyä. Ryhmittelyssä on tärkeää tietää, että objektit yhdistetään vasta renderöinnin aikana ja niitä pidetään sisäisesti erillään. Tämä mahdollistaa objektien erillisen karsinnan renderöinnissä, jos ne eivät ole näkymässä, mutta sillä on myös haittapuolia. Staattinen ryhmittely kuormittaa muistia ja tallennustilaa ja dynaaminen ryhmittely aiheuttaa lisä kuormitusta prosessorille. (Unity.)

Instanssi (Instancing) on tekniikka, jonka avulla näytönohjain voi renderöidä useita esiintymisiä samasta 3D-mallin verkosta yhdellä piirtokutsulla. Sen sijaan, että näytönohjain

renderöisi jokaisen esiintymän erikseen, se käyttää uudelleen samaa geometriaa, mutta käyttää tarvittaessa eri muunnoksia, kuten sijaintia, kiertoa tai mittakaavaa. (Epic Games; Unity.) Jokainen instanssi jakaa myös samaa materiaalia.

Verkkojen yhdistäminen (Merging) voi yhdistää useiden ainutlaatuisten tai identtisten 3D-mallien verkkoja yhdeksi suuremmaksi verkoksi, mikä vähentää yksittäisten piirtokutsujen määrää. Toisin kuin ryhmittelyssä, objektit yhdistetään ennen renderöintiä. Tämä tarkoittaa sitä, että yhdistetty verkko renderöidään aina samaan aikaan eikä yksittäisiä objekteja voida enää karsia renderöinnissä, mikä johtaa piilotetun geometrian tarpeettomaan renderöintiin. Myös yksittäisten objektien muokkaaminen muuttuu haastavaksi tai mahdottomaksi. Tämän takia verkkojen yhdistämistä tehdään yleensä manuaalisesti ja vasta sitten, kun näkymän iterointi on lopetettu. Verkkojen yhdistämisessä on tärkeää ottaa huomioon, että lopullisen piirtokutsujen määrä vastaa yhdistetyssä verkossa käytettyjen materiaalien määrää, koska materiaaleja ei yhdistetä. Tietyt pelimoottorit, kuten Unreal Engine 5, mahdollistavat myös materiaalien yhdistämisen. Tätä on kuitenkin käytettävä huolellisesti, koska se ei säilytä UV-karttoja. (Epic Games.)

Kun esineellä on useita materiaaleja, pelimoottorin on vaihdettava niiden välillä renderöinnin aikana. Materiaalin vaihtaminen lisää laskennallista kuormaa, koska näytönohjaimen on muutettava varjostusohjelmia, tekstuureja ja renderöintitiloja jokaiselle materiaalille. Jokainen vaihto voi tuoda lisää piirtokutsuja. Tämän optimoimiseksi käytäntönä on pyrkiä käyttämään yhtä materiaalia yhdessä objektissa. Jos tietystä materiaalista halutaan käyttää hieman erilaista versiota, siitä voidaan luoda materiaali instanssi. Materiaali-instanssit mahdollistavat tiettyjen ominaisuuksien muuttamisen luomatta kokonaan uutta materiaalia. Materiaali-instanssit perivät alkuperäisen materiaalin ydinominaisuudet, mutta mahdollistavat tiettyjen parametrien säädettävyyden, kuten värin, karheuden, metallisuuden ja emissioisuuden. Materiaali-instanssit voidaan käsitellä yhdellä piirtokutsulla, kun niitä käytetään oikein. (Epic Games.)

3.4 Toimialan käytännöt peliympäristöjen kehittämisessä

Peliympäristöjen kehittäminen vaatii tasapainoa taiteellisen näkemyksen ja teknisen tehokkuuden välillä. Käytetyt tekniikat ja työnkulut vaihtelevat paljon riippuen kehittäjistä ja projektista. Yksi laajimmin käytetyistä tekniikoista peliympäristöjen kehittämisessä on modulaaristen resurssien luominen. Eri pelien ympäristötaidetta tarkasti tutkiessa voidaan löytää paljon toistuvuutta, joka jää helposti huomaamatta. Tämä toistuvuus johtuu uudelleen käytetyistä ja modulaarisista objekteista.

Modulaariset 3D-mallit ovat objekteja, voiden avulla voidaan koota ja rakentaa ympäristöjä eri tavoilla ilman, että jokaiseen paikkaan tarvitaan ainutlaatuisia objekteja. Videopelit, varsinkin 3D-pelit, ovat pitkään käyttäneet modulaarista arkkitehtuuria ympäristötaiteen ja tasosuunnittelun optimointitekniikkana. Oikein toteutettuna modulaarisiin sarjoihin organisoidut objektit ja niistä kasatut ympäristöt voivat olla hyvin kustannustehokkaita ja joustavia. (Statham ym. 2022.)

Modulaarisen työnkulun päämääränä on käyttää mahdollisimman vähän eri komponentteja samalla kun säilytetään ympäristön monimuotoisuus ja toiminnallisuus. Yleinen ongelma syntyy, kun suunnitellaan modulaarisia elementtejä, joiden täytyy sopia useisiin käyttötapauksiin. Toinen haaste on visuaalinen toisto. Koska modulaarisia resursseja käytetään uudelleen usein, ympäristöt voivat nopeasti näyttää toistuvilta, jos niitä ei ole suunniteltu huolellisesti. Toistuvuutta voidaan rikkoa erilaisilla tavoilla, kuten tekstuurien variaatiolla, dekaaleilla, valaistuksella ja rekvisiitalla. (Kronenberger 2020; Statham ym. 2022.)

Yhteensopivista modulaarisista resursseista koostuvia kokonaisuuksia kutsutaan modulaarisiksi sarjoiksi. Modulaarisia sarjoja jaetaan yleensä kahteen tyyppiin: orgaanisiin ja kovapintaisiin. Orgaaniset modulaariset sarjat koostuvat orgaanisista ympäristö malleista, kuten kivistä ja kasveista, kun taas kovapintaiset modulaariset sarjat koostuvat kovapintaisista ympäristömalleista, kuten rakennuksista ja rakennelmista. (Arellano 2024.) Kuvassa 10 on modulaarinen sarja sekä kovapintaisia että orgaanisia malleja pelille God of War: Ragnarök.

Kuva 10. Modulaarinen sarja (Arellano 2023)

Toinen laajasti käytössä oleva tekniikka peliympäristöjen luomisessa on proseduraalinen generointi. Proseduraalinen generointi sisältää algoritmien käyttämisen pelisisällön, kuten maastojen, tasojen ja rakenteiden automaattisen luomisen. Tämä lähestymistapa vähentää manuaalisen suunnittelun tarvetta huomattavasti ja mahdollistaa monimutkaisten ympäristöjen luomisen tehokkaasti. Esimerkiksi pelit, kuten No Man's Sky käyttävät proseduraalista generointia kokonaisten universumien luomiseen erilaisilla planeetoilla ja ekosysteemeillä. (Game Ace 2024.)

Proseduraalisen generointi tuo paljon etuja, mutta se tuo mukanaan myös tiettyjä haasteita. Proseduuritekniikoiden liiallinen käyttö voi johtaa ympäristöihin, jotka ovat laajoja, mutta tuntuvat toistuvilta, koska niistä puuttuu käsintehty kosketus, joka lisää ainutlaatuisuutta peliympäristöihin. Proseduraalinen generointi on myös teknisesti hyvin haastavaa ja monimutkaista. Algoritmien kehittäminen ja hienosäätö edellyttävät erikoisosaamista ja lisäävät kehitysprosessin monimutkaisuutta, joka voi johtaa viivästyksiin. (Game Ace 2024.)

4 Projektin toteutus

4.1 Yleiskatsaus ja tavoitteet

Projekti keskittyy kovapintaisen modulaarisen 3D-mallisarjan luomiseen kauhupelille Thalassophobia. Modulaarinen sarja tehdään realistiselle hylätylle tutkimusasemalle, joka ilmenee vedenalaisessa ympäristössä. Ympäristö, johon tutkimusasema sisältyy, koostuu monista elementeistä, jotka yleensä ilmenevät maan päällä. Maan päällä ilmenevien elementtien omituinen esiintyminen veden alla on osa kyseisen ympäristön tarinankerrontaa, joten tutkimusasema suunnitellaan näitä edellytyksiä ajatellen.

Tavoitteena oli suunnitella ja kehittää joukko optimoituja, uudelleenkäytettäviä 3D-malleja, jotka mahdollistavat tehokkaan tasosuunnittelun säilyttäen samalla korkean visuaalisen laadun. Mallit suunniteltiin käytettäväksi Unreal Engine 5 -pelimoottorissa, mikä varmistaa yhteensopivuuden nykyaikaisten pelien kehitystyönkulkujen ja suorituskykystandardien kanssa.

