Høgskulen på Vestlandet

Bachelorrapport

Evalueringsmodul for VR-balansetrening

Evaluation module for VR balance training

Sondre Melhus

Silja Stubhaug Torkildsen

Emma Stø Vetland

Dataingeniør/Informasjonsteknologi
Institutt for datateknologi, elektroteknologi og realfag
Veileder: Harald Soleim
Innleveringsdato: 04/06/21

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel: Evalueringsmodul for VR-balansetrening	*Dato: 04.06.2021*
Forfatter(e): Sondre Melhus, Silja Stubhaug Torkildsen, Emma Stø Vetland	*Antall sider u/vedlegg: 40* *Antall sider vedlegg: 7*
Studieretning: Dataingeniør/Informasjonsteknologi	*Antall disketter/CD-er:* 0
Kontaktperson ved studieretning: Harald Soleim	*Gradering:* Ingen
Merknader:	

Oppdragsgiver: Lars Peder Bovim ved SimArena, HVL	*Oppdragsgivers referanse:*
Oppdragsgivers kontaktperson: Lars Peder Bovim	*Telefon:* 976 82 402

Sammendrag:

Prosjektet går ut på å finne ut hvordan man på en god måte kan evaluere en pasients prestasjon under balansetrening i VR.

Dette ble gjort ved å lage en asset som kan lastes ned fra Unity Asset store og enkelt implementeres i ulike VR-baserte rehabiliteringsspill i Unity. Asseten inneholder funksjoner for registrering av distanse, tid og score.

Rapporten beskriver prosessen knyttet til utviklingen og testingen av en slik asset og hvordan den kan tas i bruk i rehabilitering.

This project involves determining how to evaluate a patient's performance during balance training in VR.

This is done by creating an asset that can be downloaded from Unity Asset store and implemented in VR based rehabilitation exercises in Unity. The asset contains functions for registration of distance, time and score.

This report describes the process of developing and testing such an asset and how it can be used in rehabilitation.

Stikkord:

VR	Rehabilitering	Unity

Forord

Denne rapporten dokumenterer bachelorprosjektet «Evalueringsmodul for VR-balansetrening». Prosjektet er gjennomført av Sondre Melhus, Silja Stubhaug Torkildsen og Emma Stø Vetland.

Vi vil takke oppdragsgiver Lars Peder Bovim for hans engasjementet rundt prosjektet. Hans kunnskap og gode tilbakemeldinger under utviklingen har virkelig vært uvurderlig.

Vi vil også takke veileder Harald Soleim for oppfølging og veiledning på skriving og struktur av rapporten.

INNHOLDSLISTE

ORDLISTE

Asset	En Unity asset er en vare eller produkt som du kan bruke i et Unity prosjekt eller spill. Det kan blant annet være en 3D-modell, bilder eller script.
API	Et programmeringsgrensesnitt. Sier noe om hvordan et system er bygget opp og hvordan du kan integrere en annen applikasjon til det.
AR	Utvidet virkelighet (augmented reality). Et digitalt lag legges oppå et bilde
Backend	Del programvare som ligger nærmest databasen der dataene er lagret. Det er også her de tunge kalkuleringsoperasjonene skjer, som brukeren av systemet ikke nødvendigvis har noe forhold til (SNL, 2020).
Behandler	Person som passer på og observerer bruker/pasient, gir instrukser. Gjerne en fysioterapeut.
Bruker/pasient	Person som bruker VR-utstyret og utfører øvelsene som spores.
C #	Programmeringsspråk som brukes for å skrive script i Unity.
Enum	En enum-klasse er en klasse som representerer en gruppe konstanter (endelige variabler).
Frontend	Dele av programvare som ligger nærmest brukeren. Kode som former det visuelle og bestemmer hva som skjer når du interagerer med skjermen (SNL, 2020)

Grensesnitt	Kontaktflate mellom ulike enheter eller delsystemer som skal virke sammen.
Hjerneslag	Fellesbetegnelse på sykdomstilstander som skyldes en plutselig forstyrrelse av blodsirkulasjonen i hjernen (SNL, 2020).
KINVENT Plates (KFORCE Plates)	To uavhengige kraftplattformer som brukes i rehabilitering for å måle statisk og dynamisk balanse under ulike bevegelser (Kinvent, 2021).
Klasse (C#/Unity)	Klasser er en måte å strukturere kode ved å samle variabler og funksjoner sammen for å definere egenskaper til et objekt.
Oculus Quest	VR-utstyret som brukes i prosjektet, består av et headset og to håndkontrollere.
Plug-in	En plug-in er en programvareutvidelse som er utviklet for å tilby ekstra funksjonalitet til et eller flere programmer.
Script	Fil med kode som forteller komponentene i Unity hva de skal gjøre. I dette prosjektet skrevet i C#.
Unity	En 3D-utviklingsplattform for å bygge 2D - og 3D – applikasjoner, som spill og simuleringer, ved bruk av .NET og programmeringsspråket C#. Det kan utvikles til 25+ plattformer på tvers av mobil, stasjonær, konsoll, TV, VR, AR og nett.
VR	Virtuell virkelighet (Virtual Reality). Handler om å skape illusjonen av å være fysisk tilstede i et kunstig miljø.

VR-briller/headset	Et headset med bilde og lyd som festes over hodet og øynene til brukeren. Gir tredimensjonalitet og dybdesyn og viser en virtuell virkelighet.
XR	Samlebetegnelse for VR, MR (mixed reality) og AR. Unity sitt XR API muliggjør fleksibel utvikling på tvers av disse teknologiene.

1 INNLEDNING

Hvert år rammes cirka 12 000 mennesker her i landet, både unge og gamle, av hjerneslag (Norsk hjerneslagsregister, 2015). Forskning viser at tidlig og aktiv rehabilitering øker sjansene for å gjenvinne tapte funksjoner. Hva som er den mest ideelle formen for slagrehabilitering er ikke avklart, men det er behov for god motivasjon kombinert med tilstrekkelig mengde, intensitet og varighet av treningen (Helsedirektoratet, 2020). I Helse Førde jobbes det med integrering av VR (Virtual Reality) som del av oppfølging av slagpasienter. Etter hjerneslag er det svært varierende utfordringer, og i dagens behandling har fysioterapeuter en funksjonell tilnærming. Det vil si at man trener på det som til enhver tid er krevende for pasienten. En klassisk fellesnevner for slagpasienter (og andre) er nedsatt sittende og/eller stående balanse. For å trene på dette er det en til to fysioterapeuter som støtter pasient i forflytning til sittende eller stående, og så gjennomføres ulike øvelser i korrekt posisjon. Øvelser kan være å gripe etter en gjenstand, lene seg fra side til side, rotere hode o.l. Ulike bevegelser som gjør det mer krevende å holde balansen.

VR kan gi muligheten for en mer pasienttilpasset behandling via teknologi, deriblant VR-briller, som krever mindre oppsett og ressurser enn det som trengs i tradisjonell behandling (Rose et al., 2005). Det realistiske miljøet VR-teknologi tilbyr gjør det mulig med interaktiv samhandling med omgivelsene, mens pasienten overvåkes og bevegelser registreres (Bohil et al., 2011). Ved hjelp av god registrering vil analyse av pasientens handlinger gjøre det lettere å gi retningslinjer og presis tilbakemelding i sanntid for å fremme ønsket atferd.

I dette prosjektet vil vi prøve å finne ut hvordan man kan registrere brukerinput fra VR-briller og kontroller. Dette vil forhåpentligvis lede til en løsning på hvordan man på en god måte kan evaluere en pasients prestasjon under balansetrening i VR.

1.1 Motivasjon og mål

Målet med denne oppgaven er å gjøre det lettere for fremtidige utviklere å registrere brukerinput fra rehabiliteringsøvelser i VR. En generisk evalueringsmodul som med relativt enkle grep kan integreres i Unity-baserte VR-spill. Den vil gjøre det lettere for en terapeut å kartlegge en pasients prestasjon på diverse øvelser. Videre kan også pasientens progresjon være interessant å kunne hente ut, ved å sammenligne resultater med tidligere gjennomføringer. Et første delmål vil være å kunne registrere start og stopp på en oppgave når det kommer til posisjon på VR-briller og kontroller, bevegelse, tid og objekt. Det neste målet blir da å lage en prototype av en evalueringsmodul som gjør det mulig å standardisere hvordan man evaluerer prestasjon på en gitt øvelse i VR.

1.2 Kontekst

Ved balansetrening benyttes en mengde evalueringsmetoder for å vurdere status og fremgang i trening. Oppdragsgiver Lars Peder Vatshelle Bovim har erfart at dette er et punkt som ofte må nedprioriteres i tidsavgrensede utviklingsprosjekter i VR, men likevel er det et element han savner når spillene tas i bruk. Han har et ønske om en plug-in i Unity hvor fysioterapeut legger inn grenseverdier for en øvelse og det er et grensesnitt som viser pasients resultat etter gjennomføring av en øvelse i VR.

1.3 Avgrensninger

En klar begrensning ved prosjektstart er gruppens tidligere erfaring med VR og utvikling i Unity. Siden det vil ta litt tid å gjøre seg kjent med nye verktøy og dette er et tidsbegrenset prosjekt, vil tid kunne begrense hvor mange funksjonaliteter gruppen rekker å implementere.

Evalueringsmodellen skal brukes i rehabilitering av pasienter. Det ville vært nyttig med en brukertest blant pasienter, men testene i dette prosjektet vil fokusere på funksjonalitet og tilbakemelding fra oppdragsgiver. En brukertest ville krevd en godkjent protokoll fra helsemyndigheter noe som er en tidkrevende prosess.

1.4 Ressurser

Ressurser som er tilgjengelig for bachelorgruppen i tillegg til egen kunnskap og egne erfaringer, er oppdragsgiver Lars Peder som stiller med kompetanse og nødvendig informasjon fra Helse Vest. Gruppen har også en intern veileder og kontakt med tidligere bachelor/master studenter som har utviklererfaring.

Utstyr

- Egne PCer
- VR-sett: Oculus Quest (Figur 1 og 2). Utlån fra HVL
- Datamaskin på masterlab. På utlån fra datafag ved HVL
- Otterbox USB-A til USB-C kabel. På utlån fra HVL

Figur 1 Oculus håndkontroller

Figur 2 Oculus headset

1.5 Oppbygging av rapporten

Kapittel 1 presenterer målet, problemet, avgrensninger og ressurser som er nødvendig.

Kapittel 2 er en beskrivelse av prosjektets bakgrunn og initielle krav.

Kapittel 3 tar for seg design av produktet og hvilke metoder gruppen har brukt under utviklingen.

Kapittel 4 beskriver den detaljerte fremgangsmåten under design og arkitektur av produktet.

Kapittel 5 beskriver evalueringsmetoder som ble brukt og evalueringsresultater.

Kapittel 6 inneholder diskusjon av resultatene og konsekvenser av valg som ble tatt. Det er også forslag til forbedringer og fremtidig bruk.

Kapittel 7 er en oppsummering av prosjektets mål, resultater og fremtiden til produktet.

2 PROSJEKTBESKRIVELSE

2.1 Praktisk bakgrunn

2.1.1 Prosjekteier

Lars Peder Vatshelle Bovim er Høgskolelektor ved SimArena i 50% stilling, med hovedansvar for drift og bruk av rehabiliteringslab F327 på HVL.

Han er også leder for Energisenteret for barn og unge ved Haukeland Universitetssykehus, og har gjennomført en master ved UiB (Fysioterapivitskap) med masterprosjektet "*VR Walk - Dual-task i form av tredemøllegange i virtuell virkelighet (VR) og dets innvirkning på gangmønsteret*".

2.1.2 Tidligere arbeid

For å trene balanse, er det ofte én til to fysioterapeuter som støtter pasient i forflytning til sittende eller stående, og så gjennomføres ulike øvelser i korrekt posisjon. En tidligere bacheloroppgave ved HVL kalt VR-Walk, utviklet av Valved, Bleikli, Tellevik (2018) i samarbeid med Helse Bergen, tar utgangspunkt i å utfordre og forbedre denne prosessen. VR-Walk skaper en trygg og gøy setting ved å kombinere rehabilitering på tredemølle, med visuelle stimuli fra et VR-headset. Tredemøllen introduserer det fysiske elementet, hvor pasienten går i et eget definert tempo, og VR-headset introduserer forskjellige øvelser som skal gjennomføres mens pasienten går. VR-Walk ga gruppen et innblikk i hvordan rehabilitering i VR kan fungere.

Ved starten av prosjektfasen hadde gruppen ikke klart å finne noe konkret programvare som tilbyr en god form for evaluering av bruker/pasients prestasjon i VR-øvelser. Det nærmeste som kunne ligne en slik løsning er innebygde scoringssystemer, hvor man får opp en enkel high-score etter at man har spilt en runde. Denne formen for evaluering fungerer kanskje fint i spillsammenheng, hvor hovedfokuset er på selve prestasjonen til brukeren, men den mangler gode evalueringspunkter som er nødvendig for å kunne gi både behandlere og pasienter konstruktiv tilbakemelding.

