



# Shaderien teko ja hyödyntäminen Unityssä

## Case Kyber Knights

Esa Nord

OPINNÄYTETYÖ  
Marraskuu 2019

Tietojenkäsittely  
Pelituotanto

## TIIVISTELMÄ

Tampereen ammattikorkeakoulu  
Tietojenkäsittely  
Pelituotanto

NORD, ESA:  
Shaderien teko ja hyödyntäminen Unityssä  
Case Kyber Knights

Opinnäytetyö 25 sivua  
Marraskuu 2019

---

Tämän opinnäytetyön toimeksiantaja on Portaalin Pojat Oy, joka on Tampereella toimiva virtuaalielämysliike. Opinnäytetyössä suunniteltiin ja toteutettiin toimeksiantajalle kehitettyyn Kyber Knights -virtuaalitodellisuusmoninpeliin tarvittavat shader-ohjelmat, joilla voitiin toteuttaa pelaajan ja hahmon ja kilven tarvitsemat erikoiseffektit.

Shader-ohjelmien kehityksessä haettiin verkosta shaderiesimerkkejä, jotka toteuttavat halutun tai vastaavanlaisen efektin. Esimerkkien analysoinnilla varauduttiin mahdollisiin ongelmiin, joita efektien toteutuksessa oli virtuaalitodellisuuden yhteydessä. Ennen suunniteltujen efektien toteuttamista, voitiin hakea vaihtoehtoisia shaderiesimerkkejä korvaamaan virtuaalitodellisuuden kanssa toimimattomat efektit. Lopulliset shaderi-ohjelmat toteutettiin pitkälti suunnitelmien mukaan muutamien poikkeuksin. Shader-ohjelmien muuttujien arvojen muokkaamisessa toteutettiin Unityn dokumentaation ja haettujen esimerkkien avulla ilman suurempia ongelmia.

Shader-ohjelmien kehityksessä havaittiin, että varsinkin aloittelevana kannattaa hakea useita shaderiesimerkkejä haluttujen efektien toteuttamisen avuksi, selvittää niiden toiminta ja mahdolliset vaatimukset. Pelimoottorin ja 3D-mallinnusohjelman koordinaatiston mahdollinen eroavaisuus kannattaa myös selvittää. Vaikkakin suunnitellut shaderi-ohjelmat efekteineen saatiin toteutettua opinnäytetyön aikana, voi niiden toimintaa jatkokehittää visuaalisteneffektien parantamiseksi. Yhtenä jatkokehitysehdotuksena on lisätä pelaajan liikkumiseen teleportaatioefekti, jota harkittiin yhtenä pelaajan kuolemaefektinä.

## **ABSTRACT**

Tampereen ammattikorkeakoulu  
Tampere University of Applied Sciences  
Degree Programme in Business Information Systems  
Game Development

NORD, ESA:  
Creating and Using Shaders in Unity  
Case Kyber Knights

Bachelor's thesis 25 pages  
November 2019

---

The commissioner of this thesis was Portaalin Pojat Oy, who is a virtual experience business located in Tampere. This thesis was carried out as part of a game project where a virtual reality multiplayer game Kyber Knights was developed for the client. The objective of this thesis was to design and develop shader programs to create desired visual effects for the game. Certain variables in shader programs were also required to be modifiable while the game is running in order to animate visual effects.

Example shader programs and tutorials were searched from internet sources as well as from books on the subject. The way the code operated in the example programs was analyzed. After analyzing the code was tested by adding it into game engine's test scene and running it to test its function in virtual reality. Shader programs, with the required visual effects for the game were finished with some adjustments from the initial design of the effects. Modifying shader program variables while game is running was also carried out with only minor problems, which could have been avoided if the issues had been noticed earlier.

---

Key words: unity, shader, games, virtual reality

## SISÄLLYS

1	JOHDANTO .....	6
2	SHADEREISTÄ .....	7
3	AINEISTON KERÄÄMINEN JA TUTKIMINEN .....	8
3.1	Hahmojen kuolema efektin vaihtoehdot .....	8
3.2	Hahmojen korostamisen vaihtoehdot .....	10
3.3	Kilpi efektin tutkiminen .....	11
4	LOPULLISTEN EFEKTIEN SUUNNITTELU .....	13
4.1	Hahmojen kuolema ja kentälle takaisin siirtyminen .....	13
4.2	Hahmojen korostaminen .....	14
4.3	Kilpien shaderi.....	16
5	LOPULLISTEN SHADERIEN TOTEUTUS .....	17
5.1	Hahmojen pohja- ja kuolemaefekti .....	17
5.2	Hahmojen korostaminen .....	18
5.3	Kilpien efektien toteutus .....	20
6	ARVOJEN MUOKKAAMINEN KOODISSA .....	22
7	POHDINTA .....	24
	LÄHTEET .....	25

**LYHENTEET JA TERMIT**

Blender	3D mallinusohjelman
Cg	C kieleen pohjautuva korkeatason shader ohjelmointikieli
DeltaTime	Kameran piirtämien kuvien välinen aika sekunteina. Kuvaa edellisen ja nykyisen kuvan valmistumisen välistä aikaa.
Fixed4	11 bittinen matala tarkkuuden, 4 numeraalista arvoa sisältävä muuttuja
Shader	Ohjelma, joka kertoo näytönohjaimelle, kuinka kohde piirretään
SteamVR	Sovelluskehitysrajapinta, joka tukee yleisiä virtuaalitodellisuus laitteita
Unity	Pelimoottori
Vector3	Vektori muuttuja, joka sisältää 3 numeraalista arvoa
Vector4	Vektori muuttuja, joka sisältää 4 numeraalista arvoa
VR	Virtuaalitodellisuus

## 1 JOHDANTO

Kyber Knights on VR moninpeli, jossa pelaajat pelaavat lähiverkossa toisiaan vastaan joukkuetaistelussa tai avoimessa taistelussa toisiaan vastaan. Peli kehitettiin asiakastyönä Portaalin Pojat Oy:lle, joka on Tampereella toimiva virtuaalielämysliike ja toimii tämän opinnäytetyön toimeksiantajana.

