Pekka Heikkilä

Yhteisnäkymällinen tietokoneavusteinen suunnitteluohjelma

Opinnäytetyö
Kevät 2018
SeAMK Tekniikka
Automaatiotekniikka

SeAMK
SEINÄJOEN AMMATTIKORKEAKOULU
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö: Tekniikka
Tutkinto-ohjelma: Automaatiotekniikka
Suuntautumisvaihtoehto: Koneautomaatio
Tekijä: Pekka Heikkilä

Työn nimi: Yhteisnäkymällinen tietokoneavusteinen suunnitteluohjelma

Ohjaaja: Tapio Hellman, Petteri Mäkelä

Vuosi: 2018 Sivumäärä: 49 Liitteiden lukumäärä: 7

Opinnäytetyön tavoitteena oli tuottaa Blender Game Enginellä rakennetun tietokoneavusteisen suunnitteluohjelman prototyyppiin lisää ominaisuuksia. Kaikkien käyttäjien näkymissään tekemien muutosten tuli synkronoitua kaikkien muiden käyttäjien näkymiin. Näkymästä valittavan kappaleen muototiedot tuli pystyä tallentamaan STL-tiedostoon. Ominaisuudet lisättiin onnistuneesti kirjoittamalla Python 3-ohjelmointikielellä aliohjelmia suunnitteluohjelmatiedostoon ja erillisinä ohjelmasäikeinä ajettaviksi tiedostoiksi. Ohjelman toimintaa havainnollistettiin UML 2.0 -kaavioiden avulla. Ohjelman prototyypillä generoitua STL-tiedostoa käyttämällä valmistettiin kappale 3D-tulostimella.

Avainsanat: Joukkoistaminen, 3D-tulostaminen, tietokoneavusteinen suunnittelu, stereolitografia, pelillistäminen, verkkosovellukset

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: Technology, Communications and Transport
Degree programme: Automation Engineering
Specialisation: Machine Automation
Author: Pekka Heikkilä
Title of thesis: Shared-view computer-assisted design software
Supervisor(s): Tapio Hellman, Petteri Mäkelä
Year: 2018 Number of pages: 49 Number of appendices: 7

The aim of this bachelor's thesis was to implement new functionality to a computer assisted design software prototype built for the Blender Game Engine (BGE). The new functionality was synchronizing alteration of forms across the client and server scenes. Another feature was for any user to be able to target an object in the scene and generate an STL file from its geometry data. The features were successfully added by writing scripts into the main program and into separate files using the Python programming language. The Universal Modeling Language 2.0 was used to visualize functionality of the program. A physical object was successfully fabricated from a generated STL file using a 3D printer.

Keywords: Crowdsourcing, 3D printing, CAD, stereolithography, gamification, web applications

SISÄLTÖ

Opinnäytetyön tiivistelmä....2
Thesis abstract....3
SISÄLTÖ....4
Kuva-, kuvio- ja taulukkoluettelo....7
Käytetyt termit ja lyhenteet....9
1 Johdanto....12
2 Teoria....14
2.2 Tietokoneavusteinen suunnittelu ja valmistus....14
2.3 Joukkoistaminen....14
2.4 Pelillistäminen....15
2.5 Palvelin- ja asiakasohjelma....16
3 Toteutus....17
3.2 Ohjelman tila työn alkaessa....17
3.2.1 Mallien tuottaminen....17
3.2.2 Viestien lähettäminen tietoverkon yli....17
3.2.3 STL-tiedoston tuottaminen....18
3.3 Tietotekniset työkalut....18
3.4 Käyttötapaukset....19
3.5 Käyttöliittymä....21
3.6 Ohjelman rakenne ja toimintaperiaate....21
3.6.1 UML-mallinnus....22
3.6.2 Käyttötapauskaavio....22
3.6.3 Oliokaavio....24
3.6.4 Sekvenssikaavio....28
3.6.5 Tietorakenteet....31
3.6.6 Säikeet....34
3.6.7 Oliot....35
3.6.8 Logiikkatiilet....36
3.7 Monen käyttäjän ohjaussignaalien vastaanottaminen....36
3.7.1 UDP-viestien lähettäminen ja vastaanottaminen....36

3.7.2 Vastaanotettujen viestien tulkitseminen palvelimella....37
3.8 Kappaletietojen synkronointi verkon yli....38
3.8.1 Liiketilojen synkronointi....39
3.8.2 Muotojen synkronointi....40
3.9 Blend- ja STL-tiedoston tuottaminen ohjelman kulun aikana....41
3.9.1 3D-tulostuksen koeajo....41
4 Yhteenveto....43
LÄHTEET....45
LIITTEET....49
A. Mallinnus....1
Blender....1
B. Ohjelmointi....2
Python 3....2
C. G-koodin tuottaminen....3
Slic3r....3
D. 3D -tulostaminen....4
Repetier-Host....4
E. Tekstuurien tuottaminen....5
GNU Image Manipulation Program....5
F. Versionhallinta....6
Git....6
G. Käyttöjärjestelmä....6
Linux Lite....6
Blenderin asennus....1
Python 3:n ja lisäkirjastojen asennus....1
Python 3....2
Lisäkirjastojen tarkistus....3
Pip....5
ToMC:n käynnistys....5
Palvelimen IP-osoitteen määrittäminen....5
Valikot....6

Suunnittelunäkymä..9

Kuva-, kuvio- ja taulukkoluettelo

Kuva 1. Muuttujien alustusta ja dokumentaatiorivejä ohjelman käynnistyksen yhteydessä suoritettavassa aliohjelmassa “GlobalDictIni.py”.....33

Kuva 2. Näkymän olion Spawner sisältämiä logiikkatiiliä ja niiden välisiä yhteyksiä. 33

Kuva 3. Näkymän olion ControlHub sisältämiä pelimuuttujia.....34

Kuva 4. “HTTPSpatialServer.py”-aliohjelmalla tuotettu verkkosivu avattuna Mozilla Firefox -verkkoselaimella.....39

Kuva 5. “HTTPFormServer.py”-aliohjelmalla tuotettu verkkosivu avattuna Mozilla Firefox -verkkoselaimella.....40

Kuva 6. Vasemmalla 3D-tulostin Minifactory 1.0 valmistamassa ToMC-ohjelmalla muokatun Alvar Aallon suunnitteleman vanhan maakuntakirjaston pienoismallia ja oikealla kuva valmiista pienoismallista.....42

Kuvio 1. Kappaleen matka käyttäjien mallinnustoimenpiteistä fyysisesti valmistetuksi kappaleeksi.....20

Kuvio 2. Käyttötapauskaavio valikoista ja suunnittelunäkymästä.....23

Kuvio 3. Oliokaavio geometriamuutospyynnöstä, palvelimen vastauksesta ja muutosten päivittämisestä asiakasohjelman näkymään.....25

Kuvio 4. Oliokaavio STL-tiedoston tuottamisesta suunnittelunäkymässä kursorilla valitusta kappaleesta.....27

Kuvio 5. Sekvenssikaavio geometriamuutosten lähetyksestä, muutosten toteuttamisesta palvelimella ja päivittämisestä asiakasohjelman näkymään.....29

Kuvio 6. Sekvenssikaavio STL-kaavion tuottamisesta suunnittelunäkymässä……30

Kuvio 7. Kaavio asiakasohjelman UDP-viestinnästä palvelimen kanssa.....37

Kuvio 8. Tietorakenteiden lähettäminen ja vastaanottaminen periaatetasolla…….38

Taulukko 1. Hakurakenteen bge.logic.globalDict sisältämiä muuttujia.....32

Käytetyt termit ja lyhenteet

3D-tulostus	Materiaalia lisäävä tuotantotekniikka, jossa kappale muodostetaan kiinnittämällä sen poikkileikkauskuvioita päällekkäin
Asiakasohjelma	Tietokoneohjelma, joka ottaa yhteyttä palvelimeen toteuttaakseen osan toiminnoistaan
Asynkroninen	Prosessi, jonka tapahtumat ovat järjestykseltään riippumattomia valitusta tahdinnäyttäjäprosessista
Avatar	Ohjelman sisällä mallinnettu geometrinen kappale, johon käyttäjän kamera on ohjelman kulun aikana kiinnitetty
Datagrammi	UDP-viestien välityksen viestikokonaisuus, joka koostuu viestityypin ilmaisevasta otsikosta ja viestin sisällöstä
DBM	Database *manager, h*akurakenteen sisältävä tiedostotyyppi, joka on siirrettävissä tietokoneesta toiseen
Git	Versionhallintaohjelma, joka synkronoi käyttäjän valitseman tietokoneen kansion sisällön palvelimelle
Github	Verkkosivusto, joka toimii palvelimena ohjelmalle Git ja käyttäjän valinnan mukaan näyttää valitut tiedostot julkisesti ja mahdollistaa muiden käyttäjien esittämien muutosten tekemisen tiedostoihin
G-koodi	Numeerisen ohjauksen ohjelmointikieli, jota käytetään tietokoneavusteisessa valmistuksessa koneiden työstöprosessien ohjaamiseen
Hakurakenne	*Eng. map/dict,* abstrakti tietotyyppi, joka näyttäytyy käyttäjälle numeroituvana määränä järjestykseltään riippumattomia avain-arvopareja, joista molemmat ovat tietosisältöjä

Lista	Abstrakti tietotyyppi, joka näyttäytyy käyttäjälle tyhjänä tai järjestykseltään tunnettuna kokoelmana eri tietotyyppien tietosisältöjä
Merkkijono	Tietotyyppi, joka näyttäytyy käyttäjälle jonona kirjoitusmerkkejä
Monisto	Eng. *Manifold* - geometrinen kappale, jonka pinta on suljettu
Palvelin	Tietokone tai ohjelma, joka mahdollistaa siihen yhteyttä ottavien tai pitävien tietokoneohjelmien toimintaa
PLA	Polylaktidi, maissitärkkelys- tai sokeriruokopohjainen muovityyppi, joka soveltuu sulamislämpötilaltaan kuluttajahintaisten 3D-tulostinten käyttömateriaaliksi
Repetier-host	Tietokoneohjelma, joka ottaa syötteekseen mm. G-kooditiedoston ja ohjaa sen avulla tietokoneeseen kytketyn 3D-tulostimen työstöprosessia
Skripti	Aliohjelma
Slic3r	Tietokoneohjelma, joka muuttaa STL-tiedostoja G-koodiksi käyttäjän antamin asetuksin
Spatiaalinen	Neliulotteisen avaruuden data, jossa on ilmoitettuna kappaleiden paikat, kääntökulmat, nopeudet ja kulmanopeudet
STL	Stereolitografiatiedostotyyppi, joka sisältää mallinnetun kappaleen kolmioina esitetyn geometrian
Tangentiaalinen	Yhteydessä tangentiaalinen oppiminen tarkoitetaan prosessia, jossa ihmiset hakeutuvat oma-aloitteisesti oppimaan aiheesta, joka on esitelty heille osittain
Tietorakenne	Tietokoneen tapa järjestellä dataa käyttöä varten

ToMC	Tools of Mass Creation, sen ohjelman työnimi, jota opinnäytetyönä kehitettiin
UDP	*User Datagram Protocol*, erittäin nopea tietokoneiden välinen kommunikointitapa, joka ei varmista viestin saapumista kohteeseensa
UML	Unified Modeling Language - ohjelmistokehityksessä käytettävä mallinnuskieli, jolla kuvataan erilaisin kaavioin ohjelman rakennetta, käyttäytymistä ja vuorovaikutusta
USB	*Universal Serial Bus,* fyysinen väylä, jonka kautta tietokone viestii oheislaitteidensa kanssa
Väline-ehdollistuminen	Oppimistapa, jossa palkitun toiminnan esiintymistodennäköisyys lisääntyy ja rangaistun toiminnan esiintymistodennäköisyys pienenee

1 Johdanto

Numeerisesti ohjatut valmistusprosessit pohjautuvat tietokoneavusteisten suunnitteluohjelmien avulla tuotettuihin geometrisiin malleihin. Näiden mallien yksityiskohtainen tuottaminen on tyypillisesti teknisiin tehtäviin kouluttautuneen ammattiryhmän edustajien työtä. Työkaluna käytettävät ohjelmat tukevat tyypillisesti vain yhden käyttäjän tekemiä muutoksia kerrallaan (Schmitz 2014 ,Thomas-Lepore 2013). Ne sisältävät vaikeasti hallittavan määrän erilaisia toimintoja ja muutosten näkyminen toisille ohjelman käyttäjille vaatii erillisen paikallisten tiedostojen tilan kopioimisen palvelimelle ja palvelimelta (Fuh & Li 2005, 571-581; Graphisoft 2017).

Tässä opinnäytetyössä lisättiin Blender Game Enginellä eli BGE:llä (Pan & Felinto 2013, 1-36) tuotettuun ohjelmaan verkkotoiminnallisuus, joka synkronoi kaikkien käyttäjien tekemiä muoto- ja liiketilamuutoksia jatkuvasti kaikkien muiden käyttäjien näkymiin. Toinen toiminto, joka ohjelmaan tuli lisätä, oli STL-tiedoston tuottaminen näkymässä mallinnetusta kappaleesta. Tällä tavoin yhteistyönä mallinnetun kappaleen matka edelleen G-koodiksi ja 3D-tulostetuksi kappaleeksi saatiin lyhyeksi. Opinnäytetyö tallennettiin USB-muistille liitteenä 1.

CAD-näkymässä erilaiset simulaatiotilat on käynnistettävä erikseen. BGE:n avulla koko näkymän voi pitää jatkuvasti simulaatiotilassa, ja monimutkaisia kokonaisuuksia voi muokata keskeytyksettä. Suunnitteluympäristö voidaan rakentaa pelillistämisen suunnitteluperiaatteita noudattaen (Halevy, Doan & Ramakrishnan 2011) käyttäjäkunnan laajentamiseksi. Blender on kehitysympäristönä ilmainen käyttää ja sen pelimoottorin sisällä ajettava ohjelmakoodi on Python 3 -kieltä (Pan & Felinto 2013, 297-337; Phillips 2010, 1-62). Sen avulla on mahdollista käyttää lukuisia eri käyttäjärajapintoja ja anturidatan lähteitä (Sanner 1999, 57-84).

Tieteen ja tekniikan edistämisen näkökulmasta tietokonepelien vaikutus on näkynyt toistaiseksi mm. ongelmanratkaisun joukkoistamisena. Esimerkki tällaisesta joukkoistamisesta on proteiinien laskostumisongelman ratkaisujen etsiminen tietokonepelin avulla (Kleffner ym. 2017, 2765-2767; UW Center for Game Science ym. 2017). Peruste joukkoistamisen käyttämiseen on ongelman

suuri laskennallinen hakuavaruus. Luonnollisten toimintaympäristöjen ongelmien ja mahdollisuuksien ymmärtäminen on myös tekoälyille yhä erittäin vaikea tehtävä (Boden 1998, 347-356). Vaikka mekaanisten ominaisuusvaatimusten perusteella on onnistuttu tuottamaan koneellisesti malleja rajattuihin tarkoituksiin, kuten stadioneiden kattorakenteisiin (Shea, Aish & Gourtovaia 2005, 253-264) ja autojen runkoihin (Turney 2016), ihmisen työksi jäävät tämän vuoksi yhä monesti arkkitehtuurissa ja mekaniikkasuunnittelussa ratkaistavat ongelmat.

Opinnäytetyön ensimmäinen osa käsittelee yleisesti geometrisen mallin matkaa fyysiseksi kappaleeksi, tehtävien suorittamista mahdollistamalla suurten ihmisjoukkojen työpanos käyttöliittymäsuunnittelulla, tehtävien naamiointia vuorovaikutteiseksi viihteeksi ja palvelin- ja asiakasohjelma-arkkitehtuureja. Toisessa osassa käydään läpi ohjelman tila työn alussa, mihin sitä voi käyttää ja miten sitä ohjataan. Kolmannessa osassa tarkastellaan tiedon varastointia ja ohjelman sisäisiä dataobjekteja. Neljännessä osassa tarkastellaan ohjelmatason mekanismeja, joilla tietokone lähettää ja vastaanottaa tietoverkkoviestejä, ja kuudennessa selitetään näkymien synkronointia tietoverkon välityksellä. Valittujen ohjelmien ja kirjastojen kuvausta ja perusteluja niiden käytölle esitellään liitteessä 5.

2 Teoria

2.2 Tietokoneavusteinen suunnittelu ja valmistus

Modernien, numeerisesti ohjattujen valmistuslaitteiden käyttöön tarvitaan koneen fyysisten komponenttien numeerisen ohjauksen ohjelmointikieliä, kuten G- ja M-koodia (Brown & de Beer 2013, 1-5). G-koodin tuottaminen monimutkaisista kappaleista suoritetaan tyypillisesti ohjelmallisesti. Tästä esimerkkinä on 3D-tulostusta varten STL-tiedostosta G-koodia tuottava ohjelma Slic3r. STL-tiedosto sisältää tuotettavan kappaleen geometriset muodot ilman pinnanlaatudataa. Tällainen representaatio on tuotettavissa mistä tahansa monistosta (Chakravorty 2017; Rowland [Viitattu 24.11.2017].) Intuitiivisemmin tämä tarkoittaa, että mikäli STL-muodossa tallennetun kappalegeometrian pinnassa ei ole reikiä, siitä voi tuottaa G-kooditiedoston.

2.3 Joukkoistaminen

Kun työtehtävän osan tietojenkäsittelyn automatisointi on vaikeaa ja tehtävä itsessään on ulkoistettavissa suurelle määrälle käyttäjiä, on mielekästä käyttää verkon kautta tapahtuvaa joukkoistamista (Halevy, Doan & Ramakrishnan 2011, 86). Tällainen ohjelma vaatii käyttöliittymän, jonka suunnittelussa on otettu huomioon kohderyhmän käyttäjien tekninen osaamistaso ja heidän oppimiskykynsä, kuten esimerkiksi Clickworkers-verkkokokeilussa (Surowiecki 2004, 273-279). Kokeilussa käytetty verkkosivusto opetti lyhyellä harjoittelulla palvelun käyttäjiä tunnistamaan ja luokittelemaan Marsin kraattereita, ja he saavuttivat tehtävässä lähes ammattilaisgeologin taitotason. Mikäli ohjelman käyttäjillä ei ole valmiiksi ulkoista motivaation lähdettä, kuten palkkaa vastineeksi ohjelman käytöstä, on mielekästä toteuttaa motivointi pelillistämisen avulla.

2.4 Pelillistäminen

Työn naamiointi viihteellisen virtuaalisen käyttöliittymän taakse tunnetaan pelillistämisenä. Viihteellisillä, vuorovaikutteisilla toimintaympäristöillä on etuja monimutkaisten suoritusten opettamisessa (Hamari, Koivisto & Sarsa 2014, 3025-3034). Ohjelman käyttäjälle tämä merkitsee esimerkiksi väline-ehdollistumista ja tangentiaalista oppimista (St-Pierre 2011, 74-96; Breuer & Bente 2010, 7-24).

Käyttäjän motivaation lähteitä arvioitiin Maslown tarvehierarkian pohjalta (Maslow 1943). Esimerkiksi taidokkuus (eng. *mastery*), itsen aktualisointi ja vertaisten hyväksyntä olivat tarpeita, joiden täyttämisen tavoittelun oletettiin olevan mahdollista virtuaalisessa ympäristössä, jossa tuotetaan yhteistyönä monimutkaisia, toiminnallisia rakennelmia. Väline-ehdollistumisen oletettiin toimivan keskeisenä ohjaavana psykologisena mekanismina tarpeiden täyttämisen tavoittelussa.

Väline-ehdollistumista tapahtuu, kun toiminta johtaa palkkioon tai rangaistukseen (Kirsch ym. 2004, 369–392). Nämä palkkiot ja rangaistukset ilmenevät esimerkiksi siten, että käyttäjä tekee ennusteen toimintansa seurauksista simulaationäkymässä, minkä jälkeen hän toteuttaa toimintansa ja ehdollistuu sen mukaisesti, oliko hänen ennusteensa oikea vai väärä. Väline-ehdollistumista oletettiin tapahtuvan esimerkiksi käyttäjän tuottaessa simulaationäkymään kappaleita ja koetellessa niiden soveltuvuutta käyttötarkoituksiinsa. Palkkio- ja rangaistussignaalien lähteinä voivat toimia myös vertaisten palaute ja keinotekoiset tavoitteet, joita ohjelmaan voidaan tuottaa (Hamari, Koivisto & Sarsa 2014, 3025-3034). Lyhyt aika käyttäjän yrityksen ja palautteen välillä on väline-ehdollistamista voimistava tekijä, jonka oletettiin vaikuttavan simulaatioympäristössä (Fantino 1987).