2.1.3 Initielle krav

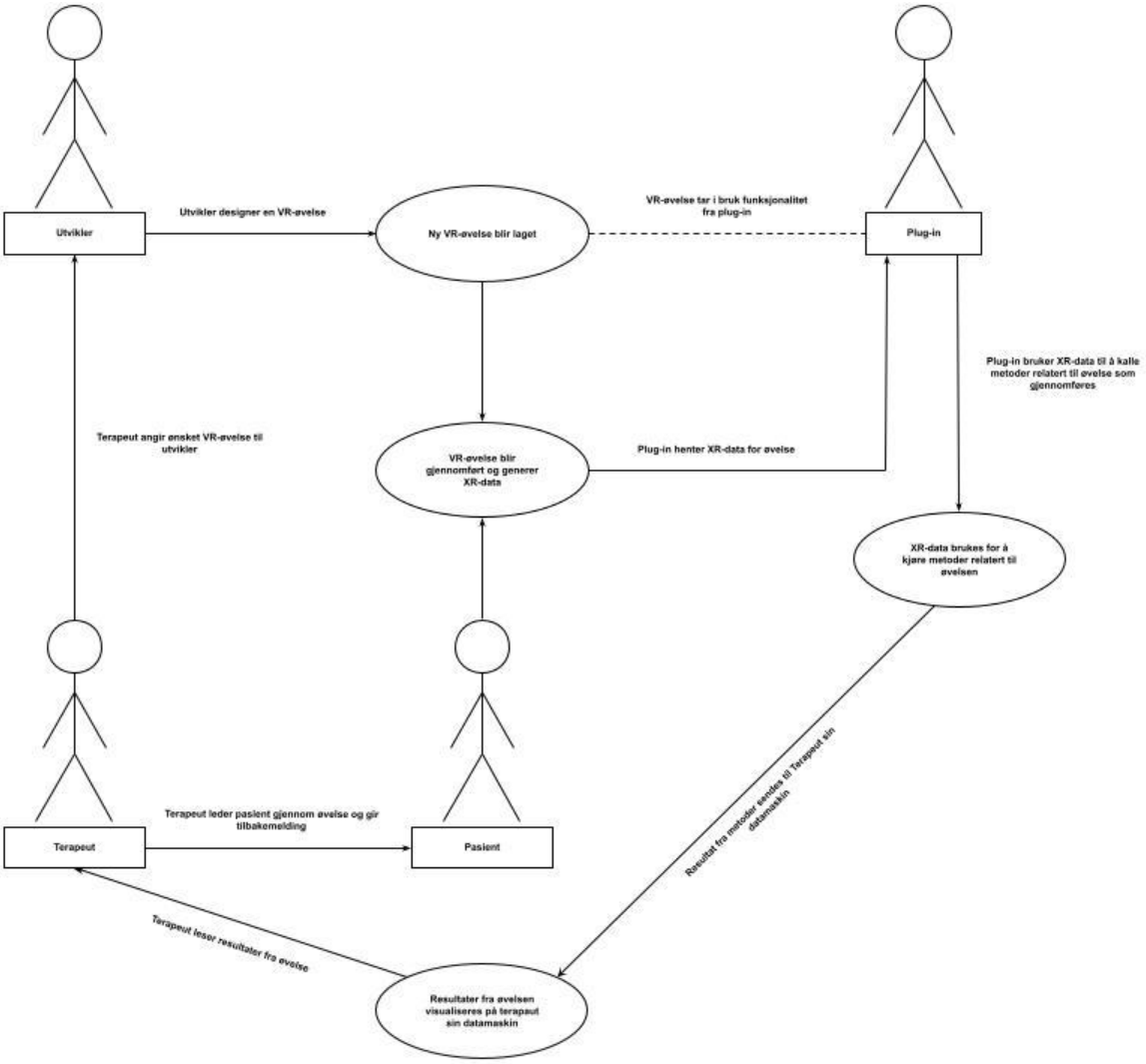
Prosjektets initielle krav er basert på flere møter med oppdragsgiver. Kravene fra oppdragsgiver ble på møtene diskutert og fremstilt slik at det ble klart for begge parter hva som var ønsket og hvordan prosjektet skulle utvikles. Oppdragsgiver beskrev en enkel plug-in som inneholdt funksjoner for å hente ut prestasjonsdata fra en gitt VR-øvelse mens den utføres. Prestasjonsdata vil i dette tilfelle være data som beskriver en brukers handlinger i en gitt oppgave. Dette kan for eksempel være posisjonsdata, som måler avstand fra start til stopp av en øvelse, eller tidsdata som forteller hvor lang tid

en pasient har brukt på å gjennomføre en oppgave. Videre var det et ønske om at en kopi av denne presentasjonsdataen skulle lagres i en ekstern database. Grunnet liten tid på oppgaven ble den eksterne databasen nedprioritert. Det ble sammen med oppdragsgiver bestemt å legge fokus på selve evalueringsmodulen som skal visualisere prestasjonsdata for en gitt øvelse. Ved videre utvikling kan databasefunksjonen legges til dersom oppdragsgiver ønsker det, mer om dette i kapittel 7.3. Dersom modulen var ferdigutviklet ønsket oppdragsgiver at det skulle lages en liten demo, for å demonstrere evalueringsmodulens funksjonalitet i en praktisk sammenheng. Demoen skulle være et enkelt program som ga en visuell og konkret fremstilling av hva plug-in kan brukes til.

Kortet ned i punktformat blir prosjektets initielle krav slik:

- Funksjonell plug-in som henter prestasjonsdata fra en VR-øvelse.
- Data standardiseres.
- En visuell fremstilling av data.
- En funksjonell demo som kan brukes i sammenheng med evalueringmodulen.

Figur 3 visualiserer hvordan evalueringsmodulen blir brukt med tanke på ulike roller. Utvikler har i oppgave å utvikle et rehabiliteringsspill med ønsket øvelse gitt av en terapeut. Terapeuten veileder og gir tilbakemelding til pasient basert på presentasjonsdata som blir returnert av plug-in. Pasienten gjennomfører øvelse i rehabiliteringsspillet som plug-in samler presentasjonsdata fra.



Figur 3 Visualisering av løsning

2.1.4 Initiell løsnings-idé

Gruppens initielle løsning er en plug-in som ikke skal kreve store mengder prosessering. Dette fordi programvaren som skal benytte seg av plug-in kjører på en trådløs Oculus Quest VR-enhet, med begrenset prosesseringskraft. Hovedfunksjonene vil være uthenting og prosessering av posisjonsdata, tid og score.

Figur 5 Stående balansetrening

Figur 4 Sittende balansetrening

Data blir hentet inn mens en pasient gjennomfører en VR-øvelse (Figur 4 og 5). Plug-in kan hente data fra flere forskjellige enheter samtidig. Dette er input-enheter som en bruker kan kontrollere. Hovedenhetene gruppen har lagt fokus på er VR-headset, samt høyre og venstre håndkontroller. Ved behov vil det være enkelt å legge til andre enheter, som for eksempel sporingsenheter for venstre og høyre bein.

Data som hentes inn vil bestå av distansen enheten har beveget seg, tiden pasienten har brukt på å fullføre en gitt oppgave og eventuelt hvor mange objekter pasienten har vært i kontakt med under øvelsen. For å gjøre løsningen mer modulær har gruppen laget flere varianter av målinger som kan gjennomføres.

Distanse kan enten måles i direkte avstand beveget eller total avstand beveget.

- Direkte avstand beveget vil kartlegge den direkte avstanden fra punkt A til punkt B. Dette vil altså si at den ikke tar hensyn til ekstra bevegelser som skjer mellom punkt A og punkt B, men kun den direkte avstanden mellom de to punktene.
- Total avstand beveget vil kartlegge den totale avstanden fra punkt A til punkt B. Dette vil altså si at den tar hensyn til ekstra bevegelser som skjer mellom punkt A og punkt B.

Tid kan enten måles i stoppeklokkeform eller nedtellingsform.

- Stoppeklokkeform vil kartlegge tiden en pasient har brukt fra de starter en oppgave til den er fullført.
- Nedtellingsform vil starte en nedtelling fra en predefinert tid når pasient starter en oppgave, og vil stoppe oppgaven når pasient har gjennomført oppgaven eller har gått tom for tid.

Score vil kun måles i teller-score.

- Teller-score vil fungere som en akkumulerende teller, som øker hver gang pasienten beveger en enhet innenfor et poenggivende felt.

Etter at data er hentet ut, skal den visualiseres for behandler. Gruppen har vurdert at en god løsning vil være en egen nedtrekksmeny som kun er synlig for utvikler/behandler. Denne menyen skal inneholde punkter som representere hver av de tre målingene som er gjennomført, samt hvilke variant av en måling som er valgt.

Gruppens initielle løsning vil ta utgangspunkt i utvikling for Oculus VR-enheter, i Unity som utviklermiljø. Dette kan potensielt skape fremtidige problemer dersom oppdragsgiver ønsker å ta i bruk VR-enheter fra en annen produsent, men i prinsippet burde dette enkelt kunne løses ved å gjøre noen endringer i plug-in sin kildekode.

Kravspesifikasjon fra oppdragsgiver

Navn nedtrekksmeny	**Alternativer/parameter**	**Forklaring**
#Oppgave start - Kommer til å starte #score	Vegg	En vegg som kan plasseres hvor en vil, i hvilken vinkel og størrelse en vil. Starter #score ved passering av vegg
	Tid	Starter #score etter å ha spilt i definert lang tid fra oppstart
	Knapp	Starter #score ved trykk på valgt knapp
	Andre?	
#Oppgave stopp - Kommer til å stoppe #score	Vegg	En vegg som kan plasseres hvor en vil, i hvilken vinkel og størrelse en vil. Stopper #score ved passering av vegg
	Score	Koblet til #score, og stopper når parameter i #score har nådd valgt verdi
	Knapp	Stopper #score ved trykk på valgt knapp
	Andre?	
#score	Ingen	Velges når #oppgave start og #oppgave stopp er tilstrekkelig (f.eks ved vegg-vegg)
	Tid	Teller tid i sekund og millisekund

	Distanse	Teller distanse i meter og desimeter for valgt sporingsenhet (briller, håndkontroll venstre, ekstern sensor osv.)(Krever sporing av tilbakelagt distanse. Om du beveger i sirkel og sånt må vi kunne måle strekning forflyttet)
	Objekt	Teller antall objekt som fanges/registreres

Tabell 1

2.2 Litteratur om problemstillingen

Gruppen har funnet mye god info i boken *Virtual Reality for Physical and Motor Rehabilitation (Weiss, Keshner, Levin, 2014)*. Boken omhandler generelle løsninger hvordan VR kan brukes i rehabilitering av fysiske og motoriske ferdigheter. Gruppen har hatt spesielt god nytte av kapittel 7, som omhandler bruken av VR i rehabilitering av slagpasienter. Dette kapittelet gir et godt innblikk i rehabiliteringsprosessen, og har gitt gruppen en bedre forståelse for hva som trengs for å ha suksess i en slik prosess.

3 DESIGN AV PROSJEKTET

3.1 Forslag til løsning

Oppdragsgiver ønsket en evalueringsmodul som enkelt kan implementeres i Unity-baserte VR-spill. Den burde altså lett kunne anvendes i ulike rehabiliteringsøvelser i VR. Med tanke på endelig bruk av verktøyet er det en fordel å forholde seg til samme datatyper og -strukturer og det blir dermed tatt utgangspunkt i Unity og C# som utviklerverktøy og programmeringsspråk. Det var også videre ønske om å koble til i K-INVENT, med utgangspunkt i balanseplater og potensielle utvidelser med tilkobling til annet testutstyr.

3.1.1 API som kjøres på trådløs VR-enhet

Den første løsningen består av å utvikle et API i Unity som er et program som tillater utvikling av spill på flere plattformer. Dette vil gjøre det enkelt for senere utvikling av VR-spill for andre utviklere og det vil være mulig å lage et lite demospill som var ønskelig fra oppdragsgiver.

APIet vil ta utgangspunkt i innhenting og prosessering av data. Fokuset vil i hovedsak være å hente ut dataene og bruke det til å måle progresjon. For å måle progresjon trengs det data fra tidligere målinger fra samme pasient. De tidligere dataene kan lagres i en testdatabase i forbindelse med dette prosjektet. Tanken er da at det skal være mulig for andre utviklere å kunne lagre data på sin egen database eventuelt skyløsning.

VR-enheten som vil bli brukt er Oculus briller, som er etter ønske fra oppdragsgiver. APIet vil også dermed være utviklet til disse brillene og vil ikke med sikkerhet virke på andre enheter.

3.1.2 Kobling til KINVENT, med utgangspunkt i Plates

Den andre løsningen blir å bygge videre på det første alternativet, ved å koble til KFORCE Plater. KFORCE Plater er to uavhengige kraftplattformer som brukes til rehabilitering av balanse og til å vurdere musklene på symmetri og styrke i beinet (underekstremiteter). Fordelen med KFORCE Plater er at de er utstyrt med elektroniske krafttransduser som blant annet gir rask respons. Platene gir også sanntid akustikk og optisk biofeedback til telefonen eller nettbrett med KFORCE-appen.

For å kunne gjøre dette vil det kreve kommunikasjon mellom KFORCE Platene og VR brillene. Informasjonen fra KFORCE som bli viktig å hente ut er COP (Center of pressure), det må da også tilrettelegges for at det er mer data som skal hentes ut.

3.1.3 Visuelle fremstillinger av resultater på tvers av testutstyr

Det tredje alternativet blir en utvidelse av første og andre alternativ. Det innebærer påkobling på annet testutstyr, og få visuelle fremstillinger på tvers av testutstyr.

3.1.4 Asset som kan lastes ned fra Unity Asset Store

En Unity asset er et produkt som du kan bruke i et Unity prosjekt eller spill. Det kan være en fil laget utenfor Unity, som en 3D-modell eller bilder, men det er også mulig å lage en asset i Unity som f.eks. en scene eller en tutorial. I Unity Asset Store kan man laste ned assets publisert av Unity Technologies, men også mange andre da alle med en Unity-konto kan laste opp og dele sine assets her.

Asseten vil bestå av en nedtrekksmeny hvor det vil være mulig å definere når spillet/øvelsen skal starte og stoppe, hvordan scoren skal vises visuelt og hva progresjonen til pasienten skal måles i. Score vil enten bli gitt i tid, distanse eller antall objekter som fanges/hentes av pasienten.

3.1.5 Diskusjon av alternativene

Hovedkriteriet i alle de forskjellige løsningene krever at det i utgangspunktet lages en god og fungerende måte å registrere bevegelsene til en pasient i VR. Løsningen for å kunne utvikle dette prosjektet vil dermed basere seg på dette. Det tas også i betraktning at gruppen har begrenset kunnskap når det kommer til Unity, Oculus Quest VR-briller og eventuelt annet testutstyr.

Den første løsningen har den fordelen at den sikter seg direkte inn mot selve registreringen av brukerinput og innhenting av data fra VR-brillene. Dette svarer til grunnkriteriene fra oppdragsgiver og det gir mindre arbeid med tanke på tidsbruk. Ved å fokusere på hovedoppgaven kan dette resultere i et bra API som kan gjenbrukes og eventuelt videreutvikles av andre utviklere. Ulempen er mangel på kunnskap i gruppen som gjør at det tar tid å sette i gang med utviklingen av selve APIet, da det er mye å sette seg inn i både nytt verktøy og en ny plattform.