Tämän opinnäytetyön tavoitteena oli suunnitella ja toteuttaa Kyber Knights pelin erikoisefektejä varten shader-ohjelmat, joiden muuttujien arvoja voidaan koodissa muokata pelin ollessa käynnissä. Peli kehitettiin Unity-pelimootorilla käyttäen SteamVR:ää VR toiminnallisuuden tuomiseksi, jolloin shaderien suunnittelussa otettiin huomioon mahdolliset ongelmat shaderien toiminnassa VR ympäristössä. Tarvittavat erikoisefektit olivat pelaajan kuolemisen ja kentälle siirtymiseen käytettävä efekti, jolloin pelaajan hahmo ei vain yhtäkkiä katoaisi kentältä ja ilmestyisi hetken päästä takaisin. Pelaajan hahmoa myös haluttiin korostaa antamalla sille väritetyt ääriviivat ja antamalla sille korostusefekti, jolla pelaaja näkyisi tiimilleen seinien läpi. Pelaajien kilpiä muokattiin testien jälkeen futuristisiksi energiakilviksi, jolloin niille suunniteltiin sopiva efekti shadereilla. Lisäksi kilville suunniteltiin osumien visualisointi, jolloin pelaajat pystyivät havainnoimaan kilven hyödyn itsepuolustuksessa.

Tarkoituksena opinnäytetyöllä on selvittää, kuinka edellä mainitut efektit voidaan toteuttaa ja kuinka niiden muuttujien arvoja voidaan muokata koodin kautta ohjelman ollessa käynnissä. Lisäksi selvitetään kuinka yhdistää useampi efekti toimimaan yhdessä shaderissä ja mitä tarvitsee ottaa huomioon shaderien kehityksessä, kun niitä käyttävä peli kehitetään VR alustalle.

## 2 SHADEREISTÄ

Shaderit ovat pieniä ohjelmia, mitkä sisältävät matemaattisia laskuja ja algoritmeja, joilla lasketaan jokaisen pikselin väri perustuen valoon ja materiaalin ase-  
tuksiin (Unity Documentation 2019a). Projektissa käytettiin surface- ja vertex and  
fragment-shadereitä eri efektien luomiseen.

Surface-shaderistä kerrotaan Unityn dokumentaatiossa (2019b), että se on paras  
vaihtoehto, jos valojen ja varjojen tarvitsee vaikuttaa shaderiin. Surface shade-  
reitä ei pidä käyttää, jos shaderi ei ole missään tekemissä valojen kanssa. Jälki-  
prosessoitujen efektien tai useiden erikoisefekti shaderien kassa surface-shade-  
rit ovat huonovaihtoehto, koska ne tekevät paljon laskelmia valaistukseen ilman  
tarvetta (Unity Documentation 2019b). Unityn editorissa uusi pohja surface-sha-  
derille luodaan luomalla uusi standard surface shader, johon Unity automaatti-  
sesti kirjoittaa shaderin perustoiminnot. Surface-shaderissä pitää surface-toimin-  
nossa määrittää SurfaceOutput rakenne, jonka avulla Unityn Surface Shader  
kääntäjä osaa luoda toimivan shaderin.

Vertex and fragment-shadereitä tarvitaan, jos shaderin ei tarvitse toimia valais-  
tuksen kanssa tai jos kyseessä on erittäin eksoottisia efektejä, joita surface-sha-  
der ei tue (Unity documentation 2019b). Vertex-toiminnossa voidaan shaderiä  
käytävän mallin kärkipisteiden sijaintia muokata, jolloin voidaan mallin muoto  
muuttaa ohjelmasta käsin. Vertex-toimintoa voidaan käyttää myös surface-sha-  
derissä, mutta vertex and fragment-shaderissä se on vaadittu toiminto. Frag-  
ment-toiminnossa lasketaan ja asetetaan pikselikohtaiset arvot, jotka lasketaan  
erikseen jokaiselle pikselille. Dokumentaatiossa (2019c) painotetaan fragment-  
shaderien optimoinnin tärkeyttä pelin suorituskyvyn optimoinnissa. Vertex and  
fragment-shaderiin pystyy lisäämään tuen valaistukselle, mutta tämä tuki pitää  
erikseen lisätä shaderin toimintaan. Uusi vertex and fragment-shader luodaan  
Unityn editorissa luomalla uusi unlit shader-tiedosto.

### 3 AINEISTON KERÄÄMINEN JA TUTKIMINEN

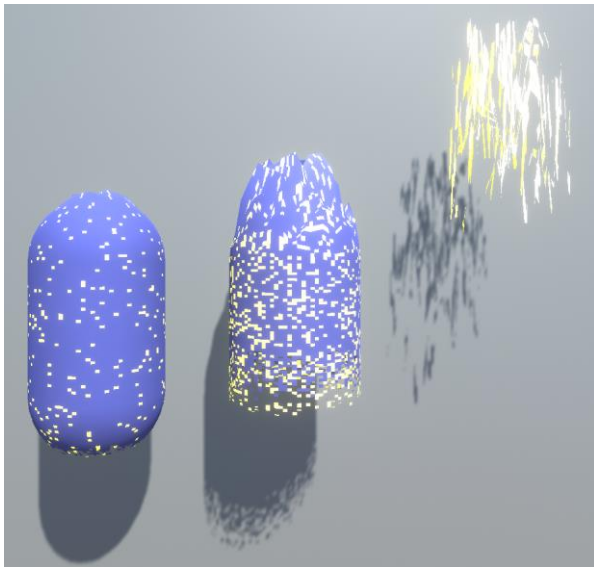
Tarvittava aineisto haluttujen shaderiefektien suunnittelemiseen ja kehittämiseen kerättiin pitkälti verkosta hakemalla shaderiesimerkkejä halutuista efekteistä. Lisäksi haettiin aineistoa perusshaderien toiminnoista, kuten emissio- ja metallikarttojen käytöstä shaderissä. Aineiston lisäksi hyödynnettiin Unityn shader -ohjelmoinnin dokumentaatiota oikean syntaksikäytön varmistamiseksi. Löydettyjä shaderiesimerkkejä testattiin VR:ssä niiden toimivuuden varmistamiseksi, ja selvittämään toimivatko shaderit halutusti VR:ssä ilman muutoksia.

#### 3.1 Hahmojen kuolema efektin vaihtoehdot

Hahmon kuolemaefektille haettaessa shaderiesimerkkiä otettiin huomiioon asiakkaan toiveet kuolema efektiin liittyen. Hahmojen kuolema efektille löydettiin kaksi visuaalisesti sopivaa shaderi vaihtoehtoa, jotka myös toimivat hahmojen siirryttäessä takaisin kentälle. Kummatkin esiteltiin Harry Alisavakissen blogissa, jossa hän käy yksityiskohtaisesti kummankin shaderin toiminnallisuuden läpi. Kummassakin shaderissä hyödynnetään yhtä muuttuvaa arvoa, jolla määritetään efektin vaihe.