Projektin ensisijaisena tavoitteena oli esittää ja dokumentoida tehokas 3D-mallinnuksen työnkulku kovapintaisten peliympäristöjen luomiseen. Tämä sisältää tekniikat, kuten modulaarisen työnkulun, tekstuurien optimoinnin ja suorituskykyä säästävien resurssien luomisen. Valmiit mallit testataan Unreal Engine 5:ssä niiden tehokkuuden arvioimiseksi todellisessa peliympäristössä. Lisäksi projekti pyrkii tasapainoittamaan teknisen tehokkuuden ja taiteellisen joustavuuden varmistaen, että modulaarinen sarja mahdollistaa erilaisia kokoonpanoja. Projekti pyrkii myös tarjoamaan näkemyksiä parhaista käytännöistä ja työnkuluista kovapintaisten videopeliympäristöjen mallintamisessa.

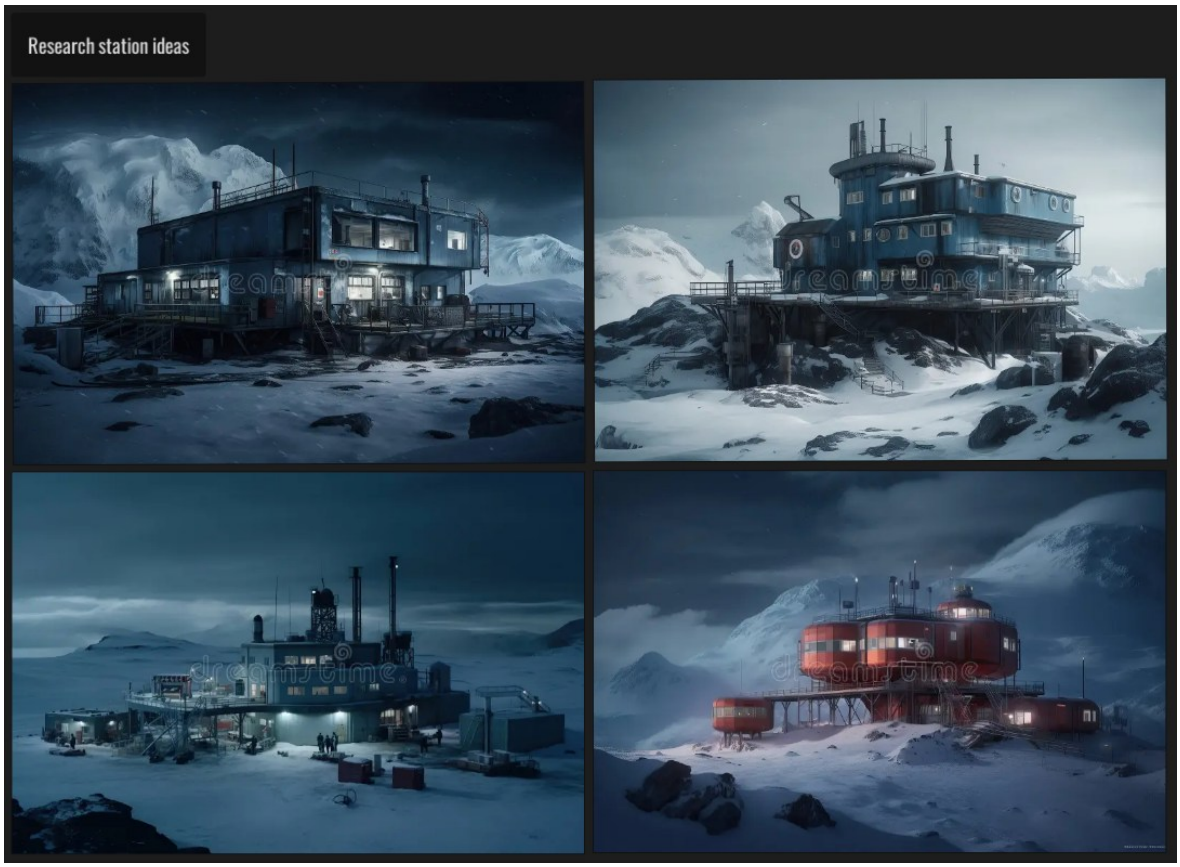
4.2 Mallien suunnittelu ja modulaarinen erittely

Tutkimusaseman toteutukseen päätettiin käyttää modulaarista työnkulkua sen tehokkuuden, joustavuuden ja optimointietujen vuoksi peliympäristöjen luomisessa. Modulaariset resurssit mahdollistavat monimutkaisten ympäristöjen nopean rakentamisen käyttämällä rajoitettua määrää esivalmistettuja komponentteja. Modulaaristen sarjojen mallintaminen on aikaa vievää, mutta koko tuotannon kannalta se säästää paljon aikaa uudelleenkäytettävyyden ja iteroinnin ansiosta, joten alussa käytetty aika on kannattavaa. Tätä lähestymistapaa käytetään laajasti pelien kehityksessä.

Modulaaristen mallien kehitys vaatii huolellista suunnittelua ja tiettyjen periaatteiden noudattamista saumattoman integroinnin ja tehokkaan käytön varmistamiseksi peliympäristössä. Toisin kuin ainutlaatuiset mallit, jotka on suunniteltu itsenäisiksi objekteiksi, modulaariset mallit on rakennettava uudelleenkäytettäviksi, johdonmukaisiksi ja tekniset

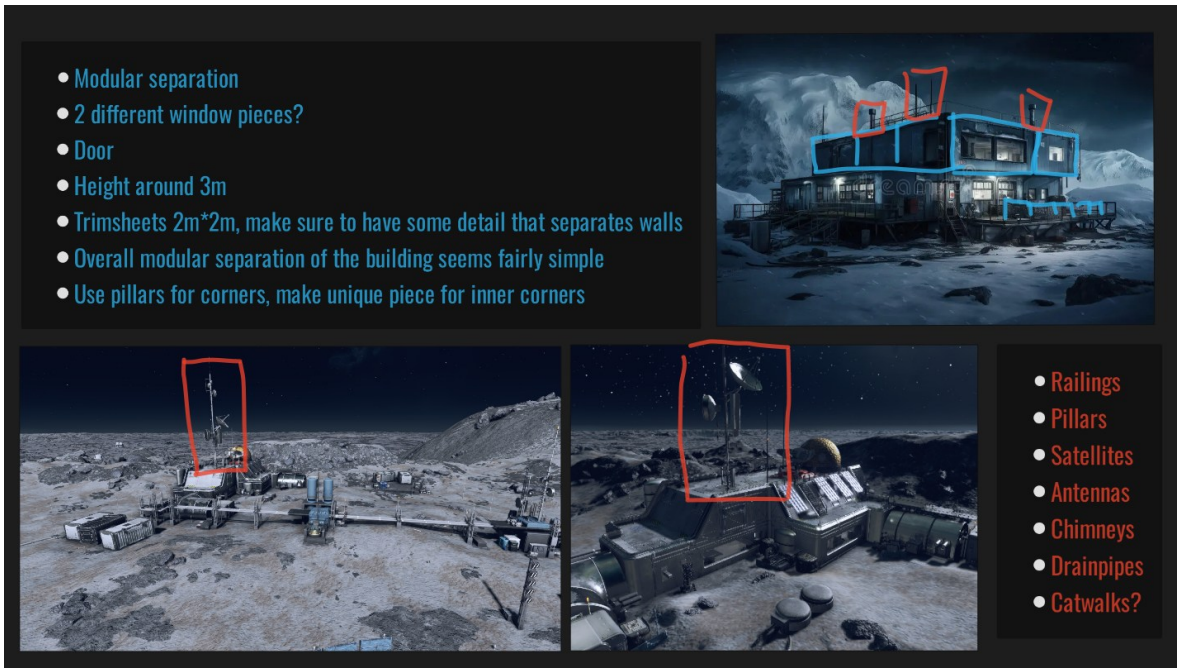
rajoitukset huomioon ottaen. Modulaarisen sarjan ei haluta koostuvan liian monesta kappaleesta tai materiaalista, koska tämä lisää piirtokutsuja ja heikentää suorituskykyä. Sarjan täytyy olla myös intuitiivinen ja skaalautuva, mikä mahdollistaa dynaamisen tasosuunnittelun. Modulaaristen sarjojen kehittämisessä täytyy ottaa huomioon, että työnkulun eri vaiheita toistetaan useaan kertaan, koska se koostuu monista erilaisista osista. Työnkulussa hypitään mahdollisesti monta kertaa suunnittelun, mallinnuksen, teksturoinnin ja testauksen välillä.

Suunnittelu aloitettiin kokoamalla erilaisia referenssikuvia (Kuva 11). Referenssikuvien kerääminen on välttämätöntä visuaalisen tarkkuuden ja yhtenäisyyden ylläpitämiseksi. Referenssit voivat tulla todellisesta arkkitehtuurista, konseptitaiteesta tai muista olemassa olevista peleistä. Yhdenmukainen taidetyyli oli säilytettävä kaikissa malleissa, jotta vältettiin silmäänpistäviä visuaalisia eroja ympäristöjä koottaessa. Koska modulaarisia kappaleita käytetään toistuvasti, kaikki visuaaliset erot korostuvat.



