TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Andre Ahuna

Loodusmaastiku genereerimine liitreaalsuses

Bakalaureusetöö (9 EAP)

Juhendaja: Madis Vasser, PhD

Tartu 2021

Loodusmaastiku genereerimine liitreaalsuses

Lühikokkuvõte:

Käesolev bakalaureusetöö kirjeldab nutiseadme liitreaalsusrakenduse arendamist ja analüüsi. Loodud rakendus võimaldab nutiseadmega skaneerida keskkonda ning genereerida skaneeritud alale kasutajat ümbritsevaid objekte arvesse võtva loodusmaastiku. Kirjatöös tutvustatakse kasutatud tehnoloogiaid ning põhjendatakse nende valikut, selgitatakse süsteemi arhitektuuri ja info samm-sammulist töötlemist ning viimaks antakse hinnang rakenduse jõudlusele ja ettepanekuid edasisteks arendusteks.

Võtmesõnad:

Liitreaalsus, LiDAR, iOS, protseduuriline genereerimine, nutiseade, Unity, mobiilirakendus, maastik, maastikuloome, võrestik.

CERCS:

P175 – Informaatika, süsteemiteooria.

Generating Natural Landscape in Augmented Reality

Abstract:

This thesis describes the development and analysis of a smart-device-based augmented reality application. The created app allows user's environment to be scanned and used for the generation of a landscape that is affected by any objects in the scanned area. The paper introduces tools used in the process and justifies choosing them, explains the system architecture and the step-by-step info processing, and finally evaluates the application's performance and offers future improvements.

Keywords:

Augmented reality, LiDAR, iOS, procedural generation, smart device, Unity, mobile application, terrain, terrain generation, mesh.

CERCS:

P175 – Informatics, system theory.

Sisukord

1. Sissejuhatus

Tänapäeval on pea igal inimesel nutitelefon või tahvelarvuti. Taolise universaalsuse tõttu on enim kasutatud liitreaalsusrakendused orienteeritud just neile. Alternatiivsed liitreaalsuse (ingl *augmented reality*, edaspidi AR) kogemise meetodid - näiteks nutiprillid HoloLens ja Magic Leap - pole peavoolus veel erilist populaarsust kogunud. Seetõttu on liitreaalsust võimaldavad teegid olnud sunnitud rakendama nutiseadmetele tüüpilisi 2D pilte salvestavaid kaameraid, mille kasutamine liitreaalsuse tarbeks on nõudnud ebaoptimaalseid lahendusi [1]. See on valdkonna arengut pidurdanud.

2020. aasta märtsis avalikustas ettevõte Apple uue generatsiooni iPhone'i ja iPadi kaamerakorpustesse lisatud LiDAR skanneri, mille integreerimine võimaldab - nende sõnadega – „täiesti uuel tasemel AR elamusi“ [2]. Tartu Ülikooli arvutigraafika ja virtuaalreaalsuse laborisse soetatud iPad Pro lubas seda võimekust järele proovida. Loodud liitreaalsusrakenduse teemavalik sündis soovist katsetada LiDAR-i võimaldatud sügavuskaarti ning autori huvist protseduurilise maailmaloome vastu.

Lõputöö eesmärgiks oli luua rakendus LiDAR skanneriga iOS seadmele, mis suudaks luua mistahes keskkonnas konteksti arvestava loodusmaastiku. Ehk: (1) võimaldada LiDAR skannerit kasutades ümbritseva kaardistamine; (2) selle põhjal genereerida mõistliku aja jooksul maastik, mis kataks realistlikku väljanägemist säilitades skanneritud ala; (3) loodut kasutajale liitreaalsuses kujutada.

Käesolevale bakalaureusetööle leidis autor ühe sarnase varasema projekti - Renan Dembogurski jt 2012. aasta paber [3] interaktiivse maastiku genereerimisest liitreaalsustähiste põhjal. Nende lõpptulemus lõi aga konstantse suurusega maapinna, mis ei võtnud arvesse kaamerat ümbritsevaid objekte. Tõenäoliselt poleks see 2012. aasta nutiseadmete AR võimekusega ka võimalik olnud. Üldiselt rakendab nende programm liitreaalsuse võimalusi vähe ning ligi kümme aastat parem AR tehnoloogia lubab luua esinduslikuma programmi.

Järgmine peatükk tutvustab töös esinevaid mõisteid ja termineid. Kui tekstis kasutatud erialane termin jääb arusaamatuks, võib sealt leida definitsiooni. Kolmas peatükk tutvustab kasutatud tehnoloogiaid ning nende eeliseid alternatiivide ees antud töö teenistuses. Neljas peatükk süüvib programmi arendusse, kirjeldades nutiseadme kaamerasse saabunud info samm-sammulist töötlemist kasutajale esitatavaks tulemuseks ning avastatud probleemkohti. Viies peatükk kirjeldab teste ning võimalusi edasisteks arendusteks.

2. Mõisted ja terminid

LiDAR (ingl *light detection and ranging*) – seade, mis mõõdab objektide kaugust valgusimpulsi tagasipeegeldumise aja kaudu [4].

Liitreaalsus (ingl *augmented reality*) – interaktiivne keskkond, milles läbi meediumi liidetakse reaalsele pildile digitaalne lisateave [5].

Mürakaart (ingl *noise map*) – n-dimensiooniline kogum arvudest kindlas vahemikus (tüüpiliselt 0 kuni 1 või -1 kuni 1). Näiteks valge mürakaart on kogum, kus iga punkti väärtus on suvaline arv etteantud vahemikust.

Sügavuskaart (ingl *depth map*) – 2D kaart, kus iga punkt kirjeldab jäädvustatud objekti kaugust vaatlejast.

Teek (ingl *library*) – korduvaks kasutamiseks määratud kollektsioon infoobjektidest (nagu klassid, dokumentatsioon, funktsioonid) [6].

Võrestik (ingl *mesh*) – kollektsioon punktidest ja servadest, mida arvutigraafikas kasutatakse pindade jäljendamiseks. Võrestike punktid grupeeritakse tüüpiliselt servadega ühendatud kolmnurkadesse, mille pinda (ingl *face*) seejärel kuvatakse.

3. Tehnoloogiad

Soov rakendust arendades kasutada liitreaalsusvaldkonna kõige uuemaid arenguid LiDAR skanneri näol limiteeris tunduvalt võimalike tehnoloogiate valikuid. Kasutades vastavat tuge mitteomavat tööriista, oleks loodav programm olnud suutmatu ära kasutama riistvara poolt pakutud võimekust. Seetõttu oli vajalik LiDAR skanneri funktsionaalsust võimaldava Apple'i liitreaalsusteegi ARKit 4 tugi. Kuna 2021 kevadel oli antud teek alla aasta vana, siis leidus seda toetavaid rakendusi vähe. See muutis arenduskeskkonna ja pistikprogrammide valiku suhteliselt konkreetseks.

