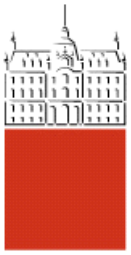
Univerza
v Ljubljani
Fakulteta
za gradbeništvo
in geodezijo



Jamova cesta 2
1000 Ljubljana, Slovenija
http://www3.fgg.uni-lj.si/

DRUGG – Digitalni repozitorij UL FGG
http://drugg.fgg.uni-lj.si/

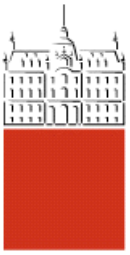
To je izvirna različica zaključnega dela.

Prosimo, da se pri navajanju sklicujte na bibliografske podatke, kot je navedeno:

Zelenbaba, D., 2016. Modeliranje podatkov terestričnega laserskega skeniranja za uporabo v navidezni resničnosti. Magistrsko delo. Ljubljana, Univerza v Ljubljani, Fakulteta za gradbeništvo in geodezijo. (mentorica Kosmatin Fras, M., somentor Grigillo, D.): 83 str.

Datum arhiviranja: 26-04-2016

University
of Ljubljana
Faculty of
Civil and Geodetic
Engineering



Jamova cesta 2
SI – 1000 Ljubljana, Slovenia
http://www3.fgg.uni-lj.si/en/

DRUGG – The Digital Repository
http://drugg.fgg.uni-lj.si/

This is original version of final thesis.

When citing, please refer to the publisher's bibliographic information as follows:

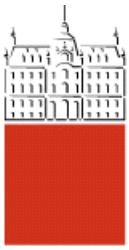
Zelenbaba, D., 2016. Modeliranje podatkov terestričnega laserskega skeniranja za uporabo v navidezni resničnosti. Master Thesis. Ljubljana, University of Ljubljani, Faculty of civil and geodetic engineering. (supervisor Kosmatin Fras, M., co-supervisor Grigillo, D.): 83 pp.

Archiving Date: 26-04-2016

Univerza
v Ljubljani

Fakulteta za
gradbeništvo in
geodezijo

Jamova 2
1000 Ljubljana, Slovenija
telefon (01) 47 68 500
faks (01) 42 50 681
fgg@fgg.uni-lj.si



MAGISTRSKI ŠTUDIJSKI PROGRAM DRUGE STOPNJE GEODEZIJA IN GEOINFORMATIKA

Kandidat:

DINKO ZELENBABA

MODELIRANJE PODATKOV TERESTRIČNEGA LASERSKEGA SKENIRANJA ZA UPORABO V NAVIDEZNI RESNIČNOSTI

Magistrsko delo št.: 14/II.GIG

MODELING THE DATA OF A TERRESTRIAL LASER SCANNING FOR USE IN VIRTUAL REALITY

Graduation – Master Thesis No.: 14/II.GIG

Mentorica:
doc. dr. Mojca Kosmatin Fras

Somentor:
asist. dr. Dejan Grigillo

Ljubljana, 25. 04. 2016

STRAN ZA POPRAVKE, ERRATA

Stran z napako	Vrstica z napako	Namesto	Naj bo

Ta stran je namenoma prazna.

IZJAVE

Podpisani **Dinko Zelenbaba** izjavljam, da sem avtor magistrskega dela z naslovom: »**MODELIRANJE PODATKOV TERESTRIČNEGA LASERSKEGA SKENIRANJA ZA UPORABO V NAVIDEZNI RESNIČNOSTI**«.

Izjavljam, da je elektronska različica v vsem enaka tiskani različici.

Izjavljam, da dovoljujem objavo elektronske različice v digitalnem repozitoriju.

Ljubljana, 15.04.2016.

Dinko Zelenbaba

Ta stran je namenoma prazna.

BIBLIOGRAFSKO – DOKUMENTACIJSKA STRAN IN IZVLEČEK

UDK:	**004.946.5:528.7(043)**
Avtor:	**Dinko Zelenbaba**
Mentor:	**doc. dr. Mojca Kosmatin Fras, univ. dipl. inž. geod.**
Somentor:	**asist. dr. Dejan Grigillo, univ. dipl. inž. geod.**
Naslov:	**Modeliranje podatkov terestričnega laserskega skeniranja za uporabo v navidezni resničnosti**
Tip dokumenta:	**Magistrsko delo – univerzitetni študij**
Obseg in oprema:	**83 str., 5 pregl., 69 sl.,**
Ključne besede:	**3D modeli, modeliranje, navidezna resničnost, terestrični laserski skener, igralni pogon**

Izvleček:

V magistrski nalogi obravnavamo področje 3D modeliranja oblakov točk, pridobljenih s terestričnim laserskim skeniranjem, načine pridobivanja 3D modelov ter njihovo uporabo in podrobneje uporabo v navidezni resničnosti na primeru video iger. Naloga je sestavljena iz teoretičnega in praktičnega dela. V teoretičnem delu so opisane vrste 3D modelov in 3D podatkov, načini njihovega pridobivanja in uporabe. Opisana je zgodovina industrije video iger, navidezna resničnost kot sestavni del te industrije, video igre in vrste video iger. Poudarek je na prvoosebnih (angl. first person) video igrah, ki se dogajajo v ustvarjenih navideznih 3D prostorih skozi katere se igralec giblje in ob tem rešuje različne naloge. Opisani so tudi elementi video iger, ki jih lahko povežemo z večpredstavnostno kartografijo oziroma splošno z geodezijo in geoinformatiko. Praktični del magistrske naloge je sestavljen iz treh delov: pridobivanje podatkov (skeniranje objekta), obdelava podatkov (registracija oblakov točk, izdelava modela) in uporaba modela (izdelava aplikacije navideznega sprehoda v igralnem pogonu). Za potrebe pridobivanja podatkov oz. skeniranja je izbran objekt, ki ima zgodovinski pomen. Pri obdelavi podatkov smo preizkusili več programov, ki omogočajo obdelavo oblaka točk in izdelavo modelov. Na koncu je razložen način, s katerim se izdela najenostavnejši nivo aplikacije sprehajanja okoli izdelanega modela objekta s pomočjo izbranega orodja (igralni pogon), v katerem so izdelane mnoge popularne video igre.

BIBLIOGRAPHIC – DOCUMENTALISTIC INFORMATION AND ABSTRACT

UDC:	**004.946.5:528.7(043)**
Author:	**Dinko Zelenbaba**
Supervisor:	**Assist. Prof. Mojca Kosmatin Fras, Ph.D.**
Cosupervisor:	**Assist. Dejan Grigillo, Ph.D.**
Title:	**Modeling the data of a terrestrial laser scanning for use in virtual reality**
Document Type:	**M. Sc. Thesis**
Scope and tools:	**83 p., 5 tab., 69 fig.**
Keywords:	**3D models, modelling, virtual reality, terrestrial laser scanner, game engine**

Abstract

In this master's thesis we explore the 3D modeling of point clouds that were obtained through terrestrial laser scanning, methods of obtaining 3D models and their usability in virtual reality and video games. The thesis consists of a theoretical part and a practical part. The theoretical part describes types of 3D models and 3D data, ways to obtain them and forms of their usage. It also deals with the history of video gaming and virtual reality as a constituent of the video game industry, types of video games with a focus on “First person” video games that take place in constructed virtual 3D spaces and consist of moving through the virtual space along with solving various tasks. Elements of video games that are connected to multi-medial cartography and geodesy and geoinformatics in general are also addressed. The practical part of the thesis consists of three parts: acquiring of the data (scanning of the object), data processing (registration of point clouds, model-building) and the use of the model (making an application through a game engine that enables a virtual walk). For the acquiring of the data (scanning), a historically important object was chosen. Several programs that enable point cloud processing and model-making were used during the data processing. Finally, the simplest way of using a game engine for building a level of a virtual walk around a model object is described. The particular game engine was used to make various video games, many of them quite well known.

ZAHVALA

Za pomoč in podporo pri izdelavi magistrske naloge se iskreno zahvaljujem mentorici doc. dr. Mojci Kosmatin Fras in somentorju asist. dr. Dejanu Grigillu.

Zahvaljujem se tudi geodetskemu podjetju Geocentar d.o.o. z Čakovca, njegovemu direktorju Đurotu Zaloviću, ki mi je dovolil uporabo terestričnega laserskega skenerja in programske opreme, ter zaposlenim za pomoč pri skeniranju in nasvetih pri izdelavi modela.

Zahvaljujem se Muzeju Međimurja Čakovec, njegovemu direktorju mr. sc. Vladimiru Kalšanu, za dovoljenje, da smo lahko skenirali kapelo Svete Jelene.

Zahvalil bi se tudi svojemu bratu Davoru Zelenbabi za sodelovanje pri ustvarjanju ideje za temo magistrske naloge ter moji družini za podporo v času študija.

Ta stran je namenoma prazna.

KAZALO VSEBINE

KAZALO SLIK

KAZALO PREGLEDNIC

Ta stran je namenoma prazna

1 UVOD

Glede na to, da obstaja že veliko bodisi diplomskih bodisi magistrskih nalog na temo laserskega skeniranja, ki opisujejo v večji meri same postopke laserskega skeniranja in načine obdelave takih podatkov, sem se odločil, da v svoji magistrski nalogi, poleg že omenjenega, raziščem in obrazložim uporabo izdelkov, pridobljenih z laserskim skeniranjem, v navidezni resničnosti. V glavnem planu bodo video igre kot možno področje uporabe modelov, pridobljenih iz podatkov laserskega skeniranja, in kot zelo razširjeno področje prikaza navidezne resničnosti. Moderne video igre vse bolj prepričljivo in realno prikazujejo svet in prostor, v katerem se določena video igra dogaja. Glede na to, da gre za prikaz prostora, obstaja povezava naloge z geodezijo, geoinformatiko, fotogrametrijo in daljinskim zaznavanjem ter z večpredstavnostno kartografijo. Na enak način kot sodobne video igre uporabljajo ustvarjen imaginaren prostor, bi lahko uporabljale tudi realen prostor. Eden izmed načinov, kako bi lahko realen prostor prenesli v neko video igro, so zagotovo izdelki iz podatkov 3D laserskega skeniranja. Po drugi strani bi na isti način, kot video igre uporabljajo imaginaren prostor, lahko naredili aplikacije za premikanje (sprehajanje) skozi realen prostor.

Z razvojem tehnologije laserskega skeniranja se razvijajo in širijo tudi področja uporabe te tehnologije. Od že znanih uporab v arheologiji, gradbeništvu in industrijski izmeri, forenziki, varovanju kulturne dediščine, se pojavljajo tudi nova področja uporabe. Eno izmed takih področji je igralniška (angl. gaming) industrija, ki se v veliki meri ukvarja s prikazom navidezne resničnosti. V svoji magistrski nalogi se želim osredotočiti predvsem na ta vidik uporabe. Igralniška industrija je ena izmed najhitrejše rastočih industrij na svetu, s prihodki večjimi kot jih ustvarjata filmska in glasbena industrija skupaj (Statista.com, 2015). Trenutno najnovejše video igre zelo realno prikazujejo prostor, v katerem se igra odvija, pa čeprav je ta ustvarjen le navidezno. Z izdelavo modelov iz podatkov laserskega skeniranja in uporabo le-teh v t.i. "igralnih pogonih", s katerimi se video igre razvijajo, bi lahko dobili možnost virtualnega sprehajanja v okolju iz realnega prostora. Poleg izdelave video iger z uporabo digitalnih podatkov realnega prostora, se ti podatki lahko na isti način uporabijo tudi v aplikacijah za navidezno sprehajanje po prostoru, ki se lahko primerjajo z znano aplikacijo "Google street view". Razlika glede na "Google street view" je v tem, da se pri "Google street view" gibanje po prostoru ustvarja s hitrim premikanjem iz fotografije v fotografijo (t.i. sferna fotografija), odvisno od tega, na kateri lokaciji je fotografija posneta, s t.i. morfingom pa se izgublja občutek stvarnega gibanja. Z uporabo 3D modelov v okolju igralnih pogonov (angl. "game engine") bi dobili občutek pravega gibanja v realnem prostoru. Glede na to, da je navidezno sprehajanje v prikazu realnega prostora eden izmed načinov prikaza prostorskih podatkov, lahko rečemo, da bi z realizacijo uporabe modelov, pridobljenih z laserskim skeniranjem v "Game engine" za video igre dobili virtualne večpredstavnostne karte.

Večpredstavnostne karte, kot so na primer Google maps in že omenjeni Google street view, so danes v večji in širši uporabi, njihova uporaba za simulacijo prostega gibanja v realnem prostoru pa bi zagotovo bila nov korak v razvoju večpredstavnostne kartografije.

Namen naloge je prikazati postopek pridobivanja laserskega oblaka točk z enim izmed najsodobnejših laserskih skenerjev, ki je trenutno na tržišču. Omenjen skener je FARO Focus3D x 330. Zajet oblak točk bomo obdelali in na različne načine izdelali 3D model s teksturami, izdelan model uvozili v igralni pogon ter izdelali aplikacijo za sprehajanje okrog skeniranega objekta. Celotna naloga bo temeljila na kratki teoretični razlagi in s poudarkom na izdelanem praktičnem primeru, kjer bo skeniran izbran zgodovinsko pomemben objekt.

1.1 Hipoteze

V nalogi smo preverili naslednje hipoteze:

- geodezija in geoinformatika ter podrobneje fotogrametrija in daljinsko zaznavanje lahko prispevajo k industriji video iger;

- 3D modeli, izdelani iz podatkov terestričnega laserskega skeniranja, se lahko uporabljajo v izdelavi videoiger, tako da prostor navidezne resničnosti gradijo iz podatkov realnega prostora;

- na način, kot se izdelujejo video igre, se lahko izdelajo tudi multimedijske (večpredstavnostne) karte.

1.2 Struktura naloge

Naloga je sestavljena iz sedmih poglavij in vsebinsko razdeljena na dva dela, in sicer teoretični in praktični. V uvodnem poglavju so opisani ideja in cilji naloge, ter predstavljene hipoteze. V drugem in tretjem poglavju so teoretične razlage, ki so povezane s tematiko naloge. V četrtem poglavju začnemo z opisom praktičnega dela naloge, kjer je opisan postopek pridobivanja podatkov oz. skeniranja, ter nekaj informacij o skeniranem objektu. V petem poglavju je opisan postopek obdelave podatkov, pridobljenih z laserskim skeniranjem, ki se deli na dva dela in sicer, registracija oblaka točk, kjer smo uporabili program proizvajalca laserskega skenerja, in izdelava 3D modela iz oblaka točk, kjer smo uporabili več različnih programov. V šestem poglavlju smo opisali postopek izdelave aplikacije navideznega sprehoda okoli skeniranega objekta oz. modela. V zaključnem (sedmem) poglavju smo potrdili postavljene hipoteze, in podali naše mnenje o uporabljeni tehnologiji.

2 3D MODELI IN MODELIRANJE

Proces ustvarjanja resničnih ali imaginarnih 3D objektov imenujemo 3D modeliranje (Bernik, 2010b). Razvoj računalniške tehnologije omogoča uporabniku za doseganje optimalne učinkovitosti izbiro med raznimi metodami in tehnikami. Izbor je povezan s klasičnim 3D modeliranjem ali 3D skeniranjem s pomočjo programskih in strojnih rešitev. Z uporabo 3D tehnik modeliranja lahko uporabnik ustvari 3D model na več načinov: uporabi lahko poligone, krivulje ali pa hibrid obeh tehnik pod nazivom subdivizijsko modeliranje. Izbor ne določa kakovosti končnega izdelka, vendar pa lahko bistveno vpliva na čas, potreben za izdelavo 3D modela. Vsaka od naštetih tehnik vključuje veliko algoritmov, ki uporabniku omogočajo izdelavo in manipulacijo tako osnovnih enot, kot tudi bolj zapletenih geometrijskih teles (Bernik, 2010b).

Rezultat je 3D model, ki se z upodabljanjem lahko prikaže kot 2D slika, lahko pa se uporabi kot vir za simulacije v realnem času (Bernik, 2010b). "Upodabljanje je proces, pri katerem se interaktivni element 3D računalniške grafike pretvarja v statični 2D objekt, sliko" (Bernik, 2010a).

3D modeli so opisani s 3D podatki, ki se zapisujejo v različnih digitalnih formatih, odvisno od programskih paketov. Večina znanih formatov je medsebojno kompatibilnih, vseeno pa lahko pri pretvorbi 3D formata pride do izgube informacij.

Vsebino 3D podatkov lahko razvrstimo v tri kategorije: geometrija, videz in scena (McHenry, Bajcsy, 2008.). Za samo ustvarjanje in modeliranje modelov je geometrija najpomembnejša, vendar pa imata pri uporabi teh modelov, v tem primeru v navidezni resničnosti, pomembno vlogo tudi preostali kategoriji.

Geometrija modela je večinoma sestavljena iz skupine 3D točk (ali vozlišč). Ploskev modela je v tem primeru ustvarjena iz serije poligonov, ki so sestavljeni iz teh vozlišč. Število vozlišč, iz katerih je ploskev sestavljena, je različno, vendar prevladujejo triangulacijske ploskve. Veliko formatov podpira tudi linije, sestavljene iz dveh vozlišč. Poligonalne mreže so primerne za grafične aplikacije in omogočajo enostaven prikaz 3D vsebin. Pogosto pa poligonalne mreže niso dovolj, zato se kot deli 3D objektov uporabljajo čvrsti volumni (McHenry, Bajcsy, 2008).

Izboljšan videz modelov v največ primerih zahteva lepljenje tekstur na njihovo površino, kar je doseženo s funkcijsko povezavo vsake tridimenzionalne točke z ustrezno točko na dvodimenzionalni sliki. Pri upodabljanju se točkam znotraj ploskev dodelijo informacije o barvi ujemajočih se točk na pripadajoči sliki (fotografiji). Model, ki vsebuje teksture, mora

vsebovati koordinate teh tekstur znotraj 3D podatkov. Večina formatov 3D podatkov podpira teksture. S kombinacijo tekstur in fizičnega videza modela lahko dosežemo zelo realističen videz (McHenry, Bajcsy, 2008).

Scena vključuje videz modela ob upoštevanju kamere kot zornega kota, izvora svetlobe in same okolice modela, ki lahko vključuje tudi druge modele. Za definiranje pogleda na 3D podatke moramo najprej opredeliti položaj kamere v 3D prostoru in vektorje, ki definirajo smer gledanja. Pri opisovanju izvora svetlobe je za videz modela pomembno, kako se bo intenziteta določene ploskve prikazala v odvisnosti od njenega položaja glede na izvor svetlobe. Ploskve, ki so direktno obrnjene k izvoru svetlobe, bodo imele večjo intenziteto, medtem ko bodo ploskve, obrnjene stran od izvora, prikazane v "senci". Izvor svetlobe je v 3D podatkih prav tako opisan s položajem v 3D prostoru in vektorjem smeri svetlobe (McHenry, Bajcsy, 2008).

Ravno za zgoraj navedene podatke, kateri opisujejo scenu je v računalniški grafiki zelo pomemben podatek tudi normala vozlišča. Normala vozlišča je vektor, ki je pravokoten na ploskev, ki vsebuje to vozlišče. Normala vozlišča je v bistvu zamenjava prave geometrijske normale ploskve. Normala vozlišča se izračuna iz normal sosednjih ploskev, ki tvorijo določeno vozlišče. Uporaba teh normal je široka, ampak v glavnem je njihov namen določanje smeri neke ploskve na modelu kot samo smer videza, prav tako pa omogočajo tudi, da se iz te smeri izračuna osvetlitev in senčenje izdelanih modelov v realnem času (Miller, 1999).

2.1 Vrste 3D modelov

Kot že navedeno, 3D modeli vsebujejo podatke o točkah 3D prostora in druge informacije kot sta videz in scena, ki jih računalnik interpretira kot navidezen objekt, ki se izrisuje na zaslonu. Ko govorimo o samih modelih, neodvisno od podatkov, ki jih opisujejo, obstajajo tri medsebojno neodvisne vrste 3D modelov: žični (ang. wireframe), ploskovni (ang. surface) in volumski (ang. solid) modeli (Cheng, 2005).

2.1.1 Žični modeli

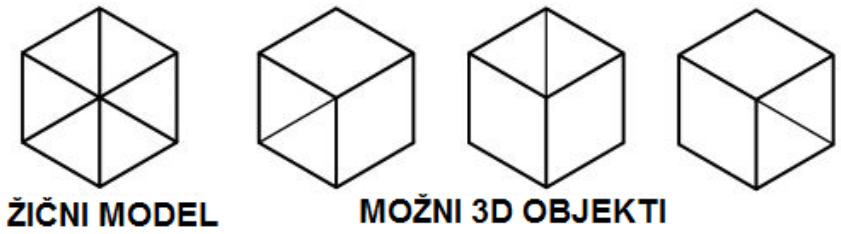
Žični modeli (ang. wireframe models) so najenostavnejša oblika 3D modelov. Niso zahtevni za strojno opremo, so enostavni za modeliranje, vendar pa imajo slabe možnosti za upodobitev.

Žični model je skeletni opis 3D objekta. V žičnem modelu ni površin, sestavljen je le iz vozlišč in ravnih ter ukrivljenih linij, ki opisujejo robove objektov.

"Žični model definira osnovne lastnosti objekta: obris objekta, orientacijo glede na okolje in funkcijo v sklopu prizora. Pri žičnem modelu ni skritih linij, zato je težko na hitro interpretirati geometrijsko obliko. Uporaba žičnega modela temelji na hitrem upravljanju z modelom in potrebi po vpogledu v skrite linije" (Jovanović, 2006). Ker žični modeli nimajo podatkov o površini in obsegu telesa, 3D model pa je prikazan na 2D zaslonu, se model ne more prikazati realno, zato pogosto pride do napačne interpretacije (Cheng, 2005).

Če je določena količina robov shranjena v računalniku pod pogojem, da ti robovi predstavljajo robove resničnega objekta, je pogosto nejasno, za kakšen objekt gre, ker ima lahko več različnih objektov enake robove in elemente (Topić, 2005).

Na sliki 1 so prikazane možne interpretacije žičnih modelov.



Slika 1: Primer možnih interpretacij žičnih modelov (Vir: Cheng, 2005).

2.1.2 Ploskovni modeli

Ploskovni 3D modeli (ang. surface models) so množice ploskev, ki se v 3D prostoru združujejo v objekt. Ploskovni modeli, poleg točk in linij, vsebujejo tudi površine, na katere se lahko lepijo teksture. S pomočjo ploskovnih modelov je lažje definirati vidne grafične entitete (Jovanović, 2006).

Ploskovno modeliranje je bolj zapleteno od žičnega, saj poleg robov objekta vsebuje tudi površine. Površine se večinoma definirajo s poligonalnimi mrežami. Vendar pa se lahko tudi v primeru ploskovnih modelov pojavi dvosmiselnost pri določevanju prostornine 3D objekta.

"Ploskovno modeliranje zadostuje za večino prikazov. Težave se lahko pojavijo pri izvajanju presekov ali podobnih izvodov modela, saj v teh primerih telesa, predstavljena s ploskvami, izgledajo "prazno" oziroma kot da nimajo prostornine. V tem primeru in tedaj, ko želimo

računati razne fizikalne velikosti telesa, je treba uporabiti volumsko modeliranje" (Topić, 2005).

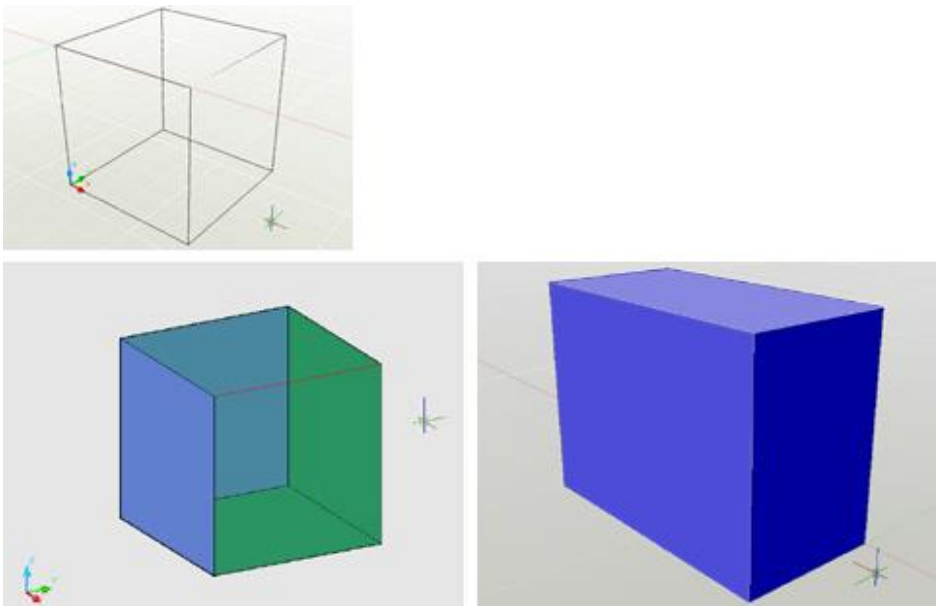
2.1.3 Volumski modeli

Volumski modeli (ang. solid models) so sestavljeni iz osnovnih 3D oblik: kocke, stožci, valji, krogle itd. Lastnost volumskih modelov je, da so popolni in nedvoumni ter imajo obliko zaprtega volumna. Volumski modeli vsebujejo največ podatkov o objektu. Možno je izračunati vse informacije od površin posameznih ploskev do prostornine celotnega telesa. Zaradi teh lastnosti je uporaba volumskih modelov zelo razširjena in znana. Glede na prikaz volumna so volumski modeli lahko prikazani kot »boundary representation« (Brep) in »constructive solid geometrey« (CSG) (Vuoskoski, 1996).

»Boundary representation« prikazuje 3D model kot množico osnovnih elementov (ploskve, robovi in točke). V tej množici je jasno določena hierarhija strukture podatkov: model je definiran s ploskvami, ki so definirane z robovi, ki pa so definirani s točkami (Topić, 2005). Brep predstavlja čvrsti volumen kot "kožo" okoli objekta (Vuoskoski, 1996).

CSG predstavlja čvrsti volumen, sestavljen iz enostavnih volumnov (Vuoskoski, 1996). Metoda CSG prikazuje 3D modele s pomočjo osnovnih 3D geometrijskih oblik, ki se samodejno generirajo v 3D programih (Topić, 2005).

Na sliki 2 so prikazane vse tri vrste 3D modelov.



Slika 2: Vrste 3D modelov – žični (zgoraj), ploskovni (spodaj levo) in volumski (spodaj desno), (Vir: Design Workshop Sidney, 2014)

2.2 Formati 3D modelov

Obstaja več kot 100 znanih formatov za zapis 3D podatkov, ki se uporabljajo v raznih programskih orodjih za različne namene. Formati so zelo pomembni, še posebno, če se nek izdelek izdeluje s pomočjo več različnih programskih paketov, saj različni programi podpirajo samo nekaj formatov. Zato se lahko zgodi, da modela, izdelanega v enem programu, ne moremo uporabiti v drugem, ker formata zapisa nista izmenljiva. Seznam nekaj najbolj znanih formatov zapisa 3D modelov je prikazan v preglednici 1.

Preglednica 1: Seznam nekaj najbolj znanih formatov 3D podatkov (Vir: McHenry, Bajcsy, 2008)

Končnica	**Naziv**
3ds	3D Studio
dwg	Legacy AutoCAD Drawing
dxf	AutoCAD Drawing Exchange Format
fbx	AutoDesk Kaydara FBX
max	3Ds Max
mb	Maya Scene binary
obj	Wavefront

V tej nalogi je za izdelavo aplikacije uporabljen format podatkov obj, ki je eden najbolj znanih in razširjenih formatov ter ga podpira večina priljubljenih programskih paketov. Format podatkov obj ima tekstualno obliko, razvit pa je bil s strani Wavefront Technologies. Vsebina formata obj je sestavljena iz tekstualnih vrstic, ki vsebujejo ključ in različne spremenljivke. Ključ na začetku vsake vrstice prikazuje tip informacije v tej določeni vrstici. V preglednici 2 so prikazani nekateri ključi v obj formatu.

Preglednica 2: Seznam nekaterih ključev, uporabljanih v obj formatu (Vir: McHenry, Bajcsy, 2008)

Ključ	**Opis**
#	Comment
v	Vertex
l	Line
f	Face
vt	Texture Coordinate
vn	Normal
g	Group

Primer zapisa obj formata:

```
v 0.0 0.0 0.0
v 0.0 1.0 0.0
v 1.0 0.0 0.0
f 1 2 3
```

Format obj podpira: geometrijo in oblike vozlov, robov ter ploskev, parametre površin, vozlišč, normal, tekstur, lastnosti materialov in skupin (McHenry, Bajcsy, 2008).

2.3 Načini pridobivanja 3D modelov

Modeliranje 3D modelov v računalniški grafiki pomeni izdelavo modela s pomočjo računalnika. Pri izdelavi 3D modela je treba razlikovati, če se ta model ustvarja oziroma oblikuje iz obstoječih elementov programskih paketov in se na ta način pridobi izmišljen oblikovani 3D model, ali se uporabijo podatki, pridobljeni iz okolja, in se z obdelavo teh podatkov izdela 3D model. Pred pojavom laserskega skeniranja je bila najbolj priljubljena metoda zbiranja podatkov iz okolja za izdelavo 3D modelov bližjeslikovna fotogrametrija (Antolković, 2014). Za izdelavo žičnih 3D modelov se je uporabljala tudi klasična točkovna meritev s pomočjo geodetskih merskih sistemov (tahimetrov).

Ustvarjanje 3D modelov se izvaja z uporabo programskih 3D paketov, na primer 3ds Max in Maya, vendar je v današnjem času vse bolj priljubljena metoda 3D laserskega skeniranja resničnih objektov in izdelava modela iz zajetega oblaka točk.

"Opisovanje tridimenzionalnih lastnosti realnih objektov v našem okolju in njihovo shranjevanje v digitalni obliki sta postali resničnost na mnogih področjih človekove aktivnosti. Poleg tega stalni napredek računalniške tehnologije in njena vse večja razširjenost vodita k povpraševanju po vse večji količini kakovostnih in podrobnih podatkov, zlasti o prostoru človekove aktivnosti in objektih, ki ga obkrožajo. Zaradi njihove kompleksnosti in izjemne raznolikosti, predvsem v geometrijskem smislu, je za kakovostno opisovanje potrebna ogromna količina merjenih podatkov. Takšno količino podatkov ni možno zbrati z ustvarjanjem fizičnega kontakta med merilno napravo in objektom za vsako merjeno točko. Zato se že celo stoletje kot učinkovit in kakovosten način meritve brez neposrednega kontakta z objektom uporablja fotogrametrija" (Lasić, 2008). V zadnjih dvajsetih letih se je uveljavila tudi tehnologija 3D laserskega skeniranja, ki predstavlja popolnoma avtomatizirano in izjemno učinkovito metodo zbiranja prostorskih podatkov.

Za zbiranje podatkov iz okolja z namenom izdelave 3D modelov je metoda laserskega skeniranja najhitrejša in najenostavnejša metoda, vendar pa je obdelava takšnih podatkov še vedno obširna, zahtevna in dolgotrajna ter zahteva veliko zmogljivost računalnika, saj se obdeluje velike količine podatkov.

2.4 Obdelava laserskega oblaka točk

3D lasersko skeniranje ali lidar (ang. Light Detection and Ranging, LiDAR ali LIDAR) je postopek pridobivanja podatkov iz okolja s pomočjo laserskega skenerja, ki deluje na principu oddajanja laserskih žarkov, ki se odbijajo od objektov v okolje in vračajo nazaj v skener. Lastnost skeniranja je pridobivanje velikega števila točk, t. j. nekaj milijonov, v kratkem času. Ta množica točk se imenuje oblak točk. Pridobljene točke imajo določene X, Y in Z koordinate v lokalnem ali referenčnem koordinatnem sistemu, z razvojem tehnologije pa mnoge laserske naprave poleg pridobivanja prostorskih koordinat omogočajo tudi pridobivanje dodatnih informacij o pridobljenih točkah kot sta intenziteta ali barva.

Pri obdelavi laserskih meritev je najprej treba povezati oblake točk, pridobljene z različnih stojišč skenerja, v celoto, kar se doseže s pomočjo registracije. Registracija oblaka točk se najpogosteje izvaja s pomočjo identičnih točk znotraj posameznih oblakov. Z razvojem programskih sistemov se vse pogosteje uporablja samodejna registracija s pomočjo algoritmov, ki prepoznajo geometrijske oblike znotraj oblaka točk, ta postopek pa še dodatno olajšujejo sistemi, ki imajo s pomočjo dodatnih senzorjev v skenerjih že približno prostorsko določeni lokacijo in orientacijo skenograma. Celoten postopek registracije oblaka točk ni zahteven, izvaja se polavtomatsko in je najenostavnejši del obdelave laserskih meritev. Edina omejitev pri tem koraku je povezana s strojno opremo, saj so za registracijo oblaka točk zaželeni zmogljivi računalniki (Miler, Đapo, Kordić, Medved, 2007).

Po izvedeni registraciji, oblak točk georeferenciramo na podlagi izmerjenih in signaliziranih oslonilnih točk, ali pa nadaljujemo obdelavo v lokalnem koordinatnem sistemu.

