TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Automaatikainstituut

ISS40LT

Dolores Freiberg 120780IASB

3DKRAANA MODELLEERIMINE JA SIMULEERIMINE VIRTUAALMAAILMAS

Bakalaureusetöö

Juhendaja: Aleksei Tepljakov
Teadur

Tallinn 2016

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor:

Kuupäev: Allkiri:

Annotatsioon

Käesoleva töö eesmärgiks on kirjeldada 3DKraana virtuaalse mudeli loomise protsessi, kasutades kolme programmi: Blender, Unreal Engine 4 ja MATLAB.

Soov oli luua hariduslik töövahend, mis töötaks ka Oculus Rifti platvormil, ning millega on võimalik erinevaid juhtimisalgoritme katsetada 3DKraana virtuaalse mudeli baasil.

Töö koosneb piltidest ja tekstist, mis kirjeldavad 3DKraana mudeli loomist ning samuti on lisatud mõningad soovitused lugejale, kes soovib iseseisvalt projekti luua.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 44 leheküljel, 4 peatükki, 31 joonist ja 1 tabelit.

Abstract

Modeling and Simulating a 3DCrane in Virtual Reality

The following thesis intends to describe the process necessary to create an interactive "3DCrane" model, that will serve as an educational tool, using mainly the programs Blender, Unreal Engine 4 and MATLAB.

The main reasons for choosing this particular project are 1) wanting to create an educational and helpful model for future students studying with the 3DCrane to help them get better understandings of its functions, 2) wanting to create something more "hands-on" and "practical" rather than theoretical or analytical, 3) wanting to learn how to work with artistic programs such as Blender and Unreal Engine 4.

The work consists of images and text related to 3D creation, manipulation, etc., in Blender, Unreal Engine 4 and MATLAB, as well as suggestions for improvements for those who would like to persue a similar project, or build upon this one.

The thesis is written in Estonian and contains 44 pages of text, 4 chapters, 31 figures and 1 table.

Sisukord

Jooniste loetelu

Tabelite loetelu

Sissejuhatus

Igapäevaselt on oluline liigutada massiivseid objekte ühest kohast teise ning inimjõud iseseisvalt sellega hakkama ei saaks. Seetõttu on loodud selleks masinad ning tänapäeval on paljud neist juba automatiseeritud. Seejuures on oluline, et säiliks turvalisus ja selleks on loodud mitmed keerukad programmid ja neid juhtivad algoritmid.

Virtuaalreaalsus on üks viisidest tegeliku reaalsuse kujutamiseks, mille abil on võimalik vähendada kulutusi, aega ning säilitada kasutajaskonnale turvalisus. Virtuaalreaalsust defineeritakse kui terminit, mis kirjeldab kolmedimensionaalset arvuti genereeritud keskkonda, mida saab kontrollida inimene [1]. Virtuaalreaalsust kasutatakse paljudes valdkondades ning see aitab oluliselt kaasa inimese arengule.

Antud töö eesmärk oli luua Tallinna Tehnikaülikooli laboratooriumis asetsevast 3DKraana mudelist omakorda mudel ja lisada sellele reaalsusele vastav simulatsioon virtuaalmaailmas. 3DKraana on tööstusliku kraana väiksem laboratoorne mudel, mis sobib erinevate juhtimisalgoritmide katsetamiseks laboratoorsetes tingimustes.

Selleks kasutasin 3D-tarkvara Blenderit, kuna erinevalt paljudest teistest samaväärsetest programmidest (Maya, 3ds Max, XSI) on see kättesaadav vabavarana ja tasuta. Eesmärk oli ka Oculus Rifti rakendamine ja seetõttu otsustasime simuleerimise teha Unreal Engine 4 mängu mootorit kasutades, kuna see on samuti tasuta kättesaadav. Viimaseks on loodud MATLABis kraana matemaatiline mudel, mille abil on teostatud virtuaalse mudeli dünaamika Unreal Engine 4-s.

Töö koosneb sissejuhatusest, neljast peatükist ja kokkuvõttest.

Esimeses peatükis antakse ülevaade 3DKraana mudelist ning tuuakse välja selle kasutusala reaalsuses.

Teises peatükis kirjeldatakse 3DKraana mudeli modelleerimisprotsessi keskkonnas Blender ja tuuakse välja terminoloogia, mis on vajalik selle mõistmiseks.

Kolmandas peatükis tutvustatakse Unreal Engine 4 ja 3DKraana simuleerimisprotsessi. Samuti esitatakse terminoloogia, mis on oluline põhifunktsioonide rakendamiseks. Lisaks on lühike kirjeldus Oculus Riftist.

Viimases peatükis on ülevaade MATLABist ja loodud juhtimisalgoritmist.

1 3DKraana ja selle ülevaade

Selles peatükis antakse ülevaade 3DKraanast ja selle rakendamisest reaalses maailmas. Samuti tuuakse välja peamised spetsifikatsioonid, mis on olulised hilisemate protsesside jaoks nagu näiteks modelleerimine ja simuleerimine.

1.1 Mis on 3DKraana?

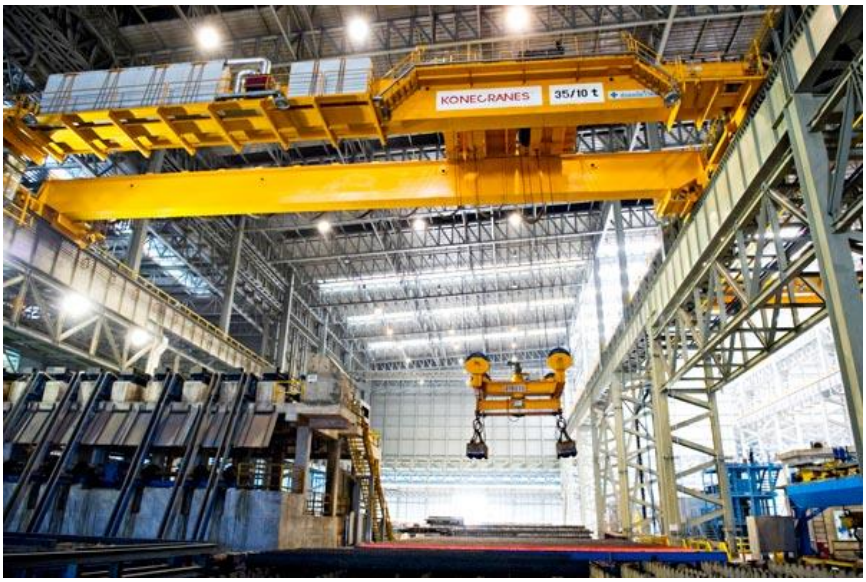
Käesolevas töös olen kasutanud modelleerimiseks ja simuleerimiseks füüsilist mudelit, mis asub Tallinna Tehnikaülikooli laboratooriumis. 3DKraana mudel on mõeldud eelkõige teadusasutustele uurimus ja haridust edendavateks töödeks.

See mudel on loodud reaalsete tööstuskraanade põhjal ja seda on võimalik juhtida kasutades PC-d. Lühidalt defineerides on 3DKraana mudel mittelineaarne elektromehaaniline süsteem, millega on võimalik luua keerulisi dünaamilisi käitumismalle [2].

1.2 Milleks reaalselt tööstuskraanasid kasutatakse?

Tööstuskraanasid on olemas mitut erinevat tüüpi, kuid antud töös kasutatav mudel imiteerib eelkõige niinimetatud sildkraanat (joonis 1).

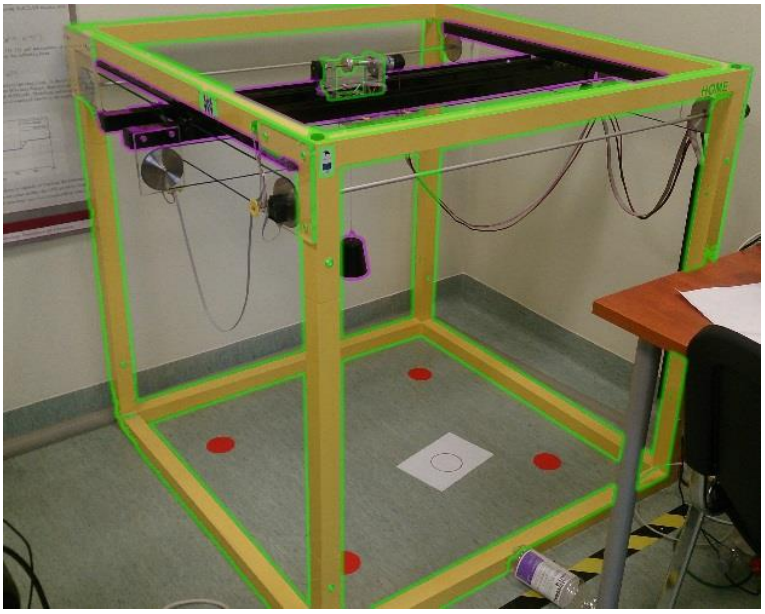
Sildkraanad on enamasti mõeldud raskete koormuste liigutamiseks ulatuslikul tööalal. Nendega on võimalik kiirendada materjalivoogu ning monteerimisprotseduure, samuti lühendada tootmisaega [3]. See koosneb rööbasteel liikuvast sõidumehhanismist – sillast, ning sillal liikuvast tõste- ja sõidumehhanismist – vankrist. Rööbastee on kinnitatud ehituskonstruktsiooni külge. Harilikult omab tõstevõimet kuni 50 tonni, harvemal juhul kuni 600 tonni [4].



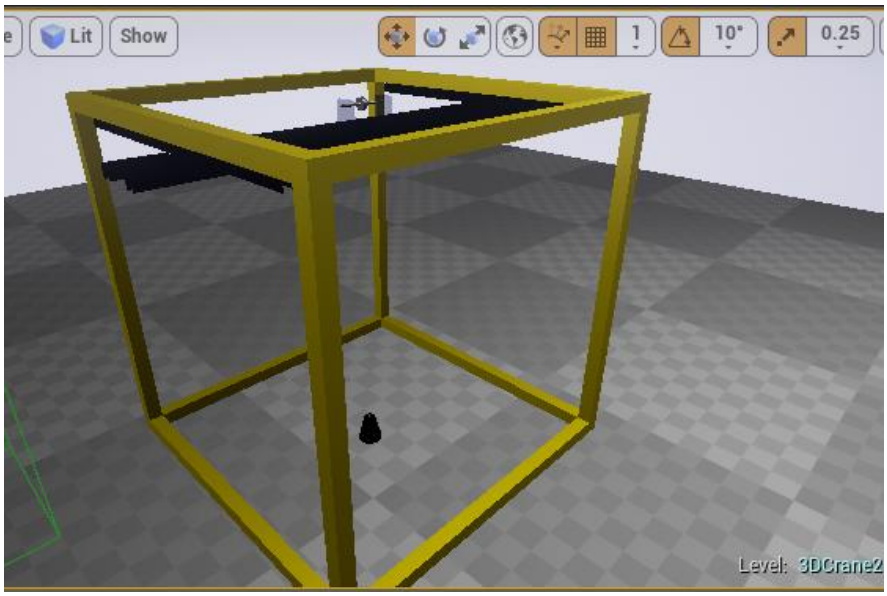
Joonis 1. Tööstuskraana rakendamine igapäevaelus.

1.3 3DKraana spetsifikatsioonid

3DKraana modellatsioonil, mis on tehtud Blenderi keskkonnas, ei ole kujutatud kõiki objekte, mis füüsilisel mudelil eksisteerib. Antud modellatsioon on koostatud põhilistest kraana komponentidest, mis on vajalikud põhifunktsioonide toimimiseks. Joonisel 2 on rohelise ja violetse värviga esile tõstetud need komponendid, millest koosneb ka käesoleva töö mudel. Joonisel 3 on kujutatud töö käigus loodud mudeli lõplik versioon.



