NTNU
Det skapende universitet

Digitale Verktøy for Historiefortelling

Andreas Ulvøen

Master i elektronikk
Innlevert: juni 2015
Hovedveileder: Andrew Perkis, IET

Norges teknisk-naturvitenskapelige universitet
Institutt for elektronikk og telekommunikasjon

Oppgavetekst:

Digitale Verktøy for Historiefortelling

Ny teknologi innen digitale verktøy for historiefortelling gjør det mulig å fortelle historier på nye måter. Hodemonterte briller (HMD) gir en stor grad av innlevelse ved å la brukeren kunne se seg om i omgivelsene på en intuitiv måte. Sporingssystemer som Microsoft Kinect og andre mer avanserte sporingssystemer gjør det mulig for en stor grad av interaksjon med omgivelsene ved å gjengi fysiske bevegelser. Spillmotorer som Unity [29] og Unreal Engine [30] gjør det enklere å lage fortellinger med ulike interaksjonspunkter uten å måtte ha store mengder med spesialisert kunnskap rundt det tekniske aspektet ved å lage en digital historie. Ved å bruke disse verktøyene sammen blir det mulig å lage nye interaktive fortellinger med en stor grad av innlevelse og tilstedeværelse.

Digitale Verktøy for Historiefortelling er en masteroppgave under Immersive Media Technology Experiences (IMTE) ved NTNU. I masteroppgaven skal det greies ut om bruken av digitale verktøy for historiefortelling for å lage en interaktiv fortelling. Dette skal gjøres ved å utvikle en prototype på en interaktiv fortelling ved hjelp av Oculus Rift [27] (HMD), Optitrack [28] (kroppssporingssystem) og Unity. (spillmotor) Oppgaven skal også vurdere hvordan en slik interaktiv fortelling oppleves av en bruker. Dette skal gjøres med en brukertest.

Faglærer og veileder: Professor Andrew Perkis

Innhold

Sammendrag

Denne masteroppgaven er et forskningsprosjekt ved Immersive Media Technology Experiences (IMTE) og tar for seg digitale fortellerverktøy og hvordan disse kan brukes til å lage en interaktiv fortelling. I oppgaven har det blitt laget en interaktiv fortelling ved hjelp av spillmotoren Unity, der interaksjon med fortellingen er i form av bevegelsessporing. Oculus Rift Development Kit 2 [27], en hodemontert brille med display ble brukt for å øke graden av innlevelse. En brukertest har blitt utført for å kunne si noe om hvordan en bruker opplever innlevelse og deltagelse i en interaktiv digital fortelling. Det har også blitt diskutert og greid ut om hvordan Unity [29] kan brukes som et digitalt fortellerverktøy.

Abstract

This master thesis is a research project at Immersive Media Technology Experiences (IMTE) and deals with digital tools for storytelling and how such tools can be used to make an interactive story. An interactive story has been made with the help of the game engine Unity, with interactions being in the form of the tracking of movements of the user. Oculus Rift Developement Kit 2 [27], a head-mounted display was used to increase immersiveness. A user test was performed to enable discussion about how a user experience immersiveness and participation in an interactive digital story. In addition Unity has been examined on how it is to use as a digital tool for storytelling.

Kapittel 1

Forkortelser og ordforklaringer

Forkortelse	Forklaring
HMD	Head-Mounted Display, hodemontert display
HTML	Hyper Text Markup Language
IMTE	Immersive Media Technology Experiences
Plugin	Lite program for ekstra funksjonalitet i et større program
Prefab	Uttrykk brukt i Unity, et ferdig sammensatt objekt
VR	Virtual Reality
HRTF	Head-Related Transfer Function
VRT	Virtual Reality Therapy
DVD	Digital Versatile Disc
NPC	Non Player Character
Script	Program
SCUMM	Script Creation Utility for Maniac Mansion
DirectX	Grafikk API for Windows.
API	Application Programming Interface
Unreal Engine	Spillmotor
Unity	Spillmotor
CryEngine	Spillmotor
Source	Spillmotor
C#	Programmeringsspråk
Optitrack	Optisk sporingssystem
LED	Light Emitting Deode
PhysX	Fysikkmotor
HDR	High Dynamic Range
GameObject	Generelt objekt i Unity, grunnleggende beholder for egenskaper
Blender	Modelleringssoftware
Kinect	Kroppssporingssystem laget av Microsoft
IK	Inverse Kinematics
VRPN	Virtual-Reality Peripheral Network
UART	Unity AR Toolkit
Mecanim	Animasjonssystem i Unity
JavaScript	Programmeringsspråk
DK	Development Kit

Kapittel 2

Introduksjon

Digitale Fortellerverktøy.

Digitale fortellerverktøy er verktøy for å fortelle digitale fortellinger. Ettersom at digital teknologi har utviklet seg har flere og flere verktøy blitt tilgjengelig for historiefortellere. Alt fra enkle verktøy som Word og Powerpoint til mer avanserte verktøy med mer funksjonalitet er tilgjengelig for den moderne forteller. I 2009 kom Oculus VR ut og kunngjorde sin hodemonterte brille, Oculus Rift. Slike briller har blitt prøvd flere ganger før, for eksempel Nintendos Virtual Boy [26] i 1995, men det er først i de senere år at datakraft og displayteknologi har blitt kraftig og billig nok for å kunne lage et produkt som holder stand. Ridende på bølgen til Oculus har flere andre selskap fått øynene opp for mulighetene og utvikler sine egne briller. I tillegg er det flere små og store firma som kommer ut med produkter ment til bruk i VR-sammenheng, ofte bevegelsessporing av forskjellige slag.

Innlevelse og deltagelse.

Innlevelse er en stor del av å fortelle historier. Ved å påvirke den som blir fortalt historiens sanser med riktige inntrykk blir fortellingen levende og virkelig for lytteren. Denne påvirkningen kan være så banal som å fortelle en skummel historie rundt et bål. Stemningen med nattemørket, bålet, rare lyder fra skogen og pusten som holdes fra dem som lytter sørger for at den emosjonelle responsen til fortellingen blir sterkere enn hvis den samme historien hadde blitt fortalt ved lunsjbordet. Nyere digital teknologi gjør det mulig å påvirke mange av våre sanser med realistiske inntrykk slik at vi kan bygge verdener som oppleves som “ekte”, Virtual Reality. Digitale verktøy gjør det også mulig for en bruker å delta i handlingen. Ved å utføre en handling og se verdenen rundt seg respondere får brukeren en sterkere følelse av tilstedeværelse.

Oculus Rift.

Oculus Rift [27] har brakt med seg en stor interesse for Virtual Reality i de senere årene. Fra Oculus VR la ut sin kickstarterkampanje i 2009 har Virtual Reality kommet på dagsordenen for flere store aktører. Oculus Rift er en såkalt Head-Mounted Display (HMD), en brille med en skjerm på innsiden montert på hodet til brukeren. Samtidig har prosesseringskraften i moderne datamaskiner blitt kraftig nok til å kunne lage realistiske virtuelle miljøer hjemme i stuen. I etterkant av Oculus har flere andre firmaer kunngjort sine egne briller, slik som HTC og Valves RE Vive [31] og Samsungs Gear VR. [32] Spillindustrien og filmindustrien er særlige pådrivere til at Virtual Reality skal bli populært ved å produsere innhold, men tiden vil vise om VR-briller kommer til å bli enkle og billige nok til at de blir allemannseie.

Virtuelt Legekontor.

Virtuelt legekontor er navnet på prototypen som har blitt laget i masteroppgaven. Den kombinerer bruken av Unity som et digitalt fortellerverktøy med bruken av Oculus Rift og Optitrack for interaksjon med brukeren. Prototypen har blitt utviklet for å kunne se på hvordan Unity kan brukes som fortellerverktøy og hvordan brukerens opplevelse av Oculus Rift og bevegelsessporing påvirker innlevelsen og følelsen av deltagelse. Brukeren opplever handlingen fra perspektivet til legen og kan bevege hender fritt rundt for å utføre en svært forenklet ultralydundersøkelse.

2.1 Presentasjon av kapitlene

Teori: Teorien tar for seg bakgrunnstoff for de ulike byggeklossene i oppgaven og forklaringer av viktige terminologier.

Virtuelt Legekontor: Kapittelet tar for seg utførelsen og innholdet av prototypen som ble laget i løpet av masteroppgaven. Det går inn på hvordan de ulike verktøyene virker sammen og hvordan det har blitt satt sammen i Unity.

Brukertesting: Her blir det forklart hvordan brukertesten ble utført og de viktigste resultatene av brukertesten blir lagt fram.

Drøfting: I dette kapittelet diskuteres det hvordan Unity er som fortellerverktøy med styrker og svakheter og bruken av Oculus Rift og bevegelsessporing blir drøftet i lys av brukertesten. Til slutt blir det lagt fram mulige feilkilder til brukertesten.

Konklusjon: Her blir det konkludert om bruken av Unity som fortellerverktøy og hvordan brukerne opplevde å bruke Oculus Rift og bevegelsessporing. Det blir også lagt fram forslag til videre arbeid.

Tillegg: Her er det vedlagt kildekode og resultatene fra brukertesten.

Kapittel 3

Teori

3.1 Digitale fortellerverktøy

Oppgaven definerer digitale fortellinger slik:

Digitale fortellinger er fortellinger bestående av et eller flere digitale media, slik som bilde, lyd og interaktivitet. I sin enkleste form vil en digital fortelling være et bilde eller en film, mer avanserte former for digitale fortellinger kan ta i bruk verktøy.

Digitale fortellerverktøy er ut i fra definisjonen over verktøy som brukes for å lage og/eller muliggjøre en digital fortelling. Begrepet er svært bredt og nye verktøy blir stadig tilgjengelig. Eksempler på eldre, mer kjente verktøy er Windows Movie Maker, Powerpoint og Photoshop. I de siste par årene har det kommet svært mange nye verktøy på banen, ikke minst gjennom folkefinansieringsportalen Kickstarter. Mange av disse verktøyene har fulgt popularitetsbølgen til Oculus Rift og er laget for å gjøre en Virtual-Reality-opplevelse bedre.

3.2 Innlevelse

> Becoming part of Las Posadas instead of merely observing it transformed the experience for me. It was like the differenve betweeen wathcing a movien and suddenly becoming a character in it. To me, it vividly demonstradet the power of immersivness-one of the most compelling and magical aspects of interactive media.
>
> *Carolyn Handler Miller, Digital Storytelling; a creator's guide to interactive entertainment.*

Innlevelse er følelsen av å være til stede og del av en fortelling. En stor grad av innlevelse kommer av at en bruker har flere muligheter for interaksjon med fortellingen. Carolyn Handler Miller oppsummerer godt hva innlevelse er i boken "Digital Storytelling, a creators guide to interactive entertainment [25]. Hun beskriver en festival i Santa Fe som har som mål å gjenskape den bibelske fortellingen om Josef og Marias jakt etter et sted å føde. Folkegruppen går fra sted til sted og synger med ønske om innpass. Djevelen vil så komme fram og synger i mot, med tilrop fra folkegruppen. Hun bestemte seg for å gå fra å være en passiv tilskuer til å bli en aktiv deltager ved å gå inn i folkemengden og synge med. De fysiske og emosjonelle inntrykkene med å følge gruppen gjorde at hun følte seg fullstendig involvert i fortellingen.

3.3 Virtual Reality

Virtual Reality-teknologier er verktøy som har som mål å gi en altoppslukende opplevelse til brukeren. Virtual Reality forer brukeren en virtuell virkelighet som prøver å føles så ekte som mulig for brukeren, selv om ikke miljøet brukeren er i er ekte. Gjennom å påvirke de ulike sansene dannes det et mer og mer komplett bilde av denne virtuelle virkeligheten og brukeren føler en sterk innlevelse i den virtuelle virkeligheten.

Syn:

- Virtual Reality krever mye ut av verktøyene som skal brukes for å påvirke synet. Hvis forsinkelsen mellom hodebevegelser og oppdatering av bildet er for høy eller hvis bildene fra de to øynene skulle være usynkroniserte, forsvinner illusjonen av at bildet er ekte. I tillegg vil mange oppleve kvalme og bilsyke hvis forsinkelsen er for stor. [24]
- Oppløsningen av skjermen må være høy. Oculus Rift har i sine prototyper Development Kit 1 og 2 hatt problemer med at brukeren ser en "nettingeffekt", som om man skulle sett verdenen gjennom en myggnetting. Mønsteret på pikslene kan også ha litt å si på hvordan denne mønstereffekten oppleves. Siden skjermene ofte er plassert tett med øynene og linser med forstørring brukes for å kunne se skjermen er det høye krav til pikseltetthet. Skjermen må også ha lav "persistence". Full-persistence vil si at alle pikslene er på til enhver tid. Lav persistence betyr at pikslene er på et svært kort øyeblikk og så blir svart rett etter. Med høy persistence vil brukeren oppleve mye motion blur. Skjermen må i tillegg støtte en høy oppdateringsfrekvens. Konsumervarianten til Oculus sikter på 90 Hz. [23]
- Brillene må ha mulighet for å dekke et bredt synsfelt. En av de største salgspunktene for Oculus Rift var det svært brede synsfeltet på 100 grader. [27] Optimalt vil VR-brillene kunne dekke hele synsfeltet til øyet.
- Bevegelsessporing av hodet må kunne gjengi brukerens bevegelser så nøyaktig som mulig. Hvis enkelte frihetsgrader mangler eller sporingen ikke oppdateres raskt nok vil brukeren oppleve ubehag. Oculus Rift Development Kit 1 hadde kun mulighet for å måle rotasjonen til hodet til brukeren, slik at det var umulig for brukeren å for eksempel lene seg ut av et vindu. Når slike bevegelser ikke blir gjengitt av synet er det ikke bare desillusjonerende men også ubehagelig.

I tillegg spiller hørselsinntrykket en stor rolle for innlevelsen. Det finnes flere måter å gjenskape realistisk lyd. En av dem er å bruke hodetelefoner og binaurale teknikker som HRTF (Head Related Transfer Functions) [22]. Forbrukervarianten av Oculus Rift vil komme med innebygde hodetelefoner og støtte for lydprosessering med HRTF.

3.3.1 Historie

Mennesker har lenge prøvd å lage historier med involverende og oppslukende. Helt tilbake i 1968 kom en av de første digitale Virtual-Reality verktøyene. Det var en hodemontert brille som var så stor at den måtte henge fra taket. Brukeren kunne se svært enkle modeller av rom. [21] En stund senere kom Nintendo på banen med sin Virtual Boy i 1995. Maskinen var ment for å brukes mens den stod på en overflate, hadde ingen bevegelsessporing, hadde kun mulighet for å vise rødfarger og en oppløsning på kun 384 x 224 piksler. [26] Virtual Boy fikk en kjølig mottagelse og ble tatt av salg året etter i 1996.

Helt opp til 2009 var det ingen store framskritt å snakke om med tanke på Virtual Reality-teknologi. Da kom Oculus VR ut med sin kickstarterkampanje for Oculus Rift. I etterkant har flere andre aktører kommet ut med sine tolkninger av HMDer, Sony med sin Project Morpheus i mars, 2014, Google med sin enkle Google Cardboard, HTC og Valves HTC Re Vive og Samsungs Gear VR. Hver har sin forskjellige måte å håndtere hardware på, spesielt med tanke på Google Cardboard og Gear VR som baserer seg på at brukeren bruker en mobiltelefon som skjerm.

3.3.2 Bruksområder

Virtual Reality har flere bruksområder:

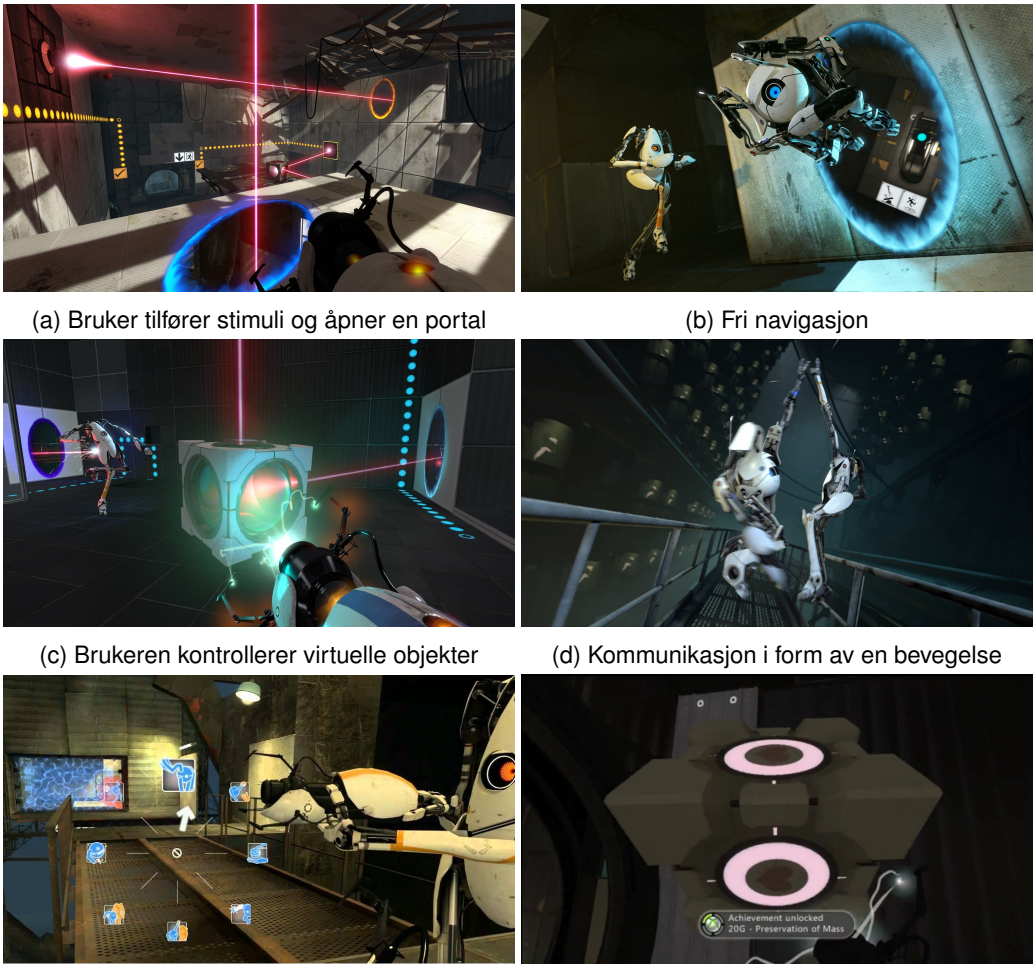
- **Journalisme** Virtual Reality gjør det mulig for en bruker å ta del i en reportasje på en mye større måte enn før. Ved å ha på seg brillen vil brukeren ha mulighet for å se seg rundt og føle seg som en del av hendelsene. Journalist Nonny De la Pena har utviklet en historie kalt "Project Syria" [20], der hun ved en hodemontert brille og posisjoneringssporing lar personer oppleve hvordan det er å gå ned en folksom gate i Syria, oppleve et bombeangrep og se situasjonen i flyktningeleirene.
- **Utdanning** For å holde på interessen til barn og unge er det lite som er bedre enn kombinasjonen av læring og teknologi. Ved å skape realistiske og involverende miljøer av historiske hendelser kan det bli en ny måte for skolen å lære fra seg kunnskap.
- **Film** 3D-briller har allerede vært i bruk en stund, spesielt på kinoer. Men hodemonterte briller vil ha en mye større grad av innlevelse der brukeren fritt kan se seg rundt for å studere omgivelsene. Hollywood lukter på bruken av VR-briller, Oculus VR har skap sitt eget interne filmstudio og pornoindustrien har sitt bidrag til filmmassen.
- **Kunst** Siden VR ofte tar i bruk dataanimert grafikk er det et særdeles nyttig verktøy for kunstnere som ønsker å lage kunst på en ny måte og som kanskje er utenfor det som er mulig i virkeligheten. Brukere vil kunne oppleve kunst i et virtuelt rom som å gå rundt i en gjenskapelse av Louvre.
- **Spill** Virtual Reality har fått et kraftig inntog i spillverdenen etter at Oculus VR kunngjorde Oculus Rift. Med den nye brillen kom det nye muligheter for å lage spill med stor grad av innlevelse. Spesielt kom det ut mange skrekkspill til Oculus Rift i den første perioden. De fleste av disse første opplevelsene var mindre spill og opplevelser laget av små spillselskaper. I senere tid har det kommet ut større titler fra store spillselskaper med integrert støtte for Oculus Rift. Ved å ta på seg brillen er det mulig å i større grad ta del i verdenen som spillutvikleren ønsker å skape.
- **Terapi** Virtual Reality Therapy (VRT) er en form for terapi brukt for å hjelpe personer med fobier og PTSD. Tradisjonelt sett har behandlingen for PTSD vært å la pasienten gjenoppleve hendelsen i fantasien. Ved bruk av VR-teknologi er det mulig å gjenskape de viktige detaljene i hendelsen og la brukeren gjenoppleve hendelsene i stor detalj. [19]
- **Opplæring** Virtual Reality er mye brukt i opplæring i situasjoner der det er sikrere å trene gjennom en simulering enn i virkeligheten. Eksempler her vil være piloter som trener på å fly i en simulator og andre stressende situasjoner med fare for liv og helse. Det amerikanske forsvaret utvikler opplevelser ved bruk av Oculus Rift for å se om det kan brukes i opplæring. [18]

3.4 Interaktivitet og digitale fortellinger

I boken “Digital Storytelling; a creators guide to interactive entertainment” blir det listet opp seks grunnleggende former for interaktivitet. Ved å kombinere disse seks grunnleggende formene er det mulig å skape svært mange ulike former for interaktive fortellinger.

