Haaga-Helia
ammattikorkeakoulu Oy

3D-grafiikan luonti tietokonepeleihin

Arto Elstelä

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2016

Tekijä(t)
Arto Elstelä

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
3D-grafiikan luonti tietokonepeleihin

Sivu- ja liitesivumäärä
38

Työn tarkoitus on perehtyä 3D-mallinnusohjelmaan nimeltä Blender, ja luoda sen avulla tietokonepeliin soveltuva hahmo. Työssä otetaan huomioon ja hyödynnetään nykyajan peliteollisuuden menetelmiä, jotta saadaan kuva siitä, millaista työtä 3D-mallintaja voisi tehdä pelinkehitystiimissä.

3D-mallinnuksella tarkoitetaan luomistyötä, jossa käytetään apuna tietokoneohjelmaa ja lopputuloksena on kolmiulotteinen objekti. 3D-mallinnuksella on nykyään hyvin monipuolisia käyttötarkoituksia aina viihdeteollisuudesta lääketieteeseen ja 3D-tulostukseen.

Työn teoreettisessa osiossa käydään läpi tietokonepelien historiaa ja tekniikkaa, joka on mahdollistanut kolmiulotteisen grafiikan tuottamisen. Osiossa esitellään myös työssä käytettävät ohjelmistot ja niiden käyttöliittymä. Työn projektiosiossa käsitellään 3D-hahmon rakentaminen alusta loppuun vaihe kerrallaan. Työvaiheisiin kuuluu yksinkertaistetusti mallin luonti, tekstuurien luonti ja luurangon luonti.

Asiasanat
3D-mallinnus, tietokonepeli, Blender, grafiikka

Sisällys

1 Johdanto

Tietokonepelit ja niiden pelaaminen ovat muihin viihdemuotoihin, kuten elokuviin ja televisioon verrattuna hyvin uusi keksintö, mutta siitä huolimatta peliteollisuus painii nykyään samassa sarjassa kuin elokuva- ja musiikkiteollisuus. Tietokone- ja konsolipelaamista harrastaa nykyään enemmän ihmisiä kuin koskaan ennemmin, eikä loppua tämän suosion kasvulle ole näköpiirissä. Suomessakin peliala on kasvamassa tasaisesti, vaikka alalla toimiikin vain muutamia tuhansia ihmisiä.

Pelinkehitys on monimutkainen prosessi, ja ajan kuluessa tämä prosessi on muuttunut, sillä uusissa peleissä käytetään aina vain kehittyneempää ja monimutkaisempaa teknologiaa. Nykyajan huippupelien kehitystiimeissä voi olla satoja kehittäjiä, joista jokaisella on omat erityistehtävänsä pelinkehitysprosessissa. Opinnäytetyöni tavoitteena on perehtyä siihen, millaista työtä 3D-mallintaja voisi tehdä peliprojektissa.

3D-mallinnuksella tarkoitetaan tietokoneella tehtyä kolmiulotteisen objektin, kuten tietokonepelihahmon, luontia. Olen kiinnostunut aiheesta, sillä 3D-mallinnus on hyvin moniulotteinen aihe, ja sillä on paljon erilaisia käyttötarkoituksia. Kolmiulotteisia malleja hyödynnetään pelien lisäksi myös elokuvien ja televisiosarjojen tehosteissa, sekä vähemmän viihdekäyttöön tarkoitetuilla aloilla, kuten arkkitehtuurisessa suunnittelussa ja fysiikantutkimuksessa. Viime aikoina myös 3D-tulostus on ollut hyvin suosittu aihe, ja siitä voi tulevaisuudessa tulla hyvin tärkeä osa jokapäiväistä elämäämme. Käyttötarkoituksesta riippuen mallinnusprosessi voi olla hyvinkin erilainen ja sisältää erilaisia vaiheita, mutta perusideana on saada aikaiseksi kolmiulotteinen objekti, jota voidaan hyödyntää pelissä tai muussa simulaatiossa.

Työssäni pyrin perehdyttämään lukijan, sekä itseni, 3D-mallinnuksen perusteiden lisäksi myös 2D-tekstuurien tekoon ja pelimoottorien toimintaan. Tarkoituksena on keskittyä pelien 3D-grafiikkaan, eli käydään läpi, miten ilmaisella Blender-ohjelmalla tehdään pelikäyttöön valmis pelihahmo ja mitä yksityiskohtia sekä erikoistapauksia kannattaa ottaa huomioon, kun mallinnetaan peligrafiikkaa.

2 Tietokonepelit

Tietokonepelit ovat nimensä mukaisesti pelejä, joiden pelaamiseen tarvitaan ohjain sekä näyttölaite, jonka kautta pelaaja näkee mitä pelissä tapahtuu. Tietokonepelejä voi pelata eri alustoilla, kuten pelikonsoleilla, matkapuhelimilla ja PC-tietokoneilla, mikä vaikuttaa pelin ohjaamiseen sekä grafiikan laatuun. Koska tietokonepelit ovat interaktiivisia, eli pelaaja

vaikuttaa itse pelin kulkuun, ne tarjoavat erilaisen kokemuksen verrattuna esimerkiksi elokuviin ja televisioon, mikä on auttanut niiden suosion kasvua. Tietokonepeleistä käytetään yleisesti myös nimitystä videopelit, joka tarkoitti alun perin vain tietyn tyyppisellä grafiikalla toteutettuja pelejä, mutta nykyään sanoilla viitataan samaan asiaan.

2.1 Tietokonepelien historia

Ensimmäinen tietokonepeliin tarkoitettu koodi kirjoitettiin jo vuonna 1947. Kyseessä oli brittiläisen matemaatikko Alan Turingin luoma shakkiohjelma, joka oli liian kehittynyt sen ajan tietokoneille, eikä ohjelmaa ikinä testattu oikeilla tietokoneilla. Sen sijaan ensimmäinen toimiva tietokonepeli saatiin valmiiksi vuonna 1951 Lontoossa pidettävää tiedeaiheista näyttelyä varten. Tietokonefirma Ferrantin työntekijä John Bennett suunnitteli koneen, joka pystyisi pelaamaan hyvin matemaattista Nim-peliä, jotta tietokoneiden matemaattiset kyvyt saataisiin suuren yleisön tietoon. Yleisö piti pelistä, mutta suurin osa oli kiinnostunut vain pelaamaan viihdetarkoituksessa, eikä oppimaan matematiikkaa. (Donovan 2010, 4–6.)

Seuraava innovaatio tietokonepelien kehityksessä tapahtui vasta vuonna 1958, kun ennen ydinaseiden parissa työskennellyt insinööri William Higinbotham keksi käyttää oskilloskoopin ruutua pelin grafiikan esittämiseksi. Hänen luomansa peli oli kahden pelattava tennispeli, joka oli hyvin suosittu yleisön keskuudessa, mutta lopulta pelikone päätettiin purkaa varaosiksi. (Donovan 2010, 8–9.)

Muutamaa vuotta myöhemmin, 1960-luvulla, kourallinen Massachusettsin teknillisen korkeakoulun opiskelijoita sai ideakseen luoda tietokonepelin koska se oli heidän mielestään hauskaa ja luovaa, eikä sen takia koska se olisi hyödyllistä. Vuonna 1961 opiskelijat saivat valmiiksi ensimmäisen version avaruustaistelupelistä, jonka nimeksi tuli ”Spacewar!”. Pelin alustana käytettiin PDP-1 tietokonetta, jollaisia oli käytössä vain muutamissa korkeakouluissa Yhdysvalloissa. Huolimatta tietokoneiden harvinaisuudesta, pelistä tuli hitti eri korkeakoulujen opiskelijoiden keskuudessa. Vuonna 1971 Nutting Associates niminen firma julkaisi oman versionsa pelistä, jonka nimeksi tuli ”Computer Space”. Tämä versio pelistä oli suunniteltu toimimaan kolikoilla, jotta se voisi syrjäyttää perinteiset mekaaniset pinball-pelit baareissa ja pelihalleissa. Pelejä tuotettiinkin yli 1500 kappaletta ja ne tuottivat yli miljoona dollaria, mikä tekee Computer Spacesta ensimmäisen suurta suosiota saavuttaneen tietokonepelin. (Donovan 2010, 10–21.)

Computer Space-kolikkopelin suosiosta huolimatta se ei koskaan saavuttanut maailmanlaajuista huomiota. Vuonna 1972 perustettu Yhdysvaltalainen pelifirma Atari sen sijaan

onnistui herättämään kiinnostusta myös Yhdysvaltojen ulkopuolella Pong-nimisellä kolikkopelillään. Pong on yksinkertainen pöytätennistä muistuttava peli, ja yksi tunnetuimmista varhaisista videopeleistä. Pelin suosion seurauksena kymmenet jäljittelijät loivat omia versioitaan pelistä, jopa Japanissa ja Italiassa asti. Atari kasvoi pikkufirmasta pelihallien mullistajaksi ja alkoi pian kehittää uusia kolikkopelejä, joista tuli myös huippusuosittuja. Pongista julkaistiin vuonna 1975 myös kotikäyttöön tarkoitettu versio, joka myi yli 150 tuhatta kappaletta ja mullisti jälleen videopelibisneksen. (Donovan 2010, 22–26.)

Pongin kotikonsoliversion suosio inspiroi jälleen monia muita yrittäjiä kopioimaan idean, ja vuonna 1977 markkinoilla oli jo yli 60 eri jäljitelmää Pong-pelistä. Monet jäljittelijöistä käyttivät uudenlaista mikropiiriä, joka mahdollisti pelin helpon kopioimisen. Suuri määrä kopiopelejä aiheutti kysynnän laskun ja monet pienet firmat joutuivat sulkemaan ovensa. Vain suuremmat firmat, kuten Atari, pysyivät markkinoilla. (Donovan 2010, 32–36.)

Videopelit olivat tuohon aikaan hyvin yksinkertaisia, koska ne rakennettiin suoraan piirilevyille juotoskolvin ja suurten komponenttien avulla. Pelikehittäjistä suurin osa oli sähköinsinöörejä, eikä ohjelmistokehittäjiä tai taiteilijoita. Asiat kuitenkin muuttuivat, kun Intel toi markkinoille uudentyyppisen mikropiirin, jota kutsutaan mikroprosessoriksi. Prosessori oli suuri edistysaskel tietokoneiden kehityksessä, sillä se mahdollisti tietokoneiden toiminnan ohjaamisen täysin ohjelmistotasolla. Pelinkehittäjille tämä tarkoitti sitä, että juotoskolvit korvattiin ohjelmakoodilla, jolloin myös oli mahdollista luoda yhä monimutkaisempia pelejä. (Donovan 2010, 36–42.)

Vuosia 1978-1982 nimitetään kolikkovideopelien kultaiseksi ajaksi, sillä kolikkopelien teknologia kehittyi nopeasti ja niiden ylläpitokustannukset olivat alhaisemmat kuin koskaan ennen. Monet klassikkopelit, kuten Space Invaders (1978), Pac-Man (1980) ja Donkey Kong (1982) pitivät pelihallit hyvin suosittuina ja tuottavina paikkoina. Monien kolikkopelien grafiikka luotiin vektorigrafiikalla, joka muodostuu geometrisistä muodoista, kuten pisteistä, viivoista ja kaarista. Vektorigrafiikan vahvuus on sen tarkkuudessa ja riippumattomuus resoluutiosta, mutta tuolloin vektoreilla pystyttiin luomaan vain esineiden ääriviivat, mikä rajoitti grafiikan käyttöä. Vaikka vektoreilla ei pystytä suoraan luomaan näyttävää grafiikkaa, ne ovat silti nykyäänkin tärkeä työkalu pelien sisäisen logiikan toiminnassa. (Donovan 2010, 76, 80, 82.)

Samaan aikaan kun kolikkovideopelit olivat suursuosiossa, myös kotikonsolit kehittyivät. Vuonna 1980 Atari julkaisi suursuositun Space Invadersin myös ROM-kasettina (**ROM-cartridge**). Atari 2600-konsolille, mikä paransi firman asemaa suuresti pelimarkkinoilla. Konsolin alkuajan suosiosta huolimatta uudet konsolille julkaistut pelit eivät menestyneet

halutulla tavalla ja suuri määrä huonolaatuisia pelejä täytti pelimarkkinat. Vuonna 1983 tilanne kääntyi niin pahaksi, että puhutaan pelimarkkinoiden vuoden 1983 romahduksesta (**North American video game crash of 1983**). Romahduksen aikana peliteollisuuden tuotto laski 3,2 miljardista vain 100 miljoonaan dollariin, eli noin 97 prosenttia. (Donovan 2010, 79, 97.)

Teollisuuden romahduksen takia monet pienemmät pelifirmat joutuivat sulkemaan ovensa Pohjois-Amerikassa, mikä antoi japanilaisille pelifirmoille markkinaraon. Nintendo julkaisi NES-konsolin, eli Nintendo Entertainment Systemin Pohjois-Amerikassa loppuvuonna 1985. Konsolista tuli suursuosittu vuoteen 1987 mennessä, ja sen suosio auttoi peliteollisuutta toipumaan romahduksesta (Donovan 2010, 165–168). Nintendo Entertainment Systemin prosessori oli 8-bittinen, eli sen tuottama pikseligrafiikka oli rajoitettu 256 värivaihtoehtoon ja vain 16-24 eri väriin samanaikaisesti ruudulla. (Wikipedia, 8-bit color.)

Nintendon suosio Pohjois-Amerikassa auttoi myös muita japanilaisia pelifirmoja, kuten Segaa ja Capcomia kasvamaan. Sega julkaisikin oman konsolinsa nimeltä Sega Genesis vuonna 1988, ja onnistui haastamaan Nintendon aseman konsolimarkkinoilla (Donovan 2010, 213–215). Nintendo vastasi haasteeseen omalla Super Nintendo Entertainment Systemillä, eli SNES:illä. Uudet konsolit olivat 16-bittisellä prosessorilla varustettuja, eli väripaletti kasvoi noin 65 tuhanteen värivaihtoehtoon sekä 256 eri väriin samanaikaisesti ruudulla. (Wikipedia, High color.)

Suurimmat innovaatiot peliteollisuudessa tapahtuivat 1990-luvulla. Ensimmäisen kerran pelien grafiikat pystyttiin toteuttamaan todellisella 3D-grafiikalla bittikarttojen ja vektorien sijaan, sillä markkinoille tuli edullisia kotitietokoneeseen sopivia näytönohjaimia (**GPU/Graphics processing unit**). Näytönohjaimen tarkoituksena on keskittyä grafiikkaan liittyvien laskutoimitusten tekemiseen, jolloin tietokoneen pääsuorittimen (**CPU/Central processing unit**) laskentateho voidaan keskittää muualle, kuten tekoälyyn ja fysiikkaan. Näytönohjainpiirit tulivat vakiokäyttöön myös uuden sukupolven pelikonsoleissa, kuten Sonyn Playstationissa ja Nintendo 64:ssä, eli suurin osa konsolipeleistä toteutettiin tästä lähtien 3D-grafiikalla. (Donovan 2010, 261-263)

Kolmiulotteinen grafiikka ja pelikoneiden laskentatehon kasvu mahdollistivat uudentyylisten pelien innovaation. 90-luvulla suosituiksi tulivat uudet peligenret, kuten ensimmäisen persoonan räiskinnät (**FPS/First-person shooter**), tosiaikaiset strategiapelit (**RTS/Real-time strategy)** ja massiiviset monen pelaajan verkkopelit (**MMO/Massively multiplayer online game**), joista jälkimmäisin oli myös internetin yleistymisen ansiota. (Donovan 2010, 263, 195–197, 307.)

3D-grafiikkaan siirtymisen jälkeen peliteollisuudessa ei ole tapahtunut samankaltaista mullistusta, ainakaan vielä. Näytönohjainpiirit ovat kehittyneet tasaiseen tahtiin, ja peleissä nähdään vuosi vuodelta näyttävämpää grafiikkaa. 2000- ja 2010-luvuilla mobiilipelaaminen on tullut hyvin suosituksi, sillä jopa pienet käsikonsolien ja älypuhelimien näytönojaimet pystyvät näyttämään moninkertaisesti parempaa grafiikkaa kuin ensimmäiset 3D-pelikoneet. (Wikipedia, History of video games.)

3 Pelimoottorit

Pelimoottorilla tarkoitetaan peleihin erikoistunutta ohjelmistokehystä, jonka kautta pelinkehittäjät saavat työkaluja pelinkehityksen eri osa-alueisiin. Tyypillisesti pelimoottorit sisältävät erilaisia komponentteja, jotka ovat vastuussa pelin eri toiminnoista. Yksi tärkeimmistä komponenteista on renderöintimoottori (**Rendering engine**), jonka vastuulla on tuottaa pelin grafiikka esimerkiksi muuttamalla 3D-mallit 2D-muotoon algoritmin avulla (**Rasterisation**), jotta ne voidaan näyttää tietokoneen ruudulta. Renderöinnin lisäksi tärkeitä komponentteja ovat muun muassa ääni-, fysiikka-, ja tekoälymoottorit, jotka ovat vastuussa nimensä mukaisista toiminnoista peleissä ja ovat erillisiä osia pelin pääohjelmasta. Monet pelimoottorit on rakennettu niin, että pelinkehittäjillä on käytössään visuaalinen käyttöliittymä ja ohjelmointiympäristö (**Integrated development environment**), jonka kautta kehitystyö on mahdollisimman nopeaa ja sopisi mahdollisimman monenlaisten pelien kehitykseen. (Kalderon 2011)

Pelimoottorit ovat välttämättömiä pelinkehityksessä, mutta ne ovat pohjimmiltaan vain työkaluja pelinkehittäjien arsenaalissa. Useat suositut kaupalliset pelimoottorit sisältävät kehitystä helpottavia ääni- ja tekstuurikirjastoja, joita tulisi kuitenkin käyttää harkitusti. Pelien sisällön, kuten grafiikan, äänien ja pelimekaniikan, tulisi olla peleissä omalaatuisia, vaikka käytössä olisi sama pelimoottori. Modernit pelimoottorit ovat jo niin kehittyneitä, että pelien sisällön rajana on lähinnä vain pelinkehittäjien mielikuvitus sekä taito luoda haluamansa sisältö, oli se sitten monimutkainen koodinpätkä tai 3D-malli.

