Høgskulen på Vestlandet

BACHELOROPPGAVE

Room of horror – VR for økt pasientsikkerhet

Room of horror – VR for improved patient safety

Gruppe D10

Adrian René Johnsen Mortensen

Kristoffer Nome

DAT190

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

Dataingeniør / Informasjonsteknologi

Veileder: Harald Soleim

Innleveringsdato: 2/6-20

Jeg bekrefter at arbeidet er selvstendig utarbeidet, og at referanser/kildehenvisninger til alle kilder som er brukt i arbeidet er oppgitt, *jf. Forskrift om studium og eksamen ved Høgskulen på Vestlandet, § 10.*

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel:	*Dato:*
Room of horror – VR for økt pasientsikkerhet	2/6-20
Forfatter(e):	*Antall sider u/vedlegg: 46*
Adrian René Johnsen Mortensen Kristoffer Nome	*Antall sider vedlegg: 7*
Studieretning: Dateingeniør / Informasjonsteknologi	*Antall disketter/CD-er:* 0
Kontaktperson ved studieretning: Harald Soleim	*Gradering:* (Velg: Ingen eller Begrenset til ddmmåå)
Merknader:	

Oppdragsgiver:	*Oppdragsgivers referanse:*
SimArena	
Oppdragsgivers kontaktperson:	*Telefon:*
Marit Vassbotten Olsen	55 58 56 26

Sammendrag:

«Room of Horror» er en metode for simuleringstrening innen bachelor- og masterutdanninger i sykepleie. Det er utviklet et grunnlag for en slik simulering i VR, som kan brukes som utgangspunkt i fremtidig videreutvikling. Spillet inneholder en virtuell operasjonsstue og en virtuell postoperativ avdeling med til sammen 20 mulige aktive feil som skal oppdages og markeres av studenter.

Stikkord:

VR	Simulering	Pasientsikkerhet

Høgskulen på Vestlandet, Fakultet for ingeniør- og natuvitskap
Postadresse: Postboks 7030, 5020 BERGEN Besøksadresse: Inndalsveien 28, Bergen
Tlf. 55 58 75 00 Fax 55 58 77 90 E-post: post@hvl.no Hjemmeside: http://www.hvl.no

FORORD

Denne rapporten beskriver arbeidet ved bachelorprosjektet “Room of Horror - VR for økt pasientsikkerhet”. I prosjektet er det utviklet et grunnlag for videreutvikling av simulering av “Room of Horror” i VR, for fremtidig bruk i et forskningsprosjekt, og i undervisning i masterutdanningene for Anestesi-, Intensiv- og Operasjonssykepleie ved Høgskulen på Vestlandet. Prosjektet er gjennomført av Adrian René Johnsen Mortensen og Kristoffer Nome.

Under utviklingen ble det opprettet en blogg med ukentlige oppdateringer som kan ses på https://dat190g10.code.blog/

Utviklerne ønsker å takke

- Marit Vassbotten Olsen og Lars Peder Vatshelle Bovim - oppdragsgivere ved SimArena, HVL - for muligheten til å delta i prosjektet, en spennende oppgave, og godt samarbeid gjennom hele prosjektperioden

- Harald Soleim - veileder for prosjektet ved Institutt for datateknologi, elektroteknologi og realfag, HVL - for gode råd, nyttige tilbakemeldinger, og konstruktive innspill under arbeidet.

- SimArena og Institutt for datateknologi, elektroteknologi og realfag for utlånt utstyr og ressurser

- Petrin Hege Eide, Hege Emilie Flakne og Tone Johnsgaard - Høgskulelektorer ved Institutt for helse- og omsorgsvitskap, HVL, for masterutdanningene innen hhv. Operasjon-, Anestesi-, og Intensivsykepleie - for forslag til implementasjoner og tilbakemeldinger.

ABSTRAKT

“ROOM OF HORROR” - VR FOR ØKT PASIENTSIKKERHET

Problemstilling: Hvordan kan man bruke metoder for spillutvikling for å utvikle et “Room of Horror” i VR for sykepleiere i masterutdanning? **Bakgrunn:** Ferdighetstrening og simulering er mye brukt i bachelor- og masterutdanninger innen sykepleie. En metode for simulering er “Room of Horror”. Oppsett av fysiske slike rom kan ta tid og være ressurskrevende, men en versjon i VR kan redusere flere ressursbehov. Det er utviklet et grunnlag for en slik simulering, som kan brukes som utgangspunkt for fremtidig videreutvikling. **Design:** Det er brukt Unity for utvikling mot Oculus Quest. Det er opprettet en virtuell operasjonsstue og en virtuell postoperativ avdeling, som til sammen inneholder 20 feil. En lærer kan velge hvilke feil som skal være aktive før simuleringen, og en student kan oppdage og markere aktiverte feil. Det har vært fokus på tilrettelegging for en målgruppe som ikke nødvendigvis har erfaring med bruk av VR. **Evaluering:** Potensialet for evaluering er redusert på grunn av koronavirus-situasjonen. Produktet er evaluert i forhold til oppdragsgivers initielle krav, ved hjelp av ukentlige videokonferansemøter med oppdragsgiver og en utvidet demonstrasjon for faglærere. **Konklusjon:** Så langt det er mulig å evaluere gitt situasjonen oppfyller produktet oppdragsgivers krav. Produktet danner et grunnlag for fremtidig videreutvikling.

“ROOM OF HORROR” - VR FOR IMPROVED PATIENT SAFETY

Problem: How can methods for game development be used to develop a “Room of Horror” in VR for nurses studying for master’s degrees? **Background:** Skills training and simulation are frequently used in bachelor’s and master’s degrees in nursing. One method for simulation is the “Room of horror”. Setup of physical versions of these rooms can demand time and resources, but a version in VR can reduce several resource demands. A basis for simulation that can be used as the starting point for future development has been developed. **Design:** Unity has been used to develop for Oculus Quest. A virtual operating room and a virtual postoperative ward has been implemented, which combined contains 20 errors. A teacher can select which errors that are to be active before the simulation starts, and a student can discover and mark activated errors. An area of focus during development has been ease of use for users that do not necessarily have experience with use of VR. **Evaluation:** The potential for evaluation has been reduced due to the coronavirus-situation. The product has been evaluated compared to the project owner’s initial requirements by using weekly videoconference-meetings with the project owner, and an extended demonstration for lecturers in the relevant educations. **Conclusion:** As far as is possible to evaluate given the situation, the product satisfies the project owner’s initial requirements. The product creates a basis for future development.

ORDLISTE

Virtual Reality (VR)	Virtuell virkelighet. Simulerte opplevelser som gir brukere økt innlevelse ved bruk av VR-briller.
VR-briller	“Briller” som lar brukere “stenge ute” andre synsinntrykk enn det som gis gjennom brillene. Inneholder en skjerm for hvert øye, som gir brukere opplevelse av dybde. Tillater også gjenskapelse av at brukere beveger hodet i VR-opplevelsen, ved hjelp av sporint av bevegelse.
Augmented Reality (AR)	I stedet for at synsinntrykk erstattes fullstendig som ved VR-briller, legges det på et lag med informasjon eller bilder over den virkelige verden når brukere ser gjennom brillene.
Oculus	Produsent av maskinvare og programvare til VR.
Oculus Quest	VR-briller lansert i 2019. Ikke avhengig av tilkobling til en datamaskin siden den inneholder egen maskinvare og bruker et eget operativsystem basert på Android.
Unity	Spillutviklingsverktøy og spillmotor.
Unity: GameObject	Den grunnleggende entiteten i Unity. Kan ha en eller flere komponenter, og kan ha flere GameObject-barn, som selv kan ha flere GameObject-barn.
Unity: Entitet	En entitet kan ha en eller flere komponenter.
Unity: Komponent	Definerer konkrete egenskaper eller oppførsler.
Unity: Prefab	“Standardiserte” GameObjects som kan instansieres omtrent som et objekt er en instans av en klasse.
Unity: Material	Noe forenklet: Definerer hvordan en 3D-modell skal fargelegges.
Oculus Integration	Verktøyspakke for lettere utvikling mot VR-plattformer i Unity.
Virtual Reality ToolKit (VRTK)	Verktøyspakke for lettere utvikling av VR-applikasjoner i Unity.
Blender	Programvare for modellering av 3D-modeller
Fusion 360	Programvare for modellering av 3D-modeller
C#	Programmeringsspråket brukt ved utvikling i Unity
C#: Event	Implementasjon av designmønsteret Observer i C#. Melding sendt av et objekt for å signalisere at en hendelse oppstod. Andre objekter kan “abonnere” på en event, og utføre en forhåndsdefinert respons når eventen oppstår - vanligvis et eget metodekall. Avhenger av

	Delegater.
C#: Delegate	Type i C# som holder referanser til en eller flere metoder. Deklareres ved å angi parametere og returtype for metoder delegaten skal holde referanser til.
Anestesi- Intensiv- og Operasjonssykepleie	Spesialiseringer innen sykepleie-faget. Master-studier. For opptak må en student ha bachelorgrad i sykepleie, og minimum to års relevant yrkespraksis innen de siste fem årene.
Anestesisykepleie	Anestesisykepleieren er den første som møter pasienten på operasjonsavdelingen, og den siste som forlater pasienten ved overføring til oppvåkningsavdelingen [1]. Svært forenklet er anestesisykepleiere spesialisert innen sykepleie til pasienter i narkose.
Intensivsykepleie	En pasient som blir akutt og kritisk syk trenger intensivsykepleie hele døgnet. Utdanningen gir kunnskap og kompetanse for å kunne ta ansvar for akutt og kritisk syke pasienter [2].
Operasjonssykepleie	Operasjonssykepleie krever kunnskap om den akutt og/eller kritisk syke pasienten sin situasjon og behov, og inngående kjennskap til operasjonstekniske prinsipp og kirurgiske undersøkelser og behandling [3].
Operasjonsstue	Rom hvor det utføres kirurgiske operasjoner. En operasjons*avdeling* kan ha flere operasjons*stuer*.
Postoperativ avdeling	Sykehusavdeling hvor pasienter overvåkes etter kirurgiske inngrep inntil de vurderes som stabile nok til å overføres til en vanlig sengepost eller utskrives. En postoperativ avdeling kan bestå av et større åpent rom med flere senger, flere lukkede pasientrom, eller en kombinasjon.
SimArena	Prosjektets oppdragsgiver. Samlebetegnelse for 25 testlaboratorium ved Høgskulen på Vestlandet (HVL). Studetner ved sykepleierutdanningene ved HVL gjennomfører ferdighetstrening og simulering ved SimArena.
Simulering / Simuleringstrening	Metode for trening og læring. Gir mulighet for å trene på etterligninger av hendelser og situasjoner som ligner på virkelige situasjoner. Tradisjonell simulering skilles fra simulering ved hjelp av datamaskiner og VR.
“Room of Horror”	Metode for simulering. Et pasientrom arrangeres med bevisste feil eller brudd på pasientsikkerhetsrutiner. En eller flere studenter skal oppdage flest mulig feil innen en gitt tid.

1 INNLEDNING

1.1 Motivasjon og mål

Uønskede hendelser i helsevesenet kan gi alvorlige konsekvenser for de som rammes. Anestesi-, operasjon- og intensivavdelinger ved norske sykehus er i dag høyteknologiske enheter med høy kompetanse, som gir pasienter et omfattende behandlingstilbud. Pasientsikkerhet er i fokus på disse avdelingene, ved bruk av blant annet sjekklister og simuleringstrening. Likevel skjer det uønskede hendelser av ulik alvorlighetsgrad.

I 2014 lanserte Helse- og omsorgsdepartementet [4] *Pasientsikkerhetsprogrammet I trygge hender 24-7*, som dannet en strategi for pasientsikkerhet i helsevesenet. Et konkret mål i strategien var å redusere pasientskader [5]:

> «*Ved utgangen av 2018 skal «andel pasientopphold med minst én pasientskade, alle alvorlighetsgrader (E-I)» reduseres til 10,3 prosent. Det tilsvarer en 25 prosents reduksjon fra 2012-nivå, som var på 13,7 prosent.»*

Dette målet ble ikke fullstendig oppnådd – i 2018 oppstod det en pasientskade ved 11,9 prosent av somatiske sykehusopphold i Norge [5]. Det er ikke registrert om en pasientskade kunne vært unngått eller ikke, men det regnes med at rundt halvparten av disse pasientskadene kunne vært forebygget [5].

Pasientsikkerhetsprogrammet opphørte i januar 2019, men konseptet, navnet *I trygge hender 24-7* og arbeidet med pasientsikkerhet og kvalitetsforbedring fortsetter [6]. Kampanjen definerer 16 konkrete innsatsområder, hvorav flere er direkte relatert til sykepleiere og spesialsykepleiere.

Utdanning av sykepleiere og spesialsykepleiere er i stadig utvikling, men ferdighetstrening med blant annet anatomiske modeller og rollespill har i mange år vært en del av utdanningene - bruk av treningsdukker og -modeller er beskrevet i litteratur fra 1800-tallet [7]. Den første treningsdukken rettet mot trening av helsepersonell ble laget i 1911, og ble på 1970-tallet erstattet av mer spesialiserte dukker med realistisk respirasjon og hjerteslag [8]. Utviklingen har fortsatt, og det finnes i dag svært avanserte, datastyrte treningsdukker med et bredt spekter av funksjoner.

Med denne utviklingen har simulering blitt en vanlig metode for trening og læring. Ved bruk av simulering kan man trene på ferdigheter og konkrete situasjoner som ligner på det faktisk arbeidet man skal utføre. Dette skaper muligheter for å prøve forskjellige tilnærminger for å løse oppgaver, og dermed få bedre innsikt i hva som vil fungere best i virkeligheten [9]. I senere år har utdanninger også begynt å benytte seg av simulering ved hjelp av dataspill og Virtual Reality (VR) [7]. Ved å gjøre slike simuleringer i VR reduseres flere ressursbehov - for eksempel trengs det ikke nødvendigvis en lærer som veileder og styrer simuleringen, simuleringen kan

gjøres uavhengig av et fysisk rom, og det kan ta betydelig mindre tid å forberede situasjoner. Samtidig har VR-teknologi de siste årene gjort store fremskritt med tanke på tilgjengelighet og brukervennlighet, etter at dagens generasjon med VR-teknologi ble lansert i 2016 [10]. VR opplevde en stor kommersiell økning i 2018 og 2019 [11], og det er forventet at nettopp bruk av VR i utdanning vil bidra sterkt til at denne veksten vil fortsette i den nærmeste fremtiden [12].

En metode for simulering er «Room of Horror» [13]–[15], som setter søkelys på pasientsikkerhet. Dette innebærer et vanlig pasientrom – ofte med en treningsdukke – som på forhånd er arrangert med et visst antall mulige feil eller brudd på pasientsikkerhetsrutiner. Dette kan for eksempel medikamentfeil, mangel på sikring mot pasientfall, mangel på hånddesinfeksjonsmiddel i rommet eller manglende tiltak ved pasientallergier. En student, eller flere studenter sammen, blir sendt i rommet med en oppgave om å oppdage så mange feil som mulig innen en gitt tid. Hensikten er at studentene skal utvikle evnen til å oppdage slike feil i det virkelige arbeidslivet, og kanskje enda viktigere - å øve opp en trygghet til å kommunisere om feil og til å si fra. Bilder av “Room of Horror” kan ses i figur 1 og figur 2 [16].

Det å forberede dette rommet kan ta tid, og når rommet er satt opp er det vanskelig å endre. En “Room of Horror”-simulering i VR vil kreve mindre ressurser for å opprette rommet, og gjøre det enklere å endre rommet mellom simuleringene. Det åpner i tillegg for muligheter for å la scenarioer utspille seg hvor det oppstår feil, som ellers ville krevd betydelige personalressurser for å gjennomføre simuleringen, da flere personer måtte ha deltatt i simuleringen. Bruk av VR i sammenligning med tradisjonelle desktop-spill kan i tillegg gi en større grad av innlevelse i rommet og situasjonene som kan oppstå.

Målet for dette prosjektet er å utvikle et minste brukbare produkt av et VR-miljø for simulering av “Room of Horror” til bruk i utdanningen av Anestesi-, Operasjon- og Intensivsykepleiere ved Høgskulen på Vestlandet (HVL), og med dette svare på problemstillingen: Hvordan kan man bruke metoder for spillutvikling for å utvikle et “Room of Horror” i VR for sykepleiere i masterutdanning?

Figur 1: Room of Horror

Figur 2: Room of Horror

1.2 Kontekst

Ved HVL skjer ferdighetstrening og simulering ved SimArena, som er prosjektets oppdragsgiver. SimArena ønsker å utvide sine muligheter for simulering ved å ta i bruk VR i tillegg til simuleringssrommene som allerede finnes, og vil derfor utvikle et VR-basert verktøy med en virtuell versjon av «Room of Horror».

I tillegg til det potensielle læringsutbyttet er simuleringen planlagt å være en del av et forskningsprosjekt som blant annet ser på nettopp læringsutbyttet blant studenter som går gjennom “Room of Horror”-simuleringer - både reelle og i VR. Til dette forskningsprosjektet er det søkt midler fra Norsk Forskningsråd (NFR), der Helse Bergen, internasjonale samarbeidspartnere og simuleringsmiljø fra ulike profesjoner er involvert. Prosjektet oppnådde ikke å få tildelt midler fra NFR, men har fått tildelt midler fra Kunnskapsdepartementet (KD), og det skal lyses ut en stipendiatstilling der deler av dette prosjektet skal gjennomføres.

Prosjektet er et samarbeidsprosjekt mellom SimArena og Institutt for datateknologi, elektroteknologi og realfag ved HVL. Prosjektet ble opprinnelig foreslått som et masterprosjekt, men er blitt omgjort til et bachelorprosjekt, med mål om en prototype eller minste brukbare produkt i stedet for et fullverdig produkt. Resultatet vil kunne brukes videre som grunnlag for et masterprosjekt, hvor det vil være aktuelt å samarbeide med stipendiat som ansettes. Kombinasjonen av muligheten for videreutvikling og det fremtidige forskningsprosjektet gjør at både oppdragsgiver og prosjektgruppe har et langsiktig perspektiv på prosjektet.

Prosjektgruppen har bestått av to bachelorstudenter på Høgskulen på Vestlandet – én ved bachelorprogrammet for Dataingeniør, og én ved bachelorprogrammet for Informasjonsteknologi.

1.3 Avgrensninger

Som nevnt var prosjektet opprinnelig tenkt som en masteroppgave, men ble noe redusert i omfang og skala. Det har tidvis vært noe vanskelig å fastslå og konkretisere begrensninger, da det ikke var et tydelig endepunkt for prosjektet. Dette kan ha blitt forsterket av at det planlegges å videreutvikle programmet i et masterprosjekt, og at det derfor kan ha vært vanskelig å avgjøre at en funksjonalitet kan implementeres på et senere tidspunkt etter dette bachelorprosjektet. Oppdragsgiver gav uttrykk for at dette i stor grad skulle være opp til prosjektgruppen.

Det har vært flere diskusjoner mellom oppdragsgiver og prosjektgruppen om utformingen av selve verktøyet, om hva en bruker faktisk skal gjøre i produktet, og hva en bruker skal oppleve. Oppdragsgiver har uttrykt et ønske om «levende» scener hvor det utspilles forskjellige scenarioer med forskjellige narrativ og feil som oppstår. Dette avgrenses til mer statiske scener med utplasserte feil.

Prosjektet avgrenses også i omfang av at det utvikles av en prosjektgruppe bestående av to studenter på bachelor-nivå i løpet av en periode på noen måneder, i stedet for et større team med profesjonelle utviklere med større tidsrammer.

1.4 Ressurser

HVL og SimArena har vært behjelpelig med fysiske ressurser. SimArena har lånt ut en Oculus Quest til utviklerne, og Institutt for datateknologi, elektroteknologi og realfag (IDER) ved HVL har utlånt en kraftig data og satt av plass på masterlaben til prosjektgruppen, hvor det finnes flere andre studenter på både bachelor- og masternivå, som kan brukes som ressurser. Denne muligheten ble noe redusert da HVL ble stengt for studenter grunnet den globale situasjonen med korona-viruset, men veileder var behjelpelig med å tilrettelegge for fortsatt kommunikasjon.

