Janne Pitkäkangas

# 3D-PELIYMPÄRISTÖN TOTEUTTAMINEN

Opinnäytetyö
Kajaanin ammattikorkeakoulu
Luonnontieteiden ala
Tietojenkäsittelyn koulutusohjelma
Syksy 2014

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

**OPINNÄYTETYÖ**
**TIIVISTELMÄ**

| | |
|---|---|
| Koulutusala<br>Luonnontieteet | Koulutusohjelma<br>Tietojenkäsittely |
| Tekijä(t)<br>Janne Pitkäkangas | |
| Työn nimi<br>3D-peliympäristön toteuttaminen | |
| Vaihtoehtoiset ammattiopinnot<br>Peligrafiikka | Ohjaaja(t)<br>Raimo Mustonen |
| | Toimeksiantaja<br>- |
| Aika<br>Syksy 2014 | Sivumäärä ja liitteet<br>55+3 |


Opinnäytetyön tutkimiskohteena on kolmiulotteisen peliympäristön luomisprosessi suunnitteluvaiheesta valmiiksi, optimoiduksi maisemakokonaisuudeksi. Työssä selvitetään lyhyesti peliympäristön merkitystä pelaajan uppoutumisessa peliin sekä yksittäisten 3D-peliobjektien ja ympäristöjen esteettistä suunnittelua, mutta sen pääaihealue on kuitenkin ennemmin kolmiulotteisten peliympäristöjen luomisprosessin teknisellä kuin taiteellisella puolella. Huomiota on kiinnitetty varsinkin optimointitekniikoihin. Peligrafiikan optimointi on tärkeää, koska pelejä suorittavilla laitteistoilla on rajallinen määrä tehoa ja muistia käytettävissään kaiken vaaditun grafiikan piirtämiseen. Huolellisesti optimoitu grafiikka mahdollistaa sulavamman pelikokemuksen laitteiston varatessa vähemmän resursseja sen piirtämiseksi. Toinen etu on, että hyvin optimoitua grafiikkaa voidaan piirtää näytölle kerralla suurempia määriä.

Työssä toteutettiin pieni osa kolmiulotteista pelimaailmaa. Projekti alkoi toimeksiantona osaksi Colossal Order -peliyrityksen kaupallista peliä, mutta projektin toteutuksen aikana pelin kehittäminen jouduttiin keskeyttämään. Tällöin päätettiin tehdä alkuperäistä toimeksiantoa kattavampi kokonaisuus, jossa tutkittiin itsenäisesti tarkemmin Blender-3D-mallinnusohjelman ja Unity-pelimoottorin yhteiskäyttöä peliympäristön rakentamisessa.



| | |
|---|---|
| Kieli | Suomi |
| Asiasanat | 3D, 3D-mallintaminen, Blender, peliympäristö, Unity, videopeli |
| Säilytyspaikka | ☒ Verkkokirjasto Theseus<br>☐ Kajaanin ammattikorkeakoulun kirjasto |

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

**THESIS ABSTRACT**

| School | Degree Programme |
|---|---|
| Natural Sciences | Business Administration |
| Author(s)<br>Janne Pitkäkangas | |
| Title<br>Implementation of a 3D Game Environment | |
| Optional Professional Studies<br>Game graphics | Instructor(s)<br>Raimo Mustonen |
| | Commissioned by<br>- |
| Date<br>Fall 2014 | Total Number of Pages and Appendices<br>55+3 |


The subject of this thesis is the creation process of a three-dimensional game environment. Topics such as the achievement of better immersion through game environments and some of the fundamentals of aesthetical design of graphical video game assets and environments are briefly introduced, but the main focus is more on the technical than the artistic side of the process. Particular emphasis has been placed on the optimization of both individual 3D game objects and entire environments. Proper optimization is important because the hardware has limited resources at its disposal for rendering. Optimized game graphics ensure a better gaming experience through improved frame rates as the hardware does not have to work as hard to render them. Another advantage is that a greater amount of well optimized 3D objects can be visible at once, which enables more impressiveness in game environments.

A small piece of a 3D game world was also created. The project began as a commission for Colossal Order, a game development company, and it was supposed to be a part of a commercial video game. Unfortunately, the game was cancelled, which led to shifts in the aim of the project. It was then decided to independently study the use of Blender, a free 3D modeling package, together with Unity, a game engine, in the creation of a game environment.



| | |
|---|---|
| Language of Thesis | Finnish |
| Keywords | 3D, 3D modeling, Blender, game environment, Unity, video game |
| Deposited at | ☒ Electronic library Theseus<br>☐ Library of Kajaani University of Applied Sciences |

SISÄLLYS

SYMBOLILUETTELO

| | |
|---|---|
| Alfa | Kuvatiedoston värikanava, jota voidaan käyttää mm. tekstuurin pikselikohtaisen läpinäkyvyyden määrittämiseen |
| Atlas | Yhteen kuvatiedostoon pakattu joukko useiden erillisten 3D-objektien tekstuureita. |
| Diffuusio (Diffuse) | Pinnan väri, joka nähdään, kun pintaan osuva valo hajoaa tasaisesti joka suuntaan. |
| Eritteleminen (Batching) | Piirtokutsujen vähentäminen yhdistelemällä yksittäisiä näytönohjaimelle piirrettäväksi lähteviä tietoeriä suuremmiksi kokonaisuuksiksi. |
| Fbx | 3D-objekteja sisältävä tiedostomuoto. |
| Immersio | Pelaajan niin syvä uppoutuminen peliin, että hänen aivonsa käsittelevät pelin tapahtumia samalla tavoin kuin tosielämänkin tapahtumia. |
| Kohoumakartta (Bump map) | Harmaasävykuva, jossa jokaisen pikselin tummuus määrittää sen korkeuden suhteessa pinnan nollatasoon. |
| Kova reuna | Varjostusryhmien välinen saumakohta, jossa kärjet kahdentuvat. |
| Kärki (Vertex) | 3D-mallin pienin rakennusosa. Piste 3D-avaruudessa, jolla voi olla ominaisuuksia kuten sijainti, normaali, väri ja tekstuurikoordinaatteja. |
| LOD | Lyhenne sanoista level of detail. Optimointikeino, jossa objektista tehdään yksinkertaistettuja versioita, jotka otetaan yksi kerrallaan käyttöön kameran etääntyessä objektista. |
| Lähialuevarjostus (Ambient Occlusion) | Tekniikka, jossa 3D-maisemaan luodaan varjostus vertailemalla pikseleiden välisiä etäisyyksiä. |
| MIP | Tekstuurista tehty joukko toinen toistaan pienemmäksi skaalattuja versioita, joista pelimoottori valitsee sopivimman kameran etäisyyden mukaan. |

| | |
|---|---|
| Normaali | 3D-grafiikassa normaali on monikulmion tai kärjen kohtisuora. 3D-mallin pinta varjostetaan kärkien normaaleiden perusteella. |
| Normaalikartta | Tekstuuri, johon on varastoitu pinnan suuntanormaalit. |
| Paistaminen (Baking) | Tietojen, kuten valaistuksen tai normaaleiden laskeminen 3D-geometriasta tai kuvatiedostosta tekstuuriin. |
| Proppi | Peliympäristöä koristavat objektit, jotka toimivat osana lavastusta. |
| Referenssi | Peligrafiikkaa tehtäessä käytettävä lähdemateriaali, jonka avulla varmistutaan siitä, että lopputulos vastaa vaatimuksia. Referenssi voi olla esimerkiksi kuvia tai videoita, ja sen avulla voidaan tavoitella mm. värejä, tyyliä tai yksityiskohtia. |
| Spekulaarisuus (Specularity) | Pinnan kiilto valon heijastuessa siitä. |
| Tekstuuri | Kaksiulotteinen kuva, joka kääritään kolmiulotteisen pinnan päälle, jolloin luodaan vaikutelma todellista yksityiskohtaisemmasta pinnasta. |
| UV-kääriminen (UV-unwrapping) | Objektin tekstuurikoordinaattien määrittäminen. |
| Valokartta (Light map) | Tekstuuri, johon paistetaan ympäristön valot ja varjot. |
| Varjostusryhmä (Smoothing group) | Yhdessä varjostetun monikulmiojoukon muodostama yhteinen pinta. |

# 1 JOHDANTO

Peliympäristö on tärkeä osa pelikokemusta. Yleensä pelimaailma toimii vähintäänkin alustana pelaajan ja pelin eri elementtien välisessä vuorovaikutuksessa, mutta sen avulla voidaan tehdä paljon muutakin. Pelimaailman avulla voidaan esimerkiksi virittää tunnelmaa tai sitä voidaan käyttää tarinankerronnan apuvälineenä.

Opinnäytetyön tavoitteena on tutkia sitä, millainen on hyvä 3D-peliympäristö ja kuinka se syntyy. Olennaisia kysymyksiä ovat seuraavat: Mistä peliin uppoutuminen eli immersio aiheutuu ja mitä asioita pelimaailmalta vaaditaan, jotta pelaajan uppoutumista peliin voitaisiin sen avulla edistää? Kuinka peliympäristöjä voidaan luoda mahdollisimman nopeasti, kuitenkin uhraten mahdollisimman vähän lopputuloksen visuaalista laatua ja uskottavuutta? Mitä keinoja 3D-peligraafikolla on käytettävissään, jotta yksittäiset objektit tai kokonaiset peliympäristöt saataisiin mahdollisimman näyttäviksi kuitenkaan unohtamatta rajallisia laitteistoresursseja? Työssä käydään läpi myös estetiikan perusasioita kuten värit ja muodot, mutta siinä on kuitenkin keskitytty enemmän peliympäristöjen tuottamisen tekniseen kuin taiteelliseen puoleen.

Pelimaailma on tärkeä osa pelin immersiivisyyttä: jos se ei tunnu hyväksyttävältä, koko pelin luoma illuusio toisesta todellisuudesta on vaarassa särkyä. Immersio edellyttää siis pelimaailmalta tietynlaista uskottavuutta, mutta sitä voidaan entisestään vahvistaa muun muassa tunnelman avulla.

Peligrafiikan näyttävyyttä rajoittavat pääasiassa laskentatehon ja muistin määrä. Laskettavaa peliympäristöissä näytönohjaimelle tuottavat muun muassa kärkien 3D-avaruuden koordinaattien muuntaminen näytön koordinaateiksi, niiden väliin muodostuvien pintojen piirtäminen ja valaistus. Muistiin varastoidaan kaikki näkyvissä olevat tekstuurit ja 3D-mallit. Suoritinkaan ei jää toimettomaksi: sen täytyy lähettää tiedot näytönohjaimelle kaikista piirrettävistä pinnoista. Usein yhtä resurssia voidaan tarvittaessa ostaa toisella, esimerkiksi valaistuksesta voidaan tietyissä tapauksissa vapauttaa laskentatehoa muistin kustannuksella niin sanotusti paistamalla tilan valot ja varjot tekstuuriin. On myös pidettävä mielessä, että kaikkia olemassa olevia resursseja voidaan harvoin käyttää pelkkään pelimaailmaan: jo pelkästään grafiikan puolella niistä usein ottavat osansa ainakin hahmot ja käyttöliittymän visuaaliset elementit.

Tuloksia kokeiltiin käytännössä luomalla Unity-pelimoottorissa kolmiulotteinen rakennus sisätiloineen. Projekti alkoi toimeksiantona, jossa siitä oli alun perin tarkoitus tulla osa kaupallisen peliprojektin pelimaailmaa. Tämä lähtökohta asetti tuotetuille 3D-malleille ja tekstuureille tiettyjä laatuvaatimuksia. Niiden piti näyttää riittävän hyviltä silti varaten mahdollisimman vähän resursseja. Peliprojekti kuitenkin peruuntui kesken toimeksiannon, jolloin opinnäytetyö muuttui itseopiskeluksi, jossa pääasiallisesti perehdyttiin Unityn käyttöön pelimaailman luomisessa. Erityisiä huomion kohteita olivat ilmaisen Blender-3D-mallinnusohjelman käyttö yhdessä Unityn kanssa ja Unityn tarjoamat peliympäristön optimointikeinot. Lopuksi kokonaisuudesta tehtiin ohjelma, jossa luotua peliympäristöä voidaan tarkastella ensimmäisessä persoonassa.

# 2 PELIYMPÄRISTÖN SUUNNITTELU

Peliympäristöjen avulla voidaan selittää pelaajalle lukuisia asioita pelin maailmasta ja taustatarinasta. Mikä paikka on kyseessä? Mitä siellä on tapahtunut aiemmin? Keitä tai mitä siellä asuu ja millaisissa oloissa? Mitä paikassa tapahtuu lähitulevaisuudessa? Entä mikä on paikan tarkoitus pelimaailmassa tai sen tunnelma? Pelimaailma kertoo paljon itsestään pelaajan liikkuessa sen läpi. (Worch 2010. 58–59.)

## 2.1 Immersio

Immersiolla tarkoitetaan pelaajan mahdollisimman täydellistä uppoutumista peliin. Tämä tarkoittaa muun muassa sitä, että pelaaja reagoi pelissä tapahtuviin asioihin ennemmin vaistonvaraisesti kuin jatkuvan, aktiivisen analysoinnin kautta. Babak Kaveh kirjoittaa artikkelissaan A Fresh Look at the Concept of Immersion, että immersoituneen pelaajan mieli tulkitsee pelin maailmaa jokseenkin samaan tapaan kuin oikeaakin maailmaa. Esimerkiksi pelaajan kohdatessa pelissä puron hän yhdistää siihen samoja tunteita kuin mitä hän tuntisi ollessaan oikeankin puron varrella. Toisin sanoen pelaaja ei huomaa pelaavansa peliä. Ihmiset ovat suurimman osan valveilla viettämästään ajasta tällaisessa immersoituneessa tilassa. Syy siihen, miksi arkielämän joka hetkeä ei muisteta tarkasti, on se, että aivot hoitavat suuren osan tarvittavista toiminnoista taustalla.

Immersio ja kiinnostus eivät ole sama asia. Pelaaja voi olla kiinnostunut pelistä kykenemättä kuitenkaan uppoutumaan siihen täysin. Kuitenkin Kavehin mukaan kiinnostus peliä kohtaan on edellytys immersiolle. Mikäli pelaaja on kiinnostunut pelistä ja sen teemasta, hän jatkaa pelin pelaamista ja tutkimista ja sitä kautta uppoutuu syvemmälle peliin ja siten altistuu muille immersiota aiheuttaville tekijöille. Tällaisia tekijöitä ovat muun muassa tunneside pelin tarinaan ja hahmoihin, järkeenkäyvät kontrollit sekä tietysti pelin audiovisuaaliset ominaisuudet. (Kaveh 2010.)

Pelimaailma on tärkeä immersiotekijä. Mielenkiintoinen pelimaailma vetää pelaajaa tutkimaan ja siten käyttämään enemmän aikaa pelin pelaamiseen (Kaveh 2010). Pelin ympäristöjen on oltava uskottavia, jopa todentuntuisia, olivatpa ne sitten tyyliteltyjä tai todellista maailmaa jäljitteleviä. Epäkohdat muistuttavat pelaajaa siitä, että hän on pelaamassa peliä. (Gard 2010.)

Immersio yhdessä tunnelman kanssa tekee peliympäristöstä mieleenpainuvan. Kenttäsuunnittelija Sjoerd De Jong mainitsee kirjassaan The Hows and Whys of Level Design, että kokemukset ja pelimaailmat jäävät paremmin mieleen silloin kun pelaaja voi yhdistää ne mielessään johonkin todellisessa elämässä kokemaansa tunteeseen. (Jong 2008, 126.)

## 2.2 Uskottavuus

Immersiivinen pelikokemus edellyttää siis todellisen tuntuista, uskottavaa ja loogista pelimaailmaa. Kuinka sitten voidaan varmistua siitä, että luotu peliympäristö täyttää nämä kriteerit, varsinkin kun jokainen pelaaja odottaa siltä hieman eri asioita? Pelisuunnittelija Toby Gardin mukaan jokaisen ihmisen maailmankuva ja ajattelu muodostuvat yksinkertaistetuista todellisuutta kuvaavista malleista, toisin sanoen joukosta käsityksiä, joiden avulla järjestelemme ympäröivän maailman ominaisuuksia. Tunnistamme ja tulkitsemme kohtaamiamme asioita ja muita kokemuksia näiden käsitysten kautta, ja siten jokaisella meistä on erilainen kuva ympäröivästä maailmankaikkeudesta. Luodessamme esimerkiksi taidetta valitsemme siihen päätyvät asiat näiden samojen mallien perusteella, ja siksi jokaisen luomukset ovat omanlaisiaan.

Vaikka nämä käsitykset auttavat meitä tulkitsemaan asioita nopeammin, niissä on se haittapuoli, että niistä pois jäänyt, taiteilijan käsitysten mukaan turha tieto saattaa olla teoksen tarkastelijalle tärkeää, ja näin hänen mielestään teoksesta jää puuttumaan jotain olennaista. Jos pelaaja ei siis kohtaa pelimaailmassa asioita, jotka hänen todellisuudenkuvansa mukaan tulisi sieltä löytyä, tai näkee siellä jotain, joka ei sinne hänen mielestään kuulu, maailma ei tunnu hänestä autenttiselta. Esimerkkinä tästä Gard mainitsee todellisen maailman Ranskalle ominaiset liikennemerkit virtuaalisessa Chicagossa. Pelin kehittäjän rajallisen todellisuuskuvan perusteella tässä ei välttämättä ole mitään väärää, mutta Chicagoa tunteva pelaaja huomaa heti, että jotain on vialla. Tällaiset epäkohdat voidaan välttää keräämällä tarpeeksi referenssimateriaalia, kuten kuvia, ennen objektin suunnittelemista. Referenssiä ei tarvitse kuitenkaan noudattaa täydellisen tarkasti; toisinaan pelattavuus saattaa asettaa pelimaailmalle vaatimuksia, joita täysin todenmukaisella ympäristöllä ei voitaisi toteuttaa (Rouse III 2010, 13). (Gard 2010, 1.)

Referenssin käyttämisestä on myös sellainen etu, että sen avulla voidaan jo etukäteen suunnitella objektin käytännön toteutusta, ennakoida mahdollisesti ongelmallisia kohtia ja siten jo

hyvissä ajoin pohtia niihin ratkaisuja. Referenssimateriaalia voidaan etsiä esimerkiksi internetistä, kirjoista tai elokuvista. (Giambruno 2003, 1.)

Vaikka uskottava peliympäristö onkin edellytys tunnelmalle ja immersiolle, Sjoerd De Jongin mukaan sen ei tarvitse olla tyyliltään fotorealistinen, kunhan sen teema, tyyli ja tarina muodostavat hyväksyttävän kokonaisuuden. Tärkeintä on, että pelin maailma vaikuttaa loogiselta. Esimerkiksi arkkitehtuurin ja muun geometrian tulisi yleensä näyttää tukevalta, eikä valaistuksessa tulisi olla kaikkia sateenkaaren sävyjä ilman syytä.

Objektien sijoittelulla on suuri vaikutus peliympäristön uskottavuuteen. Esimerkiksi luonnolliset objektit kuten kasvit ja kivet eivät koskaan esiinny täysin satunnaisissa paikoissa todellisessa maailmassa. Jong suosittelee pyrkimään sopivaan suhteeseen satunnaisuutta ja suunnitelmallisuutta, sillä liika suunnitelmallisuus saattaa kuitenkin johtaa keinotekoisen näköiseen lopputulokseen. Kivien ja kasvien tapauksessa kivet ovat todennäköisesti olleet paikoillaan jo kauan ennen kasveja, ja tämän tulisi näkyä kokonaisuutta katsottaessa. Tällöin objektien ei tulisi olla satunnaisesti sijoiteltuina ilman minkäänlaista logiikkaa, vaan esimerkiksi siten, että kasvit kukoistavat kivien tarjoamassa suojassa. (Jong 2008, 77.)

Ympäristön uskottavuutta lisää, jos sillä on selvästi jonkinlainen menneisyys. Tällöin pelaaja ei yhtä helposti tunne olevansa kaiken keskipiste (Jong 2008, 127). Ympäristön taustaa voidaan hahmottaa seuraavien kysymyksien avulla: Mikä paikka on kyseessä? Keitä siellä asuu? Onko siellä joskus tapahtunut jotain tärkeää? Voidaanko näitä asioita valaista pelaajalle visuaalisin keinoin, esimerkiksi pelimaailmaa koristamaan tehtyjen objektien eli proppien avulla? (Rouse 2010, 19.)

