

# DooVR - Interaktiv modellering i virtuell verklighet

## Projektrapport, TNM094

Team DooVR  
Isabelle Forsman  
Johan Nordin  
Jonathan Bosson  
Olle Grahn

20 juli 2016

# Sammanfattning

I denna rapport beskrivs hur en applikation som tillåter modellering i virtuell verklighet har utvecklats. De senaste åren har tekniken som används för att arbeta i virtuell verklighet blivit bättre och alltmer tillgänglig. Projektet syftar till att undersöka om modellering med sådan utrustning kan förbättra konstnärens upplevelse och förmåga, jämfört med de modelleringsapplikationer som använder tvådimensionella skärmar.

Arbetet med projektet har genomförts med den agila utvecklingsmetoden Scrum där gruppen haft nytta av de rutiner och trygghet den medfört. Utvecklingsarbetet har omfattat att sätta upp och kalibrera ett tracking-system samt att skapa en applikation som visar ett möjligt användningsområde för virtuell verklighet.

Resultatet av projektet är DooVR, en modelleringsapplikation som låter användaren modellera en yta med hjälp av tracking-enheten Intersense IS-900 samt Oculus rift som bildskärm.

# Innehåll

<b>Sammanfattning</b>	<b>i</b>
<b>Figurer</b>	<b>iv</b>
<b>Typografiska konventioner</b>	<b>v</b>
<b>1 Inledning</b>	<b>1</b>
1.1 Bakgrund . . . . .	2
1.1.1 Kravspecifikation . . . . .	2
1.2 Syfte . . . . .	3
1.3 Frågeställning . . . . .	3
1.4 Avgränsningar . . . . .	3
<b>2 Relaterat arbete</b>	<b>4</b>
2.1 Virtuell verklighet och tracking-system . . . . .	4
2.2 Kamerabaserad tracking . . . . .	4
2.3 Magnetfältsbaserad tracking . . . . .	5
2.4 3D modeller och deras datastruktur . . . . .	5
2.4.1 Mesh . . . . .	5
2.4.2 Volymrendering . . . . .	5
<b>3 Process</b>	<b>7</b>
3.1 Utvecklingsmetodik . . . . .	7
3.1.1 Roller . . . . .	7
3.2 Rutiner och verktyg . . . . .	8
3.2.1 Kodgranskning . . . . .	8
3.2.2 Dokumentation . . . . .	8
3.2.3 Versionshantering . . . . .	9
3.2.4 Testning . . . . .	9
3.3 Tidsplan . . . . .	9
<b>4 Systemöversikt och arkitektur</b>	<b>11</b>

4.1	Systemets olika delar . . . . .	11
4.1.1	Namespaces <b>oculus</b> och <b>twoD</b> . . . . .	11
4.1.2	Wand . . . . .	12
4.1.3	SceneObject . . . . .	12
4.2	Tredjepartsbibliotek . . . . .	12
<b>5</b>	<b>Hårdvara</b>	<b>13</b>
5.1	Oculus Rift . . . . .	13
5.2	InterSense . . . . .	14
5.3	Kalibrering . . . . .	15
5.4	VRPN . . . . .	16
<b>6</b>	<b>Mesh</b>	<b>17</b>
6.1	Datastruktur . . . . .	17
6.2	Retriangulering . . . . .	18
6.3	Modellering . . . . .	19
<b>7</b>	<b>Datorgrafik</b>	<b>21</b>
7.1	Användargränssnitt . . . . .	21
7.2	Texturer . . . . .	22
7.3	Ljusmodell . . . . .	23
7.4	Realtidsrendering . . . . .	24
<b>8</b>	<b>Resultat</b>	<b>25</b>
<b>9</b>	<b>Analys och diskussion</b>	<b>27</b>
9.1	Metod . . . . .	27
9.2	Resultat . . . . .	27
9.2.1	Mesh . . . . .	27
9.2.2	Kalibrering . . . . .	28
9.2.3	Hårdvara . . . . .	28
9.3	Arbetet i ett vidare sammanhang . . . . .	28
<b>10</b>	<b>Slutsatser</b>	<b>29</b>
10.1	Frågeställningar . . . . .	29
10.2	Framtida arbete . . . . .	30
<b>A</b>	<b>Arbetsfördelning och ansvarsområden</b>	<b>32</b>
<b>B</b>	<b>Ordlista</b>	<b>33</b>

# Figurer

1.1	Applikationen under användning . . . . .	2
3.1	Tidsplan för projektet, visar mål med sprintarna samt tidsfördelningen. . . . .	10
4.1	Klassdiagram över hur systemet är uppbyggt . . . . .	11
5.1	Här visas Oculus Rift systemets kamera och dess specifikationer . . . . .	13
5.2	Pincusion distortion visas till vänster och barrel distortion till höger . . . . .	14
5.3	Pitch, yaw, och roll runt de olika axlarna. . . . .	15
5.4	Oculus systemets koordinatsystem jämfört med IS-900 . . . . .	15
5.5	VRPN aktivitet under körning . . . . .	16
6.1	En mesh där bara edges renderas . . . . .	17
6.2	Översikt över den information som finns i en half edge datastruktur . . . . .	18
6.3	Den streckade linjen är för kort och edge collapse utförs, resultatet visas till höger. . . . .	18
6.4	Specialfallen som kan uppstå vid en edge collapse . . . . .	19
6.5	Den streckade linjen är för lång och edge split utförs, resultatet visas till höger. . . . .	19
6.6	Den heldragna pilen är riktningen verexpunkten flyttas i . . . . .	20
7.1	Höger och vänster öga ser olika bilder, förskjutningen mellan objekten i bilderna kallas binocular disparity. . . . .	21
7.2	Funktionspanel och markeringar av hexagon boxar. . . . .	22
7.3	Uppdelad texturbild för varje sida av en box . . . . .	23
7.4	Blinn-Phong ljusmodell . . . . .	24
8.1	En snigel som modellerats i applikationen . . . . .	26

# Typografiska konventioner

Typsnitt	Betydelse
<i>ABCacb123</i>	Engelskt ord (först förekommande kapitelvis), ordlista över dessa ord finns i bilaga <b>B</b> .
ABCabc123	Teknisk- eller dataterminologi
<b>ABCabc123</b>	Variabel- eller klassnamn.

# Kapitel 1

## Inledning

Människan har skapat saker sedan långt tillbaka i tiden. Detta skapande har fortsatt och utökats in i vår digitala era. Många sätt att skapa har förbättrats med hjälp av datorer och ny teknik. Till exempel kan målare och musiker göra saker som inte tidigare var möjliga med hjälp av datorer. Något som inte berikats på samma sätt med hjälp av datorer är skulptering och modellering. Modellering på datorer är just nu låst till tvådimensionella skärmar fastän det är volymer och kroppar som modelleras. DooVR är en modelleringssapplikation som löser detta problem.

I applikationen DooVR modellerar användaren i 3D med hjälp av en *head mounted display (HMD)* och handhållna 6 degrees of freedom (6DoF)-enheter på det vis som syns i figur 1.1. Applikationen ger konstnärer möjlighet att modellera på samma sätt som de gör i verkligheten men utan verklighetens begränsningar. Detta kräver att miljön som användaren arbetar i känns naturlig och att all teknik som används är synkroniserad och fungerar tillsammans. Det krävs också ett objekt som kan förändras på oändligt många sätt och funktioner som ger användaren både frihet och precision i sitt skapande. För att implementera all denna funktionalitet krävs det till sist en väl fungerande utvecklingsmetodik och strukturerad systemarkitektur.

Rapporten tar upp vad allt det ovannämnda innebär, hur utvecklingen gått till och vad resultatet blev. Avslutningsvis diskutas också de val och ändringar som gjorts under projektets gång och hurvida resultatet var lyckat eller inte. Rapporten är skriven med visst tekniskt kunnande i åtanke men ger grundläggande förklaringar av den teori som applikationen kräver.



Figur 1.1: Applikationen under användning

## 1.1 Bakgrund

Idén till projektet har två rötter. Den första är att det just nu lanseras många produkter inom virtuell verklighet (VR) som låter användare se och röra sig i 3D. Dessa produkter är billiga nog att konsumenter såväl som företag har möjlighet att köpa dem. Den andra är att projektgruppens medlemmar har arbetat en hel del med redan existerande 3D-modelleringsprogram. Det många av dessa program har gemensamt är att användaren modellerar i 3D på en 2D-skärm. Detta leder till att användaren ständigt måste vrida och flytta på en kamera vilket gör att det ofta blir svårt att uppfatta vad som händer med 3D-formerna.

Idén är alltså använda denna relativt billiga hårdvara för att låta användare skapa 3D-modeller i 3D istället för i 2D.

### 1.1.1 Kravspecifikation

Projektgruppen presenterade förslag på demoapplikation till kunden som sedan levererade följande kravspesifikation:

- Ett trackingsystem med en demoapplikation som visar systemets styrkor.
- Två konstnärer ska kunna arbeta samtidigt.
- Stöd för Virtual Reality Peripheral Network (VRPN) måste finnas.

## 1.2 Syfte

Projektets syfte är att implementera ett *tracking-system* och utveckla en applikation som visar på möjligheterna med virtuell verklighet. Applikationen som utvecklas ska tillåta en användare att modellera, likt lerskulptering, med hjälp av en handhållen 6DoF-enhet samt en Oculus Rift<sup>1</sup>.

Det ursprungliga syftet var att applikationen skulle stödja två typer av användare; konstnärer och åskådare. Tanken var att åskådare skulle kunna se vad som gjordes i scenen och sätta ut markeringar på meshen men de skulle inte kunna manipulera den. Detta för att möjliggöra interaktion mellan konstnären och utomstående. En åskådare skulle antingen kunna använda en Oclulus Rift eller en 2D-skärm samt använda en wand, Kinect 2<sup>2</sup> eller datormus och tangentbord som inmatningsenheter. Applikationen skulle även tillåta att flera konsnärer modellerar i samma scen på samma eller olika objekt.