- **Bruker tilfører stimuli, programmet reagerer** En av de mest grunnleggende formene for interaksjon. Kan komme i alle former, fra enkelt til komplisert. Et grafisk brukergrensesnitt er et typisk eksempel på denne type interaksjon, der bruker tilfører et stimuli i form av å trykke på en knapp med musepekeren eller tilsvarende handling og programmet reagerer.
- **Fri navigasjon** Denne typen interaksjon går ut på at brukeren er i stand til å navigere et miljø. Miljøet kan være alt fra en stor verden slik som i mange nye spill til enkle menyer i en DVD. I likhet med tilførsel av og reaksjon på stimuli er navigasjonen en universell del av interaktive programmer.
- **Kontroll av virtuelle objekter** Brukeren kan kontrollere og påvirke virtuelle objekter, slik som å åpne en dør, flytte på ting, skyte et gevær og lignende. Dette er en vanlig form for interaktivitet, spesielt i spill, men er ikke universell.
- **Kommunikasjon** Brukeren kan kommunisere med en NPC (Non Player Character) eller andre mennesker. Kommunikasjonen kan foregå med dialogvalg, stemmeaktivering, tekst og så videre. Denne kommunikasjonen er ofte toveis. Interaksjonstypen er vanlig, spesielt i historiedrevne spill, men ikke universell.
- **Brukeren kan sende informasjon** Interaksjon der brukeren sender informasjon. Et eksempel kan være et program på en flyplass som ber brukeren skrive inn kommentarer til sikkerhetskontrollen og så trykke en knapp for å sende informasjonen.
- **Brukeren kan motta ting** Brukeren kan samle ting, virtuelle eller reelle. Eksempler er å handle på Amazon.com eller å kjøpe et nytt våpen i et spill. Svært vanlig element i spill, der spilleren ofte kan motta ting i spillet og kan også motta “prestasjoner” for visse handlinger.

Et godt eksempel på bruken av alle seks interaksjonsformene er spillet Portal 2. [17] Hvert eksempel er illustrert i figuren 3.1 nedenfor. Brukeren kan tilføre stimuli i form av et trykk på en museknapp og programmet reagerer ved å lage en portal der brukeren siktet. 3.1a Mer avanserte former for denne interaksjonen er problemløsningen i spillet som ofte innebærer å manipulere objekter 3.1c og navigere et område. 3.1b Brukeren kan kommunisere med sine medspillere ved hjelp av å plassere markører 3.1e eller å bruke animasjoner. 3.1d Til slutt kan brukeren motta hatter og prestasjoner for å utføre handlinger. 3.1f



(a) Bruker tilfører stimuli og åpner en portal

(b) Fri navigasjon

(c) Brukeren kontrollerer virtuelle objekter

(d) Kommunikasjon i form av en bevegelse

(e) Brukeren sender informasjon

(f) Brukeren mottar en prestasjon

Figur 3.1: Illustrasjoner av de seks interaksjonspunktene

3.5 Spillmotorer

Spillmotorer er et eksempel på avanserte fortellerverktøy. Spillmotorer generelt er et rammeverk for å gjøre utvikling av spill enklere ved å håndtere grunnleggende aspekter ved et spill som fysikk, lyd, scripting, tegning av grafikk og så videre. Motorer har utviklet seg fra å være lite mer enn et programmeringsspråk slik som LucasArts Script Creation Utility for Maniac Mansion (SCUMM) [16], lansert så tidlig som 1987, til å bli svært fleksible integrerte utviklingsmiljø som gjør det fullt mulig for et lite team å skape gode spill og fortellinger.

3.5.1 Unreal Engine

Figur 3.2: Logoen til Unreal Engine

Unreal Engine [30] er laget av Epic Games og ble først vist fram i 1998. Hovedsakelig laget for førstepersonsskytere men har også blitt brukt til en rekke andre sjangere. Unreal Engine 4 har støtte for DirectX 11 og 12 for Windows, Xbox One og støtte for OpenGL for OS X, Playstation 4, iOS, Android, Ouya og Windows XP. Det er også støtte for å publisere spill for HTML5-kompatible webbrowsere. Ved å lage et nytt system med såkalte "blueprints", et visuelt scripting-system uten bruk av C++ mener utviklerne av motoren å ha fått ned kompileringstiden slik at det er mulig å gjøre små forandringer på store prosjekt uten å måtte kompilere alt på nytt hver gang. Motoren er gratis tilgjengelig for alle å bruke kommersielt inntil en viss grense. Over denne grensen må det betales royalties til Epic Games. Unreal Engine 4 har støtte for Oculus Rift.

3.5.2 Cryengine 3

CryEngine [15] er utviklet av spillutvikler Crytek. Motoren ble opprinnelig laget for spillet Far Cry som kom ut i 2004. Motoren har støtte for en rekke plattformer; Windows, OS X, Linux, Playstation 3/4, Wii U, Xbox 360/One, iOS og Android. Motoren har inntil nylig vært gratis å bruke for ikke-kommersielle produkter men er nå tilgjengelig enten via abonnementtjeneste på steam eller via en kommersiell lisens.

Figur 3.3: Logoen til Cryengine 3

3.5.3 Source

Figur 3.4: Logoen til Source Engine

Source [14] er Valves spillmotor og kom i bruk i 2004 når spillet Counter-Strike:Source ble lansert. Source har ingen definerte versjonsnummer men er under stadig utvikling. Skrevet i C++, motoren har støtte for følgende plattformer: Windows, OS X, Linux, Playstation 3, Xbox, Xbox 360 og Android. Source Engine er kun tilgjengelig under lisens, men det er mulig å få tak i Source Software Development Kit for å lage spill eller modifikasjoner i Source hvis man eier et Source-spill.

3.6 Bevegelsessporing

3.6.1 Optiske systemer

Optiske systemer bruker bildesensorer for å triangulere posisjonen til objektet som spores i rommet. Et typisk system består av flere kameraer som tar bilder ut av et romområde, ofte plassert rundt romområdet for å redusere muligheten for at noen av markørene blir skjult. Systemene har mulighet til å finne data som romposisjon, rotasjon, fart og akselerasjon. Fart og akselerasjon beregnes ut i fra flere bilderekker og rotasjon finnes ved å definere et objekt av tre eller flere punkter.

Det finnes flere typer av markører. Passive markører er dekt av et reflekterende materiale som reflekterer lys tilbake til kameraene. Fordelen med denne typen markør er at skuespilleren ikke trenger å ha på seg noe elektrisk utstyr. På den andre siden er det lettere for kameraene å forveksle mellom ulike markører. Enkelte oppsett prøver å forhindre denne effekten ved å ha et høyt antall kameraer. Aktive markører på den andre siden er markører som inneholder sin egen lyskilde og kan kontrolleres med software. Dette gjør det mulig å identifisere hvilken markør som er hvor, enten ved at en led lyser om gangen eller at alle lyser samtidig men med forskjellige mønstre.

Andre former for optiske systemer er markørfrie systemer. Disse tar i bruk bildebehandling og algoritmer for å gjenkjenne menneskelige former. Et eksempel er Microsofts Kinect [12], et system som prosjekterer et rutenett av infrarødt lys og beregner dybde i rommet basert på distorsjon av rutenettet. Kinect har mulighet for å gjenkjenne menneskefigurer og styre et virtuelt skjelett. Et annet eksempel er Leap Motions “Leap Motion Controller”, et lite objekt ment til å ha plassert på pulten. To infrarøde kameraer sender bildedata til Leap Motions software som lager en 3D-representasjon av bevegelsene i området. Dette gjør det mulig å lage en detaljert modell av hendene til brukeren. [13]

Figur 3.5: Leap Motion Controller i aksjon

3.6.2 Treghetssystemer

Treghetssystemer bruker gyroskoper og akselerometre for å måle bevegelse og posisjon. Ofte vil sensoren sende data trådløst for prosessering. Fordelene med å bruke treghetssystemer er at det krever mindre komplekse systemer, er lett å flytte rundt og har ofte mulighet for et stort målevolum. Bakdeler er at de fleste slike systemer må ta i bruk en annen form for sporing for å måle absolutt posisjon, som for eksempel et enkelt optisk system. Treghetssystemer kan også være mindre nøyaktig enn andre systemer, spesielt i billigere produkter.

Figur 3.6: Wii MotionPlus-kontroller

Nintendo Wii har en kontroller som er et godt eksempel på bruken av treghetssystemer. Den tar i bruk akselerometer og et optisk system for å finne kontrollerens bevegelser. Ved å bruke “Wii-Motion Plus”-tillegget har kontrolleren også to ekstra gyroskoper, slik at det er mulig å tolke mer komplekse bevegelser enn med kun kontrolleren alene. [11]

3.6.3 Magnetiske systemer

Magnetiske systemer for bevegelsessporing bruker magnetfelt for å finne posisjon og rotasjon. Systemet fungerer ved å sammenligne den magnetiske fluksen i tre ortogonale spoler i både sender og mottaker. Ofte er det målbare området mindre enn ved bruken av optiske systemer, men med den fordelen av at posisjoneringen ikke blir begrenset av ikke-metalliske objekter i målevolumet. Objekter av metall vil kunne påvirke magnetfeltet slik at det ikke er mulig å måle i området rundt objektet.

Et eksempel på magnetisk sporing er STEM-systemet til Sixense [10]. Systemet er ment til å være modulært men består hovedsakelig av en basestasjon, to håndkontrollere og tre sporingspakker som kan plasseres hvor som helst på kroppen, for eksempel rundt ankel, håndledd eller overkropp.

Figur 3.7: Stem system, hentet fra [10]

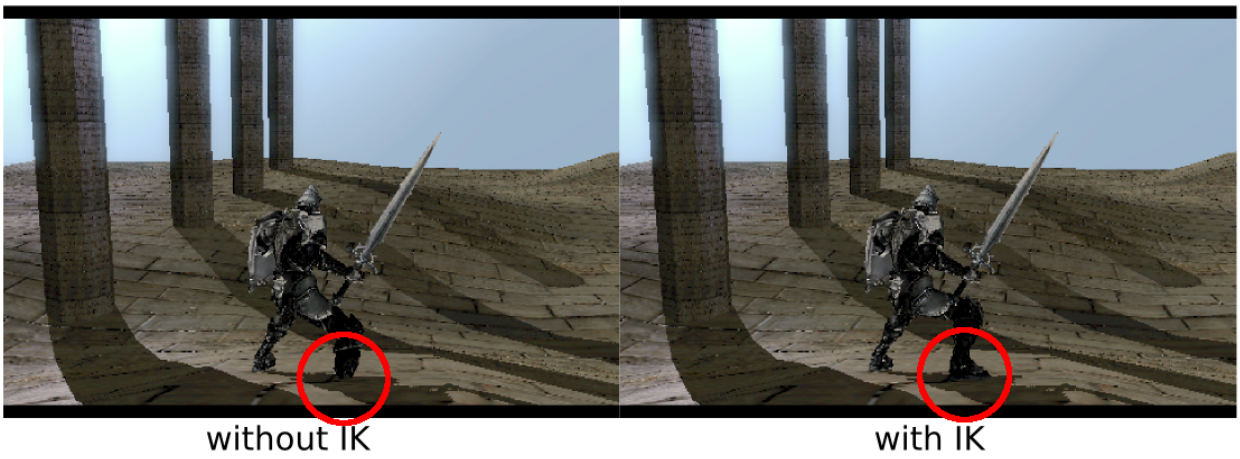
3.6.4 Mekaniske systemer

Mekaniske systemer bruker målinger fra fysiske objekter for å bestemme hvilke bevegelser er gjort, som for eksempel målinger fra et ledd som forteller i hvilken posisjon leddet er i. Et mekanisk system for bruk til å lage animasjoner er mulig ved at en skuespiller tar på seg et ytre skjelett med sensorer. Bevegelsene fra skuespilleren forandrer på posisjonen til leddene og denne posisjonen blir lest til programvaren. Slike systemer har svært stor nøyaktighet i reproduksjonen av bevegelser, er relativt rimelige i forhold til andre systemer, kan være trådløst uten muligheter for å miste data på grunn av objekter i målevolumet og har et svært stort målevolum. Enkelte systemer vil kunne gi Force Feedback.

3.6.5 Inverse Kinematics

I animasjonsarbeid er det vanlig å utstyre objekter som skal animeres med et skjelett laget av bein som henger sammen. Et vanlig eksempel vil være menneskelignende modeller som har et tilsvarende menneskelignende skjelett. Alle beinene er tilegnet vekter som bestemmer hvordan de påvirker modellen. Forward Kinematics er en teknikk som kort fortalt går ut på å flytte bein fra rot og utover for å nå et punkt. Et eksempel på det er å flytte skulder, så overarm, underarm og til slutt hånd for å nå den posisjonen som er ønskelig. Inverse Kinematics er den motsatte prosessen. Man begynner med et punkt og jobber seg bakover til rotbeinet ved å først plassere hånd, underarm, overarm og til slutt skulder. Ofte er denne prosessen under visse begrensninger slik som at bein må henge sammen og at albueledd kan roteres innenfor et avgrenset område.

Inverse Kinematics brukes ofte i spill der det er nødvendig å utføre handlinger det ikke er mulig å lage animasjoner på i forkant, slik som ofte vil oppstå hvis interaksjoner med omgivelser er dynamiske. Et eksempel er hvis en modell av en mann går opp en trapp. Ved hjelp av inverse kinematics kan man sette et “mål” for fotbladet på trappetrinnet slik at modellen kan bevege seg realistisk uavhengig av trappens egenskaper. Forskjellen mellom bruk av IK og ingen IK kan sees i figuren under. 3.8

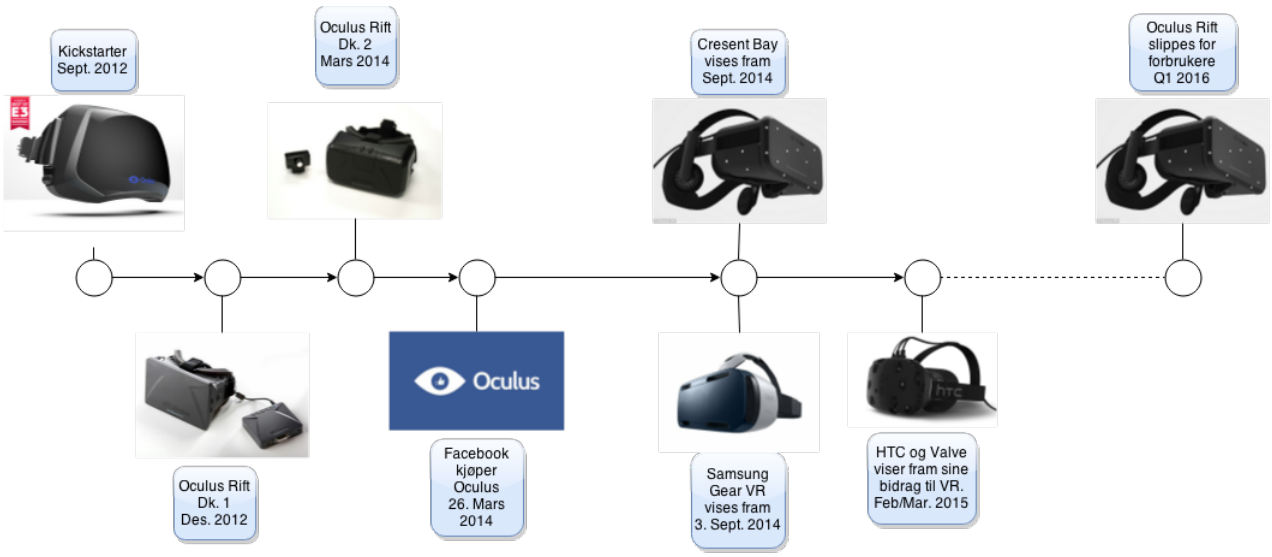


Figur 3.8: Eksempel av IK, hentet fra Gamasutra [1]

3.7 Oculus Rift

Figur 3.9: Oculus Rift Developement Kit 2

Oculus Rift er en HMD, Head-Mounted Display, en skjerm montert på hodet til brukeren. Skjermen blir fysisk delt i to og hvert øye ser bare den ene halvdelen av skjermen. Ved å sende to ulike bilder med perspektiv plassert side ved side får man en effekt av å se tredimensjonalt, en stereoskopisk effekt.



Figur 3.10: Tidslinje for Oculus Rift

I løpet av årene fra Oculus VR først viste fram en konseptmodell via Kickstarter har brillen hatt flere iterasjoner. 3.10 Den første modellen tilgjengelig for utviklere var Development Kit 1. Brillen var ute for salg fra 26 september, 2012 til 12 mars 2014. Den hadde en 18 cm stor skjerm med en oppløsning på 1280 x 800 piksler, med et synsfelt inne i brillen på 90 grader. Ved å bruke en kombinasjon av gyroskop, akselerometre og magnetometre kunne den spore rotasjonen til brillen.

Den neste viktige iterasjonen, Development Kit 2, ble kunngjort i mars 2014. DK 2 forbedret mange av aspektene som er viktig for Virtual Reality i forhold til DK 1. DK 2 hadde en bedre skjerm som både hadde en større oppløsning, bedre oppdateringsfrekvens og hadde mye mindre motion-blur enn DK 1. Det nye utviklersettet kom også med et lite infrarødt kamera, som ble brukt til å spore plasseringen til brillen i rommet. Små infrarøde LEDs under plasten i brillen dannet et mønster som gjorde det mulig å spore bevegelsene til brillen så lenge brillen var innenfor kameraets synsfelt. Det var ikke plassert LEDs på baksiden av brillen, noe som gjorde at kameraet ikke kunne spore posisjonen til brukeren dersom brukeren snudde hodet 180 grader.

Den siste store modellen blir en konsumermodell som er ment til å komme i salg i det første kvartalet i 2016. Denne modellen har igjen forbedret de fleste aspektene ved disse brillene, med høyere oppløsning og bedre bevegelsessporing ved at brillen har fått plassert infrarøde LEDs på baksiden. Den mest interessante nyheten er at brillen kommer med integrerte hodetelefoner og støtte for HRTF, som lager et svært realistisk lydbilde. Oculus VR har kommet med anbefalinger for maskinvare for å kunne ha en god opplevelse av VR [23]. Anbefalt grafikkort er en GTX 970 / AMD 290, som i 2015 er svært kraftige og dyre kort.

3.8 Optitrack

Optitrack [28] er et firma som driver hovedsakelig med utstyr for bruk i profesjonell bevegelsessporing. Firmaet leverer en komplett løsning for bevegelsessporing, både hardware og software. Kameraene er sensitive både for vanlig lys og infrarødt spekter og har sine egne kilder for infrarødt lys. Ved å montere små kuler med en reflekterende overflate vil kameraene kunne fange opp bevegelser. Ved å lage spesielle mønstre av kuler kan sporingsprogrammet beregne posisjon og rotasjon med svært stor nøyaktighet.