Pelimaailmasta saa autenttisemman tuntuisen myös lisäämällä sinne liikkuvia yksityiskohtia. Staattinen maailma tuntuu helposti keinotekoiselta, varsinkin luonnon osalta. Esimerkkejä ympäristön liikkuvista asioista ovat puiden lehdet tuulessa, sade, pilvet ja koneiden liikkuvat osat. Toisaalta joskus voi olla tarkoituksenmukaista tehdä maisemasta täysin liikkumaton. (Jong 2008, 127.)

## 2.3 Tilan vaikutus tunnelmaan

Arkkitehti Christopher W. Tottenin mukaan tilan koko ja sijainti vaikuttavat huomattavasti siihen, millaisia tuntemuksia tai ajatuksia se herättää pelaajassa. Tunnetilat voivat vaihdella esimerkiksi turvallisuudentunteen ja ahdistuksen välillä. Ahdas, suljettu tila, kuten käytävä, luo tunteen turvattomuudesta, sillä jos jokin hyökkäisi siellä pelaajan kimppuun, tälle olisi vain vähän vaihtoehtoja pakenemiseen. Ahtaat käytävät ovatkin tyypillisiä kauhupeleissä. Myös avoimessa tilassa pelaaja voi tuntea olonsa uhatuksi, sillä mahdolliset viholliset havaitsevat tämän jo kaukaa ja pääsevät hyökkäämään esteettä. Totten kertoo, että arkkitehtuurin teorisoija Grant Hildebrand kutsuu tällaista avointa tilaa, jossa on vain vähän piilopaikkoja, prospektitilaksi. Monissa vanhoissa 2D-peleissä, esimerkiksi Mega Man -sarjassa, pomohuoneet ovat tällaisia.

Jos pelaaja pystyy saavuttamaan tilan jokaisen osan, se tuo tälle turvallisuuden tunnetta. Hän tuntee olevansa tilanteen herra. Tätä Totten kutsuu henkilökohtaiseksi tilaksi. Esimerkiksi Batman: Arkham Asylum -pelissä huoneet saattavat olla hyvinkin suuria, mutta pelaajalla on aina etulyöntiasema vihollisiinsa nähden siten, että hän pääsee liikkumaan ympäriinsä näiden huomaamatta.

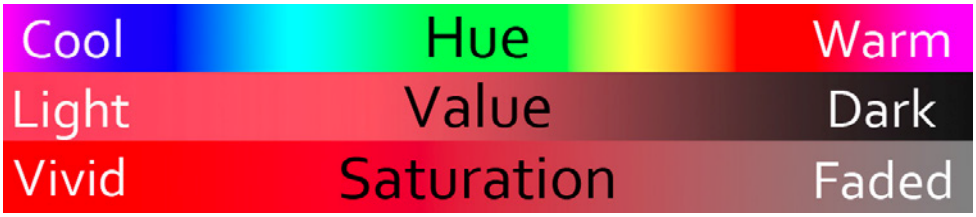
Pelin kulkua voidaan tasapainottaa vuorottelemalla prospektitilan ja sinne sijoiteltujen suojapaikkojen välillä. Suojapaikka on jonkinlainen henkilökohtainen tila, esimerkiksi katos, luola tai jopa varjo, jossa pelaaja voi hengähtää rauhassa. Suojasta voi myös toisinaan löytyä apuvälineitä. (Totten 2011.)

## 2.4 Objektin ja ympäristön suunnittelu

Videopeleissä pyritään luomaan illuusio todellisesta maailmasta samojen perusasioiden avulla kuin perinteisessäkin taiteessa. Esimerkkejä tällaisista perusasioista ovat perspektiivi, muoto ja vaaleus. Tämä illuusio ei kuitenkaan ole ainoa syy siihen, miksi näihin vanhoihin visuaalisen suunnittelun periaatteisiin kannattaa kiinnittää huomiota: niiden avulla voidaan saavuttaa entistä parempaa estetiikkaa ja niitä voidaan jopa käyttää tarinankerronnan apuvälineinä. (Solarski 2013, 1.)

### 2.4.1 Värit

Väri koostuu kolmesta komponentista: sävy, kylläisyys ja vaaleus (kuvio 1). Sävy tarkoittaa sitä, mitä väriä kyseinen väri pohjimmiltaan on, esimerkiksi punainen tai sininen. Violetti, sininen ja vihreä mielletään tunnelmaltaan kylmiksi väreiksi, kun taas keltainen, oranssi ja punainen ovat lämpimiä sävyjä. Kylläisyys on sävyn määrä. Kylläinen väri näyttää eloisalta, ja kylläisyyden vähetessä väri lähenee harmaata. Vaaleus tarkoittaa sitä, miten tumma tai vaalea väri on. 3D-peliassetin tapauksessa vaaleus ei ole välttämättä aivan yhtä tärkeä komponentti kuin sävy ja kylläisyys. Tämä riippuu siitä, paljonko pelimoottori ja varjostimet tekevät objektin varjostamiseksi.




Kuvio 1. Sävy, vaaleus ja kylläisyys. (Kuvion lähde: Mathis 2012, 158.)

Värikokonaisuus on tärkeä osa pelin maailman visuaalista kommunikaatiota. Tämä tarkoittaa sitä, että kauhupeliä varten tehdyn objektin saisi periaatteessa sopimaan pirteäänkin ympäristöön pelkästään värejä muuttamalla. Harmonisen värikokonaisuuden valinnassa voidaan käyttää apuna väriympyrää. Väriympyrässä on kaikki sävyt jaettu tasaisesti sen kehälle siten, että vastavärit ovat vastakkaisilla puolilla ympyrää.

Yleisiä harmonioita ovat yksiväriharmonia, vastaväriharmonia, kolmiväriharmonia, lähiväriharmonia ja aksentillinen lähiväriharmonia. Yksiväriharmoniassa on vain yhtä sävyä, mutta vaihtelevia määriä kylläisyyttä ja vaaleutta. Vastaväriharmonian värit on muodostettu yhdestä sävystä ja sen vastasävystä. Kolmiväriharmoniassa on pääväri sekä kaksi muuta väriä, jotka ovat ympyrän vastakkaisilla puolikkailla yhtä kaukana päävärin vastaväristä. Lähiväriharmoniassa kaikki täydentävät värit ovat päävärin lähellä. Aksentillisessa lähiväriharmoniassa on vielä lisäksi päävärille vastaväri. Peliobjektia teksturoitaessa saattaa olla kannattavaa etsiä käytettävät sävyt ennen yksityiskohtien lisäämistä. Tämä helpottaa värikokonaisuuden hahmottamista ja harmonia säilyy helposti vielä yksityiskohtien lisäämisen jälkeenkin.

Yksivärisen materiaalin pinnassa on aina vähintäänkin pieniä sävyeroja. Nämä erot riippuvat materiaalin omien ominaisuuksien lisäksi ympäröivästä valaistuksesta ja materiaalista mahdollisesti heijastuvista objekteista. Nykyaikaisten pelien hienostuneet grafiikkaominaisuudet mahdollistavat sen, että tällaisia materiaaleille ominaisia heijastuksia ja muita poikkeamia tarvitsee yhä harvemmin lisätä diffuusiotekstuuriin, mutta lisäsävyt yksivärisellä alueella saattavat kuitenkin tehdä lopputuloksesta entistä mielenkiintoisemman (kuvio 2). Sävyeroilla saadaan myös luotua lisäkontrastia valoisien ja varjoisten alueiden välille. (Mathis 2012, 158–171.)


Kuvio 2. Vasemmanpuoleiseen puutekstuuriin on käytetty pelkästään ruskean sävyjä; oikeanpuoleista on elävöitetty pienellä sävyvaihtelulla. (Kuvion lähde: Mathis 2012, 171.)

### 2.4.2 Muoto

Eri muodoilla on monenlaisia psykologisia merkityksiä. Vaikka ihmiset tulkitsevatkin näkemäänsä kaiken aiemmin kokemansa kautta, yleistyksiä voidaan kuitenkin tehdä jossain määrin. Chris Solarski esittää tällaisiksi vakioiksi ympyrän, nelikulmion ja kolmion. Hänen mukaansa ympyrä voidaan yhdistää viattomuuteen, nuoruuteen, energisyyteen ja naisellisuuteen.

Nelikulmio puolestaan antaa vaikutelman aikuisuudesta, tasapainosta, vakaudesta, jääräpäisyydestä, turvallisuudesta ja hienostuneisuudesta. Kolmion Solarski yhdistää aggressiivisuuteen, miehekkyyteen ja voimaan. Hän kuitenkin korostaa, että näitä muotojen ja tuntemusten välisiä suhteita ei pitäisi käyttää ehdottomana sääntönä suunnittelussa, vaan pikemminkin ongelmakohtien etsimisen apuvälineenä. (Solarski 2013, 1.)

Objektia suunniteltaessa kannattaa kiinnittää erityistä huomiota sen siluettiin eli ääriviivoihin. Alex Galuzin listaa hyvälle siluetille seuraavat ominaisuudet: vahvat muodot, mieleenpainuvuus, tunnistettavuus jopa kaukaa, erottuvuus muista objekteista, vetoavuus, omaperäisyys ja ainutkertaisuus. Olennaista on, että pelaaja tunnistaa jo lyhyellä vilkaisulla, mikä objekti on kyseessä. (Galuzin 2010.)

## 2.5 Ympäristön sommittelu

Sommittelu on tärkeä osa visuaalisia taiteita. Yleisesti sillä tarkoitetaan näkyvien elementtien asettelua siten, että kokonaisuus miellyttää silmää ja ohjaa katsetta kohti teoksen olennaisia yksityiskohtia. Ympäristöä suunnitellessa sommittelu toteutetaan muun muassa arkkitehtuurin, värien ja valaistuksen avulla. (Jong 2008, 74.)

Jos alueella ei ole minkäänlaista kontrastia eri elementtien välillä, lopputulos on yksitoikkoinen. Esimerkkinä Jong esittää kokonaisuuden, jossa kaikki alueet ovat keskenään samanvärisiä ja -muotoisia, ja joissa kaikissa on keskenään samanlainen valaistus. Mielenkiintoisuutta voidaan lisätä huomattavasti lisäämällä yksityiskohtia, jotka erottuvat muusta maisemasta selvästi. Kuitenkin jos ympäristöä koetetaan parantaa käyttämällä jatkuvasti yhtä ja samaa yksityiskohtaa, yksityiskohdasta itsestäänkin tulee toistuva ja pitkäveteinen. Lisäksi kontrastin määrän ja sijoittelun kanssa on oltava tarkkana. Jos maisemasta selvästi erottuvia yksityiskohtia käytetään liikaa, pelaajan katse ei enää virtaa sujuvasti näkymän läpi. Tällöin yksityiskohdat kilpailevat pelaajan huomiosta, eikä maisema tunnu tasapainoiselta. Tällainen tilanne saattaa syntyä esimerkiksi silloin, kun huoneen yksityiskohdat tai värit ovat jakautuneet liian epätasaisesti sen eri puolien välillä. Yleensä paras tulos saadaan, kun yksityiskohdat ovat tarpeeksi erottuvia tuodakseen vaihtelua, mutta eivät silti ole huomiota herättäviä. (Jong 2008, 75–76.)

Peliympäristön sommittelun tulisi parantaa pelattavuutta, esimerkiksi auttamalla pelaajaa navigoimaan pelimaailmassa. Värien, valaistuksen, arkkitehtuurin ja yksityiskohtien avulla voidaan ohjata pelaajan katsetta kohti keskeisiä objekteja, kuten tärkeitä hahmoja tai reittejä seuraavalle alueelle.

Sommittelu on helpompaa peleissä, joissa kamerat ovat ennalta määrätyissä paikoissa. Näin tiedetään aina, mitä pelaaja näkee minäkin hetkenä. On kuitenkin paljon myös sellaisia pelejä, joissa näin ei ole. Silti pelaajan pitäisi kokea suunnitellut näkymät mieluummin interaktiivisen pelin aikana kuin lukitun kameran kautta. Tällöin on kätevintä panostaa sommittelussa sellaisiin paikkoihin ja kulmiin, joihin pelaaja todennäköisimmin katsoo. Tällaisia ovat esimerkiksi oviaukot. (Rouse III 2010, 33.)

## 2.6 Valaistus

Valaistus on tärkeä osa peliympäristöä. Sjoerd De Jongin mukaan jopa kaikkein yksinkertaisin kenttä voidaan muuttaa hyvin laadukkaaksi hyvin toteutetulla valaistuksella, tai upeinkin geometria voidaan pilata valaisemalla se huolimattomasti. Valaistuksella voidaan pelaajan ohjaamisen lisäksi luoda tunnelmaa, esimerkiksi saada tila näyttämään lämpimältä tai kylmältä. Valaistus on myös sommittelun apuväline, ja sillä voidaan korostaa kolmiulotteisten objektien muotoja. (Jong 2008, 110.)

Voimakas kontrasti valon ja varjon välillä saattaa aiheuttaa illuusion kahdesta täysin erillisestä tilasta yhden tilan sisällä: varjot oma alueensa ja valo omansa. Aiemmin mainittiin, että varjo saattaa tarjota pelaajalle suojaa. Varjolla voi kuitenkin olla myös päinvastainen merkitys. Varsinkin silloin, kun valoa on vain vähän, siitä poistuminen varjon puolelle on epämiellyttävää. Pimeässä odottava tuntematon aiheuttaa jännitystä. (Totten 2011.)

On tärkeää muistaa peliympäristöä valaistaessa, että valolla on oltava aina lähde. Ympäristön uskottavuuden varmistamiseksi sen, mistä valo on peräisin, täytyisi olla ilmeistä. Käytännössä tämä tarkoittaa valojen sijoittamista mahdollisimman lähelle lähteitään. Valo täytyisi myös suhteuttaa oikein lähteeseensä nähden, esimerkiksi pienitehoisella lampulla kirkkaasti valaistu suuri tila ei ole uskottava. (Jong 2008, 111–112.)

Useimmiten on aiheellista määrittää valolle jokin muu väri kuin oletusarvona toimiva valkoinen. Todellisessa maailmassa harva valonlähde tuottaa täysin valkoista valoa, ja lisäksi valonsäteet kimpoilevat pinnoilta ottaen niistä mukaansa väriä. Tätä kimpoilua ei vielä nykyaikaisissakaan peleissä voida kunnolla mallintaa varsinkaan reaaliajassa, mutta sitä voidaan usein riittävän hyvin matkia lisäämällä valon väriin ympäristön pääsävyjä. Sommittelustakin saadaan entistä mielenkiintoisempi antamalla valoille värejä, mieluiten useampaa kuin yhtä sävyä samassa ympäristössä. Käytettyjen värien täytyisi täydentää toisiaan kuitenkin luoden samalla kontrastia. Jong suosittelee pääsävyiksi yhtä kylmää, esimerkiksi sininen tai turkoosi, ja yhtä lämmintä väriä, kuten keltainen tai oranssi. Keltainen ja oranssi tuovat tarpeeksi kontrastia viileille väreille herättämättä kuitenkaan liikaa huomiota, toisin kuin aggressiivinen punainen. Jos kontrastia näyttää syntyvän liikaa, sitä voidaan hillitä vähentämällä valojen värien kylläisyyttä. Käytännön esimerkki tällaisesta valon kautta ympäristöön tuodusta kylmän ja lämpimän värin kontrastista on auringonvalo. Aurinko tuottaa lämpimänsävyistä, kellertävää tai oranssia valoa, ja aurinkoisena päivänä varjot ovat viileän sinertäviä. (Jong 2008, 114–120.)

# 3 3D-PELIYMPÄRISTÖGRAFIIKAN TOTEUTTAMINEN

Edellisessä luvussa käytiin läpi peliobjektin visuaalinen suunnittelu, eli kuinka objekti saadaan näyttämään esteettisesti hyvältä. 3D-grafiikassa ja peligrafiikassa yleensäkin on kuitenkin otettava huomioon myös toinen, tekninen puoli. Pelaamiseen käytettävien koneiden suorituskyky ei ole loputonta: laskentateho ja muisti ovat rajallisia resursseja, ja optimaalisen suorituskyvyn saavuttamiseksi peligrafiikkaa tehtäessä täytyy varmistaa, että näitä rajoja ei ylitetä. Ymmärrys siitä, mistä osista 3D-grafiikka koostuu ja minkälaisia resursseja se vaatii laitteistolta, auttaa luomaan hyvännäköistä, mutta samalla mahdollisimman hyvin optimoitua peligrafiikkaa.

## 3.1 3D-geometria

Kolmiulotteinen objekti voidaan aina määritellä käyttäen suorakulmaisen eli karteesisen koordinaattijärjestelmän kolmea akselia. Näistä akseleista käytetään kirjaimia X, Y ja Z. Voidaan ajatella, että X edustaa leveyttä, Y korkeutta ja Z syvyyttä. Näiden kolmen akselin avulla voidaan määritellä esimerkiksi objektin sijainti kolmiulotteisessa avaruudessa suhteessa avaruuden keskipisteeseen tai niiden osien koordinaatit, joista objekti muodostuu. Tällaista digitaalista, jotakin asiaa esittävää objektia kutsutaan 3D-malliksi. (Slick 2013.)

3D-mallin pienin yksittäinen osa on kärki eli verteksi. Se on yksittäinen piste, sijainti 3D-avaruudessa. Näiden kärkipisteiden avulla voidaan muodostaa monikulmioita eli polygoneja. Tällainen monikulmio on litteä taso, jossa on yleensä 3-4 kärkeä, mutta joissakin ohjelmissa myös yli nelikulmaiset monikulmiot ovat mahdollisia. Asettelemalla useita monikulmioita kiinni toisiinsa niistä voidaan muodostaa monenlaisia 3D-malleja. Umpinainen 3D-malli saattaa antaa vaikutelman kiinteästä objektista, mutta todellisuudessa se on ontto sisältä. (Pardew & Whittington 2005, 12; Chadwick 2012.)

### 3.1.1 3D-mallinnusprosessi

3D-mallintamiseen on useita lähestymistapoja. 3D-mallit voidaan rakentaa monikulmioista (polygonal modeling), käyrien (spline) tai kehikoiden (patch) avulla, tai parametrisesti. Monikulmioista rakentaminen on vanhin tapa, ja se on edelleen suosittu varsinkin pelialalla, sillä matalayksityiskohtaisten mallien luominen ja muokkaus on yleensä siten kätevintä. Monikulmiomallinnuksessa vaikutetaan suoraan mallin muotoon liikuttelemalla sen kärkiä ja monikulmioita. Näin ollen se, kuinka paljon yksityiskohtia objektiin voidaan mallintaa, riippuu siitä, kuinka pienistä monikulmioista sen pinta koostuu.

Käyrä on eräänlainen jana, jonka pituutta ja mahdollista mutkallisuutta hallitaan kontrollipisteiden avulla. Käyrien etu monikulmiomallinnukseen verrattuna on se, että niillä tehdyt mallit ovat resoluutioriippumattomia, toisin sanoen niiden pehmeät muodot säilyvät jopa läheltä katsottuna toisin kuin monikulmioista koostuvassa mallissa, jonka pehmeistäkin muodoista saattaa paljastua näkyviä kulmia lähitarkastelussa. Käyrät ovatkin kätevä keino luoda orgaanisia, pehmeitä muotoja. Käyrien avulla luodun objektin lopullista monikulmiomäärää voidaan muuttaa vapaasti milloin tahansa mallinnusprosessin aikana.

Kehikko toimii hieman samoin kuin käyrä, mutta siinä yksittäisten janojen sijaan objektin pintaa muokataan kontrollipisteistä koostuvan verkon avulla. Kukin näistä pisteistä vaikuttaa omaan alueeseensa objektin pinnassa vetäen sitä itseään kohti hieman magneetin tavoin, ja niitä liikuttelemalla voidaan venyttää objektin pintaa suuntaan tai toiseen. Samoin kuin käyrät myös kehikko soveltuu hyvin orgaanisten muotojen mallintamiseen.

