

TowerVR

VR-spel för flera spelare med positionsbestämning
till Google Cardboard

Adam Alsegård
Jonathan Grangien
Joakim Konac
Love Lidberg
Måns Löglund
Victoria Waldemarson
Benjamin Wiberg

Examinator: Karljohan Lundin Palmerius

Sammanfattning

Denna rapport behandlar ett medietekniskt kandidatarbete inom *virtual reality* (VR). Syftet med arbetet var att utveckla ett VR-spel till Google Cardboard med flerspelarläge som använder positions- och rotationsbestämning. Det finns i nuläget (maj 2016) ingen annan applikation på marknaden som kombinerar VR, Google Cardboard, positionsbestämning samt flerspelarläge.

Spelet byggdes i spelmotorn Unity med tredjepartsbiblioteken Vuforia för positionsbestämningen samt Photon Unity Networking (PUN) för nätverkskommunikationen mellan spelarna. Utvecklingen följe systemutvecklingsprinciper, mer bestämt ett agilt arbetssätt enligt metoden *Scrum*. Projektet hade ett flertal sprintar som kan sammanfattas med en undersökningsfas för tekniska lösningar, en undersökningsfas för spelidéer samt en implementeringsfas.

Det slutgiltiga spelet går ut på att bygga ett gemensamt torn med klossar i olika former. Den spelare som lägger klossen som gör att tornet rasar åker ut. Klossarna har olika svårighetsgrader och ger därmed olika mycket poäng. Den spelare med mest poäng vid spelets slut vinner.

Projektarbetets resultat var i huvudsak det slutgiltiga spelet med underdelarna nätverk, prestanda, positionsbestämning och spelidé. Nätverket sköttes genom PUN som i grunden inte har en övre gräns för antalet spelare, men projektgruppen rekommenderar inte att fler än fyra spelare spelar samtidigt. Optimering av prestandan gjorde att den slutgiltiga bilduppdateringsfrekvensen blev 30-60 bilder per sekund under en spelomgång, beroende på telefonens hårdvara.

Då VR är ett område i ständig utveckling har projektgruppen fått vara med i att utforska nya tekniska områden. Samtidigt har många tekniska problem uppkommit under utvecklingsprocessen. Resultatet är således präglat av både nytänkande idéer och tekniska lösningar, men även omognad i tekniken.

Innehåll

Sammanfattning	i
Figurer	v
1 Inledning	1
1.1 Bakgrund	1
1.2 Syfte	2
1.3 Frågeställningar	2
1.4 Avgränsningar	2
1.5 Typografiska konventioner	3
2 Relaterat arbete	4
2.1 Outlive - ett AR-spel för flera spelare	4
2.2 Splash - sällskapsspel för mobila enheter	4
2.3 Hardcode - ett VR-spel för flera spelare	5
3 VR-principer	6
3.1 Cybersickness ochvection	6
3.2 Projektets åtgärder	7
4 Utvecklingsprocessen	8
4.1 Systemutvecklingsprinciper	8
4.1.1 Utvecklingsmetodik	8
4.1.2 Mötesrutiner	9
4.1.3 Kravhantering	9
4.1.4 Versionshantering	10
4.1.5 Kodkonventioner	11
4.1.6 QA, testningsprinciper och rutiner	11
4.1.7 Profilering	12
4.1.8 Dokumentationsprinciper	12
4.2 Tidsplanering/Sprintplanering	13

4.2.1	Undersökningsfas 1 - teknik	14
4.2.2	Undersökningsfas 2 - spelidé	14
4.2.3	Implementeringsfas	14
5	Teknisk beskrivning	15
5.1	Spårning	15
5.2	Nätverk	16
6	Implementation	17
6.1	Prototyper	17
6.2	Slutgiltiga spelet	18
6.2.1	Spelidé	18
6.2.2	Grafisk profil	19
6.2.3	Modellering	19
6.2.4	Menyer	19
6.2.5	Implementation av Photon Unity Networking	21
6.2.6	Spellogik	22
7	Resultat	23
7.1	Slutprodukt	23
7.2	Spelupplevelse	26
7.2.1	Användartester	26
7.3	Nätverk	26
7.4	Prestanda	26
8	Analys och diskussion	27
8.1	Arbetssätt	27
8.2	Analys av resultatet	28
8.3	Problem och hinder	29
8.3.1	Unity	30
8.3.2	Smarttelefoner utan gyroskop	30
8.3.3	Markörkuben	30
8.4	Arbetet i ett vidare sammanhang	31
9	Slutsatser	32
9.1	Utvecklingsområden	33
A	Gruppmedlemmar och ansvarsområden	36
B	Stories	37