## 1.3 Frågeställning

- Kan en Oculus Rift och en 6DoF enhet hjälpa konstnärer att modellera i 3D på ett sätt som gör att modelleringen blir enkel att utföra och upplevs som modellering i verkligheten?
- Vilka är fördelarna och nackdelarna med volym- och yt-baserade datastrukturer och vilken av dessa två typer passar bäst till projektet?
- Hjälps konstnären av kontakt med åskådare då denne modellerar och är förmågan att markera punkter på ett objekt tillräcklig för att åskådare ska kunna delge sina idéer och tankar med konstnären?
- Vilka alternativ finns för att ansluta användarna över nätverk och vilket alternativ ger bäst effekt med avseende på latens?
- Går det att skapa en kalibreringsfunktion som länkar ihop koordinatsystemet för HMD med 6DoF-enhetens koordinatsystem, oavsett hur användarens tracking utrustning är uppsatt?

## 1.4 Avgränsningar

Projektet har utvecklats mot operativsystemet Windows.

Hårdvaran som integreras i systemet är Oculus Rift, Intersense IS-900 genom VRPN samt den wand som utvecklas i projektet Active3D wand på Linköpings universitet. Den sistnämnda kommer inte beskrivas eller diskuteras närmare i rapporten på grund av att gruppen har skrivit under ett sekretessavtal.

---

<sup>1</sup><https://www.oculus.com/rift/>

<sup>2</sup><https://www.microsoft.com/en-us/kinectforwindows/develop/>

# Kapitel 2

## Relaterat arbete

### 2.1 Virtuell verklighet och tracking-system

Det har under flera årtionden pågått forskning om VR-produkter och de har länge funnits tillgängliga. Det som händer just nu är att tekniken har blivit så pass noggrann att dessa redskap kan användas för att göra saker i en virtuell verklighet på samma sätt och med samma känsla som om det gjordes i den fysiska verkligheten. Dessutom är dessa produkter nu tillräckligt billiga att privatpersoner har råd att köpa dem.

Det finns många olika sätt att tracka fysiska föremål och representera deras positioner och orientationer med digitala värden [11]. Just nu är två vanligt använda tekniker för positions tracking att använda en kamera eller ett magnetfält för att avläsa var i rummet enheterna befinner sig. Men det finns andra metoder som också fungerar och nya metoder utvecklas ständigt till exempel går det också använda ultraljud för att tracka enheter.

### 2.2 Kamerabaserad tracking

Kamerabaserad tracking går ut på att identifiera föremål i kamerans vy och bestämma deras position [2]. Denna identifiering görs ofta med avancerad mjukvara som analyserar kameras pixeldata. Identifieringen kan underlättas genom att använda föremål som står ut i kamerans pixeldata. Till exempel är en cyanfärgad kub som lyser starkt enklare att identifiera än ett ansikte. Det finns dock mjukvara<sup>1</sup> som är mycket bra på att identifiera alltförifrån ansikten till händer.

Men föremålen ska inte bara identifieras, deras positioner och orientationer måste också bestämmas. Föremålets position i två av de tre dimensioner som utgör dess position är enkel att bestämma. Om föremålet rör sig uppåt i kameravyn har det också rört sig uppåt i verkligheten. Samma sak gäller ifall föremålet rör sig sidledes i kameravyn. Det är dock inte lika enkelt bestämma positionen eftersom föremålet flyttas mot eller ifrån kameran. Detta går att göra på olika sätt. En metod är att använda kameras perspektiv och analysera föremålets storlek i kameravyn. En boll som är nära kameran kommer ha en stor radie i kamerans vy, flyttas bollen längre ifrån kameran kommer den ha en mindre radie i kamerans vy. Därefter kopplas denna storleksförändring till den faktiska avståndsförändringen och därmed är föremålets avstånd från kameran bestämt. En annan metod är att använda en djupkamera i samband med en vanlig kamera [5]. Djupkamerans pixel-data består bara av information om hur långt ifrån det som fanns i just den pixeln befann sig ifrån kameran.

Att bestämma ett föremåls orientation med hjälp av en kamera går ut på att beräkna geometrier,

---

<sup>1</sup><https://www.microsoft.com/en-us/kinectforwindows/develop/>

avståndsförhållanden mellan punkter och hur dessa förändras i tiden. Är det till exempel en kub som ska positions bestämmas och bara fyra hörn syns i kameravyn vid en viss tidpunkt är kuben orienterad på så sätt att en sida är riktad rakt mot kameran. Om det i nästa tidpunkt finns två nya hörn till höger om de hörn som befann sig i kameravyn vid föregående tidpunkt har en rotation åt vänster skett. Även här är färgkodning och djupkameror goda hjälpmedel.

## 2.3 Magnetfältsbaserad tracking

Det finns många intressanta projekt där ett magnetfält används för att tracka enheter. Ett exempel är Sixense<sup>2</sup> som med hjälp av elektromagnetiska fält bestämmer positionen och orienteringen för enheter. Fördelarna med en magnetbaserad tracking är t.ex. att det blir möjligt att bestämma positionen för flera enheter till samma bas-station och det är möjligt att hålla reda på positionen även om enheterna skymts från bas-stationen av en människokropp eller andra föremål [11].

## 2.4 3D modeller och deras datastruktur

Det går att representera volymer och föremål med hjälp av olika datastrukturer som alla har fördelar och nackdelar. I applikationen som utvecklas i detta projekt ska volymen även kunna förändras och formas i realtid vilket ställer ökade krav på datastrukturen. Ett grundläggande val som måste göras är ifall volymen ska representeras i form av en yta eller en faktiskt volym.

### 2.4.1 Mesh

En yta eller en så kallad mesh består av ett slutet nät av trianglar där trianglarnas tre punkter är ordnade motsols sett från meshens utsida. Detta nät består av en lista med punkter och en lista med index som länkar samman punkterna tre och tre så att de skapar trianglar. Ska det gå att göra ändringar på meshens struktur utan att behöva gå igenom båda listorna flera gånger behövs även ytterligare information. Vilken ytterligare information som behövs skiljer sig från fall till fall men det rör sig ofta om att punkterna har information om vilka andra punkter, kanter eller trianglar som finns i dess närhet. Exempel på använda datastrukturer är Face-based, HalfEdge och Winged-edge [12].

Fördelen med denna datastruktur är att den representerar volymer med en relativt liten datamängd. Det är bara ytan som behöver sparas, ingen information om hur det ser ut innanför ytan behöver sparas.

Nackdelarna är att det krävs många beräkningar för att bestämma ifall en punkt befinner sig i volymen eller utanför. Det krävs även beräkningar för att bestämma trianglarnas fram och baksida. Inte heller finns det ett enkelt sätt att bestämma ifall en del av ytan penetrerar en annan del av ytan eftersom sökning inte sker i rummet utan bara längs ytan.

### 2.4.2 Volymrendering

Om en faktisk volym ska representeras så representeras denna med hjälp av ett skalärfält. Ett skalärfält består av en mängd punkter bestående av tre koordinater som representerar dess position och ett fjärde värde som representerar punktens densitet. Ska volymen renderas i realtid krävs även här en lista

---

<sup>2</sup><http://sixense.com/wireless>

med punkter och trianglar. En metod som används för att skapa en yta av volymdata är så kallade Marching cubes [10].

Den stora fördelen med denna datastruktur är att det är enkelt att testa ifall en punkt befinner sig i eller utanför volymen vilket förenklar många funktioner och beräkningar som krävs för volymmodellering. Den stora nackdelen är att den innehåller mycket mer information än den meshbaserade datatypen. Ett skalärfält med hög upplösning får inte plats i en modern persondators internminne vilket gör att för denna datatyp krävs komprimerings och sorteringsalgoritmer om den ska användas i realtid.

# Kapitel 3

## Process

### 3.1 Utvecklingsmetodik

I projektet har den agila utvecklingsmetoden *Scrum* [7] använts. I utvecklingsmetoden Scrum delas arbetet upp i *sprintar*. En sprint definieras som en hel iteration i utvecklingsprocessen där hela gruppen under 2-4 veckor arbetar mot ett gemensamt och konkret mål. Resultatet efter varje sprint ska vara en fungerande version av systemet där ny funktionalitet har tillförts. Varje sprint inleddes med ett planeringsmöte där gruppen gemensamt tittade igenom *product backlog*. Därefter diskuterades uppgifter och mål för sprinten och *stories* togs fram. En storie är ett krav som finns på produkten och innehåller information om för vem, vad, när, var samt varför och ska kunna genomföras under en sprint. Dessa stories bröts sedan ner till *tasks* som är uppgifter som ska utföras för att klara en storie. En task ska kunna utföras på en dag. De framtagna stories och tasks fylldes sedan in i planeringsverktyget Trello<sup>1</sup>. Ansvarsfördelning utav tasks och stories har inte behövts eftersom gruppen varit få medlemmar och arbetat nära varandra.

Gruppen inledde varje projektdag med ett kort möte där alla berättade vad de arbetade med under föregående projektdag samt vad som planerades att göra under dagen. Vid varje avslutad sprint har ett *sprint review* möte ägt rum tillsammans med kund där arbetet under sprinten presenterats och kommande arbete diskuterats. Kunden har uppdaterats om hur projektet forskrider och haft möjlighet att ändra sina krav, framföra prioriteringar och komma med feedback på om utfört arbete uppfyller kundens krav. Ett retrospektivmöte utfördes sedan sist i sprinten där gruppen diskuterat hur arbetet gått, vad som varit bra och vad som behöver ändras i kommande sprint.

#### 3.1.1 Roller

De roller gruppen tillämpat är *product owner*, *scrum master*, dokumentansvarig, sekreterare, test och kravansvarig samt kodansvarig.

##### Product owner

Ansvarig för att hålla regelbunden kontakt med kunden och har sett till att *product backlog* innehåller projektets alla krav. Dessutom har denne ansvarat för att påminna utvecklarna om datum för leverabler.

---

<sup>1</sup><https://trello.com/>

## Scrum master

Scrum master har varit ansvarig för att scrum planning, daily scrum och retrospect möten hållts. Dessutom ansvarade scrum master för att ha en överblick av sprintens mål, tasken och task prioriteringar och påminna utvecklarna om dessa så att utvecklingen skett baserat på prioriteten hos tasken. Scrum master har också hållt nästa sprint i åtanke för att ha koll på hurvida sprintarna följt tidsplaneringen.

## Dokumentansvarig

Som dokumentansvarig har personen ansvarat för att idéer och skisser i projektet sparats och dokumenterats. Personen har också ansvarat för att rapportskrivningen inleds i tid.

## Sekreterare

Sekreteraren har ansvarat för att möten blir dokumenterade.

