

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Arvutitehnika instituut

Martin Luik

Kolmemõõtmelise maailma loomine virtuaalprillidele

Bakalaureusetöö

Juhendaja: Eduard Petlenkov

Tallinn 2015

Autorideklaratsioon

Olen koostanud antud töö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud. Käesolevat tööd ei ole varem esitatud kaitsmisele kusagil mujal.

Kuupäev:

Autor:

Allkiri:

Annotatsioon

Käes olev töö uurib Unreal Engine 4 mootorit ja sellega seonduvaid mõisteid. Eesmärk on teha õpetus virtuaalse maailma loomist virtuaalprillidele milleks on Oculus Rift.

Lõputöö on kirjutatud eesti keeles ning sisaldab 60 lehekülge teksti ja koosneb 3 peatükist.

Sissejuhatus.....	6
1 Unreal Engine 4 omadused	7
1.1 Graafika	7
1.1.1 Viivitusega varjutamine	7
1.1.2 Poolläbipaistev valgustus	8
1.1.3 Sub-Surface Shading	8
1.1.4 Graafikaprotsessori osakeste simuleerimine	9
1.1.5 Vektorväljad.....	9
1.2 Materjal.....	10
1.3 Järeltötluse efektid.....	10
1.3.1 Ümbritsev kuupkaart.....	10
1.3.2 Bloom.....	11
1.3.3 Silmade kohanemine	11
1.3.4 Peegeldused objektiivis	11
1.3.5 Tooni kaardistamine.....	11
1.4 Kasutajaliides (UI).....	11
1.5 Animatsioon	12
1.6 Heli	13
1.7 Füüsika simulatsioon	13
1.8 Matinee ja filmindus	13
2 Unreal Engine 4 Editor'i terminoloogia	14
2.1 Projekt	14
2.2 Objekt.....	14
2.3 Klassid	14
2.4 Actors	14
2.5 Komponendid	15
2.6 Pawns.....	15
2.7 Karakter	15
2.8 Player Controller.....	16
2.9 AI Controller	16
2.10 Brushes	16
2.11 Levels	17
2.12 World.....	17
2.13 GameModes.....	17
2.14 GameStates	18
2.15 PlayerStates	18
3 Virtuaalmaailma loomine kasutades Unreal Engine 4 Editori	18
3.1 Projekti loomine	18
3.2 Skybox'i loomine	18
3.3 LandScape loomine	27
3.4 Playeri loomine.....	32
3.5 Objektide paigutamine.....	35

3.6	Matinee rakendamine	38
3.7	Oculusrifi rakendamine	42
	Kokkuvõte.....	44
	Kasutatud kirjandus.....	45

Sissejuhatus

Unreal Engine on loodud Epic Games studio poolt. Tegemist on mängumootoriga, mida kasutavad väga paljud mänguarendajad. Esimene versioon töötati välja aastal 1998, teise versiooniga tuldi välja aastal 2002 ja kolmanda esialgse versiooniga aastal 2004. Neljanda versiooni arendamist alustati aastal 2003 ja selle stabiilne versioon tuli välja 24. veebruaril 2015. Mootori ülesehitamisel on kasutatud erinevaid programmeerimiskeeli, milleks on C++, C#, GLSL, Cg ja HLSL. Kõige uuemal versioonil Unreal Engine 4 on väga lai platvormi tugi. Sinna alla kuuluvad Microsoft Windows, Linux, OS X, Xbox One, Playstation 4, HTML5, iOS, Android, Oculus Rift, Ouya ja HTC Vive. Praeguseks on mootor kõigile tasuta kättesaadav.

Oculus Rift on virtuaalreaalsuse prillide peakomplekt, mille on välja töötanud USA firma Oculus VR. Esimene prooviversioon arendati aastal 2012 ning teisega tuldi välja aastal 2014. Prillid koosnevad ühest ekraanist, mis on jagatud kaheks ja neid katavad kaks lääts, mis tekitavad kasutaja jaoks ruumilise efekti. Lisaks on veel välja töötatud positsiooni jälitaja (*position tracker*), mis teeb kindlaks kasutaja asendi ruumis.

Bakalaureusetöö eesmärgiks on koostada algõpetus kolmemõõtmelise maailma loomiseks kasutades Unreal Engine 4.

Töö koosneb sissejuhatusest, kolmest peatükist, järeldusest, kasutatud kirjanduse loetelust ja lisadest.

Esimeses peatükis käsitletakse Unreal Engine 4 omadusi.

Teises peatükis antakse ülevaade Unreal Engine 4 virtuaalmaailma loomise terminoloogiast.

Kolmandas peatükis kirjeldatakse etapiviisiliselt virtuaalmaailma loomist.

Töö sisaldab veel jooniseid ja fotosid Unreal Engine 4 editorist, samuti soovitusi ning näpunäiteid järgmisteks samasugusteks projektideks.

1 Unreal Engine 4 omadused

1.1 Graafika

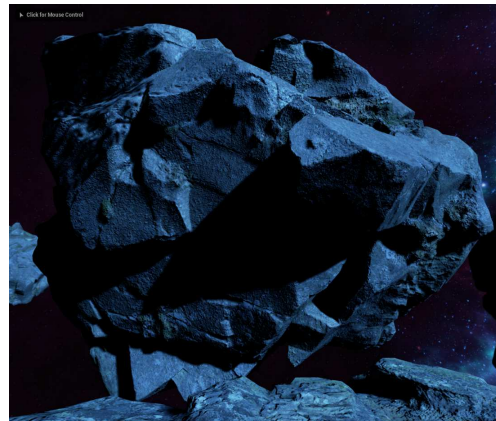
Unreal Engine 4 (edaspidi UE4) süsteemi arhitektuur on uudne, sealhulgas DirectX 11 tugi, mis sisaldab viivitusega varjundeid, globaalset valgustust, valgustatud poolläbipaistvust ja järeltöötlust, sammuti GPU osakeste simuleerimist kasutades vektorväljasid [1].

1.1.1 Viivitusega varjutamine

Varjude tekitamine 3D objektidele virtuaalmaailmas on samm reaalsusele lähemale. UE4-s on rakendatud viivitusega valgustus. See tähendab seda, et kui objekt on virtuaalmaailmas paika pandud, siis UE4 kogub kõik andmed, mida see objekt omab, geomeetrisse puhvrissse [10]. Geomeetrisse puhver kogub infot selle kohta, mis asendis objekt mänguväljal asetseb ja mis materjalist ta koosneb, näiteks puit, kivi või vesi. Pärast seda arvutatakse välja iga varju piksel seal, kus valgus paistab objektile otse peale ja seal, kus valgus paistab kõrvale, kasutades samal ajal andmeid tekstuuri puhvrissist.



Enne varjutamist



Pärast varjutamist

UE4-l on nelja eri liiki valgustamise viise.

- Directional light
- Point light
- Spot light

- Sky light

Directional light - valgus, mida rakendatakse välise keskkonna valgustamiseks virtuaalses maailmas.

Point light - valgus, mis toimib sarnaselt nagu lambipirn.

Spot light - kasutatakse siis, kui on vaja valgustada konkreetset objekti või kindlat punkti.

Sky light - on vahend millega saab määrata taevast tuleva valguse tugevust.

1.1.2 Poolläbipaistev valgustus

Poolläbipaistev valgustus efekt võimaldab varjundeid tekitada kahe poolläbipaistva objekti segunemisel, näiteks valguse segunemine suitsuga. [1].



1.1.3 Sub-Surface Shading

Tekstuuridele on loodud varjutamise mudel, milleks on "MLM_Subsurface" [1]. See on vahend, millega saab töödelda tekstuuri 3D objekti pinnal rohkem realistlikumaks, näiteks inimese nahka [16].

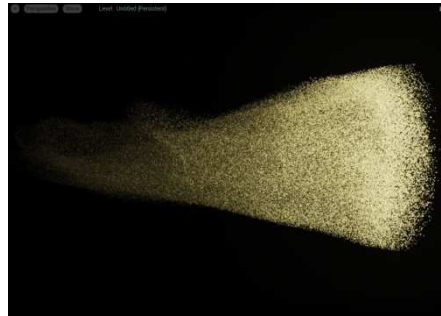


1.1.4 Graafikaprotsessori osakeste simuleerimine

UE4 võimaldab osakeste simuleerimist graafikaprotsessori peal [1]. Kui traditsiooniline keskne protsessor (CPU) lubab umbes tuhat osakest kaadris, siis graafikaprotsessor (GPU) suudab tõhusalt simuleerida ja esitada sadu tuhandeid osakesi [1]. Osakeste simuleerimise abil saab luua näiteks, tolmu, liiva või tule puhul sädemete animatsiooni.



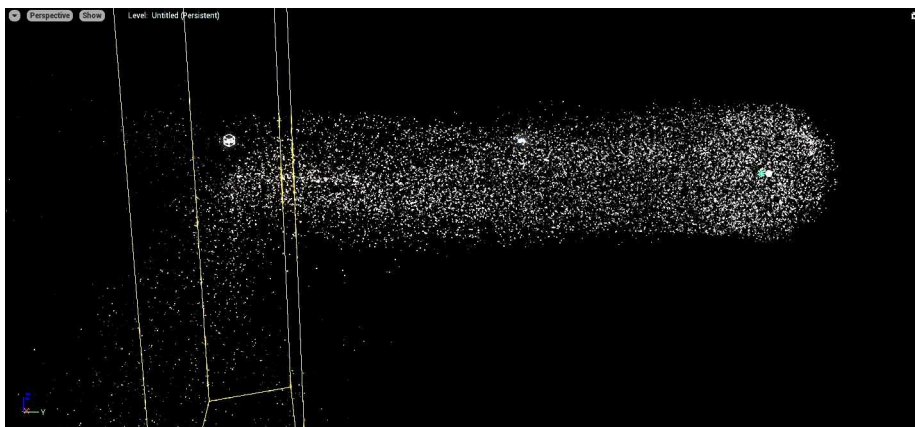
CPU osakeste simuleerimine



GPU osakeste simuleerimine

1.1.5 Vektorväljad

Peale graafikaprotsessori osakeste simuleerimise on võimalik rakendada vektorväljasid [1]. Vektorväljad on ühtne võrk vektoritest, mis määravad ära osakeste liikumise suuna [1]. Vektorväljad on paigutatud kolmemõõtmelisse maailma nagu 3D objektid ja neid saab samamoodi üle kanda/teisaldada, pöörata ja skaleerida [1]. Nad on dünaamilised ja neid saab igal ajal liigutada [1]. Vektorväljasid kohtame näiteks siis, kui meil on vaja määrata, mis suunas virtuaalne suits või vesi liiguks.



1.2 Materjal

Materjal koosneb tekstuurist, mida saab rakendada 3D objektile, et anda sellele välimus.

Lihtsamalt öeldes on materjal nagu värv, mis katab 3D objekti.

UE4-s rakendatakse füüsika reeglitel põhinevat varjutamise mudelit. See tähendab seda, et kui pinnase kiht on ebäühtlane, näiteks kivil, siis ta loob sinna varjundeid, kuhu valgus ei jõua.

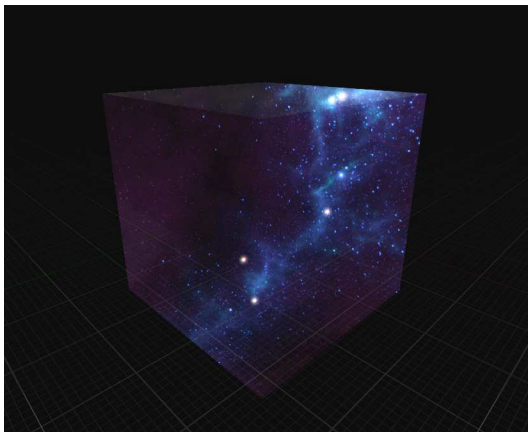
See tekitab efekti, et pinnas on ebäühtlane.

1.3 Järeltöötuse efektid

UE4 on varustatud mitmete järeltöötuse efektidega, mis võimaldab kunstnikel ja disaineritel kohendada üldvaadet ja tunnetada tegevuspaika. Näiteks 3D objektid või efektid, mis sisaldavad eredusefekti (*HDR blooming effect on bright objects*).

1.3.1 Ümbritsev kuupkaart

Ümbritsev kuupkaart on virtuaalmaailmas taust, mis tekitab vaatleja jaoks illusiooni, et maailm on suurem kui ta tegelikult on.



1.3.2 Bloom

Bloom efekt võimaldab heledat objekti tumedal taustal rohkem esile tuua.

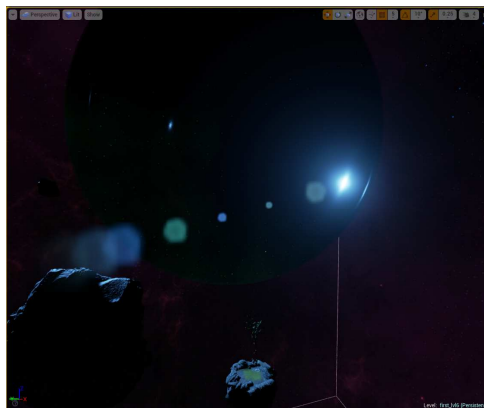


1.3.3 Silmade kohanemine

UE4 süsteem võimaldab simuleerida silmadega sarnaselt käituvat kohanemist erinevate valgus mõjutustele, näiteks kui liikuda virtuaalses maailmas heledamast keskkonnast tumedamasse keskkonda, võtab natukene aega, kuni pilt selgemaks läheb[1].

1.3.4 Pegeldused objektiivis

Lens Flare efektiga on võimalik luua/tekitada hajutatud valgus, kui vaadata eredat objekti [1].



1.3.5 Tooni kaardistamine

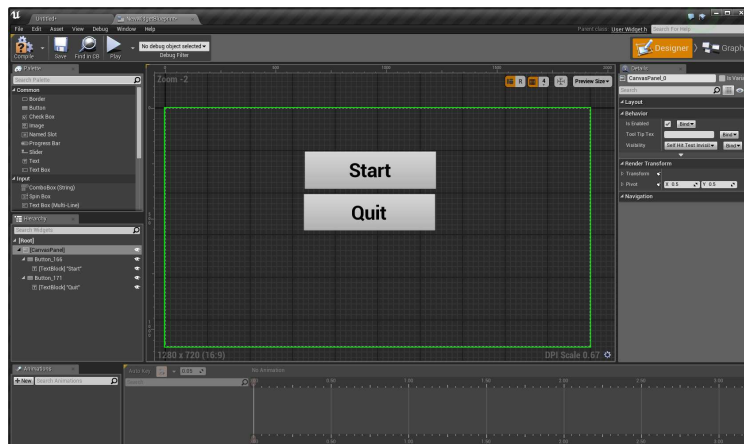
Tooni kaardistamine lubab värvi nihutada keskkonnas nii, et kohad, mis pole nähtavad, kuid vajavad esiletoomist, tuuakse rohkem esile [1].