3.1 Pelimoottorien historia

Ennen pelimoottorien suosion kasvua pelit ohjelmoitiin useimmiten yksittäisinä ohjelmina, jotka käyttivät saatavilla olevaa laitteistoa mahdollisimman tehokkaasti hyödykseen. 1980-luvun laitteistossa ongelmaksi osoittautui muistin vähyys, mikä esti kunnollisen pelimoottorin kehittämisen, joten lähes kaikki pelit olivat täysin ainutlaatuisia kovakoodattuja ohjelmia. (Lowood 2014)

Termiä ”pelimoottori” alettiin käyttää vasta 1990-luvun keskivaiheilla, kun ensimmäisen persoonan räiskintäpelit (**First-person shooter/FPS**) kuten Id Softwaren Doom ja Quake tulivat hyvin suosituiksi. Niiden teknologia oli niin toimivaa, että muut kehittäjät päättivät ennemmin lisensoida peleissä käytetyn teknologian ja rakentaa oman pelinsä sen päälle, kuin rakentaa täysin oma pelimoottori. Tällöin alettiin myös puhua erikseen pelin sisällöstä (**Game content**), johon kuuluu muun muassa pelin grafiikat, hahmot sekä kentät ja pelimoottorista (**Game engine**), joka kattaa pelin toimimiseen tarvittavan peruslogiikan. (Lowood 2014)

1990-luvun loppupuolella sekä uuden vuosituhannen alussa varsinkin kaksi tietokonepelifirmaa, Id Software ja Epic Games, kilpailivat pelimoottoribisneksen herruudesta. Id Softwaren pelimoottorina toimi Quake-pelin mukaan nimetty Quake engine, jonka versiot myöhemmin uudelleennimettiin id Techiksi. Epic Gamesin vastine id Techille oli Unreal Engine, joka ennen pitkää nousi ylivoimaisesti suosituimmaksi kuin id Tech. Molemmat moottorit ovat yhä aktiivisessa kehityksessä ja lisensoitavissa.

Lyhyessä ajassa pelimoottoreista kehittyi hyvin monipuolisia ja monimutkaisimpia ohjelmistoja, mitä on ikinä kirjoitettu. Nykyaikaiset pelimoottorit ovat kehittyneet sille tasolle, että pelinkehittäjien ei tarvitse olla itse koodareita, vaan pelinkehitystiimit voidaan jakaa graafikoihin, kenttäsuunnittelijoihin, skriptaajiin sekä muihin pelin osa-alueisiin keskittyviin ryhmiin. (Wikipedia, Game_engine)

3.2 Merkittävät pelimoottorit

Yksi merkittävimmistä pelimoottoreista PC-pelaamisen kannalta viimeisen kymmenen vuoden aikana on Valve Corporationin Source-pelimoottori. Se julkaistiin alun perin vuonna 2004 ikonisten Counter-Strike: Source ja Half-Life 2-pelien kanssa. Source pohjautuu hyvin kaukaisesti ensimmäisen Quaken moottoriin, eli nykyisellä nimellään id Tech 1. Merkittävän moottorin Sourcesta tekee sen suorituskyvyn lisäksi keskittyminen yhteisöllisyyteen. Valve on aina ollut hyvin avoin ja ystävällinen peliensä modifikaatioiden (**Modding**) suhteen, ja jopa kannustaa sellaisten tekemiseen. Sourcelle on saatavilla ilmaiset kehitystyökalut, joita kutsutaan nimellä Source SDK (**Software development kit**). Työkalujen julkaisu on saanut aikaan ilmiön, missä pienet harrastelijaporukat ovat julkaisseet omia ilmaispelejään (**Source mods**), jotka usein ovat mielikuvituksellisia ja kokeellisia pelejä verrattuna suuriin AAA-tason peleihin, jotka usein pelkäävät poiketa tutusta pelattavuudesta. Vaikka monet suosituimmista tietokonepeleistä käyttävät Sourcea, sen ikä on

selkeästi nähtävissä. Sourcen isoimpia ongelmia ovat rajoitukset kenttien kokoon, vanhentuneet grafiikat ja kankeat kehitystyökalut. Valve on kehittänyt Sourcen seuraajaa jo monia vuosia, ja luultavasti julkaisee seuraavan pelinsä uudella moottorilla. (Wikipedia, Source)

Epic Gamesin kehittämä Unreal Engine on yksi käytetyimmistä pelimoottoreista nykyajan peliteollisuudessa. Alun perin vuonna 1998 julkaistun pelimoottorin uusin versio on Unreal Engine 4, joka on hyvin moderni ja joustava kehitysalusta. Pelimoottorilla on mahdollista kehittää yksinkertaisten 2D-mobiilipelien lisäksi myös AAA-tason suurprojekteja, joten se sopii harrastelijoiden lisäksi myös suurille pelistudioille. Moottori on yhteensopiva konsoleiden, mobiililaitteiden sekä PC:n kanssa, ja tukee uusimpia grafiikka- ja ohjelmistotekniikoita, kuten DirectX 12 ja NVIDIA PhysX. Epic Games tarjoaa pelimoottorin ja hyvin kattavat kehitystyökalut ilmaiseksi, mutta vaatii maksuksi 5% tuloista, mikäli moottorilla tuotetaan kaupallisia projekteja. Pelimoottorin virallisilla sivuilla on ilmaisia tutoriaaleja sekä dokumentaatiota kehitystyön helpottamiseksi. Saatavilla on myös suuri määrä valmista sisältöä, kuten koodia ja grafiikkaa, jota kehittäjät voivat ostaa tai myydä virallisella kauppapaikalla (**UE4 Marketplace**). Kaiken kaikkiaan Unreal Engine on pätevä ja moderni pelimoottori, jonka avulla voidaan luoda lähes minkälaisia pelejä tahansa, ja lisäksi se on hyvä oppimisympäristö pelinkehityksestä kiinnostuneille. (Masters 2014)

Unity Technologiesin kehittämä Unity-pelimoottori on melko uusi tulokas pelimarkkinoilla. Kiinnostavan Unitystä tekee sen suuntautuminen enemmän pienien indie-kehittäjien tarpeiden mukaiseksi. Unity on hyvin järjestelmäriippumaton, se tukee kaikkia yleisimpiä mobiilikäyttöjärjestelmiä, pelikonsoleita ja jopa selainpohjaisia pelejä on mahdollista kehittää Unityn avulla. Lisäksi Unityn käyttöliittymä on pyritty rakentamaan hyvin intuitiiviseksi ja graafiseksi, jotta kynnys kehitystyön aloittamiseksi olisi mahdollisimman matala. Pelimoottorin alkuaikoina Unityä pidettiin lähinnä mobiilipelien kehitykseen soveltuvana moottorina, mutta varsinkin loppuvuonna 2015 julkaistun Unity 5-version jälkeen moottori kykenee hyvin moderniin grafiikkaan, joka peittoaa helposti esimerkiksi vanhan Source-moottorin tason. Monipuolisuudestaan huolimatta yksi Unityn huonoimmista puolista on kunnollisen grafiikkaeditorin puute, joten kaikki pelien sisältämät 3D-mallit pitää luoda erillisellä ohjelmalla. Hyvänä puolena Unity kuitenkin tukee lähes kaikkien mallinnusohjelmien tiedostomuotoja, ja tarjolla on Unreal Enginen tapaan kauppapaikka, josta kehittäjät voivat halutessaan hankkia peleihinsä sisältöä. (Masters 2014)

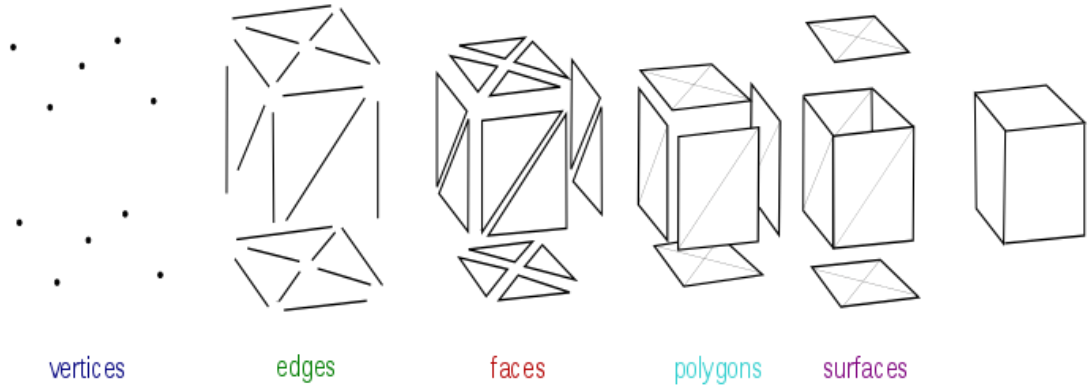
4 3D-mallinnus

3D-mallinnuksella (**3D-modeling**) tarkoitetaan tietokoneohjelman avulla tehtävää luomistyötä, jossa tuloksena on kolmiulotteinen objekti. Kolmiulotteisella mallinnuksella on laaja määrä sovelluksia erilaisilla aloilla. Terveydenhuollossa mallinnusta voidaan käyttää hyödyksi esimerkiksi sisäelinten tarkastelussa ja hoidossa, fyysikot ja kemistit luovat teoreettisia malleja kemiallisista yhdisteistä, arkkitehdit rakentavat 3D-malleja suunnittelemistaan rakennuksista ja elokuvateollisuus luo tietokonegrafiikkaa elokuviin sekä televisiosarjoihin (**Computer-generated imagery**). Näiden lisäksi tietysti peliteollisuus on täysin riippuvainen 3D-mallinnuksesta, sillä suurin osa nykypeleistä on toteutettu 3D-teknologialla. (Blender 3D: Noob to Pro 2016)

4.1 Mallin luonti

3D-mallit voidaan jakaa pääasiassa kahteen eri kategoriaan, kiinteisiin (**Solid**) ja kuorimalleihin (**Shell/boundary**). Erona malleilla on, että kiinteät mallit on mallinnettu myös sisältä, eli ne ovat paljon todellisempia kuin kuorimallit, jotka ovat täysin onttoja sisältä. Kiinteät mallit ovat paljon vaativampia mallintaa, mutta ne ovat hyödyllisiä erilaisissa simulaatioissa, joissa tavoitteena on saada oikeaan maailmaan liittyvää dataa. Kiinteiden mallien käyttötarkoituksia ovat esimerkiksi lääketeollisuus ja erilaiset insinöörisimulaatiot, kuten konesuunnittelu tai arkkitehtuurisuunnittelu. Kuorimallien vahvuus on niiden mallinnuksen helppous sekä pienempi laskentatehon tarve. Kuorimallit sopivatkin paremmin visuaalisiin tehtäviin, kuten tietokonepelien tai elokuvateollisuuden grafiikan mallinnukseen. (Blender 3D: Noob to Pro 2016)

Ennen 3D-mallin luontia pitää päättää millä tyylillä sen haluaa tehdä. Kolme yleisintä tapaa mallin esittämiseksi ovat polygonimallinnus (**Polygonal modeling**), käyrämallinnus (**Curve modeling**) ja digitaalinen veistäminen (**Digital sculpting**). Polygonimallinnuksessa käytetään kolmiulotteisessa avaruudessa sijaitsevia pisteitä, eli kärkiä (**Vertex**), joita yhdistelemällä luodaan särmiä (**Edge**) ja tahoja (**Face**). Lopputuloksena saadaan aikaan rautalankamalli (**Polygon mesh**), joka määrittää mallin muodot. Polygonimallien vahvuutena on niiden monikäyttöisyys ja pieni kuormitus tietokoneen grafiikkapiirille. Suurin ongelma polygoneissa on niiden huono soveltuvuus kaareville muodoille, sillä polygonit ovat kaksiulotteisia, eivätkä siis voi ikinä luoda täysin kaarevaa muotoa. Kuva 1 havainnollistaa polygonimallinnusta. (Digital-Tutors Team 2013)



Kuva 1. Polygonimallinnuksen peruskäsitteet

Käyrämallinnuksessa käytetään nimensä mukaisesti käyriä kolmiulotteisen mallin esittämiseksi. Käyrille annetaan hallintapisteitä (**Control point**), joiden painoarvoa (**weight**) lisäämällä käyrä kaartuu lähemmäs pistettä. Käyrämallinnuksen vahvuutena on tarkkojen ja vähän muistia vievien kaarevien muotojen esittäminen. Käyrämallinnusta voi myös käyttää apuna polygonimallinnuksessa, sillä monet mallinnusohjelmat pystyvät muuttamaan käyrän polygonimuotoon, kun mallinnustyö on tehty. (Blender 3D: Noob to Pro 2016)

Digitaalinen veistäminen on polygoni- ja käyrämallinnukseen verrattuna uusi tapa esittää 3D-malleja. Sen tarkoituksena on antaa mallintajalle työkaluja, jotka simuloivat esimerkiksi oikean saven veistämistä, eli mallintaja pääsee painelemaan, vetämään, tarttumaan, nipistämään ja tasoittamaan mallin pintaa mielensä mukaan. Veistämisen vahvuutena on sen kyky tuottaa hyvin fotorealistisia tuloksia, varsinkin kun kyseessä ovat orgaaniset mallit, jotka koostuvat suurimmaksi osaksi pyöreistä ja epätasaisista muodoista. Veistäminen on myös intuitiivisempi tapa luoda malleja, sillä se muistuttaa hyvin paljon vaikkapa muovailuvahalla leikkimistä, kun esimerkiksi polygonimallinnus vaatii hieman enemmän loogista hahmotus- ja suunnittelukykyä hyvän tuloksen aikaansaamiseksi. (Digital-Tutors Team 2013)

Useimmat digitaaliseen veistämiseen kykenevät ohjelmat käyttävät rautalankamalliin perustuvaa logiikkaa (**mesh-based geometry**), joka muistuttaa paljon polygonimallinnusta, vain työkalut rautalankamallin muokkaamiseksi ovat erilaiset. Veistämisellä tuotetut mallit ovat kuitenkin hyvin tarkkoja, eli ne koostuvat suuresta määrästä polygoneja (**High poly model**), mikä ei sovi peligrafiikaksi sen suuren tehotarpeen vuoksi. Tarkat mallit voidaan onneksi pelkistää vähentämällä polygonien määrää (**Low poly model**), jolloin malleja voidaan käyttää peligrafiikassa. Tarkkojen mallien yksityiskohdat saadaan säilytettyä pelkistetyissäkin malleissa käyttämällä apuna tekstuureja, jotka simuloivat korkeuseroja mallin pinnalla (**Normal/Bump mapping**). (Blender 3D: Noob to Pro 2016)

4.2 Mallin viimeistely

Toimivaan ja hyvännäköiseen malliin liittyy monia muitakin osia kuin vain mallinnusohjelmalla luotu kuori. Tässä osiossa listaan ja selvitän muutamia tärkeimpiä 3D-mallinnukseen liittyviä käsitteitä.

Tekstuurit (**Texture**) ovat 3D-mallin pinnalla näytettäviä kuvia tai kuvioita, jotka rikkovat pinnan pelkistetyn ulkonäön ja tekevät siitä kiinnostavan näköisen. Tekstuureita käytetään lähes kaikessa 3D-mallinnuksessa, sillä niiden avulla saadaan aikaan paljon yksityiskohtia ilman että mallien suorituskyky kärsii. Samalle 3D-mallille voidaan antaa monta erityyppistä tekstuuria, joilla kaikilla on oma vaikutuksensa mallin ulkonäköön:

- **Color/Diffuse map**: Määrittää mallin värityksen, ei tulisi sisältää valaistus tai varjostusinformaatiota. Tämän tyyppinen tekstuuri voi jo yksin luoda hyvin näyttävän mallin, ja 3D-grafiikan alkuaikoina se olikin ainoa käytössä oleva tekstuurityyppi (Digital-tutors, understanding difference texture maps).
- **Bump/Normal map**: Määrittää mallin pinnalla olevat epätasaisuudet. Tekstuurin efekti on kuitenkin vain illuusio, eli mallin siluetti ei muutu tekstuurin vaikutuksesta. Normal map on näistä kehittyneempi versio, joten se reagoi valaistuksen muutokseen uskottavammin kuin bump map. Normal mappien manuaalinen muokkaus käsittelyohjelmalla on kuitenkin haastavaa, joten useimmiten ne generoidaan suoraan mallin high-poly versiota apuna käyttäen (Digital-tutors, bump-normal-and-displacement-maps).
- **Displacement map**: Määrittää myös mallin epätasaisuuksia, mutta efekti ei ole vain illuusio, vaan mallin siluetti muuttuu tekstuurin vaikutuksesta. Huonona puolena lisägeometrian luonti mallin pinnalle on hyvin raskasta tietokoneelle (Digital-tutors, bump-normal-and-displacement-maps).
- **Specular map**: Määrittää mallin kiiltävyyden, eli kuinka paljon valoa heijastuu takaisin sen osuessa mallin pintaan. Tekstuurin voi tehdä täysin manuaalisesti tai

käyttäen apuna color mappia (Digital-tutors, understanding difference texture maps).

- **Reflection map**: Määrittää mallille heijastuksen, joka voi vaihtua ympäristön mukaan. Esimerkiksi pelikentän taivastekstuuria voidaan käyttää myös heijastustekstuurina. Efekti on myös illuusio, mutta se on paljon suorituskykyisempi kuin todellisen heijastuksen laskeminen reaaliajassa.
- **Mask map**: Helpottaa monen eri tekstuurin käyttöä samassa mallissa luomalla ”maskin”, joka määrittää missä kohdassa toinen tekstuuri näytetään. Maski on hyödyllinen, koska se antaa yhdistellä tekstuureja ilman alkuperäisten muokkaamista (Digital-tutors, understanding difference texture maps).
- **Opacity/Alpha map**: Määrittää pinnan läpinäkyvyyden esimerkiksi ikkunoita tai kasvillisuutta varten. Läpinäkyvyys voidaan määritellä myös muiden tekstuurien kautta käyttäen Alpha-väylää (**Alpha channel**) (Wikipedia, Alpha mapping).
- **UV map**: Työkalu, jonka avulla tekstuurien paikka määritetään mallin pinnalla. UV map luodaan purkamalla 3D-malli kaksiulotteisiksi koordinaateiksi mallinnusohjelmassa. UV mapin luonti on yksi kriittisimmistä ja haastavimmista vaiheista 3D-mallinnuksessa, sillä se määrittää miltä tekstuurit näyttävät mallin pinnalla. Luodessaan UV mappia mallintajan pitää miettiä mitkä osat mallista ansaitsevat eniten tilaa, ja että tekstuuri ei veny tai muutenkaan näytä huonolta (Wikipedia, UV mapping).