## Test och kravansvarig

Test och kravansvarig har ansvarat för att tester utförts och att beskrivningar av alla buggar förts in i Githubs<sup>2</sup> verktyg för issues. Denne har även ansvarat för att produktens krav varit uppfyllda och meddelat utvecklarna om eventuella problem. Dessutom ansvarade denne för att hitta lämpliga testpersoner för användartesterna och sköta kommunikationen med dessa.

## Kodansvarig

Rollen som kodansvarig innebar att den ansvarige hade översikt på kodstruktur och programvaror i projektet. Personen såg till att gruppens git-workflow utförts på rätt sätt och att regelbundna kodgranskningssmöten hållts. Dessutom har denne försäkrat att dokumentation skett på rätt sätt och att koden följt styleguiden som använts.

## 3.2 Rutiner och verktyg

### 3.2.1 Kodgranskning

Projektgruppen har under hela projektet tillämpat parprogrammering.

Kodgranskningssmöten har utförts i mitten samt i slutet av varje sprint, för att upptäcka brister i koden samt för att alla i gruppen ska få en förståelse för arbetet som gjorts. På kodgranskningssmötena har den aktuella koden visats på en stor skärm så att alla i gruppen kan läsa koden. Sedan har de som skrivit koden fått förklara vad den gör och tankarna med det valda implementationssättet.

### 3.2.2 Dokumentation

Eftersom projektet har haft flera intressenter har det varit viktigt med dokumentation för att alla involverade ska få en klar uppfattning om hur projektet strukturerats och vilka användningsfall som finns. UML modeller har därför använts under projektetsgång.

<sup>2</sup>[www.github.com](http://www.github.com)

Mötesprotokoll, övriga dokument och skisser har sparats på Google Drive<sup>3</sup>.

För dokumentation av kod har verktyget doxygen<sup>4</sup> använts. Doxygen kräver att koden dokumenteras och kommenteras efter en mall. Uppfyller koden riktlinjerna kan dokumentation som ger en bra översikt och läsbarhet om systemet genereras.

### 3.2.3 Versionshantering

Versionshanteringsprogrammet Git<sup>5</sup> har använts under projektets gång. Projektgruppen har arbetat enligt Gitflow-workflow [1] vilket innebär att all utveckling sker mot en utvecklings-branch, från utvecklings-branchen skapas en branch där utvecklingen av funktioner sker. När arbetet med funktionen är klar utförs en merge med utvecklings-branchen. Efter att koden i utvecklings-branchen granskats under ett kodgranskningssmöte och rättats därefter utförs en merge mellan master och utvecklings-branchen. På detta sätt finns alltid en fungerande version av applikationen i master branchen.

### 3.2.4 Testning

Eftersom all utveckling har skett i ett labb där den använda utrustningen alltid funnits tillgänglig har testning av applikationen kunnat ske kontinuerligt. Nästan all funktionalitet som implementerats har en visuell komponent. Det har varit relativt enkelt att testa applikationen och se ifall resultatet är som förväntat. En aspekt av applikationen som var svårare att testa visuellt var meshen och dess datastruktur. Här var det ibland svårt att se vad som hände vid triangulation och modellering. Därför skrevs enkla funktioner som renderade information och länkar som bara existerade abstrakt i datastrukturen. Tillsammans med dessa funktioner och Visual studios debug-verktyg kunde även denna del testas.

## 3.3 Tidsplan

I början av projektet tog projektgruppen fram en tidsplan över projektiden där sprintar och deras mål planerades in. Denna tidsplan uppdaterades sedan kontinuerligt allteftersom hur projektet låg till. Den senast uppdaterade tidsplanen ses i figur 3.1.

Efter Sprint 2 gjordes tidsplanen om eftersom gruppen insett att mesh hanteringen skulle ta längre tid att göra än beräknat samt att ljusmodellen som skapats under Sprint 1 inte fungerade som den skulle vid användning av HMD. Därför valde gruppen att prioritera funktionerna för konstnären och beslutade att Kinect 2 inte längre skulle integreras för observatörer.

Efter Sprint 3 hade gruppen inte slutfört målen som fanns för sprinten. Därför bestämdes att stryka nätverksdelen i projektet och föra över de resterande uppgifterna från Sprint 3 till Sprint 4. Eftersom nätverksdelen prioriterades bort föll även observatören ur projektet.

---

<sup>3</sup><https://www.google.com/drive/>

<sup>4</sup><http://www.stack.nl/~dimitri/doxygen/>

<sup>5</sup><http://git-scm.com/>

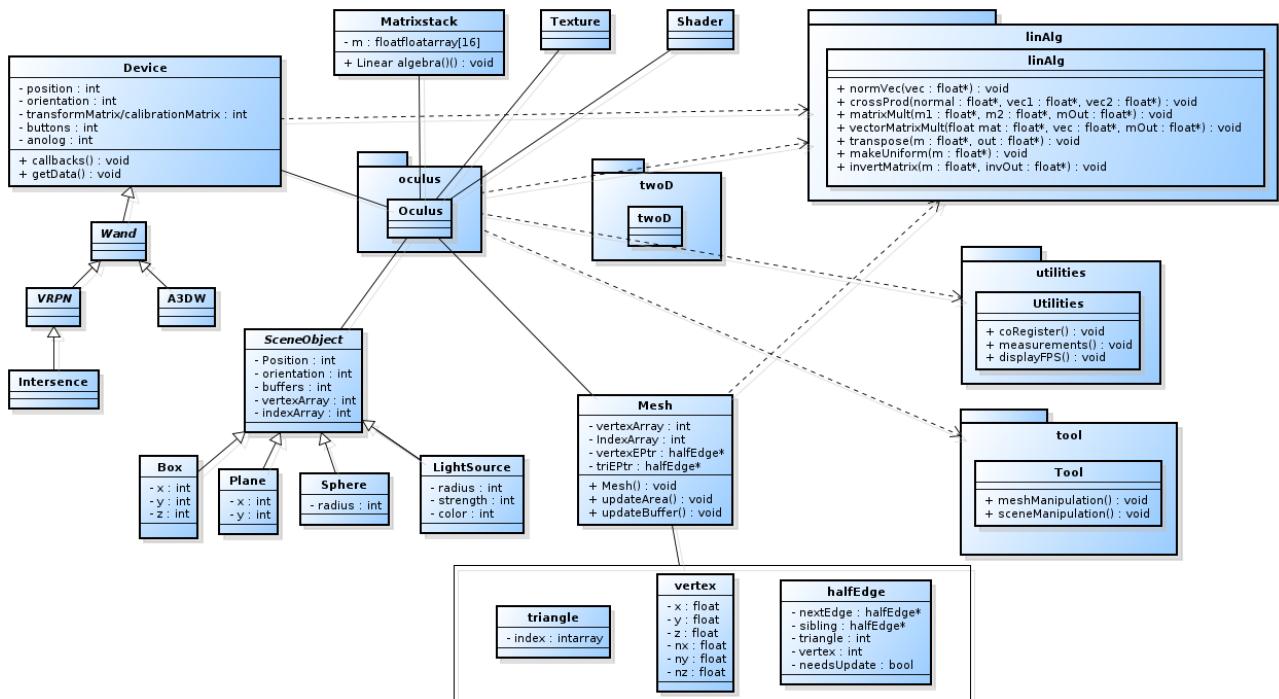


Figur 3.1: Tidsplan för projektet, visar mål med sprintarna samt tidsfördelningen.

# Kapitel 4

## Systemöversikt och arkitektur

DooVR är skapat i C++ och i detta kapitel beskrivs hur systemet är uppbyggt, vilka designval som gjorts samt vilka tredjepartsbibliotek som används.



Figur 4.1: Klassdiagram över hur systemet är uppbyggt

## 4.1 Systemets olika delar

### 4.1.1 Namespaces oculus och twoD

I Namespace **oculus** renderas scenen alltid till en Oculus Rift. **oculus** använder sig av funktioner och attribut ifrån alla delar av applikationen. I namespace **twoD** renderas scenen till en tvådimensionell skärm. Detta namespace används då användaren inte har tillgång till en HMD.

### 4.1.2 Wand

Klassen **Wand** implementerades eftersom applikationen är tänkt att fungera med en mängd olika tracking-enheter. Alla wand-enheter har en position och orientation men olika enheter har olika funktioner och egenskaper så implementeras trackingsystemen som parallella underklasser till **Wand**. Applikationen är också tänkt att fungera med VRPN som beskrivs i avsnitt 5.4. Detta bibliotek hanterar tracking-enheternas data på ett speciellt sätt så därför finns en subklass till **Wand** som tar hand om enheter som ansluter till applikationen genom VRPN.

### 4.1.3 SceneObject

SceneObject klassen är en superklass till alla statiska objekt som används i scenen. Alla objekt som har en statisk vertexstruktur och ska läggas ut någonstans i scenen är ett SceneObject. Exempel på sceneobjects som kan finnas i scenen är golv och ljuskällor.

## 4.2 Tredjepartsbibliotek

De API : er som används i projektet är OpenGL, libOVR, VRPN, GLEW, GLFW samt wand3D som är bibliotek för Active3D wand. Anledningen till att biblioteket GLM inte används som planerats i projektets början är för att datatyperna i GLM orsakade onödig kopiering när datan skickades till OpenGL. Med anledning av detta beslutade projektgruppen att inte längre använda GLM och skriva egna matematiska funktioner.

# Kapitel 5

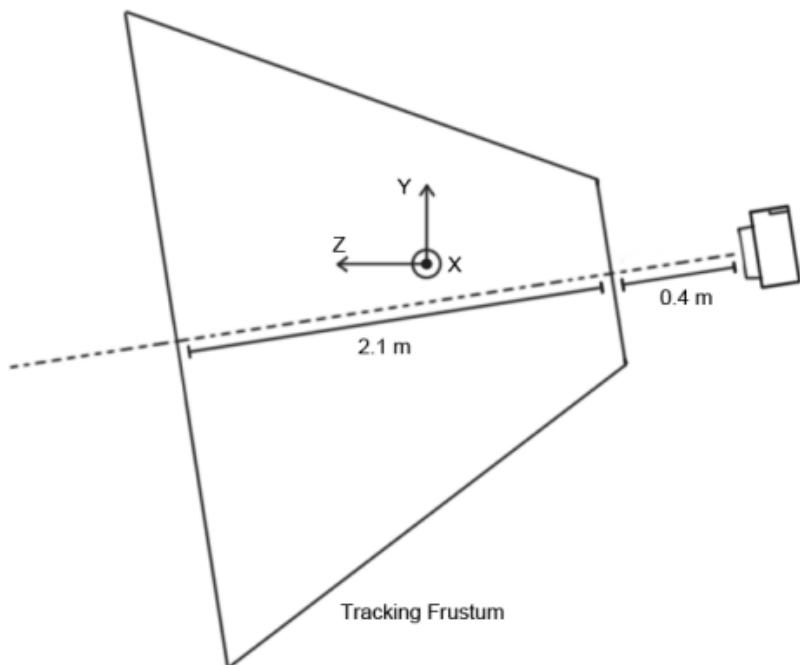
## Hårdvara

Hårdvara som används i projektet är Oculus Rift DK2, InterSense IS-900 och en prototypenhet av Active3D wand från InnovationskontorEtt.