Joonis 2. Füüsiline 3DKraana mudel.



Joonis 3. Modellatsioon Unreal Engine 4 keskkonnas.

Virtuaalmaailmas loodud modellatsioon on peaaegu üksüheses vastavuses füüsilise mudeli mõõtmetega. Mõõtmised on saadud käsitsi, kasutades mõõdulinti ja nihikut. Tabelis 1 on välja toodud mõningad 3DKraana parameetrid, mis on saadud reaalsete mõõtmiste käigus [5].

Tabel 1. 3DKraana parameetrid [5].

Parameeter	**Väärtus**
Maksimaalne kiirus v_{max} $[m/s]$	0.3
Maksimaalne kiirendus a_{max} $[m/s2]$	0.6
Liikumisala mööda X-telge $[mm]$	920
Liikumisala mööda Y-telge $[mm]$	920
Trossi pikkus l $[mm]$	820
Koormuse mass m_l $[kg]$	1
Vankri mass m_c $[kg]$	1.155
Silla mass m_r $[kg]$	2.2

1.4 3DKraana ja MATLAB

3DKraana koosneb kolmest liikuvast osast. Koormuse asukohta muudetakse mööda Z-telge. Nii sild kui vanker liiguvad horisontaalselt mööda X-telge ja lisaks on vankril

omadus liikuda sillal mööda Y-telge. Koormus, mis on ühenduses vankriga, võib liikuda vabalt igas suunas. 3DKraana juhtimine toimub DC mootorite abil [2].

3DKraanal on viis identset kooderit, mis mõõdavad viit erinevat muutujat: tõstevankri koordinaate horisontaalsel tasandil, koormuse ja tõstevankri vahelise trossi pikkust ning ka koormuse kahte kaldenurka [2].

Toiteliides võimendab juhtsignaali, mis edastatakse PC-lt DC-mootoritele. Samuti see konverteerib kooderi pulss-signaalid 16-bitisteks jadadeks, et informatsioon oleks PC-le aru saadav [2].

PC, mis on varustatud RT-DAC/PCI digitaalse I/O plaadiga suhtleb toiteliidese plaadiga. Loogika, mis on vaja kooderi signaalide aktiveerimiseks, lugemiseks ja PWM pulsi genereerimiseks seadistatakse Xilinx kiibil. Kõikidele plaadi funktsioonidele on võimalik ligi pääseda läbi 3DCrane Toolbox´i, mis toimib MATLAB/Simulink keskkonnas [2].

2 Blender ja selle ülevaade

Antud töös kasutasin Blenderit kui modelleerimisvahendit, et luua realistlik 3D mudel kraanast, mis paikneb Tallinna Tehnikaülikooli laboratooriumis.

2.1 Mis on Blender?

Blender on keskkond 2D/3D objektide loomiseks, mis toetab kõikvõimalikke operatsioone, nagu näiteks modelleerimist, UV-laotamist, tekstuurimist, skelettanimatsiooni, osakeste ja muu füüsika simulatsiooni, skriptimist, renderdamist, komponeerimist, järeltöötlust ja mängude loomist [6].

Kuna Blender on oma olemuselt vaba tarkvara ja selle litsents on registreeritud GNU GPL (General Public License) all, siis vabatahtlikud, professionaalid ja kõik nende vahepeal on oodatud lisama oma ideid, et kaasa aidata kogu projekti arengule.

Kui programm on GNU Üldise Litsentsi (GPL-i) all [7]:

- on sul õigus programmi igaks otstarbeks kohandada;
- on sul õigus programmi muuta ja õigus lähtekoodiga tutvuda;
- on sul õigus programmi kopeerida ja edasi anda;
- on sul õigus programmi täiendada ja oma versioone välja anda.

Näiteks on kasulik, kui kogenud kasutajad muudavad programmi endale kasutajasõbralikumaks seda ümber programmeerides ning seejärel saab Blender muuta oma uuendustega seda mugavamaks kogu kasutajakonnale. Kogu kood on kirjutatud, kasutades C, C++ ja Pythonit.

Blenderi avatud arhitektuur võimaldab töötada erinevatel operatsioonisüsteemidel, pakub laiendusvõimalusi, nõuab vähe mälu ning pakub ühtset töövoogu. Blender on siiani üks populaarsemaid vabavaralisi 3D-graafika loomise programme [6].

2.2 Blenderi ajalugu

Blenderi arendamine sai alguse Hollandis 1995. aastal Ton Roosendali tütarfirma NaN (Not a Number) eestvedamisel. Eesmärk oli luua ja tasuta jagada väikest igal platvormil töötavat 3D-rakendust. Seega algeliselt oli Blender loodud programmina, mida oli võimalik tasuta alla laadida, kuid see ei olnud saadaval vaba tarkvarana. Alles aastal 2002, peale mitmeid keerulisi aastaid NaN-is, otsustas Ton Roosendal teha otsustava sammu, et tagada Blenderi jätkusuutlikus. Selle tagajärjel loodi uus mittetulundusühing, Blender Foundation, mille eesmärk oli arendada Blenderit, kui kogukonnapõhise vaba tarkvara projektina [8]. Blender Foundation tegutseb tänaseni Ton Roosendali eestvedamisel ning tänu vabale tarkvarale on see olnud jätkusuutlik, eelkõige tänu kasutajakonnale, kes panustavad oma teadmisi pidevalt Blenderi arengusse.

2.3 Blenderi terminoloogia, mis on oluline kraana mudeli loomiseks

Objektirežiim ja Muutmisrežiim (Object mode and Edit mode)

Geomeetriliste objektidega on võimalik tegutseda, kasutades kahte erinevat režiimi:
1) Objektirežiimis tehtavad muudatused muudavad tervet objekti. Objekti lisamisel ollakse vaikimisi objektirežiimis, kus on võimalik limiteeritud koguses teha muudatusi. Nagu näiteks suurus, asukoht, orientatsioon.
2) Muutmisrežiimis tehtud muudatused ei mõjuta selliseid omadusi nagu näiteks objekti asukoht ja kaldenurk, kuid see-eest on võimalik muuta objekti geomeetriat.

Objekti asukoha muutmisel tuleks tähelepanelikult vaadata, mis režiimis paiknetakse, kuna hiljem võib see modelleerimisel probleeme tekitada. Nende kahe režiimi vahel on võimalik liikuda, kasutades kiirklahvi *Tab*.

3D-kursor (3D Cursor)

3D-kursoriga on võimalik määrata, kuhu stseenile lisatud uued objektid tekivad ning sellega on võimalik määrata kese, mille ümber objekti pööratakse [9].

Mediaanpunkt (Median Point)

Mediaanpunkt on peaaegu sama, mida füüsikas mõistetakse raskuskeskme all. Kui igal valitud elemendil oleks samasugune mass, siis asetseks mediaanpunkt raskuskeskmes ehk kohas, kus kõik valitu oleks tasakaalus.

Keskpunkt (Pivot Point)

Keskpunkt on ruumipunkt, mille ümber objekti/objektide pööramised, mõõtkava muutmised ja peegeldamised aset leiavad.

Objektide ühendamine ja eraldamine (Join objects and Seperate objects)

Kui on soov ühendada teatud objektid üheks ainsaks objektiks, siis on võimalik kasutada kiirklahvi kombinatsiooni *Ctrl+J*. Uus "keskpunkt" on määratud viimasena valikusse võetud objekti keskpunktiga.
Vastupidist operatsiooni ehk objektide eraldamist on võimalik sooritada, kui kasutada muutumisrežiimis kiirklahvi *P*, olles valinud soovitud tipud või küljed eraldamiseks. Seejärel tuleks valida soovitud variant.

Grupeerimine (Grouping)

Grupeerimine on mugav operatsioon, mis ei tekita olulisi muutusi seoses objektide funktsioneerimisega. Seda saab kasutada selleks, et objektid oleksid paremini organiseeritud ja näiteks, kui on soov neid objekte liigutada nii, et need grupeeritult koos liiguks. Gruppi kuuluvate objektide serv on valituna roheliselt tähistatud.

Töötlejad (Modifiers)

Töötlejad on automaatsed operatsioonid, mis mõjutavad objekti väljanägemist selle lähtekuju muutmata. Nad muudavad objekti väljanägemist, renderdamist, kuid mitte selle geomeetriat. Töötlejate abil on võimalik teha mitmeid teisendusi ja lisada efekte, mis käsitsi tehes võiks palju aega võtta [9]. Töötlejaid saab lisada ainult objektirežiimis.

Antud töös on kasutatud kahte tüüpi töötlejat:

1) *Boolean modifier*:

Selle abil on võimalik luua kahest võrest uus kombineeritud võre, kasutades teatud kahendoperatsiooni. Valikus on kolm erinevat võimalust: *Difference* (sihtvõre lahutatakse muudetavast võrest), *Union* (muudetav ja –sihtvõre liidetakse kokku), *Intersect* (tulemuseks on sihtvõre ja muudetava võre kattuv osa).

2) *Subdivision surface modifier*:

Selle abil on võimalik muuta objekti pinna väljanägemine siledamaks ning see võimaldab modelleerida keerukaid kõrge tihedusega objekte, ilma et oleks vaja salvestada väga suurt hulka andmeid. Isegi väheste tippudega võre võib pärast töötlemist välja näha ümaram ning nn "orgaanilisem".

Kihid (Layers)

Kui 3D-stseenid muutuvad keerukamaks, siis on kasulik kasutada objektikihte. Kihid on abiks mitmel moel. Näiteks kui modelleerimise käigus oleks vaja valgustada ainult teatuid objekte või näiteks määrata, milliseid objekte renderdatakse. Samuti on kihte kasulik kasutada, kui on loodud mitmeid objekte ning oleks vaja töötada detailsemalt teatud tüüpi objektide/andmetega. Kihtide vahel on võimalik liikuda, kasutades kiirklahvi *M* ja *Numbad´*il asuvaid numbreid.

Nakkumine (Snap)

Nakkumine võimaldab valitud komponente liigutada nii, et need teiste objektide külge kleepuks. See on sarnane kahe magnetilise keha kokkutõmbumisele. Selleks tuleks objektirežiimis 3D-vaate päisenuppude hulgas leida magneti ikoon ning see aktiveerida.

Mõõtkava (Scale)

Kiirklahvi S vajutamisel sisenetakse mõõtkava *Scale* teisendusrežiimi, kus on võimalik muuta objektide/andmete mõõtmeid vastavalt hiire kursori asukohale. Objekti mõõtkava suureneb, kui liigutada hiire kursorit keskpunktist eemale, ning väheneb, kui liigutada kursorit keskpunktile lähemale [9].

Pööramine (Rotation)

Kiirklahvi *R* vajutamisel sisenetakse pööramise *Rotation* teisendusrežiimi, kus on võimalik objekti/andmeid vastavalt hiire kursori asukohale pöörata ümber mitme telje.

Duplitseerimine (Duplication)

Kui on soov teha valitud objektist/objektidest samasuguse väljanägemisega koopia, siis saab kasutada kiirklahvi kombinatsiooni *Shift+D*. Duplikatsioon tekib originaaliga samasse kohta, ning seda saab soovitud asukohta lohistada. Tekitatud koopia on kui täiesti uus objekt, mis jagab originaaliga samu omadusi (materjalid ja tekstuurid) [9].