Kuva 11. Tunnelmataulu tutkimusasemalle (mukailtu Vitanovski a-d)

Tutkimusaseman referenssikuvista tehtiin lisäsuunnitelmia siitä, miten se voitaisiin jakaa erillisiin osiin. Kuvista voidaan jakaa osia esimerkiksi yleisiin seiniin, ikkuna seiniin, seinäkulmiin, pilareihin, lattiaan, kattoon, portaisiin ja rakennuksen perustaan. Kuvista voidaan saada myös rekvisiitta ideoita erilaisille tikkaille, antenneille, viemäriputkille, sekä satelliitti- ja ilmanvaihtojärjestelmille. Kuvat antavat myös hyvän yleisnäkemyksen osien teksturoinnille. Kuvassa 12 referenssejä on jaoteltu tarkemmin erilaisiin osiin ja yksityiskohtiin, jotka ilmenevät usein tutkimusasemissa.



Kuva 12. Tunnelmataulun lisäsuunnitelma (mukailtu Vitanovski a; Bethesda Softworks 2023)

Yksi modulaarisen suunnittelun tärkeimmistä näkökohdista on varmistaa, että kaikki osat kohdistetaan oikein ennalta määritetyssä ruudukkojärjestelmässä. Standardoidun yksikön koon käyttäminen estää aukot tai kohdistusvirheet koottaessa ympäristöä pelimoottorissa. Tämä lähestymistapa varmistaa, että seinät, lattiat ja muut rakenneosat napsahtavat yhteen saumattomasti.

Ennen mallien mittojen suunnittelua on hyvä tietää, mitä teksturointitekniikoita tullaan käyttämään. Suurin osa tutkimusaseman osista tullaan teksturoimaan trimmausarkeilla ja toistettavilla tekstuureilla, koska ne tarjoavat hyvän tasapainoin ulkonäön monipuolisuuden ja optimoinnin kannalta. Trimmausarkkeja voidaan käyttää uudelleen monissa eri rakennuksen osissa lisäämään yksityiskohtia. Trimmausarkit tehtiin 2 metriä leveille ja 2 metriä

korkeille pinnoille, koska tutkimusaseman monet modulaariset osat, kuten seinät ja lattiat, noudattavat metripohjaisia mittoja. Tämä varmisti saumattoman tekstuurikartoituksen tiettyjen mallien välillä ja mahdollistaa optimaalisen tekselitiheyden, mikä tarkoittaa, että tekstuurit näyttävät teräviltä ja yhtenäisiltä eri osissa. Tekselitiheys lasketaan pikselien lukumääränä metriä kohti. Trimmausarkkien tekstuuriresoluutioksi määritettiin 1024 * 1024 pikseliä, joka antoi hyvän tasapainon suorituskyvyn, muistinkäytön ja ulkonäön välillä. Tämä tarkoittaa, että tekselitiheys tuli olemaan 5.12 px/cm. Yksinkertaisia toistettavia tekstuureja käytettiin kohdissa, joissa yksityiskohtia ei tarvittu. Toistettavien tekstuurien resoluution on tärkeä olla sama, kuin trimmausarkkien, jotta tekselitiheys pysyy samana.

Koska trimmausarkit ja toistettavat tekstuurit toistuvat aina 2 metrin välein, osista, joiden kuuluu olla saumattomia, tehtiin vähintään 2 metriä leveitä tai korkeita. Jos seinien halutaan olla saumattomia pystysuunnassa, niiden täytyy olla joko 2 tai 4 metriä korkeita. 2 metriä olisi liian matala ja 4 metriä olisi liian korkea, joten seinistä tehtiin 3 metriä korkeita. Tämä tarkoittaa, että trimmausarkit ja toistettavat tekstuurit eivät tulleet olemaan saumattomia seinien välillä pystysuunnassa, joten seiniä eroteltiin jalka- ja kattolistoilla trimmausarkkien avulla. Seinistä olisi saatu saumattomia pystysuunnassa, jos trimmausarkkien ja toistettavien tekstuurien mittoja olisi muutettu, mutta tekstuurit tulisivat silloin toistumaan 1 metrin välein, joka olisi tehnyt tekstuurien ulkonäöstä liian toistuvan.

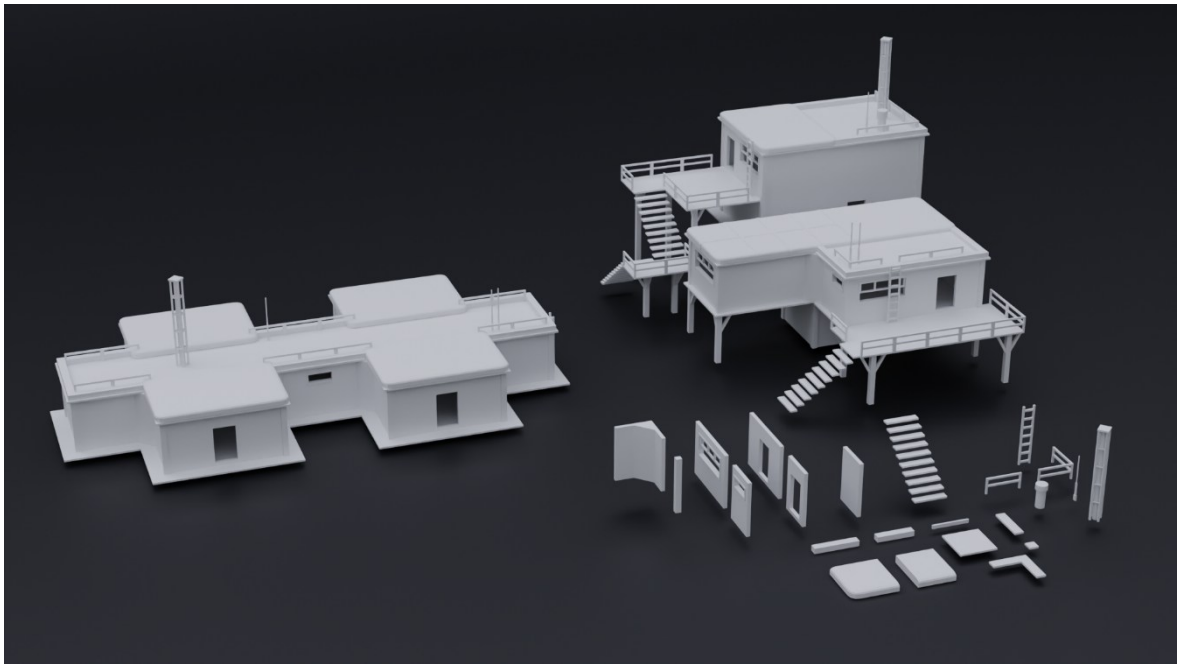
Jotta rakennuksen ulkonäkö pysyy monipuolisena, sille tehtiin useita trimmausarkkeja ja toistuvia tekstuureja. Esimerkiksi rakennuksen sisäseinille ei haluttu samoja tekstuureja, kuten ulkoseinille. Seinille haluttiin myös lisätä yksityiskohtia, jotta ne eivät näytä liian tylsiltä. Sisä- ja ulkoseinät koostuivat kuitenkin yhdestä osasta, jotta erillisten osien määrä pidetään mahdollisimman pienenä. Yksinkertaisin tapa käyttää useita trimmausarkkeja ja toistuvia tekstuureja yhdessä osassa on käyttää useita materiaaleja, mutta tämä tulisi lisäämään piirtokutsuja pelimoottorissa ja huonontamaan suorituskykyä. Jotta usean materiaalin käyttö voidaan välttää, trimmausarkkeja ja toistuvia tekstuureja yhdistettiin tekstuurikokoelmiksi. Yhteen tekstuurikokoelmaan yhdistettiin 2 trimmausarkkia ja 2 toistettavaa tekstuuria, jotta ne täyttivät koko UV-alueen samoin, kuten yksittäiset elementit. Tekstuurikokoelman tekstuuriresoluutiosta tuli siten 2048 * 2048 pikseliä, jotta tekselitiheys pysyi samana.

4.3 3D-mallinnusprosessin aloitus

Suunnitteluvaihe jatkui osittain myös mallintaessa. Mallintaminen aloitettiin tekemällä luonnosmalli. Luonnosmalli, joka tunnetaan myös nimellä greybox tai whitebox, on yksinkertaistettu versio tasosta tai resurssista, joka on tehty primitiivisillä muodoilla perusasettelun, mittakaavan ja koostumuksen määrittämiseksi ennen yksityiskohtaista mallinnusta. Tarkoituksena oli visualisoida ja testata tilasuhteita ja pelikulkua mahdollisimman aikaisessa

vaiheessa, jotta mahdollisia virheitä tai esteitä voitiin välttää myöhemmissä vaiheissa. Modulaaristen resurssien kanssa työskennellessä luonnosmalli on vielä tärkeämpää, koska modulaariset työnkulut perustuvat saumattomasti yhteen sopivien uudelleenkäytettävien osien suunnitteluun. Luonnosmallin avulla suunniteltiin, kuinka monta erilaista kappaletta tarvittiin halutun ulkonäön saavuttamiseen.