Tangentiaalinen oppiminen tarkoittaa käyttäjän itsenäistä tiedonhakua aiheesta, jolle he ovat altistuneet (Brown ym. 2013). Ohjelman sisällä ilmenevät haasteet toimivat lähtökohtana aihetta koskevaan tietoon, mutta käyttäjän tiedonhaku aiheeseen laajenee ja syvenee hänen kiinnostuksensa mukaisesti. Tiedon käyttökelpoisuuden on ehdotettu olevan tangentiaalista oppimista edistävä tekijä (Jilani, Kadobayashi & Jozen 2014, 496-499).

Viihdepelaajat käyttävät valtavia määriä vaihtelevan mittaisia sessioita erilaisissa virtuaalisissa ympäristöissä suorittaen monipuolisia tehtäviä (McGonigal 2011, 1-5). Tehtävän joukkoistaminen tarvitsee usein suuren käyttäjämäärän. Yksi suuri kohderyhmä ToMC-ohjelman kaltaiselle mallinnus- ja simulaatioympäristölle on viihdepelien monipuolinen yleisö.

Pelillistettyjen työtehtävien ihmisille, ympäristölle tai koneelle itselleen aiheuttamien vaaratilanteiden tapahtumistodennäköisyydet ja seuraukset oli kyettävä tekemään mahdollisimman pieniksi. ToMC-ohjelmassa tämä ei ollut suuri ongelma, sillä kaikki 3D-tulostimen fyysisten komponenttien ohjaaminen tapahtui automaattisesti Repetier-Host-ohjelman kautta. Suurimmat riskit sisältyivät ToMC-ohjelman verkon kautta siirrettävien blend-tiedostojen sisältämien kappaleiden aliohjelmiin ja palvelinohjelman verkkotoiminnallisuuteen. Piilottamalla blend-tiedostojen skripteihin vahingollisia komentoja ne oli mahdollista suorittaa tietokoneella, jonne ne oli kopioitu. Pythonin SimpleHTTPServer-palvelimen kautta oli mahdollista nähdä palvelinkoneelta tiedostoja, joita ei ollut tarkoitettu jaettaviksi (Freeney 2014).

Markkinoilla on jo digitaalisia pelejä, jotka perustuvat peliympäristön muokkaamiseen (Deffenbaugh 2012, 1-4; Short 2012; Kaplan 2009, 563-572). Ympäristön muokkaamisen mekanismit perustuvat näissä tapauksissa kappaleiden tuottamiseen, poistamiseen, siirtelyyn ja kiinnittämiseen toisiinsa. Kappaleet ovat teksturoituja ja fysiikkamallinnettuja, pelaajat voivat muuttaa niiden muotoja vain rajoitetusti, usein esimerkiksi ennalta määritellyn hajoamisanimaation mukaisesti.

2.5 Palvelin- ja asiakasohjelma

Asiakasohjelma ottaa tietoverkon kautta yhteyttä palvelinohjelmaan toteuttaakseen osan toiminnoistaan. Asiakasohjelma voi pystyä esimerkiksi itsenäiseen kappaleiden törmäysfysiikoiden simulointiin, mutta näkymien yhdenmukaisuuden varmistamiseksi kaikkien asiakasohjelmien on haettava jatkuvasti kappaleiden paikka-, nopeus- ja muototietoja palvelinohjelmalta.

3 Toteutus

3.2 Ohjelman tila työn alkaessa

Ensimmäiset merkinnät ToMC-ohjelman Github-versionhallintahistoriassa ovat heinäkuulta 2013. Tuona aikana kirjoitettiin ensimmäiset HTTP-säieskriptit, näkymässä liikkuminen sekä näkymän lataaminen että tallennus. BGE-moottori muutti oletusarvoisesti kaikkien kappaleen kopioiden geometrioita samalla tavalla. Tämän ongelma ratkaistiin kirjoittamalla BGE-moottorin Libload- ja Libnew-funktioita (Blender Foundation 2017) hyödyntävää Python 3-ohjelmakoodia ToMC-ohjelmaan ennen opinnäytetyön aloittamista.

3.2.1 Mallien tuottaminen

Työn alkaessa kappaleiden yksilöllinen muokkaaminen oli ominaisuutena valmis vain palvelimen näkymässä. Näkymän tallentaminen DBM-tiedostoon toimi, mutta näkymän uudelleen tuottaminen tiedostosta ei toiminut. Indeksointivirheiden vuoksi näkymän kappaleet olivat väärissä paikoissa ja värähtelivät jatkuvasti lattiatason läpi. Toiminnallisuus, jonka avulla muotojen kopiointi palvelimelta asiakasohjelmille onnistuisi, oli tuotettava opinnäytetyön aikana. Ohjelmassa oli valmiina ominaisuutena olevaa liiketilatietojen synkronointi HTTP-tiedonsiirron avulla, mikä toimi pohjana muototietojen lähettämiseen tarvittavan toiminnallisuuden rakentamiselle.

3.2.2 Viestien lähettäminen tietoverkon yli

Verkon kautta kommunikointiin oli valmiina opinnäytetyön alkaessa pohjakoodia, jonka avulla kappaleiden spatiaalista dataa oli voitu lähettää, vastaanottaa ja tulkita asiakasohjelman ja palvelimen välillä. Tämä vanha koodi tuli muokata sopivaksi asynkronisen kappaleiden tuottamisen kanssa. Kappaleiden geometrista dataa välittäviä viestejä ei oltu vielä implementoitu.

3.2.3 STL-tiedoston tuottaminen

Työn alussa ohjelman sisältä käsin ei ollut mahdollista tuottaa Blend-tiedostoja näkymässä esiintyvistä kappaleista. Tämän vuoksi 3D-tulostamiseen vaadittua G-koodia edeltävää STL-tiedostoa ei vielä voitu tuottaa.

3.3 Tietotekniset työkalut

Tietotekniset työkalut ja tarvittavat lisäkirjastot on esitelty lyhyesti liitteessä 5. Työkalut on jaettu käyttötarkoituksen mukaan mallinnukseen, ohjelmointiin, G-koodin tuottamiseen, 3D-tulostamiseen, tekstuurien tuottamiseen, versionhallintaan ja käyttöjärjestelmään.

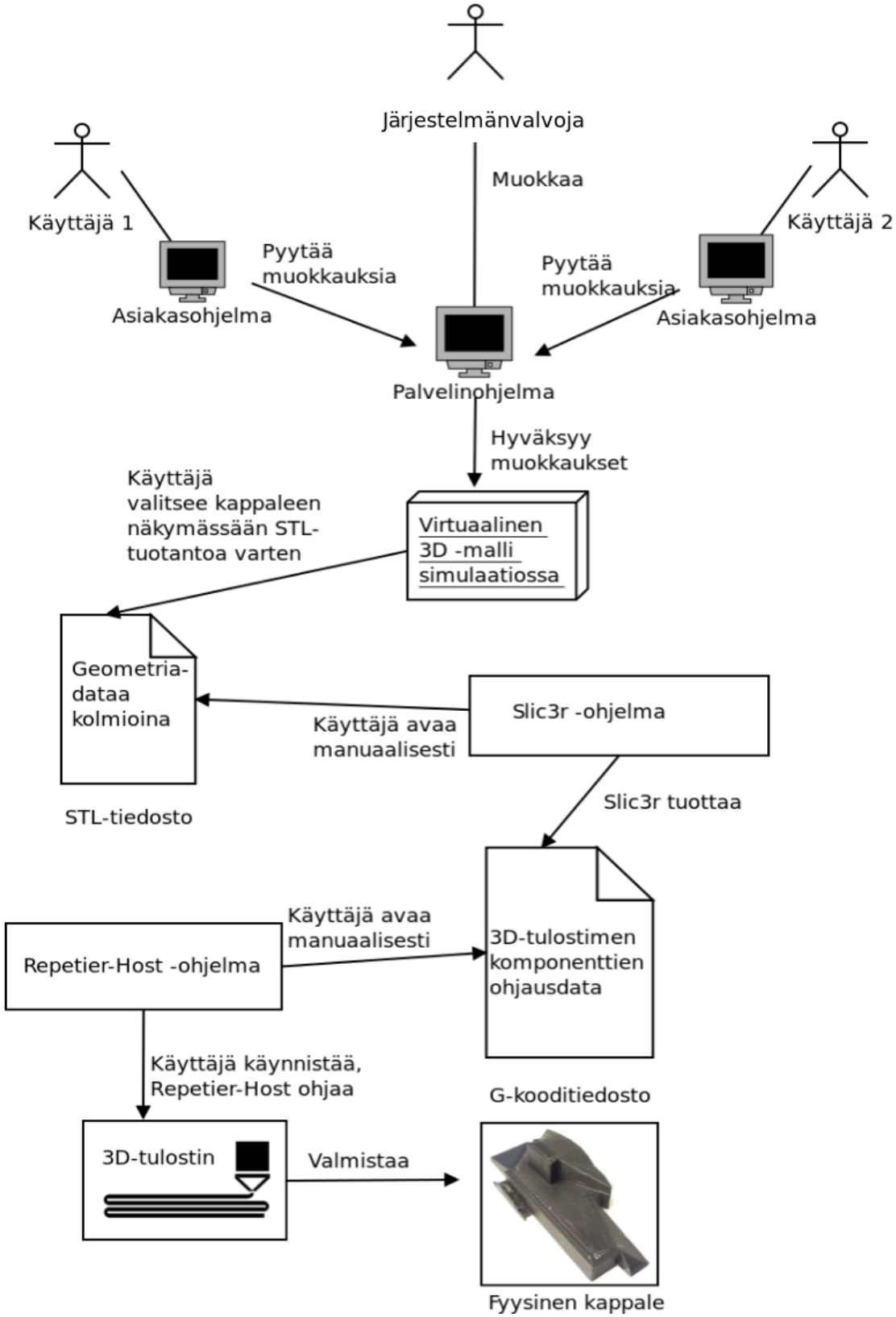
3.4 Käyttötapaukset

ToMC-ohjelmaa voidaan käyttää ohjelman ObjectLibrary-kansion blend-tiedostoihin tallennettujen mallien tuottamiseen ja muokkaamiseen. Työn päättyessä oli käytettävissä perusgeometrioita, kuten kuutioita ja sylintereitä, sekä käsin mallintaen tuotettuja STL-malleja Seinäjoella sijaitsevista Alvar Aallon suunnittelemista rakennuksista.

Käyttäjä voi määrätä ohjelmainstanssinsa joko asiakas- tai palvelinohjelmaksi. Palvelimelle voi liittyä monta asiakasohjelmaa samanaikaisesti, ja kaikkien ohjelmainstanssien käyttäjät voivat muokata synkronoidun näkymän kappaleita yhdessä (kuvio 1).

Tuotettu näkymä voidaan tallentaa uutta muokkaussessiota varten. Vaikka kommunikaatio palvelimen kanssa keskeytyisi, asiakasohjelmien käyttäjät voivat keskeytymisen jälkeen tallentaa paikalliset näkymänsä, aloittaa uusia muokkaussessioita ja jatkaa siitä, mihin näkymän tilanne jäi. Tämä onnistuu avaamalla keskeytetyn tilanteen paikka- ja geometriatiedot sisältävä tiedosto uudessa ohjelmainstanssissa, jonka käyttäjä määrää palvelinohjelmaksi.

Näkymän yksittäisiä kappaleita voidaan tallentaa STL-muotoon. Tässä tapauksessa Slic3r-ohjelmaa käyttämällä STL-tiedoston sisältämästä datasta voi generoida G-koodin. G-koodi syötetään Repetier-Host-ohjelmalle, joka ohjaa 3D-tulostimen fyysisiä komponentteja kappaleen valmistuksessa. Tällainen tiedosto löytyy työn liitteestä 4.



Kuvio 1. Kappaleen matka käyttäjien mallinnustoimenpiteistä fyysisesti valmistetuksi kappaleeksi

3.5 Käyttöliittymä

Ohjelman käyttöliittymä jakautuu kahteen osuuteen: valikoihin ja suunnittelunäkymään. Valikoissa käyttäjä voi määrätä, onko ohjelmainstanssi asiakas vai palvelin. Tämän valinnan muuttaminen vaatii ohjelman käynnistämisen uudelleen. Suunnittelunäkymässä käyttäjä voi liikuttaa avatarinaan toimivaa kuutiota, käännellä ja zoomata kameraa, tuottaa uusia kappaleita, muokata niiden muotoja ja tuottaa kursorin alla olevasta kappaleesta STL-tiedoston.

Viimeaikaisia kehityksiä (Solidface 2017) lukuun ottamatta tietokoneavusteiset suunnitteluohjelmat yhteistyöympäristöinä vaativat käyttäjältä resurssienvaraus- tai tiedostonhallintatoimenpiteitä. Toimenpiteiden jälkeen hän voi nähdä muiden käyttäjien kappaleisiin tekemät muutokset (Graphisoft 2017) tai muuttaa kappaleita itse. ToMC ohittaa tällaiset vaiheet, sillä ne eivät edistä mallinnusprosessia.

Avatarin ja kameran liikuttamiseen käytettyjä oletusohjaussyötteitä (liite 7) käytetään laajasti esimerkiksi massiivisissa verkkoroolipeleissä (Messinger, Stroulia & Lyons 2008, Blizzard 2017, 13; Electronic Arts 1999, 5; The Rift Development Team 2011, 48). Viihdepeleille on tyypillistä, että käyttäjä voi vaihtaa ohjaussyötettä kaikille toiminnoilleen, ja oletusohjaussyötteet (liite 7) on valittu ennakoiden tätä kehitystä.

3.6 Ohjelman rakenne ja toimintaperiaate

Ohjelman ydin on tiedosto ”ToMC_va62.blend”, jonka käynnistäminen vaatii Blenderin (liite 7). Tiedosto sisältää äänet ja tekstuurit sekä kappaleet, joita ei voida muokata, kuten valikoiden painikkeet. Se sisältää myös Python-skriptit, joiden toiminta ei edellytä niiden ajamista omissa säikeissään.

Ydintiedoston ulkopuolella olivat verkkopalvelimia ylläpitävät, paikallisen IP-osoitteen hakuun käytettävät ja STL-tiedoston generointiin käytettävät Python-skriptit kansiossa ”Extra_python_modules_ToMC”. Tallennetut näkymät varastoituvat kansioon ”Worlds” ja ohjelmasta STL-muodossa tuotetut tiedostot kansioon

”Exported_Objects/STL”. Kaikki muokattavissa olevien kappaleiden blend-tiedostot on varastoitu myös omaan kansioonsa ”ObjectLibrary”.

Kun ”ToMC_va62.blend” käynnistetään Blenderin pelimoottorissa, moottori aloittaa näkymän piirtämisen ja kappaleiden fysiikkamallinnuksen. Ydintiedostoon sisäänkirjoitetut Python-skriptit käynnistävät tarvittaessa muita skriptejä omissa säikeissään. Kaikki näkymään tuotettavat kappaleet haetaan omista yllä mainituista blend-tiedostoistaan. Käyttäjän sulkiessa ohjelman myös käynnistetyt palvelinsäikeet suljetaan.

3.6.1 UML-mallinnus

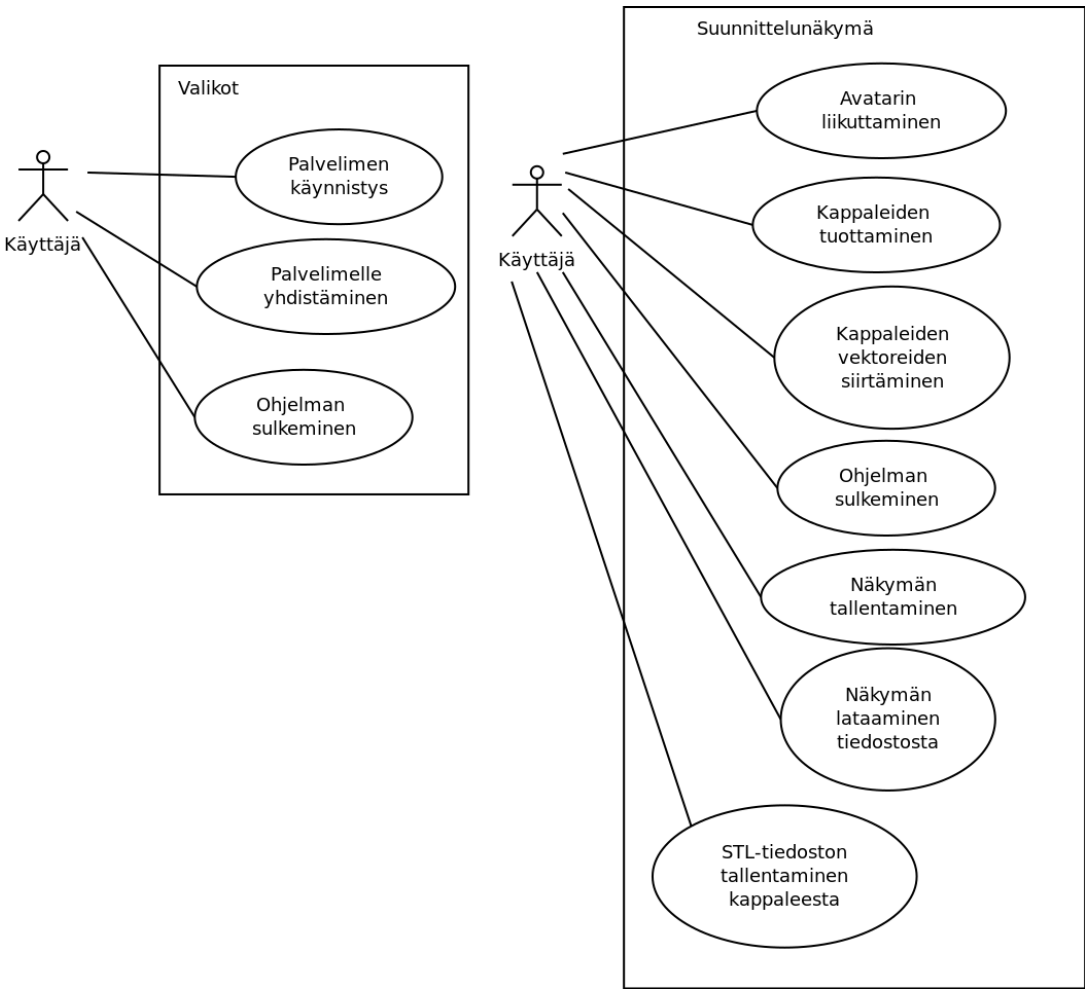
Tuotettujen muutosten visualisointiin käytettiin UML 2.0 -yleismallinnuskieltä (Grässle, Baumann & Baumann 2005). UML on usein ohjelmistoteollisuudessa käytetty kieli, joka tarjoaa laajan valikoiman kaaviotyyppejä ohjelman toiminnan ja rakenteen kuvaamiseen. Kaaviotyypeistä valittiin soveltaen käytettäviksi käyttötapauskaavio, oliokaavio ja sekvenssikaavio. Kaavioista selviää, mitä toimintoja ohjelma sisältää käyttäjän näkökulmasta, mitä informaatiota ohjelman komponentit viestivät keskenään, ja missä järjestyksessä komponenttien keskinäiset vuorovaikutukset tapahtuvat. Oliokaaviota on havainnollisuuden parantamiseksi laajennettu sisältämään aliohjelmia, niiden muuttujia ja tiedostoja ohjelman ulkopuolella.

3.6.2 Käyttötapauskaavio

Käyttötapauskaavio koostuu käyttäjäikoneista, ellipsien sisään nimetyistä käyttötapauksista, janoista näiden kahden välillä ja suorakulmiosta, joka merkitsee ohjelman rajaa (kuvio 2). Janat merkitsevät käyttäjän vuorovaikutusta ohjelman käyttöliittymän näkyvien tai ohjekirjassa mainittujen osien kanssa. Käyttötapaukset on merkitty vapaassa järjestyksessä.

Käyttötapauksiin voi olla myös yhteydessä sisällytettyjä (eng. *include*) toimintoja, jotka tapahtuvat käyttäjältä näkymättömissä aina suoritettaessa käyttötapaus.