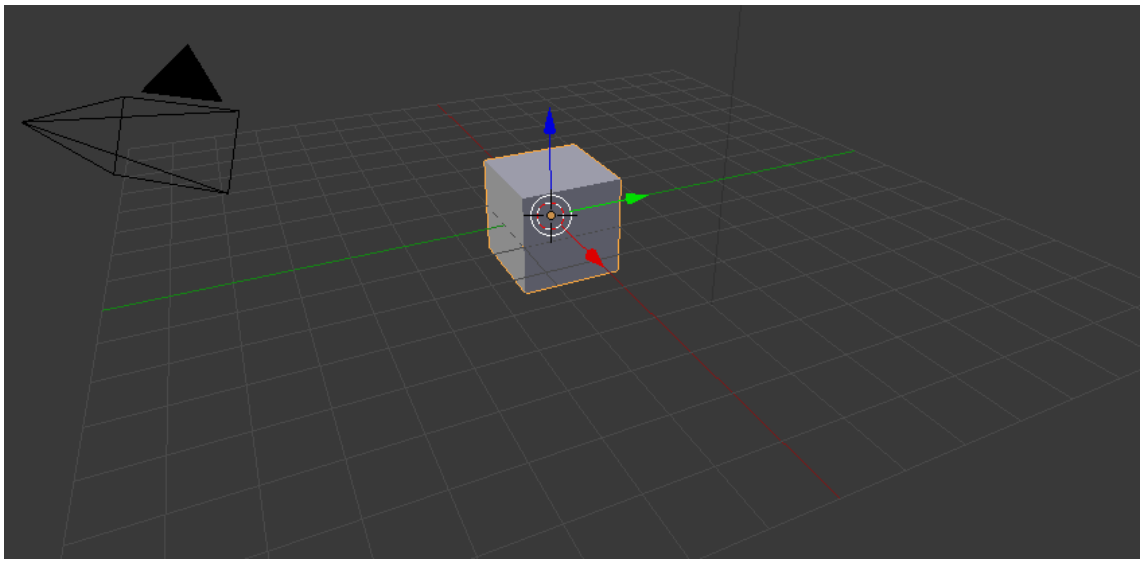
2.4 3DKraana mudeli loomine Blenderi keskkonnas

3D-stseeni loomiseks on minimaalselt vaja kolme komponenti: mudelit, materjali ning valgusallikat [9].

Mudeleid on võimalik luua erinevaid funktsioone kasutades – mõned neist on efektiivsemad ja mõned vähem efektiivsemad. Enamasti see oleneb kasutaja kogemusest, kuid kõige olulisem on siiski antud olukorras lõpptulemus.

2.4.1 Vajalikud sätted enne modelleerimist

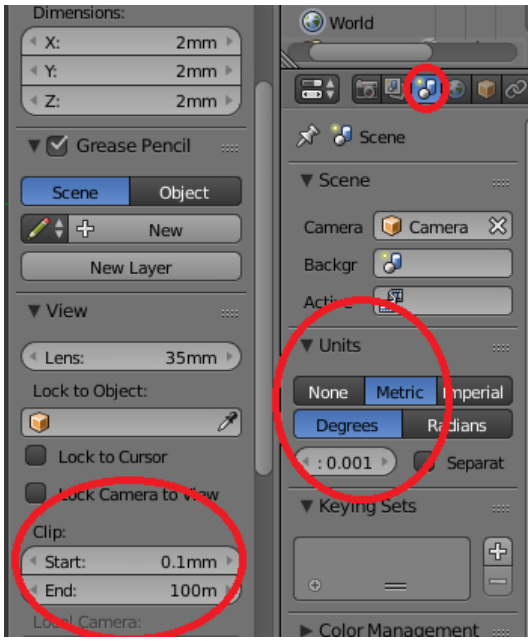
Blenderit on võimalik tasuta alla laadida kompileeritud programmina või lähtekoodina nende ametlikult kodulehelt www.blender.org, kus on samuti olemas manuaal erinevate mõistete ja võimaluste jaoks. Kui avada programm, siis vaikeekraan kujutab endast kuubikujulist võreobjekti, kaamerat, ja valgusallikat (joonis 4).



Joonis 4. Blenderi vaikeekraan.

Et alustada kraana modelleerimist, oleks kõige pealt oluline muuta mõõtühikute süsteem meetriliseks. Blenderis on võimalik valida kolme erineva mõõtkava vahel: *"None"*, *"Metric"* ja *"Imperial"*.

Vaikeseadena on valitud *None*, mida teisiti kutsutakse ka "Blenderi ühikuks", kuid antud projekti puhul on kasulikum teisendada süsteem meetriliseks, sest mõõtmised on tehtud millimeetrites. Peale teisendust on näha, et algselt olev *grid* ei ole selle mõõtühikute süsteemiga sobilik ja seetõttu tuleks see muuta meie projektile kohaseks [10]. Joonisel 5 on välja toodud muudatused, mis tuleks enne modelleerimist teha.



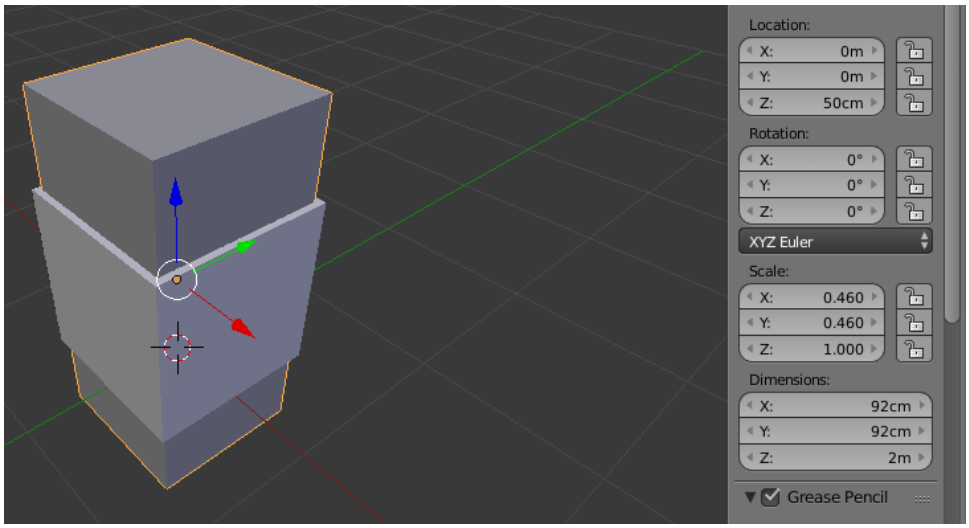
Joonis 5. Mõõtkava meetriliseks muutmine.

Järgmise sammuna saaks alustada modelleerimist ja selleks võiksime kasutada algset võrekujulist kuupi, muutes selle kuubi mõõtmed meile vajalikuks. Objekti dimensioone on võimalik muuta *View Properties* paneelilt.

On mitmeid erinevaid mooduseid, et jõuda sarnase lõpptulemuseni, kuid antud töös toon välja mooduse, mida kasutasin.

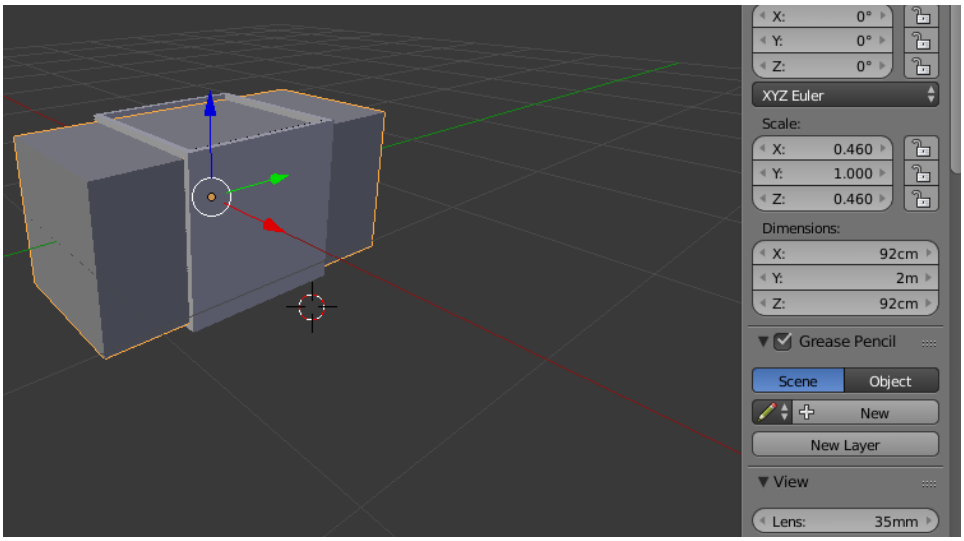
2.4.2 3DKraana modelleerimine

Üks lihtne moodus kraana raamistiku loomiseks on kasutada *Boolean modifier´*it. Selleks oleks vaja luua objektirežiimis teine võrekujuline kuup kasutades kombinatsiooni *Shift+A*. On oluline muuta ka teise kuubi mõõtmed selliselt, et peale *Boolean modifier´*i rakendamist saaksime soovitud paksusega raami (joonis 6).



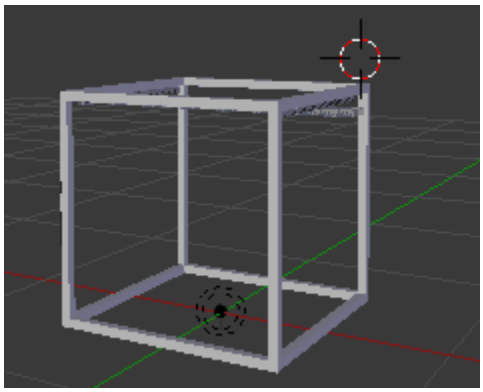
Joonis 6. *Boolean modifier´*i *"Difference"* operatsiooni rakendamine.

Seejärel on vajalik rakendada *Boolean modifier´*i *"Difference"* funktsiooni ka teist korda (joonis 7). Töötlejaid on võimalik rakendada objektirežiimis.



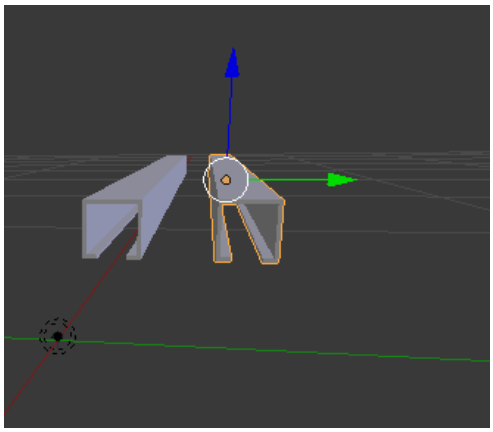
Joonis 7. *Boolean modifier´*i rakendamine teistkordselt.

Kui põhiraam (joonis 8) on valmis, siis tuleb alustada järgmiste detailide loomist, mille puhul järjekord pole oluline. Kõige lihtsam moodus nn "aukude" tegemiseks on kasutades *Boolean Modifier´*it.



Joonis 8. 3DKraana põhiraam.