Det har vært flere ressurspersoner involvert. Oppdragsgivere og veileder har samarbeidet tidligere på lignende prosjekter som involverte utvikling av prosjekter for VR til bruk ved SimArena, og kan derfor tilby relevant rådgiving og veiledning. Blant studentene som har gjennomført prosjektet, har Adrian tidligere erfaring med flere av utviklingsverktøyene som brukes, og Kristoffer er tidligere utdannet operasjonssykepleier, og har derfor kjennskap til og kunnskap om helsefaglige utfordringer og spesialsykepleie, som er deler av problemdomenet.

For å hjelpe utviklingen har også SimArena bidratt med opp til kr. 10 000,- for innkjøp av 3D-modeller til bruk i scenene.

Programvare som brukes i prosjektet:

- Unity for utvikling av selve VR-programmet
- Blender for modellering og animasjon av modeller
- Fusion360 for modellering

For versjonskontroll brukes Unity Teams, og for kommunikasjon og koordinering brukes Discord. For kommunikasjon og møter med veileder og oppdragsgiver brukes Microsoft Teams og Zoom.

1.5 Oppbygging av rapporten

I denne rapporten beskrives først prosjektet og problemstillingen, før løsningen og implementasjonen beskrives. Dette følges av en mer detaljert beskrivelse av løsningens design, evaluering av resultatet, diskusjon av valg som ble gjort og konsekvenser av disse, og til slutt en konklusjon.

2 PROSJEKTBESKRIVELSE

I dette kapittelet beskrives prosjektets bakgrunn ved å beskrive oppdragsgiver og deres forventninger til prosjektet, samt beskrivelser av relevante metoder og relevant forskning.

2.1 PRAKTISK BAKGRUNN

2.1.1 Prosjekteier

SimArena er en avdeling ved Fakultet for helse- og sosialvitskap (FHS) ved HVL [17]. SimArena er et simuleringssenter med 25 testlaboratorium, hvor studenter kan få ferdighetstrening og veiledning relatert til utfordringer de kan møte i sitt fremtidige yrkesliv, for å øke sikkerheten til fremtidens pasienter [18]. Ved SimArena finnes laboratorier for klinisk sykepleie, som inkluderer Operasjonslaboratoriet med en nesten fullverdig operasjonsstue hvor studenter i Anestesi- og Operasjonssykepleie kan trene, og Intensivlaboratoriet hvor studenter i Intensivsykepleie kan trene. Figur 3 viser bilde av SimArenas operasjonslaboratorium [19].

SimArena planlegger ikke å erstatte disse laboratoriene med VR-utgaver. Både grunnleggende sykepleieutdanning og de tre relevante masterutdanningene involverer praktisk trening og oppbygging av erfaring med konkrete arbeidsoppgaver som per i dag er vanskelig å gjenskape i VR. Simulering generelt innen sykepleiefaget er ment å forbedre og utvide læring [20], ikke erstatte ferdighetstrening og utplassering i praksis, noe som også gjelder for simulering med VR. SimArena ønsker å bruke VR for å øke bevisstheten blant studenter rundt det å oppdage potensielt farlige feil i arbeidssituasjoner, og å øke tryggheten til å kommunisere disse feilene. Deres mål er å bidra til å redusere uønskede hendelser og pasientskader.

Figur 3: SimArenas operasjonslaboratorium

2.1.2 Tidligere arbeid

Det er gjort mye tidligere arbeid innen simulering i VR for helsevesenet. Datagrafikk og VR har spilt en viktig rolle i flere tiår for diagnosering og behandling av pasienter, og simulering blir i økende grad brukt for trening til og planlegging av kirurgiske inngrep [21]. Det finnes flere verktøy for bruk til slik trening og opplæring (se for eksempel [22]–[25]), som allerede brukes ved Haukeland Universitetssykehus - og det finnes spesialiserte verktøy med spesialiserte håndkontrollere for konkrete typer operasjoner, som for eksempel VR-løsningen LapSim [26] for trening til laparoskopiske operasjoner ("kikkhull-operasjoner"). Det finnes også VR-program med fokus på trening i kommunikasjon som tilsynelatende har gitt gode resultater ved bruk av medisinstudenter [27].

Det finnes forskningslitteratur om tradisjonelle "Room of horror" som er beskrevet i kapittel 2.2.2, men det lykkes ikke å finne mye informasjon om "Room of Horror" i form av VR-implementasjoner. Det finnes beskrivelser av at det er tatt i bruk ved noen amerikanske utdanningsinstitusjoner allerede i 2018 [28], men ingen informasjon om dette er et kommersielt produkt som kan kjøpes eller hvordan dette fungerer. Etter internettsøk oppdages det filmer fra en VR-implementasjon som lot til å være under utvikling da filmene ble lagt ut høsten 2019 [29], men det lykkes ikke å finne informasjon om verktøyet er ferdig utviklet og tatt i bruk, eller noen beskrivelser av verktøyets funksjonalitet. Uansett, selv om dette verktøyet skulle være tilgjengelig for andre er det rettet mot medisinstudenter og sykepleierstudenter på bachelornivå, og finner sted i tradisjonelle sykehusavdelinger, ikke i spesialiserte enheter som anestesi-, operasjon- og intensivavdelinger som dette prosjektet tar plass i. Dette prosjektet baserer seg derfor ikke på et tidligere prosjekt eller program.

2.1.3 Initielle krav

I prosjektet skal det utvikles en virtuell versjon av "Room of horror". VR-programmet skal inneholde to scener:

- En operasjonsstue som inkluderer to leger - en kirurg og en anestesilege - en anestesisykepleier og en operasjonssykepleier
- En postoperativ enhet som inkluderer en anestesilege, en anestesisykepleier og en intensivsykepleier

Scenene skal animeres og innredes med aktuelt utstyr. Det skal lages en algoritme eller flere standardiserte grupper med feil og mangler, med utgangspunkt i veiledning fra oppdragsgiver. Det skal implementeres at en lærer kan forhåndsvelge feil som skal være "aktivert" i scenen, eller en student skal kunne starte programmet med tilfeldig utvalgte aktive feil eller grupper med feil. En student skal i løpet av en gitt tid identifisere feil som eksisterer i scenen.

Ved starten av prosjektperioden hadde ikke oppdragsgiver bestemt seg for om studenten skulle få umiddelbar tilbakemelding om en feil ble korrekt oppdaget, eller om studenten etter simuleringen skulle få opplysninger om hvor mange feil som ble oppdaget av totalt antall feil i scenen - eller ingen tilbakemelding i det hele tatt. Etter diskusjoner mellom prosjektgruppe og

oppdragsgiver og en tidlig implementasjon av feil-markering ble det oppnådd enighet om at en bruker skal kunne markere feil i scenen, med muligheter for tilbakemelding etter simuleringen.

Det stilles i tillegg krav om fokus på brukervennlighet og brukbarhet. Det kan ikke forventes at målgruppen for programmet har erfaring med bruk av VR.

2.1.4 Initiell løsnings-idé

Siden prosjektet reduseres i skala fra et masterprosjekt til et bachelorprosjekt settes det noen begrensninger.

Oppdragsgiver ville i utgangspunktet ha et program som kunne brukes både på en tradisjonell data-skjerm og med VR-briller, men etter diskusjon mellom prosjektgruppe og oppdragsgiver velges det å ta utgangspunkt i at programmet i første omgang skal bruke VR-briller. Oppdragsgiver ville også i utgangspunktet ha en kryssplattform-løsning som kan brukes på VR-briller fra flere produsenter. Siden oppdragsgiver har gjort en Oculus Quest tilgjengelig for utviklerne og dette ifølge oppdragsgiver sannsynligvis vil være deres mest brukte plattform, velges det i første omgang å fokusere på denne plattformen. Under utviklingen gjøres det tiltak for å opprettholde størst mulig grad av kryssplattform-kompatibilitet for senere utvikling.

Oppdragsgiver uttrykte i starten av prosjektperioden ønsker om å la scenarioer og situasjoner utspille seg i scenene, og det ble foreslått at brukere skal kunne stoppe scenarioet når det oppstår en feil brukeren reagerer på. Dette ville krevd betydelig opplæring og arbeid med animasjoner. Prosjektgruppen var usikre på om det var tilstrekkelig tid i utviklingsperioden til å gjennomføre dette, og det oppnås derfor enighet om at det i første omgang opprettes scener med relativt statiske feil, men der personene i scenen er animerte og kommuniserer, og en mulig “aktivert” feil kan involvere noe som blir sagt i scenen.

Som nevnt i avsnitt 1.2 har både oppdragsgiver og prosjektgruppe et langsiktig perspektiv på prosjektet. Dette medførte at det ble enighet om å fokusere på å lage et solid rammeverk og grunnlag for videreutvikling, i stedet for å fokusere på å fylle scenene med mest mulig innhold.

Oppdragsgiver foreslår at programmet implementeres i Unity, da de har tidligere erfaring med bachelor- og masterprosjekter for VR som brukte dette.

Løsning og design beskrives nærmere i kapittel 3, og beskrives i detalj i kapittel 4.

2.2 LITTERATUR OM PROBLEMSTILLINGEN

En av utviklerne i prosjektet har tidligere erfaring med 3D-utvikling i Unity, men ingen har erfaring med VR-utvikling. Det brukes derfor mye litteratur og dokumentasjon om dette. I dette avsnittet beskrives først anbefalinger og metoder for utvikling av VR, og deretter litteratur om simulering med “Room of Horror”.

2.2.1 Metoder for utvikling i VR

Som nevnt i avsnitt 2.1.3 kan det ikke forventes at brukere har erfaring med bruk av VR. Dette betyr at det ikke bør implementeres avansert bruk av håndkontrollere, og at det bør gjøres tiltak for å forebygge VR-sjøsyke. Dette er en tilstand som kan minne om reisesyke som enda ikke er helt forstått, men som kan oppstå ved bruk av VR når en bruker visuelt opplever at kroppen beveger seg gjennom VR-brillene, men fysisk opplever at kroppen står i ro [30] - eller ved lite optimalisering som fører til lav bildefrekvens og en forsinkelse mellom når bruker utfører en bevegelse, og når den samme bevegelsen vises i VR-brillene [31].

Metoder og teknikker for implementasjon av grunnleggende VR-funksjonalitet baserer seg på Oculus sitt kurs i VR-utvikling for Unity, som er utviklet i samarbeid med Unity og finnes på nettressursen Unity Learn: “Design, Develop and Deploy for VR” [32]. I denne beskrives blant annet implementasjon av bevegelse, interaksjon med scenen ved brukerens hender, og anbefalinger for brukergrensesnitt - som alle baserer seg på råd og erfaringer fra utviklere ved Oculus.

For brukbarhet og optimalisering følges også metoder og praksiser angitt i Oculus Developers sine “VR Guidance and Best Practices” [33]. Blant disse er det noen seksjoner som spesielt legges merke til:

- Vision [34]: For å oppnå opplevelse av dybdesyn bruker VR to skjermer, en for hvert øye. Dette kan føre til ubehag for brukere. Det må tas hensyn til at objekter i scenen en bruker skal fiksere øynene sine på - som menyer eller i dette prosjektet feil i sykehusrommet - bør tegnes minst ½ meter unna.

  Det anbefales å ikke bruke “Heads-Up Display” som viser viktig informasjon - i tradisjonelle spill for eksempel spillerens helse eller tilgjengelig ammunisjon. I VR kan dette i større grad enn på en vanlig skjerm blokkere objekter i scenen eller potensielt skape ubehag og frustrasjoner ved brukerens tolking av dybde. Slik viktig informasjon bør heller bygges inn i scenens omgivelser, eller gjøres til en del av brukerens representasjon, for eksempel ved å vise informasjon på brukerens hender.

- Locomotion [35]: Hvordan en bruker flytter seg i scenen er viktig å implementere riktig. Dårlige implementasjoner kan føre til ubehag og VR-sjøsyke. Det er foreløpig ikke oppdaget en universal løsning for bevegelse som fungerer for alle brukere, men noen metoder har blitt etablerte og mye brukt av VR-utviklere:

    - Brukeren kan ikke flytte seg i scenen på andre måter enn å bevege seg selv i det virkelige rommet.
    - Teleportering - brukeren peker med håndkontrollere på bakken, og flytter seg til det markerte stedet på et øyeblikk, uten at brukeren faktisk opplever å bevege seg dertil. Dette unngår ubehag som kan oppstå ved akselerering, men kan føre til en desorienterende følelse ved store endringer, spesielt hvis brukeren ikke gis klare og tydelige tegn for å forutsi hvor man ender etter flyttingen.
    - Kunstig bevegelse lar en bruker starte og stoppe bevegelser, men bevegelsen skjer i en konstant hastighet, langs en rett linje. Dette unngår potensielt ubehagelig akselerasjon.
    - Blinking og "snap turns" unngår å vise visuell informasjon som potensielt kan føre til ubehag ved å gjøre skjermen svart et øyeblikk eller å umiddelbart forandre synsvinkelen. Dette utnytter at menneskets hjerne er flink til å "fylle inn" små avbrytelser i synet, som når man blinker med øynene eller flytter blikket.
    - Akselerasjon ved bruk av VR-brillene lar brukere styre bevegelse ved å vippe hodet i retningen man vil bevege seg. Dette kan føre til ubehag hvis vipping av hodet også roterer brukeren.
    - Tunnelsyn ved bevegelse blokkerer brukerens sidesyn, som har vist seg å være en effektiv måte å redusere ubehag på.
    - Bruk av håndkontrollere med en liten forsinkelse fra når brukeren begynner å bevege på tommelstikken for å bevege seg, til bevegelsen starter lar brukeren forutse og forberede seg på bevegelsen, som fører til mindre ubehag.
    - Brukeren flytter på omgivelsene i stedet for at brukeren beveger seg selv i omgivelsene

- User Input [36]: I dette prosjektet skal en bruker peke ut og markere feil i rommene. Det må derfor defineres hvordan en bruker skal gjøre dette med håndkontrollere. Det anbefales at representasjoner av brukerens hender eller håndkontrollere nøyaktig følger bevegelsene brukeren faktisk gjør, både med hensyn til rotasjon og bevegelser.
  Det må gjøres bevisste valg om handlinger som utføres når brukeren trykker gitte knapper på håndkontrollerne. Brukere som har tidligere erfaring med bruk av VR kan ha vaner og forventninger til at en knapp utfører en bestemt handling, og for brukere uten erfaring bør bruk av knappene være så intuitiv som mulig. Det anbefales å følge standard kartlegging av knappene på håndkontrollerne, som definert av Oculus [37].

  Feil implementasjon av håndregistrering og posisjonering kan føre til ubehag og frustrasjon for brukere, og selv et lite avvik i rotasjon kan oppleves forvirrende.
  Hvis det velges å representere brukerens hender med 3D-modeller av hender bør det tas hensyn til situasjoner hvor hendene går gjennom geometri og solide objekter. Det kan oppstå ubehag både ved at en hånd da forsvinner "inn i" objektet, og ved å la hånden kollidere og stoppe med objektet, mens brukerens virkelige hånd fortsatt kan bevege seg "dypere" inn i objektet. Et alternativ til å representere brukerens hender med modeller av hender er å heller representere håndkontrollerne som brukes grafisk. Dette kan fungere bra til å peke på brukergrensesnitt eller å peke på objekter i scenen.

- Rendering [38]: Tekst både i brukergrensesnitt og i scenen bør være lesbar. Det anbefales å bruke en *signed distance field* font - en skrifttype som opprettes med spesielle teknikker, som ser skarp og tydelig ut også etter skalerings- og rotasjonstransformasjoner [39]. Fontstørrelse og utplassering av tekst i scenen er viktig.

  Det gis også anbefalinger for å optimalisere VR-programmer. Det bør tilstrebes at tiden mellom øyeblikket en bruker beveger hodet og det oppdaterte bildet vises i VR-brillene er så liten som mulig. Dette behovet forsterkes da det tas utgangspunkt i utvikling til den trådløse Oculus Quest-plattformen, som har noe redusert yteevne og mindre kraftig hardware i forhold til VR-briller som bruker hardware i en tilkoblet datamaskin for å tegne grafikk.

2.2.2 Litteratur om "Room of horror"

Etter litteratursøk er det funnet to forskningsartikler som spesifikt omhandler "Room of horror" i helsevesenet.

Farnan et al [13] beskriver et "Room of horror"-forsøk med til sammen 214 medisinstudenter. Det ble opprettet et realistisk klinisk scenario i en vanlig sykehusavdeling med spesifikke pasientsikkerhetsrisikoer, og det ble laget en pasientjournal med informasjon om casen, pasientens allergier og pasientens faste medikamenter. Pasienten ble representert med en treningsdukke. Feil som skulle bli oppdaget av studenter inkluderte blant annet at pasientjournalen viste smittefare men rommet hadde manglende smittevernstiltak, at pasientjournalen manglet viktige medikamenter som er vanlige ved sykehusinnleggelser, at hånddesinfeksjonsdispenser i rommet var tom, og at det ikke var gjort tiltak mot fallrisiko. Noen feil ble oppdaget av opptil 80% av deltakere, mens noen av så få som 6%. Flere deltakere markerte potensielle feil som ikke var tiltenkt som identifiserbare feil eller simulerte farer. Som en konklusjon oppleves simuleringen som en effektiv måte å gjøre deltakere oppmerksom på pasientsikkerhet, og til tross for at flere deltakere oppdaget få feil, var de fornøyd med den interaktive opplevelsen og den kliniske nytteverdien.

Et viktig aspekt fra dette forsøket som er relevant for prosjektet denne rapporten handler om, er at flere deltakere markerte feil som ikke var en del av simuleringen. Den virtuelle representasjonen av sykehusrom vil ikke være en perfekt gjenskapelse av virkeligheten, og kan inkludere flere feil som er et resultat av at for eksempel modeller ikke er realistiske nok, eller plassering av objekter ikke er nøyaktige nok i forhold til virkeligheten. Avhengig av implementasjon bør det forsøkes å forebygge at brukere henger seg opp i ikke-relevante feil, eller at brukere bruker mye tid på å forsøke å markere feil som i programmet ikke er definert som feil, som i stedet vil kunne oppleves som dårlig brukergrensesnitt.

Clay et al [40] beskriver forsøk med 51 sykepleiestudenter og 93 medisinstudenter, som alle utførte to "Room of horror"-simuleringer satt i en intensivavdeling. En student gikk først gjennom simuleringen alene. En uke senere tok deltakerne del i undervisning og diskusjon som tok

grunnlag i data fra simuleringene. Innen 48 timer etter undervisning gjennomførte deltakerne en ny simulering som et tverrprofesjonelt team med tre til fire deltakere. Deltakere ble gitt syv minutter i selve simuleringen, men så mye tid som ønsket på forhånd for å gå gjennom pasientjournalen. Studentene noterte observasjoner på papir under selve simuleringen, og ble etterpå bedt om å beskrive sine funn elektronisk i et åpent skjema. Det ble bevisst ikke brukt sjekklister eller noen form for multiple-choice, for å unngå noen form for bias eller forhåndsantagelser om feil i simuleringene, da dette kunne ført til at studenter var forberedt på å oppdage bestemte feil, eller i etterkant "oppdaget" feil på skjemaer som ikke faktisk ble observert. Resultatene fra undersøkelsen viste at enkelte feil var mer sannsynlig å bli oppdaget av medisinstudenter, og enkelte feil var mer sannsynlig å bli oppdaget av sykepleierstudenter. Som i forsøkene beskrevet av Farnan et al [13], oppdaget studentene flere feil som ikke var en bevisst del av simuleringen. Generelt oppnådde team bedre resultater enn individer.

Det bemerkes i artikkelen at flere studenter beskrev den første simuleringen som "enkel", som førte til følelser av sårbarhet når de faktiske resultatene ble presentert. Diskusjonene som ble gjort i etterkant av den individuelle simuleringen inkluderte bevisst humor for å oppfordre studentene til å slappe av, og bidro til å bli oppmerksom på andre personer og profesjoner sine roller innen pasientsikkerhet. Team-simuleringen ga studenter mulighet til å fortelle andre studenter om sine roller og ansvar, og gjorde studenter oppmerksom på det store antallet oppgaver og sikkerhets-sjekker som er nødvendig for pasientsikkerhet, og viktigheten av samarbeid for å utføre disse oppgavene. Artikkelen foreslår også at deltakere kan ha nytte av å bruke sjekklister som brukes i virkelige arbeidssituasjoner - både for å hjelpe dem i simuleringen, og å gjøre dem kjent med sjekklistene.