Najzahtevnejši del obdelave laserskih meritev je sama izdelava modela. Aplikacije za obdelavo podatkov so v stalnem razvoju, vendar načeloma zaostajajo za hitrostjo tehničnega razvoja skenerjev. Cilj obdelave oblaka točk je pridobivanje smislenih ploskev iz množice točk znotraj oblaka. Obdelava podatkov za pridobivanje modela se mora pogosto izvajati ročno, kar pa zahteva veliko časa. Programske pakete, ki omogočajo obdelavo laserskih meritev, večinoma razvijajo sami proizvajalci laserskih skenerjev, zato so ti programi prilagojeni ravno skenogramom, pridobljenih z uporabo njihovih laserskih skenerjev. Vendar pa se v zadnjem času pojavlja vse več neodvisnih programskih paketov, prav tako pa tudi odprtokodna ali programska oprema "Open source", ki jo razvijajo uporabniki s celega sveta in je prilagojena širši uporabi. Danes že lahko najdemo programske pakete, ki popolnoma samodejno izdelujejo 3D modele iz oblaka točk, pri katerih mora uporabnik le postaviti določene parametre, ki definirajo način izdelave modela. Čeprav je na ta način možno zelo hitro in enostavno pridobiti končni 3D model, še vedno obstajajo slabosti, vprašljivi pa sta

tudi kakovost in uporabnost takega modela. Vsekakor je pri izdelavi modela potrebno upoštevati njegov končni namen in v skladu s tem prilagoditi način obdelave.

2.5 Uporaba 3D modelov, pridobljenih z laserskim skeniranjem

"Možnosti uporabe velike količine podatkov, pridobljenih s terestričnimi laserskimi skenerji, so danes zelo razsežne. Stalno povečevanje zmogljivosti modernih računalnikov praktično vsakodnevno odpira nove možnosti" (Lasić, 2008).

Dandanes je 3D lasersko skeniranje, zahvaljujoč velikemu številu informacij in podrobnem prikazu okolja, v zelo široki uporabi. Tehnologija laserskega skeniranja se že v veliki meri uporablja v topografskih meritvah, pri katerih so potrebni podrobnejši prikazi z veliko gostoto in natančnostjo, kot sta npr. meritev arheoloških najdb in izdelava digitalnih modelov kamnolomov. Široko uporabo najdemo tudi v gradbeništvu in industrijskih meritvah, pri katerih sta pomembni natančnost in podrobnost prikaza, zato je obdelava na tem področju najzahtevnejša. Tehnologija laserskega skeniranja se pojavlja tudi v medicini in forenziki. V navedenih primerih ni nujno, da se uporabljajo 3D modeli. 3D modeli, pridobljeni iz podatkov laserskega skenirana, se največ uporabljajo v dokumentaciji in ohranjanju kulturne dediščine, prav tako pa tudi v urbani topografiji ter novih področjih, ki se šele pojavljajo v tej sferi, kot sta animacija in zabava. V primeru dokumentacije in ohranjanja kulturne dediščine je cilj izdelava teksturiranih 3D modelov določenih objektov ali predmetov, ki se dokumentirajo. Pod pojmom "zabava" ima zagotovo velik potencial področje video iger, in sicer v segmentu navidezne resničnosti, saj bi se lahko prostor dogajanja video igre izdelal iz modelov, pridobljenih z laserskim skeniranjem, s čimer bi dobili video igre s prikazom realnega prostora. Ravno ta način uporabe 3D modelov, pridobljenih z laserskim skeniranjem, nas zanima v tej nalogi, v kateri bomo prikazali način, kako se to lahko realizira. Uporaba 3D modelov v navidezni resničnosti še ni tako razširjena in znana, vendar bi bil način izdelave podoben dokumentaciji kulturne dediščine, saj je tudi v tem primeru cilj pridobivanje teksturiranega 3D modela. Razlike bi bile:

- v zahtevani kakovosti končnega modela, ki bi zaradi manjšega obsega obdelave za uporabo v navidezni resničnosti gotovo bila nižja kot v primeru ohranjanja kulturne dediščine;

- pri ohranjanju kulturne dediščine so pomembne tudi najmanjše podrobnosti na objektu oz. predmetu, medtem ko pri navidezni resničnosti temu ni tako, saj obstaja določena stopnja generalizacije;

- pri obsegu podatkov, pri čemer je pomembno, da bi se v primeru navidezne resničnosti uporabljala celotna področja nekega prostora, medtem ko je to pri ohranjanju kulturne dediščine le eden objekt ali predmet;

- in na koncu v ceni ter stroških.

3 GAMING INDUSTRIJA (INDUSTRIJA VIDEO IGER)

"Industrija video iger v sedanjem času predstavlja najdonosnejšo in najbolj razširjeno vejo zabavne industrije. Z razvojem novih tehnologij se stalno išče nove načine, ki vplivajo na razširitev človeškega doživljanja interakcije z računalniškim svetom" (Abdagić, 2011).

Industrija video iger ali interaktivna zabava je področje ekonomije, ki označuje vrsto aktivnosti, kot so razvoj, marketing in prodaja video iger. Je ena od najhitrejše rastočih industrij v Združenih državah Amerike (Bareta, 2013).

Nekaj podatkov o industriji video iger glede na portal Statista.com (2015):

- Vrednost globalnega trga: 1,9 trilijon meriških dolarjev
- Vrednost ameriškega trga: 594,7 milijard ameriških dolarjev
- Prihodki od industrije video iger: 101,62 milijard ameriških dolarjev

Evolucija omenjene industrije ima vse večji vpliv na neposredno povezana področja, kot so industrija strojne opreme, avdio in video industrija, industrija mobilnih telefonov ipd. Geodezija in geoinformatika zaenkrat nimata večjega vpliva, vendar menim, da imata velik potencial za vključitev v doprinos k razvoju video iger, še posebej na področju prikazovanja navidezne resničnosti.

3.1 Navidezna resničnost

"Navidezna resničnost (ang. virtual reality) je skupina tehnologij, s katerimi se uporabnikova slika resničnosti skuša karseda popolno nadomestiti s sliko navideznega okolja. Izdelek, ustvarjen na računalniku, zamenjuje sliko resničnosti. Navidezna resničnost vključuje različne vhodno/izhodne naprave, ki uporabnika neposredno povezujejo z računalnikom, prav tako pa tudi neposredno interakcijo uporabnika in računalnika" (Pandžić, 2004).

Navidezna resničnost daje uporabniku občutek navideznega okolja nekega prostora, kar je doseženo s pomočjo določenega medija, kot so računalniški zaslon, TV sprejemnik ali kino. Ti mediji zagotavljajo 2D prikaz tridimenzionalnega sveta.

Definicija navidezne resničnosti ali virtualne realnosti izhaja iz pojmov "virtualno" in "realnost". "Virtualno" je nekaj, kar je blizu, realno pa je tisto, kar ljudje izkusimo. Navidezna resničnost torej pomeni bližja resničnost. To bi lahko pomenilo kar koli, vendar se običajno nanaša na specifičen način posnemanja resničnosti.

Ljudje svet doživljamo s pomočjo čutil in zaznavanja. Splošno znano je, da ima človek pet čutov: vid, tip, voh, okus in sluh. Vendar pa ima v resnici človek še več čutov, kot je na primer občutek za ravnotežje. Če upoštevamo tako dodatne čute, kot tudi osnovne in njihovo obdelavo ter zaznavo v možganih, dobimo v naših mislih bogat pretok informacij o okolju, ki ga zaznavamo. Vse, kar vemo o naši resničnosti, izvemo preko naših čutil. Z drugimi besedami, celotna izkušnja resničnosti je enostavna kombinacija informacij, dobljenih preko čutil, in njihove smiselne interpretacije v možganih. V skladu s tem, če naša čutila sprejemajo izmišljene informacije, bo, kot odgovor na to, tudi naša zaznava resničnosti prilagojena. Ustvarila se bo predstava o resničnosti, ki v bistvu ne obstaja, vendar je z naše perspektive zaznana kot resnična. To je nekaj, čemur lahko rečemo navidezna resničnost. Lahko bi se reklo, da navidezna resničnost pomeni predstavljanje računalniško ustvarjenega navideznega okolja našim čutilom (Virtual Reality Site, 2015).

3.2 Zgodovina in razvoj industrije video iger

Industrija video iger je zelo mlada, z zametki razvoja v sedemdesetih letih 20. stoletja. Razvoj se je že v petdesetih letih začel kot hobi maloštevilnih programerjev, ki ga v začetku najverjetneje niso dojemali kot vir zaslužka, ampak le kot zabavo in razvedrilo. Čeprav so se prve video igre pojavile že pred tem, se sedemdeseta leta šteje za začetek razvoja industrije video iger, bolj natančno pojav prve komercialne arkadne video igre "PONG", leta 1972. Video igra "PONG", ki je bila simulacija tenisa, je postala prava uspešnica in je bila prodana v 12.000 izvodih ter tako postala prva uspešna video igra. V poslovnem svetu so v tem prepoznali veliko možnost zaslužka, zato so mnoga podjetja začela kopirati igro "PONG" in izdelovati lastne verzije, podobne originalu. To se šteje kot začetek industrije video iger. S pojavom komercialnih video iger so se pojavile tudi prve igralne konzole. Prva hišna igralna konzola za video igre je bila "Magnavox Odyssey", ki jo je kasneje prevzelo podjetje Philips (Saxena, Laze, Kurata, 2009).

V osemdesetih letih so mnoga podjetja začela razvijati lastne igralne konzole, vendar se je samo enemu uspelo prebiti na trg. To je bil japonski "NINTENDO". Poleg tega se je v osemdesetih pojavilo vse več hišnih računalnikov, prihajalo je do razvijanja vrat oziroma priključnih mest (prilagajanje video igre, namenjene enem sistemu, za delovanje na nekem drugem sistemu) in do kopiranja priljubljenih arkadnih video iger. S pojavom hišnih računalnikov se je populariziral tudi razvoj video iger s strani uporabnikov, t. j. omogočeno jim je bilo, da sami "programirajo" video igre na svojih računalnikih. Najbolj znani računalniki tega časa so Commodore, Apple in Tandy. Vse večja zastopanost hišnih računalnikov je privedla do razvoja novih priljubljenih video iger, ki pa so jih prvič začeli razvrščati tudi glede

na zvrst. V tem času se prav tako pojavijo eni najbolj znanih predstavnikov v svetu video iger, kot je Super Mario (Saxena, Laze, Kurata, 2009).

Če so osemdeseta bila leta nagle rasti industrije video iger, lahko devetdeseta označimo za leta zrelosti. Tedaj se pojavi napredna 3D grafika, kar omogoči razvijalcem veliko več ustvarjalne svobode. Pred pojavom 3D video iger je ciljna skupina industrije vključevala predvsem otroke, saj so 2D videoigre vsebovale enostavne naloge z enostavnimi zgodbami in so bile enostavne za igro. S pojavom 3D tehnologije video igre postanejo vse bolj zahtevne in se odpirajo širšemu občinstvu, s tem pa postanejo pomemben dejavnik v globalnem svetu zabave. Poleg tega nova era 3D tehnologije v industriji video iger omogoča nastanek nove zvrsti – prvoosebne video igre (ang. first person), od katerih so najbolj priljubljene strelske prvoosebne igre ali FPS ("first person shooter").

"Sama 3D grafika se začne z enostavnimi, ravnimi, osenčenimi objekti, kasneje pa se vpeljejo še enostavne teksture" (Bareta, 2013).

Pojav 3D tehnologije je zagotovo povzročil revolucijo v industriji video iger. O nadaljnjem razvoju industrije v poznih devetdesetih in vse do danes je veliko za napisati, od pojava mobilnih iger in razvoja mobilne industrije pa do pojava spletnih iger z razvojem spleta in olajševanjem pristopa vse večjemu številu uporabnikov. Vendar pa razvoj 3D tehnologije ostaja eden najpomembnejših dejavnikov v sodobni industriji video iger. Od njenega pojava do danes se je prikaz okolja oz. prostora, v katerem poteka dogajanje določene video igre, bistveno spremenil. V primerjavi s prvotnimi, enostavnimi prikazi 3D prostora, imajo današnje video igre že tako prepričljiv prikaz okolja, da le-to deluje kot resnično – od tod tudi izraz "navidezna resničnost". Primerjava je prikazana na slikah 3 in 4.

Primera:

Slika 3: Videz ene od prvih prvoosebnih strelskih iger na začetku razvoja 3D grafike (video igra: Doom), (Vir: Flaterco.com)

Slika 4: Videz modernih video iger v dizajniranem 3D prostoru (video igra: GTA V), (Vir: GameSpot, 2015)

Prav tako se pojavlja vse več video iger, ki imajo dogajanje postavljeno v prikaz resničnih prostorov. Izdelava takšnih iger se danes dosega na več načinov. Eden od običajnih načinov je fotografiranje prostora po pravilih terestrične fotogrametrije in izdelava modela iz večih fotografij ali lepljenje fotografij na približno oblikovane 3D modele. Že za obstoječe načine izdelave takih video iger lahko rečemo, da v njih posredno sodelujeta geodezija in geoinformatika, bolj natančno fotogrametrija. Primerov, v katerih bi bili uporabljeni modeli, pridobljeni s pomočjo terestričnega laserskega skeniranja, še ni oz. tudi če obstajajo, zaenkrat še niso komercialni in objavljeni.

Eden od primerov video iger z dogajanjem v prikazu resničnega prostora je "Gas Guzzlers: Extreme", ki so ga razvili zagrebški razvijalci "Gamepires". Dogajanje je postavljeno v navideznem strogem centru Zagreba (slika 5).

Slika 5: Primer video igre z navideznim prikazom resničnega prostora: Trg Bana Jelačića v Zagrebu v video igri Gas Guzzlers: Extreme (Vir: VGBlogger, 2014)

3.3 Delitev video iger

Obstaja več načinov delitve video iger:

- glede na zvrst: akcija, avantura, arkada, strategija, športna, simulacija itd.
- glede na podzvrst: otroške/izobraževalne, konstrukcijske, glasbene, besedilne itd.
- glede na tematiko: abstraktne, antične, fantazijske, futuristične, srhljive, sedanjost itd.
- glede na grafiko: 2D, 3D, besedilne
- glede na zorni kot (ang. point of view): prvoosebne (ang. first person), tretjeosebne (ang. third person) in nedoločeno (n/a)
- mnoge druge delitve

Za temo te naloge je najpomembnejša delitev glede na zorni kot (POV) in sicer prvoosebne igre. Aplikacija, katere izdelava je opisana v tej nalogi, je izdelana kot prvoosebna metoda (Wolf, 2001).

3.3.1 Prvoosebne videoigre

Prva oseba ali "First Person" v video igrah pomeni grafično perspektivo zornega kota lika, s katerim igralec upravlja. Prva oseba se pojavlja v mnogih primerih in zvrsteh video iger, od avantur, v katerih je igralec v perspektivi lika (človeka ali izmišljenega lika), do simulacij voženj in letenja, v katerih prvoosebnost pomeni kokpit vozila oz. letala. Vsekakor pa je najbolj priljubljena in razširjena zvrst, v kateri se uporablja prvoosebnost, FPS ("first-person-shooter") ali prvoosebne strelske igre, pri katerih ima grafična perspektiva neizmeren vpliv na igranje (Wikipedia, 2015b).

Prvoosebne video igre večinoma temeljijo na avatarju, pri čemer se na zaslonu prikazuje tisto, kar bi igralčev avatar videl s svojimi očmi.

"Avatar je "objekt", ki predstavlja utelešenje uporabnika ali t. i. alter ego uporabnika v dvodimenzionalnem ali tridimenzionalnem svetu. Uporablja se v računalniških igrah, na spletnih forumih in drugih mestih" (Wikipedia, 2015a).

Značilnost prvoosebnih video iger vseh zvrsti je ta, da se dogajajo v 3D prostoru, ki je večinoma namišljen in oblikovan. Igralec se prostovoljno premika v tem prostoru s pomočjo ukazov preko igralnih konzol ali tipkovnice in miške. Večina teh video iger prikazuje tudi tlorisni načrt oz. zemljevid tega prostora, nekatere pa imajo celo možnost navigacije skozi prostor in izračune oddaljenosti ter časa potovanja, kot da bi bil ta prostor resničen (slika 6).

Slika 6: Primer video igre s prostovoljnim gibanjem skozi oblikovani prostor (Video igra: Need For Speed - Most Wanted), (Vir: 3.bp.blogspot.com, 2015)

Kot je razvidno iz primera na sliki 6, mnogo video iger prikazuje situacije iz resničnega življenja v svetu navidezne resničnosti, v tem primeru vožnjo po mestu z zemljevidom in navigacijo, ki olajšujeta orientacijo v oblikovanem, namišljenem prostoru. Ta zemljevid je lahko tudi resničen zemljevid nekega resničnega mesta, medtem ko je navidezni prikaz vožnje skozi mesto lahko prikaz vožnje skozi resnično mesto, če bi video igro izdelali s pomočjo 3D modelov, pridobljenih s terestričnim laserskim skeniranjem. Prvoosebne video igre se izdelujejo s pomočjo programov, ki jim rečemo "igralni pogoni" (ang. game engine).

3.4 Igralni pogoni

Igralni pogoni (ang. game engine) so programski sistemi, dizajnirani za ustvarjanje in razvoj video iger. Razvijalci iger jih uporabljajo za izdelavo video iger za igralne konzole, mobilne naprave in računalnike. Igralni pogon združuje mnoge funkcije, potrebne za izdelavo video iger, kot so: upodabljanje 2D in 3D grafike, ustvarjanje zvoka, ustvarjanje simulacije sistema v skladu s fizikalnimi zakoni, programiranje, animacija, umetna inteligenca, zaznavanja trčenja med objekti itd. Igralne pogone v glavnem razdelimo glede na vrsto upodabljanja in sicer na 2D in 3D sisteme (Gamecareerguide.com, 2008).

Upodabljanje se izvaja s pomočjo računalniško ustvarjenih modelov. Namen igralnega pogona je, da uporabniku ni treba skrbeti, kako se bo model prikazal na zaslonu, ampak za prikaz geometrije, tekstur, osvetlitve in senc tega objekta. Večina igralnih pogonov ima možnost izdelave modelov znotraj samega programa, medtem ko imajo nekateri omogočen

tudi uvoz modelov, izdelanih v 3D programih za modeliranje. Pri uvozu 3D modelov je pomemben dejavnik izmenljivost formata 3D objekta z določenim igralnim pogonom (Wikipedia, 2015c).

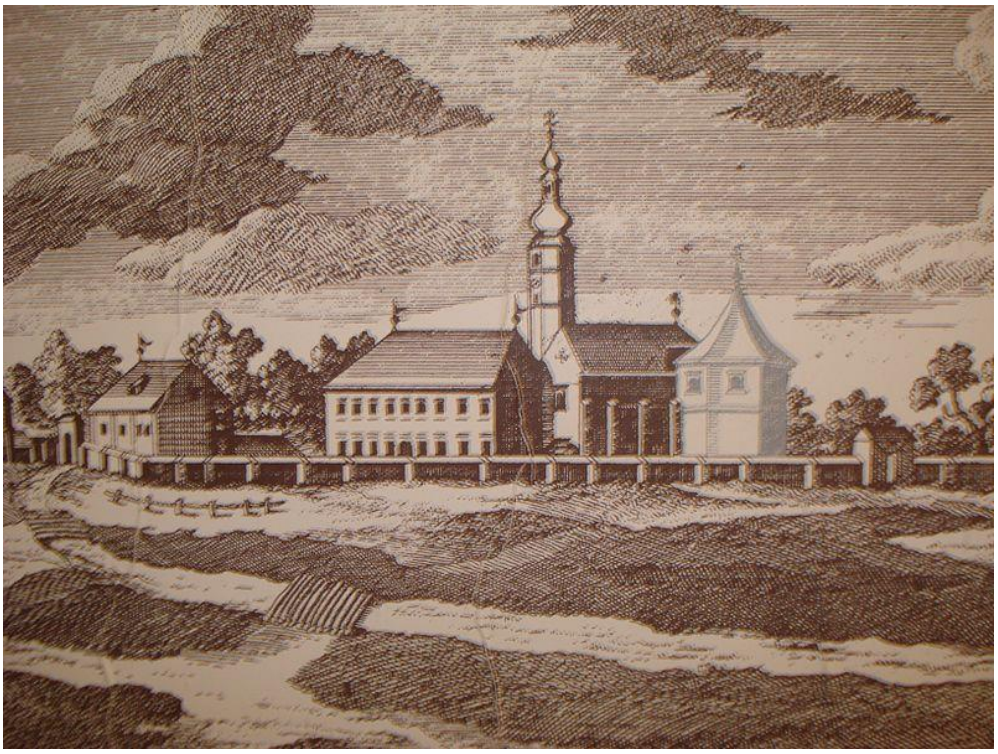
Nekateri trenutno najbolj znani igralni pogoni so: RAGE Engine, CryEngine, Naughty Dog Game Engine, The Dead Engine in Unreal Engine (IGN.com, 2009).

V tej nalogi smo uporabili Unreal Engine.

4 SKENIRANJE OBJEKTA

4.1 Kapelica Svete Jelene

Kapelica Svete Jelene v Šenkovcu je bila cerkveno svetišče pavlinskega samostana, posvečenega Blaženi Devici Mariji in Vsem svetim, ki ga je leta 1376. osnovala plemiška družina Lacković. Izgradnjo samostana je nadaljevala družina Celjskih grofov, ki je postala lastnik leta 1404. Od 1546. leta je bila gospodar Međimurja družina Zrinski, ki je spremenila ime samostana v Svetojelenski in ob južnem crkvenem zidu postavila heksagonalno kapelo za pokop njenih članov. Protestantski nemiri leta 1570 so bili vzrok poškodovanju samostanskega kompleksa, leta 1695 pa se je zgodil velik požar, po katerem je cerkev obnovljena v baročnem stilu. Močan potres leta 1738 je povzročil nove poškodbe, po katerih so sledili popravki in zadnji večji posegi na stavbi. Samostansko posest je prevzela Kraljevska komora. V času Napoleonovih vojn je bil samostan uporabljan kot vojno skladišče, cerkev pa kot pekarna. Leta 1802 je dotedanja pavlinska posest prešla v last barona Vinka Kneževića, ki je preuredil samostan v družinski dvorec in pri tem zrušil dotrajano cerkveno ladjo ter bivšo kapelo Zrinskih, cerkveno svetišče pa preuredil v dvorsko kapelico. Leta 1859 je tedanji lastnik posesti, grof Jurij Feštetić, dal zrušiti celoten dotrajal samostan, z izjemo jugovzhodnega krila, ki pa je bilo uničeno v potresu l. 1880" (Webograd, 2015). Na sliki 7 je videz nekdanjega samostanskega kompleksa v 18. stoletju.



Slika 7: Svetojelenski pavlinski samostan v 18.st. (Vir: Wikipedia, 2015d)

Danes je kompleks pavlinskega samostana Svete Jelene v Šenkovcu zapleteno arheološko najdišče, sestavljeno iz ohranjene arhitekture nekdanjega svetišča cerkve (danes Kapelica Svete Jelene) in arheoloških ostankov arhitekture samostana, cerkvene ladje ter grobnice Zrinskih (slika 8). Kapelica je obnovljena in preurejena v muzej, najdišče pa je v celoti arheološko raziskano (Muzej Međimurja Čakovec, 2015).

Slika 8: Sedanji videz kapelice Svete Jelene

4.2 FARO Focus3D x 330

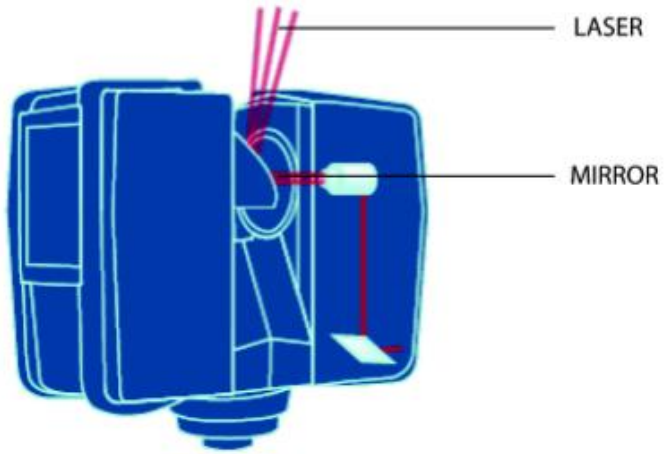
FARO Focus3D x 330 (slika 9) je fazni panoramski laserski skener z velikim dometom. Focus 3D uporablja lasersko tehnologijo za izdelavo podrobnih tridimenzionalnih prikazov okolja in geometrijskih oblik v kratkem času. Izdelani prikazi so množice nekaj milijonov točk v 3D prostoru.

Slika 9: FARO Focus 3D X 330 (Vir: FARO Technologies Inc., 2015a)

Najpomembnejše lastnosti tega skenerja so:

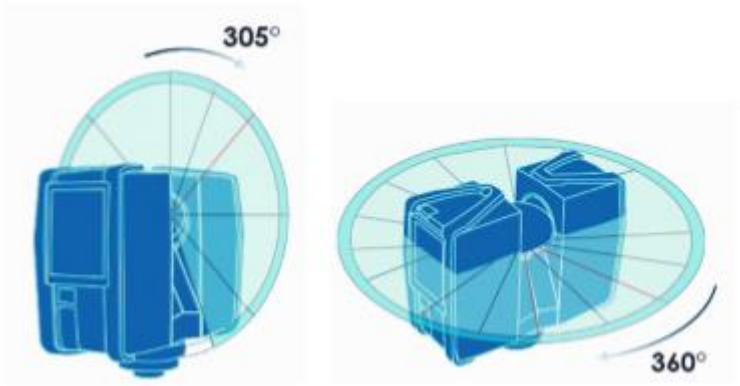
- hipermodulacija
- visoka natančnost
- visoka ločljivost
- velika hitrost
- intuitivno upravljanje preko zaslona, občutljivega na dotik (ang. touchscreen display) velika mobilnost, male dimenzije, majhna teža in integrirana baterijo, ki se lahko polni
- fotorealistični 3D obarvani skenogrami, pridobljeni s pomočjo integriranega barvnega fotoaparata
- integrirani dvoosni kompenzator za samodejno horizontiranje skeniranih podatkov
- integrirani kompas in altimeter za pridobivanje orientiranih skenogramov z višinskimi informacijami
- WLAN za daljinsko upravljanje
- integrirani GNSS sprejemnik za olajšan proces registracije in določevanje točnega časa ter lokacije uporabnikovega skenograma

Focus 3D skenerji praviloma delujejo tako, da v center rotirajočega se zrcala oddajajo infrardeči laserski žarek (slika 10). Zrcalo odbija laserski žarek z rotiranjem v vertikalni ravnini, pri čemer se skener vrti okoli vertikalne osi. Žarki se odbijajo od okolja in vračajo nazaj v skener.



Slika 10: Način snemanja skenerja (Vir: FARO Technologies Inc., 2015a)

Merjenje razdalje s tem skenerjem deluje na principu merjenja razlike v fazi med oddanim in sprejetim valovanjem. Hipermodulacija zmanjšuje razmerje med šumom in signalom (ang. signal to noise ratio) s pomočjo posebne modulacijske tehnologije. Iz pridobljenih razdalj in izračunanih trenutnega položaja rotacijskega zrcala, iz katerega se dobi vertikalni kot, in trenutnega horizontalnega položaja skenerja, iz katerega se dobi horizontalni kot, se izračunajo 3D koordinate za vsako točko. Razdalja, vertikalni in horizontalni kot tvorijo polarne koordinate (δ, α, β), ki se transformirajo v kartezični koordinatni sistem (x, y, z). Skenerjevo vidno polje snemanja (ang. field of view) je 360° x 305° (slika 11) (FARO Technologies Inc., 2015a).



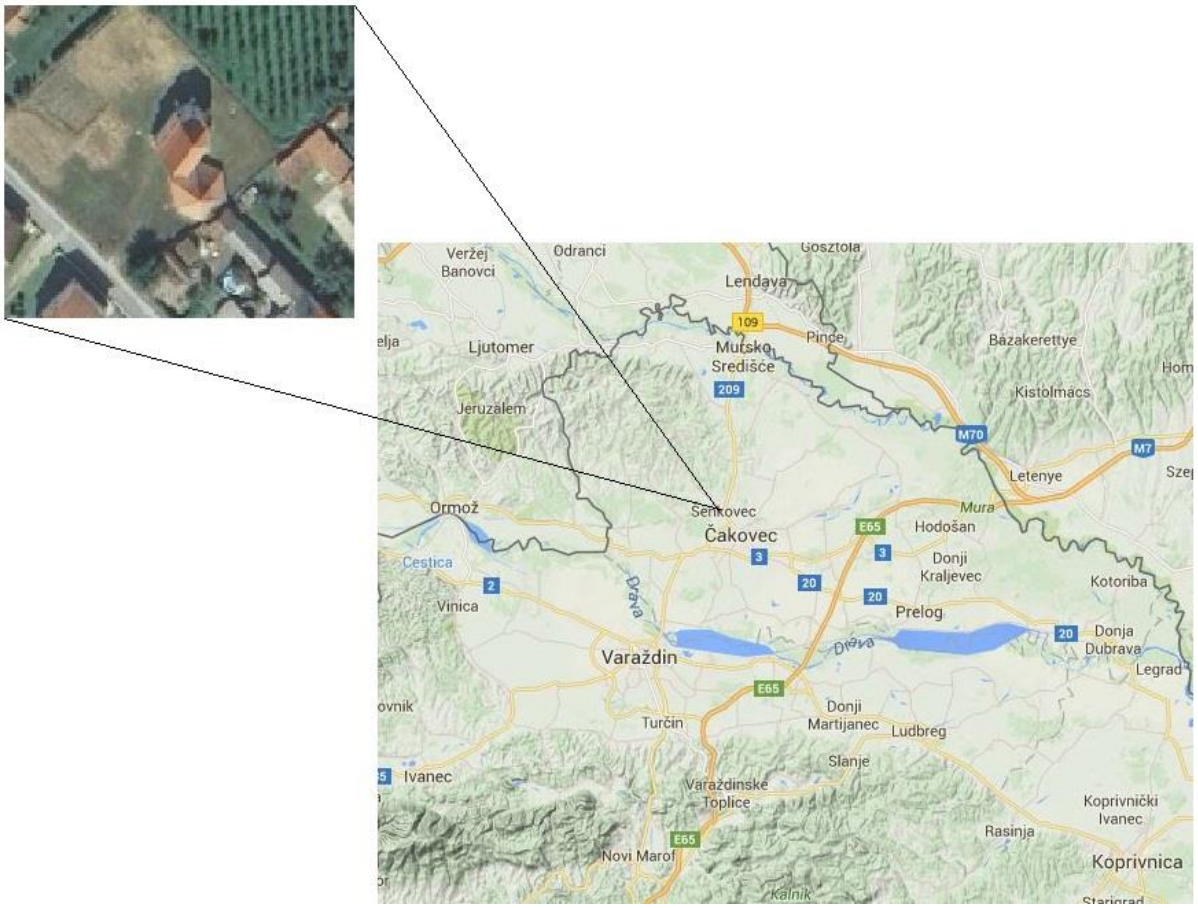
Slika 11: Vertikalna in horizontalna rotacija skenerja (Vir: FARO Technologies Inc., 2015a)

Preglednica 3: Tehnične specifikacije skenerja (Vir: FARO Technologies Inc., 2015a)

Domet skenerja	0.6 - 330m
Hitrost skeniranja	do 976000 točk v sekundi
Natančnost	+/- 2mm
Integrirana barvna kamera	70 mpx
Teža	5 kg
Multisenzor	GNSS, kompas, altimeter, dvoosni kompenzator
Velikost	240 x 200 x 100 mm
Upravljanje s senzorjem	Zaslon, občutljiv na dotik in WLAN

4.3 Opis terenskega dela

Laserska meritev objekta je bila izvedena 16.05.2015. v Šenkovcu, na lokaciji kapelice Svete Jelene (slika 12). Meritev je bila izvedena iz šestnajstih stojišč. Skupno trajanje meritve je bilo tri in pol ure.



Slika 12: Lokacija skeniranega objekta

Nastavitve skenerja so bile privzete (ang. default):

- Profile: Outdoor ...20 m
- Resolution: 28.0 Mpts
- Quality: 4 x
- Point Distance: 7.670 mm/10m
- Scan Size: 8192 x 3414 [pt]

Z razvojem tehnologije postaja zbiranje podatkov vse enostavnejši in hitrejši del skupnega dela, medtem ko mora uporabnik za opravljanje tega dela uporabiti vse manj znanja. Naprednost tehnologije omogoča uporabo "črne škatle". Glede na to, da nastavitev skenerja nismo spreminjali, ampak smo uporabili vse privzete nastavitve, vključno z vklopljenostjo vseh senzorjev, je skeniranje predstavljalo enostavno delo in sicer: postavitev inštrumenta na stativ, vključitev inštrumenta in pritisk tipke "Start Scan" na zaslonu (slika 13). Število in lokacije stojišč snemanja smo izbrali neposredno na terenu, brez predhodnega načrtovanja, s čimer smo želeli dobiti prekomerno število meritev in čim bolj podroben oz. zgoščen skenogram, saj se nepotrebni presežek lahko v kasnejši obdelavi odstrani. Pri skeniranju za vezne točke nismo uporabili markiranih oznak, saj program SCENE, s katerim smo izvedli registracijo, omogoča samodejno registracijo. Prav tako nismo uporabili niti oslonilnih točk, saj nam za končni izdelek v tem primeru ni bil potreben georeferencirani oblak točk, ampak so bili pridobljeni podatki v lokalnem koordinatnem sistemu, kar nam je za namen izdelave te naloge zadoščalo.