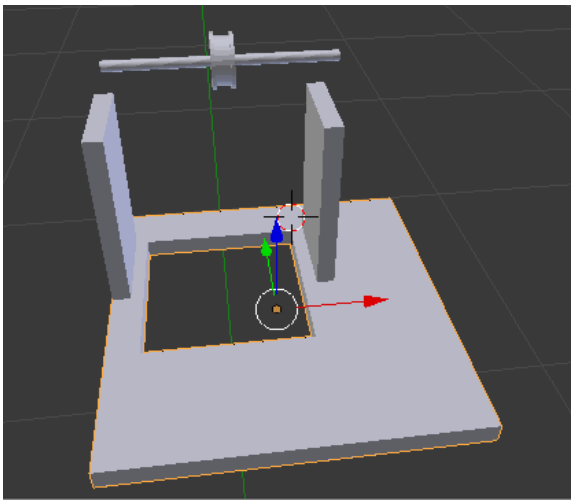
Näiteks kui järgmise objektina luua üks taladest, siis võiks meeles pidada asjaolu, et saada teine samasugune objekt, on võimalik seda objekti duplitseerida. See muudab modelleerimisprotsessi palju kiiremaks (joonis 9).



Joonis 9. Duplitseeritud objekt.

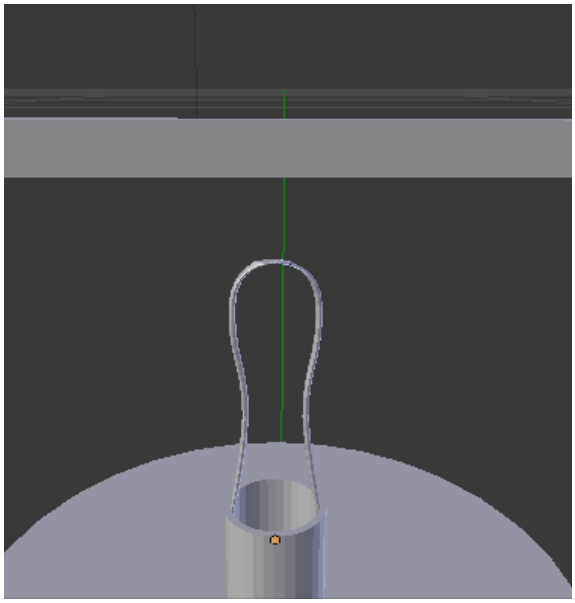
Seejärel on oluline paigutada need objektid oma täpsete mõõtudega soovitud asukohta. Blenderis on hea lihtne moodus selleks kasutada koordinaate ning õigete koordinaatide leidmiseks tuleks kasutada tavalist matemaatikat. Hea moodus objekte üksteise vastu kleepida on kui kasutada *Snap* funktsiooni ning kui on soov neid koos liigutada, siis võiks objekte grupeerida.

Kui objekte on mitmeid ja see hakkab töötamist segama, siis hea on kasutada *layer´*eid. Näiteks hulk detaile (joonis 10) on loodud algselt teisel kihil ning hiljem on neid võimalik liigutada algsele kihile.



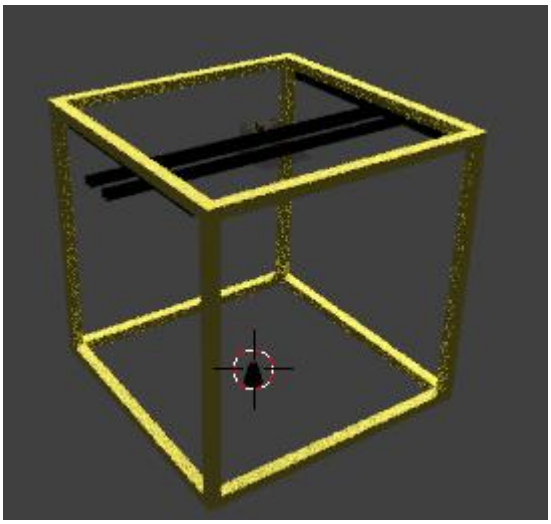
Joonis 10. Detailide loomine erinevatel kihtidel.

Peaaegu kõikide objektide loomisel on kasutatud eelnevaid põhitõdesid. Kuid kui on soov luua objekte, millel reaalsuses on ümaram ja siledam vorming, siis saab kasutada *Subdivision surface modifier´*it (joonis 11).



Joonis 11. *Subdivision surface modifier´*i kasutamine.

Kui soovitud detailid on loodud, siis viimase sammuna võiks neile lisada materjalid. See on eelkõige kasulik visualiseerimiseks (joonis 12), kuid pole antud olukorras prioriteet, kuna plaan on eksportida mudel Unreal Engine 4 ja lisada materjalid hiljem. Seetõttu pole oluline, et materjalid Blenderis reaalsusele vastaks.

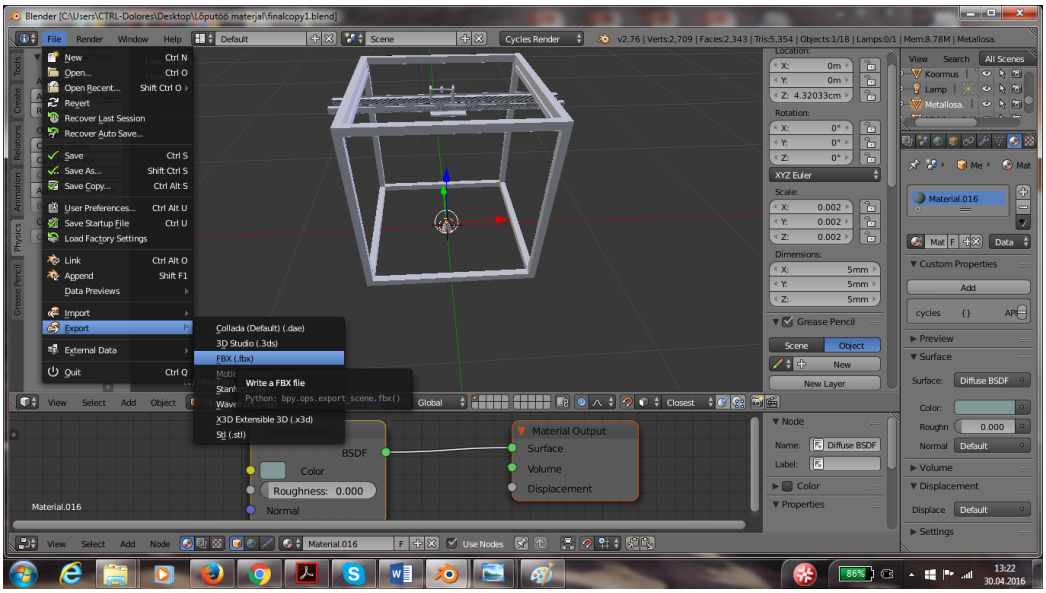


Joonis 12. Kraana mudel lisatud materjalidega renderdatud kujul Blenderis.

Peale materjalide lisamist on modellatsioon valmis ning seda võib eksportida Blenderist Unreal Engine 4. Kuid enne eksportimist oleks oluline kõik detailid objektirežiimis seada (0;0;0) koordinaatidele, kuna UE4 importimisel salvestatakse automaatselt kõikide objektide *pivot point* telgede tsentrisse.

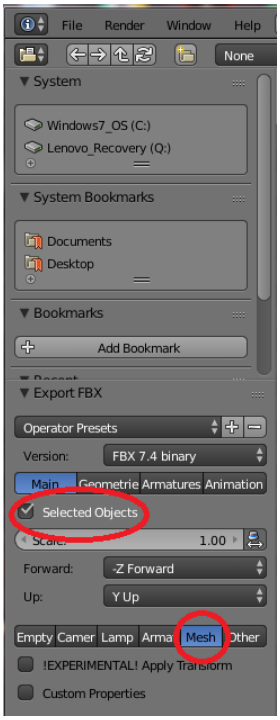
2.4.3 Modellatsiooni eksportimine

Edasine eesmärk on mudeli simuleerimine, kasutades Unreal Engine 4. Selleks on vaja valmis modellatsioon importida UE4, mis toetab .fbx failivormingut ning Blender võimaldab selle loomist. Selleks oleks vaja *File* menüüst valida *Export* ning seejärel *FBX* (joonis 13).



Joonis 13. Mudeli eksportimine .fbx failivormingus.

On oluline, et objektid, mida soovitakse eksportida, oleksid kõik aktiivsed. Kuna antud juhul pole oluline valgusallika, kaamera ja animatsiooni eksportimine, siis tuleks valida lihtsalt *Mesh*, mis garanteerib, et eksporditakse 3D kraana mudel ja sellega seonduv (materjalid, tekstuurid) (joonis 14). Edasine protsess jätkub Unreal Engine 4-s.



Joonis 14. Eksportimine.

3 Unreal Engine 4 ja selle ülevaade

Selles peatükis antakse ülevaade mängu mootor Unreal Engine 4-st ning kirjeldatakse peamisi funktsioone, mis on vajalik 3DKraana virtuaalse mudeli simuleerimiseks. Samuti on lühike tutvustus Oculus Rift virtuaalprillide kohta.

3.1 Mis on Unreal Engine 4?

Unreal Engine 4, mida lühidalt kutsutakse UE4, on Epic Games ettevõtte poolt loodud mängu mootor, mille eesmärk on võimaldada mugavamalt erinevaid mänge luua. Enamasti mängu mootorid koosnevad mitmetest programmidest erinevate funktsioonidega ning üks populaarsematest mängu mootoritest on praegusel ajal UE4. See on alates 2015. aasta märtsist tasuta kättesaadav, kuid kui selles keskkonnas loodav mäng hakkab kasumit teenima, siis 5% tulust läheb Epic Games stuudiole. UE4 populaarsus tuleb eelkõige selle kasutajasõbralikkusest ning võimalusest luua väga heal tasemel mänge [11].

3.2 Unreal Engine ajalugu

Unreal Engine arendus sai alguse Epic Games inseneride poolt, mille algne eesmärk oli luua arvutimäng Unreal, mis kujutas endas tulistamismängu. Unreal Engine esimene versioon (Unreal Engine 1) avaldati aastal 1998 ning iga uue versiooniga kaasnesid uued funktsioonid, mis võimaldasid luua veelgi paremaid kõrgetasemelisi mänge. Kõige esimene versioon toetas mõningaid platvorme: PC, Mac, Linux, PlayStation 2, kuid nüüdseks võimaldavad uued versioonid kasutada ka teisi platvorme nagu näiteks Android, iOS, Wii U, Xbox 360/One, Oculus Rift, PlayStation 3 ja 4. Aastal 2002 avaldati teine versioon (Unreal Engine 2), mille aluskood oli täielikult ümber kirjutatud. Iga uue väljalaskega kaasnesid paremad funktsioonid ja efektid, mis tekitaksid realistlikuma tunde mängijale. Kolmas versioon (Unreal Engine 3) avaldati aastal 2006. Sel oli varasematest versioonidest parem võimalus muuta valgustust ning varjutust. UE3 kasutati ka muuks otstarbeks, kui lihtsalt mängude loomiseks. Selle abil on loodud olulisi simulaatoreid õpinguteks ning samuti animafilme. Kõige uuem versioon, mis on siiani avaldatud on Unreal Engine 4 ning peamiselt erineb see eelnevast versioonist, kuna kasutab Kismet asemel Blueprinte [11].

3.3 UE4 terminoloogia, mis on vajalik simulatsiooni loomiseks

Project

Projekt koosneb kõiksugu andmetest, mis omakorda moodustavad mängu. UE4-s on korraga võimalik tegutseda mitme erineva projektiga.

Actor

Iga objekt (valgusallikas, karakter, võre, jne), mis lisatakse maailma, kvalifitseerub *Actor´*i mõiste alla. *Actor* on programmeeritav klass, mida kasutatakse objekti asukoha, pöörlemise ning mõõtkava defineerimiseks.