For oppdragsgiver hadde det vært en stor fordel om KFORCE Plater også ble koblet til. Dette er et sentralt verktøy for fysioterapeuter som jobber med balansetrening og det ville gitt mer data å måle progresjonen til pasienten på. Ulempen er at det blir mer arbeid, som videre fører til tidspress. Siden gruppen har minimalt med kunnskap om verktøyet som brukes i første løsning, blir det vurdert at introduseringen av enda et verktøy ikke er hensiktsmessig.

Den tredje løsningen bygger i hovedsak på at både den første og den andre løsningen blir gjennomført dersom det er mer tid igjen.

Etter hvert som gruppen ble mer kjent med Unity, ble en fjerde løsning tydelig. Denne løsningen gikk ut på å lage en asset. Fordelen med dette alternativet er at det gjør det mulig utvikle et ferdig produkt som inneholder alt som trengs for registreringen av en pasients bevegelser og det kan legges direkte inn i et rehabiliteringsspill utviklet i Unity.

3.2 Valgt løsning

Ved starten av prosjektet bestemte gruppen seg på å fokusere på å utvikle et API i Unity. Det ble begrunnet med at de andre løsningene i hovedsak bygger på den første løsningen og det ville da gi bedre tid på å strukturere og programmere APIet. Denne løsningen svarte også til de initielle grunnkravene til oppdragsgiver og det ville være mulighet til å videreutvikle APIet til å inkludere andre plattformer og testutstyr.

Etter å ha gjort mye undersøkelser og flere møter med oppdragsgiver kom det en justering i hva det endelige produktet skulle være. Likevel førte ikke dette til store endringer da dette valget kom før det måtte tas beslutninger som ville skille et API fra en asset. Løsningen med en asset ble valgt da dette virket mer hensiktsmessig for videre bruk og gjennomføring. En asset vil kunne forenkle prosessen for fremtidige utviklere som vil bruke produktet, da de vil kunne laste det ned fra asset store og bruke den direkte i sitt spill.

Dersom gruppemedlemmene hadde hatt mer erfaring eller tidsrammen var større hadde det være muligheter for å koble til annet utstyr/plater og lage et mer grafisk brukergrensesnitt. Det er i hovedsak derfor valget ble en asset, og forhåpentligvis vil det i senere tid være mulighet for å videreutvikle produktet.

3.3 Valg av verktøy

Unity

Unity er en 3D-utviklingsplattform for å bygge 2D - og 3D – applikasjoner, som spill og simuleringer, ved bruk av .NET og programmeringsspråket C#. Unity kan utvikle til 25+ plattformer på tvers av mobil, stasjonær, konsoll, TV, VR, AR og nett.

C#

C# er et moderne, objektorientert og type-safe programmeringsspråk. C# gir muligheten til å lage mange sikre og robuste applikasjoner i .NET echo systemet. Det er et naturlig språk å bruke til å lage software komponenter grunnet at det gir språkkonstruksjoner som støtter konseptene direkte.

Quest 1

Quest VR-briller er plattformen som testspillet skal kjøres på og data kan hentes ut fra. Her ble funksjonene testet virtuelt og det var mulig å se hvilke eventuelle endringer som måtte gjøres. Fordelen med Oculus Quest er at man ikke trenger å være koblet til en datamaskin, men kan kjøre spill og programvare trådløst. Oppsettet er enkelt og Oculus Quest tilpasser seg omgivelsene dine, så du kan spille mens du står eller sitter, på et lite eller stort område.

Google Drive

Google Drive ble brukt til lagring av dokumenter som er relevant til bacheloroppgaven. Det er et godt verktøy når en skal dele og redigere dokumenter mellom flere.

Team Gantt

Team Gantt er en webapplikasjon som enkelt håndterer opprettelse av framdriftsplan, gjøremål og milepæler. Det er også enkelt for flere brukere å redigere samtidig, i tillegg til at det er mulig å opprette en PDF-fil.

Microsoft Visual Studio

Unity har en built-in support for å åpne script i Visual Studio Code som er en source-code editor. Siden produktet som utvikles i dette prosjektet skal kunne brukes i Unity ved spillutvikling til VR-briller er Visual Studio Code et naturlig valg.

GitHub

GitHub ble brukt til deling og lagring av kode. Det ble opprettet et privat delingsområde for prosjektet hvor gruppen har tilgang med sine egne GitHub-kontoer.

3.4 Prosjektmetodikk

3.4.1 Utviklingsmetodikk

Gruppene valgt en smidig (agile) utviklingsmetode, som vil si at det er iterativ og inkrementell utvikling av løsninger. Da kan man bruke tiden på det det som gir nytteverdi, ha klare prioriteringer og rammer og utvikle prototyper som kan testes og verifiseres. En slik fleksibel metode gjorde det lettere å gjøre endringer under utviklingen og respondere på tilbakemeldinger fra oppdragsgiver.

Koden ble i hovedsak utviklet mens gruppen satt sammen på masterlab og diskuterte løsningene underveis. Da var det også mulig for en på gruppen å ha på seg VR-

brillene, mens de andre instruerte eller gjorde justeringer i programmet. Det har også vært individuelt arbeid med rapport og programmering, både på skolen og hjemmefra. Da har gruppemedlemmene ofte avtalt på forhånd hva som må gjøres og siden oppdatert hverandre på fremgang som er gjort.

Det ble også opprettet en blogg som ble oppdatert ukentlig med hva som hadde skjedd og ville skje den neste uken.

3.4.2 Prosjektplan

Det har blitt laget et GANTT-skjema (Vedlegg 10.2) som viser milepæler og tidsfrister gruppen må overholde. Det ble satt av tid til møter ukentlig med både intern veileder og Lars Peder fra Helse Vest, slik at møter kunne avholdes ved behov.

3.4.3 Risikovurdering

Vedlegg 10.2 er en risikoanalyse som viser sannsynligheten for at en risiko inntreffer og hvor stor konsekvensen er. En slik analyse med forebyggende tiltak og strategier minsker sannsynligheten for at noe oppstår.

3.5 Evalueringsplan

Et nyttig verktøy som kan brukes til å måle brukbarheten til et produkt eller system er system usability scale (SUS) (Brooke, 1986). Da kan en bruker svare på 10 spørsmål og scoren for hvert spørsmål konverteres til et nytt tall, plusset sammen og ganget med 2.5 for å konvertere original score av 0-40 til 0-100.

	The System Usability Scale Standard Version	Strongly Disagree	1	2	3	4	Strongly Agree 5
1	I think that I would like to use this system frequently.		O	O	O	O	O
2	I found the system unnecessarily complex.		O	O	O	O	O
3	I thought the system was easy to use.		O	O	O	O	O
4	I think that I would need the support of a technical person to be able to use this system.		O	O	O	O	O
5	I found the various functions in this system were well integrated.		O	O	O	O	O
6	I thought there was too much inconsistency in this system.		O	O	O	O	O
7	I would imagine that most people would learn to use this system very quickly.		O	O	O	O	O
8	I found the system very awkward to use.		O	O	O	O	O
9	I felt very confident using the system.		O	O	O	O	O
10	I needed to learn a lot of things before I could get going with this system.		O	O	O	O	O

Figur 6 The standard System Usability Scale

Med utgangspunkt i en slik skala har gruppen utarbeidet noen påstander som kan brukes i evaluering av dette prosjektet. Da disse ikke er helt de samme som i SUS, vil resultatene ikke kunne sammenlignes med andre tilsvarende produkters score. Likevel kan de være nyttige for å si noe om resultatene i eget prosjektet.

Evalueringspunkter (Sterkt uenig (1) – Sterkt enig (5)):

1. Produktets målefunksjoner for distanse fungerer på en tilfredsstillende måte
2. Produktets målefunksjoner for distanse gir svar med tilfredsstillende presisjon
3. Produktets målefunksjoner for tid fungerer på en tilfredsstillende måte
4. Produktets målefunksjoner for tid gir svar med tilfredsstillende presisjon
5. Produktet har funksjonalitet som er til forventet nivå
6. Produktet oppfyller mine forventninger
7. Produktet er noen jeg kan se for meg å ta i bruk
8. Jeg er totalt sett fornøyd med det endelige sluttproduktet

Tilleggsspørsmål

9. Dersom prosjektet skulle startes på nytt med din nåværende erfaring, hvilke forandringer ville du gjort?

10. Ved videre arbeid på prosjektet, hva ville vært dine topp tre ønsker over nye funksjonaliteter?

4 DETALJERT DESIGN

4.1 Innledende fase

I innledningen av prosjektet ble veileder Harald Soleim og oppdragsgiver Lars Peder Vatshelle Bovim introdusert. Videre ble det lagt frem en kravspesifikasjon med ønskelige funksjoner, som Lars Peder hadde laget med tanke på hva som er etterspurt og nyttig i henhold til VR-baserte rehabiliteringsspill.

Plattform og relevante verktøy ble så videre introdusert. Sammen med oppdragsgiver, veileder og andre fra HVL med relevant kunnskap ble det bestemt at Unity var best å bruke som utviklerverktøy. Unity har blitt brukt av tidligere master- og bachelorstudenter, som også presenterte programmet gjennom en liten demo. Denne presentasjonen ga en grunnleggende forståelse for hvordan programmet fungerer og hvordan det kan brukes til å utvikle på VR-plattformer.

Videre i den innledende fasen ble utstyret satt opp og kontoer som var nødvendig for å kunne bruke programmene ble lagd. Det ble også gitt informasjon om hvem som kunne kontaktes om det skulle være spørsmål knyttet til programmeringen. Dette var andre internt på HVL med relevant kunnskap, samt noen tidligere studenter.

4.2 Planlegging

Planleggingsfasen startet med å planlegge de ulike fasene i prosjektet. Dette ble gjort i et GANTT-diagram som er nevnt tidligere. I dette diagrammet ble det lagt inn diverse innleveringsfrister og ulike faser av prosjektet, deriblant utvikling- og testfasen. Videre ble det bestemt hvilke programmer som skulle brukes og prosessen med å utforske programmene ble satt i gang. Sammen med oppdragsgiver ble det bestemt at Unity skulle brukes. Som nevnt tidligere var det manglende erfaring med dette og dermed gikk det mye tid i den innledende fasen på å opparbeide seg kunnskap som var nødvendig og finne ut hvilke funksjoner som kunne være nyttige.

Etter å ha gjort undersøkelser rundt oppbygningen av Unity og hvordan utvikling i programmet fungerer, ble det sammen med oppdragsgiver Lars Peder utarbeidet en ny kravspesifikasjon. Det var fortsatt ønske om de samme funksjonene som i første kravspesifikasjon, men funksjonene skulle bli levert i form av en asset istedenfor i et API.

4.3 Design and Creation

4.3.1 **Arkitektur**

Koble opp Oculus Quest med Unity

Unity er et program som gir muligheten til å utvikle spill i 3D og 2D. Dette programmet inneholder noen nøkkelegenskaper, blant dem muligheten til å utvikle på flere ulike plattformer. Dette prosjektet skulle utvikles til VR-briller og utstyret som ble brukt var Oculus Quest 1. For å få Unity til å kjøre på Oculus-brillene ble det brukt en USB-kabel som koblet VR-headset til datamaskinen. Videre ble appen Oculus Link installert på pc-en. Den viser oversikt over hvilket utstyr som er koblet til. Unity prosjektet som skulle kjøres på VR-brillene ble så definert som et "Universal Render Pipeline" prosjekt. Grunnen til det er at URP gjør det enkelt å lage optimalisert grafikk på tvers av plattformer.

Unity lifecycle - monobehaviour class

Monobehaviour class er basisklassen som alle Unity-script kommer fra. Ved oppretting av C# - script gjennom Unity sitt prosjektvindu vil det automatisk arve fra denne klassen. Den tilbyr metoder som gjør det enkelt å knytte script til objekter og hendelser i et spill. Det er blitt brukt følgende funksjoner fra monobehaviour klassen i dette prosjektet:

Start() - denne metoden blir kjørt én gang når scriptet starter. Her er det vanlig å initialisere tilstanden til scriptet og eventuelle objekter eller variabler.

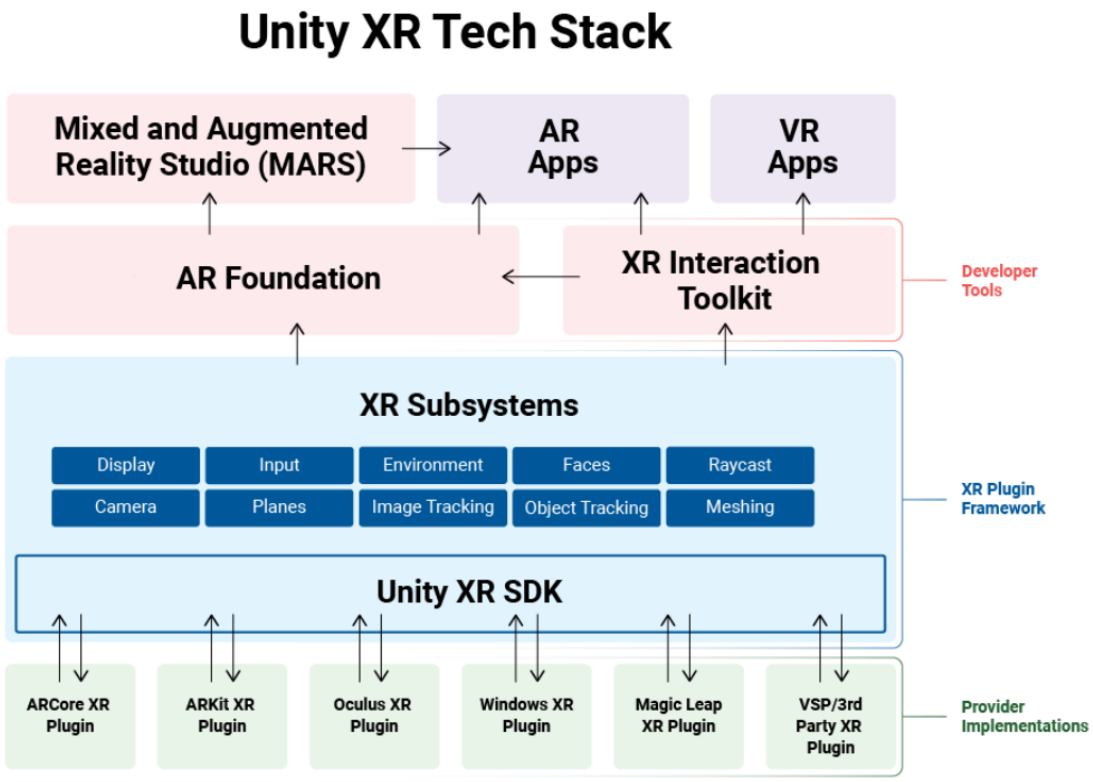
Update() - det som blir skrevet i update blir kalt hver gang skjermbildet blir oppdatert. I prosjektet blir update brukt til å spore posisjon, tid og distanse til pasienten.