Laajentavat toiminnot (eng. *extend*) ovat myös käyttötapausta suorittaessa näkymättömissä tapahtuvia prosesseja, jotka tapahtuvat vain tiettyjen ehtojen täyttyessä. Laajentavaa toimintoa kohti piirretty nuoli ilmaisee, että nuolen alkupäässä oleva käyttötapaus voi johtaa nuolen loppupäässä olevan toiminnon käynnistymiseen. Sisällytetystä toiminnosta ulospäin piirretyt nuolet osoittavat käyttötapauksiin, joiden kanssa sisällytetty toiminto aina käynnistyy.

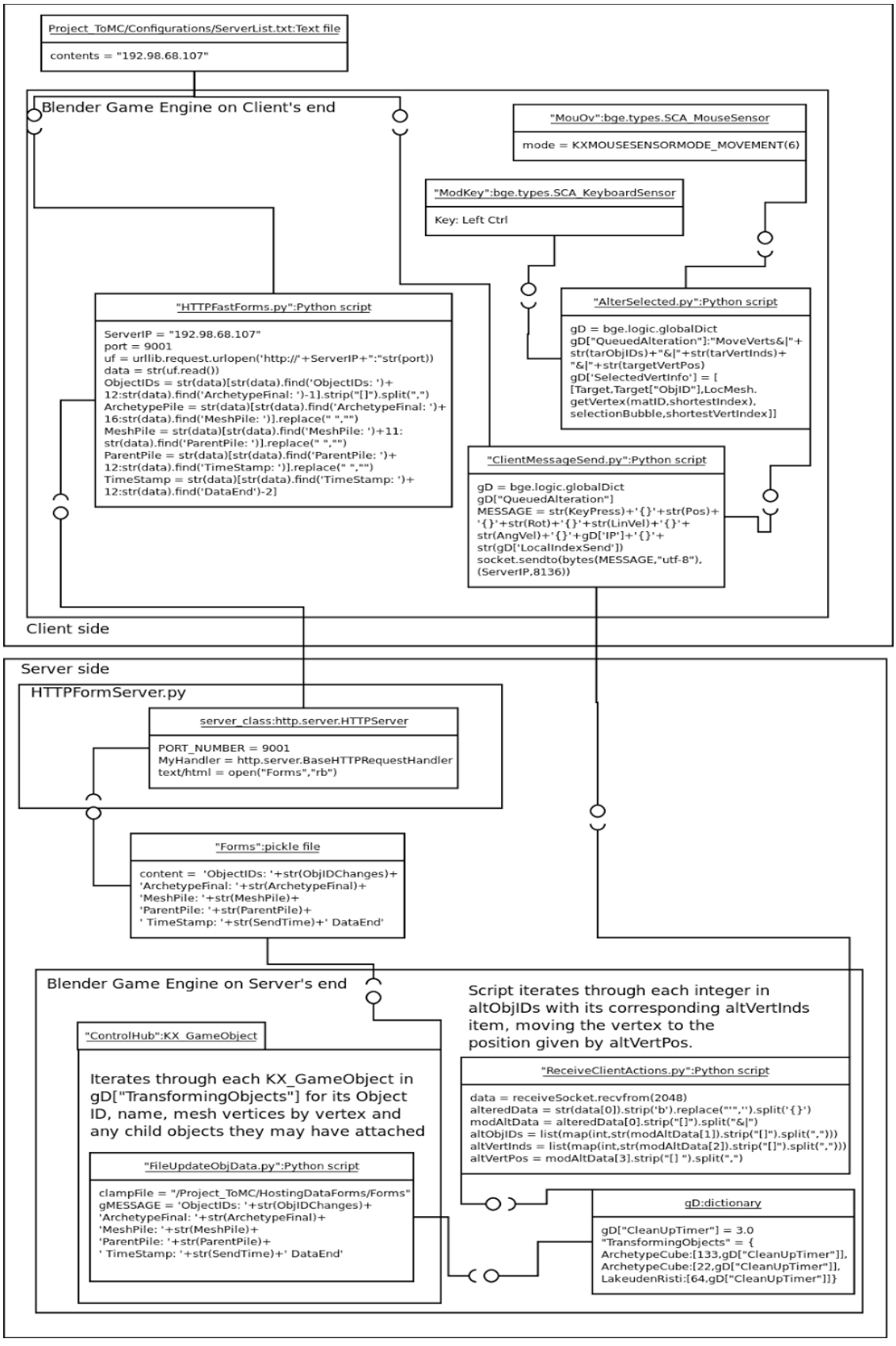


Kuvio 2. Käyttötapauskaavio valikoista ja suunnittelunäkymästä

3.6.3 Oliokaavio

Ohjelman toimintaan osallistuvat luokista tuotetut oliot kuvataan oliokaaviossa (kuvio 3 ja kuvio 4). Toisiinsa kiinnitetyt suorakulmiot merkitsevät olioita ja niiden nimi ja tyyppi on määritelty ylemmässä suorakulmiossa. Alemmassa osassa on lueteltu olion sisältämät attribuutit.

Oliot on suljettu ohjelman rajoja kuvaavien suorakulmioiden sisään. Näiden rajojen ulkopuolelle jäävät oliot ovat tiedostoja, jotka toimivat datan väliaikaisina sijoituspaikkoina ennen lähettämistä verkon yli. Suorakulmioiden väliset konnektorit ilmaisevat, minne suuntaan data siirtyy. Data on saatavilla konnektorin ympyräosaan kiinnitetystä suorakulmiosta ja data ohjataan konnektorin puoliympyräosaan kiinnitettyyn suorakulmioon.



Kuvio 3. Oliokaavio geometriamuutospyynnöstä, palvelimen vastauksesta ja muutosten päivittämisestä asiakasohjelman näkymään

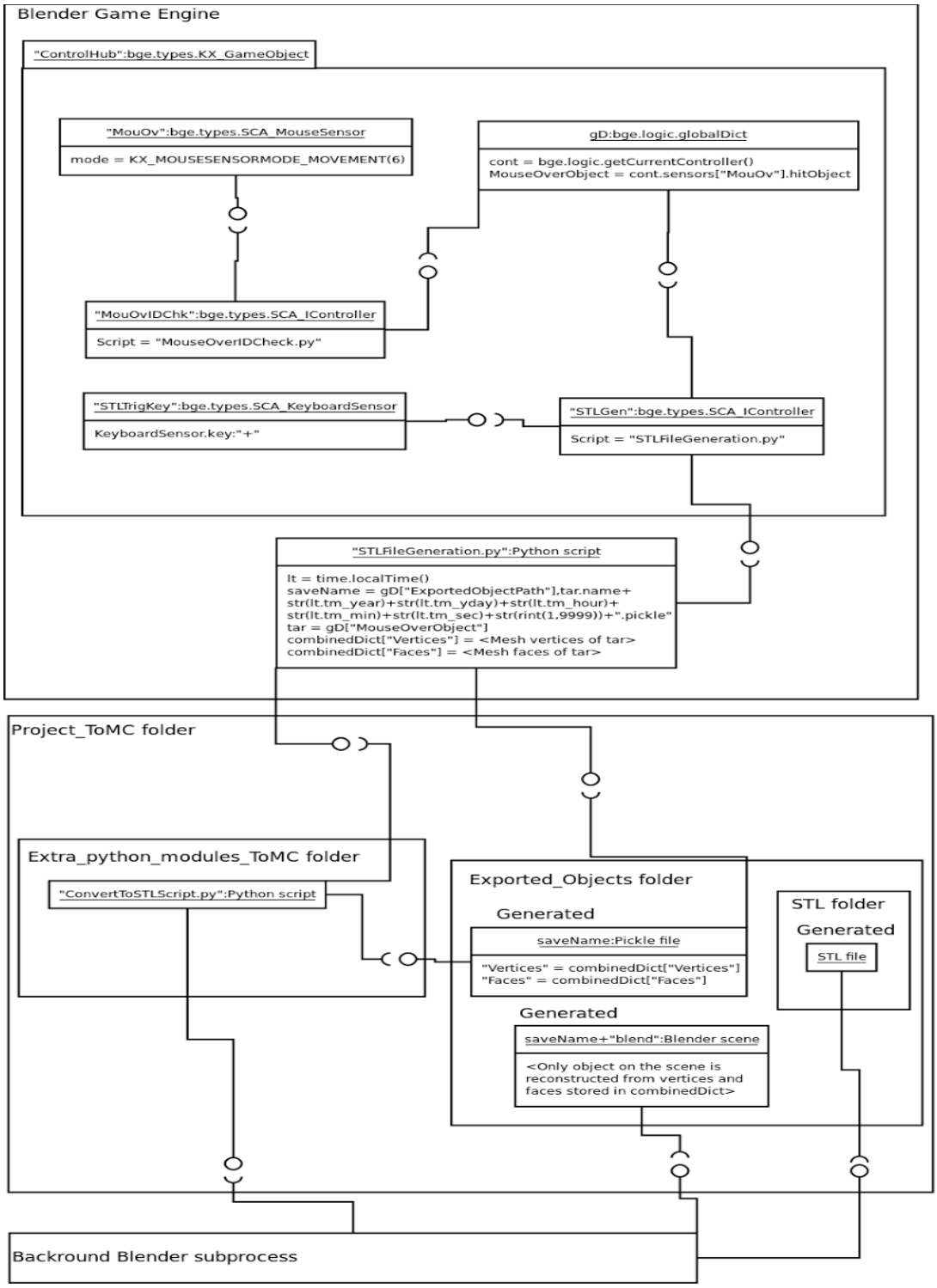
Kuviossa 3 on kuvattu prosessiketju, joka alkaa asiakasohjelman käyttäjän muutospyynnöstä kappaleen geometriassa ja päättyy palvelimen näkymässä muutettujen geometrioiden synkronointiin asiakasohjelmassa. Muuttaakseen kappaleen geometriaa asiakasohjelman käyttäjä valitsee hiiren vasemmalla näppäimellä kappaleista kulmia, painaa Ctrl-näppäimen pohjaan ja liikuttaa hiirtä. Aliohjelma “AlterSelected.py” suorittaa matriisilaskennan, joka liikuttaa vektoreita huomioiden käyttäjän kameran paikan ja asennon. Kun laskenta on valmis, sen tulokset annetaan edelleen aliohjelmalle “ClientMessageSend.py”. Se lähettää pyynnön vektorien uusista paikoista palvelimelle Internetin läpi UDP-datagrammeina.

Palvelimen puolella saapuvat datagrammit vastaanottaa aliohjelma “ReceiveClientActions.py”. Saapuessaan datagrammi on UTF-8-merkistöä noudattava tavujono. Tavujonosta geometriamuutokset erotellaan viestin MESSAGE osuudesta ”KeyPress”. ”KeyPress” sisältää kohdekappaleen tunnistusnumeron, muutettavien vektoreiden indeksit kappaleen sisällä ja vektoreiden uudet sijainnit. Tavujono muutetaan merkkijonoksi, siitä poistetaan laskentaan vaikuttamattomat merkit, kuten yläpilkut ja tavujonon aloittanut b. Merkkijonoa pilkotaan eri symboliyhdistelmien kohdilta ja eroteltujen merkkien tietotyyppejä muutetaan, kunnes jäljellä on palvelimen näkymässä tehtävien geometriamuutosten suorittamiseen tarvittavat tietotyypit.

Asiakasohjelman lähettämät geometriamuutokset suoritetaan palvelimen näkymässä. Tästä eteenpäin muutoksia kohdellaan samalla tavoin kuin ne olisi tehty suoraan palvelimen näkymässä. Muutosten kohteina olevat kappaleet tallennetaan globaaliin hakurakenteeseen ”gD”, ja ne poistetaan hakurakenteesta kolmen sekunnin umpeuduttua. Muutosten kohteina olevat kappaleet tallennetaan listaan ja tallennetaan pakattuun tiedostoon Forms. BGE:n ulkopuolinen aliohjelma “HTTPFormServer.py” lukee Forms-tiedoston ja pitää merkkijonon sisältävää HTTP-sivua yllä Internetissä.

Asiakasohjelman aliohjelma “HTTPFastForms.py” lukee HTTP-verkkosivun sisällön ja jakaa merkkijonon kappaleiden tunnistusnumeroihin, kappaleiden nimiin, kappaleet muodostaviin vektoreihin, kappaleiden kiinnityssuhteisiin ja aikaleimaan. Jako tapahtuu etsimällä merkkijonosta osia kuten “ObjectIDs: “ ja

“MeshPile: “. Parsittuaan geometriamuutoksiin tarvittavan datan viestistä “HTTPFastForms.py”-skripti muokkaa paikallisessa näkymässä kappaleiden geometriat palvelimen näkymän tilannetta vastaaviksi.

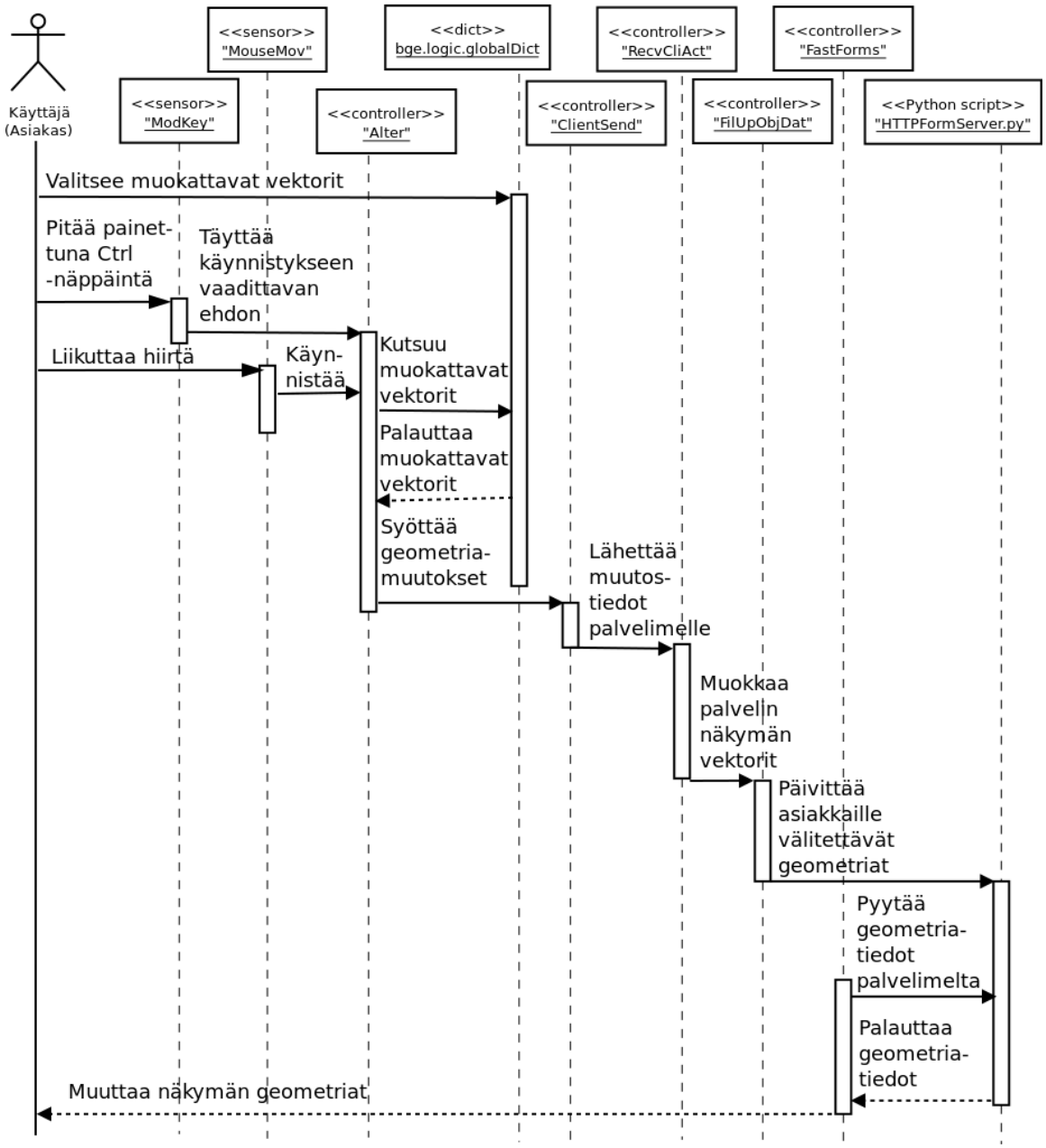


Kuvio 4. Oliokaavio STL-tiedoston tuottamisesta suunnittelunäkymässä kursorilla valitusta kappaleesta.

Kuvio 4 on oliokaavio STL-tiedoston tuottamisesta näkymässä valitusta kappaleesta. Kun käyttäjä siirtää hiiren kursorin kappaleen päälle, kappale tallennetaan BGE-moottorin sisällä kaikkialta saatavilla olevaan kirjastoon. Käyttäjän painaessa +-näppäintä aliohjelma “STLFileGeneration.py” käynnistyy. Aliohjelma kokoaa tallennettavan kappaleen vaipan paikallisen koordinaatiston vektorit ja vektorien järjestysluvuista koostuvat pinnat seuraavalle aliohjelmalle pickle-pakkausmuotoisessa ohjelmassa (liite 6, taulukko 1). Pickle-tiedostoa käyttävän aliohjelman nimi on “ConvertToSTLScript.py”. Se käynnistää Blender -editorin taustaprosessina, tuottaa vastaanottamastaan geometriadatasta uuden Blender-näkymätiedoston ja uuden STL-tiedoston.

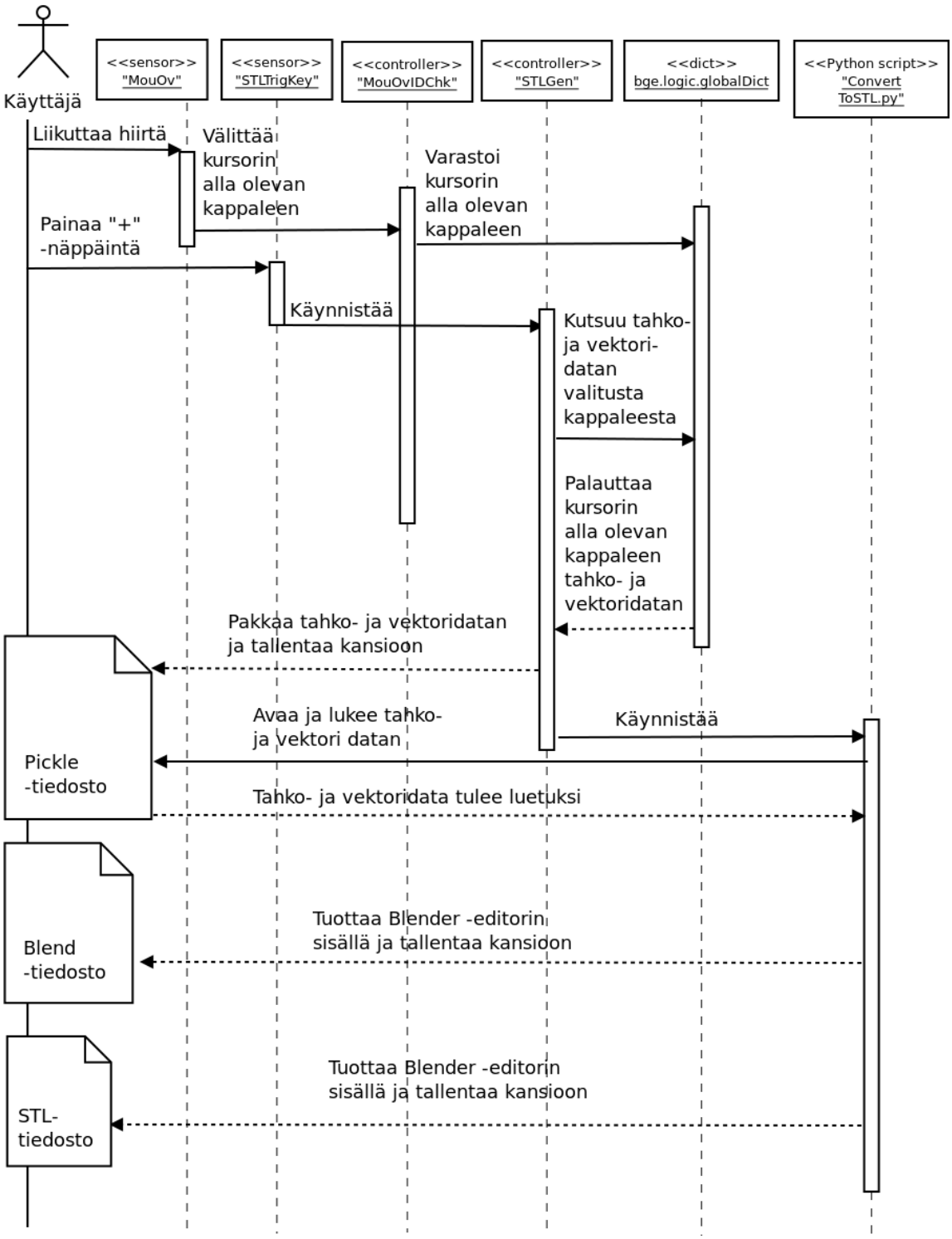
3.6.4 Sekvenssikaavio

Ohjelman toiminnan esittäminen aikajärjestyksessä on osa UML 2.0 -kielen tarkoitusta. Aikajärjestystieto esitetään sekvenssikaavion muodossa. Kaaviossa ylimpänä ovat käyttäjät ja ohjelman komponentit, jotka ovat vastuussa käyttäjän tavoittelemien toimintojen suorittamisesta. Objekteista alaspäin johtavat katkoviivat ovat aikajanoja. Aikajana on levennetty niiltä osin, joiden aikana komponentti on toiminnassa. Täytetyt nuolet merkitsevät synkronisia pyyntöjä toimintojen suorittamiseksi tai datan noutamiseksi. Vaakasuorat katkoviivanuolet ilmaisevat palautettua dataa tai käyttäjälle näkyviä muutoksia.