Denne artikkelen inneholder flere punkter som kan være potensielle utvidelser av dette prosjektet i en senere master-oppgave. I denne omgang ble det avgjort å utvikle simuleringene for én person om gangen, men det kunne vært nyttig å senere utvide dette til at flere går gjennom en simulering samtidig, og å kunne inkludere pedagogiske hjelpemidler og læringsmuligheter i VR-programmet. Forslaget om bruk av sjekklister kan potensielt implementeres som en valgmulighet i senere versjoner, og det kan være nyttig å implementere at en bruker leser en pasientjournal før simuleringen.

3 DESIGN AV PROSJEKTET

I dette kapittelet gis et overblikk over utviklingsprosessen, ved å se på valg av løsning og valg av utviklingsverktøy, samt prosjektmetodikk og hvordan prosjektet evalueres ved slutten av utviklingsperioden.

3.1 FORSLAG TIL LØSNING

Som nevnt i avsnitt 2.1.4 velges det å fokusere utviklingen på muligheten for å bruke VR-briller til å oppleve scenarioene som skal implementeres. Disse scenarioene kan blant annet være de samme scenarioene som er brukt i undervisning ved det fysiske Room of horror, men med mindre bruk av plass. Dermed vil alternativene for løsningen omhandle hvilken VR-teknologi løsningen skal designes rundt.

3.1.1 Alternativ løsning 1 Alt-i-ett VR hodesett

Et alternativ er å bruke en enhet hvor alt kommer i en pakke. Den spesifikke enheten prosjektgruppen har tilgang til er Oculus Quest.

Oculus quest er et VR-hodesett hvor alt er på enheten [41]. Denne løsningen krever lite oppsett av sluttbruker ettersom det ikke omhandler programvare på en datamaskin og det ikke er nødvendig å koble den til en datamaskin. Enheten har et eget operativsystem basert på Android. All maskinvare som behøves for å drive enheten er også bygd inn. Dette fører til en høy grad av tilgjengelighet, siden lite må settes opp og klargjøres før bruk.

Siden all maskinvare er integrert i enheten har den limitasjoner i ytelse [42]. Enheten skal festes til brukere, noe som setter krav til enhetens formfaktor. Dette har konsekvenser for hvor kraftig maskinvare som kan brukes. Resultatet av disse konsekvensene betyr at en bruker kan oppleve noe lavere oppløsning og lavere bildefrekvens i skjermene, som kan føre til en noe mindre realistisk opplevelse. Dette alternativet krever at løsningen optimaliseres for bruk av integrert maskinvare.

3.1.2 Alternativ løsning 2: Datamaskin avhengig hodesett

Ved datamaskin-avhengige VR-briller vil man bruke maskinvaren og ressursene til en tilkoblet datamaskin for å kjøre scenene. Dette innebærer at hodesettet altså har kabler som går til datamaskinen med programvaren.

Datamaskinen kan ha gode komponenter og dermed kan simuleringen ha meget god ytelse med mange objekter og detaljer, og en høy grad av realisme. Med et godt skjermkort kan også disse VR-brillene ofte ha veldig god oppløsning.

Eksempler på datamaskin-avhengige VR-briller er Oculus Rift og HTC Vive.

3.1.3 Alternativ løsning 3: AR-hodesett

«Augmented reality» (AR) eller Utvidet Virkelighet på norsk, er litt annerledes fra VR siden det i stedet for å erstatte synsinntrykk legges på et lag med informasjon på bildet av den virkelige verden [43]. Denne løsningen kan også være en alt-i-ett enhet hvor det som trengs for å lage dette ekstra laget på bildet er innebygd i enheten. Denne løsningen krever da også at man har en fysisk scene man kan legge til digitale elementer til.

Et eksempel på AR-briller er Microsoft HoloLens.

3.1.4 Diskusjon av alternativene

I dette prosjektet skulle det lages et alternativ til det fysiske «Room of horror» som brukes av sykepleiere i videreutdanning ved HVL. Dermed ville løsningen med AR ikke være av interesse for oppdragsgiver.

Utvidet virkelighet ville krevd at det var et fysisk rom på størrelsen av scenen og dette rommet måtte enten inneholdt utstyret som trengs i scenen eller digitale gjenstander. Dette vil dermed ikke være et godt alternativ til det fysiske rommet som brukes til “Room of Horror”. Det ble derfor fokusert på VR løsninger hvor alt er digitalt.

Datamaskin-avhengig hodesett vil kunne kjøre det meste med god ytelse om maskinen som er tilkoblet har de nødvendige ressursene. Problemet kommer ved bruk av dette, da det krever en del forberedelser og mer plass for å bruke ettersom datamaskinen som tar plass må være koblet til hodesettet. Selve datamaskinen kan i tillegg være kostbar.

Alt-i-ett hodesett vurderes som en god løsning ettersom det er lærere og studenter - som ikke nødvendigvis har høy kompetanse i oppsett av spill og VR-plattform - som skal sette i gang øvelsene, og man slipper å ha en datamaskin koblet til. Dette gjør løsningen meget tilgjengelig. I tillegg er det billigere å kjøpe inn utstyr for oppdragsgiver, da det ikke trengs separate datamaskiner for flere samtidige brukere. Problemet med denne løsningen er at ressursene i enheten har begrensninger på hvor detaljert og hvor store scener kan være. Det må derfor gis mer oppmerksomhet til optimalisering og bruk av optimaliseringsteknikker. I tillegg kan resultatet ha noe lavere grad av realisme enn ved bruk av datamaskin-avhengige hodesett, da 3D-modeller som brukes må være noe mindre komplekse. Bruk av dynamiske lys må også begrenses. Lys og skyggelegging kan være svært viktig for opplevelsen av realisme, men dette kan være svært ressurskrevende.

Høsten 2019 lanserte Oculus konseptet *Oculus Link* [44], som åpner for muligheten til å bruke Quest-brillene koblet til en stasjonær PC ved en USB 3-kabel, og bruke PC-en sin maskinvare for utregninger og opptegning av 3D-grafikk. Dette tilfører disse VR-brillene økt fleksibilitet, og

kan unngå problemet med maskinvare-begrensninger - med unntak av en noe lavere oppdateringsfrekvens i brille-skjermene enn nyere VR-briller bygd spesifikt for tilkobling til PC. Dette har også store betydninger for utviklingsarbeidet, da det ikke er nødvendig å utføre tidkrevende forberedelser og kompilering for å teste at selv svært små endringer fungerer som planlagt under kjøring av spillet.

3.2 VALGT LØSNING

Etter diskusjon med oppdragsgiver velges det å utvikle for et alt-i-ett hodesett, og dermed prioritere tilgjengelighet over realisme i denne omgang. Det skal tilrettelegges for muligheten til også å kunne kjøre programmet ved å bruke et hodesett som er tilkoblet en datamaskin, noe som i fremtidig videreutvikling åpner for muligheten til å kunne øke realismen avhengig av hvilke VR-briller programmet kjøres på, eller om det brukes Oculus Quest-briller med bruk av Oculus Link.

Dette valget har konsekvenser for utviklingen. Som nevnt må det brukes tid på optimalisering, og det må tas hensyn til kompleksiteten av 3D-modellene som brukes i scenene. Samtidig betyr arbeid gjort for å optimalisere kjøring på en alt-i-ett-enhet at spillet i høy grad også vil være optimalisert for kjøring på datamaskin-avhengige VR-briller.

3.3 VALG AV VERKTØY

Prosjektet krevde flere kategorier med verktøy for å fullføres. Prosjektet måtte planlegges, og på grunn av Covid-19 situasjonen, måtte planlegging skje virtuelt over internett. I prosjektet skulle det også utvikles en simulering som det skal kunne interageres med, og dermed trengtes det spillutviklingsverktøy. Prosjektgruppen ønsket også at det skulle brukes modeller i simuleringen som er realistiske, og som ligner på utstyr som brukes i virkeligheten. Dermed trengtes modelleringsverktøy for å lage modeller som kanskje ikke kunne finnes til salgs.

3.3.1 Utviklingsverktøy

Utviklingen av prosjektet ble gjort i spillutviklingsverktøyet Unity med programmeringsspråket C#. Unity er en anerkjent spillmotor med god dokumentasjon og flere gode ressurser for veiledning [45]. Utviklingsverktøyet finnes i flere tilgjengelige kategorier av versjoner, som grovt kan oppsummeres til tre forskjellige: Nye alfa- og beta-versjoner i 2020-kategorien, nye stabile versjoner i 2019-kategorien som anbefales for utviklere som vil bruke nyere, oppdaterte Unity-funksjoner, og eldre, svært stabile versjoner i 2018-kategorien eller enda eldre versjoner, med såkalt *Long Term Support (LTS)*, som anbefales for utviklere som ikke har behov for nye

funksjoner, og ikke vil forholde seg til endringer i API-er, for eksempel hvis en produktlansering er nært forestående [46].

Prosjektgruppen ville utvikle på en mest mulig stabil plattform, så før utviklingen ble startet ble det bestemt å bruke en eldre LTS-versjon - Unity 2018.4.18f1 - og ikke gå gjennom noen versjonsoppdateringer hvis ikke dette var absolutt nødvendig. Siden utviklingstiden var relativt kort, ville ikke prosjektgruppen måtte bruke tid på potensielle feil som kunne oppstå i nyere versjoner, eller problemer som kunne oppstå ved oppdatering av utviklingsverktøyet.

3.3.2 Modelleringsverktøy

For modellering av mekaniske deler og apparater for operasjonssal og postoperativ avdeling, ble det brukt CAD programmet Autodesk Fusion 360. Dette programmet er også godt anerkjent for design til produksjon [47]. Programmet brukes på grunn av tidligere erfaring fra prosjektgruppemedlemmer.

For redigering av modeller og modellering av mer organiske elementer ble det valgt å bruke modelleringsverktøyet Blender. Blender er et åpen-kilde-prosjekt som er blitt en av de markedsledende for spillutvikling og filmskaping blant amatør og selv-publiserte skapere [48].

3.3.3 Planleggingsverktøy

Ved oppstart av prosjektet ble alle HVL campus stengt på grunn av Covid-19 situasjonen [49]. Dermed ble det nødvendig med verktøy for å samarbeide virtuelt.

Oppgaver ble planlagt ved hjelp av sprint-tavler på Trellos webløsning. Unitys innebygde *Unity Collaborate* ble brukt for versjonskontroll, og for deling av filer og selve prosjektet. Dokumentasjon ble gjort i Googles skytjeneste «Google Drive».

3.4 PROSJEKTMETODIKK

3.4.1 Utviklingsmetodikk

Prosjektgruppen valgte å bruke prosessrammeverket SCRUM under utviklingen av prosjektet. SCRUM er et smidig utviklings rammeverk hvor valg blir gjort ut fra erfaring og det som er kjent for prosjektet. Dette rammeverket er ledende i programvareutvikling [50].

SCRUM bruker tre hovedroller under utviklingen. Produkteier eller i dette prosjektet SimArena, utviklingsteamet eller prosjektgruppen, og SCRUM master som er definert for å passe på at det tilrettelegges slik at utviklingen går som planlagt [50].

Utvikling i SCRUM skjer på inkrementell måte, som gjøres i «Sprinter» - en fast leveransesyklus [50]. Gruppen valgte å ha sprintsykluser på en uke. Sprintsyklusene begynte med et planleggingsmøte hver mandag hvor prosjektgruppen gikk gjennom en overordnet backlog som ble laget av ønsker fra produkteier, som da ble lagt til en mindre sprint-backlog hvor de viktigste funksjonene basert på produkteiers ønske ble prioritert først. Sprint backloggen ble bestemt av prosjektgruppen; basert på prioritet og hvor mange oppgaver prosjektgruppen mente kunne gjøres i løpet av sprinten. Hver syklus ble avsluttet med et prosjektgruppe-møte hver fredag, hvor ukens utvikling ble oppsummert, og det ble påbegynt planlegging av neste uke.

I sprinten valgte prosjektdeltakere selv hvilke oppgaver de tok ansvar for ved å markere oppgaven som påbegynt. Oppgaven ble gjennomført og markert oppgaven som ferdig. Oppgavene ble representert som lapper på en tavle med kolonner for de forskjellige fasene en oppgave kan være i [50]. I dette prosjektets tilfelle ble Trello brukt til å vise status for de forskjellige oppgavene.

Prosjektgruppen valgte å bruke SCRUM metodikken ettersom den passer godt til arbeid hvor det ikke er et klart og kjent produkt, og ettersom gruppen har ukentlige møter med produkteier er det viktig å kunne utvikle på en smidig måte.

I dette prosjektet ble det brukt objektorientert programmering (OOP). Et objektorientert programmeringsspråk er en modell som organiserer designet av programmet rundt data, eller objekter [51]. Grunnen til at prosjektgruppen valgte denne typen programmering var at Unity som standard bruker programmeringsspråket C# som er objektorientert [52].

3.4.2 Prosjektplan

Prosjektplanen ble utarbeidet slik at rapport ble skrevet parallelt med utviklingen av produktet. Planen er også delt i 2 hovedgrupper: Forprosjekt og hovedfase. Forprosjektperioden varte fra 9. mars frem til 3. april, og innebærte planlegging av prosjektet og grunnleggende implementasjon av VR-interaksjon, samt utforming av scenene som skal være med i sluttløsningen.
Hovedfasen av prosjektet varte fra 14. april til 2. juni, og inkluderte å samle det som ble gjort i forprosjektet, lage ferdigbygde pakker som kunne trekkes inn for videre utvikling, samt fullføre rapport og scenarioer ønsket av produkteier. Fullt Gantt-skjema med planen kan finnes i vedlegg 2.

3.4.3 Risikovurdering

Risikoliste kan finnes i Vedlegg 1.

	Risikodiagram			
Svært sannsynlig				
Sannsynlig			1	
Lite sannsynlig		3	4, 5	
Ikke sannsynlig			2	
	Ubetydelig	Mindre alvorlig	Alvorlig	Svært alvorlig

Mal fra arbeidstilsynet [53]

Ut fra evaluering av risiko har prosjektgruppen en risiko med høy sannsynlighet og alvorlig konsekvens. Dermed ble det utviklet tiltak for å redusere risikoen og konsekvensen av mangel på VR testing med testpersoner.

Etter risikovurdering anså prosjektgruppen disse risikoene som håndterbare, da det ble gjort tiltak for å redusere konsekvenser om noe skulle skje.

3.5 EVALUERINGSPLAN

Ved slutten av prosjektperioden er det utført en evaluering av prosjektet.

Ved starten av prosjektperioden var det planlagt å fokusere prosjektets evaluering på spillets brukbarhet og brukervennlighet, og at sluttbrukere - faglærere og studenter fra de relevante videreutdanningene ved HVL - kunne bruke spillet på en tilfredsstillende måte. Dette skulle evalueres ved å gjennomføre brukertesting mot slutten av prosjektperioden, hvor noen testpersoner fra målgruppen kunne prøve spillet i VR og bli intervjuet av utviklerne etter en gjennomført simulering. Dette ville også kunne gitt innsikt i om spillets funksjonalitet ble oppfattet som å ha tilstrekkelig potensiale for senere videreutvikling. Disse planene måtte endres da HVL stengte campus for studenter på grunn av koronavirus-situasjonen, som betydde at det ikke ville være mulig å gjennomføre en tilfredsstillende brukertesting, hvor brukere og umiddelbare reaksjoner kunne observeres.

Det ble derfor bestemt å fokusere prosjektets evaluering på spillets funksjonalitet i forhold til oppdragsgiver sine krav, og dermed evaluere prosjektet ved hjelp av akseptansetesting. Det er lagt større vekt enn planlagt på hyppige tilbakemeldinger og evalueringer av fremgang i ukentlige møter med oppdragsgiver, og en større demonstrasjon over videokonferanse for noen faglærere i målgruppen. Dette beskrives nærmere i kapittel 5.

4 DETALJERT DESIGN

I dette kapittelet beskrives hvordan prosjektets resultat er oppnådd. Dette gjøres ved å se på hvordan VR-funksjonalitet er implementert, etterfulgt av hvordan prosjektet er bygget opp, og en beskrivelse av brukeropplevelsen.

4.1 Integrasjon av VR

4.1.1 Verktøyspakker

Unity har innebygget støtte for utvikling til VR, men lite funksjonalitet som støtter eller hjelper selve utviklingen. Det velges derfor å importere Oculus Integration-pakken [54], som inneholder flere SDK-er, C#-scripts, Unity *Prefabs* og andre ressurser, som alle gjør utviklingen i Unity lettere. Spesifikt for dette prosjektet brukes denne pakken blant annet for å håndtere grafisk opptegning til de to skjermene i VR-brillene, og innlesing av knappetrykk på håndkontrollere. Pakken håndterer i tillegg oversettelse av sporing av kamera og håndkontrollere fra bevegelser gjort av brukere, til bevegelser av kamera og hender i spill-scenene. Som nevnt i avsnitt 2.1.4 skulle det tilstrebes fremtidige muligheter for å bruke spillet på andre VR-plattformer enn Oculus Quest, og denne pakken bidrar til dette, da det å tilrettelegge for kryssplattform-kompatibilitet og å gjøre utvikling for flere plattformer lettere er et eksplisitt mål med verktøyet [55].

I tillegg importeres verktøyspakken *Virtual Reality ToolKit* (VRTK) [56], som er en samling med gjenbrukbare løsninger på utfordinger som er vanlige ved utvikling til VR. Konkret i dette prosjektet brukes VRTK for lettere å implementere at en bruker kan plukke opp objekter i scenen, og å implementere at en bruker kan trykke inn fysiske knapper i scenen. I tillegg utnyttes kombinasjonen av Oculus Integration og VRTK: Der Oculus Integration kan fortelle spillet at brukeren trykket en knapp på håndkontrolleren, gjør VRTK det lettere å programmere hva som skal skje når brukeren trykket på nettopp den spesifikke knappen.

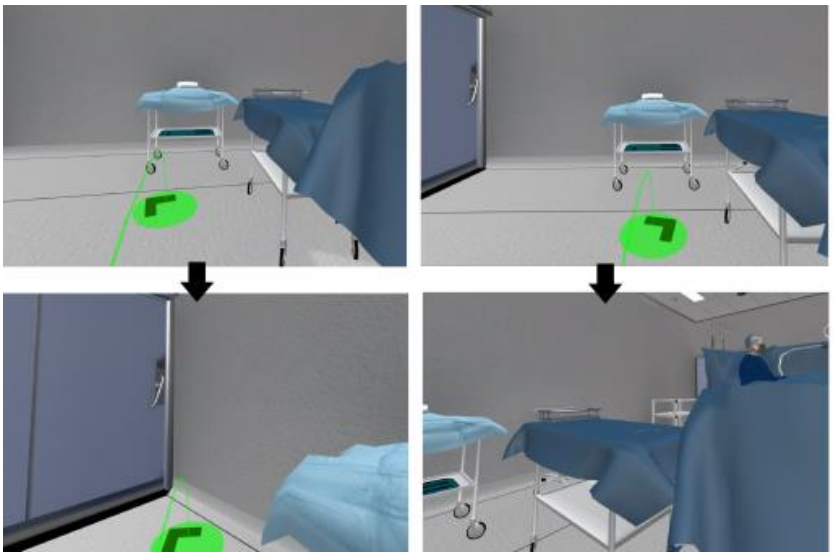
4.1.2 Bruk av metoder

Implementasjon av grunnleggende VR-funksjonalitet baserer seg på Oculus sitt innføringskurs “Design, Develop and Deploy for VR” [32], som nevnt i avsnitt 2.2.1. I Unity brukes prefaben OVRCameraRig fra Oculus Integration for å definere VR-briller og håndkontrollere for Unity, og prefaben TrackedAlias fra VRTK brukes for lettere å kunne ha tilgang til objektene som representerer VR-brillene og håndkontrollere i Unity, og å kunne behandle disse. Oppsett av disse krever en del spesifikk referering mellom objekter i Unity-editoren, som demonstreres i kurset.