### 5.1 Oculus Rift

Ett Oculus Rift system består av en kamera och ett *headset*, som består av en skärm, linser och sensorer. Linserna används för att sträcka ut det som visas på skärmen och simulera människans verkliga synfält. Sensorerna gör att kameran kan bestämma huvudenhetens position.

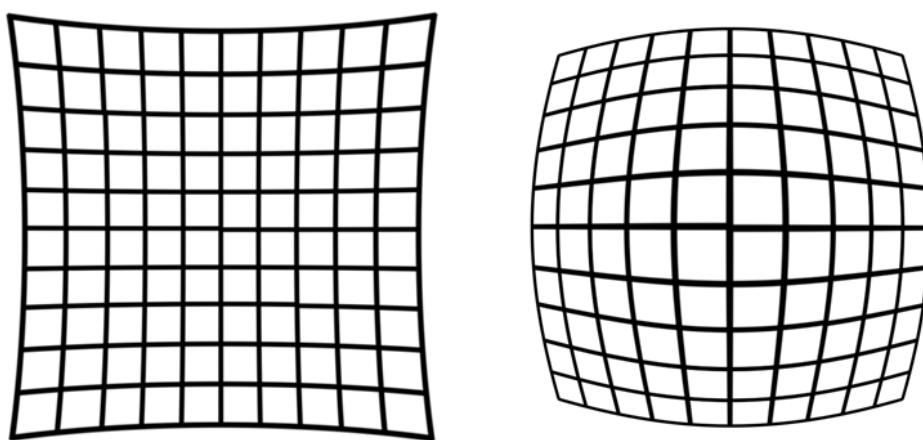
Kamerans specifikationer kan ses i figur 5.1. Även om kameran är lutad precis som i figuren kommer Oculus systemets position definieras i det koordinatsystem som visas i bilden där z-axeln alltid är riktad ifrån kameran och längs med golvet. Den kommer också bara kunna hitta kameran när Oculus finns inom det markerade frustumet. Skulle användaren vända sig mer än 90 grader ifrån kameran eller på något sätt dölja HMD:n ifrån kamerans vy kommer den inte heller kunna bestämma positionen.



Figur 5.1: Här visas Oculus Rift systemets kamera och dess specifikationer

Oculus systemets orientation bestäms inte med hjälp av kameran utan av inbyggda mätare som sparar rotationsförändringarna i det koordinatsystem som visas i bild 5.1. Orientationen kan alltså alltid spåras oavsett hur användaren rör sig. Positionen och orienteringen används sedan för att rendera ut scenen på skärmen som om enheten rörde sig i en verlig scen.

Eftersom linserna är placerade nära skärmen och är mycket kraftiga går det inte rendera på samma sätt till Oculus Rift skärmen som till en vanlig 2D skärm. Kraftiga linser ger upphov till något som kallas *pincusion distortion*. Detta hanteras genom att först rendera scenen till en textur som sedan mappas till skärmen med *barrel distortion*, se figur 5.2. Linserna ger också upphov till ett fenomen som kallas *chromatic aberrations* vilket innebär att färger i skärmens utkanter blir förvrängda. Även detta fenomen måste korrigeras i texturen innan den renderas ut. För projektet har valts att använda libOVR för att sköta dessa korrektioner. Detta medföra att scenen först renderas till en textur istället för ett fönster. Texturen skickas till biblioteket som utför korrigeringarna och sedan renderar resultatet.



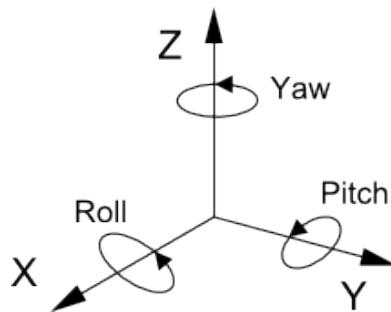
Figur 5.2: Pincusion distortion visas till vänster och barrel distortion till höger

## 5.2 InterSense

I DooVR är det viktigt att kunna beskriva exakt hur ett föremål (i detta fall handen) rör sig i det verkliga livet för att representera den i VR-världen. För att göra detta används en 6DoF tracking-enhet. I arbetet med det här projektet har IS-900<sup>1</sup> från Intersense används som 6DoF tracking-enhet [4].

En 6DoF tracking-enhet innebär att enheten har sex frihetsgrader vid förflyttning av en stelkropp. Det kan också beskrivas som att föremålet har sex olika sätt den kan röra sig på. För trackingenheten kan de sex frihetsgraderna delas upp i translations- och rotationrörelse. Translationen för en trackingenhet blir i tre frihetsgrader: framåt/bakåt, upp/ned, vänster/höger. De tre rotationerna beskrivs i figur 5.3 med de engelska termerna: pitch, yaw, och roll.

<sup>1</sup><http://www.intersense.com/pages/20/14>

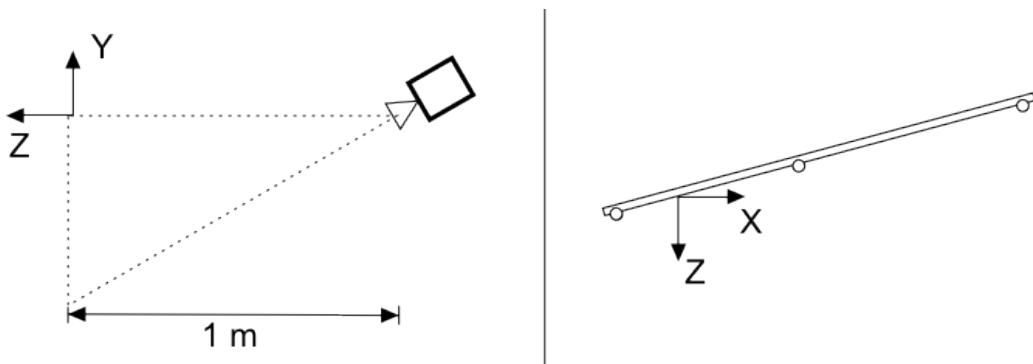


Figur 5.3: Pitch, yaw, och roll runt de olika axlarna.

På IS-900 finns det fem stycken knappar och en analog styrspak. Intersense IS-900 trackar ett område under ställningen som finns monterad i taket. Systemet använder sig av en hybridteknik av tröghetsmoment och ultraljud för att tracka enheten.

## 5.3 Kalibrering

Eftersom DooVR använder sig av två olika system, Oculus Rift DK2 och IS-900, som har olika definitioner av sina koordinatsystem behövs en kalibrering mellan dessa utföras. Målet med kalibreringen är att hitta transformationen som beskriver hur punkter i InterSense koordinatsystem kan översättas till Oculus koordinatsystem. Co-registration är en teknik som används för att synkronisera två koordinatsystem. Origo för Oculus och IS-900 finns beskrivna i figur 5.4. Båda dessa är höger system.



Figur 5.4: Oculus systemets koordinatsystem jämfört med IS-900

För att bestämma en entydig lösning till ekvationssystemet krävs att exakt fyra par punkter väljs. Om antalet punkter är mindre än fyra kan det bli mer än en lösning. Om det blir fler än fyra punkter får ingen lösning alls för ekvationssystemet.

Fyra punkter väljs i ett av koordinatsystemen med punkterna utsprida i alla koordinat-led. Resultatet blir en interpolation innanför de valda punkterna och en extrapolering utanför. Därför är det bra om punkterna ligger en liten bit ifrån varandra för att minska felmarginalen, ju större avstånd mellan punkterna desto mindre blir felmarginalen. Användarens uppgift är att välja punkter i det andra koordinatsystemet som motsvarar samma plats som första koordinatsystemets punkter.

Två matriser, A och B, skapas av de koordinater som valdes till punkterna i de två koordinatsystemen. En transform  $x$  kan nu beskriva hur punkterna i A kan konverteras till punkterna i B genom  $A \cdot x = B$ , det vill säga  $x = B \cdot A^{-1}$ .

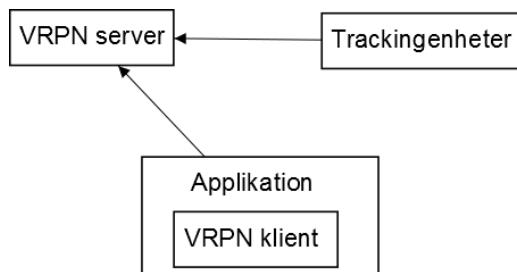
Den virtuella verkligheten är uppbyggd utefter Oculus där koordinatsystemet med HMD som centrum. Oculus Rift är begränsad av längden på sladden och IS-900 trackar enbart inom ett litet område så

är avståndet mellan de valda punkterna begränsat. En annan begränsning med co-registration mellan Oculus och IS-900 är att användaren inte kan se sin hand under kalibreringen. Detta innebär att metoden förlitar sig till användarens proprioception, det vill säga det som känns rätt för användaren.

## 5.4 VRPN

VRPN är ett *Open-source* projekt vars syfte är att hantera flera VR-enheter. VRPN består av ett antal klasser och servrar som används för att skapa ett nätverksgränssnitt mellan applikationen och tracking-enheter. VRPN har även stöd för många av de etablerade tracking-systemen och det är möjligt att integrera nya system i VRPN [9]. Nätverket som skapas med VRPN består av en klient och server struktur, se figur 5.5. Servern kan antingen byggas med CMake eller skrivas själv. Om en server ska byggas med CMake måste enheterna vara inlagda i VRPN projektet. Därefter måste sökvägar till enheterna som ska anslutas till servern anges, i detta fall InterSense.