OnTriggerEnter(Collider) - denne metoden kalles når et kollisjonsobjekt kolliderer med et utløserobjekt. I dette prosjektet blir metoden kalt når valgt kontroller kolliderer med en usynlig kube som fungerer som en startvegg.

OnTriggerStay(Collider) - det som står i denne metoden blir kalt hver gang skjermbildet oppdateres etter at kollisjonsobjektet har kollidert med utløserobjektet.

OnTriggerExit(Collider) - Denne metoden kalles etter OnTriggerEnter(Collider) når kollisjonsobjektet forlater utløserobjektet. I dette prosjektet blir utløserobjektet kuben brukt som en stoppvegg også.

Unity XR Plug-in Framework (Figur 7)
Rammeverk som åpner for direkte integrering for flere plattformer, som PC-er, konsoller, mobile enheter, nettsider og VR. Et API tilbyr vanlige funksjoner på tvers av plattformene Unity støtter og det er mulig for XR-maskin- og programvaretilbydere å utvikle egne Unity plug-ins. Et eksempel på utviklingsverktøy som tilbys og er tatt i bruk i dette prosjektet er XR Interaction Toolkit.



Figur 7 Unity XR Plug-in Framework (Unity Technologies, 2021).

XR Interaction Toolkit

Et verktøy som gjør det lettere å implementere interaktivitet i VR uten kode. Det inneholder komponenter som kan gi objekter i Unity visse egenskaper uten at man må lage egne script eller laste ned noe fra eksterne biblioteker. Det gjør at utviklingen av XR-apper er raskere og mer fleksibel.

XR Rig Settings (Figur 8 og 9)

XR Rig er brukerens øyne, ører og hender i den virtuelle verden. Sammen med et bevegelses- og teleportering- system som tillater brukeren å bevege seg i den virtuelle verden.

Figur 8 Roon-Scale XR Rig Settings

Rig Base Game Object: Dette er rot-objektet til XR Rig-hierakiet. Som standard er det GameObject som er koblet til XR Rig.

Camera Floor Offset Object: Dette er objektet kameraet er forskjøvet fra.

Camera Game Object: Kameraet som blir brukt til å gjengi scenen.

Tracking Orgin Mode: Denne innstillingen bestemmer plasseringen av den virtuelle verdens sentrum.

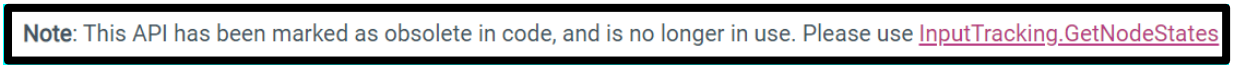
Figur 9 XR Rig Hierarchy

XR Rig tar kontroll over hovedkameraet (main camera) i scenen. Det blir plassert under kontroll av Camera Offset som plasserer kameraet i høyden til XR Rig GameObject, med mindre det er spesifisert en høyde i Camera Y Offset. Høyre og venstre kontroller er også inkludert.

XRNode

For å få tilgang til de ulike input-enhetene i et script brukes blant annet XR-noder. XR-noder representerer de fysiske punktene i XR-systemet. Brukers hodeposisjon, høyre og venstre hånd og andre sporingsreferanser som Oculus kamera er eksempler på XR-noder.

InputTracking.GetLocalPosition

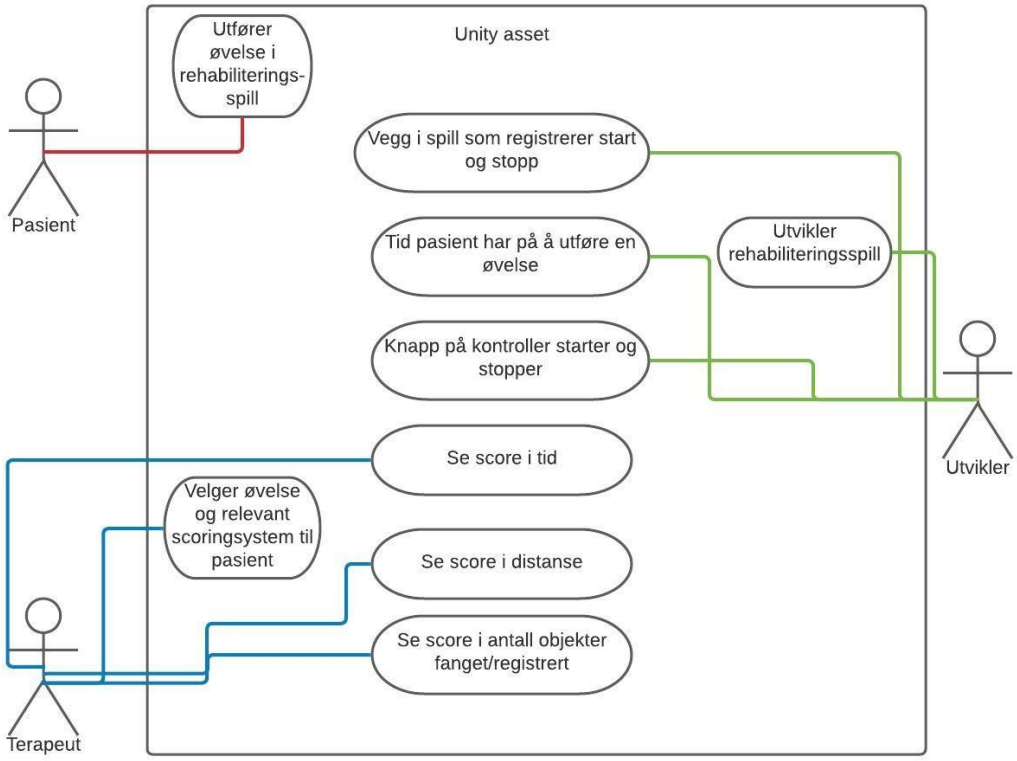


Figur 10 Unity InputTracking.GetLocalPosition statement

Gruppen valgte å fortsette å bruke metoden *InputTracking.GetLocalPosition* selv om den fases ut av Unity (Figur 10). Dette ble begrunnet med at det var lett å finne dokumentasjon på nettet, og mesteparten av bugs og feil luket vekk. Når dette APIet først var tatt i bruk ble det sett på som beste løsningen å fortsette med det, heller enn å gjøre endringer til nytt bibliotek. Det viktigste på dette tidspunktet var å komme i gang med utviklingen og så kunne en heller komme tilbake og gjøre endringer senere.

4.3.2 **Brukstilfeller**

Basert på den siste kravspesifikasjonen ble det kommet frem til følgende brukstilfeller, vist i en brukstilfellemodell (Figur 11).



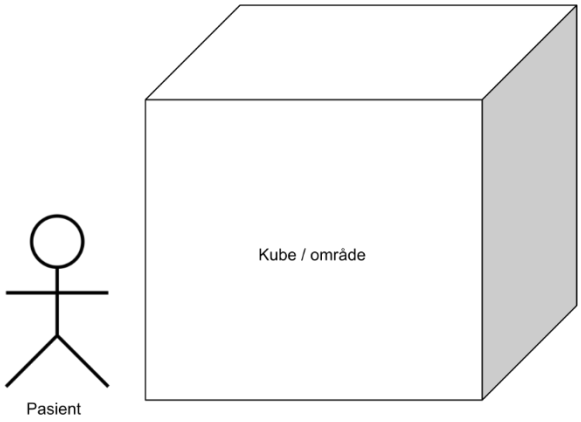
Figur 11 Brukstilfellediagram

Figur 11 viser hvordan de ulike rollene pasient, terapeut og utvikler vil ta i bruk asseten. Det er utvikler som laster ned asseten fra Unity for å så implementere det i rehabiliteringsspillet. Etter at utvikler har implementert de nødvendige metodene blir spillet gjennomført av pasienten. Terapeuten vil så kunne velge hvilket scoringssystem som er mest relevant for å måle progresjonen til pasienten.

4.3.3 Funksjonalitet

Unity-asseten består av funksjoner som er relevant for rehabilitering og opptrening av slagpasienter. Hver funksjon vil returnere data i form av tid, distanse eller antall objekter fanget. Dette er data som for terapeut er nyttig for å måle progresjon og nåværende tilstand til pasient. Asseten vil inneholde følgende funksjoner:

- Registrering av tid og distanse ved passering av vegg/kube. En kube vil blant annet kunne brukes til å definere et gyldighetsområde for øvelsen (Figur 12).



Figur 12 Illustrasjon av hvordan en kube i Unity, usynlig for pasient, kan brukes til å definere et område

- Registrering av tid og lengste distanse ved trykk av knapp på kontroller.
- Registrering av total distanse ved trykk av knapp på kontroller.
- Registrering av score, for eksempel antall objekter pasienten har fanget/hentet.

Målingene gjort i Unity vil samsvare med virkelige mål, kapittel 5 tar for seg en test som viser dette.

Asseten er bygd opp slik at en utvikler skal kunne velge hva som er nødvendig å bruke i sitt spill ut fra øvelsen pasienten skal utføre. Det er ikke nødvendigvis slik at terapeuten har bruk for all data som blir registrert, men da har utvikler muligheten til å sortere ut det som er irrelevant.

Asseten skal benyttes i rehabiliteringsspill i VR utviklet i Unity. Det er ikke sikkert at asseten vil fungere på andre VR-plattformer, siden det i utviklingsfasen kun har blitt testet på Oculus Quest VR-briller. Dette var etter avtale med oppdragsgiver.

4.3.4 Grafisk grensesnitt

Grafisk grensesnitt blir definert gjennom tre ulike funksjoner avhengig av hva som er mest relevant for øvelsen som rehabiliteringsspillet skal baseres på (Figur 13).

Kube – colliders

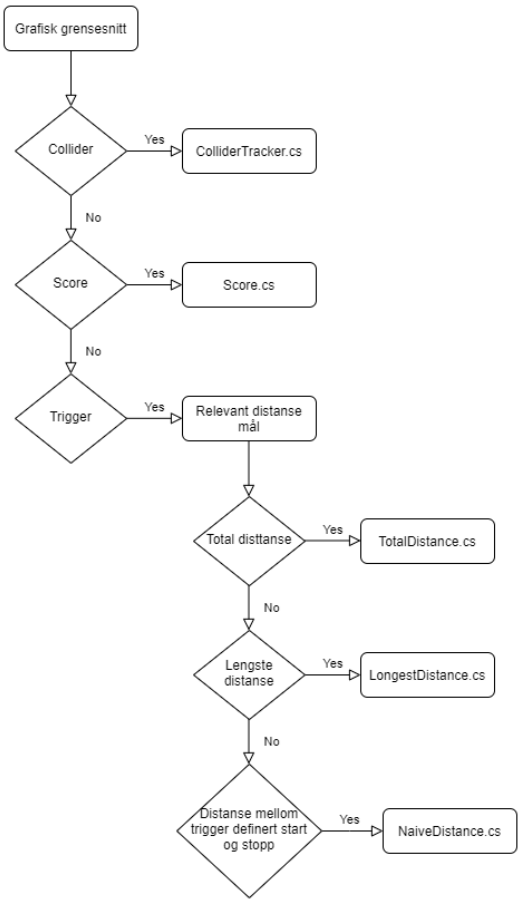
I kravspesifikasjonen fra oppdragiver var det ønskelig med en vegg som kunne registrere start og stopp i en øvelse. Denne veggen skulle kun være synlig for utvikler eller terapeut og kunne plasseres på valgfritt sted. Etter å ha undersøkt Unity og ulike metoder, ble det konkludert med å bruke en 3D-kube. Den vil da kunne plasseres, skaleres og roteres etter ønske fra utvikler. Kuben vil registrere start ved at pasienten entrer kuben og det settes i gang registrering av distanse og tid. Det vil i tillegg registreres antall ganger pasienten entrer og forlater kuben i form av en score. Registreringen vil så stoppe når pasienten forlater kuben. Om det kun ønskes start eller stopp, vil kubene kunne skaleres slik at de fungerer som en vegg.

Knapp på kontroller

En annen måte å registrere start og stopp på en oppgave er ved at pasienten trykker inn en knapp definert av utvikler. Registrering av tid og distanse vil da starte ved første trykk, og stoppe ved andre trykk. Pasienten vil kunne starte registreringen igjen ved å trykke en tredje gang. Det er også mulig å la registreringen gå fra første trykk til øvelsen avsluttes på andre måter.

Scoringssystem

Det tredje grensesnittet vil stoppe øvelsen dersom pasienten når gitt score. Score kan være gitt i tid, distanse eller antall objekter som fanges/registreres. Det vil være mulig å stoppe øvelsen når tiden er gått ut, når pasienten har oppnådd gitt distanse eller dersom pasienten har fanget gitt antall objekter. Antall objekter fanget er ment som et eksempel på hvordan en score kan defineres, men dersom utvikler har en annen scorefunksjon vil den lett kunne implementeres. Scoring vil fungere sammen med en av de to andre funksjonene nevnt tidligere når det kommer til å starte øvelsen.