Det tas også hensyn til noen av prinsippene i “VR Guidance and Best Practices” [33] som beskrives i avsnitt 2.2.1. Etter disse retningslinjene velges det å gjøre brukergrensesnitt til en del av omgivelsene i VR, i stedet for tradisjonelle menyer som er “låst” til skjermen, som potensielt kan skape ubehag for brukere. Brukergrensesnittet vises derfor på skjermer i scenen. Det anbefales også at objekter brukeren skal fokusere på bør være minst en halv meter fra

brukeren. For menyene løses dette ved å plassere brukerens startposisjon over en halv meter fra meny-skjermene. Anbefalingen er derimot vanskeligere å følge for objekter i scenene hvor brukeren skal oppdage feil, og selv kan ta initiativ til å studere et objekt på nært hold for å finne en detalj-feil. Det kan tenkes at dette kan føre til ubehag for brukeren, som potensielt kan konkludere med at bruk av VR generelt er en ubehagelig opplevelse. I programmet er det derfor forsøkt å gjøre feil tydelige nok til å ses fra en halv meters avstand. Som nevnt i avsnitt 2.2.2 beskriver litteratur om tradisjonelle “Room of horror” at deltakere ofte registrerer feil som ikke var ment som en feil i simuleringen. Dette betyr at det vil være vanskelig å kunne forutsi hva en bruker vil studere på eget initiativ, selv om det er gjort tiltak for å tydeliggjøre de implementerte feilene.

Som nevnt i avsnitt 2.2.1 finnes det flere muligheter for implementasjon av bevegelse av spilleren i scenene. Et mål med spillet er å la brukere bevege seg fritt i rommet for å lete etter feil, men samtidig må det forebygges VR-sjøsyke for brukere med mindre erfaring med VR. Det velges derfor å bruke metoden teleportering. Denne metoden gir brukere en stor grad av frihet, med lav sannsynlighet for ubehag. Når brukerens venstre tommel rører venstre tommelstikke vises en indikasjon på bakken over hvor brukeren kommer til å flytte seg, og ved å rotere venstre tommelstikke kan brukeren velge hvilken retning kamera skal se etter flyttingen. Ved å trykke venstre tommelstikke ned utføres selve teleporteringen. Selve bevegelsen i scenen skjules ved at skjermene i VR-brillene blir svarte et øyeblikk. Figur 4 viser teleportering i spillet. Venstre side viser øverst at brukeren holder venstre tommelstikke mot venstre når tommelstikken trykkes inn, og underst at brukeren ser mot venstre etter teleporteringen. Høyre side viser det motsatte - brukeren holder venstre tommelstikke mot høyre når tommelstikken trykkes inn, og ser mot høyre etter teleporteringen.



Figur 4: Implementert teleportering. Bordet i midten av de øverste bildene kan ses i ytterkanten av synsfeltet etter teleportering.

Et diskusjonspunkt som oppstår fra denne implementasjonen er om dette kan være vanskelig å gjennomføre for brukere som ikke er vant med VR eller bruk av håndkontrollere. Det ble vurdert å implementere faste punkter på bakken en bruker kunne velge å flytte seg til og dermed redusere potensialet for at en bruker blir overveldet. Dette ville også kunne virke forebyggende mot at en deltaker studerer objekter på for nært hold. Siden en del av brukeropplevelsen er ment å inkludere å kunne inspisere detaljer i scenen, ble det likevel valgt å beholde bevegelsesfriheten. I tillegg til teleportering velges det å implementere "snap turns". Når brukeren beveger høyre tommelstikke til venstre eller høyre roteres brukeren 45 grader om sin vertikale akse i scenen - i Unity tilsvarer dette brukerens y-akse. Denne funksjonen kan ignoreres ved at brukeren heller snur kroppen sin, men dette vil være vanskelig hvis brukeren er sittende. Kombinasjonen av venstre og høyre tommelstikker er bevisste valg for å følge Oculus sin anbefalte praksis om å tilrettelegge for brukere som har tidligere erfaring med VR og bruk av håndkontrollere - i 3D-spill har det i mange år vært en vanlig praksis å bevege spilleren med venstre tommelstikke og å rotere kamera om spillerens y-akse med høyre tommelstikke.

I både innføringskurset fra Oculus og i anbefalte praksiser fra Oculus anbefales det å bruke en *signed distance field* font for tekst. I Unity importeres derfor pakken *TextMesh Pro*, som anbefales i innføringskurset. Denne pakken tilbyr bruk av en slik font på en enkel måte, og brukes for å vise all tekst i spillet. Igjen kan dette virke forebyggende mot at en bruker føler behovet for å studere tekst på for nært hold.

Begge kildene gir også anbefalinger om at det bør gis spesiell oppmerksomhet til ytelse og bildefrekvens ved utvikling til VR. Innføringskurset gir konkrete forslag til tiltak som kan implementeres. Under utviklingen av prosjektet ses det også på andre kilder som gir optimaliseringsråd for utvikling til Oculus Quest [57], [58]. Noen konkrete punkter som implementeres:

- For å unngå store mengder beregninger under kjøring brukes få dynamiske lys og bruk av lightmapping - at belysning av elementer i scenen utføres ved kompilering og gjøres statisk, i stedet for kontinuerlig oppdatering av belysning under kjøring.
- Liten bruk av skygger for å unngå store mengder beregninger under kjøring
- De fleste objektene i Unity markeres som statiske, som noe forenklet medfører at Unity kan unngå en del beregninger under kjøring
- *Occlusion culling* - det brukes ikke ressurser på å tegne opp objekter i scenen som ligger bak noe annet i forhold til kameraenes synsvinkler
- Liten bruk av gjennomsiktige objekter, som kan medføre gjentatte utregninger for den samme skjermpikselen. I sykehusomgivelser finnes en del gjennomsiktige objekter som anses som nødvendig for å opprettholde realisme i scenene, så det kan ikke unngås fullstendig.
- Unngå for mange utregninger i Unity-klassenes *Update()*-metoder (beskrives nærmere i avsnitt 4.2.2)

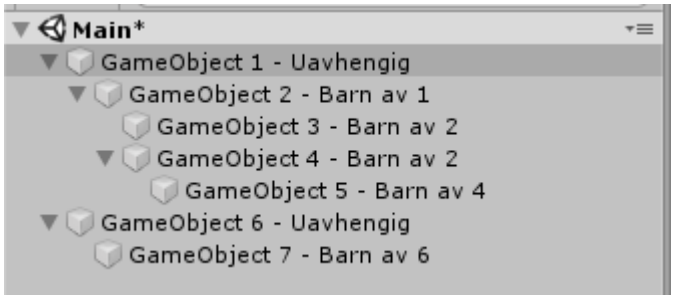
Oculus anbefaler å tilstrebe en bildefrekvens på 72 bilder per sekund under kjøring [59], som er det høyeste som kan oppnås på Quest-brillene. I samtaler med oppdragsgiver oppgis det at

dette ikke er et absolutt krav for prosjektet, og at SimArena allerede bruker noen kommersielle VR-programmer som ikke oppfyller denne anbefalingen. Under utviklingen har prosjektgruppen likevel tilstrebet en så høy som mulig bildefrekvens.

4.2 Prosjektstruktur og arkitektur

Utvikling i Unity er i utgangspunktet basert på en Entity-Component-System-arkitektur, som er mye brukt i spillutvikling [60]. I denne arkitekturen tilstrebes det å erstatte polymorfi med komposisjon i oppbygging og strukturering av programkode [61]. Det er et stort fokus på lav kobling, som fører til høy grad av gjenbrukbarhet. I stedet for at det oppstår flere ledd med arv som kan føre til stor og uoversiktlig kode med flere avhengigheter, skilles kode grovt sett i *entiteter* og *komponenter*. En *entitet* kan bestå av en eller flere flere *komponenter*, der komponenter kan definere tradisjonelle objekter, eller konkrete egenskaper eller oppførsler. Både entiteter og komponenter kan brukes i flere forskjellige sammenhenger. I Unity brukes *GameObject* som den grunnleggende entiteten - et slikt objekt kan ha flere GameObject-entiteter som "barn", som hver kan bestå av flere komponenter, og igjen inneholde flere andre entiteter. Et eksempel på forhold mellom flere GameObjects vises i figur 5. Et GameObject-barn vil arve en forelders transformasjons-egenskaper, som medfører at et barn vil bevege seg eller bli rotert i scenen hvis forelderen beveges eller roteres. Slike GameObjects kan også pakkes sammen til standardiserte *Prefabs*, som kan instansieres nesten slik et objekt er en instans av en klasse - en Prefab består av en definert struktur og med definerte egenskaper, men én instans av Prefaben kan ha annet innhold eller andre verdier for disse egenskapene enn en annen instans av den samme Prefaben - slik en klasse kan angi et objekts struktur og attributter, mens et instansiert objekt av klassen kan ha andre attributt-verdier enn et annet instansiert objekt av den samme klassen.

GameObjects i Unity inneholdes i Unity-*Scener*. En *Scene* kan ses på som en logisk inndeling av spillets innhold. I praksis angir en Scene hva som skal lastes inn i maskinens minne - når Scenen startes lastes Scenens innhold inn, og når det startes en ny Scene gir maskinen slipp på ressursene som krevdes av den forrige Scenens innhold. I figur 5 vises innholdet i en eksempel-Scene kalt *Main*, som ses øverst i hierarkiet.



Figur 5: Forhold mellom GameObjects

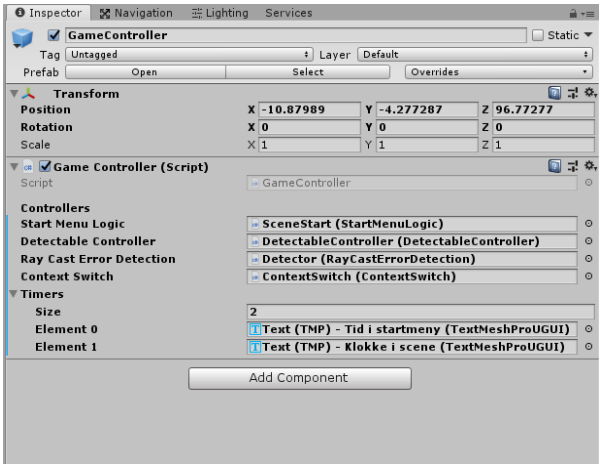
«Room of horror»-prosjektet baserer seg på denne arkitekturen. I dette delkapittelet beskrives dette nærmere ved å se på prosjektets struktur – først i Unity, og deretter prosjektets kodestruktur. For både GameObjects og i programkode er det gjort tiltak for å opprettholde størst mulig grad av uavhengighet, for å tilrettelegge for senere videre utvikling. Dette beskrives nærmere i avsnitt 4.4.

4.2.1 Unity-struktur

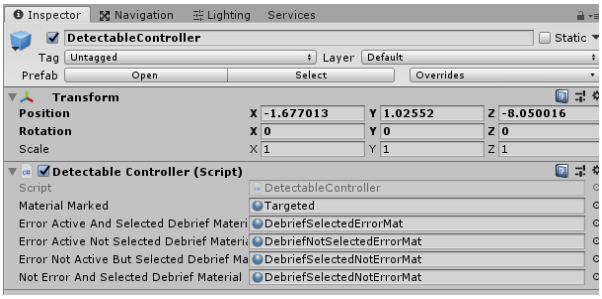
I prosjektets oppdragsbeskrivelse oppgis som tidligere nevnt to omgivelser: En operasjonsstue og en postoperativ avdeling. Dette gir en naturlig inndeling i to forskjellige Unity-scener. Det opprettes i tillegg en oppstartsscene hvor en bruker kan velge hvilken scene som skal startes, og en "Tutorial"-scene hvor brukere blir vist hvordan håndkontrollerne brukes og får trene på bruken av disse. Alle scenene inneholder prefaben *Player*, som selv inneholder prefabs for *VR Player* og *Desktop Player*. Disse inneholder kameraer for de forskjellige bruksområdene, og håndtering av bevegelse av VR-briller og håndkontrollere for bruk av VR, og håndtering av tastatur, mus, og eventuelle håndkontrollere for spilling uten VR-briller.

Scenene for operasjonsstue og postoperativ avdeling - hvor selve spillet foregår - inneholder flere felles prefabs for å håndtere start og avslutning av spillet, og spill-logikk. Den viktigste av disse er prefaben *GameController*. Objektet består av én komponent: C#-scriptet med samme navn. Denne fungerer som et bindeledd som knytter entiteter og de forskjellige delene av spillet sammen ved hjelp av C#-kode, som beskrives nærmere i avsnitt 4.2.2. Komponenten gis eksplisitte referanser til disse entitetene gjennom Unity-editoren. Entiteten har ett barn: En klokke i scenen som viser gjenværende tid i simuleringen. Figur 6 viser komponentene til en instans av prefaben i Unity-editoren. *Transform*-komponenten angir objektets posisjon, rotasjon, og skalering i scenen, og er en del av alle GameObject i Unity. *Game Controller*-komponenten viser hvordan et C#-script kan motta referanser til andre objekter ved hjelp av Unity-editoren. I bildet er komponenten gitt referanser til andre GameObject som inneholder angitte komponenter - for eksempel er medlemsvariabelen Start Menu Logic i C#-scriptet *GameConroller* her tildelt referansen til det konkrete GameObjectet *SceneStart* i scenen, som inneholder komponenten StartMenuLogic.

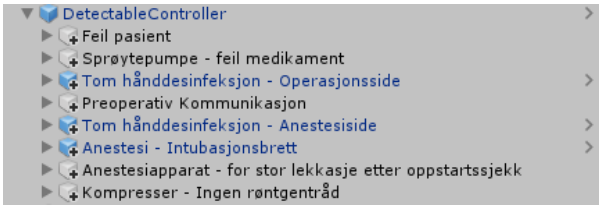
Prefaben *DetectableController* består også av én komponent, som er opprettet av C#-scriptet med samme navn. Komponenten er ansvarlig for å holde oversikt over objektene i scenen som kan markeres av en bruker, og gis kun referanser til materialer som brukes for å farge et objekt når det blir markert, og for å farge et objekt i det som kalles *Debrief*, hvor en bruker blir vist korrekt markerte feil, og feil som ikke ble funnet. Prefabens barn består av GameObjects som inneholder potensielt aktive feil. Dette betyr at instansen av *DetectableController* som brukes i operasjonsstue-scenen har andre barn enn instansen som brukes i postoperativ-scenen. Figur 7 viser komponentene til prefab-instansen som brukes i operasjonsstue-scenen, og figur 8 viser en oversikt over den samme entitetens barn.



Figur 6: Bilde fra Unity-editor: Entiteten GameController med den tilknyttede komponenten Game Controller



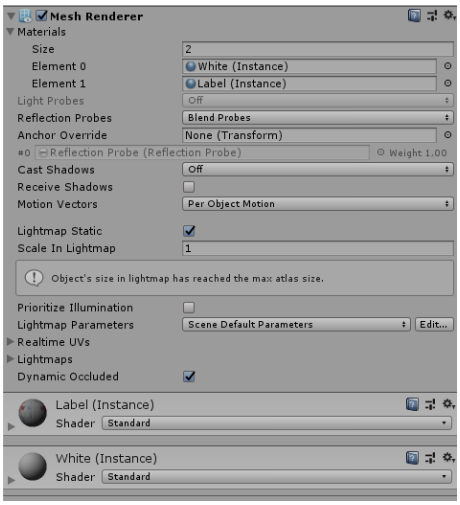
Figur 7: Komponenter for DetectableController-entiteten



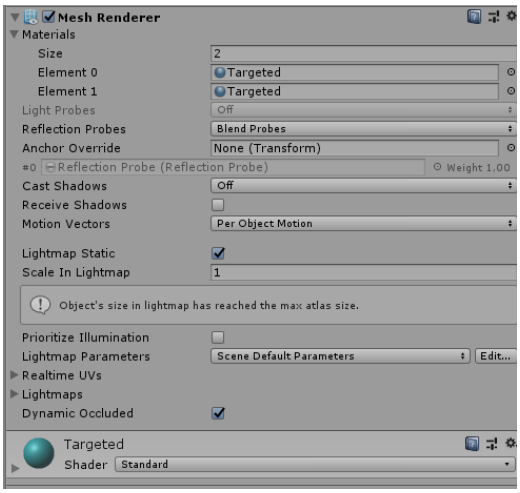
Figur 8: DetectableController-entiteten vist med barn i Operasjonsstue-scenen

DetectableController oppdager på egenhånd alle entiteter i scenen som er tilknyttet komponenten *Detectable*, som gjør at 3D-modeller tilknyttet entiteten eller entitetens barn endrer farge når en bruker markerer en av modellene. En 3D-modell tegnes i Unity ved hjelp av komponenten *Mesh Renderer*. Ett element i denne komponenten som er viktig for implementasjon av objekt-markering er hvilke Unity-*Materialer* som brukes - noe forenklet forteller dette Unity hvordan 3D-modellen skal fargelegges. En 3D-modell kan bruke en eller flere slike materialer. Figur 9 viser komponenten *Mesh Renderer* for tegning av modellen til en pose med hånddesinfeksjon, og figur 10 viser den samme komponenten etter at hånddesinfeksjonen er markert av brukeren - materialene er endret.

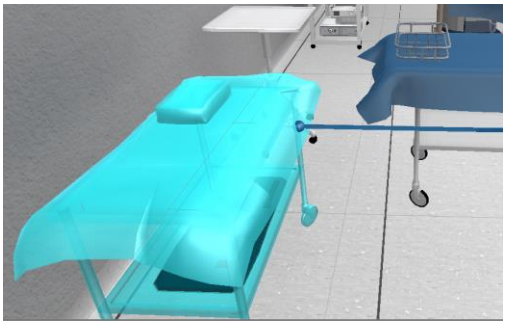
En *Detectable*-entitet krever også en tilknytning til en variant av Unity-komponenten *Collider* - for eksempel *BoxCollider* eller *MeshCollider* - som tilfører entiteten kollisjonsdeteksjon. Dette definerer et område hvor entiteten kan bli truffet av spillets laserpeker, og bidrar til å tydeliggjøre for brukeren hvilket objekt i scenen som pekes på. Figur 11 viser en entitet tilknyttet en *MeshRenderer*-komponent som definerer 3D-modellen for et bord, en *Detectable*-komponent som er markert av bruker, og at laserpekere kolliderer med entitetens *Collider*-komponent.



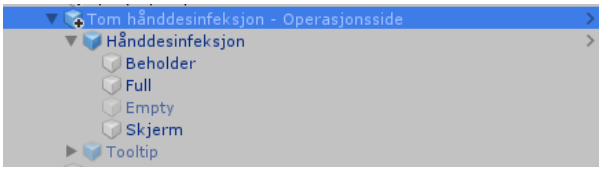
Figur 9: Mesh Renderer for en pose med hånddesinfeksjon



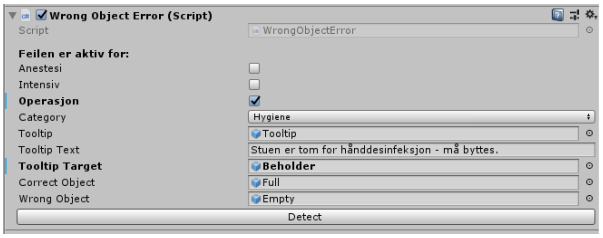
Figur 10: Mesh Renderer for en pose med hånddesinfeksjon som er markert av brukeren


Figur 11: Bord som er markert av bruker. Laserpekeren kolliderer med bordets *Collider*

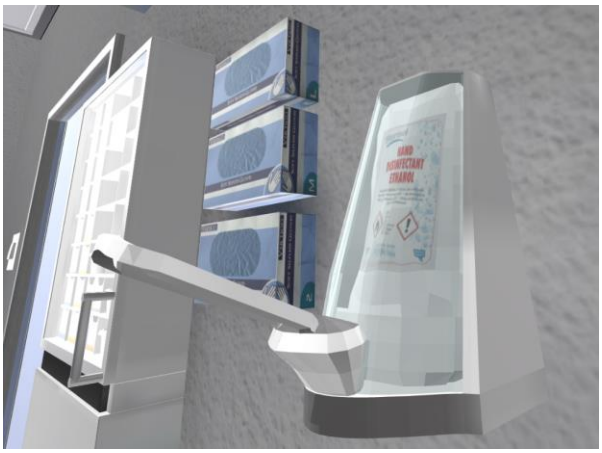
Komponenten *Detectable* danner også grunnlaget for å definere feil i scenen ved bruk av polymorfisk arv i programkoden, som beskrives nærmere i avsnitt 4.2.2. Et GameObject som definerer en feil kan være oppbygd på flere forskjellige måter avhengig av feilen, og det er definert fem forskjellige feil-typer: *CommunicationError, SomethingIsMissingError, VideoError, WrongObjectError,* og *WrongTextError*. Antall feiltyper kan enkelt utvides i fremtidig utvikling. Som et eksempel er feilen "Tom hånddesinfeksjon" knyttet til komponenten *WrongObjectError*, som definerer denne feil-typen. I denne feilens tilfelle legges en instans av prefaben *Hånddesinfeksjon* som barn av feil-entiteten. Prefaben har selv en *BoxCollider* for kollisjonsdeteksjon, og inneholder GameObject-barn som er knyttet til 3D-modeller. Feil-komponenten gis referanser til en 3D-modell som skal vises hvis feilen er aktiv, og en modell som skal vises hvis feilen ikke er aktiv. Det angis i tillegg hvilken sykepleier-spesialisering feilen er aktiv for og hvilken feil-kategori feilen tilhører, og komponenten gis en referanse til til et *Tooltip* - en Prefab fra verktøyspakken VRTK - som vises i Debrief, samt tekst som skal vises i tooltipet, og hvilket GameObject tooltipet skal peke til. Figur 12 viser feil-entiteten med sine barn, og figur 13 viser komponenten *Wrong Object Error*. Figur 14 viser entiteten hvor feilen ikke er valgt som aktiv, figur 15 viser entiteten hvor feilen er valgt som aktiv, og figur 16 viser entiteten i Debrief, med tooltip.