Luonnosmallin tekemisessä on tärkeää pitää se mahdollisimman yksinkertaisena ajan säästämisen vuoksi. Se aloitetaan mahdollisimman pienestä osa määrästä ympäristön rakentamiseen, koska modulaarista sarjaa voidaan aina laajentaa myöhemmin. Mitä enemmän sarjaa käytetään, sitä enemmän siihen halutaan lisätä osia. Luonnosmallintaminen voidaan tehdä joko valitussa mallinnusohjelmassa tai usein myös itse pelimoottorissa. Pelimoottorissa luonnosmallin luominen mahdollistaa sen jatkuvan testauksen pelitilanteessa, mutta mallinnustyökalut ovat usein huomattavasti edistyneempiä ja nopeampia käyttää. Kuvassa 13 on tutkimusaseman luonnosmalli tehtynä Blenderissä. Luonnosmallia testattiin myös pelimoottorissa.



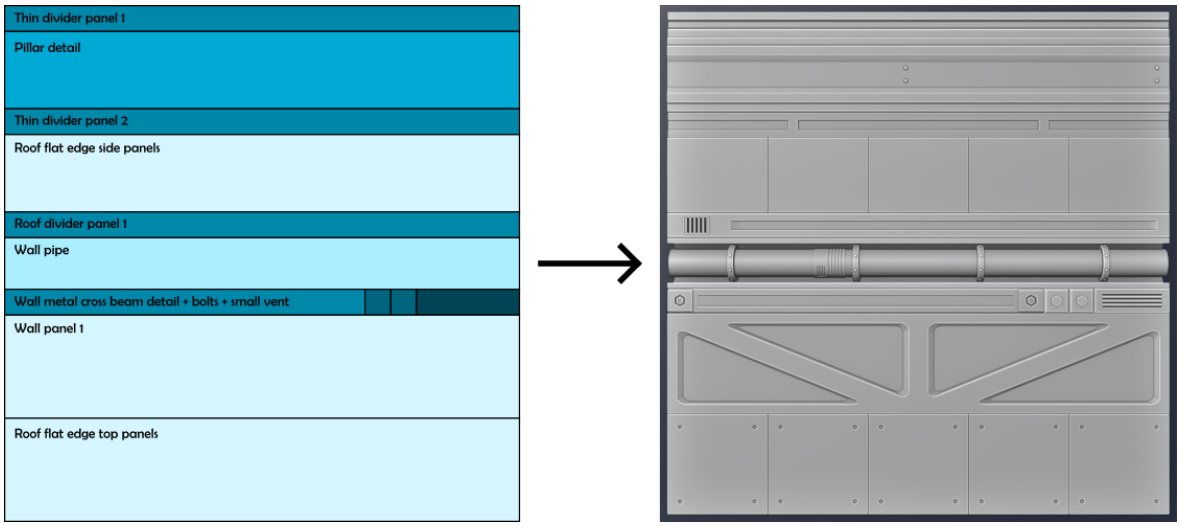
Kuva 13. Luonnosmalli

4.4 Trimmausarkit ja toistettavat tekstuurit

Modulaarisessa työnkulussa tekstuurien ja trimmausarkkien luominen tehdään tyypillisesti luonnosmallinnuksen jälkeen, toisin kuin perinteisessä työnkulussa, jossa

luonnosmallinnusta jatketaan yksityiskohtaisten mallien luomisella. Tämä johtuu siitä, että modulaarisissa työnkuluissa tekstuurit ovat tärkeitä muotoiltaessa sitä, kuinka uudelleenkäytettävät kappaleet laatoitetaan ja miltä ne tulevat näyttämään koko ympäristössä. Koska samoja tekstuureja tullaan käyttämään uudelleen useissa moduuleissa, niiden määrittäminen ajoissa auttaa ohjaamaan kaikkien myöhemmin mallinnettavien osien mittasuhteita, mittakaavaa ja visuaalista koheesiota.

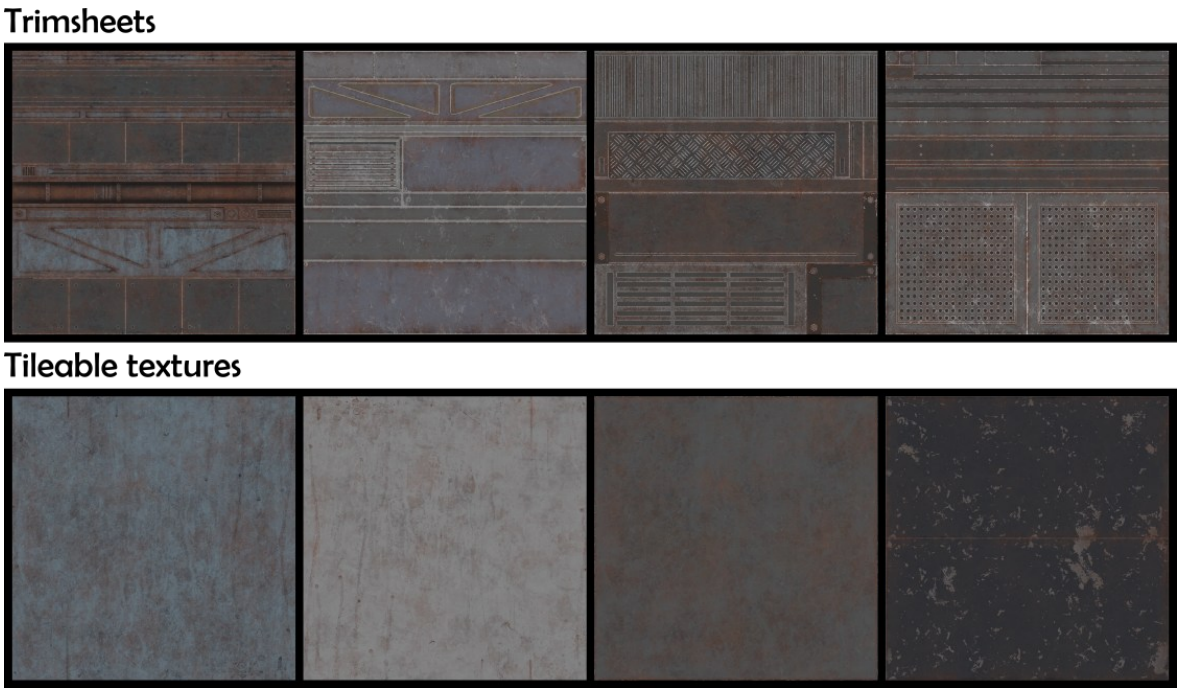
Prosessi alkoi analysoimalla luonnosmallia ja referenssikuvia, jotta saatiin selville, millaisia yksityiskohtia ympäristöön tarvitaan. Tutkimusasemissa yleisiä toistuvia yksityiskohtia ovat esimerkiksi paneelit, pultit, putket, viisteet ja reunalistat. Kaikki yksityiskohdat mallinnettiin 2 m * 2 m kokoiselle levylle. Trimmausarkkien työnkulkua voidaan nopeuttaa ja helpottaa suunnittelemalla sen asettelua vektorigrafiikkaohjelmistolla, kuten Inkscapella. Kuvassa 14 on suunnitelma ensimmäisestä trimmausarkista ja sen perusteella mallinnetuista yksityiskohdista.



Kuva 14. Trimmausarkin suunnittelu

Kun trimmausarkit oli mallinnettu, ne teksturoitiin Adobe Substance 3D Painterissä. Substance 3D Painterissä trimmausarkkien high-poly-yksityiskohdat siirretään yksinkertaiselle low-poly pinnalle generoimalla tekstuurikarttoja Bake-työkalulla. Näiden tekstuurikarttojen avulla voidaan simuloida erilaisia pintojen ominaisuuksia, kuten kaarevuutta, normaalidataa ja ympäristön okkluusiota. Tekstuurien ulkonäkö suunniteltiin vastaamaan ympäristön visuaalista tyyliä. Tekstuureissa piti tulla ilmi ruosteisuutta ja kulumista, mutta jos niille annettiin liikaa ruostetta, tekstuureista tulisi liian yksitoikkoisia ympäristön kokonaisuudessa.