Component

Komponent on funktsionaalsus, mida saab lisada *Actor´*ile. *Actor´*il on võimalik kasutada kõiki erinevaid funktsionaalsusi, mida võimaldavad komponendid. Näiteks *Audio Component* võimaldab *Actor´*il tekitada heli. Kui luua UE4 auto, mis koosneb mitmest osast, siis võib öelda, et iga detail esindab ühte komponenti ning koos need moodustavad ühe *Actor´*i.

Level (Editor)

Level Editor on mõeldud põhifunktsioonide rakendamiseks *Unreal Editor´*is. *Level Editor´*is on võimalik luua erinevaid kihte, et mugavamalt hallata mängu(de) stseene. Põhimõtteliselt on kiht 3D-keskkond, kuhu saab lisada mitmeid objekte, et kujundada maailm, kus mängija liigub.

Material (Editor)

*Material Editor´*is on võimalik luua või kasutada olemasolevaid materjale. UE4-s on materjalidega ümber käimiseks loodud mugav kasutajaliides, mis kujutab endast sõlmetaolist graafi. Materjal on vahend, mille abil antakse objektile soovitud väljanägemine. See defineerib pinna tüübi, millest objekt on tehtud. Tehnilise poole pealt käsitletakse materjali kui funktsiooni, millega määratletakse valguse käitumist teatud pinnaga [12].

Blueprint (Editor)

Blueprint on üks viis, kuidas luua uut tüüpi *Actor´*eid ja skriptimise tasemel sündmuseid, ilma et peaks kasutama programmeerimiseks C++. Selle eelkäijaks oli varasemates versioonides UnrealScript ning suures osas on neil kahel sarnane funktsionaalsus. Blueprint kujutab endast materjalidele sarnast sõlmetaolist graafi, millega on võimalik kontrollida ja luua kõiksugu sündmusi mängus. UE4 *Blueprint Editor´*is on võimalik töötada erinevate Blueprintidega ning neid omavahel ühendades on võimalik luua keerulisi funktsioone, mis on olulised mängu toimimiseks.

Matinee Editor

*Matinee Editor´*is on võimalik kiirelt luua lihtsamaid animatsioone. Selleks kasutatakse *keyframe´*e, et seada mingil kindlal ajahetkel *Actor´*i omadused. Näiteks kui jaotada teatud ajavahemik ja seada kindlale ajaühikule *Actor´*i asukoht, siis on võimalik animeerida *Actor´*i liikumist soovitud marsruudil.

Static Mesh (Editor)

Static Mesh on geomeetriline objekt, mis enamasti kujutab endast polügooni, mida puhverdatakse video mälus ja renderdatakse graafikakaardi poolt. *Static Mesh´*id võivad olla geomeetriliselt keerukad, kuna renderdamine on efektiivne. Kuna nad on puhverdatud video mälus, siis on neid võimalik teisaldada, pöörata ja mõõtkava muuta, kuid nende animeerimine pole kuidagi võimalik. Enamasti on *Static Mesh´*id 3D mudelid, mis on loodud 3D modelleerimiskeskkonnas (Blender, Maya, 3dsMax) ning siis imporditakse need *Unreal Editor´*i. Suures jaos moodustub mäng *Static Mesh´*idest. *Static Mesh Editor´*is on võimalik vaadata ja muuta *Static Mesh* omadusi [12].

Physics Constraint Actor

Physics Constraint Actor kujutab endast üht tüüpi lüli, mille abil saab ühendada kahte *Actor´*it, ning millest vähemalt üks simuleerib füüsikalisi omadusi. Selle abil saab lisada ka piiranguid ja kasutada füüsikalisi jõude. Eelkõige kasutatakse seda pendeldavate/kiikuvate objektide loomiseks, kus on oluline, et füüsika mõjuks limiteeritud piirkonda [12].

Cable Actor

Unreal Engine 4 programmeerija, James Golding, poolt loodud *Cable Actor/Component plugin* on lihtne viis luua kaablitaoline ühendus kahe *Actor´*i vahel. Kaabli parameetreid (pikkus, paksus) saab muuta *Editor´*is [13]. *Cable Actor* meenutab omaduste poolest kummitaolist kaablit ning seetõttu on seda mugav kasutada olukordades, kus on oluline kaablipikkuse muutumine ajas.

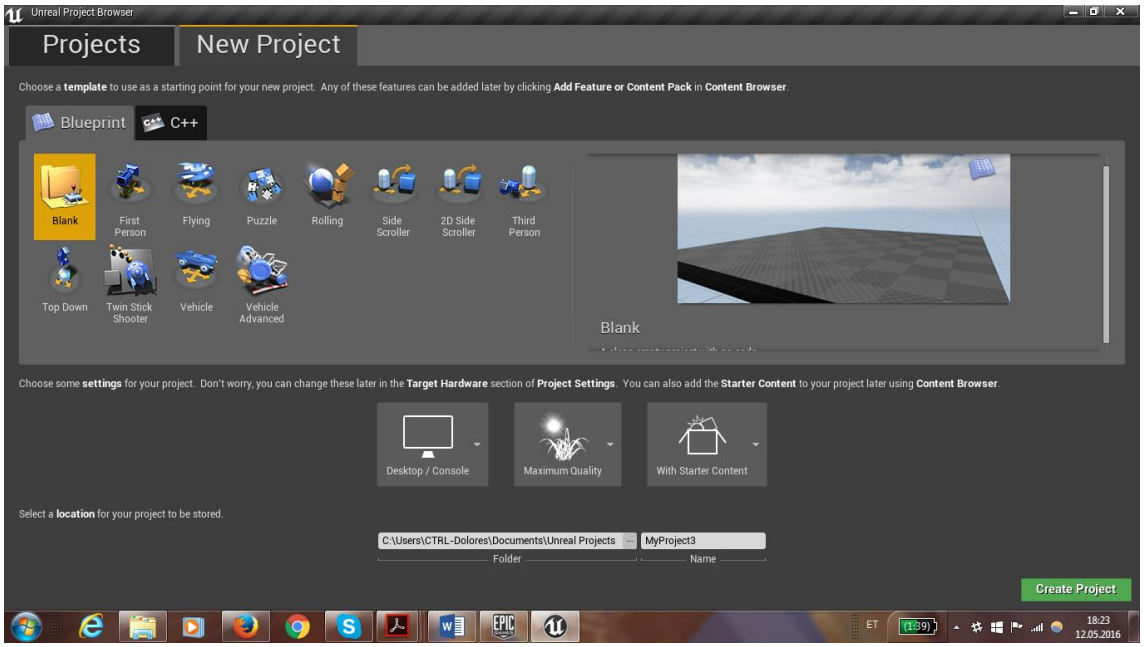
3.4 3DKraana simuleerimine Unreal Engine 4-s

Antud töös kasutasin UE4 3DKraana virtuaalse mudeli simuleerimiseks. Kuna üks eesmärkidest oli 3DKraana mudelit visualiseerida Oculus Rifti kasutades, siis UE4 kasutamine oli vägagi otstarbekas.

3.4.1 Uue projekti loomine Unreal Engine 4-s

Unreal Engine 4 on võimalik alla laadida nende ametlikult kodulehelt www.unrealengine.com, kuid kõigepealt on oluline ennast registreerida Epic Games kasutajaks. Peale seda laetakse alla Epic Games Launcher, kus on võimalik installida UE soovitud versioon. Selles töös on kasutatud versiooni Unreal Engine 4.10.4.

Kui avada UE4, siis esimesena on võimalik valida erinevate projektide vahel, mis on eelnevalt loodud. Samuti on võimalik alustada uut projekti, kus mängulooja saab seadistada projekti endale kohaseks (joonis 15).

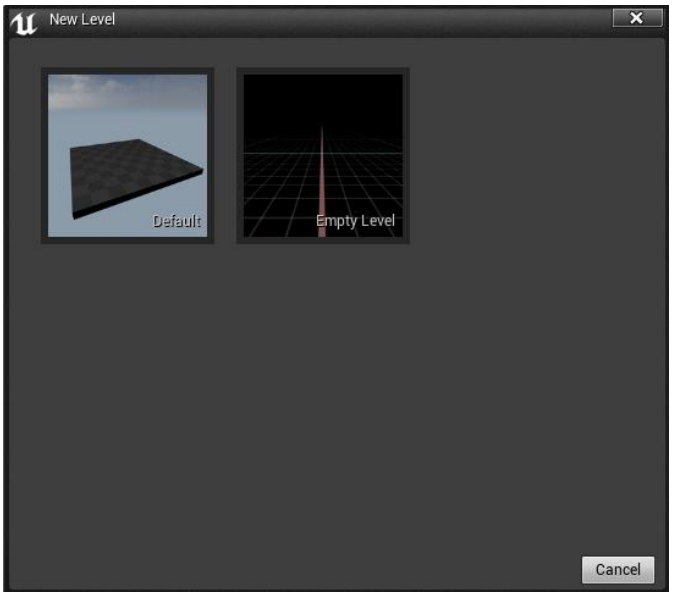


Joonis 15. Uue projekti loomine UE4-s.

Kui uus projekt on loodud koos *Starter Content*´iga, siis vaikeekraan kujutab endast lauda, põrandat, kuju ja kahte tooli koos valgusallika ning kaameraga (joonis 16).

Joonis 16. UE4 vaikeekraan.

Kuna on soov importida kraana mudel nn "tühja ruumi", siis selleks tuleks *File* menüüst avada *Open New Level* ja seejärel valikust *Default* (joonis 17), mille tagajärjel luuakse uus kiht.

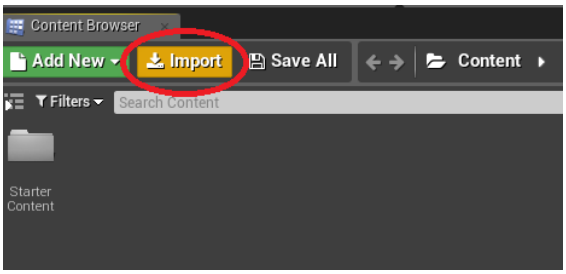


Joonis 17. Uue kihi loomine UE4-s.

Järgmise sammuna oleks vaja 3DKraana mudel, mis sai loodud Blenderi keskkonnas importida Unreal Engine 4, et alustada mudeli simuleerimist.

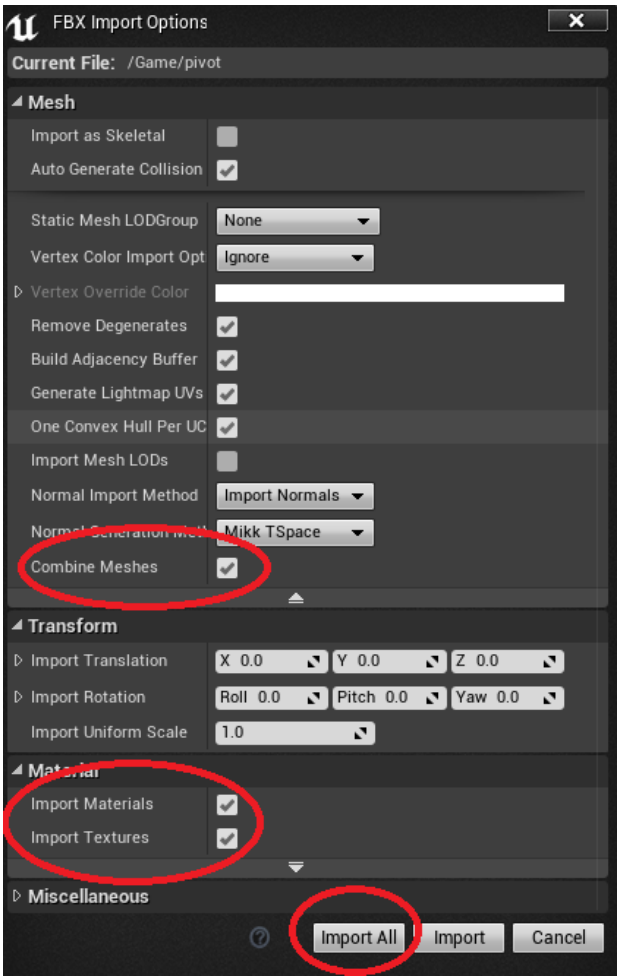
3.4.2 Mudeli importimine 3D modelleerimiskeskkonnast UE4

Mudeli importimine on tehtud UE4-s mugavaks ning selleks on vaja *Content Browser´*is avada *Import* aken (joonis 18). Peale seda on võimalik importida .FBX failid, mis loodi Blenderi keskkonnas.