Parametrisessä mallinnuksessa objektia muokataan säätelemällä sen ominaisuuksia kuten alkumuotoa, kokoa ja resoluutiota. Ohjelma muistaa jokaisen tehdyn muutoksen ja jokaista näistä ominaisuuksista voidaan muokata uudestaan missä tahansa mallinnusprosessin vaiheessa niin kauan kuin objekti pidetään parametrisenä. Tavallisesti parametrinen malli täytyy muuttaa monikulmioiksi viimeistään sitä vietäessä peliin. (Giambruno 2003, 1.)

Tyypillisesti mallintaminen aloitetaan joko kaksiulotteisesta muodosta tai mallinnusohjelmaan sisäänrakennetusta perusobjektista eli primitiivistä. Kaksiulotteisesta objektista aloittamisen vahvuuksia ovat muodon nopea muokkaaminen ja mittakaavan helppo kokeileminen. Kun nämä seikat on saatu kuntoon, voidaan objekti tai osia siitä pullistaa kolmiulotteiseksi työntämällä sen monikulmioita ulospäin siten, että ulostyöntyneen alueen sivuille syntyy uu-

det monikulmiot (extrude), toisin sanoen objektille annetaan syvyys Z-akselilla. Esimerkiksi neliö muuttuu ektruusiossa kuutioksi. Toinen yleinen keino muuttaa kaksiulotteisia muotoja kolmiulotteisiksi on sorvaaminen. Tämä tarkoittaa sitä, että muodosta tehdään pieniä ekstruusioita samalla, kun sitä käännetään määrätyn akselin ympäri. Sorvaaminen on kätevä tapa tehdä sylinterimäisiä, koristeellisia objekteja, kuten tuolinjalkoja.

Primitiivejä ovat esimerkiksi taso, kuutio, pyramidi, pallo, kartio ja torus. Primitiivien vahvuus kaksiulotteisista, käsin tehdyistä muodoista aloittamiseen verrattuna on siinä, että niiden avulla monimutkaistenkin kolmiulotteisten muotojen tekeminen on nopeaa. Niitä voidaan muotoilla uudelleen eri muunnosten (transform) avulla. Muunnokset ovat toimintoja, joilla voidaan muokata objektin tai jonkin sen osan sijaintia, suuntaa tai kokoa, toisin sanoen siirtäminen, pyörittäminen ja skaalaus. Yleensä 3D-mallinnusohjelmissa muunnokset suoritetaan gizmon avulla. Gizmo on eräänlainen apuväline, joka esittää graafisesti objektin sijaintia, kulmaa ja kokoa, samalla tarjoten eräänlaiset kahvat, joita vetämällä näitä ominaisuuksia pystytään muokkaamaan. Muunnokset voidaan rajoittaa tietylle tai tietyille akseleille, mikä estää muutosten tekemisen vääriin suuntiin.

Muunnoksissa voidaan käyttää useita erilaisia koordinaattijärjestelmiä. Muutamia esimerkkejä ovat näkymään perustuva, globaali ja paikallinen koordinaatisto. Lisäksi muunnoksen keskuspistettä voidaan tarvittaessa siirtää objektin ulkopuolelle. Tavallisesti objektia pyöritettäessä muunnoksen keskipisteenä toimii objektin keskipiste ja objekti pyörii oman akselinsa ympäri. Jos muunnoksen keskipiste asetetaan objektin ulkopuolelle, pyöritettäessä objekti kiertääkin tätä määriteltyä pistettä. Yksi käytännön esimerkki on kuution skaalaaminen siten, että skaalauksen keskipisteenä toimii yksi sivuista, jolloin kyseinen sivu pysyy vanhassa sijainnissaan ja toiset sivut siirtyvät sen keskipisteestä kauemmaksi. Muuten kuution jokainen sivu liikkuisi poispäin kuution keskipisteestä. (Giambruno 2003, 3–6.)

Useimmat pelit muuttavat 3D-mallia tuotaessa sen geometrian kolmioiksi, sillä suurin osa näytönohjaimista on suunniteltu juuri kolmioiden tehokkaaseen piirtämiseen. Siitä huolimatta kannattaa kuitenkin useimmiten pyrkiä rakentamaan 3D-mallit nelikulmioista, koska useimmissa 3D-mallinnusohjelmissa on joukko eri valintatyökaluja, jotka tarvitsevat nelikulmioista koostuvaa topologiaa toimiakseen. Tällainen on esimerkiksi reunasilmukan (edge loop) valinta. Tällaiset työkalut nopeuttavat huomattavasti 3D-mallintajan työtä.

Jokaisen nelikulmion voi muuttaa kolmioiksi kahdella tavalla. Se, kummin päin monikulmio jaetaan, vaikuttaa siihen, kuinka sen pinta varjostuu. Tästä johtuen kannattaa aina lopuksi tarkastaa, että pelimoottoriin tuotu, kolmioiksi muutettu geometria varjostuu toivotulla tavalla. Vääränlaisen lopputuloksen voi korjata muuttamalla ongelmalliset nelikulmiot kolmioiksi käsin. (Chadwick 2012.)

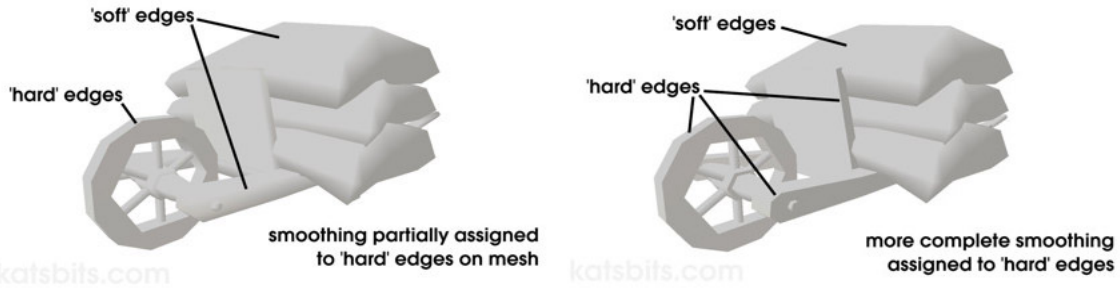
### 3.1.2 Kärkien ominaisuudet

Kuten jo edellä todettiin, suurempi määrä geometriaa mahdollistaa tarkempien yksityiskohtien mallintamisen, mutta vaatii toisaalta enemmän tehoa näytönohjaimelta. Yleensä 3D-mallin piirtämisen raskautta arvioitaessa on perinteisesti puhuttu monikulmio- tai kolmiomääristä (polygon count, triangle count). On kuitenkin kannattavampaa kiinnittää huomiota kärkien määrään (vertex count), sillä niillä on tiettyjä ominaisuuksia, jotka käytännössä aiheuttavat niiden monistumista. Jokaiseen kärkeen voidaan 3D-avaruuden sijainnin lisäksi varastoida yksi normaali, yksi materiaali sekä yksi sijainti tekstuuriavaruudessa eli toisin sanoen yksi UV-sijainti. Jos siis kahdella vierekkäisellä kolmiolla on esimerkiksi erilliset materiaalit, niiden väliset kärjet käytännössä kahdentuvat pelimoottorissa, vaikka näitä kärkiä käsiteltäisiin 3D-mallinnusohjelmassa edelleen yksinkertaisina. (Chadwick 2012; Galanakis 2008.)

Geometriassa normaali tarkoittaa suoraa, joka on kohtisuorassa jotain suoraa tai pintaa vastaan (Kivelä 2000). 3D-mallinnuksessa kärjen normaali on oletusarvoisesti keskiarvo niiden kolmioiden normaaleille, joita kyseinen kärki yhdistää. Kärkien normaaleista on hyötyä varjostuksessa: jos 3D-malli varjostettaisiin siten, että jokaisen kolmion jokainen kärki osoittaisi samaan suuntaan kuin kolmion normaali, lopputuloksena kolmioiden välillä olisi selvät kulmat ja jokaisen kolmion litteys olisi ilmeinen. Kun kärkien normaaleita käännetään, kolmioon, joka on todellisuudessa litteä, saadaan pehmeä varjostus, aivan kuin kolmiota olisi taivutettu. Näin ollen laskemalla kolmioiden normaaleista keskiarvot kärkien normaaleihin ja varjostamalla 3D-malli niiden perusteella sen pinnanmuodot saadaan näyttämään pehmeiltä. Tällaista varjostustapaa kutsutaan interpoloiduksi varjostukseksi. (Wagner 2004, 1; Summers 2004, 95.)

Interpoloidun varjostuksen avulla 3D-mallin pintaan saadaan siis tarvittaessa varjostettua pehmeitä muotoja käyttämättä paljoa geometriaa. Siinä on kuitenkin yksi ongelma: kaikista kulmista ei yleensä haluta pehmeästi varjostettuja. Tietynlaisissa tapauksissa tarvitaan selvä

ero kahden vierekkäisen monikulmion varjostuksessa. Varsinkin matalapolygonisissa malleissa käytetään siksi tekniikkaa, jota voidaan tekotavasta riippuen kutsua koviksi reunoiksi (hard edges) tai varjostusryhmiksi (smooth groups). Molemmilla keinoilla päädytään kuitenkin loppujen lopuksi samaan tulokseen – joukkoon monikulmioista koostuvia ryhmiä, joista jokainen varjostetaan interpoloidusti pelkästään samaan ryhmään kuuluvien monikulmioiden kesken (kuvio 3). Tähän lopputulokseen päästään myös fyysisesti hajottamalla objekti osiin niitä reunoja pitkin, joihin halutaan kova varjostus. Kun objekti piirretään pelissä, kärjet jakautuisivat näiden varjostussaumojen kohdalla joka tapauksessa (KatsBits 2014; Crytek 2014). Pehmeiden ja kovien reunojen tai varjostusryhmien määrittelyssä onkin lähinnä se ero kärkien jakamiseen verrattuna, että objektin muokkaaminen on helpompaa sen pinnan ollessa yhtenäinen. (KatsBits 2014.)

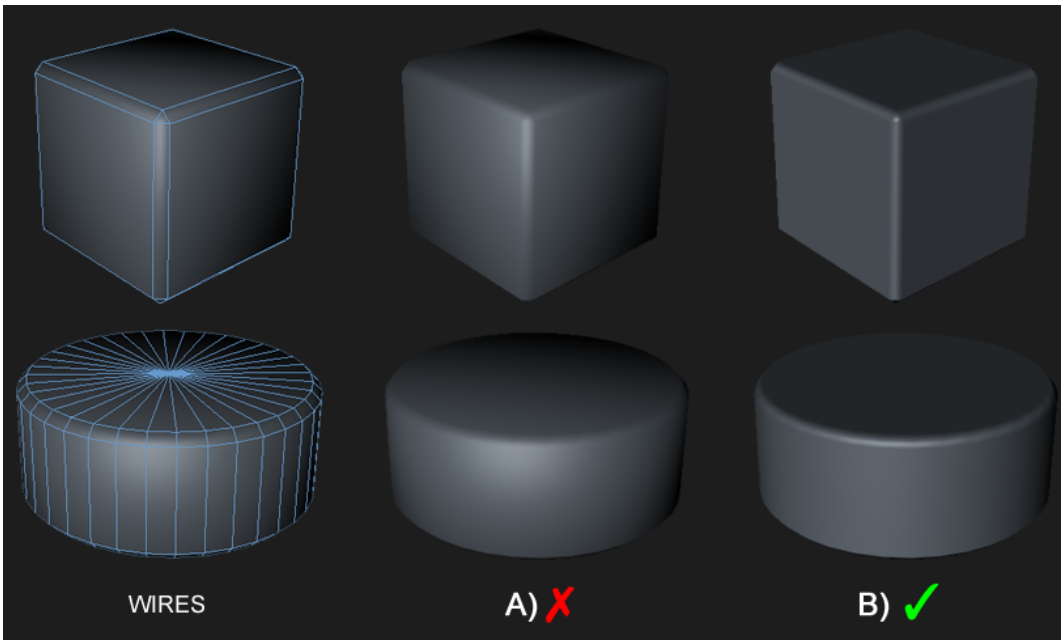



Kuvio 3. Pehmeiden ja kovien reunojen ero varjostuksessa. Vertailun vuoksi vasemmanpuoleisen kärryn aisa on jätetty pehmeäreunaiseksi. (Kuvion lähde: KatsBits 2014.)

Jotkin 3D-mallinnusohjelmat tarjoavat mahdollisuuden käännellä kärkien normaaleita käyttäjän haluamiin suuntiin. Tällainen mahdollistaa hyvin halpojen pehmeiden kulmien tekemisen. Pehmeät kulmat näyttävät teräviä luonnollisemmilta, sillä tosimaailmassa harva reuna on täysin terävä (Crytek 2014). Koska kärjet lopulta kahdentuvat varjostusryhmien välillä, kuutio, jonka jokainen tahko on oma varjostusryhmänsä, koostuu viime kädessä 12 kolmiosta, joissa on yhteensä 24 kärkeä. Jos kuution jokainen särmä levennetään kapeaksi nelikulmioksi ja kuutio varjostetaan interpoloidusti, saadaan 24 kolmiota, jotka koostuvat yhteensä 24 kärjestä. Näin ollen näytönohjaimella on molemmissa tapauksissa yhtä monta kärkeä käsiteltäväksi, jos kärjillä ei ole muita niitä jakavia ominaisuuksia. (Galanakis 2008.)

Pehmeäsärmäinen kuutio ei kuitenkaan näytä toivotunlaiselta vielä oletusnormaaleilla. Pehmennettyinä särminä toimivien monikulmioiden kanssa interpoloidut tahkojen normaalit aiheuttavat sen, että kuutio varjostuu aivan kuin pinnat eivät olisikaan tasoja. Haluttu loppu-

tulos saadaan vasta, kun kärkien normaalit käännetään osoittamaan tahkojensa normaaleiden suuntaisesti (kuvio 4). (FrostSoft 2012.)




Kuvio 4. Käsin muokatut kärkinormaalit. A-sarakkeen objekteilla on muokkaamattomat kärkinormaalit, eli jokaisen kärjen normaali on sen yhdistämien neljän monikulmion normaaleiden keskiarvo. Tällöin suoriksi tarkoitetut tahkot varjostuvat kuin ne olisivat kuperia. B-sarakkeen objektien kärkien normaalit on käännetty osoittamaan lähimmän suuren tahkon normaalin suuntaan, jolloin nämä tahkot varjostuvat oikein ja niiden väliin muodostuu pehmeästi varjostuvat särmät. (Kuvion lähde: FrostSoft 2012.)

Kärjelle voidaan antaa jokin RGB-väriarvo. Yleinen käyttötapa on lisätä 3D-mallin pintaan väriä. Tällöin kunkin kärjen kohdalla väritekstuurista saatu väri kerrotaan kärjen värillä, jolloin lopputuloksena saadaan luotua vaihtelua objektin pintaan, vaikka tekstuuri olisi toistuva. Kärkien värien avulla voidaan hallita objektin muitakin ominaisuuksia kuin sen pinnan näkyvää väriä, kuten vaihtumista eri tekstuurien välillä tai kasvien lehtien heilumista tuulessa. (Chadwick & Montero 2011; Crytek 2014.)

## 3.2 Teksturointi

Tekstuurit ovat tärkeä osa 3D-grafiikkaa. Niiden avulla 3D-mallien pinnat saadaan näyttämään paljon todellista geometriaansa yksityiskohtaisemmilta. Tekstuurit ovat siis tavallaan

väärennettyjä pintoja 3D-malleille. Objektin varsinaisten pinnanmuotojen ja pintakuviointien lisäksi niihin voidaan tehdä tarvittaessa myös yksityiskohtia, jotka olisivat todellisuudessa hieman irti kyseisestä pinnasta, esimerkiksi putkia tai johtoja. Tekstuurien avulla voidaankin usein saada sama lopputulos kuin lisäämällä paljon geometriaa, mutta huomattavasti vähemmällä laskentateholla. (Franson 2004, 2.)

### 3.2.1 UV-koordinaatisto

Koska tekstuurit ovat kaksiulotteisia kuvia, ne täytyy ikään kuin kääriä (unwrap) 3D-mallin pinnan päälle. Tätä käärimistä voisi verrata lahjan paketoimiseen: jos lahja ei ole laatikossa, voi olla työlästä saada paperi käärittyä sen ympärille siististi ilman ylimääräisiä ryppyjä.

Samoin kuin 3D-avaruuden ulottuvuudet ovat X, Y ja Z, tekstuurikoordinaatit käyttävät omaa koordinaatistoaan, jonka ulottuvuuksia merkitään kirjaimilla U, V ja W. Näistä U edustaa leveyttä, V korkeutta ja W syvyyttä. Yleisesti kuitenkin W jätetään mainitsematta ja puhutaan vain UV-koordinaateista. (Summers 2004, 88.)

Jokaiselle XYZ-avaruuden kärjelle on vastaava pisteensä UV-avaruudessa. Lisäksi jokainen näistä UV-pisteistä sijaitsee jossain kohti tekstuuria. Näin voidaan tarkasti määrittää, mikä kohta tekstuurista on minkin kärjen kohdalla 3D-mallin pinnassa. Toinen hyöty käärimisestä on, että objektin UV-koordinaatteja voidaan käyttää oppaana sen tekstuurien teossa. (Summers 2004, 310.)

Useimmissa 3D-mallinnusohjelmissa on mukana muutamia valmiita käärimismalleja eli projektioita. Niissä tekstuurikoordinaatit projisoidaan 3D-mallin pinnalle käyttäen apuna jonkin yksinkertaisen objektin eli primitiivin pintoja. Tällaisia primitiivejä ovat laatikko, sylinteri, pallo ja taso. Näistä on kuitenkin yleensä hyötyä vain hyvin yksinkertaisten muotojen teksturoinnissa. Objektin kaikkien tekstuurikoordinaattien ei kuitenkaan tarvitse olla yhtenäisenä palana, vaan niitä voidaan tarvittaessa hajottaa useiksi saariksi. Projektiomappaus on oikeanlaisten objektien kanssa käytettynä nopea keino luoda 3D-mallin geometrialle vastaavat UV-koordinaatit, mutta varsinkin monimutkaisten objektien kohdalla niitä käytettäessä saattaa olla vaikea ennustaa, missä kohti aikaansaaduista koordinaateista aiheutuu tekstuurin venymistä objektin pinnalla tai muita vastaavia artefakteja.

Jos objektissa on sellaisia muotoja, että primitiiviprojektiolla ei saada haluttua lopputulosta, voidaan valita objektin pinnasta pieniä alueita kerrallaan ja kääriä niitä yksitellen esimerkiksi tasoprojektiolla. Joissakin ohjelmissa on myös mahdollisuus tehdä kääriminen automaattisesti monikulmioiden välisten kulmien suuruuden perusteella. Ohjelmaan syötetään jokin arvo, minkä jälkeen ohjelma repii UV-koordinaatit erilleen niistä kohdista, joissa monikulmioiden välisen kulman asteluku ylittää syötetyn arvon. (Summers 2004, 308–310.)

### 3.2.2 Materiaalit

3D-grafiikassa materiaalilla yleensä tarkoitetaan joukkoa muuttujia, jotka määräävät sen, kuinka valo käyttäytyy 3D-mallin pinnalla. Tällaisia muuttujia ovat diffuusio, spekulaarisuus, pinnan epätasaisuus, peilaavuus, valon taittuminen objektin sisällä, näkyvyys tai läpinäkyvyys ja valaisevuus. Kaikkia näitä ominaisuuksia voidaan hallita pikselikohtaisesti kaksiulotteisten kuvien eli tekstuurien avulla. (Summers 2004, 77, 81.)

Diffuusio on yleensä materiaalin pääominaisuus. Se on väri, joka nähdään, kun materiaaliin osuva valo hajoaa tasaisesti joka suuntaan. Useimmiten se on materiaalin pääasiallinen väri: jos materiaalin halutaan olevan punainen, sen diffuusiokomponentista tehdään punainen. Diffuusio- eli värikarttaan tehdyt yksityiskohdat pysyvät objektin pinnassa paikoillaan riippumatta katsojan ja valonlähteen sijainnista. Tätä vastoin spekulaariheijastus liikkuu katsojan liikkuessa objektiin nähden. Pinnan spekulaarinen alue on se piste, josta suurin osa valoa heijastuu suoraan katsojaan. Spekulaarisen alueen väriä, kokoa, näkyvyyttä ja reunojen pehmeyttä voidaan säätää, ja näin saadaan aikaiseksi vaikutelma todellisesta materiaalista. Esimerkiksi kiinteissä, sileäpintaisissa esineissä, kuten biljardipalloissa, on pieni, kirkas ja teräväreunainen heijastus. Pehmeiden pintojen heijastus on yleensä vastaavasti suurempi, himmeämpi ja pehmeäreunaisempi. (Summers 2004, 100–102.)