Slika 13: Zaslon skenerja

5 OBDELAVA PODATKOV

Obdelava podatkov, pridobljenih z laserskim skeniranjem, je bila v našem primeru sestavljena iz dveh delov: združevanje (registracija) oblaka točk in izdelava 3D modela iz oblaka točk. Kar se tiče same obdelave podatkov terestričnega laserskega skeniranja, je registracija najenostavnejši in najhitrejši del obdelave. Ta proces se z razvojem tehnologije in podpornih programskih paketov vse bolj avtomatizira in poenostavlja ter zahteva vse manj časa, še posebno v primeru, če gre samo za registracijo v lokalnem koordinatnem sistemu, brez georeferenciranja. Edina možna težava oz. zahteva pri registraciji je zmogljivost računalnika, s katerim se izvaja proces, saj je potrebno obdelati ogromne količine podatkov. Kar se tiče programskih paketov za registracijo skenov, ima vsak proizvajalec skenerjev razvit tudi svoj program za registracijo in georeferenciranje skenov, zato uporabniku ni treba skrbeti, kateri program bo izbral za opravljanje tega koraka obdelave. Pri skenerju Faro je ta program SCENE. Veliko takšnih programov nudi tudi nadaljnjo obdelavo laserskih skenov, kar vključuje tudi izdelavo 3D modelov, vendar jim to ni primarni namen in zanj niso izpopolnjeni. Uporabnik ima odprto pot pri izbiri programskega paketa in načina obdelave, odločitev pa je odvisna od namena končnega izdelka.

Izdelava modela je najbolj zahteven in dolgotrajen proces celotnega dela, še posebej, če je cilj pridobiti čim bolj podroben in kakovosten model. Čeprav danes že obstajajo programski paketi, ki omogočajo samodejno izdelavo modelov, imajo tako izdelani modeli pomanjkljivosti glede uporabnosti in verodostojnosti prikaza originala. Glede na to, da je tema te naloge modeliranje podatkov, je velik delež praktičnega dela vključeval iskanje načina izdelave 3D modelov iz oblaka točk, ki bo uporaben v izbranem igralnem pogonu. Prva težava, ki se pojavi pri modeliranju podatkov terestričnih laserskih skeniranj je ta, da so aplikacije za izdelavo modelov v stalnem razvoju in večinoma zaostajajo za tehničnim razvojem laserskih skenerjev in njihovimi zmožnostmi. Druga možna težava pa je ta, da je to še vedno relativno nova in mlada tehnologija in posledično izobraževanje o izdelavi modelov ter uporabi takšnih aplikacij še vedno ni širše uvedeno v visokošolske sisteme. Znanje s tega področja se tako drago zaračunava in skrbno hrani, zato je izobraževalne materiale za uporabo posameznih programskih paketov skorajda nemogoče pridobiti brezplačno. Prav tako ima pri izdelavi modelov zelo pomembno vlogo zmogljivost računalnika, s katerim se model izdeluje. Programski paketi za modeliranje podatkov so izredno zahtevni, kakovost in podrobnost končnega izdelka pa sta odvisni tudi od zmogljivosti računalnika, t. j. strojne opreme, in ne samo od načina obdelave podatkov.

Zaradi vsega navedenega je praktični del tega magistrskega dela temeljil na raziskovanju oz. samostojnem iskanju enostavnega in relativno hitrega načina izdelave modelov iz skeniranih

podatkov, ki bo uporaben za izdelavo končne aplikacije, s čimer bi potrdili zastavljene hipoteze. V tem delu raziskovanja smo preizkusili več programskih paketov, ki omogočajo modeliranje oblaka točk in vizualizacijo modelov oz. dodajanje tekstur. Od vseh preizkušenih programskih paketov smo se odločili v magistrsko delo vključiti tri izbrane programe: Kubit PointSense Heritage, Pointfuse in Meshlab. Končni cilj je pridobiti vizualno prepoznaven model, medtem ko nam zaradi prej navedenih dejstev podrobnosti na končnem modelu in končna vizualna kakovost modela nista pomembni. S potrditvijo zastavljenih hipotez bomo prikazali način, s katerim je to možno izvesti, z uporabo zmogljivejših računalnikov pa je seveda možno dobiti vizualno bolj kakovostne rezultate.

Ker ima zmogljivost računalnika pomembno vlogo v obdelavi podatkov, pridobljenih z laserskim skeniranjem, sta v spodnjih preglednicah (Preglednici 4 in 5) prikazani konfiguraciji strojne opreme računalnikov, uporabljenih pri praktičnem delu te magistrske naloge. Registracija oblaka točk v programu SCENE, ki je računalniško najzahtevnejši del celotne obdelave, je bila izvedena na izposojenem računalniku, preostanek obdelave pa sem izvedel na lastnem računalniku.

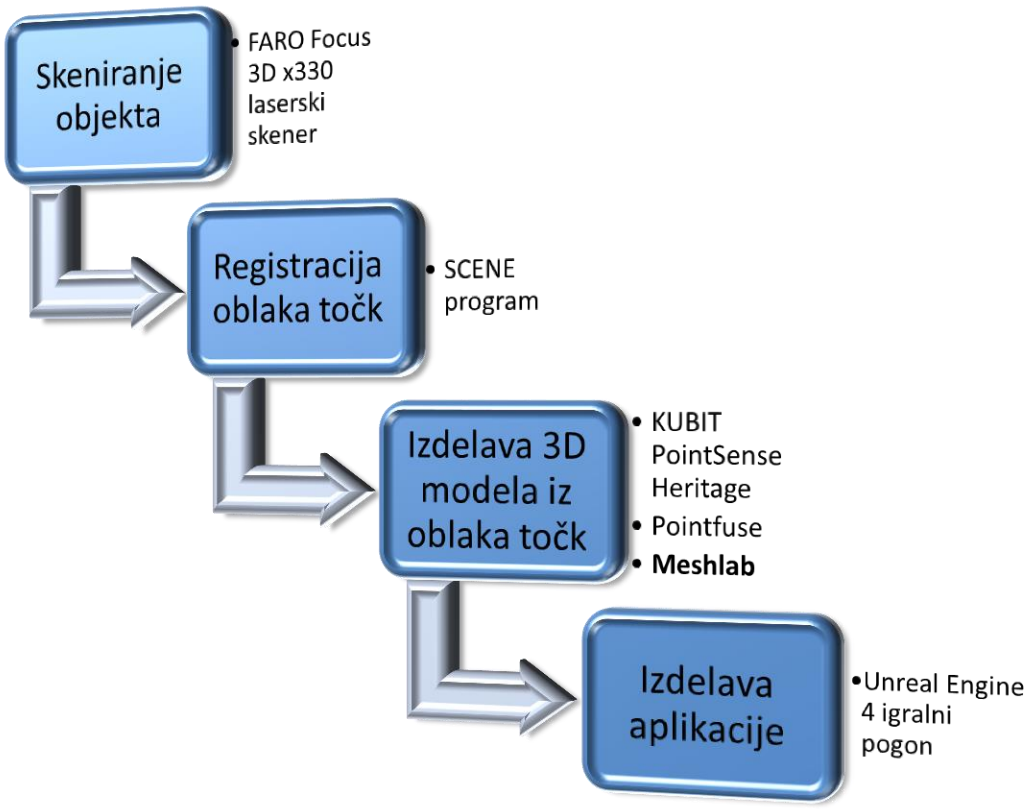
Preglednica 4: Konfiguracija strojne opreme računalnika, uporabljenega za registracijo oblaka točk

Komponenta	**Specifikacija**
Procesor	2.6 GHz Quad-core Intel i7
Delovni pomnilnik (RAM)	16GB 1600MHz DDR3L SDRAM
Grafična kartica	NVIDIA GeForce GT 750M (2GB)

Preglednica 5: Konfiguracija strojne opreme računalnika, uporabljenega za izdelavo modela in aplikacije

Komponenta	**Specifikacija**
Procesor	AMD Athlon II X4 640 3.0 GHz Quad-core
Delovni pomnilnik (RAM)	Adata DDR3 4GB 1600MHz
Grafična kartica	AMD Radeon HD 6850 (1GB)

Grafična shema korakov izdelave naloge:



5.1 Registracija oblaka točk v programu SCENE

Program SCENE je obsežno programsko orodje za obdelavo in upravljanje s 3D oblakom točk. Še posebej je namenjen pregledovanju, administraciji in delu z obsežnimi 3D skeniranimi podatki, pridobljenimi z visokoločljivostnimi 3D skenerji, kot so npr. skenerji iz serije FARO Focus3D X. SCENE na zelo učinkovit in enostaven način obdeluje in upravlja s skeniranimi podatki ter nudi širok razpon funkcij in orodij, kot so filtriranje, samodejno prepoznavanje objektov, registracija skenov in samodejno barvanje skenov.

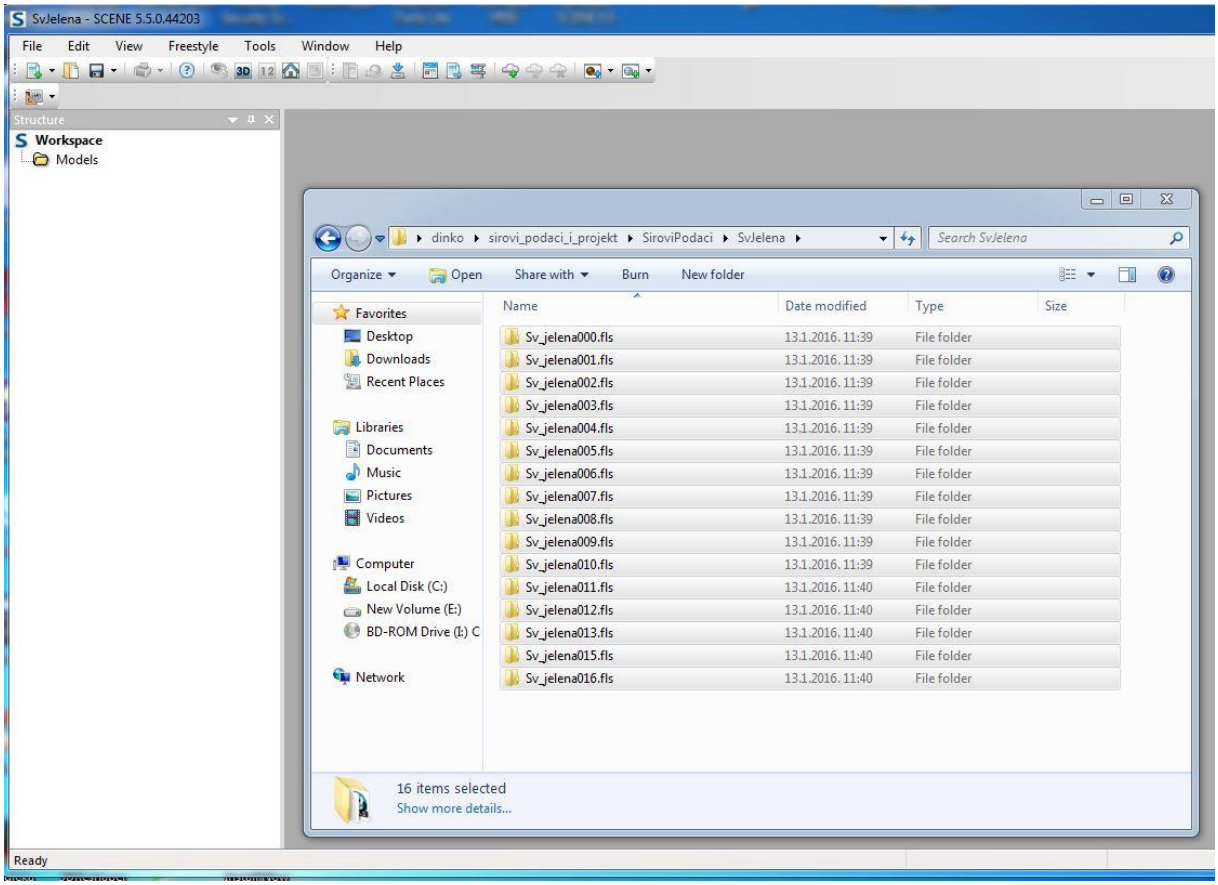
Registracijo oblaka točk lahko razdelimo v več korakov. Pomembno je, da se projekt po vsakem koraku shrani, ker program nima možnosti "razveljavi" (ang. undo) (FARO Technologies Inc., 2015b).

5.1.1 Ustvarjanje projekta in nalaganje podatkov skeniranja

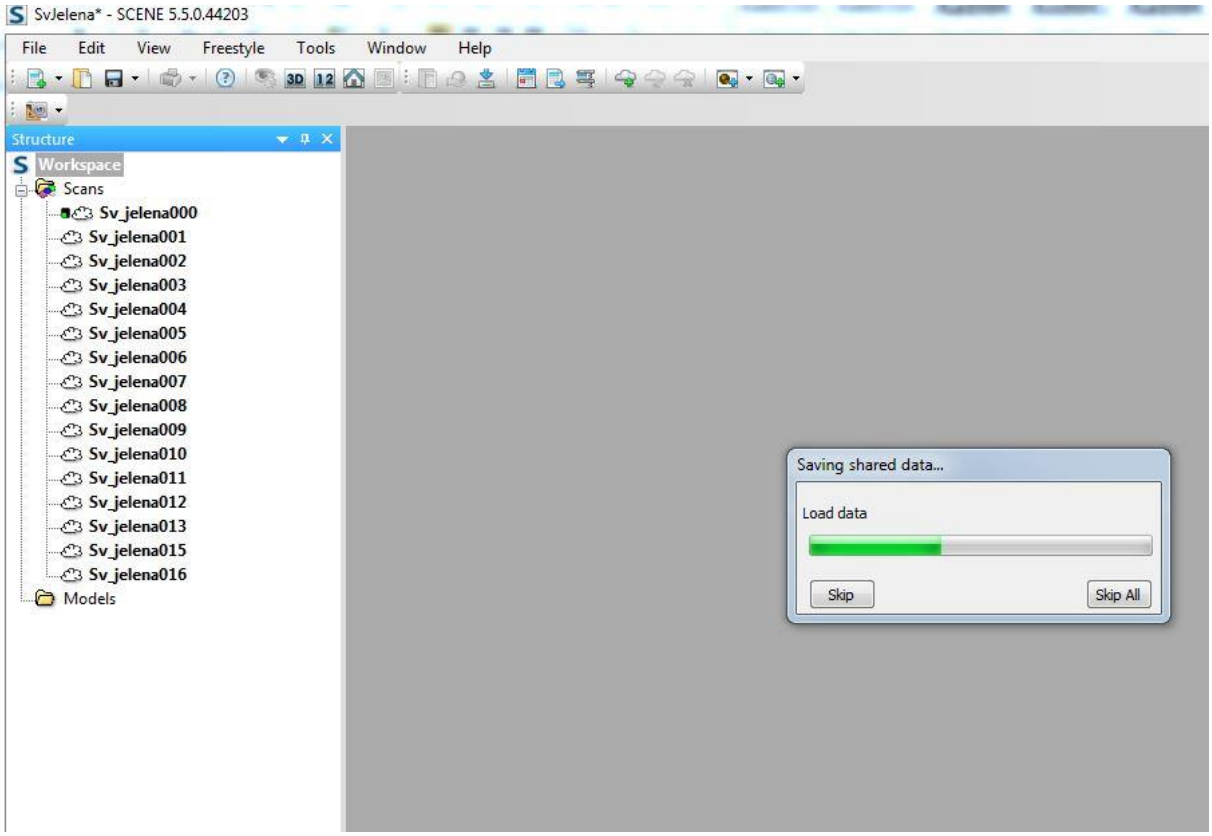
V *Workspace*-u programa SCENE moramo najprej ustvariti nov projekt s pomočjo ukaza:

File – New – Project.

Pri tem nas program vpraša po lokaciji shranjevanja projekta in imenu projekta, po vnosu le-teh pa je s pomočjo ukaza "Create" projekt ustvarjen ter lahko naložimo podatke skeniranja. Podatki skeniranja so shranjeni v mape, in sicer na način ena mapa – eno stojišče. Vnos podatkov lahko izvedemo na dva načina, t. j. s pomočjo ukaza *File – Open* in označimo datoteke glede na stojišče, ali pa na enostavnejši način, da najprej označimo vse datoteke skeniranja glede na stojišče in jih s pomočjo miške "povlečemo in spustimo" v *Workspace* programa SCENE (slika 14). Skenirani podatki se na ta način samodejno prikažejo v programu, razvrščeni glede na stojišče, vendar zaenkrat še niso naloženi. Projekt je potrebno shraniti in šele tedaj se podatki skeniranja naložijo, kar lahko traja nekaj časa (slika 15) (FARO Technologies Inc., 2015b).



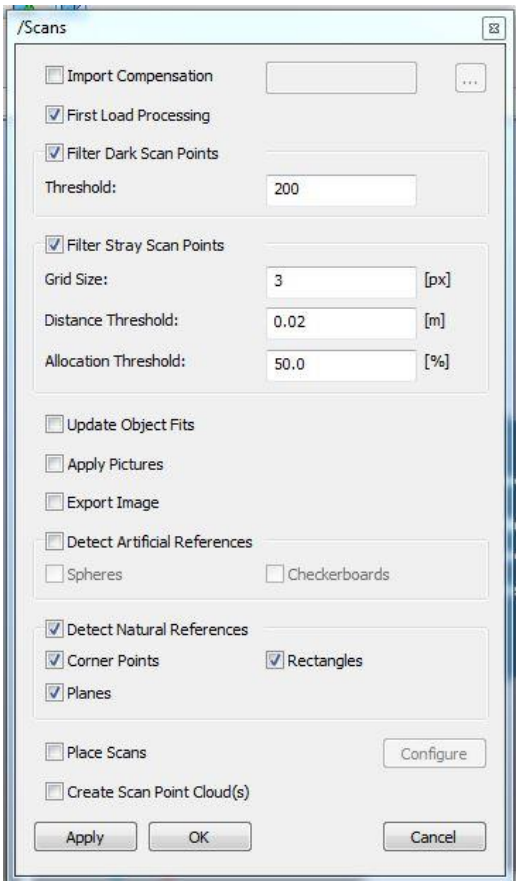
Slika 14: Vnos podatkov skeniranja v *Workspace* programa SCENE



Slika 15: Nalaganje podatkov posamezno glede na stojišče

5.1.2 Samodejna registracija

Vse ukaze, ki želimo izvesti z naloženimi skenogrami, izvedemo tako, da izvedemo desni klik miške na ikono *Scans* v *Workspace*-u. Prvi korak združevanja oblaka točk je predprocesiranje s pomočjo ukaza *Operations – Preprocessing – Preprocess Scans.* Ta ukaz je način skupne obdelave, s katero se izvaja več korakov obdelave skeniranih podatkov hkrati. S predprocesiranjem filtriramo skenograme, zaznamo in prepoznamo objekte ter iščemo skupne točke na skenogramih, ki so lahko markirane oznake na terenu ali v našem primeru izpostavljene podrobnosti na skeniranem okolju. Z izvedbo ukaza predprocesiranja se nam odpre okno, v katerem smo nastavitve nastavili kot je vidno na sliki 16.



Slika 16: Nastavitve predprocesiranja

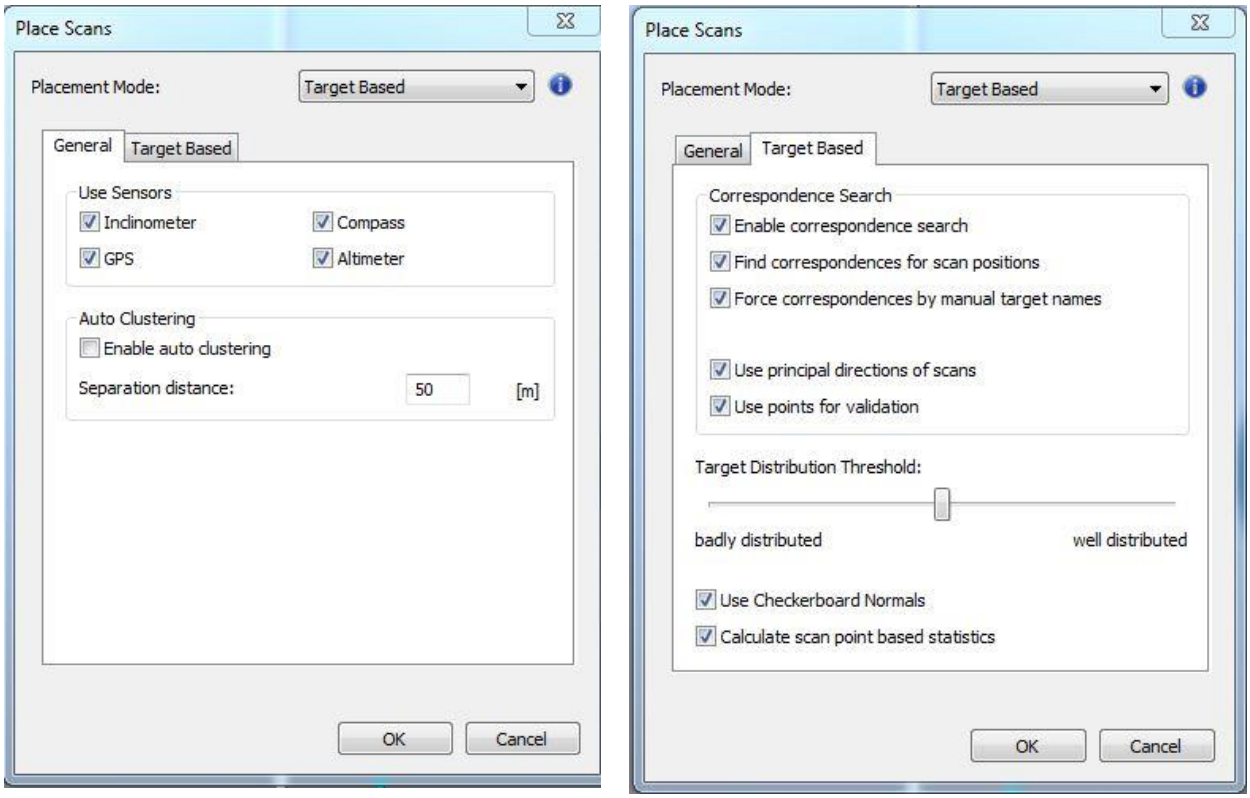
S pomočjo nastavitve *Firts Load Processing* izvedemo filter, ki je samodejno privzet v nastavitvah programa in je priporočen za prvo nalaganje skenogramov. Filter vključuje *Filter Dark Scan Points*, ki filtrira točke glede na intenziteto, in *Filter Stray Scan Points*, ki filtrira t. i. osamelce. Pri predprocesiranju izbiramo tudi način prepoznavanja skupnih točk, ki je lahko s pomočjo markiranih oznak ali sfer – možnost *Detect Artificial References*, ali s pomočjo naravnih podrobnosti kot so robne točke, ploskve in pravokotniki (možnosti *Corner Points, Planes, Rectangles*) z možnostjo *Detect Natural References*, ki smo jo tudi označili. Celoten

postopek predprocesiranja lahko traja nekaj časa, odvisno od zmogljivosti računalnika. Po končanem predprocesiranju je potrebno projekt ponovno shraniti, s čimer sprejmemo spremembe.

Naslednji korak je združevanja skenogramov s pomočjo registracije. Program omogoča več načinov registracije, ki jih lahko tudi kombiniramo, da bi dobili čim bolj natančne rezultate. Registracijo smo izvedli na dva načina:

- Prva registracija "*Target Based*" s pomočjo skupnih točk, najdenih med predprocesiranjem
- Druga registracija "*Cloud to Cloud*", za katero je predhodno potrebno izvesti katerokoli vrsto registracije, da dobimo približno združene skene

Registracijo skenogramov izvedemo s pomočjo ukaza *Operations – Registration – Place Scans*, ki nam odpre okno z vprašanjem, kakšno registracijo želimo izvesti in katere parametre želimo vključiti v registracijo. Okno in nastavitve parametrov so prikazani na sliki 17. Prva registracija je bila, kot že navedeno, "*Target Based".*

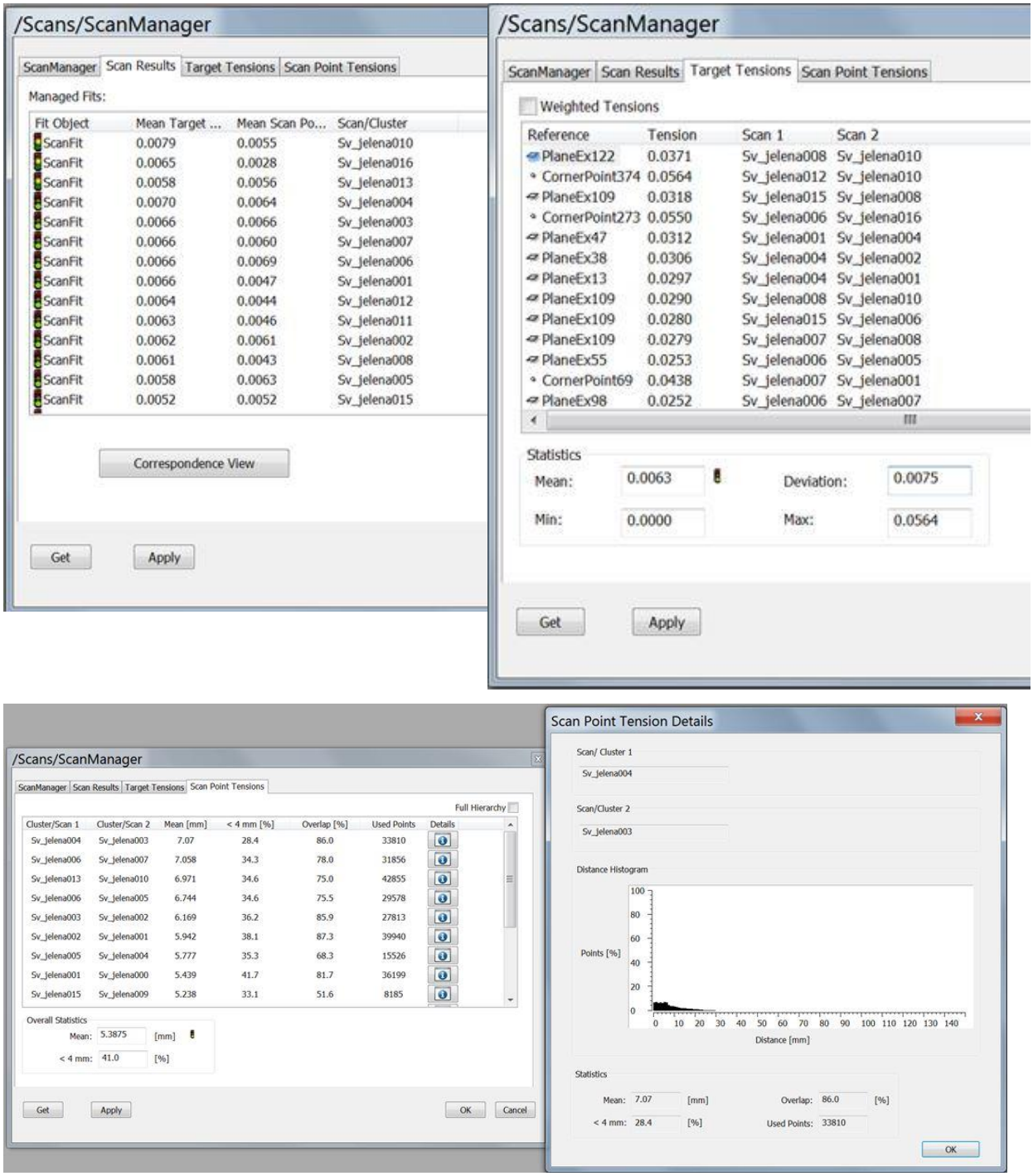


Slika 17: Nastavitve "Targed Based" registracije

"*Target Based"* registracija deluje odlično, če uporabimo natančno merjene sfere ali oznake, medtem ko ima v našem primeru, v katerem gre za naravne podrobnosti, ki jih program sam prepoznava med predprocesiranjem, ta način registracije manjšo natančnost in služi samo

za približno združevanje oblaka točk, da lahko izvedemo "*Cloud to Cloud"* registracijo. Naravne podrobnosti, definirane v predprocesiranju, uporabimo s pomočjo funkcije *Enable correspondence search*, približno znane lokacije in orientacije skenogramov pa s pomočjo *Find correspodences for scan positions*, ki so pridobljene z uporabo senzorjev skenerja (GPS, inklinometer, kompas in altimeter).

Rezultati "*Target Based"* registracije:



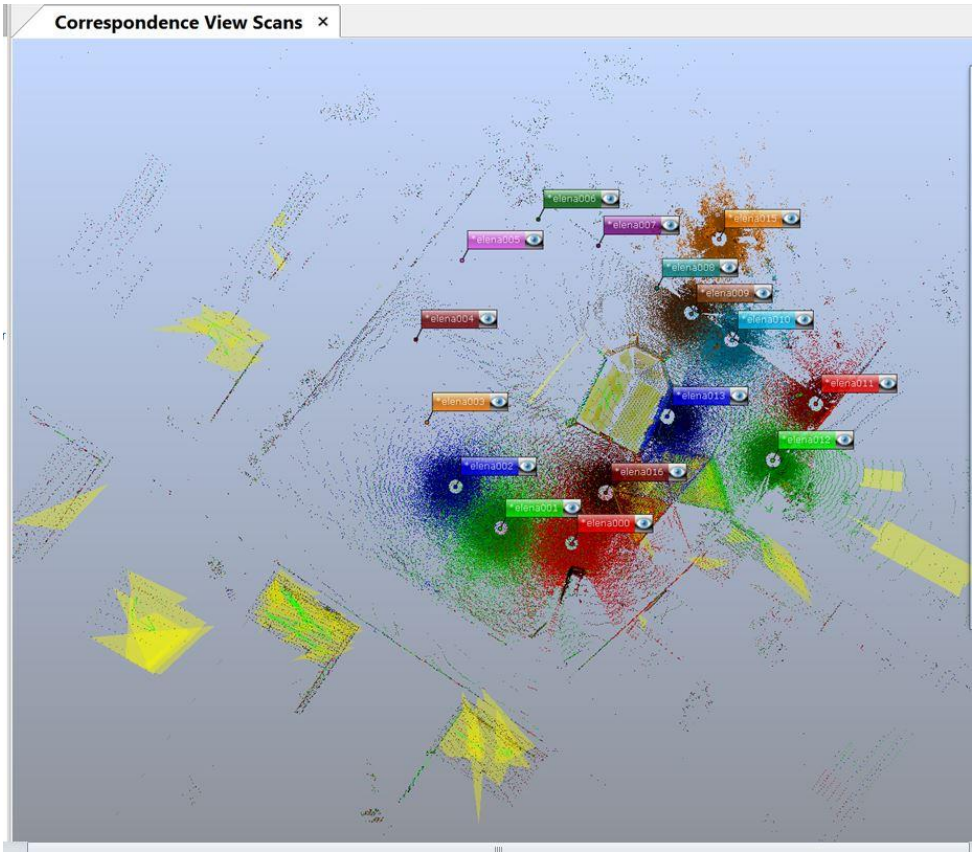
Slika 18: Rezultati "Target Based" registracije

Rezultate registracije pridobimo s pomočjo možnosti *ScanManager*, ki se samodejno odpre z zaključkom postopka izbranega registriranja. *Scan results* nam prikazuje skupno kakovost posamezne registracije. *Target Tensions* omogoča iskanje kritičnih parov, ki imajo slabe rezultate in kvarijo natančnost. Zadnji zavihek *Scan Point Tensions* prikazuje vse referenčne pare skenogramov, uporabljene za registracijo, ter končno skupno statistično natančnost registracije. Podatki o natančnosti, ki jih dobimo med dvema referenčnima paroma skenogramv so: aritmetična sredina, ki opisuje neskladje med skenogramoma; število točk z napako, manjšo od 4 mm; število prekrivajočih se skeniranih točk v deležih in število skeniranih točk, uporabljenih za izračun statistike.

Podatki skupne statistike:

- Aritmetična sredina: 5,3875 [mm]
- Odstotek točk z napako, manjšo od 4 mm: 41 [%]

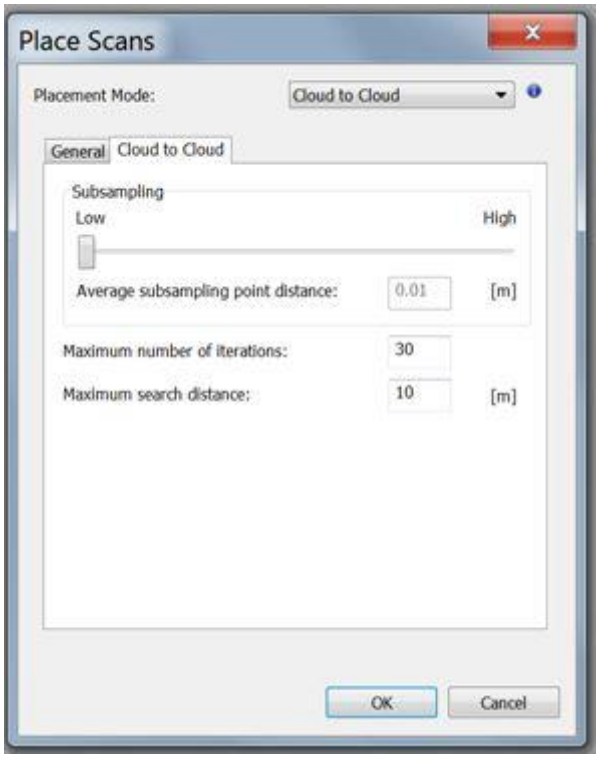
Prikaz za preverjanje združenih skenogramov lahko dobimo s pomočjo ukaza *Correspondence View* v *Scan Manager*-ju, kjer so prikazane tudi skupne točke in ravnine, uporabljene za registracijo vsakega od stojišč (slika 19). Ob uporabi ukaza nas program vpraša po želenem prikazu maksimalnega števila točk – 10 milijonov.