Materiaalit (**Material**) ovat tekstuurien ohella tärkeä osa nykyaikaista 3D-grafiikkaa. Materiaalit eroavat tekstuureista siten, että niiden avulla voidaan määrittää paljon enemmän erilaisia tehosteita mallin pinnalle. Yksinkertainen tehoste on esimerkiksi mallin ja tekstuurin kiiltävyys, joka voidaan määrittää suoraan materiaalin avulla. Vaikka käytössä olisi sama tekstuuri, materiaalin kiiltävyyden eroilla voidaan saada aikaan uudenkarhea tai käytössä tummunut metallipinta. Materiaalisysteemiä (**Materials system**) voidaan käyttää hyödyksi myös itse pelimekaniikassa, jolloin eri materiaaleille määritetään omat fyysiset ominaisuutensa ja äänensä. Kun määritykset on tehty, voidaan luoda samasta 3D-mallista helposti kaksi versiota, joista toinen on puuta ja toinen metallia. (Wikipedia, Material)

Varjostimet (**Shader**) ovat pieniä tietokoneohjelmia, joiden tehtävänä on luoda näyttäviä tehosteita ruudulle. Esimerkkejä shadereista ovat reaaliaikainen valaistus ja varjostus 3D-mallin pinnalla. Varjostimet toimivat usein yhteistyössä tekstuurien ja materiaalien kanssa, esimerkiksi normal-mappien sisältämä informaatio prosessoidaan varjostimen kautta, jolloin mallin pinnalle saadaan luotua epätasaisuuksia. (Wikipedia, Shader)

5 Blender

Blender on Blender Foundationin kehittämä ilmainen ja avoimeen lähdekoodiin perustuva 3D-grafiikan tekoon tarkoitettu ohjelma. Blender on ominaisuuksiltaan hyvin monipuolinen 3D-ohjelma, jonka sisällä käyttäjät voivat tehdä lähes kaiken 3D-mallinnukseen liittyvän, kuten itse mallinnuksen (**modeling**), luurankojen mallinnuksen (**rigging**), animaation, dynaamisen simulaation sekä tarvittaessa jopa videoeditoinnin tai pelien kehityksen. Tehokäyttäjät voivat avoimen lähdekoodin ansiosta kirjoittaa Python-ohjelmointikielellä omia skriptejä, jotka tuovat Blenderiin uusia ominaisuuksia. Useat käyttäjien tekemät skriptit ja työkalut ovat päätyneet seuraaviin Blenderin virallisiin versioihin, eli kyseessä on aidosti yhteisön panostuksella toimiva projekti. (Blender 2016)

Blender on lisäksi hyvin järjestelmäriippumaton, eli se toimii yhtä hyvin Linuxilla, Windowsilla sekä Macilla. Toimiakseen Blender tarvitsee kaksiytimisen prosessorin, 2 GB muistia sekä OpenGL 2.1 kykenevän näytönohjaimen, eli vaatimukset eivät ole huomattavan suuret. Tietysti tehokkaampi kone on aina parempi sulavan käyttökokemuksen kannalta, ja Blender varmasti osaa ottaa kaiken irti tehokkaasta moniydinprosessorista ja suuresta muistimäärästä esimerkiksi mallien piirtämisvaiheessa (**rendering**). Blender tukee useita yleisiä 3D-tiedostomuotoja, kuten **.3DS** ja **.OBJ**, mutta työn alla olevan projektit kannattaa tallentaa Blenderin omaan projektimuotoon, eli **.blend**-tyypin tiedostoksi. (Blender 2016)

5.1 Blenderin käyttö media-alalla

Blenderin vahvuus muihin 3D-grafiikkaohjelmiin verrattuna on sen ilmaisuus ja yhteisöllähtöinen kehitys. Tämän takia Blender on suosittu harrastelijoiden ja pienempien yritysten keskuudessa, kun taas suuremmat yritykset usein käyttävät maksullisia ja laajempia ohjelmistokokonaisuuksia grafiikan tuottamiseen. Siitä huolimatta Blenderiä on käytetty suurissakin projekteissa, kuten vuonna 2004 julkaistun Spider-Man 2 -elokuvan alkutuotannossa sekä useissa televisiomainoksissa ja -sarjoissa. Myös NASA, eli Yhdysvaltojen ilmailu- ja avaruushallintovirasto käyttää Blenderiä julkisesti saatavilla olevien 3D-mallien luontiin. (Wikipedia, Blender)

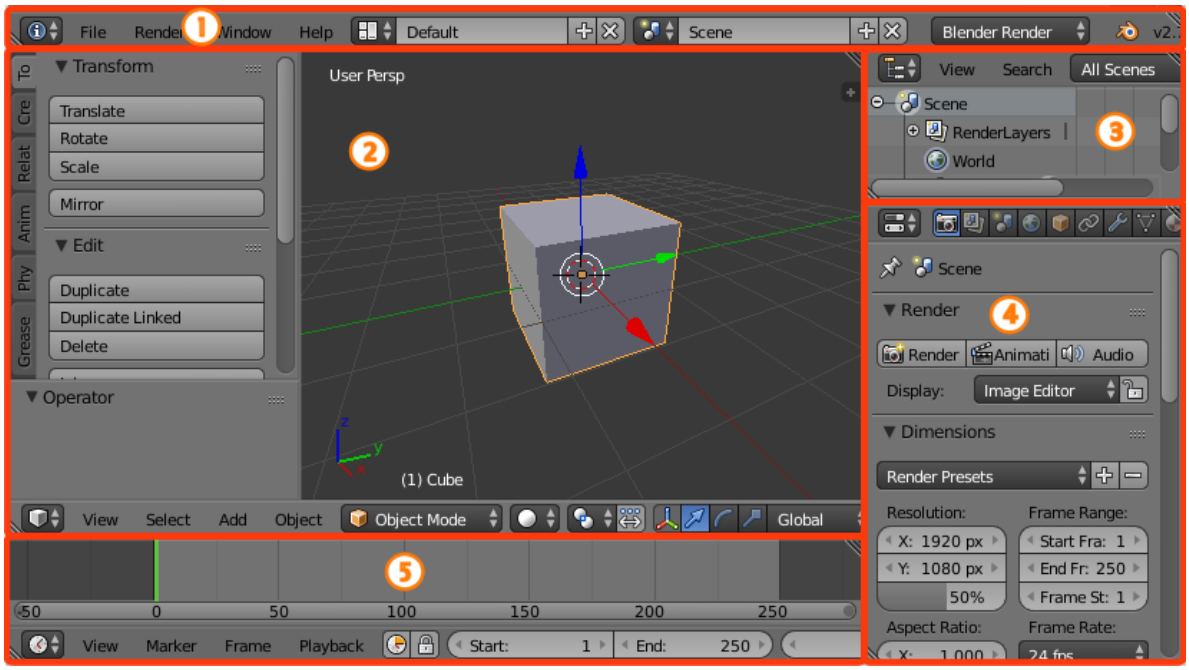
5.2 Käyttöliittymä

Kun Blenderin avaa ensimmäistä kertaa, se tervehtii käyttäjää pienellä ponnahdusikkunalla, joka sisältää linkkejä Blenderin virallisille sivuille. Linkeistä ylivoimaisesti tärkein on virallinen manuaali, josta löytyy apua aloittelijoille sekä vastauksia erilaisiin ongelmatilanteisiin mitä käyttäjä tulee varmasti kohtaamaan jossakin vaiheessa mallinnusprosessia.

Ponnahdusikkunan jälkeen aukeaa suoraan 3D-näkymä, joka on oletusarvoisesti varustettu harmaalla kuutiolla sekä kymmenillä erilaisilla työkalupalkeilla ja paneeleilla.

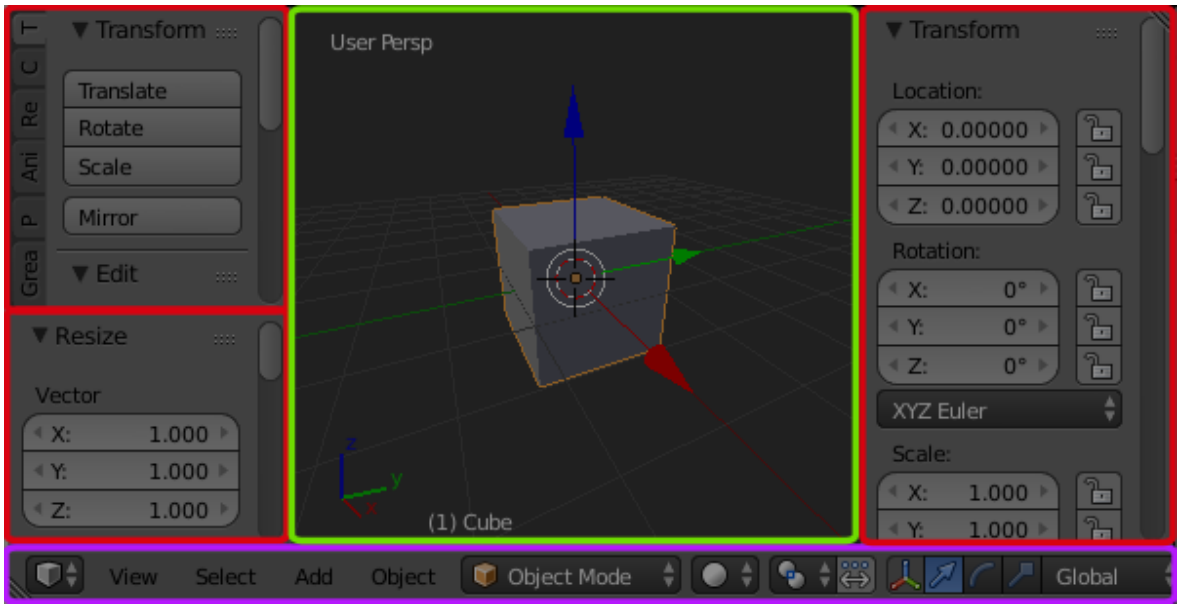
Aloittelijan silmissä Blenderin käyttöliittymä on sekava ja epäintuitiivinen, koska painikkeet sekä tekstit ovat paikoin hyvin pieniä, ja niitä on vakionäkymässä todella suuri määrä. Suurin osa Blenderin saamasta kritiikistä liittyykin sen sekavaan käyttöliittymään. Käyttöliittymällä on kuitenkin myös omat vahvuutensa, kuten käyttäjien mahdollisuus kustomoida sitä oman mielensä mukaan, kunhan mielikuvitusta ja koodaustaitoa löytyy. Kuten aikaisemmin mainittu, Blender on täysin avoimen lähdekoodin ohjelma, joten käyttäjät saavat tehdä täysin vapaasti omia muutoksia ohjelmakoodiin ja jakaa luomuksensa muiden käyttäjien kanssa.

Blenderin käyttöliittymä pohjautuu erityyppisiin editoreihin, joiden avulla käyttäjä voi tarkkailla ja tehdä muutoksia työhönsä. Vakiona käyttöliittymä avaa viisi eri editoria; Info-palkin (**Info**), 3D-näkymän (**3D-view**), ääriviivatyökalun (**Outliner**), ominaisuuspaneelin (**Properties**) ja aikajanan (**Timeline**). Kuvassa 2 esimerkkejä editoreista.



Kuva 2. Blenderin editorit

Editorit koostuvat useista komponenteista, kuten työkalupalkeista tai tietopaneeleista. Kuvassa 3 näkyvät 3D-näkymän eri komponentit.



Kuva 3. Blenderin 3D-näkymän komponentit

Suosittelen aloittelijoita etsimään videotutoriaaleja esimerkiksi Youtubesta, sillä niitä seuraamalla käyttöliittymän perusteiden opettelu tuntuu paljon luontevammalta kuin pelkkää manuaalia lukemalla. Blender Foundation tarjoaa myös kuukausimaksullista palvelua, jonka kautta pääsee katselemaan ammattilaisten tekemiä tutoriaaleja.

6 Photoshop

Adobe Photoshop on Adobe Systemsin kehittämä kuvankäsittelyohjelma, joka on tarkoitettu bittikarttagrafiikan (**Raster graphics**) luomiseen ja muokkaukseen. Photoshop sisältää laajan määrän työkaluja ja ominaisuuksia, kuten kerrokset (**Layer**), maskit (**Mask**), läpinäkyvyyden (**Transparency/Alpha compositing**) ja eri värimallit (**Color model**). Photoshop käyttää omia tiedostomuotojaan, jotka ovat .**PSD** ja .**PSB**, mutta näiden lisäksi ohjelma tukee myös useimpia yleisiä graafisia tiedostomuotoja. Bittikarttagrafiikan lisäksi Photoshopissa on rajalliset työkalut muun muassa vektorigrafiikan (**Vector graphics**), tekstin, 3D-grafiikan ja videon muokkaamiseksi. Photoshopin työkaluvalikoimaa voi halutessaan myös laajentaa liitännäisten (**Plug-in**) avulla, joita on tarjolla ilmaiseksi sekä maksullisina versioina. Photoshop on yhteensopiva vain Windows- ja OS X-käyttöjärjestelmien kanssa, eli Linux-käyttäjät joutuvat etsimään toisen vaihtoehdon.

Alun perin vuonna 1990 julkaistu Photoshop on kokenut pitkän historiansa aikana muutoksia eri versioiden nimeämiskäytäntöihin ja liiketoimintamalleihin. Vuoteen 2003 asti Photoshopin versiot nimettiin yksinkertaisesti versionumeroilla, minkä jälkeen Photoshop 8:n sijaan uusi versio saikin nimekseen Photoshop CS (Creative Suite). Creative Suiten

ideana oli yhdistää kaikki Adoben ohjelmistot, kuten Acrobat, Premiere ja Illustrator, saman palvelun ja nimeämiskäytännön alle. Creative Suiten kehitys jatkui vuoteen 2013 saakka, jolloin sen seuraaja Creative Cloud julkistettiin. Photoshop CS6 jäi viimeiseksi kertamaksulliseksi Photoshop-versioksi, sillä sen seuraaja Photoshop CC on saatavilla vain kuukausimaksullisen Creative Cloud-palvelun kautta. Käytän itse työssäni vuonna 2010 julkaistua Photoshop CS5-versiota, sillä Photoshopin perusominaisuudet ovat pysyneet vuosien aikana hyvin samanlaisina, eikä ole ollut tarvetta päivittää uudempaan. Yleisesti on kuitenkin suositeltua käyttää aina uusinta versiota, mikäli se on mahdollista.

6.1 Photoshopin käyttö media-alalla

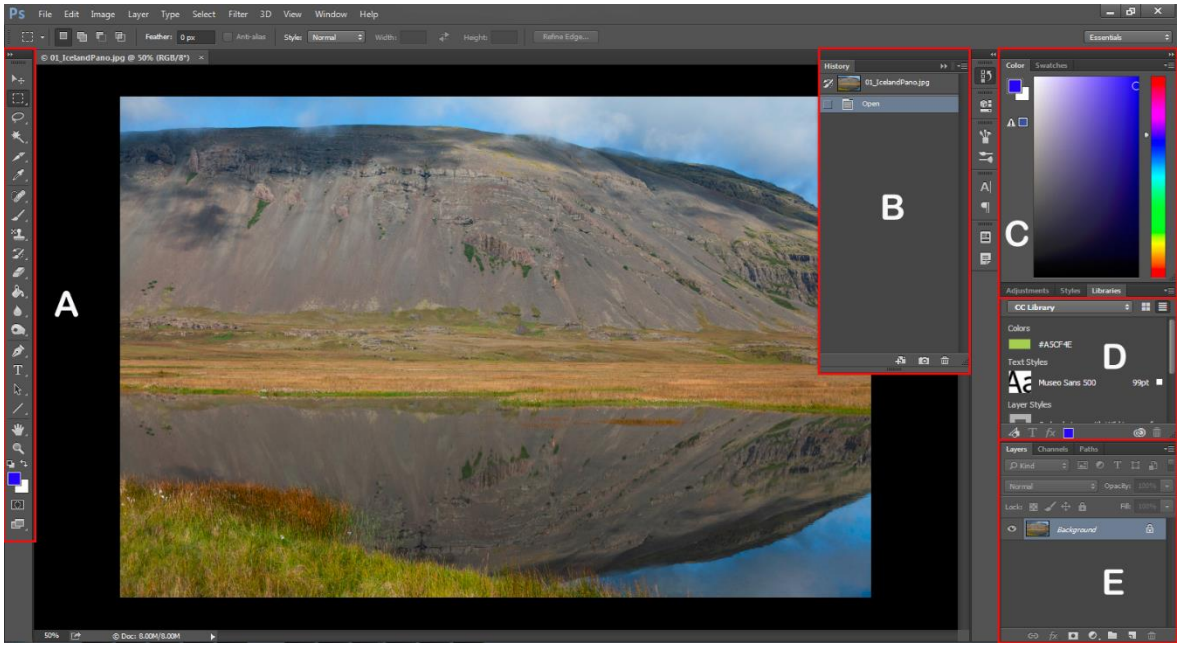
Toisin kuin Blender, Adoben Photoshop on jo vuosikymmeniä ollut mediateollisuuden vakiotyökalu kaikenlaisessa grafiikkaan, valokuvaukseen ja kuvanmuokkaukseen liittyvässä työssä. Photoshopin vahvuutena on sen monipuolisuus ja tehokkaat työkalut, jotka tekevät siitä korvaamattoman yleistyökalun, vaikka muut ohjelmat olisivatkin parempia tietyissä erikoistapauksissa, kuten konseptitaiteen piirtämisessä tai vektorigrafiikassa. Photoshopin alkuperäisestä julkaisusta lähtien se on pitänyt johtoaseman alalla, eikä vaihtoehtoisia ohjelmia juurikaan löydy. Harrastelijoiden keskuudessa vaihtoehtoisia ohjelmia ovat esimerkiksi Paint.NET ja varsinkin GIMP. Molemmat ohjelmat ovat ilmaisia, ja GIMP on myös avoimen lähdekoodin ohjelma, joten se sopii hyvin aloittelijoille sekä harrastelijoille.

6.2 Käyttöliittymä

Verrattuna Blenderiin, Photoshopin käyttöliittymä on hyvin selkeä ja intuitiivinen. Jokainen joka on koskaan käyttänyt Microsoftin Paintia tai vastaavaa piirto-ohjelmaa huomaa selkeitä yhtäläisyyksiä ohjelmien välillä, vaikka Photoshopissa onkin paljon enemmän työkaluja.