3.1 LiDAR

Nutiseadetel liitreaalsust pakkuvad teegid on ajalooliselt olnud sunnitud opereeruma seadme kaamera poolt varustatud informatsioonil [5]. See on taganud AR rakenduste leviku, kuid mõneti ka nende tagurlikkuse. Nimelt: kaamerasilm loob nähtust 3D keskkonnast inimsilmale mõistetava 2D pildi, mida AR teegid peavad tõlkima tagasi 3D kaardiks[1]. Seda saavutatakse peamiselt pildile jäänud kontrastsete punktide (nn ankrute) omavahelisel võrdlemisel. See lähenemine on aga arvutuslikult kallis ning jääb hätta vähekontrastsete pindadega (nt ühevärvilised seinad). [5]

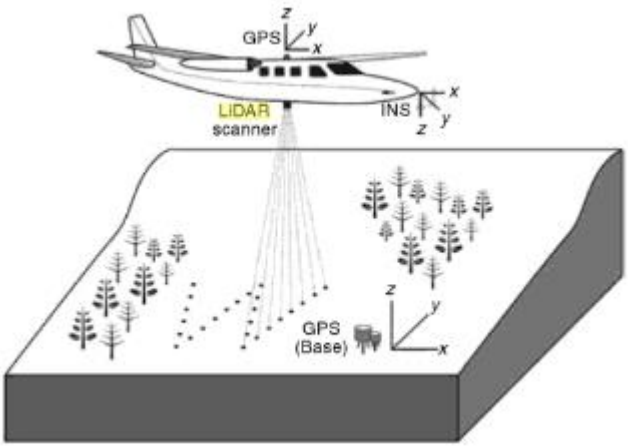
LiDAR (*Light Detection and Ranging*) on laserskaneerimisseade, mis kaardistab väljastatud laserimpulsi peegeldumist kasutades objektide asukohti [4]. Selle idee on lihtne: punkti kaugust allikast saab laseri peegeldumisel välja arvutada valemiga

$$\text{kaugus} = (\text{valguse kiirus} * \text{peegeldumisele kulunud aeg}) / 2,$$

misjärel teades seadme asukohta ning laserkiire suunda, on võimalik välja arvutada punkti koordinaadid [4].

[1] Erandiks on siinkohal markeri-põhine AR, mis keskkonna kaardistamise asemel otsib sisendist kokkuleppelisi märke (nt QR-kood).

Joonis 1 illustreerib LiDAR-i tüüpilist rakendust maapinna kaardistamisel. Lisaks on seade kasutust leidnud üha enam ka isejuhtivate sõidukite pardal, kuid 2020 jõudis see liitreaalsuskogemuse parandamise eesmärgil esmakordselt ka nutiseadmetesse iPad Pro 2020 ja iPhone Pro 2020 kujul [7]. Spetsiaalselt AR kogemuse parandamise eesmärgil kaamerakorpusesse lisatud skanner võimaldab tuvastada kuni viie meetri kaugusel olevaid objekte varasemast suurema täpsusega [7].



Joonis 1. Maapinna kaardistamine LiDAR skanneriga lennukist [4].

Kuna käesolev lõputöö eeldab kasutajalt ümbritseva keskkonna kaardistamist, on ajalooliselt just sellel eesmärgil loodud tehnoloogia siinkohal teoreetiliselt suurepärane.

3.2 Unity

Unity on üks maailma populaarsemaid mängude ja rakenduste loomevahendeid [8]. Tänu suurele kasutajas- ja loojaskonnale toetab see väga erinevaid platvorme ning omab elavat foorumit ja laialdaselt dokumentatsiooni. Samuti on selle kasutamine tasuta kui loodu ei teeni aastas üle 100 000 dollari [9].

Unity liitreaalsusrakenduste tugi on üks eesrindlikumaid, mitte ainult võimaldades uusimate AR teekide kasutamist, vaid ka integreerides need oma multiplatvormilisse liidesesse AR Foundation. Käesolevas töös kasutamise kasuks rääkis ka selle tugev võrestikuloome süsteem, liitreaalsusrakenduse loomise abistamiseks koostatud koodinäidised[2] ning autori varasem kogemus selle programmi käsitlemisel.

Unity ainsad tõsiseltvõetavad alternatiivid olid Unreal Engine ja Vuforia Engine. Unreal Engine omab sarnaselt Unityle piisavalt suurt kasutajabaasi, et pakkuda mängumootorina laialdast funktsionaalsust. Üldjuhul on nende kahe vahe üpris väike ning enamasti taandub inimeste eelistus sellele, kumma nad mugavama leiavad. Töö kirjutamise hetkel Unreal Engine aga veel LiDAR-i kasutamiseks vajalikku ARKit 4 ei toeta ning nende teekaardilt (ingl

[2] https://github.com/Unity-Technologies/arfoundation-samples

roadmap) selle plaani ka lähitulevikus ei paista[3]. Vuforia Engine teisalt on varasematest erinev – tegu on spetsiifiliselt liitreaalsuskogemuste ehitamiseks loodud mootoriga. Nende liitreaalsusvõimekus tuleneb mootori enese koodibaasist, milles kahjuks ARKit 4 funktsionaalsused veel implementeeritud pole[4].

3.3 ARKit

Otsus keskenduda arendamisel just iOS seadmele ei tulnud koheselt. Kuigi Tartu Ülikooli kaudu oli võimalik kasutada kõige uuemat AR riistvara, oli autori esmainstinkt rakendus luua nõrgemale, kuid tuttavamale Android-seadmele. Seda kahel põhjusel: laenatud iPadi pidi jagama kaastudengitega ning Apple'i ökosüsteem, mis ei võimalda loodud rakendust väljaspool MacOS operatsioonsüsteemiga arvutit nutiseadmes käivitada. Need takistused said ületatud, ent ajaliselt kulukalt. Periooditi arendus seiskus, kuna iPad oli parajasti kaasõpilase käes ning rakenduse käivitamiseks oldi sunnitud Tartu Ülikoolilt lisaks laenama 2014. aasta Apple Mini. Viimase arvutusliku nõrkuse tõttu toimus arendus Windows arvuti peal. Rakenduse käivitamise konveier nägi seega välja järgmine:

1) IOS-rakendusfailide kompileerimine PC arenduskeskkonnas (Unity).
2) Loodud failide kopeerimine Apple Minisse läbi jagatud kausta.
3) Rakenduse kompileerimine ja käivitamine Apple'i Xcode tööriistas.

Kokku võttis see protsess tõrgete mitteilmnemisel aega umbes 15-20 minutit, mis muutis vigade tuvastamise valulikuks tööks.

Lõpliku otsuse keskenduda arendamise iOS seadmele põhjustas Apple'i konkurentidest võimsam liitreaalsusteek ARKit ja LiDAR skanneri tugi. Tabelis 1, kus on välja toodud eri AR teekides saadaolevad funktsionaalsused, on vahe Androidi vastava teegiga ARCore

[3] https://trello.com/b/TTAVI7Ny/ue4-roadmap

[4] https://developer.vuforia.com/forum/unity/get-depth-map-lidar-sensor-vuforia-fusion

silmnähtav. See, kombineeritult lihtsama installimise ja Apple-seadete ühemõistelise AR-Kit-i toega[5], on peamised põhjused, miks hoolimata Androidi ligi 85% turuosast nutitelefonide operatsioonisüsteemides on ARCore'i osakaal liitreaalsusrakendustes ARKit-ist väiksem [10].

Tabel 1. Liitreaalsusteekide võrdlus [11].