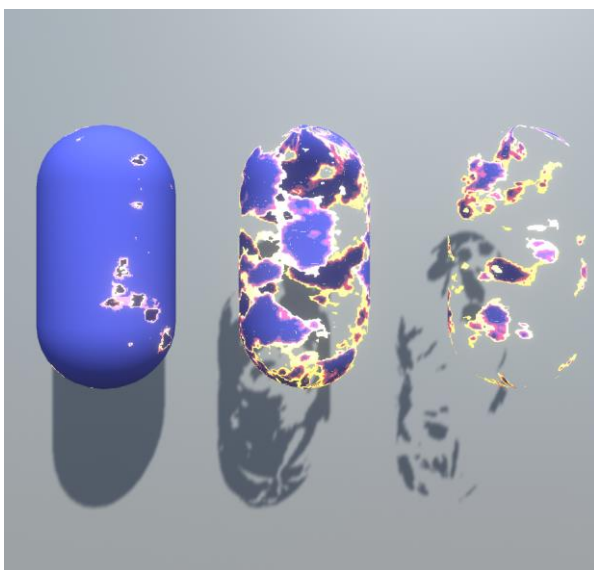
Ensimmäinen vaihtoehtoinen shaderiefekti on nimeltään teleportation dissolve, jossa hahmo piilotetaan näkyvistä satunnaismelua hyödyntäen ja korvaamalla osan piilotetuista paloista emissioivalla palalla (kuva 1). Efektissä myös siirretään mallin kärkipisteitä erikseen lasketulla kertoimella teleportaatiosuuntaan päin. Alisavakis (2018) toteaa kertovansa kertoimen kärkipisteen sijaintiin pohjautuvalla satunnaisluvulla, jolloin jokaista kärkipistettä ei siirretä yhtä paljon ja efekti näyttäisi vain hahmon levitoivan pois. Efektin satunnaismelu tehtiin suoraan shaderiohjelmassa, jolloin visuaalisesti hahmo näyttää pikselöityvän efektin alkaessa.





KUVA 1. Teleportation dissolve-efekti

Toinen vaihtoehtoinen shaderiefekti on nimeltään dissolve, jossa hahmo piilote-  
taan hyödyntäen ohjetekstuuria (kuva 2). Ohjetekstuuria käytetään shaderissä  
antamaan jokaiselle pikselille arvon 0 ja 1 väliltä, josta efektin edetessä vähen-  
netään efektin tilaa edustava arvo. Lopullista laskettua arvoa käytetään clip-toi-  
minnossa, jossa jätetään piirtämättä kaikki pikselit joiden arvo on alle nolla.  
Shaderissä myös käytetään toista tekstuuria väriramppina, jolla lisätään emis-  
siota leikatun alueen reunalle. Reunaefektin luomisesta Alisavakis (2017) kertoo  
käyttävänsä tekstuuria reunavärin kertoimena, jolloin ei tarvitse useita väriramp-  
peja värin vaihtamiseen.



KUVA 2. Dissolve-efekti

### 3.2 Hahmojen korostamisen vaihtoehdot

Hahmon ääriivakorostuksen mallina käytettiin Unify community wikin sivuilta löydettyä silhouette-outlined diffuse nimistä shaderiä, jossa ääriivakorostuksen lisäksi hahmo korostettiin seinien läpi. Efekti toteutettiin mallissa vertex and fragment-shaderissä, jossa voidaan hallita mallin pinnan muotoja ja väriä ilman varjostukseen tarvittavia toimintoja. Efekti toteutetaan luomalla vertex-toiminnossa hahmosta isompi versio, joka värjätään yksiväriseksi ja jätetään kameraa päin olevat pinnat piirtämättä. Vertex-toiminnossa asetettu väri annetaan seuraavaksi fragment-toiminnolle, jossa se asetetaan jokaiselle pikselille. Efektin testauksen yhteydessä sen huomattiin näyttävän VR:ssä isommalta kuin miltä se editorissa näytti. Tämä ongelma efektissä saatiin korjattua helposti asettamalla pienempi arvo efektin koolle. Efektissä hahmon korostaminen seinien läpi tehdään käyttämällä syvyystestiä määrittämään, koska hahmo yksivärinen siluetti piirretään hahmon päätekstuurin sijasta.

Hahmon korostamiselle seinien läpi oli kaksi toiminnallisuudeltaan lähes samantyyppistä vaihtoehtoa, joiden ainoana erona toisen efektin käyttämä toinen kamera efektin luomisessa. Ensimmäinen vaihtoehto hahmon korostamiselle käytti toista kameraa efektin toteuttamiseksi, ja tämän efektin malli löydettiin Making' Stuff Look Good Youtube-kanavalta Stealth Games' XRay Vision -videosta. Efektissä hahmo piirretään kahdesti kahden kameran kautta, joista toinen piirtää hahmon käyttäen korvaavaa shaderiä. Toisen kameran piirtosyvyys on myös asetettu korkeammaksi kuin pääkameran, jolloin sen näkemä kuva piirretään pääkameran kuvan päälle. Efektin reunat luodaan kertomalla annettu väriarvo pikselikohtaisella kertoimella. Tarvittava kerroin saadaan laskemalla pikselikohtaisesti pinnan normaalin ja kameran katsomissuunnan välinen pistetulo ja kertomalla se halutulla arvolla. Laskettu arvo tämän jälkeen vähennetään yhdestä, jolloin saadaan pikselille kerroin. Efektissä stencil-operaatioita käytetään määrittämään, koska hahmon piilossa olevat alueet piirretään käyttäen korvaavaa shaderiä. Tämä toteutetaan antamalla molemmille hahmon piirtämiseen tarkoitetuille shadereille omat stencil-referenssiarvot, joita käytetään määrittämään kummalla shaderillä hahmo piirretään ruutuun. VR:ssä efektin toiminnallisuuden kanssa huomasi olevan ongelmia, koska toinen kamera ei toiminut pääkameran tavoin.

SteamVR:n kamera hallitseva ohjelma kuvasi pääkameralla VR laitteen kummallekin linssille oman kuvan, mutta toista kameraa ohjelma ei käyttänyt. Tämä johti tilanteeseen, jossa vasemmalle silmälle näkyvässä kuvassa oli aina korvaavalla shaderillä piirretty korostettu alue.