Kun käyttäjä avaa Photoshopin, ohjelma avautuu työtilaan (**Workspace**), joka sisältää kaikki tarvittavat perustyökalut. Vasemmassa laidassa on työkalupaneeli (**Tools panel**) ja oikeassa laidassa on väripaneeli (**Colors panel**), säätöpaneeli (**Adjustments panel**) ja kerrospaneeli (**Layers panel**). Näiden lisäksi käyttäjä saa halutessaan auki muutoshistorian (**History panel**) sekä muita paneeleita tarkempia säätöjä varten. Näiden lisäksi tärkeimpiä elementtejä käyttöliittymässä ovat työtilan yläosassa sijaitseva sovelluspalkki (**Application bar**), joka sisältää sovelluksen tärkeitä valikoita ja säätöjä. Palkin alla sijaitsee valitun työkalun säädöistä vastuussa oleva palkki (**Options bar Control panel**).

Vaikka käyttämäni esimerkkikuva (Kuva 4) onkin Photoshopin uusimmasta CC 2015-versiosta, ohjelman käyttöliittymä on lähes identtinen vanhemman CS5-version kanssa.



Kuva 4. Photoshopin paneelit

Vaikka Photoshopin käyttöönotto on paljon helpompaa Blenderiin verrattuna, kannattaa silti lukea Adoben oppaasta läpi työtilan perusteet (**Workspace basics**), tai vaihtoehtoisesti katsoa esimerkiksi Youtubesta tutoriaali aiheesta. Joskus kokeneetkin käyttäjät löytävät uutta tietoa kertaamalla ohjelman perusteet.

7 Pelihahmon 3D-mallin luonti

Työn ensimmäisessä vaiheessa tavoitteena on luoda hahmon 3D-malli valmiiksi, jotta se voidaan myöhemmissä vaiheissa teksturoida sekä viedä pelimoottoriin. Tämä työn vaihe keskittyy mallinnustyöhön, joten käytössä on suurimmaksi osaksi Blender. Käytän työssäni apuna Blender Cloud-palvelussa olevaa video-opasta nimeltä ”Blender Game Asset Creation”.

7.1 Mallin luonnin vaiheet

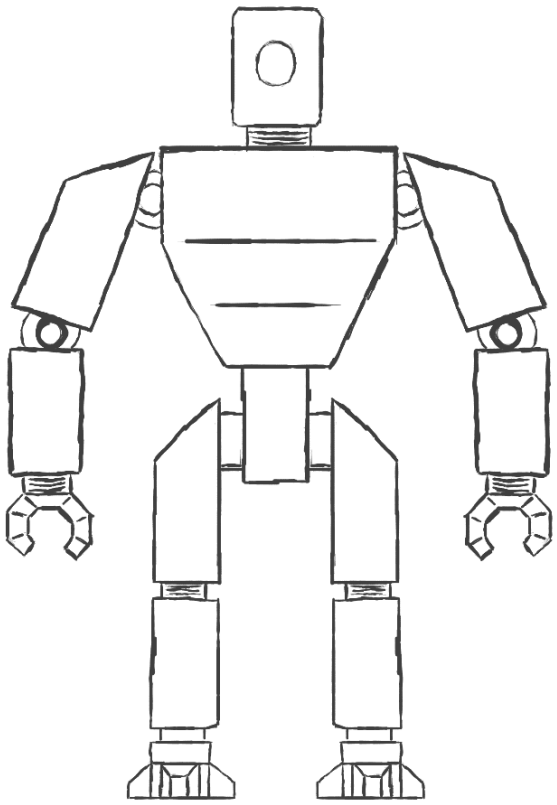
3D-mallinnuksessa seurataan usein hyväksi havaittua työnkulkua. Työ aloitetaan lähes aina etsimällä referenssikuva aiheesta, jota lähdetään mallintamaan. Referenssikuvat voivat olla valokuvia oikeista esineistä tai vaikkapa jonkun piirtämä konseptikuva. Seuraavana vuorossa on rautalankamallin rakentaminen referenssikuvan pohjalta. Kun yksinkertainen perusmalli on saatu rakennettua, voidaan aloittaa yksityiskohtien lisääminen. 3D-

mallin yksityiskohtainen geometria on hyvä saada kokonaan valmiiksi, ennen kuin siirtyy seuraaviin vaiheisiin, jotta vältytään työn uudelleentekemiseltä. Mallin viimeistelyn jälkeen voidaan joko alkaa luoda tekstuureita (**Texture**) tai luurankoa (**Armature/Skeleton**).

7.2 Konseptitaiteen luonti

Opinnäytetyössäni tarkoituksena on luoda pelihahmo, joten päätin tekeväni ihmismäisen robottihahmon. Robottihahmo on mielestäni kiinnostavampi mallintaa ja varsinkin teksturoida kuin vaikkapa tavallinen ihmishahmo. Ensimmäinen askel robotin luomisessa oli piirtää konseptikuva Photoshopin avulla. Luomani konseptitaide on melko nopeasti piirretty kuva (Kuva 5), eikä siinä ole esimerkiksi lopullisessa mallissa olevia tekstuureita ollenkaan, koska halusin päästä mahdollisimman nopeasti mallintamaan hahmoa.

Koska halusin hahmoni olevan suhteellisen yksinkertainen ja aloittelijaystävällinen, päätin käyttää vain yhtä referenssikuvaa, jossa näkyy hahmon etuprofiili. Monimutkaisempien mallien kanssa on hyvä olla vähintään referenssikuvat hahmon etu- ja sivuprofiilista. Mitä parempia referenssikuvat ovat, sen mieluisampaa niiden pohjalta on mallintaa. Varsinkin peligrafiikan teossa taitavasti tehty konseptitaide on tärkeää, sillä kuvat toimivat myös inspiraationa 3D-mallintajille. Nykyaikaisissa pelinkehitystiimeissä on jopa jäseniä, jotka keskittyvät vain konseptitaiteen tekoon, jonka pohjalta 3D-mallintajat tuovat konseptit peleihin. Omaan projektiin voi etsiä referenssikuvia vaikkapa Googlen kuvahausta, eikä sillä ole suurempaa merkitystä ovatko kuvat valokuvia vai konseptitaidetta, kunhan mallinnettavan asian muodot ovat hyvin näkyvillä.



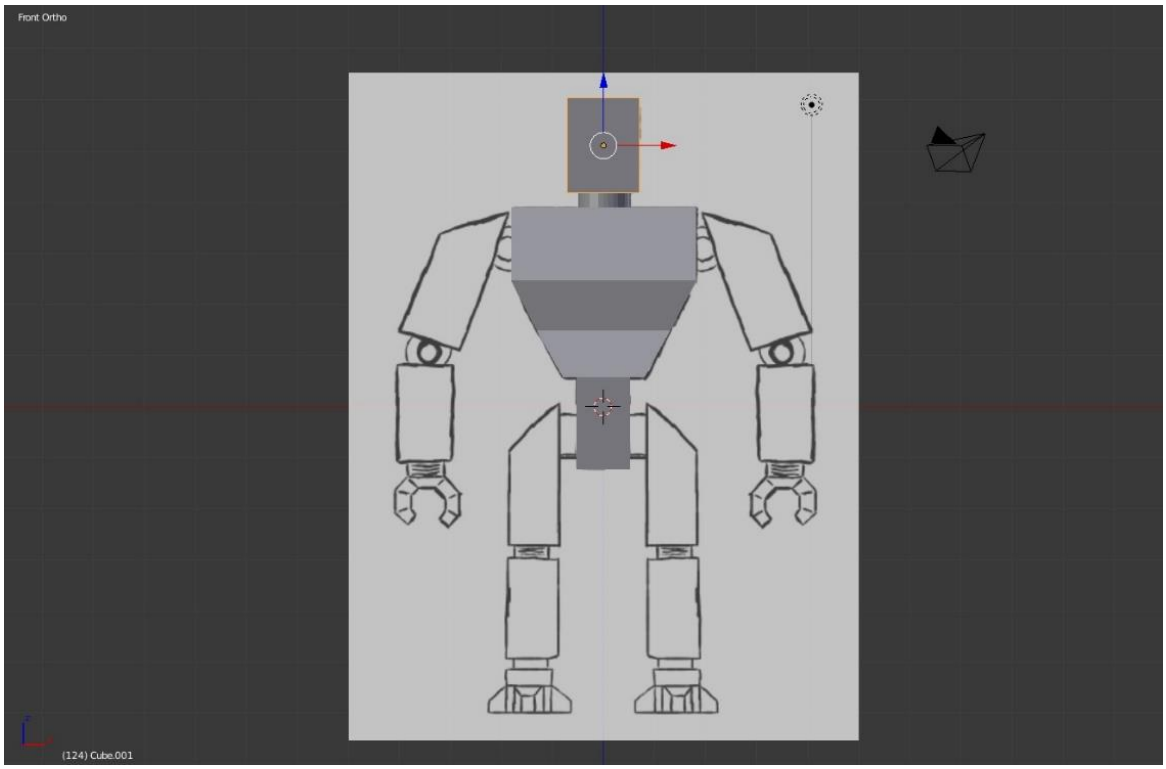
Kuva 5. Piirtämäni konseptikuva

7.3 3D-mallinnuksen aloittaminen

Aloitin hahmon luomisen tuomalla referenssikuvan Blenderiin taustakuvaksi. Taustakuvat näkyvät vain ortografisessa näkymässä, mihin pääsee kätevästi painamalla numpadin numeroa 5 ja sen jälkeen vielä 1, jolloin näkymä on ortografinen etunäkymä (**Front Ortho**). Referenssikuva kannattaa sijoittaa siten, että sen keskikohta on z-akselin kohdalla, varsinkin jos mallinnettavana on hahmo tai esine, joka on symmetrinen. Oma referenssikuvani sijoittui z-akselin kannalta lähes täydellisesti, joten pääsin heti jatkamaan mallinnusta.

Päätin aloittaa mallinnuksen hahmon keskiruumiista ja päästä, jotta hahmolle saadaan tehtyä jonkinlainen perusrunko, johon raajat voidaan liittää myöhemmin. Hahmon pää ja keskiruumis voidaan aloittaa lisäämällä objektitilassa (**Object Mode**) kolme kuutiota, sekä yksi sylinteri niskaa varten. Nämä perusobjektit saadaan lisättyä helposti pikavalikosta, joka aukeaa näppäinkomennolla Shift+A. Asettelin kuutiot suurin piirtein oikeille kohdilleen objektitilassa, käyttäen hyödyksi Blenderin muunnostyökaluja (**Transformations**), kuten g-näppäimellä tehtävää liikutusta (**Grab/Move**) ja s-näppäimellä tehtävää skaalausta (**Scale**).

Sen jälkeen ryhdyin muokkaamaan objekteja editointitilassa (**Edit Mode**), johon päästään siirtymään helposti tab-näppäimellä. Jotta referenssikuva näkyisi lisättyjen objektien alta, voidaan näkymä muuttaa z-näppäimellä rautalankanäkymäksi (**Wireframe**) ja takaisin kiinteäksi (**Solid**). Hahmon keskiruumiin muodot saadaan tehtyä lisäämällä keskimmäiseen kuutioon kaksi uutta silmukkaa (**Loop cut**) Ctrl+R näppäinkomennolla, ja sen jälkeen skaalaamalla tai liikuttamalla kärjet oikeille kohdilleen. Kuvassa 6 näkyy työvaiheen tulos.



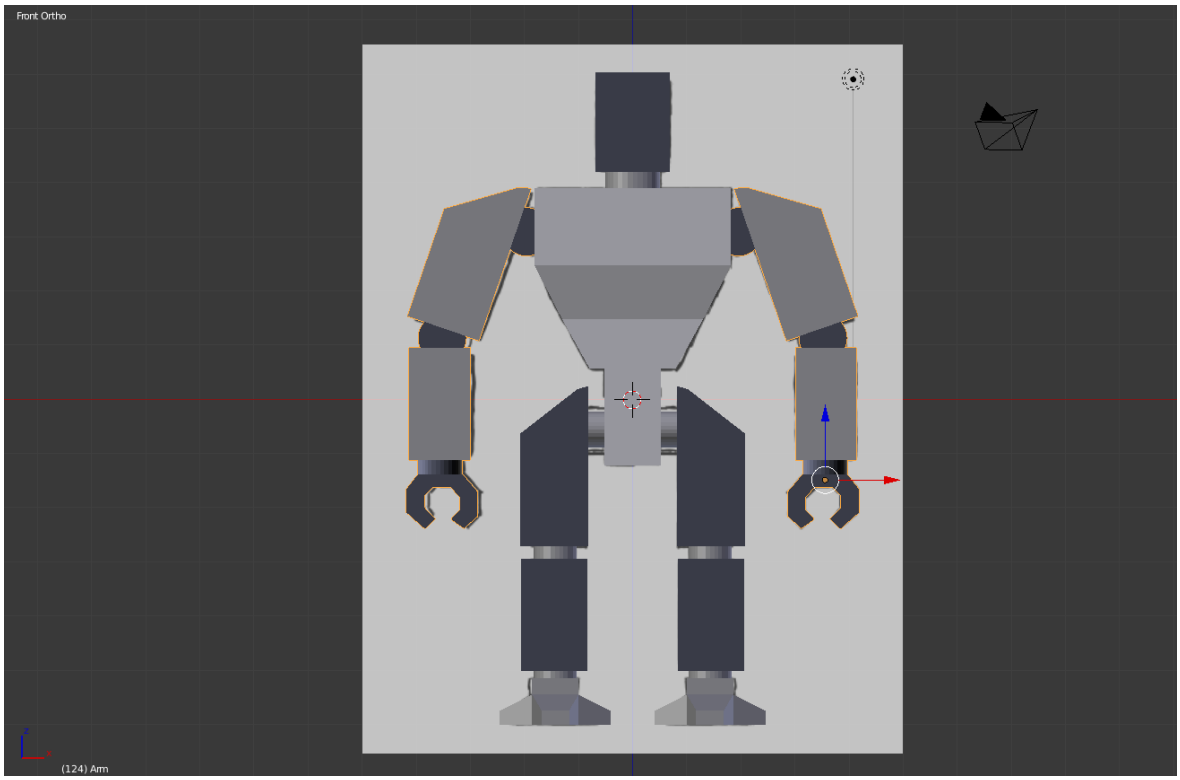
Kuva 6. Hahmon perusrunko

7.4 Raajojen mallinnus

Kun keskiruumiin perusmalli on saatu valmiiksi, voidaan siirtyä tekemään raajoja. Käden perusmallin teko on hyvin samankaltainen operaatio kuin aikaisemminkin, eli lisätään objektitilassa kaksi kuutiota ja kolme sylinteriä, jotka asetellaan paikalleen referenssikuvan perusteella, käyttäen muunnostyökaluja. Työkaluja käyttäessä tulee muistaa, että muutokset voi lukita yhdelle akselille, esimerkiksi tässä tapauksessa sylinterit pyöritettiin (**Rotate**) 90 astetta x-akselin suuntaisesti, mikä onnistuu painamalla x-näppäintä pyörityskomennon jälkeen. Tässä työvaiheessa aikaa säästää myös samanlaisten objektien kopiointi (**Duplicate**) näppäimillä Shift+D, sillä kopioidut objektit käyttävät samoja muunnosarvoja kuin alkuperäiset, eli kaikkia palikoita ei tarvitse erikseen pyöritellä. Kädessä olevan koukun tein lisäämällä ensin kaksiulotteisen tason ja käyttämällä e-näppäimellä pursutustyökalua (**Extrude**), joka luo uusia kärkiä valituista kohdista.

Suurin osa jalan geometriasta saatiin valmiiksi kopioimalla paloja kädestä, mutta jalkaterän mallinnus oli kiinnostavampi operaatio. Käytin sen mallinnuksessa hyödykseni tuttua silmukan leikkaus-menetelmää ja pursutustyökalua, jonka jälkeen skaalasin ulokkeet kapenemaan kärkeä kohti, jolloin tuloksena oli kiinnostava muoto hahmon jaloille. Tässä vaiheessa työtä päätin yhdistellä lisäämiäni objekteja toisiinsa, jotta hahmon ruumiinosista saadaan helpommin hallittavia kokonaisuuksia. Käytin tähän tehtävään Ctrl+J yhdistämistyökalua (**Join**), joka yhdistää valitut objektit yhdeksi. Tämän jälkeen nimesin uudet objektit kuvaamaan paremmin niiden tarkoitusta (Head, Body, Arm, Leg).

Kun raajat on saatu muokattua yhtenäisiksi kokonaisuuksiksi, voidaan lisäksi käyttää Blenderin muuttujia (**Modifiers**) hyödyksi, jotta saadaan peilattua raajat myös hahmon toiselle puolelle. Muuttujat löytyvät oikeasta sivupaneelista, pienen jakoavainsymbolin alta. Tässä tapauksessa haluamme käyttää peilausta (**Mirror**), ja valita peilaavaksi objektiksi hahmon keskiruumiin. Koska keskiruumis on mallinnettu symmetrisesti, raajat peilautuvat juuri niin kuin halutaan ilman suuria lisäsäätöjä. Muuttujat ovat hyödyllisiä monissa tilanteissa, ja ne ovat myös hyvin käyttäjäystävällisiä, sillä niiden tekemät muutokset eivät ole lopullisia ennen kuin niiden halutaan olevan. Peilattua kättä voi muokata ja muutokset peilautuvat myös. Kuvassa 7 näkyy työvaiheen lopputulos.