1.4 Kasutajaliides (UI)

Unreal Motion Graphics UI Designer (UMG) on visuaalne kasutajaliidese haldamise töövahend, mille abil saab luua kasutajaliidese elemente nagu näiteks mängudes HUDs, menüüd või ükskõik missugune liides, mis on graafiliselt esitatud kasutajale. UMG on

tööriist, mis sisaldab eelnevalt tehtud valmis funktsioone, mida saab rakendada oma loodud liidesel, näiteks button, checkboxes, slider, progress bars. Tööriist on spetsiaalselt tehtud nii, et seda saab rakendada Blueprindi keskkonnas, millel on kahte sorti koostamise viise.

Disaineril on võimalik visuaalselt kujundada liidest ja aluspõhi valmis ehitada kasutades funktsiooni, milleks on sündmustikust koosnev trepitatud süsteem, mille abil saab toimima panna loodud kasutaja liides.



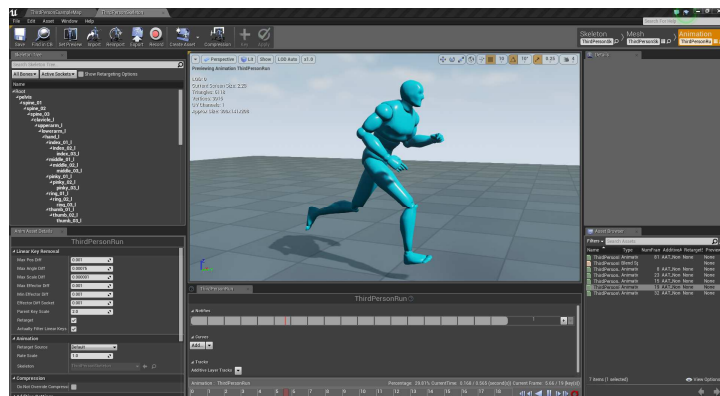
1.5 Animatsioon

UE4 animatsioonisüsteem võimaldab väga põhjaliku kontrolli loodud karakteri ja tema skeleti võrgustiku üle. Animeerimiseks on UE4 välja pakkunud kolm erinevat funktsiooni:

AnimationSequens - funktsioon, mis võimaldab mängija liikumist muuta rohkem realistlikumaks.

MorphTargets - funktsioon, mis annab meil võimaluse luua kahjustuste või näoilmete animatsioone.

SkeletalControls - funktsioon, mille abil saame luua muutusi skeletoni luudes.



1.6 Heli

Helil on väga oluline roll mängumaailmas, kuna see loob usutava ümbritseva keskkonna. Loodud virtuaalmaailm muutub kasutaja jaoks tõeliseks elamuseks, kui seda ümbritsevad helid, helid loodusest või masinatelt, muusika, näitlejate dialogid [5]. UE4 helisüsteem on varustatud töövahenditega, mis võimaldavad vormida heli, et anda kasutajale soovitud meeleolu [5]. See on hea süsteem, sest kui midu peab heli töötlemiseks kasutama lisaprogramme, siis UE4 võimaldab heli töödelda väga heal tasemel süsteemi enda sees [5].

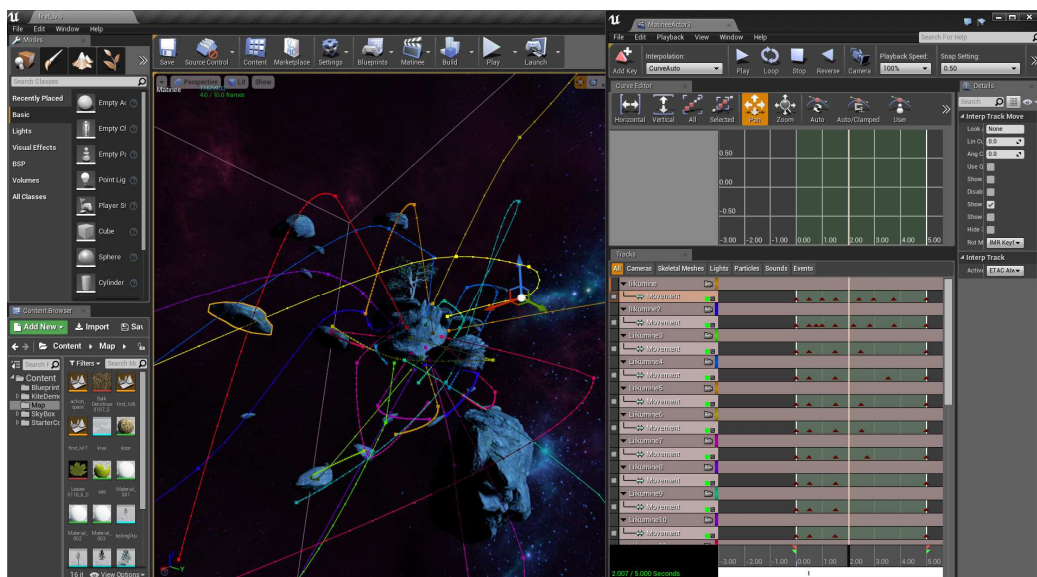
UE4 on välja pakkunud heli töötlemiseks rakenduse "*Sound Cues Editor*" [5]. Selle abil on võimalik töödelda heli ja luua erinevaid süsteeme, kuidas heli peaks keskkonnas toimima [5].

1.7 Füüsika simulatsioon

UE4 kasutab PhysX 3.3 füüsika mootorit, mis simuleerib, kalkuleerib ja teostab kõiki kokkupõrkeid virtuaalses maailmas[6]. Kasutades füüsikat virtuaalses maailmas võib parandada/täiustada tunduvalt virtuaalset reaalsust.

1.8 Matinee ja filmindus

Matinee on animatsiooni töövahend, mis võimaldab animeerida "Actorit", näiteks filmiklipina või mängu sisest hetkesündmust. Süsteem on üles ehitatud nii, et võimaldab klipitada stsene andes Actorile kindla osa, mida ta peab mingi aja jooksul täitma. *The Matinee* on väga sarnane teiste professionaalsete videotöötlusprogrammidega.



2 Unreal Engine 4 Editor'i terminoloogia

Kolme- või kahemõõtmelise virtuaal maailma loomiseks on UE 4 loonud editori, mille abil saab valmis ehitada mängu. Editor on loodud ainult taseme disainimiseks ja mängu reeglistiku üles ehitamiseks. Editori kasutades on vaja teada terminoloogiat.

2.1 Projekt

Uue projekti loomiseks on UE4 editor välja toonud kaks valikut milleks on blueprint või C++. Blueprint süsteem on rohkem mõeldud neile, kes ei oska väga programmeerida [7]. Tegemist on lihtsa mugava visuaalse süsteemse trepitatud programmeerimiskeelega. Lisaks pakutakse projekti loomisel valmis aluspõhjasid, mis koosnevad erinevatest mängu moodustest.

2.2 Objekt

Objekt on unreal engine 4 vundament, mis on ühenduses erinevate funktsioonidega [7]. Kõik asjad Unreal Engine 4 päritakse objektilt [7]. Programmeerimis keeles C++ on tähistatud objekt nimega UObject, mis on põhiklassiks teiste objektide jaoks [7].

2.3 Klassid

Klass defineerib UE4 jaoks ära, milline peaks olema "Actor" või objekti omadus ja käitumine mängu siseselt [7]. Klassidest moodustub puu, kus on näidatud ära superklass ja tema alla kuuluvad alamklassid. Klasse saab moodustada kasutades C++ või Blueprinti [7].

2.4 Actors

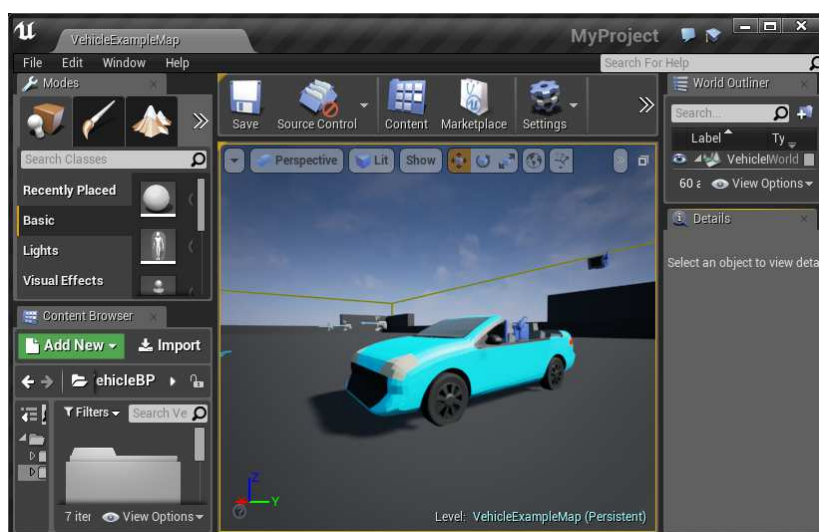
Actor on mistahes visuaalne objekt, mida saab rakendada taseme loomisel [7]. Ta kuulub üldisesse klassi, mis toetab lihtsamat kolmemõõtmelist muundamist [7]. Programmeerimis keeles C++ on "Actor" alamklassina teistele Actoritele tähistatud AActor [7].

2.5 Komponentid

Komponent on osa funktsionaalusest, mida saab lisada Actorile [7]. Komponent ilma Actorita ei saa eksisteerida [7]. Selleks, et lisada komponente, peab kindlasti aluseks olema "Actor" [7]. Näiteks me ei saa kasutada materjali enne, kui me pole loonud Actorit, kelle peal seda rakendada saaks.

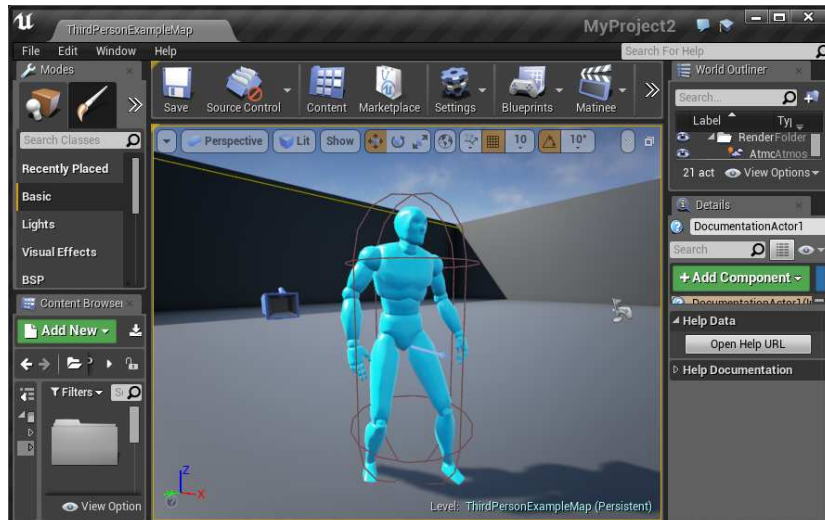
2.6 Pawns

Pawns ehk käsilane kuulub Actori alamklassi, ta töötab kui mängusisese avatarina [7]. Käsilast saab kontrollida mängusisene mängija või siis tehisintellekti poolt ilma mängija sekkumiseta [7].



2.7 Karakter

Karakter kuulub "Pawn Actor" alamklassi, mida saab kasutada kui mängija karakterina [7]. Karakteri alamklass sisaldab vahendeid, millega saab kasutaja kontrollida tema liikumist näiteks kasutades klaviatuuri või kontrollerit [7].



2.8 Player Controller

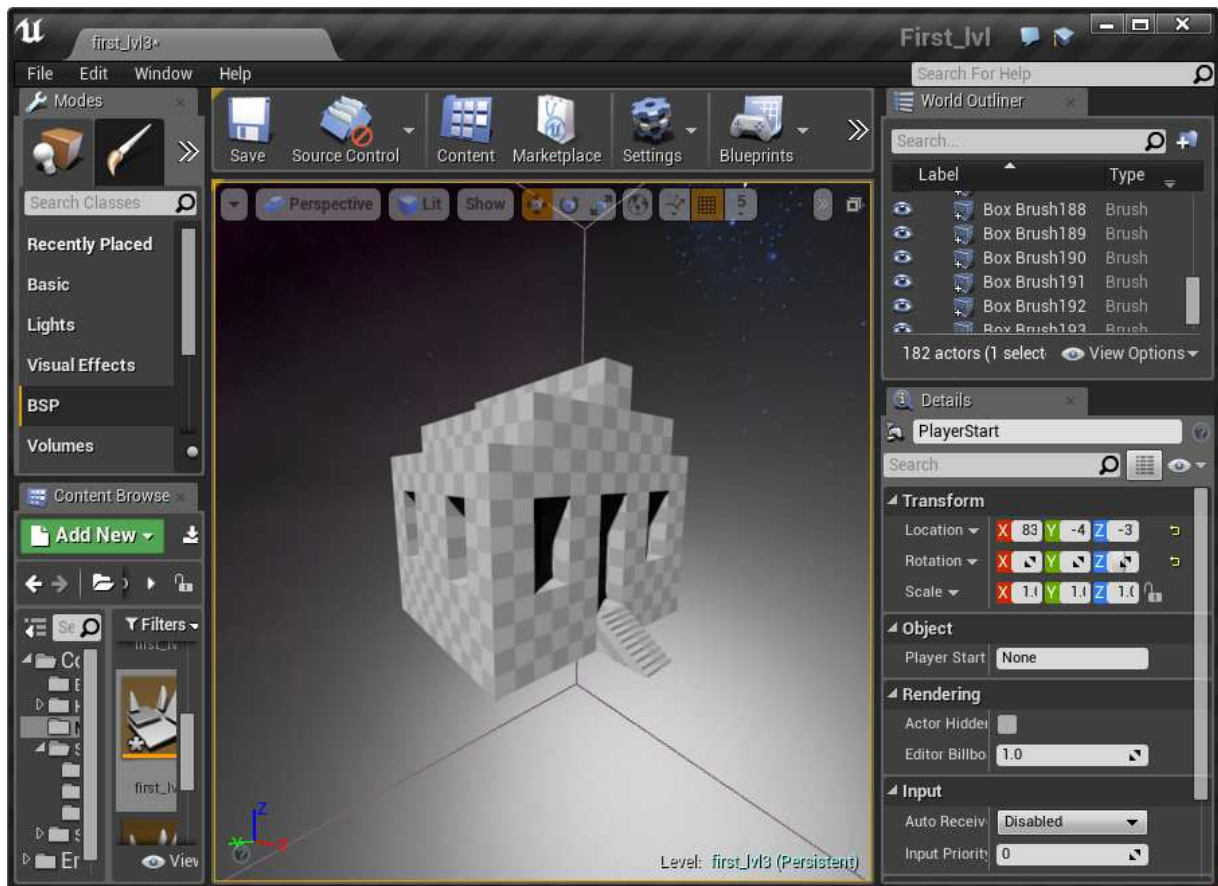
Player Controller on liides Pawns ja inimese vahel. Kontrollides annab kasutaja signaali väljaspool virtuaalmaailma, mis edasi kandes tõlgitakse käsuks ning peale seda hakkab player käsujärgi rolli täitma. UE4 pakub projekti loomisel lisaks juba valmis kontrollereid.

2.9 AI Controller

Tehisintellekti kontroller toimib sarnaselt nagu PlayerController ainult, et erinevus seisneb selles, et AI kontroller täidab käsku ilma kasutaja sekkumiseta [7].