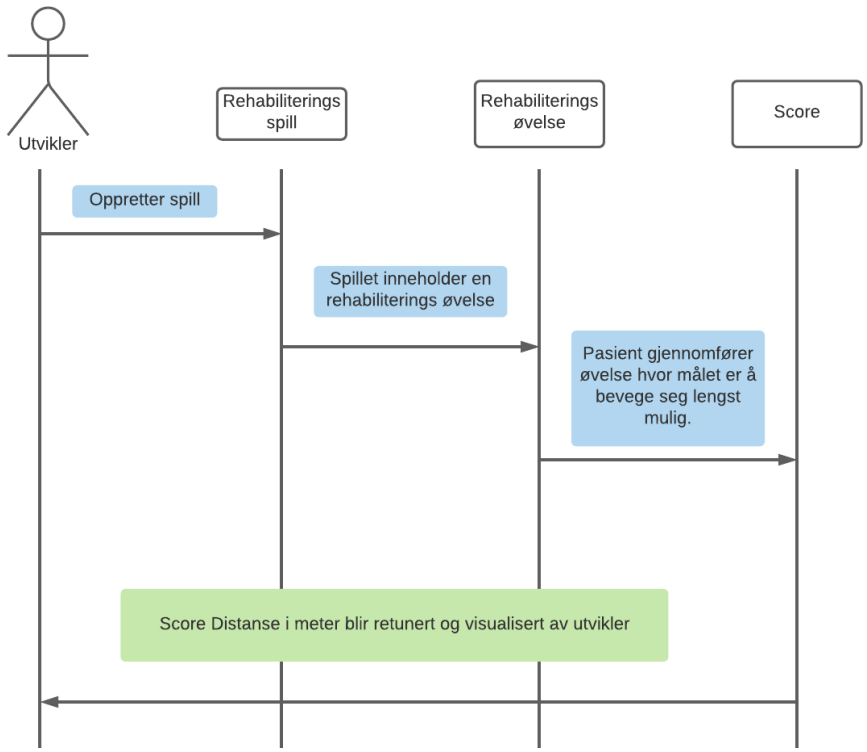
Figur 13 Kobling mellom klasser og de ulike grensesnittene

4.4 Use case

For å få en oversikt over hva asseten skal inneholde ble det delt inn i tre hovedbrukstilfeller, som også samsvarer med de tre ulike score-typene ønsket fra oppdragsgiver. De er laget med tanke på at en utvikler skal kunne bruke asseten i et rehabiliteringsspill og vurdere hvilken score som er mest relevant for visualisering for terapeuten. De tre forskjellige brukstilfellene blir beskrevet med sekvensdiagram i de kommende avsnittene.

Use case 1 - Distansen pasienten har beveget seg

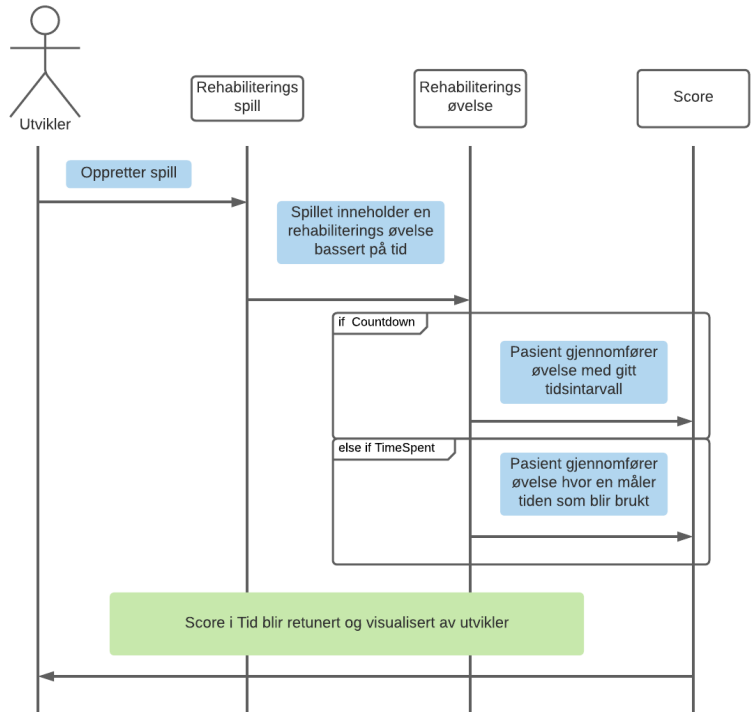
I use case 1 (Figur 14) blir det tatt utgangspunkt i at rehabiliteringspillet skal måle distansen pasienten har beveget seg. Dersom pasienten klarer å strekke seg eller bevege seg lenger enn det første registrerte, blir det lagret som lengste distanse til pasienten. Det er også det som vil bli returnert som pasienten sin score.



Figur 14 Use case 1

Use case 2 - Tiden pasienten har brukt eller har på å utføre øvelsen

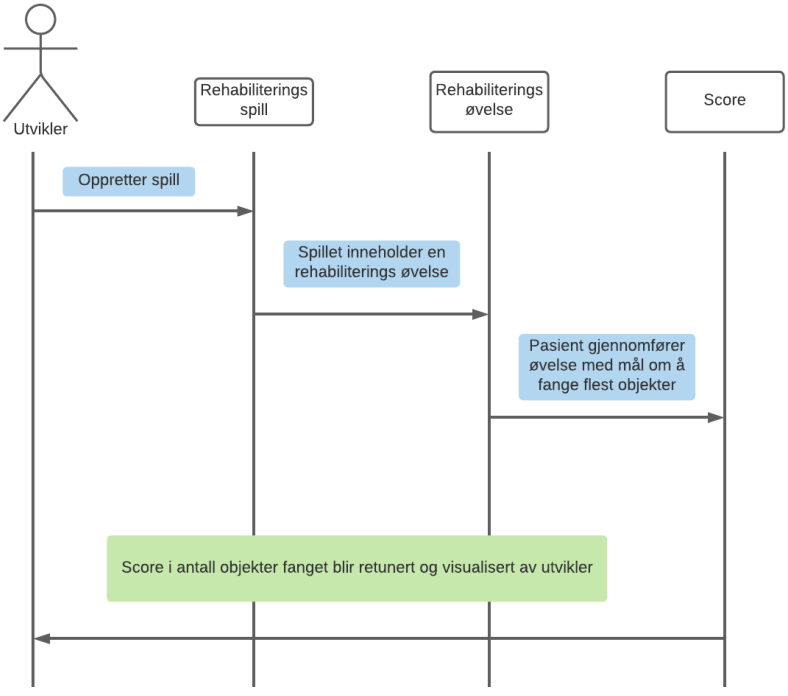
I use case 2 (Figur 15) er det tiden som bestemmer scoren til pasienten. Det er to muligheter å definere tiden på. Den første er å måle tiden mellom pasienten starter øvelsen til pasienten har fullført, altså total tid brukt på øvelsen.
Det vil også være mulig for utvikler å legge inn nedtelling, altså at pasienten har en gitt tid å gjennomføre øvelsen på. Scoren vil da bli definert ut fra om pasienten klarte å gjennomføre på tiden eller ikke.



Figur 15 Use case 2

Use case 3 - Antall objekter pasienten klarer å fange/hentet

I use case 3 (Figur 16) er det tatt utgangspunkt i at scoren defineres ut ifra hvor mange objekter pasienten klarer å fange eller hente. Det kan da for eksempel være terapeut som setter ut objekter som pasienten skal fange. Terapeuten vil kunne bestemme avstand mellom objekt og pasient og hvor mange objekter pasienten har som mål å fange/hente. Antall objekter vil også kunne bli lagt inn av utvikler ut fra vanskelighetsgrad.



Figur 16 Use case 3

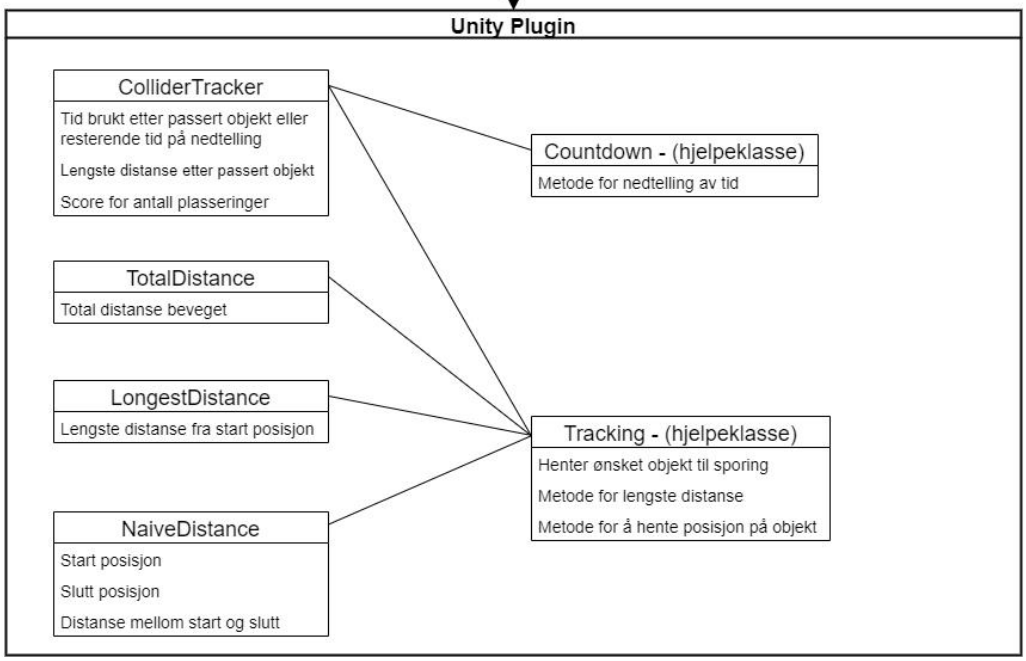
4.5 Klasser

Under utvikling er det laget flere script for registrering av input. Her følger en oversikt over ulike klasser (Figur 17) og bilder som fremhever viktige deler av koden.

(Facebook Technologies, LLC, 2021)

Oculus Quest - plattform og sporingsenhet

Unity - Utviklingsplattform (Microsoft.Net, 2021)



Figur 17 Oversikt over ulike klasser

ColliderTracker

I denne klassen vil det bli registret tid og lengste distanse pasienten har beveget seg etter å ha entret et gitt område. Registreringen av tid og distanse vil enten stoppe etter en gitt tid, ved å ha nådd en viss score eller ved å forlate dette området. Det vil være opp til utvikler av rehabiliteringsspillet å se hvilken måte som er mest relevant for øvelsen pasienten skal utføre.

I Unity blir det brukt komponenter som kalles colliders. Disse komponentene tillater at en fysikkmotor kan håndtere kollisjoner mellom to objekter. Fysikkmotoren simulerer kollisjoner ved hjelp av colliders. Det er da mulig å bestemme hvilke objekter som vil kollidere med hverandre og hvordan de vil oppføre seg under en kollisjon. I dette

prosjektet blir det brukt noe som kalles triggers, som er et spesielt oppsett for colliders, der hensikten er å utløse visse hendelser. I scriptet vist i kodesnutt 1 vil det utløses en sporingshendelse i form av tid og distanse for å registrere hvor lenge og hvor langt pasienten beveger seg inn i kuben. Metodene *OnTriggerEnter*, *OnTriggerStay* og *OnTriggerExit* ble også brukt for å oppdage når to objekter overlapper og for å starte hendelser.

```
60      private void OnTriggerEnter(Collider obj)
61      {
62
63          if (stopTyp == stopType.COUNTDOWN)
64          {
65              Debug.Log("Countdown started at " + countDown.targetTime);
66              countDown.start();
67          }
68          else
69          {
70              Debug.Log("Entered wall");
71              timer.Start();
72          }
73      }
74
75      private void OnTriggerStay(Collider other)
76      {
77          pos = t.returnTracked(tracedDevice);
78      }
79
80      private void OnTriggerExit(Collider other)
81      {
82          if (stopTyp == stopType.COUNTDOWN)
83          {
84              countDown.stop();
85          }
86          else if (stopTyp == stopType.SCORE)
87          {
88              score++;
89              Debug.Log("Exiting wall, you have gained a point!");
90              Debug.Log("Your current score is: " + score);
91          }
92          else
93          {
94              Debug.Log("Exiting wall");
95              timer.Stop();
96              Debug.Log("Time spent inside the object :" + timer.Elapsed.ToString("mm\\:ss\\.ff"));
97              Debug.Log("The longest distance moved past entering the wall :" + t.distance(pos));
98              pos.Clear();
99          }
100     }
```

Kodesnutt 1

TotalDistance

TotalDistance (Kodesnutt 2) registrer totaldistansen pasienten har beveget seg mellom første og andre gang pasienten trykker inn en knapp på kontrolleren. Den totale distansen samsvarer med virkelige mål (Se måletest i kapittel 5 for mer info).