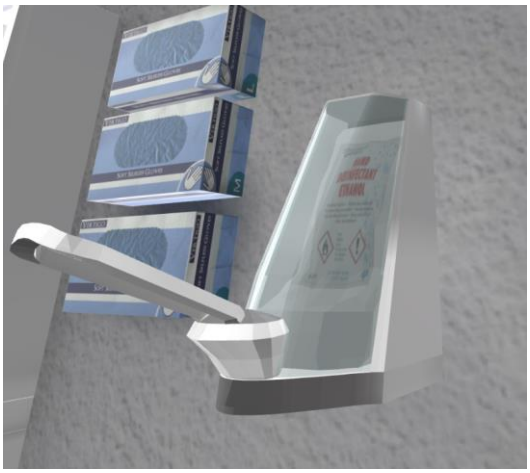
Figur 12: Instans av prefaben *Tom hånddesinfeksjon* med barn



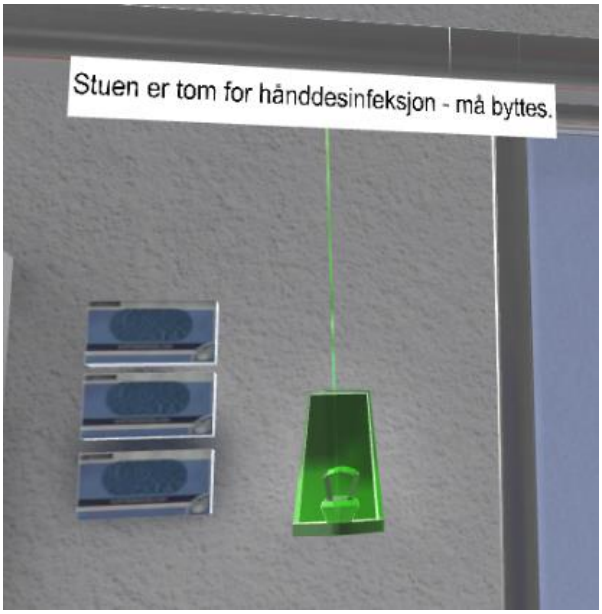
Figur 13: Komponenten *Wrong Object Error* tilknyttet en instans av prefaben *Tom hånddesinfeksjon*



Figur 14: Tom hånddesinfeksjon - Feil ikke aktiv



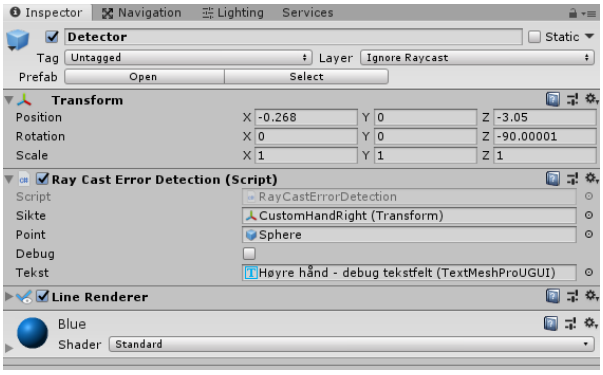
Figur 15: Tom hånddesinfeksjon, feil aktiv.



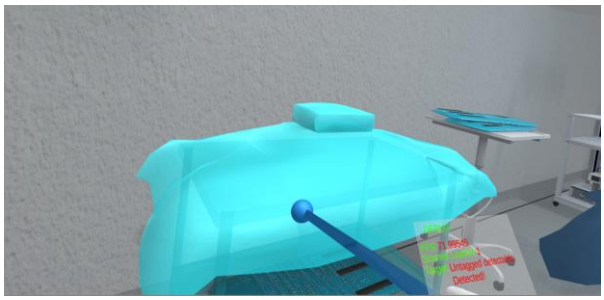
Figur 16: Tom hånddesinfeksjon i Debrief med tooltip. Farget grønn siden bruker markerte feilen.

Prefaben *DetectableController* hører sammen med prefaben *Detector*, men det er ingen eksplisitt kobling mellom disse i Unity - dette skjer i *GameController*-scriptet. *Detector* er ansvarlig for å lage laserpekeren en bruker benytter seg av for å markere feil i scenen. Som et instansiert GameObject ligger entiteten som et barn av GameObjectet *VR Player*. Komponenten som dannes av C#-scriptet *RayCastErrorDetection* trenger referanser til VR-spillerens høyre hånd siden denne danner utgangspunktet for laserpekeren, og laserpekeren følger håndens bevegelser ved hjelp av VR-implementeringen. I tillegg gis en referanse til en kule som brukes for å tydeliggjøre nøyaktig hvor i scenen laserpekeren peker. Implementasjonen av laserpeker har tatt utgangspunkt i prefaben *UIHelpers* og denne prefabens barn *LaserPointer*, som er en del av Oculus sitt Unity-rammeverk. Det ble valgt å implementere en egen versjon av en laserpeker for å lettere kunne tilpasse denne til prosjektets behov. *Detector* bruker i tillegg Unity-komponenten *Line Renderer*, som er ansvarlig for å tegne grafisk selve linjen som utgjør laserpekeren. Figur 17 viser entiteten *Detector* i Unity-editoren, og figur 18 viser laserpekeren i bruk, hvor det vises at en bruker peker på et bord i operasjonsstuen.

Under utvikling var det nødvendig å opprette en liten debug-skjerm tilknyttet spillerens høyre hånd for å observere at rett objekt ble utvalgt i scenen når det ble pekt på av laserpekeren. Det velges å beholde denne funksjonaliteten, med mulighet for å slå dette av og på under utvikling. Denne kan ses i Figur 18.

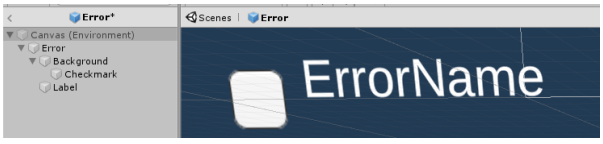


Figur 17: Entiteten *Detector* med komponenter


Figur 18: Entiteten *Detector*: LineRenderer-komponent tegner strek, og en kule tydeliggjør hva det pekes på. Debug-skjerm kan også ses nede til høyre.

Prefaben *SceneStart* inneholder geometri for et rom som danner oppstartsomgivelser, og flere skjermer som brukes som bakgrunn for brukergrensesnitt. I tillegg defineres selve brukergrensesnittet i prefaben, ved hjelp av Unity-komponenter. Komponenten *Menu Logic* gis referanser til entiteter og komponenter som brukes i brukergrensesnittet - både knapper i selve grensesnittet, og entiteter som brukes som en mal for elementer i lister over mulige feil som skal aktiveres. Disse listene bygges dynamisk fra entiteter tilknyttet en *Detectable*-komponent i scenen, og resultatet blir en rekke GameObjects med en Unity *Toggle*-komponent. Figur 19 viser prefaben *ErrorToggle* i Unity-editoren som danner malen, og figur 20 viser en ferdig opprettet liste med instanser av prefaben. Selve funksjonaliteten i menyene defineres i C#-kode, som beskrives nærmere i avsnitt 4.2.2.

Til sammen er det utviklet 11 forskjellige feil, der noen kan gjenbrukes på forskjellige steder i samme scene, eller gjenbrukes i forskjellige scener. For eksempel kan feilen “Tom hånddesinfeksjon” brukes to ganger i operasjonsstuen da det er vanlig å ha en hånddesinfeksjonsbeholder på hver side av rommet - og feilen kan brukes i scenen for den postoperative avdelingen, da hånddesinfeksjonsbeholdere også er til stede i denne settingen. Med disse 11 forskjellige feilene er det implementert 10 feil i operasjonsstuen, og 10 feil i den postoperative avdelingen. Dette vurderes som tilstrekkelig for å kunne gi et bredt inntrykk av feilene som er mulig å implementere i fremtidig utvikling.



Figur 19: Prefaben *ErrorToggle* for feil-elementer i startmeny

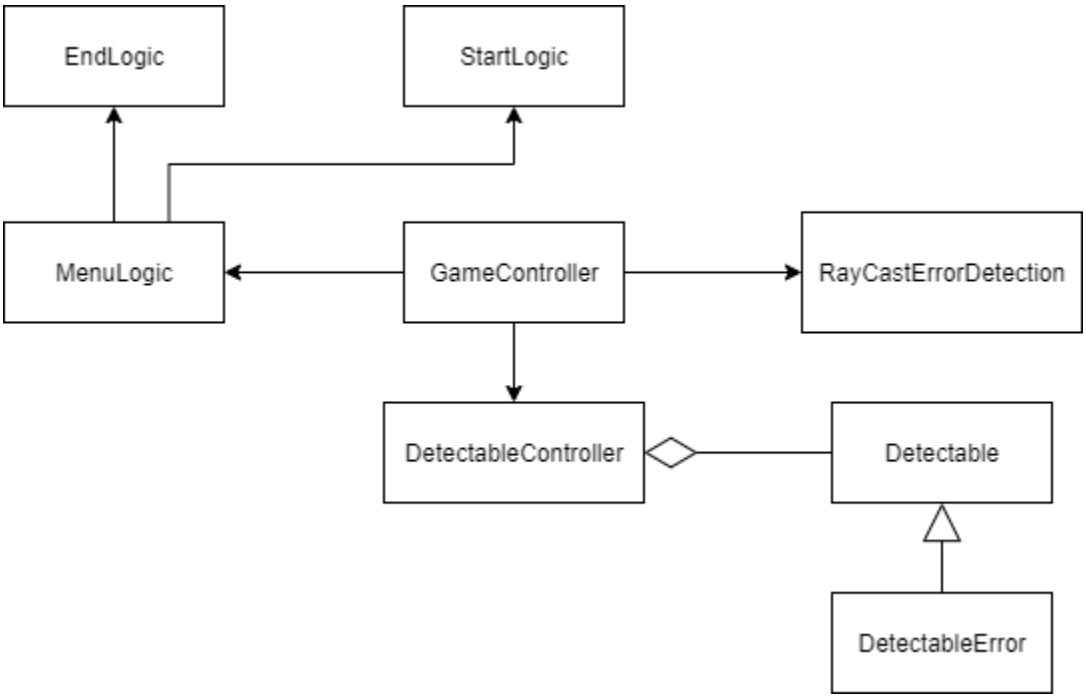
Figur 20: Ferdig opprettet liste med tre instanser av *ErrorToggle*

4.2.2 Programkodestruktur

I dette delkapittelet beskrives de viktigste klassene i prosjektet - som vises i et UML-diagram i figur 21 - og noen av de viktigste metodene i disse klassene.

For programmering i Unity brukes C#. Klasser som skal brukes aktivt i Unity-editoren må arve fra klassene *MonoBehaviour* eller *ScriptableObject*. Begge disse klassene inneholder metoden Awake(), som kalles når scriptet først lastes inn og kan tenkes på som en konstruktør, men kun *MonoBehaviour* inneholder metodene Start(), som kalles etter initialisering men før noe annet skjer i spillet, og Update(), som kalles for hvert bilde som tegnes i spillet [62], [63].

Klassen *GameController* har ansvaret for å styre spillets gang, og fungerer som et bindeledd mellom andre, uavhengige objekter. Dette bidrar til å opprettholde lav kobling og å tilrettelegge for videre utvikling, siden andre, viktige objekter kan fungere og gjøre sin jobb uten å bli påvirket av at andre klasser endres. Slike koblinger utføres i klassens Start()-metode. Spesielt viktig er koblingen mellom *RayCastErrorDetection*- og *DetectableController*-objektene.



Figur 21: Klassediagram over de viktigste klassene i prosjektet: GameController bruker objekter av klassene MenuLogic, DetectableController og RayCastErrorDetection. DetectableController har en aggregasjon av flere Detectable-objekt. DetectableError arver fra Detectable. MenuLogic bruker objekter av klassene StartLogic og EndLogic.

RayCastErrorDetection er ansvarlig for å oppdage hva en bruker peker på med laserpekeren og håndtere at en bruker velger å markere objektet, og *DetectableController* er ansvarlig for å holde rede på objekter i scenen som kan markeres, og fortelle slike objekter at de skal bytte farge når de blir markert. Disse funksjonalitetene er relaterte, og kunne implementeres ved høy grad av kobling. I stedet velges det å implementere dette ved hjelp av en C# event. Ved å bruke en event kan *RayCastErrorDetection* leve i spillet uten å være avhengig av andre objekter og deres implementasjon og funksjonalitet. I stedet for at det gjøres et eksplisitt metodekall fra *RayCastErrorDetection* på en metode i *DetectableController* som ville skapt en avhengighet, settes metoden detectDetectable(Detectable detectedObject) i *DetectableController* til å abonnere på eventen, og *RayCastErrorDetection* kan aktivere sin event uten å måtte bry seg om hva en annen klasse - hvis noen - gjør med objektet som ble detektert, i metoden detect(), som blir kalt når en bruker trykker på knappen på håndkontrollerne som markerer objekter..

I *GameController.Start()* - RayCastErrorDetection blir bedt om å legge til metoden detectDetectable som abonnent på sin detectEvent:

```
rayCastErrorDetection.SubscribeToDetectEvent(detectableController.detectDetectable);
```

I *RayCastErrorDetection* - metoden gitt som parameter settes som abonnent på detectEvent:

```
public void SubscribeToDetectEvent(Action<Detectable> action)
    {
        detectEvent += action;
    }
```

Når brukere markerer objekter med laserpekeren kalles metoden detect() som aktiverer Eventen, og sender objektet i scenen som pekes på som argument til metoder som abonnerer.

```
public void detect(){
        Detectable detectable = hit.transform.GetComponent<Detectable>();
        if (detectable != null)
        {
            detectEvent?.Invoke(detectable);
        }
    }
```

I dette tilfellet fører aktivering av detectEvent ved å kalle på Invoke()-metoden altså til at *DetectableController* utfører sin detectDetectable-metode.

Det gjøres i tillegg en lignende kobling til start-menyens start-knapp. Dette starter spillet, og spesifikt i *GameController* fører dette til at klokken starter hvis det er valgt å gjøre simuleringen på tid, og andre objekter som kontrolleres av dette objektet gjøres oppmerksom på at spillet starter, i metoden StartSim():

```
menuLogic.SubscribeToStartButtonEvent(() => StartSim());
```

Klassen *MenuLogic* er ansvarlig for meny-funksjonaliteten, og bruker klassen *StartLogic* for å opprette menyene som brukes før simuleringen starter, og klassen *EndLogic* for å opprette menyene som brukes etter simuleringen er ferdig. Metoden *SetupMenu(List<DetectableError> PotentialErrors)* kalles fra *GameController* når den er initialisert. Parameteren PotentialErrors inneholder alle objekter av klassen *DetectableError* som ble oppdaget i scenen av *DetectableController*. Metoden kaller videre på startSimLogic.SetupListOfErrorsInMenu(). De viktigste linjene fra denne metoden:

```
foreach (DetectableError d in potentialErrors)
{
    listeItem = UILogic.AddElementToList(gameObjectList, listItemPrefab);
    listeItemToggle = listeItem.GetComponent<Toggle>();
    listeItemToggle.onValueChanged.AddListener(d.ToggleError);
}
```

Den statiske metoden UILogic.AddElementToList:

```
public static GameObject AddElementToList(GameObject gameObjectList, GameObject
listElementPrefab)
    {
        GameObject scrollItemObj = Instantiate(listElementPrefab);
        scrollItemObj.transform.SetParent(gameObjectList.transform, false);
        return scrollItemObj;
    }
```

For hver feil i scenen instansieres et GameObject fra prefaben *ErrorToggle*. Komponenten *Toggle* danner en “checkbox” i spillet som brukes for å velge om en feil skal være aktiv i simuleringen. Komponenten har eventen *onValueChanged* som aktiveres når en bruker klikker på liste-elementet, og for hvert liste-element legges den relaterte feilen til som abonnent på denne eventen. Resultatet er at når en bruker haker av for at feilen skal aktiveres eller tar vekk haken i menyen, kalles den relevante feilen sin ToggleError()-metode, som faktisk aktiverer eller deaktiverer feilen i spillet, før brukeren har startet simuleringen.

Som tidligere nevnt er det i prosjektet brukt en arkitektur hvor det tilstrebes komposisjon over polymorfisk arv - men det brukes likevel noe arv, da klassen *DetectableError* arver fra klassen *Detectable*, og de forskjellige typene feil som beskrevet i avsnitt 4.2.1 implementeres i klasser

som arver fra *DetectableError* - hovedsakelig for å kunne overskrive metoden ToggleError() da forskjellige typer feil krever forskjellige implementasjoner av aktivering og deaktivering av feil.

Klassen *DetectableController* kaller på metoden GetDetectableChildren() i sin Start()-metode. Metoden finner alle entiteter i den innlastede scenen som er tilknyttet en *Detectable*-komponent, ved hjelp av Unity-funksjonen FindObjectsOfTypeAll. Disse lagres i en liste, og i tillegg lagres alle *DetectableError* i en separat liste:

```
List<Detectable> allDetectablesInScene = Resources.FindObjectsOfTypeAll<Detectable>().
 Where(
 go => go.gameObject.scene.name != null //Ikke inkluder ikke-instantierte prefabs
  ).ToList();

foreach(Detectable d in allDetectablesInScene)
{
        detectables.Add(d);

 //Oversikt over potensielle feil
        if (d is DetectableError)
        {
        PotentialErrors.Add((DetectableError)d);
        }
}
```

Klassen har også to viktige metoder som kalles når en bruker er ferdig i simuleringen: FinnAntallMarkerteObjekter() som danner en oversikt over objekter som er markerte i scenen, og debriefDetectables(), som ber alle markerte *Detectable* forandre farger for å vise brukeren feil som ble funnet og ikke funnet, og markerte objekter som ikke var en aktiv feil.

Klassen *Detectable* er ansvarlig for å holde rede på hvilke Unity-*Materials* som brukes i opptegning av tilhørende 3D-modeller. I klassens Start()-metode lagres dette i en liste av lister:

```
public List<List<Material>> orgMats;
internal virtual void Start()
    {
        orgMats = new List<List<Material>>();
        childrenderers = GetComponentsInChildren<Renderer>(true);
        //Lag oversikt over originale materialer
        foreach (Renderer render in childrenderers)
        {
            List<Material> m = new List<Material>();
            render.GetMaterials(m);
            orgMats.Add(m);
        }
    }
```

Denne informasjonen må lagres for at de tilhørende 3D-modellene kan fargelegges annerledes når de blir markert av brukeren, og for å kunne gå tilbake til de opprinnelige materialene hvis brukeren ombestemmer seg.