2.10 Brushes

Brushes ehk geomeetriline pintsel, kasutatakse selleks, et luua esialgset taseme konstruktsioon [7]. See on algne töövahend, et luua lihtsamaid ehitisi [7]. Kui ei ole kolmemõõtmelise modelleerimise tarkvara, siis algajatel on see töövahend heaks asenduseks. Brushes on kõige kasulikum kasutada leveli protüüpimiseks [7].



2.11 Levels

Level ehk tase on kasutaja poolt defineeritud mängu ala [7]. Level koosneb valmis kujundatud näitlejate objektidest. Unreal Engine monteerija on iga level salvestatud ".umap" faili formaadina [7].

2.12 World

Maailm koosneb leveli listidest, mida iga kord laaditakse mängu jooksul. See käsitleb leveli voogu ja Actorite tekitamist [7].

2.13 GameModes

GameMode koosneb reeglistikest, mida looja määrab. Reegleid saab määrata mängijale, kes siseneb virtuaalsesse maailma [7]. Gamemodes on vaikimisi määratud Pawn klass, HUD class ja PlayerController Class [7].

2.14 GameStates

GameStates on mängu olek, mis annab kasutajale informatsiooni mängu seisust, näiteks tuuakse välja skoor, nimekiri mängijatega, kes on mitmikmänguga ühenduses või ülesande seisund mängu hetkel [7]. Üldiselt öeldes on gamestate mängurada, mille omadused mängides muutuvad.

2.15 PlayerStates

PlayerStates on mängija olek mängus. Sellega näidatakse näiteks mängija nime ja üldist seisundit [7].

3 Virtuaalmaailma loomine kasutades Unreal Engine 4 Editori

Selleks, et luua kolmemõõtmelist maailma on vaja mudeleid, mida saaks rakendada taseme loomisel. UE4 on ainult taseme disainimiseks ja mängu reeglistike üles ehitamiseks, seega ei saa luua mudeleid täiesti nullist, selleks on vaja eraldi programmi. Hea tasuta modelleerimise programm on näiteks "Blender", aga selles projektis seda ei rakendata. Alustuseks pakub UE4 enda pakutud pakke, mis sisaldavad juba valmis tehtud mudeleid.

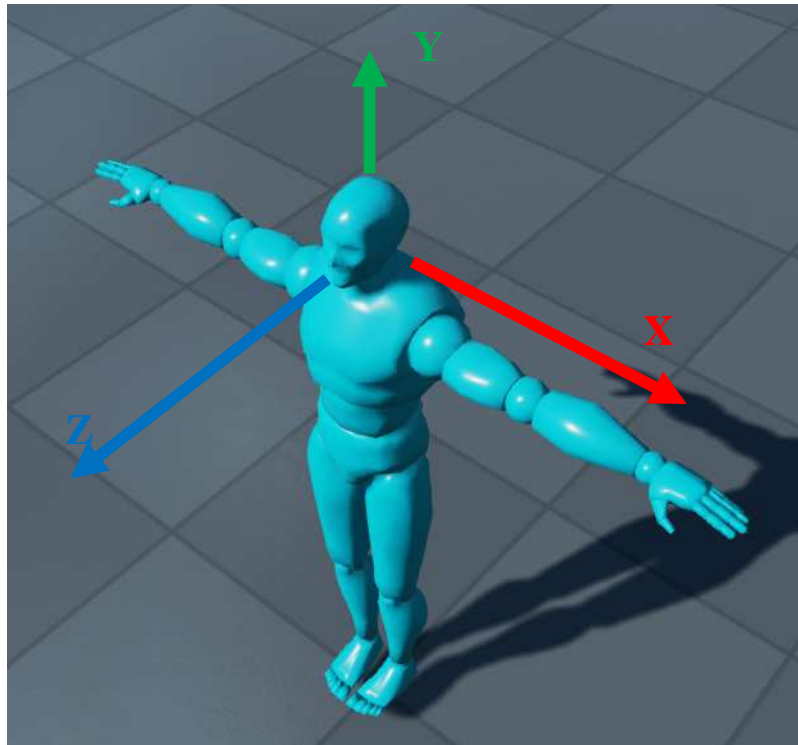
3.1 Projekti loomine

Projekti loomisel on antud kaks valikut, kuidas virtuaalset maailma üles ehitada. Esimene valik on Blueprint ja teine on C++. Mõlema valiku puhul jäävad aluspõhjad samaks. Esmane valik võiks olla Blueprint, sest sellega on lihtsam ja mugavam süsteemi üles ehitada. Projekti seadistamiseks on soovitatav valida Blueprinti alt "Blank", et alustada üles ehitamist täiesti algusest. Seejärel tuleks valida, mis platvormile virtuaalmaailma luua. Esmane valik võiks jääda Desktop/Console ja kuna lisaks on veel vaja 3D mudeleid, siis määrata lisaks veel "With Starter Content", mis sisaldab valmis mudeleid ja tekstuure. Kui projekt on loodud, avaneb töölaud.

3.2 Skybox'i loomine

Skybox on mõeldud selleks, et luua virtuaalmaailma taust, mis tekitab illusiooni, et loodud maailm on palju suurem kui ta tegelikult on. See koosneb kuuest pildist ja pildid sisaldavad laiali laotuna ühte kindlat ruumi.

Piltide lõikamisel peame arvestama vaatleja jaoks X, Y, Z telge nii positiivses kui ka negatiivses suunas.



X - positiivne ja negatiivses suunas on pildid parem ja vasak

Y -positiivne ja negatiivses suunas on pildid ülevalt ja alt vaates

Z - positiivne ja negatiivne suunnas on pildid eest ja tagant vaates

UE 4 jaoks peame pildid paigutama vastavalt antud joonise järgi.

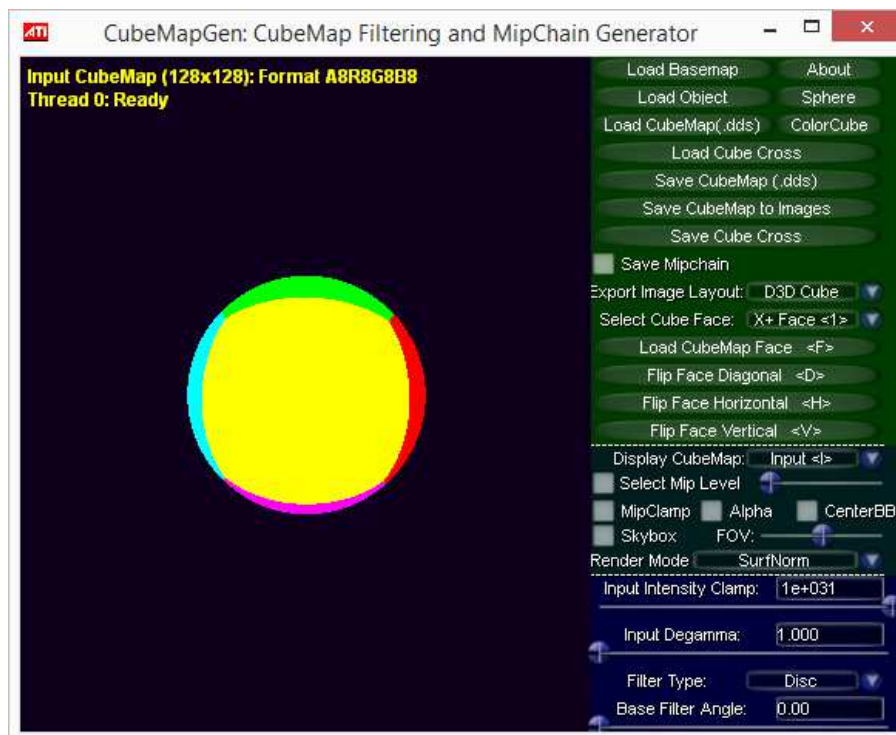


Alustades Skyboxi tegemist täiesti nullist, läheks vaja lisa programmi, mis genereerib pildi UE4 jaoks õigesse formaati. Nendeks on kas ATi poolt välja töötatud CubeMapGenerator või siis Nvidia Texture Tools.

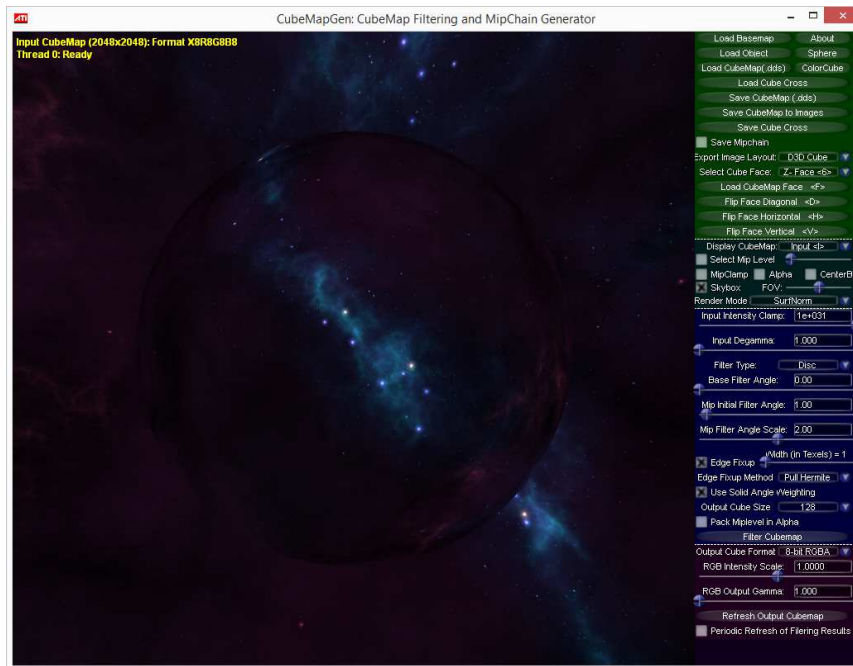
Projektis on plaanis luua keskkonnaks fantaasiamaailm, selleks on hea lisaprogramm Spacescape, mis sisaldab erinevaid pilte kosmosest ja on tasuta allalaetav. Programmil on hea

lisafunktsioon mis lõikab välja Skybox'i jaoks vajalikud pildi osad nii et sätib x, y, z positiivse ja negatiivse piirkonnad vaikumisi ära.

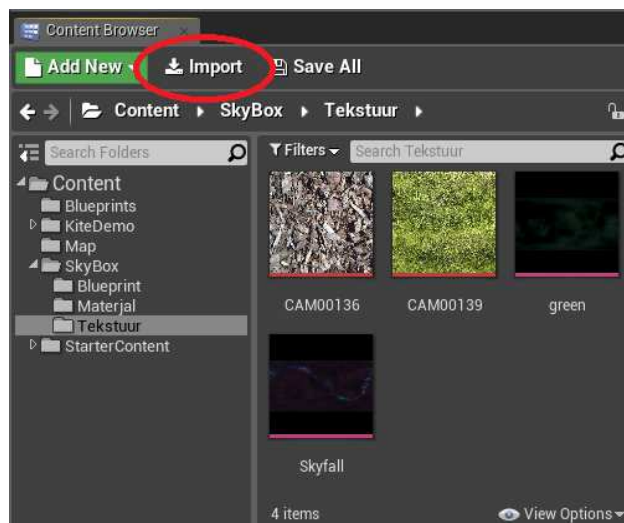
Enne UE4 Skybox'i importimist on vaja pildid paika panna, selleks tuleks kasutada lisa programmi CubeMapGen'i. Märkuseks võib öelda, et maksimum resolutsioon, mida CubeMapGen suudab genereerida on 1024x1024, sellest kõrgemat genereerides programm läheb lihtsalt katki. Kui on soovi parema resolutsiooniga Skybox'i teha, siis on välja pakkuda programm Nvidia Texture Tools, aga selleks on vaja lisaks pildi töötlemiseks Photoshopi.



"CubeMapGen" piltide lisamiseks tuleb ükskõik enne valida "Select CubeMap Face" ja siis "Load CubeMap Face". Kuna Spacescape programm lõikab pildid kuueks osaks ja sätib nad õigesse järjekorda, siis saab CubeMapGen programmis mugavalt makro nuppe kasutada. Numbrid ühest kuni kuueni saab valida "Cube Face"-i ja vajutades klaviatuuril "F"-i saab valida pilti.

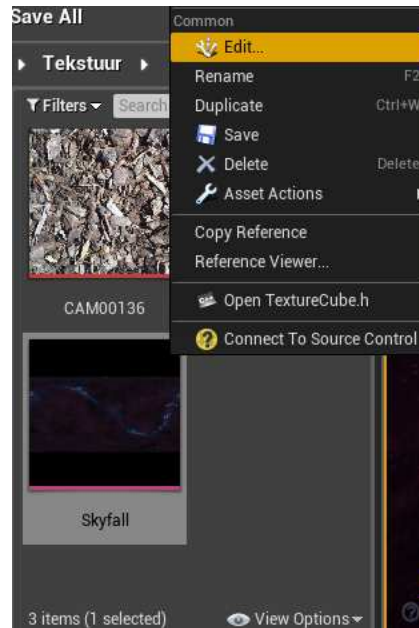


Kui pildid on CubeMapGeneraatoris paika saadud, tuleks salvestada Skybox UE4 jaoks „dds” formaati ning seejärel tuleb editoris importida "dds" fail.

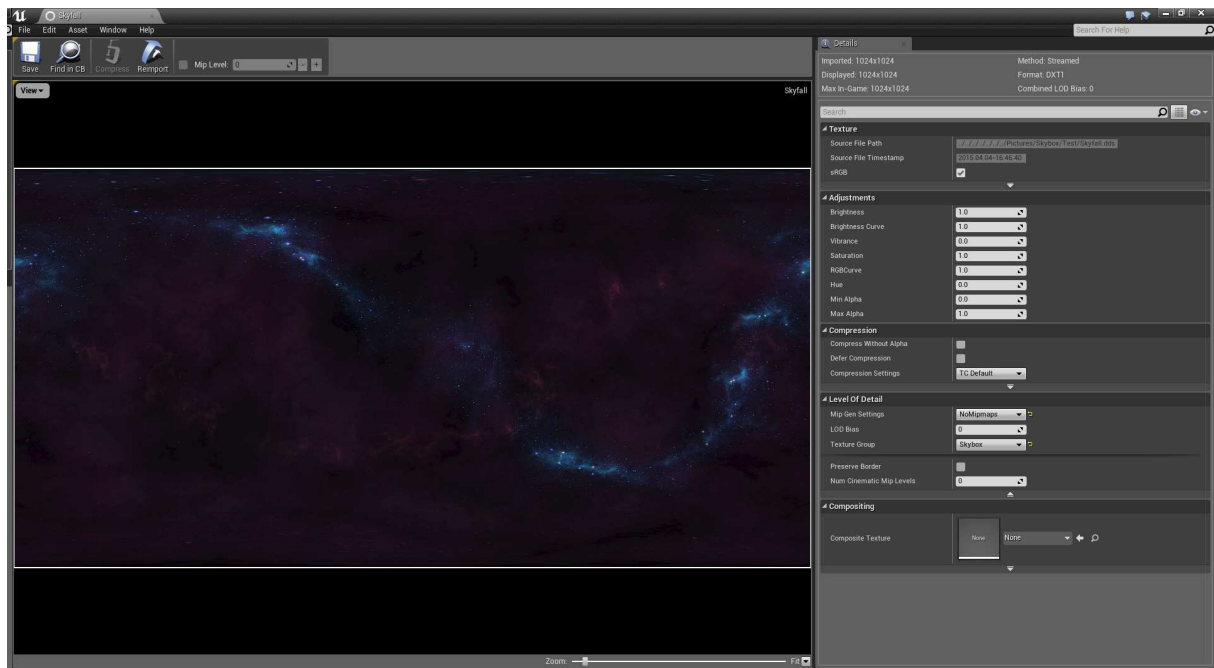


Peale improtimist tuleb muuta tekstuuri sätteid, sest UE4 on vaikimis keernud tekstuuri kvaliteedi madalaks.