Kuvio 5. Sekvenssikaavio geometriamuutosten lähetyksestä, muutosten toteuttamisesta palvelimella ja päivittämisestä asiakasohjelman näkymään.

Kuvio 5 on sekvenssikaavio, joka käsittelee geometriadatan muuttamisen palvelimella, muutosviestin lähettämisen palvelimelle ja palvelimella tapahtuneiden geometriamuutosten synkronoinnin asiakasohjelman näkymään. Kaavion alkuoletuksena on, että asiakasohjelman käyttäjä on valinnut ainakin yhden kappalegeometrian vektorin muokattavaksi.



Kuvio 6. Sekvenssikaavio STL-kaavion tuottamisesta suunnittelunäkymässä

Kuvio 6 on sekvenssikaavio, joka käsittelee STL-tiedoston tuottamisen suunnittelunäkymässä valittavasta kappaleesta. Hiiren kursorin alla olevaa kappaletta päivitetään jatkuvasti, ja kun käyttäjä painaa näppäintä +, viimeisin kursorin koskettama kappale haetaan ”globalDict”-hakurakenteesta. Kappaleen paikalliskoordinaattivektorit ja kappaleen tahkojen vektori-indeksit pakataan pickle

-muotoiseen tiedostoon. Aliohjelma “ConvertToSTL.py”-skripti avaa pickle-tiedoston, käynnistää uudessa ohjelmasäikeessä taustalla käyttäjältä piilotetun Blender -editorin ja rakentaa mallin uudelleen pickle-tiedoston sisältämästä datasta. Tämä blend-näkymä tallennetaan tiedostoksi, ja saman ajon aikana siitä tuotetaan STL-tiedosto.

3.6.5 Tietorakenteet

Näkymän tietoja voidaan tallentaa hakurakenteeseen nimeltä ”bge.logic.globalDict”. Näihin tietoihin on pääsy helposti kaikkialta muualta ohjelman sisäpuolelta paitsi näkymään ulkopuolisesta tiedostosta tuotettujen kappaleiden logiikkatiilistä. Globaldictiin alustetaan ja tallennetaan ohjelman käynnistyessä erilaisia muuttujia, joista on tyyppeineen listattu esimerkkejä taulukkoon 1.

Taulukko 1. Hakurakenteen bge.logic.globalDict sisältämiä muuttujia

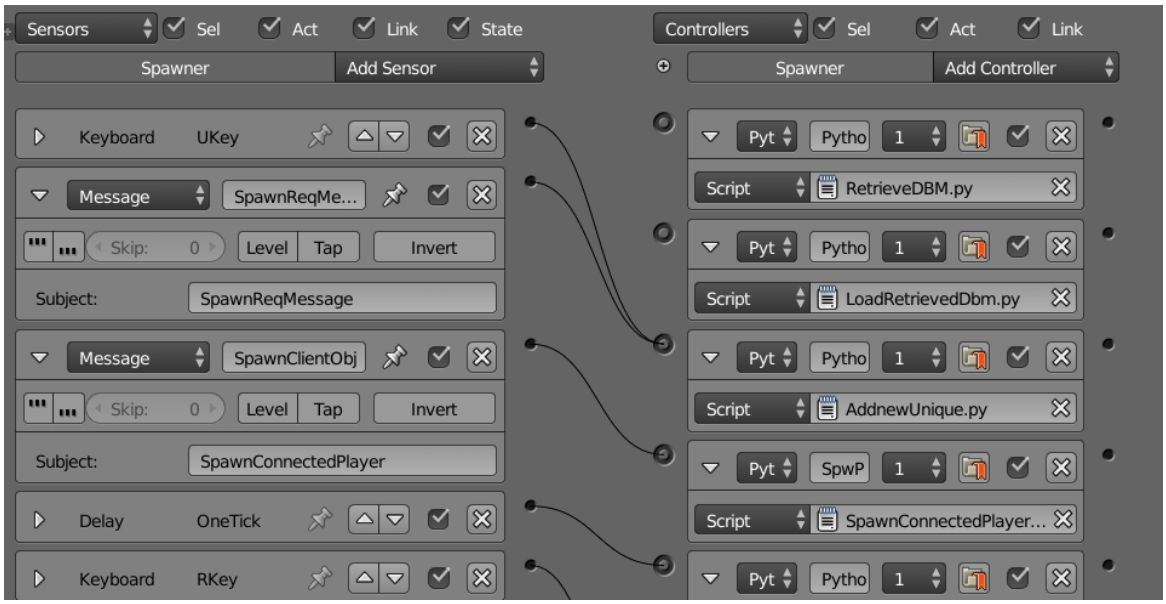
Muuttujan nimi GlobalDictissä	Tyyppi	Tietorakenteen sisältämä tieto
ObjLibraryFolder	merkkijono	Polku kansioon, jossa näkymään lisättävät paikalliset .blend -tiedostot ovat
addUnique	funktio	Mekanismi, jolla kappale lisätään näkymään
saveWorldName	merkkijono	Tallennustiedoston nimen runko
debugCategories	lista	Aiheet, joista ohjelma tulostaa komentoriville korjausta ja kehitystä helpottavia viestejä
PersonalKeys	dict	Painikkeet ja niitä vastaavat viestit, jotka asiakasohjelma lähettää palvelimelle ohjauspyyntöinä
ObjectIDs	lista	Kappaleiden tunnistusnumerot synkronoinnissa
STLObjects	merkkijono	Tiedostopolku, jonne tuotetut STL-tiedostot tallennetaan
refreshIPID	funktio	Päivittää pelaajakappaleiden IP-osoite - tunnistusnumeroparit
importObject	merkkijono	Näkymään tuotettavaksi valitun kappaleen nimi

Näkymän kaikki kappaleet sisältävät logiikkatiiliä, jotka kommunikoivat asynkronisen näkymään saapumisestaan näkymän sisältävän Spawner -peliobjektin ”attrDict” -hakurakenteiden kanssa. Jokainen peliobjekti sisältää hakurakenteen nimeltä Game Properties. Hakurakenteen muuttujatyypit ovat boolean, ajastin, kokonaisluku, liukuluku tai merkkijono.

```
#The object which will be produced as default. Can be removed or altered with little consequence.
gD['importObject'] = "ArchetypeCube"
#The list of IP addresses of connected players
gD['playerIPList'] = []
#The dictionary in which IPs are connected to their respective
#controlled objects
gD['IPsToObjects'] = {}
#The dictionary in which IPs are connected to their respective
#controlled object IDs
gD['IPsToObjIDs'] = {}
#The dictionary which connects IDs to their respective
#game objects
gD['IDsToObjects'] = {}
#Have each ID-GameObject pair be stored in this dictionary
#when the ObjID is assigned during object creation. Reset this
#list when the scene is being loaded.
#Exception: Player objects on the client side will be added
#to the dict when they receive their correct player IDs from the server.
scene.objects["Spawner"].attrDict["IDsToObjects"] = {}
```

Kuva 1. Muuttujien alustusta ja dokumentaatiorivejä ohjelman käynnistyksen yhteydessä suoritettavassa aliohjelmassa “GlobalDictIni.py”

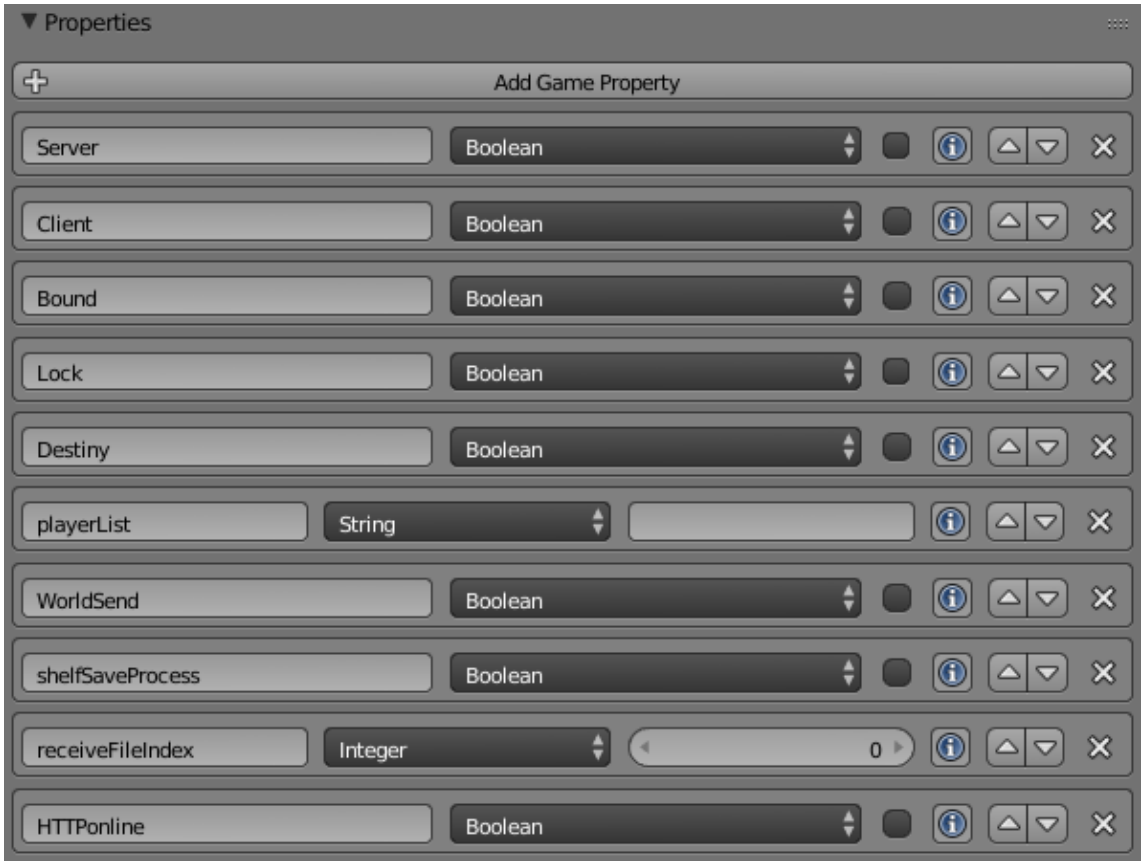
Esimerkki logiikkatiilen sisältämästä koodista on esitetty kuvassa 1. Näiden aliohjelmien sisältöä voi tarkastella ja muuttaa Blenderin Text Editor -näkymässä. Suurin osa ToMC-ohjelman toiminnallisuudesta on kirjoitettu tällaisiin aliohjelmiin.



Kuva 2. Näkymän olion Spawner sisältämiä logiikkatiiliä ja niiden välisiä yhteyksiä.

Kuvassa 2 vasemmalla näkyvien anturien vaikuttuessa suoritetaan oikealla puolella olevien ohjainten sisältämiä Python-aliohjelmia. Musta viiva logiikkatiilten välillä ilmaisee anturin ja ohjaimen välisen yhteyden. Ohjaimesta on vedettävissä yhteys edelleen toimilaitteisiin toiminnallisuuden lisäämiseksi, mutta suurin osa vastaavista ominaisuuksista on kirjoitettu sisään Python-aliohjelmiin. Logiikkatiilet

ja kuvassa 3 esiintyvät pelimuuttujat ovat löydettävissä Blenderin Logic Editor -ikkunasta.



Kuva 3. Näkymän olion ControlHub sisältämiä pelimuuttujia.

Kuvassa 3 on esitetty näkymän objektin “ControlHub” pelimuuttujia. Logiikkatiilet voivat käyttää muuttujia datan varastointiin. Esimerkiksi tiedot siitä, onko ohjelma määritelty palvelin- vai asiakasohjelmaksi, tallennetaan Server- ja Client-muuttujaan. Tieto siitä, että tämä valinta on tehty, varastoidaan muuttujaan Bound.

3.6.6 Säikeet

BGE-moottorin sisällä ajettavat ohjelmasilmukat on ajettava loppuun ennen näkymän seuraavaa virkistystä (Blender Foundation 2017). Esimerkiksi HTTP-palvelinaliohjelmat on pidettävä tämän vuoksi omissa säikeissään, jotta ne eivät pysäyttäisi ohjelman näkymän piirtämistä. Näiden säikeiden kautta tietoverkkoon tuodaan asiakasohjelmien ladattaviksi kappaleiden paikkatiedot, muototiedot,

kappaleiden geometriat sisältävät blend-tiedostot ja pakatut tiedostot, joista palvelinohjelman näkymä puretaan näkyville asiakasohjelmassa.

3.6.7 Oliot

Luokat ovat digitaalisia sapluunoita, joiden avulla on mahdollista tuottaa ohjelmoinnissa monipuolisia tietorakenne- ja funktiovarastoja. Näitä varastoja kutsutaan olioiksi, niiden sisältämiä tietorakenteita attribuuteiksi ja funktioita metodeiksi.

ToMC-ohjelman merkityksellisimmät luokat opinnäytetyön kannalta ovat:

- KX_Scene
- KX_GameObject
- KX_MeshProxy
- SCA_ISensor
- SCA_IController
- SCA_IActuator

”KX_Scene” on näkymä, joka sisältää kaiken ohjelman visuaalisesti esittämän eli ”KX_GameObject”-oliot. Näkymässä on myös ohjelmalogiikkaa sisältäviä tyhjiä ”KX_GameObject”-akseliobjekteja sekä valonlähteitä. Kamerat, joiden kautta näkymää käännetään ja siirretään, ovat myös KX_GameObjecteja.

Käyttäjälle näkyviä osia sisältävät ”KX_GameObject”-oliot sisältävät ”KX_MeshProxy”-oliot, joiden kautta on pääsy kappaleen geometrian muokkaamiseen. Kaikki luettelon SCA-alkuiset oliot ovat myös tavoitettavissa ”KX_GameObject”-olioiden kautta. SCA-tyypin oliot ovat logiikkatiilten pohjarakenteita.

3.6.8 Logiikkatiilet

Blender-editorinäkymässä ToMC-ohjelman ohjauslogiikka on muokattavissa Logic Editor -näkymässä. Jokaiseen ”Game Object”-olioon on määrättävissä omat anturinsa (*eng. sensor*), ohjaimensa (*eng. controller*) ja “toimilaitteensa” (*eng. actuator*). Sensorit tarkkailevat pelinäkymän tai käyttäjän ohjauslaitteiden tiloja, joilla aktuaattorin toimintaa kytketään päälle tai pois. Ohjaimet toimivat logiikkaportteina tai sisältävät itsessään Python-skriptin. Aktuaattorit tekevät muutoksia näkymässä sensorin lähettämien signaalien seurauksena.

3.7 Monen käyttäjän ohjaussignaalien vastaanottaminen

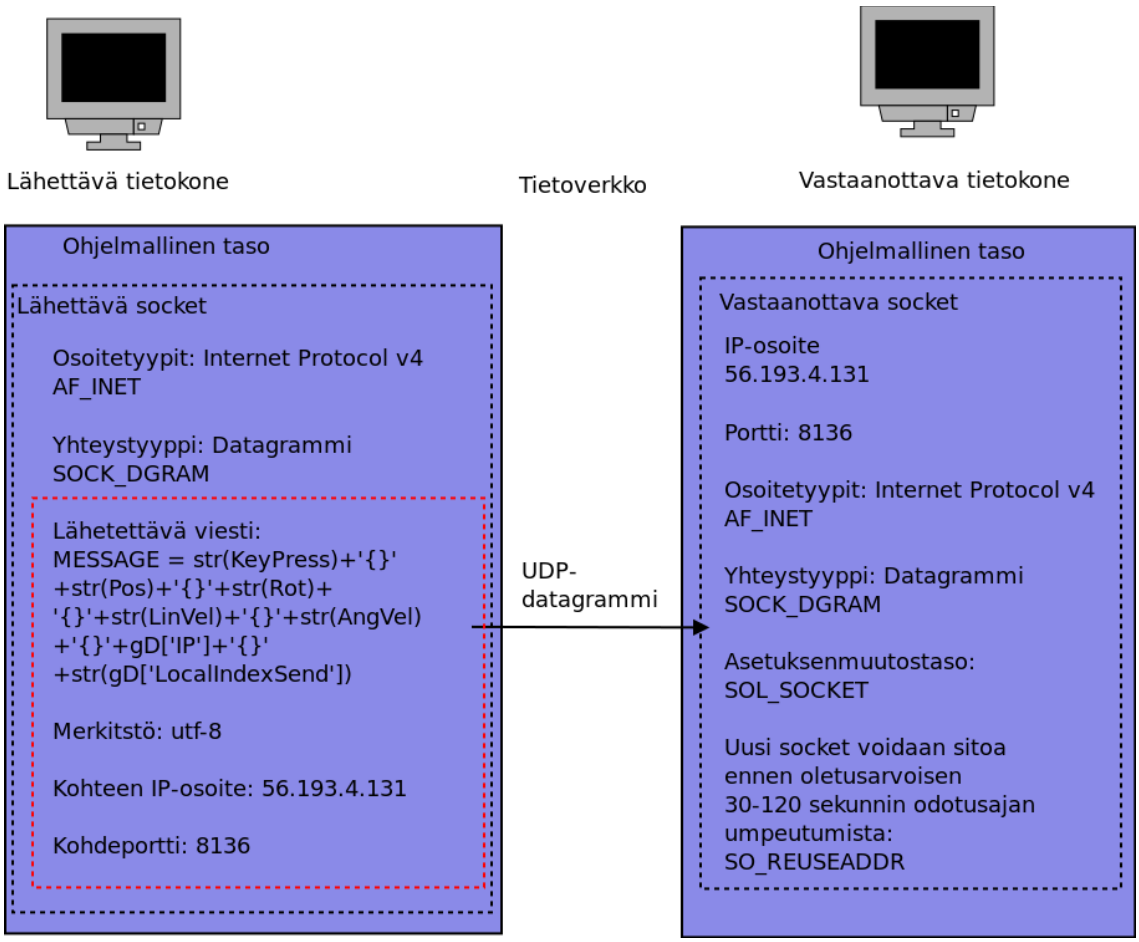
Jotta ohjelman käyttäjät pystyvät mallintamaan kappaleita yhteistyönä reaaliajassa, ohjelman on pystyttävä vastaanottamaan ja lähettämään komentoja tietoverkon yli. Pythonin tietorakenteita ei voi lähettää verkon yli sellaisinaan, ne on muutettava UDP-protokollaa käyttäessä tavujonoiksi ja HTTP-verkkosivua käyttäessä merkkijonoiksi. ToMC-ohjelmassa näihin merkkijonoihin on lisätty erikoismerkkejä, kuten &, | ja #.