Joonis 18. Mudeli importimine UE4-s 1.

Joonisel 19 on välja toodud olulised kohad, mida tuleks järgida importimisel.

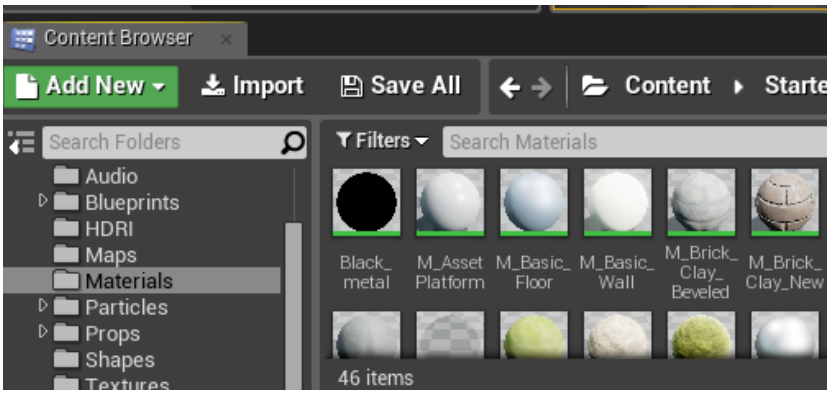


Joonis 19. Mudeli importimine UE4-s 2.

Kuna UE4-s on mugav luua uusi materjale ja tekstuure olemasolevaid kasutades, siis pole oluline nende importimine Blenderi keskkonnast. Samuti ei ole Blenderi materjalid kõige kvaliteetsemad, kui UE4-s mudelit renderdada. Vaikeseadena imporditakse kogu mudel kui ühe tervikuna, kuid kui on oluline säilitada detailsus, siis oleks enne importimist vaja elimineerida linnuke *Combine Meshes* valiku juurest.

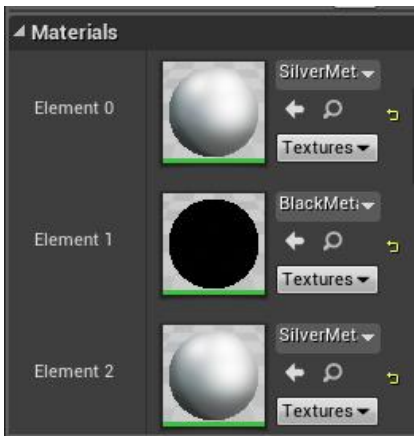
Antud töös on imporditud kraana liikuvad objektid eraldi, et hiljem rakendada neile MATLABi juhtimisalgoritmi.

Peale mudeli importimist saab objekte vedada kihile ja määrata korrektsed koordinaadid. Peale seda võiks lisada detailidele materjalid, et luua realistlikum väljanägemine. Selleks on võimalik kasutada olemasolevaid materjale (joonis 20) ning neid duplitseerides ja teatuid omadusi muutes saab luua realistliku välimusega mudeli.



Joonis 20. Algpaketiga kaasa saadud materjalid UE4-s.

Materjale saab lohistades *Actor´*ile peale kanda. Kindlasti on oluline kasutada Blenderis materjale, kuna UE4-s võimaldab see ühe objekti igale detailile erisugune materjal lisada. Näiteks antud töös koosneb kraana koormus kolmest erinevast osast ja materjalist ning UE4 importimisel tunneb programm selle omaduse ära (joonis 21). Seetõttu on võimalik ühele *Actor´*ile lisada mitu materjali.



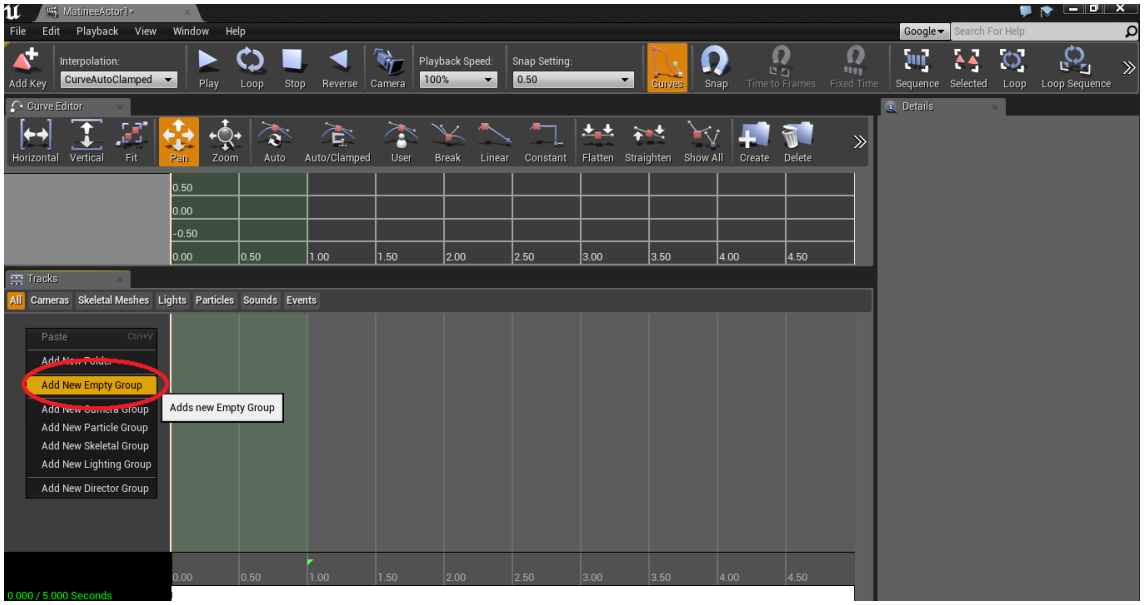
Joonis 21. *Actor* ja temale kuuluvad materjalid.

3.4.3 UE4-s 3DKraana mudeli simuleerimine

Üks viis UE4-s liikumist imiteerida on *Matinee* funktsiooni kasutades. Selleks tuleks tööriistaribalt avada *Cinematics* ning valida *Matinee* (joonis 22).

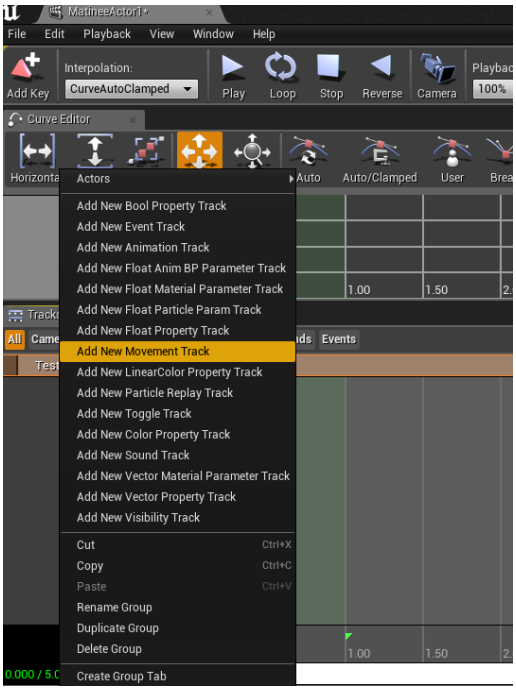
Joonis 22. *Unreal Editor´*i tööriistariba.

Peale seda avaneb *Matinee* kasutajaliides, kus on võimalik seadistada objekti liikumist. Alustuseks tuleks objekt kihil aktiviseerida ning seejärel parema hiireklõpsuga lisada uus tühi grupp (joonis 23). Peale *Enter* klahvivajutust saab loodud grupile nime lisada.



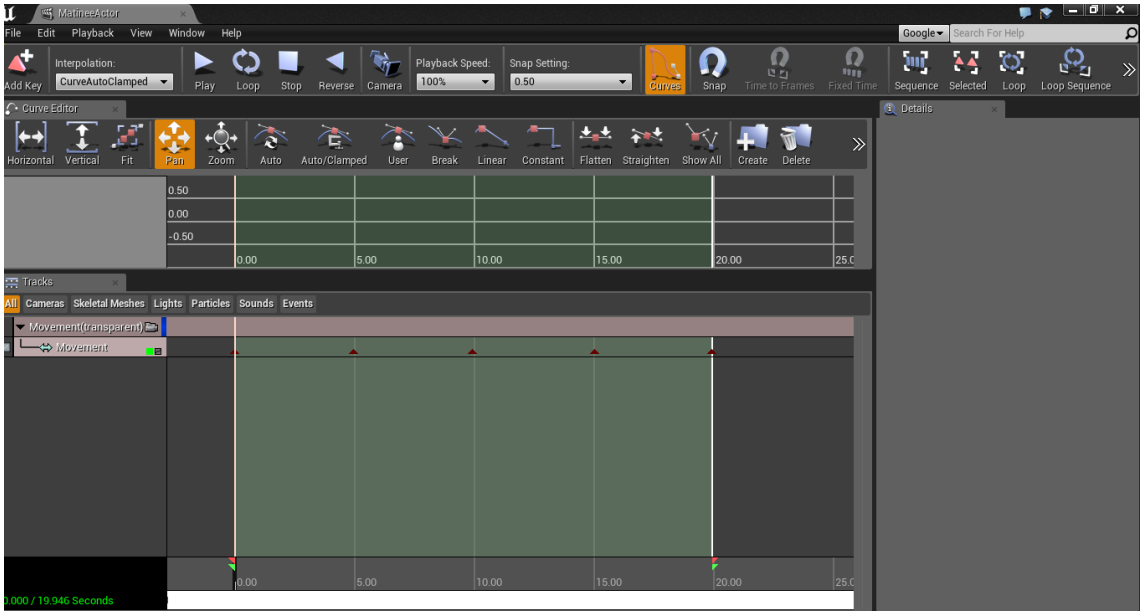
Joonis 23. Uue grupi loomine *Matinee Editor´*is.

Seejärel on oluline lisada *Matinee Track*. Valikuid on mitmeid, kuid kui on soov genereerida *Actor´*i liikumine mööda kindlat trajektoori, siis tuleks menüüst valida *Add New Movement Track* (joonis 24).