UE4-s on kõik tekstuuri ikoonid märgitud punase ribaga. Tekstuuri settinguid saab muuta topelt hiire klõpsuga ikoonile peale vajutades või siis parema hiire klõpsuga menüüst valides edit.



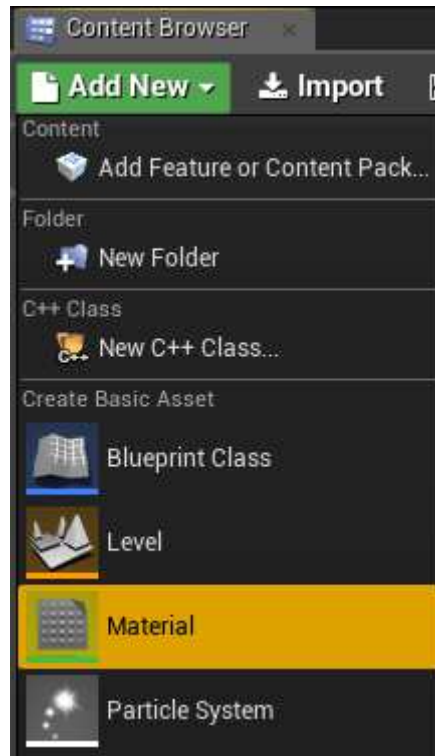
Peale seda avaneb uus aken, kus me saame muuta tekstuuri detaile.



Selleks, et muuta pildi kvaliteet paremaks tuleks "Mip Gen Setting" alt valida "NoMipmaps".

Järgmise etapina tuleks luua uus materjal, et rakendada seda hiljem objektile. Selleks tuleb

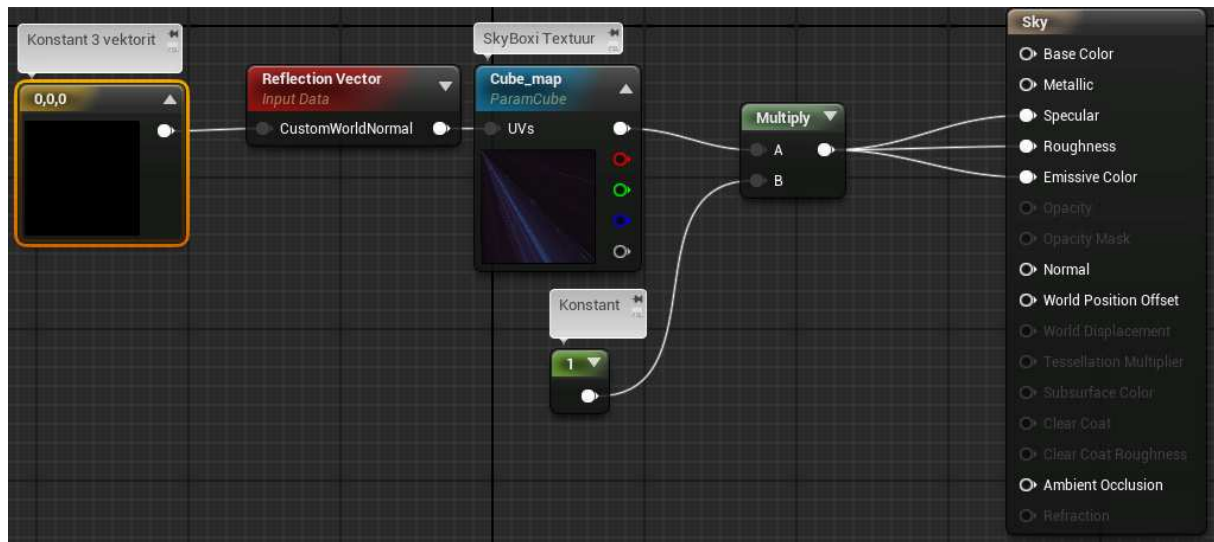
valida "Content Browser" alt "Add New" ja "Material" ning seejärel anda nimi.



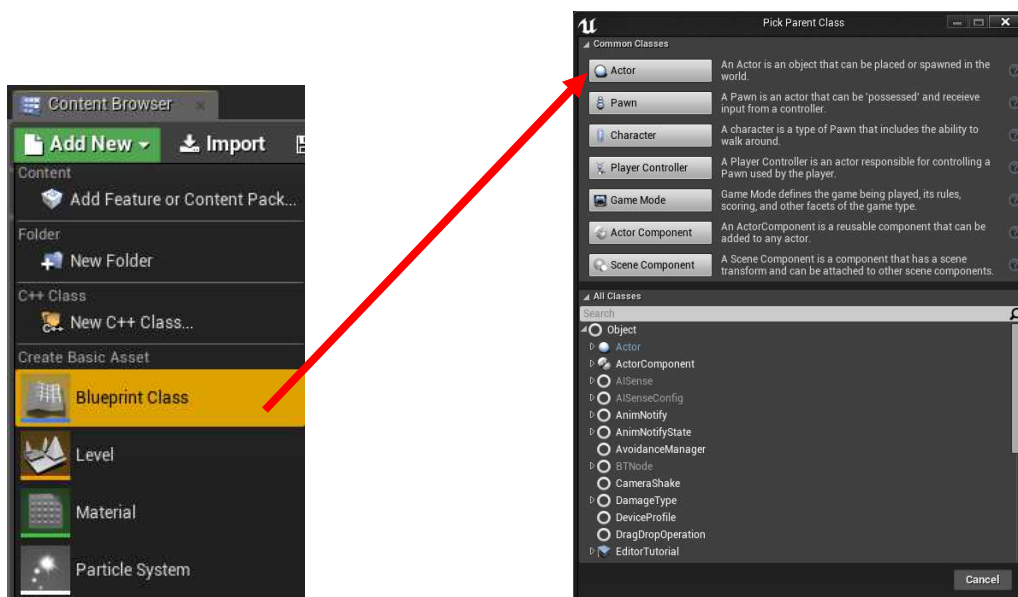
Parema hiire klõpsuga loodud materjali ikooni peale vajutades ja sealt valides "Edit", avaneb visuaalne skriptimise süsteem, kus saab materjali programmeerida. Vaikimisi on antud materjali omadusi kirjeldav sisenditega funktsioon. See funktsioon väljastab lõpptulemuse. Materjali töötlemiseks pakutakse "Palette" menüüs suures koguses erinevaid funktsioone, aga kui klikkida parema hiire klahviga ruudustiku väljale, avaneb menüü koos otsinguga. Selle abil saab kiiremini oma soovitud funktsioonid välja kutsuda.

Skyboxi materjali loomiseks oleks vaja tekstuuri ja nelja funktsiooni rakendada:

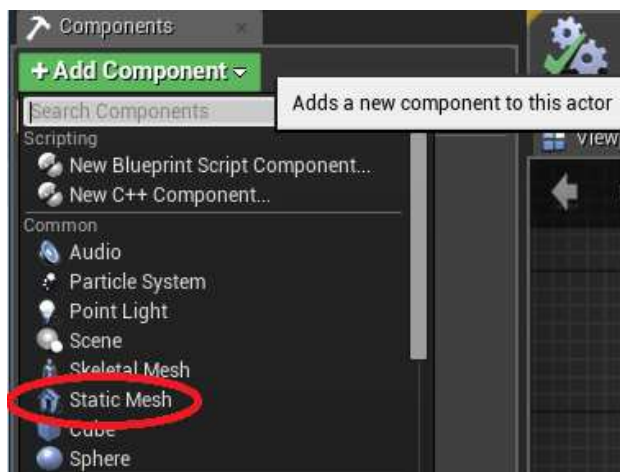
- ReflectionVectorWS
- Constant3Vector
- Constant
- Multiply



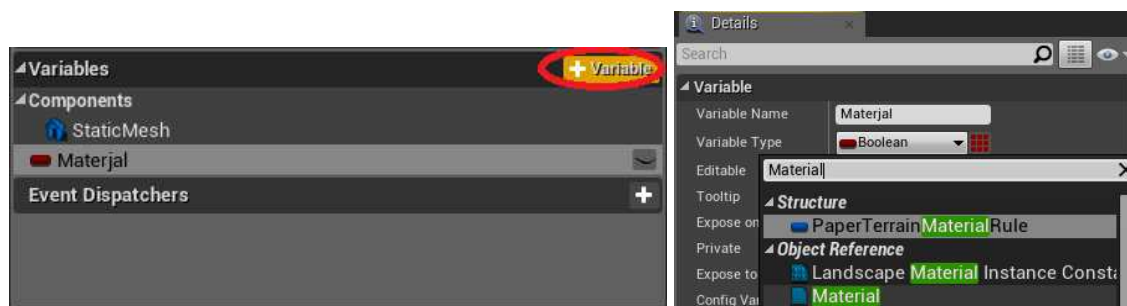
Materjali rakendamiseks on vaja luua uus "Blueprint Actor" klass. Selleks tuleb valida "Add New" ja "Blueprint Class". Ette kuvatakse valikutega menüü, kus tuleks valida "Actor"



"Blueprinti Actor" klassile on vaja lisada kaks komponenti. Selleks tuleb klikkida parema hiire klavviga loodud "Blueprint Actor"-ile peale ja valida edit. Seejärel avaneb uus aken, kus saab loodud Blueprinti visuaalse skriptimise süsteemiga programmeerida. Esmalt on vaja luua SaticSmesh komponent. Selleks tuleb valida "Add Component" ja "Static Mesh".

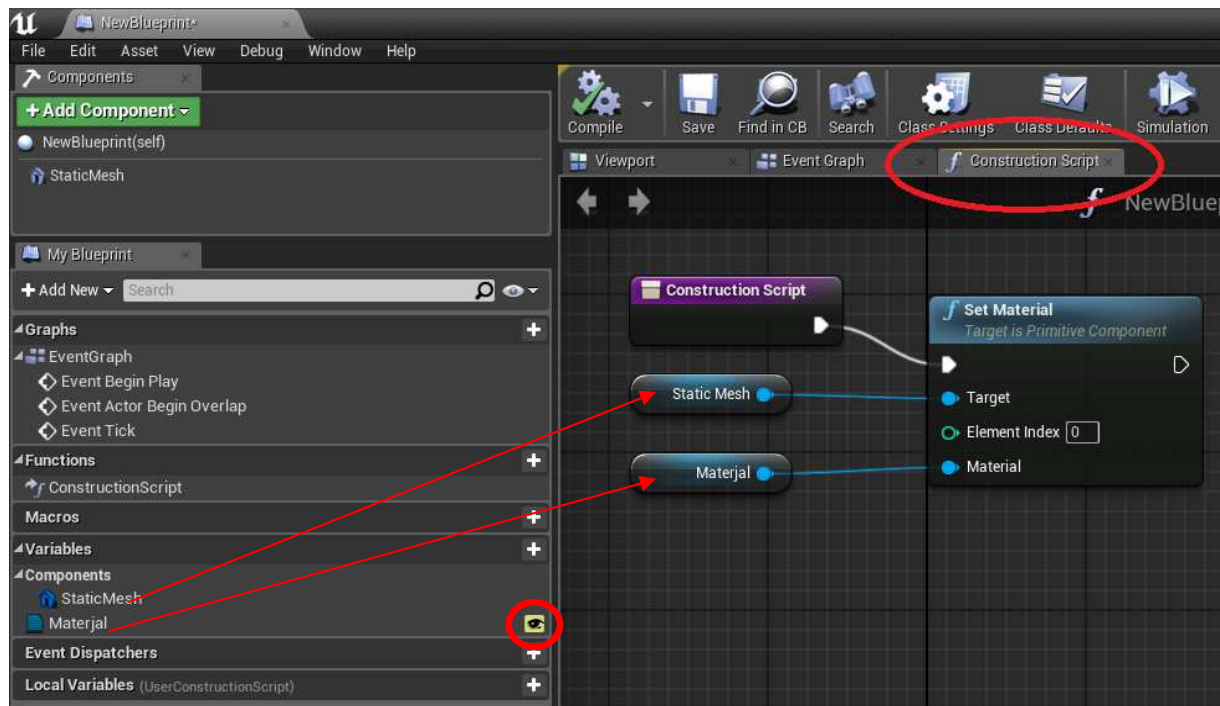


Kui "StaticMesh" on loodud, siis järgmisena peab tekitama uue muutuja, et saaks objektile materjali rakendada. Selleks tuleb "Variables" väljal plus märgile vajutada. Seejärel tuleks anda muutujale nimi ja muuta muutuja tüüp "Details" väljal valides "Variable Tye". Avaneval listi otsinguribale tuleks sisestada "Material" ning valida leitud tulemus listist.

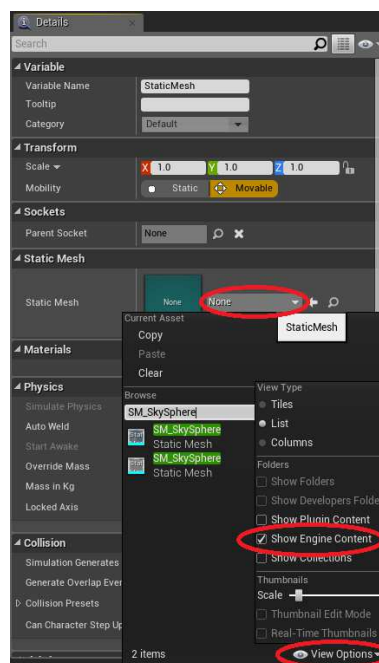


Järgmise sammuna tuleks töötlemise hakata konstruktsiooni skripti funktsiooni. Selleks tuleb valida ülevalt reast "Construction Script" ning seejärel avaneb ruudustikuga väli, kus saab hakata funktsioone välja kutsuma. Esmalt tuleb lohista hiirega ükshaaval kaks loodud komponenti väljale. Materjali puhul küsitakse "Get" või "Set" ja valida tuleb "Get". Seejärel tuleb parema hiire klõpsuga väljale vajutada ja avanevast listist võtta linnuke maha "Context Sensitive" kõrvalt ning sisestada otsingusse "Set Material" ja valida leitud tulemus.