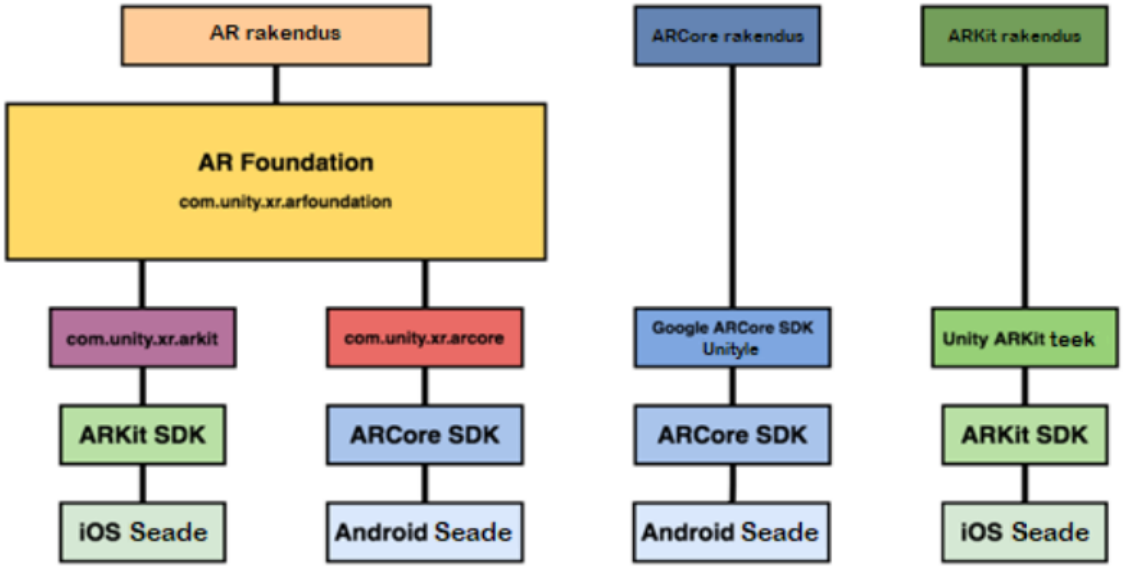
	ARCore	**ARKit**	**Magic Leap**	**HoloLens**
Seadme jälgimine (ingl *device tracking*)	✓	✓	✓	✓
Pindade jälgimine (ingl *plane tracking*)	✓	✓	✓	
Punktipilved (ingl *point clouds*)	✓	✓		
Ankrud (ingl *anchors*)	✓	✓	✓	✓
Valguse hindamine (ingl *light estimation*)	✓	✓		
Keskkonna sondid (ingl *environment probes*)	✓	✓		
Näo jälgimine (ingl *face tracking*)	✓	✓		
2D pildi jälgimine (ingl 2D *image tracking*)	✓	✓	✓	
3D objekti jälgimine (ingl 3D *object tracking*)		✓		
Võrestikuloome (ingl meshing)		✓	✓	✓
2D & 3D keha jälgimine (ingl 2D & 3D *body tracking*)		✓		
Koostöö (ingl *collaborative participants*)		✓		
Inimese segmentimine (ingl *human segmentation*)		✓		
Kiire heitmine (ingl *raycast*)	✓	✓	✓	
Edastatav video (ingl *pass-through video*)	✓	✓		
Seansikorraldus (ingl *session management*)	✓	✓	✓	✓
Oklusioon (ingl *occlusion*)	✓	✓		

Käesoleva töö huvides oli neist funktsionaalsustest iOS-i eelistamise kõige kaalukamateks argumentideks tabelist 1 ilmnev Androidi ARCore'i võrestikuloome võimekuse puudumine ning muidugi Apple'ile unikaalne LiDAR.

[5] Kuna Google ei oma Androidi jooksutava riistvara üle pea mingit sõnaõigust, on nende liitreaalsuse teek ARCore sunnitud pakkuma sõltuvalt riistvarast erinevat AR funktsionaalsust. Vaata lähemalt https://developers.google.com/ar/discover/supported-devices.

3.4 AR Foundation

Unity liitreaalsust võimaldava teegi valik oli raskem. Soov kasutada LiDAR skannerit vähendas valiku kahele: Apple'i ARKit ja Unity AR Foundation. AR Foundation ei ole teek ARCore'i või ARKiti mõistes, vaid pigem üldisem Unity liides eri riistvara teekidega suhtlemiseks. Seda suhtlust kirjeldab joonis 2.



Joonis 2. AR Foundationi arhitektuur [12].

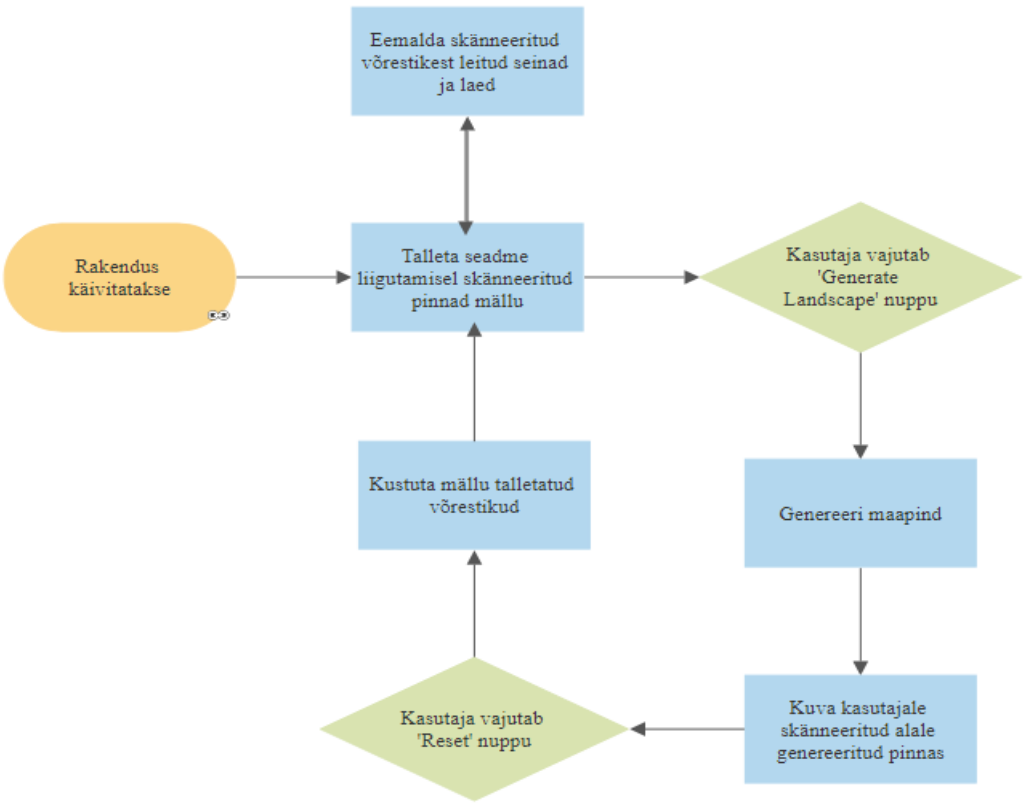
Selle kasutamise peamine põhjus – ühe rakenduse tugi eri platvormidel – antud töö skoobis tähtsust ei omanud. Valiku peamised põhjused olid mugavus ja dokumenteeritus. Kuna AR Foundation on Unity loodud, sobib see hästi viimase ökosüsteemi ning samuti omab see hulgaliselt avalikku näidiskoodi. Tegelikkuses oleks siinkohal võinud valida ka ARKiti, et korraldada koodi optimiseerimist madalamal tasemel, ent lõputöö maht ei võimaldanud niivõrd täpselt teeki süvenemist. Seetõttu sai selles usaldatud AR Foundationi võimekust.

4. Arendamine

Valmiva programmi disain ideest realiseerumiseni oli pidevalt ajas arenev koos autori teadmiste laienemisega. Käesolev peatükk tutvustab esmalt programmi lõplikku ülesehitust, järgnevalt sisendi samm-sammulist töötlemist ning viimaks esinenud probleeme koos leitud lahenduste ja kompromissidega.

4.1 Rakenduse kulg

Rakenduse kasutamise üldine kulg on kujutatud joonisel 3. Sellel kujutatud tegevustest toimusid „Talleta seadme liigutamisel skanneritud pinnad mällu“ ja „Kuva kasutajale skanneritud alale genereeritud pinnas“ Unity liitreaalsusteegi AR Foundationi-siseselt. Programmi loomisel seetõttu LiDAR skannerist lähtuva informatsiooni töötlemist ning genereeritu kujutamist liitreaalsuses polnud vaja taasluua. Sel põhjusel jäävad need antud kirjatööst välja.



Joonis 3. Rakenduse vooskeem (siniselt rakenduse, roheliselt kasutaja tegevus).

4.1.1 Pindade tuvastamine

AR Foundationi genereeritud võrestikest polnud kõik kasutuskõlblikud. Kuna eesmärgiks oli genereerida usutav loodusmaastik, tuli siseruumides kasutamise eesmärgil välja filtreerida tuvastatud seinad (sealhulgas aknad ja uksed) ja laed. Laed, kuna looduses maapinnaga paralleelseid pinde valdavalt ei leidu, ning seinad, kuna nende kõrgus teiste siseruumi objektidega võrreldes oleks põhjustanud nende domineerimise pea mistahes ruumis loodud maastikus.