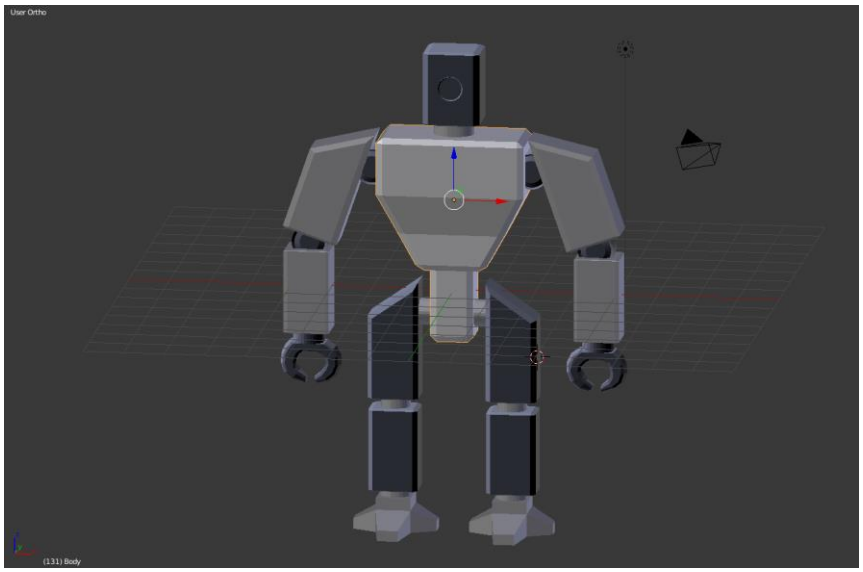


Kuva 7. Hahmon raajojen mallinnus

7.5 Yksityiskohtien mallinnus

Hahmon perusmuodot ovat nyt hyvällä mallilla, joten voidaan alkaa lisätä yksityiskohtia ja valmistella mallia seuraavia työvaiheita varten. Tässä vaiheessa lisäsin mallille silmän, jonka muodostin skaalaamalla sylinterin hyvin ohueksi ja pursuttamalla sylinterin reunasta pyöreän muodon, josta saatiin silmälle reunus. Lisäksi hahmon ruumis, pää ja raajat ovat nyt yksityiskohtaisempia, sillä tasoitin teräviä reunoja käyttäen hyödyksi Ctrl+B hiomistyökalua (**Bevel**). Hahmo on yhä kulmikas, mutta näyttää nyt paljon laadukkaammalta kuin aiemmin. Myös raajojen taitekohdissa olevat sylinterit saivat lisää yksityiskohtia, ja näyttävät nyt enemmän mekaanisilta konepyöriltä. Sylinterien yksityiskohdat luotiin samalla menetelmällä kuin hahmon silmä. Lisäsin hahmon koukkukäsiin lisää tahoja, että sain muokattua käden haluamaani muotoon.

Yksityiskohtien lisäksi siistin hieman tekemääni rautalankamallia poistamalla turhia tahoja ja kärkiä, sekä asettelemalla esimerkiksi raajojen välikappaleita paremmin paikoilleen. On kannattavaa poistaa kaikki kärjet, särmät ja tahot jotka ovat päätyneet jostain syystä muun geometrian sisäpuolelle, ellei sille ole jotain erityistä syytä. 3D-mallin tarkastelija, kuten tietokonepelin pelaaja, ei ikinä tule näkemään niitä, mutta ne syövät silti tietokoneen laskentatehoa. Jos peleissä on suuri määrä huonosti viimeisteltyjä 3D-malleja, voi pelin suorituskyky heikentyä huomattavastikin. Blenderissä on automaattinen työkalu (**Remove Doubles**), joka poistaa päällekkäiset kärjet valitusta objektista. Suosittelen käyttämään tätä työkalua usein, sillä se on nopea ja helppo tapa välttyä ongelmilta mallinnuksen aikana. Mallin eri osien asetteluun on myös kätevä työkalu, jonka avulla osat napsahtavat paikoilleen (**Snap during transform**) juuri kuten haluat, eikä osia tarvitse itse tähdätä millintarkasti paikoilleen. Kuvassa 8 näkyy työvaiheen lopputulos.

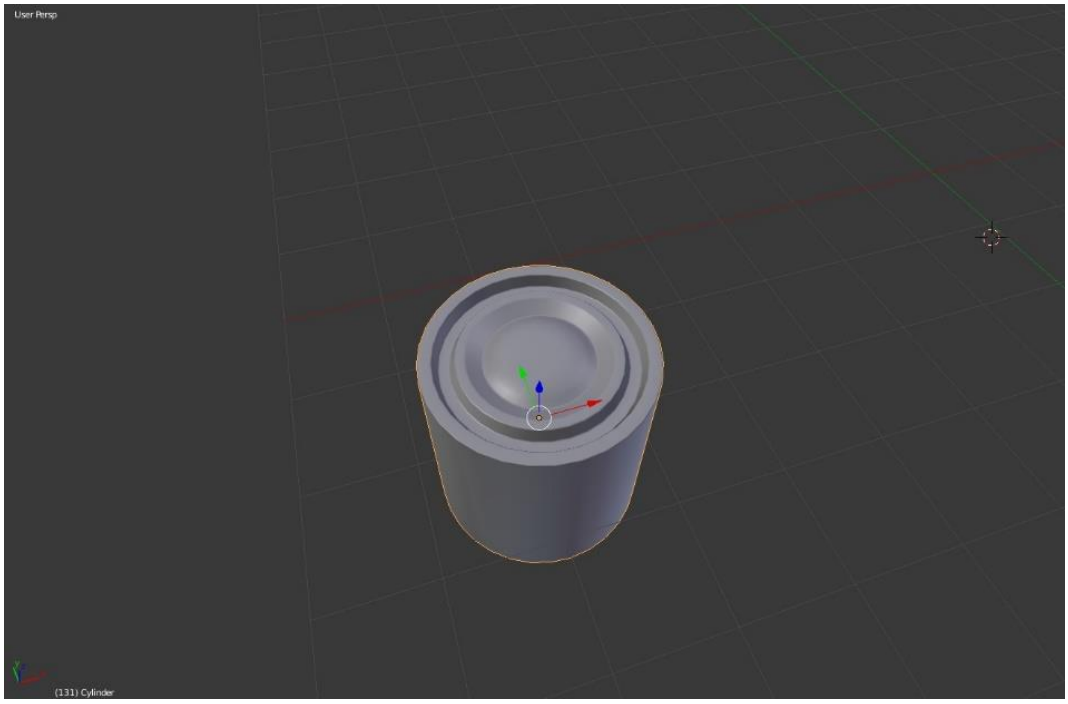


Kuva 8. Hahmon yksityiskohdat

7.6 Pienet yksityiskohdat

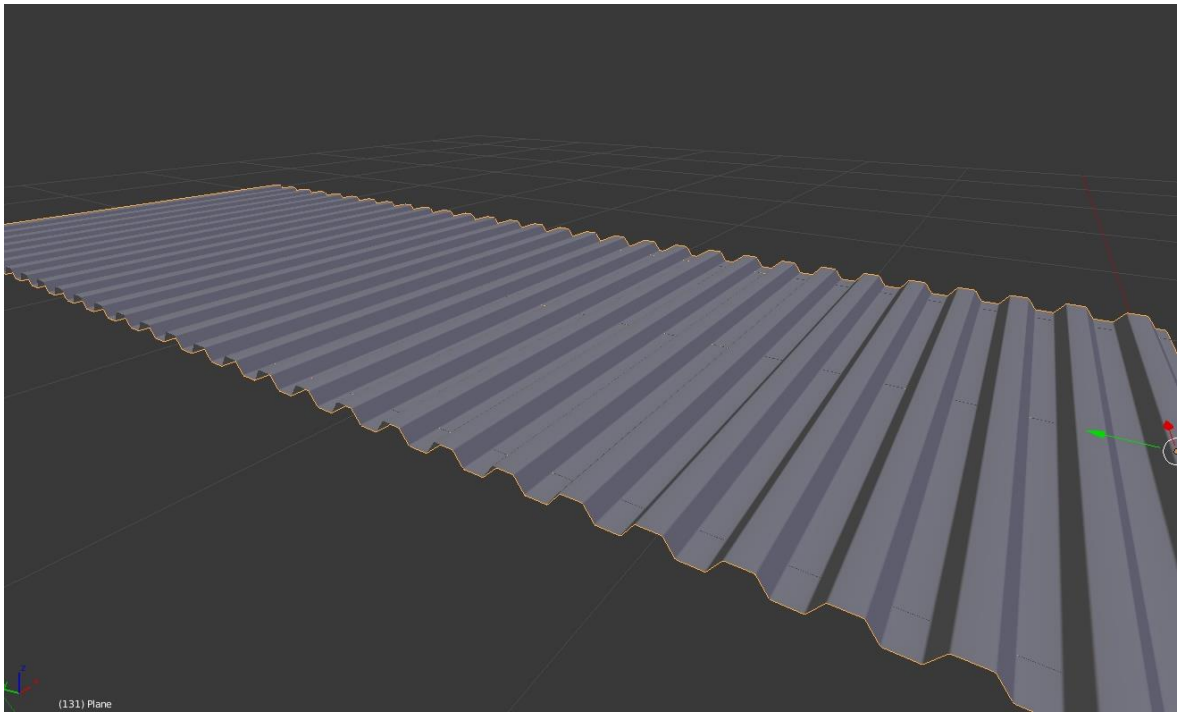
Koska hahmon mallinnustyö alkaa olla valmis, päätin keskittyä seuraavaksi pieniin yksityiskohtiin, jotka vaativat erityistä huomiota. Tarkoituksena on, että robotin silmä muistuttaa kameran linssiä. Linssin muoto on hyvin monimutkainen, eli tarvitaan yksityiskohtaista mallinnusta hyvin pienelle osalle hahmosta. On tietysti mahdollista liittää muoto suoraan itse malliin, mutta tavoitteena on luoda suorituskykyistä grafiikkaa. Onkin siis järkevämpää mallintaa yksityiskohtainen linssi erikseen ja luoda sen avulla normal map, joka antaa illuusion kolmiulotteisuudesta.

Linssin mallintaminen on melko yksinkertainen prosessi. Lisätään uusi sylinteri, jonka toiseen päätyyn mallinnetaan kameran linssiä muistuttavaa muotoa. Tärkein työkalu tässä prosessissa on i-näppäimellä toimiva upotustyökalu (**Inset**), jonka avulla saadaan luotua sylinterin keskiosaan lisää tahoja. Tahot skaalataan ja liikutellaan paikoilleen, jolloin lopputuloksena on epäsäännöllinen muoto, joka muistuttaa kameran linssiä. Mallinnuksen jälkeen pitää vielä muuttaa objektin varjostus (**Shading**) pehmeäksi (**Smooth**), jotta tahojen reunat eivät näy normal mapissa. Haluan kuitenkin, että linssissä olevat reunakohdat olisivat terävämpiä. Voisin merkitä käsin haluamani reunat teräviksi (**Mark Sharp**), mutta tässä tapauksessa käytän mielummin muuttujaa. Valitaan muuttujalistasta hiontamuuttuja (**Bevel**) ja muutetaan rajausmetodi (**Limit Method**) painoarvon mukaiseksi. Lisäksi annetaan muuttujan leveysarvoksi (**Width**) noin 0.0110 ja lohkoja (**Segments**) 4 kappaletta. Pienellä säätämisellä tuloksena on nätin näköinen kameran linssi, joka näkyy kuvassa 9.



Kuva 9. Linssin malli

Samaa periaatetta käyttäen luodaan myös erillinen objekti putkimaiselle muodolle, jonka on tarkoitus tulla hahmon kaulaan ja raajojen liitoskohtiin. Putken muoto on hyvin vaihteleva, joten sen mallintaminen suoraan hahmoon veisi paljon resursseja. Muoto saadaan aikaiseksi lisäämällä 2D-taso, jonka keskikohtaan leikataan kaksi lisäsärmää. Keskikohta korotetaan ylöspäin jonkin verran, jotta putken epätasainen muoto saadaan esiin. Skaalataan taso 0.1 kertaiseksi sen alkuperäisestä koosta, ja sen jälkeen venytetään se x-akselin suuntaisesti. Myös tämän objektin varjostus pitää muuttaa pehmeäksi, ja lisätään myös hiontamuuttuja, mutta hieman voimakkaampana. Lisätään lopuksi vielä ryhmämuuttuja (**Array**), joka luo objektista kopioita halutun määrän. Muutetaan muuttuja y-akselin suuntaiseksi ja kopioiden määräksi asetetaan 10 kappaletta. Näin objekti on valmis tekstuurien tekoa varten. Kuvassa 10 näkyy mallinnuksen lopputulos.



Kuva 10. Putken pintatekstuurin malli

8 Pelihahmon teksturointi

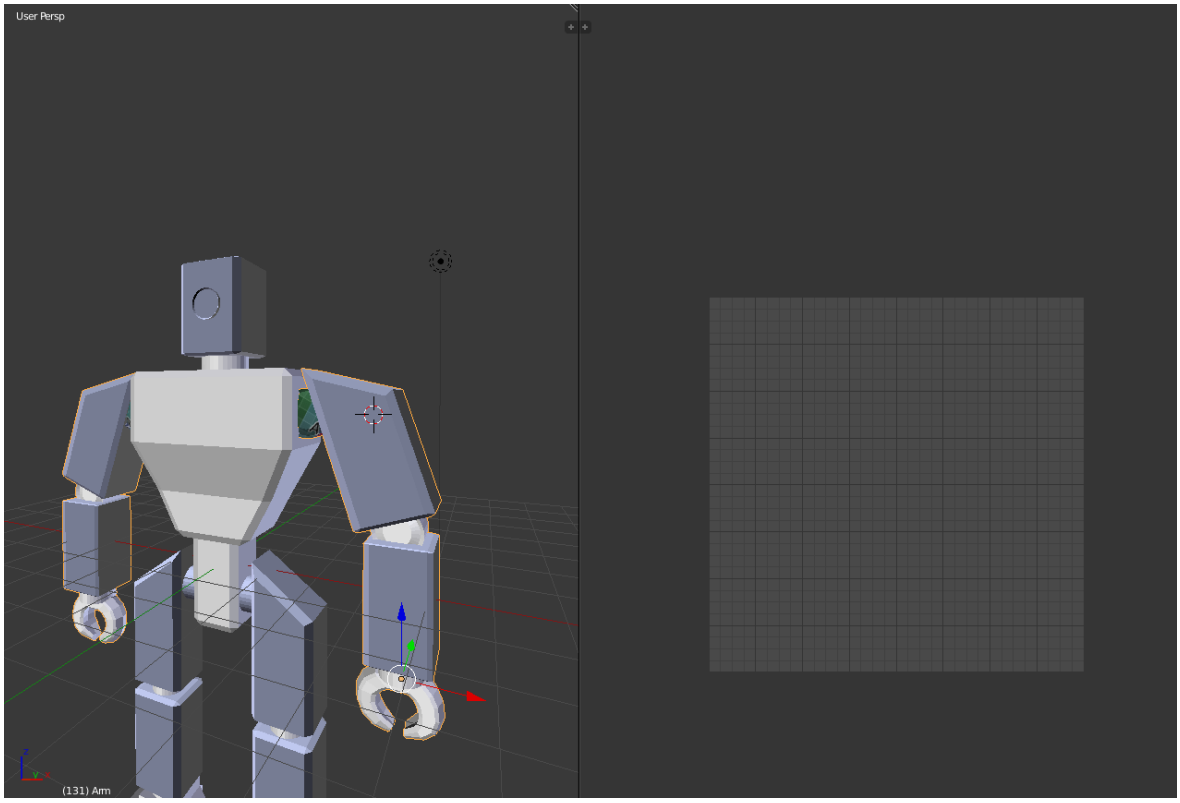
Teksturointivaiheessa tarkoituksena on luoda 2D-kuvia, jotka asetellaan mallin pinnalle, luoden näyttävämmän kokonaisuuden. Tässä vaiheessa käytetään hyödyksi Blenderin lisäksi myös Photoshoppia.

8.1 UV-kartoitus

Ennen kuin 3D-mallin teksturoinnin voi aloittaa, mallille pitää luoda UV-kartta. UV-kartta on kaksiulotteinen versio luodusta mallista, johon mallin tahot sijoitetaan purkamalla (**UV**

Unwrap). Blenderissä on muutamia erilaisia työkaluja, mitä voidaan käyttää hyödyksi mallin purkamisessa. Mikäli tietää mitä tekee, varmin tapa purkaa malli UV-muotoon on lisätä manuaalisesti mallin särmiin saumoja (**Seam**), jotka kertovat Blenderille mistä kohdasta mallia halutaan leikata. Saumojen asettelu onnistuu valitsemalla haluamansa särmän ja valitsemalla Mark Seam Ctrl+E pikavalikosta, mallin purkaminen onnistuu painamalla Unwrap U-näppäimen pikavalikosta. Yksinkertaisille objekteille saumojen asettelu on helppoa, mutta prosessi vaikeutuu mitä enemmän yksityiskohtia objektissa on, sen takia tämä työvaihe voi olla hyvin aikaa vievä ja tylsähkö prosessi. Hyvänä puolena mallin voi purkaa UV-kartalle niin monta kertaa kuin haluaa aiheuttamatta pysyviä muutoksia, eli työvaiheen voi suorittaa ainakin alkuun yrityksen ja erehdyksen kautta. Blenderistä löytyy myös automatisoitu työkalu UV-kartoittamiseen (**Smart UV Project**). Työkalu löytyy samasta U-näppäimen valikosta kuin Unwrap-komento, ja sen käyttö voi olla helpompaa ja nopeampaa kuin manuaalinen saumojen lisääminen.

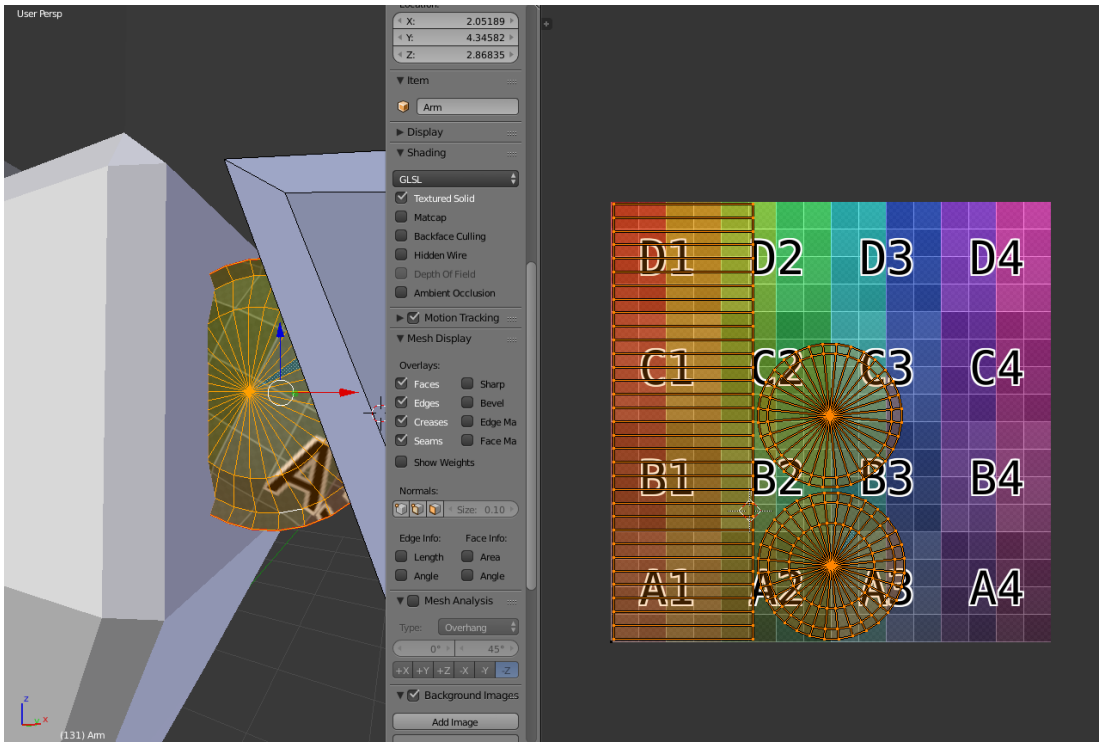
UV-kartoituksen aluksi lisään Blenderin 3D-näkymän (**3D-view**) lisäksi toisen näkymän, jonka tyypiksi valitsen UV/kuvaeditorin (**UV/image editor**). Näkymäikkunan voi jakaa kahdeksi vetämällä näkymäikkunan oikeassa yläkulmassa olevasta nuolesta, ja näkymät voi yhdistää vetämällä vasemmassa alakulmassa olevasta nuolesta. Kaksi näkymää helpottaa UV-kartoitusprosessia huomattavasti, sillä näkymät on suunniteltu toimimaan yhdessä mahdollisimman hyvin, kuten kuvassa 11 näkyy.