4.3 Brukeropplevelse

Design av brukeropplevelsen er basert på retningslinjer fra Oculus og Unity. Det er blitt utført hyppige gjennomganger både internt i gruppen og med visning til oppdragsgiver for å sikre at programmet var brukervennlig og brukeren fikk god nok tilbakemelding på hva som skjedde i scenen. Det ble fokusert på en rekke funksjoner i opplevelsen som måtte utvikles med brukeropplevelse som prioritet.

4.3.1 Brukergrensesnitt

Brukergrensesnittet består av flere deler deriblant hva bruker ser i opplevelsen, hvordan brukeren bruker håndkontrollere for å nå de forskjellige funksjonene i opplevelsen som å slå på laserstrålen, og hvilke andre knapper på kontrollen som skal brukes.

Prosjektet bruker i hovedsak tre knapper. Brukeren kan gå til pausemenyen ved å bruke “start-knappen” på Oculus kontrollen. Dette er etter retningslinjene fra Oculus [36] som nevnt i avsnitt 2.2.1, hvor noen knapper er beskrevet som faste - deriblant pause til “start-knappen” Brukeren kan også slå på laserpekeren med “Knapp 2” på høyre oculus kontroll. Dette er en av knappene Oculus-dokumentasjonen setter som flerbruks knapp. Knappen for markering av objekter brukes “Knapp 1” på høyere oculus kontroll. Denne knappen er også åpen for applikasjonsspesifikk bruk. Det var naturlig å ha disse knappene nærme når applikasjonen var i bruk. I tilegg til disse hovedknappene er det utviklet en metode for “gripe-knapp” som følger Oculus retningslinjer.

Brukeren bruker laserpekeren for markering av feil i scenen. Dermed var det viktig at den var godt synlig for brukeren. For å oppnå dette ble det brukt en farge på strålen med god kontrast i de utviklede scenene, og en tydelig markør som følger hvor strålen treffer. Dermed vil brukeren kunne se at det som blir markert i scenen er det brukeren faktisk mente å markere.

I henhold til oculus retningslinjer er menyene i opplevelsen i “Gamespace” i motsetning til festet til kamera, som beskrevet i avsnitt 4.1.2. Dette betyr at menyene ses på en fast posisjon i rommet i motsetning til at menyen følger brukeren. Disse skjermene er forsøkt å gjøre store nok til at brukeren har god oversikt uten å ta brukeren ut av opplevelsen.

4.3.2 Gjennomgang av normal bruk

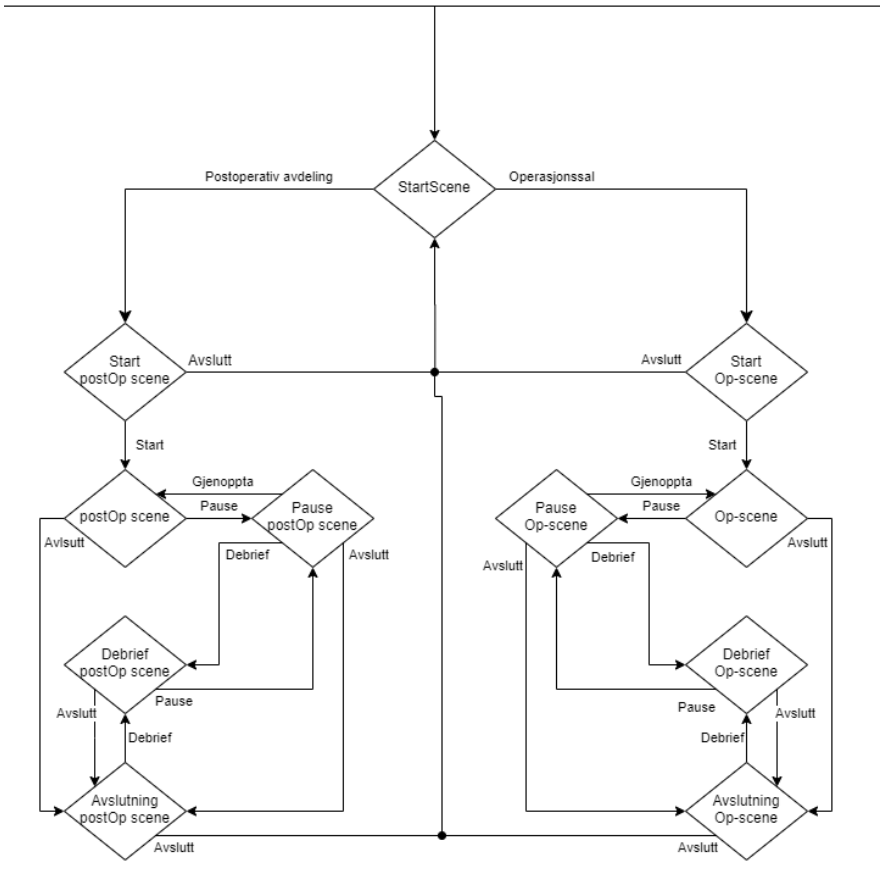
I dette delkapittelet beskrives normal gjennomgang av en simulering. Gjennomgangen vises også i et prosessdiagram i figur 22.

Ved oppstart av en scene blir spilleren plassert i prefaben *SceneStart*, hvor det tenkes at en lærer skal velge feil som skal aktiveres i scenen. En bruker kan velge å aktivere alle feil i scenen, utvalgte konkrete feil som vises på en av skjermene, eller tilfeldige feil. Brukeren kan også velge om simuleringen skal være tidsbestemt, og i tilfelle hvor mye tid som gis.

Deretter viser spillet en blank skjerm med mulighet til å starte simuleringen eller å avbryte. Dette gir brukeren som er ansvarlig for oppsett av opplevelsen mulighet til å overlevere VR-briller til brukeren som skal gå gjennom opplevelsen slik at det ikke er mulig for brukeren å jukse før man begynner opplevelsen. I denne mellom-scenen vil brukeren ha muligheten til å trykke start-knappen. Når brukeren klikker på knappen merket *Start* flyttes brukeren til en forhåndsdefinert posisjon i scenen, hvor simuleringen kan starte.

Brukeren vil deretter være presentert scenen. I scenen ligger det 3D-modellert medisinsk utstyr og personell. Det vil kunne forekomme kommunikasjon mellom personene i scenen som i seg selv kan være feil. Brukerens oppgave vil være å gå igjennom denne scenen og merke av de objektene den mener er feil. Brukeren har fri bevegelse i rommet og kan undersøke det meste av utstyr og informasjon.

Når brukeren mener seg ferdig med å inspisere omgivelsene - eller at angitt tid går ut - gis brukeren to muligheter for å avslutte: “Klikke” med laserpekeren på knapper i scenene merket “Avslutt simulering”, som er plassert på scenenes naturlige utgangsdører - eller bruke pauseknappen på håndkontrollen for å gå til pausemenyen. Her vil bruker kunne velge å fortsette inspeksjonen / opplevelsen, gå til Debrief eller avslutte. Figur 23 viser en avslutningsknapp plassert på en utgangsdør i operasjonsstuen, og figur 24 viser pause-menyen.

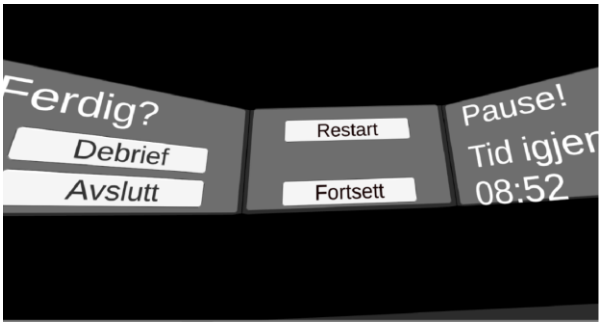


Figur 22 Prosessdiagram

Velger brukeren å avslutte vil brukeren tas tilbake til startmenyen nå med informasjon om hvor mange feil brukeren fant, hvilke feil brukeren fant, og hvilke andre objekter i rommet som ikke var ment som feil som ble markert. Dette vises til den som var ansvarlig for oppsettet og det kan dermed diskuteres hva brukeren har gjort i scenen. I fremtidige versjoner kan denne informasjonen tenkes å bli sendt til en database for det planlagte forskningsprosjektet. Menyen gir også brukeren muligheten til å gå til Debrief. Figur 25 viser avslutningsmenyen.

Om brukeren velger å gå til Debrief - enten direkte via pause-menyen, eller via statistikk-menyen - vil brukeren flyttes tilbake til scenen. Feil som ikke er markert vil nå ha rød farge, feil som ble markert vil ha grønn farge og andre objekter som ikke var ment som feil vil ha gul farge. Det vil også være en tekstboks over koblet til objektet med en forklaring på feilen. Figur 26 viser en bruker i Debrief. Etter at bruker har valgt å avslutte eller gå direkte til Debrief er tiden stoppet. Brukeren kan gå frem og tilbake mellom avslutningsmenyen og Debrief.

Figur 23: Avslutningsknapp på en naturlig utgangsdør



Figur 24: Pausemeny med mulighet for å avslutte eller å gå direkte til Debrief. Restart-knapp vil ikke være tilgjengelig for vanlige brukere.

Figur 25: Avslutningsmeny for å se oversikt over feil og mulighet for å gå til Debrief.

Figur 26: Debrief. Korrekt markerte objekter vises grønne, og ikke markerte aktive feil vises røde.

4.4 Tilrettelegging for videreutvikling

Det ble tidlig etablert at prosjektet skulle være utviklingen av en plattform for videreutvikling. Det var derfor viktig at designet av prosjektet var skalerbart og mulig å videreutvikle. For å oppnå dette ble det laget prefabs med scene-kritisk logikk, og i utvikling av prosjektets kode var det fokus på uavhengighet.

4.4.1 Prefabs

Det er laget prefabs med kritisk logikk som skal være i alle scenene. Dette er for å tilrettelegge for utvikling av flere scener i fremtidig videreutvikling. Det er opprettet prefabs med definerte materialer for tilbakemelding på deteksjon, prefabs med scene kritisk logikk som scene kontroller, start- og avslutningsmenyer, og de forskjellige feiltypene. Dette skal gjøre utviklingen av nye scener rask i videreutvikling.

4.4.2 Kodebase

Prosjektets kodebase er generisk nok til å utvikles videre til spesialiseringer og alt er laget i eget namespace slik at utvidelse av kode ikke skal skape konflikter med navn og lignende laget for dette prosjektet. Som beskrevet i avsnitt 4.2.2 er det tilstrebet lav kobling mellom objekter, og at objekter i så stor grad som mulig skal være uavhengige av hverandre. Dette er oppnådd ved blant annet mye bruk av C# Events, og fører til at klasser kan byttes ut med andre, potensielt mer avanserte klasser, eller klassenes funksjonalitet kan utvides. Hvis det implementeres nye feil i scenene vil disse feilene bli oppdaget dynamisk, i stedet for at noe relatert til konkrete feil er hardkodet.

5 EVALUERING

Som nevnt i avsnitt 3.5 velges det å fokusere på akseptansetesting for evaluering: Oppfyller prosjektet og det resulterende spillet oppdragsgiver sine krav? I dette kapittelet beskrives dette nærmere.

5.1 Evalueringsmetode

For å vurdere om produktet oppfyller oppdragsgiver sine krav, ble det avholdt ukentlige møter med oppdragsgiver via Zoom, hvor forrige ukes fremgang og nye implementasjoner ble diskutert og demonstrert, og oppdragsgiver fikk mulighet til å gi hyppige tilbakemeldinger. Ideelt sett burde dette vært gjennomført ved at oppdragsgiver selv jevnlig forsøkte spillet i VR, men på grunn av koronavirus-situasjonen lot dette seg ikke gjennomføre.

I tillegg ble det avholdt en større demonstrasjon mot slutten av utviklingstiden for oppdragsgiver og tre personer i målgruppen - en faglærer fra hver av de tre relevante masterutdanningene ved HVL. Denne løsningen er heller ikke optimal, da de ikke får oppleve innlevelsen som kan oppnås ved bruk av VR, som var en sterk motivasjon for å utvikle spillet i VR i motsetning til en tradisjonell desktop-applikasjon. Et av oppdragsgiver sine initielle krav var fokus på brukbarhet og brukervennlighet, da det ikke kan forventes at brukere i målgruppen har erfaring med bruk av VR. Viktige fokusområder under utviklingen har derfor vært å tilrettelegge og å forebygge eller redusere potensielt ubehag for uerfarne brukere. Det vil være vanskelig å evaluere om dette kravet er oppfylt ved en demonstrasjon over videokonferanse hvor brukere ikke får oppleve spillet gjennom VR-briller.

Mer konkret har det under utviklingsprosessen spesielt dukket opp to spørsmål relatert til de anvendte spillutviklings-metodene: om den implementerte teleporterings-metoden for forflytning av spilleren i scenene er enkel nok å bruke, og om feil i scenen er tydelige nok til at det ikke er nødvendig for en bruker å studere objekter på svært nært hold, noe som kan føre til ubehag. Teleporteringen vil ikke være mulig å evaluere uten at en testperson faktisk holder en håndkontroll i sine egne hender og forsøker å utføre kombinasjonen av knappetrykk som kreves. Det vil til en viss grad være mulig å evaluere om feil i scenen er tydelig nok via videokonferanse, ved at deltakere vurderer feilenes synlighet. Dette vil likevel ikke skikkelig gjenskape situasjoner hvor en bruker vil studere et objekt nærmere, da VR-brillene styres av utviklere som etter utviklingen har bygd opp en del erfaring med bruk av VR, og ikke er like utsatt for potensielt ubehag. I tillegg vet utviklerne hvor feilene ligger i scenene og hva som skal ses etter. Dette betyr også at det vil være vanskelig å vurdere om det er gjort tilstrekkelige tiltak for å gjøre opplevelsen generelt behagelig for brukere, da utviklerne kan ha en høyere toleranse enn en uerfaren bruker. Det velges likevel å inkludere opplevd komfort som en subjektiv vurdering, og som en kvantitativ vurdering ses det på målt bildefrekvens i spillet ved å bruke den implementerte debug-skjermen som beskrives i avsnitt 4.2.1, siden lav bildefrekvens kan være sterkt bidragsytende til opplevd ubehag [31].

Den større demonstrasjonen ble gjennomført ved at utviklerne styrer VR-brillene, og viser det som ses i VR-brillene ved å vise skjermbildet i Unity-editoren til oppdragsgivere og faglærere via Zoom. Faglærere inntok først rollen som student, og ble bedt om å fortelle utviklere hvor de skal gå i rommet, og hva som skal markeres som feil. Det ble gått gjennom både operasjonsstuen og den postoperative avdelingen, med en tidsavgrensning satt til 10 minutter. Deretter ble oppdragsgiver og faglærere vist menyene de er tenkt å bruke før en student overtar. Gjennomgangen var tenkt å gi utviklerne et inntrykk av hvordan brukeropplevelsen evalueres av brukere i målgruppen.

Etter demonstrasjonen ble oppdragsgiver og faglærerne stilt spørsmål som var forberedt på forhånd, hovedsakelig om spillet er noe de ser for seg de kan bruke som et hjelpemiddel i undervisning i fremtiden.

For vurdering av om konkrete funksjonaliteter fungerer som forventet, ble det gjennomført fortløpende testing av utviklerne etter at en ny funksjonalitet ble implementert. Unity lar utviklere kjøre utviklingsprosjekter direkte i Unity-editoren, som om det er gjennomført en full kompilering av prosjektet. For dette prosjektet kan en slik full kompilering ta opp mot 15-20 minutter. Ved utvikling mot Oculus Quest har dette tidligere ikke vært mulig siden disse alt-i-ett-brillene benytter seg av sitt eget Android-operativsystem - men ved hjelp av Oculus Link som nevnes i delkapittel 3.1.4, kan brillene brukes som datamaskin-avhengige VR-briller, og dermed kunne kjøre prosjekter umiddelbart i Unity-editoren. Dette kan derimot ikke brukes for å vurdere bildefrekvens, siden brillene bruker den tilkoblede datamaskinens ressurser for grafiske beregninger og opptegning.

5.2 Evalueringsresultat

5.2.1 Oppdragsgivers evaluering

Oppdragsgivers initielle krav om scener som skulle implementeres og hva en bruker skal gjøre i spillet ble i stor grad møtt tidlig i utviklingsprosessen. Mye av utviklingen har derfor vært preget av implementasjon av konkrete feil i scenene, og tilrettelegging for senere videreutvikling, uten et tydelig endepunkt. Som nevnt i avsnitt 1.3 var det tidvis vanskelig å avgjøre om elementer skulle implementeres i løpet av denne utviklingsprosessen, eller om en implementasjon skulle utsettes til senere videreutvikling i et eventuelt masterprosjekt. Dette har vært diskusjonstema ved flere av de ukentlige møtene med oppdragsgiver, og har ført til at ønskede implementasjoner og krav har utviklet seg underveis gjennom en samarbeidsprosess.

Tilbakemeldingene fra oppdragsgiver har vært positive. Mot slutten av utviklingsprosessen gir oppdragsgiver uttrykk for å være fornøyd med resultatet, og at det ses frem til videre utvikling av produktet. Grunnleggende funksjonalitet er tilfredsstillende implementert.

5.2.2 Målgruppes evaluering

Den utvidete demonstrasjonen for faglærere i målgruppen ble avholdt to uker før avslutning av utviklingstiden. Demonstrasjonen anses som vellykket. De gir uttrykk for at de kan se for seg at spillet er noe de kan bruke som et hjelpemiddel i undervisning, spesielt etter videre utvikling. Det bemerkes også at spillet ikke bare kan gjøre simulering av "Room of Horror" enklere, men at det også kan bidra til å skape nye undervisningsmuligheter. Lærerne sier at spillet kan bidra til å gi nye perspektiver som har vært vanskelig å undervise om, og at spillet kan skape muligheter for å lære studenter på en ny og annerledes måte. Det å oppdage feil i rommene på denne måten gir rom for refleksjon og diskusjon, og spesifikke feil kan føre til samtaler om hva disse feilene kan føre til.

Faglærerne bemerker flere feil i scenene som er resultater av manglende detaljer i de brukte 3D-modellene. Noen av disse endres før utviklingen avsluttes - for eksempel endres noen farger på enkelte 3D-modeller - men noen krever mer modelleringsarbeid enn utviklerne har tid til, og må derfor forbedres i en videreutvikling. Generelt følte faglærerne at feilene var tydelige nok, uten å være for lett å oppdage. Det bemerkes at noen feil kanskje var lett å oppdage for dem som er erfarne spesialsykepleiere, men kan være vanskeligere for studenter med mindre erfaring. Under demonstrasjonen ble det bevisst forsøkt å holde en avstand på ca. en halv meter, som beskrevet i avsnitt 2.2.1. Som nevnt i avsnitt 5.1 er det vanskelig å evaluere om feil er tilfredsstillende tydelig gjennom en videokonferanse i stedet for gjennom utprøving av spillet i VR, men med evalueringen som var mulig å gjennomføre under utviklingstiden velges det å konkludere med at de implementerte feilene er akseptabelt tydelige.

Faglærerne etterspør spesifikt et større fokus på kommunikasjon mellom personene som ses i simuleringene i fremtidige versjoner. Feil eller misforståelser i kommunikasjon kan være en viktig kilde til uønskede hendelser. Dette bør derfor være en prioritet i videre utvikling. Oppsummert er faglærerne positiv til spillet, og entusiastiske over fremtidige versjoner.

5.2.3 Tekniske krav

Kravet om tilrettelegging for brukere med lite erfaring med bruk av VR blir vanskelig å evaluere uten brukertesting. De største spørsmålene forbundet med dette er om markering av objekter og bevegelse av spilleren i scenen er lett å gjennomføre. Det er implementert en innføring i bruk av håndkontrollerne før brukeren plasseres i rommene det skal letes etter feil i slik at en bruker kan øve på forhånd - men det at en bruker *klarer* å bruke håndkontrollerne til å flytte seg i scenen betyr ikke nødvendigvis at bruken av håndkontrollerne føles naturlig og enkelt.