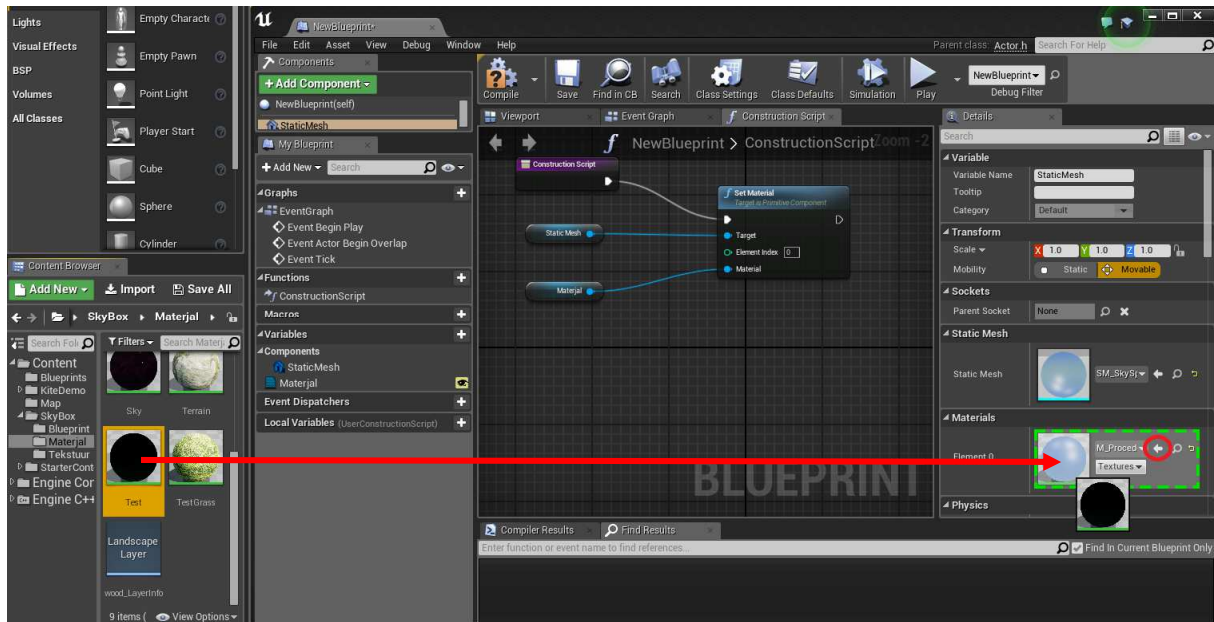
"StaticMesh" tuleb ühendada "Target" sisendisse ja Materjal "Materjal" sisendisse. Materjali ikooni kõrval olev silma ikoon, tuleb teha aktiivseks, et hiljem "Detail" menüüs tekiks materjali määramise koht.



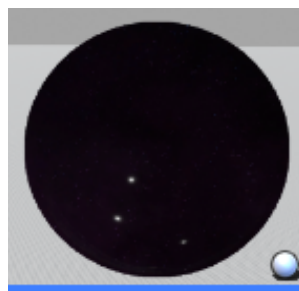
Vahepeal tuleb teha "Compile" ja "Save". Kui need kaks asja on tehtud, siis tuleks järgmise sammuna klikkida "Components" listi all olevat "StaticMesh"-i peale. Kõrval avaneb "Detail" menüü. Seal oleks vaja määrata "Static Mesh", selleks tuleb avada list ja enne otsingusse sisestamist on vaja klikkida "View Options" peale. Seal oleks vaja märkida linnuke "Show Engine Content"-i kõrval. Seejärel tuleb sisestada otsingu reale "SM_SkySphere" ja valida esimene välja toodud tulemus.



Peale seda tekib materjali määramise võimalus. Loodud materjali saab kahte moodi rakendada, kas lohistades või siis teha materjali ikoon aktiivseks ja "Detail" menüüs "Materjal" noolekese peale vajutada.



Kui kogu eelnev protsess on tehtud, tuleb teha uuesti "Compile" ja "Save". Nüüd on valmis "SkyBox Actor", mida saab rakendada leveli disainimisel.



3.3 LandScape loomine

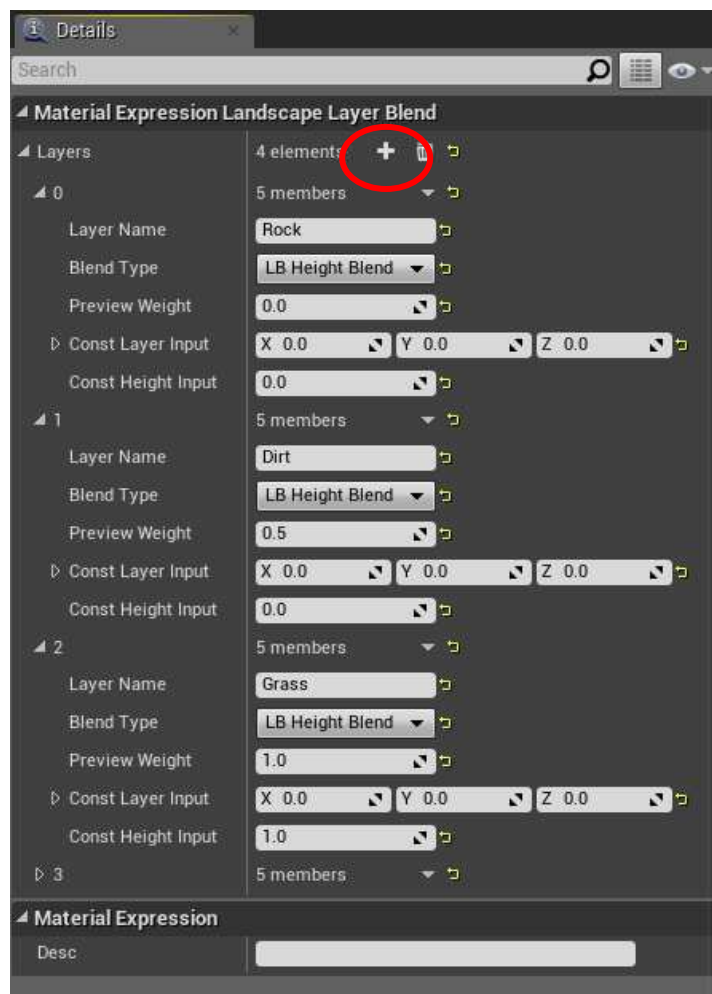
Kui skybox on valmis, oleks vaja pinnast, mille peal saaks kõndida. Selleks oleks soovitatav valida LandScape funktsioon, mille abil saab pinnase tasapinda muuta ja määrata erinevaid materjali kihte.

Plaanis on kasutada nelja erinevat tekstuuri, et luua mitmekihiline pinnas. Esmalt oleks vaja tekitada uus material. Rakendada tuleks kokku kaheksa tekstuuri ja kolme funktsiooni:

- LandscapeCoords
- 8 x Texture Sample
- LandscapeLayerBlend
- Constant

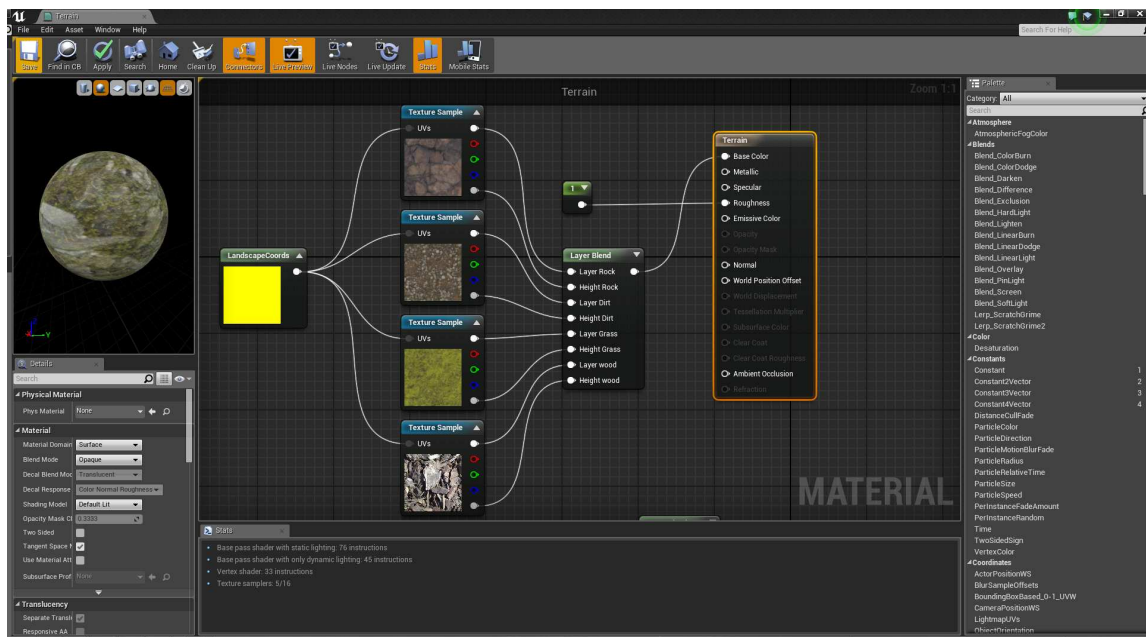
Tekstuure saab UE4 enda pakist. Kui ise on soovi teha tekstuuri, siis võib lihtsalt pildistada päris looduses erinevaid pinnaseid ja pärast töödelda pilti nii, et tekiks võrdsete külgedega ruut ning seejärel importida UE4.

Kui kõik vajalikud funktsioonid on välja kutsutud, siis on vaja "Layer Blend" funktsiooni täiendada. Selleks tuleb teha funktsioon aktiivseks, klikates hiirega selle peale. Detailmenüüs tekib võimalus kihtide arvu määrata, selleks tuleb klikkida plus nuppu peale. Seejärel tuleb määrata kihtidele nimed ja mis sorti kihiga on tegemist valides "Blend Type".

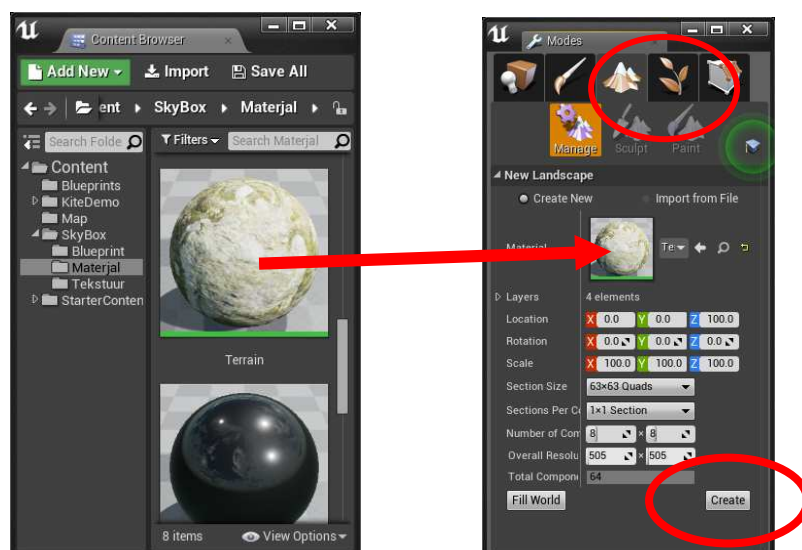


Kui see etapp on tehtud, tekkis mitme sisendiga ja ühe väljundiga "Layer Blend" funktsioon. "LandscapeCoords" funktsioon on ühendatud kõikide tekstuuridega ja need omakorda on ühendatud "Layer Blend" funktsiooniga, mis omakorda on ühendatud "Material" funktsiooniga sisendiga "Base Color". "Constant" mille väärtuseks sai määratud 1, tuli ühendada "Terrain" "Roughness" sisendisse, sellepärast, et pinnas valguse käes läikima ei hakkaks.

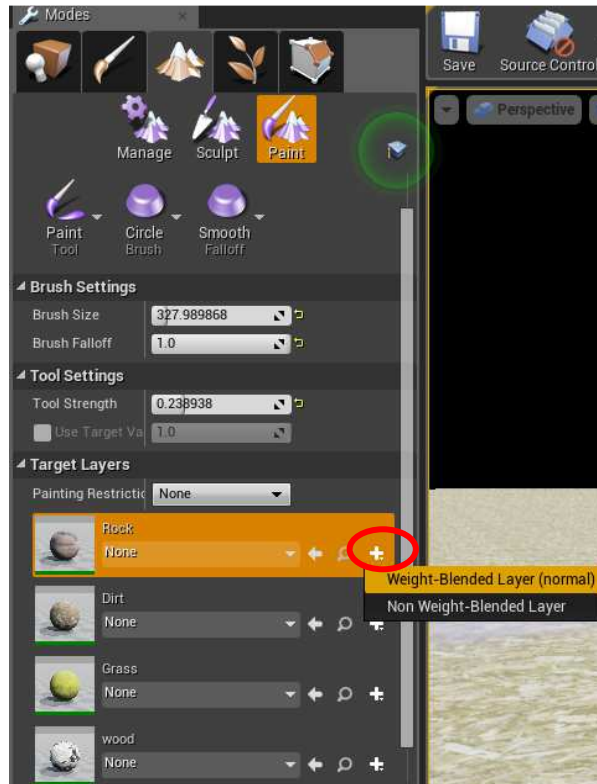
Kui kõik komponendid on ühendatud, siis tuleks järgmise sammuna kõik salvestada.



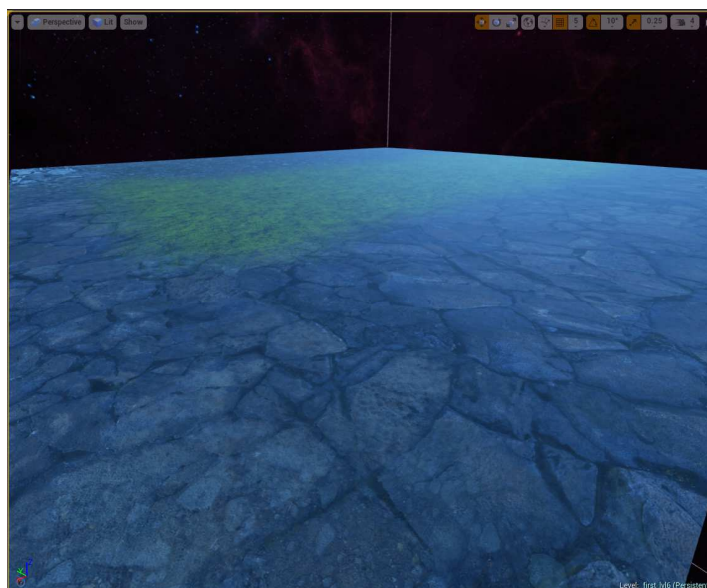
Enne kihtide rakendamist tuleks "Mode" menüüst valida "LandScape". Seejärel sättida materjal ja määrata pinnase tihedus võimalikult suur, et tekstuuri muster ei hakkas korduma. Peale sätete seadistamist tuleb klikkida "Create" nupu peale.