Slika 19: "Correspondence view" v programu SCENE

Druga registracija je bila "Cloud to Cloud." Kot že prej omenjeno, je za "Cloud to Cloud" registracijo potrebna predhodna registracija skenogramov z eno od ponujenih vrst registracije. Vzrok za to je, da "Cloud to Cloud" registracija analizira skenirane točke sosednih skenogramov in popravlja lokacije referenčnih objektov. "Cloud to Cloud" registracija deluje po principu iterativne izravnave, ta postopek pa zahteva veliko časa, kar pa je odvisno od zmogljivosti računalnika, na katerem se izvaja.

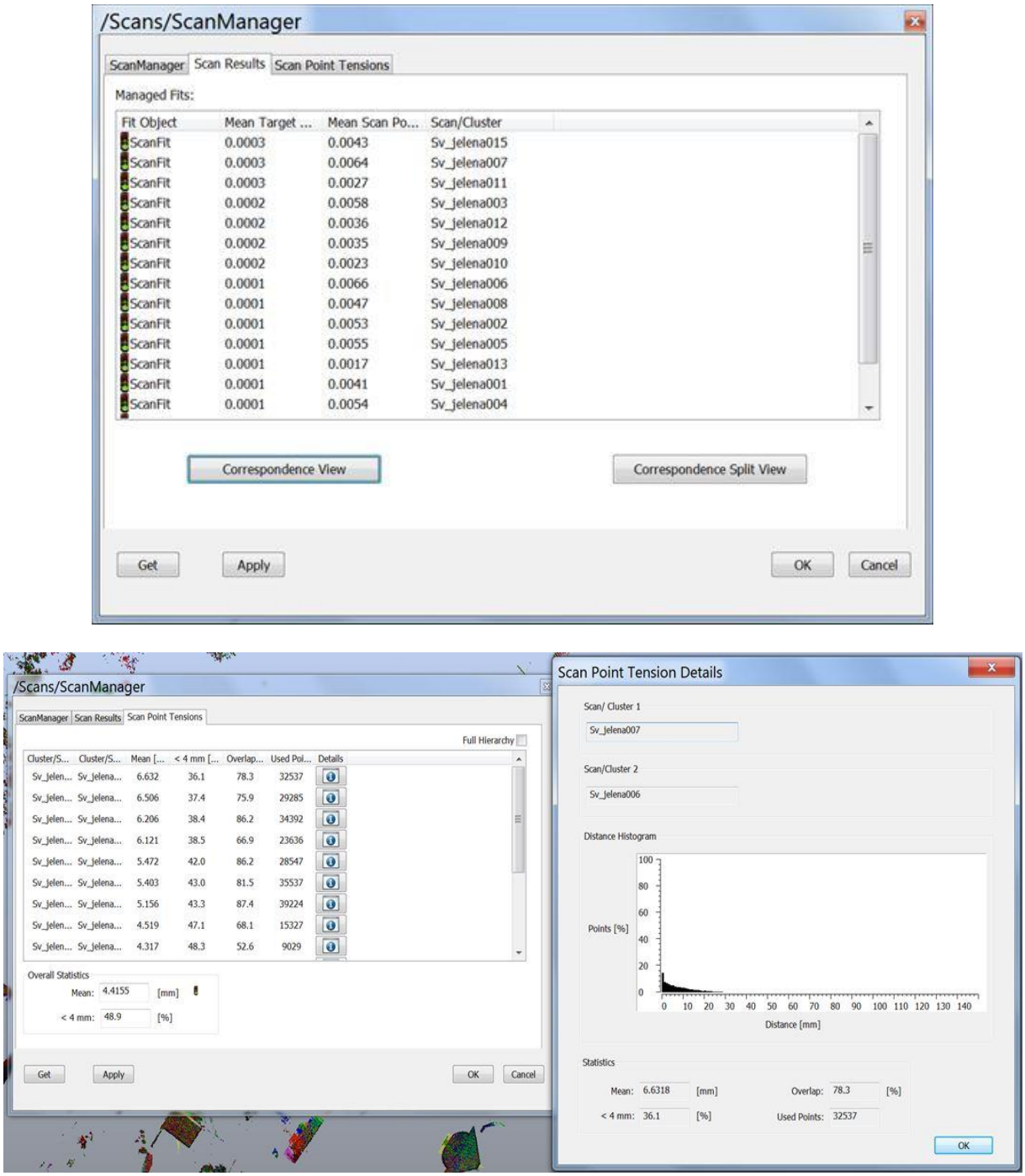
"Cloud to Cloud" registracijo lahko zaženemo na enak način kot "Target Based" in sprejmemo ponujene nastavitve (slika 20).



Slika 20: Nastavitve "Cloud to Cloud" registracije

Po končani registraciji se ponovno odpre okno *Scan Manager*, v katerem so prikazani rezultati registracije. Že iz zavihka *Scan Results* je razvidno, da smo s "Cloud to Cloud" registracijo pridobili veliko boljše rezultate, podatki skupne statistike pa to potrjujejo. S tem procesom je registracija zaključena in imamo v projektu združen oblak točk. Pred nadaljevanjem obdelave je potrebno projekt ponovno shraniti, s čimer sprejmemo narejene spremembe. (FARO Technologies Inc., 2015b).

Rezultati "Cloud to Cloud" registracije:



Slika 21: Rezultati "Cloud to Cloud" registracije

Prikaz rezultatov je enak kot po "Target Based" registraciji z izjemo zavihka *Target Tensions*, ki ga v tem primeru ni, saj pri "Cloud to Cloud" registraciji niso bile uporabljene skupne točke ali ravnine (slika 21).

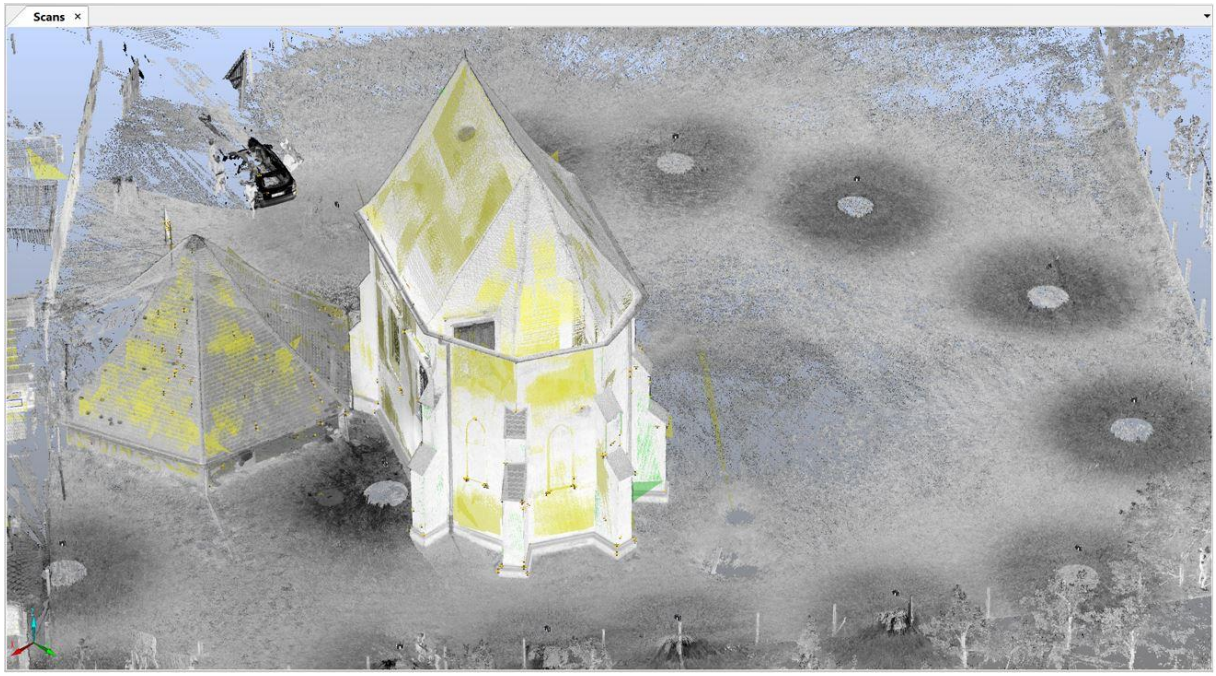
Podatki skupne statistike:

- Aritmetična sredina: 4,4155 [mm]
- Odstotek točk z napako, manjšo od 4 mm: 48,9 [%]

5.1.3 Končna obdelava in izvoz oblaka točk

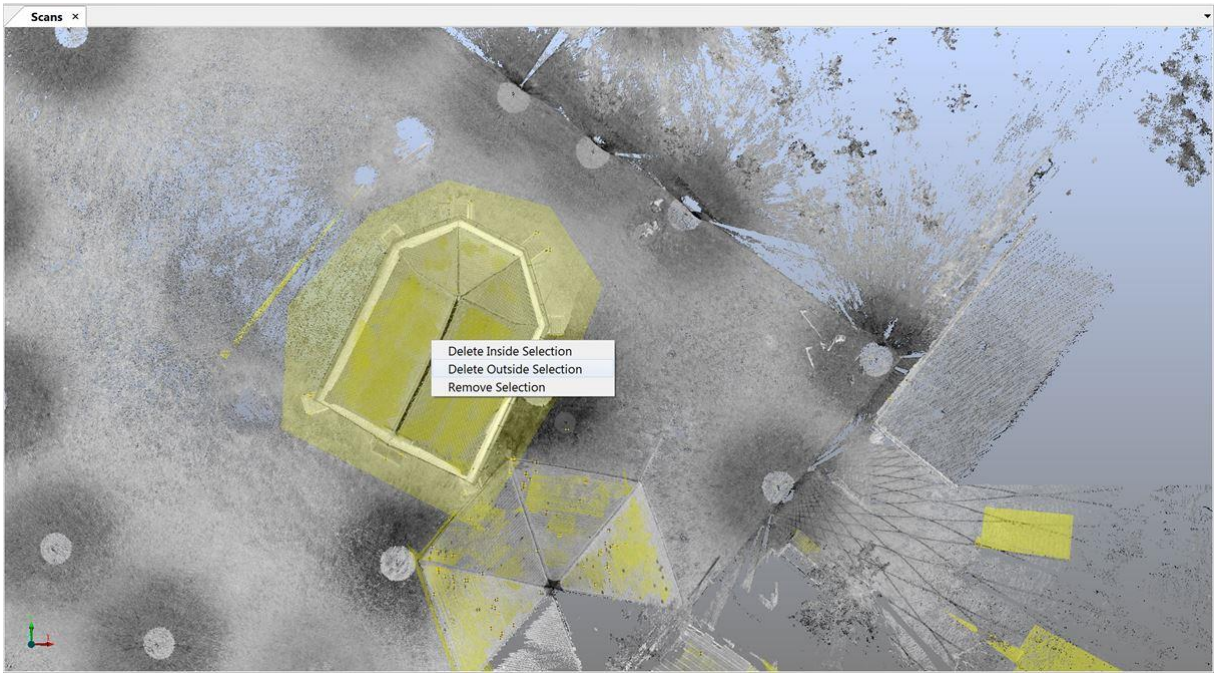
Pred izvozom oblaka točk je poleg obarvanja skenograma z realnimi barvami iz fotoaparata skenerja, zaželeno izbrisati nepotrebne, odvečne skenirane točke okoli objekta, s čimer se zmanjša količina podatkov, ki jih mora računalnik predelati, in s tem pospeši njegovo delovanje, prav tako pa se izboljša tudi preglednost skengrama.

Najprej lahko v programu SCENE pregledamo registrirani oblak točk, ki smo ga dobili s pomočjo "Target Based" in "Cloud to Cloud" registracij. To storimo s pomočjo desnega miškinega klika na *Scans* v *Workspace*-u in *View – 3D view* (slika 22).



Slika 22: "3D view" registriranega oblaka točk v programu SCENE

Na pridobljenem oblaku točk smo najprej izbrisali nepotrebne podatke okoli skeniranega objekta. To smo izvedli tako, da smo okno pregledovanja v programu SCENE nastavili na "Top View" s pomočjo ikone *Top View*, nato s pomočjo ikone *Polygone Selector* na tlorisnem prikazu z miško označili področje skena, ki smo ga želeli obdržati, in poligon zaprli z dvojnim levim miškinim klikom. Nato smo z desnim miškinim klikom izbrali možnost *Delete Outside Selection*, s katero smo izbrisali vse odvečne točke okoli objekta (slika 23).

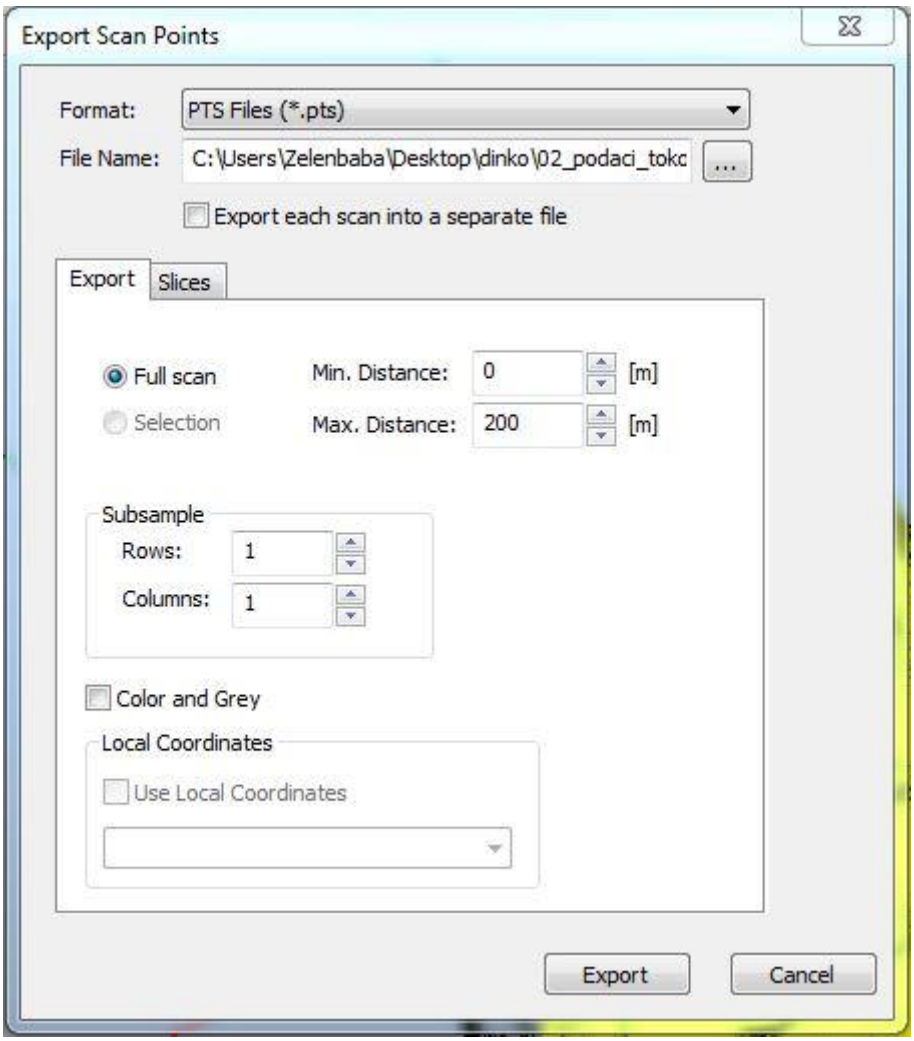


Slika 23: Brisanje nepotrebnih podatkov okoli skeniranega objekta

Po brisanju odvečnih točk lahko preostalemu oblaku točk, ki opisuje naš skenirani objekt, dodamo realne barve iz kamere skenerja. To izvedemo s pomočjo desnega miškinega klika na *Scans,* kjer izberemo *Operations – Color/Pictures – Apply Pictures.* Operacija najprej naloži vse skenograme in njihove fotografije, nato na fotografijah odpravi vse distorzije in na koncu vsaki skenirani točki doda informacijo o barvi. Na ta način pridobimo obarvani in registrirani oblak točk skeniranega objekta, ki ga lahko izvozimo in nadaljujemo z obdelavo oz. izdelavo modela (slika 24).

Slika 24: Obarvani in registrirani oblak točk

Na koncu je pridobljeni oblak točk treba izvoziti (ang. export) v želeni format 3D podatkov, da lahko nadaljujemo z obdelavo oz. izdelavo 3D modela. Program SCENE omogoča izvoz v le nekaj različnih formatov 3D podatkov, in sicer: E57, VRML, DXF, XYZ Text, XYZ Binary, IGES, FLS in PTS. Glede na to, da bomo poskušali modelirati z več programskimi paketi, smo izvoz izvedli v dva formata in sicer: XYZ Ascii in PTS. Izvoz skeniranih točk se izvede s pomočjo desnega miškinega klika na *Scans* in ukaza *Import/Export – Export Scan Points* (slika 25) (FARO Technologies Inc., 2015b).

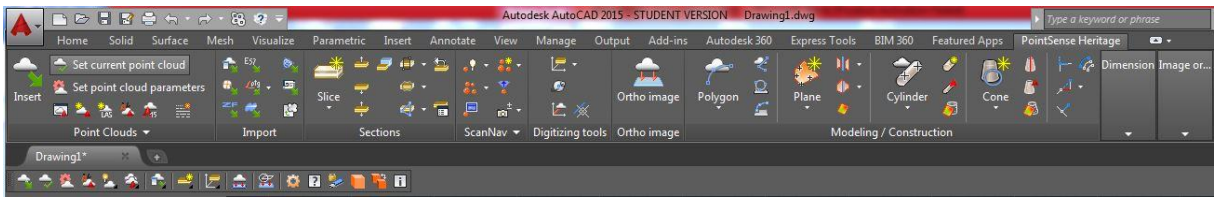


Slika 25: Nastavitve za izvoz skenograma

5.2 Izdelava modela v programu Kubit PointSense Heritage

Kubit PointSense Heritage je programsko orodje, ustvarjeno kot dodatek (ang. plug-in software) za AutoCAD, ki omogoča lažjo obdelavo laserskih skenov in fotografij znotraj programa AutoCAD z namenom pridobivanja 3D modelov in drugih izdelkov. Glede na to, da je AutoCAD razširjen in znan velikemu številu uporabnikov, način obdelave v njem olajša obvladovanje novih izzivov. Tako ima Kubit PointSense Heritage poleg že znanih AutoCAD-ovih funkcij dodane tudi lastne funkcije, ki v veliki meri olajšajo delo z oblakom točk. Že v AutoCAD-u je prisotnih nekaj funkcij za delo z oblakom točk, kot so npr.: prikaz, rezanje (ang. cropping), segmentiranje (ang. sectioning), pripenjanje (ang. snapping). Kubit PointSense Heritage pa omogoča še: rezanje na manjše dele (ang. Slicing), Section Manager, ortofotografije, risanje linij in poligonov na oblaku točk, ekstrakcijo cilindrov, detekcijo ravnin, analizo površin in delo s fotogrametrijo, lepljenje fotografij na ravnine iz ortofoto-ja ter mnoge druge. Kubit PointSense Heritage se največ uporablja v arheologiji in ohranjanju kulturne dediščine za dokumentiranje s pomočjo 3D modelov in drugih podatkov, kot so razne analize skeniranih površin (kubitTV, 2015). Kubit PointSense Heritage je zelo obširen program z veliko izbiro možnosti. V tem delu nas zanima samo način, s katerim lahko izdelamo 3D model, in ali ga lahko dalje uporabimo v izbranem igralnem pogonu, zato smo se omejili na le en del njegovih možnosti. Kubit je leta 2015. prevzel FARO, ampak uradno se še vedno za ime programa uporablja Kubit.

Kubit PointSense Heritage deluje samo na novih različicah AutoCAD-a, in sicer od 2010 naprej. Namesti se ga ločeno, z zagonom AutoCAD-a po namestitvi pa se dodatek samodejno naloži v program AutoCAD. Vse funkcije Kubit PointSense Heritage-a se nahajajo na zavihku PointSense Heritage znotraj AutoCAD-a (slika 26).



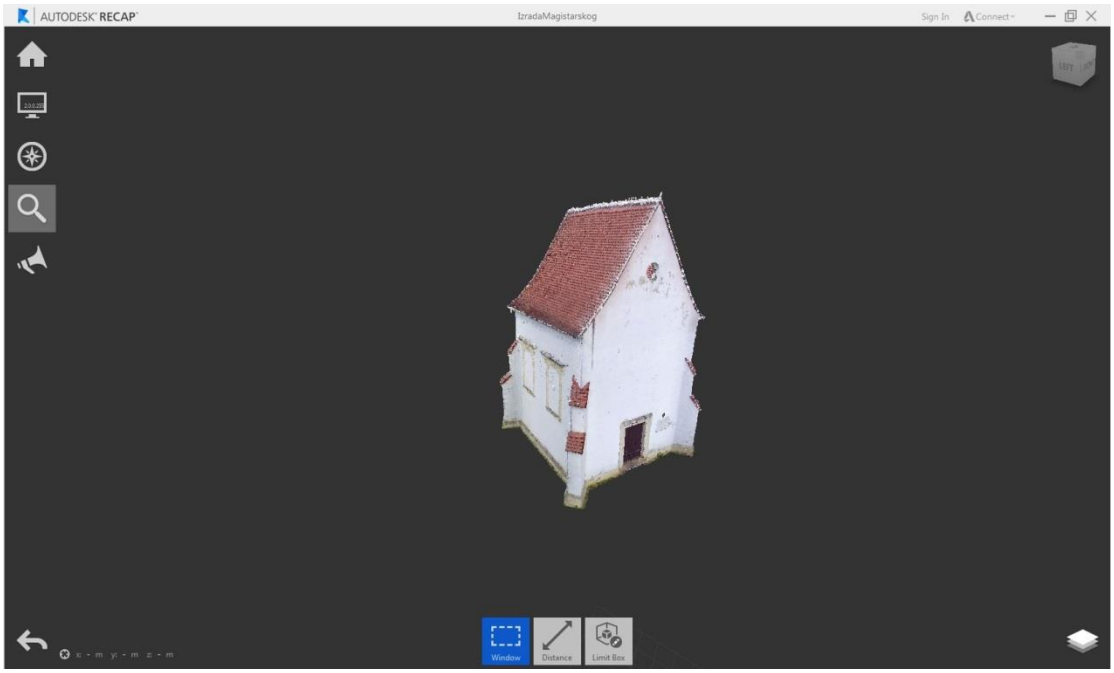
Slika 26: Videz Kubit PointSense Heritage dodatka znotraj programa AutoCAD 2015

Format podatkov za oblak točk, ki ga program AutoCAD podpira, je RCP. Pridobimo ga lahko s pomočjo Autodeskovega programa ReCap. Zato je pred začetkom dela z AutoCAD-om potrebno podatke pripraviti v programu ReCap.

5.2.1 Priprava podatkov v programu Autodesk ReCap 2016

Autodesk ReCap je program, ki se uporablja za ustvarjanje in upravljanje 3D podatkov iz laserskih skenogramov ter fotografij po poenostavljenem postopku. S pomočjo programa ReCap smo naložili skenirani oblak točk, pridobljen s programom SCENE, ter ga izvozili v format RCP, ki ga lahko uporabimo v programu AutoCAD.

Delo s programom ReCap je zelo enostavno. Ob zagonu programa ustvarimo nov projekt, izberemo podatek (v našem primeru PTS format, ki smo ga pred tem izvozili), ki ga želimo uvoziti v program (ang. import), in s pomočjo ikone *index scans* zaženemo nalaganje izbranega skena (slika 27). Ko se sken naloži, je treba projekt zagnati s pomočjo ikone *launch project*.

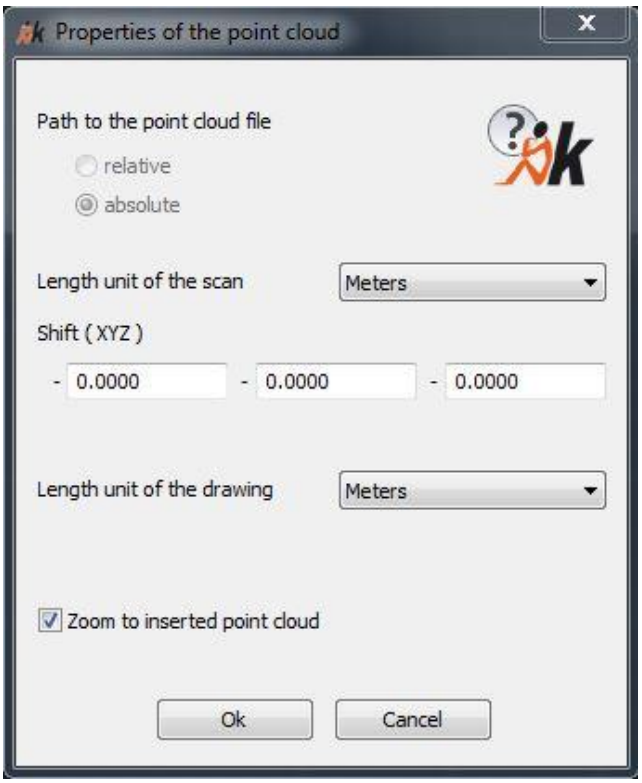


Slika 27: Uvoženi oblak točk v programu ReCap

Čeprav ima program ReCap možnost obdelave oblaka točk, npr. odstranjevanje odvečnih delov ipd., smo ga uporabili le za izvoz v željen format podatkov. S pomočjo ukaza *Export* odpremo okno, v katerem izberemo lokacijo in format shranjevanja.

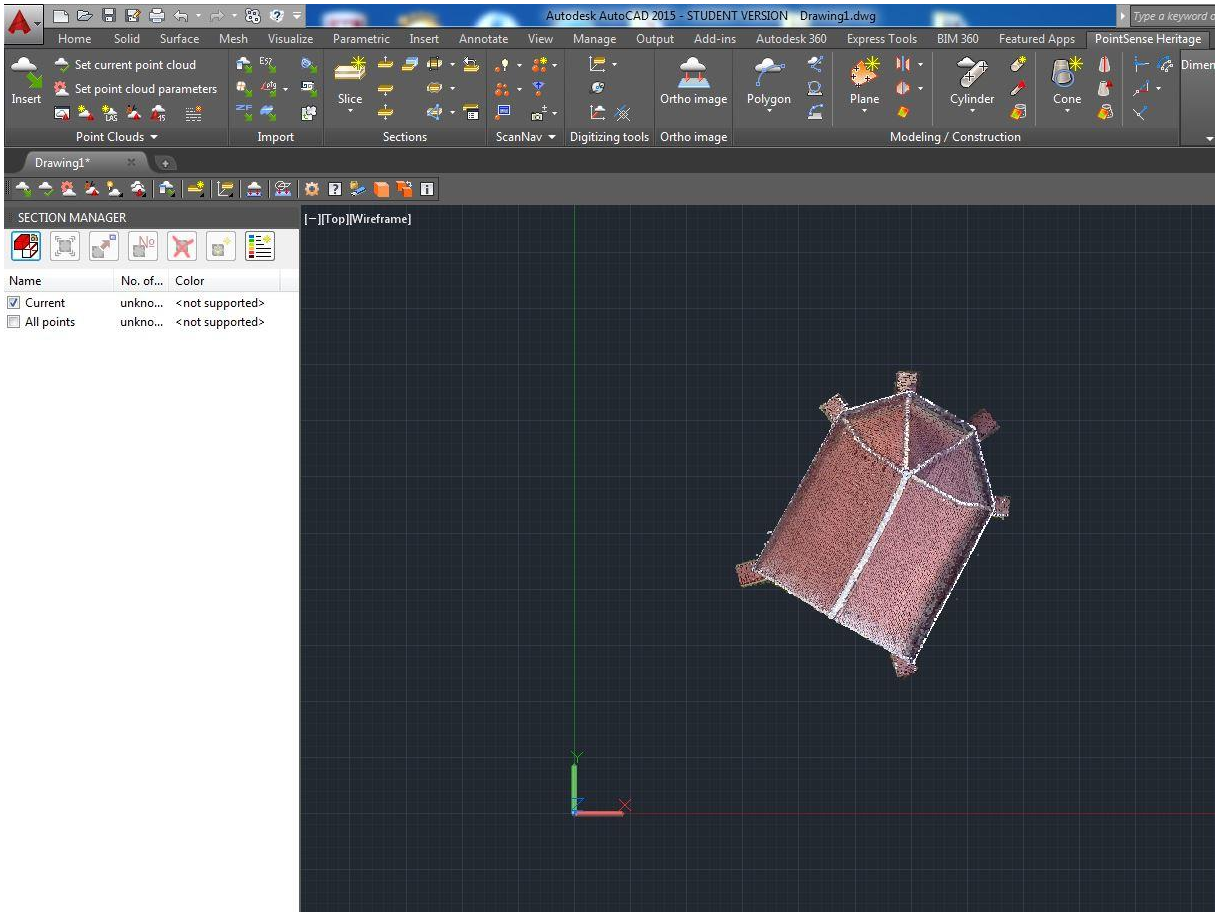
5.2.2 Modeliranje

Za uvoz skeniranega oblaka točk lahko uporabimo AutoCAD-ov ukaz *Attach Point Cloud* v zavihku *Insert.* Z uporabo tega ukaza se nam odpre okno, v katerem izberemo datoteko formata RCP, ki smo jo ustvarili s programom ReCap. Alternativno lahko podatke uvozimo s pomočjo dodatka Kubit PointSense Heritage, in sicer z ukazom *Insert*, ki nam prav tako odpre okno za izbiro datoteke, ki jo želimo naložiti (slika 28).



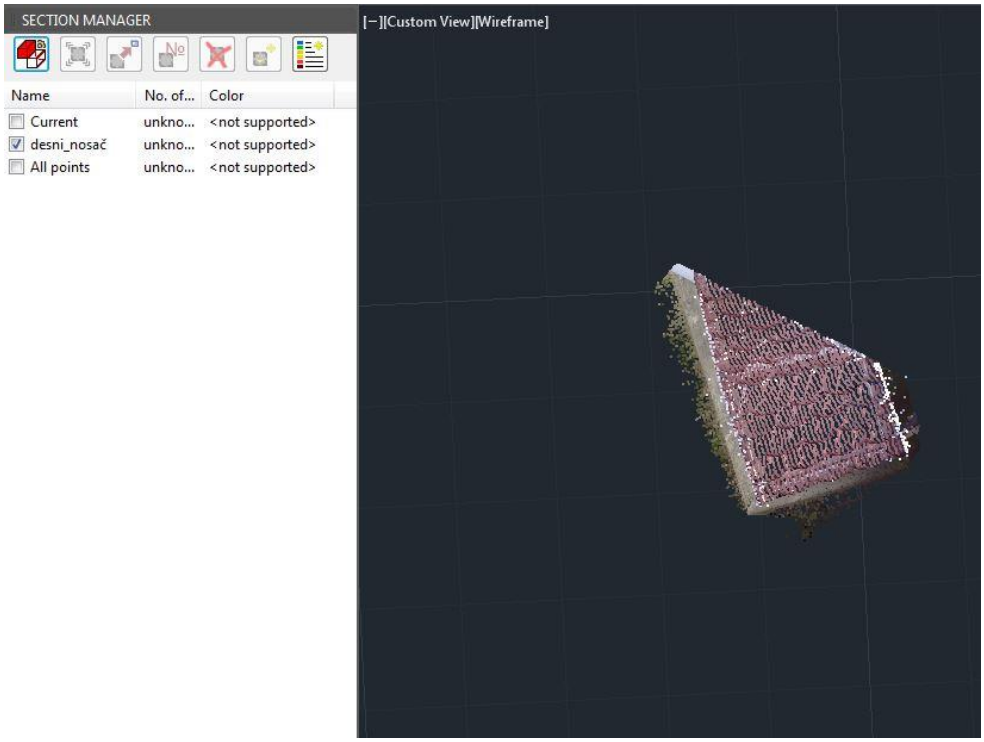
Slika 28: Nastavitve uvoza oblaka točk v AutoCAD

Po naložitvi oblaka točk v AutoCAD samodejno dobimo "Top View" pogled na le-tega in odpre se nam okno "Section Manager", ki nam omogoča upravljanje s skupinami oblaka točk, ki jih bomo vzpostavili zaradi lažje obdelave celotnega oblaka. "Section Manager" ima na začetku dve skupini – *Current* in *All points* (slika 29).