Figur 3.11: Avansert rigg, hentet fra optitracks hjemmesider [28]

3.9 Unity

Figur 3.12: Logoen til Unity

> Few companies have contributed as much to the flowing of independently produced games as Unity Technologies.
>
> *Dean Takahashi, 2012 [9]*

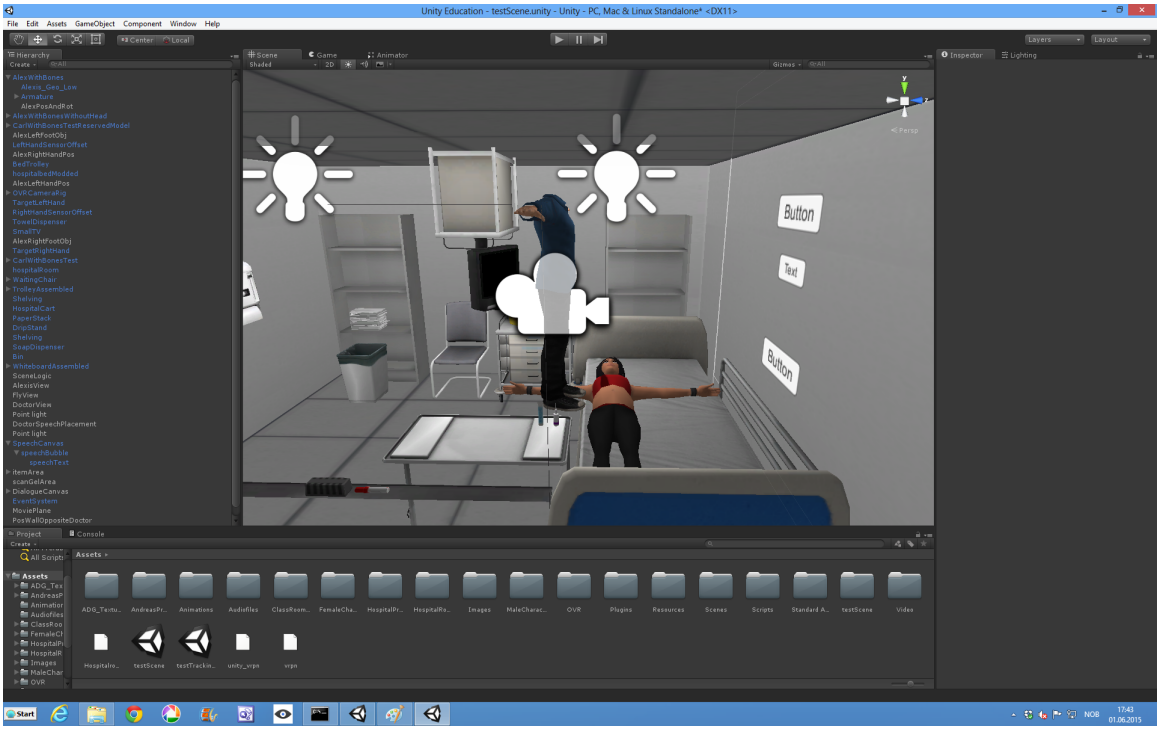
Unity er en spillmotor brukt i utviklingen av en rekke spill. [8] Spillmotoren ble først lansert den 8 juni, 2005 for Mac OS. Motoren hadde på det tidspunktet støtte for en shader-basert OpenGL renderer, en tidlig variant av PhysX (fysikkmotor), lydhåndtering og scripting i C#. To år senere kom 2.0 som brakte med seg en terrenggenerator, nettverksstøtte, dynamiske skygger i sanntid og et system for å bygge brukergrensesnitt. I oktober 2008 kom det støtte for å publisere spill for iPhone og et par måneder senere kom det støtte for å lage spill for Windows. Unity ble da virkelig en spillmotor for kryssplattformutvikling.

Oktober 2010 kom Unity 3.0. Med denne versjonen kom det mer avanserte verktøy med støtte for lightmapping, occlusion culling, lydbehandling i sanntid og support for en nyere variant av C#. I 2010 kom Unity Asset Store, et online marked der brukere av Unity kan selge alt av resurser som modeller, scripts, audio og så videre. Senere kom det også et partikkelsystem, stifinning, level-of-detail-støtte for 3D-objekter, HDR og støtte for å lage spill for Android-plattformen.

Unity 4 ble lansert i 2012 og brakte støtte for DirectX 11, Linux, Windows Phone 8, OpenGL ES 3.0 og en rekke andre forbedringer. I tillegg kom “Mecanim”, et avansert animeringssystem. [7]

3 Mars 2015 kom Unity 5.0. Denne versjonen gikk bort fra den tidligere modellen med å ha en gratisvariant og en profesjonell variant med bredere støtte og gjorde hele motoren tilgjengelig for begge varianter. Unity 5 har den bredeste støtten for ulike plattformer, med støtte for blant annet Windows, OS X, Linux, Android, iOS, BlackBerry 10, Windows Phone, WebGL, Playstation 3/4/Vita, Wii U, New 3DS, Xbox 360/One, Oculus Rift og Gear VR.

Unity baserer seg på å kombinere bruken av programmering og drag-and-drop metodikk for å lage “scener”, en samling av miljø, modeller, audio og så videre. Alt som går inn i scenen er definert som et GameObject". Et GameObject kan ha et svært variert spekter av egenskaper

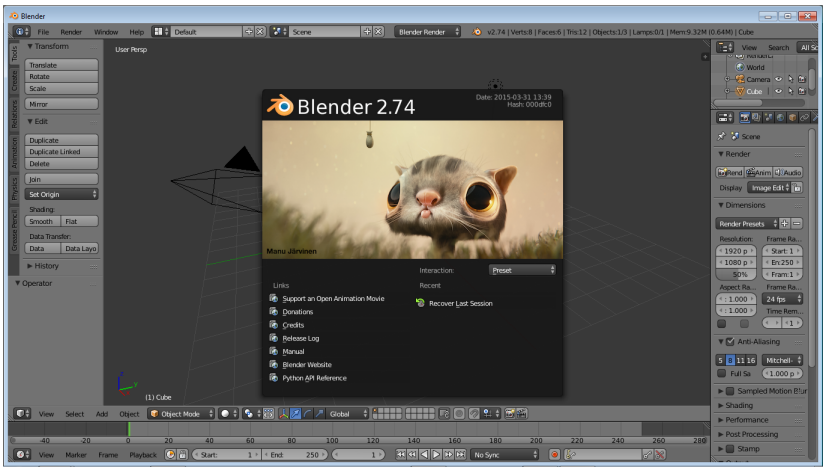


Figur 3.13: Unity 5

assosiert ved seg. For eksempel kan et GameObject ha en transform (posisjon, rotasjon og skala), modell, tekstur, script, animasjoner og så videre. Det er også mulig å kombinere de fleste av disse egenskapene i tillegg til at et GameObject kan ha sine egne GameObjects som underobjekter. Man kan også lagre denne konfigurasjonen av objekter, underobjekter og egenskaper som et forhåndslaget objekt, en prefab. Dette åpner for å lage svært avanserte objekter med mye funksjonalitet på en enkel måte. Et eksempel vil være å lage fiendesoldater i et spill. Man kan lage en prefab av en soldat som inneholder selve modellen til soldaten, animasjoner, teksturer og annet som måtte være nødvendig. Soldat-objektet kan ha et eller flere scripts som styrer logikken til objektet. Dette forhåndslagde objektet kan så instansieres enten i editoren eller via scripts.

3.10 Blender

Blender er et gratis modelleringssoftware, ofte brukt for å lage modeller og animasjoner men også brukt til mer avansert 3D-rendering og videoredigering. Programmet startet som et internt program for det nederlandske animasjonsstudioet Neo Geo og Not a Number Technologies [6] men ble gjort open-source i 2002. Blender har en renderer, verktøy for en rekke animasjonsformer, scripting, teksturmapping og en rekke andre viktige verktøy.



Figur 3.14: Blenders velkomstbilde. [6]

3.11 Brukertesting

Brukertesting for følelse av innlevelse er vanskelig å utføre. Et par alternativ er presentert i “Measuring and Managing Presence in Virtual Environments”. [33]. Av dem er en subjektiv undersøkelse rundt prototypen enkelt gjennomførbart. En feilkilde i denne formen for brukertesting er at brukerene som tar testen må ha det klart for seg hva “innlevelse” og “deltagelse” er, og denne definisjonen bær være lik for alle brukerene. Spørsmålene kan være påstander som “Jeg følte en stor grad av innlevelse” med fem svaralternativer gradert fra Meget Uenig, via Uenig, Nøytral og Enig til Meget Enig.

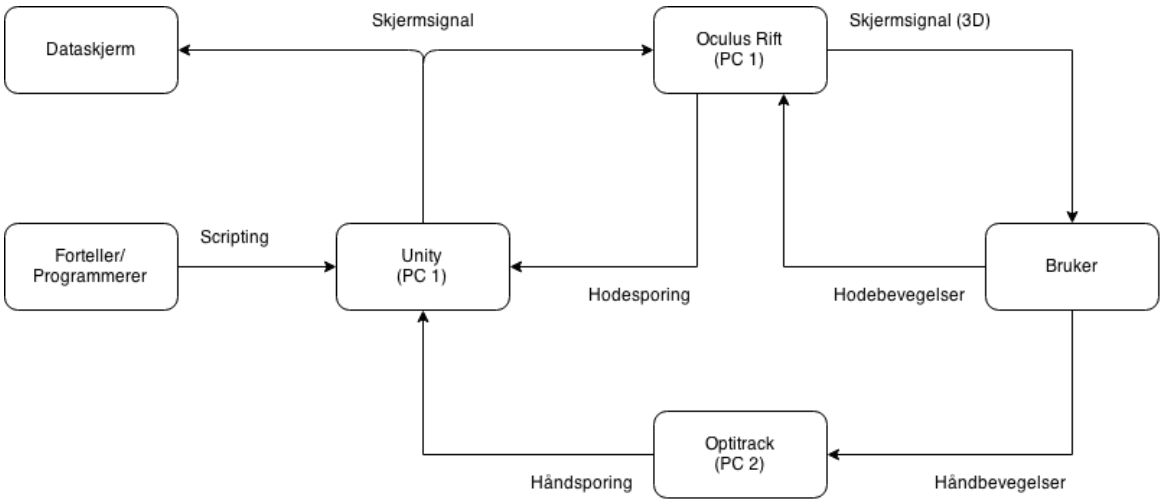
Kapittel 4

Virtuelt legekontor

4.1 Formål

Bruken av nye digitale fortellerverktøy åpner for nye muligheter for interaksjon med digitale fortellinger. Prototypen Virtuelt legekontor ble utviklet med to formål; utforske bruken av Unity som et fortellerverktøy og hvordan bruken av Oculus Rift og bevegelsessporing oppleves av en bruker av fortellingen.

4.2 Oversikt



Figur 4.1: Flytkart for informasjonsstrømmen i det virutelle legekontoret

Det virtuelle legekontoret er en kort fortelling laget i Unity. Figuren ovenfor 4.1 er en visuell oversikt over de ulike elementene som inngår i prototypen samt informasjonsflyten mellom dem. To datamaskiner ble brukt for å kunne håndtere både kroppssporing og generering av det digitale miljøet. Brukeren sender inn informasjon til Oculus Riften via hodebevegelser og informasjon til sporingssystemet ved håndbevegelser. Hodebevegelsene registreres av Oculus Rift som så sender bevegelsene til Unity. En plugin i Unity håndterer informasjonen slik at de to virtuelle kameraene som representerer brukerens øyne i det digitale miljøet beveger seg på tilsvarende måte. Unity sender i retur bildene fra de to kameraene til Oculus Rift slik at brukeren opplever miljøet i 3D. Brukerens håndbevegelser registreres av Optitrack-systemet koblet til en annen datamaskin. Denne maskinen beregner posisjon og rotasjon til brukerens hender og strømmer data over nettverket til Unity. En plugin i Unity tar inn posisjoneringsdata fra nettverket og påvirker det virtuelle miljøet basert på scripting.

4.3 Utførelse

I masteroppgaven var formålet å utforske bruken av et fortellerverktøy, Oculus Rift og bevegelsessporing. Mulige alternativer for bevegelsessporing inkluderte Kinect og Optitrack. Optitrack ble valgt på grunn av at det ikke var nødvendig for oppgaven å spore hele kroppen og at det var ønskelig med en stor grad av presisjon i sporingen. Oculus Rift har støtte via utviklingssett for to spillmotorer, Unity og Unreal Engine. Unity ble valgt grunnet erfaring med bruken av spillmotoren og muligheten for gjenbruk av resurser. Den betalte varianten var nødvendig for å kunne bruke biblioteker for å ta i bruk Oculus Rift og Optitrack.

4.3.1 Unity Pro

I en tidligere rapport "Digital Storytelling Tools" [5] ble det bygget en prototype for å undersøke visse aspekter ved bruken av VR. Denne prototypen ble basis for videre utvikling av en ny prototype. Ved å bruke både betalte og gratis modeller fra Unitys "Asset Store" [4] var det mulig å lage et legekontor ved å plassere rekvisita i et rektangulært rom. De fleste av modellene i scenen ble plassert for å gjøre legekontoret mer troverdig og hadde ingen annen betydning for programmet. Et overblikksbilde av scenen er vist i figur 4.2.

Figur 4.2: Legekontoret i Unity

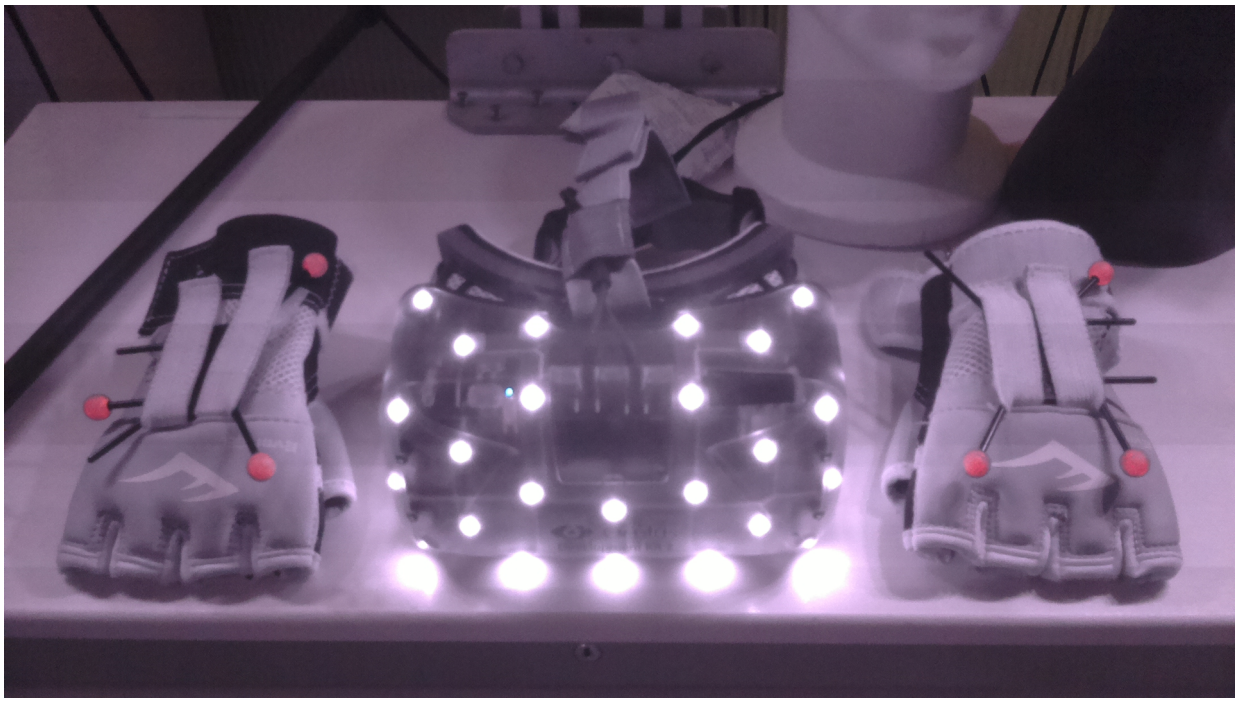
I oppgaven var det hensiktsmessig å gjengi bevegelsene til brukeren så godt som mulig i bevegelsene til brukerens avatar. I planleggingsfasen ble det vurdert å utstyre brukeren med sporingsobjekter for hender, albuer og brystkasse og så oppdatere avataren til brukeren basert på sporingsinformasjonen fra disse objektene. Dette ville både krevd en del fysisk utstyr og virket også komplekst å implementere i Unity. Ved å utforske dokumentasjonen til Unity sitt animeringssystem, Mecanim [7] ble det funnet en mulighet for å utnytte en innebygget funksjonalitet for Inverse Kinematics. Ved å bruke Inverse Kinematics var det mulig å animere avataren til brukeren på en rimelig nøyaktig måte. IK virker ved å gi Unity en posisjon og rotasjon hånden skal være i og Unity beregner hvordan underarm, albue og overarm må være posisjonert for at det skal være mulig innenfor et sett med begrensninger for leddene i skulder og albue. Siden IK finner en "naturlig" løsning i de fleste tilfeller ble dette alternativet valgt for å redusere kompleksiteten i implementasjonen.

4.3.2 Optitrack

Figur 4.3: Oppsett av Optitrack-systemet

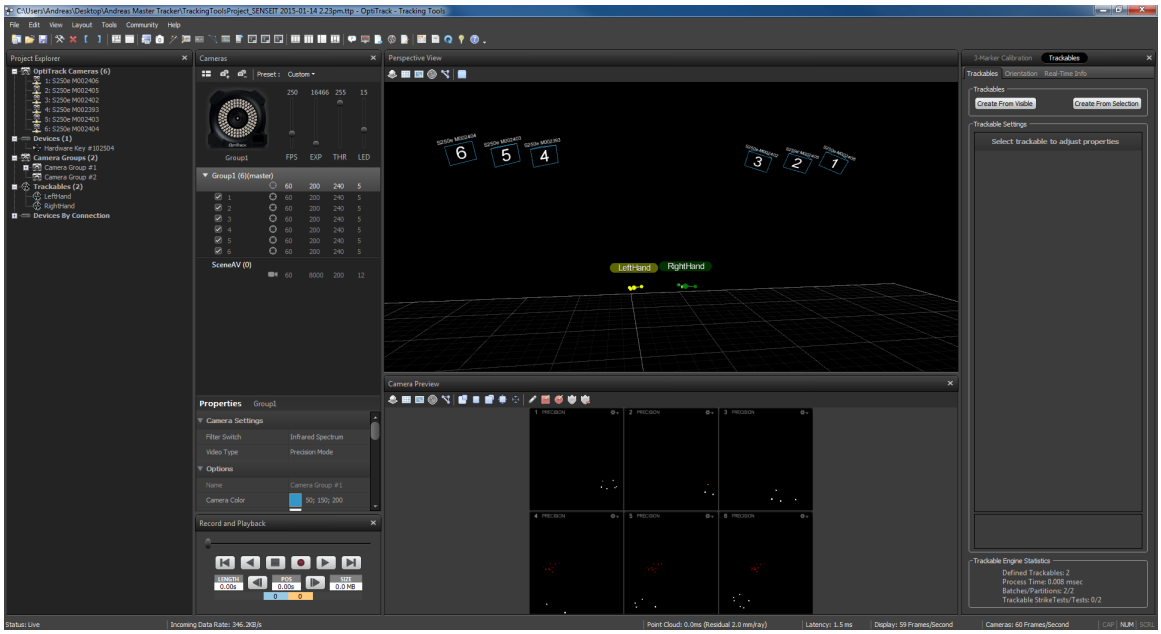
For å ha mulighet for en svært nøyaktig sporing av bevegelser ble Optitrack valgt til fordel for Kinect. I oppgaven ble det brukt en eksisterende rigg, vist ovenfor i figur 4.3. Seks kameraer plassert på en linje sender bildedata til en datamaskin, som bruker bildene for å beregne posisjon og rotasjon til brukerdefinerte objekter. I oppgaven ble kameraene satt til å kun se refleksjoner i det infrarøde spekteret og små kuler av et retro-reflekterende materiale ble festet til to plater. 4.4,4.5 Programvaren knyttet til Optitrack har mulighet for å definere vilkårlige mønstre av refleksjoner slik at det var mulig å definere de to platene som høyre og venstre hånd. Disse to platene ble så festet til hver sin hanske slik at det var mulig å spore håndbevegelser på en naturlig måte.

Figur 4.4: Hansker for sporingsobjekter og Oculus Rift



Figur 4.5: Hansker for sporingsobjekter og Oculus Rift -Infrarødt-

Programvaren til Optitrack, illustrert i figur 4.6, har mulighet til å strømme posisjoneringsdata i et utvalg av formater. Ved å utforske hva som var tilgjengelig av plugins for Unity falt valget på et åpent format, VRPN (Virtual-Reality Peripheral Network) [3]. Unity AR Toolkit (UART) [2] ble valgt for å gjøre implementasjonen av sporing så enkel som mulig men det viste seg at programvaren var både utdatert og vanskelig å bruke. Pluginen og programvaren ble brukt som en base for en modifisert plugin som var mer tilpasset oppgaven.