Toiseen vaihtoehtoiseen efektiin malli löydettiin Linden Reidin blogista X-ray shaderitutoriaalista, jossa hän hyödyntää shaderissä useampaa eri vaihetta stencil-operaatioiden ja syvyys testin lisäksi. Stencil-operaatiot käyttävät toiminnassaan Unityn stencilpuskuria. Stencilpuskurista Unityn dokumentaatiossa kerrotaan sen olevan monikäyttöinen pikselinaamio, jota voidaan käyttää pikselien tallentamiseen tai hylkäämiseen (Unity Documentation 2019d). Efekti toteutetaan samassa shaderissä pohjaefektin kanssa, jolloin samassa shaderissä on useampi Pass-toiminto efektien toteuttamiseen. Syyksi usealle Pass-toiminnolle Linden Reid (2018) kertoo, että näin varmistetaan mallin takapintojen piirtäminen ennen etupintoja. Kummallekin Pass-toiminnolle annetaan omat stencil-referenssiarvonsa, efektin toimiakseen oikein täytyy pohjaefektin arvo olla isompi kuin x-ray-efektin. Shaderissä ensiksi pohjaefektissä kirjoitetaan stencil-puskuriin sen referenssiarvo syvyydestin onnistuessa, muutoin pidetään jo stencil puskurissa oleva arvo. Tämän jälkeen Pass-toiminnossa, jossa hahmon korostaminen tehdään, verrataan stencil-puskurin arvoa toiminnon omaan referenssiarvoon. Jos arvo on isompi kuin toiminnon oma arvo ei tämän toiminnon efektiä piirretä, muussa tapauksessa efekti piirretään ja puskurissa oleva arvo korvataan toiminnon arvolla. VR:ssä tämä efekti toimi ensimmäistä paremmin, joten tätä efektiä päätettiin käyttää pohjana hahmon korostamisessa seinien läpi.

### **3.3 Kilpi efektin tutkiminen**

Hahmon kilven efektille haettiin mallia Overwatch-pelistä ja sen hahmojen energiakilvistä. Malliksi efektille löytyi Youtubesta Making' Stuff Look Good kanavalta tapaustutkimus Overwatchissa olevasta Winston-hahmon kilvestä. Mallin shadeerissa luotiin efekti tekemällä kohde mallista läpinäkyvä ja antamalla sille väri, jonka vahvuutta hallitaan värissä olevalla alpha arvolla. Efektin kuvio luotiin valmiilla tekstuurilla, joka oli muuten näkymättömissä paitsi efektin aallon kohdalla. Aalto laskenta mallissa suoritettiin lisäämällä saturaatiota käytettävän tekstuurin

kahdelle eri arvolle, lopulta tämä lisättiin lopulliseen väriin shaderin lopussa. Efektissä käytettiin kahta muuta tekstuuria luomaan efektistä elävämpi, luomalla kuvion sykkivän kuvioinnin kilpeen. Makin' Stuff Look Good (2016b) kanavan videolla todettiin, että tekstuurit voidaan pakata yhteen tekstuuriin, koska ne ovat lähes samanlaisia ja ne voidaan erottaa toisistaan käyttämällä kuvan r-, g-, b-arvoja. Nämä arvot ovat kuvan punaiselle, vihreälle ja siniselle värille, joita voidaan käyttää erottamaan yhdestä tekstuuri kuvasta kolme hieman erilaista ohjetekstuuria. Efektin reunat luotiin lähes samalla tavalla kuin hahmon ääriiviiva korostaminen, jossa otettiin pistetulo pikselin normaalista ja kameran katsomissuunnasta. Hohtava alue mallin yläosaan saatiin vähentämällä pikselin y-koordinaatin arvosta 0,45 ja kertomalla se halutulla kertoimella, jolloin hohtavasta alueesta saatiin voimakkaampi ja tarkempi.

Kilven osumien visualisointiin esimerkki löydettiin Unity 2018 Shaders and Effects Cookbook-kirjasta, jossa kirjoittajat John P. Doran ja Alan Zucconi kertovat kuinka toteuttaa lämpökartta Unityn shadereilla. Kirjassa kerrotaan, kuinka lämpökartta toteutetaan hyödyntäen vektori listoja Unityn shadereissa, jonka avulla myös osumien visualisoinnin kilpiin on mahdollista toteuttaa. Doran ja Zucconi (2018, 343) toteavat että Unity sallii listojen asettamisen usean eri metodin kautta, kuten `SetVectorArray`, `SetColorArray`, `SetFloatArray` ja `GetMatrixArray`. `SetVectorArray` metodin avulla on mahdollista antaa kilven shaderille tieto mihin kohtaan kilpeä osuttiin ja koska kyseessä on lista, voidaan useita osumia ottaa ylös ja näyttää yhtäikaa. Kirjan esimerkki koodissa myös annetaan shaderille muuttuja mikä kertoo listassa olevien arvojen määrän, jolloin tyhjiä arvoja ei käydä läpi. Doran ja Zucconi (2018, 344) myös mainitsevat, että `SetVectorArray` metodi tukee kirjoittamisen hetkellä vain `Vector4` muuttujia. Doran ja Zucconi (2018, 344) lisäävät, että Unity automaattisesti lisää nollan neljänneksi arvoksi, kun `Vector3` muuttuja asetetaan `Vector4` muuttujaksi. Tämä ei kilpien osumien visualisoinnin kanssa haittaa, koska tässä tapauksessa osuman koordinaattien lisäksi voimme antaa neljänneksi arvoksi osuman voimakkuuden `Vector4` muuttujaan.

## 4 LOPULLISTEN EFEKTIEN SUUNNITTELU

Kyber Knights pelin lopulliset shader-ohjelmat suunniteltiin haetun taustamateriaalin pohjalta, yhdistelemällä ja muokkaamalla haluttuja efektejä. Pelaajan käyttöliittymässä käytettiin Unityn UI-mask komponenttia, jolla voitiin rajata käyttöliittymässä olevan elämäpalkin ylitse näkyvää tekstuuria elämäpisteiden vähentyessä. Komponentti ja hahmon korostamiseen tarkoitetut shader-efektit käyttävät molemmat Unityn stencilpuskuria, joten suunnittelussa huomioon otettiin shaderien ja UI-mask komponentin mahdolliset ongelmat.

### 4.1 Hahmojen kuolema ja kentälle takaisin siirtyminen

Hahmojen kuolemaefektiksi valittiin löydetyistä shaderiesimerkeistä dissolve-shaderi, jota voitiin käyttää myös hahmon palatessa kentälle. Valittu efekti oli helposti toteutettavissa ilman suurempia muutoksia, ja sen visuaalinen efekti oli nopeasti muokattavissa vaihtamalla kahta efektin käyttämää tekstuuria. Kuolemaefektiä suunniteltiin käytettävän myös pelaajan siirtyessä takaisin kentälle, joka toteutettaisiin tekemällä efekti vastakkaiseen suuntaan. Yhdelle hahmoista myös suunniteltiin lasinen säiliö selkään, joten tarvittava efekti täytyi suunnitella kahteen shaderiin. Lasiselle säiliölle oma versio hahmon shaderistä, jolloin efektien lisäksi shaderistä saatiin visuaalisesti lasisen näköinen.