LongestDistance

LongestDistance (Kodesnutt 2) registrerer lengste distansen pasienten har beveget seg fra første gang pasienten trykker på en knapp til øvelsen avsluttes Om pasienten ikke klarer å bevege seg lenger på andre forsøk vil ikke lengste distanse bli oppdatert. Et eksempel på bruk kan være å finne ut hvor langt pasient kan strekke seg fra et startpunkt gitt ved knappetrykk.

```
52      void Update()
53      {
54          inputDevice.TryGetFeatureValue(CommonUsages.triggerButton, out isPressed);
55
56          if (inputDevice != null && isPressed && !wasPressed)
57          {
58              wasPressed = true;
59          }
60
61          if (inputDevice != null && wasPressed)
62          {
63              pos = t.returnTracked(tracedDevice);
64              Debug.Log(t.distance(pos));
65          }
66
67      }
```

Kodesnutt 2

NaiveDistance

NaiveDistance (kodesnutt 3) registrerer distansen pasienten har beveget seg fra startposisjon til sluttposisjon som blir bestemt av at pasienten trykker inn en knapp.

```
60      void Update()
61      {
62          //Will track position of a given device while you press the trigger button on targetDevice
63          inputDevice.TryGetFeatureValue(CommonUsages.triggerButton, out isPressed);
64
65          if (inputDevice != null && isPressed && !wasPressed)
66          {
67              pos = t.returnTracked(tracedDevice);
68
69              wasPressed = true;
70          }
71
72
73          if (inputDevice != null && !isPressed && wasPressed)
74          {
75              if (counter % 2 == 0)
76              {
77                  wasPressed = false;
78                  Debug.Log("The start pos :" + t.returnPos(tracedDevice));
79                  counter++;
80              }
81              else
82              {
83                  wasPressed = false;
84                  Debug.Log("The stop positon:" + t.returnPos(tracedDevice));
85                  Debug.Log("The distance moved :" + System.Math.Round(t.distance(pos), 2));
86                  counter++;
87              }
88          }
89      }
```

Kodesnutt 3

TrackedDevice

TrackedDevice (Kodesnutt 4) er enum-klasse hvor det blir definert hvilket objekt som skal spores. I dette prosjektet har det kun blitt lagt inn hender og hode, men det er enkelt å legge inn andre elementer, for eksempel om en ønsker sporing av føtter.

```
6       public enum trackedDevice
7       {
8           HEAD,
9           RIGHTHAND,
10          LEFTHAND
11      }
```

Kodesnutt 4

Tracking

Tracking er en hjelpeklasse som foretar sporing på de ulike elementene som blir gitt i enumklassen. Her ligger flere av funksjonene som kalles i de andre klassene. Eksempler på slike funksjoner er *returnTracked* (Kodesnutt 5*)* og *distance* (Kodesnutt 6).

```
142     public List<Vector3> returnTracked(trackedDevice device)
143     {
144         switch (device)
145         {
146             case trackedDevice.HEAD:
147                 pos.Add(InputTracking.GetLocalPosition(XRNode.Head));
148                 return pos;
149
150
151             case trackedDevice.LEFTHAND:
152                 pos.Add(InputTracking.GetLocalPosition(XRNode.LeftHand));
153                 return pos;
154
155
156             default:
157                 pos.Add(InputTracking.GetLocalPosition(XRNode.RightHand));
158                 return pos;
159
160
161         }
162
163     }
```

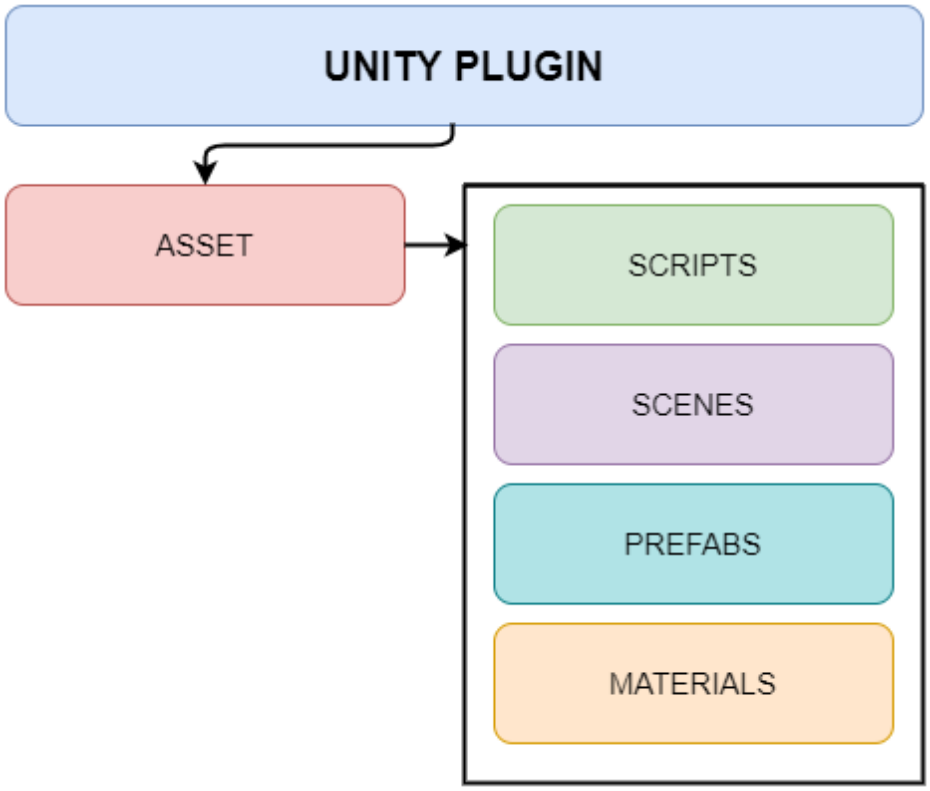
Kodesnutt 5

```
//Calculates the longest distance traveled by using the pos List
public float distance(List<Vector3> pos)
{
    Vector3 first = pos.First();
    float largest = 0;

    foreach (Vector3 vector in pos)
    {
        if (Vector3.Distance(vector, first) > largest)
        {
            largest = Vector3.Distance(vector, first);
        }
    }
    return largest;
}
```

Kodesnutt 6

4.6 Mappestrukturering av Unity plug-in



Figur 18 Unity plugin stucture

Unity plug-in (Figur 18) inneholder følgende mapper:

Asset: Denne mappen inneholder alle assetene som blir brukt i Unity-prosjektet.

Scripts: Inneholder alle scriptene som ble nevnt i kapittel 4.5.

Scenes: Inneholder ulike scener som igjen inneholder ulikt innhold. For eksempel kan en scene inneholde et spill. Dette prosjektet har en egen scene.

Prefabs: Inneholder lagrede spillobjekter med alle komponenter, verdier og eventuelle underordnede spillobjekter som har blitt opprettet i Unity. I dette prosjektet inneholder mappen en venstrehånd, høyrehånd og en kube til ColliderTracker-klassen.

Materials: Dette er en mappe som inneholder ulike materials som inneholder data, som farger eller referanser til tekststrukturer. For eksempel kan et spillobjekt som skal brukes til gress få tildelt et material med fargen grønn. I dette prosjektet har ikke materials vært relevant, siden prosjektet baserer seg på å utvikle metoder (backend) og ikke det visuelle (frontend).

5 EVALUERING

5.1 Evalueringsmetode

Under utviklingsperioden har det blitt utført funksjonell testing etter hvert som nye funksjoner har blitt implementert. Dette har blant annet blitt gjort ved å skrive ut informasjon til konsollen i Unity gjennom *Debug.Log()* mens VR-kontrollere ble beveget på for å teste nye funksjoner.

Som tidligere nevnt har det vært endringer i kravspesifikasjonen underveis. Kontinuerlig evaluering, samtaler med oppdragsgiver og testing av hva som faktisk er mulig i Unity har gjort det mulig å komme frem til mer spesifiserte krav.

Det ble planlagt at de ulike funksjonalitetene i produktet skulle testes ved å sammenligne tallverdier, altså ved hjelp av en kvantitativ metode. En av de mest sentrale funksjonene er målingen av distanse. Dette er noe som raskt lar seg teste ved hjelp av å sammenligne distansen i Unity og distanse tilbakelagt i rommet med et målebånd (Tabell 2). For å kunne vurdere kvaliteten av resultatene er det viktig å avklare med oppdragsgiver hvor store avvik som kan tolereres.

For å måle nøyaktigheten til håndsporingen, føres håndkontrolleren langs et målebånd (Figur 19). Midten av kontrolleren markerer starten. Under måling var det eneste avviket i målingen av total distanse, men dette kan skyldes at denne funksjonen er svært sensitiv og måler alle bevegelser, også om man skjelver litt på hånden.

Figur 19 Oppsett for måletest

Funksjon	Resultat i Unity, utført med stående grense	Resultat målebånd/tidtaker
Longest Distance Største distanse fra startpunkt	Hånd: 10 cm Hode: 10 cm	Hånd: 10 cm Hånd: 10 cm
Total Distance Total distanse tilbakelagt under øvelse	Hånd: 22 cm	Hånd: 20 cm
Naive Distance Avstand fra start til stopp	Hånd: 51 cm Hode: 50 cm	Hånd: 51 cm Hode: 50 cm
Collider Tracker - Tid brukt innenfor grenseområde - Nedtelling innenfor område	10 sek	10 sek

Tabell 2 Måletest

En mer kvalitativ tilnærming er nødvendig for å måle produktets brukervennlighet.

Målet er at en utvikler skal kunne ta i bruk asseten med relativt enkle grep etter å ha lest brukermanualen. Det ideelle vil være å få medstudenter eller andre som holder på med rehabilitering i VR i Unity til å teste asseten i sitt spill. Dette avhenger imidlertid av at asseten er kompatibel med spillet, noe som på grunn av begrenset med tid til utvikling ikke kan garanteres. Derfor ble det planlagt å lage et enkelt lite spill, slik at kompatibiliteten ikke kommer i veien for testing. Da kan oppdragsgiver eller medstudenter teste bruk av asset lokalt på gruppen sin datamaskin på masterlab.

Gruppen har på grunn av corona-situasjonen hatt begrenset tilgang på relevante testere. Det fremstod derfor at en god løsning ville være å vise frem en enkel demo som viser generell funksjonalitet til Lars Peder. Dette er på bakgrunn av hans erfaring innenfor fagområdet og rehabiliteringsprosessen som asseten skal

brukes i. I tillegg til å ha god kompetanse på fagområdet er han også oppdragsgiver, som vil si at denne testingen også ga oss en god mulighet til å vise frem det endelige sluttproduktet til oppdragsgiver. Testen kunne også vært brukt med flere terapeuter dersom det hadde vært mulig å avvikle.

5.2 Evalueringsresultat

5.2.1 Første demo for oppdragsgiver (11 mai)

Gruppen inviterte Lars Peder til masterlab for en demo av en prototype av asseten. Dette var et minimal viable product (MVP) for å kunne vise hvordan de ulike funksjonene fungerte. Dette var nyttig for å finne ut om det virker slik som oppdragsgiver hadde sett for seg.

Det ble gitt gode tilbakemeldinger på de funksjonene som var implementert og vi forsikret oss om at vi var på rett spor. Score var under demoen ikke ferdigstilt, men dette var planlagt fra gruppen sin side. Det ble etter demoen snakket om hvordan score skulle fungere slik at det ble tilpasset oppdragsgivers ønsker

5.2.2 Test av ferdig asset

Lars Peder ble igjen invitert på masterlab og fikk en demonstrasjon av en mer ferdig asset. Etter dette ble han tilsendt et skjema med påstandene nevnt i kapittel 3.
Under følger hans svar og eventuelle kommentarer.

Evalueringspunkter (Sterkt uenig (1) – Sterkt enig (5)):

1. **Produktets målefunksjoner for distanse fungerer på en tilfredsstillende måte.**
   4
   5 forutsatt at distanse blir korrekt på den som var feil på demonstrasjonstidspunkt.
2. **Produktets målefunksjoner for distanse gir svar med tilfredsstillende presisjon.**
   5
3. **Produktets målefunksjoner for tid fungerer på en tilfredsstillende måte.**
   5
4. **Produktets målefunksjoner for tid gir svar med tilfredsstillende presisjon.**
   3
   Nøytralt svar da jeg ikke har sett testing på dette.
5. **Produktet har funksjonalitet som er til forventet nivå.**
   5

6. **Produktet oppfyller mine forventninger.**
   5
7. **Produktet er noen jeg kan se for meg å ta i bruk.**
   4
   Ikke jeg personlig, men ta med i aktuelle prosjekter.
8. **Jeg er totalt sett fornøyd med det endelige sluttproduktet.**
   4

Tilleggsspørsmål

Dersom prosjektet skulle startes på nytt med din nåværende erfaring, hvilke forandringer ville du gjort?

Ville vært tydeligere på rammer og definerte formål tidligere i prosessen, samt innhentet veilederkompetanse på utvikling av lignende oppsett i Unity.

Ved videre arbeid på prosjektet, hva ville vært dine topp tre ønsker over nye funksjonaliteter?

1. *Tilknytning til et utvalg med måleverktøy.*
2. *Generisk kobling til tallbaserte måleparametre (generalisering av 1).*
3. *Dynamiske UI som kan integreres i spill med drag and drop.*

Kommentar til evalueringsresultat

Disse funksjonene har blitt implementert i det ferdige produktet:

- Måling av direkte distanse fra punkt A til B
- Måling av total distanse pasienten har beveget seg fra punkt A til B
- Måling av den lengste distansen pasienten har beveget seg fra startpunkt
- Måling av diverse egenskaper relatert til pasients samhandling med et gitt objekt
    - Hvor lang tid har en pasient tilbrakt inne i objektet
    - Om pasient har brukt mer enn gitt tidsintervall inne i objektet
    - Hvor langt har pasient beveget seg inne i objektet
    - Hvilken score har pasienten klart å oppnå inne i objektet

Ved slutten av prosjektet virker oppdragsgiver og gruppen å dele oppfatningen av hva som er oppnådd av resultater. I tilbakemeldingen fra oppdragsgiver svarer han at assetens målefunksjoner for distanse, som er de mest sentrale funksjonene, fungerer på en svært tilfredsstillende måte. Da testen ble gjennomført var det en liten bug i den ene målefunksjonen, men dette ble raskt fikset, slik at disse funksjonene ville fått høyeste score mulig fra oppdragsgiver. Han sier også at han kan se for seg at andre utviklere kan ta i bruk produktet, noe som er selve målet med asseten. Videre kommenterer han ønsker for videre arbeid, deriblant kobling til annet måleverktøy, dette er noe som vil utdypes i kapittel 7.3

6 DISKUSJON

Dette kapittelet vil diskutere arbeidet som er blitt gjort i prosjektet, måloppnåelsen av prosjektets problemstilling og hvilket videre arbeid som kan gjøres.

6.1 Fremgangsmåte

I prosjektets startfase valgte gruppen å ta i bruk et Gantt-diagram for å hjelpe med planleggingen av prosjektet. Diagrammet bestod av alle arbeidsoppgavene som måtte gjennomføres, og hver av disse oppgavene ble tildelt et eget tidsintervall for fullføring. Etter hvert som gruppen fullførte oppgaver ble diagrammet oppdatert for å reflektere prosjektets progresjon. Gruppen møtte på enkelte utfordringer med å holde disse angitte tidsintervallene. Problemene kan i stor del spores tilbake til gruppens manglende erfaring innen utvikling i Unity miljøet og C# som programmeringsspråk. Dette gjorde at mye av prosjektets tidlige fremdrift måtte settes på vent, og tiden som var satt av til utvikling gikk i stedet til opplæring i programvaren. Dette førte igjen til at tidsfristene som ble satt i Gantt-diagrammet måtte tilpasses. Gruppen kunne ha unngått dette problemet ved å ha mer fokus på å sette seg inn i programvaren før prosjektet startet.