Filtreerimise eesmärgil sai kasutatud ARKiti pindade klassifitseerimise funktsionaalsust[6]. See suudab skanneritust tuvastada järgmisi objekte: lage, ust, põrandat, istet, lauda, seina, akent. AR Foundation polnud seda funktsionaalsust töö kirjutamise hetkel veel enda teeki lisanud, küll aga oli võimaldatud vajadusel otse ARKiti Unity teegi poole pöördumine ning avaldatud selle kohta näidiskood[7]. Joonis 4 kujutab selle najal valminud tulemust.

Joonis 4. Kuvatõmmis loodud rakendusest enne maastiku loomist.
Tumedad alad on skaneeritud pinnad pärast sobimatute elementide eemaldamist.

[6] Vaata lähemalt https://developer.apple.com/documentation/arkit/armeshclassification

[7] https://github.com/Unity-Technologies/arfoundation-demos/blob/master/Assets/Meshing/Scripts/MeshClassificationFracking.cs

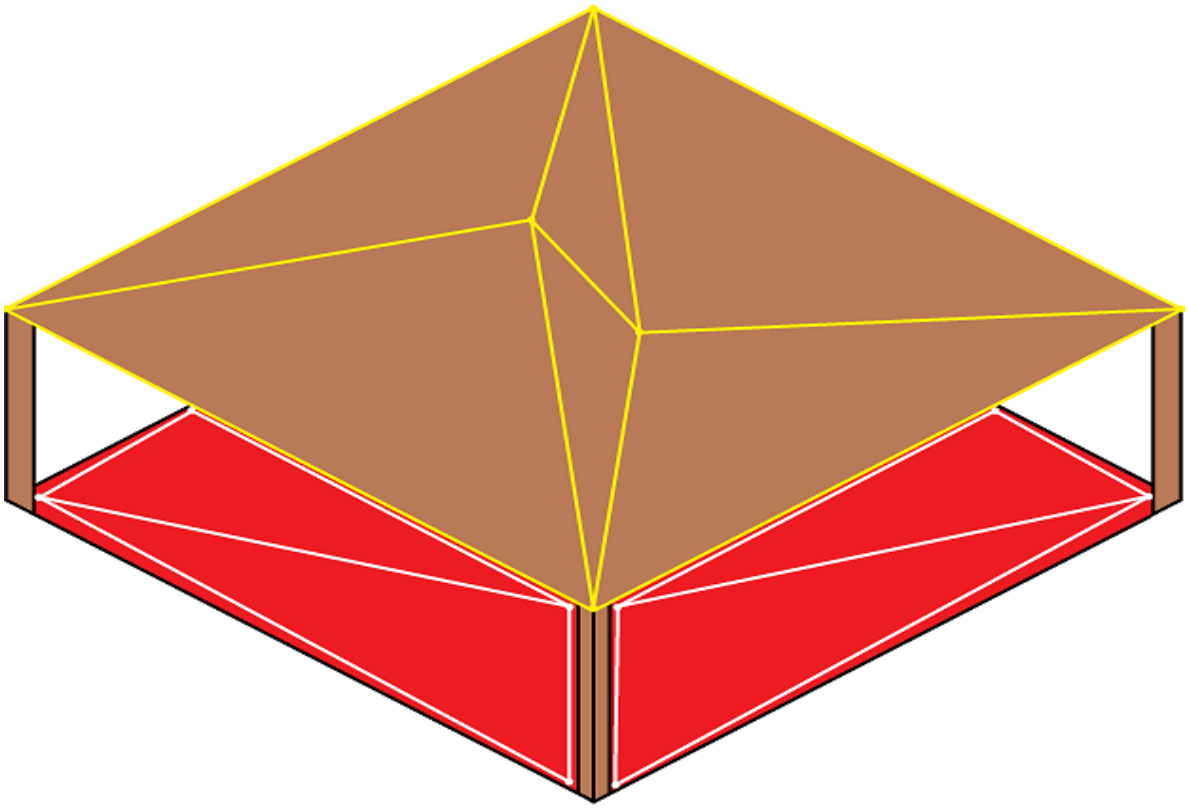
ARKiti klassifitseerimisfunktsionaalsus tagastab iga võrestiku kohta massiivi iga kolmnurga klassifikatsiooniga. Seetõttu oli vaja võrestikud taasluua kasutades vaid laena ja seinana mittetuvastatud kolmnurki.

Seinte ja lae väljafiltreerimine toimub rakenduses reaalajas, et maastiku genereerimisel aega kokku hoida.

Järgnevates sektsioonides kirjeldatu leiab rakenduses aset peale nupu „*Generate Landscape*“ (tlk genereeri maastik) vajutamist.

4.1.2 Reljeefkaardi konstrueerimine

Tuvastatud võrestike kõik punktid maastiku genereerimisel orientiiriks ei sobinud. Näiteks kui skaneerida joonisel 5 kujutatud lauda, siis oli vaja laua all tuvastatud punktid (joonisel punane ala) võrestikest eemaldada. Looduses taolisi horisontaalseid õõnsusi tavakorras ei esine. Selle vältimiseks oli lihtsaim meetod võrestikest kõikide punktide, mille kohal eksisteerib teine punkt, eemaldamine ehk keskkonna reljeefkaardi loomine.



Joonis 5. Laud (pruun) ja skannerile nähtav põrand selle all (punane). Kollased jooned illustreerivad skanneri moodustatud võrestiku kolmnurki laua peal, valged jooned kolmnurki laua all.

Sarnaselt joonisel 5 kujutatud võrestikele, ei asunud LiDAR-i tuvastatud punktid täpselt teineteise kohal. Seetõttu tuli reljeefkaardi konstrueerimisel igale punktile määrata nn mõjusfäär, mille seest leitud madalamad punktid võrestikust eemaldati. Kasutatud algoritmi pseudokood on järgmine:

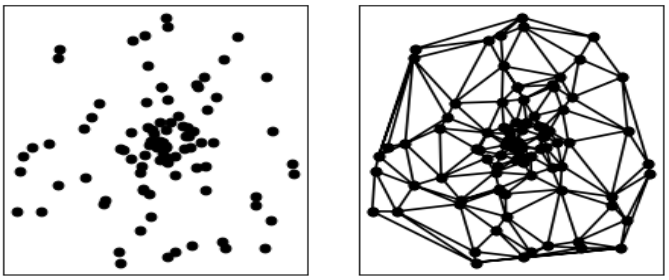
```
SORDI kõik punktid kõrguse järgi vähenevas järjekorras
IGA ( int i = punktid.koguarv-1; i >= 0; i-- )
    IGA ( int j = i+1; j < punktid.koguarv; j++ )
        KUI ( absoluutväärtus( punktid[i].asukoht – punktid[j].asukoht ) < täpsus )
            EEMALDA punktidest punkt[i]
            SUURENDA i ühe võrra,
```

kus täpsus oli testimise käigus leitud ujukomaarv, mis tähistas selle punkti mõjusfääri suurust; teisisõnu maksimaalset kaugust, mil punkt asus teisega võrdlemiseks piisavalt sarnastel koordinaatidel. Kui täpsus sai liiga väike, siis pääsesid võrestikku ka punktid, mis asusid tegelikkuses kõrgema objekti all (joonisel 5 punktid punaselt alalt). Liialt suure täpsuse korral sai aga eemaldatud väga palju punkte, mistõttu tuli nende põhjal loodud võrestik väga nurklik.

Võrestikust punktide eemaldamine ei ole aga sugugi triviaalne probleem ning nõuab võrestiku parandamist või täiesti uue loomist. Kuna ülalkirjeldatud algoritm jättis kõrvale suure arvu punkte, oli lihtsam säilitatud punktidest ehitada täiesti uus võrestik.