Figur 4.6: Skjermbilde fra Optitrack

Pluginene ble skrevet i C# og biblioteket ble skrevet i C++. Pluginene bestod av to C#-scrips, ”VrpnManager.cs” og ”trackerInstance.cs”. ”VrpnManager.cs” initialiserer biblioteket, starter en server som lytter til en gitt nettadresse og håndterer oppretting av nye sporingsobjekter. Scriptet ”trackerInstance.cs” legges til objektene som skal motta sporingsdata og tar inn informasjon om hvilket sporingsobjekt det skal hente data fra på formen ”objektnavn”@”nettverksadresse”. Pluginen sørger for at posisjon og rotasjon til objektet scriptet er koblet til blir oppdatert etter som ny sporingsdata kommer inn til biblioteket. Funksjonen for å oppdatere posisjonen blir kalt hver gang Unity lager et nytt bilde.

4.3.3 Oculus Rift

Oculus VR har implementert pluginer for bruk i Unity, som gjorde det svært enkelt å bruke Oculus Rift med Unity. Et sett med resurser ble lastet ned fra Oculus VR og importert inn i Unity. En prefab (prefabrikkert objekt) kalt “OVRCameraRig” ble plassert inn i scenen 4.7. Et prefabrikkert objekt er et objekt som har et ferdig sammensatt sett med egenskaper slik som scripts, “barn” og lignende. Ved å referere til OVRCameraRig i scriptet som kontrollerer scenen var det mulig å kontrollere hvor kameraet var plassert.

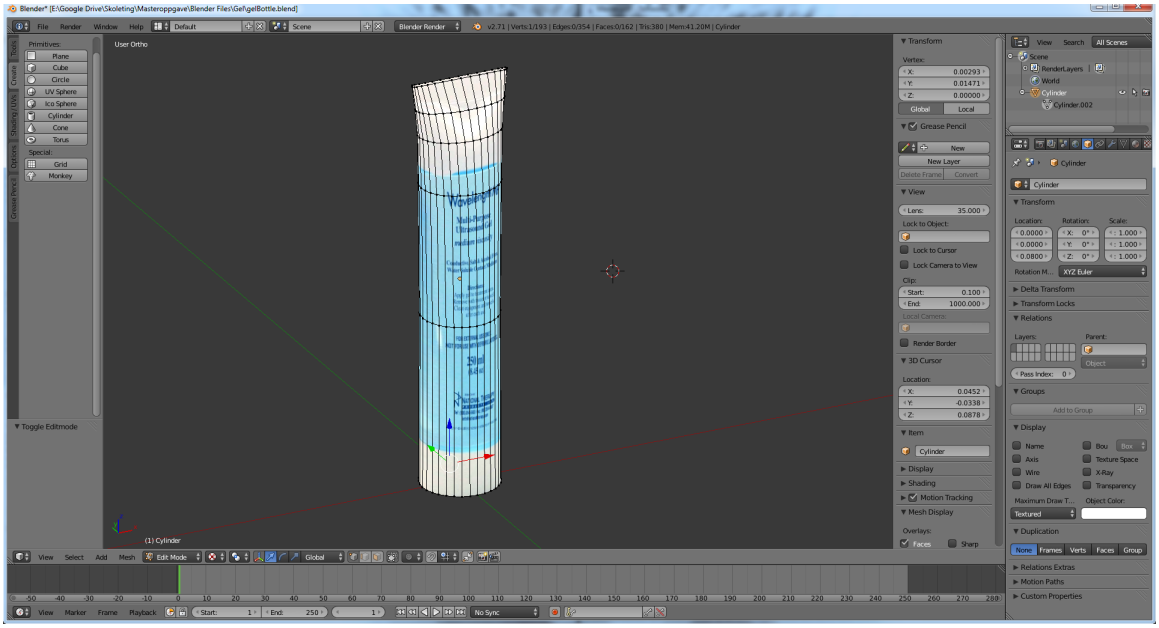
Figur 4.7: Plassering av OVRCameraRig i Unity

4.3.4 Blender

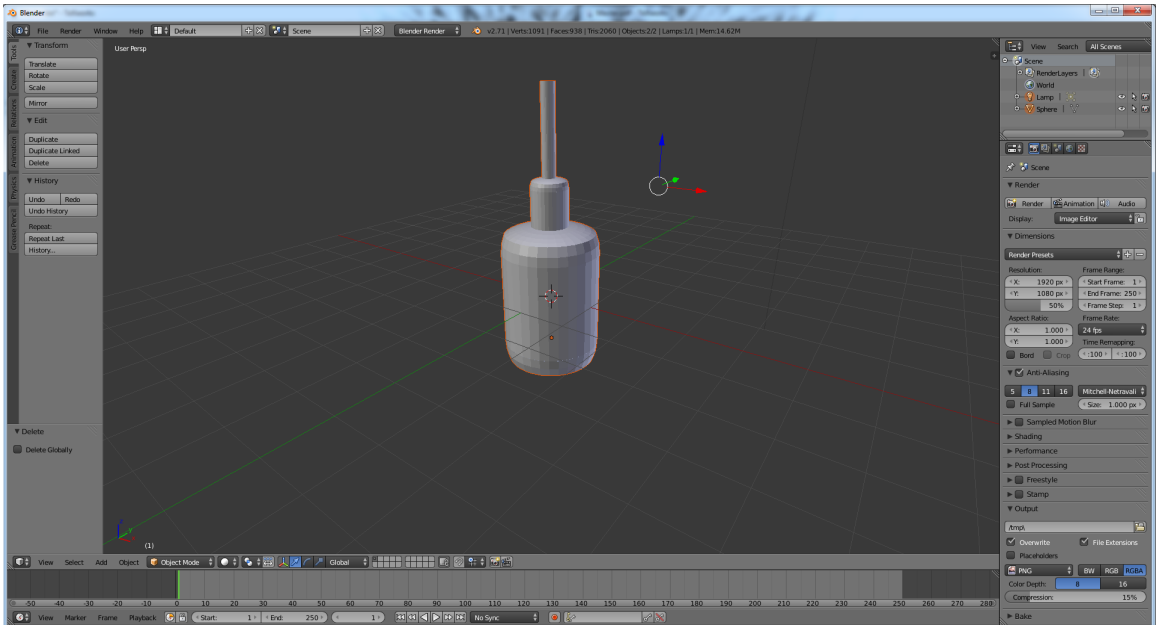
Blender ble brukt for et par viktige oppgaver. For å kunne ta i bruk Mecanim-systemet til Unity var det nødvendig å modifisere modellene for legen og pasienten ved å legge til et “skjelett” til modellen. Alle beinene i modellen ble plassert på riktig sted, med beinet i korsryggen som en rot for alle de andre. Beinplassering er illustrert i figur 4.8 Ved å eksportere modellene til FBX-formatet var det mulig å importere det i Unity, der Unity automatisk tilegnet vekter for hvordan beinene påvirker modellen etter som beinene beveger seg.

Figur 4.8: Legens skjelett i Blender

Den andre viktige oppgaven var å opprette modeller som det var nødvendige for oppgaven, men som ikke det var mulig å få tak i på Unitys Asset Store. Modeller for tuben med gel 4.9 og ultralyd-transduceren 4.10 ble opprettet ved å kombinere enkle former som sylindre og halvkuler.



Figur 4.9: Tube med gel i Blender



Figur 4.10: Ultralydtransducer laget i Blender

4.4 Virtuelt Legekontor

For å styre selve historien ble det benyttet et script, “ScenarioScript.cs”. Scriptet i all sin helhet kan sees i vedlegget. Dette hovedscriptet kontrollerte all framdrift i historien til prototypen:

1. Brukeren kan bevege seg fritt rundt og plukke opp objekter i den initielle fasen av historien. Denne biten er ment for å la brukeren forstå hvordan bevegelsessporingen fungerer og la brukeren orientere seg i det virtuelle miljøet. Scriptet beveger seg videre til neste del når mellomromstasten blir trykket ned. I det tasten blir trykket ned starter opptaket av bevegelsene til brukeren.

2. Historien begynner. I denne fasen må brukeren plukke opp tuben med ultralyd-gel og føre denne over magen til pasienten. Bruker brukeren for lang tid, over 15 sekunder, vil pasienten reagere med dialog. Pasienten vil også reagere hvis brukeren tar transduceren over magen før det har blitt påført gel.

3. Brukeren blir presentert med to dialogvalg. Dialogvalg velges ved å se på valget i 2 sekunder. Pasienten reagerer med dialog.

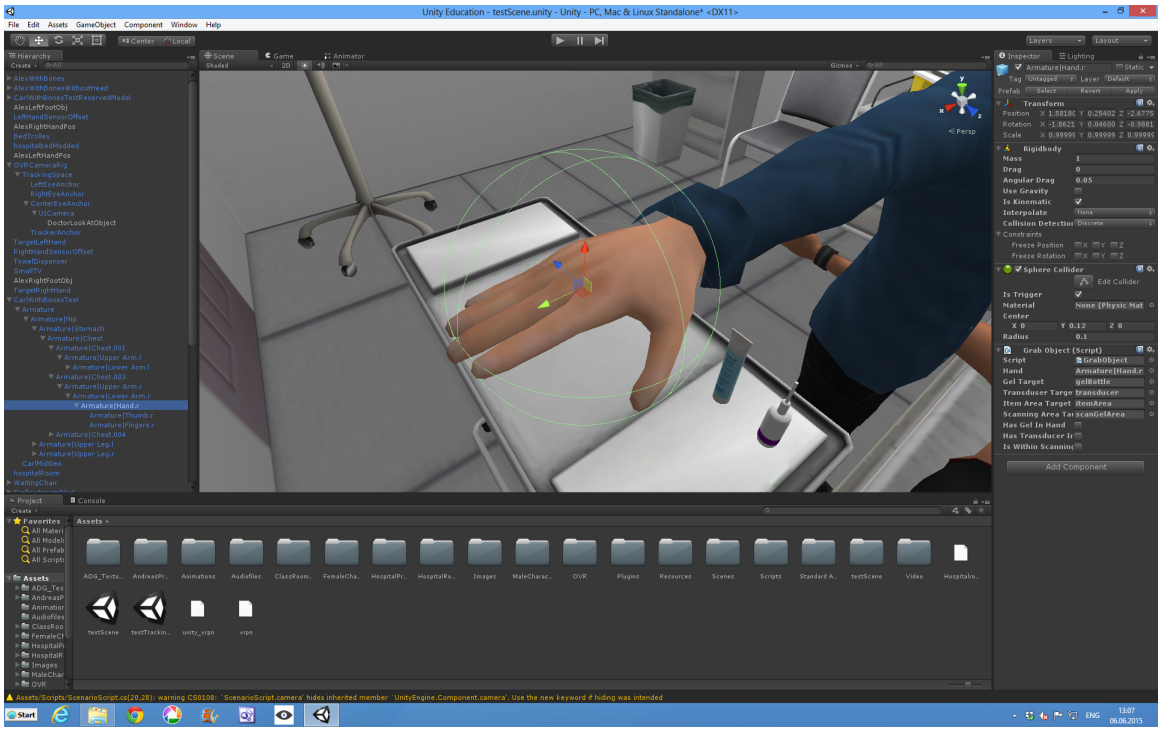
4. Brukeren må plukke opp og føre ultralydtransduceren over magen til pasienten. Når transduceren er i området over magen vises en repeterende film på monitoren til venstre for legen. Pasienten reagerer ved å se mot monitoren som viser filmen. Brukeren må holde transduceren på riktig sted i 7 sekunder i strekk.

5. Avsluttende fase. Dialog fra legen kommer opp på veggen foran legen som impliserer at brukeren må legge fra seg transduceren før historien er ferdig. Når brukeren har lagt fra seg alt brukeren måtte holde slutter opptaket av bevegelsene til brukeren.

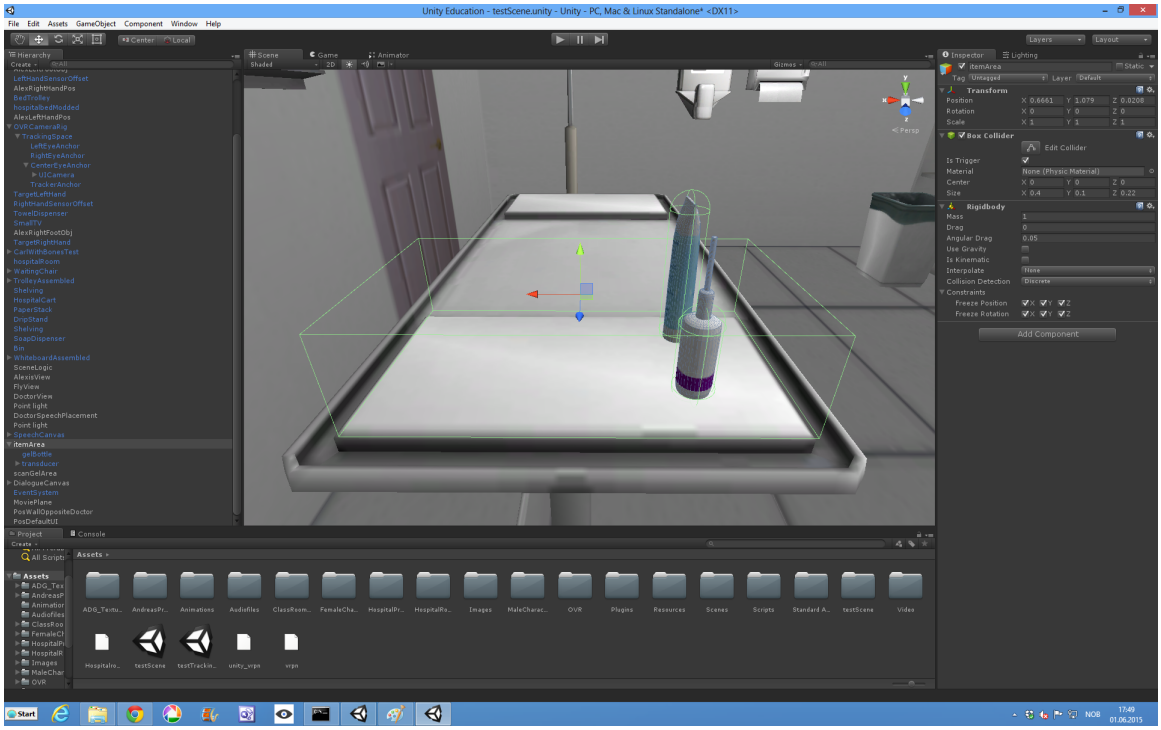
6. Når mellomromstasten blir trykket, skifter historien perspektiv til pasientens synspunkt. Modellen til legen blir byttet ut med en med hode og modellen til pasienten blir byttet ut med en uten hode. Nå blir alle bevegelser og dialogvalg som brukeren gjorde spilt tilbake for brukeren.

4.4.1 Interaksjonspunkter

To interaksjonspunkter ble valgt for å drive historien framover. Den første interaksjonen var i form av å gripe og føre objekter i rekkefølge. Dette ble valgt ut i fra konteksten til historien og grunnet at handlingen i seg selv var veldig enkel å forklare. For å implementere denne funksjonaliteten i Unity ble både høyre og venstre hånd i skjelettet til avataren utstyr med en kollisjonssone og et script for å håndtere griping og slipping av objekter. Kollisjonsområdet er markert i figur 4.11 Ved å markere for at kollisjonssonen var en "triggersone" i inspektørvinduet i Unity var det mulig å lage et script som ble kalt hver gang sonen gikk inn i en annen kollisjonssone. Ved å gi objekter kollisjonssoner var det mulig å gjøre hånden til "parent" av objektet slik at hånden da kunne påvirke posisjonen til objektet. Videre ble det definert to statiske soner, "Item Area" 4.12 og "Scanning Area" 4.13. Hvis en hånd kolliderte med Item Area og i tillegg hadde et objekt i hånden, ville objektet bli "parented" til Item Area slik at det var mulig å legge fra seg objektet. "Scanning Area"-objektet ble brukt av scriptet som styrte handlingen.



Figur 4.11: Kollisjonsområder for doktorens hender

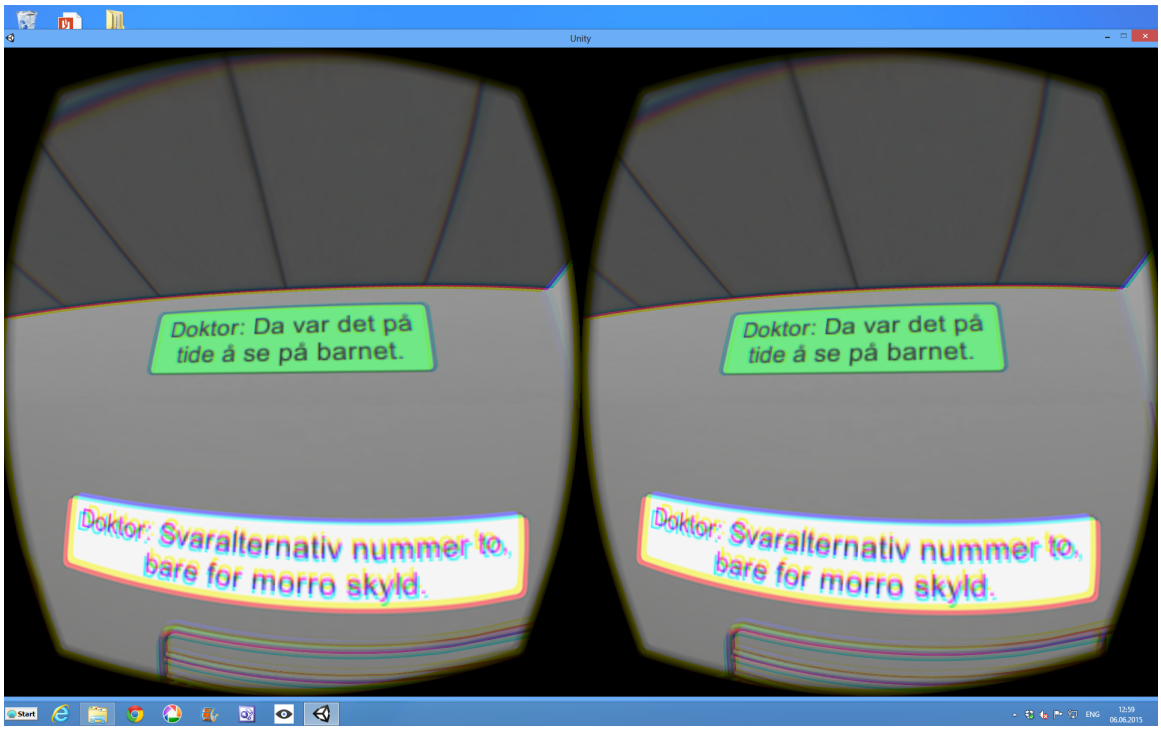


Figur 4.12: Kollisjonsområder for gel, transducer og målområde for nevnte objekter

Figur 4.13: Scanneområde

Det andre punktet var et enkelt dialogvalg. Det ble vurdert flere løsninger for hvordan brukeren kunne velge et alternativ, blant annet å ta på alternativet med fysiske bevegelser. For å ha ulike interaksjonspunkter falt valget på at brukeren kunne se på dialogalternativet brukeren ønsket å velge. Dette ble gjort ved å plassere to objekter i i scenen, et canvas-objekt og et “EventSystem”-

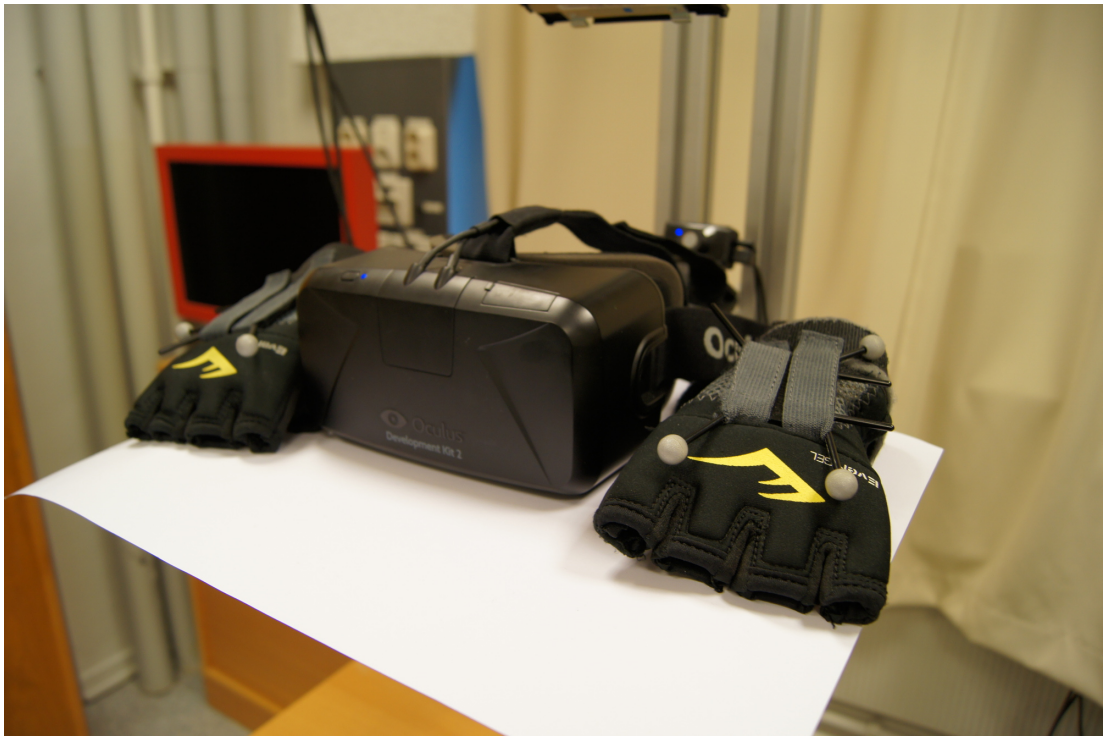
objekt. Canvas-objektet gjorde det mulig å plassere knapper med variabel tekst i scenen. For å gjøre det mulig å velge knapper ble et dummy-kamera lagt til Oculus-riggen. Et script ble lagt til EventSystem-objektet for å håndtere hva som skulle skje når dummy-kameraet så på knappene i canvas-objektet. Knappene selv var spesielle Unity-objekter som hadde mulighet for å gjøre en handling basert på en definert trigger. Et script ble lagt til canvas-objektet som oppdaterte et sett variabler slik at det var mulig å hente ut hvilken knapp som brukeren så på og når brukeren begynte å se på knappen. Disse variablene ble lest og brukt av hovedscriptet. En illustrasjon på hvordan knappene forandret farge når de ble sett på er i figur 4.14.