Merkit erottelevat tietorakenteiden rajoja ja tietorakenteiden sisälle varastoitujen artikkeleiden rajoja. Vastaanottava ohjelma jakaa viestit näistä kohdista ja palauttaa osat niiden oikeisiin tietorakennemuotoihin. Tietojenkäsittelytieteissä tällaista merkkijonojen käsittelyä kutsutaan parseroinniksi.

3.7.1 UDP-viestien lähettäminen ja vastaanottaminen

ToMC-ohjelman simulaatio-objekteja synkronoiva verkkoviestintä tapahtuu käyttäen UDP-datagrammeja. Koska tähän viestintämuotoon ei sisälly viestin vastaanottamisen varmistamista (Postel 1980), viimeiset simulaatio-objektien tilat julkaistaan asiakasohjelmien saataville varmuuden vuoksi muutaman sekunnin

ajan muutosten päättymisen jälkeen. Palvelimelle liittyneiden asiakkaiden pyynnöt oman avatar-objektinsa liikuttamisesta lähetetään datagrammeina.



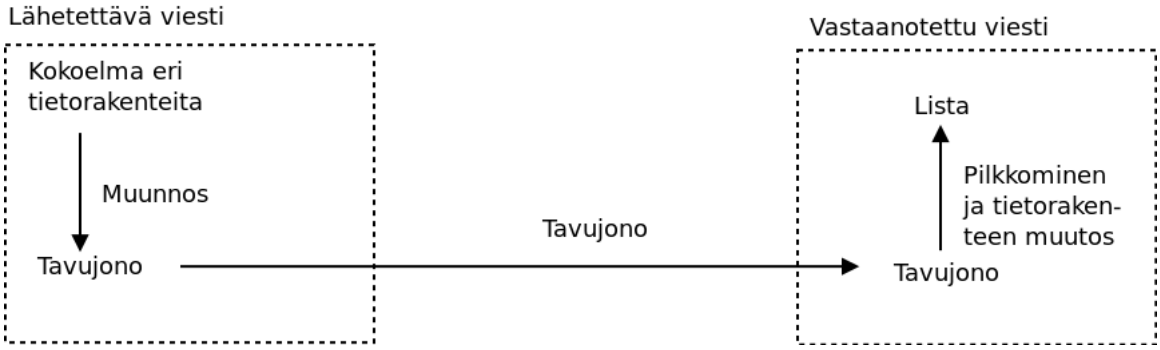
Kuvio 7. Kaavio asiakasohjelman UDP-viestinnästä palvelimen kanssa

Datagrammien lähetys ja vastaanottaminen vaativat ohjelmallisten socket-objektien määrittelyn. Vastaanottava socket on lisäksi sidottava (*eng. bind*) socket-osoitteeseen, joka sisältää tietokoneen IP-osoitteen ja portin, jonka kautta datagrammi vastaanotetaan. Lisätietoja socket-olioihin liitettävistä asetuksista on löydettävissä eri lähteistä (Systutorials 2017, Linux man-pages project 2017).

3.7.2 Vastaanotettujen viestien tulkitseminen palvelimella

Datagrammi saapuu UTF-8-merkistön mukaisena tavujonona, joka käännetään takaisin merkkijonoksi (kuvio 8). Merkkijonon alusta ja lopusta osuudet, jotka eivät vaikuta simulaationäkymään, minkä jälkeen merkkijono jaetaan osiin lähettäjän

merkitsemien vastakkaisten aaltosulkujen {} kohdilta. Viestien vastaanottamista suorittava aliohjelma tarkistaa, että edellisen viestin vastaanotettu aikaleima “gD[‘LocalIndexSend’]” (kuvio 7) on pienempi kuin juuri saapuneen. Viestin Keypress-osa sisältää komennon, joka pyytää avatarin liikettä tai kohdekappaleen geometrian muutosta.



Kuvio 8. Tietorakenteiden lähettäminen ja vastaanottaminen periaatetasolla

3.8 Kappaletietojen synkronointi verkon yli

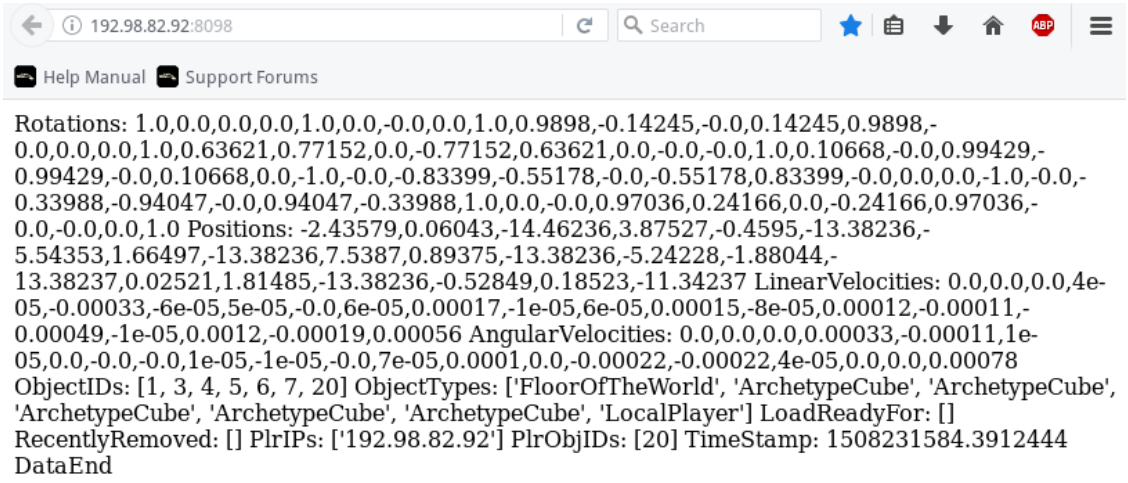
Asiakasohjelmien näkymien synkronointiin tarvittavien tietojen jakamista palvelimelta suorittavat kolme Python-HTTP-ohjelmasäiettä. Säikeet käynnistyvät, kun käyttäjä määrittää ohjelmainstanssin palvelimeksi painikkeella ”Start Game”. Tämän jälkeen asiakasohjelmat avaavat palvelinohjelman verkkosivuja 4 Hz:n taajuudella päivittääkseen näkymänsä muodot ja 7,5 Hz:n taajuudella päivittääkseen näkymänsä liiketilatiedot (Python Software Foundation 2017). Taajuuksien valinnassa on pyritty ehkäisemään palvelimen liiallista kuormittamista. Säikeiden käyttämät portit on valittu vain sillä perusteella, että useimmat verkko-ohjelmat eivät käytä niitä ja siten kärsi niiden kautta kulkevan tietoliikenteen lisääntymisestä (Microsoft 2017).

3.8.1 Liiketilojen synkronointi

Aliohjelmaa tiedostosta “HTTPSpatialServer.py” ajava ohjelmasäie käynnistää verkkosivun, josta löytyvät merkkijonona järjestyksessä seuraavat tiedot kaikista kappaleista:

- kappaleiden kääntökulmat
- kappaleiden paikat avaruudessa
- kappaleiden lineaarinopeudet
- kappaleiden kulmanopeudet
- kappaleiden tunnistusnumerot
- äskettäin näkymästä poistettujen kappaleiden numerot
- näkymässä olevien käyttäjäkappaleiden IP-osoitteet
- pelaajaobjektien tunnistusnumerot
- aikaleima koko näkymästä.

Säie toimii portin 8098 läpi, joten mikäli käyttäjä tietää palvelimen IP-osoitteen olevan esimerkiksi 56.193.4.131, hän voi pyytää palvelimelta edellä listatut tiedot kirjoittamalla tavallisen verkkoselaimen osoitekenttään 56.193.4.131:8098.



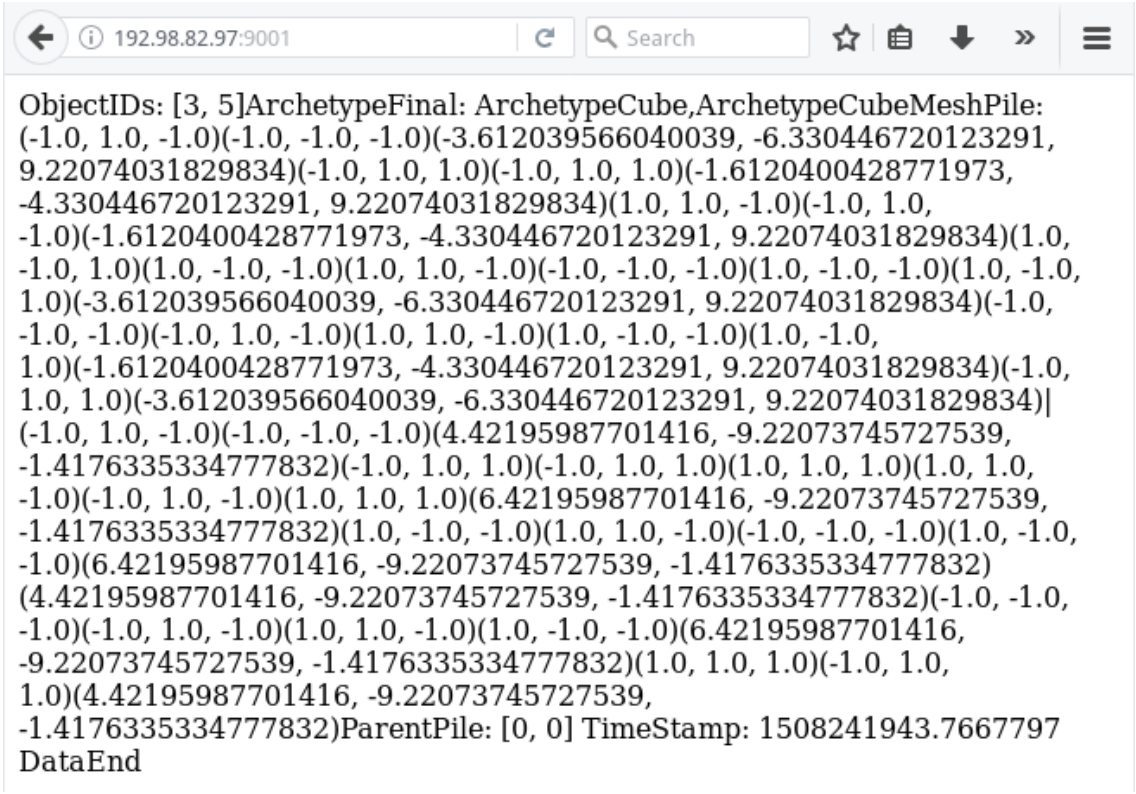
Kuva 4. “HTTPSpatialServer.py”-aliohjelmalla tuotettu verkkosivu avattuna Mozilla Firefox -verkkoselaimella

3.8.2 Muotojen synkronointi

Kun kappale muokkautuu palvelimen näkymässä paikallisen käyttäjän tai asiakasohjelman lähettämän viestin seurauksena, se merkitään Python-listaan, ja

sen geometriatietoja pidetään näkyvillä samanlaisella verkkosivulla kuin liiketiloja kaikille asiakasohjelmille. Verkkosivua pitää yllä BGE-moottorin ulkopuolisessa säikeessä ajettava aliohjelma “HTTPFormServer.py”. Kappale säilyy listassa kolmen sekunnin ajan laskettuna viimeisestä siihen kohdistuneesta muutoksesta. Verkkosivuun otetaan yhteyttä portin 9001 läpi, ja se sisältää järjestyksessä seuraavat tiedot:

- kappaleiden tunnistusnumerot
- kappaleiden tyyppien nimet
- kappaleiden muodot vektoreina paikallisina koordinaatteina
- kappaleiden kiinnitykset toisiin kappaleisiin
- aikaleima.



Kuva 5. “HTTPFormServer.py”-aliohjelmalla tuotettu verkkosivu avattuna Mozilla Firefox -verkkoselaimella.

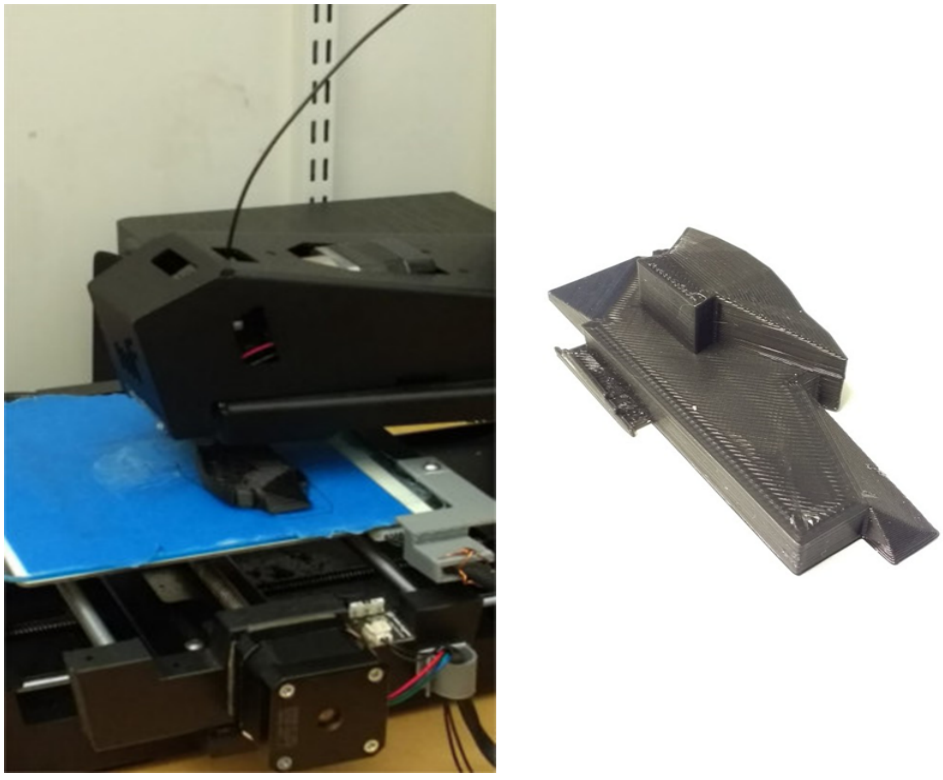
3.9 Blend- ja STL-tiedoston tuottaminen ohjelman kulun aikana

BGE-moottorin sisäinen kappale on mahdollista kopioida omana Blend -tiedostonaan näkymän ulkopuolelle Python-komennolla (liite 2). Blender on

mahdollista käynnistää käyttäjän näkymän ulkopuolella ja antaa käynnistyskäskyn yhteydessä Python-komentoja (Blender Documentation Team 2017). Tässä tapauksessa komento luo ToMC-näkymässä valitun kappaleen nimen perusteella STL-tiedoston, kuten liitteessä 3. Tallennuskansion polku on ohjelman kotikansion sisällä kansiossa “/Exported_Objects/STL/”. Blend-tiedosto tallentuu kansioon “/Exported_Objects”.

3.9.1 3D-tulostuksen koeajo

Tuotettu STL-tiedosto avattiin Slic3r-ohjelmassa, joka tuotti sen avulla G-kooditiedoston. G-kooditiedosto avattiin 3D-tulostuslaitteiston ohjaamiseen suunnitellulla Repetier-Host-ohjelmalla, jonka avulla fyysinen kappale tulostettiin PLA-muovista. Tulostusprosessista ja valmiista tuotteesta otettuja valokuvia on kuvassa 6.



Kuva 6. Vasemmalla 3D-tulostin Minifactory 1.0 valmistamassa ToMC-ohjelmalla muokatun Alvar Aallon suunnitteleman vanhan maakuntakirjaston pienoismallia ja oikealla kuva valmiista pienoismallista

4 Yhteenveto

ToMC on prototyyppinä onnistunut koe käyttää tietoverkkoja synkronoimaan kappaleiden geometria- ja liiketilatietoja Blender Game Engine -näkymien välillä. Ohjelma mahdollistaa alkeellisen kappaleiden tietokoneavusteisen suunnittelun yhteistyönä Internetin vaikutusalueella. Simulaatio-ominaisuudet tekevät näkymästä vuorovaikutteisen ympäristön, mikä pelillistää ohjelman käyttöä.

Prototyypin verkkotoiminnallisuudessa oli työn päättyessä selkeitä kehityskohteita. Asiakasohjelmien näkymissä näkyi värähtelyä geometria- ja spatiaalidatan synkronoinnin yhteydessä. HTTP-sivujen käytön vaihtaminen UDP-protokollaan nopeuttaisi synkronointia, ja tietojen päivittäminen palvelimen näkymään välivaiheittain vähentäisi nähdyn liikkeen katkonaisuutta. Käytetyt Python-palvelimet ovat hyvin läpinäkyviä (Freeney 2014), ja käyttäjien yksityisyyden parantamiseksi on siirryttävä tietoturvaltaan kehittyneempään ympäristöön, kuten Flaskiin (Grinberg 2014, Chapter 8 [Teoksessa ei sivunumeroita]).

Digitaalisia pelejä kritisoidaan siitä, että ajan käyttäminen niihin ei tuota mitään konkreettista tai hyödyllistä. ToMC vastaa tähän ominaisuudella välittää virtuaalisissa maailmoissa tuotettu malli eteenpäin 3D-tulostettavaksi. Vaikka käyttäjällä ei olisi pääsyä tulostimen käyttöön, mallintamistaitojen harjoittelemisen aloittaminen oppilaitoksissa tapahtuvan koulutuksen ulkopuolella luo tulevaisuuden ammattilaissukupolville valmiuksia muotoilla geometrioita yhdessä.

Rakentamis- ja suunnittelupelien kohderyhmät tarjoavat osallistumisellaan valtavan virtauksen henkilötyötunteja toimijalle, joka kykenee tarjoamaan heille riittävät puitteet viihtymiseen. ToMC-ohjelman jatkokehityksen keskiössä olivat tämän virtauksen tavoittamiseksi parempi käytettävyys ja monipuolisemmat geometrianmuokkaustoiminnot. Kun STL-tiedosto on tuotettu, käyttäjän on ohjattava käsin Slic3r- ja Repetier-Host-ohjelmia, eikä loppukäyttäjältä tule tällaista osaamista vaatia. Kuluttajahintaisten tulostimien yleisyys ei työn päättyessä kuitenkaan ollut tarpeeksi suuri, jotta ominaisuuden kehittäminen lähitulevaisuudessa olisi ollut kiireellisin implementoitava toiminnallisuus.

Myös ammattilaiskäyttöön suunnattuja CAD-ohjelmia tuottavat yritykset ovat viime aikoina keskittyneet sujuvoittamaan käyttäjien yhteistyötä ohjelmiensa kautta. Vaikka nämä ohjelmat pitäisivät asemansa työkäytössä vielä vuosikymmeniä, aiheeseen kouluttautumattoman yleisön viihdyttäminen ei todennäköisesti tule olemaan niiden suunnittelun keskiössä. Viihdepelien piirissä dynaaminen visuaalisten- ja törmäysgeometrioiden muokkaus ei ollut toistaiseksi yleistynyt. Tämä tarjosi ToMC-ohjelmalle tyhjän ekologisen lokeron, jossa se ei joutunut kilpailemaan samoilla ominaisuuksilla varustettujen ohjelmien kanssa.

LÄHTEET

Blender Documentation Team. 2017. Command Line Arguments - Blender Manual [www-dokumentti]. Blender Documentation Team. [Viitattu 15.11.2017]. Saatavissa: https://docs.blender.org/manual/en/dev/advanced/command_line/arguments.html

Blender Foundation. 2017. Gotchas — Blender 2.78.0 e8299c8 - API documentation. [www-dokumentti]. Blender Foundation. [Viitattu 14.11.2017]. Saatavissa: https://docs.blender.org/api/blender_python_api_current/info_gotcha.html

Blizzard. 2017. World of Warcraft Beginner's Guide. [Verkkokirja]. Blizzard Entertainment. [Viitattu 9.11.2017]. Saatavissa: http://media.battle.net/documents/wow/WoW-BradyGAMES-enUS-Guide.pdf

Boden, M. A. 1998. Creativity and artificial intelligence. Artificial Intelligence. 103(1–2), 347–356.

Breuer, J. & Bente, G. 2010. Why So Serious? On the Relation of Serious Games and Learning. Eludamos. Journal for Computer Game Culture 4(1), 7–24.

Brown, A. C. & de Beer, D. 2013. Development of a stereolithography (STL) slicing and G-code generation algorithm for an entry level 3-D printer. Teoksessa 2013 Africon. IEEE, 1–5.