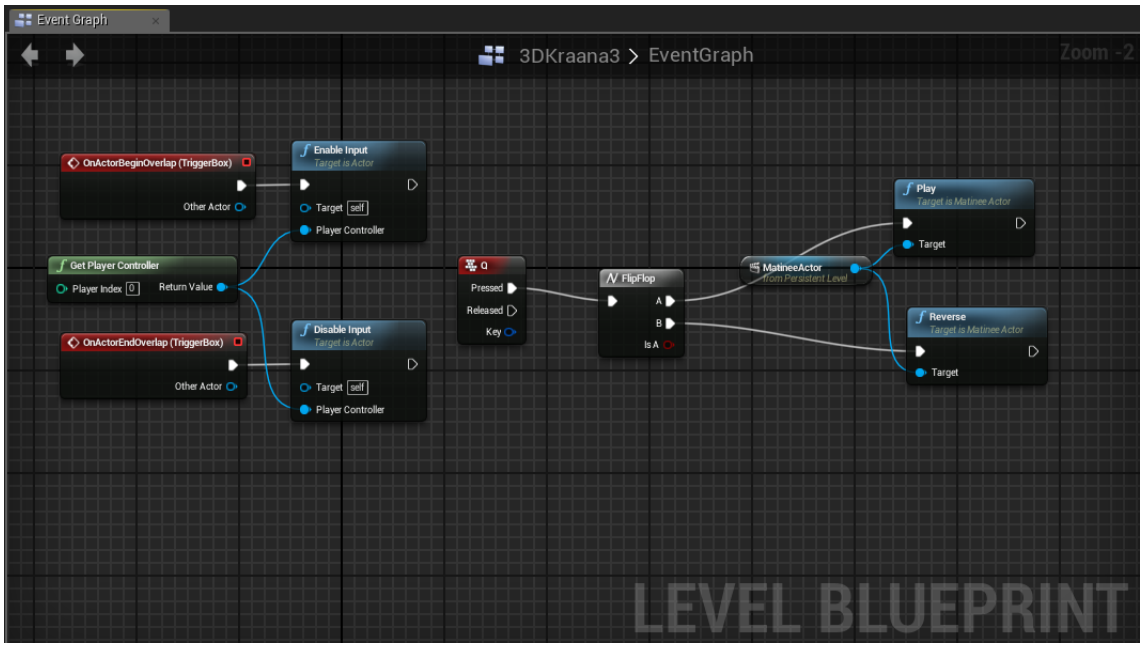
Joonis 24. *Matinee Track* menüü.

Peale *Movement Track* lisamist on võimalik alustada objekti liikumise animeerimist. Selleks saab muuta *Matinee Editor´*is animatsiooni pikkust ja tänu *keyframe´*idele saab määrata kindlas ajahetkes *Actor´*i asukoht. Joonisel 25 on tehtud vajalikud seadistused kraana ühe liikuva osa animeerimiseks. Kogu simulatsiooni kestus on 20 sekundit ning see on jaotatud neljaks osaks. Igale osale on määratud kindel asukoht *Level Editor´*is.



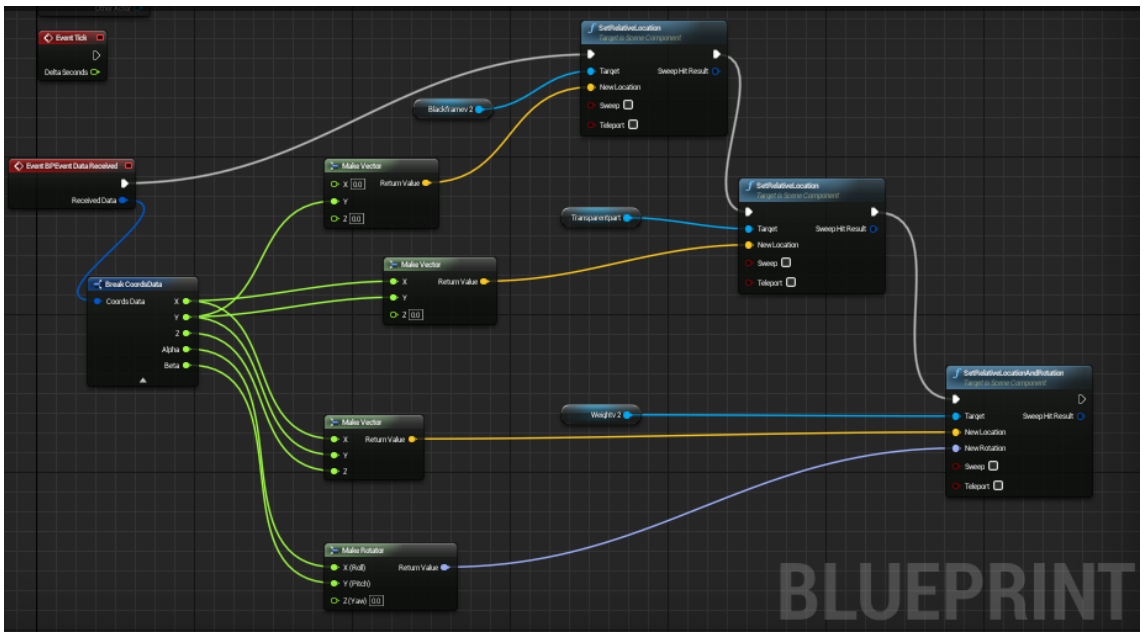
Joonis 25. *Actor´*i liikumise animeerimine kasutades *Matinee Editor´*i.

Viimase sammuna oleks mõistlik luua uus *Level Blueprint*, millega saaks kontrollida *Play Mode´*is kraana liikumist määratud klahvikombinatsiooniga (joonis 26). Üks lihtne moodus oleks kasutada komponenti *Box Trigger*. Tänu Blueprindile võimaldab see mängimise aeg teatud asukohas käivitada soovitud sündmus. Antud juhul tähendab see, et mängija sattumisel *Box Trigger´*i piirkonda on võimalik käivitada ja lõpetada klahvikombinatsiooniga *Actor*i liikumine.



Joonis 26. Blueprint *Actor*´i liikumise kontrollimiseks.

Matinee kasutamine on üks võimalikest viisidest UE4-s liikumist simuleerida, kuid antud töö eesmärk oli 3DKraana mudeli liikumist juhtida MATLAB-i kasutades. Selleks on oluline luua C++ programmikood või Blueprint mis suudaks suhelda MATLABiga. Oluline on, et MATLABis genereeritud koordinaadid jõuaksid UE4 ning *Actor* (3DKraana komponendid) liiguks MATLABis ette antud asukohta (koordinaatidele). Joonisel 27 on näidatud Blueprint, mille abil on võimalik 3DKraana koordinaadid MATLABist kätte saada.



Joonis 27. Blueprint 3DKraana koordinaatide kättesaamiseks MATLABist.

Kuna Unreal Engine ei võimalda modelleerimist, kuid võimaldab olemasolevaid komponente kasutada, siis hea võimalus on kasutada *Cable Actor´*it koormuse ja tõstevankri vahelise trossi kujutamiseks. *Cable Actor´*it on hea kasutada, kuna see imiteerib kummilaadset kaablit ning vastavalt koormuse asukohale muudab ta oma pikkust. *Cable Actor´*i mitmeid omadusi (pikkus, jämedus, segmentide arv) on võimalik muuta, mille tõttu on selle kasutamine veelgi mugavam.

3.5 Oculus Rift ja selle ülevaade

Oculus Rift on virtuaalreaalsuse peakomplekt, mille loomist alustas Palmer Luckey ning esimene prototüüp (Development Kit 1) avaldati aastal 2012 [14]. See on eelkõige mõeldud PC-dele, kuid kuna Windows 10 on platvormitundetu ning see töötab Xbox One-ga, siis on võimalik *stream´*ida mõningaid mänge Oculus Riftil. Rift toetab kahte põhilist mängu mootorit Unity ja Unreal Engine. Nüüdseks on mitmeid mänge, mis toetavad Oculus Rifti kasutamist ning tulevikus nende osakaal suureneb jõudsalt [15].

Virtuaalreaalsuse peakomplekte hakati looma 1950. aastast, kuid ükski neist pole olnud niivõrd edukas kui Oculus Rift, eelkõige puudulikule tehnoloogiale tol ajal. Oculus Rift pole mitte ainult meelelahutuslik, vaid see väldib ka nn "merehaiguse" tekkimist. Samuti on välditud enimlevinud probleeme nagu ajaline viivitus, piksliline graafika ja liikumise tuvastamine.

Oculus Rift tehnoloogia põhineb kahel ekraanil, mis kuvavad stereoskoopilist 3D pilti. Pilt saadakse tänu inertsiaalanduri, magnetomeetri ja prillide liikumist jälgiva välise kaameraga, mis aitab määrata kasutaja liikumist ruumis [15].

Märtsis 2016 tuli müügile ametlik versioon Oculus Rifti peakomplektist. Enne seda on avalikkusele tutvustatud mitmeid prototüüpe ning kahte neist (Development Kit 1 ja Development Kit 2) oli võimalik isiklikus otstarbeks sooritada [16]. Antud töös on kasutatud Development Kit 2 (joonis 28), mis oma omadustelt on kvaliteetsem kui Development Kit 1.

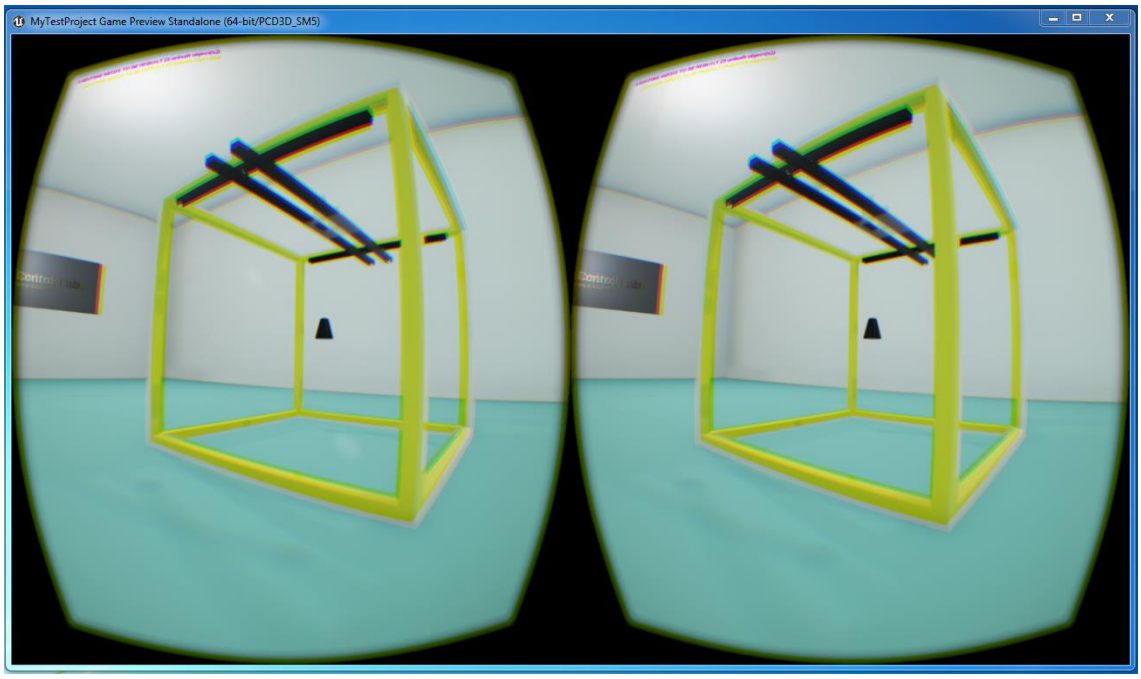
Joonis 28. Development Kit 2.

Oculus Rifti arendusmeeskond on märkimisväärne, kuna iga uue prototüübiga proovivad nad elimineerida probleemid, mis eelneval versioonil esinesid ning samuti loovad nad täiesti uusi funktsioone. Arendajaversioonide kättesaadavaks muutmine aitas kõvasti kaasa lõpliku versiooni arendusele.