Figur 4.14: Brukerenes to dialogvalg

Kapittel 5

Brukertesting



Figur 5.1: Klar for brukertest

5.1 Utførelse

Brukertestene ble utført i perioden 29. Mai til 02. Juni på 21 personer. I løpet av brukertestene ble det klart at det var flere små problemer med prototypen men disse problemene var ikke så store at det var behov for å forandre på prototypen i løpet av testene.

1. Testperson blir ønsket velkommen.
2. Testperson får en muntlig beskrivelse av hva de skal oppleve og hvilke oppgaver de skal utføre.
3. Bruker tar på seg hansker og Oculus Rift og blir plassert på riktig plass.
4. Kalibreringsrutine blir gjennomført.
5. Bruker får tid til å gjøre seg vant med utstyret.
6. Testinstruktør starter historien.
7. Bruker utfører oppgavene med muntlig hjelp fra testinstruktør.
8. Bruker fullfører oppgavene og testinstruktør starter tilbakespilling av brukerens bevegelser.
9. Bruker fyller ut spørreskjema på pc.

5.1.1 Kalibreringsrutine

For å sørge for at bevegelsene til brukeren ble gjengitt så nøyaktig som mulig var det viktig å finne den mest optimale posisjonen for brukeren å stå. Brukeren ble plassert foran sporingsriggen, med kropp og ansikt rett fram mot riggen. Brukeren ble bedt om å gå litt bakover, strekke armene rett fram og så gå sakte framover. Testinstruktør fulgte med på speilbildet av brukerens synspunkt på en ekstern skjerm og ba brukeren om å stoppe når avatarens armer nådde en utstrakt posisjon. Testinstruktøren tilbakestilte posisjonen til det virtuelle kameraet slik at brukeren hadde riktig perspektiv. Etter dette ble brukeren bedt om å bevege rundt på hendene og brukeren ble spurt om brukeren følte at bevegelsene ble gjengitt på en realistisk måte. Hvis svaret var negativt ble rutinen gjort om igjen.

5.2 Resultater

Bra opplegg med mykje potensiale!

God oppgave med store muligheter i fremtiden

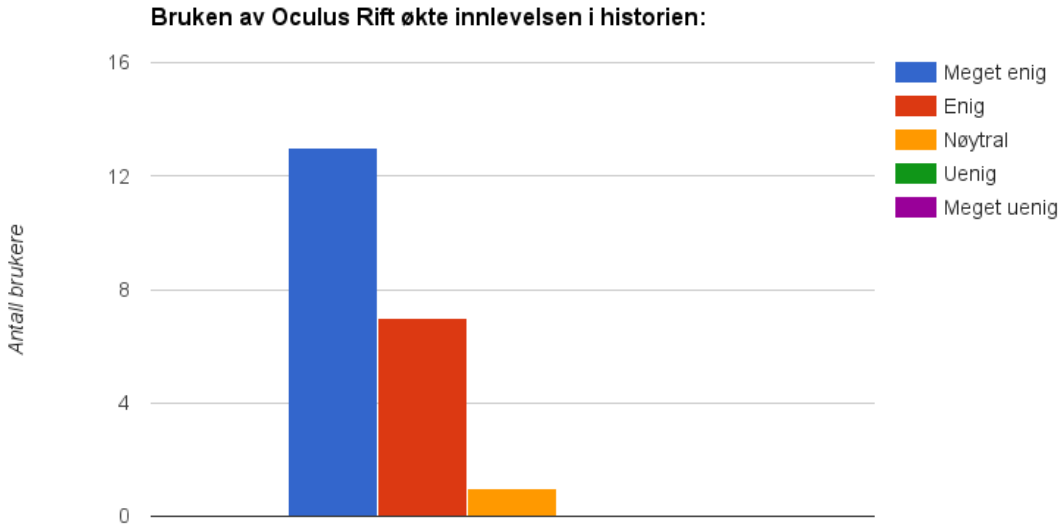
The future is here!

Kjempekult! Tror historiefortelling kan bli noe helt nytt, særlig for de som har litt problemer med å sitte stille og sliter med å følge med

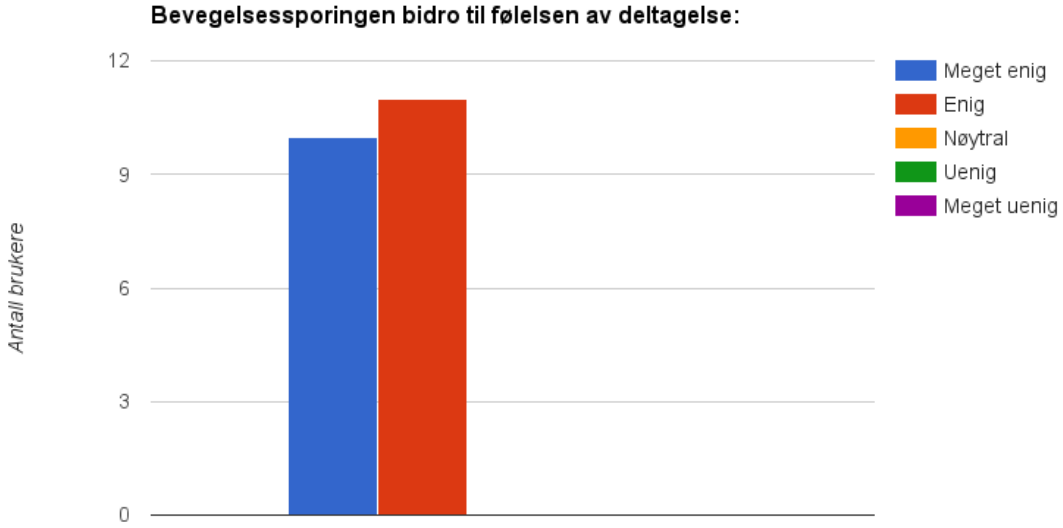
Ble litt bilsyk kvalm.

Det var veldig gøy!

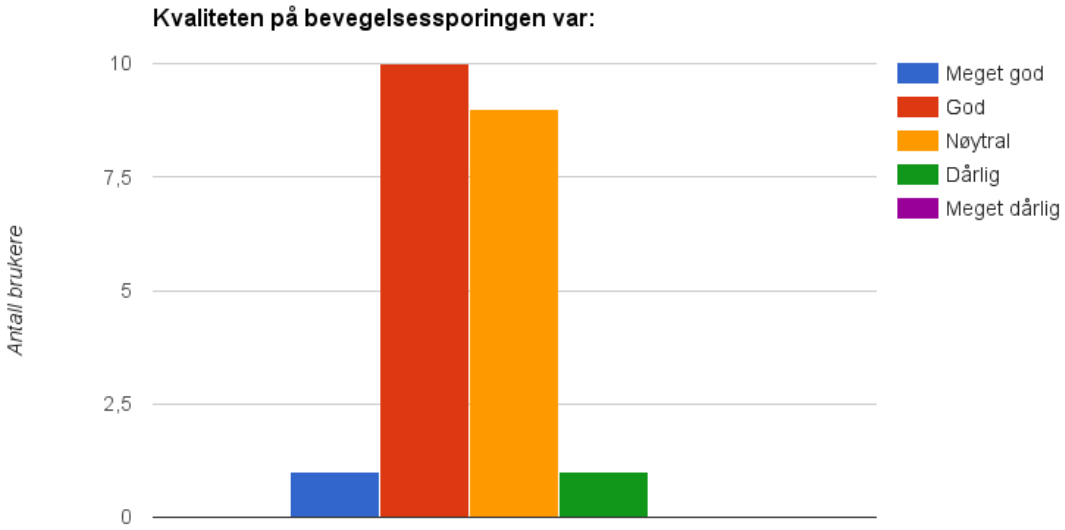
Følgende er de viktigste resultatene av brukertesten. Den komplette listen med svar og kommentarer er lagt ved i vedlegget.



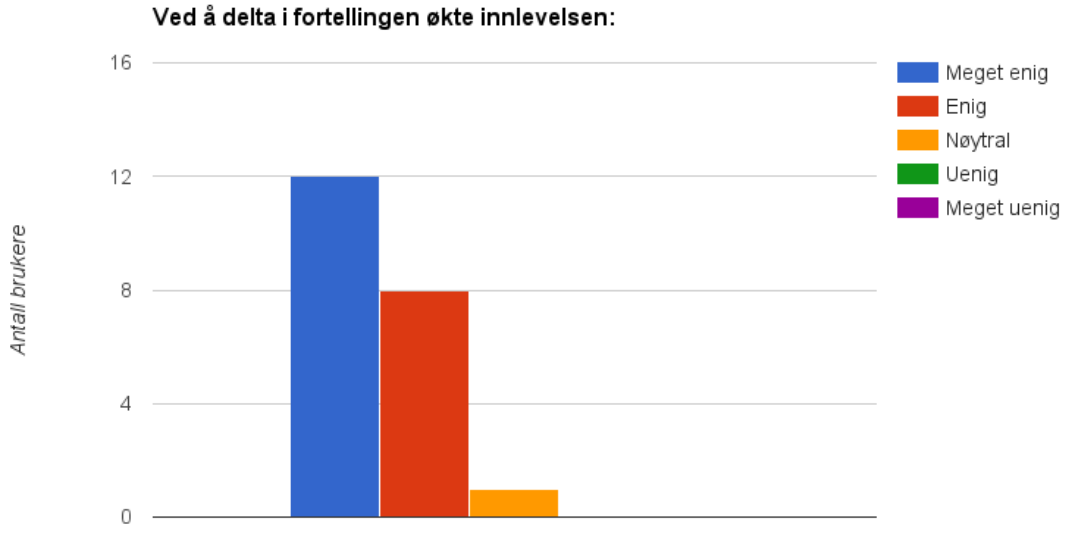
Figur 5.2: Spørsmål om bruken av Oculus Rift



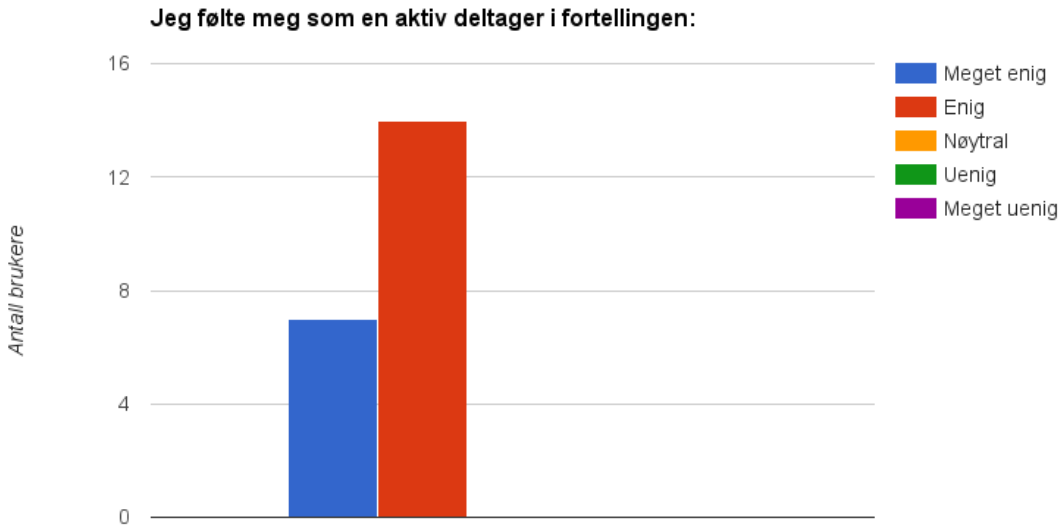
Figur 5.3: Spørsmål om bevegelsessporing og deltagelse



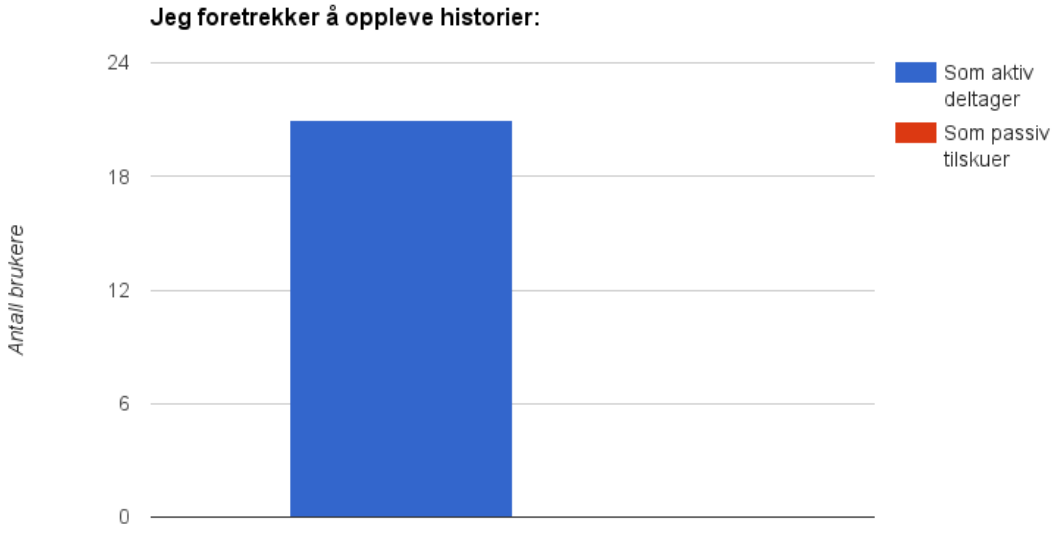
Figur 5.4: Spørsmål om kvaliteten på bevegelsessporingen



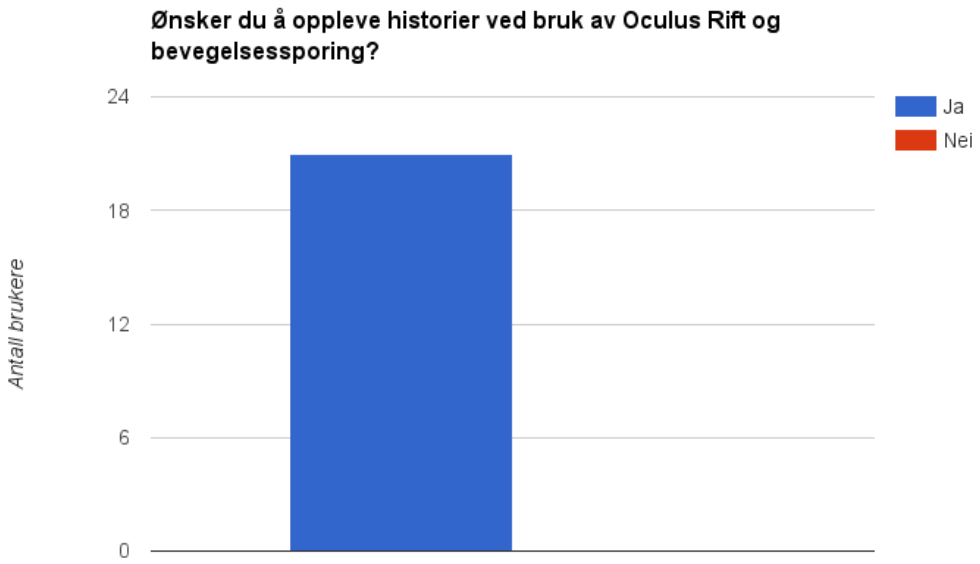
Figur 5.5: Spørsmål rundt deltagelse og innlevelse



Figur 5.6: Spørsmål om brukeren følte seg som en aktiv deltager



Figur 5.7: Spørsmål om hvordan brukeren ønsker å oppleve fortellinger



Figur 5.8: Spørsmål om brukeren ønsker å oppleve historier med digitale historiefortellingsverktøy

Kapittel 6

Drøfting

6.1 Unity som fortellerverktøy

Gjennom utviklingen av prototypen ble det klart at Unity var et svært nyttig og brukbart verktøy. Ved bruken av det grafiske grensesnittet og scripting var det mulig å få til mye til tross for lite resurser og få personer. Unity er også godt dokumentert selv om det i enkelte tilfeller kan være vanskelig å finne det man trenger. Bruken av en spillmotor i seg selv for historiefortelling gir store muligheter for ulineære historier og gjør det enkelt å benytte seg av eksterne verktøy.

Her følger er en oversikt over viktige fordeler og bakdeler ved å bruke Unity som et verktøy for historiefortelling:

- **Scripting** Spesielt nyttig for oppgaven var bruken av Unitys scriptingsystem. I oppgaven ble det valgt å bruke C#, men det er i tillegg mulig å bruke UnityScript (JavaScript). Ved hjelp av disse scriptene var det mulig å legge til alt av den nødvendige logikken som historien trengte. Muligheten for å bruke scripting sammen med et svært stort og brukbart API gjør at det er enkelt å oppnå avanserte situasjoner og samhandlinger og gjør at det i stor grad er mulig for en bruker å ha interaksjon med programvaren. Ved å bruke plugins og ekstern kode er det mulig å ta i bruk svært mange digitale verktøy.
- **Skalering** Unity er svært skalerbart og passer for små og store prosjekter. Unity tilbyr også bruk av programvare ment for å gjøre det enklere for et team av personer å samarbeide. Begrensningen i Unity ligger i hvor kraftig maskinvare programmet skal kjøre på til forskjell fra hva som er teknisk mulig å få til.
- **Asset Store** Når historier utvikles er det ofte greit å ha tilgang på et bredt spekter av modeller og funksjonalitet. Ved å bruke Unitys Asset store er det muligheter for å spare utviklingstid og penger ved å bruke betalte eller gratis resurser der det er mulig.
- **Eksterne verktøy** Ved bruk av plugins, scripting og ekstern kode åpner det for bruk av svært mange eksterne verktøy slik som Oculus Rift, Optitrack, Kinect og så videre. Dette gir da historiefortellere mange valg i hvordan de ønsker at brukeren skal interaktere med historien.
- **Kompleksitet** På grunn av at Unity er ment til å ha et svært stort spekter av muligheter så er det mye å sette seg inn i. Til tross for mye dokumentasjon og flere opplæringsvideoer er det også enkelte tilfeller der funksjonalitet ikke er godt nok forklart. Et eksempel på dette var at det var lite informasjon rundt spesielle regler når det gjelder interaksjon mellom kollisjonsobjekter.