Brown, T., Li, H., Nguyen, A., Rivera, C. & Wu, A. 2013. Development of Tangential Learning in Video Games. [www-dokumentti]. [Viitattu 26.2.2018]. Saatavissa: http://www.seas.upenn.edu/~cse400/CSE400_2013_2014/reports/07_report.pdf

Chakravorty, D. 2017. STL File Format for 3D Printing - Simply Explained. [www-dokumentti]. All3DP. [Viitattu 13.11.2017]. Saatavissa: https://all3dp.com/what-is-stl-file-format-extension-3d-printing/#pointone

Deffenbaugh, M. D., Johnson, D.,Yuan, B., Lutz, P. 2012. A Physical Channel in a Digital World. Teoksessa Proceedings of the International Conference on Security and Management (SAM). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp) 1–4.

Electronic Arts. 1999. System Shock 2 User Manual. [Verkkokirja]. Electronic Arts Limited. [Viitattu 9.11.2017]. Saatavissa: http://cdn.akamai.steamstatic.com/steam/apps/238210/manuals/System%20Shock%202%20-%20Manual.pdf

Fantino, E. 1987. Operant Conditioning Simulations of Foraging and the Delay-Reduction Hypothesis. Teoksessa: Kamil A.C., Krebs J.R., Pulliam H.R. (eds) Foraging Behavior. Boston. Springer. (Abstrakti)

Freeney, S. 2014. SimpleHTTPServer considered harmful. [Blogikirjoitus]. [Viitattu 9.11.2017]. Saatavissa: https://scott.mn/2014/01/05/simplehttpserver_considered_harmful/

Fuh, J.Y.H. & Li, W. D. 2005. Advances in collaborative CAD: the-state-of-the-art. Computer-Aided Design 37, 571–581.

Graphisoft. 2017. Teamwork Functions at a Glance | Help Center | ARCHICAD, BIMx, BIM Server knowledge base from GRAPHISOFT. [www-dokumentti]. Graphisoft SE. [Viitattu 10.11.2017]. Saatavissa:https://helpcenter.graphisoft.com/guides/archicad-20/archicad-20-reference-guide/collaboration/teamwork/teamwork_functions_at_a_glance/#1153399

Grässle, P., Baumann, H. & Baumann, P. 2005. UML 2.0 in Action: A project-based tutorial. Birmingham: Packt Publishing.

Grinberg, M. 2014. Flask Web Development: Developing Web Applications with Python. 2. uud. p. Sebastopol: O'Reilly Media Incorporated.

Halevy, A., Doan, A. & Ramakrishnan, R. 2011. Crowdsourcing systems on the World-Wide Web. Communications of the ACM 54(4), 86.

Hamari, J., Koivisto, J. & Sarsa, H. 2014. Does Gamification Work? -- A Literature Review of Empirical Studies on Gamification. Teoksessa: 2014 47th Hawaii International Conference on System Sciences. IEEE, 3025–3034.

Jilani, Y., Kadobayashi, R. & Jozen, T. 2014. Do adults want to learn by playing digital games? Teoksessa: 2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE). IEEE, 496–499.

Kaplan, A. M. & Haenlein, M. 2009. The fairyland of Second Life: Virtual social worlds and how to use them. Business Horizons 52(6), 563–572.

Kirsch, I., Lynn, S. J., Vigorito, M. & Miller, R. R. 2004. The role of cognition in classical and operant conditioning. Journal of Clinical Psychology 60(4), 369–392.

Kleffner, R., Flatten, J., Leaver-Fay, A., Baker, D., Siegel, J. B., Khatib, F., Cooper, S. 2017. Foldit Standalone: a video game-derived protein structure manipulation interface using Rosetta. Oxford: Oxford Academic Press. Bioinformatics 33(17), 2765–2767.

Linux man-pages project. 2017. Socket(2) - Linux manual page. [www-dokumentti]. [Viitattu 3.11.2017]. Saatavissa: http://man7.org/linux/man-pages/man2/socket.2.html

Maslow, A. H. 1943. A theory of human motivation. Psychological Review 50(4), 370–396.

McGonigal, J. 2011. Reality Is Broken: Why Games Make Us Better and How They Can Change the World. London: Vintage.

Messinger, P. R., Stroulia, E. & Lyons, K. 2008. A Typology of Virtual Worlds: Historical Overview and Future Directions. Journal of Virtual Worlds Research 1(1), 1–18.

Microsoft. 2017. Avoiding TCP/IP Port Exhaustion.[www-dokumentti]. Microsoft Corporation. [Viitattu 20.11.2017]. Saatavissa: https://msdn.microsoft.com/en-us/library/aa560610(v=bts.20).aspx

Pan, M. & Felinto, D. 2013. Game Development with Blender. 1. p. Boston: Cengage Learning PTR.

Phillips, D. 2010. Python 3 Object Oriented Programming. 1st p. Birmingham: Packt Publishing.

Postel, J. 1980. User Datagram Protocol, [www-dokumentti]. Internet Engineering Task Force. [Viitattu 9.11.2017] Saatavissa: https://www.ietf.org/rfc/rfc768.txt

Python Software Foundation. 2017. 21.6. urllib.request — Extensible library for opening URLs — Python 3.6.3 documentation. [www-dokumentti]. Python Software Foundation. [Viitattu 14.11.2017]. Saatavissa: https://docs.python.org/3/library/urllib.request.html

Rintala, M. & Jokinen, J. 2000. Olioiden ohjelmointi C++:lla. Jyväskylä: Suomen ATK-kustannus.

Rowland, T. Ei päiväystä. Manifold -- from Wolfram MathWorld. [www-dokumentti]. Wolfram Research Inc. [Viitattu 24.11.2017]. Saatavissa: http://mathworld.wolfram.com/Manifold.html

Sanner, M. F. 1999. Python: A Programming Language for Software Integration and Development. Journal of Molecular Graphics and Modelling 17(1) 57–84.

Schmitz, B. 2014. How Engineers are Collaborating Today [www-dokumentti]. 3D CAD World. [Viitattu 8.11.2017]. Saatavissa: http://www.3dcadworld.com/engineers-collaborating-today/

Shea, K., Aish, R. & Gourtovaia, M. 2005. Towards integrated performance-driven generative design tools. Automation in Construction 14(2), 253–264.

Short, D. 2012. Teaching scientific concepts using a virtual world – Minecraft. [www-dokumentti]. Teaching Science 58(3).[Viitattu 19.11.2017]. Saatavissa: https://www.researchgate.net/profile/Dan_Short/publication/236587414_Teaching_Scientific_Concepts_using_a_Virtual_World_-_Minecraft/links/00b49518172e4dc83d000000/Teaching-Scientific-Concepts-using-a-Virtual-World-Minecraft.pdf.

Solidface. 2017. SolidFace Collaborative 3D CAD software. [Video]. SolidFace Technology Incorporated. [Viitattu 8.11.2017]. Saatavissa: https://www.youtube.com/watch?v=Oh6xJK4TNIE

Surowiecki, J. 2004. The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations. Anchor: Doubleday.

St-Pierre, R. 2011. Learning with Video Games. Teoksessa: Felicia, P. (toim.) Handbook of Research on Improving Learning and Motivation through Educational Games: Multidisciplinary Approaches. Irlanti: Waterford Institute of Technology. 74-96.

Systutorials. 2017. Setsockopt - get and set options on sockets - Linux Man Pages (2). [www-dokumentti]. Systutorials. [Viitattu 20.11.2017]. Saatavissa: https://www.systutorials.com/docs/linux/man/2-setsockopt/

The Rift Develompent Team. 2011. Rift Online Game Manual 1.0. [Verkkokirja] Trion Worlds Incorporated. [Viitattu 17.11.2017]. Saatavissa: http://file3.guildlaunch.net/196511/RIFT_Manual_EN.pdf

Thomas-Lepore, G. 2013. Top Five CAD Collaboration Fails. [www-dokumentti]. 3D CAD World. [Viitattu 8.11.2017]. Saatavissa: http://blog.grabcad.com/blog/2013/05/07/top-five-cad-collaboration-fails/

Turney, D. 2016. Revving Up for a Manufacturing Revolution with Hack Rod, Redshift by Autodesk. [www-dokumentti]. Autodesk Incorporated. [Viitattu 14.11.2017]. Saatavissa: https://www.autodesk.com/redshift/hack-rod

UW Center for Game Science, UW Institute for Protein Design, Northeastern University, Vanderbilt University Meiler Lab, UC Davis. 2017. The Science Behind FoldIt | Foldit. [Verkkosivusto]. [Viitattu 24.10.2017]. Saatavissa: http://fold.it/portal/info/about

LIITTEET

Ulkoisella USB-muistilla:

Liite 1: ToMC:n versio opinnäytetyön päättyessä oheistiedostoineen

Liite 2: Blend -tiedosto, jota käytettiin STL-tiedoston tuottamiseen

Liite 3: STL-tiedosto, jota käytettiin G-kooditiedoston tuottamiseen

Liite 4: G-kooditiedosto, jonka avulla tulostettiin fyysinen kappale

Liite 5: G-koodin tuottamiseen käytetty Slic3r –konfiguraatiotiedosto

Paperiliitteinä:

Liite 6: Tiivistelmä käytetyistä tietoteknisistä työkaluista ja niiden lisäkirjastoista

Liite 7: ToMC käyttöohje

ToMC - käytetyt tietotekniset työkalut ja niiden lisäkirjastot

Tämä dokumentti sisältää verkkosivut, joista ohjelman tuottamiseen tarvittavat työkalut tai moduulit löytyvät. Lisäksi mukana on linkkejä ToMC:n jatkokehityksen kannalta olennaisimpiin käyttöohjeisiin, kuten ohjelmointirajapinnan dokumentaatioon. Kaikki ohjelmat ja niiden kirjastot olivat työn aikana saatavilla Internetistä veloituksetta.

Tavoitteena on tarjota lähtötiedot samanlaisen ohjelmaympäristön tuottamiseen kuin opinnäytetyön lopussa oli käytössä. USB-muistille tallennettujen tiedostojen virkistäminen kopioimalla sisältö laitteelle uudelleen 10 vuoden välein on suositeltavaa tietojen häviämisen ehkäisemiseksi. ToMC:n viimeisin versio löytyy osoitteesta https://github.com/Blastra/ToMC.

A. MALLINNUS

Blender

Blender on ilmainen mallinnus- ja simulaatio-ohjelma, jonka lähdekoodi on avointa. Yksi sen toiminnoista on sisäänrakennettu simulaatiotila, BGE eli Blender Game Engine. ToMC käyttää BGE:n sisältämiä toimintoja, kuten joustamattomien objektien (eng. Rigid body) fysiikkamallinnusta ja uusien objektien tuottamista näkymään ohjelman ulkopuolisista tiedostoista.

Blender on saatavilla osoitteessa https://www.blender.org/download/ [Viitattu 23.10.2017]

Blenderin fysiikkamallinnuksesta voi lukea lisää osoitteessa http://bulletphysics.org/wordpress/ [Viitattu 23.10.2017]

B. OHJELMOINTI

Python 3

Python on ohjelmointikieli, jonka avulla BGE:n simulointinäkymän toimintaa voi ohjata. Se on korkean tason yleisohjelmointikieli, johon on saatavilla Internetistä lisäominaisuuksia sisältäviä moduuleja.

Alla olevassa taulukossa on luetteloituna ohjelmassa käytettyjä Python -kirjastoja ja lyhyet kuvaukset niiden funktioiden ja muuttujien tarkoituksesta ohjelmakokonaisuudessa.

bge	Blenderin pelimoottorin näkymän kappaleet, toiminnot ja muuttujat
bpy	Blenderin editorinäkymän toiminnot
dbm	Rajapinta Unix-tietokantoihin
http	Verkkosivujen käynnistäminen ja tiedostonsiirto niiden kautta
math-utils	Matemaattisia apufunktioita ja muuttujatyyppejä
os	Tiedostopolut ja -koot, kansioiden tuottaminen ja käyttöjärjestelmien erot näissä
pickle	Muuttujien tallentaminen ulkoiseen binääritiedostoon
select	Internet-sockettiin saapuvan datan odottaminen

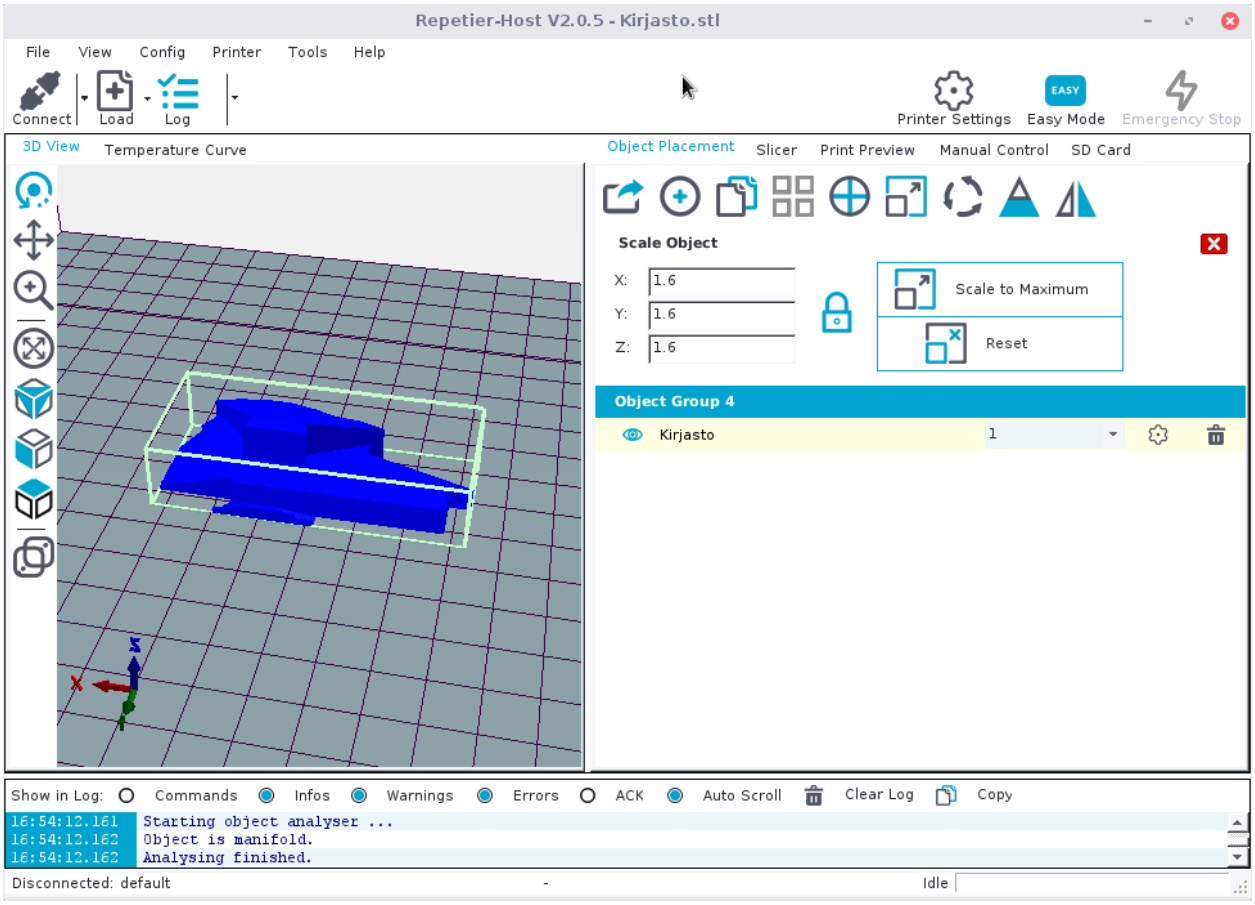
shelve	Käyttäjäkohtaisen konfiguraatiotiedoston tallentaminen ja avaaminen
socket	UDP-viestien lähettäminen ja vastaanottaminen
sys	Käyttöjärjestelmästä johtuvat tiedostopolut skripteihin ja aliohjelmien käynnistämiseen
threading	Ohjelmasäikeiden käynnistäminen ja sulkeminen
time	Aikaleimojen tuottaminen ja vertailu
urllib	Datan hakeminen verkkosivuilta

Taulukko 1: Käytetyt Python -ohjelmointikirjastot ja niiden käyttö ohjelmassa

C. G-KOODIN TUOTTAMINEN

Slic3r

STL-tiedostomuodon muuttaminen 3D-tulostuksessa tarvittavaksi G-koodiksi edellyttää ohjelmallista käsittelyä. Slic3r on tätä tarkoitusta varten tuotettu ohjelma, ja sen kalibrointi käytettävän 3D-tulostuslaitteiston mukaiseksi parantaa työkappaleen onnistuneen tulostuksen todennäköisyyttä. Slic3r on saatavissa verkosta ilmaiseksi, ja sen lähdekoodiin voi tarjota ohjelmoimiaan parannuksia Githubin kautta.



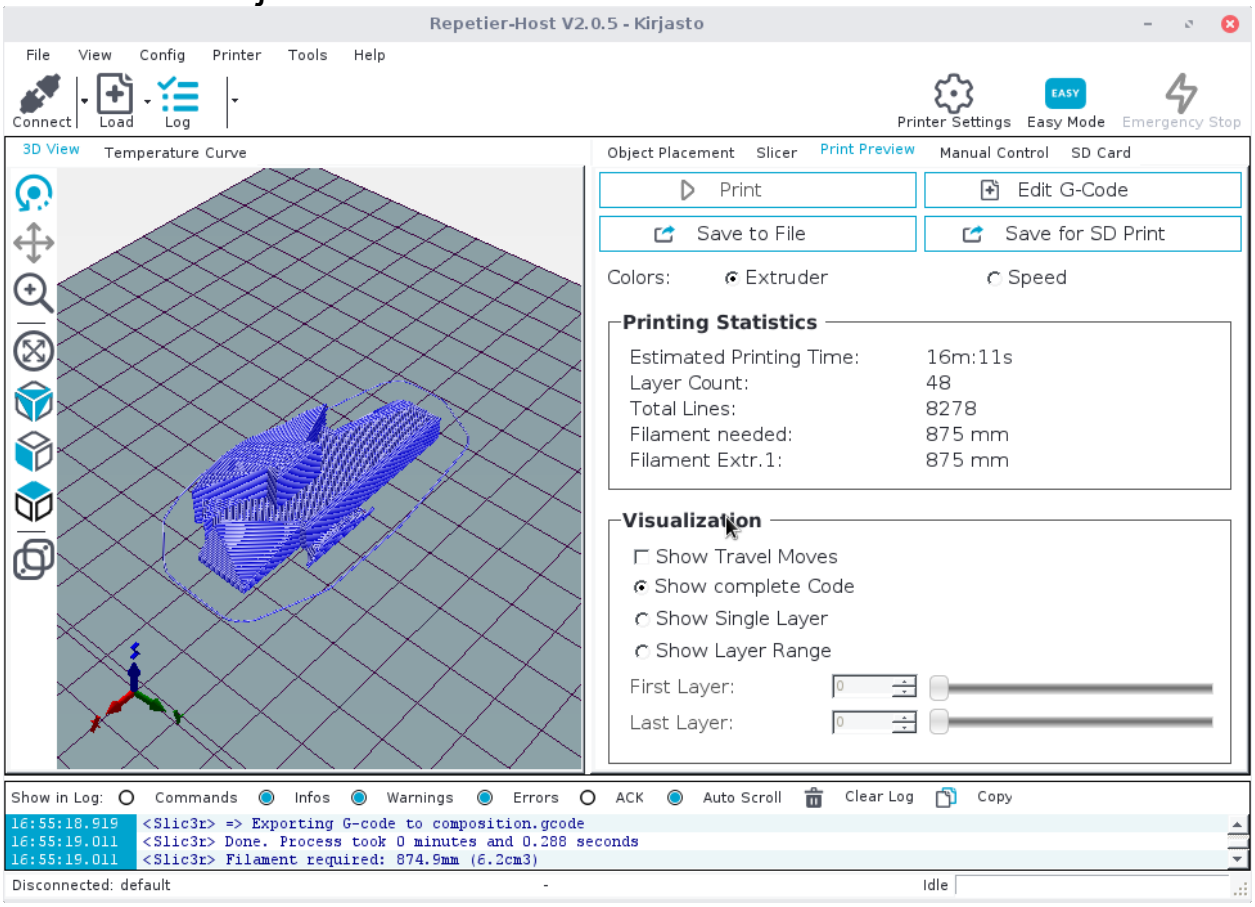
Kuva 1. Repetier-Hostin näkymä STL-tiedoston tuonnin jälkeen

D. 3D -TULOSTAMINEN

Repetier-Host

3D-tulostimen ohjaukseen voi käyttää esimerkiksi ilmaiseksi saatavilla olevaa ohjelmaa Repetier-Host. Se ottaa syötteekseen Slic3r -ohjelman tuottaman G-koodin ja ohjaa tulostimen X- Y- ja Z-suuntaisia servomoottoreita, filamentinsyöttömoottoria, filamentin sulattamiseen käytettävää lämmitysvastusta sekä mahdollisia jäähdytystuulettimia ja tulostusalustan lämmitysvastuksia. Ohjaukseen käytettävä tietokone ja 3D-tulostin viestivät tämän opinnäytetyön tapauksissa USB-väylän kautta. Repetier-Host valittiin käyttötarkoitukseen, koska se oli

ilmainen ja yleisesti käytetty ratkaisu kuluttajahintaisten 3D-tulostinten ohjaamisessa.