Servern körs som en separat process utanför applikationen. Klient delen i VRPN integreras i applikationen som en egen klass. I klassen för klienten definieras vilken server och tracking-enhet som klienten ska ansluta emot. Den kommer även hantera den data som fås tillbaka från servern.



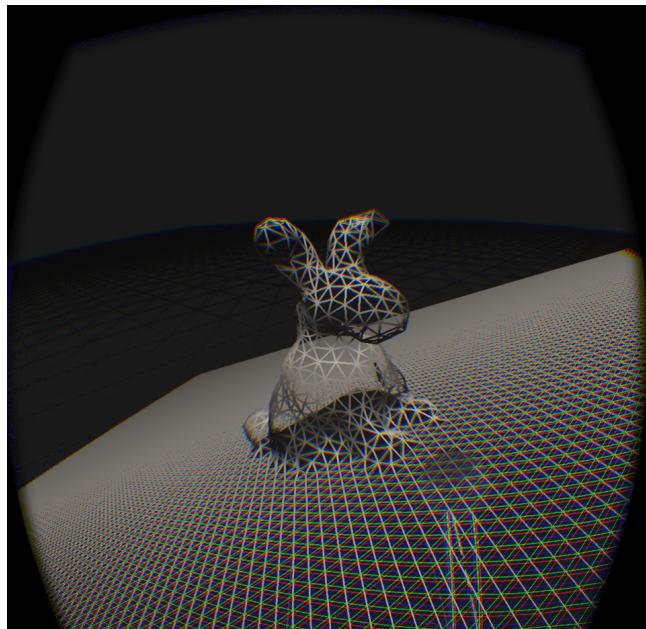
Figur 5.5: VRPN aktivitet under körning

En anledning till att använda sig av VRPN är att det ger en struktur som gör alla enheter är av samma basklass. Till exempel kommer olika typer av tracking-enheter alltid se ut som de är av typen `vrpn_Tracker`. Det gör att det inte spelar någon roll vilken typ av tracker som används då alla enheter skickar tillbaka data på samma sätt. Det gör att är möjlig att hantera flera olika tracking-enheter utan att systemet behöver göras om.

# Kapitel 6

## Mesh

Som nämntes i avsnitt 2.4.1 består en mesh av punkter, kanter och trianglar, se figur 6.1. Hur dessa representeras beror på vad som ska göras med meshen. Generellt gäller att ju mer information som sparas desto snabbare går beräkningarna. Det gäller alltså identifiera vilka beräkningar som krävs och vilken information dessa beräkningar behöver tillgång till. Därefter måste val göras så att beräkningarna inte ska ta för lång tid och datastrukturen inte tar för stor plats.

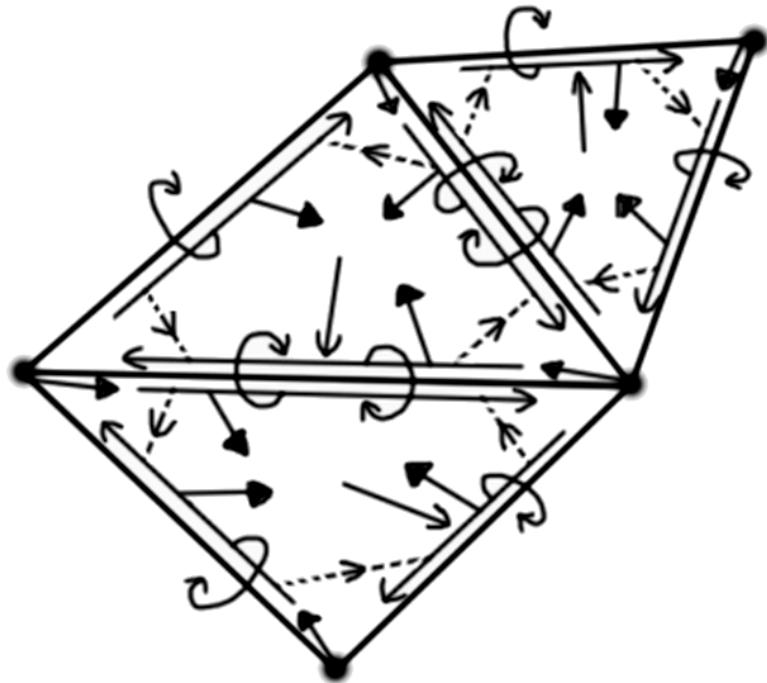


Figur 6.1: En mesh där bara edges renderas

### 6.1 Datastruktur

I detta projekt används en struktur som kallas `half edge`, se figur 6.2. En `half edge` datastruktur är baserad på information om kanter, vertex punkter samt trianglar. Den är uppbygd så att varje kant representeras utav två stycken `halfEdge`, alltså en för varje triangel den är del av. Varje `halfEdge` har i sin tur en pekare på nästkommande `halfEdge` i moturs riktning i triangeln, en pekare på en av `vertex` punkterna som är del av kanten i medurs riktning, en pekare till den `halfEdge` som är på andra sida av kanten som representeras, samt en pekare till den triangel som den är en del av. `Vertex` punkterna har även en pekare till en `halfEdge`, den `halfEdge` som `vertex` punkten pekar på får i sin

tur inte peka på tillbaka på *vertex* punkten. Slutligen har varje triangel en pekare till en **halfEdge** som den är uppbyggd av.



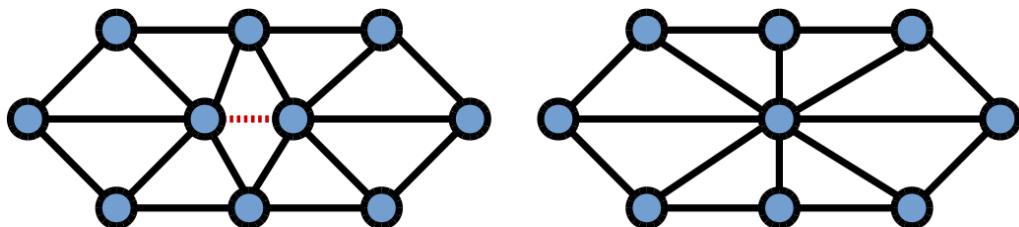
Figur 6.2: Översikt över den information som finns i en `halfEdge` datastruktur

Den använda datastrukturen gör det möjligt att snabbt beräkna avståndet mellan *vertex* punkter som flyttats samt att retriangulera vid behov.

## 6.2 Retriangulering

För att modellering av meshen ska fungera som önskas krävs det att *vertex* punkter läggs till och tas bort i realtid. Dessutom bör meshens trianglar hållas så liksidiga som möjligt.

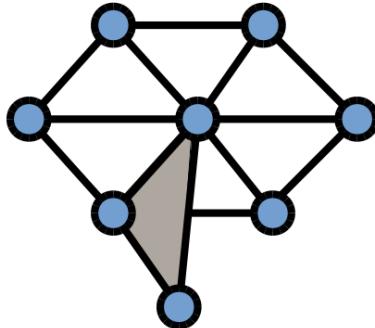
Tekniken som används för att ta bort en *vertex* punkt om de befinner sig för nära varandra kallas *edge collapse* [10]. Vilket innebär att när två *vertex* punkter befinner sig för nära varandra flyttas den ena *vertex* punkten halvägs mot veretxpunkten som ligger för nära. *Vertex* punkten som flyttats ärver den andras *vertex*- och triangelgrannar. Trianglarna mellan dessa två *vertex* punkter och *vertex* punkten som inte flyttades tas bort se figur 6.3.



Figur 6.3: Den streckade linjen är för kort och *edge collapse* utförs, resultatet visas till höger.

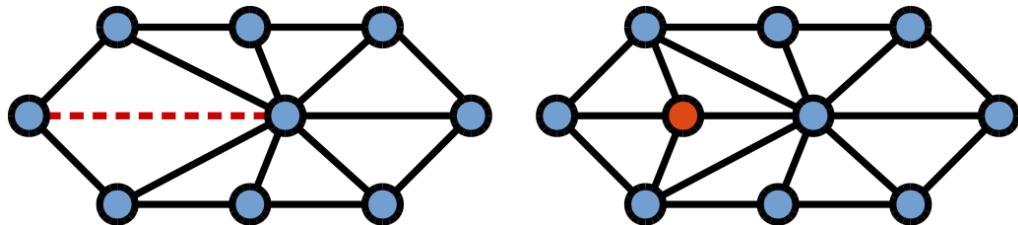
Efter *edge collapse* utförs kan ibland ett specialfall uppstå, se figur 6.4. I vanliga fall ingår en kant endast i två trianglar, i specialfallet ingår kanten i fyra trianglar. På grund av detta kan inte denna kant retrianguleras därför utförs alltid en kontroll efter varje *edge collapse* för att kontrollera

om fallet uppstått och i sånafall ta bort den vertexpunkt som inte är kopplad till resterande delar av meshen.



Figur 6.4: Specialfallet som kan uppstå vid en edge collapse

Om avståndet mellan två vertexpunkter är för långt placeras en ny vertexpunkt ut mitt i mellan dessa och binds till deras gemensamma vertexgrannar, se figur 6.5. Vertexpunkterna som var problematiska binds i sin tur om till den nya. Detta kallas edge split.



Figur 6.5: Den streckade linjen är för lång och edge split utförs, resultatet visas till höger.

## 6.3 Modellering

Meshen ändras med hjälp av verktyg som har olika egenskaper och former. Först testas ifall verktyget vidrör meshens vertexpunkter, om det är fallet flyttas vertexpunkterna i en riktning som beror av verktygets egenskaper. När alla markerade punkter har flyttats måste deras normaler uppdateras och området måste retrianguleras. Till sist skickas uppdateringarna till datorns GPU, se listing 6.1.