### 3.2.3 Alfa

Digitaalisessa kuvankäsittelyssä väri syntyy sekoittamalla punaista, vihreää ja sinistä (RGB). Bittikartassa jokaiselle näistä väristä on oma kanavansa. 24-bittisessä tekstuurissa bitit on jaettu tasan punaisen, vihreän ja sinisen kanavan kesken, joten jokaisella näistä kanavista on

kahdeksan bittiä. Käytännössä tämä tarkoittaa sitä, että jokainen näistä kanavista koostuu 256 asteesta väriä. Kun kanavat yhdistetään, saadaan 256*256*256 eli yhteensä 16 777 216 eri sävyä. (Summers 2004, 47.)

Joissakin tapauksissa tekstuurin jokaisen pikselin läpinäkyvyyttä halutaan hallita erikseen. Tällöin tehdään erillinen läpinäkyvyyskartta, joka useimmiten varastoidaan diffuusiokartan neljänteen eli niin sanottuun alfakanavaan. Se, miten tarkkaan läpinäkyvyyttä voidaan hallita, riippuu siitä, montaako harmaan sävyä läpinäkyvyyskarttaan on varaa käyttää. Mitä enemmän sävyjä on, sitä useampia läpinäkyvyysasteita on käytettävissä. Toisaalta muistinkäyttö kasvaa sävyjen määrän myötä. (Chadwick & Montero 2013.)

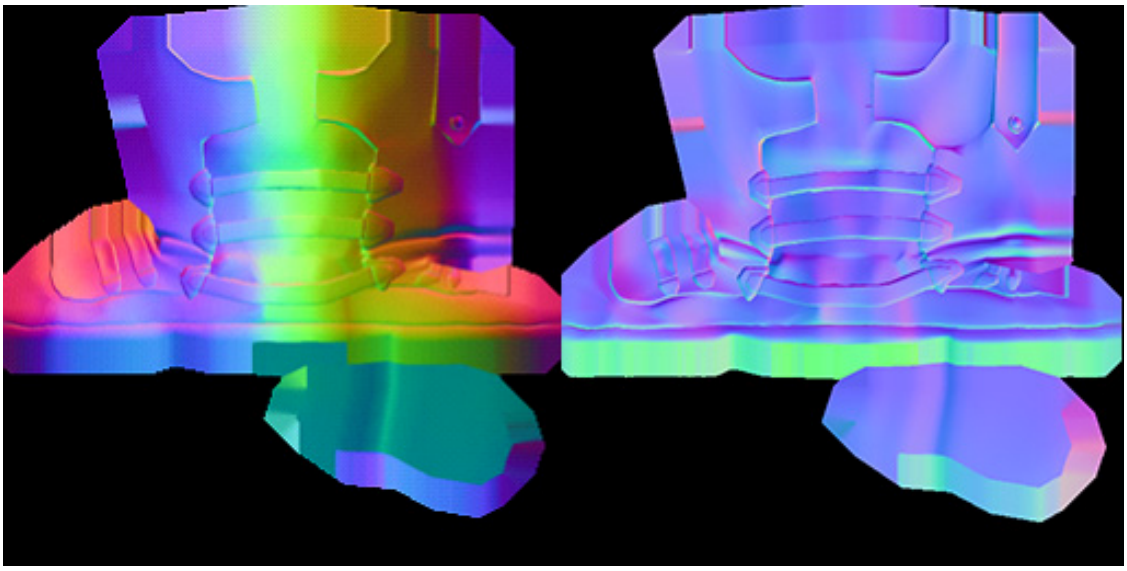
### 3.2.4 Normaalikartta

Aiemmissa luvuissa oli puhetta siitä, että mikäli objektiin halutaan mallintaa pieniä yksityiskohtia, sen pinnan topologia täytyy jakaa pienempiin monikulmioihin, mikä tuottaa näytönohjaimelle paljon uutta laskettavaa ja siten hidastaa kuvan piirtämistä näytölle. Pinnan hienoille yksityiskohdille on kuitenkin kehitetty huomattavasti tätä kevyempi toteutuskeino: normaalikartta.

Normaalikartan avulla pinnan todelliset normaalit korvataan joukolla uusia, yksi tekstuurin jokaista pikseliä kohti (van Waveren & Castaño 2008, 2). Normaalikarttaan varastoidaan suuntanormaaleita siten, että bittikartan punaiseen kanavaan tallennetaan X-, vihreään kanavaan Y- ja siniseen kanavan Z-suuntavektori. Vektorit voivat osoittaa positiiviseen tai negatiiviseen suuntaan, mutta värikanavilla on vain positiivisia arvoja. Toisin sanoen suuntavektorin arvot [-1,1] täytyy pystyä ilmoittamaan värikanavan arvoilla [0,1]. Värikanavan arvolla 0 suuntavektori on siis täysin negatiivinen, arvolla 0,5 se on neutraali ja arvolla 1 täysin positiivinen.

Normaalikarttoja on kahdenlaisia: värikkäitä objektitilanormaalikarttoja ja sinertäviä tangenttitilanormaalikarttoja (kuvio 5). Nämä eroavat toisistaan siten, että objektitilanormaalikarttojen suuntavektorit on laskettu suhteessa koko objektin sijaintiin ja kulmaan ja ne ovat siten absoluuttisia, kun taas tangenttitilanormaalikartan vektoreiden kulmat perustuvat siihen pintaan, johon ne on osoitettu. Tästä syystä objektitilanormaalikartat ovat hieman nopeampia renderöitäviä, mutta toisaalta ne ovat myös hyvin epäkäytännöllisiä. Tällaista normaalikarttaa

käyttävälle objektille ei oikein voida tehdä mitään sellaista animaatiota, joka muuttaisi sen geometriaa. Lisäksi jokainen objekti tarvitsee oman normaalikarttansa, kaikenlainen geometrian peilaaminen on mahdotonta eikä näitä normaalikarttoja voida kovin helposti toistaa pinnan yli. (Liman3D 2013; van Waveren & Castaño 2008, 2–3.)



Kuvio 5. Esimerkkejä normaalikartoista. Vasemmalla on objektitilanormaalikartta ja oikealla samasta objektista tehty tangenttitilanormaalikartta. (Kuvion lähde: Crytek 2013.)

Tangenttitilanormaalikarttojen sininen kanava eli Z-komponentti on aina positiivinen eli normaali osoittaa pinnasta poispäin (Liman3D 2013; van Waveren & Castaño 2008, 2–3). Tästä johtuu näille kartoille ominainen sinertävä väri. Koska tangenttitilanormaalikarttojen Z-komponentti on aina positiivinen, sininen kanava voidaan hylätä kokonaan ja laskea se nopeasti uudelleen X- ja Y-komponenteista renderöitäessä. Vaikka tangenttitilanormaalikartat ovatkin objektitilaan perustuvia karttoja hitaampia renderöitäviä, niitä käyttämällä voidaan toisaalta yhden kanavan hylkäämisen ansiosta säästää enemmän arvokasta tekstuurimuistia. (van Waveren & Castaño 2008, 9.)

### 3.2.5 Kohoumakartta

Ennen normaalikarttojen yleistymistä pinnanmuotojen varjostamiseen voitiin käyttää harmaasävykorkeuskarttoja. Tällainen kohoumakartta (bump map) toimii siten, että jokaiselle bittikartan pikselille lasketaan korkeusvektori sen perusteella, kuinka tumma kyseinen pikseli

on. Tummilla sävyillä saadaan painaumia ja vaaleilla kohoumia. Näistä korkeusvektoreista saadaan sitten laskettua suuntavektorit, joiden perusteella objektin pinta varjostetaan. (Blinn 1978, 290.)

Nykyisin suositaan normaalikarttoja, sillä ne ovat paljon kohoumakarttoja nopeampia renderöitäviä. Tämä johtuu siitä, että niissä on suuntanormaalit laskettuina valmiiksi, toisin kuin kohoumakartoissa, joiden korkeusvektoreista lasketaan suuntavektorit vasta renderöitäessä (van Waveren & Castaño 2008, 2.)

### 3.2.6 Spekulaarisuuskartta

Jos objektin pinnan spekulaarisuutta eli valon heijastumista halutaan hallita pikselikohtaisesti, se onnistuu spekulaarisuuskartan avulla. Perinteisesti se on ollut harmaasävykuva, jonka vaaleammat alueet saavat aikaan kirkkaamman spekulaarisen heijastuksen ja tummemmat alueet vastaavasti himmeämmän heijastuksen. Jos spekulaariheijastuksille halutaan tietty väri, spekulaarisuuskartasta voidaan tehdä myös värikäs. (Ward 2008, 138.)

### 3.2.7 Valokartta

Valokartta on tekstuurityyppi, johon varastoidaan valmiiksi laskettu valaistus. Todenmukainen reaaliaikainen valaistus vaatisi laitteistolta mahdottomia tehoja, joten on kätevämpää paistaa ympäristön valaistus valmiiksi tekstuuriin, jonka väreillä diffuusiotekstuurin värit sitten kerrotaan peliä suoritettaessa. Valokartoitusta käytettäessä voidaan hyödyntää sellaisiakin tekniikoita, joiden laskemiseen tarvitaan paljon aikaa, kuten valonsäteiden kimpoilu pinnalta toiselle.

Mikäli valokartoitettavan objektin UV-koordinaatteja ei ole järjestelty siten, että jokaisella pinnalla on oma alueensa UV-tilassa, niille täytyy useimmiten tehdä toisetkin UV-koordinaatit valokarttaa varten. Näin jokaisen erillisen pinnan ainutkertaiset valaistusolosuhteet saadaan tallennettua todenmukaisesti. On olemassa automaattisia UV-koordinaattien käärimis- ja pakkausalgoritmeja, mutta jos aika riittää, parhaan lopputuloksen saa kuitenkin määrittällä valokartoille tekstuurikoordinaatit käsin. (Chadwick, Klausen & Phoenix 2014.)

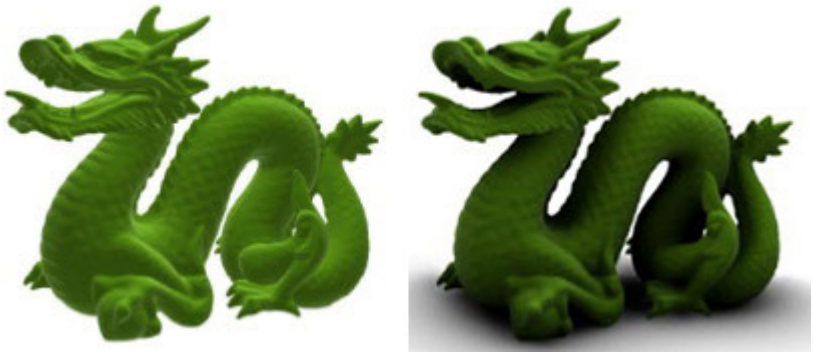
### 3.2.8 MIP-kartat

MIP tarkoittaa joukkoa tekstuurista tehtyjä pienempiä versioita, joihin tekstuuri vaihdetaan asteittain objektin siirtyessä kauemmaksi kamerasta. Jollei mipejä käytettäisi, tekstuurin siirtyessä kauemmaksi useita tekstuurin pikseleitä saatettaisiin joutua ahtamaan yhdelle näytön pikselille, mistä seuraisi välkkymistä pikselin vaihtaessa usein väriä. MIP-kartoitettu tekstuuri tarvitsee kolmasosan enemmän muistia, mutta hyöty kuvan laadussa on huomattava. Useimmat pelimoottorit kykenevät luomaan MIP-kartat automaattisesti, mutta ne voidaan myös tehdä käsin. Lyhenne MIP tulee latinan sanoista multum in parvo, paljon pienessä tilassa. (Bjorke 2005; Chadwick 2011.)

Mipit täytyy ottaa huomioon tekstuureita tehdessä. Yleensä peliobjektin UV-koordinaatit on tehty siten, että ne on jaettu erillisiksi alueiksi eli saariksi ympäri tekstuuria. UV-saaria ei kannata pakata liian lähelle toisiaan, vaan jokaisen UV-saaren ympärille tulisi jättää muutama pikselirivi kyseisen UV-saaren sisältämän tekstuurinosan pääväriä. Muuten eri väriset kohdat tekstuurista saattavat vuotaa toistensa päälle MIP-kartoissa. (Chadwick 2013.)

## 3.3 Lähialuevarjostus

Valaistuksen tarkka mallintaminen reaaliajassa on haastavaa käytettävissä olevan laitteiston laskentatehon rajallisuuden vuoksi. Ambient occlusion eli lähialuevarjostus on tekniikka, jolla voidaan huomattavasti parantaa valaistuksen laatua ilman liiallista suorituskyvyn uhraamista. Lähialuevarjostuksessa lasketaan pikselin varjostus suhteessa sen lähellä oleviin objekteihin, eli toisin sanoen tarkistetaan, varjostaako jokin objekti kyseistä pikseliä. Käytännössä lähialuevarjostus tuo maisemaan pehmeästi valoon vaihtuvat varjot (kuvio 6). Muuten peleissä tavallisesti nähdään kauttaaltaan tasaisesti tummia varjoja, jotka ovat hieman pehmeitä reunoiltaan.



Kuvio 6. Vasemmanpuoleinen objekti on valaistu pelkästään ympäristöön sijoitetuilla valonlähteillä. Oikeanpuoleiseen on lisäksi laskettu lähialuevarjostus, joka tuo paremmin sen muotoja esille. Samalla objektin ympärille saatiin hyvin pehmeäreunainen varjo. (Kuvion lähde: Geforce 2014.)

Lähialuevarjostus on periaatteessa yksinkertaistettu globaalin valaistuksen malli. Globaalissa valaistuksessa jokaisen pikselin väri lasketaan sitä ympäröivän puolipallon valaistuksen perusteella. Täysin ympäristölle paljastuvat alueet ovat kirkkaampia, kun taas muiden pintojen peittoon jäävät alueet ovat tummempia. Globaali valaistus kuitenkin vaatii monimutkaisia laskutoimituksia. Lähialuevarjostuksessa samankaltainen lopputulos saavutetaan nopeammin vertaamalla vierekkäisten pikseleiden suhteellisia syvyyksiä. (Geforce 2014.)

## 3.4 Optimointi

Virkistystaajuus eli ruudunpäivityksen tiheys riippuu kaikista yhdellä kertaa näytölle piirrettävistä objekteista, ja siksi se on yleensä huonoimmillaan pelin näyttävimpinä hetkinä. Usein nämä ovat samalla niitä hetkiä, jolloin pelaajat odottavat peliltä mahdollisimman sujuvaa pelattavuutta. Siksi suorituskyvyllisistä ongelmista muodostuu helposti huomattava turhautumisen aiheuttaja. Pelejä varten tehtyjen 3d-mallien ja ympäristöjen optimoinnissa onkin yhtä paljon kyse tällaisten peliä hidastavien tilanteiden välttämisestä kuin virkistystaajuuden yleisestä parantamisesta. (Provost 2003 a.)

### 3.4.1 Kärjet ja pinnat

Peliympäristöä optimoitaessa olennaisia tekijöitä ovat kärkien tiheys, tekstuurien tiheys ja näkyvissä olevan alueen koko. Pelin maisema tulisi rakentaa tasapainoisesti näiden välillä.

Ruudulle ei voida sijoitella rajattomasti kärkiä ja tekstuureita yhdellä kertaa. Näin ollen mitä suurempi alue on näkyvissä kerralla, sitä harvempaan yksityiskohdat täytyy jättää. Pienestä huoneesta voidaan siis suorituskyvyn osalta tehdä paljon yksityiskohtaisempi kuin laajasta ulkoilma-alueesta. Näkyvän alueen suuruus on siis tärkeä tekijä peliympäristön piirtämisen monimutkaisuuden kannalta. Mitä pienempi näkyvä alue on, sitä tiheämpään yksityiskohtia voidaan sijoitella ja sitä parempi virkistystaajuus saavutetaan.

Kärkien tiheys on yksinkertaisesti se, paljonko kärkiä on käytetty tietyllä alueella. Jos samalla alueella on useita tiheästä geometriasta koostuvia objekteja, virkistystaajuus laskee huomattavasti näiden objektien tullessa näkyviin. Tällaisten kalliiden objektien hajauttaminen tasaisesti ympäri aluetta ja niiden sijoittaminen mahdollisimman pieniin tiloihin on tärkeämpää ja usein myös helpompaa kuin geometrian vähentäminen yksittäisestä objektista. Kannattaa myös pitää mielessä, että alueen raskautta lisää myös se, jos esimerkiksi tekoälyn ohjaama objekti sattuu kulkemaan sinne.

Tekstuurien tiheys tarkoittaa sitä, paljonko tekstuurimuistia voidaan hyödyntää kullakin alueella. Suorituskyky kärsii, jos samalla alueella käytetään liikaa eri tekstuureita. Näin ollen myös eri tekstuurit tulisi hajauttaa tasaisesti ympäri aluetta. (Provost 2003 a.)

Grafiikkaprosessorit eivät käsittele peligrafiikkaa objektikohtaisesti, vaan eri objektit jakaantuvat erillisiin pintoihin materiaaliensa perusteella. Pinnan piirtäminen näytölle koostuu kahdesta toimenpiteestä: kärkien koordinaatit 3D-avaruudessa täytyy muuntaa kaksiulotteisen näytön koordinaateiksi ja niiden väliin muodostuvat pinnat täytyy piirtää. Molemmat toimenpiteet tapahtuvat samanaikaisesti, joten objektin piirtämisessä näytölle menee juuri yhtä kauan kuin hitaammassa näistä operaatioista. Se, kummassa menee kauemmin, riippuu objektista. Jos objektissa on paljon kärkiä pienellä alueella, niiden koordinaattien muuntamiseen kuluu enemmän aikaa kuin pintojen piirtämiseen. Guillaume Provost kutsuu tällaista pintaa muunnossidonnaiseksi (transform-bound). Muunnokseen kuluvaa aikaa pidentää myös, jos pintaa valaistaan usealla valonlähteellä tai jos siinä on monimutkaisia animaatioita. Mikäli pinnassa on kärjet harvemmassa ja niiden väliset kolmiot ovat siten näytöllä suuria, pinta on

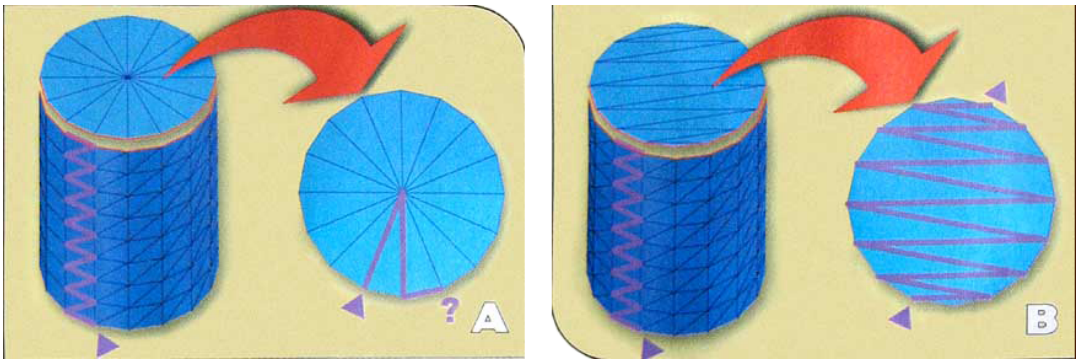
täyttösidonnainen (fill-bound), eli pintojen piirtäminen on kärkien muuntamista hitaampaa. Täyttöön kuluvaan aikaan vaikuttaa myös pinnassa käytettyjen tekstuurien määrä, tyyppi ja koko. Varsinkin pikselikohtaisesti valaistukseen vaikuttavat tekstuurit kuten normaalikartat ja valoa taittavat, läpinäkyvät materiaalit ovat raskaita piirrettäviä. Yksittäistä objektia optimoitaessa on tärkeää tunnistaa, kummanlaisesta objektista on kyse ja tehdä muutokset sen perusteella. Yleensä suuret pinnat kuten seinät ovat useimmiten täyttösidonnaisia ja animoitu geometria muunnossidonnaista.