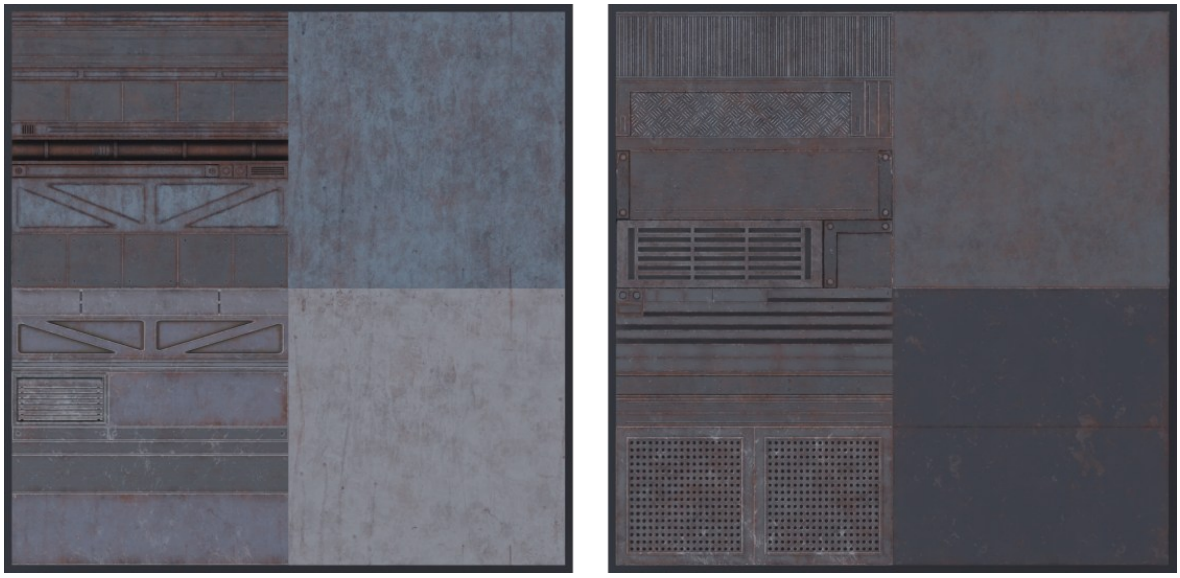
Trimmausarkkeja tehtiin 4 ja niiden kanssa käytettiin yksinkertaisempia toistuvia tekstuureja, jotka tehtiin myös Substance 3D Painterissä. Kun yksi tekstuuri oli valmis, se vietiin ulos kolmena 1024 * 1024 pikselin kokoisena diffuusio-, ORM- ja normaalikarttana. Kuvassa 15 on trimmausarkkien ja toistuvien tekstuurien valmiit diffuusiokartat.



Kuva 15. Trimmausarkit ja toistettavat tekstuurit

Trimmausarkkeja ja toistuvia tekstuureja suunniteltiin käytettäväksi tekstuurikokoelmassa, jotta yksittäisten kappaleiden materiaalimäärä pysyy mahdollisimman pienenä. Tekstuurikokoelmia voidaan tehdä tietyillä kuvankäsittelyohjelmistoilla, kuten Photoshopilla. Diffuusio-, ORM- ja normaalikartat tuotiin kuvankäsittelyohjelmaan, jonka avulla eri tekstuurien samat kartat yhdistettiin 2048 * 2048 pikselin kokoiseksi kuvaksi. Tämä suoritettiin jokaiselle karttatyypille erikseen. Muistin käytön optimoimiseksi ja suorituskyvyn parantamiseksi yhdistettyjen ORM-tekstuurikarttojen kokoa muutettiin 1024 * 1024 pikseliin. Tämä muutos tehtiin sen jälkeen, kun testit osoittivat, että ORM-karttojen resoluution alentamisen visuaalinen vaikutus oli minimaalinen. Karheuskartat säätelevät, kuinka sileältä tai karkealta pinta näyttää vuorovaikutuksessa valon kanssa, mutta ne eivät useimmissa tapauksissa tarvitse erittäin tarkkoja yksityiskohtia tai pikselitarkkuutta. Tämä auttaa suorituskykyä, koska tekstuurien pakkausalgoritmit toimivat paremmin kuvissa, joissa on vähemmän kohinaa ja yksityiskohtia. Vähemmän yksityiskohtainen karheuskartta tiivistyy tehokkaammin, mikä

vähentää tekstuurin muistitilaa ja näytönohjaimen VRAMin käyttöä pelimoottorissa. Kuvassa 16 on valmiitten rakennusosien tekstuurikokoelmien diffuusiokartat.



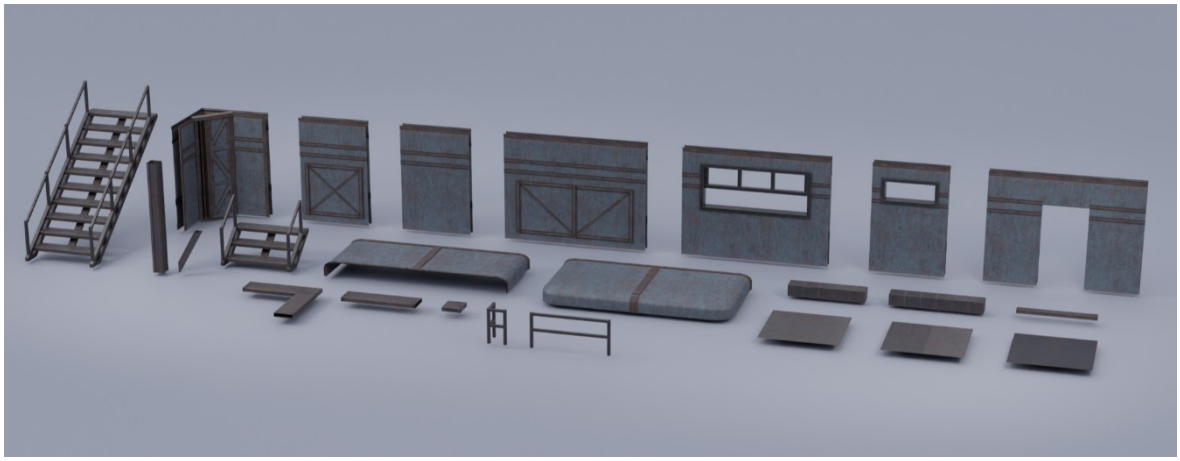
Kuva 16. Valmiit tekstuurikokoelmat

Trimmausarkkien ja toistettavien tekstuurien yhdistäminen tekstuurikokoelmiksi tuo suorituskyvyllisiä etuja pelimoottorissa, mutta sillä on myös haittapuolia. Yksi yleinen ongelma on reunojen vuotaminen eli edge bleeding, jota ilmenee, kun kokoelmien vierekkäiset UV-saaret ovat liian lähellä toisiaan. Tämä aiheuttaa tekstuuritietojen vuotamista tekstuurisuodatuksen aikana. Toinen ongelma on UV-kartoituksessa. Yksittäisillä trimmausarkeilla ja toistettavilla tekstuureilla on mahdollista ylittää UV-rajat, koska ne ovat saumattomia. Tekstuurikokoelmissa UV-karttojen on pysyttävä niille osoitetulla alueella, mikä rajoittaa tätä saumatonta joustavuutta. Lähes yhtä hyvään saumattomaan lopputulokseen voidaan kuitenkin päästä tarkalla UV-saarten sijoittelulla.

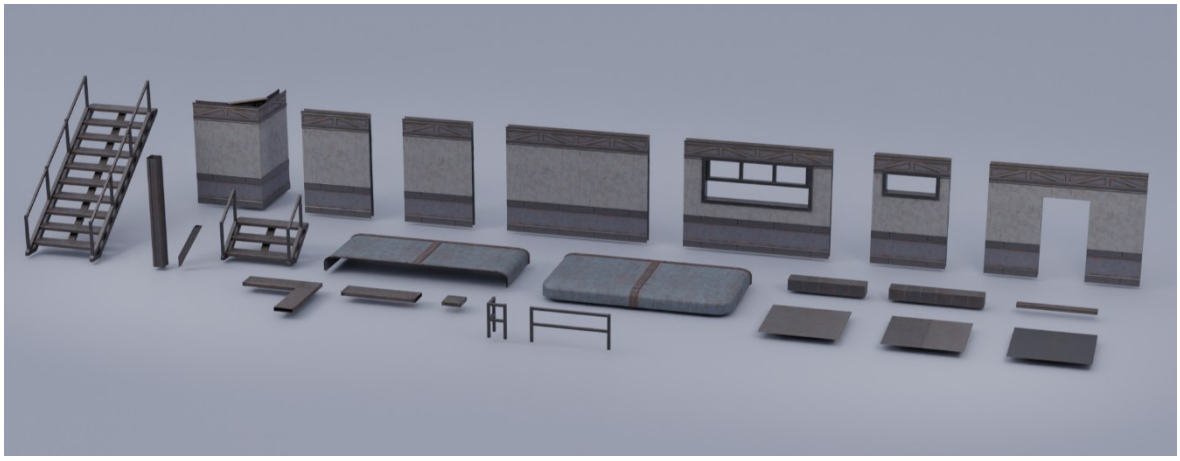
4.5 Modulaaristen osien viimeistely ja rekvisiitta

Kun tekstuurikokoelmat olivat valmiit, jatkettiin modulaarista mallinnusvaihetta. Tässä vaiheessa painopiste siirtyi lopullisten modulaaristen low-poly osien luomiseen aikaisempien luonnosmallien pohjalta. Lopulliset mallit, kuten seinät, lattiat ja katot mallinnettiin ottamalla tarkasti huomioon niiden mittakaavat, yhteensopivuus ja kuinka ne sopivat pelimoottorissa käytettävään ruudukkojärjestelmään. Osat mallinnettiin trimmausarkkien yksityiskohtien avulla ja ne lisättiin malleihin UV-kartoituksen avulla. Koska trimmausarkit luotiin aiemmin,

mallit voitiin suunnitella suoraan niiden perusteella ja mittakaavoilla. Valmiit modulaariset mallit ovat kuvissa 17 ja 18.