C Resultat från användartester	39
D Misslyckat försök till utskriven kub	41
E GitHub - organisationen AndroidHMD	42
F Doxygen - klassindex	43

Figurer

1.1	Google Cardboard	1
4.1	Trello - överblick	9
4.2	Kodexempel för C#	11
4.3	Doxygen-dokumentation av en C#-klass	12
4.4	Sprintplanering	13
5.1	Markörkub	16
6.1	Flödeschema över scener i spelet	20
6.2	Exempel på en tredimensionell knapp från spelet	21
6.3	Klassdiagram	21
7.1	Byggklossar	23
7.2	Spelarnas avatarer - "wisps"	24
7.3	Byggklossar	24
7.4	Scen 1, huvudmeny, visat i stereo-läge. Här syns även ett slumpvist genererat namn för en spelare, vilket redogörs för i avsnitt 6.2.6	25
7.5	Scen 2, rumsinställningar	25
7.6	Scen 3, spelrumslobby	25
C.1	Användartester	39
C.2	Användartester	40
D.1	3D-printad kub som böjdes under utskrivningen. Den gick inte att använda till projektet.	41
E.1	GitHub - organisationen AndroidHMD	42
F.1	Doxygen - klassindex	43

Kapitel 1

Inledning

Den tekniska utvecklingen har ökat exponentiellt de senaste decennierna. Saker som visionärer bakom tidiga *science fiction*-böcker och filmer knappt kunde drömma om är nu vardagsredskap för gemene man. Ett område i ständig utveckling är *virtual reality* (VR), ett samlingsnamn för tekniker med avsikt att skapa en illusion för användaren av att vara i en annan verklighet. Denna rapport kommer behandla ett medietekniskt kandidatarbete inom VR.

1.1 Bakgrund

Under våren 2016 släppte VR-företaget Oculus sina VR-glasögon Rift på den internationella marknaden. Rift var redan innan det släpptes ett flaggskepp för VR då tidiga versioner av Rift fanns tillgängliga för utvecklare under ett flertal år. I takt med att intresset för VR har ökat så har dock fler produkter tagit sig ut på marknaden. Idag finns ett flertal varianter av VR-glasögon i olika prisklasser. Ett av de mer populära alternativen både för användare och utvecklare är Google Cardboard, vilket är en hållare för smarttelefoner anpassad för VR-applikationer. Google Cardboard består endast av ett par linser, en magnetbaserad knapp och ett hölje i pappkartong (se Figur 1.1), vilket gör den mycket billigare än majoriteten av sina konkurrenter.



Figur 1.1: Google Cardboard

Både Rift och smarttelefonvarianterna skapar den virtuella verkligheten genom att ta över flödet för ett eller flera sinnen, vanligtvis enbart synen. VR-glasögonen stänger ute den visuella verkligheten och visar istället en datorgenererad värld. När användaren vrider på huvudet spårar enheten huvudets rörelser och speglar dessa i den virtuella verkligheten. Det finns vissa installationer där även ljud, lukt och/eller balans påverkas.

En nära besläktad teknik till VR är *Augmented reality* (AR) eller förstärkt verklighet. AR kräver en enhet med videokamera då användaren, till skillnad från VR, är kvar i den verkliga världen och att virtuella objekt endast överlagras ovanpå kamerans videoflöde.

I de flesta VR-applikationer och spel som finns idag bestäms användarens position i världen inte av användarens verkliga, fysiska position. Antingen förflyttas användaren i en fördefinierad bana, åker med konstant hastighet i ögonens riktning eller har en statisk position. Det finns vissa applikationer till Rift där användaren kan styra sin position med hjälp av en spelkontroller. Det finns även exempel på AR-applikationer till mobila enheter där både rotation och position bestäms utefter ett spårat objekt. I dagsläget finns det dock ingen VR-applikation till mobila enheter som använder sig av både positions- och rotationsbestämning. För att få positionsbestämning till Google Cardboard måste tekniken för AR integreras med tekniken för VR.

1.2 Syfte

Målet med projektet är att utveckla ett VR-spel som använder både positions- och rotationsbestämning. Spelet ska gå att spelas av både en och flera spelare. Spelet ska utvecklas i spelmotorn Unity och målplattformen är Android-smarttelefoner med VR-glasögonen Google Cardboard. Google Cardboard SDK används för den grundläggande funktionaliteten för VR-applikationer och Vuforia används för positionsbestämning.

SDK står för *Software Development Kit* och är en uppsättning utvecklingsverktyg som gör det möjligt för mjukvaruvecklare att bygga applikationer mot specifik hårdvara, programpaket eller liknande.