Kui LandScape on loodud, siis menüüs valides Paint, on tekkinud lisa valik kihtide määramiseks, aga enne nende kasutamist tuleks saada infot kihtide jaoks. Selleks tuleb valida "Create Layer Info" vajutades plus ikoonile, siis valida "Weight-Blended Layer" ning määrata asukoht, kuhu info salvestatakse.

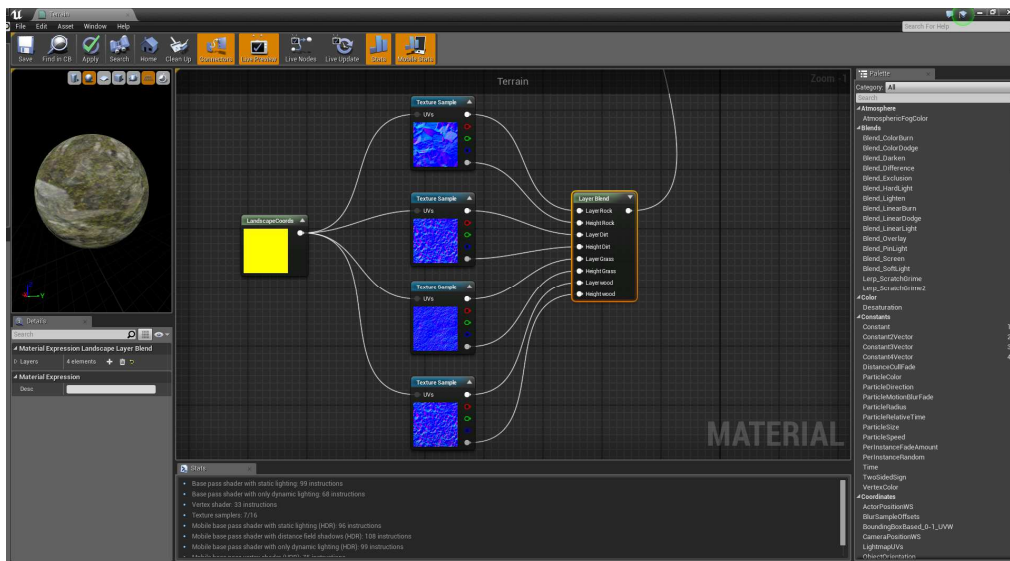


Peale informatsioonikihtide salvestamist saab alustada pinnase töötlemisega. Valides menüüst "Sculpt", on võimalik pinnase tasapinda muuta ja valides "Paint", saab pinnast värvida kasutades erinevaid kihte.

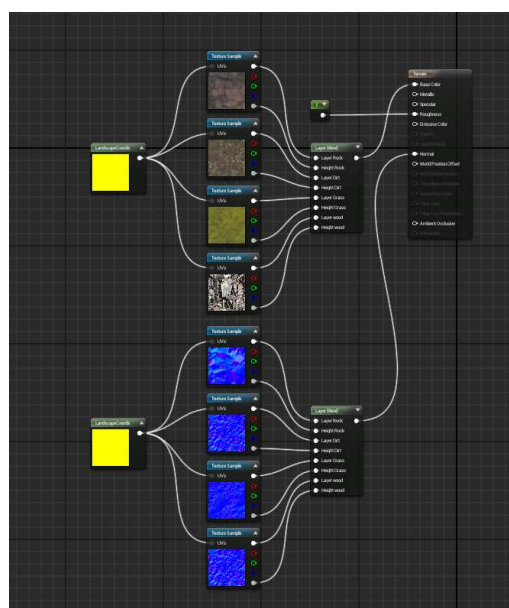


Hetke tulemus on selline, et saab kasutada erinevaid kihte, aga kõik kihid on ühtlaselt siledad. Et muuta kihid eritasandiliseks, oleks vaja kasutusele võtta UV mapping tekstuurid. Need loovad illusiooni, et pinnase materjal oleks ruumiline. Seega tuleks võtta UE4 pakist tekstuuridele vastavad "UV mapping" tekstuurid ja täiendada materjali kasutades visuaalset skripti.

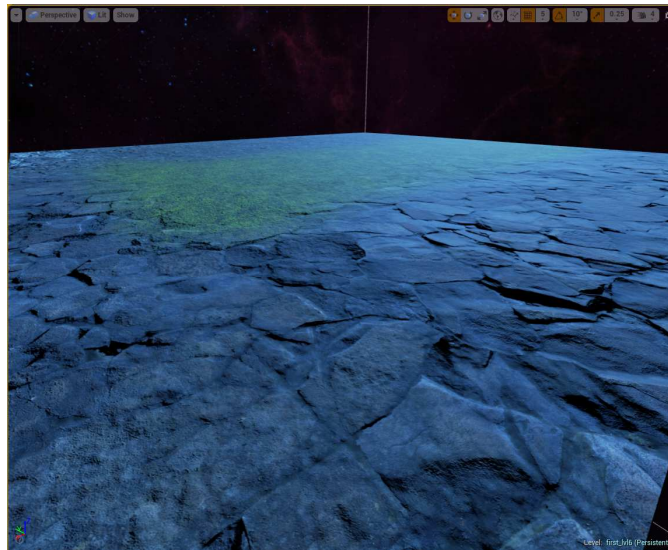
Protsess on täpselt sama, mis eelnevalt kihtide loomisel rakendati.



Mõlemad komponendid tuleb ära ühendada materjal sisendisse, et saada ruumiline efekt kihtide jaoks. See osa, mis koosneb "UV mapping" tekstuuridest, tuleb ühendada "Normal" sisendisse. Seejärel tuleks täiendatud materjal salvestada.



Rakendades UV mappingut pinnasele, tekkis illusioon, et kiht on ruumiline.

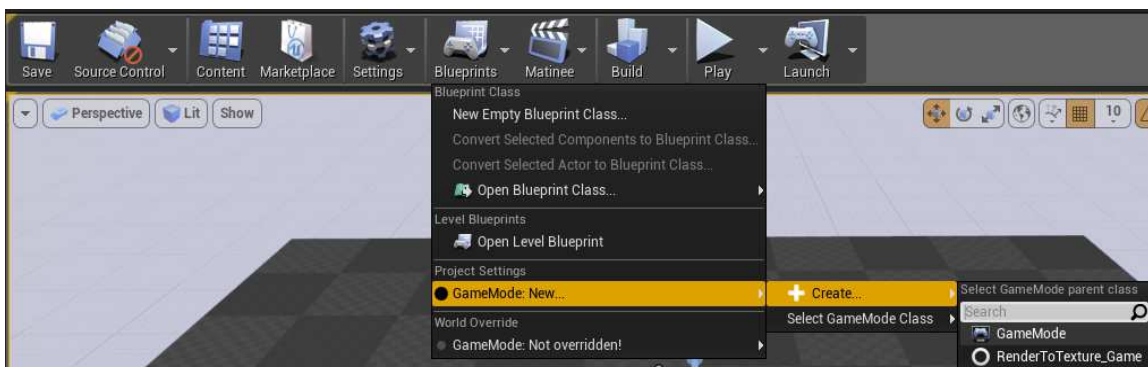


3.4 Playeri loomine

Playeri loomisel peab esmalt selgeks tegema, milline saab olema mängu olek. Kuna eesmärgiks on kasutusele võtta Oculus Rift, tuleks mängu olekuks valida esimese vaate ehk siis "First Person View".

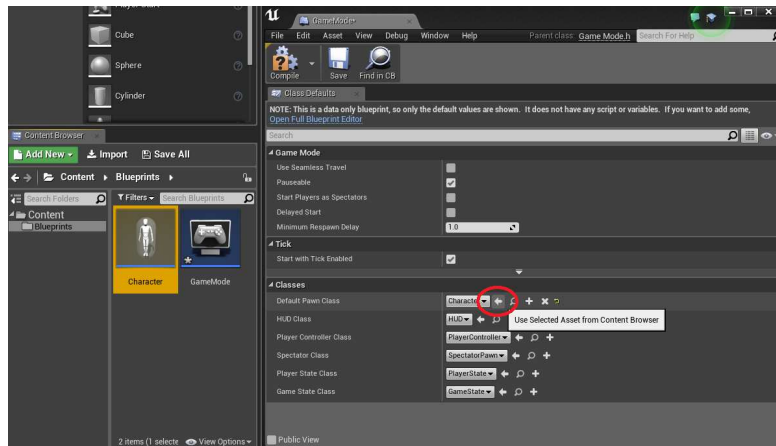
Esmalt on vaja luua Character Blueprint klass. Selleks tuleb teha samad sammud, mis SkyBoxi valmistamisel, kui oli vaja luua "Actor", aga seekord tuleb selle asemel valida "Character".

Järgmise sammuna on vaja luua "GameMode". Selleks tuleb valida ülevalt menüüs Blueprint, projekti sätete alt "GameMode", "Create" ja seejärel valida "GameMode".



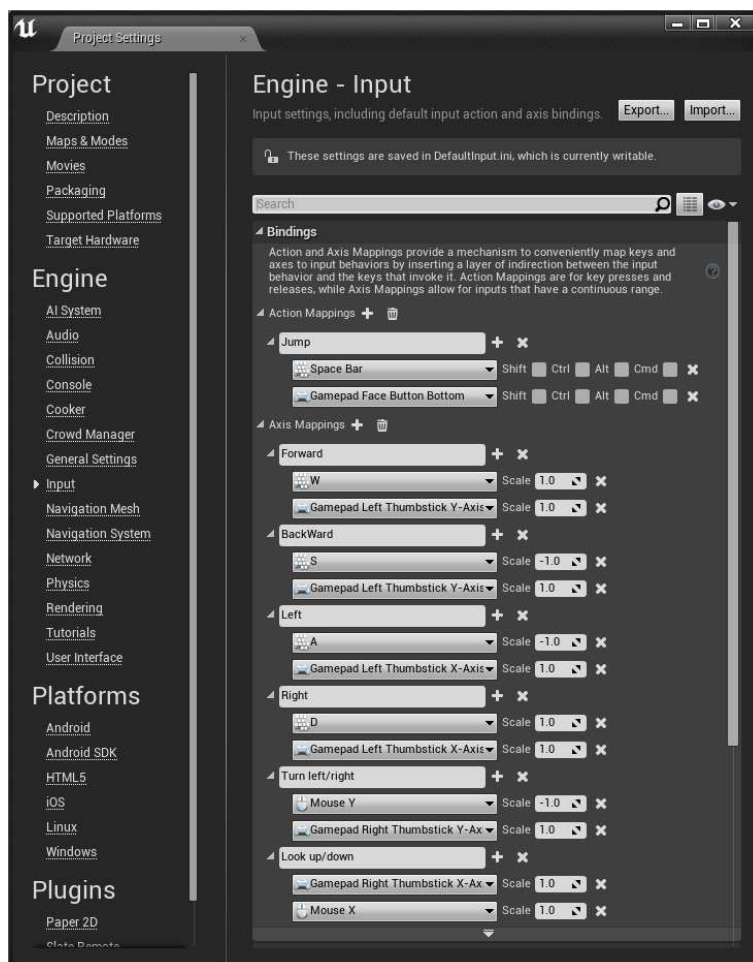
Peale seda protsessi tekkis meil uus folder ja Blueprint klass. Kui vajutada mitu korda hiirega "GameMode" blueprindile, avaneb meile menüü. Seal tuleks määrata "Default Pawn Class"

alt äsja loodud "Character" klass. Seejärel tuleb teha "Character" klass aktiivseks, klikates selle peale.



Kui eelnevad etapid on tehtud, tuleb Input settinguid muuta. Selleks tuleb valida ülevalt menüü reast "Edit" -> "Project Settings" -> "Input".

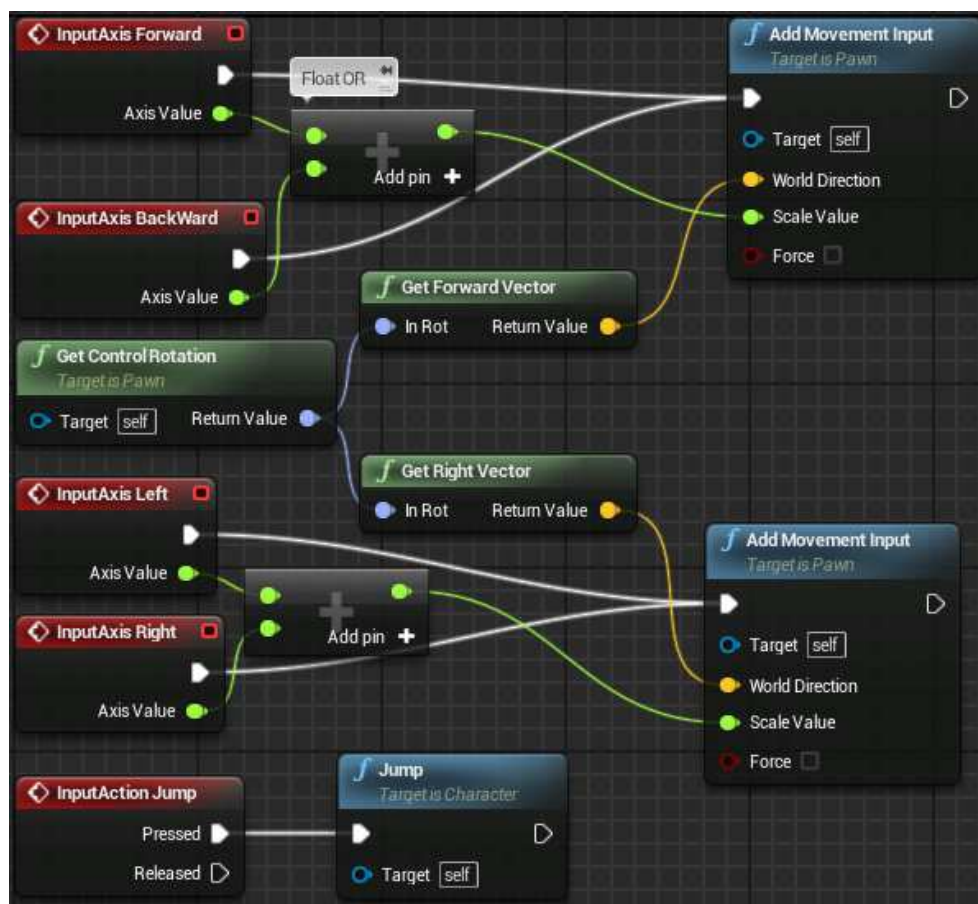
Seal saab määrata ära mis nuppudega kasutaja saaks kontrollida Playerit. Kasutusele tuleks võtta nii klaviatuur kui ka controller, mis on mõeldud konsoolidele.