6.2 Oculus Rift og bevegelsessporing

I brukertesten ble det spurt flere spørsmål rundt brukerens opplevelse av bruken av Oculus Rift og bevegelsessporing. Som vist i figur 5.2 var de fleste brukerene meget enig (13) til enig (7) i at det å ha på seg Oculus Rift økte innlevelsen i historien. Kun en person var nøytral til påstanden. Noe som er interessant å se på er at selv om det var mange som mente at kvaliteten til bevegelsessporingen var Nøytral (8) til God(10) som vist i 5.4 mente svært mange at bevegelsessporingen økte innlevelsen og deltagelsen i historien. Til tross for at brukerene ofte måtte tilpasse seg hvordan systemet fungerte er det klart at de fleste følte at bevegelsessporingen økte følelsen av tilstedeværelse ut ifra svarene som vist i figur 5.3. Alle brukerene var Meget Enig (10) eller Enig (11) i påstanden om at bevegelsessporingen bidro til følelsen av deltagelse.

I brukertesten ble det spurt flere spørsmål rundt brukerens opplevelse av bruken av Oculus Rift og bevegelsessporing. Som vist i figur 5.5 var de fleste av brukerene enig(8) til meget enig(12) i at ved å delta i handlingen økte følelsen av innlevelse. Kun en var nøytral og ingen var uenig i påstanden. Dette viser sammen med 5.6 at ved å bruke digitale fortellerverktøy slik at brukeren tar en aktiv del i handlingen så øker følelsen av tilstedeværelse og innlevelse. Alle brukerene oppgav at de foretrakk å oppleve historier som en aktiv deltager. 5.7 Spesielt interessant er svarene brukerene oppgav når de ble spurt om de ønsket å oppleve historier ved bruk av Oculus Rift og bevegelsessporing som vist i figur 5.8. Svaret var et enstemmig ja, som vitner om at det er ønske om å bruke disse verktøyene for å oppleve handlinger på nye måter. Dette kom også fram av kommentarer fra brukerene i løpet av testen og kommentarer skrevet i svarskjemaet.

6.3 Feilkilder til brukertesten

- **Unity** Unity er en avansert programvare med svært mange muligheter og systemer som skal virke sammen. Selv om det gjør Unity til et svært nyttig verktøy betyr det også at det er muligheter for at uforutsette feil kan oppstå. Under brukertestene sluttet Unity å fungere to ganger.
- **Oculus Rift** Oculus Rift DK 2 er et utviklerverktøy og ikke ment for bruk av vanlige konsumere. Brillen har en lavere oppløsning enn nødvendig for å ikke legge merke til en nettingeffekt og posisjonssporingen slutter å fungere hvis brukeren beveger seg utenfor synsfeltet til kameraet. Hver bruker vil også ha ulik avstand mellom pupillene slik at individuell tilpassing er nødvendig for å få en korrekt bilde av brillene, en prosess som er upraktisk å gjøre for hver ny bruker.
- **Optitrack** Optitrack fungerer svært bra i de fleste tilfeller men har en del muligheter for feil og mangler. Den største feilkilden ved måten Optitrack-systemet var satt opp under brukertestene var at det var mulig å holde hendene i visse posisjoner og vinkler slik at ingen av kameraene hadde syn til sporingsobjektene. Siden kameraene var satt opp på en linje foran brukeren ville systemet miste sporingen hvis brukeren holdt hendene med håndbaken opp eller hvis brukeren holdt hånden til høyre eller venstre side med håndbaken mot kameraene. Dette problemet var spesielt til stede under brukertestene når brukeren skulle gripe et av objektene på sin høyre side. De fleste av brukerene fant fort ut av hvordan de var nødt til å holde hånden for at sporingen skulle fungere som normalt men det er verdt å vurdere om dette kan ha hatt en innvirkning på hvor realistisk brukerene følte at sporingen var. I andre tilfeller kunne systemet plukke opp refleksjoner fra brukerens klokker og klær slik at sporingen sluttet å fungere.
- **Virtuelt Legekontor** Scenen “Virtuelt Legekontor” var et avansert system satt sammen av flere scripts. Dette gjorde at det var muligheter for uforutsette feil å oppstå. Enkelte kombinasjoner av handlinger og tilstander gjorde at det ikke var mulig å fortsette videre med testen. En av brukerene prøvde to ganger uten å lyktes med å fullføre historien.

Kapittel 7

Konklusjon

Hovedtemaene i oppgaven var hvordan en spillmotor som Unity kan brukes sammen med Oculus Rift og kroppssporing som digitale fortellerverktøy. Ved å utvikle en kort historie i Unity som tar i bruk disse verktøyene og så kjøre prototypen gjennom brukertester har oppgaven prøvd å svare på disse spørsmålene.

7.1 Unity

Unity viste seg å være et svært anvendbart verktøy med mange nyttige funksjoner og et stort mangfold av resurser. Spesielt var det muligheten for å scripte med et godt API som gjorde verktøyet nyttig som et digitalt fortellerverktøy. Digitale fortellinger lages ved å kombinere ulike former for digitale media. Unity kan kombinere visuelle inntrykk som video og grafikk, håndtere lyd og gjennom bruk av ekstern kode er det mulig å anvende svært mange digitale verktøy. Selv om Unity er en svært omfattende og avansert programvare som det er nødvendig å bruke tid til å sette seg inn i så gjør all funksjonaliteten Unity til et utmerket verktøy for digital historiefortelling.

7.2 Brukertester

For å svare på hvordan brukerene opplevde bruken av Oculus Rift og kroppssporing ble det utført en brukertest. Resultatene viste at brukerenes følelse av innlevelse og deltagelse økte ved å bruke disse verktøyene. Selv om mange av brukerene opplevde problemer under bruken av bevegelsessporingen og at kvaliteten på bevegelsessporingen ble vurdert som god til nøytral var de fleste enig og meget enig i at bevegelsessporingen bidro til følelsen av deltagelse. Bruken av Oculus Rift økte innlevelsen i historien for 20 ut av 21 testpersoner, der 13 stykker var meget enig og 7 var enig. Kun en av testpersonene var nøytral. Det ble også spurt om brukeren ønsket å oppleve historier ved bruken av Oculus Rift og bevegelsessporing der alle svarte ja. Dette viser at det er viktig å se på bruken av disse verktøyene i utviklingen av nye historier.

12 stykker var meget enig i at å delta i fortellingen økte innlevelsen og 8 var enig. Kun en person var nøytral. Dette viser at det å delta aktivt i en historie er viktig for å føle seg som en del av den. Dette ble også reflektert i at samtlige testpersoner ønsket å være en aktiv deltager til fordel for å være en passiv tilskuer.

7.3 Videre arbeid

For å se på videre arbeid er det lurt å se på hva som finnes av kommende teknologier. Det vil spesielt være interessant å se på hvordan følelsen av innlevelse øker ved å ta i bruk flere sanser og former for interaktivitet. Forbrukervarianten av Oculus Rift som er ment å komme for salg i første kvartal av 2016 vil ha et innebygget par med hodetelefoner og støtte for avansert lydprosessering for å gi et svært nøyaktig lydbilde. Hvis det i tillegg er mulig å ta i bruk andre sanser som luktesans og berøring vil man etter hvert kunne skape digitale miljøer der brukeren vil oppnå en sterk følelse av innlevelse. Dette er et svært interessant område for forskning.

I tillegg til å kartlegge nye digitale verktøy som påvirker andre sanser vil det være nødvendig å se på hvordan historiefortelling blir nødt til å tilpasse seg de nye verktøyene. Hvis målet er å skape en film for bruk I Oculus Rift må filmskaperne ta høyde for brukerens muligheter for interaksjon ved å snu på hodet. Da blir det viktig å passe på hva som er bak kameraet for å ikke bryte illusjonen. Det vil da også være nødvendig med nye metoder for å dirigere brukeren mot der hendelsen skjer. Hvis det i tillegg skal bli muligheter for å aktivt delta i scenen vil det være nødvendig med nye måter å fortelle historier på.

Det vil også være lurt å undersøke de andre store spillmotorene for å kartlegge styrker og svakheter med hver av dem. De ulike motorene kan ha forskjellige muligheter når det kommer til å involvere flere typer media i samme program.

Tillegg A

Brukertester

Under følger de komplette resultatene fra brukertesten.

Kjønn	Alder	Kvaliteten på bevegelsessporingen var:	Bevegelsessporingen økte innlevelsen i historien:	Jeg følte meg som en aktiv deltager i fortellingen:	Ved å delta i fortellingen økte innlevelsen:
Mann	22	Nøytral	Enig	Enig	Enig
Mann	23	Nøytral	Enig	Meget enig	Meget enig
Mann	22	Nøytral	Enig	Enig	Enig
Mann	21	God	Enig	Enig	Meget enig
Mann	21	Nøytral	Enig	Enig	Enig
Mann	21	Nøytral	Enig	Enig	Nøytral
Mann	23	God	Enig	Enig	Enig
Mann	22	God	Meget enig	Meget enig	Meget enig
Mann	21	God	Enig	Enig	Enig
Mann	24	God	Meget enig	Meget enig	Meget enig
Mann	22	God	Enig	Enig	Meget enig
Mann	22	God	Meget enig	Enig	Meget enig
Mann	22	Dårlig	Enig	Enig	Enig
Mann	28	Nøytral	Nøytral	Enig	Enig
Mann	21	Nøytral	Enig	Meget enig	Meget enig
Mann	28	Nøytral	Meget enig	Meget enig	Meget enig
Kvinne	22	Meget god	Enig	Meget enig	Meget enig
Kvinne	22	God	Enig	Enig	Meget enig
Kvinne	23	God	Meget enig	Meget enig	Meget enig
Kvinne	25	God	Enig	Enig	Enig
Mann	21	Nøytral	Enig	Enig	Meget enig

Figur A.1: Brukertest Del 1

Erfaring med bruk av Oculus Rift?	Erfaring med bruk av sporingssystemer?	Bevegelsessporingen bidro til følelsen av deltagelse:	Bruken av Oculus Rift økte innlevelsen i historien:
Ja	Nei	Enig	Enig
Nei	Nei	Enig	Enig
Nei	Ja	Enig	Enig
Ja	Nei	Meget enig	Enig
Nei	Nei	Enig	Meget enig
Ja	Ja	Meget enig	Enig
Ja	Ja	Meget enig	Meget enig
Nei	Nei	Meget enig	Meget enig
Nei	Ja	Enig	Meget enig
Ja	Ja	Meget enig	Meget enig
Nei	Nei	Enig	Enig
Nei	Nei	Meget enig	Meget enig
Nei	Ja	Meget enig	Meget enig
Nei	Nei	Enig	Enig
Nei	Nei	Enig	Meget enig
Ja	Nei	Meget enig	Meget enig
Ja	Nei	Enig	Meget enig
Nei	Nei	Meget enig	Meget enig
Nei	Nei	Meget enig	Meget enig
Nei	Nei	Enig	Nøytral
Nei	Nei	Enig	Meget enig

Figur A.2: Brukertest Del 2

Bruken av Oculus Rift bidro til følelsen av deltagelse:	Jeg foretrekker å oppleve historier:	Ønsker du å oppleve historier ved bruk av Oculus Rift og bevegelsessporing?
Meget enig	Som aktiv deltager.	Ja
Enig	Som aktiv deltager.	Ja
Meget enig	Som aktiv deltager.	Ja
Enig	Som aktiv deltager.	Ja
Meget enig	Som aktiv deltager.	Ja
Enig	Som aktiv deltager.	Ja
Enig	Som aktiv deltager.	Ja
Meget enig	Som aktiv deltager.	Ja
Enig	Som aktiv deltager.	Ja
Meget enig	Som aktiv deltager.	Ja
Enig	Som aktiv deltager.	Ja
Meget enig	Som aktiv deltager.	Ja
Meget enig	Som aktiv deltager.	Ja
Enig	Som aktiv deltager.	Ja
Enig	Som aktiv deltager.	Ja
Meget enig	Som aktiv deltager.	Ja
Meget enig	Som aktiv deltager.	Ja
Enig	Som aktiv deltager.	Ja
Meget enig	Som aktiv deltager.	Ja
Enig	Som aktiv deltager.	Ja
Enig	Som aktiv deltager.	Ja

Figur A.3: Brukertest Del 3

Her er kommentarer fra brukerene:

Kommentarer
Ble litt "bilsyk"-kvalm.
Bra opplegg med mykje potensiale!
Jeg prøvde to ganger, begge gangene ville ikke ultralydbildet dukke opp på skjermen. Jeg valgte svaralternativ 1 og jeg hadde på meg singlet. Vet ikke hvorvidt dette kan ha påvirket bildene, men det er kanskje relevant.
God oppgave med store muligheter i fremtiden
The future is here!
Kjempekult! Tror historiefortelling kan bli noe helt nytt, særlig for de som har litt problemer med å sitte stille og sliter med å følge med
Det var veldig gøy!
Litt feil kalibrert med hensyk på høyde. Derfor var hånda lavere i virkeligheten enn den var på bildet. Litt uvant første gangen å oppleve å styre noen andre.

Figur A.4: Kommentarer til brukertesten

Tillegg B

C# scrips

IKControl.cs

```
ï»¿
using UnityEngine;
using System;
using System.Collections;

[RequireComponent(typeof(Animator))]

public class IKControl : MonoBehaviour
{

    protected Animator animator;

    public bool ikActive = false;
    public Transform rightHandObj = null;
    public Transform leftHandObj = null;
    public Transform lookObj = null;

    void Start()
    {
        animator = GetComponent<Animator>();
    }

    //a callback for calculating IK
    void OnAnimatorIK()
    {
        if (animator)
        {

            //if the IK is active, set the position and rotation directly to the goal.
            if (ikActive)
            {

                // Set the look target position, if one has been assigned
                if (lookObj != null)
                {
                    animator.SetLookAtWeight(1);
                    animator.SetLookAtPosition(lookObj.position);
                }

                // Set the right hand target position and rotation, if one has been
                    assigned
                if (rightHandObj != null)
                {
                    animator.SetIKPositionWeight(AvatarIKGoal.RightHand, 1);
                    animator.SetIKRotationWeight(AvatarIKGoal.RightHand, 1);
                    animator.SetIKPosition(AvatarIKGoal.RightHand, rightHandObj.position
                        );
                    animator.SetIKRotation(AvatarIKGoal.RightHand, rightHandObj.rotation
                        );
                }
```

```
48
49                  // Set the right hand target position and rotation, if one has been
                        assigned
50                  if (leftHandObj != null)
51                  {
52                      animator.SetIKPositionWeight(AvatarIKGoal.LeftHand, 1);
53                      animator.SetIKRotationWeight(AvatarIKGoal.LeftHand, 1);
54                      animator.SetIKPosition(AvatarIKGoal.LeftHand, leftHandObj.position);
55                      animator.SetIKRotation(AvatarIKGoal.LeftHand, leftHandObj.rotation);
56                  }
57
58              }
59
60              //if the IK is not active, set the position and rotation of the hand and
                    head back to the original position
61              else
62              {
63                  animator.SetIKPositionWeight(AvatarIKGoal.RightHand, 0);
64                  animator.SetIKRotationWeight(AvatarIKGoal.RightHand, 0);
65                  animator.SetIKPositionWeight(AvatarIKGoal.LeftHand, 0);
66                  animator.SetIKRotationWeight(AvatarIKGoal.LeftHand, 0);
67                  animator.SetLookAtWeight(0);
68              }
69          }
70      }
71  }
```

scripts/IKControl.cs

ButtonHandler.cs

```
ï»¿
using UnityEngine;
using System.Collections;

public class ButtonHandler : MonoBehaviour {

 public float firstPressed = 0.0f;
 public int currentSelectedButtonID = -1;

 public void selected(int buttonID){

  if (currentSelectedButtonID == -1) {
   Debug.Log ("UI-button selected");
   firstPressed = Time.time;
   currentSelectedButtonID = buttonID;
  }
 }

 public void deselected(){
  currentSelectedButtonID = -1;
  firstPressed = 0.0f;
 }
}
```

scripts/ButtonHandler.cs

GrabObject.cs

```
ï»¿
using UnityEngine;
using System.Collections;

public class GrabObject : MonoBehaviour {

 public GameObject hand;

 public GameObject gelTarget;
 public GameObject transduserTarget;
 public GameObject itemAreaTarget;
 public GameObject scanningAreaTarget;

 private GameObject grabbedObject = null;
 private Transform objectOrgParent = null;

 private bool itemInHand = false;
 private float coolDownTimer = 0.0f;

 public bool hasGelInHand = false;
 public bool hasTransducerInHand = false;
 public bool isWithinScanningArea = false;

 void onStart(){
  coolDownTimer = Time.time;
 }

 void OnTriggerExit(Collider collision) {

  isWithinScanningArea = false;
 }

 public void ResetGrabbing(){

  if (itemInHand) {
   //Remove item from hand and place it back onto the tray:
   grabbedObject.GetComponent<Rigidbody> ().isKinematic = false;
   grabbedObject.transform.parent = objectOrgParent;

   if (grabbedObject == gelTarget) {
    grabbedObject.transform.position = objectOrgParent.transform.position;
    grabbedObject.transform.rotation = objectOrgParent.transform.rotation;
    grabbedObject.transform.localPosition += new Vector3 (-0.11f, 0, -0.06f);
    grabbedObject.transform.Rotate (270, 270, 0);
   } else if (grabbedObject == transduserTarget) {
    grabbedObject.transform.position = objectOrgParent.transform.position;
    grabbedObject.transform.rotation = objectOrgParent.transform.rotation;
    grabbedObject.transform.localPosition += new Vector3 (-0.11f, 0, 0.06f);
    grabbedObject.transform.Rotate (0, 0, 0);
   }
   itemInHand = false;
   hasGelInHand = false;
   hasTransducerInHand = false;
   coolDownTimer = Time.time;
   return;
  }

  //This might possible be a bit of a bear.
  itemInHand = false;
  hasGelInHand = false;
  hasTransducerInHand = false;
  coolDownTimer = Time.time;
  return;
 }

 void OnTriggerEnter(Collider collision){
  //Debug.Log ("Grabbed something");

  //If there's no item in the hand
```

```
70  if (!itemInHand) {
71
72   //Check if the cooldown is done
73   if ((Time.time - coolDownTimer) < 1.0f) {
74    return;
75   }
76
77   //Check if the collision is the gel target or transducer
78   if (collision.gameObject == gelTarget) {
79    grabbedObject = collision.gameObject;
80    //Check if the object is already being held.
81    if (grabbedObject.GetComponent<Rigidbody> ().isKinematic) {
82     return;
83    }
84    // Make it kinematic as we are holding it now
85    grabbedObject.GetComponent<Rigidbody> ().isKinematic = true;
86    // Store the original parent to restore it when letting loose
87    objectOrgParent = grabbedObject.transform.parent;
88    // And parent it to the hand
89
90    //Position the object correctly in the hand
91    grabbedObject.transform.parent = hand.transform;
92    grabbedObject.transform.position = hand.transform.position;
93    grabbedObject.transform.rotation = hand.transform.rotation;
94    grabbedObject.transform.localPosition += new Vector3 (-0.02f, 0.1f, 0);
95    grabbedObject.transform.Rotate (0, 180, 0);
96
97    itemInHand = true;
98    hasGelInHand = true;
99    coolDownTimer = Time.time;
100   return;
101
102  } else if (collision.gameObject == transduserTarget) {
103   grabbedObject = collision.gameObject;
104   //Check if the object is already being held.
105   if (grabbedObject.GetComponent<Rigidbody> ().isKinematic) {
106    return;
107   }
108   // Make it kinematic as we are holding it now
109   grabbedObject.GetComponent<Rigidbody> ().isKinematic = true;
110   // Store the original parent to restore it when letting loose
111   objectOrgParent = grabbedObject.transform.parent;
112   // And parent it to the hand
113
114   //Position the object correctly in the hand
115   grabbedObject.transform.parent = hand.transform;
116   grabbedObject.transform.position = hand.transform.position;
117   grabbedObject.transform.rotation = hand.transform.rotation;
118   grabbedObject.transform.localPosition += new Vector3 (-0.02f, 0.1f, 0);
119   grabbedObject.transform.Rotate (-90, 0, 0);
120
121   hasTransducerInHand = true;
122   itemInHand = true;
123   coolDownTimer = Time.time;
124   return;
125  }
126 }
127
128 //If there is an item in hand
129 if (itemInHand) {
130
131  //Check if the cooldown is done
132  if ((Time.time - coolDownTimer) < 1.0f) {
133   return;
134  }
135
136  if (collision.gameObject == itemAreaTarget){
137   //Remove item from hand and place it back onto the tray:
138   grabbedObject.GetComponent<Rigidbody> ().isKinematic = false;
139   grabbedObject.transform.parent = objectOrgParent;
```

```
140
141     if (grabbedObject == gelTarget) {
142      grabbedObject.transform.position = objectOrgParent.transform.position;
143      grabbedObject.transform.rotation = objectOrgParent.transform.rotation;
144      grabbedObject.transform.localPosition += new Vector3 (-0.11f, 0, -0.06f);
145      grabbedObject.transform.Rotate (270, 270, 0);
146     } else if (grabbedObject == transduserTarget) {
147      grabbedObject.transform.position = objectOrgParent.transform.position;
148      grabbedObject.transform.rotation = objectOrgParent.transform.rotation;
149      grabbedObject.transform.localPosition += new Vector3 (-0.11f, 0, 0.06f);
150      grabbedObject.transform.Rotate (0, 0, 0);
151     }
152     itemInHand = false;
153     hasGelInHand = false;
154     hasTransducerInHand = false;
155     coolDownTimer = Time.time;
156     return;
157    }
158    if (collision.gameObject == scanningAreaTarget) {
159
160     isWithinScanningArea = true;
161     coolDownTimer = Time.time;
162     return;
163    }
164   }
165
166  }
167
168 }
```