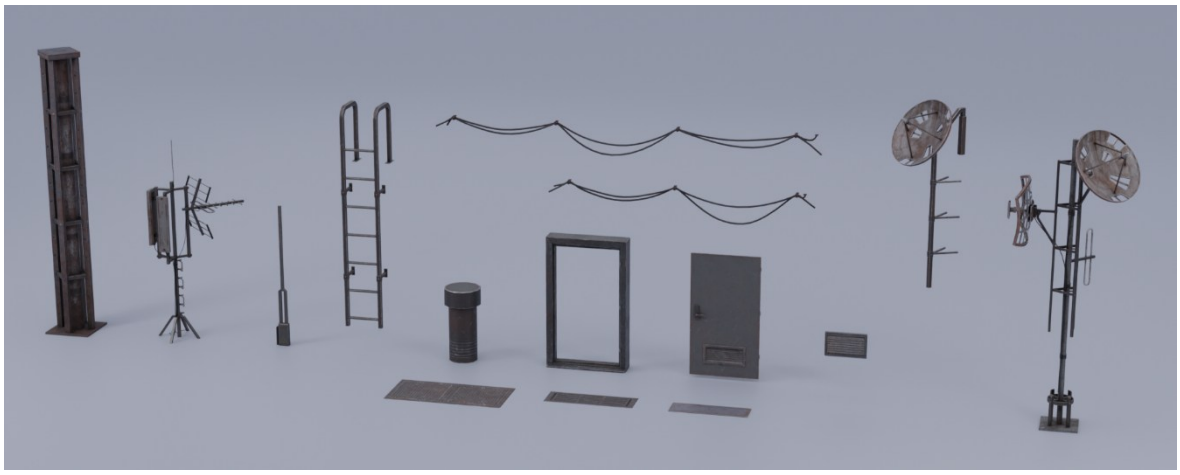
Kuva 17. Modulaariset mallit



Kuva 18. Modulaaristen mallien sisäseinät

Modulaaristen resurssien viimeistelyn lisäksi luotiin erilaista rekvisiittaa ja tukirakenteita. Rekvisiittoja, kuten ovia, tikkaita, kaapeleita, antenneja ja muita pienempiä arkkitehtonisia yksityiskohtia, mallinnettiin täydentämään modulaarisia osia. Osa rekvisiittamalleista voitiin tehdä helposti trimmausarkkien avulla. Lopuille monimutkaisimmille rekvisiittamalleille tehtiin sekä low-poly- että high-poly-mallit, joiden avulla niistä saatiin yksityiskohtaisempia ja ainutlaatuisempia.

Rekvisiittamalleille, jotka eivät käyttäneet trimmausarkkeja, tehtiin uniikit tekstuurikokoelmat, jotka kohdistuvat suoraan näiden tiettyjen mallien pintaan. Tiettyjen mallien UV-karttoja yhdisteltiin, jotta tekstuurikartoista saatiin mahdollisimman optimoituja ulkonäön, suorituskyvyn ja muistinhallinnan kannalta. Tekstuurikarttojen resoluutioksi päätettiin 2048 * 2048 pikseliä, jotta rekvisiittamallien tekselitiheys olisi mahdollisimman lähellä modulaaristen osien tekselitiheyttä. Blenderissä valmiitten UV-karttojen tekselitiheyttä voidaan tarkistaa ilmaisella Texel Density Checker -lisäosalla. Rekvisiittamallien ORM-karttojen resoluutiota pienennettiin 1024 * 1024 pikseliin optimointia varten samoin, kuin modulaaristen mallien. Myös rekvisiittamallien teksturointi tehtiin Substance 3D Painterissä. High-poly-yksityiskohtia siirrettiin low-poly-mallien pinnoille Bake-työkalun avulla, ja niiden ulkonäkö suunniteltiin samojen periaatteiden mukaisesti kuin modulaaristen osien. Kuvassa 19 on valmiit rekvisiittamallit ja kuvassa 20 on lopullisen mallisarjan avulla tehty rakennus Blenderissä.



Kuva 19. Rekvisiittamallit

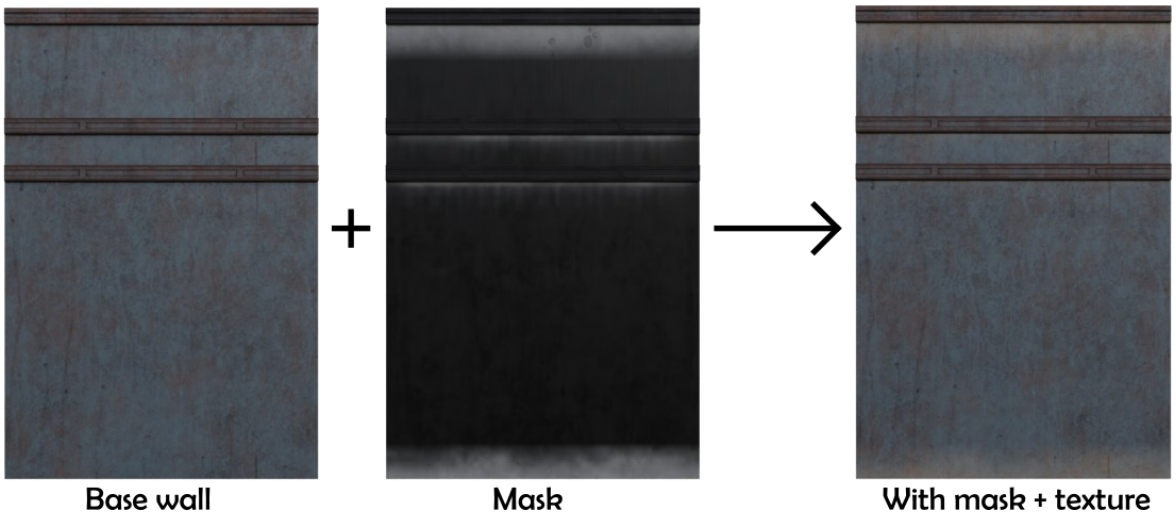
Kuva 20. Mallisarjan avulla tehty rakennus

Mallien viimeistelyn lisäksi luotiin mukautettu kanavapakattu tekstuurimaski Substance 3D Painterissä, jonka avulla voitiin lisätä tietyille modulaarisille osille lian ja hiekan kertymisen yksityiskohtia. Malleille, jotka tulevat käyttämään maskia, tehdään toinen UV-kartta Blenderissä. Yhdellä mallilla voi olla monta eri UV-karttaa ja niitä voidaan käyttää eri tarkoituksiin Unreal Enginessä. Kanavien pakkaus yhdistettynä useiden UV-karttojen käyttöön mahdollisti yksityiskohtaiset likatehosteet, joilla oli minimaalinen vaikutus tekstuurimuistiin, varjostimen monimutkaisuuteen ja materiaalin suorituskykyyn.

Maskille käytettiin yhtä hyvin kevyttä 8-bittistä 512 * 512 suurta RGB-tekstuuria, jossa jokainen värikanava edusti eri maskia. Punainen kanava sisälsi pohjan likamaskin, joka korosti seinien ja rakenteiden pohjan lähellä olevia alueita, joihin hiekka luonnollisesti kerääntyy. Vihreää kanavaa käytettiin määrittämään likaantuminen seinien yläosissa ja sinistä kanavaa käytettiin käsinmaalatuille yksityiskohtamaskeille, mikä mahdollisti paikallisen lian sijoittumisen rakoihin, nurkkiin ja muille halutuille alueille. Maskin vaikutus ja miltä se näyttää käytännössä ilman tekstuurien diffuusiokarttoja näkyy kuvassa 21.

Unreal Enginessä materiaalin asennuksen aikana näihin maskeihin viitattiin erikseen kanavajaon kautta, mikä mahdollisti tarkan hallinnan lian jakautumiseen ja sekoittamiseen. Lika- ja hiekkatehosteet saatiin aikaan yksinkertaisella toistuvalla hiekkatekstuurin diffuusiokartalla yhdistettynä pakattuun maskidataan. Jotta kertymä ei näyttäisi liian yhtenäiseltä tai aiheuttaisi selviä suoria artefakteja, käytettiin myös harmaasävyistä grunge-karttaa

hajottamaan maskin sekoittumista, mikä lisäsi luonnollista satunnaisuutta lopulliseen materiaaliin.