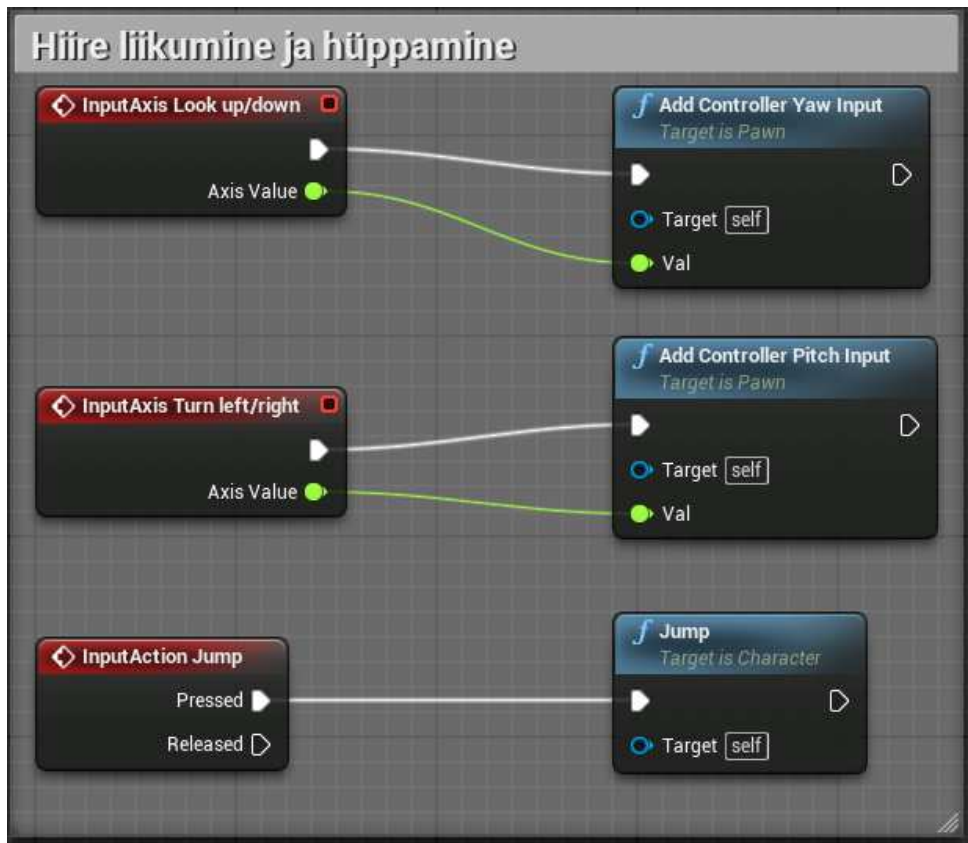


Kui Player kontrolleri sätted on paika pandud, siis järgmise sammuna tuleb minna Player Blueprint klassi. Seal tuleks hakata kokku panema sündmuste jada kasutaja jaoks, näiteks kui ta vajutab klaviatuuril nuppu, liigutab hiirt või kasutab kontrollereid. Sisuliselt antakse kasutajale võimaluse Playerit kontrollida.

Vajutades "Character" blueprindile peale, avaneb uus aken koos "EventGraph". Esmalt on vaja välja otsida klahvi vajutus funktsioonid. Kuna eelnevalt sai "Input" sätted paika pandud siis pakutakse neid funktsioone eelisjärjekorras. Lisaks on vaja veel kümnet funktsiooni.

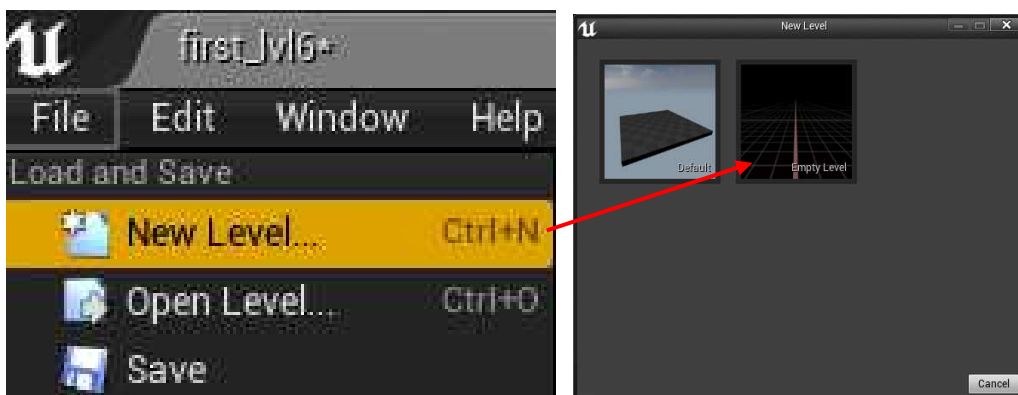
- **2 x Add Movement Input**
- **2 x float OR element**
- **Jump**
- **Get Control Rotation**
- **Get Forward Vector**
- **Get Right Vector**
- **Add Controller Yaw Input**
- **Add Controller Pitch Input**



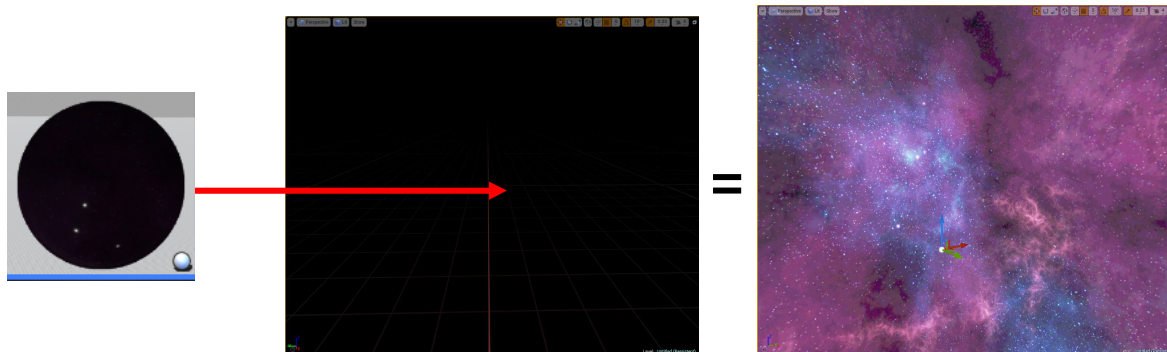


3.5 Objektide paigutamine

Kui "GameMode" on paigas ,siis võib alustada taseme disainimisega. Kuna plaan on luua virtuaalne maailm täiesti nullist, tuleks valida menüüst "File"-i alt "New Level" ning sealt omakorda "Empty Level". Avanev pilt on ruudustik mustal taustal.

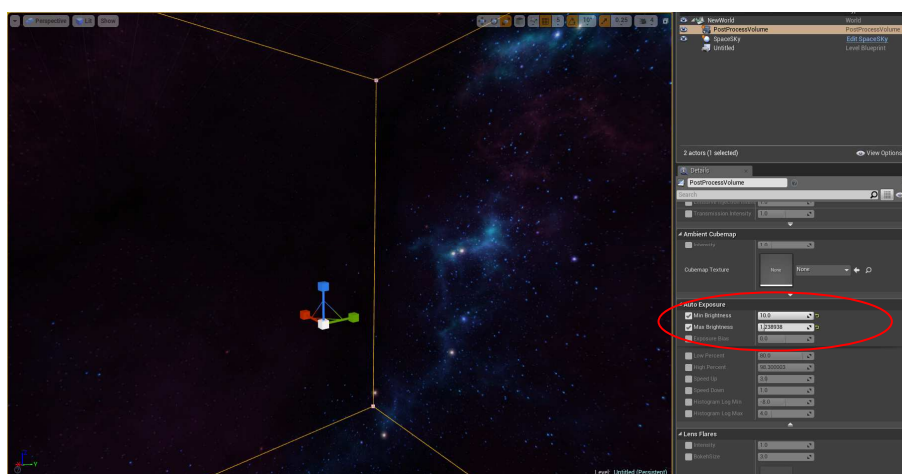


Esmalt sätime paika SkyBoxi. Selleks tuleb meil lohistada Skybox "Actor" tööväljale.



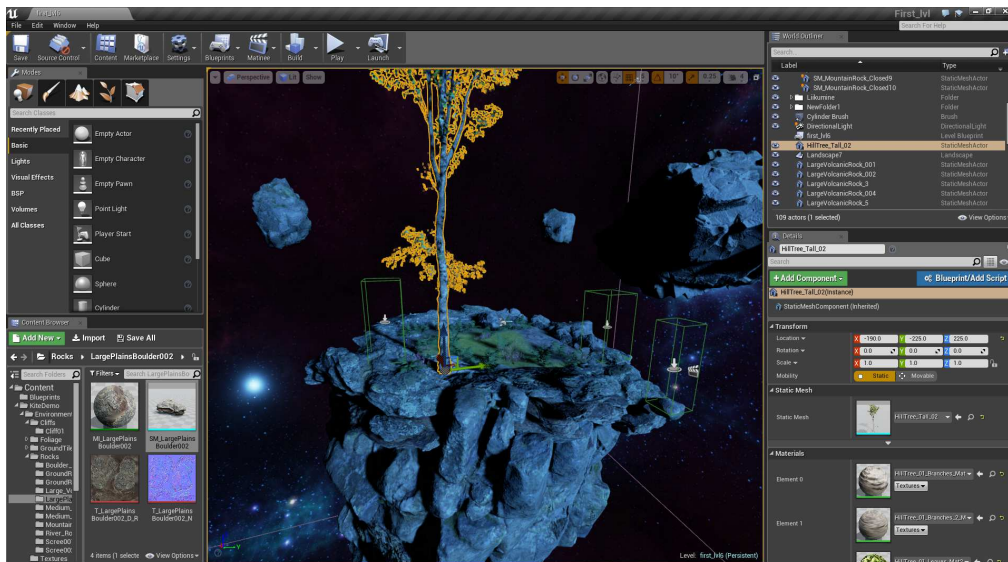
Enne 3D objektide paigutamist tuleks paika panna valgus ja PostProcessVolume, mis on nähtamatu kastike, mille sisse tuleks paigutada objekte.

PostProcess tuleks võtta kasutusele sellepärast, et kontrollida heledust. Ilma selleta tekiks probleem, et Skyboxi heledus kasvaks automaatselt. "PostProcessVolume" objekti leiab "Modes" menüüst "Visual Effects" listi alt. "Details" menüü alt on vaja muuta AutoExposure sätete alt Min Brightness ja Max Brightness.

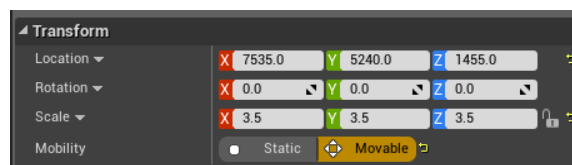
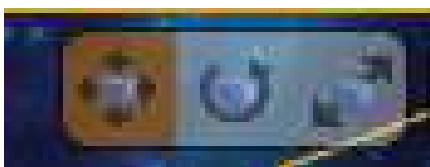


Valguseks oleks soovitatav valida DirectionalLight'i, mille saab Modes menüüst Lights listi alt. Valguse värvi ja tugevust saab "Details" menüü alt sättida.

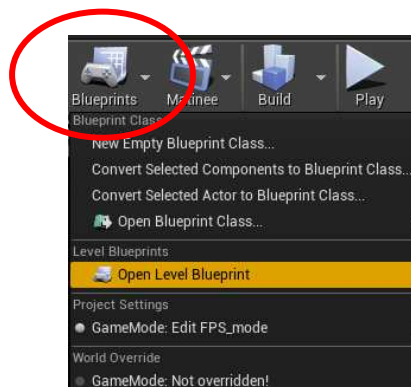
Kui need etapid on paigas, võib alustada 3D objektide rakendamist taseme loomiseks. Objekte saab lisada väga lihtsalt, selleks tuleb Content Browseri alt lohistada Actor mängu väljale.



3D-objekti suurust, asendit ja asukohta saab määrata kahte moodi. Üks võimalus on valida see ülevalt menüüribalt ja teine võimalus on sisestades numbreid Details väljal.



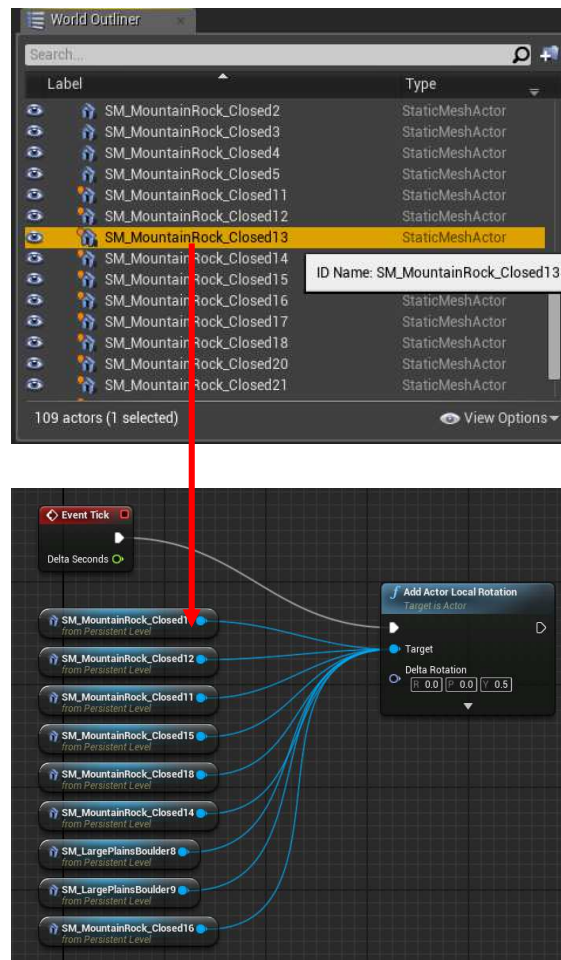
Et virtuaalmaailm oleks natukene elavam, võiks mõned objektid liikuma panna, selleks tuleb avada "Level Blueprint" klass. Selle leiab ülevalt menüüst valides "Blueprints" ja "Open Level Blueprint".



Et 3D-objektid liiguksid stabiilselt terve mängu tööloleku aja, läheks vaja objekti ja kahte funktsiooni:

- Event Tick
- Add Actor Local Rotation
- Objekt

Selleks, et rakendada objekte Blueprint keskkonnas, tuleb "World Outliner" listist valida objekt ja lohistada see hiirega töölauale.

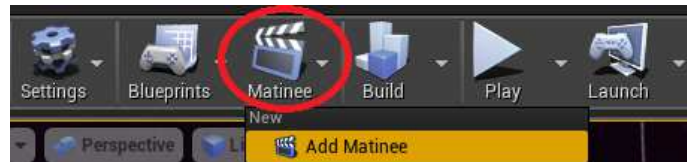


Kui "Level Blueprint" klass ära salvestada ja seejärel simuleerida valides ülevalt menüüst "Play" alt "Simulate", näeme, et objekt tiirleb ümber oma telje

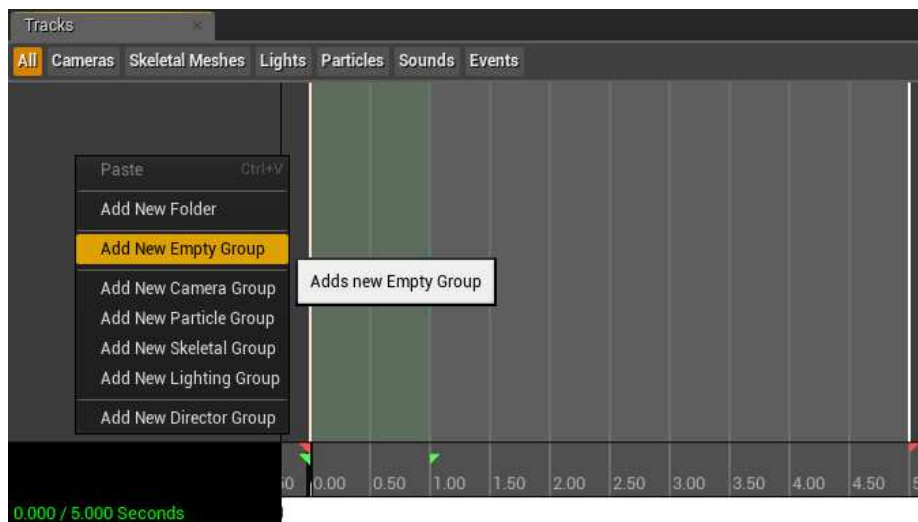
3.6 Matinee rakendamine

Matinee rakendusega on võimalik luua animatsiooni. Sellega on võimalik üles ehitada sündmustik, mis kindla ajavahemiku jooksul peab toimuma.