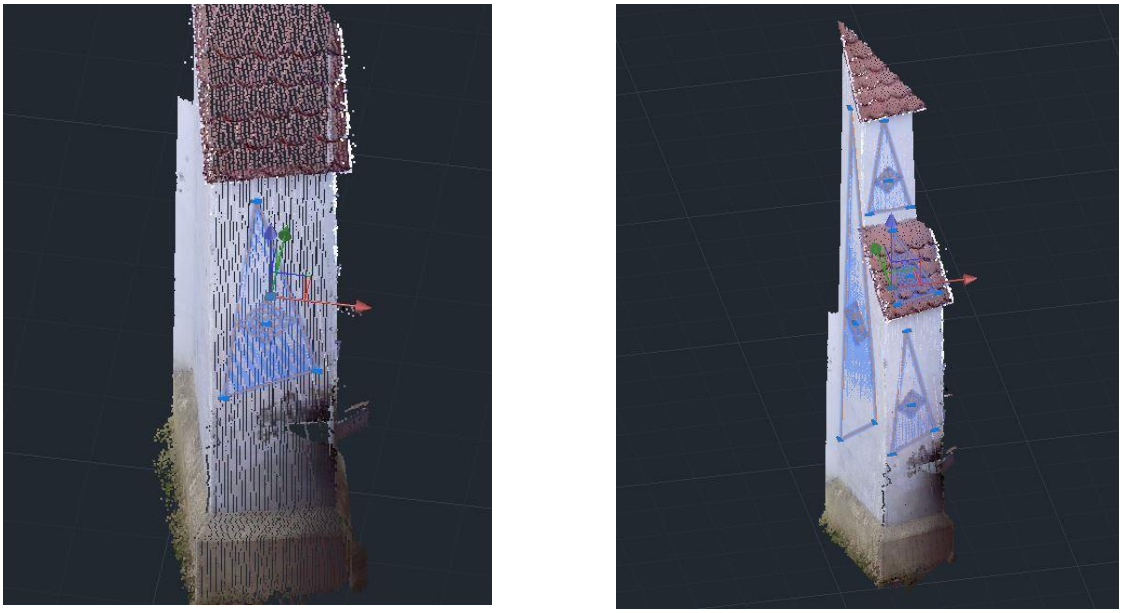
Slika 29: Pogled na uvoženi oblak točk v AutoCAD-u

Pogled na oblak točk lahko upravljamo s pomočjo AutoCAD-ovih standardnih funkcij v 3D Modeling načinu, kot sta *Pan* in *Orbit.* Najprej smo organizirali skupine oblaka točk, kar nam je olajšalo postopno obdelavo celotnega oblaka. Organizacijo izvajamo s pomočjo ukaza *Deffine clipping polygon* v skupini *Sections,* s katerim označimo podrobnost na oblaku točk, ki jo želimo izpostaviti, in se ta prikaže v oknu "Section Manager" pod skupino *Current.* Ko preimenujemo skupino *Current* v poljubni naziv, nam je ta razdelek shranjen v tej skupini in ga vključujemo ter izključujemo po potrebi. Na enak način pripravimo vse dele celotnega oblaka točk. Delitev na manjše dele v veliki meri olajša postopek, saj pri delu s celotnim oblakom vedno obstaja nevarnost, da zajamemo točko, ki je nismo želeli. Na sliki 30 lahko vidimo skupino "desni_nosač", ki smo jo izločili iz celotnega oblaka točk.



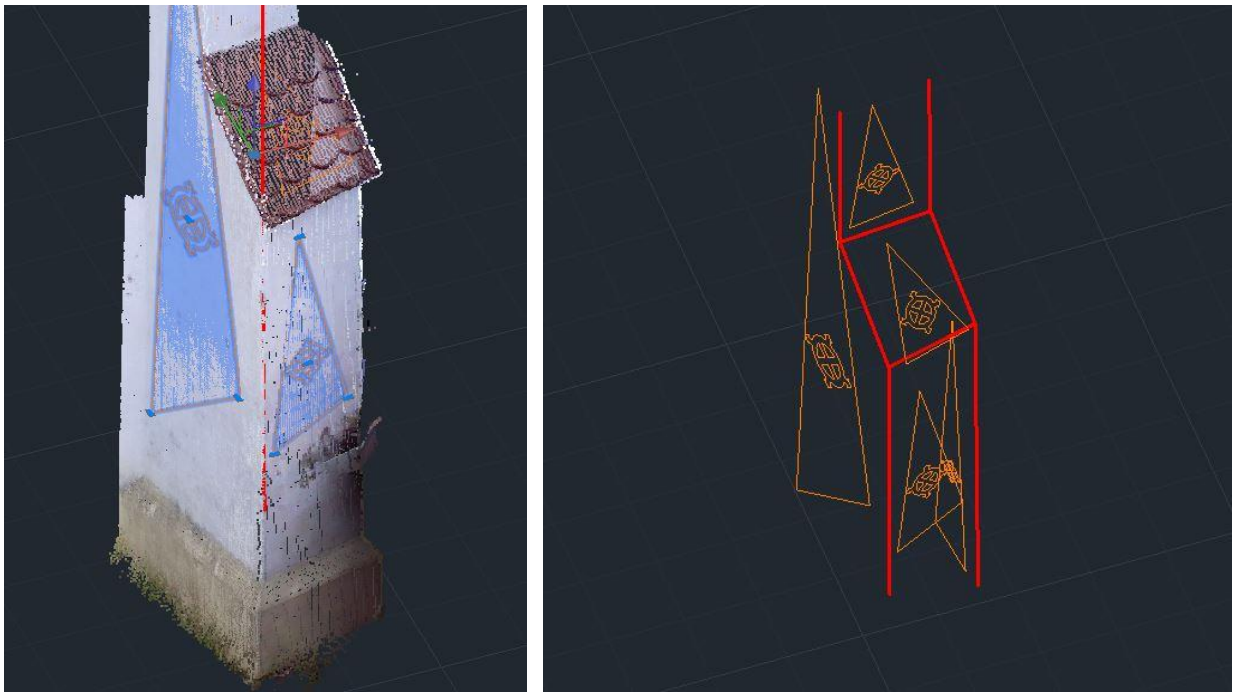
Slika 30: Skupina "desni_nosač", s katero smo izločili del oblaka točk

Naslednji korak je bil definiranje ravnin in presekov ravnin, s katerimi smo izgradili žični model. Ravnina se definira s pomočjo ukaza *Draw plane* v skupini *Modeling/Construction.* Ta ukaz od nas zahteva, da v oblaku točk označimo tri točke, katerim želimo definirati ravnino. S pomočjo tega ukaza definiramo vse sosednje ravnine na izbranemu segmentu (slika 31).



Slika 31: Določene ravnine na izbranem delu oblaka točk

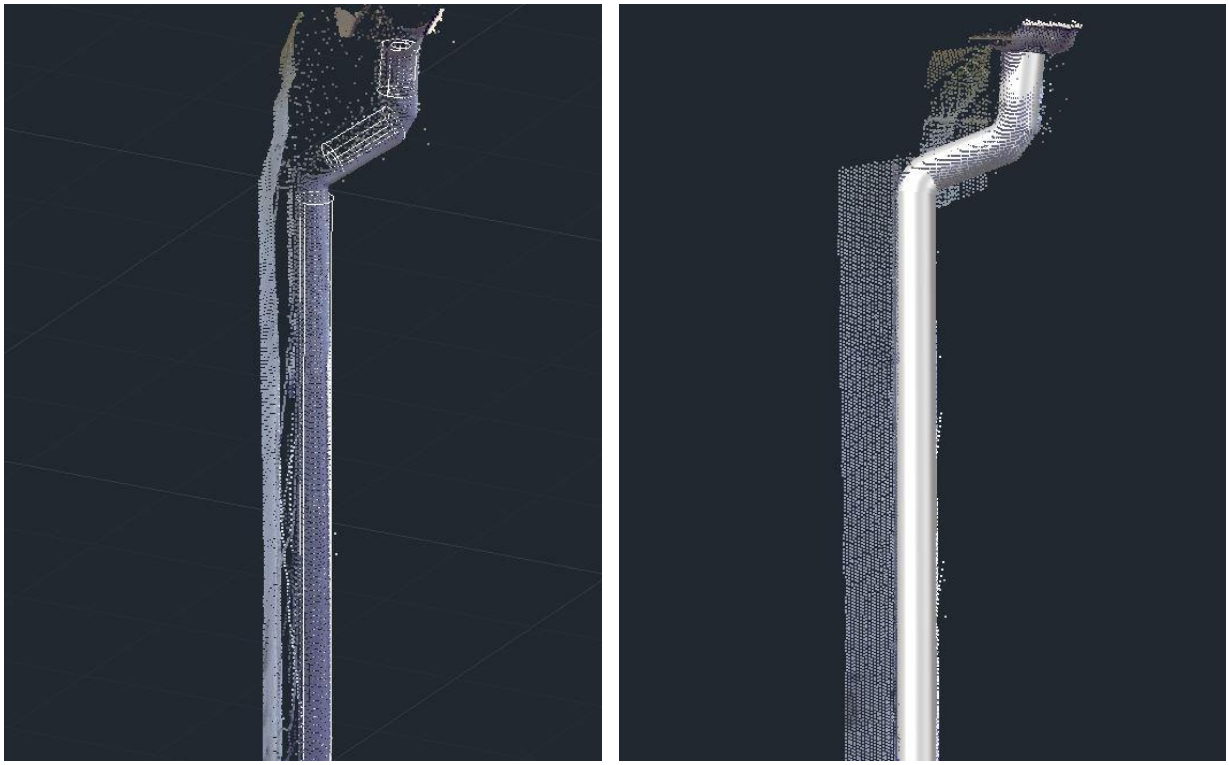
Po definiranju ravnin, med katerimi nas zanimajo robovi, smo nadaljevali s konstrukcijo linij, ki definirajo te robove. Najprej smo zaradi lažje organizacije na risbi ustvarili nov sloj (layer), v katerem se bodo robovi izrisovali, ga označili z rdečo barvo in povečali debelino zaradi boljše opaznosti. Nov sloj mora med uporabo ukaza, s katerim rišemo nove elemente, biti postavljen na trenutnega. Linijo (rob) med dvema ravninama smo dobili s pomočjo ukaza *Intersection line*, ki od nas zahteva, da označimo želeni ravnini. Za vsaki naslednji ravnini ponovno uporabimo isti ukaz in tako označujemo želene ravnine. Program pri izrisovanju skupne linije, ki definira rob, ne prepoznava mej te linije, zato jih je treba s pomočjo AutoCAD-ovega ukaza *Fillet* združiti s sosednimi robovi. Na sliki 32 se vidi izrisani del, ki smo ga predhodno izločili kot skupino.



Slika 32: Izrisani robovi na odseku oblaka točk – na levi sliki s programsko izrisano linijo, na desni sliki s popravljenimi linijami s pomočjo ukaza *Fillet*

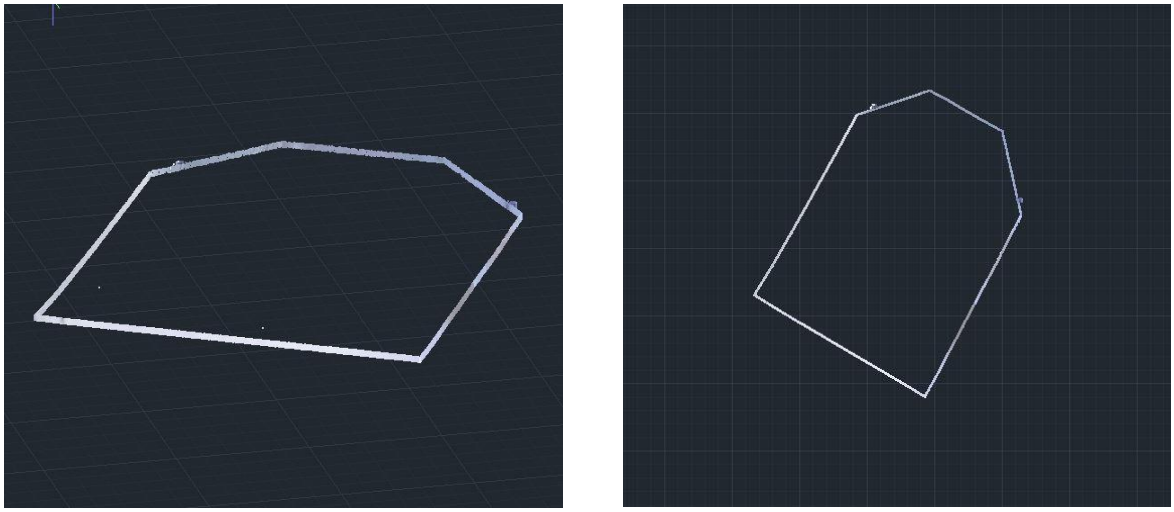
Enak postopek smo nadaljevali po celem oblaku točk, postopoma izrisujoč žični model, iz katerega bomo kasneje dobili površine. Omenjeni način se lahko uporablja za ravne ploskve, vendar lahko tudi pri tem pride do težav, če te niso popolnoma ravne, saj je treba zid razdeliti na več ravnin, da izrisane linije robov ne bi preveč odstopale od pravih robov. Za neravne ploskve obstajajo za izdelavo žičnega modela druge funkcije. V našem primeru smo za takim načinom obdelave posegli pri obdelavi strehe objekta in konstrukciji cevi žleba. Pri izdelavi cevi smo po že opisanem postopku izrezali del oblaka točk in ga shranili v poseben segment. Prav tako smo ustvarili posebni sloj za cevi. Za izdelavo cevi smo uporabili posebni ukaz, specializiran ravno za to, in sicer *Fit cylinder using 2 click point*. S pomočjo tega ukaza na oblaku točk izberemo dve točki na cevi in nam program samodejno izriše cev s primernim

premerom in smerjo. Ker je celoten žleb sestavljen iz treh cevi, smo jih združili z ukazom *Connect (elbow)*. Rezultat lahko vidimo na sliki 33.



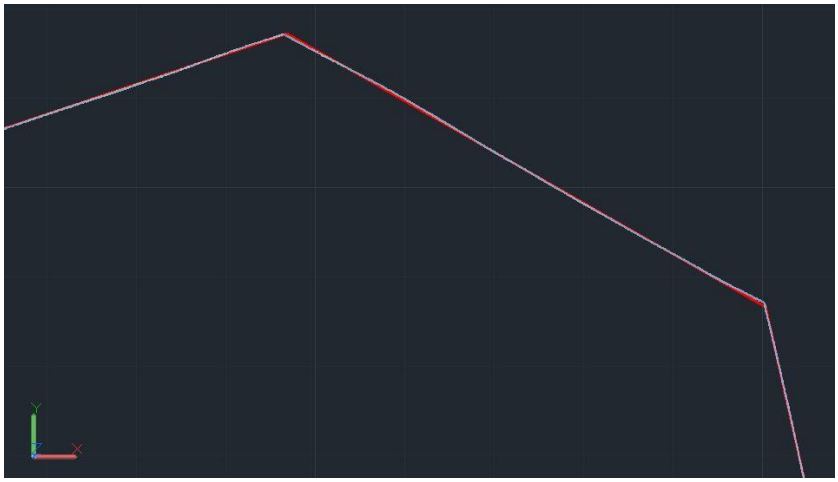
Slika 33: Izdelava cevi - na levi sliki so definirane cievi žleba , na desni pa združena ena cev

Pri naslednjem koraku smo se lotili izrisovanja bolj nepravilnega dela objekta, t.j. strehe. Pri tem smo imeli drugačno organizacijo prikaza delov oblaka točk. Uporabili smo Kubitovo funkcijo *Define slice*, s pomočjo katere definiramo poljubno debel presek oblaka točk, vzporeden z izbrano ravnino koordinatnega sistema. Ukaz od nas zahteva, da najprej določimo ravnino, s katero želimo "prerezati" oblak točk. V našem primeru je bila to XY ravnina. Nato moramo določiti točko v oblaku točk, v kateri želimo ta del (ang. slice) in nazadnje še debelino preseka. Izbrana točka je bila na zidu, neposredno pod streho, izbrana debelina pa je bila 10 cm.



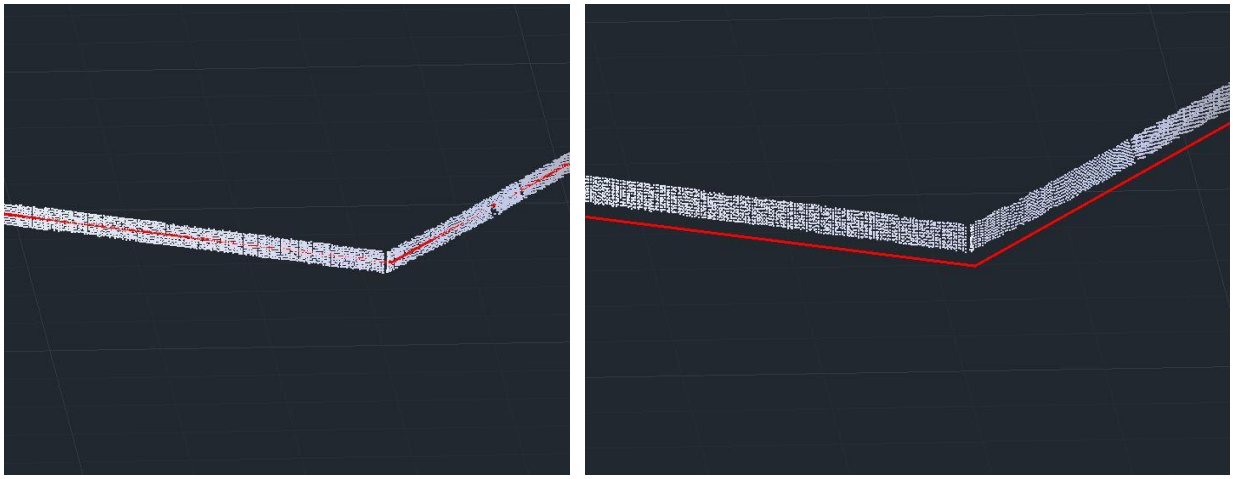
Slika 34: Definiran presek oblaka točk, vzporeden z XY ravnino, debeline 10 cm

Ko imamo definiran presek oblaka točk, preklopimo v tlorisni prikaz (ang. Top view), kot je razvidno iz desnega dela slike 34. S pomočjo ukaza *Fit polygon* smo približno izrisali tlorisni oris objekta (potrebno je izklopiti OSNAP). Nato je program samodejno izračunal najboljši položaj novonastalega poligona med vsemi točkami (slika 35).



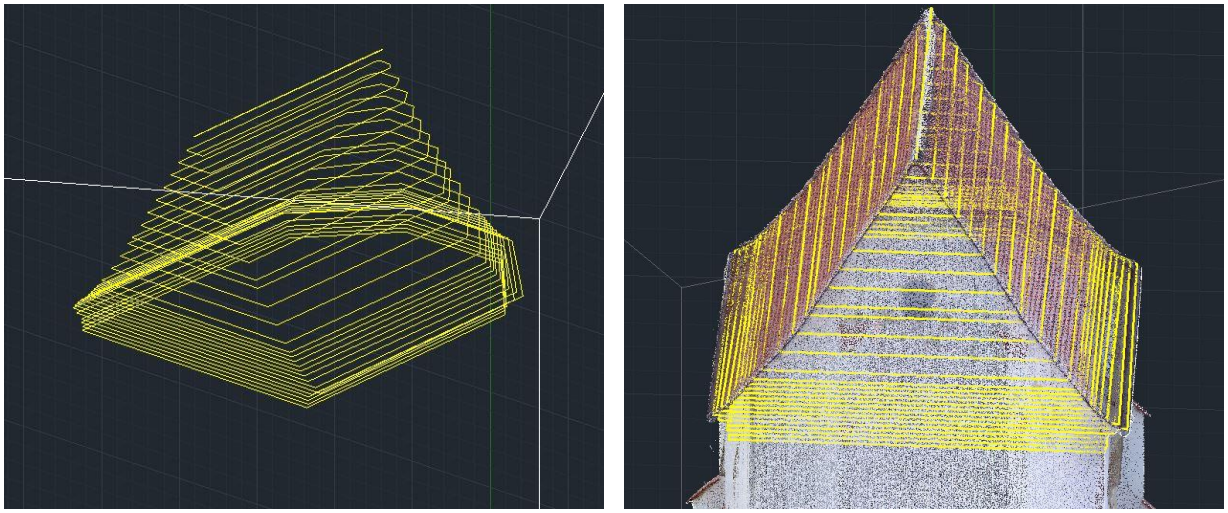
Slika 35: Prekirivanje izračunane lomljenke (rdeče) z oblakom točk

Pri vsakem naslednjem koraku smo presek dvignili za 10 cm na podrobnejšem delu pregiba oz. za 40 cm na strehi. To smo izvedli s pomočjo ukaza *Shift slice upwards.* Na sliki 36 lahko vidimo ujemanje poligona s presekom pred uporabo ukaza *Shift slice upwards* (leva slika) in po tem, ko je presek dvignjen za 10 cm (desna slika).



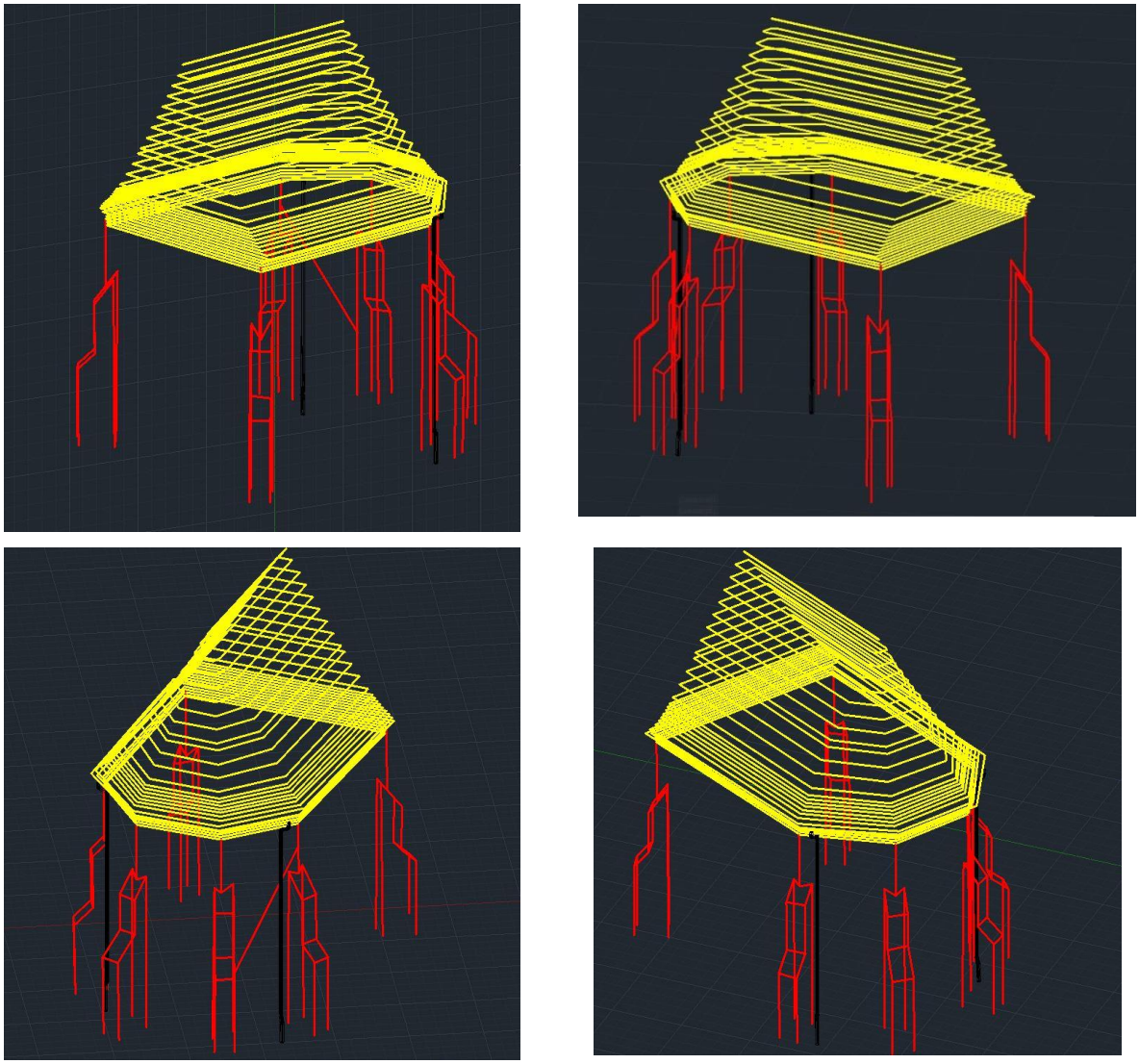
Slika 36: Del preseka z ujemajočim poligonom

Po vsakem dvigu preseka smo ponovno izrisali poligon, ki se najbolj ujema z določenim presekom. Postopek smo ponavljali vse do vrha strehe in s tem dobili žični model strehe, sestavljen iz niza vzporednih poligonov, ki približno opisujejo oris strehe. Poligoni so nam potrebni za poznejše definiranje nepravilne ravnine. Na sliki 37 lahko vidimo dobljeni žični model strehe (leva slika) in ujemanje žičnega modela strehe z oblakom točk (desna slika).



Slika 37: Žični model strehe

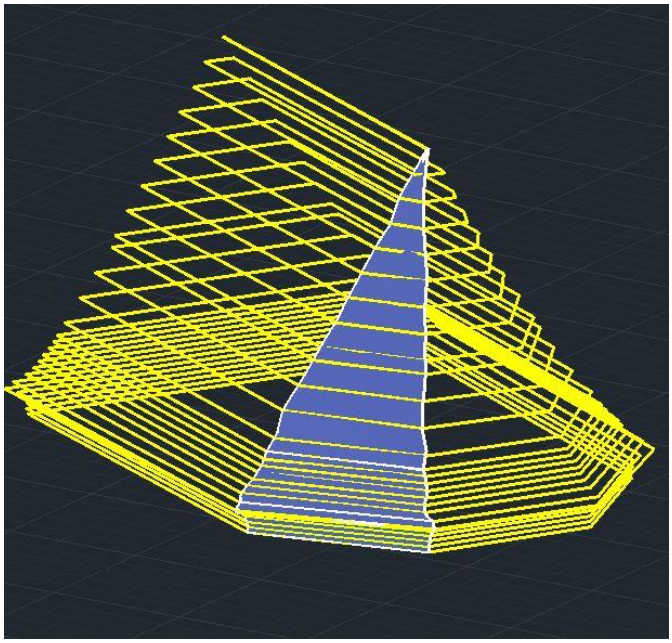
Z zaključkom žičnega modela strehe oz. zgornjega dela skeniranega objekta smo končali izdelavo žičnega modela celotnega objekta in se lotili definiranja ravnin, da bi dobili ploskovni model. Na sliki 38 lahko vidimo končan žični model iz štirih različnih zornih kotov.



Slika 38: Žični model kapele iz štirih zornih kotov

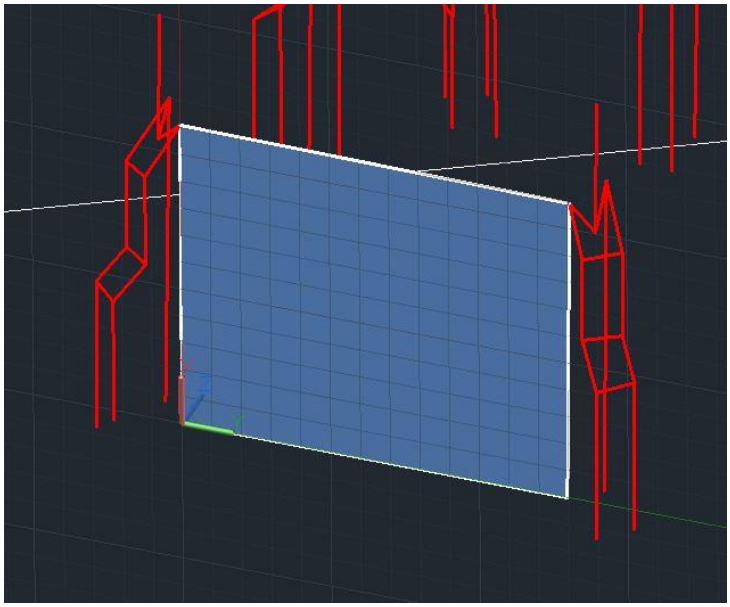
Kot je razvidno iz slike 38 in navedeno v poglavju 2.1.1, žični model ni popoln in se ga lahko napačno interpretira, zato v smislu vizualizacije modela ni zadosten. V tem primeru je njegova uporaba le za nadaljnje definiranje ploskev. Na sliki so podrobnosti, ki smo jih izrisali na različne načine, obarvane z različno barvo, in sicer: robovi rdeče, žlebovi črno, streha rumeno.

Najprej smo s pomočjo AutoCAD-ovega ukaza *Loft* izdelali ploskovni model strehe. Da smo na posameznih delih strehe lahko ustvarjali ploskve, smo poligone strehe razbili na posamezne linije s pomočjo ukaza *Explode*. S pomočjo ukaza *Loft* z označevanjem vzporednih linij od temena strehe proti vrhu ustvarjamo nepravilno ploskev, kot je razvidno iz slike 39.



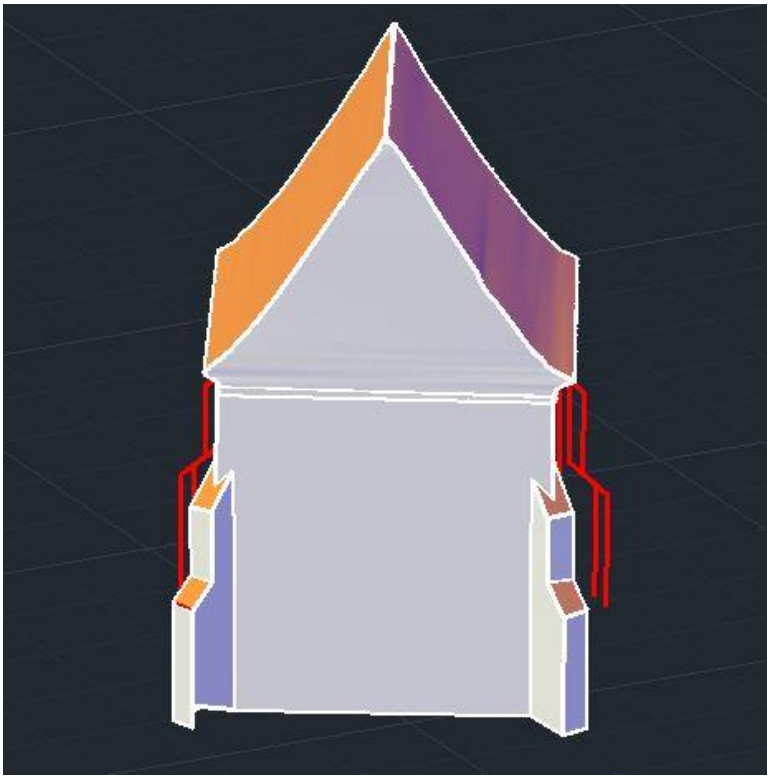
Slika 39: Ploskev na eni strani strehe, konstruirana s pomočjo ukaza *Loft*

Za določevanje ploskev na ravnih površinah uporabljamo ukaz *Region.* Najprej moramo definirati koordinatni sistem ravnine, ki jo želimo izrisati. To storimo s pomočjo ukaza *UCS* z izhodiščno točko na robu te ravnine in še dvema točkama. Ko je želena ravnina definirana, oris ploskve izrišemo z navadno lomljeno črto (polilinijo) programa AutoCAD, potem zaženemo ukaz *Region* in označimo lomljeno črto izbrane ploskve. Dobljeni rezultat je treba pretvoriti v ploskev (ang. Surface) s pomočjo ukaza *Convert to Surface.* "Region" smo po tem izbrisali, da nam je ostal samo "Surface". Rezultat je prikazan na sliki 40.



Slika 40: Definirana ploskev (Surface) na modelu

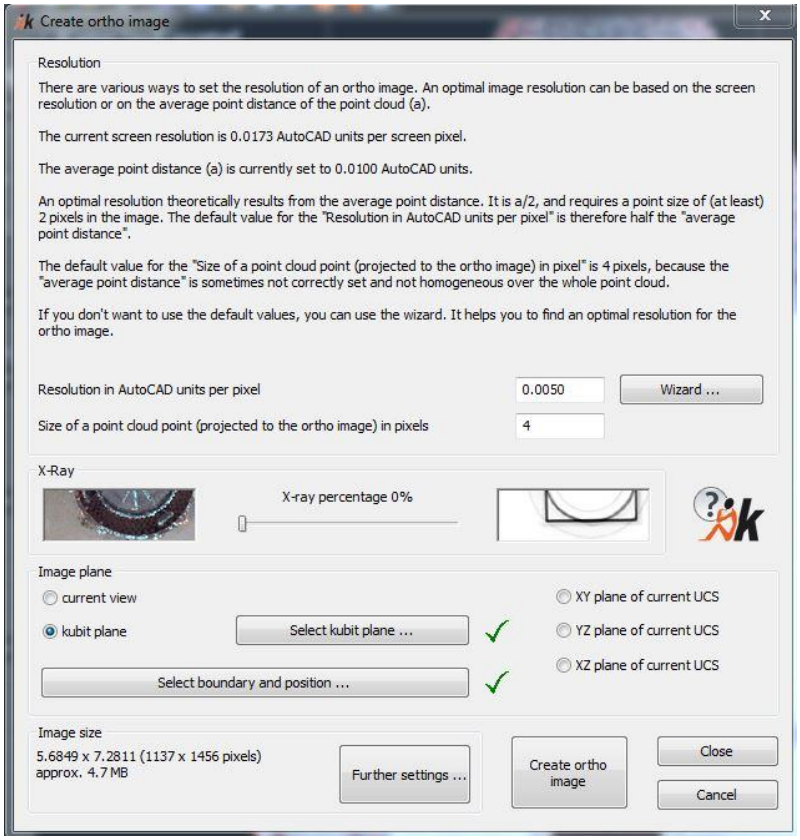
Na enak način smo obdelali celoten sprednji del objekta in skupaj s predhodno izdelanim modelom strehe dobili spodnji rezultat (slika 41).