Kuva 2. Repetier-Hostin näkymä G-koodin generoinnin jälkeen

E. TEKSTUURIEN TUOTTAMINEN

GNU Image Manipulation Program

Ohjelman valikoiden tekstuurit tuotettiin ohjelmalla GNU Image Manipulation Program (GIMP).

GIMP on saatavilla osoitteesta https://www.gimp.org/downloads/, ja sen ohjekirja on myös saatavilla samasta osoitteesta.

F. VERSIONHALLINTA

Git

Versionhallinnan työkaluna käytettiin Git -versionhallinta-ohjelmaa, joka toi opinnäytetyön etenevän ohjelmalopputuloksen julkiseen jakoon Github-verkkosivun kautta. Git valittiin tarkoitukseen niillä perusteilla, että se on ilmainen ja käyttöliittymältään kevyt. Muutosten päivittäminen julkiseen tietokantaan onnistuu kolmella terminaalikomennolla ja salasanan kirjoittamisella. Paikallisen ohjelman päivittäminen onnistuu yhdellä komennolla.

Git -ohjelma saatavilla:
https://git-scm.com/download [Viitattu 23.10.2017]

GitHub -sivusto:
https://github.com/ [Viitattu 23.10.2017]

G. KÄYTTÖJÄRJESTELMÄ

Linux Lite

Ohjelman toimintaa on testattu enimmäkseen lähiverkoissa käyttäen ulkoisia IP-osoitteita tiedonsiirrossa. Käyttöjärjestelminä toimivat Ubuntu Linux -pohjainen Linux Lite 3.6 sekä Ubuntu 16.04. Opinnäytetyön työosuuden päättyessä käytetty Blenderin versio oli 2.79. Palvelinohjelmaa ajavan käyttäjän on avattava portit 8098, 8099, 9001 ja 9002 tcp-viestien vastaanottamista varten. Porttien avauskomennot tcp-liikenteelle Linuxin komentorivillä ovat:

```
sudo iptables -A INPUT -p tcp --dport 8098 -j ACCEPT
sudo iptables -A INPUT -p tcp --dport 8099 -j ACCEPT
```

```
sudo iptables -A INPUT -p tcp --dport 9001 -j ACCEPT
sudo iptables -A INPUT -p tcp --dport 9002 -j ACCEPT
```

Huomioitavaa on, että portit ovat auki tällöin liikenteelle kaikista IP-osoitteista, mikä on tietoturvariski. Komennon suorittaminen tarvitsee järjestelmänvalvojan salasanan. Myös käytettävät HTTP-säikeiden käyttämät palvelinobjektit ovat tietoturvariski.

Mikäli udp-liikenne portista 8136 ei saavuta palvelinta, komento

```
sudo iptables -A INPUT -p udp --dport 8136 -j ACCEPT
```

avaa saapuvalle liikenteelle portin.

Linux Lite on saatavilla osoitteesta: https://www.linuxliteos.com/download.php [Viitattu 23.10.2017]

Liite 7. Ohjelman käyttöohje

ToMC asennus- ja käyttöohje

Tämän käyttöohjeen tarkoituksena on neuvoa käyttäjä läpi tietokoneohjelma ToMC:n käynnistämiseen tarvittavien eri ohjelmien asennusten ja ohjelman omien valikoiden suunnittelunäkymään asti.

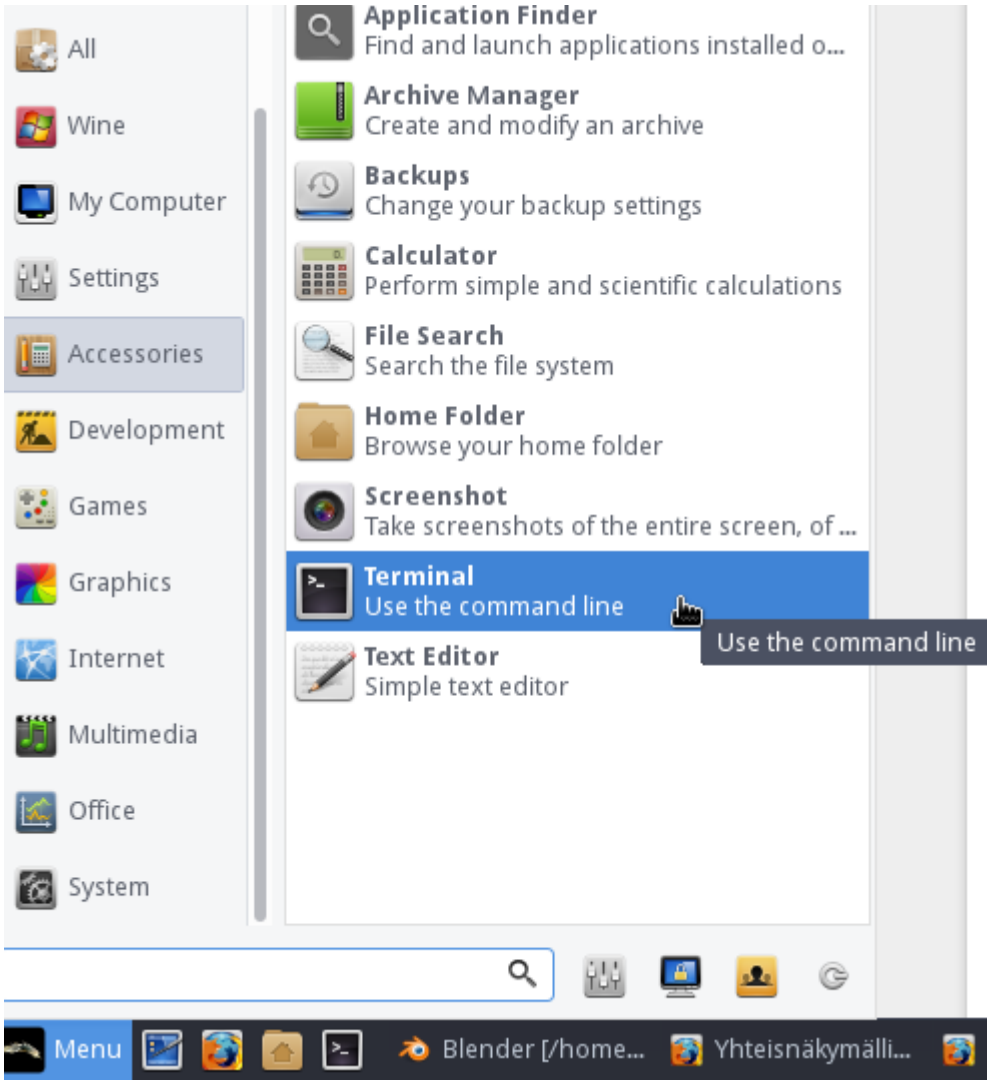
Mikäli ohjelma on jo asennettu, voit siirtyä kohtaan ToMC:n käynnistys. Ohjelman viimeisin versio on saatavilla verkosta osoitteessa: https://github.com/Blastra/ToMC

BLENDERIN ASENNUS

Blenderin viimeisin versio on todennäköisesti yhteensopiva ToMC:n nykyisen tiedoston kanssa, mutta liitteestä 6 löytyy Blender 2.79, jonka kopiointi käytettävälle tietokoneelle käy asennuksesta Linuxilla.

PYTHON 3:N JA LISÄKIRJASTOJEN ASENNUS

Python 3:n asentaminen ja käyttäminen Blenderin ulkopuolella onnistuu Linuxin terminaalista käsin. Terminaalin voi avata näppäinyhdistelmällä Ctrl + Alt + T tai vasemman alakulmasta löytyvän Menun Accessories -valikosta. Terminaalia kutsutaan myös komentoriviksi.



Terminaalin käynnistys Menu -valikosta käsin

Python 3

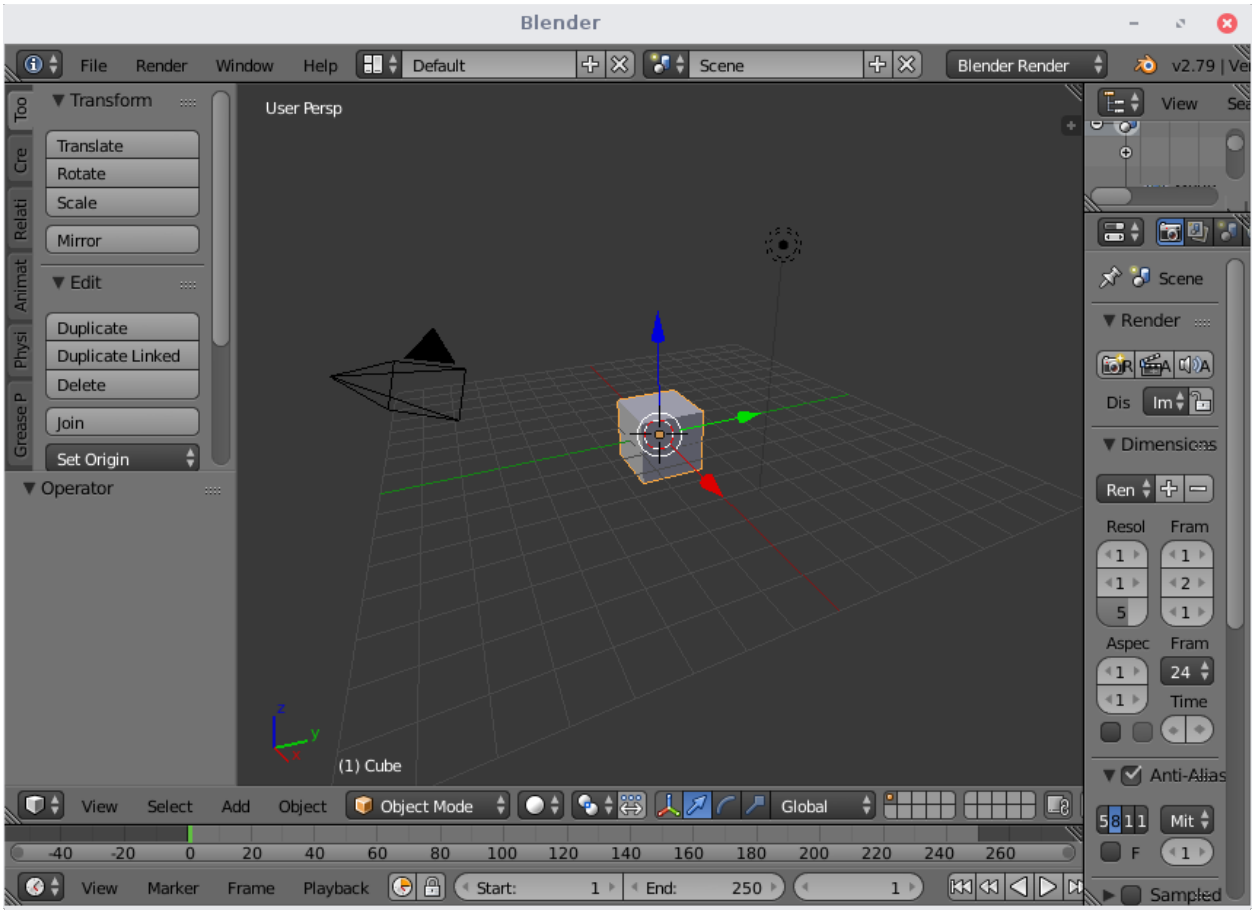
Python 3:n asennus Linuxilla, tässä tapauksessa, vaatii seuraavan komennon terminaalissa:

```
sudo apt-get install python3
```

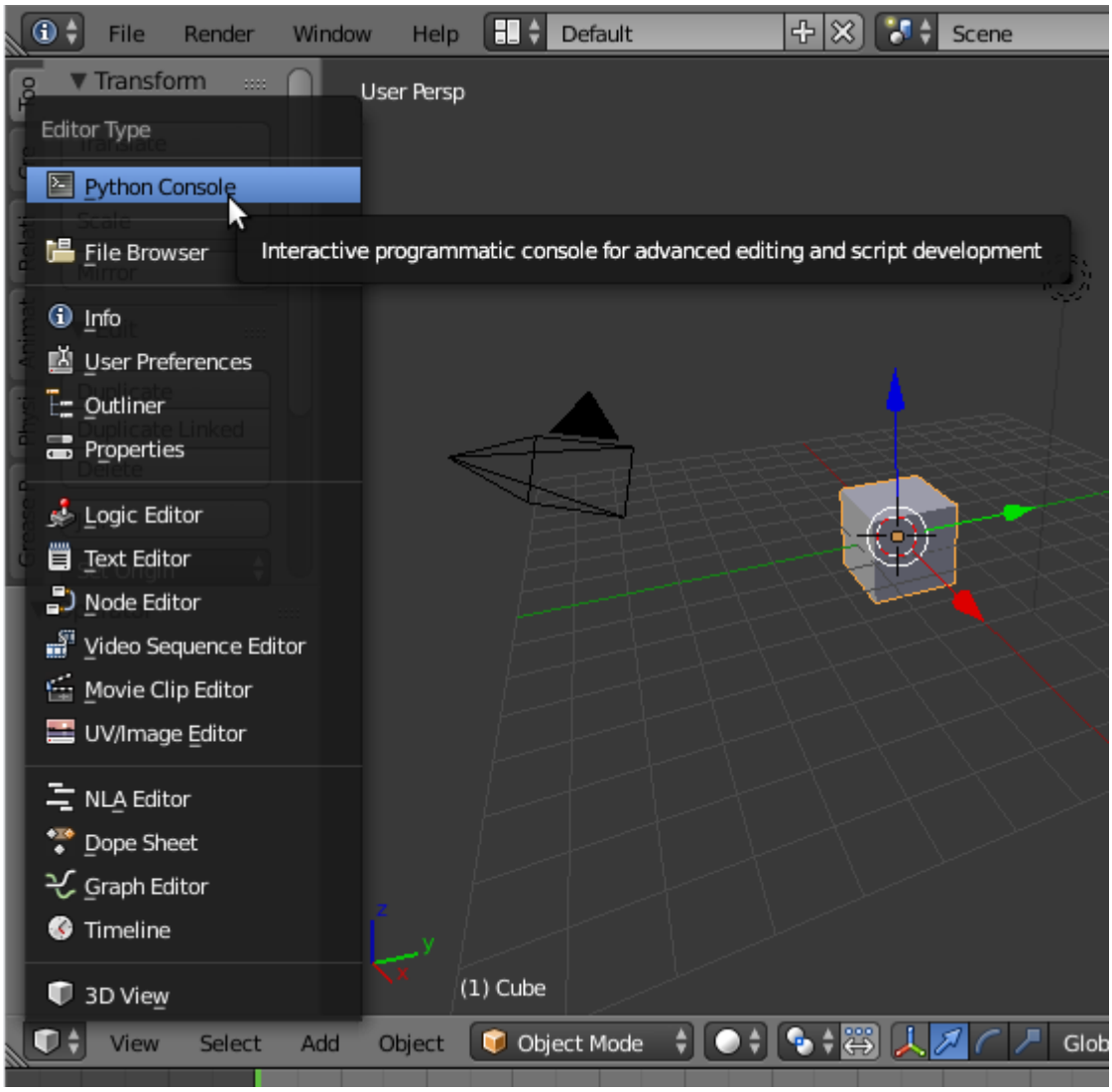
Ohjelmien asennus vaatii käyttäjän salasanan, ja että järjestelmänvalvoja ei ole rajoittanut ohjelmien asentamista käyttäjältä.

Lisäkirjastojen tarkistus

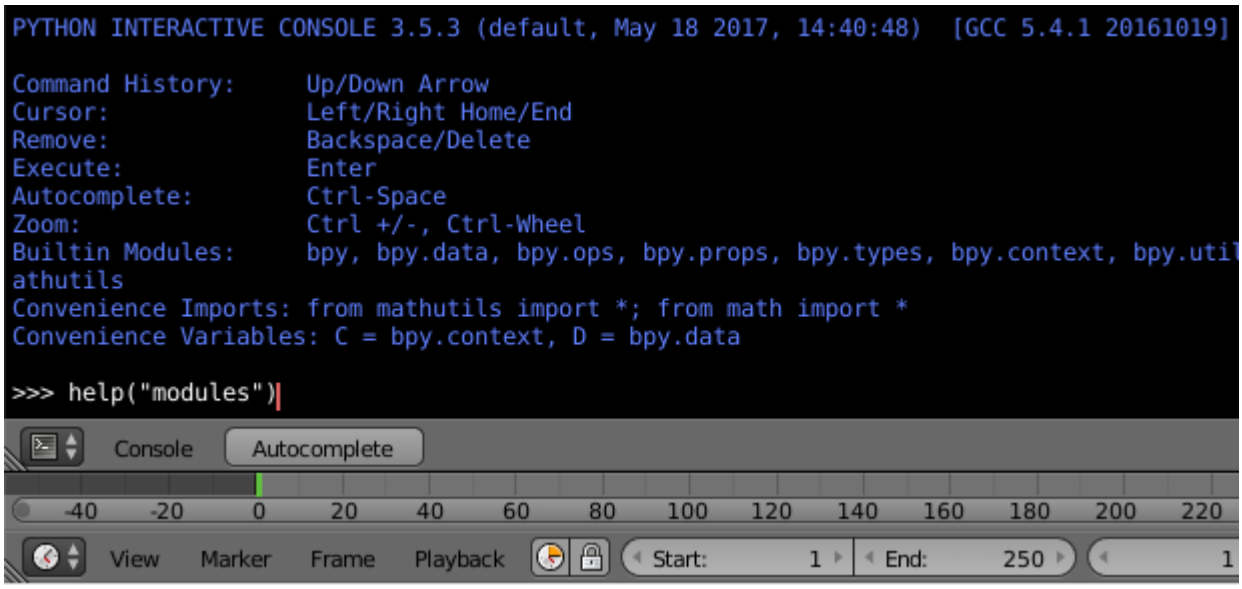
Tarkistaaksesi, ovatko tarpeelliset lisäkirjastot Blenderin käyttämän sisäänrakennetun Pythonin käytettävissä, suorita seuraavat askeleet:



Käynnistä Blender.



Tee yhdestä editorin ikkunasta Python Console.



Kirjoita Python Consoleen help(“modules”).

Jos tulostuvassa lisäkirjastojen listassa ei ole jotakin liite 9B:ssä mainituista lisäkirjastoista bge:tä lukuun ottamatta, kirjaston voi asentaa käyttämällä pip-pakettienhallintaohjelmaa.