1.3 Frågeställningar

De vetenskapliga frågeställningarna detta projekt avser svara på är:

- Vilka begränsningar finns för speldesign och interaktionsmöjligheter medför Google Cardboard som VR-plattform?
- Hur kan biblioteket Vuforia integreras i ett Unity-projekt för att få positions- och rotationsbestämning till en VR-applikation?
- Vilka nätverksalternativ finns för smarttelefoner och Unity, och vilket är bäst för realtidsinteraktion mellan spelare?
- Hur avancerad kan spelets grafiska stil vara (i form av 3D-modeller, visuella effekter osv.) samtidigt som en tillräckligt bra uppdateringsfrekvens bibehålls för att maximera *immersion* och undvika *cybersickness*?

Immersion syftar på känslan att bli helt innesluten i en virtuell verklighet. *Cybersickness* är åksjukekänslan som kan inträffa när sinnenas intryck i den virtuella världen inte stämmer överens med verkligheten. Detta diskuteras vidare i kapitel 3.

1.4 Avgränsningar

Gruppen kommer att arbeta i programmet Unity för att utveckla VR-spelet då Unity har en hel del inbyggda funktioner som hjälper till med VR-portering till smarttelefoner och för positionsbestämning.

Spelet kommer utvecklas för Android-smarttelefoner och endast utvecklas för iOS-smarttelefoner i mår av tid. Google Cardboard används som VR-glasögon och biblioteket Vuforia används för positionsbestämningen.

1.5 Typografiska konventioner

Engelska namn eller uttryck skrivs i kursiv stil. Alla dessa uttryck förklaras i löpande text efter att de nämns första gången. Programnamn skrivs med stor bokstav men ej i kursiv stil, även i de fall namnen är på engelska. Även förkortningar förklaras i löpande text efter att de nämns första gången. Fetstil används för att markera namn på klasser, filmer, spelidéer eller liknande.

Kapitel 2

Relaterat arbete

Google Cardboard släpptes i juni 2014 och är därmed fortfarande en relativt ny produkt. Populariteten verkar dock växa exponentiellt och det finns i nuläget väldigt många tillgängliga applikationer. Som det nämntes i föregående kapitel (1.1) finns det idag inte några VR-applikationer till Google Cardboard som använder sig av positionsbestämning [1]. Det finns dock ett flertal AR-applikationer som gör det, varav en handfull även har lägen för flera spelare. Applikationer till mobila enheter som använder sig av AR har varit på marknaden längre än de som tillämpar VR. Redan 2013 skrev Ibañez et. al en master-uppsats som analyserade ramverket Vuforia [2]. Vuforia används i detta projekt för positionsbestämning. Positionen beräknas utifrån en markör som mobilkameran spårar. En utförligare beskrivning av ramverket återfinns i avsnitt 5.1. Ibañez et. al rekommenderar att använda Vuforia tillsammans med Unity och sätter upp vilka begränsningar ramverket har med sig. Exempelvis bevisar de att Vuforia lättast kan bestämma positionen på ett avstånd på 0.5 till 3 meter från den spårade markören. Alla förslag som framförs i uppsatsen på användningsområden handlar dock bara om AR-applikationer. Den första frågeställningen i denna rapport behandlar hur Vuforia kan användas för en VR-applikation.

Det finns vissa spel som behandlar liknande områden som detta projektarbete. Några exempel följer nedan.

2.1 Outlive - ett AR-spel för flera spelare

Ett av de första exemplen på ett AR-spel för flera spelare är Outlive [3]. Spelet är byggt i Unity med Vuforia för positionsbestämningen. Alla spelare spelar dock på samma telefon, enheten lämnas helt enkelt över till nästa spelare när turen går vidare. Spelet använder sig av flera viktiga VR-aspekter för hur ljud och bild integreras men saknar stereoskopisk vy, rotationsbestämning och andra fundamentala VR-element.

2.2 Splash - sällskapsspel för mobila enheter

Splash är ett AR-spel för flera spelare som utvecklades i ett tidigare kandidatprojekt i kursen TNM094 på Linköpings Universitet. Till skillnad från Outlive använder sig Splash av en nätverkslösning och upp till fyra spelare kan spela samtidigt. Holst et. al [4] använde sig utav en *peer-to-peer*-lösning (P2P) för nätverket och ramverket Metaio för positionsbestämningen. P2P är ett nätverk av sammankopplade noder som inte kommunicerar enligt klient-server-modellen [5]. Detta innebär att man inte tilldelar olika enheter specifika roller i kommunikationen och att inga noder har några speciella

privilegier gentemot de övriga, utan att alla noder i nätverket kan agera i alla roller.