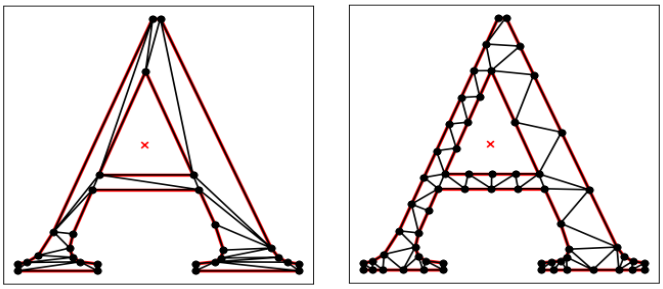
4.1.3 Delaunay triangulatsioon

Delaunay triangulatsioon on NSVLi teadlase Boriss Delaunay järgi nimetatud lähenemine punktihulga (P) ühendamiseks kolmnurkadega nii, et ühegi kolmnurga igat tippu läbiva ringi sisse ei jääks ükski teine punkt hulgast P ning ühegi kolmnurga külg ei kattuks teisega [13] (kujutatud joonisel 6). Taoline lähenemine garanteerib, et loodavate kolmnurkade iga nurk on maksimaalne võimalik [13], niiviisi tõstes neist loodava võrestiku kvaliteeti. Kuna reljeefkaardi loomisel jäid alles vaid iga koordinaadi kõrgeimad punktid, võis olemasolevad punktid taandada kahele dimensioonile, ignoreerides kõrgusi, ning neist Delaunay triangulatsiooniga konstrueerida loodava maastiku võrestik.



Joonis 6. Suvalistest punktidest (vasakul) konstrueeritud Delaunay triangulatsioon (paremal) [13].

Kuna püstitatud eesmärgiks oli maastiku genereerimine mõistliku aja jooksul ning käsitletavate punktide arv suur, oli mõistlik delegeerida triangulatsiooni teostus välisele, optimeeritud algoritmile. Selleks sai kasutatud California ülikooli professori Jonathan Shewchuki vabavaralise koodiprojekti Triangle[8] tõlget C# keelde Triangle.Net-i[9] näol, mis mõlemad on kasutatavad MIT litsentsi alusel. Lisaks Delaunay võrestiku loomisele pakub Triangle loodule ka kvaliteedigarantiid – kui loodavate kolmnurkade vähim nurk langeb alla kindla kraadi, siis lisatakse võrestikku kvaliteedi säilitamiseks lisapunkte [13]. Seda illustreerib joonis 7.



Joonis 7. Delaunay triangulatsiooni (vasakul) tihendamine lisatud punktidega (paremal) [13].

Võrestikku lisatud punktid parandasid esimese kvaliteeti, kuid kuna Delaunay triangulatsioon opereerus punktide 2D koordinaatide põhjal, vajasid nende kõrgused korrektuuri. See sai saavutatud omistades igale lisatud punktile seda puutuvate kolmnurkade keskmise kõrguse. Optimeerimise huvides sai siinkohal ka järskude vertikaalpindade sujumiseks tõstetud (kuid mitte langetatud) iga kolmnurga punktid selle kolmnurga keskmisele kõrgusele üldise võrestiku sujumiseks.

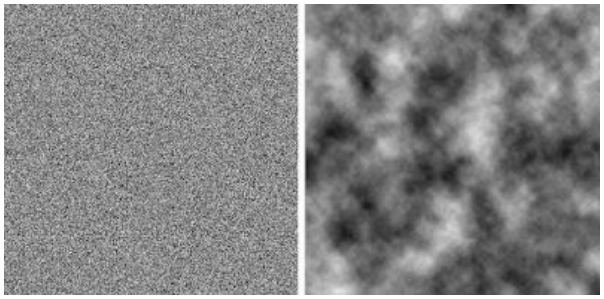
4.1.4 Müra lisamine

Saadud võrestik, ükskõik kui sujuv, ei sarnanenud veel looduses esinevatele pinnavormidele. Skaneeritud keskkonnad dikteerisid küll genereeritud maastiku suuremaid pinnavorme (mäed, orud), ent siseruumide normaalselt siledad pinnad põhjustasid nende detailivaesuse. Seega oli vaja olemasolevasse võrestikku tuua detailsust, samas vältides loodavasse maastikku silmnähtavate mustrite loomist. Sellel otstarbel sai võrestikule rakendatud mürakaarti.

[8] http://www.cs.cmu.edu/~quake/tripaper/triangle0.html

[9] https://archive.codeplex.com/?p=triangle

Mürakaardi konstrueerimisel sai kasutatud sujuvat müraalgoritmi, täpsemalt Unity Mathf klassi sisseehitatud Perlini müra. Sujuvad müraalgoritmid on protseduurilises maailmaloomes kontekstivaba, ühtlase pinnase loomisel levinud tööriistad. Seda osalt – nagu on nähtav joonisel 8 – algoritmi loodud konteksti illusiooni tõttu (naabruses paiknevate punktide kõrgusevahed pole suured, üleminekud on sujuvad). Illusioon kontekstist tegelikkuses naabritest sõltumata oli antud töö skoobis oluline, kuna loodud võrestik ei olnud konstantne (punkte ei saanud asetada ruudustikule, nende vahed varieerusid).

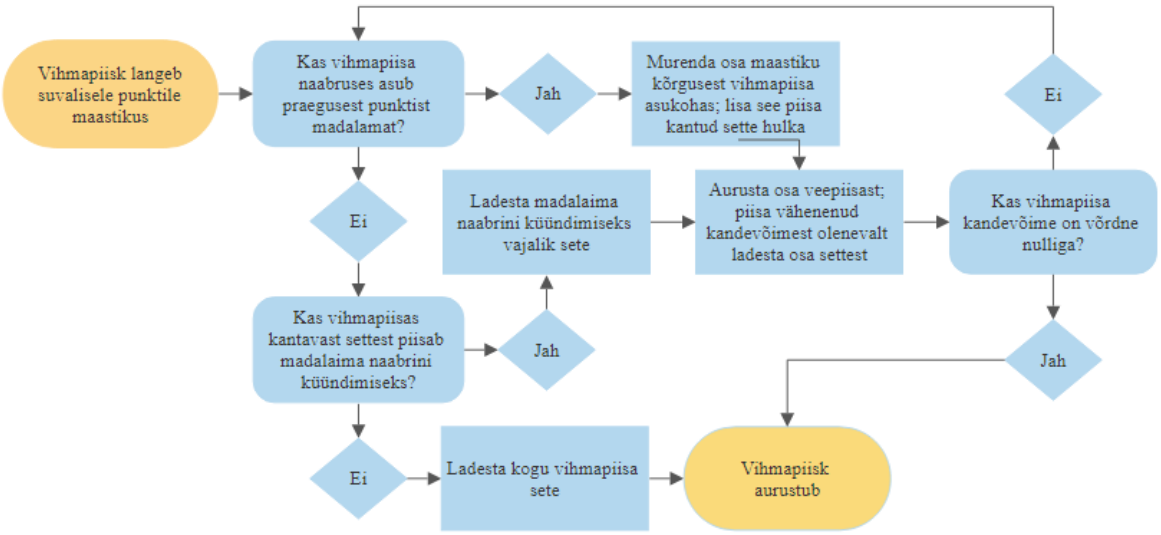


Joonis 8. Valge mürakaart (vasakul) vs Perlini mürakaart (paremal) [14].

Eri parameetritega mürakaarte kombineerides sai leitud antud rakendusele sobiv väiksemate pinnavormide detailsustase.

4.1.5 Erosioon

Loodud pinnavormi veel looduslikuma väljanägemise saamiseks sai võrestikul jäljendatud sademetest põhjustatud erosiooni. Vihmapiisad murendavad aastate jooksul pinnast ning kannavad seda allamäge, orgudes vee aurustumisel neid ladestades. Programmiliselt on seda efekti võimalik jäljendada üsna täpselt, asendades protsessi tegeliku aastatepikkuse kestvuse individuaalsete veepiiskade drastilisema mõjuga maapinnale.



Joonis 9. Erosiooni vooskeem.