Nagu eelnevalt mainitud on võimalik Oculus Rifti kasutada koos Unreal Engine 4-ga. Eelnevalt on vaja Oculus Rift ühendada PC-ga ning teha mõningad seadistused, et peakomplekt töötaks Unreal Engine 4-ga [17]. Igal järgneval korral on mugav *Unreal Editor´*i tööriistaribalt valida *Play Mode´*iks *VR Preview* (joonis 29).

Joonis 29. UE4-s loodud mudeli kujutamine virtuaalreaalsuses.

Peale *VR Preview Mode* aktiveerimist avaneb aken PC ekraanil, kus on kujutatud loodud mudel ning samuti kuvatakse 3D stereoskoopiline pilt Rifti ekraanil (joonis 30).



Joonis 30. 3DKraana kujutis Oculus Rifti ekraanil.

4 3DKraana virtuaalse mudeli juhtimine MATLABi kasutades

Füüsilist 3DKraana mudelit on võimalik kontrollida läbi MATLABi. Selleks on MATLABis olemas 3DKraana kooderitel põhinev juhtimismudel, mis tarkvara installides kaasa tuli. Käesolevas peatükis on lühike ülevaade MATLABist ja olulistest funktsioonidest, mis on vajalikud virtuaalse mudeli kontrollimiseks UE4-s.

4.1 MATLAB ja Simulink

MATLAB on neljanda põlvkonna programmeerimiskeel ja keskkond numbrilisteks uuringuteks. Peamisteks tegevusaladeks on tehted maatriksitega, algoritmide loomine ja andmete visualiseerimine. MATLABi kasutavad enamasti insenerid ja teadlased mitmetest valdkondadest teadustöödeks [18].

Simulink kujutab endast plokkdiagrammi, mille abil on võimalik modelleerida, analüüsida ja simulatsioonina jälgida süsteeme, mille väljundid muutuvad ajas [19]. Simulink võimaldab koostada süsteemi struktuurile vastava aseskeemi, valida üksikute lülide võimendustegureid ja ajakonstante, anda süsteemile ette vajalikke juhttoimeid ja häiringuid ning jälgida süsteemi reaktsiooni nendele toimetele [20].

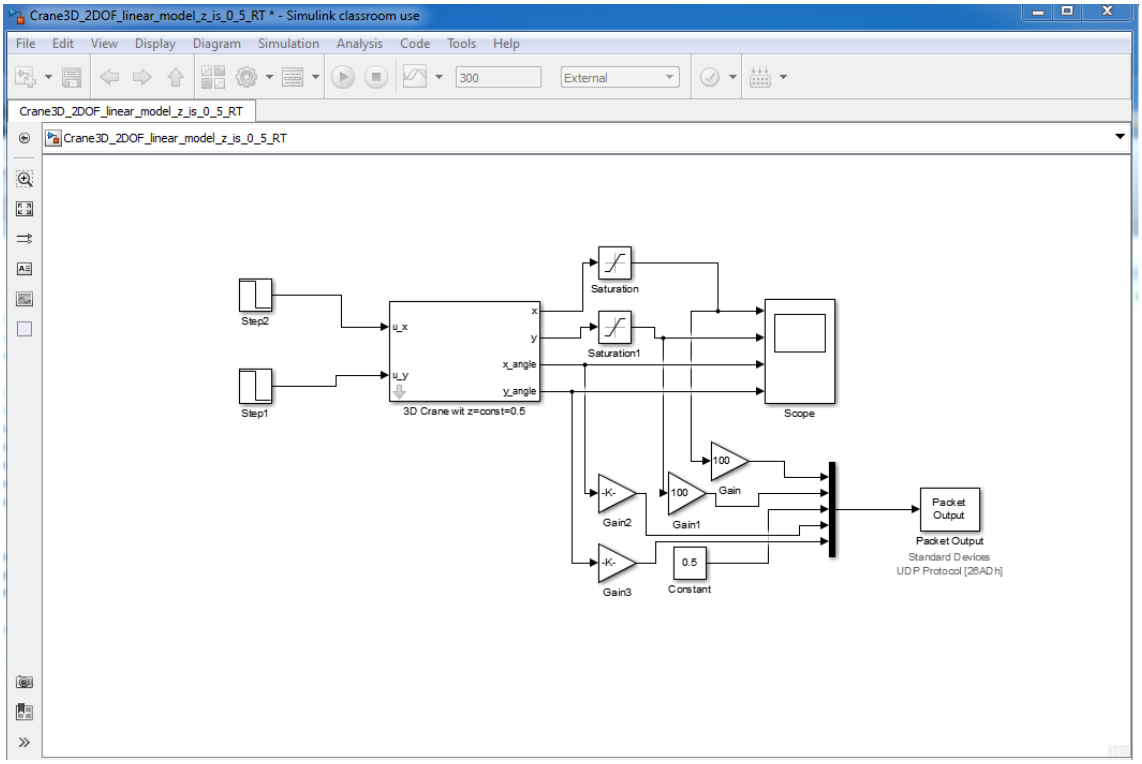
4.2 UDP ja *Real-Time Windows Target*

UDP protokoll on üks transpordikihi protokollidest, mida kasutatakse multimeedia edastamisel ning olukordades, kus on tähtis andmeside kiirus ja vähemtähtis kvaliteet, kuna pakettide kadumisel neid uuesti ei saadeta. UDP ei tekita virtuaalkanalit nagu TCP ning samuti ei nõua see kviitungeid, vaid lihtsalt saadab pakette.

Real-Time Windows Target võimaldab Simulinkis loodud mudeli abil kontrollida reaalajas teatud objekti (antud projekti puhul 3DKraanat). See koosneb sisend -ja väljund plokkidest, mille abil on võimalik pakette saata ühest kohast teise. Pakettide saatmiseks kasutatakse UDP protokolli.

4.3 3DKraana virtuaalse mudeli juhtimisalgoritm MATLABis

Käesolevas töös on kasutatud virtuaalse mudeli juhtimiseks lineaarset mudelit (joonis 31). Sellega kontrollitakse 3DKraana mudeli dünaamikat UE4-s, kuid selle käitumine ei ole üksüheses vastavuses reaalse mudeli omadustega. Lineaarne mudel on üks lihtsustatud viisidest, millega saab simuleerida virtuaalset mudelit. Mudelil on määratud silla ja vankri liikumisala piirid ning trossi pikkus on konstantne (0,5 m), mis lihtsustab simuleerimist.



Joonis 31. 3DKraana lineaarne juhtimismudel.

Kahe rakenduse (MATLAB ja UE4) vaheline suhtlus toimub, kasutades UDP protokolli. Kraana koordinaadid ja nurgad, mis on sisestatud MATLABis, saadetakse pakketidena Unreal Engine 4. UE4-s on loodud Blueprint kraana andmete lahti pakkimiseks ning simulatsiooni tekitamiseks. Kahe rakenduse vaheliseks suhtluseks on loodud programmikood C-keeles, mis võimaldab MATLABist pakettide saatmist UE4.

Kokkuvõte

Käesoleva töö lõpptulemusena loodi Tallinna Tehnikaülikooli laboratooriumis olevast 3DKraanast virtuaalne mudel Unreal Engine 4-s. Virtuaalse mudeli dünaamika kontrollimine toimub läbi MATLABi. Loodud mudelit on võimalik kasutada õppe- ning teadustöös 3DKraana käitumismudeli paremaks mõistmiseks.

Virtuaalset mudelit, mis on loodud Unreal Egine 4 keskkonnas, saab kuvada Oculus Riftil, mis tekitab realistlikuma tunde kasutajale. Kuna UE4 ei võimalda modelleerimist, siis on oluline kasutada spetsiaalset 3D-keskkonda mudelite loomiseks, milleks sobib väga hästi Blender.

Antud tööd edasi arendades on võimalik luua reaalsest mudelist täielik koopia, mida saaks kasutada erinevates valdkondades, säästes aega ja kulutusi. Kuna Oculus Rift on mõeldud virtuaalse reaalsuse tekitamiseks ning ei toeta liitreaalsust, siis on üsna keeruline liikumist kontrollida klaviatuuriga. Seetõttu oleks mugavam juhtimist kontrollida mängukontrolleriga, mille funktsiooni on võimalik edasi arendades lisada.

Algselt oli soov luua juhtimissüsteem, millega saaks katsetada erinevaid juhtimisalgoritme virtuaalsel mudelil, kuid lineaarne mudel seda täielikult ei võimalda. Edaspidi võiks töös kasutada 3DKraana tarkvaraga kaasa tulevat kooderitel põhinevat mudelit või automaatikainstituudi teaduslaboris töötava doktorandi, Andrei Aksjonovi poolt loodud matemaatilist mudelit, kuna antud süsteemid on kalkuleeritud matemaatika ja füüsika seadustele vastavalt. Eelnimetatud mudeleid oleks vaja modifitseerida antud projektile kohaseks, mis on üsna keerukas ja aeganõudev, kuid lõpptulemus oleks märkimisväärne ja igati kasulik virtuaalse õppelabori loomiseks.

Kasutatud kirjandus

[1] Virtual Reality Society. What is Virtual Reality? [WWW] http://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html (14.05.2016)

[2] 3DCrane User's Manual. [WWW] http://a-lab.ee/man/3DCrane-user-manual.pdf (21.04.2016)

[3] Overhead Cranes. [WWW] http://www.mhi.org/fundamentals/cranes (20.04.2016)

[4] Lukašin, A. Tööstuslikud elektriseadmed ja –paigaldised. [WWW] http://opiobjektid.tptlive.ee/Paigaldised/kraanad.html (21.04.2016)

[5] Aksjonov, A. 3D Kraana juhtimissüsteem : üliõpilastöö. Tallinn, Tallinna Tehnikaülikool, 2015.

[6] Blender.org. [WWW] https://www.blender.org/about/ (14.04.2016)

[7] Vaba Tarkvara ja GPL-i litsents. [WWW] https://wiki.blender.org/index.php/Doc:ET/2.6/Manual/Introduction/License (28.04.2016)

[8] Blender's History [WWW] https://www.blender.org/foundation/history/ (14.04.2016)

[9] Blender Reference Manual. [WWW] https://www.blender.org/manual/ (20.04.2016)

[10] Metric & Imperial Units in Blender. [WWW] http://www.katsbits.com/tutorials/blender/metric-imperial-units.php (20.04.2016)

[11] Lee J. Learning Unreal Engine Game Development. [Online] https://www.packtpub.com/packtlib/book/Game-Development/9781784398156 (13.05.2016)

[12] Unreal Engine 4 Documentation. [WWW] https://docs.unrealengine.com/latest/INT/index.html (02.04.2016)

[13] Golding, J. Cable Component Plugin for UE4. [WWW] https://www.unrealengine.com/blog/cable-component-plugin-for-ue4 (10.05.2016)

[14] Kumparak, G. A Brief History of Oculus. [WWW] http://techcrunch.com/2014/03/26/a-brief-history-of-oculus/ (14.05.2016)

[15] Smith, S. L., Andronico, M. What is the Oculus Rift? [WWW] http://www.tomsguide.com/us/what-is-oculus-rift,news-18026.html (14.05.2016)

[16] Oculus Rift. [WWW] https://en.wikipedia.org/wiki/Oculus_Rift (14.05.2016)

[17] Oculus Rift. [WWW] https://wiki.unrealengine.com/Oculus_Rift (14.05.2016)

[18] Rouse, M. MATLAB. [WWW] http://whatis.techtarget.com/definition/MATLAB (15.05.2016)

[19] Miidla, P. Simulink. [WWW] http://kodu.ut.ee/~peepm/matlab/simulink1.pdf (16.05.2016)

[20] Sillamaa, H. Süsteemiteooria. Tallinn : Tallinna Tehnikaülikool, 2003.