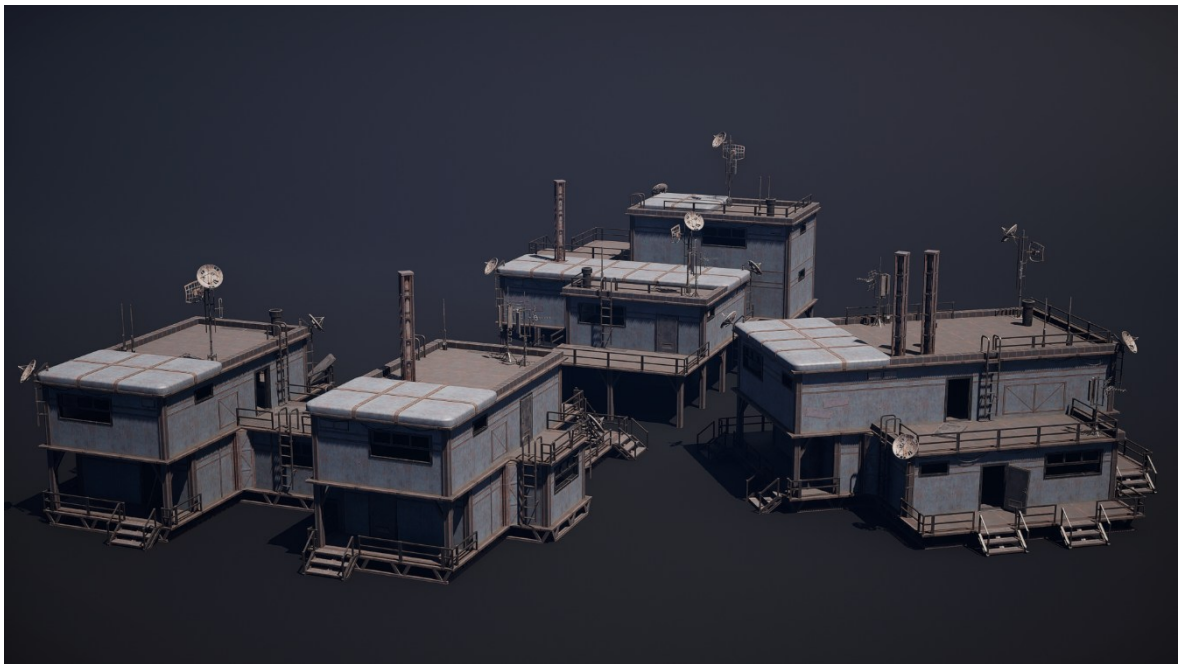


Kuva 21. Yksityiskohtamaski seinässä

4.6 Mallien tuonti, optimointitekniikat ja suorituskyvyn testaus

Kun lopulliset mallit ja tekstuurit oli saatu valmiiksi, kaikki resurssit tuotiin Unreal Engine 5:een integrointia ja suorituskykytestausta varten. Tuontiprosessin aikana käytettiin FBX-tiedostoja. Osa modulaarisista resursseista sisälsi sekä ensisijaisen UV-kanavan teksturointia varten että toissijaisen kanavan maskeja varten. Mallien tuonnissa Unreal Engine luo myös lisäksi yhden uuden UV-kanavan valokartoitusta varten automaattisesti.

Optimointi oli olennainen osa työnkulkua sen varmistamiseksi, että ympäristö pysyi suorituskykyisenä ja visuaalisesti yksityiskohtaisena. Kokonaisia ympäristöalueita rakennettiin modulaarisen rakennussarjan avulla testausta varten, ja suorituskykyä parannettiin hyödyntämällä Unreal Enginen Batch-työkalua, joka järjestää identtiset malliverkot Hierarchical Instanced Static Mesh -komponenteiksi. Tämä tekniikka vähentää piirtokutsujen määrää ja parantaa renderöintitehokkuutta muuntamalla useat identtiset staattiset malliverkot instansseiksi taustalla. Vaikka rakennetussa testiympäristössä oli lähes 1000 mallia, piirtokutsut pysyivät tyypillisesti 80-100 välillä pelkkiä rakennuksia sisältävissä näkymissä instanssien ja materiaalien jakamisen ansiosta. Kuvassa 22 näkyy yksi rakennettu testiympäristö Unreal Enginessä, joka koostuu 3 erilaisesta rakennuksesta.



Kuva 22. Testirakennuksia Unreal Enginessä

Unreal Enginen editorin sisäisiä työkaluja käytettiin laajasti suorituskyvyn analysointiin ja parantamiseen. Shader Complexity -näkymätila auttoi tunnistamaan, jos materiaalit olivat liian monimutkaisia tai kalliita näytönohjaimen prosessorille ja Lightmap Density -näkymätila auttoi tunnistamaan eri mallien automaattisesti luotujen valokarttojen resoluution ympäristössä. Jotkin Unreal Enginen automaattisesti luodut valokartat olivat oletusarvoisesti liian korkeita, joten niitä laskettiin manuaalisesti muistin säästämiseksi ja tarpeettoman ylimääräisen kuormituksen välttämiseksi.

Visuaalisten virheenkorjaustyökalujen lisäksi käytettiin konsolikomentoja kuten ”stat scenerendering” ja ”stat gpu”, jotta saatiin yksityiskohtaisia tietoja siitä, miten resurssit vaikuttivat renderöinnin suorituskykyyn. Nämä työkalut tarjosivat reaaliaikaisia tilastoja piirtokutsuista, materiaalien monimutkaisuudesta, kolmioiden määrästä ja valaistuskustannuksista, mikä helpotti pullonkaulojen tunnistamista ja korjaavien toimenpiteiden tekemistä.

Suorituskyvyn tehostamiseksi edelleen luotiin materiaali-instansseja päämateriaalien perusteella. Tämä mahdollisti dynaamisen materiaalihallinnan parametreillä, kuten karheuden tai maskien intensiteetin pitäen piirtokutsut minimissä. Kanavapakattujen ORM-tekstuurien ja maskien käyttö edisti myös optimointia vähentämällä tekstuurinäytteiden määrää materiaalia kohden.

5 Yhteenveto ja pohdinta

5.1 Tulosten tarkastelu ja kohdatut haasteet

Opinnäytetytön tavoitteena oli suunnitella ja toteuttaa modulaarinen mallisarja hylätylle tutkimusasemalle videopeliympäristöön ottaen huomioon visuaalisen laadun ja optimoidun suorituskyvyn. Lopputuloksena syntyi selkeästi jäsennelty modulaarinen mallisarja, jonka teksturointi perustuu trimmausarkkeihin ja toistuviin tekstuureihin, ja jota täydentävät erilaiset rekvisiittamallit, kuten satelliitit ja antennit. Hyödyntämällä menetelmiä, kuten materiaali-instansseja, Batch-työkalua sekä tekstuurioptimointia, modulaarinen ympäristö saavutti sekä teknisen tehokkuuden että visuaalisen yhtenäisyyden. Suorituskykytestien perusteella piirtokutsujen määrä ja materiaalien monimutkaisuus pysyivät tavoitetasoilla myös laajoissa rakennetuissa testiympäristöissä. Lopputulos vastaa asetettuja tavoitteita ja tuotetut resurssit soveltuvat suoraan pelinkehityksen tuotantoputkeen.

Projektin aikana kohdattuja merkittävimpiä haasteita olivat suunnittelu, visuaalisen yksityiskohtaisuuden ja suorituskyvyn yhteensovittaminen, liian toistuvuuden estäminen ja tekstuurien reunavuodot tekstuurikokoelmissa. Kaikki nämä vaiheet vaativat paljon iteratiivista testausta ja aikaa. Suurin osa haasteista syntyi materiaalimäärän pitämisestä mahdollisimman alhaisena. Lopullinen päätös oli tehdä trimmausarkeista ja toistuvista tekstuureista tekstuurikokoelmia, mutta tämä johti myös tarkempien UV-karttojen tekemiseen ja täydellisen saumattomuuden rikkomiseen reunavuodon estämiseksi.

Toistuvuuden rikkominen modulaarisessa lähestymistavassa oli myös haastavaa suunnitteluvaiheessa, koska tutkimusaseman haluttiin pysyä mahdollisimman realistisena. Realististen mallien suunnittelu on huomattavasti vähemmän luovaa, koska yksityiskohtien ei haluta olevan liian futuristisia tai silmäänpistäviä. Realistiset tutkimusasemat eivät ole arkkitehtonisesti hyvin monimutkaisia, joten toistuvuuden rikkominen on vaikeampaa.