Listing 6.1: pseudokod för manipulering av meshen

---

```

float verktygsPosition[3];
float förflyttning[3];
vector<int> måsteUpdateras;

for( int i = 0:vertexArray.size() ) {
    if( vertexPosition inom verktygets radie) {
        vertexPosition + förflyttning;
        måsteUpdateras.push_back(i);
    }
}

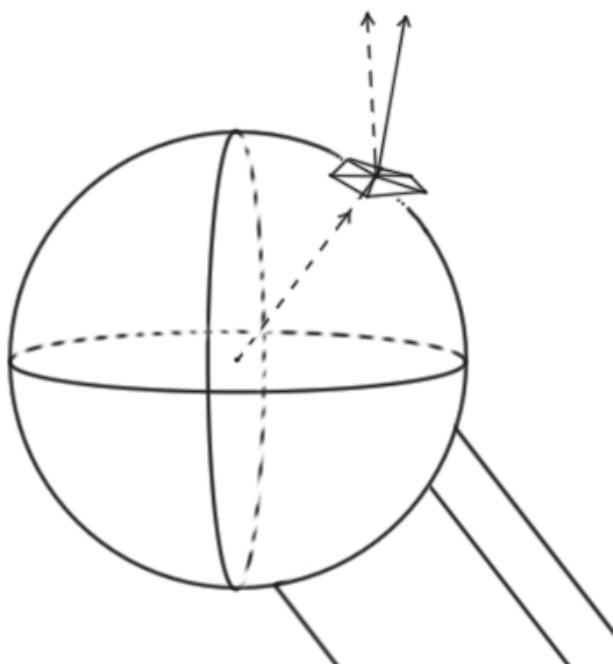
for (int i = 0:måsteUpdateras.size()) {
    updateNormal();

    if(avståndet mellan punkten och någon av grannarna är för stort) {
        skapa en ny vertexpunkt;
        skapa nya halfEdges och trianglar;
        skapa bindningarna i de nyskapade elementen;
        bind om sibling länkarna;
    }
}

```

```
if (avståndet mellan punkten och någon av grannarna är för litet) {  
    flytta punkten halvägs mot grannen;  
    ta bort grannen;  
    ta bort de två gemensamma triangeln och dess halfEdges;  
    bind om siblings;  
    bind om vertex punkternas halfEdge pekare;  
}  
}  
  
skickaUppdateringarGPU();
```

För att undvika att meshen växer in i sig själv sker modifieringen av meshen alltid i riktningen mellan vertexpunktens normalriktning och riktningen som bildas mellan verktygets origo och verexpunkten, se figur 6.6.



Figur 6.6: Den heldragna pilen är riktningen verexpunkten flyttas i

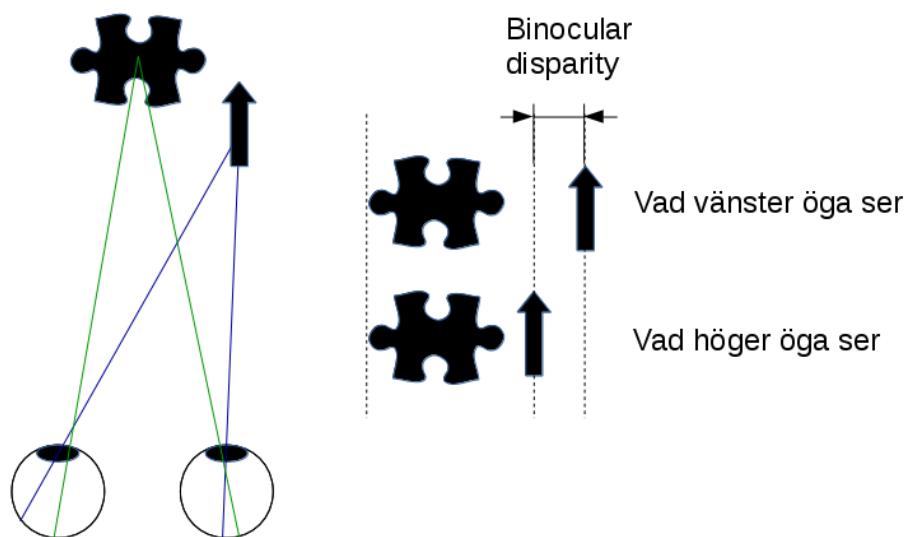
# Kapitel 7

## Datorgrafik

### 7.1 Användargränssnitt

När applikationen använder en HMD som skärm är det viktigt att inte ha en statisk heads-up display (HUD). Anledningen till detta är för att det kan vara obehagligt för användaren samt att det är opraktiskt.

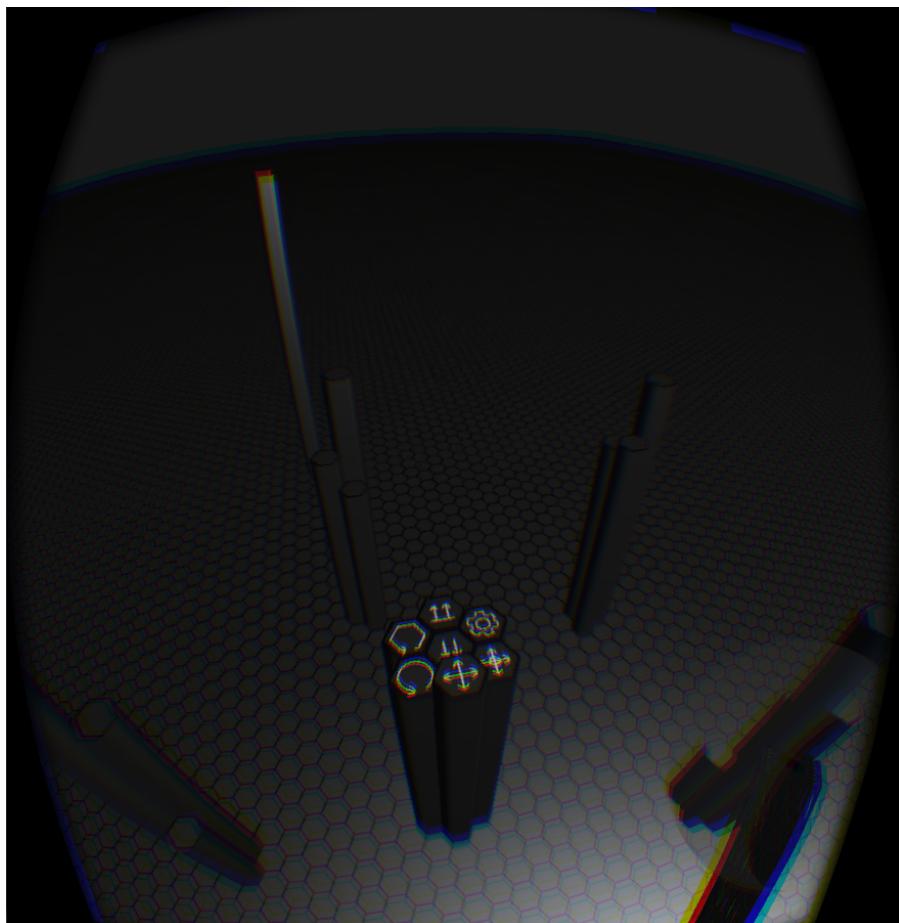
Ett av problemen är att människan har ett avstånd mellan båda ögonen som bidrar till att ögonen inte ser exakt samma sak. Läggs ögonens bilder över varandra kommer objekten att vara försjutna, se figur 7.1 denna försjutningen kallas *binocular disparity*. Hjärnan använder binocular disparity för att tolka hur långt bort ett föremål är, ju längre bort ett föremål är desto mindre blir skillnaden mellan ögonens bilder. På grund av detta finns det en liten skillnad mellan de renderade bilderna för båda ögonen i Oculus. Att använda en statisk HUD kan orsaka problem eftersom en HUD alltid renderas främst. Om ett objekt kommer närmre användaren än HUD:en uppstår en kontradiktion när binocular disparity antyder att objektet är närmre än HUD:en som renderas främst [6]. Med anledning av detta används texturer på wanden för att ge återkoppling om vilket verktyg som är aktiverat, hur detta åstadkommits beskrivs i ansnitt 7.2. Till exempel byts texturen på wanden till en bild av två korsade pilar när verktyget move väljs.



Figur 7.1: Höger och vänster öga ser olika bilder, försjutningen mellan objekten i bilderna kallas binocular disparity.

För att användaren lätt ska kunna se inom vilket områden som meshen kommer att manipuleras renderas en sfär med hög opacitet ovanpå wandens handtag. Denna sfär påverkas inte heller av ljus i scenen då det kan försämra sikten. För att åstakomma detta renderas denna sfär med egna shaders.

Eftersom applikationen är tänkt att kunna användas av flera olika typer av wands med olika många knappar behövs en meny för att välja funktioner. Figur 7.2 visar denna meny som representeras genom upphöjda hexagon boxar. Genom att föra wanden så att den kolliderar med en box och klicka på trigger knappen väljs funktionen. För att ge feedback till användaren att funktionen är vald höjs detta menyval upp några centimeter över de andra. På så vis behöver en wand endast en knapp för att vara kompatibel med applikationen



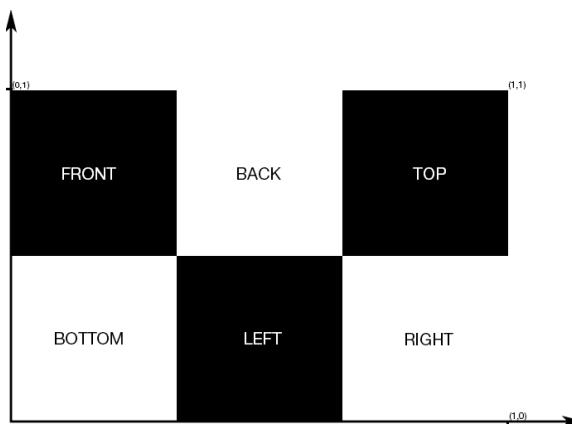
Figur 7.2: Funktionspanel och markeringar av hexagon boxar.

Användaren har även möjlighet att höja upp hexagonboxar från golvet. Dessa är tänkt att användas som en återkoppling till modelleraren om hur verkligheten ser ut omkring denne. Till exempel kan användaren markera ut gränserna för hårdvarans trackingområde eller de möbler och hinder som finns i rummet denne befinner sig i.

## 7.2 Texturer

Texturer används i applikationen för att ge mer detalj på de objekt som finns i scenen samt ge återkoppling när användaren byter funktion att interagera med. Bilderna läses från en DirectDraw Surface (.DDS) fil, vilket är ett kompressions format optimerat för effektiv inläsning av GPU:n [8].

För att mappa texturen på korrekt sätt används texturkoordinater i varje vertexpunkt av objektet. Figur 7.3 visar hur en texturfil kan mappas till en box. Framsidan av boxen ska alltså mappas med texturkoordinaterna  $(0.0, 0.5)$ ,  $(0.33, 0.5)$ ,  $(0.33, 1.0)$  och  $(0.0, 1.0)$  för att få rätt del av texturen på sin sida.