Joonis 9 kirjeldab ühe loodud vihmapiisa elukäiku jäljendavat algoritmi. Piiskade koguarv sai võrdne maastikuvõrestikus asuvate punktide koguarvuga ning nende aurustumine toimub üle 20 tsükli (tsükkel = madalamale punktile voolamine).

4.1.6 Maastiku värvimine

Loodud võrestiku toonimine toimub loodud programmi järeltöötluses. Unity materjalide .shader failid võimaldavad kasutades HLSL/Cg programmeerimiskeelt loodu kuvamist otse kontrollida. Selle funktsionaalsust kasutades sai maastiku värvus sõltuvaks selle kõrgusest ning järsakust. Kõrgus dikteerib punkti tooni skaalal kollane (liiv) – roheline (muru) – valge (lumi); kallaku järskus punkti tooni skaalal pinnavärv (joonisel 10 kujutatud kollane-roheline-valge) – hall (kalju).

Joonis 10. Mitte-kaljuse maastiku värvitoon.

4.2 Probleemkohad

Eespool kirjeldatud programmi ülesehitus valmis katsetamise ja eksimise teel. Osad lähenemised töötasid, teised vajasid parandamist ning kolmandad nõudsid kompromisse. Järgnevas sektsioonis on välja toodud olulisimad esinenud probleemid koos lahendustega.

4.2.1 Programmi kompileerimine

Kirjutatud koodist iPadil jooksutatava programmi kompileerimine oli ebameeldivalt veniv protsess. Lõplik ahel nägi ette Unitys programmifailide ehitamise, nende saatmise üle kohaliku võrgu teise, Tartu Ülikoolilt laenatud 2014. aasta Apple Minisse, kus failid Xcode'is viimaks programmiks sai kompileeritud. Arendamise jooksul sai selle kestvus lõpuks saadud umbes 20 minuti peale ning seegi kompileerimisahela sujumisega (Unity mängumootorile on võimalik projektifailide ehitamisel kaasa anda mitmed Apple'i kompileerimisprogrammi Xcode'i parameetrid, et vältida nende korduvat sisestamist). Võib väita, et see protsess jäigi probleemiks, mis lahendust ei leidnud.

Parandusi sai otsitud. Esmane reaktsioon oli macOS operatsioonisüsteemi emuleerimine autori Windows-arvutil, et kasutada selle suuremat võimsust Xcode'is rakenduse kiiremaks

kompileerimiseks ning säästa aega projektifailide võrgus transportimise pealt. Kiire netiotsing paljastas aga, et igasugune Apple tarkvara jooksutamine mitte-Apple riistvaral on nende litsentsi vastu, mistõttu jäi antud lähenemine kõrvale. Veel sai lahendusena otsitud piisavalt võimekat macOSi terve projekti arendamiseks, kuid taolisi masinaid kuskilt laenamiseks ei leitud. Kuna LiDAR-i kasutamiseks oli Xcode möödapääsmatu, jäi kirjeldatud aeganõudev ahel kasutusse.

Üks kasutuseloleva kompileerimisprotsessi nõrkusi oli veatuvastuse kaotsiminek. Nimelt ei kuvanud Xcode vea toimumisel korrektset virna jälge (ingl *stack trace*). See probleem sai ületatud lihtsa, kuid koodi loetavuse huvides väga tüütu lahendusega – hetkese koodi täitmise asukoha väljaprintimisega konsooli.

4.2.2 Koordinaadid

Seoses nutiseadmete liitreaalsusrakenduste tööpõhimõttega, et rakendust võib käivitada mil iganes, peavad AR funktsionaalsust pakkuvad teegid suutma tööd alustada olenemata kaamera/skanneri positsioonist. Sel põhjusel alustab Unity liitreaalsusrakendustes kaamera positsioonilt (0, 0, 0) ning kõikide tuvastatud objektide positsioonid talletatakse selle algse asukoha suhtes. Õnneks ei dikteeri nutiseadme esialgne positsioon ka x, y ja z telgede suundi - seda teeb seadme güroskoop [1]. Seetõttu ei pea iga rakendus ise välja nuputama, mis suunas asub „all“ või „üles“. Kuna algpositsioon võib nutiseadme asukohast olenevalt olla ükskõik kus, peavad kõik kindlal kõrgusel/kaugusel orienteeruvad funktsioonid opereeruma suhtelistel vahedel. See pole niivõrd suur probleem, kuivõrd asi mida liitreaalsusrakenduse arendamisel silmas pidada, et mitte konfigureerida parameetreid mis programmi uuel käivitamisel enam asjakohased pole.

Koordinaatide osas põhjustas veel probleeme otsus luua ebakorrapärane maastikuvõrestik. See tekitas vajaduse kasutada ajaliselt ebaefektiivseid algoritme (täpsemalt kirjeldatud järgmises peatükis). Parem lahendus oleks olnud skaneeritud maastiku ruudustikuks jagamine ning iga ruudu väärtustamine kõige kõrgema selle alalt leitud punktiga. See lähenemine oleks loonud reljeefkaardi, mille võrestiku kolmnurkade küljed oleks konstantsete pikkustega. Ka kasutatud Delaunay triangulatsiooni-algoritmi lubatud kvaliteedigarantii oleks siis juba eos täidetud olnud.

4.2.3 Vähene materjal

Kõige akuutsem probleem antud töö loomisel oli vähene allikmaterjal. Liitreaalsus on väga kiiresti arenev valdkond, kus vaid paari aasta tagused materjalid on sisult vananenud. Eriti tuntav on see AR-i uusimate innovatsioonide – nagu LiDAR-i – puhul. AR Foundationi juures eksisteerib funktsionaalsus, kuid dokumentatsioon ning arutelu selle kohta on käesoleva töö kirjutamise hetkel puudulik.

Tulevastele LiDAR-it Unity mängumootoriga kombineerida soovijatele on seega siinkohal välja toodud antud projekti arendamiseks kasulikuks osutunud ressursid:

- Unity AR Foundationi manuaal[10] – Tarkvara tootja loodud manuaal peaks olema alati esimene koht kust informatsiooni otsida. LiDAR-i kohta jääb see napisõnaliseks, kuid on hea koht kust alustada.
- Unity AR Foundationi koodinäidised[11] - Olulisim kasutatud materjal. Pakub hästidokumenteeritud näidisprojekte demonstreerimaks põhilisi funktsionaalsusi.
- Dilmer Valecillose YouTube'i kanali AR Foundationi esitusloend[12] - Väga kasulik koht nägemaks kindla funktsionaalsuse rakendamist algusest lõpuni. Kuna kanali omanik jälgib liitreaalsuse uusimaid arenguid, leiab sealt ka LiDAR-it puudutavaid videoid.

Loodetavasti on tulevastele arendajatele kasulik ka käesolevas töös loodud programmi lähtekood (vt lisa I. Lisatud failid).

AR Foundationi üldisema funktsionaalsuse kohta on materjalid internetist lihtsasti leitavad, seega neid siinkohal välja ei tooda.

[10] https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/manual/index.html

[11] https://github.com/Unity-Technologies/arfoundation-samples

[12] https://www.youtube.com/playlist?list=PLQMQNmwN3FvzLN-8moCKmZb00gr7sdcrZ

5. Tulemus

Iga tarkvaraarendusprojekti korral on oluline tulemuse kriitiline analüüs käesoleva ja/või edasiste projektide arengu tagamiseks. Siinkohal autor teadvustab, et valminud rakendus omab väga mitmesuguseid võimalusi edasisteks arendusteks, millest paljud nõuaksid aga süsteemi arhitektuuris kardinaalseid muutusi.

Joonis 11. Kuvatõmmised rakendusest enne (vasakul) ja pärast (paremal) maapinna genereerimist.