Kuva 11. Kaksi näkymää Blenderissä

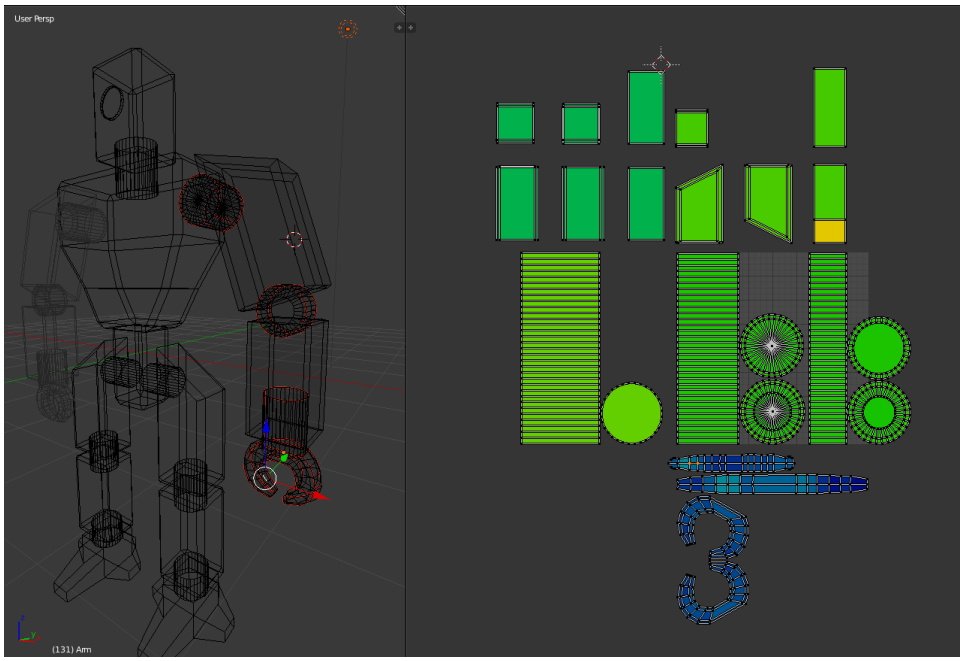
Päätin aloittaa luomani mallin purkamisen UV-kartalle käsiobjektista, sillä se vaikutti helpoimmalta aloituskohteelta. Käytännössä ei pitäisi olla suurta merkitystä mistä kohdasta mallia purkamisen aloittaa, koska objektien paikkaa UV-kartalla voi muokata milloin vain. Ensimmäisenä puran hahmon olkapäässä olevan sylinterin käyttämällä apuna aiemmin mainittua automaattityökalua. Työkalulle voidaan antaa 3 eri arvoa, joista tärkein on kulman rajoitus (**Angle Limit**). Mitä suuremman arvon tähän antaa, sen vähemmän vääristymää ja yhtenäisempiä saarekkeita luodulle UV-kartalle tulee. Joskus on kuitenkin hyödyllisempää olla enemmän saarekkeita, joten työkalu kannattaa ajaa muutamaan kertaan ja valita paras tulos. Käyttäessäni työkalua huomasin, että UV-kartalle purettu kuva vaikuttaa oudolta ja epäkäytännölliseltä. Ongelmana oli käsiobjektin yhdistämisessä luotu skaalausarvo, mikä korjaantui valitsemalla objektimoodissa käsiobjekti ja painamalla Ctrl+A valikosta Apply Scale, minkä jälkeen työkalun tuottama UV-kuva oli kunnollinen.

Automaattityökalun tuottama jälki on usein melko hyvää, mutta parannuksen varaa on yhä. UV-karttaa voidaan parannella valitsemalla UV-editorissa yksittäisiä särmiä ja ommella (**Stitch**) niitä kiinni toisiinsa V-näppäimellä, jotta UV-kartan saarekkeista saadaan yhtenäisempiä kokonaisuuksia. Tässä vaiheessa kannattaa myös UV-editori ja 3D-näkymä synkronoida UV-editorin työkalupalkissa sijaitsevasta napista. Synkronoinnin avulla molemmissa näkymissä näkyy korostettuna valitut kärjet, särmät tai tahot, jolloin työn kokonaiskuva on helpompi hahmottaa. UV-saarekkeiden ompelun lopuksi voidaan valita UV-editorissa saumojen luonti saarekkeista (**Seams from islands**), jolloin saarekkeiden reunoista generoidaan objektiin täsmäävät saumat. Tämän jälkeen objekti puretaan vielä kerran, jolloin UV-kartalle muodostuu hyvin käyttökelpoinen kuva. Kannattaa vielä luoda UV-editoriin uusi kuva, jonka resoluutioksi syötetään 512x512 pikseliä ja tyypiksi väriruudukko (**Color Grid**), joka näkyy kuvassa 12. 3D-näkymässä pitää lisäksi muuttaa varjostusasetuksista päälle teksturoitu kiinteä näkymä (**Textured Solid**), jolloin luotu väriruudukkotekstuuri näkyy objektin päällä. Ruudukko on hyödyllinen työkalu, jonka avulla näkee helposti valmiiden tekstuurien koon ja esimerkiksi tekstuurien venymisongelmat.



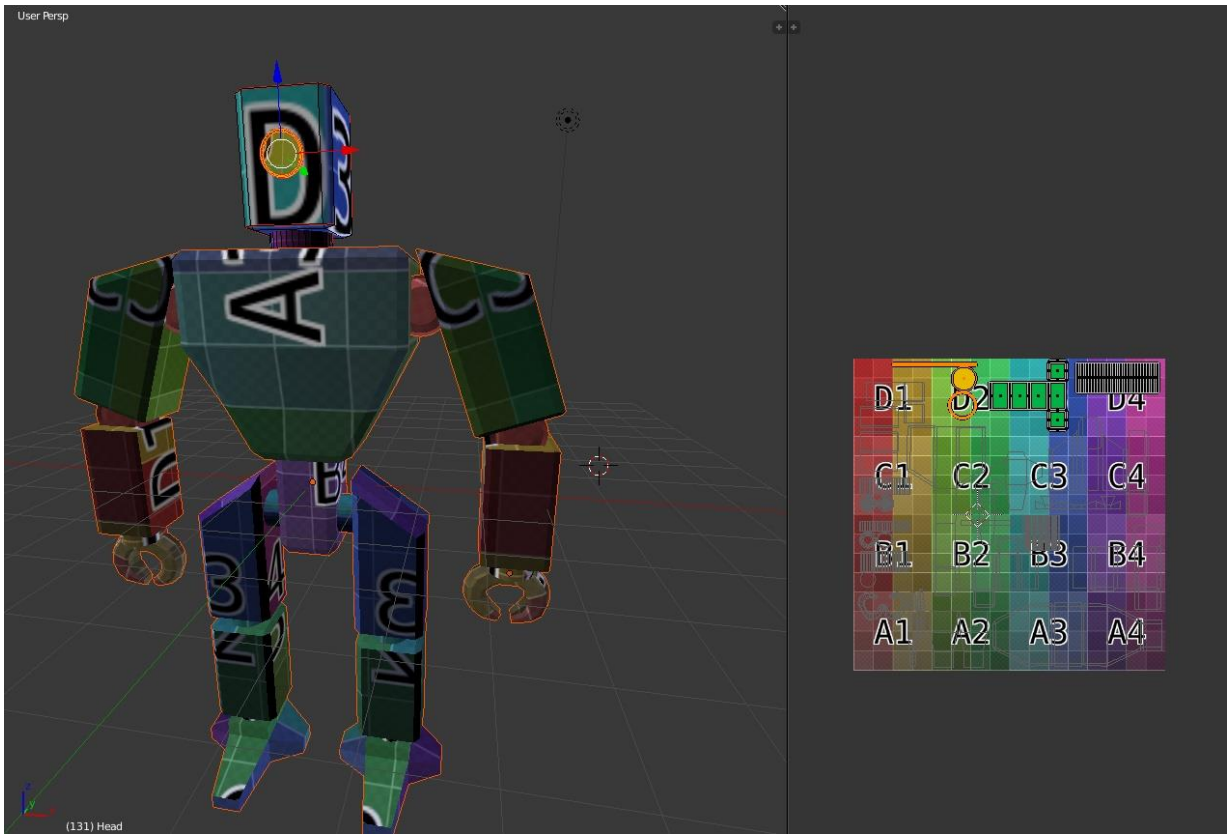
Kuva 12. UV-kartoituksen alku ja väriruudukko

Jatkoin käden UV-kartan purkamista aiemmin mainituilla tekniikoilla. Huomasin, että yksinkertaisemmat laatikkomaiset osat kädestä olivat helppo purkaa automaattityökalun avulla ilman ongelmia. Monimutkaisemmat osat, kuten sylinterit ja käden koukut vaativat hieman enemmän ajattelua ja saumojen asettelua loogisiin kohtiin. Tässä työvaiheessa on suureksi avuksi olla tarkka kolmiulotteinen hahmotuskyky, jotta osaa hahmottaa miten 3D-malli muutetaan 2D-muotoon saumojen avulla.



Kuva 13. Käden purettu UV

UV-kartan purkuvaiheessa ei ole vielä tarpeen skaalata ja asetella saarekkeita kovin tarkasti väriruudukkoon, mutta kannattaa silti asetella palaset suurin piirtein sellaisiin kohtiin, jotta ne muistuttavat 3D-mallin osia. Järkevällä asettelulla on helpompi hahmottaa miten UV-kartan koordinaatit vastaavat 3D-mallia. Lopullinen palojen asettelu kannattaa tehdä vasta kun koko malli on purettu. UV-kartoituksen haastavuus piilee juuri palojen asettelussa kartalle. Ideana on, että mahdollisimman vähän tekstuurin pinta-alasta menisi hukkaan, mutta myös että mallin näkyvimmät ja tärkeimmät osat saisivat mahdollisimman paljon tilaa tekstuurista. Mitä enemmän tilaa jollekin 3D-mallin osalle varataan UV-kartalta, sen tarkempi ja yksityiskohtaisempi tekstuuri voidaan luoda siihen kohtaan. Käytin hyväkseni palojen asettelussa työkalua, joka värikoodaa (**Stretch**) palaset UV-kartalla riippuen niiden koosta itse 3D-mallissa, kuten näkyy kuvassa 13. Työkalun avulla on helppo nähdä, mikäli palanen on liian iso tai pieni. Työvaiheen lopuksi 3D-malli on päällystetty väriruudukon tekstuurilla, eli tekemäni UV-kartoitus toimii. Kuvassa 14 näkyy toimivan UV-kartan vaikutus.



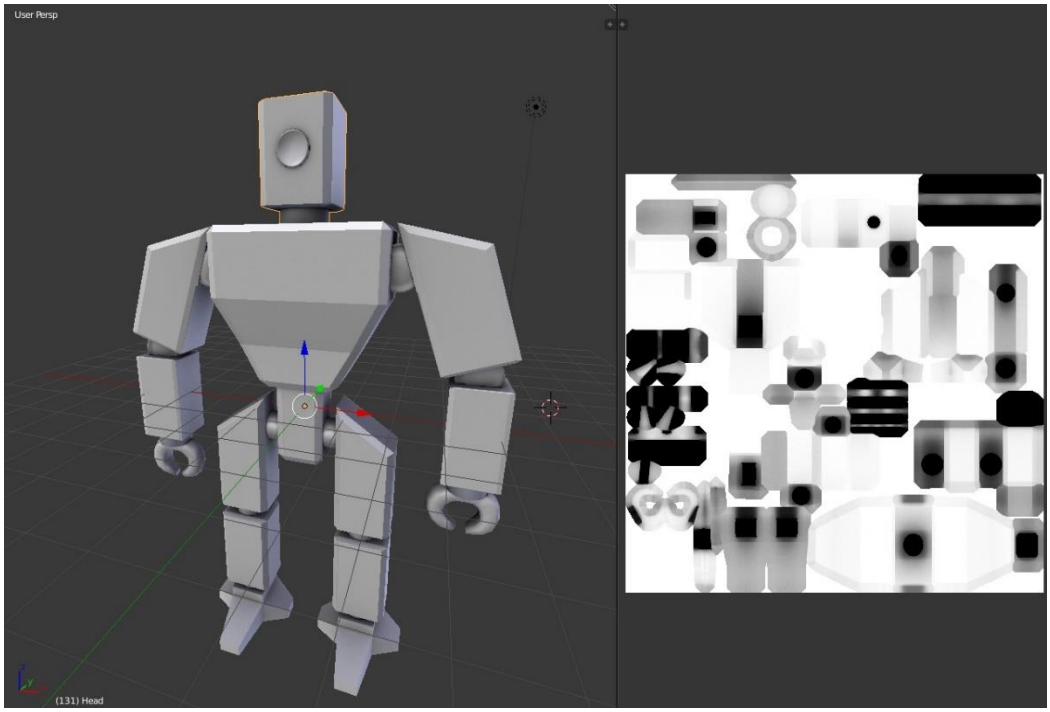
Kuva 14. Toimiva UV-kartta

8.2 Pohjatekstuurin luonti Blenderillä

Nyt kun UV-kartta on määritetty, seuraavassa työvaiheessa luodaan hahmolle pohjatekstuuri, jonka pohjalta saadaan luotua myöhemmin yksityiskohtaisempi tekstuuri. Uuden

tekstuurin lisääminen tapahtuu UV-editorin Image-valikosta painamalla ”New Image”. Luodaan uusi 512x512 pikselin kokoinen kuva, jonka väriksi valitaan valkoinen, sekä määritetään uusi tekstuuri kaikille objekteille valitsemalla ne 3D-näkymässä ja valitsemalla UV-editorissa luotu tekstuuri. Tekstuurin voi tallentaa suoraan blend-tiedostoon valitsemalla samasta valikosta ”Pack as PNG”. Tässä vaiheessa huomasin, että jotkut osat mallista näyttivät valaistuvan eri tavalla, vaikka käytössä ei ole materiaaleja tai muuta mikä voisi aiheuttaa sen. Ongelman syynä oli jostain syystä väärin päin oleva valaistusinformaatio (**Normals**), mikä on helposti korjattavissa valitsemalla ongelmalliset tahot Edit Modessa ja valitsemalla W-napin valikosta ”Flip Normals”. Lisäksi tässä vaiheessa muutin mallin valaistustyylin pehmeäksi (**Smooth**), koska muuten esimerkiksi normal-mapin luonti ei onnistu. Valaistustyylin voi muuttaa painamalla 3D-näkymän vasemmasta työkalupaneelista Smooth shading.

Pohjatekstuurin luonti Blenderillä onnistuu valitsemalla haluttu osa mallista, ja menemällä oikeanpuoleisen työkalupaneelin Render-osioon, ja sieltä Bake-työkalun valikoihin. Valitaan moodiksi ”Ambient Occlusion”, sillä aluksi haluamme luoda kevyen varjoefektin mallille. Valitaan ”Normalize” ja poistetaan täppä ”Clear” valinnasta, sillä emme halua, että työkalu poistaa kaiken tekstuurista, kun se ajetaan. Valitaan ”Margin” valikosta 8 pikseliä, jotta efekti varmasti ulottuu haluttuihin kohtiin. Kun valinnat on tehty, voidaan painaa Bake-painikkeesta, ja tarkastella millaista jälkeä Blender saa aikaiseksi. Tuloksen pitäisi olla melko kevyt, mutta silti huomattava, kuten kuvassa 15.

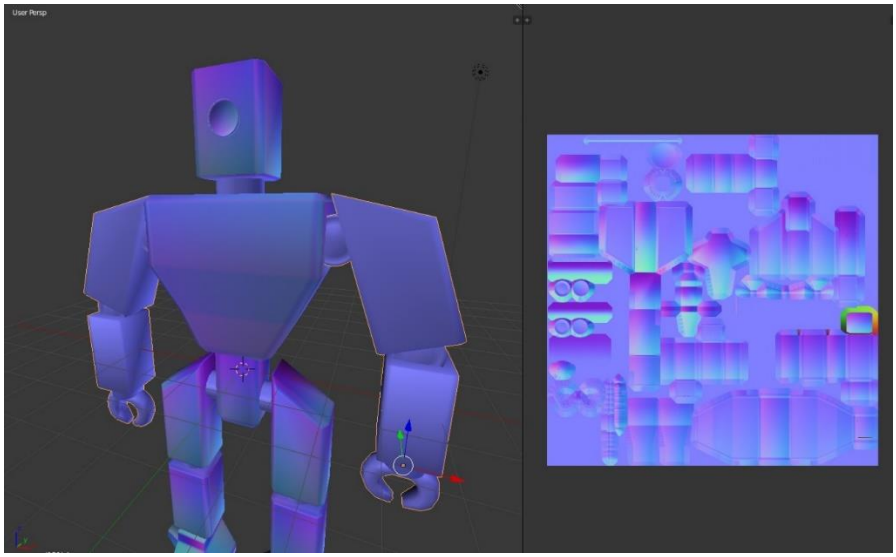


Kuva 15. Varjotekstuuri

8.3 Normal-mapin luonti Blenderillä

Normal-mapin tarkoituksena on lisätä malliin lisää yksityiskohtia ilman, että käytetään lisää polygoneja, jolloin mallin suorituskyky on parempi. Tämä työvaihe alkaa melkein samoin kuin edellinen, eli lisätään uusi tekstuuri resoluutiolla 512x512 pikseliä. Tekstuurin väriksi valitaan tällä kertaa 0.5 punainen, 0.5 vihreä ja 1.0 sininen. Tämä väri määritetty normal-mapin neutraaliksi väriksi, eli se ei aiheuta muutoksia mallin ulkonäköön.