Täyttösidonnaista objektia on sikäli vaikea optimoida, että pinnasta ei oikein voida tehdä pienempää kuin sen tarvitsee olla. Suorituskykyä voidaan kuitenkin tarvittaessa parantaa käyttämällä suurilla pinnoilla mahdollisimman halpoja ja yksinkertaisia materiaaleja (Provost 2003 a) sekä välttämällä useiden suurien, läpinäkyvien pintojen latomista päällekkäin (Provost 2003 b). Muunnossidonnaisten objektien kanssa on tärkeää huomioida varsinkin se, että siirtyessään kauemmaksi kamerasta niissä on yhä sama määrä kärkiä laskettavana, vaikka ne veisivät ruudulla miten vähän tilaa tahansa. Näin täyttösidonnainen objekti voi etääntyessään muuttua muunnossidonnaiseksi. Jos samaa objektia voidaan tarkastella sekä kaukaa että läheltä, ongelmaa voidaan helpottaa tekemällä objektista yksinkertaistettuja, alkuperäistä vähemmästä geometriasta koostuvia versioita eli lodeja (sanoista Level of Detail, eli yksityiskohtien määrä). Objektin siirtyessä tarpeeksi kauas kamerasta se voidaan vaihtaa tällaiseen lod-versioon kuvanlaadun huonontumatta. Lodien avulla tehdään siis geometrialle jotain pitkälti samantapaista kuin mitä mip-kartoilla tehdään tekstuureille.

Erityisen ongelmallisia saattavat olla sellaiset objektit, joista vain osa piirretään ruudulle kerrallaan. Tällaisen objektin jokainen, myös näkymän ulkopuolelle jäänyt kärki täytyy kuitenkin muuntaa ja objektin jokainen tekstuuri täytyy varastoida muistiin. Jos tällainen objekti tuottaa ongelmia, kannattaa kokeilla sen osittamista. (Provost 2003 a.)

Kärkien muuntamista voidaan nopeuttaa tekemällä objektin topologia siten, että sen kolmiot ovat mahdollisimman pitkälti siisteissä riveissä, joissa ensimmäisen kolmion jälkeen jokainen uusi kolmio jakaa kaksi kärkeä sitä edeltävän kolmion kanssa. Tällöin, kun rivin ensimmäisestä kolmiosta on kaikki kärjet muunnettu, laitteisto pystyy hyödyntämään pinnan topologiaa siten, että ensimmäisen kolmion kärjet käsiteltyään sen tarvitsee muuntaa jokaisesta seuraavasta kolmiosta vain yksi uusi kärki, kunnes rivi lopulta päättyy. Yleensä kolmiorivi päättyy sellaisessa kohdassa, jossa yksi kärki on jaettu hyvin monen kolmion kesken. Tällaista kolmioiden rivittämistä (triangle stripping) voidaan auttaa välttämällä tällaisia alueita ja hyvin ka-

peita kolmioita sekä muutenkin jakamalla geometria mahdollisimman tasaisesti pinnan halki (kuvio 7). (Provost 2003 b.)

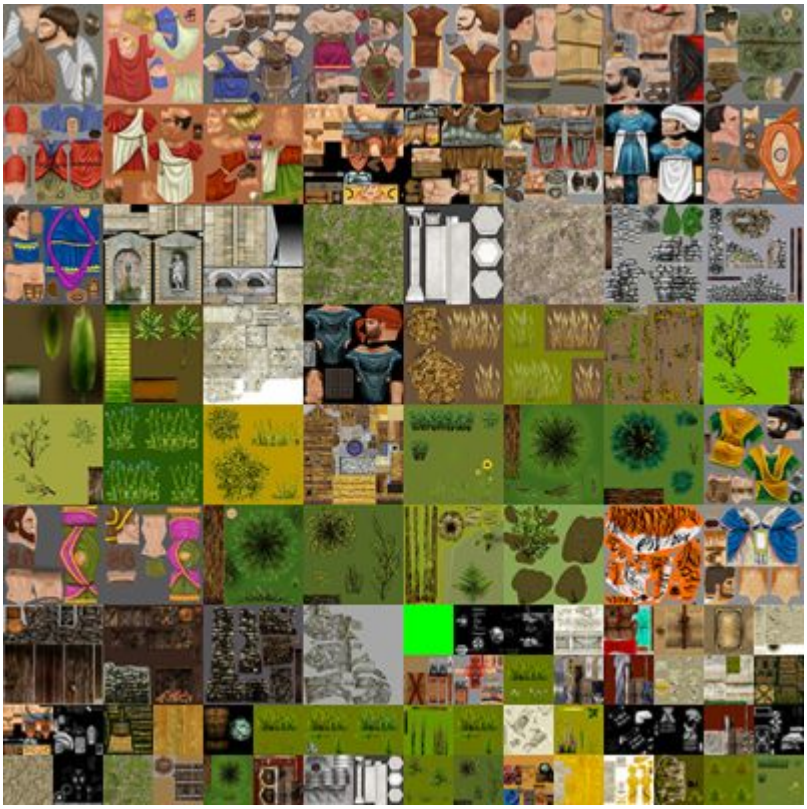



Kuvio 7. Lieriön A päädyssä kolmioista muodostuu niin sanottu tuuletin, eikä näytönohjain pysty jatkamaan kolmioiden rivittämistä tehokkaalla tavalla. Lieriössä B kolmiorivit jatkuvat koko objektin läpi, mikä tehostaa kärkien muuntamista. (Kuvion lähde: Provost 2003 b.)

### 3.4.2 Tekstuuriatlas

Kaikki näytölle piirrettävä kulkee jossain välissä suorittimen kautta. Tällaista yksittäistä suorittimen kautta kulkevaa pakettia kutsutaan eräksi (batch). Jokainen tekstuuri, objekti ja muut pelin osat ovat yleensä omina erinään. Erien määrää voidaan jossain määrin vähentää yhdistelemällä niitä isommiksi kokonaisuuksiksi. Tällainen eritteleminen (batching) nopeuttaa ruudunpäivitystä suorittimen työn määrän vähentyessä. (Jones 2011, 13.)

Yksi pahimmista piirtokutsujen lisääjistä on tekstuurin vaihtuminen, erityisesti jos näkyvissä on jopa tuhansia objekteja, jotka käyttävät yhteensä satoja erillisiä tekstuureita. Ongelmaa voidaan helpottaa paljon pakkaamalla useiden objektien tekstuurit yhteen suureen tekstuuriin (kuvio 8). Tällaista kokoelmatekstuuria kutsutaan tekstuuriatlakseksi. Niitä voidaan koostaa käsin tai apuohjelmien kuten Nvidian tekstuuriatlastyökalun avulla. Käsin tehtäessä lopputulosta on helppo hallita, mutta siihen saattaa kulua paljon aikaa. (Ivanov 2006, 1.)



Kuvio 8. Esimerkki yhdeksi atlakseksi pakatusta joukosta tekstuureita. (Kuvion lähde: Ivanov 2006.)

Tekstuuriatlasten käytöllä on kuitenkin myös mahdolliset varjopuolensa. Atlakseen pakattujen tekstuurien toistaminen yksittäisen pinnan yli on mahdotonta, ja kaikkien tekstuurikoordinaattien on sijaittava yksikköneliön sisäpuolella, eli arvojoukossa (0,0)-(1,1). Lisäksi kannattaa varautua siihen, että erillisten objektien tekstuurit vuotavat toistensa päälle, kun atlaksesta tehdään mip-karttoja. (Ivanov 2006, 2.)

### 3.4.3 Piilotettujen pintojen poisto

Toisinaan näyttötilassa saattaa olla objekteja, joita kuitenkaan ei juuri sinä hetkenä nähdä jonkin toisen objektin peittäessä ne näkyvistä. Piilotettujen pintojen poisto (occlusion culling) on tekniikka, jonka avulla tällaiset piilotetut objektit voidaan tilapäisesti jättää kokonaan renderöimättä, eikä niihin siten haaskata resursseja turhaan. Tämä ei kuitenkaan tapahdu automaattisesti, sillä yleensä 3D-tietokonegrafiikassa kauimpana kamerasta olevat objektit piirretään ensin. (Bushnaief 2013; Unity Technologies 2014 a.)

Esimerkki piilotettujen pintojen poistotyökalusta on Unityssä käytetty Umbra Softwaren väliohjelmisto Umbra (Bushnaief 2013). Se luo virtuaalisen kameran avulla kaikista pelialueen objekteista hierarkian, joka koostuu erillisistä joukoista mahdollisesti näkyvissä olevia objekteja. Tämä data koostuu käytännössä soluista, joihin koko pelialueen tilavuus jaetaan. Näiden solujen sisältämistä objekteista Unity pystyy nopeasti valitsemaan peliä ajettaessa, mitkä piirretään ja mitkä jätetään piirtämättä.

Tällaisessa soluihin perustuvassa menetelmässä solujen ja objektien koko vaikuttaa siihen, miten tehokkaasti näkymättömiä pintoja voidaan piilottaa. Esimerkiksi objektia, joka kattaa useita soluja, ei tyypillisesti piiloteta yhtä helposti kuin sellaista objektia, joka mahtuu yhden solun sisään. Näin ollen piilotettujen pintojen poistoa voidaan toisinaan tehostaa jakamalla suuria objekteja pienempiin palasiin. (Unity Technologies 2014 a.)

## 3.5 Modulaarisuus

Modulaarisuudella tarkoitetaan pelimaailman luomista siten, että koko kokonaisuuden mallintamisen sijaan tehdään huolella pienempiä osia, joita pystytään sitten sovittamaan toisiinsa kiinni ja näin voidaan koostaa yksityiskohtainen kokonaisuus vähemmässä ajassa (Jones 2011, 7). Samalla varmistutaan, että alueen ulkoasu on yhdenmukainen, kun tarvitaan vähemmän artisteja työstämään sen rakennuspaloja, mikä on aiheellinen huoli varsinkin isolla tiimillä (Perry 2002. 35). Lisäksi modulaariseen peliympäristöön voidaan tarvittaessa nopeasti tehdä korjauksia. Jos alue koostuu modulaarisista palasista, joissa myöhemmin ilmenee jotain muutettavaa, riittää, että korjattu versio palasesta tuodaan pelimoottoriin ja siten koko alue korjautuu kerralla. Tällaista alkuperäisen assetin mukana päivittyvää kopiota kutsutaan instanssiksi. (Jones 2011, 11; Perry 2002., 34–35.)

Modulaarisuudessa on sellainen ongelma, että pelaaja saattaa huomata maailman tai jonkin sen osan koostuvan toistuvista palasista. Tätä palasten toistumista voidaan kuitenkin helposti naamioida lukuisin keinoin. Muuten modulaarista maisemaa voidaan koristella irrallisilla objekteilla, esimerkiksi samojen seinäpalojen toistumista voidaan peittää asettelemalla seinän viereen proppeja. Myös maiseman koristeluun käytettävät propit voidaan tehdä modulaarisesti: jos huoneeseen halutaan tehdä kolme patsasta, voidaan tehdä erikseen kolme erilaista päätä, kolme torsoa ja kolmet jalat, ja koota näistä suuri joukko erilaisia patsaita (Perry 2002, 34). Jos palasten valaistuksissa on eroja, toistuminen on entistä vähemmän ilmeistä. Modu-

laarisissa paloissa on myös sellainen lisähyöty, että niitä voidaan joskus käyttää muutenkin kuin niiden alkuperäisessä käyttötarkoituksessa, esimerkiksi pylvästä ovenpielinä. Tästäkin voi olla apua modulaarisuuden piilottamisessa. (Jones 2011, 7–11.)

Ympäristön tekemiseen kuluvan ajan säästämisen lisäksi modulaarisuus auttaa myös suorituskykyä. Jokaista ympäristössä käytettyä instanssia samasta assetista ei tarvitse varastoida erikseen muistiin eikä lähettää näytönohjaimelle erillisessä erässä, toisin kuin jos koko ympäristö olisi mallinnettu yhdeksi objektiksi. Näin myös suorittimen läpi kulkevien tietoerien lukumäärää saadaan pienennettyä, mikä parantaa virkistystaajuutta. (Jones 2011, 13–16.)

Modulaarisuutta voidaan toteuttaa tarpeen mukaan erilaisissa mittakaavoissa (kuvio 9). Voidaan esimerkiksi tehdä kokonaisia rakennuksia tai ainakin muutamia toistuvia palasia, joista rakennukset koostetaan, tai yksityiskohtaisempaa lopputulosta tavoitellessa yksittäinen seinäkin voidaan rakentaa useista modulaarisista paloista. Yleensä yksityiskohtien tiheyden tarve määrittyy sen mukaan, kuinka nopeasti pelaaja liikkuu alueen läpi.

Kuvio 9. Modulaarisia rakennuspalikoita kahdessa eri mittakaavassa. (Kuvion lähde: Perry 2002, 32.)

Yleisesti modulaariset assetit suunnitellaan siten, että ne voidaan asetella pelimaailmaan käyttäen apuna ruudukkoa. Tällöin pienemmät yksityiskohdat tulisi toteuttaa siten, että isompien palasten koko on pienempien palasten koolla jaollinen. Esimerkiksi jos pelimaailman rakennuspalasten leveys olisi yleensä 256 yksikköä, pienemmistä yksityiskohdista olisi järkevää tehdä 128 tai 64 yksikköä leveitä. Näiden yksiköiden avulla voidaan myös määritellä pelin

hahmojen liikkeitä, esimerkiksi hyppy- ja istumisanimaatioiden asettamat vaatimukset objekteille, ja käyttää näitä tietoja apuna ympäristön luomisessa. Näiden selvittäminen hyvissä ajoin säästää paljon työtä myöhemmin. (Perry 2002, 32.)

Modulaarisuus ei kuitenkaan aina ole paras tapa peliympäristön toteuttamiseen. Joskus ympäristö saadaan toteutettua nopeammin ja halvemmin epämodulaarisesti. Esimerkiksi jos muuten modulaarisesti tehtyyn rakennukseen halutaan yksi huone, jolla on muista selvästi erottuva ulkoasu, tämä ainutkertainen huone on usein helpompaa tehdä luomalla joukko uniikkeja, epämodulaarisia 3d-malleja sitä varten. (Jones 2011, 30.)

# 4 PROJEKTI: BAARI SISÄTILOINEEN

Työ alkoi toimeksiantona, jossa tehtävänä oli suunnitella ja toteuttaa baari-ravintola sisätiloineen Colossal Order -peliyrityksen tuolloin kehitteillä olleeseen peliprojektiin. Kyseinen projekti jouduttiin valitettavasti perumaan opinnäytetyön toteutuksen aikana, mistä seurasi muutoksia työn tavoitteissa ja aikataulussa.

Projektin alussa toimeksiantaja toimitti listan objekteista, jotka rakennuksen sisältä tulisi löytyä, kuvauksen paikasta ja pelin graafisesta tyylistä, sekä esimerkkejä siitä, kuinka kyseinen tyyli voitaisiin käytännössä toteuttaa. Tavoitteena oli luoda uusklassistyylinen rakennus keveillä tieteisfiktioelementeillä. Sisätiloista pyrittiin tekemään likaiset ja hieman kärsineet – sellaiset, joissa moottoripyöräjengiläiset ja rikolliset saattaisivat viihtyä. Alustavasti oli myös puhetta rakennuksen propittamisesta, mutta peliprojekti peruuntui ennen kyseistä vaihetta. Tällöin päätettiin hyödyntää jo tehtyä työtä mahdollisimman perusteellisesti ja tutustua itsenäisesti Unity-pelimoottorin käyttöön peliympäristön tuotannossa. Proppien sijoittelun lisäksi opiskeltiin siis myös valaistusta ja perehdyttiin Unityn tarjoamiin optimointimenetelmiin.

## 4.1 Projektin tiedot

Suunnitteilla olleesta pelistä oli tarkoitus tulla läheltä kuvattu. Tällaisen pelin ympäristöjen täytyisi olla sen verran yksityiskohtaisia, että ne ovat hyvännäköisiä myös läheltä tarkasteltaessa. Tähän vaikuttavat sekä objekteihin käytetty geometrian määrä että tekstuurien tarkkuus.

### 4.1.1 Tekniset vaatimukset

Kaikki tekstuurit oli sovittava kolmeen tekstuurikarttaan: yksi rakennuksen ulkopinnoille, yksi rakennuksen sisäpinnoille ja näiden lisäksi sisäpuolen propeille yksi tekstuuriatlas. Kaikille näistä tuli tehdä diffuusio- ja normaalikartat. Tarvittaessa diffuusiokarttaan sai sisällyttää yksibittisen alfan, tosin rakennuksen ikkunoissa sai käyttää täydellistä alfaa. Jokaisen tekstuurin kooksi määrättiin 2048*2048. Mitään tiettyä määräystä sen suhteen, montako pikseliä täy-

tyisi mahduttaa metrille, ei annettu. Valon käyttäytyminen pinnoilla oli tarkoitus määrittää materiaalien kautta, eikä spekulaarisuuskarttoja siksi tarvittu.

Toimeksiantajan puolelta ei tullut erikseen määräyksiä käytettävissä olevasta geometrian määrästä, joten objekteihin käytettiin juuri sen verran, että ne saatiin näyttämään tarpeeksi hyviltä, kuitenkin välttäen tekemästä niistä liian raskaita. Tavoitteena oli luoda kilpailukykyistä peligrafiikkaa, joten esimerkkiä geometrian määrästä ja tekstuurien tarkkuudesta katsottiin viime vuosien peleistä, joita olivat muun muassa Call of Duty: Ghosts, Dishonored, Max Payne 3 ja Spec Ops: The Line.

### 4.1.2 Projektissa käytetyt työkalut

Toimeksiannon toteutuksessa käytetyt työkalut valittiin niiden teknisten ominaisuuksien, hintojen ja lisenssien perusteella. Lopputuloksesta oli alun perin tarkoitus tulla osa kaupallista projektia, joten vaikka ohjelmien täytyi olla halpoja, tai vielä mieluummin ilmaisia, niiden käyttöehtojen täytyi silti sallia ohjelmalla tuotettujen tiedostojen käyttäminen osana kaupallista projektia. Lisäksi ohjelmien täytyi soveltua peligrafiikan tehokkaaseen tuotantoon.

Mallintamiseen ja tekstuurikoordinaattien määrittelyyn sekä lähialuevarjostuksen paistamiseen käytettiin ilmaista Blender-työkalua. Blender oli jo aiemmissa projekteissa todettu tehokkaaksi 3D-työkaluksi, jossa on kuitenkin peliympäristögraafikon näkökulmasta yksi ongelma: sillä ei voi muokata kärkien normaaleita. Tämän toimeksiannon toteuttamista se ei kuitenkaan merkittävästi haitannut.

Tekstuureiden tekemiseen yritettiin alun perin käyttää ilmaista GIMP-kuvankäsittelyohjelmaa, mutta Photoshopiin tottuneelle yritys osoittautui turhauttavaksi ja tehottomaksi. Aikaa olisi kulunut liikaa uuden opettelemiseen ja eri tavalla toteutettujen perustoimintojen omaksumiseen, joten kahden viikon kokeilun jälkeen GIMPin tilalle vaihdettiin kuukausimaksullinen Photoshop CC. Normaalikartat luotiin paistamalla ne xNormalissa korkeuskartoista, jotka tehtiin Photoshopilla. Lopuksi luodut peliassetit vietiin Unity-pelimoottoriin, jossa ne järjesteltiin pieneksi maisemakokonaisuudeksi.

## 4.2 Projektin toteuttaminen

Rakennuksen tekeminen oli monivaiheinen projekti, jossa luotiin palanen peliympäristöä tyhjästä. Kokonaisuus täytyi ensin suunnitella, mallintaa ja teksturoida sekä vielä lopuksi rakentaa ja valaista pelimoottorissa. Eteneminen ei ollut aina suoraviivaista, vaan välillä asioita jouduttiin tekemään uusiksi ja siten palaamaan edelliseen vaiheeseen.