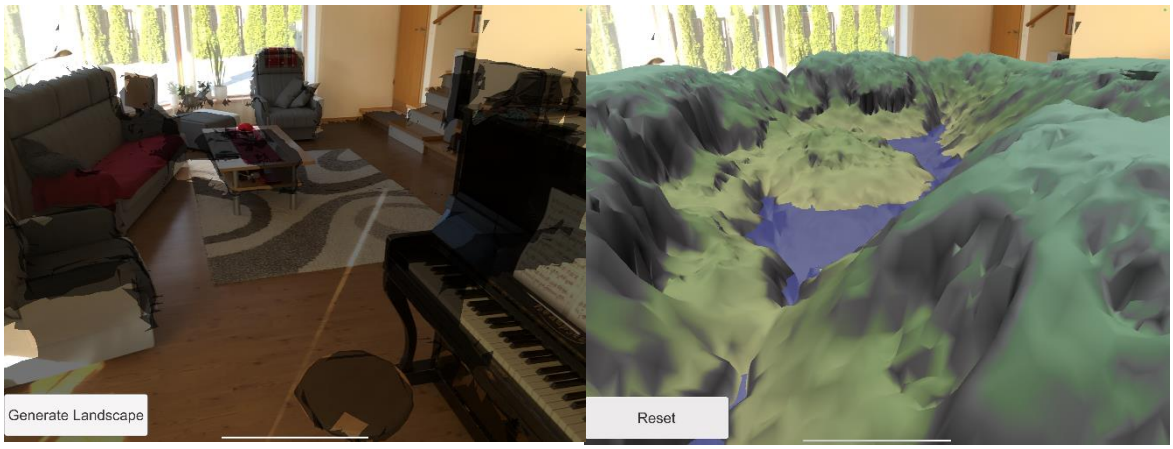
Antud peatükk kirjeldab joonisel 11 kujutatu-laadse tulemuse genereerimise kiirust ja võimalikke edasiarendusi.

5.1 Jõudlus

Kuna LiDAR skanneriga nutiseadmeid on töö kirjutamise hetkel vaid kaks – 2020 iPad Pro ja 2020 iPhone Pro -, ning ühele neist autoril ligipääs puudub, on rakenduse võimekust mõõdetud vaid arendamiseks kasutatud iPadil. See seade kasutab 8-tuumalist Apple'i M1 protsessorit ja omab 8 GB muutmälu[13].

Algoritmi ajalise keerukuse hindamiseks sai rakendusele edastatud maksimaalselt võrgustikus lubatud punktide arv, misjärel sai maastiku genereerimise eri etapid ajaliselt jäädvustatud.

Joonis 12. 1000 punktiga võrestik enne ja pärast maapinna genereerimist.



Joonis 13. 12 000 punktiga võrestik enne ja pärast maapinna genereerimist.

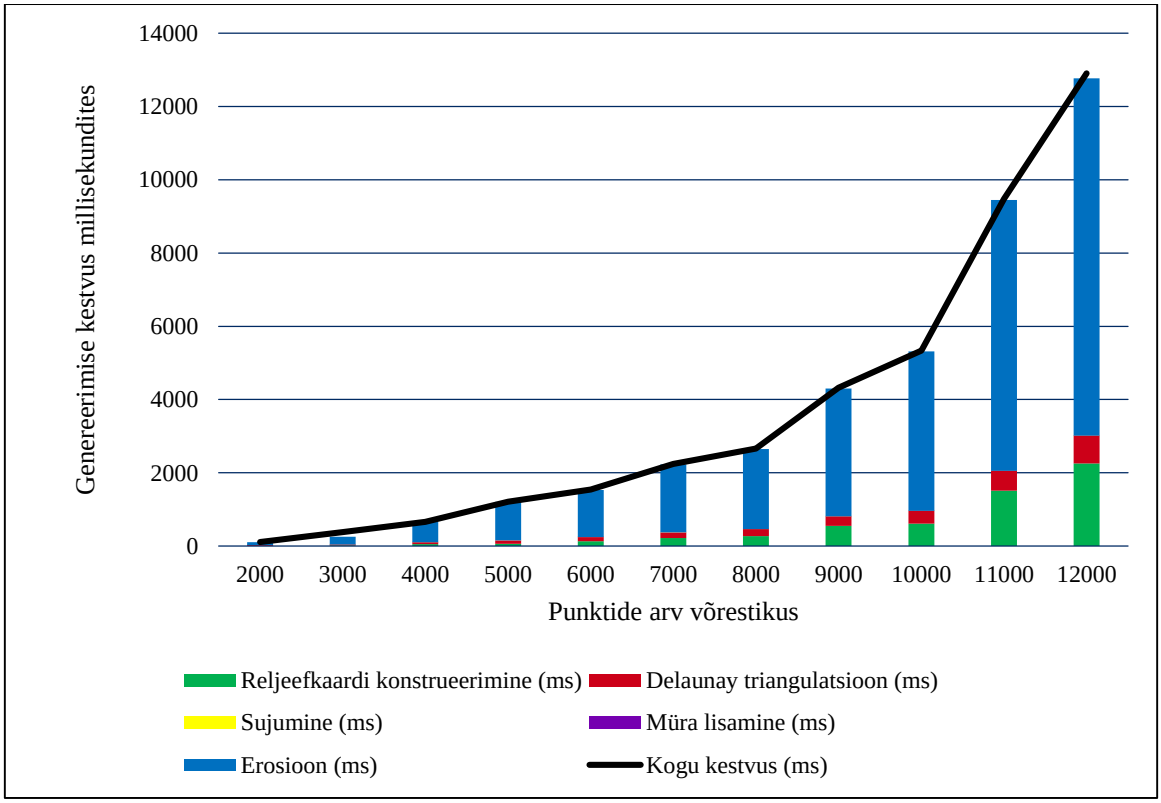
[13] https://www.apple.com/ipad-pro/specs/

Orientiiriks järgneva tabeli 2 mõistmiseks kujutab joonis 12 rakenduse loodud pinnavormi, kus võrestikus on 1000 punkti, ning joonis 13, kus võrestikus punkte on 12 000.

Tabel 2. Maastiku genereerimise ajakulu

Punkte võrestikus	**2000**	**3000**	**4000**	**5000**	**6000**	**7000**	**8000**	**9000**	**10000**	**11000**	**12000**
Kogu kestvus (ms)	112	378	665	1214	1547	2241	2662	4321	5338	9482	12907
Reljeefkaardi konst-rueerimine (ms)	1	20	50	63	132	220	274	555	611	1510	2257
Delaunay triangulat-sioon (ms)	28	28	60	91	115	153	191	257	353	547	757
Sujumine (ms)	0.1	0.1	0.2	0.3	0.3	0.3	0.4	0.5	1	0.6	1
Müra lisamine (ms)	0.3	0.6	0.6	0.8	0.9	1	1.5	1.8	2	2	2.7
Erosioon (ms)	81	207	549	1050	1287	1852	2179	3488	4349	7393	9751

Maastiku genereerimise ajalise keerukuse kasvutrendi kujutab joonis 14:



Joonis 14. Maastiku genereerimise ajakulu kasv.

Nagu graafikult ilmneb, on ajalise keerukuse eksponentsiaalse kasvutrendi põhjus erosiooni simuleerimine. Erosiooni praegune naiivne rakendus (iga sademe iga tsükli uurimine) on selge koht, mida algoritmi kiiruse arendamiseks parandada. Mööndusena peab küll mainima, et 11 000 - 12 000 punkti omav maastik on küllaltki ekstreemne näide kuna enamik siseruumidest sedavõrd suurt võrestikku vajavat ala ei oma. Odavama erosiooni algoritmi rakendamine on siiski vajalik kui plaanis on maastiku resolutsiooni tõstmine.