Det må derfor ses på hvordan utviklerne opplever bruk av spillet, fra perspektivet til brukere som har erfaring med bruk av håndkontrollere og VR. Markering av objekter i scenen oppleves enkelt, og det er lett å se hvilket objekt som skal markeres. Bevegelse i scenen føles naturlig. Enkelte objekter i scenen kan være vanskelig å bevege seg forbi, og det kan enkelte ganger

oppleves at en posisjon man flytter seg til har et dårligere perspektiv for å studere en feil enn man så for seg. Det oppleves likevel enkelt å korrigere slike perspektiv ved å flytte seg på nytt.

Utviklerne opplever ikke noe ubehag ved bruk av spillet. Spillet oppleves responsivt, og det er ikke observert forsinkelser mellom at en bruker beveger hånden, og den samme bevegelsen gjengis i spillet - noe som kan føre til ubehag, som beskrevet i avsnitt 2.2.1.

Det ses på bildefrekvens som et objektivt mål for potensiale for ubehag. Spillet opprettholder stort sett 72 bilder per sekund. Når brukeren ser fra et perspektiv med oversikt over hele scenen kan det oppleves et fall i frekvens til ca 65 bilder per sekund. Spillet når derfor ikke opp til Oculus sine anbefalinger om å opprettholde konstante 72 bilder per sekund, som nevnt i avsnitt 4.1.2. Kravet anses likevel som oppfylt etter samtaler med oppdragsgiver.

Oppdragsgiver anser sine initielle krav som oppfylt. Produktet aksepteres, og akseptansetest anses dermed som bestått.

6 DISKUSJON

Utviklingen har vært preget av koronavirus-situasjonen. Flere spørsmål om brukeropplevelsen er ubesvarte, da det ikke har vært mulig å gjennomføre skikkelige brukertester.

Valget om å utvikle mot alt-i-ett-enheten Oculus Quest har hatt konsekvenser for utviklingen i spillet. Arbeid med å oppnå så høy grad av realisme som mulig innen begrensningene gitt av enhetens ressurser har vært tidkrevende. I den initielle tidsplanen ble det planlagt å utsette arbeid med optimalisering til de siste to ukene av utviklingstiden, men dette arbeidet viste seg å kreve mer tid, og preget de siste fire ukene. Optimalisering og vurderinger av hva som er grafisk mulig burde sannsynligvis vært en mye større del av utviklingsforløpet, og vært viet mer oppmerksomhet fra starten av utviklingen. Ett av de opprinnelige målene var å fokusere arbeidet på funksjonalitet og tilrettelegging for videreutvkling, men tidvis har sannsynligvis arbeid med å legge til innhold og feil i scenene hatt et større fokus enn nødvendig, og krevd tid og ressurser som kunne vært brukt på optimalisering. Som nevnt i avsnitt 5.2 kan spillet i enkelte situasjoner ha en bildefrekvens på under 70 per sekund, og dette kunne sannsynligvis forbedres med ytterligere optimaliseringsarbeid. Samtidig betyr dette arbeidet at spillet sannsynligvis vil kunne kjøre med en høyere bildefrekvens på datamaskin-avhengige VR-briller, som nevnt i avsnitt 3.2. I fremtidig videreutvikling bør det diskuteres om det skal endres VR-plattform, eller om spillet skal ha et annet grafisk uttrykk som er mindre realistisk, men mindre grafisk krevende.

Alt-i-ett-enheten gir begrensede muligheter for realistisk opplysning og skyggelegging i spillet. Dette kan være godt nok for relativt statiske og stillestående scener, men kan ha stor betydning for opplevd realisme i mer animerte scenarioer. Det har vært forsøkt å importere noen svært realistiske 3D-modeller, men enkelte av disse har alene ført til lav bildefrekvens - helt ned mot 20 bilder per sekund på grunn av én modell - og måtte derfor erstattes med mindre realistiske modeller. I tillegg har det vært nødvendig å bruke mye tid på å gjøre enkelte andre 3D-modeller mindre komplekse. Dette har bidratt til en høyere bildefrekvens og dermed en bedre brukeropplevelse, men dette kommer på bekostning av noe realisme. Samtidig - hvis det hadde blitt valgt å utvikle mot en annen plattform og å prioritere realisme over tilgjengelighet, ville realismen som var mulig å oppnå være noe begrenset av utviklernes erfaringer innen 3D-modellering. Det har vært nødvendig å lage flere modeller selv, og enkelte av disse er av en tydelig lavere kvalitet enn modellene som ble innkjøpt fra profesjonelle utviklere. En oversikt over innkjøpte modeller kan ses i rapportens appendix, i kapittel 10.3. Grafisk realisme i spillet har vært lite diskutert med oppdragsgiver, men det kan potensielt være nyttig å ta dette opp mer eksplisitt for fremtidig videreutvikling, spesielt hvis animerte situasjoner skal være en større del av brukeropplevelsen, siden animasjoner som ikke er gode nok kan oppleves som svært lite realistisk. Det har heller ikke vært mye diskutert i hvor stor grad en bruker opplever innlevelse i scenene, siden dette ville vært vanskelig å vurdere uten mer brukertesting. Brukerinnlevelse kan tenkes å være et viktigere spørsmål enn opplevd realisme, spesielt siden dette var en motivasjon for valg av utvikling til VR, som nevnt i avsnitt 5.1.

Dette gir også en grunn for å diskutere simuleringens realisme uavhengig av grafisk realisme. Tidlig i prosjektarbeidet nevnte oppdragsgiver mulighet for å gi studentene en omvisning ved SimArena og potensielt også relevante avdelinger ved Haukeland Universitetssykehus. Dette lot seg ikke gjennomføre på grunn av koronavirus-situasjonen. Scenenes utseende og innredning har derfor basert seg på bilder fra internettsøk, og at en av utviklerne er tidligere utdannet operasjonssykepleier, som nevnt i avsnitt 1.4. Denne tidligere utdanningen og erfaringen fra tidligere arbeid ved Haukeland Universitetssykehus har ført til mye kjennskap til hvordan en operasjonsstue ser ut, men ikke fullt så mye kjennskap til postoperative avdelinger. Det anses derfor som sannsynlig at operasjonsstue-scenen er generelt representativ for en faktisk operasjonsstue, i større grad enn scenen av den postoperative avdelingen er representativ for en faktisk postoperativ avdeling. I begge scenene mangler sannsynligvis flere detaljer, som kan være nødvendig for å oppnå høy grad av brukerinnlevelse. Noen av disse manglende detaljene ble påpekt av faglærere under den utvidete demonstrasjonen. Manglende detaljer anses likevel som akseptabelt da både utviklere og oppdragsgiver har sett for seg at slike detaljer kan legges til scenene i senere videreutvikling, da det kan disponeres mer tid til innredning av rommene. For fremtidig utvikling og implementasjon av flere feil vil det sannsynligvis være nødvendig med besøk på SimArena sine øvingslaboratorium for å få bedre innsikt i rommene og rommenes inventar, og tettere samarbeid med faglærerne.

Utviklerne er fornøyde med valg av verktøy. Bruk av Unity har vært relativt uproblematisk, og siden en av utviklerne hadde erfaring med verktøyet fra før, gikk det raskt å etablere grunnleggende funksjonalitet. *Unity Collaborate* som versjonskontroll-verktøy har fungert bra, men prosjektgruppen har savnet mulighet for å opprette “branches” i koden for å utforske eksperimentelle implementasjoner. I tillegg har det tidvis oppstått frustrasjon over at det ikke er mulig at flere utviklere gjør endringer i samme scene samtidig, men dette kunne sannsynligvis vært unngått ved i større grad å utvikle ved bruk av prefabs. Ved senere videreutvikling bør det vurderes å bruke nyere versjoner av Unity. Det ble valgt å bruke Unity-versjonen 2018.4 for å bruke minst mulig tid på potensielle feil som kunne oppstå i oppdateringer og nye versjoner. Dette var i stor grad vellykket, men det betydde at det ikke var mulig å benytte seg av nye funksjoner i senere versjoner. I 2019- og 2020-versjoner er det blant annet implementert en betydelig forbedring i utvikling av menyer og brukergrensesnitt som kunne spart mye tid i dette prosjektet.

Det har også vært mye bruk av modelleringsverktøyet *Blender*. Verktøyet kan ha en bratt læringskurve, og tidlig i utviklingsprosessen ble det brukt mye tid på å tilegne seg erfaring. Som nevnt er det svært varierende kvalitet mellom enkelte av 3D-modellene som ble opprettet, og etterhvert som utviklerne ble mer vant med verktøyet er det brukt en del tid på å forbedre modeller som ble utviklet tidlig. Utviklerne opplevde en tydelig progresjon og læring i modelleringsarbeidet, og kunne modellere betydelig raskere og bedre mot slutten av utviklingsprosessen.

Når det gjelder selve implementasjonen av spillet kan det stilles spørsmål ved implementasjonen av *DetectableController* og *DetectableError* i Unity. Som i avsnitt 4.2.1 ses det på feilen “Tom hånddesinfeksjon” som et eksempel. I den nåværende implementasjonen

ligger en instans av *DetectableController*-prefaben blant de øverste elementene i scene-hierarkiet, som et GameObject. GameObjectet som definerer feilen “Tom hånddesinfeksjon” er tilknyttet den passende feil-komponenten, og dette GameObjectet ligger som barn av *DetectableController*-GameObjectet. Videre ligger GameObjectene som definerer hånddesinfeksjonsbeholderen og dens grafiske egenskaper - hvor i verden befinner objektet seg, hvordan det ser ut og hvordan det skal fargelegges - som et barn av GameObjectet som definerer feilen. Strukturen kan ses i figur 27. Dette betyr at 3D-modellene som viser selve hånddesinfeksjonsbeholderen er posisjonert i scenen relatert til GameObjectet som definerer feilen, som igjen er posisjonert i scenen relatert til GameObjectet som er tilknyttet *DetectableController*. Denne strukturen gjør det svært lett å ha kontroll over feil i scenen, og i den tilhørende C#-koden er det svært lett å få oversikt over materialer tilknyttet 3D-modellene, som er nødvendig for å kunne tydelig markere modellene for brukeren. Ikke minst er det enkelt å definere at brukeren markerer selve feilen, i stedet for at brukeren markerer 3D-modellen uten at feilen er klar over at den er markert. Samtidig har det store konsekvenser for håndtering av feilenes transformasjons-egenskaper, da det å reposisjonere feil i scenen kan føre til problemer, siden modellens altså er posisjonert relativt til to andre abstrakte GameObjects - enda flere for enkelte andre feil - som strengt tatt ikke trenger en posisjon i verdenskoordinater i det hele tatt. Dette har hatt lite betydning når scenene er relativt statiske og stillestående da det skjer lite reposisjonering under kjøring, men kan gi utfordringer i senere utvikling hvis det implementeres mer dynamiske scener hvor objekter beveger seg i forhold til hverandre. Det bør derfor utforskes en annen struktur der en feil for eksempel heller gis referanser til 3D-modellene, som i større grad kan være posisjonert relativt til andre objekter i scenen, og ikke til abstrakte, usynlige elementer.

Som nevnt i avsnitt 5.2 har det tidvis vært vanskelig å definere et endepunkt for utviklingen på grunn av muligheten for senere videreutvikling. Det er derfor vanskelig å anse spillet som ferdigutviklet. Det opprinnelige målet for prosjektet var å utvikle en prototype som et grunnlag for videreutvikling, men i løpet av utviklingsprosessen har det oppstått spørsmål om det faktisk i større grad er utviklet et minste brukbare produkt. Dette kan diskuteres, og det kan være vanskelig å sette en grense for å skille de to konseptene. Spillet er fullt funksjonelt, og det er mulig for en bruker å gå gjennom en “Room of Horror”-simulering i VR. Det er ikke stor variasjon og bredde i feilene brukeren kan finne og markere, men den implementerte funksjonaliteten kan benyttes til å opprette flere feil, og spillets funksjonalitet og spillet i seg selv er fullt mulig å bygge videre på og å utvide. Mot slutten av utviklingsperioden velges det derfor å beskrive prosjektets resultat som et minste brukbare produkt.

Figur 27: DetectableController øverst i scene-hierarkiet. Feilen “Tom hånddesinfeksjon” som barn av DetectableController. Hånddesinfeksjon som barn av feilen. GameObjects med 3D-modeller som barn av Hånddesinfeksjon-prefaben.

7 KONKLUSJON

Målene for dette prosjektet innebar å utvikle et VR-miljø for simulering av “Room of horror” for sykepleiere i masterutdanning på HVL med scener for postoperativ avdeling og en operasjonsstue. Prosjektet har nådd disse målene ved utvikling av de grunnleggende funksjonene for en opplevelse av “Room of horror” i spillmotoren Unity, ved bruk av metoder for spillutvikling.

Det er utviklet et minste brukbare produkt av et “Room of Horror” i VR. Produktet anses som et grunnlag for videre utvikling. Produktet er et spill som inneholder en virtuell operasjonsstue og en virtuell postoperativ avdeling. Det er utviklet og designet algoritme for hvordan man merker, aktiverer og viser feil, og det er implementert funksjonalitet for å bevege seg i rommene. En lærer kan før simuleringen starter velge hvilke feil som skal være aktive i scenene, før en student kan markere objekter i scenen som oppfattes som feil. Etter simuleringen får studenten tilbakemelding om hvor mange feil som ble funnet, og hvor mange feil som ikke ble funnet.

Oppdragsgiver sine initielle krav om spillets innhold er oppfylt. Så langt det har vært mulig å vurdere gitt koronavirus-situasjonen er det tilrettelagt for brukbarhet og brukervennlighet. Oppdragsgiver aksepterer produktet.

Resultatet av denne prosjektperioden kan videreutvikles til å gå dypere i de nåværende scenene, men det kan også videreutvikles til å inneholde andre situasjoner i helseinstitusjoner. Som nevnt i avsnitt 2.1.2 ble det funnet en video av et lignende produkt som ser ut til å omhandle en vanlig sengepost ved et sykehus. Det kan tenkes at “Room of horror” kan være et nyttig verktøy også for bachelorutdanninger i sykehus, hvor slike situasjoner vil være mer relevante.

Produktet kan også utvikles som en plattform til å utvikle opplæringsverktøy for andre læringssituasjoner hvor detaljer i omgivelsene er viktig. Dette kan tenkes for eksempel for pilot- eller mekaniker-utdanninger med simulert inspeksjon av maskineri eller instrumenter, men også for trening av andre yrker hvor inspeksjon eller protokoller skal følges. Det er derfor mange muligheter for videreutvikling.

Konkrete ideer og forslag til videreutvikling i en masteroppgave beskrives i kapittel 8.

Før overlevering av produktet til oppdragsgiver ble det skrevet en installasjonsveiledning, etter ønske fra oppdragsgiver. Det ble i tillegg skrevet en utfyllende teknisk dokumentasjon for fremtidige utviklere.

8 VIDEREUTVIKING

Videreutvikling av dette prosjektet kan gå mange veier på grunn av prosjektets design og fokus på videreutvikling. Prosjektet har noen funksjoner som eksplisitt savnes av faglærere og oppdragsgiver, og enkelte scenarioer og feil oppdragsgiver ønsker at vil bli utviklet senere. Faglærere har kommet med flere forslag til flere feil som kan implementeres, og konkrete scenarioer som kan utspille seg.

Oppdragsgiver ønsker at produktet i fremtiden skal handle mer om situasjoner og kommunikasjon mellom karakterer. Dette er lagt et grunnlag for, men siden det ville vært nødvendig å bruke mye tid og ressurser for å utvikle manus og innspilling av kommunikasjon, animasjoner og lignende, ble fokuset lagt på å lage mer visuelle statiske feil. Det er store muligheter for videre forbedring av dette. Et eksempel på et konkret scenario som har vært tatt opp gjentatte ganger er at en anestesisykepleier kan følge en pasient fra operasjonsstuen til den postoperative avdelingen, og gå gjennom en simulert overlevering av pasienten og oppleve kommunikasjon mellom anestesisykepleieren og intensivsykepleieren, som kan være en viktig kilde til uønskede hendelser.

Prosjektgruppen har også sett for seg muligheter for å gjøre slik at erfaringen ikke nødvendigvis er nødt til å starte med at lærer eller ansvarlig administrator setter opp aktive feil og scenen i VR-brillene før en student kan begynne simuleringen, men at læreren heller kan være på en annen maskin og sette opp og endre på scenen mens brukeren er i opplevelsen. Dette ble ikke gjort i løpet av denne prosjektperioden ettersom det krever utvikling av kommunikasjon mellom enheter - som er tidkrevende og utfordrende å utvikle.

Med kommunikasjon mellom enheter vil det også være mulig å hente ut data fra en students gjennomgang av en simulering som kan brukes til å måle læring og påvirke diskusjonen etter opplevelsen er gjort. Med denne funksjonaliteten vil det være mulig for en lærer å ha flere studenter i opplevelsen og å hente data fra alle studentene uten å måtte følge opp hver enkelt i opplevelsen. Dette vil i tillegg være svært nyttig for det planlagte forskningsprosjektet.

Som nevnt i avsnitt 2.2.2 kunne det også vært spennende å utforske muligheten for å utvikle simuleringer som kan gjennomgås av flere personer sammen, noe som kan gi et eget læringsutbytte.

9 REFERANSER

[1] Høgskulen på Vestlandet, «Klinisk sykepleie - Anestesisykepleie». https://www.hvl.no/studier/studieprogram/2020h/ma-ksa/ (åpnet mai 25, 2020).
[2] Høgskulen på Vestlandet, «Klinisk sykepleie - Intensivsykepleie». https://www.hvl.no/studier/studieprogram/2020h/ma-ksi/ (åpnet mai 25, 2020).
[3] Høgskulen på Vestlandet, «Klinisk sykepleie - Operasjonssykepleie». https://www.hvl.no/studier/studieprogram/2020h/ma-kso/ (åpnet mai 25, 2020).
[4] Helse- og Omsorgsdepartementet, «Pasientsikkerhetsprogrammet I trygge hender 24-7», *Regjeringen.no*, okt. 02, 2014. https://www.regjeringen.no/no/dokumenter/Pasientsikkerhetsprogrammet-I-trygge-hender-24-7/id2005291/ (åpnet mar. 24, 2020).
[5] Helsedirektoratet, «Pasientskader i Norge 2018». https://www.helsedirektoratet.no/rapporter/pasientskader-i-norge/sammendrag (åpnet mar. 24, 2020).
[6] Pasientsikkerhetsprogrammet, «I trygge hender 24 - 7», *Pasientsikkerhetsprogrammet*. https://pasientsikkerhetsprogrammet.no/om-oss/om-pasientsikkerhetsprogrammet/i-trygge-hender-24-7 (åpnet mar. 24, 2020).
[7] W. M. Nehring og F. R. Lashley, «Nursing Simulation: A Review of the Past 40 Years», *Simul. Gaming*, bd. 40, nr. 4, s. 528–552, 2009, doi: 10.1177/1046878109332282.
[8] Connell School of Nursing - Boston College, «A brief history of nursing simulation», *Boston College*. https://www.bc.edu/bc-web/schools/cson/cson-news/Abriefhistoryofnursingsimulation.html (åpnet mai 01, 2020).
[9] M. Barland, «Teknologi for livslang læring - fjernt, nært og simulert», *Teknologirådet*, nov. 01, 2018. https://teknologiradet.no/publication/teknologi-for-livslang-laering-fjernt-naert-og-simulert/ (åpnet mar. 24, 2020).
[10] C. Anthes, R. J. Garcia-Hernandez, M. Wiedemann, og D. Kranzlmuller, «State of the art of virtual reality technology», bd. 2016-, s. 1–19, 2016, doi: 10.1109/AERO.2016.7500674.
[11] S. Rogers, «2019: The Year Virtual Reality Gets Real», *Forbes*. https://www.forbes.com/sites/solrogers/2019/06/21/2019-the-year-virtual-reality-gets-real/ (åpnet mar. 24, 2020).
[12] B. D. Nagel, «Education to Help Drive VR Growth», *THE Journal*, feb. 26, 2020. https://thejournal.com/articles/2020/02/26/education-to-help-drive-vr-growth.aspx (åpnet mar. 24, 2020).
[13] J. M. Farnan *mfl.*, «Patient safety room of horrors: a novel method to assess medical students and entering residents' ability to identify hazards of hospitalisation», *BMJ Qual. Saf.*, bd. 25, nr. 3, s. 153, 2016, doi: 10.1136/bmjqs-2015-004621.
[14] A. S. Olson, N. Olson, J. L. Wilson, A. E. Muck, R. Garcia, og K. S. Balhara, «38 Room of Horrors: A Pilot Curriculum to Enhance Nurses' Patient Safety Awareness», *Ann. Emerg. Med.*, bd. 72, nr. 4, s. S18–S19, 2018, doi: 10.1016/j.annemergmed.2018.08.043.
[15] B. Murphy, «4 hazards medical trainees face in hospital "room of horrors"», *American Medical Association*, okt. 22, 2018. https://www.ama-assn.org/education/accelerating-change-medical-education/4-hazards-medical-trainees-face-hospital-room (åpnet mar. 24, 2020).
[16] M. Molloy og A. Clay, «"Room of Horrors": Engaging Interprofessional Students in a Hazards of Hospitalization Simulation», Duke University School of Nursing, Åpnet: mar. 26, 2020. [Online]. Tilgjengelig på: https://sigma.nursingrepository.org/bitstream/handle/10755/622529/Malloy+Room+of+Horrors.pdf;jsessionid=EF74485FCC99681692C51BCA9A052988?sequence=1.