### 4.2.1 Suunnittelu

Projektin ensimmäinen vaihe oli referenssin kerääminen. Tarvittiin paljon valokuvia uusklassisista rakennuksista, sekä sisä- että ulkopuolelta, jotta saataisiin mahdollisimman kattava ymmärrys kyseiselle tyylille ominaisista piirteistä. Lisäksi kuvia etsittiin myös baareista ja projektiin tulevista propeista. Referenssin tarkoituksena oli varmistaa, että kaikilla objekteilla on niille olennaiset yksityiskodat, sekä antaa osviittaa siitä, minkälaista tunnelmaa ja millaisia yksityiskohtia ja värejä tosimaailman vastaavanlaisissa rakennuksissa on yleensä. Lisäksi toimeksiantajalta pyydettiin kuvia jo olemassa olevista peliasseteista tyylin yhtenäisyyden varmistamiseksi.

Kun kuvia oli kerätty tarpeeksi, oli aika tehdä mahdollisimman nopeita luonnoksia idean esittämiseksi toimeksiantajalle. Nämä luonnokset tehtiin Blenderissä, osittain piirtotaidon puutteesta, mutta pitkälti myös muista käytännön syistä: tällöin lopullista mittakaavaa oli helppo arvioida, ja saatiinpa rakennuksesta samalla jo yksinkertainen versio, jota voitaisiin jo sovittaa paikalleen pelimaailmassa, mikäli tällaiselle olisi tarvetta. Samalla tehtiin myös suurimmista huonekaluista ja muista tilaa vievistä objekteista yksinkertaiset versiot tarvittavan lattiapinta-alan arvioinnin helpottamiseksi. Tämäkin edesauttoi osaltaan idean esittämistä toimeksiantajalle. Samalla kokeiltiin myös alustavasti uusklassisten yksityiskohtien mallintamista rakennuksen julkisivuun.

Rakennuksesta luonnosteltiin yhteensä viisi eri versiota. Vain kaksi viimeistä näistä tuli valmiiksi asti ja ne esiteltiin toimeksiantajalle. Ensimmäinen luonnos hylättiin epäkäytännöllisyytensä vuoksi. Siinä keittiö, josta ruoka haetaan, ja tiski, jolta se ostetaan, päätyivät eri puolille rakennusta. Lisäksi rakennus jakautui lukuisiin pienempiin huoneisiin, mikä tuntui sekavalta. Toinen versio aloitettiin varaamalla paljon tilaa esiintymislavan ympärille. Huoneeseen teh-

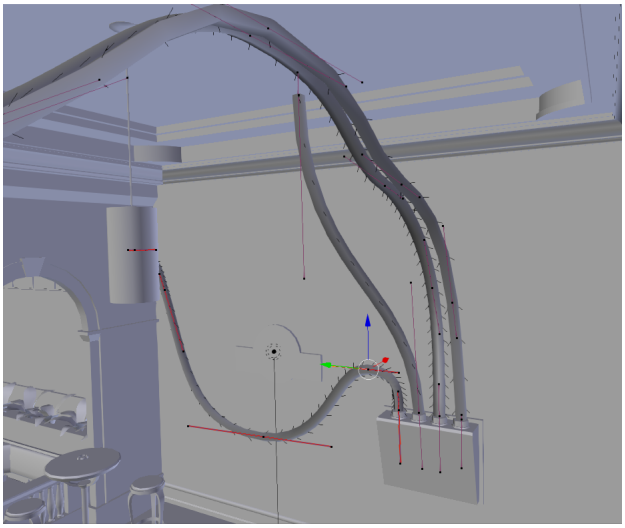
tiin myös ylös suuri parveke, jolta käsin asiakkaat voisivat seurata esityksiä. Tästä tosin syntyi aivan vääränlainen tunnelma: tarkoituksena ei kuitenkaan ollut tehdä suurta yökerhoa. Kolmas versio oli jälleen yksikerroksinen. Tässä versiossa ongelmana oli sen pohjaratkaisu, jonka vuoksi uusklassisille rakennuksille tyypillinen symmetrinen julkisivu ei olisi ollut sille mahdollinen. Jokainen näistä luonnoksista hylättiin aikaisessa vaiheessa ja ne jätettiin keskeneräisiksi.

Neljäs yritys alkoi jo tuottaa tulosta. Tällä kertaa rakennuksesta tehtiin hyvin pieni, hieman amerikkalaisten diner-ravintoloiden henkinen (kuvio 10). Ravintolassa oli tällöin vain kaksi pöytää, jotka sijaitsivat etuovelta katsottuna huoneen oikealla puolella. Vasemmalla oli baaritiski. Tähän versioon tehtiin myös yksinkertaiset ikkunat ja yksinkertainen ovi sekä ulko- ja sisäkatto. Lisäksi rakennuksen ulkopuolelle merkittiin paikat kahdelle takaovelle. Huoneen perällä oli pieni esiintymislava, ja sen vieressä oviaukko pienempään tilaan, jossa sijaitsi takaovi ja käymälöiden sisäänkäynnit. Levyautomaatti sijoitettiin pääoven lähelle baaritiskin viereen. Tämä suunnitelma vaikutti jo toimivalta, joten se esiteltiin toimeksiannon ohjaajalle. Hän sanoi sen olevan muuten ihan hyvä, mutta näin pieneen tilaan olisi ollut vaikea tehdä mitään pelattavaa, joten ruokailutilasta täytyi tehdä huomattavasti leveämpi. Katto poistettiin väliaikaisesti muokkaamisen helpottamiseksi ja ruokailutilan sivuseinät siirrettiin kauemmas toisistaan. Samalla rakennuksen takaosaa siirrettiin pari metriä taemmas. Vapautuneeseen tilaan saatiin kuusi uutta pöytää ja suurempi esiintymislava. Laajennettu huone vaikutti tosin hieman tylsältä ja hallimaiselta, joten keskelle huonetta lisättiin neljä pylvästä vaihtelun lisäämiseksi. Samalla tila jakautui entistä selvemmin ruokailutilaksi ja kulkuväyläksi. Vaikutelmaa tehostettiin lisäämällä pylväiden väliin matala seinä. Levyautomaatti siirrettiin tämän uuden seinän viereen, sillä pääoven viereisten seinien eteen mahtui nyt ruokapöytiä. Toimeksiantaja hyväksyi suunnitelman, joten oli aika aloittaa lopullisten 3D-mallien luominen.

Kuvio 10. Viimeisiä luonnoksia. Pohjaratkaisu alkoi jo olla tässä selvillä.

### 4.2.2 Mallintaminen ja UV-kääriminen

Mallinnusprosessi aloitettiin koristeobjekteista eli propeista, jotta lopullista tyyliä voitaisiin jo kokeilla ottamatta kuitenkaan vielä paineita koko kokonaisuudesta. Tällaisia proppeja olivat esimerkiksi pöydät, baarijakkarat ja levyautomaatti. Useimpien objektien mallinnus aloitettiin primitiivistä, joka oli objektista riippuen joko taso, kuutio tai lieriö. Käyriäkin päädyttiin hyödyntämään, tosin vasta projektin loppuvaiheessa, katossa ja lavan takana kiemurtelevien koristevalojen johtoihin sekä ilmastointijärjestelmän letkuihin (kuvio 11).



Kuvio 11. Pehmein kurvein kiemurtelevien objektien, kuten näiden letkujen, mallintaminen onnistuu näppärästi käyrien avulla.

Alustavan irtaimiston jälkeen tehtiin paranneltu versio itse rakennuksesta, jotta siihen kiinteästi liittyvät propit, kuten baaritiski tai kattoa ja seiniä pitkin kulkevat putket, olisi helpompi sovittaa paikoilleen, ja lopuksi sitten mallinnettiin nämä suuremmat koristeobjektit. Mallinnusta nopeutti se, että luonnoksen geometriaa pystyttiin pitkälti hyödyntämään edelleen. Pieniä muutoksia jouduttiin tosin tekemään, esimerkiksi suuria pintoja jouduttiin jakamaan pienemmiksi monikulmioiksi reunasilmukoiden avulla, jotta niiden tekstuurikoordinaatit saataisiin käärittyä mahdollisimman vähäiseen tilaan. Tällöin tekstuuritilaa ei tarvitse haaskata tekemällä koko pinnalle omaa tekstuuria, vaan riittää, että tekstuuri tehdään vain lyhyelle pätkälle seinää ja sitten tämä sama osa tekstuurista toistetaan jokaisella yksittäisellä palasella. Haasteellista tässä on lähinnä sellaisen tekstuurin tekeminen, jossa pintakuvion toistuminen ei ole liian ilmeistä.

Toimeksiantajalta ei tullut määräyksiä sen suhteen, paljonko geometriaa olisi varaa käyttää suorituskyvyn vaarantumatta, joten esimerkkiä katsottiin viime vuosien läheltä kuvatuista peleistä, niin geometrian määrän kuin tekstuurien tarkkuudenkin suhteen. Käytännössä geometriaa käytettiin mahdollisimman vähän niin, että haluttu muoto saatiin sillä kuitenkin aikaan. Luonnollisesti pienikokoisiin objekteihin käytettiin vähemmän geometriaa. Lisäksi kärkien määrää saatiin vähennettyä jättämällä pois sellaiset pinnat, joita pelaaja ei varmasti näe koskaan ja jotka eivät vaikuta pelaajan näkemien pintojen varjostukseen. Tällaisia pintoja ei kuitenkaan loppujen lopuksi ollut montaa, sillä oletettiin, että kamera pääsee liikkumaan rakennuksessa vapaasti. Yksi esimerkki tällaisesta poisjätetystä pinnasta on katon syvennyksiä reunustavien puulistojen kattoa kohti osoittavat pinnat.

Kuten luvussa 3.1.2. mainittiin, jokainen kärki voi olla osa vain yhtä varjostusryhmää, ja jokaisella kärjellä voi olla vain yksi U- ja yksi V-tekstuurikoordinaatti. Kärkien määrää pyrittiin siis mahdollisuuksien mukaan säästämään myös tekemällä UV-saumat varjostusryhmien ehdoilla, toisin sanoen objektin ehdottomasti vaatiessa useamman varjostusryhmän UV-saumat aseteltiin koville reunoille. Lieriömäisten muotojen pinnat varjostetaan yhtenä ryhmänä, joten UV-saumojen aiheuttamilta kärkien kahdentumisilta ei voida näissä tapauksissa säästyä samalla tavalla. Tällaisia tapauksia ovat esimerkiksi suorat putket ja pullojen sivut. Putkien osalta ongelma voitiin ratkaista mallintamalla ne käyttäen parillista määrää sivuja, joista puolille määriteltiin UV-koordinaatit ja toisen puolen UV:t peilattiin sitten näiden päälle. Pullojen kohdalla tämä ei ollut mahdollista, sillä toisen puolen peilautuessa sen etiketin tekstit olisivat tulleet takaperin. Myös putkien mutkat tehtiin käyttäen yhtä saumaa tekstuurien vääristymisen välttämiseksi. Saumattomalla UV-käärimisellä saavutettu säästö ei yleensä ole yksit-

täisen objektin kohdalla suuri: esimerkiksi rakennuksen sisäkattoon tehtiin kahdenlaisia putkia, joista paksummissa on kuusi sivua, eli niiden jokaisen palasen kummassakin päässä on kuusi kärkeä. Mikäli UV-koordinaatit olisi tehty käyttäen yhtä saumaa, yhdessä putken pätkässä olisi 12 kärjen sijaan 14. Lopullisessa tuotoksessa on kuitenkin satoja, jopa tuhansia kohtia, joissa kärkiä on säästetty tällä tavalla, eikä se olennaisesti lisää objektin toteuttamiseen kuluvaa aikaa. Paremminkin päinvastoin: teksturointiin tarvitaan vähemmän aikaa, kun teksturoitavaa pinta-alaa on vähemmän.

Tällainen optimointiajattelu näkyy myös joidenkin proppien suunnittelussa. Esimerkiksi baarijakkarassa on kolme jalkaa, sillä uusklassista estetiikkaa pyrittiin hakemaan laittamalla siihen useampi, koristeellisesti muotoiltu jalka. Tällainen muotoilu ei kuitenkaan onnistu ilman jalan jakamista useilla reunasilmukoilla, joten jokaiseen jakkaraan jalkoja käytettiin mahdollisimman vähän. Kolme on pienin määrä, jolla istuin näytti vielä tukevalta.

Toimeksiantajalta ei tullut myöskään mitään tiettyä toivetta tekstuurien keskimääräiseksi pikselimääräksi metriä kohden, joten objektien UV:t toteutettiin tasapainotellen visuaalisen laadun ja tekstuuritilan mahdollisimman tehokkaan käytön välillä. Apuvälineeksi tehtiin testitekstuuri, joka koostui 32 pikselin levyisistä ruuduista, joista joka toisen sisällä oli tiheässä pisteitä helpottamaan pinnan lopullisen tarkkuuden arviointia, ja joka toisessa ruudussa oli nuoli, joka osoitti joko ylös tai oikealle. Näiden nuolien avulla kartoitettiin tekstuurien peilautumista ja mahdollisia uusia saumoja, joita voitaisiin mahdollisesti poistaa peilaamalla.

Projektin alussa objektit mallinnettiin ensin loppuun asti ja niiden UV-koordinaatit tehtiin vasta sitten. Kokemuksen myötä omaksuttiin kuitenkin tehokkaampi, modulaarisempi työskentelytapa, jossa tehtiin teksturointia vaille valmiita paloja, joista lopullinen objekti sitten koostettiin. Esimerkiksi lopulliseen versioon hyllystä (kuvio 12) mallinnettiin ja myös alustavasti UV-käärittiin ensin kaikki toistuvat elementit, joista se koostuu, kuten kaappien ovet, hyllyt ja niitä erottavat väliseinät. Lopullinen objekti kasattiin näistä elementeistä vasta sitten. Näin saatiin valmiiksi jokaisen toistuvan elementin kaikki UV-saarekkeet siisteinä nippuina, jotka voitiin sitten lopullisen kokonaisuuden kasaamisen jälkeenkin lajitella helposti tekstuurissa sopivaan vapaaseen kohtaan. Tämä on tehokasta työajan säästämisen lisäksi myös tekstuuritilan käytössä. Varjopuolena tosin tekstuurin toistuminen elementin kopioissa on näkyvää. Hyllyn tapauksessa varsinkin suurissa elementeissä tekstuurin toistuminen käy ilmeiseksi. Tätä voidaan kuitenkin peittää tehokkaasti laittamalla hyllyyn esineitä, kuten pulloja. Alussa modulaarista työskentelyä jarrutti lähinnä se, että ei ollut kunnollista käsitystä siitä, mitä

oikeastaan oltiin tekemässä. Kun tyylistä saatiin kunnolla kiinni, yksittäisten objektien lopputulosta oli helpompaa visioida ennen mallintamista, ja niiden tekoprosessi pystyttiin siksi suunnittelemaan tapauskohtaisesti mahdollisimman tehokkaaksi.

Kuvio 12. Modulaarisesti toteutettu hylly.

### 4.2.3 Teksturointi ja lähialuevarjostuksen paistaminen

Ennen teksturoinnin aloittamista päätettiin jo kokeilla lähialuevarjostuksen paistamista. Heti huomattiin vakava ongelma: tekstuurikoordinaatit oli tehty keskittyen pelkästään mahdollisimman suuren pikselimäärän pakkaamiseen jokaiselle pinnalle. Tästä tietenkin seurasi, että lähialuevarjostusta ei saatu toteutettua tekstuureissa riittävän tarkasti. Lähialuevarjostuksessa kun esimerkiksi seinät varjostuvat eri tavoin nurkissa kuin pitkillä, suorilla pätkillä. Kun seinän jokainen pätkä on pakattu täsmälleen samaan kohti tekstuuria, lopputuloksena saadaan joko suoran seinän valoisuus myös nurkkiin tai vaihtoehtoisesti nurkan varjo toistumaan seinän pitkillekin suorille. Seurauksena kaikkien objektien tekstuurikoordinaatit järjesteltiin uu-

delleen, mikä joissakin tapauksissa oli helpompaa tekemällä UV-kääriminen kokonaan uusiksi, ja tällä kertaa tekstuuritila jaettiin lähialuevarjostuksen ehdoilla. Viimein varjostus näytti halutunlaiselta, mutta koska samaa tekstuuritilaa käyttikin nyt moninkertainen määrä pintoja, jokaiselle pinnalle riitti tekstuurin pikseleitä huomattavasti pienempi määrä, eikä tekstuureista olisi siksi saatu riittävän teräviä. Tämän toisen epäonnistumisen seurauksena saatiin sentään kunnollinen käsitys siitä, kuinka lähialuevarjostus käytännössä toimii erilaisissa paikoissa, ja tätä ymmärrystä voitiin hyödyntää lopullisessa UV-järjestelyssä, jossa lähialuevarjostuksen laatua uhrattiin kuitenkin jälleen tekstuurien terävyyden hyväksi, sillä suttuiset tekstuurit pistävät silmään selvästi puolihyvin tehtyä lähialuevarjostusta enemmän – varsinkin kun sen tarkoitus oli kuitenkin olla vain yksi läpinäkyvä komponentti diffuusiotekstuurin päällä. Sisäpuolella tämä kävi erityisen selväksi: pintoja oli niin paljon, että jokainen mahdollinen pikseli jouduttiin käyttämään niiden tekstuurien terävyyteen, eikä tilaa lähialuevarjostuksen eroille käytännössä jäänyt. Yleispätevän varjostuksen paistaminen ei näyttänyt vaivan arvoiselta, joten sellainen päädyttiin maalaamaan käsin.

Lähialuevarjostuksen paistaminen vaatii valmisteluja. Jos samassa tekstuurissa on useita UV-saarekkeita päällekkäin, lopputulos on useiden erilaisten päällekkäisten varjostusten sekasotku. Lopullisissa objekteissa tekstuuritila optimoitiin siten, että diffuusiokartan puolesta samanlaiselle elementille suotiin kopio lähialuevarjostusta varten vain, jos varjostuksen ero oli tarpeeksi selvä. Esimerkiksi ulkopuolta kiertäville tiiliseinille varattiin tekstuuriin kaksi palaa: yksi suoralle seinälle ja yksi nurkalle. Blender paistaa tekstuurit ainoastaan tekstuuritilan yksikköneliön sisäpuolella oleville saarekkeille, joten varjostuksen puhtaus pystyttiin varmistamaan siirtämällä kaikki kopiot jokaisesta elementistä tämän neliön ulkopuolelle paistamisen ajaksi.

Diffuusiokartat luotiin sekä valokuvia muokkaamalla että käsin maalaamalla. Kunkin objektin kohdalla arvioitiin erikseen, kumpaa keinoa käyttäen haluttu lopputulos saavutettaisiin mahdollisimman nopeasti. Useimmiten käytettiin molempia tekniikoita yhdessä. Valokuvia käytettäessä niistä pyrittiin tekemään hieman maalatun näköisiä Surface Blur- ja Paint Daubs -suodattimien avulla. Surface Blur -suodatin sumentaa osia kuvasta sävyerojen perusteella, jolloin suuremmilta pinnoilta katoaa kaikkein pienimmät yksityiskohdat. Paint Daubs muuttaa kuvan sen näköiseksi kuin se koostuisi siveltimellä kankaalle painetuista maalitäplistä. Lopuksi diffuusiokarttoihin lisättiin pienellä näkyvyydellä aiemmin paistettu lähialuevarjostus.

Normaalikartat tehtiin xNormalilla Photoshopissa tehdyistä kohoumakartoista. Nämä kohoumakartat luotiin maalaamalla ensin syvennykset ja kohoumat uuteen kerrokseen diffuusiotekstuurin päälle, minkä jälkeen tämän uuden kerroksen alle luotiin toinen uusi kerros, joka täytettiin 50% harmaalla. Photoshop CC:ssä on omakin kohoumakarttoja käyttävä normaalikarttageneraattori, mutta se teki pinnanmuodoista vähän liian jyrkkiä, eikä siinä ollut muokattavia asetuksia.