5.2 Edasiarendused

Olemasoleva projekti edasiarendustes on esmane lühendada genereerimisprotsessi ajakulu. Seda saab saavutada efektiivsemate või tulemust jäljendavate algoritmidega. Rakendada saaks ka paralleliseerimist – näiteks skaneerimise jooksul istub seadme CPU praeguse seisuga pea tegevusetult, kui sellal võiks rakendus juba hilisema maastiku genereerimise tarbeks esmaseid arvutusi teha. Kui ka optimeeringutega saab algoritm liiga aeglane, peaks kasutusele võtma sisuplokid, et kasutajale oleks võimalik jooksvalt vähemalt mingisugust tulemust näidata. Maastiku genereerimise kiiruse parandamine võimaldaks välja vahetada ka senise robustse reljeefkaardi konstrueerimise ning seeläbi paranda läbi maastikuvõrestiku resolutsiooni tõstmise loodud maapinna kvaliteeti. Suurem resolutsioon lubaks omakorda kasutusele võtta erinevad tulemuse kujutamiseskaalad (nutiseadmega kindlale pinnavormile lähenedes saaks genereerida/mälust laadida kõrgema detailsusega võrestiku, ehk võimaldada kasutajal saada nii üldine või spetsiifiline ülevaade nagu ta soovib).

Projekti arendamine süsteemi arhitektuuri poolelt avaks veelgi enam võimalusi. Peamine neist on loodusmaastiku genereerimine reaalajas. See vajab aga väga efektiivset koodi ning tihedat ARKitiga suhtlemist, mis ei pruugi AR Foundationiga võimalik olla. Tõenäoliselt oleks samuti areng võrestikus ruudustiku kasutuselevõtt, kuna praegused suvalistest vahedest koosnevad kolmnurgad sunnivad algoritmis puntki naabrite leidmiseks iga kolmnurga uurimist, niiviisi taaskord maastikuloomet aeglustades. Näiteks lubaks ruudustik võrestikus kasutusele võtta lineaarse ajalise keerukusega erosiooni algoritmi [15].

Kuna projekti eesmärk on loodusmaastiku jäljendamine, leidub potentsiaalselt lisatavaid elemente pea lõpmatult. Lisada võib taimestikku, bioome, erisuguseid veekogusid ja palju muud.

6. Kokkuvõte

Käesoleva bakalaureusetöö käigus loodi kasutajat ümbritsevat keskkonda arvestav loodusmaastikku genereeriv liitreaalsusrakendus. Uusimate iOS seadmete LiDAR skannerit kasutav valminud rakendus suutis kaardistada orientiiridena sobivaid pinnad, genereerida sellele alale looduses esinevat maastikku jäljendava pinnase, ning kuvada loodut liitreaalsuses. Püstitatud sihtidest võis mittesaavutatuks lugeda vaid pinnase genereerimiseks ette nähtud mõistlikku ajakulu, kuid selle eesmärgi teostamiseks sai kirjeldatud potentsiaalset lahendust.

Loodud rakendus saavutas looduslikku maastikku meenutava pinnase genereerimise baseerudes läbi Unity mängumootori AR Foundationi teegi LiDAR skannerist lähtuval informatsioonil objektide asukohtadest. Avastatud pindadest sai filtreeritud välja skanneritud laed, seinad, uksed ja aknad. Maastikuvõrestik tekitati peale nupu „*Generate Landscape*“ vajutamist. Võrestiku moodustamiseks leidis rakendus kõik keskkonnas tuvastatud punktid mille kohal ei eksisteerinud kõrgemat punkti ning kombineeris need kasutades Delaunay triangulatsiooni. Saadud pinde katvaid võrestikku siluti, selle detailsust parandati mürakaardiga ning sellel emuleeriti sademete põhjustatud erosiooni mõju. Loodu tooniti järeltöötluses ning esitati kasutajale liitreaalsuses.

Peale rakenduse arhitektuuri ja tööprotsessi kirjeldamise sai tutvustatud ka kasutatud tehnoloogiaid, tuues põhjendusi nende kasutamiseks alternatiivide ees, analüüsitud tulemuse jõudlust algoritmi kiiruse seisukohast ning loetletud võimalikke edasisi arendusi.

Antud töös käsitletud projekti arendamise algusjärgus oli autor nutiseadmete LiDAR skanneri kasulikkuse osas kõhkleval seisukohal. Selle arvamuse lükkas aga kiiresti ümber kasutatud seadme täpsus ümbritseva keskkonna kaardistamisel ning selles orienteerumises. Lisaks ajas muutuvate asupaikade ja objektide kestvale jäädvustamisele võimaldab LiDAR tehnoloogia levik ja areng nutiseadmetes liitreaalsusrakenduste uuele tasemele viimist. Autori arvates on ka käesoleval programmil edasise arendamisega potentsiaali pakkuda kasutajaile ainulaadset võimalust mistahes keskkonnas pääseda hetkeks meeldiva loodusmaastiku keskmesse.

7. Viidatud kirjandus

1. Luong H., Philips W., Vlaminck M. A markerless 3D tracking approach for augmented reality applications. *2017 International Conference on 3D Imaging (IC3D)*. Brussels: IEEE, 2017, pp 1-7.
2. Apple Inc. Apple unveils new iPad Pro with breakthrough LiDAR Scanner and brings trackpad support to iPadOS. *Newsroom*, 2020. https://www.apple.com/newsroom/2020/03/apple-unveils-new-ipad-pro-with-lidar-scanner-and-trackpad-support-in-ipados (14.04.2021).
3. Dembogurski R., Filho J.L., Sad D.O., Silva R., Vieira M.B. Interactive Virtual Terrain Generation using Augmented Reality Markers. *SBC Journal on 3D Interactive Systems*. Niterói: Brazilian Computer Society, 2012, pp 29-36.
4. Heritage G., Large A. Laser Scanning for the Environmental Sciences. Manchester: John Wiley & Sons, 2009, pp 2-7.
5. Borkar B., Kote S. A Survey on Marker-less Augmented Reality. *International Journal of Engineering Trends and Technology (IJETT)*. Tiruchirappalli: Seventh Sense Research Group, 2014, pp 639-641.
6. AKIT: Andmekaitse ja infoturbe leksikon. https://akit.cyber.ee/.
7. Apple Inc. Introducing ARKit 4. https://developer.apple.com/augmented-reality/arkit (14.04.2021).
8. Ghouri T.L., Hussain A., Hussain F., Shakeel H., Uddin N. Unity Game Development Engine: A Technical Survey. *University of Sindh Journal of Information and Communication Technology*. Jamshoro: University of Sindh, 2020, pp 73-81.
9. Unity Technologies. https://store.unity.com (14.04.2021).
10. Abderrahmani A.E., Oufqir Z., Satori K. ARKit and ARCore in serve to augmented reality. *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*. Fez: IEEE, 2020, pp 1-7.
11. Unity Technologies. About AR Foundation, 2021. https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/manual/index.html (14.04.2021).

12. Miller D., Mowrer T., Weiers B. Unity’s Handheld AR Ecosystem: AR Foundation, ARCore and ARKit. 2018. https://blogs.unity3d.com/2018/12/18/unitys-handheld-ar-ecosystem-ar-foundation-arcore-and-arkit (14.04.2021).
13. Shewchuk J.R. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. 1996. http://www.cs.cmu.edu/~quake/tripaper/triangle0.html (14.04.2021).
14. Strochinsky B. Random Map Generator. 2015. http://brandenstrochinsky.blogspot.com/2015/05/random-map-generator.html (14.04.2021).
15. Beyer H.T. Implementation of a method for hydraulic erosion. Technische Universität München Department of Informatics. 2015.

Lisad

I. Lisatud failid

ARTerrainGeneration – Loodud programmi lähtekoodi sisaldav kaust. Kahjuks ei võimalda Apple'i ökosüsteem valmisrakenduse jagamist väljaspool nende turgu App Store. Lähtekood on litsentseeritud käesoleva kirjatööga samadel alustel.

Demo.mp4 – Loodud programmi tööd demonstreeriv video.

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina,

Andre Ahuna,

(autori nimi)

annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

Loodusmaastiku genereerimine liitreaalsuses,

(lõputöö pealkiri)

mille juhendaja on

Madis Vasser,

(juhendaja nimi)

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

1. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.
2. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Andre Ahuna

06/05/2021