Slika 41: Del ploskovnega modela, pridobljenega s pomočjo programa Kubit PointSense Heritage

Ker je modeliranje v Kubitu v primerjavi z ostalimi uporabljenimi programi najobsežnejše in zahteva največ časa, nismo dokončali ploskovnega modela, ampak smo se takoj lotili teksturiranja nedokončanega modela, da smo preverili, če je model, pridobljen s pomočjo programa Kubit PointSense Heritage, sploh mogoče uporabiti v izbranem igralnem pogonu.

Program Kubit omogoča dodajanje tekstur s pomočjo orientiranih fotografij, narejenih ločeno od skeniranja (klasični način), ali pa s pomočjo ustvarjanja ortofotografij iz obarvanega oblaka točk. Glede na to, da imamo v našem primeru obarvan oblak točk, smo uporabili to metodo. Pri teksturiranju modela je treba s pomočjo ukaza UCS za vsako ploskev posebej definirati ravnino, vendar tokrat s pomočjo ravnine, ki smo jo ustvarili na začetku z določitvijo treh točk na oblaku točk (*UCS from plane)*. Nato je treba zagnati ukaz *Create ortho image,* s katerim se odpre okno, kot je razvidno s slike 42.



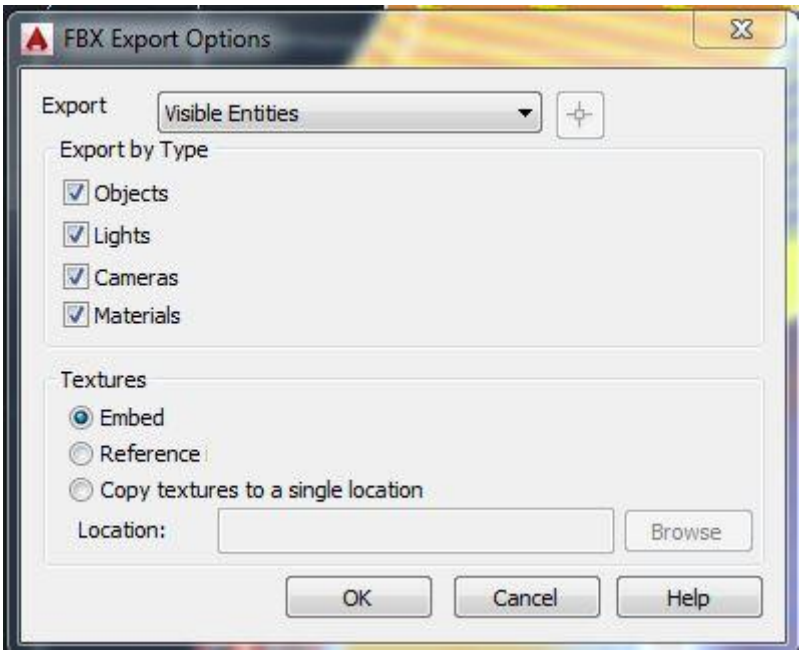
Slika 42: Nastavitve ustvarjanja ortofotografije iz oblaka točk

V nastavitvah določimo ločljivost fotografije (slike), z ukazom *Select kubit plane* izberemo ravnino, ki sovpada z ravnino želene fotografije, ter z ukazom *Select Boundary and position* določimo meje fotografije, ki bo ustvarjena iz oblaka točk z navpičnim pogledom na predhodno izbrano ravnino. S pritiskom na ikono *Create ortho image* se ustvarjena fotografija shrani v izbranem formatu (bmp, rle, jph, tif itd.). Po shranitvi nam program samodejno prikaže novonastalo teksturirano površino, kot lahko vidimo na sliki 43.

Slika 43: Ortofotografija na modelu, ustvarjena iz oblaka točk

Po končanem dodajanju tekstur lahko končni model izvozimo s pomočjo AutoCAD-ovega ukaza *Export – FBX* (slika 44). FBX je eden najbolj razširjenih formatov v 3D modeliranju in ga naš izbrani igralni pogon podpira.

Razlog za opustitev nadaljnje obdelave s programom Kubit PointSense Heritage je, da je obdelava dolgotrajna in bi za pridobitev vizualno prepoznavnega modela morali za vsako ravnino posebej ustvariti ortofotografijo in jo prilagoditi tej ravnini, kot je opisano v navedenem primeru. Program je namenjen predvsem dokumentiranju kulturne dediščine, zato se z njim zelo enostavno ustvarjajo žični in ploskovni modeli, medtem ko bi pridobivanje večjega modela s pravimi teksturami bilo zelo obsežno delo. Pomanjkljivost programa Kubit PointSense Heritage je, da nima možnosti samodejnega lepljenja tekstur iz oblaka točk, zato smo se lotili nadaljnjega iskanja enostavnejših programov, s katerimi lahko izdelamo končni cilj, ki je vizualno prepoznaven model.



Slika 44: Nastavitve izvoza modela v AutoCAD-u

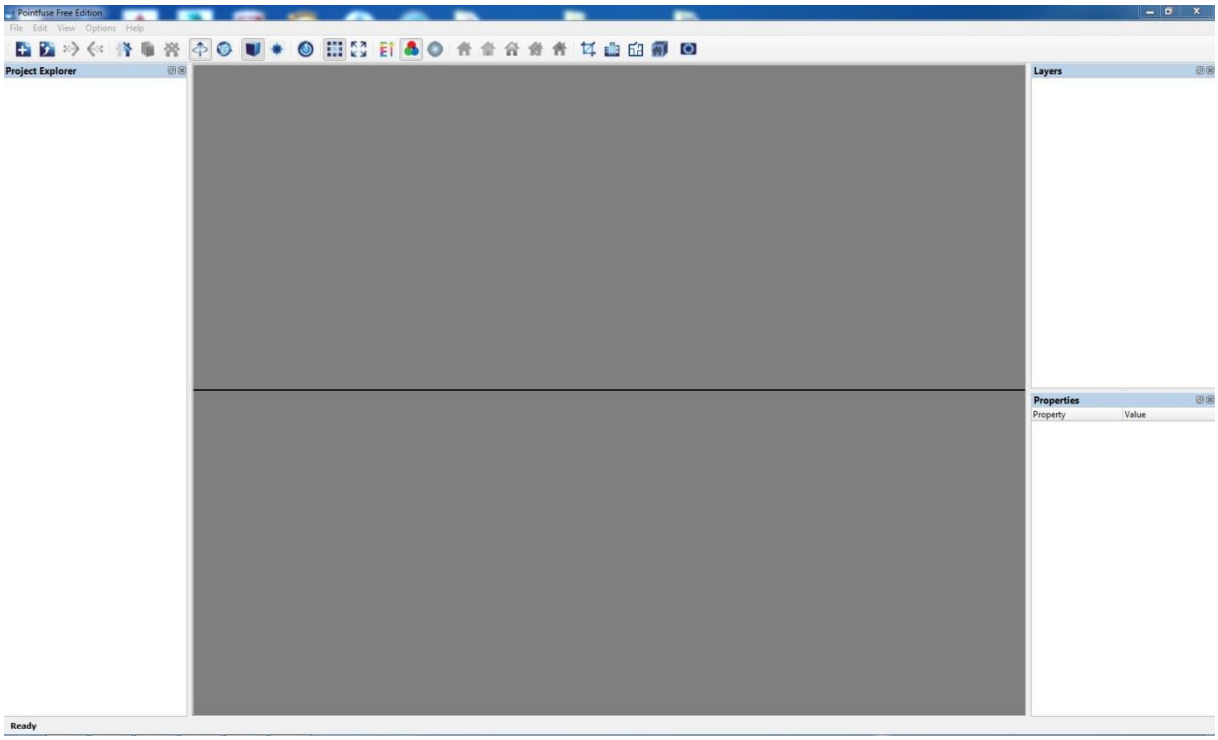
5.3 Izdelava modela v programu Pointfuse

Pointfuse je program podjetja Arithmetica, ki samodejno pretvarja obsežne oblake točk v zelo realistične teksturirane vektorske modele. Oblaki točk so lahko generirani iz katerega koli izvora, vključujoč lasersko skeniranje in fotogrametrične metode. Vektorski modeli se lahko ustvarijo iz celotnega oblaka točk, njegovega dela ali pa iz več oblakov točk. Sestavljeni so iz lomljenih črt in površin. Na sliki 45 vidimo vmesnik programa.

Najpomembnejše lastnosti programa, pomembne za to nalogo so:

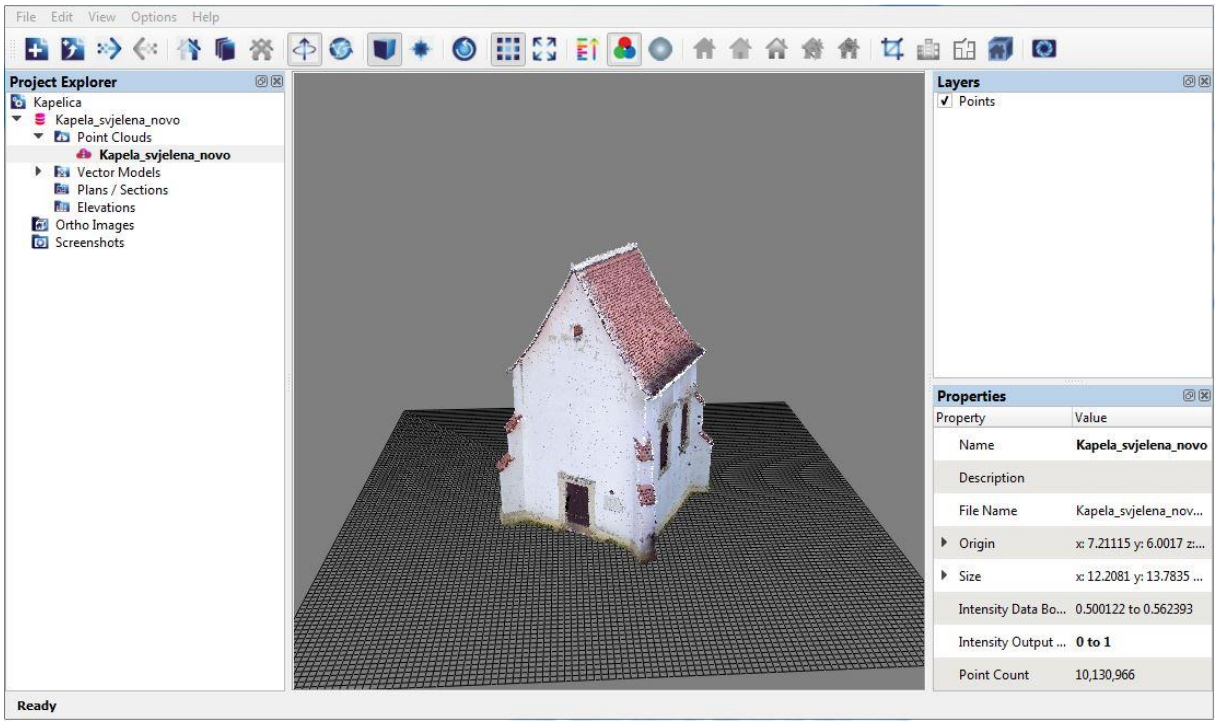
- samodejna izdelava vektorskih modelov iz oblaka točk s pomočjo ustvarjanja sklopov iz površin in robov s teksturami, ki vključujejo: RGB, intenziteto, gostoto ali oddaljenost od površine,
- uvoz oblakov točk, pridobljenih s terestričnimi LiDAR skenerji, podpirajoč formate DP, E57, FLS, FWS, LAS, LAZ, PTS, PTX in XYZ,
- Vizualizacija vektorskih modelov na več načinov, ki so lahko obarvani z realnimi barvami direktno iz oblaka točk
- Izvoz (ang. export) pridobljenih modelov v DAE (COLLADA), DXF, IFC ali OBJ (Wavefront) formatih, od katerih lahko format OBJ vsebuje dodane teksture.

(Arithmetica, 2015)



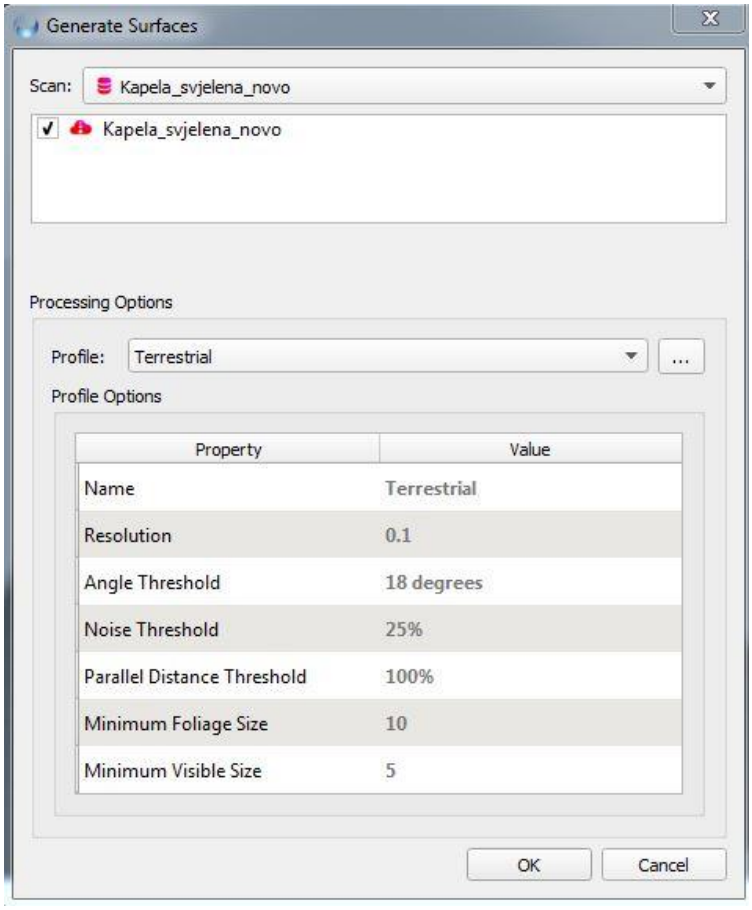
Slika 45: Vmesnik programa Pointfuse

Program Pointfuse je zelo enostaven za uporabo, saj je proces v celoti avtomatiziran. Do končnega modela se pride zelo hitro in enostavno. Prvi korak je ustvarjanje novega projekta (*File – New),* pri katerem poleg naziva projekta in lokacije shranjevanja, definiramo tudi oblak točk, s katerim bomo delali. Definirani oblak točk je bil PTS format, ki smo ga izvozili iz programa SCENE. Ko se izbrani podatki naložijo v program, se nam prikažejo v oknu *Project Explorer.* V tem oknu z desnim miškinim klikom na naloženi oblak točk in izbiro *Add to View* prikažemo podatke (slika 46).



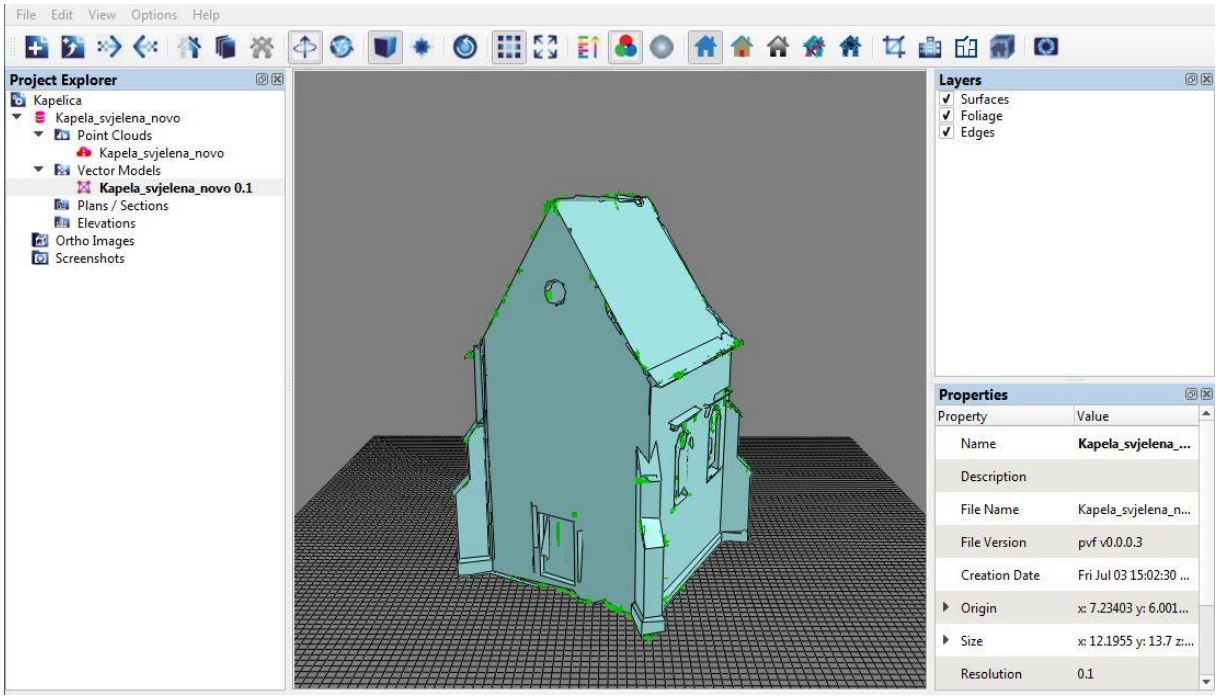
Slika 46: Oblak točk v programu Pointfuse

V nadaljevanju smo se lotili ustvarjanja modela s pomočjo ukaza *Edit – Generate Surfaces*, ki nam odpre okno z nastavitvami za generiranje površin. Program ima ponujene privzete nastavitve po profilih *Terestrial, Mobile* in *Airborne*, prav tako pa lahko tudi uporabnik sam izbere posamezne vrednosti generiranja površin. Izbrali smo profil *Terestrial,* ki ima nastavitve, prikazane na sliki 47.

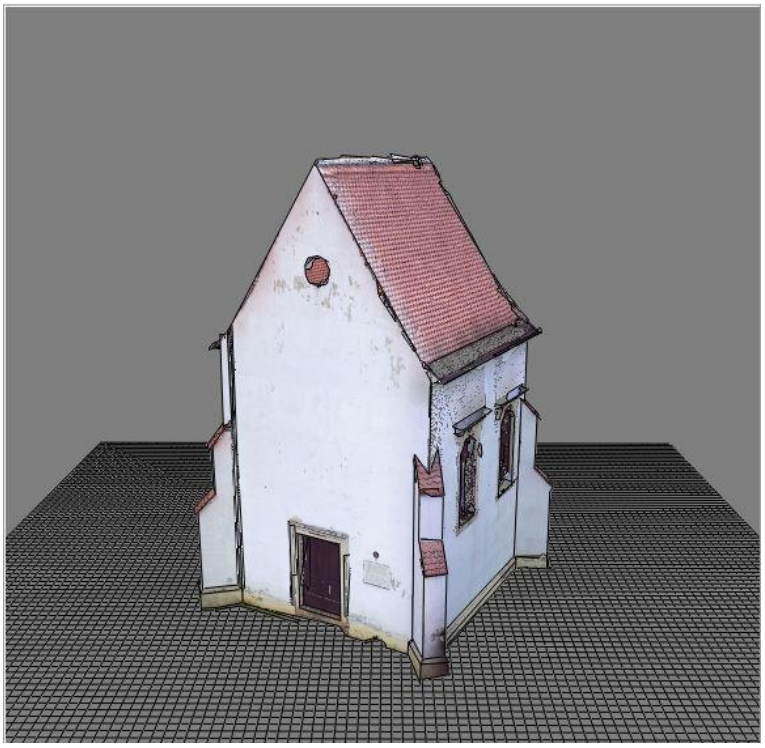


Slika 47: Nastavitve ukaza *Generate Surfaces*

Z zagonom ukaza program prične izdelovati model, kar lahko traja nekaj minut. Po končani operaciji se v *Project Explorer*-ju pod datoteko *Vector Models* pojavi ikona z izdelanim modelom, ki ga v pregledovalnik dodamo s pomočjo desnega miškinega klika in ukaza *Replace View.* Izdelani model je prikazan v treh slojih (layerjih): *Surface, Foliages* in *Edges.* Glavni sloji končnega modela so površine (*Surface),* ki so po privzetih nastavitvah prikazane enobarvno, in robovi (*Edges)*. Sloj *Foliages* vsebuje zelo majhne površine, ki ne bodo vključene v končni model. Dobljeni model je po privzetih nastavitvah prikazan enobarvno (slika 48). Za dodajanje teksture na model imamo v orodni vrstici ukaze za izbiro načina prikaza (*Solid Colour, RGB Texture, Intesity Texture, Ortogonal Distance Texture* i *Density Texture*), od katerih izberemo *RGB Texture.* Rezultat je 3D model z realističnimi teksturami, kar lahko vidimo na sliki 49.



Slika 48: Pridobljeni enobarvni model z vklopljenimi vsemi tremi sloji



Slika 49: Končni model z realističnimi teksturami, pridobljen v programu Pointfuse

Kot je razvidno iz dobljenega rezultata (slika 49), je model, pridobljen s programom Pointfuse, vizualno kakovosten in na prvi pogled zadovoljuje naše zahteve (vizualna prepoznavnost).

Končni model s teksturami lahko izvozimo v želeni format s pomočjo ukaza *File – Export*, pri katerem izberemo lokacijo shranjevanja in format. Izbrali smo format OBJ, v nastavitvah izvoza pa izberemo še, katero vrsto tekstur želimo izvoziti, v našem primeru *RGB.*

Vendar pa uporabnost modela, pridobljenega s programom Pointfuse, ni zadovoljiva zaradi pomanjkanja možnosti izračuna normal (poglavje 2: 3D Modeli in modeliranje). Program namreč ne nudi te možnosti, zato z uporabo takšnega modela v drugih programih dobimo nepopoln model, t. j. vse ploskve modela so obrnjene v isto stran, zato prikazan model ni pravilen. Primer te pomanjkljivosti lahko vidimo na spodnjem primeru, kjer smo model uvozili v igralni pogon (slika 50). Iz tega sledi, da model, pridobljen s programom Pointfuse, ne moremo uporabiti za izdelavo želene aplikacije, zato smo bili prisiljeni poiskati drug način izdelave modela.

Slika 50: Težava z modelom, izdelanim v programu Pointfuse, brez izračunanih normal

5.4 Izdelava modela v programu Meshlab

Meshlab je odprtokodni program za ustvarjanje, obdelavo in urejanje nestrukturiranih 3D triangulacijskih mrež (ang. mesh). Sistem je namenjen obdelavi tipičnih modelov večjega obsega, ki izvirajo iz 3D skeniranja, in ponuja sklop orodij za urejanje, čiščenje, pregled, upodabljanje in pretvorbo take vrste mrež.

Najpomembnejše značilnosti programa, pomembne za to nalogo so:

- Interaktivna selekcija in detekcija delov mrež s podporo velikih modelov
- Omogoča uvoz/izvoz velikega števila formatov 3D podatkov, med katerimi so:
    - Uvoz: PLY, STL, OFF, OBJ, 3DS, COLLADA, PTX, V3D, PTS, APTS, XYZ, GTS, TRI, ASC, X3D, X3DV, VRML, ALN
    - Izvoz: PLY, STL, OFF, OBJ, 3DS, COLLADA, VRML, DXF, GTS, U3D, IDTF, X3D
- Omogoča izračun normal vozlišča
- Samodejna izdelava mrežnega modela s pomočjo mnogih filtrov
- Teksturiranje modela iz obarvanega oblaka točk

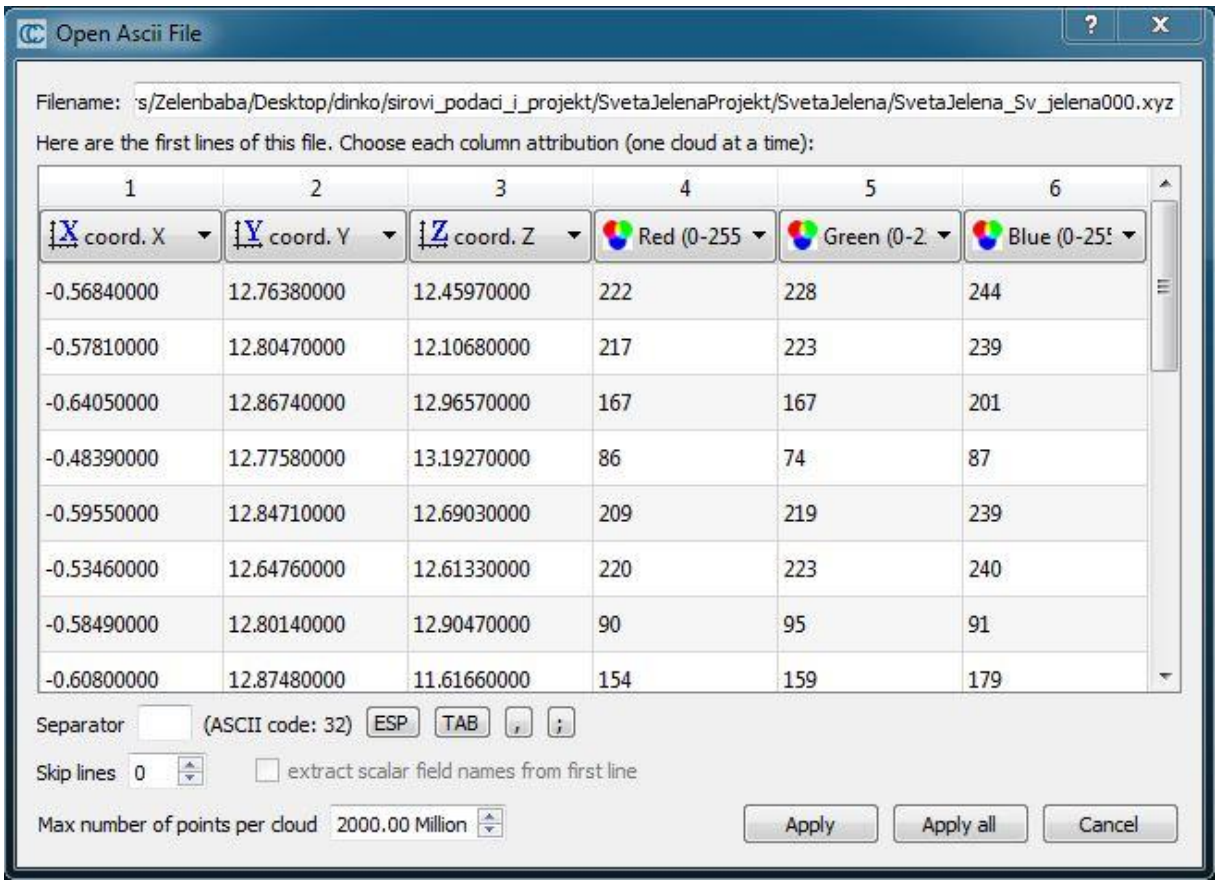
(Sourceforge.net, 2016)

Čeprav Meshlab omogoča delo z raznimi formati, je delo najenostavnejše s formatom PLY, ki ima najenostavnejši tekstualni zapis 3D podatkov, in sicer na prvih treh mestih prostorske koordinate, na drugih treh pa RGB sistem barv. Prav tako je delo s programom Meshlab izrazito zahtevno za strojno opremo, zato smo morali prilagoditi velikost (gostoto) oblaka točk z zmanjšanjem števila vključenih stojišč skeniranja. Vse navedeno smo storili med pripravo podatkov s pomočjo programa CloudCompare, ki je prav tako odprtokodni in je namenjen obdelavi oblakov točk ter omogoča delo z več oblaki hkrati.

5.4.1 Priprava podatakov s pomočjo programa CloudCompare

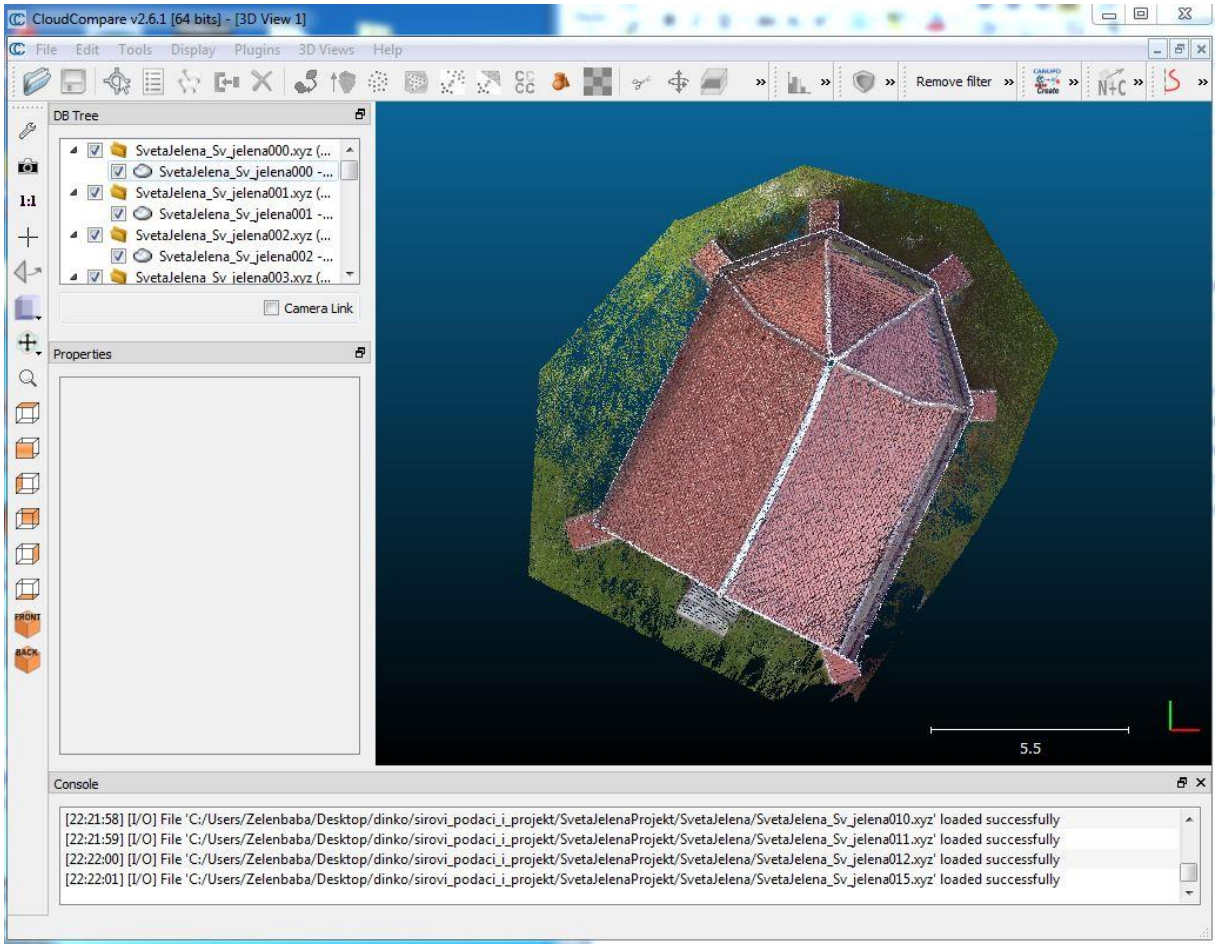
Pred uvozom skeniranega oblaka točk v program CloudCompare smo v programu SCENE izvozili registrirani oblak točk tako, da smo vsak sken izvozili v posamezno datoteko, pri čemer smo označili operacijo *Export each scan into separate file* (Poglavje 5.1.13, slika 25). Tako smo dobili 17 posameznih datotek z registriranimi skeni. Ker smo za obdelavo uporabljali poskusno verzijo SCENE, smo za redukcijo oblaka točk, uporabili odprtokodni program CloudCompare. Delo s programom CloudCompare je tudi bolj enostavno kot s programom SCENE.

V programu CloudCompare izvedemo uvoz podatkov s pomočjo ikone *Open* in označitvijo posameznih skenov. Po nekaj preizkusih zmogljivosti računalnika, je bila končna izbira stojišč: 0, 2, 3, 5, 7, 8, 10, 11, 12, 15. Izpustili smo stojišča 1, 4, 6 in 9, ki pokrivajo ista področja kot sosednja stojišča, ter 13 in 16, ki sta bila najbližje objektu in sta imela najgostejša skenograma in s tem tudi največ podatkov (Poglavje 5.1.2, slika 19). Po izbiri skenogramov nam program prikaže način zapisa uvoženega formata (slika 51). Uvozili smo XYZ Ascii format, ki smo ga izvozili iz programa SCENE.



Slika 51: Poizvedba programa CloudCompare po parametrih uvoženih podatkov

Po uvozu izbranih podatkov se nam prikažejo vsi oblaki točk hkrati. Ker so ti oblaki registrirani, izgledajo v prikazu kot en sam oblak, vendar lahko v levem oblaku *DB free* vidimo, da je sestavljen iz tolikšnega števila oblakov točk, kolikor smo jih uvozili v program.