Hahmon teleportaatioefektin lisäksi shaderin piti olla visuaalisesti samantapainen kuin Unityn standardi-shaderi, joten suunnittelussa täytyi ottaa huomioon, miten tekstuuri-, normaali- ja emissiokarttoja tuetaan. Lopulliseen shaderiefektiin lisättiin efektin lisäksi tarvittavat toiminnallisuudet, jolloin visuaalisesti siitä saatiin Unityn standardi-shaderin näköinen kuolemaefektillä (kuva 3). Suunnitteluvaiheessa oli myös mietitty, mitkä shaderin muuttujien arvoista olisi hyvä muuttaa koodin kautta pelin ollessa päällä ja mitä tarvitsee muuttaa koodin kautta, jotta halutut toiminnot toimivat.



KUVA 3. Valmis efekti

## 4.2 Hahmojen korostaminen

Hahmoja haluttiin korostaa väritetyin ääriivoin ja seinien läpi (kuva 4), jotta pelaajien olisi helpompi tunnistaa tiimiinsä kuuluvat hahmot. Ääriivakorostuksen tarkoitus on auttaa pelaajia tunnistamaan hahmon tiimi ja se oli helpoimpia efektejä suunnitella ja toteuttaa. Efekti suunniteltiin toteutettavan malliefektin mukaan ilman seinien läpi korostavaa efektiä, joka toteutettiin eri malliefektiä käyttäen. Ääriivakorostuksen suunnittelussa otettiin myös huomioon koodinkautta tehtävät muokkaukset, joilla efekti saatiin toimimaan halutulla tavalla.

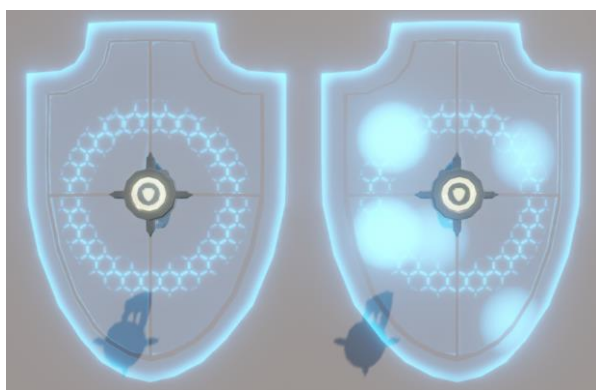


KUVA 4. Hahmon valmiit korostamisefektit

Hahmon seinien läpi korostamisefektiä valittiin pelkästään stencilpuskuria hyödyntävä efekti, joka toimi toista kameraa hyödyntävää efektiä paremmin VR:ssä. Stencilpuskuria käytettäessä shaderille annetaan stenci-osio, jossa annetaan referenssiarvo, vertailu funktio ja määritetään mitä tapahtuu vertailun epäonnistuksessa, onnistuessa tai syvyys testin epäonnistuuessa. Mahdollisia ongelmatilanteita stenciliä käytettäessä on efektin näkyminen tilanteessa, jossa sen ei haluta näkyä. Kyseinen ongelma johtuu usein siitä, että kohde shaderin lisäksi jokin muu shaderi tuottaa saman tuloksen testeistä ja tällöin shaderin efekti näkyy. Tämän takia pitää olla tarkkana referenssiarvojen ja vertailu funktioiden kanssa stenciliä käytettäessä. Hahmon korostamista seinien läpi myös suunniteltiin hyödynnettävän korostamaan pelaajan tappanut hahmo, jolloin pelaaja näkisi, kuka hänet tappoi ja missä hänet tappanut hahmo on.

### 4.3 Kilpien shaderi

Aluksi kilvistä aiottiin tehdä keskiaikaismallisia metallisia kilpiä, joiden visuaaliset efektit olisivat olleet identtiset pelaajan omiin efekteihin. Testien jälkeen huomattiin ongelmia kilpien käytössä, koska pelaajat eivät nähneet kilpien takaa kunnolla ja täten eivät käyttäneet niitä. Joten kilvet päätettiin jakaa kahteen eri osaan eli kilpi- ja kahvaosaan ja molemmat saivat oman shaderin omilla efekteilänsä. Kahvaosan shaderin toiminnot olivat samoja kuin pelaajan shaderin toiminnot, mutta kilpiosa sai toiminnoiltaan uuden shaderin. Tämä muutos uudisti kilven ilmeen vanhasta keskiaikaisesta kilvestä futuristiseksi energiakilveksi (kuva 5).



KUVA 5. Kilven valmiit visuaaliset efektit emission kanssa

Kilven haettu mallieffekti oli visuaalisesti sopiva, mutta sitä tarvitsi muokata paljon kilpien mallien käyttöön. Pohjaksi kilpi osaan valittiin osittain läpinäkyvällä pohjalla oleva kuusikulmainen heksagoni kuvio, jonka väritys asetetaan tiimin mukaan kuviolle ja pohjalle. Mallieffektissä oleva aalto suunniteltiin muokattavan toiminnaltaan siten, että se alkaisi kilven keskeltä ja liikkuisi sen reunoja kohti. Kilpeen lisättiin myös visuaalinen efekti osumia varten, jotta pelaajien olisi helpompi havaita kilven hyöty itsensä suojaamisessa.



## 5 LOPULLISTEN SHADERIEN TOTEUTUS

Lopullisia shadereitä alettiin toteuttaa sopivien malliefektien löytymisen jälkeen ja niiden pohjalta tehtyjen lopullisten efektien suunnitelmien valmistuttua. Toteutuksen yhteydessä tuli vielä ongelmia hahmo mallien ja shaderien välillä, jotka toivat lisää muutoksia lopullisiin shadereihin toteutuksen aikana. Shaderien toteutuksen aikana myös hahmojen Unityn perusshaderiä hyödyntävät materiaalit saatiin valmiiksi, jolloin itse tehtyihin shadereihin tarvitsi tehdä muutoksia ja lisäyksiä, jotta ne olivat visuaalisesti samankaltaisia artistien tekemien materiaalien kanssa.

### 5.1 Hahmojen pohja- ja kuolemaefekti

Hahmon pohjaefektin tekeminen aloitettiin luomalla uusi shader-ohjelman tiedosto, joka tehtiin luomalla Unityssä uusi Standard Surface shader. Unity kirjoittaa uuteen shaderi pohjaan tarvittavat toiminnallisuudet varjostuksen toimintaan, jolloin efektin lisääminen ja muiden haluttujen muutoksien teko oli helppoa. Malli efektiksi valitun dissolve-shaderin toiminto lisättiin hahmon pohjaefektiin pienen muutoksen kera, jolla varmistettiin efektin piilottaneen hahmon kaikki pikselit näkyvistä hohtavan reunaefektin kera.