5.2 Hyödynnettävyys ja jatkokehittämisideat

Modulaarinen mallisarja ja siihen liittyvät muut resurssit osoittautuivat hyvin käyttökelpoisiksi Unreal Engine 5 -ympäristössä. Koska mallit rakennettiin alusta alkaen ruudukon kohdistus ja modulaarinen logiikka mielessä pitäen, niiden uudelleenkäytettävyys tasosuunnittelijoille tai muille kehittäjille on korkea. Mallisarjasta voidaan rakentaa monia erilaisia rakennuksia ja tarkan optimoinnin ansiosta voidaan myös halutessa luoda hyvin laajoja ympäristöjä.

Jatkokehityksen näkökulmasta modulaarinen mallisarja tarjoaa vahvan pohjan mahdollisille laajennuksille. Yksi jatkokehitysalue olisi lisätä arkkitehtonisia variaatioita tai joidenkin

moduulien enemmän vaurioituneita versioita ympäristön tarinankerronnan parantamiseksi ja toiston vähentämiseksi. Toinen jatkokehittämisidea erilaisten vaurioyksityiskohtien lisäämiseksi ja toiston vähentämiseksi olisi tehdä Decal-sarjoja, joiden avulla pinnoille voidaan simuloida lisää ruosteisuutta, likaa tai muita vaurioita. Eri ympäristöjen visuaalisen identiteetin laajentamiseksi voitaisiin ottaa käyttöön lisää rekvisiittamalleja tai mallintaa sarja erilaisia orgaanisia ympäristömalleja, kuten hiekkakasoja ja kasvistoa, jotka ovat kasvaneet hylätyn rakennuksen pinnoille. Tärkein jatkokehittämisidea on kuitenkin sisätilojen sisustuskalusteiden rekvisiittamallit, jotta rakennukset saadaan mielenkiintoisiksi myös sisältä. Tekniseltä puolelta mallisarja voisi hyötyä täysin manuaalisesti toteutetusta yksikohtaisuustason (LOD) järjestelmästä, mikä parantaisi suorituskykyä edelleen varsinkin suuremmissa kohtauksissa ja heikomman tason järjestelmissä.

Lähteet

A23D. 2023. Decals – All you need to know. Blogi. Viitattu 15.3.2025. Saatavissa https://www.a23d.co/blog/decals-all-you-need-to-know

Agladze, M. 2024. Game Engine Basics: Rendering Techniques. E-kirja. IntelliSoft Ventures. Amazon.

Arellano, J. 2023. Modulaarinen sarja. Viitattu 12.4.2025. Saatavissa https://www.artstation.com/artwork/r9gBnO

Arellano, J. 2024. Modular Kits for Game Environments w/ Jon Arellano. Beyond Extent. YouTube-video. Viitattu 29.3.2025. Saatavissa https://www.youtube.com/watch?v=77xPHfzciiY&t=4754s

Awati, R. 2024. 3D mesh. TechTarget. Viitattu 9.3.2025. Saatavissa https://www.techtarget.com/whatis/definition/3D-mesh

Belec, A. 2022. Blender 3D Incredible Models. E-kirja. Packt. Packtpub.

Bethesda Softworks. 2023. Tunnelmataulun lisäsuunnitelma. Viitattu 5.4.2025. Saatavissa https://starfieldwiki.net/wiki/File:SF-location-Nova_Galactic_Research_Station.png

Blender. 2025. Blender 4.3 Manual. Viitattu 16.3.2025. Saatavissa https://docs.blender.org/manual/en/latest/render/materials/introduction.html

Burns, S. 2023. Trimsheets. Beyond Extent. Viitattu 15.3.2025. Saatavissa https://www.beyondextent.com/deep-dives/trimsheets

Caulfield, B. 2018. What's the Difference Between Ray Tracing and Rasterization? Nvidia-blogi 19.3.2018. Viitattu 23.3.2025. Saatavissa https://blogs.nvidia.com/blog/whats-difference-between-ray-tracing-rasterization/

Chadwick, E. 2025. ORM textures and how they work in 3 simple steps. RapidPipeline. Viitattu 12.4.2025. Saatavissa https://rapidpipeline.com/en/a/orm-map-how-it-works/

Epic Games. Unreal Engine 5.5 Documentation. Viitattu 22.3.2025. Saatavissa https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-5-documentation

Furneri, F. 2022. 3D Modeling Part 6: Low-Poly Models and Texture Baking. Shutterstock. Viitattu 15.3.2025. Saatavissa https://www.shutterstock.com/blog/low-poly-models-and-texture-baking

Game Ace. 2024. Procedural Generation in Games. Blogi. Viitattu 29.3.2025. Saatavissa https://game-ace.com/blog/procedural-generation-in-games/

GarageFarm. Understanding Topology in 3D Modeling. Blogi. Viitattu 9.3.2025. Saatavissa https://garagefarm.net/blog/understanding-topology-in-3d-modeling

Gorman, K & Vakulich, A. 2021. What is 3D Modeling and Design? A Complete Guide to 3D. Chudovo. Viitattu 9.3.2025. Saatavissa https://chudovo.com/what-is-3d-modeling-and-design-a-complete-guide-to-3d/

Katatikarn, J. 2024. 14 Most Popular 3D modeling Software Tools in 2024. Academy of animated art. Viitattu 9.3.2025. Saatavissa https://academyofanimatedart.com/3d-modeling-software/

Koneteo Stories. 2024. 3D Modeling for Games: Unveiling the Best Software Tools. Medium. Viitattu 9.3.2025. Saatavissa https://medium.com/@koneteo.stories/3d-modeling-for-games-unveiling-the-best-software-tools-648a70c1bd23

Kronenberger, L. 2020. Balancing Modularity and Uniqueness in Environment Art. Beyond Extent. Viitattu 29.3.2025. Saatavissa https://www.beyondextent.com/articles/balancing-modularity-and-uniqueness-in-environment-art

Lark. 2024. Texture Atlas. Viitattu 15.3.2025. Saatavissa https://www.larksuite.com/en_us/topics/gaming-glossary/texture-atlas#how-texture-atlas-works-for-gaming-businesses

Magee, R. 2022. Procedural Primer for Artists. Houdini. Vimeo-video. Viitattu 5.4.2025. Saatavissa https://vimeo.com/692391880

Maxon. 2025. Plans and pricing. Viitattu 9.3.2025. Saatavissa https://www.maxon.net/en/buy

Mitra, R., Hile, B. & Khayyat, M. 2025. 24 Best Video Game Engines, Ranked. Game Rant. Viitattu 22.3.2025. Saatavissa https://gamerant.com/best-video-game-engines/

Perforce. 2023. What Are the Best Game Engines? Blogi 24.1.2023. Viitattu 22.3.2025. Saatavissa https://www.perforce.com/blog/vcs/most-popular-game-engines

Pluralsight Content Team. 2022. Understanding UVs: Love or Hate Them, They’re Essential. Viitattu 11.3.2025. Saatavissa https://www.pluralsight.com/resources/blog/software-development/understanding-uvs-love-them-or-hate-them-theyre-essential-to-know

Statham W., Jacob, J., Fridenfalk, M. 2022. Game environment art with modular architecture. Science Direct. Viitattu 29.3.2025. Saatavissa https://www.sciencedirect.com/science/article/pii/S1875952121000732

Unity. Unity 6 User Manual. Viitattu 15.3.2025. Saatavissa https://docs.unity3d.com/Manual/index.html

Vehkala, M. 2023. About game engines. Remedy Entertainment. Viitattu 22.3.2025. Saatavissa https://www.remedygames.com/article/about-game-engines

Vitanovski, J. a. Tunnelmataulu tutkimusasemalle. Viitattu 5.4.2025. Saatavissa https://www.dreamstime.com/science-research-station-antarctica-generative-ai-science-research-station-antarctica-created-generative-ai-image275388805

Vitanovski, J. b. Tunnelmataulu tutkimusasemalle. Viitattu 5.4.2025. Saatavissa https://www.dreamstime.com/science-research-station-antarctica-generative-ai-science-research-station-antarctica-created-generative-ai-image275388787

Vitanovski, J. c. Tunnelmataulu tutkimusasemalle. Viitattu 5.4.2025. Saatavissa https://www.dreamstime.com/science-research-station-antarctica-created-generative-ai-science-research-station-antarctica-generative-ai-image275383642

Vitanovski, J. d. Tunnelmataulu tutkimusasemalle. Viitattu 5.4.2025. Saatavissa https://www.dreamstime.com/science-research-station-antarctica-generative-ai-science-research-station-antarctica-created-generative-ai-image275388769

Wingfox. 2022. What is Hard Surface Modeling? Viitattu 9.3.2025. Saatavissa https://blog.wingfox.com/what-is-hard-surface-modeling/

Yang, R. 2025. The Level Design Book. GitBook. Viitattu 16.3.2025. Saatavissa https://book.leveldesignbook.com/process/env-art/optimization