scripts/GrabObject.cs

ScenarioScript.cs

```
ï»¿
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class ScenarioScript : MonoBehaviour
{
 //Variables for the speech bubble.
 public Canvas speechCanvas;
 private Text speechButtonText;
 private Text buttonOne;
 private Text buttonTwo;

 public Canvas dialogueCanvas;
 private ButtonHandler dialogueButtonScript;

 private Transform doctorsSpeechTransform;
 private Transform defaultUITransform;
 private Transform oppositeWallTransform;

 private GameObject camera;
 private GameObject UiCamera;
 private GameObject alexViewObject;

 public GameObject Alex;
 private GameObject AlexRenderer;
 private GameObject AlexMesh;
 private IKControlAlex alexIKScript;

 private GameObject AlexWithoutHeadMesh;

 public GameObject Doctor;
 private GameObject DoctorMesh;
 public GameObject DoctorWithHead;
 private GameObject DoctorWithHeadMesh;

 Texture alexGelledTex;
 Texture alexNormal;

 //Variables for hand
 public GameObject RightHand;
 private GrabObject RightHandScript;

 public GameObject LeftHand;
 private GrabObject LeftHandScript;

 private StoreMovementScript movScript;

 //Movie variables
 public GameObject moviePlane;
 private MovieTexture movie;

 int STATE = 1;
 const int STATE_START = 1;
 const int STATE_APPLY_GEL = 2;
 const int STATE_SCAN = 3;
 const int STATE_SCANNING = 4;
 const int STATE_QUESTION_FROM_ALEX = 5;
 const int STATE_ANSWER_FROM_DOCTOR = 6;
 const int STATE_LEARNING = 7;
 const int STATE_END = 8;

 private bool alexView = false;
 private bool setup = true;

 const int STATE_WAIT = 100;
 float timer = 0.0f;

 // Use this for initialization
```

```
70  void Start ()
71  {
72   RightHandScript = RightHand.GetComponent<GrabObject>();
73   LeftHandScript = LeftHand.GetComponent<GrabObject>();
74   movScript = GetComponent<StoreMovementScript>();
75
76   speechButtonText = speechCanvas.GetComponentsInChildren<Text> () [0];
77   buttonOne = dialogueCanvas.GetComponentsInChildren<Text> () [0];
78   buttonTwo = dialogueCanvas.GetComponentsInChildren<Text> () [1];
79
80   dialogueButtonScript = dialogueCanvas.GetComponent<ButtonHandler> ();
81   alexGelledTex = Resources.Load("AlexGelled") as Texture;
82   alexNormal = Resources.Load("Alex") as Texture;
83
84   DoctorMesh = GameObject.Find ("CarlWithBonesTest/CarlMidGeo");
85   DoctorWithHeadMesh = GameObject.Find ("CarlWithBonesTestReservedModel/CarlMidGeoHead")
         ;
86
87   AlexMesh = GameObject.Find("AlexWithBones/Alexis_Geo_Low");
88   AlexWithoutHeadMesh = GameObject.Find("AlexWithBonesWithoutHead/Alexis_Geo_Low");
89   alexIKScript = GameObject.Find ("AlexWithBones").GetComponent<IKControlAlex>();
90
91   camera = GameObject.Find ("OVRCameraRig");
92   UiCamera = GameObject.Find ("UICamera");
93   alexViewObject = GameObject.Find ("AlexisView");
94
95   doctorsSpeechTransform = GameObject.Find ("DoctorSpeechPlacement").GetComponent<
         Transform>();
96   defaultUITransform = GameObject.Find ("PosDefaultUI").GetComponent<Transform>();
97   oppositeWallTransform = GameObject.Find ("PosWallOppositeDoctor").GetComponent<
         Transform>();
98
99   //Set up speech canvas
100  speechCanvas.transform.position = defaultUITransform.position;
101  dialogueCanvas.transform.position = defaultUITransform.position;
102
103  Renderer r = moviePlane.GetComponent<MeshRenderer>();
104  movie = (MovieTexture)r.material.mainTexture;
105  moviePlane.GetComponent<MeshRenderer> ().material.color = Color.black;
106  movie.loop = true;
107
108  //Set state intro
109  STATE = STATE_LEARNING;
110  timer = Time.time;
111
112 }
113
114 // Update is called once per frame
115 void Update ()
116 {
117  switch (STATE) {
118
119  case STATE_LEARNING:
120
121   movScript.pause();
122   //Learning phase in order to let the user adapt to the controls
123   if (Input.GetKeyDown ("space")) {
124    STATE = STATE_START;
125   }
126   break;
127
128  case STATE_START:
129
130   //Start of the thing. Check if it's the first run through or second
131   //If it's the run through the view of Alex, do some shit to make that possible.
132
133   if (alexView) {
134
135    Debug.Log ("Changing to Alexis View");
136
```

```
137     //Stop the movie from playing all the bloody time
138     movie.Stop ();
139     moviePlane.GetComponent<MeshRenderer> ().material.color = Color.black;
140 
141     //Change mesh of Alex to headless, reset skin
142     AlexMesh.GetComponent<SkinnedMeshRenderer> ().sharedMesh = AlexWithoutHeadMesh.
            GetComponent<SkinnedMeshRenderer> ().sharedMesh;
143     Alex.GetComponentInChildren<Renderer> ().material.mainTexture = alexNormal;
144 
145     //Change mesh of Doctor to headed
146     DoctorMesh.GetComponent<SkinnedMeshRenderer> ().sharedMesh = DoctorWithHeadMesh.
            GetComponent<SkinnedMeshRenderer> ().sharedMesh;
147 
148     //Reset the doctors grabby hands
149     GrabObject leftArmGrabScript = Doctor.GetComponentsInChildren<GrabObject> () [0];
150     leftArmGrabScript.ResetGrabbing ();
151 
152     GrabObject rightArmGrabScript = Doctor.GetComponentsInChildren<GrabObject> () [1];
153     rightArmGrabScript.ResetGrabbing ();
154 
155     //Change viewpoint to Alex's
156     camera.transform.position = alexViewObject.transform.position;
157     camera.transform.rotation = alexViewObject.transform.rotation;
158 
159     UiCamera.transform.parent = null;
160 
161     //Initiate playback of movement from script
162     movScript.playback ();
163 
164     //Start the bollocks anew
165     STATE = STATE_APPLY_GEL;
166     timer = Time.time;
167     break;
168   }
169 
170   Debug.Log ("Starting␣from␣Doctor's␣view");
171   movScript.record ();
172 
173   STATE = STATE_APPLY_GEL;
174   timer = Time.time;
175   break;
176 
177 
178 //Waiting for the gel to cross the gel area.
179 case STATE_APPLY_GEL:
180 
181   //Check for tardyness
182   if (Time.time - timer > 10) {
183     speechCanvas.transform.position = oppositeWallTransform.position;
184     speechCanvas.transform.rotation = oppositeWallTransform.rotation;
185     speechButtonText.text = "Alex:␣Tok␣ikke␣dette␣\n␣litt␣lang␣tid?␣";
186   };
187 
188   if (setup) {
189     Debug.Log ("STATE_APPLY_GEL:␣" + alexView.ToString());
190 
191     if (alexView) {
192       speechCanvas.transform.position = doctorsSpeechTransform.position;
193       speechCanvas.transform.rotation = doctorsSpeechTransform.rotation;
194     } else {
195       speechCanvas.transform.position = oppositeWallTransform.position;
196       speechCanvas.transform.rotation = oppositeWallTransform.rotation;
197     }
198 
199     speechButtonText.text = "Doktor:␣Hei.␣Velkommen␣til␣ultralydundersøkelse.\nDet␣fø
          rste␣vi␣skal␣gjøre␣er␣å␣legge␣på␣en␣\n␣gel␣som␣hjelper␣med␣å␣få␣et␣klart␣bilde,\
          n␣og␣så␣skal␣vi␣ta␣en␣titt␣på␣fosteret.␣";
200 
201     setup = false;
202   }
```

```
203
204     //Check if the hand is within scanning area
205     if (RightHandScript.isWithinScanningArea || LeftHandScript.isWithinScanningArea){
206
207       //Check if the gel is in hand
208       if (RightHandScript.hasGelInHand || LeftHandScript.hasGelInHand) {
209
210         Debug.Log ("Gel has entered target area");
211         Alex.GetComponentInChildren<Renderer> ().material.mainTexture = alexGelledTex;
212
213         speechCanvas.transform.position = defaultUITransform.position;
214
215         STATE = STATE_QUESTION_FROM_ALEX;
216         setup = true;
217         timer = Time.time;
218         break;
219       }
220
221       //Check if the transducer is in hand
222       if (RightHandScript.hasTransducerInHand || LeftHandScript.hasTransducerInHand) {
223
224         Debug.Log ("Transducer has entered target area");
225         speechCanvas.transform.position = oppositeWallTransform.position;
226         speechCanvas.transform.rotation = oppositeWallTransform.rotation;
227         speechButtonText.text = "Alex: Uhm, har du ikke glemt noe nå? ";
228         timer = Time.time;
229         break;
230       }
231     }
232     break;
233
234   case STATE_QUESTION_FROM_ALEX:
235
236     //Check for tardyness
237     if (Time.time - timer > 14) {
238       speechCanvas.transform.position = oppositeWallTransform.position;
239       speechCanvas.transform.rotation = oppositeWallTransform.rotation;
240       speechButtonText.text = "Alex: Er det noe gale? ";
241     };
242
243     if (setup) {
244       //Set up answers to question
245       Debug.Log ("STATE_QUESTION_FROM_ALEX: " + alexView.ToString());
246       buttonOne.text = "Doktor: Da var det på \ntide å se på barnet. ";
247       buttonTwo.text = "Doktor: Svaralternativ nummer to ,\n bare for morro skyld. ";
248       dialogueCanvas.transform.position = oppositeWallTransform.position;
249       dialogueCanvas.transform.rotation = oppositeWallTransform.rotation;
250       setup = false;
251       break;
252     }
253
254     if (Time.time - dialogueButtonScript.firstPressed > 2.0){
255       if (dialogueButtonScript.currentSelectedButtonID == 0) {
256
257         dialogueCanvas.transform.position = defaultUITransform.position;
258
259         if (alexView) {
260           speechCanvas.transform.position = doctorsSpeechTransform.position;
261           speechCanvas.transform.rotation = doctorsSpeechTransform.rotation;
262           speechButtonText.text = buttonOne.text;
263
264         } else {
265           speechCanvas.transform.position = oppositeWallTransform.position;
266           speechCanvas.transform.rotation = oppositeWallTransform.rotation;
267           speechButtonText.text = "Alex: Ah, okay! ";
268         }
269
270         STATE = STATE_SCAN;
271         timer = Time.time;
272         setup = true;
```

```
273       break;
274      } else if (dialogueButtonScript.currentSelectedButtonID == 1) {
275
276       dialogueCanvas.transform.position = defaultUITransform.position;
277
278       if (alexView) {
279        speechCanvas.transform.position = doctorsSpeechTransform.position;
280        speechCanvas.transform.rotation = doctorsSpeechTransform.rotation;
281        speechButtonText.text = buttonTwo.text;
282
283       } else {
284        speechCanvas.transform.position = oppositeWallTransform.position;
285        speechCanvas.transform.rotation = oppositeWallTransform.rotation;
286        speechButtonText.text = "Alex: Huh? Hva mener du? ";
287       }
288       STATE = STATE_SCAN;
289       timer = Time.time;
290       setup = true;
291       break;
292      }
293     }
294     break;
295
296    //Waiting for the scanner to get within the scanning area
297    case STATE_SCAN:
298
299     alexIKScript.lookObj = UiCamera.GetComponent<Transform>();
300
301     //Check if the hand is within scanning area
302     if (RightHandScript.isWithinScanningArea || LeftHandScript.isWithinScanningArea){
303
304      //Check if the transducer is in hand
305      if (RightHandScript.hasTransducerInHand || LeftHandScript.hasTransducerInHand) {
306
307       Debug.Log ("Transducer has correctly entered target area");
308       STATE = STATE_SCANNING;
309       timer = Time.time;
310       break;
311      }
312     }
313     break;
314
315    case STATE_SCANNING:
316
317     alexIKScript.lookObj = moviePlane.GetComponent<Transform>();
318
319     if ((Time.time - timer) > 7) {
320      STATE = STATE_END;
321      timer = Time.time;
322      movie.Stop();
323      moviePlane.GetComponent<MeshRenderer> ().material.color = Color.black;
324      break;
325     }
326
327     if (!movie.isPlaying) {
328      movie.Play ();
329      moviePlane.GetComponent<MeshRenderer> ().material.color = Color.white;
330     }
331
332     if (RightHandScript.isWithinScanningArea || LeftHandScript.isWithinScanningArea) {
333      break;
334     }
335     movie.Stop();
336     moviePlane.GetComponent<MeshRenderer> ().material.color = Color.black;
337     STATE = STATE_SCAN;
338     break;
339
340    case STATE_END:
341
342     //Endcase.
```

```
343
344     if (setup) {
345
346      Debug.Log ("STATE_END: " + alexView.ToString());
347
348      alexIKScript.lookObj = UiCamera.GetComponent<Transform>();
349
350      if (alexView) {
351       speechCanvas.transform.position = doctorsSpeechTransform.position;
352       speechCanvas.transform.rotation = doctorsSpeechTransform.rotation;
353      } else {
354       speechCanvas.transform.position = oppositeWallTransform.position;
355       speechCanvas.transform.rotation = oppositeWallTransform.rotation;
356      }
357      speechButtonText.text = "Doktor: Det var det. \nAlt virker som det er i orden med
             forsteret. \nDa skal jeg bare legge fra meg apparatet så er vi ferdig. ";
358      setup = false;
359     }
360
361     if (RightHandScript.hasTransducerInHand || LeftHandScript.hasTransducerInHand ||
             RightHandScript.hasGelInHand || LeftHandScript.hasGelInHand) {
362
363      //Items still in hand, wait.
364      break;
365     }
366
367     alexView = true;
368     setup = true;
369     STATE = STATE_LEARNING;
370     speechCanvas.transform.position = defaultUITransform.position;
371     break;
372    }
373   }
374
375 }
```

scripts/ScenarioScript.cs

StoreMovementScript.cs

```
ï»¿
using UnityEngine;
using System.Collections.Generic;
using System;

public class StoreMovementScript : MonoBehaviour {

 private GameObject RightHandTarget;
 private GameObject LeftHandTarget;
 public Transform HeadTarget = null;

 private State scriptState;

 enum State{
  pause,
  recording,
  playback
 };

 struct transformStruct{
  public Vector3 position;
  public Quaternion rotation;
 }

 //Make a queue for the movements
 private Queue<transformStruct> RightHandTransformQueue = new Queue<transformStruct>();
 private Queue<transformStruct> LeftHandTransformQueue = new Queue<transformStruct>();
 private Queue<transformStruct> HeadTransformQueue = new Queue<transformStruct>();

 private transformStruct tempTransform;

 // Use this for initialization
 void Start () {
  scriptState = State.pause;
  RightHandTarget = GameObject.Find("TargetRightHand");
  LeftHandTarget = GameObject.Find("TargetLeftHand");
 }

 public void pause(){

  scriptState = State.pause;
  //Debug.Log("State: Pause");
 }

 public void record(){
  scriptState = State.recording;
  //Debug.Log("State: Recording");
 }

 public void playback(){
  scriptState = State.playback;
  //Debug.Log("State: Playback");

  //Access tracker scripts and disable tracking
  trackerInstance rTracker = RightHandTarget.GetComponent<trackerInstance>();
  rTracker.ApplyTracking = trackerInstance.Transform_Type.None;

  trackerInstance lTracker = LeftHandTarget.GetComponent<trackerInstance>();
  lTracker.ApplyTracking = trackerInstance.Transform_Type.None;
 }

 // Update is called once per frame
 void Update () {

  switch (scriptState){

  case State.pause:

   break;
```

```
70
71    case State.recording:
72      tempTransform.position = RightHandTarget.transform.position;
73      tempTransform.rotation = RightHandTarget.transform.rotation;
74      RightHandTransformQueue.Enqueue(tempTransform);
75
76      tempTransform.position = LeftHandTarget.transform.position;
77      tempTransform.rotation = LeftHandTarget.transform.rotation;
78      LeftHandTransformQueue.Enqueue(tempTransform);
79
80      tempTransform.position = HeadTarget.transform.position;
81      tempTransform.rotation = HeadTarget.transform.rotation;
82      HeadTransformQueue.Enqueue(tempTransform);
83      break;
84
85    case State.playback:
86      if (RightHandTransformQueue.Count > 0){
87        tempTransform = RightHandTransformQueue.Dequeue();
88        RightHandTarget.transform.position = tempTransform.position;
89        RightHandTarget.transform.rotation = tempTransform.rotation;
90
91        tempTransform = LeftHandTransformQueue.Dequeue();
92        LeftHandTarget.transform.position = tempTransform.position;
93        LeftHandTarget.transform.rotation = tempTransform.rotation;
94
95        tempTransform = HeadTransformQueue.Dequeue();
96        HeadTarget.transform.position = tempTransform.position;
97        HeadTarget.transform.rotation = tempTransform.rotation;
98      }
99
100     break;
101   }
102 }
103
104 void OnDisable(){
105   //Disable!
106 }
107 }
```

scripts/StoreMovementScript.cs

trackerInstance.cs

```
ï»¿
using UnityEngine;
using System.Collections;
using System;
using System.Runtime.InteropServices;
using System.IO;

public class trackerInstance : MonoBehaviour {

 [ StructLayout( LayoutKind.Sequential, Pack=0 )] // set Pack=0 for Windows and Pack=1
     for OSX
 public struct TrackerReport
 {
  public VrpnManager.TimeVal msg_time;
  public int sensor;
  [ MarshalAs( UnmanagedType.ByValArray, SizeConst=3 )]
  public double[] pos;
  [ MarshalAs( UnmanagedType.ByValArray, SizeConst=4 )]
  public double[] quat;
 }

 // Class Properties
 public static int num_trackers = 0;
 public enum Derivation_Type { None=1, Velocity=2, Acceleration=4 };
 public enum Transform_Type { None=1, Position=2, Orientation=3, Both=4 };
 public bool trackRelativeToObjectPos = false;

    //Object Properties
    private Vector3 objectInitialPos = Vector3.zero;
    private Quaternion objectInitialQuat = Quaternion.identity;

    //Tracker Properties
    private bool hasFoundFirstPos = false;

 // Public Properties
 public VrpnManager.Tracker_Types TrackerType;
 public string TrackerName;
 public Transform_Type ApplyTracking = Transform_Type.Both;
 public Derivation_Type Derivation = Derivation_Type.None;
 public Transform DeviceToUnity;
 public Transform SensorOffset;
 public int SensorNumber = 0;
 public int MaxReports = 20;
 public bool ShowDebug = false;

 // Private Variables
 private bool initialized = false;
 private VrpnManager.TimeVal LastReport;
 private IntPtr[] repsPtr;

 private Vector3 lastPos = Vector3.zero;
 private Quaternion lastQuat = Quaternion.identity;
 private Vector3 trackerPos = Vector3.zero;
 private Quaternion trackerQuat = Quaternion.identity;
 private Vector3 pos = Vector3.zero;
 private Quaternion quat = Quaternion.identity;
    private Vector3 trackerFirstPos = Vector3.zero;
    private Quaternion trackerFirstQuat = Quaternion.identity;

    private Vector3 scalePos = Vector3.zero;

 private string debug_text = "";
 private int debug_xoffset;

 public Vector3 GetPosition()
 {
  return trackerPos;
 }