Pip

Yksi vaihtoehto Python 3:n lisäkirjastojen asentamiseen on pakettienhallintaohjelma pip. Sen asentaminen vaatii komennon:

```
sudo apt-get install python3-pip
```

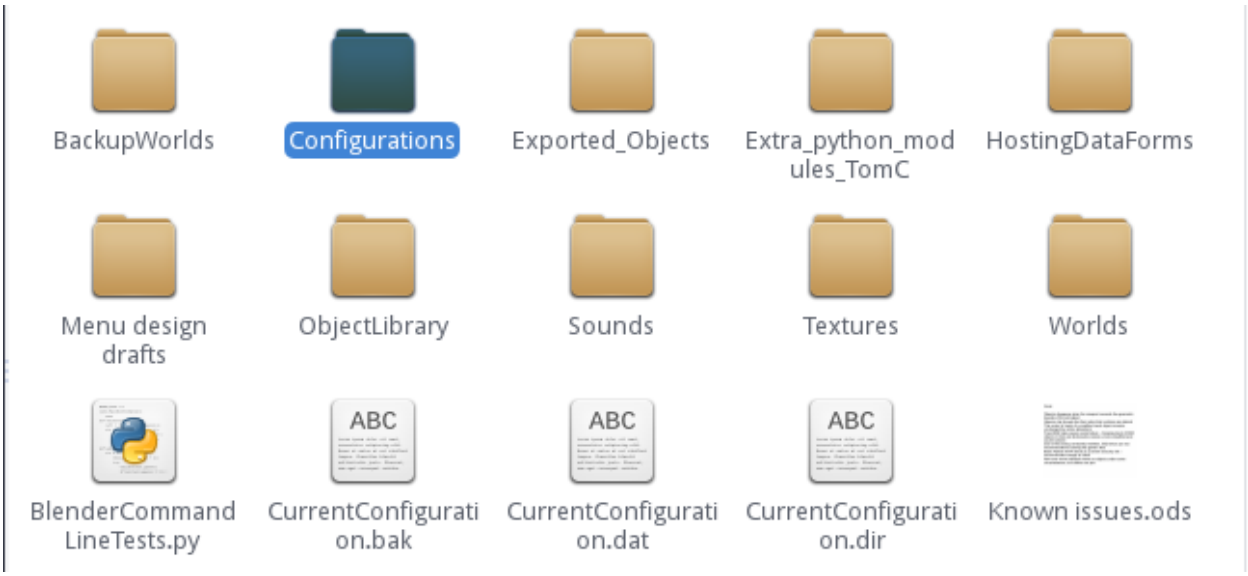
Tämän jälkeen Python 3:een voi asentaa lisäkirjaston terminaalissa komennolla (esimerkkinä urllib):

```
sudo pip3 install urllib
```

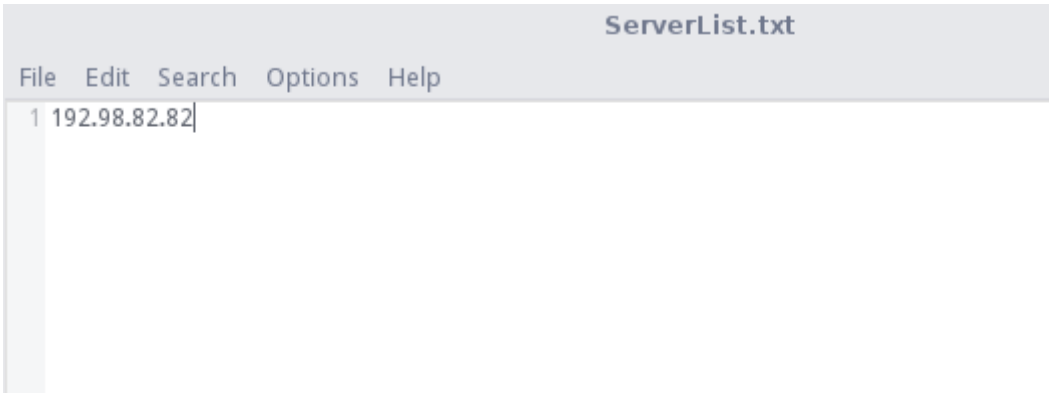
TOMC:N KÄYNNISTYS

Palvelimen IP-osoitteen määrittäminen

ToMC:n juurikansiossa (Kuva) sijaitsee Configurations -kansio. Tässä kansiossa on tekstitiedosto ServerList.txt.



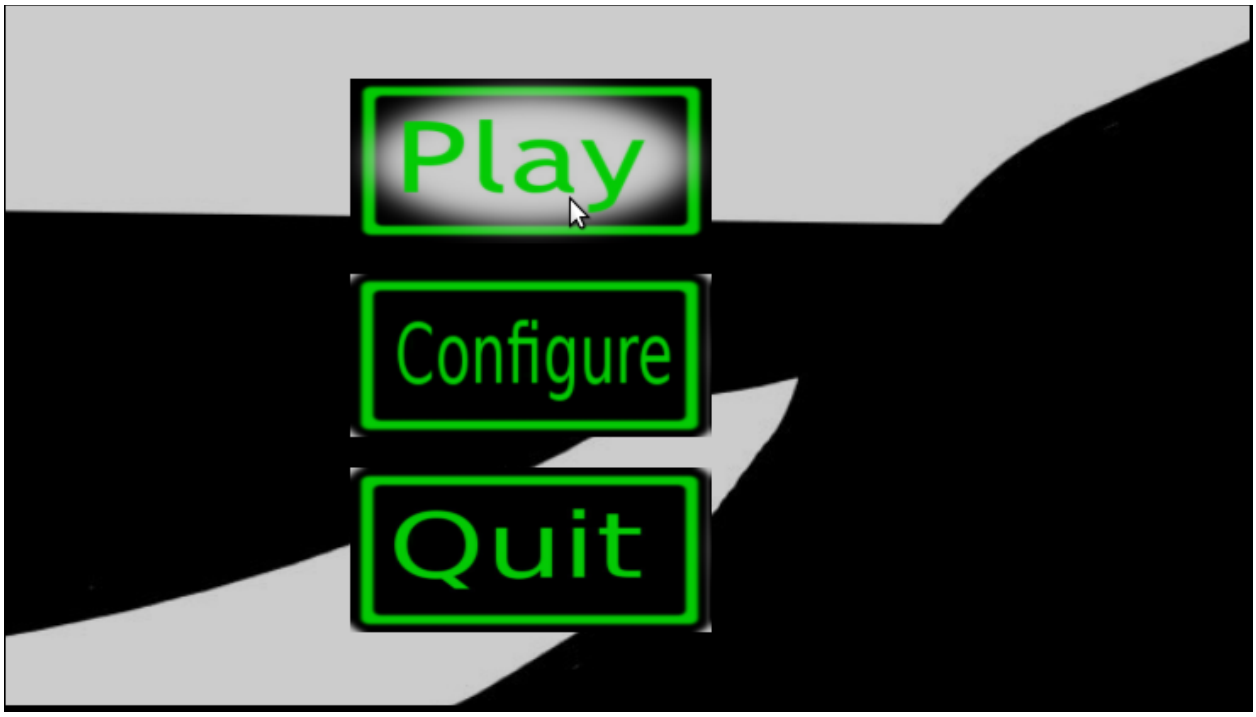
Mikäli käyttäjän tietokoneen on liityttävä ulkoiselle palvelimelle, tämä tekstitiedosto on avattava ja tiedostoon on kirjoitettava kohdepalvelimen IP-osoite. Huom! Älä lisää tiedoston loppuun rivinvaihtoa enter-näppäimellä.



Kuvassa esimerkkipalvelin opinnäytetyön kehittämisen ajalta.

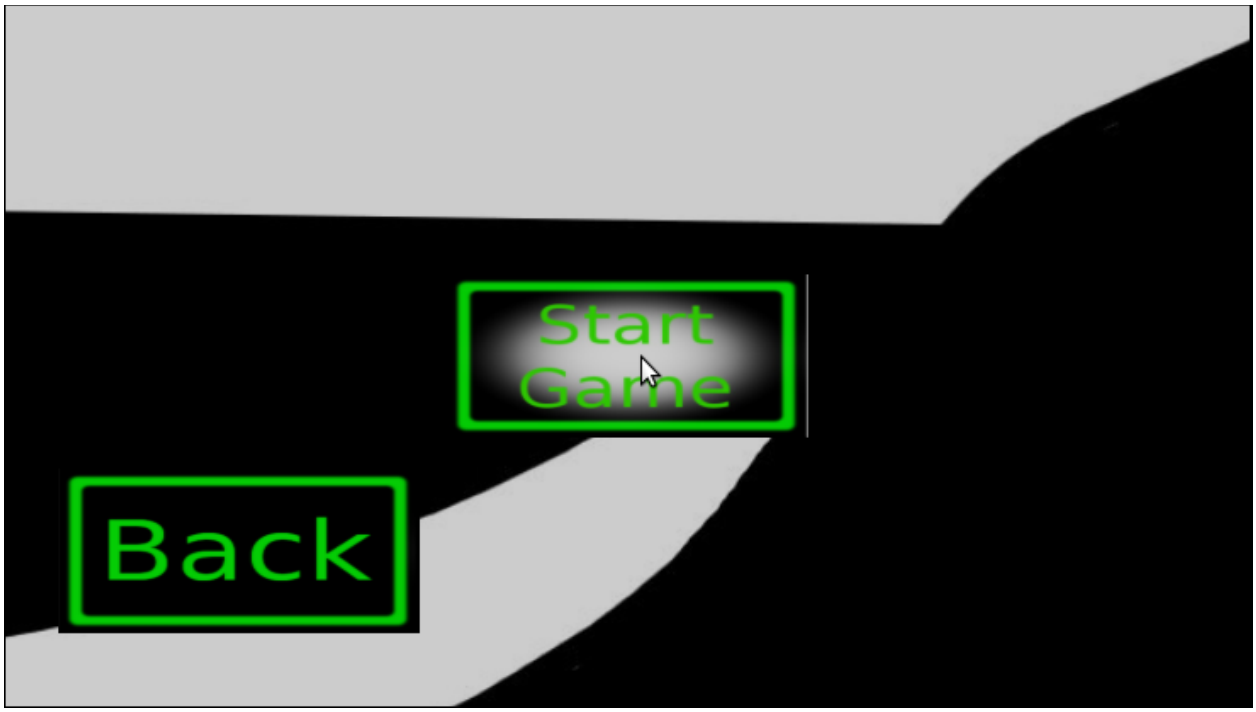
Valikot

Valikot avautuvat ohjelman käynnistyessä. Ohjelmainstanssin voi sulkea ensimmäisestä valikosta painikkeella Quit. Määrätäkseen ohjelmainstanssin palvelimeksi on valittava seuraavat painikkeet: Play, Host ja Start Game.



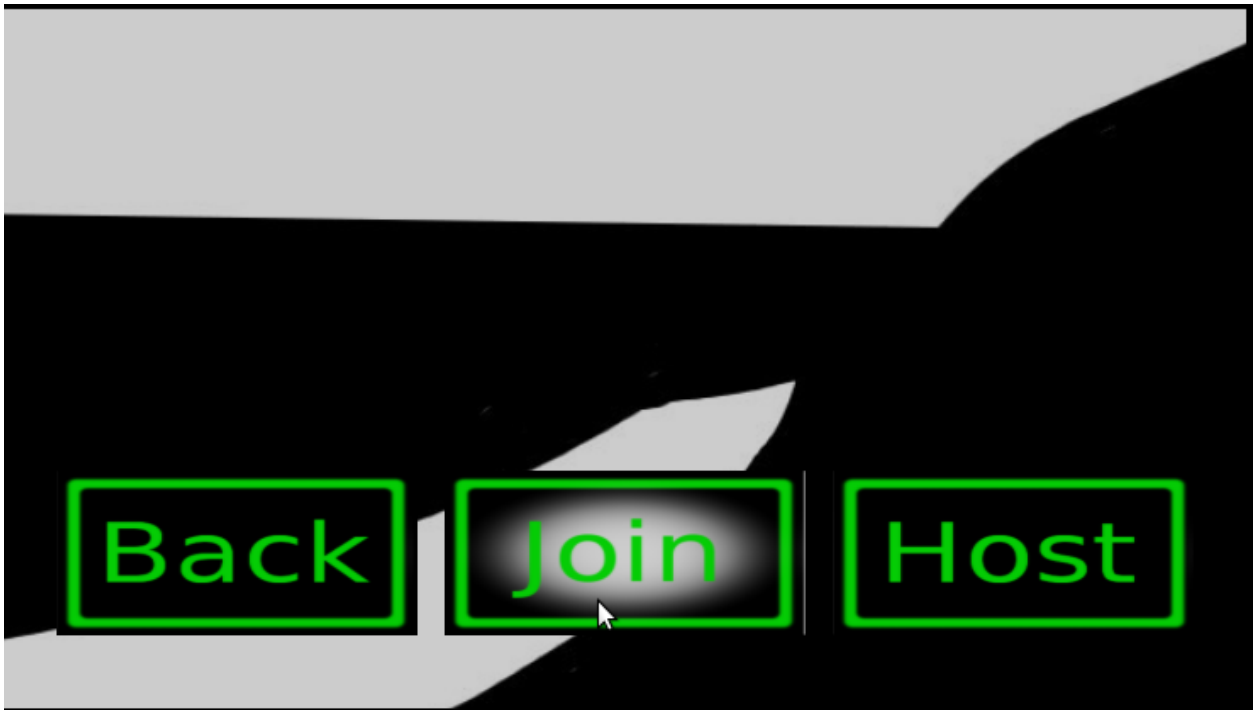
Valitse Play.

Valitse Host.

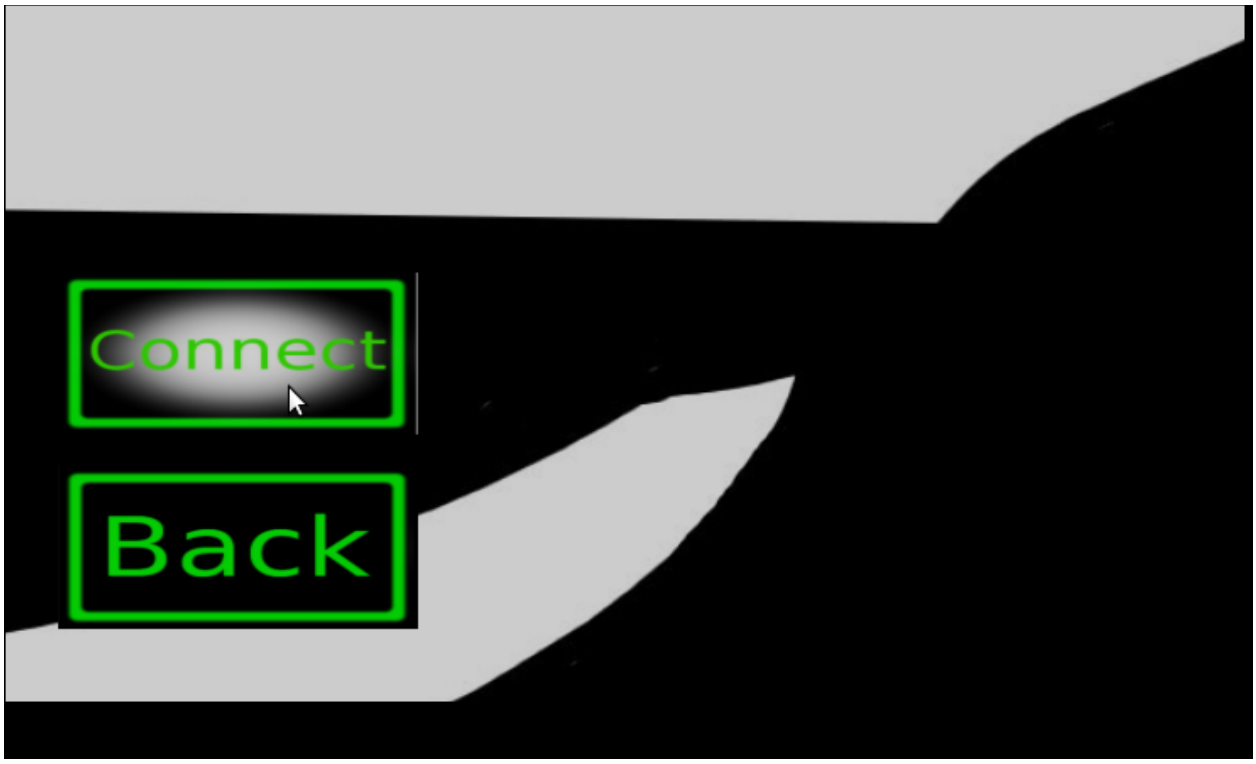


Valitse Start Game. Suunnittelunäkymä avautuu ja palvelin on valmis vastaanottamaan asiakasohjelmia.

Asiakasohjelmaksi instanssin voi valita Play-valikosta lähtien seuraavasti:



Valitse Join.

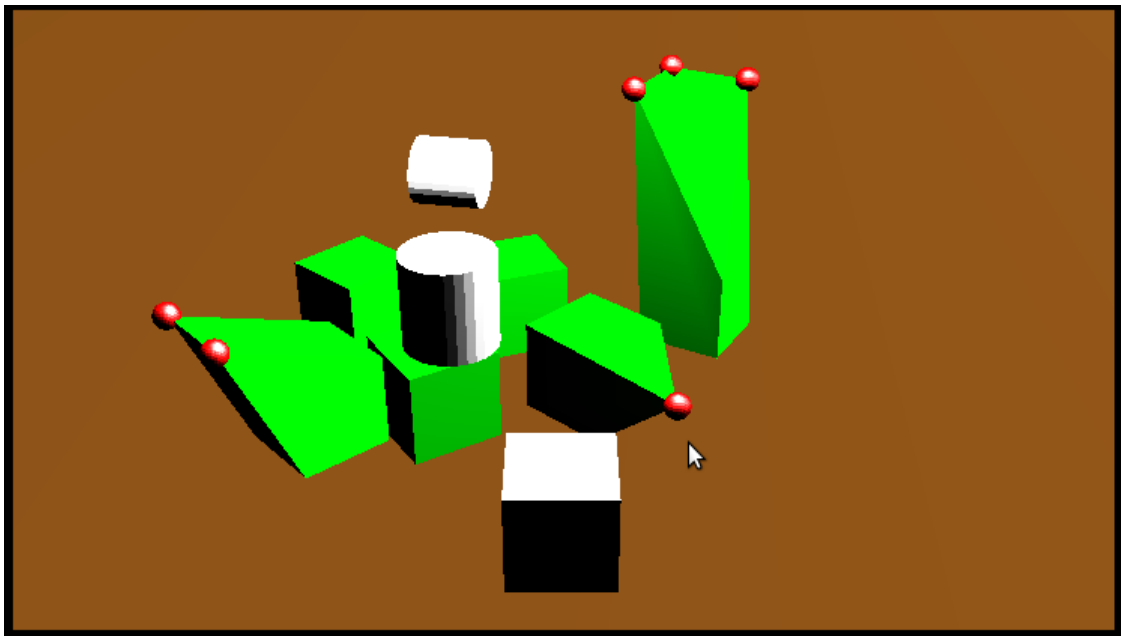


Valitse Connect. Ohjelma hakee verkon kautta tarvittavat tiedostot ja synkronoi näkymän palvelimen kanssa.

Aikaisempaan valikkoon voi siirtyä painikkeella Back.

Suunnittelunäkymä

Suunnittelunäkymässä kamera seuraa käyttäjän avatar-kappaletta, joka on valkoinen kuutio.



Suunnittelunäkymässä oli työn päättyessä seuraavat toiminnot, ja niiden oletuspainikkeet löytyvät käyttöohjeen taulukosta 1.

Toiminto	Ohjaussyöte
Avatarin liikuttaminen	
Liike eteenpäin	W
Liike taaksepäin	S
Käännös vasempaan	A
Käännös oikeaan	D
Liike ylöspäin	Välilyönti
Kappaleiden muokkaaminen	
Muokattavan vektorin valinta	Vasen hiiren painike
Muokattavan vektorin valinnan peruutus	Oikea hiiren painike
Valitun vektorin siirto kameran näkymän suhteen	CTRL + hiiren liike
Tallennus- ja lataustoiminnot	
Nykyisen näkymän tallennus DBM-muodossa	J
Näkymän lataaminen DBM-tiedostosta	F8
Kappaleen tallentaminen STL-muodossa	+
Kappaleiden tuottaminen	
Tuotettavan kappaleen vaihtaminen sylinteriin	T
Tuotettavan kappaleen vaihtaminen kuutioon	R
Tuotettavan kappaleen vaihtaminen Apila-kirjastoon	V
Tuotettavan kappaleen vaihtaminen Lakeuden Ristiin	B
Tuotettavan kappaleen vaihtaminen kaupungintaloon	N
Kappaleen tuottaminen	U

Kameratoiminnot	
Zoomaus	**Hiiren rulla eteen/taakse**
Kameran kääntäminen avatarin suhteen	**Hiiren rullapainike + hiiren liike**
Muut	
Simulaationäkymän sulkeminen	**ESC**

Taulukko 1. Oletusarvoiset ohjauspainikkeet ja -syötteet suunnittelunäkymässä