Figur 7.3: Uppdelad texturbild för varje sida av en box

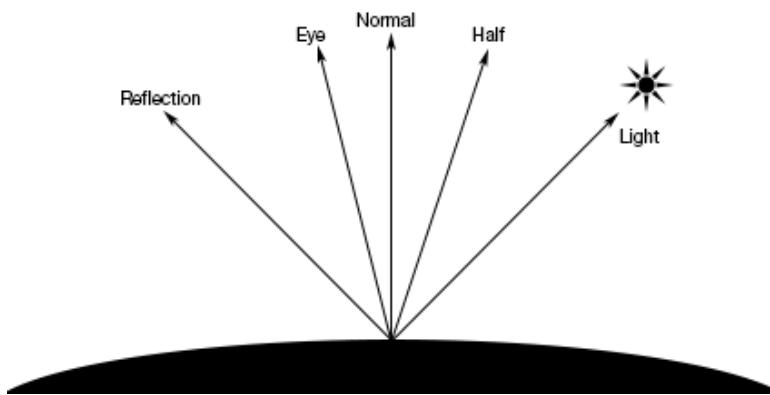
Eftersom mesh-objektet förändras under `runt i mē` och retrianguleras skulle en textur förstöras vid manipulation. Med anledning av detta renderas meshen endast med en färg och ljusmodell. För att åstadkomma detta renderas meshen med en enskild shader. Detta bidrar till att applikationen kräver mindre minne eftersom applikationen inte behöver texturkoordinater för varje vertexpunkt i meshen.

### 7.3 Ljusmodell

För att ge användaren en bra upplevelse när denne modellerar så är det en förutsättning att scenen har ett ljus som liknar ljuset utanför den virtuella världen. Ljus i den verkliga världen är komplicerat och beror på många faktorer. Ljuset i scenen är baserad på approximationer av verkligheten med hjälp av en förenklad ljusmodell.

Projektgruppen har valt att använda en ljusmodell som heter Blinn-Phong. Blinn-Phong är ett vidareutveckling av Phong's modell som huvudsakligen är uppbyggd av tre komponenter; ambient, diffuse och specular [3]. I Blinn-Phong modellen hanteras många saker på samma sätt som i Phong modellen. En sak som skiljer dessa är hur Blinn-Phong beräknar specular-termen. Istället för att använda en reflektions vektor för att räkna specular-termen definieras en vektor som kallas *halfway vector*. Det är en enhetsvektor exakt halvvägs mellan blickriktningen och ljusriktningen. Ju närmare denna *halfway vector* ligger i linje med ytans normalvektor, desto högre blir specular termens bidrag.

Med Blinn-Phong får inte lika skarpa kanter från specular termen som i Phong. En annan fördel med Blinn-Phong modellen är att den också är mer effektiv jämfört med Phong. Detta eftersom den beräkningstunga reflektionsvektorn i Phong inte längre behöver beräknas. Ytterligare en skillnad som kan ses i figur 7.4, är att vinkeln mellan *halfway vector* och ytnormalen är ofta kortare än vinkeln mellan blickriktningen och reflektionsvektorn. För att få liknande resultat som i Phong måste därför shininess exponenten vara något högre i Blinn-Phong.



Figur 7.4: Blinn-Phong ljusmodell

## 7.4 Realtidsrendering

Eftersom applikationen är tänkt att användas genom Oculus är det viktigt att garantera att applikationen inte dyker under *60 frames per second (FPS)*. Renderas scenen med mindre än 60 FPS så börjar använda uppleva en fördröjning som kan orsaka illamående och yrsel.

# Kapitel 8

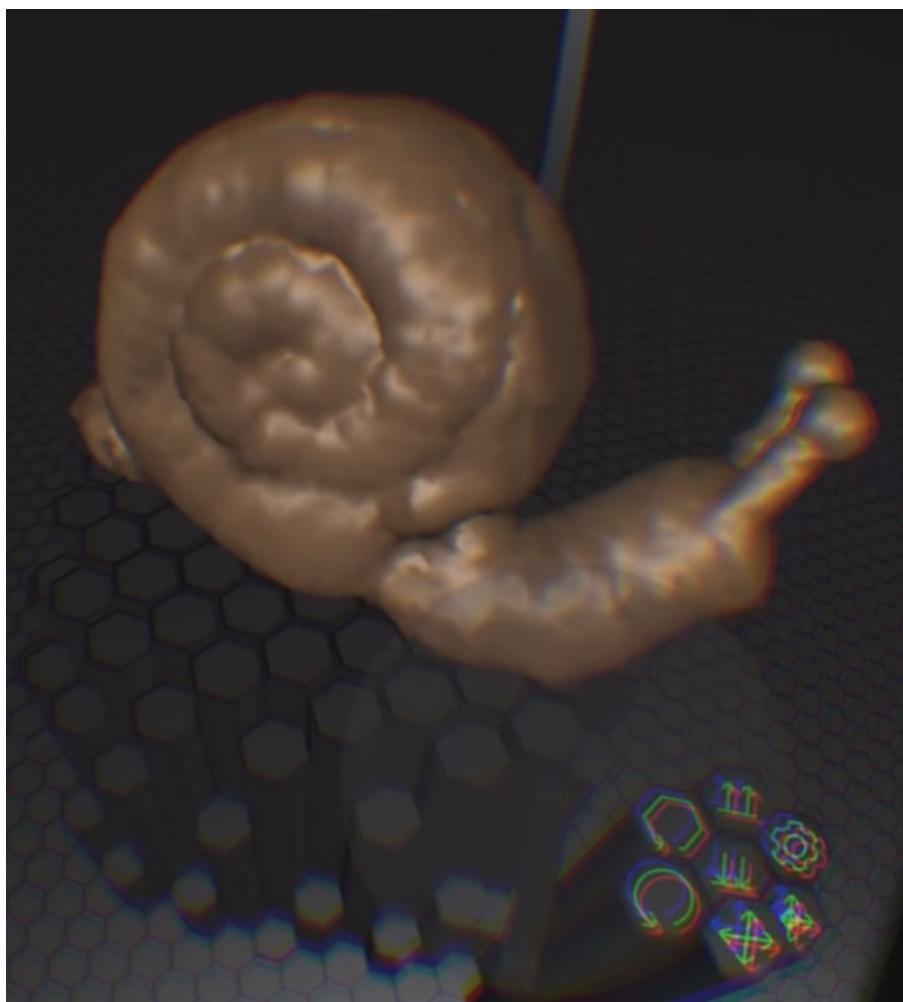
## Resultat

Projektet har resulterat i en applikation där användare i realtid kan forma en mesh med en HMD och en handhållen 6DoF-enhet.

- En enkel men välgjord scen har skapats för att användaren ska kunna orientera sig på ett bra sätt.
- Stöd för att använda en Oculus Rift, IS-900 och en prototyp av en magnet-baserad tracking-enhet har implementerats.
- Stöd för att ansluta tracking-enheter via VRPN.
- Funktioner för att enkelt kunna kalibrera och synkronisera olika tracking-enheter har implementerats.
- En mesh-struktur har skapats för att kunna förändra objekt i realtid.
- Funktioner för att förändra och retriangulera en mesh i realtid har implementerats. Ett exempel på en formad mesh kan ses i figur 8.1.

All funktionalitet har implementerats på ett strukturerat och begripligt sätt med syfte att underlätta för utvecklare att förstå och använda sig av källkoden.

Ett videoklipp av resultatet finns tillgänglig på: <https://youtu.be/UXwzPF6bLn8>



Figur 8.1: En snigel som modellerats i applikationen

# Kapitel 9

## Analys och diskussion

### 9.1 Metod

Projektgruppen har haft nytta av att utvecklingsmetoden är robust mot förändringar som uppstår under projektets gång. Projektet fick en vändning när beskedet att STEM system inte skulle levereras innan projektet tog slut. STEM system var det tracking-system som gruppen i första hand planerade att integrera och att använda. Beskedet var inte helt oväntat eftersom det redan innan beställningen fanns en viss osäkerhet när det skulle levereras. Men det blev ändå en besvikelse över att inte kunna använda sig av STEM system. Det innebar också att gruppen var tvungna att göra om sin tidsplanering. Kravet om att systemet skulle innehålla minst två konstnärer kunde inte uppfyllas eftersom projektgruppen inte längre hade hårdvaran för att testa dessa funktioner. Efter beskedet om att STEM system inte skulle levereras innan projektets slut fick gruppen ett samarbete med ett annat projekt som utvecklar en ny 6DoF-enhet. Dessa förändringar bidrog till att projektplanen fick ses över och skrivas om samt att de nya förutsättningarna diskuterades med kund.

Mycket av scrum-metoden har följts naturligt av gruppmedlemmarna samtidigt som vissa saker har kännts onödiga och begränsande. På grund av detta har scrum-metoden anpassats så att den passar bättre med gruppens struktur och krav.

Det har ibland varit svårt att bryta ner *stories* till flera mindre *tasks* som kunde utföras under en dag. Detta bidrog till att projektet vid vissa tillfällen inte kändes som det gick framåt.

### 9.2 Resultat

#### 9.2.1 Mesh

Anledningen till att en mesh valdes till projektet över att använda volymbaserad data och marching cubes var för gruppen inte hade tidigare erfarenhet av att arbeta med volymbaserad data och marching cubes. Projektgruppen såg även vid projektstart problem med att det skulle ta mycket minne att använda det volymbaserade sättet och då med risk för låg upplösning eller mycket projekttid för att hantera detta. Med avseende på projektets längd ansågs därför en mesh vara bättre anpassad.

Att triangulera en mesh visade sig dock vara mer invecklat än förväntat och tog lång tid att implementera. Eventuellt är den tid det tog jämförbar med den tid som krävs för att implementera de sök och komprimeringsalgoritmer som behövs vid en volymbaserad datastruktur.

En anledning till att det tog lång tid var att trianglar och punkter i en mesh kan bindas till varandra på många olika sätt och extremfall kan uppstå. Dessa extremfall måste tas om hand men är svåra att

upptäcka och förstå. Detta hade eventuellt kunnat gå bättre om utvecklarna gjort en avancerad studie om meshes innan implementationen började. Det är osäkert om tid hade vunnits på detta sätt.