```

```
69  public Quaternion GetRotation()
70  {
71   return trackerQuat;
72  }
73
74  [DllImport("E:/Users/CafeMedia/Google␣Drive/Skoleting/Masteroppgave/Unity/Assets/
        Plugins/VRPN/VrpnProject.dll")]
75  private static extern int VRPNTrackerStart(string name, int deriv);
76
77  void Start () {
78
79        //Get the object position and rotation:
80        objectInitialPos = transform.position;
81        objectInitialQuat = transform.rotation;
82
83        scalePos[0] = -1;
84        scalePos[1] = 1;
85        scalePos[2] = 1;
86
87   //allocate unmanaged memory for tracker reports
88   repsPtr = new IntPtr[MaxReports];
89   TrackerReport report = new TrackerReport();
90   report.sensor = 0;
91   report.pos = new double[3];
92   report.quat = new double[4];
93   report.quat[3] = 1.0f;
94   for(int i=0; i<MaxReports ; i++)
95   {
96    repsPtr[i] = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(TrackerReport)));
97    Marshal.StructureToPtr(report, repsPtr[i], true);
98   }
99   if (DeviceToUnity == null || SensorOffset == null)
100  {
101   if (DeviceToUnity == null)
102    Console.WriteLine("Warning:␣DeviceToUnity␣field␣not␣referencing␣a␣transform");
103   if (SensorOffset == null)
104    Console.WriteLine("Warning:␣SensorOffset␣field␣not␣referencing␣a␣transform");
105  }
106
107  // Setup last report time memory
108  LastReport = new VrpnManager.TimeVal();
109 }
110
111 bool StartTracker () {
112  if (VrpnManager.initialized)
113  {
114   // Register Tracker Device
115   int debug = VRPNTrackerStart(TrackerName,(int)Derivation);
116   num_trackers++;
117   initialized = true;
118           //Console.WriteLine(debug.ToString());
119   return true;
120  }
121  return false;
122 }
123
124 void Update () {
125  // Ensure device is ready
126  if (!initialized && !StartTracker()) return;
127
128        //Debug.Log("TrackerInstance: Laawrd Almaighty");
129  GetLatestPosQuat();
130  //GetAveragePosQuat();
131
132        if (!hasFoundFirstPos) //Find first position
133        {
134            trackerFirstPos = trackerPos;
135            trackerFirstQuat = trackerQuat;
136            if (ShowDebug)
137            {
```

```
138                 debug_text = this.name + System.String.Format("\nORI[X:{0:F2},Y:{1:F2},Z
                        :{2:F2}]", trackerFirstPos.x, trackerFirstPos.y, trackerFirstPos.z);
139                 //System.String.Format("\nPOS[X:{0:F2},Y:{1:F2},Z:{2:F2}]", pos.x, pos.y
                        , pos.z) +
140                 Debug.Log(debug_text);
141             }
142             hasFoundFirstPos = true;
143         }
144
145   if (ApplyTracking == Transform_Type.Both)
146   {
147             pos = (trackerPos - SensorOffset.position);
148             pos.x = pos.x * -1;
149             lastPos = pos;
150
151             //Interchange x and z orientation
152             quat = trackerQuat;
153             quat.z = quat.z * -1;
154
155             lastQuat = quat;
156    UpdateTransform(pos, quat);
157   }
158   else if (ApplyTracking == Transform_Type.Position)
159   {
160             if (trackRelativeToObjectPos)
161             {
162                 pos = ((trackerPos - trackerFirstPos) - SensorOffset.position);
163                 pos.x = pos.x * -1;
164                 lastPos = pos;
165                 quat = lastQuat;
166             }
167             else
168             {
169                 pos = trackerPos;
170                 lastPos = pos;
171                 quat = lastQuat;
172             }
173    UpdateTransform(pos, quat);
174   }
175   else if (ApplyTracking == Transform_Type.Orientation)
176   {
177    pos = lastPos;
178    quat = trackerQuat;
179    lastQuat = quat;
180    UpdateTransform(pos, quat);
181   }
182   else
183   {
184    //pos = lastPos;
185    //quat = lastQuat;
186   }
187
188
189
190         /*if (ShowDebug)
191         {
192             debug_text = this.name + System.String.Format("\nORI[X:{0:F2},Y:{1:F2},Z:{2:
                    F2}]", quat.eulerAngles[0], quat.eulerAngles[1], quat.eulerAngles[2]);
193             //System.String.Format("\nPOS[X:{0:F2},Y:{1:F2},Z:{2:F2}]", pos.x, pos.y,
                    pos.z) +
194             Debug.Log(debug_text);
195         }*/
196 }
197
198 protected virtual void UpdateTransform(Vector3 pos, Quaternion quat)
199 {
200    if (ApplyTracking == Transform_Type.Both || ApplyTracking == Transform_Type.Position)
        {
201    transform.localPosition = pos;
202
```

```
203   }
204   if (ApplyTracking == Transform_Type.Both || ApplyTracking == Transform_Type.
          Orientation) {
205    transform.localRotation = quat;
206   }
207  }
208
209  [DllImport("E:/Users/CafeMedia/Google␣Drive/Skoleting/Masteroppgave/Unity/Assets/
         Plugins/VRPN/VrpnProject.dll")]
210  private static extern int VRPNTrackerPosReport(string name, [In,Out] IntPtr rep, [Out]
         IntPtr ts, int sensor);
211
212  void GetLatestPosQuat()
213  {
214          VRPNTrackerPosReport(TrackerName,repsPtr[0],IntPtr.Zero,SensorNumber);
215   TrackerReport rep = (TrackerReport)Marshal.PtrToStructure(repsPtr[0],typeof(
          TrackerReport));
216          trackerPos.x = (float)rep.pos[0];
217   trackerPos.y = (float)rep.pos[1];
218   trackerPos.z = (float)rep.pos[2];
219   trackerQuat.x = (float)rep.quat[0]; //CHANGED THIS WITH THE Z ONE
220   trackerQuat.y = -(float)rep.quat[1];
221   trackerQuat.z = (float)rep.quat[2];
222   trackerQuat.w = (float)rep.quat[3];
223   LastReport.tv_sec = rep.msg_time.tv_sec;
224   LastReport.tv_usec = rep.msg_time.tv_usec;
225  }
226
227  [DllImport("E:/Users/CafeMedia/Google␣Drive/Skoleting/Masteroppgave/Unity/Assets/
         Plugins/VRPN/VrpnProject.dll")]
228  private static extern int VRPNTrackerNumPosReports(string name);
229
230  [DllImport("E:/Users/CafeMedia/Google␣Drive/Skoleting/Masteroppgave/Unity/Assets/
         Plugins/VRPN/VrpnProject.dll")]
231  private static extern void VRPNTrackerPosReports(string name, [In,Out] IntPtr[] repsPtr
         , [In,Out] ref int nmbr);
232
233  void GetAveragePosQuat()
234  {
235   int num = MaxReports;
236   VRPNTrackerPosReports(TrackerName,repsPtr,ref num);
237   float[] repsSum = new float[7];
238   int repsCount = 0;
239   TrackerReport[] reps = new TrackerReport[num];
240   for (int i=0; i<num; i++)
241   {
242    reps[i] = (TrackerReport)Marshal.PtrToStructure(repsPtr[i],typeof(TrackerReport));
243    if (reps[i].sensor == SensorNumber && VrpnManager.TimeValGreater(ref reps[i].msg_time
           ,ref LastReport))
244    {
245     repsSum[0] += (float)reps[i].pos[0];
246     repsSum[1] += (float)reps[i].pos[1];
247     repsSum[2] += (float)reps[i].pos[2];
248     repsSum[3] += (float)reps[i].quat[2];
249     repsSum[4] += -(float)reps[i].quat[1];
250     repsSum[5] += (float)reps[i].quat[0];
251     repsSum[6] += (float)reps[i].quat[3];
252     LastReport.tv_sec = reps[i].msg_time.tv_sec;
253     LastReport.tv_usec = reps[i].msg_time.tv_usec;
254     repsCount++;
255    }
256   }
257   if (repsCount > 0)
258   {
259    trackerPos.x = repsSum[0]/(float)repsCount;
260    trackerPos.y = repsSum[1]/(float)repsCount;
261    trackerPos.z = repsSum[2]/(float)repsCount;
262    trackerQuat.x = repsSum[3]/(float)repsCount;
263    trackerQuat.y = repsSum[4]/(float)repsCount;
264    trackerQuat.z = repsSum[5]/(float)repsCount;
```

```
265     trackerQuat.w = repsSum[6]/(float)repsCount;
266    }
267    else
268    {
269     trackerPos = lastPos;
270     trackerQuat = lastQuat;
271    }
272   }
273
274   Vector3 LatLongToMeter(float latitude)
275   {
276    // Convert latitude to radians
277    Vector3 result = Vector3.one;
278    double lat = (latitude * 2.0 * Math.PI)/360.0;
279
280    // Set up "Constants"
281    double m1 = 111132.92;  // latitude calculation term 1
282    double m2 = -559.82;  // latitude calculation term 2
283    double m3 = 1.175;   // latitude calculation term 3
284    double m4 = -0.0023;  // latitude calculation term 4
285    double p1 = 111412.84;  // longitude calculation term 1
286    double p2 = -93.5;   // longitude calculation term 2
287    double p3 = 0.118;   // longitude calculation term 3
288
289    // Calculate the length of a degree of latitude and longitude in meters
290    double latlen = m1 + (m2 * Math.Cos(2 * lat)) + (m3 * Math.Cos(4 * lat)) + (m4 * Math.
           Cos(6 * lat));
291    double longlen = (p1 * Math.Cos(lat)) + (p2 * Math.Cos(3 * lat)) + (p3 * Math.Cos(5 *
           lat));
292
293    result.x = (float)longlen;
294    result.z = (float)latlen;
295    return result;
296   }
297
298   public static Quaternion QuaternionFromMatrix(Matrix4x4 m)
299   {
300    // Adapted from: http://www.euclideanspace.com/maths/geometry/rotations/conversions/
           matrixToQuaternion/index.htm
301    Quaternion q = new Quaternion();
302    q.w = Mathf.Sqrt( Mathf.Max( 0, 1 + m[0,0] + m[1,1] + m[2,2] ) ) / 2;
303    q.x = Mathf.Sqrt( Mathf.Max( 0, 1 + m[0,0] - m[1,1] - m[2,2] ) ) / 2;
304    q.y = Mathf.Sqrt( Mathf.Max( 0, 1 - m[0,0] + m[1,1] - m[2,2] ) ) / 2;
305    q.z = Mathf.Sqrt( Mathf.Max( 0, 1 - m[0,0] - m[1,1] + m[2,2] ) ) / 2;
306    q.x *= Mathf.Sign( q.x * ( m[2,1] - m[1,2] ) );
307    q.y *= Mathf.Sign( q.y * ( m[0,2] - m[2,0] ) );
308    q.z *= Mathf.Sign( q.z * ( m[1,0] - m[0,1] ) );
309    return q;
310   }
311
312  }
```

scripts/trackerInstance.cs

VrpnManager.cs

```
ï»¿
using UnityEngine;
using System.Collections;
using System;
using System.Runtime.InteropServices;
using System.IO;

public class VrpnManager : MonoBehaviour {

 [StructLayout(LayoutKind.Sequential, Pack=0)]
 public struct TimeVal
 {
  public UInt32 tv_sec;
  public UInt32 tv_usec;
 }

 public string ServerAddress = "";
 public static bool debug_flag = false;
 public bool ShowDebug = false;
 public static bool initialized = false;

 private StringWriter debug_writer;
 private string[] debug_text;
 private int debug_index = 0;
 private int debug_buffer_pos = 0;
 private int debug_buffer_max = 500;
 private int debug_lines = 8;

 [DllImport ("E:/Users/CafeMedia/Google␣Drive/Skoleting/Masteroppgave/Unity/Assets/
     Plugins/VRPN/VrpnProject.dll")]
 private static extern void VRPNServerStart(string location);

 [DllImport("E:/Users/CafeMedia/Google␣Drive/Skoleting/Masteroppgave/Unity/Assets/
     Plugins/VRPN/VrpnProject.dll")]
 private static extern void VRPNServerStop();

 [DllImport("E:/Users/CafeMedia/Google␣Drive/Skoleting/Masteroppgave/Unity/Assets/
     Plugins/VRPN/VrpnProject.dll")]
 private static extern void VRPNServerLoop();

 [DllImport("E:/Users/CafeMedia/Google␣Drive/Skoleting/Masteroppgave/Unity/Assets/
     Plugins/VRPN/VrpnProject.dll")]
 private static extern void VRPNSetOutput(int handle);

 // Use this for initialization
 void Start () {

  Debug.Log("VRPNManager:␣Start");

  //Access external server
  VRPNServerStart(ServerAddress);
  initialized = true;
 }

 void OnDisable()
 {
  Debug.Log("VRPNManager:␣OnDisable");
  // Shut down connection to server
  VRPNServerStop();
 }

 // Update is called once per frame
 void Update () {
  //Run the server loop
  VRPNServerLoop();
 }

 public static bool TimeValGreater(ref TimeVal tv1, ref TimeVal tv2)
 {
```

```
  if (tv1.tv_sec > tv2.tv_sec)
   return true;
  if ((tv1.tv_sec == tv2.tv_sec) && (tv1.tv_usec > tv2.tv_usec))
   return true;
  return false;
 }

 public enum Tracker_Types {
  vrpn_3DConnexion_Navigator,
  vrpn_3DConnexion_SpaceBall5000,
  vrpn_3DConnexion_SpaceExplorer,
  vrpn_3DConnexion_SpaceMouse,
  vrpn_3DConnexion_Traveler,
  vrpn_3DMicroscribe,
  vrpn_5dt,
  vrpn_5dt16,
  vrpn_Analog_USDigital_A2,
  vrpn_Auxiliary_Logger_Server_Generic,
  vrpn_Button_5DT_Server,
  vrpn_Button_NI_DIO24,
  vrpn_Button_PinchGlove,
  vrpn_Button_Python,
  vrpn_Button_SerialMouse,
  vrpn_CerealBox,
  vrpn_Dial_Example,
  vrpn_DirectXFFJoystick,
  vrpn_DirectXRumblePad,
  vrpn_GlobalHapticsOrb,
  vrpn_Imager_Stream_Buffer,
  vrpn_ImmersionBox,
  vrpn_JoyFly,
  vrpn_Joylin,
  vrpn_Joystick,
  vrpn_Joywin32,
  vrpn_Keyboard,
  vrpn_Magellan,
  vrpn_Mouse,
  vrpn_National_Instruments,
  vrpn_NI_Analog_Output,
  vrpn_nikon_controls,
  vrpn_Phantom,
  vrpn_Poser_Analog,
  vrpn_Radamec_SPI,
  vrpn_raw_SGIBox,
  vrpn_SGIBOX,
  vrpn_Spaceball,
  vrpn_Tek4662,
  vrpn_TimeCode_Generator,
  vrpn_Tng3,
  vrpn_Tracker_3DMouse,
  vrpn_Tracker_3Space,
  vrpn_Tracker_AnalogFly,
  vrpn_Tracker_ButtonFly,
  vrpn_Tracker_Crossbow,
  vrpn_Tracker_DTrack,
  vrpn_Tracker_Dyna,
  vrpn_Tracker_Fastrak,
  vrpn_Tracker_Flock,
  vrpn_Tracker_Flock_Parallel,
  vrpn_Tracker_GPS,
  vrpn_Tracker_InterSense,
  vrpn_Tracker_Liberty,
  vrpn_Tracker_MotionNode,
  vrpn_Tracker_NULL,
  vrpn_Tracker_PhaseSpace,
  vrpn_VPJoystick,
  vrpn_Wanda,
  vrpn_WiiMote,
  vrpn_XInputGamepad,
  vrpn_Xkeys_Desktop,
```

```
136     vrpn_Xkeys_Jog_And_Shuttle,
137     vrpn_Xkeys_Joystick,
138     vrpn_Xkeys_Pro,
139     vrpn_Zaber };
140 }
```

scripts/VrpnManager.cs

Bibliografi

[1] Simon Yeung. Gamasutra. Inverse Kinematics (two joints) for foot placement. http://www.gamasutra.com/view/news/129168/Inverse_Kinematics_two_joints_for_foot_placement.php. 20.01.2012

[2] Georgia Institute of Technology. UART - Unity AR Toolkit. https://research.cc.gatech.edu/uart/. 2015

[3] University of North Carolina at Chapel Hill. VRPN - Department of Computer Science. http://www.cs.unc.edu/Research/vrpn/. 2015

[4] Unity. Asset Store - Unity. https://www.assetstore.unity3d.com/. 2015

[5] Andreas Ulvøen. Rapport - Digital Storytelling Tools. 2014

[6] Blender. https://www.blender.org/. 2015

[7] Unity. Unity Manual Animation FAQ. http://docs.unity3d.com/Manual/MecanimFAQ.html. 2015

[8] Wikipedia. List of Unity games. http://en.wikipedia.org/wiki/List_of_Unity_games. 27.05.2015

[9] Dean Takahashi. VB GAMESBEAT. Game developers, start your Unity 3D engines (interview). http://venturebeat.com/2012/11/02/game-developers-start-your-unity-3d-engines-interview/. 02.11.2012

[10] Sixense. STEM System | Sixense. http://sixense.com/wireless. 2015

[11] Wikipedia. Wii MotionPlus. http://en.wikipedia.org/wiki/Wii_MotionPlus. 04.03.2015

[12] Microsoft. Kinect for Windows. https://www.microsoft.com/en-us/kinectforwindows/. 2015

[13] Leap Motion. Leap Motion | Mac & PC Motion Controller for Games. https://www.leapmotion.com/. 2015

[14] Wikipedia. Source (game engine). http://en.wikipedia.org/wiki/Source_%28game_engine%29. 06.06.2015

[15] Crytek. CryENGINE 3. http://www.crytek.com/cryengine/cryengine3/overview. 2015

[16] Wikipedia. SCUMM. http://en.wikipedia.org/wiki/SCUMM. 02.05.2015

[17] Portal 2. Official Portal 2 Website. http://www.thinkwithportals.com/ 2011

[18] Andre Infante. MakeUseOf. Going “Hands On” at the US Army Virtual Reality Game Studio http://www.makeuseof.com/tag/us-army-virtual-reality-studio/. 16.03.2015

[19] Kathleen McAuliffe. Discover Magazine. Curing the Wounds of Iraw with Virtual Therapy. http://discovermagazine.com/2008/oct/17-curing-the-wounds-of-iraq-with-virtual-therapy. 17.09.2008

[20] Nonny De La Pena. Project Syria: Premieres at the World Economic Forum. http://www.immersivejournalism.com/project-syria-premieres-at-the-world-economic-forum/. 23.01.2014

[21] Wikipedia. Virtual reality. http://en.wikipedia.org/wiki/Virtual_reality#1950_.E2.80.93_1970. 29.05.2015

[22] Wikipedia. Head-related transfer function. http://en.wikipedia.org/wiki/Head-related_transfer_function. 06.06.2015

[23] Atman Binstock. Blog - Powering the Rift. https://www.oculus.com/blog/powering-the-rift/. 15.05.2015

[24] Wikipedia. Virtual reality sickness. http://en.wikipedia.org/wiki/Virtual_reality_sickness. 06.05.2015

[25] Carolyn Handler Miller. Digital Storytelling, Second Edition: A creator's guide to interactive entertainment. Elinor Actipis. 2013.

[26] Wikipedia. Nintendo Virtual Boy. http://en.wikipedia.org/wiki/Virtual_Boy. 7.06.2015

[27] Oculus VR. Oculus Rift Development Kit 2. https://www.oculus.com/dk2/. 2015

[28] OptiTrack. http://www.optitrack.com/. 2015

[29] Unity 5. http://unity3d.com/. 2015

[30] What is Unreal Engine 4. https://www.unrealengine.com/. 2015

[31] HTC Re VIVE. http://www.htcvr.com/. 2015

[32] Samsung Gear VR - Features. http://www.samsung.com/global/microsite/gearvr/gearvr_features.html. 2015

[33] Wallace Sadowski Jr., Kay Stanney. Measuring and Managing Presence in Virtual Environments. http://web.cs.wpi.edu/ gogo/courses/imgd5100/papers/Sadowski_HVE_2002.html. 2002