Gruppen hadde også noen utfordringer knyttet til valg av hvilke XR-bibliotek som skulle brukes. Et XR-bibliotek er en verktøykasse bestående av mange forskjellige funksjoner og egenskaper relatert til kartlegging og bruken av VR-enheter. Gruppen valgte å gå for Unity sitt innebygde XR-bibliotek ettersom at dette var godt dokumentert. Ulempen med dette valget var at Unity på dette tidspunktet holdt på å lage en fullstendig ny implementasjon av biblioteket, som vil si at noen av løsningene prosjektet bygger på vil bli faset ut når det nye XR-biblioteket er ferdigstilt. Igjen kunne dette problemet kanskje vært unngått ved å legge inn mer tid i forarbeidet før prosjektet startet. Mot slutten av prosjektet fant gruppen også et alternativ til Unity XR, kalt OpenXR. OpenXR er et open-source prosjekt som er tilpasset utvikling på blant annet Oculus plattformen, og som er under løpende utvikling og vedlikehold.

6.2 Konsekvens av fremgangsmåte

Dersom prosjektet hadde startet i dag, med den erfaringen og kunnskapen vi nå har opparbeidet oss, ville vi ha kunnet vise til konkrete resultater på et tidligere tidspunkt i utviklingen. Det ville da mest sannsynlig også ha blitt implementert en fullt fungerende databaseløsning, som nå er blitt utelatt grunnet manglende tid. I tillegg hadde gruppen tatt en ny vurdering på hvilke XR-bibliotek som skulle brukes, og ville da mest sannsynlig ha valgt å bruke OpenXR biblioteket. Dette kunne totalt sett gitt et bedre og allsidig produkt.

I ettertid er det lett å si at det ville vært en fordel om vi hadde satt oss enda mer inn i verktøyene vi skulle bruke før prosjektstart. I dette tilfellet er det likevel ikke sikkert at det hadde gjort selve utviklingsprosessen enklere da det var vanskelig å vite på forhånd hvilke biblioteker og programvare som faktisk ville være relevant. I tillegg er det noe manglende dokumentasjon på biblioteker som faktisk har blitt tatt i bruk, slik at den beste tilnærmingsmetoden har vært prøving og feiling i utviklingsfasen av prosjektet.

7 KONKLUSJON OG VIDERE ARBEID

7.1 Måloppnåelse

Gruppen har utviklet en asset som inneholder funksjoner for å hente ut distanse, tid og score fra et VR-spill. Målet for prosjektet var å utvikle en løsning som ville gjøre det enklere for fremtidige utviklere å implementere metoder som kan registrere brukerinput fra øvelser i VR. Dette vil gjøre det lettere og raskere å utvikle nyttig programvare, som så kan brukes til å kartlegge pasients prestasjon på diverse øvelser. Ved bruk av asseten i sitt VR-spill vil nå en utvikler ha mulighet til å registrere hvor mye en pasient har beveget seg, hvor lang tid han/hun har brukt eller få en score.

På grunn av liten tid har ikke gruppen implementert all funksjonalitet som oppdragsgiver ønsket fra produktet, på grunn av manglende tid. Likevel inneholder asseten hovedfunksjonene beskrevet av oppdragsgiver, og dette danner et godt grunnlag for å bygge videre og utvide asseten.

7.2 Begrensninger

Gruppens asset har i hovedsak to store potensielle begrensninger. Den første begrensninger er den langsiktige holdbarheten til asseten. Dette er på bakgrunn av at gruppen gjorde et lite gunstig valg når det kom til hvilke XR-bibliotek asseten skulle bygges på. I stedet for å velge Unity sitt eget XR-bibliotek som holder på å bli faset ut, burde vi i stedet ha valgt å basere prosjektet på OpenXR bibliotektet. Dette valget kan potensielt skape store begrensinger når det kommer til assetens langsiktige bruk, og det er også av den grunn uvisst hvordan koden vil samhandle med fremtidige versjoner av Unity og Oculus Quest plattformen.

Et annet potensielt problem er den mulige mangelen av brukbarhet på andre VR-plattformer. Dette er på bakgrunn av gruppens manglende tid til å teste asseten på andre VR-plattformer enn Oculus Quest, men det er ikke sikkert at dette er en relevant begrensning. Dessverre er ikke dette mulig å fastslå uten den nødvendige testingen.

7.3 Implikasjoner og videre arbeid

Prosjektets manglende database-implementasjon, endring av XR-bibliotek og utvidet funksjonalitet danner et godt grunnlag for videre arbeid. Første prioritet ved videre utvikling burde være database-implementasjonen. Både på kortsiktig og langsiktig bruk vil muligheten til å kartlegge pasienters ytelse gi stor verdi til assetens funksjonalitet. En slik løsning åpner i tillegg opp for å danne større datasett som potensielt kan brukes til å lære opp maskinlæringsalgoritmer som videre kan assistere terapeuter i behandlingsprosessen.

Videre ville et smart valg vært å fokusere på å fase ut Unity XR-biblioteket til fordel for OpenXR. Dette ville gitt asseten en lenger levetid før videre vedlikehold hadde vært nødvendig. I tillegg ville dette gjort asseten mer fremtidssikret og gjøre det lettere å implementere andre assets som er utviklet for OpenXR. I dette tilfellet ville det mest sannsynlig være mest hensiktsmessig å ta en ny vurdering av hvilke metoder og biblioteker som er mest relevant å bruke i asseten.

Videre arbeid etter dette punktet vil være mer fokusert på hvordan man potensielt kan introdusere nye former for input-enheter, som for eksempel KFORCE balansebrett. På dette punktet er det mange potensielle muligheter, og dette åpner også opp for å utvide assetens funksjonalitet til å være kjørbar på andre VR-plattformer som HTC Vive.

Asseten vil ifølge oppdragsgiver kunne brukes innen ulike former for rehabilitering i VR. Han trekker blant annet frem at den kan brukes av utviklere som lager spill for rehabilitering av barn og unge der poengsum vil være en motiverende faktor. Den vil også være nyttig i generell rehabilitering som viser om det er fremgang eller ikke.

En mulig bacheloroppgave for neste års studenter kan være å bygge videre på asseten og koble til flere input-enheter.

8 REFERANSER

Thomassen, Lars. (2020) *hjerneslag* i *Store medisinske leksikon* på snl.no. Tilgjengelig fra: https://sml.snl.no/hjerneslag (Hentet 14. mai 2021).

Granevang, Merethe: backend i Store norske leksikon på snl.no. Tilgjengelig fra: https://snl.no/backend og: frontend i Store norske leksikon på snl.no. Tilgjengelig fra: https://snl.no/frontend (Hentet 3. juni 2021).

Kinvent (2021) *KFORCE Plates*. Tilgjengelig fra: https://k-invent.com/k-force-plates/

Norsk hjerneslagsregister, Årsrapport for 2015. Trondheim: St. Olavs Hospital HF

Helsedirektoraet (2020) *Hjerneslag Nasjonal faglig retningslinje*. Tilgjengelig fra: https://www.helsedirektoratet.no/retningslinjer/hjerneslag (Hentet 14. mai 2021)

Bohil, C. J., Alicea, B., & Biocca, F. A. (2011). Virtual reality in neuroscience research and therapy. Nature Reviews. Neuroscience, 12, 752–762.

Rose, F. D., Brooks, B. M., & Rizzo, A. A. (2005). Virtual reality in brain damage rehabilitation: Review. CyberPsychology and Behaviour, 8, 241–271.

Valved, L.S, Bleikli, B.M. og Tellevik, M. (2018) *VR Walk Vandring i VR* Bacheloroppgave. Bergen: Høgskulen på Vestlandet.

Weiss, P.L, Keshner, E.A. og Levin, M.F. (2014) *Virtual Reality for Physical and Motor Rehabilitation*. New York: Springer.

Brooke, J. (1986). *SUS: a "quick and dirty" usability scale.* P. W. Jordan; B. Thomas; B. A. Weerdmeester; A. L. McClelland (eds.). *Usability Evaluation in Industry*. London: Taylor and Francis.

Unity Technologies (2020) *Unity Scripting Reference*. Tilgjengelig fra:

https://docs.unity3d.com/ScriptReference

Unity Technologies (2020) *Unity User Manual 2020.3*. Tilgjengelig fra: https://docs.unity3d.com/Manual

Bilder og figurer

Unity Technologies (2020) *XR Plug-in Framework*. Tilgjengelig fra: https://docs.unity3d.com/Manual/XRPluginArchitecture.html (Hentet 21. mai 2021).

Facebook Technologies, LLC (2021) *Oculus Quest 64GB VR Headset*. Tilgjengelig fra:

https://www.amazon.co.uk/Oculus-Quest-All-Gaming-Headset/dp/B07P6RJ39C (Hentet 01. juni 2021).

Microsoft.Net (2021) *Unity*. Tilgjengelig fra:

https://dotnet.microsoft.com/apps/games/unity (Hentet 01. juni 2021).

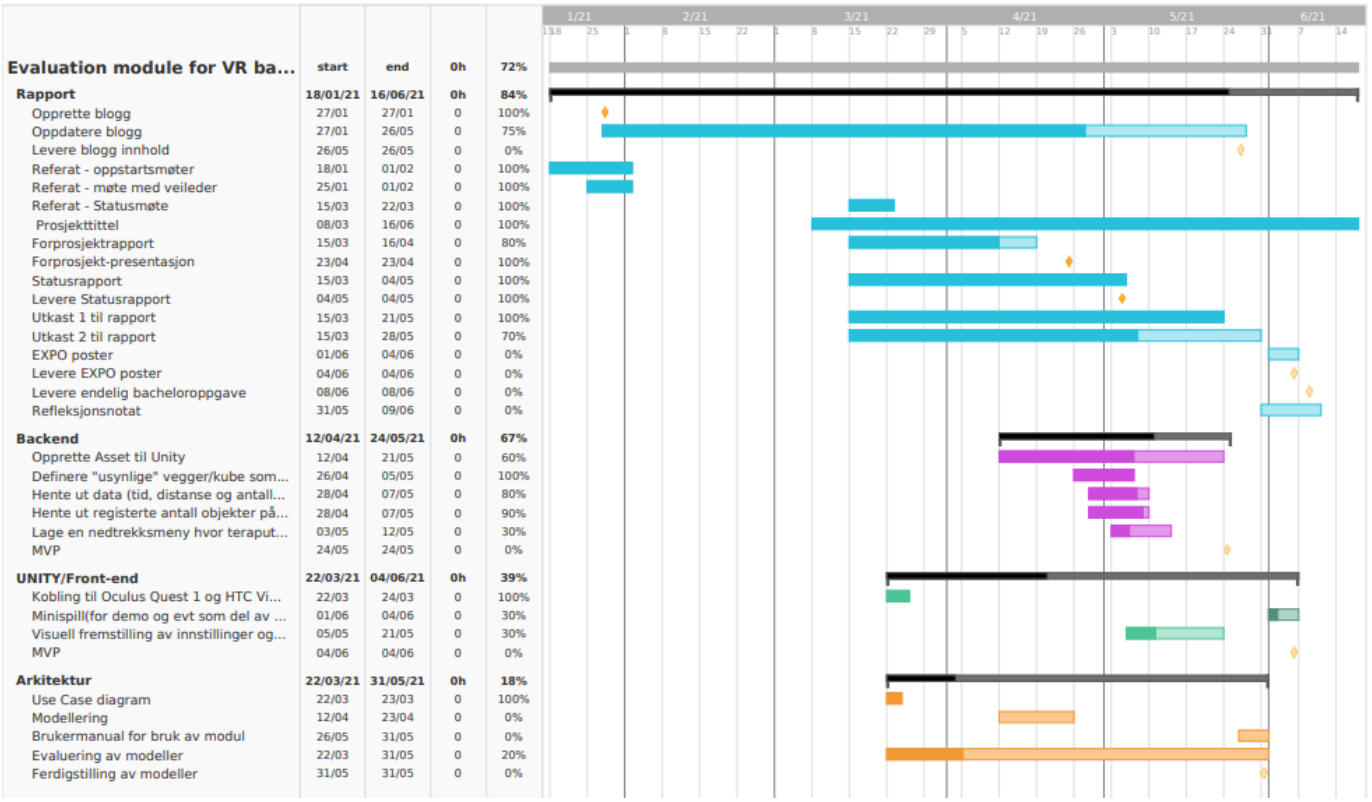
10 VEDLEGG

10.1 Risikoanalyse

Risiko	S	K	RF	Tiltak
Corona	5	2	10	Sørge for at vi har utstyr tilgjengelig og det er mulig å jobbe med prosjektet selv om vi ikke kan møtes på campus. Opprettholde kontakt gjennom Zoom.
For høye forventninger hos oppdragsgiver	1	4	4	Viktig med god kommunikasjon og oppdatere hverandre ofte
Misforståelser mellom gruppen og oppdragsgiver/veileder	2	3	6	Hyppige møter og skrive møtereferater
Blir ikke ferdig eller arbeidsmengden er for stor	2	5	10	Følge prosjektplanen og jobbe kontinuerlig gjennom semesteret
Teknisk svikt	1	4	4	Tidlig gjøre seg kjent med utstyr og programvare, lagre hyppig, ha alternativer.