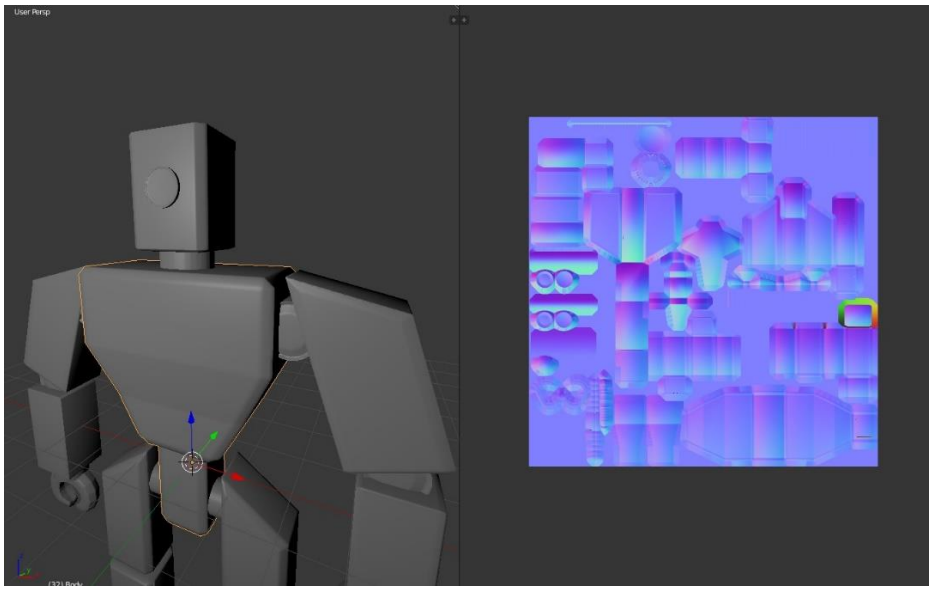
Normal-mapin hyöty tulee kunnolla esille vasta kun sen luomiseen käytetään hi-poly mallia, joten luodaan sellainen. Valitsin kaikki mallin osat ja tein niistä kopion Shift+D näppäinyhdistelmällä. Jotta mallista saadaan korkeatasoisempi, käytin apuna Bevel-muuntajaa, joka löytyy oikeasta työkalupaneelista Modifiers-välilehden alta. Bevel pyöristää mallin reunoja ja tekee siitä yksityiskohtaisemman näköisen. Tämän jälkeen on aika luoda itse normal-map. Se onnistuu lähes samalla tavalla kuin pohjatekstuurin luonti, mutta Bake Mode pitää muuttaa vaihtoehtoon ”Normals”. Tärkeää on tässä kohdassa, että hi-poly malli on tasan low-poly mallin päällä, muuten tekstuuri ei täsmää malliin. Tärkeää on myös valita ensin hi-poly ja vasta sitten Shift-klikkauksella low-poly malli, jolloin normal-map luodaan hi-poly mallin mukaan.



Kuva 16. Väärin toimiva normal-map

Normal-map on nyt luotu, mutta Blender pitää sitä tavallisena tekstuurina (Kuva 16), eli malli värjäytyy erilaisin sinisin ja violetein sävyin, eikä mallin pinnalla näy merkkiäkään syvyydestä. Normal-mapin tekstuuri pitääkin määritellä erikseen, jotta Blender osaa käyttää sitä oikein. Määrittelyn voi tehdä oikealta työkalupaneelista ”Textures”-välilehdestä. Lisätään uusi tekstuuri, jonka tyypiksi määritetään ”Image or Movie”. Image-valikosta määritellään käytettäväksi tekstuuriksi luotu normal-map. Influence-valikosta määritellään lisäksi tekstuurin vaikuttavan vain kohtaan ”Normal”. Viimeiseksi pitää vielä määritellä kohdasta

”Mapping” tekstuurin koordinaateiksi UV-kartta ja vielä ”Image Sampling”-kohdasta ruksi kohtaan ”Normal Map”. Kun kaikki nämä määritykset on tehty, tekstuuri toimii kuten on tarkoitus, eli se vaikuttaa vain mallin pinnan muotoihin, eikä väreihin (Kuva 17).

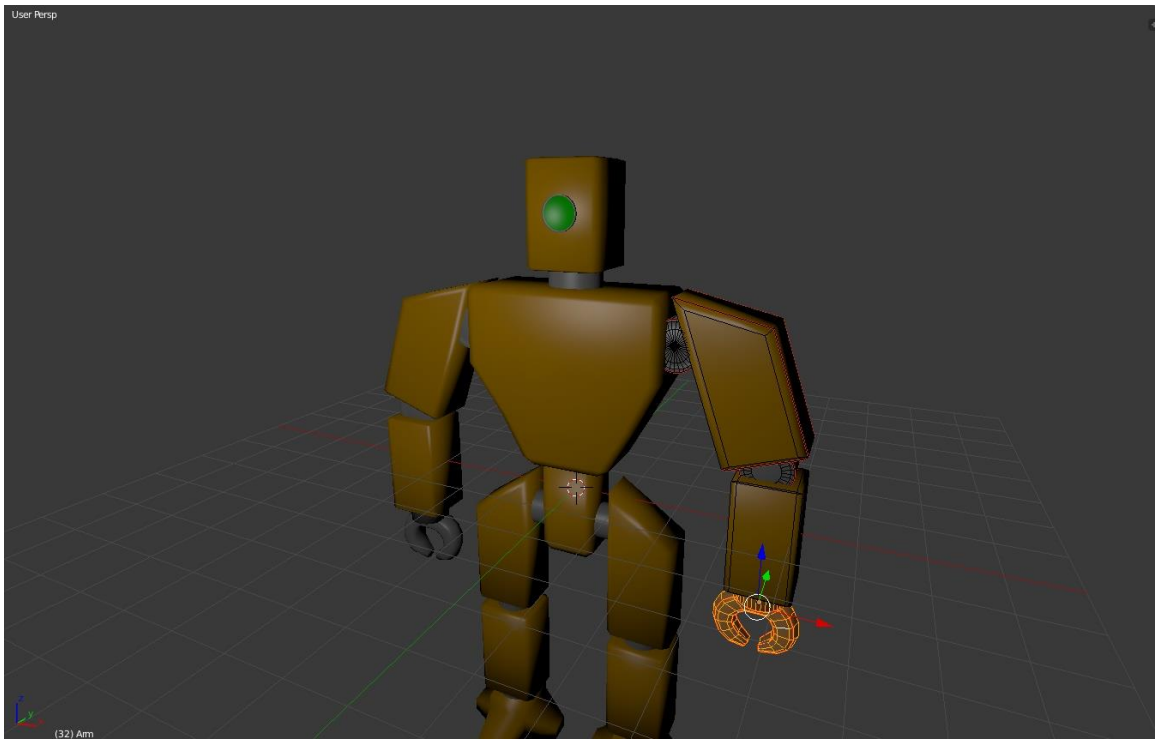


Kuva 17. Oikein toimiva normal-map

8.4 Pohjavärien luonti Blenderillä

Ennen kuin lopullista tekstuuria alkaa tekemään, on kätevää luoda mallille pohjavärit Blenderin avulla. Tämä helpottaa hahmottamaan miltä lopullinen tuotos tulee näyttämään, ja se antaa myös kätevän pohjan mistä 2D-tekstuuria voi alkaa muokata.

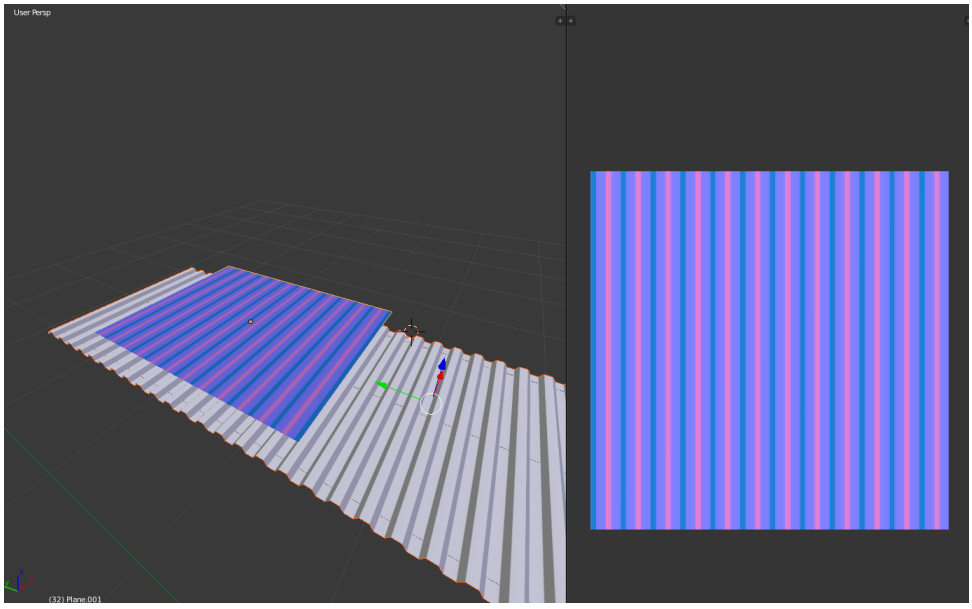
Pohjavärin lisääminen on helppoa, tarvitsee vain valita halutut tahot, ja lisätä uusi materiaali oikean työkalupaneelin Materials-välilehdeltä. Materiaali pitää vielä määrittää halutuille tahoille painamalla Assign-painiketta editointitilassa. Materiaalit ovat hyvin moniulotteinen työkalu, mutta tässä työvaiheessa kiinnostuksen kohteenamme on ainoastaan materiaalin väri (**Diffuse color**). Paletista voi valita materiaalille minkä värin tahansa, ja myös värin intensiteettiä voi muuttaa halutessaan. Valitsin omalle robottihahmolleni yksinkertaisen kelta-harmaan väriteeman, joka saa aikaan vaikutelman työmaakoneesta. Robotin silmän väriksi valitsin kirkkaanvihreän (Kuva 18).



Kuva 18. Pohjavärit mallin pinnalla

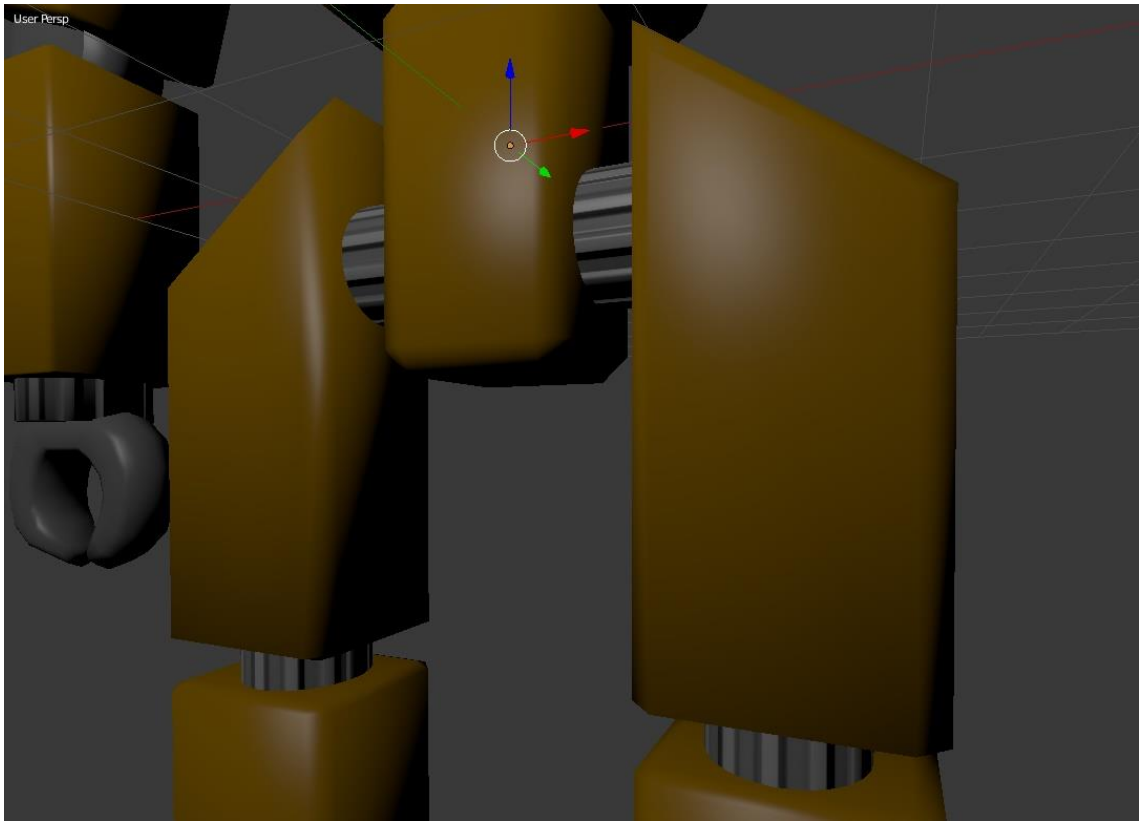
8.5 Yksityiskohtien tekstuurit

Mallinsin aikaisemmin hahmon silmää ja raajojen liitoskohtia varten yksityiskohtaiset mallit, joiden avulla voidaan luoda näihin kohtiin tekstuurit. Tekstuurien luonti onnistuu yksinkertaisesti luomalla uusi litteä taso mallin päälle, ja painamalla tekstuurin luontinappia, kuten edellisissä vaiheissa (Kuva 19). Tärkeää on muistaa tehdä näitä tekstuureja varten uudet imaget, jotta vanhat tekstuurit säilyvät.



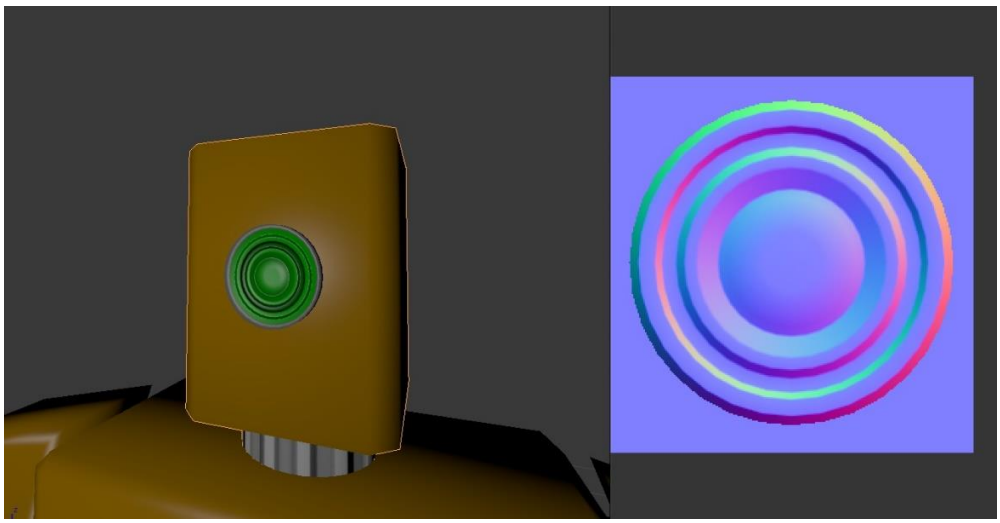
Kuva 19. Putkien normal-mapin luonti

Jotta nämä uudet tekstuurit saadaan käyttöön hahmolle, tarvitaan uusi materiaali jokaiselle eri objektille, johon tekstuurit halutaan asettaa. Tarvitaan myös erilliset UV-mapit, jotka ottavat tekstuuri-informaation uusista tekstuureista. Määritetään materiaalit käyttämään uusia tekstuureita (Kuva 20).



Kuva 20. Normal-mapin vaikutus

Raajojen lisäksi käydään sama prosessi läpi myös kamerasilmän kanssa, lopputuloksena saadaan kiinnostavan näköinen yksityiskohta (Kuva 21).



Kuva 21. Silmän normal-map

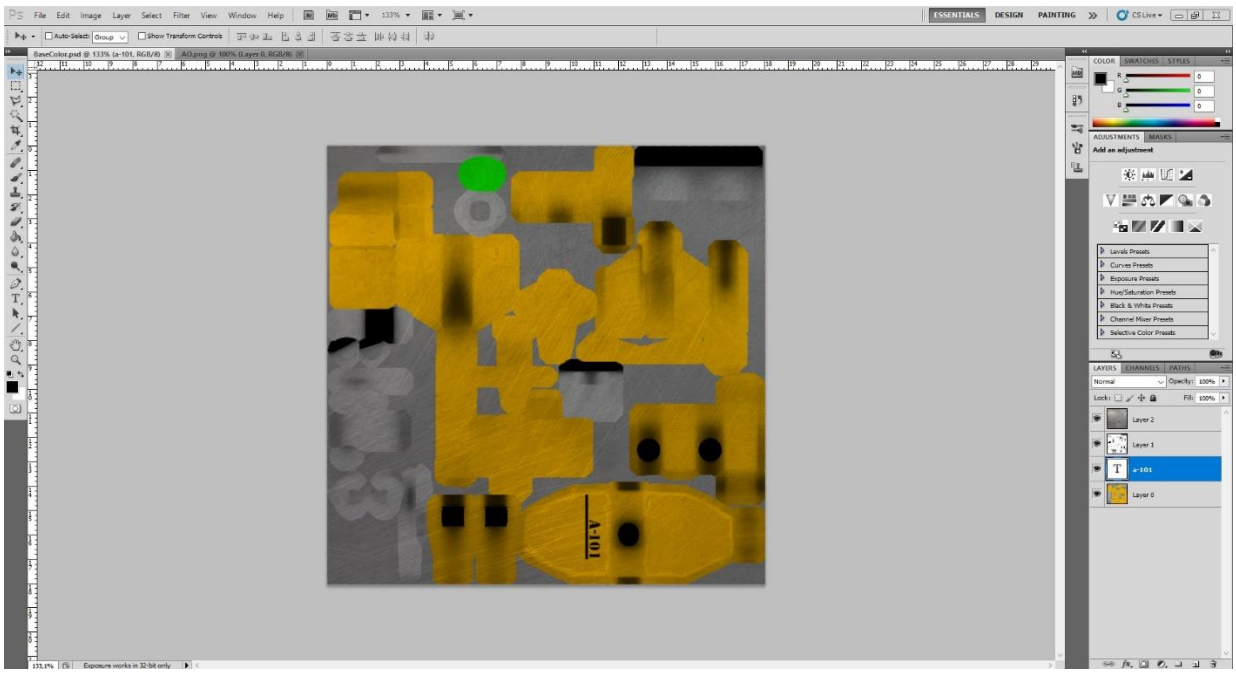
8.6 Tekstuurien viimeistely Photoshopilla

Tässä vaiheessa työtä oli aika siirtyä toisen ohjelman pariin, ja viimeistellä tehdyt tekstuurit. Loin kopiot Blenderissä luoduista tekstuureista .png-muodossa, jotta voisin avata ne Photoshopissa, ja jotta alkuperäiset tekstuurit eivät vahingossa joudu ylikirjoitetuksi.