Rakennuksen sisälle valittiin tällaiselle baari-ravintolalle tyypillinen lämmin, viihtyisä väriteema. Seinistä tehtiin pääasiallisesti vaaleanruskeat, ja suurin osa huonekaluista päällystettiin tummanruskealla puutekstuurilla. Toiseksi pääväriksi otettiin punainen, jota sijoiteltiin aika tasaisesti ympäri rakennusta, esimerkiksi istuinten pehmusteisiin, puisten pintojen koristeellisiin paneeleihin ja verhoihin. Katto jätettiin uusklassismille ominaiseksi vaaleaksi, jotta tila tuntuisi avarammalta. Punertaviin pääväreihin tuotiin vaihtelua pääasiallisesti vihreällä, jota käytettiin esimerkiksi huonekasveissa ja katon koristelampuissa. Lisäksi sinne tänne ripoteltiin pieniä määriä sinistä ja keltaista. Lopputuloksena saatiin eräänlainen lähiväriharmonia kolmella aksenttisävyllä. Tavoitteena oli saada aikaan yhtenäinen värikokonaisuus kuitenkin siten, että ihan jokainen proppi ei näyttäisi siltä, kuin se olisi varta vasten luotu tätä paikkaa varten. Puisten objektien kohdalla se ei niinkään haittaa, sillä niitä on tosimaailmassa kuitenkin aika helppo maalata, mutta tässä suhteessa ongelmallisempia olivat esimerkiksi pullot ja kasvit sekä kuvitteellisen juomatuotemerkin logolla koristettu levyautomaatti.

### 4.2.4 Tyylin toteutus ja kokonaisuuden sommittelu

Tavoitteena oli luoda uusklassistyylinen, kulunut ja likainen baari-ravintola, jossa on myös joitain tieteisfiktioelementtejä mukana. Uusklassismia pyrittiin toteuttamaan erityisesti rakennuksen arkkitehtuurin kautta. Seiniä reunustavat koristeelliset lattia- ja kattolistat, kattoa tukevat muotoillut pylväät ja osissa rakennusta ja monissa huonekaluissa on nähtävissä koristekaiverruksia. Suuressa osassa propeista on muutenkin paljon uusklassisia vaikutteita.

Kulumat ja lika toteutettiin pitkälti tekstuureissa. Kulmiin maalattiin hankaumia ja suurille puupinnoille sellaisia kohtia, joissa pinnan lakka- tai maalipinta on pahasti kärsinyt. Metallisilla pinnoilla on paljon ruostetta. Geometrian avulla iän merkkejä saatiin muun muassa lisäämällä lattiaan sojottavia lautoja ja paikoitellen vääntelemällä putkia hieman. Lisäksi seinille tehtiin leijuvien monikulmioiden ja alfan avulla tapettiin repeämiä. Muutamia hämähäkin-

verkkoja tehtiin erillisinä proppeina. Tieteisfiktiopuolta toteutettiin johtojen ja putkien lisäksi laitteiden suunnittelussa, esimerkiksi kassaan ja levyautomaattiin laitettiin kosketusnäytöt.

Ulkopuolen muotoilussa ja teksturoinnissa uusklassismin juhlavuutta pyrittiin yhdistämään rakennuksen vähemmän arvokkaaseen luonteeseen. Etuseinästä tehtiin vaalea, koristeellinen ja symmetrinen, ja sitä varjostamaan tehtiin suuri, kaiverruksin koristeltu korniisi. Sivuilla etuseinä kuitenkin vain juuri ja juuri ohittaa sivuseinien tasot, ikään kuin yrityksenä peittää nämä paljon halvemmalla tavallisista tiilistä, laastista ja betonista koostetut seinät.

Sommittelu suunniteltiin alustavasti Blenderissä, jotta objekteja voitaisiin tarvittaessa vielä muokata. Erityisesti sommitteluun vaikuttivat tietysti suurimmat objektit, eli rakennuksen oman geometrian lisäksi putket, johdot, baaritiski ja sen takana olevat hyllyt, esiintymislava ja katon ilmanvaihtokanava. Huomiota kiinnitettiin myös valoa tuottavien objektien, eli kattovalaistuksen lisäksi esimerkiksi uloskäyntikylttien, neonvalojen ja levyautomaatin sijoitteluun sekä värien käyttöön.

Lopuksi rakennuksen ulkopuolelle tehtiin nopeasti pieni, yleisesti kaduilla ja kujilla käytettävistä objekteista koostuva proppisarja. Nämä propit teksturoitiin 1024*1024-kokoisella diffuusioatlaksella.

### 4.2.5 Objektien viimeistely ja pelimoottoriin vieminen

Ennen objektien viemistä Unity-pelimoottoriin varmistettiin, että niiden pivot-pisteet olivat oikeilla kohdillaan ja että niillä oli oikea skaala, kulma ja sijainti. Näin varmistettiin, että objekti oli mahdollisimman helppo sijoittaa pelimaailmaan, ja että se näytti juuri siltä kuin sen pitikin. Useimmissa objekteissa pivot sijoitettiin sen kohdan keskelle, jossa objekti kohtaisi toisen objektin pinnan. Esimerkiksi vaasin pivot sijaitsee sen ulkopohjan keskellä. Seinäkellon pivot on leveys- ja pituussuunnissa sen keskikohdassa, mutta syvyyssuunnassa siinä kohti, jossa kello koskettaa seinää. Poikkeuksia tällaisesta käytännöstä ovat uniikit objektit, jotka mitoitettiin tiettyihin kohtiin rakennusta. Tällaisia objekteja ovat baaritiski ja sen takana olevat hyllyt, katossa ja seinillä kiemurtelevat putket ja johdot sekä ruokailutilaa reunustavat saniaiset. Näiden objektien pivot-pisteet sijoitettiin samaan kohtaan rakennuksen pivotin kanssa, jotta niiden asettaminen pelimaailmaan kävisi nopeammin. Toinen poikkeus projektin yleisestä pivot-käytännöstä oli pöytäobjektin penkkien selkänojiin kiinnittyvä naulakko. Sen

pivot asetettiin samaan kohtaan sellaisen pöydän pivotin kanssa, johon naulakko on kiinnitettynä. Naulakko tehtiin kuitenkin erillisenä objektina, koska se tarvitsee pöydän kummallekin puolelleen näyttääkseen tukevalta. Jos naulakko olisi laitettu pöydän kanssa yhteen objektiin, olisi tarvittu erikseen naulakolliset ja naulakottomat versiot ravintolan kummastakin ruokapöytätyypistä, mikä ei olisi ollut yhtä optimaalista.

Kun kaikki objektit saatiin vietyä Blenderistä fbx-tiedostoiksi, ne tuotiin Unityyn. Tässä vaiheessa havaittiin muutama mielenkiintoinen asia. Ensinnäkin objektit olivat erikoisessa kulmassa: se, mikä oli Blenderissä objektin etupuoli, osoittikin Unityssä alaspäin. Blenderissä ja Unityssä on 3D-avaruuden koordinaatit määritelty siten eri tavoin, että Blenderissä Z-akseli on korkeussuunnassa, kun taas Unityssä korkeutta edustaa Y. Kuitenkin Unityssä objektin oletustuontiasetuksissa objektia käännetään 270 astetta X-akselin ympäri, joten objektit ilmestyivät pelimaailmaan toivotulla tavalla. Ei kuitenkaan tuntunut hyvältä ajatukselta, että objektilla täytyisi olla väärä kulma, jotta se näyttäisi oikealta. Ongelmaa yritettiin ratkaista Benjamin Schaafin FixBlenderImportRotation-skriptillä (Schaaf 2014), mutta tällä saatiin 270 asteen käännös poistettua vain objektin tuontiasetuksista. Objekti ilmestyi nyt pelimaailmaan siten, että sen etusivu osoittikin alaspäin. Skripti pidettiin käytössä, koska -270 asteen kääntämistä ei jostain syystä pystytä poistamaan Unityn valikoista käsin, mutta vielä sen lisäksi tarvittiin muutos objektien geometrioihin ennen Blenderistä poisvientiä. Kaikkien objektien geometriaa käännettiin niiden pivottien ympäri -90 astetta X-akselilla, eli nyt niiden etusivu osoitti Blenderin koordinaattijärjestelmän mukaan ylös ja alasivu eteen.

Havaittiin myös, että asiat saattavat näyttää huomattavan erilaisilta 3D-mallinnusohjelmassa ja pelissä. Koska 3D-mallinnusohjelmassa on vapaa kamera, mittasuhteiden arviointi on toisinaan hankalaa. Pelimaailman katseluun käytettiin Unityn omaa ensimmäisen persoonan ohjauspakettia, sillä sen käyttöönotto kävi nopeasti ja helposti. Kun rakennusta tarkasteltiin nyt ensimmäisessä persoonassa (kameran korkeudeksi asetettiin noin 170cm), havaittiin, että jotkin alueet olivat hieman ahtaita. Esiintymislava oli vähän turhan pieni, ja etuovelta katsottuna rakennuksen oikealla puolella sijaitsevan ruokailutilan pöytien väliin jäävä käytävä epämiellyttävän ahdas. Lavan suurentamiseen ei löytynyt sopivaa, nopeaa vaihtoehtoa, mutta käytävää saatiin helposti leveämmäksi kaventamalla sitä reunustavia pöytiä.

Myös objektit, jotka oli toteutettu luvussa 4.2.2. kuvaillulla kärkien kahdentumista vähentävällä, saumattomalla UV-käärimismenetelmällä, tuottivat päänvaivaa. Halvaksi hyllyntäytteeksi oli tehty pullo, jonka kärkien määräksi Blender ilmoitti 70. Korkin päällinen ja pullon

pohja olivat omina UV-saarekkeinaan (ja korkin huippu varjostettiin erillään muusta objektista), mikä tarkoitti, että kärkien olisi kuulunut kahdentua vain näillä saumoilla. Tämä tarkoittaisi 10-sivuisessa pullossa siis 20 kärkeä lisää, eli yhteensä 90 kärkeä. Kuitenkin Unityn mukaan pullossa olikin 159 kärkeä! Kun objektin saumattomuus oli turhaan tarkistettu Blenderissä, tutkittiin jälleen Unityn fbx-tuontiasetuksia. Kokeilemalla eri asetuksia selvisi ongelman aiheuttajaksi se, että oletusarvoisesti Unity laskee itse tuoduille objekteille uudet tangentit. Asetus muutettiin, ja kaikki objektit vietiin Blenderistä fbx-tiedostoiksi jälleen, tällä kertaa tangentteineen (aiemmat fbx-tiedostot oli itse asiassa tallennettu edellisellä Blenderin versiolla, jonka fbx-vientitoiminto ei edes pystynyt tallentamaan objektien tangentteja!). Nyt kärkien lopullinen määrä vastasi arvioita.

### 4.2.6 Peliympäristön rakentaminen

Objekteista koostettiin Unityssä pieni alue, jonka keskelle rakennus sijoitettiin. Alustaksi laitettiin tasainen maa. Se tehtiin käyttäen Unityn maastotyökalua, jotta sen pintaan voitaisiin suoraan maalata tekstuureita. Maaston pintaan maalatut tekstuurit vaihtuvat toiseen pehmeästi häipyen. Tekstuureita tehtiin neljä: perusasfaltti, halkeillut asfaltti, ruoho sekä hiekka, jonka avulla asfaltti saatiin vaihdettua siistimmin ruohoon.

Pientä liikettä muuten staattiseen ympäristöön saatiin lisättyä tuulettimien avulla. Niiden lavat tallennettiin erilliseen objektiin muusta laitteesta, ja lapaobjekteille tehtiin yksinkertainen skripti, joka pyöritti niitä tasaisesti nopeudella, joka voitiin kätevästi määrittää objektikohtaisesti niiden asetuksista käsin.

Rakennus valaistiin suhteellisen hämärästi. Kuten luvussa 2.6. kuvailtiin, valojen väri määriteltiin lisäämällä lamppujen varjostimien kellertävään vaaleanharmaaseen hieman ympäristön punertavaa sävyä valonsäteiden kimpoilun väärentämiseksi reaaliaikaisissa valonlähteissä. Valosommitteluun pyrittiin tuomaan lisää mielenkiintoisuutta asettelemalla sinne tänne värikkäitä valonlähteitä, esimerkiksi levyautomaatti valaisee lähialuettaan kellertävällä valolla ja katossa on paljon pieniä koristevaloja. Rakennuksen perällä olevaa pientä huonetta, jossa käymälöiden sisäänkäynnit ja varauloskäynti sijaitsevat, eristettiin entisestään omaksi tilakseen muusta rakennuksesta jättämällä se pimeäksi. Sitä valaisee ainoastaan takaoven yläpuolella vihreänä loistava uloskäyntikyltti. Toinen valaistuksen puolesta omanlaisensa alue rakennuksen sisällä on esiintymislavan ympäristö. Sitä on valaistu pääasiallisesti koristevaloilla, ja sen

takana on punainen verho, jonka heijastamaa valoa mallinnettiin asettamalla keskelle esiintymislavaa himmeä, punainen, jokaiseen suuntaan tasaisesti valaiseva valonlähde. Tällaista suunnatonta valoa kutsutaan Unityssä pistevaloksi.

Rakennuksen ulkopuoli valaistiin koko pelialueen laajuisesti yhteen suuntaan valaisevalla niin sanotulla suuntavalolla. Sen tarkoitus oli mallintaa auringonvaloa, joten sen väriksi asetettiin lämmin, hieman kellertävä valkoinen. Käyttöön otettiin myös yksi Unityn mukana tulevista taivaslaatikoista (skybox). Se on kaikkien peliobjektien taakse piirrettävä, kuudesta neliönmuotoisesta tekstuurista koostuva kuutio, jonka tekstuureiden kautta voidaan simuloida taivasta. Laatikkoon valittiin kauniin, aurinkoisen keskipäivän taivastekstuurit, jotta ravintola saataisiin tuntumaan ympäristöään hämärämmältä. Etuseinän ikkunoita peittävät sälekaihtimet estävät päivänvalon pääsyä sisälle.

Pelialueelle lisättiin vielä hyvin vähäinen määrä sinertävää ambient-valoa. Se on tasaista, koko pelialueella vaikuttavaa valoa, joka ei varjosta objekteja. Tällä saatiin hillittyä ulkopuolen varjoja hieman, ikään kuin niihin heijastuisi epäsuoraa valoa ympäröiviltä pinnoilta. Sisätilan varjoihin se toi mukavaa pientä kontrastia siellä vallitsevalle punaruskealle väriteemalle.

## 4.3 Peliympäristön optimointi

Projektissa ei ollut käytettävissä pelihahmoja eikä peliympäristön itsensä lisäksi mitään muitakaan suorituskykyyn vaikuttavia tekijöitä, kuten äänet tai tekoäly, joten mitään kokonaista peliprojektia täysin hyödyttävää suorituskykyanalyysiä ja sen mukaisia muutoksia olisi näiden kokeiden perusteella vaikeaa tehdä. Tuotettu peliympäristö oli kuitenkin riittävän monimutkainen, jotta sen avulla pystyttiin kokeilemaan eri optimointimenetelmiä käytännössä ja havaitsemaan niiden vaikutuksia eri resurssien kulutuksessa. Käyttöön otettiin Unity Pron ilmainen 30 päivän kokeiluversio, sillä Unity Prossa on ominaisuuksia, joita sen ilmaisversiossa ei ole, kuten piilotettujen pintojen poisto ja staattisten objektien eritteleminen.

Ensimmäiseksi kiinnitettiin huomiota materiaaleihin. Alkuperäinen toimeksianto määräsi, että spekulaarisuuskarttoja ei tarvita, sillä pelissä valon käyttäytyminen pinnoilla määriteltäisiin itse tehtyjen varjostimien kautta. Unityssä varjostin määrittää, mitä ominaisuuksia sitä käyttävällä materiaalilla voi olla. Esimerkiksi pikselikohtaista läpinäkyvyyttä käyttävä materiaali tarvitsee kyseistä ominaisuutta tukevan varjostimen. Projektin lopuksi tehdyissä kokeissa

käytettiin pelkästään Unityn mukana tulleita varjostimia, joten yhdelle objektille saatettiin tarpeen mukaan antaa useita erillisiä materiaaleja. Rakennuksen ulko- ja sisäpintojen tekstuurit eivät niinkään kaivanneet useita materiaaleja, mutta propeille tehtiin erikseen metalli-, lasi- ja perusmateriaali. Nämä materiaalit ovat muistinkäytöllisesti ja ainakin teoriassa pinnan täytön nopeuden puolesta hieman halvempia kuin yhteinen materiaali, jolla olisi myös spekulaarisuuskartta. Käytännössä laskentatehon säästön varmistaminen olisi kuitenkin ollut vaikeaa. Myös proppien itsevalaisevat pinnat saivat oman materiaalinsa. Kaikki materiaalit käyttivät yksibittistä alfaa (pikseli on joko täysin näkyvä tai läpinäkyvä), paitsi juomalasien materiaalit. Nämä lasit suunniteltiin sellaisiksi, että niissä olisi kaksi läpinäkyvyyden astetta. Suurin osa astiasta olisi läpinäkyvämpää näistä kahdesta, ja vähemmän läpinäkyvää osaa käytettäisiin lasin reunoilla ja pohjassa. Tämä olisi teoriassa toteutettavissa yksibittisellä alfalla, sillä läpinäkyvyyden tasoja tarvitaan vain kaksi, mutta varjostimeen se vaatisi muutoksia eikä sellaisen opetteluun ollut aikaa. Toteutettu ratkaisu kaikessa epäoptimaalisuudessaankin (proppien alfan kaikki kahdeksan pikselikohtaista bittiä täytyi pitää tekstuurimuistissa ainoastaan muutaman juomalasin takia) päätettiin säilyttää, koska luodun pelialueen tarkoitus oli kuitenkin pääasiallisesti tuoda esille luotujen objektien ja tekstuurien visuaalista laatua.

Useamman proppimateriaalin tekeminen ei kuitenkaan ollut täydellisen ongelmatonta: objektin jokainen erillinen materiaali lisää yhden piirtokutsun, ja erilliset materiaalit haittaavat erittelyä, vaikka ne käyttäisivätkin yhteistä tekstuuriatlasta. Unityllä on kaksi keinoa jakaa piirtokutsuja eriin: staattinen ja dynaaminen. Dynaaminen erittely tapahtuu automaattisesti tiettyjen ehtojen toteutuessa. Ensinnäkin kaikilla samassa erässä piirrettäväksi lähetettävillä pinnoilla täytyy olla sama materiaali-instanssi (Unity Technologies 2014 b). Koska tässä projektissa yhdellä objektilla saattaa olla useita materiaaleja, yksittäinenkin objekti saattaa vaatia useamman piirtokutsun. Periaatteessa pintojen täyttönopeutta ja tekstuurimuistia on siis ostettu suorittimen työmäärän kustannuksella. Toinen dynaamisen erittelemisen vaatimus on, että objektilla saa olla korkeintaan 900 kärkiominaisuutta (Unity Technologies 2014 b). Tässä projektissa kärjillä oli sijainti, yksi normaali ja yksi UV-koordinaatti, joten käytännössä korkeintaan 300-kärkiset objektit voitiin eritellä dynaamisesti. Tämän kriteerin täyttivät lähinnä kaikkein pienimmät objektit, kuten pullot ja astiat, joskin yksittäistä pöytää koristavilla, alle 300-kärkisillä objekteilla saattoi olla yhteensä neljäkin eri materiaalia (yleinen proppimateriaali, kiiltävä metalli haarukoissa ja veitsissä, kiiltävä lasi suolasirottimessa ja pulloissa sekä juomalaseissa käytetty läpinäkyvä lasi), mikä tarkoitti myös neljää erillistä piirtokutsua. Objektien kärkimääriä ja mahdollista optimoinnin tarvetta arvioitiin luomalla Unityssä uusi, valaisema-

ton pelialue, jonne tutkittavat objektit sijoiteltiin sen verran harvaan, että ne saatiin tuotua näytölle yksi kerrallaan. Tämä oli tarpeen, sillä Unityssä valaistus vaikuttaa lopulliseen kärkimäärään. Näin yksittäisten objektien kärkimääriä ja erilaisten objektijoukkojen aiheuttamia piirtokutsuja voitiin tutkia helposti Unityn statistiikka-ikkunan (kuvio 13) avulla. Kyseinen ikkuna kertoo numeroina muun muassa sen, paljonko tekstuureita, kolmioita ja kärkiä on minäkin hetkenä näkyvissä ja paljonko resursseja niihin tarvitaan.