[17] Høgskulen på Vestlandet, «Velkomen til SimArena - labber for simulering», *Høgskulen på Vestlandet*, 15/1-20. https://www.hvl.no/om/organisering/fhs/simarena/ (åpnet apr. 02, 2020).
[18] Høgskulen på Vestlandet, «Velkommen til SimArena», sep. 25, 2019. https://www.hvl.no/om/simarena/ (åpnet mar. 24, 2020).
[19] Høgskulen på Vestlandet, «SimArena», *YouTube,* jun. 29, 2017. https://www.youtube.com/watch?v=w2V74eFGesM&feature=emb_logo (åpnet mai 31, 2020).
[20] M. Ravik, A. Havnes, og I. T. Bjørk, «Defining and comparing learning actions in two simulation modalities: students training on a latex arm and each other's arms», *J. Clin. Nurs.*, bd. 26, nr. 23–24, s. 4255–4266, 2017, doi: 10.1111/jocn.13748.
[21] C. Indhumathi, W. Chen, Y. Cai, og C. Buche, «Multi-Modal VR for Medical Simulation», *Int. J. Virtual Real.*, bd. 8, nr. 1, s. 1–7, 2015, doi: 10.20870/IJVR.2009.8.1.2707.
[22] Acadicus, «Acadicus», *Acadicus VR Training and Education Platform*. https://acadicus.com/ (åpnet mar. 25, 2020).
[23] Oxford Medical Simulation, «Oxford Medical Simulation - Virtual Reality Healthcare Training», *Oxford Medical Simulation*. http://oxfordmedicalsimulation.com/ (åpnet mar. 25, 2020).
[24] SimX, «SimX VR and AR Medical Simulation – The most advanced medical simulation software on the market». https://www.simxar.com/ (åpnet mar. 25, 2020).
[25] Osso VR, «Osso VR - Virtual Reality Surgical Training Platform», *Osso VR*. https://ossovr.com/ (åpnet mar. 25, 2020).
[26] LapSim, «Surgical Training for Everyone | LapSim® Essence», *Surgical Science*. https://surgicalscience.com/systems/lapsim/lapsim-essence/ (åpnet mar. 25, 2020).
[27] F. W. Kron *mfl.*, «Using a computer simulation for teaching communication skills: A blinded multisite mixed methods randomized controlled trial.», *Patient Educ. Couns.*, bd. 100, nr. 4, s. 748–759, 2017, doi: 10.1016/j.pec.2016.10.024.
[28] J. LaMantia, «Virtual-reality simulations offer medical residents hands-on practice», *Modern Healthcare*, aug. 20, 2018. https://www.modernhealthcare.com/article/20180820/NEWS/180829997/virtual-reality-simulations-offer-medical-residents-hands-on-practice (åpnet mar. 25, 2020).
[29] Jiayi, «VR-Room of horror (Beta v1)», *Vimeo*. https://vimeo.com/375317730 (åpnet mar. 25, 2020).
[30] A. Somrak, I. Humar, M. S. Hossain, M. F. Alhamid, M. A. Hossain, og J. Guna, «Estimating VR Sickness and user experience using different HMD technologies: An evaluation study», *Future Gener. Comput. Syst.*, bd. 94, s. 302–316, 2019, doi: 10.1016/j.future.2018.11.041.
[31] C. Williams, «How to Prevent Simulator Sickness», *GameDev Academy*, mar. 29, 2018. https://gamedevacademy.org/how-to-prevent-simulator-sickness-vr/ (åpnet mar. 26, 2020).
[32] Unity Learn, «Design, Develop, and Deploy for VR», *Unity Learn*. https://learn.unity.com/course/oculus-vr (åpnet mar. 25, 2020).
[33] Oculus Developers, «Introduction to Best Practices». https://developer.oculus.com/design/ (åpnet mar. 26, 2020).
[34] Oculus Developers, «Vision». https://developer.oculus.com/design/bp-vision/ (åpnet mar. 26, 2020).
[35] Oculus Developers, «Locomotion». https://developer.oculus.com/design/bp-locomotion/ (åpnet mar. 26, 2020).
[36] Oculus Developers, «User Input». https://developer.oculus.com/design/bp-userinput/ (åpnet mar. 26, 2020).
[37] «Tech Note: Touch Button Mapping Best Practices | Oculus». https://developer.oculus.com/blog/tech-note-touch-button-mapping-best-practices/ (åpnet apr. 02, 2020).

[38] Oculus Developers, «Rendering». https://developer.oculus.com/design/bp-rendering/ (åpnet mar. 26, 2020).
[39] Jas, «Distance field fonts», *GitHub*. https://github.com/libgdx/libgdx (åpnet mar. 26, 2020).
[40] A. Clay *mfl.*, «How Prepared Are Medical and Nursing Students to Identify Common Hazards in the Intensive Care Unit?», *Ann. Am. Thorac. Soc.*, bd. 14, nr. 4, s. 543–549, 2017, doi: 10.1513/AnnalsATS.201610-773OC.
[41] Oculus, «Oculus Quest: Alt-i-ett VR-briller | Oculus». https://www.oculus.com/quest/?locale=nb_NO (åpnet mar. 30, 2020).
[42] S. Rogers, «Oculus Quest: The Best Standalone VR Headset», *Forbes*. https://www.forbes.com/sites/solrogers/2019/05/03/oculus-quest-the-best-standalone-vr-headset/ (åpnet mar. 30, 2020).
[43] Compello, «AR - Augmented Reality», *Compello*. /no/ordbok/ar-augmented-reality/ (åpnet mar. 30, 2020).
[44] Oculus, «Play Rift Content on Quest with Oculus Link, Available Now in Beta!» https://www.oculus.com/blog/play-rift-content-on-quest-with-oculus-link-available-now-in-beta/ (åpnet mai 13, 2020).
[45] P. Polsinelli, «Why is Unity so popular for videogame development? - Design a Game». https://designagame.eu/2013/12/unity-popular-videogame-development/ (åpnet mar. 30, 2020).
[46] Unity, «New plans for Unity releases: Introducing the TECH and Long-Term Support (LTS) streams - Unity Technologies Blog», apr. 09, 2018. https://blogs.unity3d.com/2018/04/09/new-plans-for-unity-releases-introducing-the-tech-and-long-term-support-lts-streams/ (åpnet mai 12, 2020).
[47] Autodesk, «Cloud Powered 3D CAD/CAM Software for Product Design | Fusion 360». https://www.autodesk.com/products/fusion-360/overview (åpnet mar. 30, 2020).
[48] Blender Foundation, «About Blender», *blender.org*. https://www.blender.org/about/ (åpnet mar. 30, 2020).
[49] HVL, «Informasjon om koronatiltak». https://hvl.no/student/utveksling/reiserad-og-sykdomsutbrudd/ (åpnet mar. 30, 2020).
[50] Visma, «En kort introduksjon til Scrum», *Visma Blogg*, mai 16, 2019. https://www.visma.no/blogg/en-kort-introduksjon-til-scrum/ (åpnet mar. 30, 2020).
[51] S. Lewis, «What is object-oriented programming (OOP)? - Definition from WhatIs.com», *SearchAppArchitecture*. https://searchapparchitecture.techtarget.com/definition/object-oriented-programming-OOP (åpnet mar. 30, 2020).
[52] Unity Technologies, «Unity - Manual: Unity User Manual (2018.4)». https://docs.unity3d.com/2018.4/Documentation/Manual/index.html (åpnet mar. 30, 2020).
[53] Arbeidstilsynet, «Risikovurdering». https://www.arbeidstilsynet.no/hms/risikovurdering/ (åpnet mar. 30, 2020).
[54] Unity Asset Store, «Oculus Integration | Integration | Unity Asset Store». https://assetstore.unity.com/packages/tools/integration/oculus-integration-82022 (åpnet mai 12, 2020).
[55] Oculus Developers, «Oculus Unity Integration: Experimental Cross-Platform Support | Oculus». https://developer.oculus.com/blog/oculus-unity-integration-experimental-cross-platform-support/ (åpnet mai 12, 2020).
[56] VRTK, «VRTK - Virtual Reality Toolkit». http://www.vrtk.io (åpnet mai 12, 2020).
[57] Oculus Developers, «Android Development». https://developer.oculus.com/documentation/unity/unity-mobile-performance-intro/ (åpnet mai 12, 2020).
[58] Unity, «Seven stages of optimizing mobile VR content in Unity», *Unity*. https://unity3d.com/how-to/optimize-mobile-VR-games (åpnet mai 12, 2020).
[59] Oculus Developers, «Testing and Performance Analysis».

https://developer.oculus.com/documentation/unity/unity-perf/ (åpnet mai 12, 2020).
[60] E. D. Da Costa, «Unity with MVC: How to Level Up Your Game Development», *Toptal Engineering Blog*. https://www.toptal.com/unity-unity3d/unity-with-mvc-how-to-level-up-your-game-development (åpnet mai 02, 2020).
[61] R. Nystrom, «Component · Decoupling Patterns», *Game Programming Patterns*. http://gameprogrammingpatterns.com/component.html (åpnet mai 02, 2020).
[62] Unity Technologies, «Unity - Scripting API: ScriptableObject». https://docs.unity3d.com/ScriptReference/ScriptableObject.html (åpnet mai 04, 2020).
[63] Unity Technologies, «Unity - Scripting API: MonoBehaviour». https://docs.unity3d.com/ScriptReference/MonoBehaviour.html (åpnet mai 04, 2020).

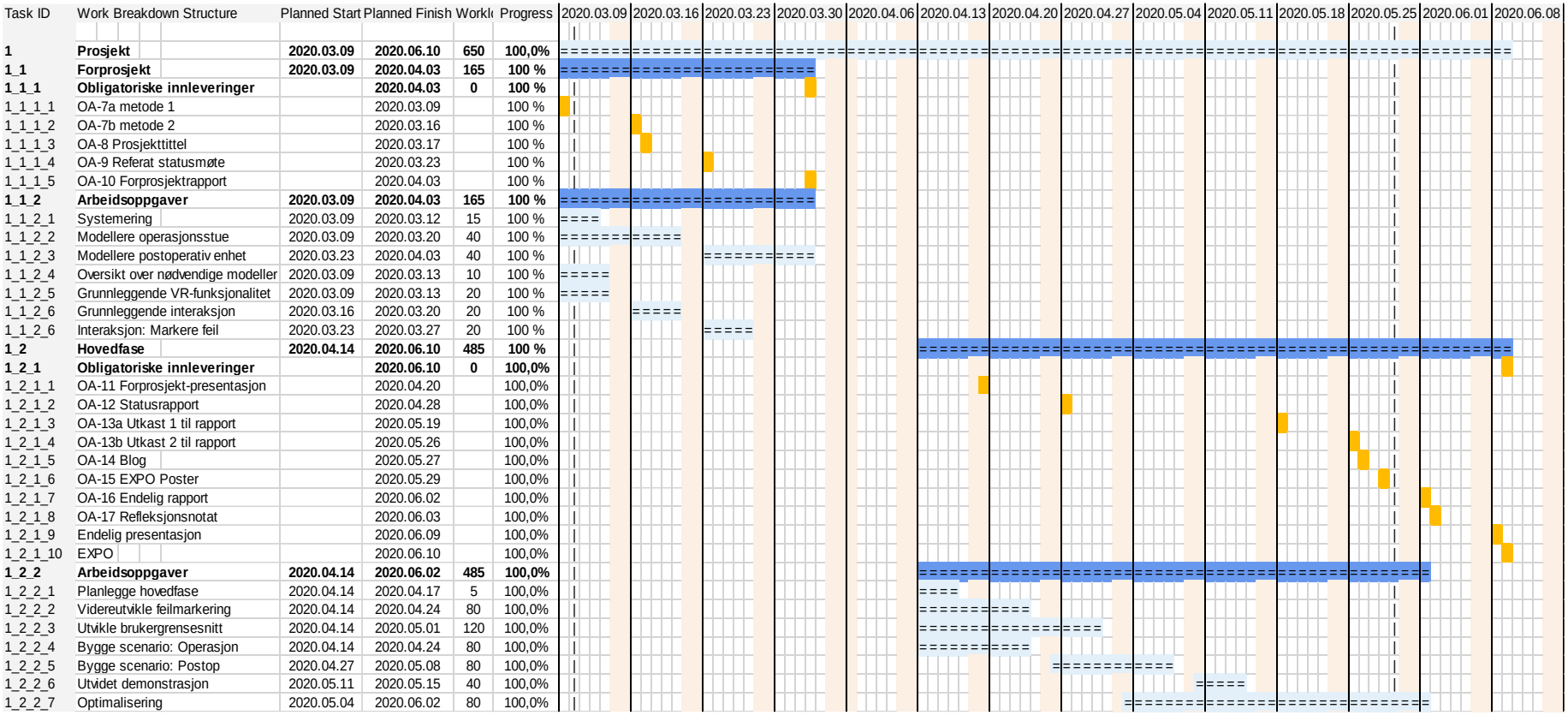
10 APPENDIX

10.1 Risikoliste

Nr.	Hva kan gå galt	Beskriv konsekvensen hvis det skjer	Alvorlighet	Kommentar	Vurdert
1	Mangel på testing	Da vil det bli vanskelig å evaluere prosjektet	Alvorlig	Det er utviklet måter å få testet uten VR	Dette er en reell risiko på grunn av Covid-19.
2	Mangel på kompetanse i gruppen	Da vil gruppen ikke kunne innfri ønsket resultat fra produkteier	Alvorlig	Gruppen har ressurser som er behjelpelig om kompetansen svikter.	Gruppen har brukt tid på å gjøre seg kjent med teknikker og verktøy som skal brukes i oppgaven.
3	Friksjon i utviklingsgruppen	Dette kan føre til mindre effektivt arbeid.	Mindre alvorlig	Gruppen har jobbet sammen over lengre tid med tidligere prosjekter uten problem.	Gruppen har vurdert dette, og har avtaler for at arbeid ikke skal senkes over personkonflikt.
4	Feil i tidsestimat / Uforusette tidsforsinkelser	Produktet kan bli forsinket slik at det ikke oppfyller produkteiers krav	alvorlig	Dette kan skje ettersom dette er gruppens første prosjekt med utvikling av spill	Vurdert og planen evalueres fortløpende med daglige møter i gruppen.
5	Misforstått ordre	Produktet som er utviklet kan være forskjellig fra produkteiers ønske	Svært alvorlig	Dette er bekjempet med ukentlig demo for produkteier samt møter.	Vurdert som ikke sannsynlig på grunn av hyppige møter.

Mal fra arbeidstilsynet [40]

10.2 Gantt Diagram



Task ID	Work Breakdown Structure	Planned Start	Planned Finish	Workl	Progress
1	**Prosjekt**	**2020.03.09**	**2020.06.10**	**650**	**100,0%**
1_1	**Forprosjekt**	**2020.03.09**	**2020.04.03**	**165**	**100 %**
1_1_1	**Obligatoriske innleveringer**		**2020.04.03**	**0**	**100 %**
1_1_1_1	OA-7a metode 1		2020.03.09		100 %
1_1_1_2	OA-7b metode 2		2020.03.16		100 %
1_1_1_3	OA-8 Prosjekttittel		2020.03.17		100 %
1_1_1_4	OA-9 Referat statusmøte		2020.03.23		100 %
1_1_1_5	OA-10 Forprosjektrapport		2020.04.03		100 %
1_1_2	**Arbeidsoppgaver**	**2020.03.09**	**2020.04.03**	**165**	**100 %**
1_1_2_1	Systemering	2020.03.09	2020.03.12	15	100 %
1_1_2_2	Modellere operasjonsstue	2020.03.09	2020.03.20	40	100 %
1_1_2_3	Modellere postoperativ enhet	2020.03.23	2020.04.03	40	100 %
1_1_2_4	Oversikt over nødvendige modeller	2020.03.09	2020.03.13	10	100 %
1_1_2_5	Grunnleggende VR-funksjonalitet	2020.03.09	2020.03.13	20	100 %
1_1_2_6	Grunnleggende interaksjon	2020.03.16	2020.03.20	20	100 %
1_1_2_6	Interaksjon: Markere feil	2020.03.23	2020.03.27	20	100 %
1_2	**Hovedfase**	**2020.04.14**	**2020.06.10**	**485**	**100 %**
1_2_1	**Obligatoriske innleveringer**		**2020.06.10**	**0**	**100,0%**
1_2_1_1	OA-11 Forprosjekt-presentasjon		2020.04.20		100,0%
1_2_1_2	OA-12 Statusrapport		2020.04.28		100,0%
1_2_1_3	OA-13a Utkast 1 til rapport		2020.05.19		100,0%
1_2_1_4	OA-13b Utkast 2 til rapport		2020.05.26		100,0%
1_2_1_5	OA-14 Blog		2020.05.27		100,0%
1_2_1_6	OA-15 EXPO Poster		2020.05.29		100,0%
1_2_1_7	OA-16 Endelig rapport		2020.06.02		100,0%
1_2_1_8	OA-17 Refleksjonsnotat		2020.06.03		100,0%
1_2_1_9	Endelig presentasjon		2020.06.09		100,0%
1_2_1_10	EXPO		2020.06.10		100,0%
1_2_2	**Arbeidsoppgaver**	**2020.04.14**	**2020.06.02**	**485**	**100,0%**
1_2_2_1	Planlegge hovedfase	2020.04.14	2020.04.17	5	100,0%
1_2_2_2	Videreutvikle feilmarkering	2020.04.14	2020.04.24	80	100,0%
1_2_2_3	Utvikle brukergrensesnitt	2020.04.14	2020.05.01	120	100,0%
1_2_2_4	Bygge scenario: Operasjon	2020.04.14	2020.04.24	80	100,0%
1_2_2_5	Bygge scenario: Postop	2020.04.27	2020.05.08	80	100,0%
1_2_2_6	Utvidet demonstrasjon	2020.05.11	2020.05.15	40	100,0%
1_2_2_7	Optimalisering	2020.05.04	2020.06.02	80	100,0%

10.3 Innkjøpte modeller

- **Hospital room**
  Av *studio lab*
  https://assetstore.unity.com/packages/3d/environments/industrial/hospital-ward-80735
  (åpnet mai 22, 2020)

- **Operating room**
  Av *Vertigo Games*
  https://assetstore.unity.com/packages/3d/props/interior/operating-room-18295 (åpnet mai 22, 2020)

- **Surgical Instruments - Medical Equipment Collection**
  Av *AssetKit*
  https://www.turbosquid.com/FullPreview/Index.cfm/ID/1072101 (åpnet mai 22, 2020)

- **Laryngoscope**
  Av *Glucius*
  https://www.turbosquid.com/FullPreview/Index.cfm/ID/763042 (åpnet mai 22, 2020)

- **3D Anaesthesia Medical Equipment model**
  Av *3dartmasterwork*
  https://www.turbosquid.com/FullPreview/Index.cfm/ID/1213055 (åpnet mai 22, 2020)

- **Anesthesia Face Mask**
  Av 3d_molier International
  https://www.turbosquid.com/FullPreview/Index.cfm/ID/907903 (åpnet mai 22, 2020)

- **3D Medical Equipment Collection 15 in 1 model**
  Av 3D_Shakh Dzmitry
  https://www.turbosquid.com/FullPreview/Index.cfm/ID/1157620 (åpnet mai 22, 2020)