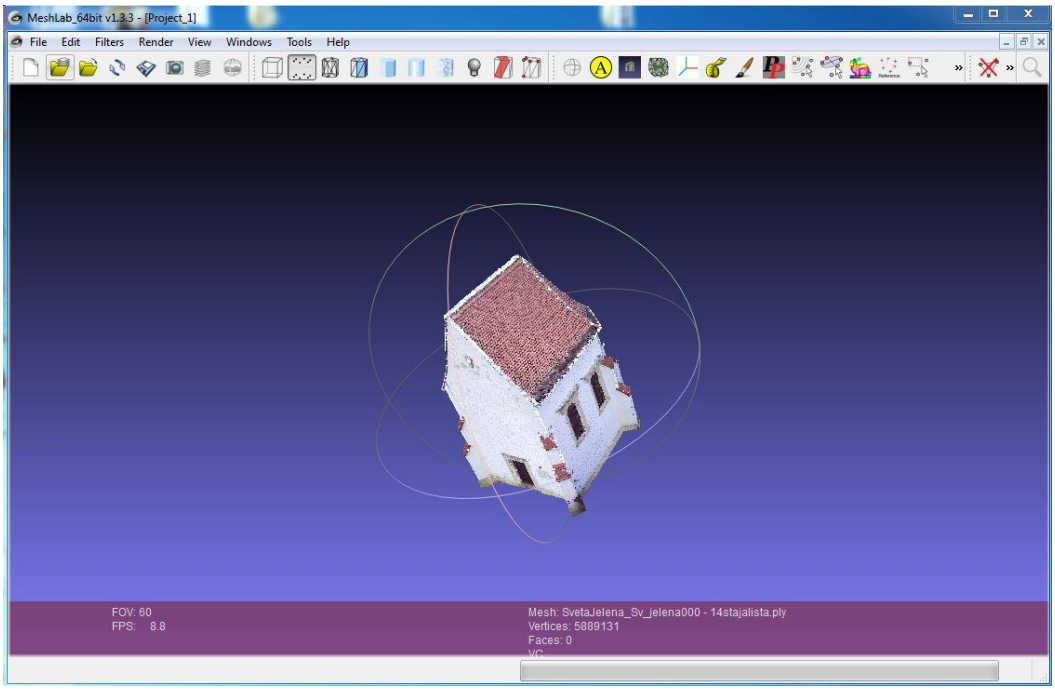


Slika 52: Izbrani oblaki točk v programu CloudCompare

Potem ko smo uvozili poljubno število oblakov točk glede na izbrane stojišča, smo jih združili v eden oblak točk s pomočjo ukaza *Merge multiple clouds.* Tako smo dobili en oblak točk, ki ga lahko dalje obdelujemo. Najprej smo izbrisali odvečne točke okoli skeniranega objekta, ki so nam ostale od obdelave v programu SCENE. To smo storili s pomočjo ukaza *Segment*, ki odpre orodno vrstico *Segmentation*, v kateri smo za izbrane dele oblaka točk uporabili ukaz *Segment out,* ki skrije označeni del točk, in potem *Confirm and delete hidden points,* ki izbriše skrite točke. Na koncu smo oblak točk izvozili v format PLY Ascii s pomočjo ukaza *Save.*

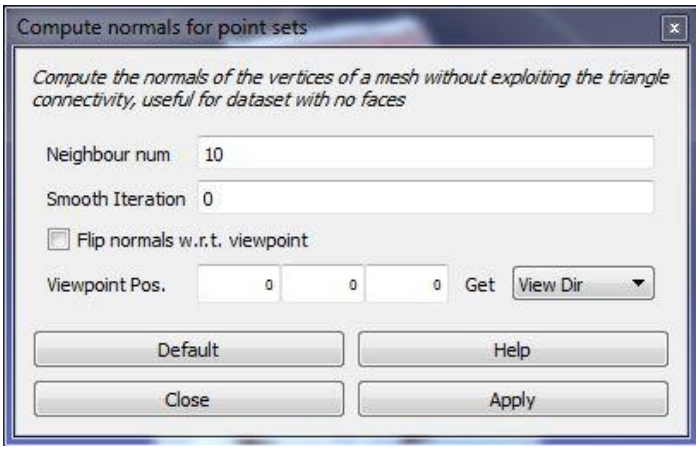
5.4.2 Obdelava s programom Meshlab

S predhodno obdelavo smo zmanjšali količino podatkov in tako zagotovili hitrejšno strojno obdelavo podatkov v programu Meshlab. Za primerjavo, datoteka PTS, s katero smo delali obdelavo v programih Kubit in Pointfuse, je velika okoli 400 MB, medtem ko je datoteka PLY Ascii, ki smo jo uporabili v Meshlabu, velika okoli 130 MB. Uvoz podatkov se v Meshlab-u izvede s pomočjo ukaza *Import Mesh,* po katerem se nam oblak točk samodejno prikaže v vmesniku programa (slika 53).



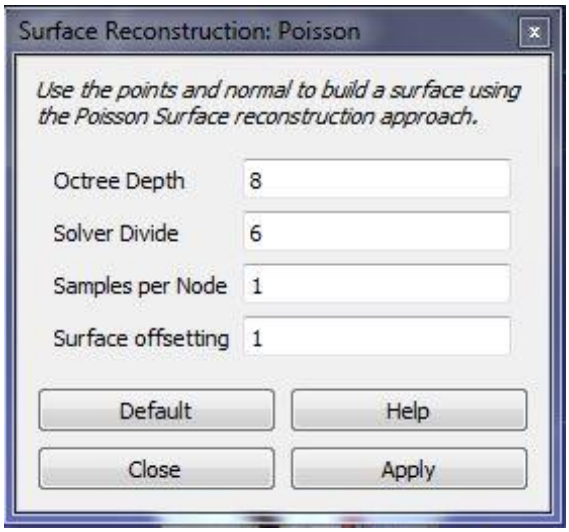
Slika 53: Vmesnik programa Meshlab z uvoženim oblakom točk

Prvi korak pri izdelavi modela je izračun normal. To smo izvedli s pomočjo ukaza *Filters - Normals - Curvatures and Orientation - Compute normals for point sets* (slika 54).



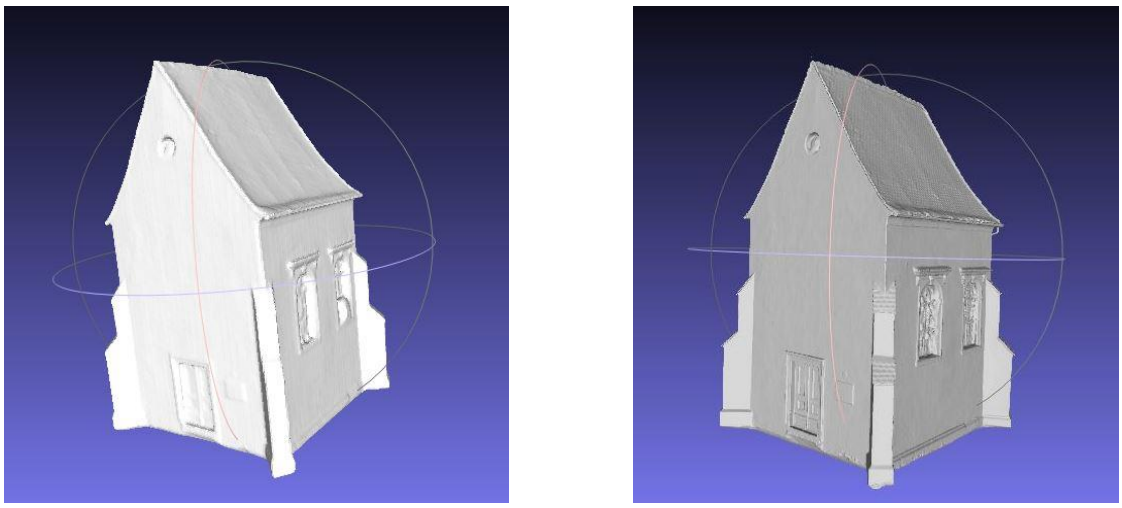
Slika 54: Nastavitve izračuna normal

Ko program izračuna normale, se lahko lotimo konstrukcije modela s pomočjo več načinov, ki jih nudi program Meshlab. Izbrali smo filter, ki je priporočen kot največkrat uporabljen, s pomočjo ukaza *Filters - Remeshing, Simplification and Reconstruction – Surface Reconstruction: Poisson.* Ta filter s pomočjo izračunanih normal in točk ter ustreznega algoritma ustvarja triangulacijsko mrežo površin. Med nastavitvami filtra je najpomembnejši parameter *Octree Depth*, s katerim določamo podrobnost končnega modela. Večja kot je vrednost, bolj podroben bo model. Program priporoča vrednost med 5 in 10. Čeprav je v našem primeru uspela tudi obdelava z vrednostjo 10, smo se za končni model odločili uporabiti vrednost 8 (slika 55), ki je glede na vizualni pregled dala popolnoma zadovoljive rezultate končnega modela (MeshLAB Tutorial, 2009).



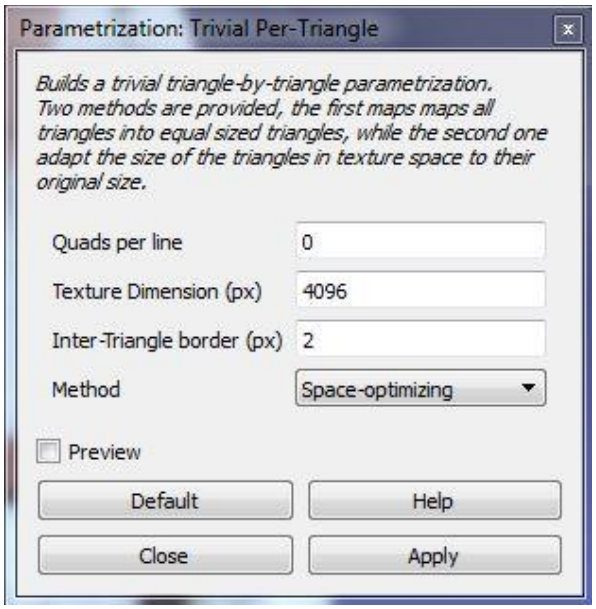
Slika 55: Nastavitve filtra *Poisson Surface Reconstruction*

Rezultat izbranega filtra je enobarvna mreža poligonov, razlika v podrobnosti zaradi različnega parametra *Octree Depth* pa je vidna na sliki 56.



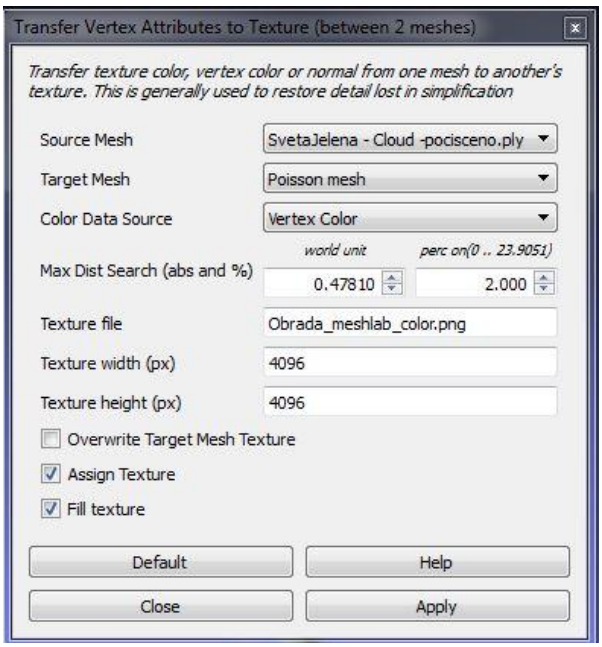
Slika 56: Pridobljeni "Mesh" z različno vrednostjo parametra *Octree Depth* - levo=8, desno=10

Za dodajanje tekstur smo najprej naredili parametrizacijo, ki je sestavljena iz dveh metod. Prva mapira vse trikotnike v trikotnike enake velikosti, druga pa v teksturi prilagaja trikotnike njihovi originalni velikosti. Ukaz *Filters - Texture - Parametrization: Trivial per – Triangle* (slika 57).



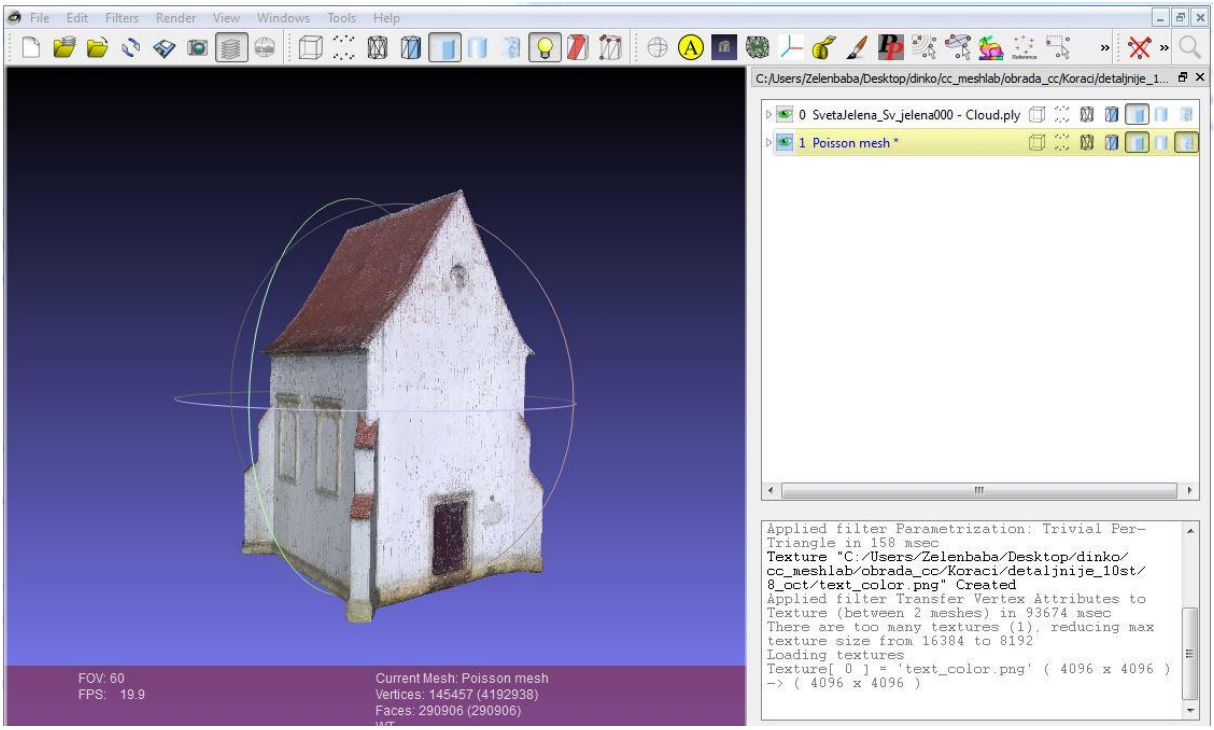
Slika 57: Nastavitve parametrizacije *Trivial per - Triangle*

Pri naslednjem koraku smo teksture prenesli direktno iz oblaka točk na pridobljeno mrežo s pomočjo ukaza *Filters – Texture – Transfer Vertex Attributes to Texture (between 2 meshes)* (slika 58).



Slika 58: Nastavitve ukaza *Transfer Vertex Attributes to Texture*

S tem smo dobili 3D model kapelice Svete Jelene s teksturami, ki je vizualno prepoznaven in ga bomo v nadaljevanju uporabili za izdelavo aplikacije (slika 59).



Slika 59: Končni 3D model, pridobljen s pomočjo programa Meshlab

Izdelani model je treba le še izvoziti v želeni format, kar se izvede s pomočjo ukaza *File – Export Mesh As.* Izbrali smo format OBJ. Model se izvozi v tri datoteke:

- datoteka OBJ: vsebuje 3D model brez tekstur
- datoteka mtl: tekstovna ASCII datoteka, ki opisuje material modela v datoteki OBJ, s katero je povezana
- datoteka PNG: vsebuje teksture

5.5 Primerjava uporabljenih programih za izdelavo modela

Kot je že napisano prej v nalogi, smo se za program Kubit PointSense Heritage odločili zaradi tega, ker je dodatek k nam znanemu programu AutoCAD. Med primerjanimi programi je Kubit edini v katerem se model popolnoma ročno izdeluje (riše) na oblaku točk, ostala dva (Pointfuse in Meshlab) imata avtomatsko generiranje površin iz oblaka točk in tako avtomatsko izdelavo 3D modela. Zaradi tega razloga je izdelava modela v Kubitu dolgotrajen in zahteven proces. Seveda pa je na koncu izdelek, izdelan s programom Kubit PointSense Heritage, v podrobnostih in vizualno najboljše kakovosti. V primerjavi z ostalima dvema programoma lahko rečemo, da je Kubit PointSense Heritage za uporabnika najbolj zahteven program, za izdelavo modela je potrebno veliko več časa kot v drugih dveh omenjenih programih, vendar je na koncu rezultat tudi boljši.

Program Pointfuse je najenostavnejši program za uporabo, in model izdelamo v zelo kratkem času, v našem primeru v nekaj minutah. Uporabniku ni potrebno poznati in razumeti procese, ki tečejo v ozadju programa. Lepljenje tekstur je v tem programu, v primerjavi z drugima dvema programoma, najlažje in tudi po subjektivnem vtisu so te teksture dobre kakovosti. Po drugi strani pa je dobljeni model, gledano celostno, najslabše kakovosti in ima veliko pomanjkljivosti. Končni model ima na svoji površini majhne praznine, katere v postopku generiranja površin niso bile zapolnjene. Največja pomanjkljivost tega programa pa je v tem, da program ne omogoča izračuna normal vozlišč.

Meshlab, s katerim smo na koncu izdelali končni model, je odprtokodni program z veliko možnosti in ponujenih algoritmov za obdelavo oblakov točk in izdelavo modelov ter njihovo filtriranje ipd. Mi smo preizskusili samo en majhen del možnosti, ki jih program ponuja. Pri uporabi tega programa, mora uporabnik, za razliko od programa Pointfuse, poznati funkcije, in parametre teh funkcij, ki jih uporablja. Čeprav je celotni proces dokaj zahteven in traja več časa kot s programom Pointfuse, je še vedno precej bolj enostaven in krajši od obdelave s programom Kubit PointSense Heritage. Model, pridobljen s programom Meshlab, je popoln in nima pomanjkljivosti za nadaljnjo uporabo v drugih programskih paketih. Tudi vizualna kakovost je subjektivno nekoliko boljša od modela, pridobljenega s programom Pointfuse. Kakovosti končnega modela, izdelanega s programom Kubit PointSense Heritage, ne moremo realno oceniti, ker ga nismo dokončali.

6 IZDELAVA APLIKACIJE ZA NAVIDEZNO "SPREHAJANJE" OKOLI 3D MODELA Z UPORABO IGRALNEGA POGONA UNREAL ENGINE 4

V tem poglavju opisujemo končni izdelek te naloge, to je izdelava aplikacije, v kateri se uporabnik lahko prosto giblje v vizualiziranemu prostoru okoli 3D modela kapelice Svete Jelene, ki smo jo poskenirali s terestričnim laserskim skenerjem. Model, ki smo ga uporabili za izdelavo aplikacije, je model, izdelan s programom Meshlab v zapisu OBJ, saj je združljiv z izbranim igralnim pogonom. Pri iskanju načina, s katerim bi lahko realizirali idejo, da izdelamo aplikacijo za sprehajanje okoli nekega modela, smo prišli do zaključka, da bi lahko poskusili to izdelati na način kot se izdelujejo video igre, to je z uporabo igralnega pogona. Ker obstaja veliko število igralnih pogonov, smo za optimalno rešitev preiskusili nekaj teh programskih paketov, podobno kot pri modeliranju. Med igralni pogoni, ki smo jih preiskusili, so tudi Autodeskov Stingray in Crytekov CryEngine 3. Na koncu smo se odločili za programsko orodje Unreal Engine 4 zaradi njegovega uporabniku prijaznega okolja, posebej za začetnike v razvoju video iger, zato opis dela z drugima dvema programoma ne bomo opisovali.

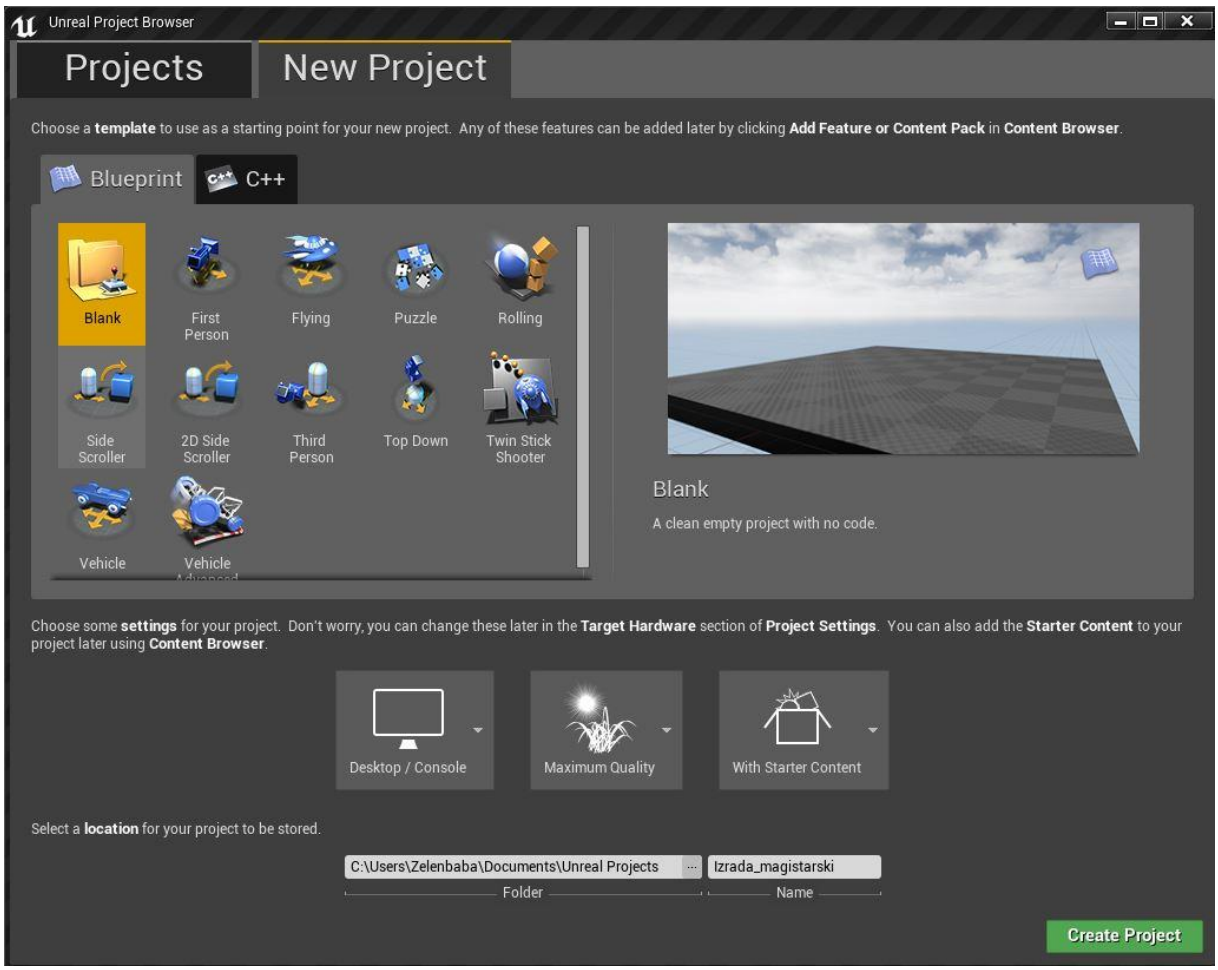
„Unreal Engine 4 je kompletni paket orodij za razvoj iger, ki so ga razvili razvijalci iger za izvajalce iger. Uporabniku omogoča izdelavo česar koli, od najenostavnejše 2D mobilne igre do »blockbusterja« za znane konzole z visoko stopnjo občutenja navidezne resničnosti.“ (Unreal Engine, 2016.) Razvilo ga je podjetje za razvoj video iger Epic Games. Čeprav je bil prvotno razvit za FPS (poglavje 3.3.1.), se danes zelo uspešno uporablja v različnih igrah in aplikacijah.

Čeprav je Unreal Engine 4 izdelan na način, da se začetniki čim lažje znajdejo in sicer tako, da ima znotraj programa že izdelane vodiče, ki uporabnika vodijo po korakih čez izdelavo, je sam program zelo kompleksen in izdelava video igre od uporabnika zahteva določanje velikega števila nastavitev za delovanje končne aplikacije oz. video igre. Ker je naš cilj izdelati samo virtualen sprehod okrog objekta in ne video igro, ki bi vsebovala razne efekte, animacije, naloge ipd., je v nalogi zajeta samo izdelava najosnovnejšega nivoja. Pokazali bomo način izdelave najosnovnejšega nivoja video igre z uporabo elementov, brez katerih niti ta najosnovnejši nivo ne more delovati. Glede na to, da želimo dobiti aplikacijo kot prvoosebno video igro, je potrebno definirati avatarja (poglavje 3.3.1.). Pomembno je, da imamo avatarja že oblikovanega, poleg tega pa moramo imeti definirane vrste animacij in ostale podrobnosti kot je način upravljanja z avatarjem, hitrost njegovega gibanja ipd. Oblikovanje avatarja in njegovih lastnosti bi bilo zelo obširno za to nalogo, zato tega nismo vključili, smo se pa odločili, da uporabimo že izdelan primer lika za prvoosebni nivo, ki nam

ga igralni pogon ponuja pri kreiranju projekta. V tem primeru ima določeni lik, kot avatar, že definirane z vse potrebne lastnosti.

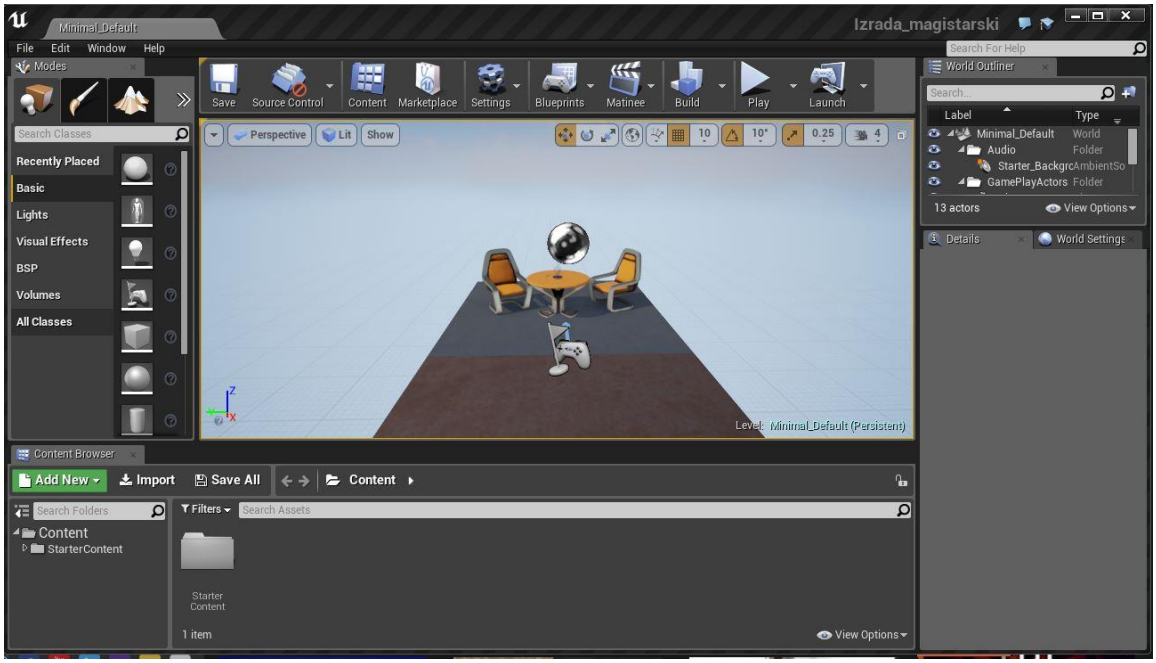
6.1 Uvoz podatkov in izdelava osnovnega nivoja

Pri zagonu samega igralnega pogona moramo biti registrirani z uporabniškim imenom in geslom. Po avtorizaciji in zagonu programa se nam odpre okno, kjer izbiramo med že obstoječimi projekti ali želimo ustvariti novi projekt (New Project). Če izberemo možnost za ustvarjanje novega projekta, poleg lokacije, kjer bo projekt shranjen in imena projekta, izbiramo tudi osnovne nastavitve (način kodiranja, začetni nivo, kakovost ipd.).



Slika 60: Ustvarjanje novega projekta v Ureal Engine 4

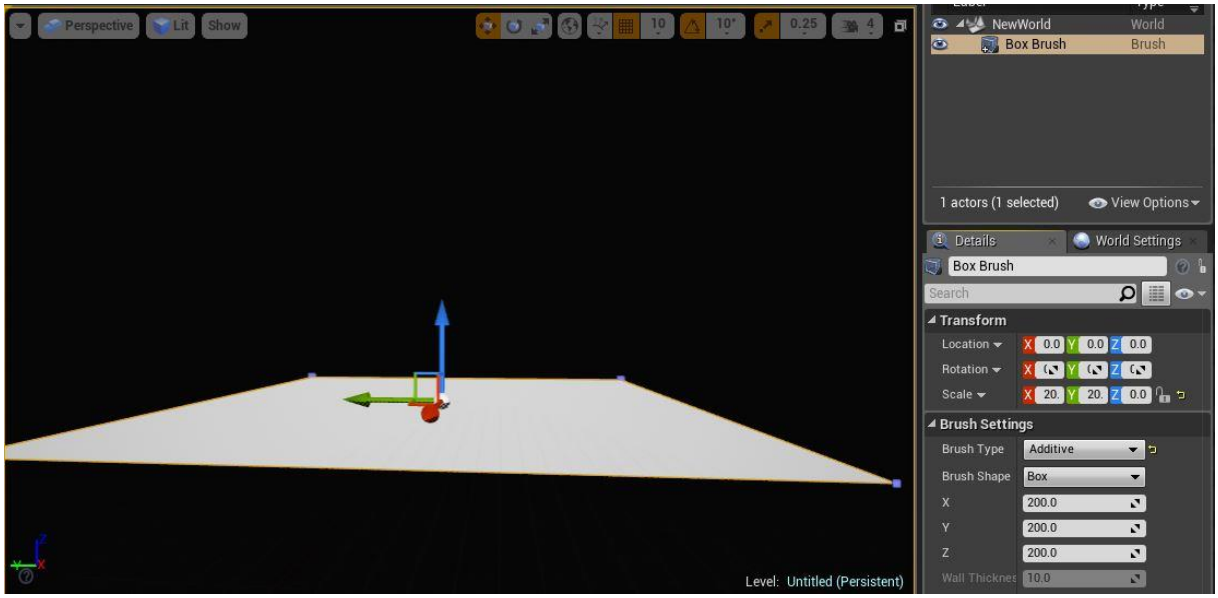
Kot lahko vidimo na sliki 60, že pri ustvarjanju projekta program ponuja že nekatere v naprej nastavljene primere, katere lahko izberemo in zaženemo ter začnemo iz njih razvijati svojo verzijo. Izbrali smo prazen projekt za samostojno izdelavo osnovnega nivoja. Potem, ko ustvarimo projekt, se nam odpre delovno okolje programa (slika 61).



Slika 61: Glavni vmesnik programa Unreal Engine-a

Najprej smo ustvarili nov nivo, ki bo v celoti prazen, s pomočjo ukaza *File – New Level – Empty Level.* V praznem nivoju začenjamo s postavitvijo objektov oz. likov. „Lik (angl. Actor) je katerikoli objekt, ki ga lahko postavimo v določen nivo. Likovi so generični razredi, ki podpirajo 3D transformacije kot so translacija, rotacija in sprememba merila.“ (Unreal Engine 4 Documentation, 2016a.)

Likovi so tako lahko kar koli, od geometrijskih teles, igralnih likov, kamere, izvirov svetlobe, do vizualnih efektov ipd. Naš prvotni lik je bila podlaga, ki smo jo postavili na način, da smo v ukazni vrstici *Modes*, pod zavihkom *Place,* nastavili skupino *BSP,* kjer so nam ponujeni osnovni geometrijski elementi. Izbrali smo kocko – *Box* in jo postavili v okno za urejanje. Lokacijo kocke smo v oknu *Transform* postavili v samo izhodišče z koordinatami X=0.0, Y=0.0, Z=0.0 pod zavihkom *Location*, velikost kocke pa smo določili s pomočjo nastavitve *Scale* v razmerju X=20.0, Y=20.0, Z=0.1 (slika 62). Te nastavitve se lahko spreminjajo in prilagajajo kadar koli tako, da velikost podlage, ki jo uporabljamo, ni fiksna. Tu lahko že vidimo, da se v igralnem pogonu dela v 3D koordinatnem sistemu in se vsak objekt lahko postavi na natančno določene poljubne lokacije.



Slika 62: Podlaga v praznem nivoju

Potem, ko smo približno definirali podlago, na kateri bomo gradili nivo, smo v naslednjem koraku, zaradi lažje orientacije v navideznem prostoru, določili izvor svetlobe, ki je zelo pomemben pri kasnejšem prikazu našega 3D objekta v navideznem okolju, ki ga ustvarjamo.