Kuvio 13. Optimointia helpottava statistiikkaikkuna. Dynaamisen erittelyn ansiosta näille seitsemälle taululle riittää yhteensä yksi piirtokutsu. Yhdessä taulut koostuvat 126 kolmiosta, joissa on yhteensä 252 kärkeä, mihin sisältyvät myös kaikki UV-saumojen ja varjostusryhmien aiheuttamat jakautumiset. Tekstuureita taulujen käytössä on yhteensä kaksi: sisäproppimateriaalin diffuusio- ja normaalikartta, jotka vievät yhteensä 10,6 megatavua muistia.

Unity Pro tarjoaa myös toisen keinon piirtokutsujen yhdistelemiseen: staattisten objektien erittely. Toisin kuin dynaamista erittelyä, sitä voidaan käyttää objekteissa niiden kärkien lukumääristä riippumatta, tosin staattinen erittely vaatii objektilta täydellistä liikkumattomuutta pelin aikana. Koska se yhdistää staattisten objektien geometriat isommiksi kokonaisuuksiksi, sen käyttö lisää muistin kulutusta hieman (Unity Technologies 2014 b). Sitä hyödynnettiinkin vain suuremmissa objekteissa, jotka eivät varmasti liikkuisi, kuten rakennuksen geometriassa, baaritiskissä ja pöydissä.

Piirtokutsujen määrää saatiin jossain määrin leikattua myös piilotettujen pintojen poiston avulla, mutta hyöty oli tässä tapauksessa kuitenkin aika pieni, sillä suurin osa objekteista ja vaativin valaistus olivat molemmat rakennuksen sisällä, missä ei ollut mistään kohti katsottuna suuria objekteja piilottamassa niitä. Käytännössä sillä saatiin lähinnä eliminoitua rakennuksen sisällä olevien näkymättömien objektien piirtäminen silloin, kun rakennusta katsotaan ulkoa. Projektin geometria ja mittakaava ei kuitenkaan oikein soveltunut piilotettujen pinto-

jen poiston kunnolliseen hyödyntämiseen, ja toiminto jopa aiheutti eri asetuksillaankin vääristä paikoista katoavia pintoja, joten se poistettiin käytöstä.

Kärkien ja piirtokutsujen määrä moninkertaistui nopeasti, kun peliympäristöön lisättiin valoja. Kuitenkin tehdyn kokoiseen rakennukseen tarvittiin useita valonlähteitä, jotta se saataisiin valaistua tasaisesti, ja halutun tunnelman aikaansaamiseksi näiden lisäksi vielä useita koristevaloja. Liikkumattomien objektien muuttumaton valaistus voidaan tallentaa valokarttaan, mutta nekään eivät ole ilmaisia. Kuten kaikki tekstuurit, myös valokartat käyttävät arvokasta muistia. Lisäksi, kuten jo luvussa 3.2.7. kerrottiin, tehokkaaseen valokarttojen käyttöön tarvitaan objektin jokaiselle pinnalle uniikit UV-koordinaatit. Unity pystyy itse luomaan ne pyydettäessä nopeasti, mutta tämä saattaa joissain tilanteissa lisätä kärkien määriä, ja joka tapauksessa se on objektin jokaiselle kärjelle yksi ominaisuus lisää, mikä saattaisi nostaa pienen objektin kärkien ominaisuuksien määrän liian korkeaksi dynaamiselle erittelylle.

Valokarttoja päädyttiin hyödyntämään vain niissä staattisissa objekteissa, joissa niiden tuoma tarkka valaistus olisi selvästi näkyvissä. Tällaisia olivat rakennuksen sisäpinnat, baaritiski hyllyineen ja suurimmat pöydät. Koska Unityssä valokartan maksimikoko on 1024*1024, rakennuksen sisäpuoli jouduttiin jakamaan neljäksi erilliseksi objektiksi, jotta valokarttojen resoluutio saatiin riittäväksi. Baaritiskin ja pöytien valokartat saivat yhden, yhteisen atlaksen. Valokartoilla saatiin sattumalta kunnollista lähialuevarjostusta vaille jääneisiin rakennuksen sisäpintoihinkin yksityiskohtainen varjostus. Useimmissa valokartoitetuissa objekteissa käytettiin Unityn automaattista UV-käärijää. Osalle pöydistä jouduttiin kuitenkin tekemään valokarttojen UV-kanava käsin, sillä automaattisesti tehdyissä tekstuurikoordinaateissa ilmeni päällekkäisyyksiä, mistä seurasi näkyviä virheitä varjostuksessa.

Valojen optimointiin on Unityssä myös toinen väline: objektit voidaan jakaa virtuaalisille kerroksille (layer), ja jokaisen valon asetuksissa on culling mask -kohta, josta voidaan määrätä, minkä kerroksen objekteihin mikäkin valo vaikuttaa. Laskentatehon käyttöä saatiin leikattua siirtämällä valokartoitetut objektit omaan kerrokseensa reaaliaikaisten valojen ulottumattomiin. Lisäksi tällä saatiin poistettua virheitä, kuten seinien läpi kulkevia valonsäteitä.

Osalle objekteista tehtiin lod-versioita. Lodien avulla saatiin kärkien määrän lisäksi vähennettyä myös piirtokutsuja. Esimerkiksi maustesetin täysiyksityiskohtaisessa versiossa on kolme erillistä materiaalia: muovisten ja puisten pintojen yhteinen perusmateriaali sekä lasi ja metalli. Setin lodeissa sen sijaan kaikki pinnat jakavat yhteisen materiaalin, joka on sama kuin mitä

useimmat propit käyttävät muutenkin. Lodit tehtiin Blenderissä poistamalla alkuperäisestä objektista tehdystä kopiosta mahdollisimman monta reunasilmukkaa kuitenkin siten, että objektin tärkeimmät muodot säilyisivät. Prosessi oli nopea, koska reunasilmukat pyrittiin valitsemaan sellaisista paikoista, joissa niiden poisto ei sotkisi objektin UV-koordinaatteja. Samalla lodit saatiin automaattisesti käyttämään oikeaa kohtaa tekstuurista.

Unity Prossa lodien käyttö on tehty helpoksi: objektille voidaan lisätä LODgroup-komponentti, jossa voidaan määrittää objektin eri yksityiskohtatasoihin käytetyt 3D-mallit vetämällä ja pudottamalla. Lisäksi LODgroupissa on liukusäädin, josta voidaan määrittää, kuinka kaukana kameran on oltava kunkin version käyttöönottamiseksi.

Lopuksi kokonaisuus käännettiin valmiiksi Windows-sovellukseksi kuvakaappausten ottamista varten. Sovelluksen kääntäminen onnistuu Unityssä nappia painamalla.

Aikaa ei riittänyt pelialueen huolelliseen optimointiin, mutta lopputuloksena saatiin kuitenkin sujuvasti toimiva, asiallisen näköinen peliympäristö. Samalla opittiin perusteet erilaisten optimointimenetelmien toimintaperiaatteista ja käytöstä.

# 5 POHDINTA

Yksityiskohtaisen, kolmiulotteisen peliympäristön luominen voi olla todella työlästä varsinkin aloittelijalle, mutta se on myös hyvin palkitsevaa. On hienoa nähdä itse tehdyt 3D-mallit ja tekstuurit kunnolla valaistuna pelimoottorissa.

Projekti päätyi venymään yli vuoden mittaiseksi, mikä johtui pääasiassa kokemattomuudesta. Toimintatavat olivat projektin alussa tehottomia ja epäröintiä oli paljon. Alussa yksittäisten objektien kohdalla lopputuloksessa oli monesti parantamisen varaa. Tekstuuritilan käyttö oli tehotonta ja tekstuurien terävyys riittämätöntä. Ennen tietotaidon karttumista oli vaikeaa arvioida, paljonko geometriaa ja tekstuuritilaa eri objekteihin olisi varaa käyttää. Useita asioita päädyttiinkin tekemään monta kertaa kokonaan uusiksi, erityisesti ajan rajallisuuden aiheuttamien paineiden helpotettua peliprojektin peruuntumisen jälkeen.

Projektin ensimmäinen ongelma oli, että suunnitelma tehtiin ainoastaan yksinkertaisena 3D-rakennuksena. Näin ollen kun lopullista työtä alettiin toteuttaa, ei ollut minkäänlaista ajatusta esimerkiksi siitä, mitä värejä käytettäisiin ja missä. Vastaavanlaisen kokonaisuuden suunnittelusta ei myöskään ollut aiempaa kokemusta. Lopulliset värit haettiin pitkälti kokeilujen kautta tekstuureita tehtäessä ohjenuorana ainoastaan referenssikuvien värimaailmat. Tämä huomioonottaen lopputulos näyttää yllättävän asialliselta.

Modulaarista työskentelytapaa olisi pitänyt käyttää alusta lähtien, varsinkin rakennuksen geometriassa. Jos pohjan suunnittelussa olisi hyödynnetty ruudukkoa, oltaisiin voitu tehdä joukko toisiinsa kiinni sopivia rakennuspaloja. Tällaisia paloja voisivat olla esimerkiksi toistettava, koristeellinen kattopala, parimetrinen pätkä seinää katto- ja lattialistoineen sekä seinään vaihtelua tuova koristepylväs. Nämä palat olisi voitu myös UV-kääriä ja ehkä jopa teksturoida ennen rakennuksen kasaamista. Nykyisellään rakennuksen monimutkaisten sisäpintojen UV-kääriminen oli äärimmäisen työlästä. Toisaalta modulaarisuus ei olisi kuitenkaan välttämättä auttanut siinä vaiheessa, kun tekstuurikoordinaatteja jouduttiin järjestelemään uusiksi ensimmäisen lähialuevarjostuksen paistoyrityksen epäonnistuttua. Ylipäätään tällainen modulaarinen toteutus olisi kunnolla onnistuakseen vaatinut enemmän käytännön kokemusta kuin mitä projektin alkuun mennessä oli ehtinyt kertyä.

Lisäviivästystä aiheutui myös GIMP-kuvanmuokkausohjelman kokeilusta. Ohjelmaa yritettiin käyttää kaksi viikkoa teksturointivaiheen alussa, mutta vieras käyttöliittymä oli vaikea

omaksuttava Photoshopia jo useamman vuoden ajan käyttäneelle. GIMPistä luovuttiinkin kokonaan ja teksturointityökaluksi otettiin kuukausimaksullinen Photoshop CC.

Suurin yksittäinen viivyttäjä oli kuitenkin paistetun lähialuevarjostuksen laadun ja tekstuurien terävyyden välillä tasapainottelu. Aiemman käytännön kokemuksen puutteessa oli vaikeaa arvioida, minkälaiset alueet tarvitsisivat ehdottomasti oman varjostuksensa eli oman alueensa tekstuurista ja kuinka paljon tekstuuritilasta olisi varaa antaa sellaisille osille, jotka diffuusio- ja normaalikarttojen puolesta voisivat olla kaikki pakattuina samaan kohtaan. Lopputulosta haettaessa tekstuurikoordinaatit päädyttiinkin tekemään alusta yhteensä kolme kertaa.

Teoriapuolella uutta asiaa olivat lähinnä valaistus ja pelimoottorissa tehtävä ympäristön optimointi. Varsinkin jälkimmäiseen olisi ollut aiheellista perehtyä tarkemminkin, mutta aikaa oli kulunut jo liikaa projektin muihin vaiheisiin.

Kaiken kaikkiaan projektissa oli liian monta uutta asiaa aloittelijan hallittavaksi. Lopputuloksesta saatiin kuitenkin viimein asiallinen kokonaisuus, sisältöä portfolioon sekä suunnattomasti arvokasta kokemusta. Projektin parissa työskenneltäessä opittiin huomattavasti uutta sekä omaksuttiin entistä tehokkaampia työskentelytapoja. Oli myös mielenkiintoista huomata, kuinka peliympäristön optimointi on pitkälti tasapainottelua eri resurssien käytön välillä.

# LÄHTEET

Bjorke, K. 2005. Let's Get Small: Understanding MIP Mapping. Nvidia. Saatavilla PDF-muodossa: ftp://download.nvidia.com/developer/presentations/2005/Misc/Lets_Get_Small-2005.pdf (luettu 9.10.2014).

Blinn, J. 1978. Simulation of Wrinkled Surfaces. SIGGRAPH 1978. Saatavilla PDF-muodossa: http://research.microsoft.com/pubs/73939/p286-blinn.pdf (luettu 29.4.2014).

Bushnaief, J. 2013. Occlusion Culling in Unity 4.3: The Basics. Julkaissut Kristyna Paskova Unity Blogissa. http://blogs.unity3d.com/2013/12/02/occlusion-culling-in-unity-4-3-the-basics (luettu 5.10.2014).

Chadwick, E. 2011. Mip Map. Polycount wiki. http://wiki.polycount.com/MipMap (luettu 12.5.2011).

Chadwick, E. 2012. Polygon Count. Polycount wiki. http://wiki.polycount.com/PolygonCount (Luettu 12.5.2014).

Chadwick, E. 2013. Edge Padding. Polycount wiki. http://wiki.polycount.com/EdgePadding (Luettu 12.5.2014).

Chadwick, E., Klausen, J. & Phoenix, B. 2014. Light Map. Polycount wiki. http://wiki.polycount.com/wiki/Light_map (Luettu 4.10.2014).

Chadwick, E. & Montero, C. 2011. VertexColor. Polycount wiki. http://wiki.polycount.com/VertexColor (Luettu 16.5.2014).

Chadwick, E. & Montero, C. 2013. Transparency Map. Polycount wiki. http://wiki.polycount.com/TransparencyMap (Luettu 15.5.2014).

Crytek 2013. Tangent Space Normal Mapping. Cryengine SDK Documentation. http://docs.cryengine.com/display/SDKDOC4/Tangent+Space+Normal+Mapping (Luettu 9.9.2014).

Crytek 2014. Asset Creation Guidelines. Cryengine SDK documentation. http://docs.cryengine.com/display/SDKDOC3/Asset+Guidelines (Luettu 29.4.2014)

Franson, D. 2004. The Dark Side of Game Texturing. Boston, MA, USA: Course Technology / Cengage Learning.

FrostSoft 2012. SHADING INTERPOLATION (VNT documentation/ Superspecular soft edges). http://frostsoft.blogspot.fi/2012/01/shading-interpolation-vertex-normals.html (Luettu 29.4.2014).

Galanakis, R. 2008. Beveling. Tech Artists Wiki. http://tech-artists.org/wiki/Beveling (Luettu 29.4.2014).

Galuzin, A. 2010. Silhouette Design Game Environments. World of Level Design. http://www.worldofleveldesign.com/categories/game_environments_design/silhouette-design-game-environments.php (Luettu 28.9.2014).

Gard, T. 2010. Action Adventure Level Design: Kung Fu Zombie Killer. Gamasutra. http://www.gamasutra.com/view/feature/132801/action_adventure_level_design_.php (Luettu 25.4.2014).

Geforce 2014. Enabling Ambient Occlusion in Games. http://www.geforce.com/whats-new/guides/ambient-occlusion#1 (Luettu 8.9.2014).

Giambruno, M. 2003. 3D Modeling Basics. (Ote teoksesta 3D Graphics & Animation, 2nd Edition. New Riders 2002.) www.peachpit.com/articles/article.aspx?p=30594 (Luettu 28.4.2014).

Ivanov, I. 2006. Practical Texture Atlases. Gamasutra. http://www.gamasutra.com/view/feature/130940/practical_texture_atlases.php (Luettu 9.9.2014).

Jones, S. 2011. Investigation into Modular Design Within Computer Games. Saatavilla PDF-muodossa: http://www.scottjonescg.co.uk/FYPResearch/Investigation_into_modular_design_within_computer_games_v1.0.pdf (luettu 20.9.2014).

Jong, S. 2008. The Hows and Whys of Level Design - Second Edition. Hourences.com - Unreal Engine and Level Design Tutorials. http://www.hourences.com (Luettu 11.10.2014).

KatsBits 2014. What Are Smooth Grounps & Mesh Smoothing And Forcing It In Blender http://www.katsbits.com/tutorials/idtech/what-are-smooth-groups-and-mesh-smoothing-on-models.php (Luettu 18.10.2014).

Kaveh, B. 2010. A Fresh Look at the Concept of Immersion. http://www.gamedesignideas.com/video-games/a-fresh-look-at-the-concept-of-immersion.html (Luettu 18.10.2014).

Kivelä, S. 2000. M niinkuin matematiikka – Lukiotason matematiikan tietosanakirja. http://matta.hut.fi/matta2/isom/html/tangnorm3.html (Luettu 5.7.2013).

Liman 2013. Understanding Normal Maps. http://www.liman3d.com/tutorial_normalmaps.html (Luettu 11.10.2014).

Mathis, B. 2012. Color. Julkaistu teoksessa Ryan Hawkins's Vertex 1, 156-162. Saatavilla PDF-muodossa: http://artbypapercut.com (Luettu 8.9.2014).

Pardew, L. & Whittington, D. 2005. Beginning Game Art in 3DS Max 8. Boston, MA, USA: Course Technology / Cengage Learning.

Perry, L. 2002. Modular Level Design. Epic Games. Saatavilla PDF-muodossa: http://udn.epicgames.com/Three/rsrc/Three/ModularLevelDesign/ModularLevelDesign.pdf (Luettu 19.10.2014).

Provost, G. 2003 1. Beautiful, Yet Friendly Part 1: Stop Hitting the Bottleneck. http://www.ericchadwick.com/examples/provost/byf1.html (Luettu 19.5.2014).

Provost, G. 2003 2. Beautiful, Yet Friendly Part 2: Maximizing Efficiency. http://www.ericchadwick.com/examples/provost/byf2.html (Luettu 20.5.2014).

Rouse III, R. 2010. Environmental Narrative - Your World Is Your Story. Game Developers Conference 2010. PowerPoint-esitys. http://www.paranoidproductions.com/miscwritings/EnvironmentalNarrative.ppt (Luettu 2.5.2014).

Schaaf, B. 2014. FixBlenderImportRotation. Unify Community Wiki. http://wiki.unity3d.com/index.php/FixBlenderImportRotation (Luettu 30.9.2014).

Slick, J. 2013 3D Defined – What is 3D? http://3d.about.com/od/3d-101-The-Basics/a/3d-Defined-What-Is-3d.htm (Luettu 18.10.2014).

Solarski, C. 2013. The Aesthetics of Game Art and Design. Gamasutra. http://www.gamasutra.com/view/feature/185676/the_aesthetics_of_game_art_and_.php (Luettu 9.5.2014).

Summers, D. 2004. Texturing: Concepts and Techniques. Hingham, MA, USA: Charles River Media / Cengage Learning.

Totten, C. 2011. Designing Better Levels Through Human Survival Instincts. Gamasutra. http://www.gamasutra.com/view/feature/134779/designing_better_levels_through_.php? (Luettu 21.5.2014)

Unity Technologies 2014 a. Occlusion Culling (Pro Only). Unity Documentation. http://docs.unity3d.com/Manual/OcclusionCulling.html (Luettu 5.10.2014).

Unity Technologies 2014 b. Draw Call Batching. Unity Documentation. http://docs.unity3d.com/Manual/DrawCallBatching.html (Luettu 7.10.2014).

van Waveren, J.M.P. & Castaño, I. 2008. Real-Time Normal Map DXT Compression. Nvidia. Saatavilla PDF-muodossa: http://developer.download.nvidia.com/whitepapers/2008/real-time-normal-map-dxt-compression.pdf (Luettu 30.4.2014).

Wagner, M. 2004. Generating Vertex Normals. Saatavilla PDF-muodossa: http://www.emeyex.com/site/tuts/VertexNormals.pdf (Luettu 9.10.2014).

Ward, A. 2008. Game Character Development: Digital Sculpting for the Realtime Artist. Boston, MA, USA: Course Technology / Cengage Learning.

Worch, M. 2010. Level Design Workshop. Game Developers Conference 2010. PowerPoint-esitys. http://www.worch.com/files/gdc/GDC_LD_workshop_LDIntro_web.zip (Luettu 12.10.2014).

LIITE: KUVAKAAPPAUKSIA VALMIISTA PELIYMPÄRISTÖSTÄ

Kuvakaappaukset on otettu Unitystä valmiiksi sovellukseksi käännetystä ohjelmasta. Tässä testiohjelmassa pelialuetta pystyttiin tarkastelemaan ensimmäisessä persoonassa.



trial version


trial version


trial version


REDSTAR
trial version


trial version


trial version