Projektis on plaan tekitada olukord, kui kasutaja asetseb kindlase punktis, hakkavad teatud sündmused toimuma. Selleks tuleb ülevalt menüüst valida "Matinee" ja "Add Matinee", seejärel avaneb uus aken, kus me saame hakata looma animatsiooni.



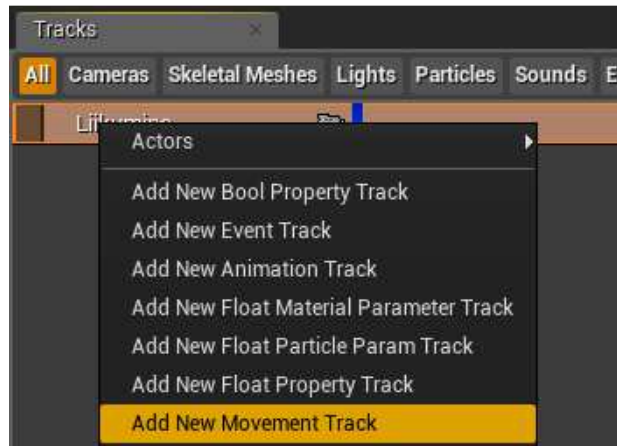
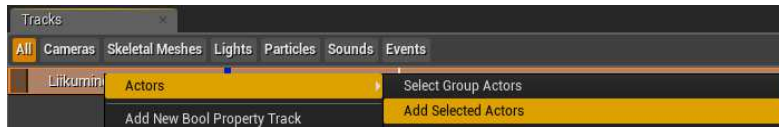
Esmalt tuleks luua uus grupp, selleks tuleks vajutada paremat hiire klahvi ja valida "Add New Empty Group" ning anda sellele nimi.



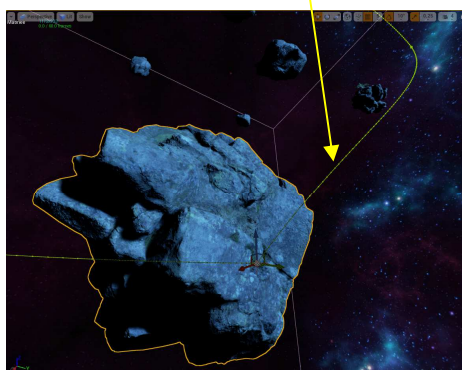
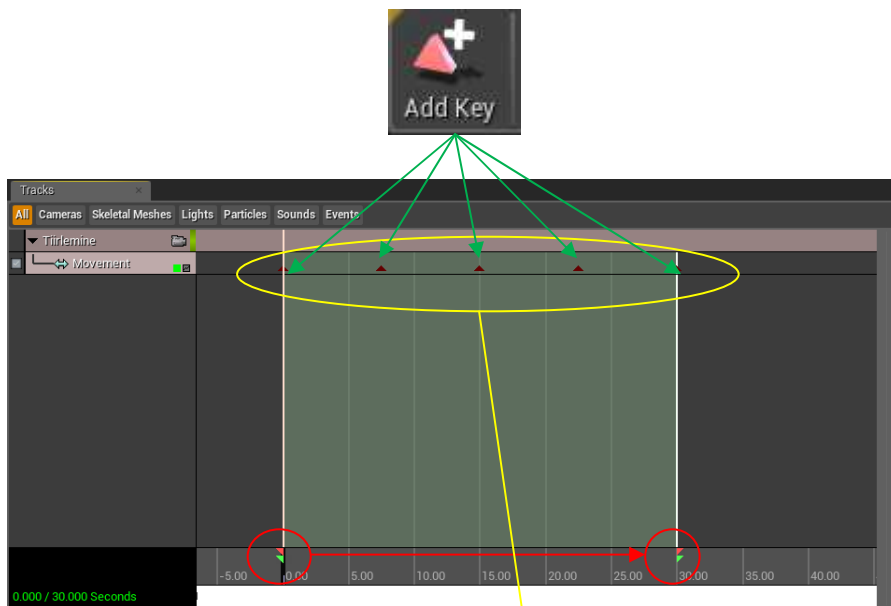
Seejärel tuleb suunduda mänguväljale ja valida klikates 3D-objektile, mida me tahame animeerida.



Kui objekt on valitud, siis tuleb tagasi minna Matinee editori ja seal klikata parema hiire klahviga loodud grupi peale ja valida Actor ->Add Selected Actor. Kui Actor on valitud, siis tuleb uuesti parema hiire klahviga valida loodud grupp ning seejärel valida Add New Movement Track.



Punase kolmnurgaga saab paika panna, kui kaua see sündmus kestab ja rohelisega saab näidata, mis selle sündmuse hetkel toimub. Add Key nupu abil saab määrata objekti trajektoori, mis ajahetkel ta kuskil peab asuma.

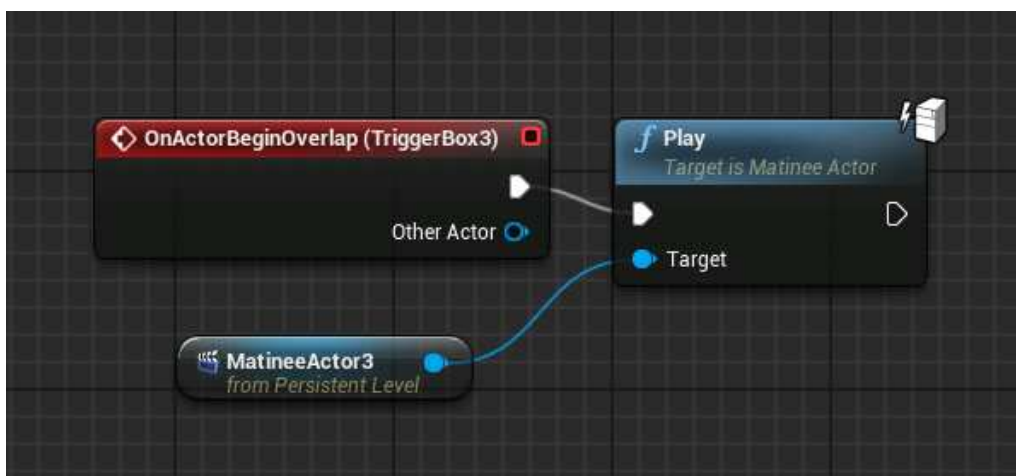


Eesmärk on programmeerida nii, et animatsioon hakkab tööle siis, kui player on mingis kindlas punktis. Selleks võib võtta kasutusele näiteks "Box Trigger"-i, mille leiab "Modes" menüüst "Basic" alt. Box Trigger on nähtamatu kastike, mis täidab lülitifunktsiooni.



Kui animatsiooni sündmused on paika saadud, siis tuleb minna Level Blueprinti ning seadistada ära "Box Trigger" funktsioon. Selleks on vaja välja kutsuda kaks funktsiooni ja "MatineeActor" objekt.

- OnActorBeginOverlap
- Play
- MatineeActor



Kui see on valmis, peaks lõpptulemus olema selline, et kui siseneda "Box Trigger" alasse, siis hakkab animatsioon tööle.

3.7 Oculusrifi rakendamine

Enne Oculuse rakendamist, tuleks prillid loodud virtuaalmaailma jaoks ära seadistada. Kõige pealt on vaja muuta ära DefaultEngine.ini fail. Selle leiab projekti folderist Config folderi alt. Failile tuleb lisada selline koodijupp:

```
[Oculus.Settings]
```

```
; In meters, not Unreal Units (cm)
```

```
; Recommended for true first person viewpoint
```

```
; since you are simulating the head.
```

```
;HeadModel_v2=(X=0.0,Y=0,Z=0.0)
```

```
; Default values
```

```
HeadModel_v2=(X=0.12,Y=0,Z=0.17)
```

```
[/Script/EngineSettings.GameMapsSettings]
```

```
EditorStartupMap=/Game/Map/first_lv16
```

```
LocalMapOptions=
```

```
TransitionMap=
```

```
bUseSplitscreen=True
```

```
TwoPlayerSplitscreenLayout=Horizontal
```

```
ThreePlayerSplitscreenLayout=FavorTop
```

```
GameInstanceClass=/Script/Engine.GameInstance
```

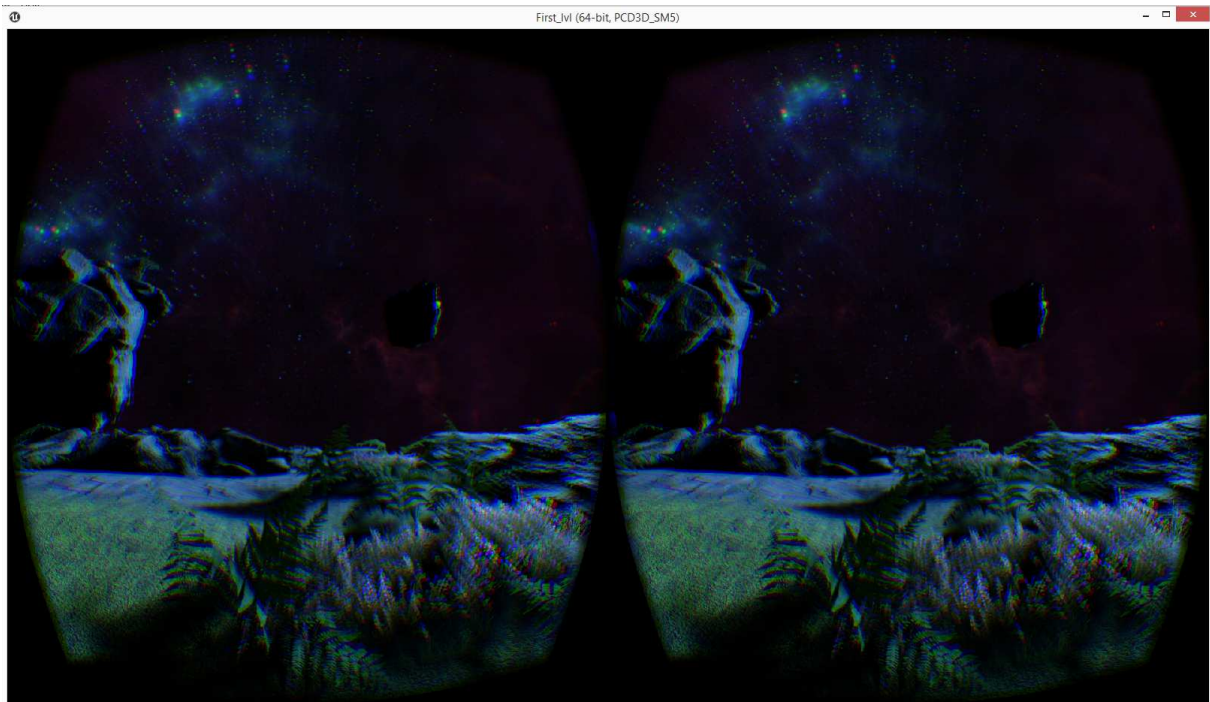
```
GameDefaultMap=/Game/Map/first_lv16
```

```
ServerDefaultMap=/Engine/Maps/Entry
```

```
GlobalDefaultGameMode=/Game/Blueprints/FPS_mode.FPS_mode_C
```

GlobalDefaultServerGameMode=None

Kui see on tehtud, tuleb teha projekte "Build", seejärel valida ülevalt menüüst "File", "Package Project" ja määrata, mis platvormile maailm kokku pakitakse. Järgmisena tuleb minna folderisse, kuhu failid tekitati ja seejärel käivitada sealt mäng. Et Oculus Rift aktiveeruks, tuleks vajutada "Alt+Enter".



Kokkuvõte

Bakalaureusetöö lõpptulemusena valmis virtuaalprillide Oculus Rifti jaoks virtuaalne maailm,. Selle töö valmimise eelduseks oli vaja põhjalikult tundma õppida mängumootorit Unreal Engine 4 ja tema kasutusvõimalusi. Töö käigus sai selgeks, et Unreal Engine 4 Editor on mõeldud ainult virtuaalse maailma disanimiseks, tal puudub täielikult modeleerimise rakendus. Soovitatav modelleerimise tarkvara, mida toetab ka Unreal Engine 4 on Blender.

Unreal Engine 4 on väga võimas tarkvara ja see pakub väga palju erinevaid võimalusi virtuaalmaailma ülesehitamiseks. Mootor on küll rohkem mõeldud mängude loomiseks, aga tegelikult võiks seda rakendada ka teistes valdkondades, näiteks animafilmide tegemisel või simulaatorite ülesehitamiseks.

Kasutatud kirjandus

- [1] <https://docs.unrealengine.com/latest/INT/Engine/Rendering/Overview/index.html>
- [2] <https://docs.unrealengine.com/latest/INT/Engine/Rendering/Materials/index.html>
- [3] <https://docs.unrealengine.com/latest/INT/Engine/UMG/index.html>
- [4] <https://docs.unrealengine.com/latest/INT/Engine/Animation/Overview/index.html>
- [5] <https://docs.unrealengine.com/latest/INT/Engine/Audio/index.html>
- [6] <https://docs.unrealengine.com/latest/INT/Engine/Physics/index.html>
- [7] <https://docs.unrealengine.com/latest/INT/GettingStarted/Terminology/index.html>
- [8] http://http.download.nvidia.com/developer/presentations/2004/6800_Leagues/6800_Leagues_Deferred_Shading.pdf
- [9] <http://kayru.org/articles/dssdo/>
- [10] http://web.archive.org/web/20090306111435/http://www710.univ-lyon1.fr/~jciehl/Public/educ/GAMA/2007/Deferred_Shading_Tutorial_SBGAMES2005.pdf
- [12] http://www.gamasutra.com/view/feature/2107/realtime_glow.php
- [13] <http://resources.mpi-inf.mpg.de/hdr/datmo/mantiuk08datm.pdf>
- [14] <http://www.cs.nott.ac.uk/~qiu/webpages/Papers/icpr2006-hdri-camera.pdf>
- [15] <http://www.realworldtech.com/physx87/>
- [16] Borshukov, G; Lewis, J. P. 2005. <http://www.scribblethink.org/Work/Pdfs/Face-s2003.pdf>