Efektin jälkeen shaderiin lisättiin metalli- ja normaali kartat, ja myös selvittää kuinka käyttää kyseisiä kartoja oikein shaderissä. Metallikartalla pystyttiin tarkasti asettamaan hahmon tekstuurin metallisten osien metallisuus. Tämä tehtiin pikseli kohtaisesti hakemalla metallikartan avulla tekstuurista fixed4 muotoinen muuttuja, josta ensimmäiset kolme arvoa asetettiin shaderin metalli muuttujaan. Viimeinen arvo muuttujasta asetettiin shaderissä tekstuurin kiillon määrittävään muuttujaan, joka lisäksi kerrottiin itse asetetulla kertoimella. Normaali kartan avulla hahmon tekstuuriin saadaan korkeuseroja ilman, että hahmon mallia pitää muuttaa. Unityn shadereissa voi hyödyntää UnpackScaleNormal toimintoa, jolla saa normaali kartan purettua ja skaalattua tekstuuriin halutulla kertoimella.

Hahmon silmiin ja muihin osiin haluttu emissio tehtiin hyödyntämällä emissio karttaa, jolla saatiin asetettua emissio haluttuihin paikkoihin tekstuurissa. Emissio

kartan avulla saatiin pikselikohtainen kerroin emissiolle nollan ja yhden väliltä, joka kerrottiin shaderissä asetetulla värin arvolla. Shaderin emission arvo kerrotaan lisäksi intensiteetti arvolla, jolla emission väriin lisätään valotusta luoden siitä luonnollisemman. Tarvittavia muutoksia shaderiin tarvitsi muuttaa emission asettaminen tapahtumaan ennen dissolve-efektiä, jonka jälkeen dissolve-efektissä lisättiin emissioon efektin oma emissio. Emissio karttaa pystytään myös hyödyntämään värien asettamiseen emissio alueisiin, jotta emissio tulisi valkoisen alueen sijasta emission väriseltä alueelta. Koska emissio kartta on mustavalkoinen, pystyttiin sen väriarvoja hyödyntämään väritettävän alueen määrittämiseen. Määrittämisen jälkeen tekstuurin väri kerrottiin asetetulla väriarvolla, jolloin valkoinen väri korvattiin hahmon tiimin värillä.

Plague doctor-hahmolla on selässään lasinen pullo, jota varten tarvittiin oma shaderi jolla saadaan lasinen ilme dissolve-efektin lisäksi. Lasi versiossa shaderin metalli ja tasaisuus arvot asetettiin ilman erillisiä tekstuureja, jolloin lasin visuaalista ilmettä pystyttiin vapaammin hallitsemaan. Metallin arvo asetettiin suoraan shaderiin, mutta tasaisuus arvo kerrottiin metallin arvolla ennen sen asettamista. Lasi shaderissä käytettiin tekstuuria pohja shaderin tavoin, mutta shaderin läpinäkyvyys asetettiin värin alpha arvon mukaan tekstuurin alpha arvon sijasta.

## 5.2 Hahmojen korostaminen

Toteutuksessa malliefektin toiminnot lisättiin muokattuna hahmon pohjashaderiin omaan Pass-toimintoonsa, jonka sisälle tehtiin efektille oma vertex and fragment-shaderin toiminnallisuus. Toteutuksessa mallista poiketen verteksi toiminto tehtiin Pass toiminnon sisällä fragment toiminnon kanssa. Lisäksi toteutuksen verteksi toiminnossa ei asetettu väriä vaan se asetettiin teksturiin fragment toiminnossa, jossa asetetaan pikselikohtaiset arvot. Toteutuksessa käytettiin mallista poiketen tekstuuria, jolla taattiin dissolve efektin toiminta. Ilman tätä muutosta olisi hahmon kuollessa siitä jäänyt tiimiläisille hahmon muotoinen värjätty muoto. Ongelmia ääriivakorostuksen kanssa tuli pohjashaderistä toteutetun lasiversion kanssa, jossa lasin läpinäkyvyys ei toiminut efektin kanssa. Ongelmana oli objektin takana olevat pinnat, jotka normaalisti jäivät piiloon objektin taakse, mutta lasilla on tapana näyttää pinnan takana olevat asiat. Ongelmaa

yritettiin ratkaista leikkaamalla pullon taakse jääneet pinnat pois, joka alkuun näytti lupaavalta, mutta lasi vain näytti olevan läpinäkyvä. Ongelmana tuli myös, ettei pullon sisältö näkynyt. Aika rajoitteiden takia lasi shaderistä jätettiin ääriiva toiminto pois, koska sitä ei saatu toimimaan.

Hahmon seinien läpi korostusefekti toteutettiin myös löydetyn mallin mukaan vertex and fragment-shaderiin, jolloin ei tarvinnut laskea valoja ja varjoja pinnalle. Efekti testattiin ennen pelattavien hahmojen valmistumista testi hahmolla, jolla se toimi mallin mukaan ilman ongelmia. Ongelmia efektin kanssa tuli uusien hahmojen mallien kanssa, koska efektin toimintaa ei ollut ajoissa tutkittu ja täten sitä ei oltu otettu huomioon hahmojen teossa. Yksi vaihtoehto tämän ongelman korjaamiseen oli tehdä hahmot uudelleen yhtenäisellä pinnalla, jolloin efekti olisi toiminut halutusti. Tämä vaihtoehto tarkoitti paljon ylimääräistä työtä artisteille, joten ongelman ratkaisemiseksi päädyttiin toiseen vaihtoehtoon ja efektistä muokattiin siluettiversio ääriviivojen sijaan. Efektin siluettiversioon muokkauksen yhteydessä lisättiin myös dissolve-efektin toiminnon shaderiin, koska hahmo näkyisi nyt kokonaan seinien läpi ja kuollessaan hahmon siluetti vain katoaisi äkkinäisesti.

Täten seinien läpi korostusefektiä muutettiin käyttämään tekstuuria pelkän värin sijasta, jotta dissolve-efekti saatiin toimimaan efektissä. Dissolve-efektin toteutus tehtiin shaderin fragment-toiminnossa, jossa efektistä jätettiin pois reuna emissio lisääminen. Lisäksi tämä efekti toteutettiin lopulta omassa shaderi-ohjelmassa, jolloin hahmon pohja ja lasi shadereissa käytettiin UsePass-toimintoa efektin käyttämiseen. Tällä tavalla pystyttiin käyttämään yhtä versiota efektistä ja tekemään muokkauksia vain yhteen ohjelmaan usean eri ohjelman sijasta.