S = sannsynlighet
K = konsekvens
RF = risikofaktor

10.2 GANTT-diagram



Evaluation module for VR ba...	start	end	0h	72%
Rapport	**18/01/21**	**16/06/21**	**0h**	**84%**
Opprette blogg	27/01	27/01	0	100%
Oppdatere blogg	27/01	26/05	0	75%
Levere blogg innhold	26/05	26/05	0	0%
Referat - oppstartsmøter	18/01	01/02	0	100%
Referat - møte med veileder	25/01	01/02	0	100%
Referat - Statusmøte	15/03	22/03	0	100%
Prosjekttittel	08/03	16/06	0	100%
Forprosjektrapport	15/03	16/04	0	80%
Forprosjekt-presentasjon	23/04	23/04	0	100%
Statusrapport	15/03	04/05	0	100%
Levere Statusrapport	04/05	04/05	0	100%
Utkast 1 til rapport	15/03	21/05	0	100%
Utkast 2 til rapport	15/03	28/05	0	70%
EXPO poster	01/06	04/06	0	0%
Levere EXPO poster	04/06	04/06	0	0%
Levere endelig bacheloroppgave	08/06	08/06	0	0%
Refleksjonsnotat	31/05	09/06	0	0%
Backend	**12/04/21**	**24/05/21**	**0h**	**67%**
Opprette Asset til Unity	12/04	21/05	0	60%
Definere "usynlige" vegger/kube som...	26/04	05/05	0	100%
Hente ut data (tid, distanse og antall...	28/04	07/05	0	80%
Hente ut registerte antall objekter på...	28/04	07/05	0	90%
Lage en nedtrekksmeny hvor teraput...	03/05	12/05	0	30%
MVP	24/05	24/05	0	0%
UNITY/Front-end	**22/03/21**	**04/06/21**	**0h**	**39%**
Kobling til Oculus Quest 1 og HTC Vi...	22/03	24/03	0	100%
Minispill(for demo og evt som del av ...	01/06	04/06	0	30%
Visuell fremstilling av innstillinger og...	05/05	21/05	0	30%
MVP	04/06	04/06	0	0%
Arkitektur	**22/03/21**	**31/05/21**	**0h**	**18%**
Use Case diagram	22/03	23/03	0	100%
Modellering	12/04	23/04	0	0%
Brukermanual for bruk av modul	26/05	31/05	0	0%
Evaluering av modeller	22/03	31/05	0	20%
Ferdigstilling av modeller	31/05	31/05	0	0%

10.3 Brukermanual

Laste ned asset
Add to my assets
Download

Installering i Unity

Gjennom Unity Engine
Etter nedlasting vil **Download** erstattes av **Import**. Ved å trykke her importeres asseten inn i ditt åpne Unity Project. En sjekkliste lar deg velge hvilke filer som skal importeres. Unity vil laste filene inn i Assets-mappen.

Forklaring av klasser

Naive Distance
Registrerer distansen pasienten har beveget seg fra startposisjon til sluttposisjon som blir bestemt av at pasienten trykker inn en knapp.

Longest Distance
Registrerer lengste distansen pasienten har beveget seg fra første gang pasienten trykker på en knapp til øvelsen avsluttes. Om pasienten ikke klarer å bevege seg lenger på andre forsøk vil ikke lengste distanse bli oppdatert. Et eksempel på bruk kan være å finne ut hvor langt pasient kan strekke seg fra et startpunkt gitt ved knappetrykk.

Total Distance
Registrer den totale distansen pasienten har beveget seg mellom første og andre gang pasienten trykker inn en knapp på kontrolleren. Den totale distansen samsvarer med virkelige mål.

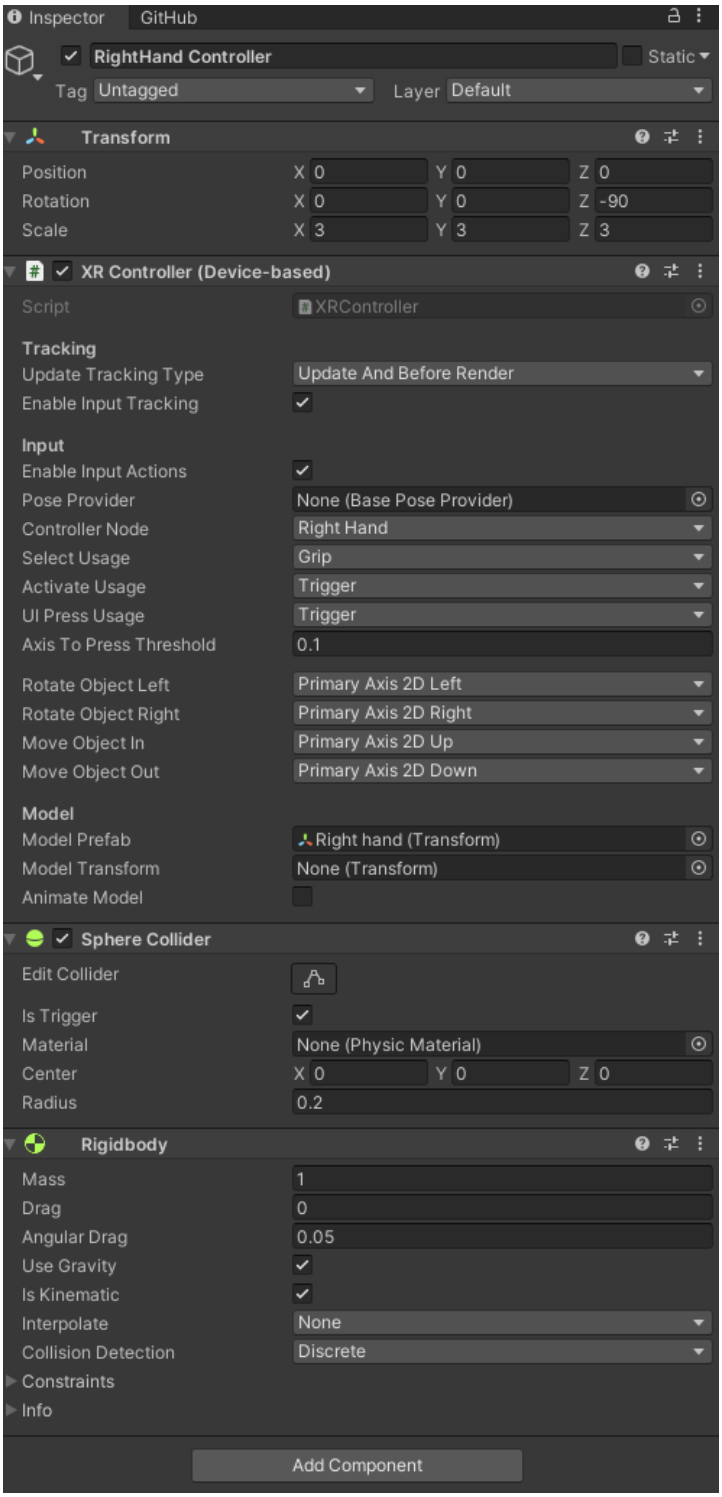
Instillinger

Spillområde/vaktsystem på Oculus Quest bør være i romskala for best resultat

Instillinger for håndkontroller

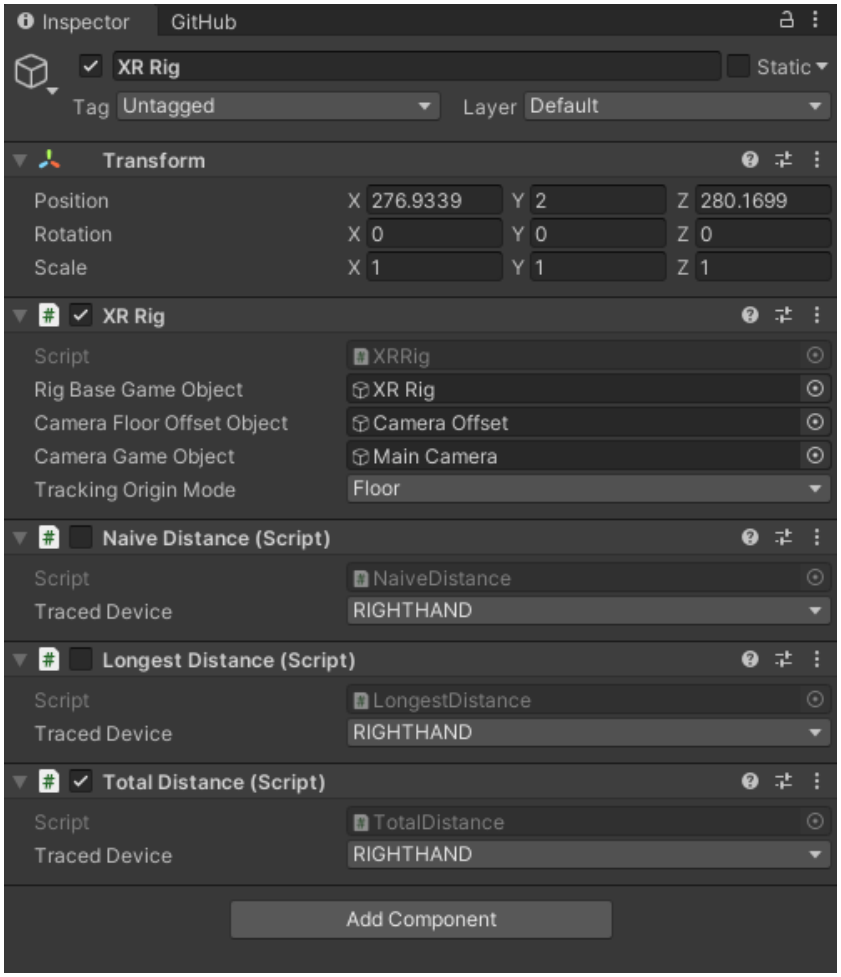
Sphere Collider
Is Trigger

Rigidbody
Is Kinematic
Use Gravity



Longest Distance, Naive Distance, Total Distance
Legg til script en vil bruke i inspector (dra inn fra Assets eller **Add component**)
Om man legger til flere script kan man velge hvilke som kan brukes ved å hake av

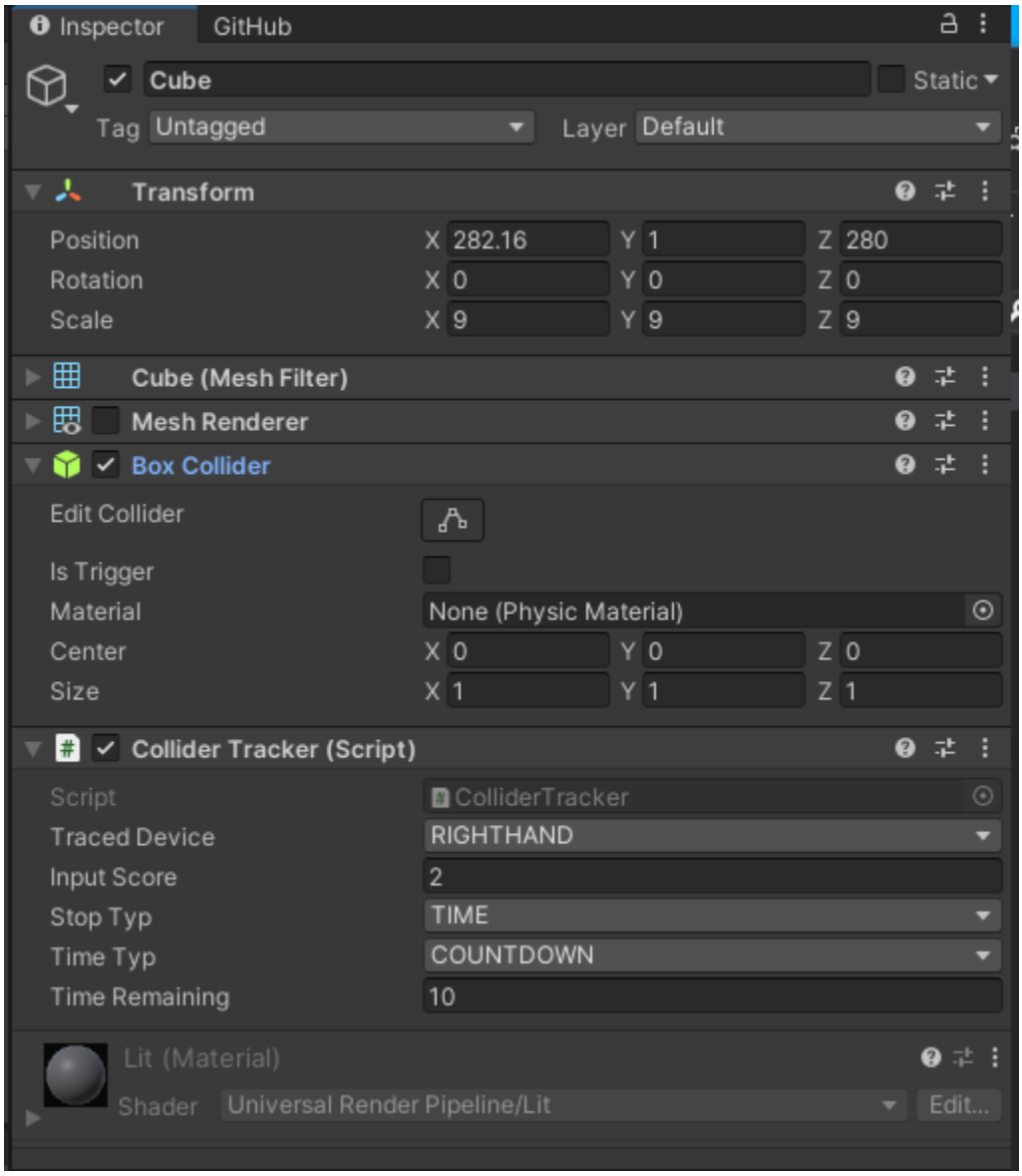
Valg i Inspector
Traced Device: Head, right hand, left hand. Hvilken input-enhet som skal spores.



ColliderTracker
Registrerer tid og lengste distanse pasienten har beveget seg etter å ha passert vegg/kube. Registreringen av tid og distanse vil enten stoppe etter en gitt tid, ved å ha nådd en viss score eller ved å passere ny vegg/forlate kuben

Legges på et objekt som definerer et område
Valg i Inspector:

- Traced Device: Head, right hand, left hand
- StopType: wall, time, score
- TimeType: countdown, stopwatch
- Timeremaining: antall sekunder i nedtelling



Eksempel med kube, som da må ha en **Box Collider**

Data/informasjon

Variabler som kan hentes ut av utvikler ved behov

double **naiveDistance**

double **distanceMovedInArea**

float **totalMovedDistance**

double **longestDistance**