En annan anledning var att gruppen valde att testa tre olika datastrukturer för att hitta en som hade liten påverkan på prestandan.

Att triangulationen tog lång tid gjorde också att det blev lite tid över till att implementera olika sorters modelleringsfunktioner. Därför finns nu inte alla de funktioner som var planerade.

## 9.2.2 Kalibrering

Kalibreringsfunktionen bygger just nu på proprioception då användaren får uppskatta var wanden borde finnas i VR-världen. Ett problem med denna metod är att en felmarginal kan uppstå vilket påverkar resultatet. Ett annat problem är att människor upplever en framåtsträckt arm som kortare än om armen var utsträckt åt sidan. Ett alternativ är att hämta HMD positionen samtidigt som wand positionen. Genom att på så sätt välja fyra unika punkter med samma förhållande mellan wand och HMD skulle felmarginen minska.

Ett problem med båda metoderna är att ge återkoppling som användaren förstår. Det är svårt att göra konfigurationsfunktionen användarvänlig. Detta är ett område som kan drastiskt förbättras och skulle ha arbetats vidare på om det funnits tid till det.

För en generell lösning av position och orientation kalibrering behövs ytterliggare fyra punkter till en andra transform som hanterar orientationen. Ytterliggare fyra punkter som väljs på användarens proprioception skulle innebära större felmarginal. Ett fel i orientationen är även svårare för användaren att kompensera för än positionsfel.

## 9.2.3 Hårdvara

Den hårdvara som används i projektet ansluts via sladd samt att tracking området är begränsat både för Intersense IS-900 och Oculus rift. Detta begränsar användaren som hela tiden måste se till att denne befinner sig inom det trackade området och att sladdar inte rycks ut. Detta har gjort att applikationen inte har kunnat användas med rörelsefrihet som var tänkt.

Hårdvaran som har använts delas av andra utvecklare och det har därför inte varit möjligt att ändra plats och inställningar för dessa. Detta tillsammans med att gruppen inte haft administrativa rättigheter på de datorer som har använts har bidragit till att det tagit längre tid att få hårdvaran att fungera som önskat.

## 9.3 Arbetet i ett vidare sammanhang

3D-modelleringsverktyg används mer och mer i dagens samhälle. Både spelindustrin och filmindustrin växer snabbt och båda dessa behöver hela tiden bättre och bättre modelleringsverktyg. Samtidigt utvecklas och används 3D-printers alltmer och även modellerare som behöver fysiska modeller så som industridesigners använder digitala modelleringsverktyg allt oftare. Tillsammans med det ökade intresset för virtuell verklighet och relaterad hårdvara gör detta att DooVR är mycket relevant i dagens samhälle.

DooVR är verktyg som med fördel kan användas i alla ovanstående industrier. Men applikationen kan också användas av amatörer då hårdvaran som krävs nu är så pass billig att många har möjlighet att köpa den.

# Kapitel 10

## Slutsatser

### 10.1 Frågeställningar

En stor del av projektets ursprungliga frågeställningar har inte kunnat bli besvarade då de inte hanterats. Under projektets gång byttes fokus vilket innebar att nätverksfunktionalitet som två modellerare och åskådare inte kunde uppnås. Med fokusbyte skapades nya frågeställningar till projektets nya riktning.

- Kan en Oculus Rift och en 6DoF-enhet hjälpa konstnärer att modellera i 3D på ett sätt som gör att modelleringen blir enkel att utföra och upplevs som modellering i verkligheten?

Fler användartester behöver göras innan denna fråga kan besvaras objektivt och till fullo. Det subjektiva svaret är att projektmedlemmarna tycker att det fungerar bättre med denna applikation än de som körs på 2D-skärmar. Samtliga utomstående som prövat applikationen har uttryckt positiva åsikter om att det känns bra och naturligt att modellera i applikationen.

- Vilka är fördelarna och nackdelarna med volym och ytbaserade datastrukturer och vilken av dessa två typer passar bäst till projektet?

Fördelen med en ytbaserad datastruktur är den tar relativt lite plats. Nackdelarna är det måste göras avancerade triangulationsberäkningar och många beräkningar behöver göras för att veta ifall en punkt finns innanför eller utanför volymen.

Fördelarna med en volymbaserad datastruktur är att det inte krävs avancerade beräkningar för att ta reda på ifall en punkt är i volymen. Nackdelen är att strukturen tar så stor plats att sök och komprimeringsalgoritmer krävs.

Det är svårt att bedömma vilken struktur som passar applikationen bättre. En ytbaserad lösning ger möjlighet att öka modellens upplösning medan en volymbaserad lösning gör att fler modelleringsfunktioner kan fungera i realtid. Optimalt så implementeras båda metoderna så att användaren kan välja vilken metod som används beroende på vad som ska göras.

- Går det att skapa en kalibreringsfunktion som länkar ihop koordinatsystemet för HMD med 6DoF-enhetens koordinatsystem, oavsett hur användarens tracking utrustning är uppsatt?

I nuvarande lösning kalibreras positionen av 6DoF-enheten men inte orientationen. En generell lösning behöver spara ytterliggare rotationsdata till en andra transform som hanterar orientationen. Då även orientationen beror på proprioceptionen leder det till större felmarginal, vilket resulterar i att 6DoF-enheten blir svårhanterlig.

## 10.2 Framtida arbete

Eftersom vertexpunkter och trianglar hela tiden tas bort och läggs till ska det implementeras vectorer som innehåller de lediga platserna i indexArray samt vertexArray vectorerna. Genom att använda den sista elementet i vektorn som pekar på lediga element undviks borttagning av element i vectorer, som är kostsamt för prestandan, utan endast sista elementet behöver tas bort. Idagsläget nollställs platsen i vektorn när en triangel eller vertexpunkt tas bort men den ersätts aldrig med en ny vilket innebär att dessa vectorer blir större och större vilket påverkar prestandan.

Även sökalgoritmer behöver implementeras för att vertexpunkterna ska kunna sökas igenom i 60 FPS. Just nu kan cirka 10,000 vertexpunkter sökas igenom samtidigt med 60 FPS. Denna upplösning måste ökas minst tiofaldigt för att ge konstnären den precision som krävs. Då kommer meshen behöva delas upp i olika partitioner som alla innehåller högst 10,000 punkter. Den struktur som är planerad att implementeras är ett *octree*<sup>1</sup>.

---

<sup>1</sup><http://en.wikipedia.org/wiki/Octree>

# Litteratur

- [1] Atlassian. *Gitflow Workflow*. URL: <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow> (hämtad 2015-04-24).
- [2] Alan C. Bovik. *Handbook of Image and Video Processing (Second Edition)*. Elsevier Academic Press, 2005.
- [3] John F Hughes och James D Foley. *Computer graphics: principles and practice (3rd Edition)*. Pearson Education, 2013.
- [4] InterSense. *IS-900 System*. URL: <http://www.intersense.com/pages/20/14> (hämtad 2015-04-27).
- [5] Krystof Litomisky. *Consumer RGB-D Cameras and their Applications*. URL: <http://alumni.cs.ucr.edu/~klitomis/files/RGBD-intro.pdf> (hämtad 2015-07-08).
- [6] Oculus. *Best Practices Guide (9 Januari, 2015)*. URL: [http://static.oculus.com/sdk-downloads/documents/Oculus\\_Best\\_Practices\\_Guide.pdf](http://static.oculus.com/sdk-downloads/documents/Oculus_Best_Practices_Guide.pdf) (hämtad 2015-05-01).
- [7] Ken Schwaber och Jeff Sutherland. *The Scrum Guide™ The Definitive Guide to Scrum: The Rules of the Game*. Scrum.Org och ScrumInc.
- [8] Dominé Sébastien. *Texture Compression in OpenGL*. URL: [http://www.oldunreal.com/editing/s3tc/ARB\\_texture\\_compression.pdf](http://www.oldunreal.com/editing/s3tc/ARB_texture_compression.pdf) (hämtad 2015-05-12).
- [9] VRPN. *VRPN*. URL: <http://www.cs.unc.edu/Research/vrpn/> (hämtad 2015-05-01).
- [10] Allan Watt. *3D Computer Graphics (3rd edition)*. Pearson Education 2000.
- [11] Greg Welch och Eric Foxlin. *Motion Tracking: No Silver Bullet, but a Respectable Arsenal*. URL: [http://www.cs.unc.edu/~tracker/media/pdf/cga02\\_welch\\_tracking.pdf](http://www.cs.unc.edu/~tracker/media/pdf/cga02_welch_tracking.pdf) (hämtad 2015-06-18).
- [12] Denis Zorin. *Mesh data structures*. URL: <http://mrl.nyu.edu/~dzorin/ig04/lecture24/meshes.pdf> (hämtad 2015-05-12).

## Bilaga A

### Arbetsfördelning och ansvarsområden

Gruppmedlem	Ansvarsområden	Arbetsuppgifter
Isabelle Forsman	Scrum master	Mesh, Oculus rift
Johan Nordin	Sekreterare och dokumentansvarig	VRPN, UI, kalibrering, texturer, wand3D
Jonathan Bosson	Kodansvarig	VRPN, UI, kalibrering, texturer, wand3D
Olle Grahn	Productowner, test och kravansvarig	Mesh, Oculus rift

# Bilaga B

## Ordlista

Ord	Betydelse
Daily scrum	Ett kort möte där dagens och gårdagens uppgifter diskuteras.
Frames per second (FPS)	Bilder per sekund
Head mounted display (HMD)	Huvudmonterad skärm, ett exempel på sådan är Oculus Rift
Headset	
Open-source	Öppen källkod, programvaran är fri att använda och får modifieras
Product backlog	Innehåller de stories som ska uppfyllas innan projektets deadline
Productowner	Produktägare, ansvarsroll inom Scrum
Scrum	En agil utvecklingsmetodik
Scrum master	Ansvarsroll inom scrum
Sprint	Ett tidsintervall då projektgruppen arbetar för att uppfylla sprint backlogen
Sprint review	Möte där projektgruppen presenterar och diskuterar utfört arbete med kund
Storie	Ett krav som ska uppfyllas under sprinten
Task	En uppgift som ska avklaras för att uppfylla en storie, en storie har flera tasks
Tracking-system	Spårningssystem för VR-enheter
Virtual reality (VR)	Virtuell verklighet