### 5.3 Kilpien efektien toteutus

Kilpi efektin toteutus pohjautuu myös paljon löydetyn mallin toimintaan, mutta tarvitsi paljon muokkauksia, jotta efekti saatiin toimimaan halutusti tasaisella pinnalla pallon sijasta. Efektin toiminta akseli muutettiin y-akselista z-akseliin, jolloin efektin korkein piste saatiin osoittamaan eteenpäin. Efektissä aalto laskettiin pikselin y koordinaatin, ajan ja erikseen määritetyn kertoimen mukaan ja täten saatiin laskettua pikselikohtainen saturaatio, jolloin saatiin pallon yläosasta alaspäin ajan kanssa etenevä aalto. Vastaavan efektin toteuttamiseksi tasaiselle pinnalle pikselien y-koordinaatin sijasta, tarvitsi shaderin sisällä laskea ympyrän kehä. Tälle kehälle annettiin mallin tavoin kerroin, jolloin sen koko pystyttiin muokkaamaan ajan kanssa ja täten animoimaan se. Tämä toteutettiin kertomalla pikselin x-koordinaatti ja z-koordinaatti itsellään ja laskemalla neliöjuuri niiden summasta, lisäksi kaavassa käytetään kerrointa määrittämään aallon koko. Toteutuksessa lisättiin efektiin myös emissio kartta, jolloin saatiin annettua kilpien reunoille lisää hohtoa. Lisäksi useita arvoja joita mallissa oli laitettu muutettiin muuttujiksi, jotta efektiä pystyisi helpommin ja nopeammin muokkaamaan Unityn editorissa.

Kun efekti lisättiin kilpien malleihin ilmeni ongelmia efektin toiminnan kanssa, joka selvisi johtuvan Unityn ja Blenderin koordinaatistojen eroavaisuudesta. Unity käyttää vasenkätistä koordinaatistoa, jossa y akseli osoittaa ylöspäin ja Blenderi puolestaan käyttää oikeakätistä koordinaatistoa, jossa ylöspäin osoitava akselin on z. Ongelma päätettiin korjata koodin kautta, jossa z-akseli korvattiin y-akselilla.

Osumien visualisoinnin toteutettiin käyttämällä muuttuja listaa osuma koordinaattien ja voimakkuuden lähettämiseen shaderille. Efekti toteutettiin for-silmukassa, jossa osuma kohdassa lisättiin shaderin emissio arvoon osuman laskettu saturaatio perustuen pikselin etäisyydestä osumapisteeseen kerrottuna osuman voimakkuudella. Osuman koko määritettiin erikseen shaderissä kahdella eri muuttujalla, joista toista käytettiin myös osuman emission laskennassa. Koska osumien emissio kerrottiin lisäksi vielä kertoimella, päätimme rajoittaa tuloksen sopivaan maksimiarvoon. Tällä tavalla estimme pelaajien sokaistumisen, jos samaan kohtaan tulee useampi osuma lähes samaan aikaan. Doran ja Zucconi

(2018, 343) kertovat Cg:n sallivan listojen käytön, mutta se ei salli listojen luomista joiden koko on tuntematon. Tämän takia toteutuksessa käytettävä lista luotiin 20 paikkaisena, jolloin listassa oli varmasti tilaa kaikille mahdollisille osuille. Osumakohdan ylösottaminen ja animointi tapahtui koodissa eikä shaderissä, joten siitä tarkemmin seuraavassa luvussa.

Shaderiin lisättiin myös dissolve-efekti, koska kilpi shaderiä haluttiin myös hyödyntää pelaajien kentälle siirtymisessä. Tämä auttaa muita pelaajia tunnistamaan pelaajat ketkä ovat kentälle siirtymisen jälkeen hetken kuolemattomia tai, jos he liikkuvat tai ampuvat. Dissolve-efekti lisättiin kilpi efektiin tekemään kuolemattomuus kilven sammumisesta paremman näköisen, sen sijaan että se vain häviäisi yhtäkkiä. Toiminto lisättiin kilpeen samalla tavalla kuin pelaajan korostamisessa seinien läpi, eli ilman reuna emissiota fragment-osiossa shaderiä.

## 6 ARVOJEN MUOKKAAMINEN KODISSA

Shadereissä käytettävien muuttujien arvojen muokkaaminen koodin kautta on useamman aiemmin esitelty efektin toiminnan kanssa pakollista, jotta ne toimivat mallien mukaan ja kun halutaan. Dissolve-efektissä ainoa arvo, jota tarvitsee muokata määrittää efektin vaiheen ja koska efektiä käytetään pelaajan kuollessa tai siirtyessä kentälle, voidaan arvon muokkaus tehdä metodissa mikä on toiminnassa vain tarvittaessa. Arvon muokkaus tehdään Lerp-toiminnossa, jossa muutetaan arvo ensimmäisestä arvosta toiseen arvoon määritetyssä ajassa, joka lasketaan vähentämällä aloitusaika nykyisestä ajasta ja jakamalla arvo määritetyllä arvolla. Lerp-toiminnossa muutettu arvo annetaan jokaisen kuvan aikana shaderille Setfloat-toiminnolla, jossa annetaan halutun muuttujan nimi ja sen uusi arvo.

Aiemmassa luvussa esitelty osumien visualisoinnin arvojen muokkaaminen koodissa tapahtui useammassa eri metodissa, joissa lisättiin uudet osumat listaan ja hallintointiin osumien animointia kuvien välissä. Uusien osumien lisäämisessä osuman koordinaatit täytyi muuttaa Unityn maailman koordinaatistosta kilven koordinaatistoon käyttäen InverseTransformPoint-toimintoa, jolloin osuma pysyy oikeassa kohdassa kilpeä sitä liikuteltaessa. Osumien koordinaateissa tarvitsi myös y- ja z-koordinaatit vaihtaa toistensa kanssa ennen lisäämistä shaderille annettavaan listaan, jotta osumat näkyisivät oikein kilven pinnassa. Shaderille annettavassa listassa oli osumien koordinaattien lisäksi neljäntenä arvona osuman kesto, joka päivitettiin listaan lisäämisen jälkeen vähentämällä siitä delta-Time-arvo, jolloin efektistä saatiin sulavampi. Osumien päivityksessä kaikki osumat, joiden kesto laski nolnaan tai sen alle poistettiin osuma listasta, jolloin seuraavan päivityksen yhteydessä niitä ei enää lisätty shaderille annettavaan listaan. Listan lisäksi shaderille annettiin listassa olevien muuttujien määrä, jolloin kaikkia listan paikkoja ei käyty shaderissä läpi turhaan. Shaderielle annettavien listojen täytyy olla saman kokoisia kuin shaderin listan koko, koska kooltaan pienempi lista skaalaa shaderin listan vastaavaan kokoon. Ongelma tässä on se, että shaderin lista ei enää skaalaudu takaisin sen suurimpaan sallittuun kokoon ohjelman ollessa päällä.