Ensimmäisenä avasin Photoshopissa varjostustekstuurin ja pohjaväritekstuurin, ja kopioin ne samaan kuvaan, mutta eri kerroksille. Jotta varjostukset saadaan järkevästi tuotua pohjavärien päälle, pitää varjostuskerros olla pohjavärikerroksen päälle, ja varjostuskerroksen sekoitusmoodi (**Blending mode**) muutetaan vaihtoehtoon ”Multiply”. Tämän moodin ansiosta suurimmaksi osaksi valkoinen tausta ei näy pohjavärien päällä, vain varjostukset näkyvät.

Halusin robotin olevan kuluneen ja vanhan näköinen, joten latasin netistä ilmaistekstuurin, joka esittää kulunutta metallipintaa. Kopioin metallitekstuurin samaan kuvaan, ja muutin myös sen moodin ”Multiply”-vaihtoehtoon, ja laskin vielä läpinäkyvyyttä, jotta tekstuuri ei ole liian voimakas. Tämän jälkeen lisäsin vielä robotin rintaan mustan raidan, sekä Photoshopin tekstityökalulla robotin nimen ”A-101” (Kuva 22).

Photoshopissa tehdyt projektit kannattaa aina tallentaa .psd-muodossa, koska se säilyttää projektissa käytetyt kerrokset ja muut säädöt ennallaan, jotta projektia voi halutessaan jatkaa suoraan siitä mihin jäi.



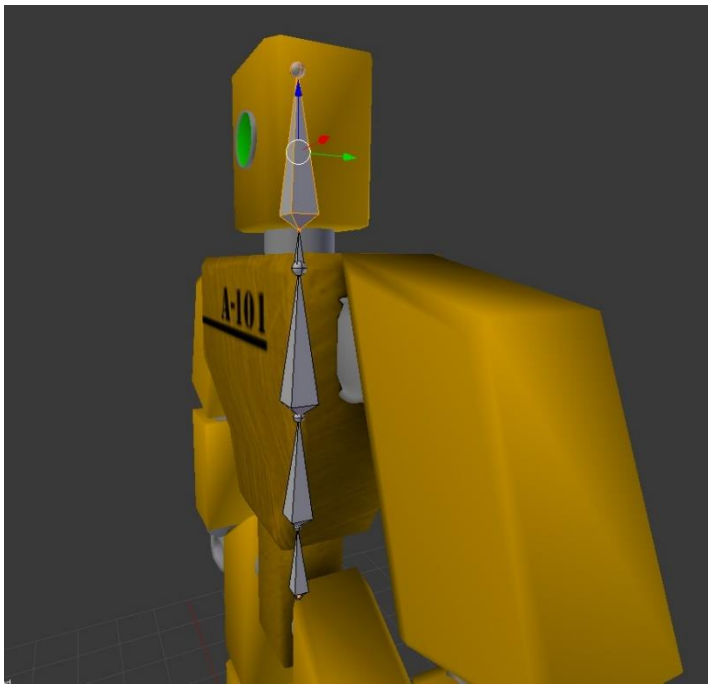
Kuva 22. Viimeistelty tekstuuri Photoshopissa

9 Pelihahmon luuranko

Seuraavassa työvaiheessa tarkoituksena on luoda pelihahmolle luuranko (**Skeleton/Rig**). Luurangon ideana on antaa animaattorille työkalut, jonka avulla hahmolle voidaan luoda animaatioita pelejä tai videota varten. Peleissä luurankoa voidaan käyttää myös muihin tarkoituksiin, kuten mallin fysiikoiden laskentaan tai dynaamiseen muovaukseen (**Deform**). Mitä enemmän ominaisuuksia mallille halutaan, sen haastavampi tehtävä luurangon luonnista tulee.

Aloitin hahmon luurangon luomisen ottamalla oikeanpuoleisen näkymän hahmosta, ja sijoittamalla ensimmäisen luun hahmon jalkojen tasolle. Uuden luun saa luotua painamalla Shift+A ja valitsemalla luun armature-valikosta. Tässä vaiheessa kannattaa myös kytkeä päälle luurangon X-Ray, jotta luut näkyvät hahmomallin läpi.

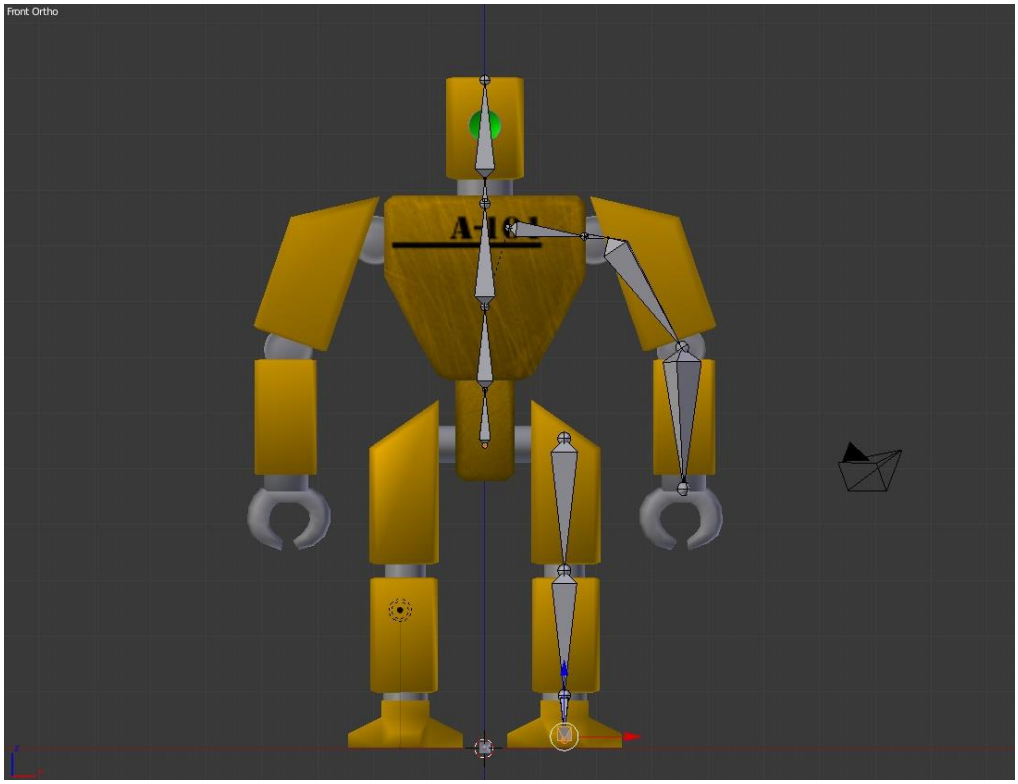
Seuraavaksi kopioin ensimmäisen luun painamalla Shift+D, ja sijoitin kopion hahmon lantion kohdalle. Tämän jälkeen jatkoin luiden lisäämistä valitsemalla lantioluun kärjen ja painamalla E-näppäintä. Tarkoituksena tässä vaiheessa on luoda hahmon tukiranka (Kuva 23).



Kuva 23. Perusranka

Seuraavaksi vuorossa on raajojen luiden luonti, mikä tapahtuu samalla periaatteella. Tärkeää on kuitenkin myös nimetä luut niiden sijainnin perusteella, ja varsinkin raajojen luissa

nimen perään kannattaa kirjoittaa kumman puolen luita ne ovat, sillä luut on tarkoitus myös peilata hahmon ruumiin toiselle puolelle (Kuva 24).



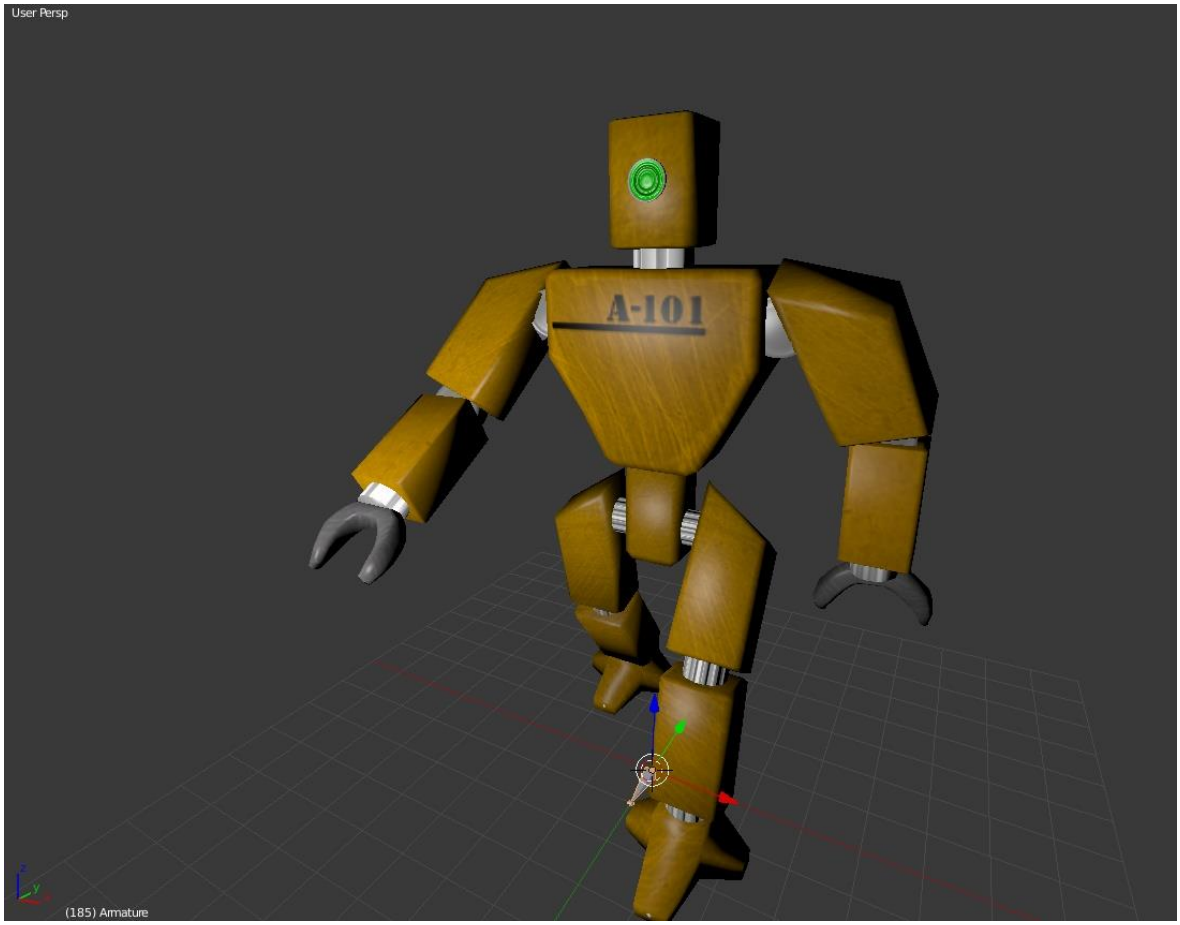
Kuva 24. Raajojen luut

Kun raajojen luut on luotu, on aika peilata ne myös hahmon ruumiin toiselle puolelle. Tämä onnistuu kopioimalla raajan luut näppäinyhdistelmällä Shift+D, ja valitsemalla 3D-näkymän armature-valikosta ”Mirror > X Local”. Tässä vaiheessa on tärkeää, että 3D-kursori on hahmon keskellä, jotta peilaus onnistuu halutulla tavalla.

Peilauksen jälkeen luurangon muoto on valmis, joten voidaan siirtyä luurangon ja mallin yhteistoimintaan. Toimiakseen oikein luurangon luut pitää määrittää liikuttamaan tiettyä osaa mallista. Luut voi määritellä täysin manuaalisesti, mutta Blenderissä on myös automaattinen toiminto tätä tarkoitusta varten. Operaation voi suorittaa valitsemalla ensin mallin, sitten luurangon ja painamalla Ctrl+P. Avautuvasta valikosta valitaan ”Armature Deform > With Automatic Weights”, jolloin Blender yrittää määrittää luut täsmäämään mallin osiin.

Automaattityökalu ei kuitenkaan ole täydellinen, ja esimerkiksi omassa mallissani se jätti osia raajoista ilman luita. Ongelman voi korjata valitsemalla haluamansa osan, ja navigoimalla ”Object Data” valikoihin. Valitulle osalle pitää määrittää oikea ”Vertex Group”, eli käytännössä valitaan listasta haluttu luu, johon osa liitetään.

Luiden toimintaa pääsee testaamaan valitsemalla luurangon ja painamalla Ctrl+Tab, jolloin Blender siirtyy poseerausmoodiin (**Pose Mode**). Tässä tilassa luita liikuttamalla voi muuttaa hahmon asentoa ja tarkastella luiden toimivuutta (Kuva 25).



Kuva 25. Valmis malli poseerattuna luurangon avulla

Tuloksena syntyi yksinkertainen, mutta toimiva luuranko. Tämän luurangon avulla on mahdollista tuottaa ainakin yksinkertaisia animaatioita hahmolle.

10 Pohdinta

Tuloksena opinnäytetyöstä syntyi 3D-hahmo, jonka luomisprosessissa käytettiin yleisiä peliteollisuuden menetelmiä. Pelkän mallin lisäksi luotiin myös erilaisia tekstuureja ja luuranko, jonka avulla mallille voidaan luoda animaatioita. Pelihahmo on siis viimeistelty animaatioita lukuun ottamatta, ja sen voisi viedä pelimoottoriin, mikäli animaatiot olisi luotu. Tämä lopputulos vastaa alkuperäistä suunnitelmaa, mutta pelihahmon vienti pelimoottoriin piti rajata pois työstä sen monimutkaisuuden takia.

Opinnäytetyöprosessi oli kiinnostava ja työläs, sillä 3D-mallinnus on hyvin monimutkainen ja laaja aihe. Huomasin opetellessani Blenderin käyttöä, että perustaidot, kuten 3D-näkymässä navigointi ja mallinnuksen perustyökalut vievät melko paljon aikaa opetella, mutta sen jälkeen ohjelman käyttö on sujuvaa. Opinnäytetyön lopussa olin käyttänyt Blenderiä noin 100 tuntia, minkä aikana olen oppinut paljon ohjelman toiminnasta, mutta tiedän että opittavaa olisi vielä valtavasti.

Jatkona opinnäytetyölle voisi selvittää jo edellä mainittuja animaatioita ja mallin viemistä pelimoottoriin. Varsinkin pelimoottorit olisivat kiinnostava aihe, sillä kaikissa pelimoottoreissa on pieniä eroja, joita varten mallit pitää valmistella ennen kuin ne toimivat oikein. Lisäksi myös tekstuurien luontiin voisi syventyä enemmän, koska opinnäytetyössäni esitellyt tekniikat ovat lähinnä pintaraapaisu aiheesta.

11 Lähteet

Aidy Burrows 2015. Blender Game Asset Creation. Luettavissa: https://cloud.blender.org/p/game-asset-creation/

Blender.org 2016. Luettavissa: https://www.blender.org/about/

Digital-Tutors Team 2013. Key 3D Modeling Terminology Beginners Need to Understand. Luettavissa: http://blog.digitaltutors.com/basic-3d-modeling-terminology/

Eddie Russell 2014. Understanding the Difference between Texture Maps. Luettavissa: http://blog.digitaltutors.com/understanding-difference-texture-maps/

Eddie Russell 2014. Eliminate Texture Confusion: Bump, Normal and Displacement Maps. Luettavissa: http://blog.digitaltutors.com/bump-normal-and-displacement-maps/

Eyal Kalderon 2011. Game engines: What they are and how they work. Luettavissa: https://nullpwd.wordpress.com/2011/05/09/game-engines-what-they-are-and-how-they-work/

Henry Lowood 2014. Game Engines and Game History. Luettavissa: http://www.kinephanos.ca/2014/game-engines-and-game-history/

Mark Masters 2014. Unity, Source 2, Unreal Engine 4, or CryENGINE - Which Game Engine Should I Choose? Luettavissa:
http://blog.digitaltutors.com/unity-udk-cryengine-game-engine-choose/

Tristan Donovan 2010. Replay: The History of Video Games. Yellow Ant, Lewes, United Kingdom

Wikibooks 2016. Blender 3D: Noob to Pro.
https://en.wikibooks.org/wiki/Blender_3D:_Noob_to_Pro

Wikipedia 2016. Source Engine. Luettavissa:
https://en.wikipedia.org/wiki/Source_(game_engine)

Wikipedia 2016. Game Engine. Luettavissa:
https://en.wikipedia.org/wiki/Game_engine

Wikipedia 2016. Blender. Luettavissa:
https://en.wikipedia.org/wiki/Blender_(software)

Wikipedia 2016. Alpha Mapping. Luettavissa:
https://en.wikipedia.org/wiki/Alpha_mapping

Wikipedia 2016. UV Mapping. Luettavissa:
https://en.wikipedia.org/wiki/UV_mapping

Wikipedia 2016. 8-bit color. Luettavissa:
https://en.wikipedia.org/wiki/8-bit_color

Wikipedia 2016. High color. Luettavissa:
https://en.wikipedia.org/wiki/High_color

Wikipedia 2016. Material. Luettavissa:
https://en.wikipedia.org/wiki/Material_(computer_graphics)

Wikipedia 2016. History of Video Games. Luettavissa:
https://en.wikipedia.org/wiki/History_of_video_games