Določili smo:

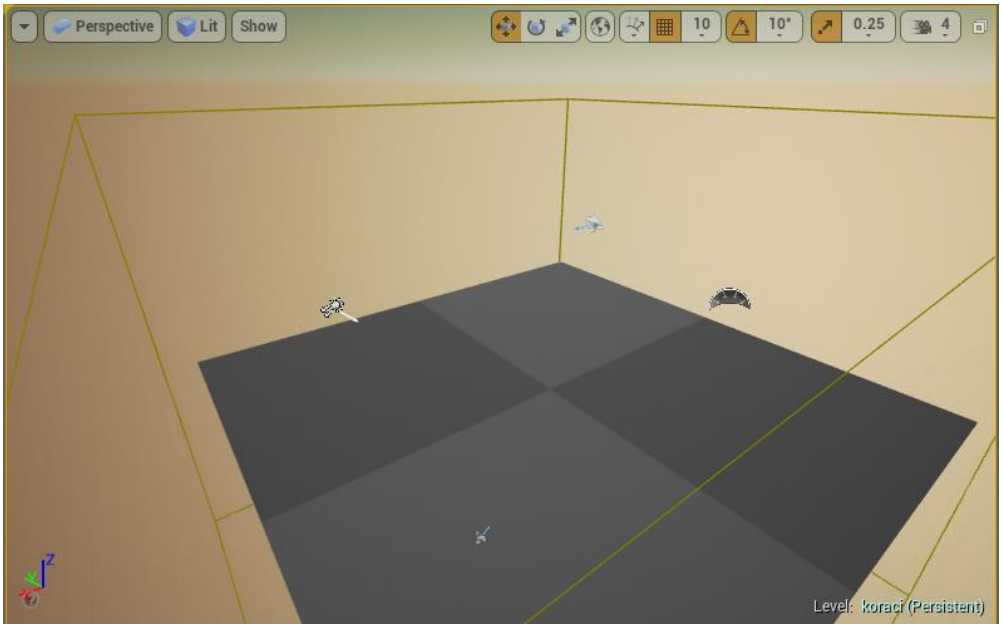
- direktno sončno svetlobo – skupina ukazov *Lights*, ukaz *Directional Light;*

- svetlobo, pridobljeno kot odsev neba –skupina ukazov *Lights*, ukaz *Sky Light.*

Določene likove svetlobe postavljamo na isti način, kot smo tudi podlago in sicer tako, da jih povlečemo v okno za urejanje. Lokacijo smo določili poljubno, smer pa je bila usmerjena proti središču podlage. Bolj točno lokacijo in smer bomo določili kasneje.

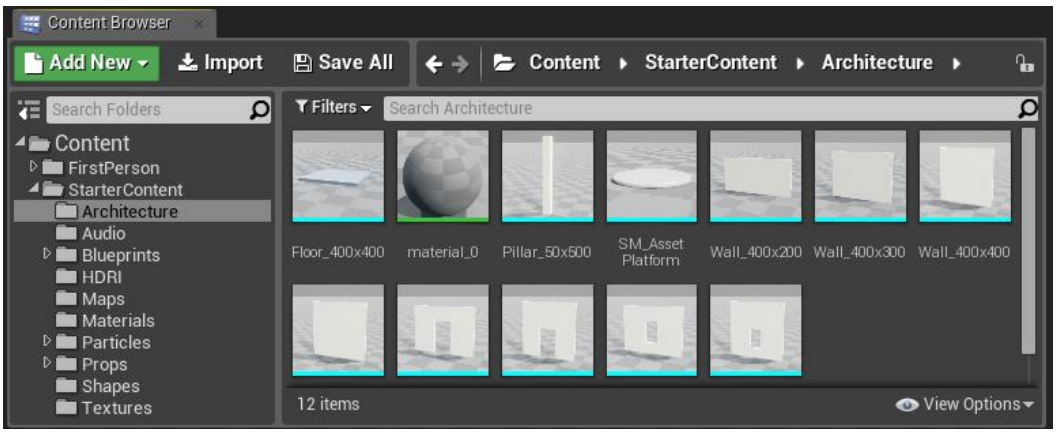
Naslednji lik, ki smo ga dodali v nivo, je vizualni efekt neba, da ozadje ne bi bilo črno. Na enak način kot prej smo uporabili možnost *Atmospheric Fog* v skupini *Visual Effects*. Pod zavihki *Directional Light*, ki smo jih prej postavili, smo vključili *Atmosphere Sun Light*, da nebo dobi ustrezno barvo glede na lokacijo direktne sončne svetlobe.

Pred vizualnim urejanjem našega nivoja je bilo potrebno definirati tudi začetni položaj uporabnika s postavljanjem lika *Player Start* v skupini ukazov *Basic*, in „*LightMass Importance Volume*“ lika v skupini *Volumes.* „*LightMass Importance Volume*“ se uporablja za kontrolo in koncentracijo svetlobe ter ustvarjanje senčenja znotraj mej definiranega volumna. S postavitvijo omenjenega lika v okno za urejanje je potrebno ročno nastaviti meje volumna na način, da vsi določeni likovi padejo znotraj definiranega območja.



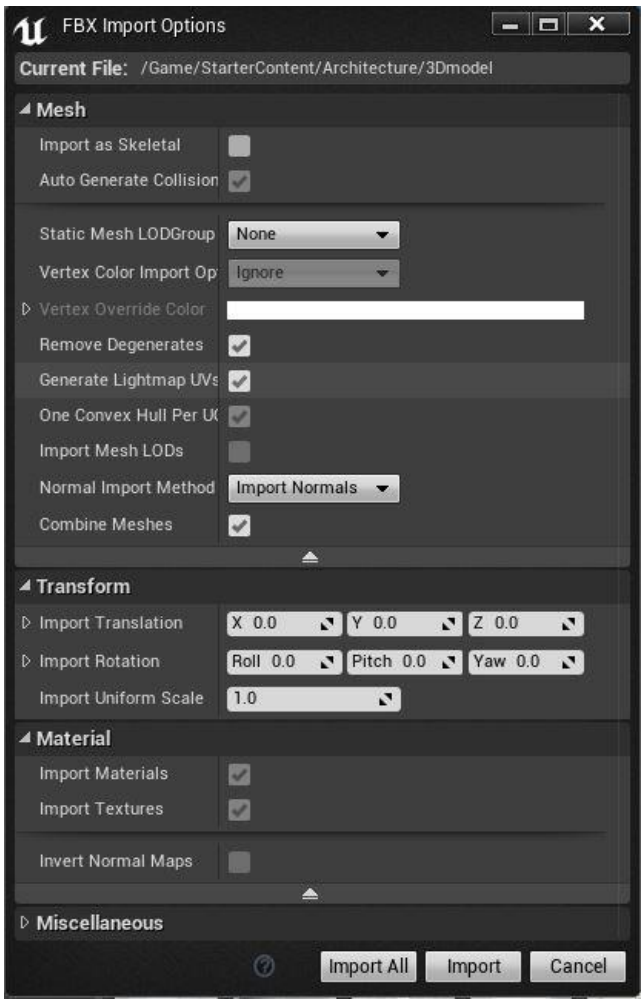
Slika 63: Nivo z definiranimi likovi potrebnimi za delovanje aplikacije

V zadnjem koraku smo končali s postavljanjem likov, ki so potrebni za delovanje aplikacije (slika 63), in smo začeli z vizualnim urejanjem nivoja. Že definirani podlagi smo dodali teksturo trave (označi se podlaga in v oknu *Details*, kategorija *Surface Materials*, zavihek *M_Ground_Grass*). Dodali smo tudi zid, ki bo določal meje našega nivoja. Zid kot lik dodamo iz okna *Content Browser* in datoteke *Architecture*. Okno *Content Browser* ima že samodejno organizirane datoteke z najosnovnejšo vsebino, ki jo lahko uporabljamo pri izdelavi nivoja iz konstruktivnih delov kot so zidovi, okna ipd., do tekstur, zvočnih efektov ipd. V *Content Browser* oknu se lahko uvozi tudi lastni material kot so 3D objekti, že izdelane like, teksture ipd. (slika 64)



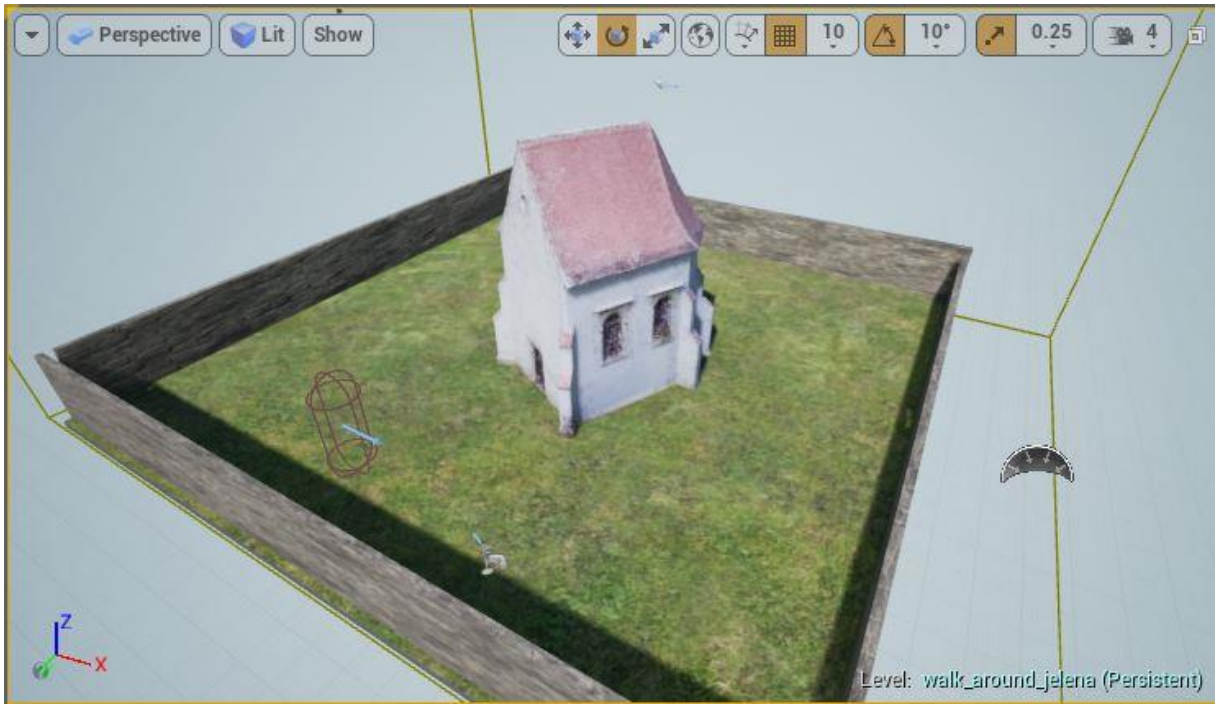
Slika 64: Content Browser

S pomočjo ukaza *Import* v *Content Browser*-ju smo uvozili naš 3D model. Ob zagonu ukaza Import izbiramo istočasno datoteke OBJ in PNG, ki smo jih pridobili s programom Meshlab. Pri izbiri teh datotek se nam ponudi okno z nastavitvami, ki ga lahko vidimo na sliki 65.



Slika 65: Nastavitve uvoza 3D modela

S potrditvijo teh nastavitev smo naredili uvoz datoteke modela in tekstur v izbrano mapo *Content Browser*-ja. Model smo dodali kot lik v naš nivo na enak način kot vse dosedanje, se pravi s potegom ikone 3D modela v okno za urejanje. Teksture smo na 3D model dodali na enak način z izbiro ikone, ki smo jo uvozili skupaj s 3D modelom. Ker v nastavitvah uvoza nismo podajali koordinat lokacije in merila (angl. Scale) za naš 3D model, smo ga dodali prostovoljno v oknu za urejanje, ter v oknu *Details* pod zavihkom *Scale* povečali 100 krat. To smo naredili, ker je po že določenih nastavitvah koordinatni sistem v programskem okolju Unreal Engine 4 v centimetrih, medtem ko so 3D koordinate našega modela v metrih. S postavitvijo našega 3D modela v okno za urejanje smo dobili končni izgled našega nivoja (slika 66).



Slika 66: Končni izgled nivoja

Pred zagonom vizualno izdelanega nivoja je potrebno zagnati tudi avtomatski proces izgradnje nivoja samega igralnega pogona, ki preračuna podatke svetlobe in vidljivosti, generira navigacijske mreže in ostale potrebne podatke v ozadju. Da bi dobili čim kakovostnejši izgled svetlobe in refleksije v našemu modelu, je bilo potrebno povečati ločljivost osvetlitve našega modela, kajti v nasprotnem primeru bi 3D model imel črne madeže. To smo naredili tako, da smo odprli *Mesh Editor* znotraj igralnega pogona z dvakratnim klikom na ikono uvoženega 3D modela u *Content Browser*-ju, ter v oknu *Details* v ukazni vrstici *Static Mesh Settings,* zavihek *Light Map Resolution* vnesli namesto nastavljenih 64 vrednost 512, in shranili spremembe.

Light Map Resolution nam omogoča kontrolo podrobnosti osvetlitve, ki jo pridobivamo na podlagi statičnih izvorov svetlobe na določenih modelih. Višje vrednosti pomenijo večjo ločljivost, vendar tudi zavzamejo več prostora na računalniku in podaljšajo čas samega procesa izgradnje. (Unreal Engine 4 Documentation, 2015b)

Proces samostojne izdelave nivoja smo zagnali z ukazom *Build* na glavni orodni vrstici okna za urejanje. Ta proces traja dlje časa, posebej zaradi tega, ker smo izbrali zelo veliko ločljivost osvetlitve. Po končanem procesu izdelave smo dobili končni nivo, ki ga pred izvozom v samostojno aplikacijo lahko preiskusimo znotraj igralnega programa in sicer z zagonom ukaza *Play,* ki se tudi nahaja v glavni orodni vrstici okna za urejanje. S preizkusom

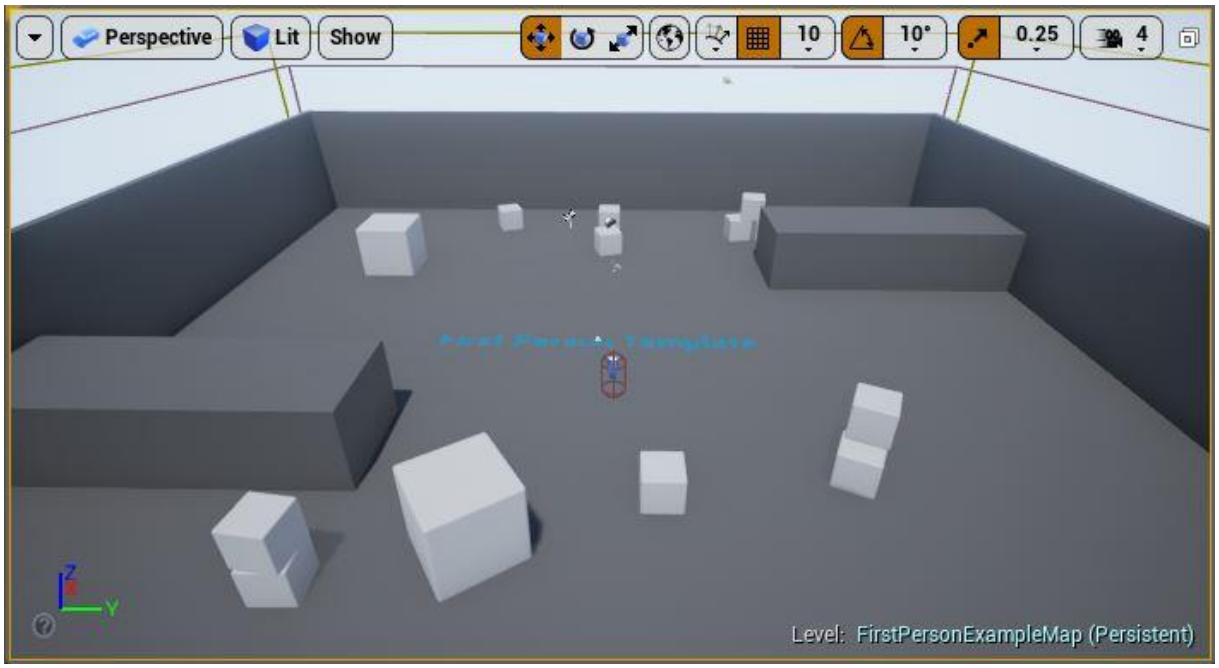
v samem igralnem pogonu, dobimo vpogled v končni izgled video igre, v tem primeru aplikacije sprehoda, ki jo bomo dobili, ko določeni nivo izvozimo kot samostojno aplikacijo. Da bi dobili pravi virtualni sprehod v tem prostoru, moramo še definirati lik, ki bi ga predstavljal naš avatar in s pomočjo katerega bi imeli omejeno hitrost gibanja glede na hitrost realnega sprehoda in zorni kot gledanja navidezne osebe. V tem primeru, ko nimamo definiranega lika prve osebe kot udeleženca (avatarja), je zorni kot oz. pogled določen s zornim kotom kamere, gibanje v prostoru pa je določeno kot prelet skozi zgrajeni prostor (slika 67.), ne pa kot simulacija sprehajanja. Čez prostor se gibljemo s pomočjo tipk na tipkovnici in sicer:

- Tipka W – gibanje naprej
- Tipka D – gibanje v desno
- Tipka S – gibanje nazaj
- Tipka A – gibanje v levo
- Kontrola z miško – prilagajanje kota pogleda

Slika 67: "Screenshot" podobe testiranega nivoja znotraj igralnega pogona

6.2 Izdelava končne aplikacije in izvoz aplikacije

Da bi dobili simulacijo pravega sprehoda z določeno hitrostjo sprehajanja in definiranim likom oz. avatarjem, smo se odločili za uporabo že pripravljenega primera prvoosebnega nivoja. V prvoosebni opciji novega projekta tako imamo že definiran osnovni nivo video igre (slika 68). Ponujeni nivo vključuje vse potrebne like za funkcioniranje video igre, ki smo jih na primer v prejšnjem praznem primeru morali izdelati, ter ob tem definiran tudi lik v obliki robotske roke, ki ga uporabljamo kot avatarja v naši prvoosebni aplikaciji.



Slika 68: Izgled ponujenega primera *prvoosebnega* nivoja v igralnem pogonu

Nivo smo vizualno uredili na enak način kot predhodni primer. Dodatno pa smo izbrisali odvečne objekte, ki obstajajo v izbranem primeru. Osvetlitev in ostale efekte smo prilagodili našim potrebam na način, da smo jim spremenili lokacijo, smer in merilo. Vse ostale lastnosti smo ponovili kot v izdelavi osnovnega nivoja. Pri končnem koraku avtomatske izdelave nivoja (*Build* procesa) smo dobili končni izgled aplikacije (slika 69.).

Slika 69: Končni izgled aplikacije virtualnega sprehoda okoli 3D modela kapelice Svete Jelene

Zadnji korak pri izdelavi aplikacije je bil izvoz (pakiranje) za določeno programsko okolje, da imamo kasneje možnost uporabe aplikacije neodvisno od igralnega pogona. Unreal Engine podpira pakiranje video iger za različne konzole oz. platforme med katerima so: Android, iOS, PlayStation 4, Windows in ostale. Pred izvozom aplikacije v prvem primeru smo pri izvozu morali v nastavitvah igre nastaviti (Edit – Project Settings – Maps & Modes) Game Default Map in izbrati FirstPersonExampleMap, da bi definirali način na katerega bo igra (aplikacija) delovala. V drugem primeru (končnem izdelku) smo ta korak izpustili, saj so potrebne nastavitve nastavljene avtomatsko.

Izvoz projekta v samostojno aplikacijo smo izvedli s pomočjo ukaza *File – Package – Project – Windows – Windows (64-bit)*. Končni izdelek je mapa z aplikacijo in spremljajočimi datotekami, ki jo lahko zaženemo na katerem koli 64-bitnem operacijskem sistemu Windows. (Unreal Engine 4 Documentacion, 2016c)

7 ZAKLJUČEK

Ob uspešni izdelavi aplikacije navideznega sprehoda okoli 3D modela, ki smo ga izdelali iz podatkov terestričnega laserskega skeniranja, ter ob kratki teoretični razlagi, smo uspeli dokazati na začetku zastavljene hipoteze. Končna aplikacija je izdelana v igralnem pogonu, v katerem izdelujejo video igre, uporabljeni pa so podatki, pridobljeni iz realnega okolja s pomočjo terestričnega laserskega skenerja, ter na podlagi realnih barv, pridobljenih z integriranim fotoaparatom v laserskem skenerju, kar potrjuje prvotno postavljeno hipotezo te naloge, da geodezija in geoinformatika ter podrobneje fotogrametrija in daljinsko zaznavanje lahko prispevajo k industriji video iger. Iz omenjenega lahko zaključimo, da je potrebna vključitev geodezije in geoinformatike v prikaz navidezne resničnosti, kjer se uporabljajo podatki iz realnega okolja. S tehnološkim razvojem se zelo povečujejo možnosti pridobitve velikih količin podatkov v zelo kratkem času in njihovih različnih vrst uporabe.

Kot je že zgoraj omenjeno smo uporabili 3D model, ki smo ga izdelali iz oblaka točk, pridobljenega s terestričnim laserskim skeniranjem, kar potrjuje drugo hipotezo te naloge, to je, da se 3D modeli, pridobljeni s terestričnim laserskim skeniranjem, lahko uporabljajo v izdelavi videoiger, tako da prostor navidezne resničnosti gradijo iz podatkov realnega prostora. Pokazali smo tudi, da se 3D model oblaka točk lahko izdela na več načinov in sicer: s pomočjo mnogih že obstoječih programskih paketov ter, da je z vidika uporabe navidezne resničnosti potrebno upoštevati ustreznosti formatov končnih 3D modelov. Dodatno smo ugotovili tudi to, da je pri izdelavi 3D modela za uporabo v navidezni resničnosti pomembna tudi vizualna kakovost končnega modela in ne le tehnična točnost posameznih elementov na 3D modelu. V ta namen lahko uporabljamo programska orodja, ki avtomatsko generirajo ploskve 3D modela iz oblaka točk, katerih točnost verjetno ni zadovoljiva, če bi izdelek uporabljali v rekonstrukciji fasade ali dokumentiranju kulturne dediščine. Seveda to ne pomeni, da za isti namen ne moremo uporabljati bolj precizne in detajlnejše izdelane 3D modele, če bi za to imeli ustrezne pogoje glede zahtevanega časa meritev in obdelave ter zmožnosti strojne opreme.

Končni izdelek te naloge ni izdelava video iger, ampak izdelava aplikacije, v kateri se lahko prosto navidezno sprehajamo okoli objekta iz realnega prostora v navidezni resničnosti. S tem smo na nek način pridobili majhno multimedijsko karto prostora okrog kapelice Svete Jelene in tako lahko potrdimo tudi zadnjo hipotezo in sicer, da se na način, kot se izdelujejo video igre, lahko izdelajo tudi multimedijske (večpredstavnostne) karte. V tej nalogi je prikazan sprehod okoli enega objekta. Seveda lahko na enak način skeniramo na primer celoten center oz. staro jedro nekega mesta ter na ta način izdelamo navidezni sprehod v večjem prostoru, kar se lahko šteje kot multimedijska karta. Igralni pogoni imajo široke

zmožnosti mi smo pa v tej nalogi pokazali le osnovne in izdelali najenostavnejši nivo. Kot je prikazano v teoretičnem delu naloge, je možno izdelati video igre s spremljajočimi virtualnimi kartami, ki prikazujejo zgrajeni svet znotraj navideznega prostora s trenutno lokacijo igralca oziroma sprehajalca, če govorimo o aplikaciji sprehajanja. Pokazali smo, da tudi igralni pogoni delajo na podlagi koordinatnih sistemov kar pomeni, da lahko poleg vsega ostalega izdelamo aplikacijo, s katero se sprehajamo v virtualnem prikazu realnega prostora kot je Google Street View, na način kot je v videoigrah, kjer dobimo virtualni vtis realnega sprehajanja.

Glede same praktične izvedbe naloge lahko zaključimo, da so nove tehnologije vse bolj enostavne za uporabo, kot je v našem primeru omenjen laserski skener FARO Focus3D x330, kjer uporabniku ni potrebno imeti veliko znanja o tehnologiji, ki jo uporablja, ampak le njen namen in svoj lastni cilj. Vendar morajo imeti ob tem uporabniki sposobnost kritične presoje kakovosti in ustreznosti končnih izdelkov. Skener je zelo hiter in tih pri skeniranju. Pridobljeni podatki so kakovostni in detajlni. V naši nalogi smo končni oblak točk zelo reducirali in s tem zmanjšali njegovo kakovost, vendar smo to morali narediti zaradi omejene zmogljivosti strojne opreme. Danes pa obstajajo že zelo zmogljivi računalniki, s katerim bi celoten postopek lahko izvedli hitreje in na koncu dobili zelo verodostojen prikaz objekta, kar nam sama tehnologija zajema podatkov tudi omogoča.

VIRI

Abdagić, A. 2011. Razvoj igara u okruženju dopunjene stvarnosti. Završni rad. Zagreb, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva (samozaložba A. Abdagić): 38 f.

Antolković, A. 2014. 3D skeniranje arheoloških artefakata. Diplomski rad. Zagreb, Sveučilište u Zagrebu, Geodetski fakultet (samozaložba A. Antolković): 56 f..

Arithmetica. 2015. Pointfuse User Guide 1.2.2. http://pointfuse.com/wp-content/uploads/userguide/Pointfuse%20User%20Guide%201.2.2.pdf (Pridobljeno 10. 01. 2016.)

Bareta, Ž. 2013. Savremena industrija video igara. Master rad. Beograd, Univerzitet Singidunum, Departman za poslediplomske studije (samozaložba Ž. Bareta): 59 f.

Bernik, A. 2010a. Analiza i tehnike renderiranja. Tehnički glasnik 4, 1-2: 42-44.

Bernik, A. 2010b. Vrste i tehnike 3D modeliranja. Tehnički glasnik 4, 1-2: 45-47.

Cheng, R.K.C. 2005. Autodesk Mechanical Desktop 2005. Poglavlje 1: Računarsko modelovanje: str. 1-7. http://www.mikroknjiga.rs/Knjige/MDE5/01_MDE5.pdf (Pridobljeno 25. 11. 2015.)

Deisgn Workshop Sidney. 2014. AutoCAD 3D models. Slika 3D modelov. http://designworkshopsydney.com.au/wp-content/uploads/2014/02/autocad-3d-models.jpg (Pridobljeno 09.12.2015.)

FARO Technologies Inc. 2015a. FARO LS SDK 5.5 MANUAL.

FARO Technologies Inc. 2015b. SCENE 5.5 USER MANUAL.

Flaterco.com. Slika video igre: Doom. http://www.flaterco.com/kb/DOOM/ChocolateDoom_0.1.4.png (Pridobljeno 04. 12. 2015.)

Gamecareerguide. 2008. What is a Game Engine?. http://www.gamecareerguide.com/features/529/what_is_a_game.php (Pridobljeno 18. 11. 2015.)

GameSpot. 2015. Slika video igre: GTA V. http://static4.gamespot.com/uploads/original/1365/13658182/2843118-rsg_gtav_pc_screenshot_057.jpg (Pridobljeno 14. 12. 2015.)

Geocentar. 2013. Video Tutorial Kubit PointCloud Pro 8.0.

IGN.com. 2009. The best game engines of this generation. http://www.ign.com/articles/2009/07/15/the-10-best-game-engines-of-this-generation?page=1 (Pridobljeno 18. 11. 2015.)

Jovanović, M. 2006. Geometrijsko modeliranje 3D objekta. Predavanje. http://ttl.masfak.ni.ac.rs/CAD/Predavanje-3%20CAD%202007.pdf (Pridobljeno 10. 11. 2015.)

kubitTV. 2015. PointSense Heritage 16 5 Webinar: Point Clouds and Photogrammetry in AutoCAD. https://www.youtube.com/watch?v=Ajis0OeicuA (Pridobljeno 15. 01. 2016.)

Lasić, Z. 2008. Primjena laserskih uređaja. Skripta. Zagreb, Sveučilište u Zagrebu, Geodetski fakultet.

LGDB. 2015. Game categories. http://www.lgdb.org/game_categories (Pridobljeno 17. 11. 2015.)

Lušenc, S. 2014. Pretvorba 3D modelov iz programa Blender v binarni zapis igralnega pogona PRISM 3D. Magistrsko delo. Maribor, Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko (samozaložba S. Lušenc): 47 f.

McHenry, K., Bajcsy, P. 2008. An Overview of 3D Data Content, File Formats and Viewers. National Center for Supercomputing Applications. University of Illinois at Urbana-Champaign, Urbana, IL 2008. https://isda.ncsa.illinois.edu/drupal/publication/overview-3d-data-content-file-formats-and-viewers (pridobljeno 25. 11. 2015.)

MeshLAB Tutorial. 2009. http://www.cse.iitd.ac.in/~mcs112609/Meshlab%20Tutorial.pdf (pridobljeno 15. 09. 2015.)

Meshlabstuff. 2009. Meshing Point Clouds. http://meshlabstuff.blogspot.hr/2009/09/meshing-point-clouds.html (Pridobljeno 15. 09. 2015.)

Miler, M., Đapo, A., Kordić, B., Medved, I. 2007. Terestrički laserski skeneri. Ekscentar 10: 35-38.

Miller, K. 2009. Vertex Normals. http://www.flipcode.com/archives/Vertex_Normals.shtml (Pridobljeno 27. 11. 2015.)

Muzej Međimurja Čakovec. 2015. Sveta Jelena u Šenkovcu. http://mmc.hr/info/sveta-jelena-u-senkovcu/ (Pridobljeno 22. 11. 2015.)

Pandžić, I.S. 2004. Virtualna okruženja: računalna grafika u stvarnom vremenu i njene primjene. Zagreb, Element.

Pejić, M.M. 2013. Tačnost modeliranja objekata tehnologijom terestričkog laserskog skeniranja. Doktorska disertacija. Beograd, Univerzitet u Beogradu, Građevinski fakultet (samozaložba M.M. Pejić): 203 f.

Saxena, A., Laze, J., Kurata, T. 2009. History of The Video Game Industry. The Growing Gaming Industry. Pro Quest Library. http://www.pages.drexel.edu/~as3445/history.html (Pridobljeno 16. 11. 2015.)

Sourceforge.net. 2016. MeshLab. http://meshlab.sourceforge.net/ (Pirdobljeno 12. 01. 2016.)

Statista.com. 2015. Statistics and facts about the Video Game Industry. http://www.statista.com/topics/868/video-games/ (Pridobljeno 16. 11. 2015.)

Topić, I. 2005. Obrada podataka laserskog skaniranja RapidForm-om. Diplomski rad. Zagreb, Sveučilište u Zagrebu, Geodetski fakultet (samozaložba I. Topić): 63 f.

Unreal Engine 4 Documentation. 2015a. Actors and Geometry. https://docs.unrealengine.com/latest/INT/Engine/Actors/index.html (Pridobljeno 16.01.2016.)

Unreal Engine 4 Documentation. 2015b. Lightmap Resolution. https://docs.unrealengine.com/latest/INT/Engine/Rendering/LightingAndShadows/LightMobility/StaticLights/index.html (Pridobljeno 16. 01. 2016.)

Unreal Engine 4 Documentation. 2015c. Unreal Editor Manual. https://docs.unrealengine.com/latest/INT/Engine/Editor/index.html (Pridobljeno 20. 09. 2015.)

Unreal Engine. 2016. FREQUENTLY ASKED QUESTIONS (FAQ) https://www.unrealengine.com/vr-page (Pridobljeno 16. 01. 2016.)

VGBlogger. 2014. Slika video igre: Gas Guzzlers: Extreme. http://www.vgblogger.com/wp-content/uploads/2014/05/06/GasGuzzlersExtreme_FullMetalFrenzy_DLC_003.jpg (Pridobljeno 14. 12. 2015.)

Virtual Reality Site. 2015. What is Virtual Reality?. http://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html (Pridobljeno 15. 11. 2015.)

Vuoskoski, J. 1996. Exchange of Product Data between CAD systems and a Physics Simulation Program. Thesis. Tampere, Tampere University of Technology, Pori Unit. http://cadd.web.cern.ch/cadd/cad_geant_int/thesis/vaitos_main.html (Pridobljeno 09. 12. 2015.)

Webograd. 2015. Kapela svete Jelene u Šenkovcu. http://webograd.tportal.hr/josnov/fotogalerija/foto (Pridobljeno 22. 11. 2015.)

Wikipedia. 2015a. Avatar (informatika). https://hr.wikipedia.org/wiki/Avatar_(informatika) (Pridobljeno 17. 11. 2015.)

Wikipedia. 2015b. First person (video games). https://en.wikipedia.org/wiki/First_person_(video_games) (Pridobljeno 17. 11. 2015.)

Wikipedia. 2015c. Igralni pogon. https://sl.wikipedia.org/wiki/Igralni_pogon (Pridobljeno 18. 11. 2015.)

Wikipedia. 2015d. Pavlinski samostan Sveta Jelena. https://hr.wikipedia.org/wiki/Pavlinski_samostan_Sveta_Jelena (Pridobljeno 22. 11. 2015.)

Wolf, M.J.P. 2001. The Medium of the Video Game. Genre and the Video Game. http://www.robinlionheart.com/gamedev/genres.xhtml (Pridobljeno 17. 11. 2015.)

3.bp.blogspot.com. Slika video igre: Need for speed - Most wanted. http://3.bp.blogspot.com/-s-bF6I-bqu4/UH8qGrO8P0I/AAAAAAAACRg/bCpVk9kLgYM/s400/Need-for-speed-Most-wanted_Black-Eddition2.jpg (Pridobljeno 14. 11. 2015.)