Arvojen muokkaaminen koodissa helpottaa myös pelaajien tiimin vaihtoa, jossa pelaaja vaihtaa oranssista tiimistä siniseen. Tällöin hahmon materiaalia ei tarvitse kokonaan vaihtaa vaan voi vaihtaa vain tarvittavat arvot, kuten hahmon tekstuurin ja väriarvot. Nämä arvot pystytään vaihtamaan komennoilla `SetColor` ja `SetTexture`, joihin kumpaankin tarvitsee vain antaa muuttujan nimi ja uusi arvo. Lisäksi muuttujien arvojen muokkaaminen koodissa mahdollisti Kyber Knights pelissä tappajan korostamisen, jossa pelaajan tappaneen hahmon ääriviivat ja seinien läpi korostuksen väri muutettiin punaiseksi vähäksi aikaa. Lisäksi normaalisti vastapuolen pelaajien seinien läpi korostus ei ole päällä toisen tiimin pelaajille, mutta pelaajalle laitetaan hänet tappaneen pelaajan korostus väliaikaisesti päälle.

## 7 POHDINTA

Opinnäytetyön tavoiteena oli toteuttaa Kyber Knights peliin tarvittavat shaderit niiden efektit ja varmistaa niiden toimivuus VR alustalla. Asetetut tavoitteet pitkälti saavutettiin opinnäytetyössä, vaikkakin projektin aikana tuli eri syistä muutoksia alkuperäiseen suunnitelmaan shaderien toiminnoissa. Osan muutoksista olisi voinut välttää aloittamalla shaderien kehityksen aikaisemmin projektissa, jolloin eri efektien toimintaa olisi keretty perehtyä ja tiedetty mitä efektit vaativat toimiakseen. Alkuperäisen suunnitelman mukaan toteutetut shaderien efektit toimivat pitkälti kuten niiden piti. Muutetut efektit toteutukset saatiin myös toimimaan projektin aikana halutusti, ja osassa efekteistä muutettu versio oli alunperin suunniteltua versiota parempi. Shaderien efektien animointi koodin kautta toteutettiin suunnitelmien mukaan ja saatiin toiminnot toimimaan vain silloin kun niiden tarvitsi olla päällä. Mahdollisiin ongelmiin olisi tämänkin kohdalla varautua, etenkin selvittämällä Unityn ja Blenderin koordinaatisto eron. Tällöin ylimääräisiltä yllätyksiltä kehityksessä olisi selvitty ja säästetty aikaa efektien kehityksessä.

Shaderien jatkokehityksen kannalta projektin jälkeen tuli muutamia kehitysideoita, joilla voisi parantaa jo tehtyjä shadereita ja lisätä uusia efektejä shadereilla. Yksi uusi lisättävä efekti olisi pelaajan liikkumiseen vaikuttava, jolla siitä voisi tehdä näyttävämmän. Tämän efektin voisi toteuttaa käyttämällä hahmon kuolema efektin toista vaihtoehtoa teleportation dissolve-efektiä, jolloin teleporttaamisesta saataisiin hieman samanlainen kuin Overwatch-pelin Sombra-hahmon teleportaus. Tämän toteuttaminen vaatisi korvaavan shaderin käyttöä, joka vaihdettaisiin hahmon siirtyessä kentällä tai hahmon kuollessa. Nykyisiä shadereita pystyisi parantamaan lisäämällä toimintoja tai muokkaamalla nykyisiä, kuten lisäämällä metalli kartalle kertoimen hallitsemaan sen vahvuutta.

Mahdollisia jatkotutkimusaiheita olisi voiko toista kameraa hyödyntää shaderi efektien toteuttamiseen VR:ssä, koska opinnäytetyössä yritettiin hyödyntää toista kameraa ilman tekemättä muutoksia SteamVR:n kameraa hallitsevaan ohjelmaan. Muita aiheita olisi eri shaderi efektien toteuttaminen VR peliin, joita tässä opinnäytetyössä ei toteutettu. Lisäksi yksi hyvä aihe olisi selvittää kannattaako VR peliin tehdä efektit pääosin shadereilla vai hiukkasefekteillä.



## LÄHTEET

- Harry Alisavakis. 2018. Teleportation dissolve. Blogi. Luettu 9.10.2019. <https://halisavakis.com/my-take-on-shaders-teleportation-dissolve/>
- Harry Alisavakis. 2017. Dissolve shader. Blogi. Luettu 9.10.2019. <https://halisavakis.com/my-take-on-shaders-dissolve-shader/>
- John P. Doran & Alan Zucconi. 2018. Implementing Heatmaps with arrays. Unity 2018 Shaders and Effects Cookbook. Luettu 16.10.2019.
- Linden Reid. 2018. X-ray shader tutorial. Blogi. Luettu 9.10.2019. <https://linden-reid.wordpress.com/2018/03/17/x-ray-shader-tutorial-in-unity/>
- Makin' Stuff Look Good. 2016a. Stealth Games' XRay Vision. Video. Katsottu 9.10.2019. <https://www.youtube.com/watch?v=OJkGGuudm38>
- Makin' Stuff Look Good. 2016b. Overwatch: Winston's Barrier Projector. Video. Katsottu 16.10.2019. <https://www.youtube.com/watch?v=C6lGEgcHbWc>
- Unify community. 2013. Silhouette outline. Wiki. Luettu 9.10.2019. [http://wiki.unity3d.com/index.php/Silhouette-Outlined\\_Diffuse](http://wiki.unity3d.com/index.php/Silhouette-Outlined_Diffuse)
- Unity Documentation. 2019a. Material, Shaders & Textures. Luettu 14.11.2019 <https://docs.unity3d.com/Manual/Shaders.html>
- Unity Documentation. 2019b. Shader Reference. Luettu 14.11.2019. <https://docs.unity3d.com/Manual/ShaderOverview.html>
- Unity Documentation. 2019c. Vertex and fragment shader examples. Luettu 15.11.2019. <https://docs.unity3d.com/Manual/SL-VertexFragmentShaderExamples.html>
- Unity Documentation. 2019d. ShaderLab:Stencil. Luettu 28.10.2019. <https://docs.unity3d.com/Manual/SL-Stencil.html>