2.3 Hardcode - ett VR-spel för flera spelare

Hardcode är ett spel till Android med Google Cardboard som har ett läge för flera spelare. Flerspelarläget gör spelet relevant då detta är det enda VR-spelet med ett sådant till plattformen Android som projektgruppen har hittat i dagsläget, och det är därför intressant att jämföra implementeringen. Hardcode är ett skjutspel i tredjepersonsperspektiv där spelaren kontrollerar sin karaktär med en handkontroll; detta för att undvika *cybersickness* då spelet inte har positionsbestämning. Det här gör att det är diskuterbart om spelet ens är virtuell verklighet då viktiga principer saknas och den immersiva upplevelsen inte är lika stark som den hade varit om spelet implementerade ett förstapersonsperspektiv [6].

Kapitel 3

VR-principer

Vid utveckling av en VR-applikation finns det ett flertal principer som utvecklaren behöver ta hänsyn till. Detta kapitel förklarar dessa principer samt hur projektet anpassades för att efterfölja dem.

Målet med VR är att användaren ska uppleva sig att vara i en annan värld, vilket uppnås genom att ge användaren en fysisk representation i en virtuell miljö. Applikationen ska i största grad försöka få användarens fokus att ligga i den simulerade världen, detta för att maximera upplevelsen av *immersion* och därmed graden av "verklighet"^[7]. Eftersom människan ej är van vid att ha en virtuell verklighet runt sig medföljer dock en hel del problem, som *cybersickness* och *vection*. Nedan redogörs för hur dessa uppkommer och vilka teorier som finns för att reducera åkommorna.

3.1 Cybersickness och vection

Cybersickness, eller *simulation sickness*, är upplevelsen av åksjukeliknande obehag som uppstår vid användande av virtuella miljöer. Det finns idag flera teorier om orsaken bakom *cybersickness* varav *sensor conflict theory*, att kroppens olika sinnen och rörelsesensorer hamnar i konflikt, *postural control instability*, att nervsystemets kontroll av kroppens orientering och stabilitet rubbas, och *eye movement hypothesis optokinetic reflex*, att ögonen följer objekt som rör sig medan huvudet är stilla, är tre av dem. Det finns dock inga konkreta bevis för varför vissa personer upplever *cybersickness* och andra inte, eller huruvida konflikt mellan sinnen faktiskt orsakar obehag [8].

vection är känslan av rörelse under avsaknad av fysisk rörelse, rörelsen är alltså helt imaginär. LaViola påstår att "vection is required for cybersickness to occur"^[8]. Det påstås därmed finnas ett starkt samband mellan fenomenen. *vection* kan dock vara eftertraktat inom vissa områden i VR, såsom under militärträning och rehabilitering, eftersom det kan bidra till en mer realistisk och övertygande upplevelse [9]. Problem uppstår vid försök att maximera *vection* samtidigt som *cybersickness* skall undvikas, detta eftersom relationen mellan de två är nära sammankopplade. Andra faktorer som kan påverka upplevelsen av *cybersickness* är användarens kön, ålder, eventuella sjukdomar, hur lång tid användaren befinner sig i VR-världen samt hur bra positioneringen fungerar. Även tekniska problem som hack eller jitter i positionsbestämningen kan påverka upplevelsen [10].

För att undvika *vection* i VR kan utvecklaren undvika virtuell acceleration och istället ha en konstant hastighet i spelet. Kameran bör heller inte flyttas i oväntade riktningar utan styras av användaren. Med ökad speltid ökas risken för *cybersickness* och användarens tidsuppfattning kan rubbas. Om fokus ligger fel, om fokuseringen tar för lång tid eller om synfälts-vinkeln är onormal kan användaren även lätt få huvudvärk [10]. Problem kan även uppstå när användaren går in och ut från VR-världen. *Flashbacks*, korta tillbakablickar, och andra liknande fenomen kan dröja sig kvar efter användandet, vilket kan skapa jobbiga bieffekter [8]. Forskning visar dock att en större exponering till VR kan göra

att användaren får färre obehag med tiden [9].

3.2 Projektets åtgärder

En optimal VR-applikation bör reducera eller helt få bort *cybersickness*. Det aktuella projektet försöker åstadkomma detta genom att ha positionsbestämning runt ett bestämt origo. Rörelserna i VR-världen följer därmed användarens fysiska rörelser och så länge användaren inte flyttar sin position medan denne saknar spårning av markörkuben kommer orienteringen i VR-världen alltid stämma med sinnenas orientering. Den modellerade världen är även skalenlig och användaren kommer alltid ha ett golv eller en plattform att stå på. Avatarer för de andra spelarna kommer ritas ut för ökad orientering.

Det finns heller ingen extern kontroll av kameran utan kamerans alla rörelser följer huvudets egna. Spelet kommer ha korta spelomgångar och användaren kan när som helst lämna spelet ifall obehagen blir för stora. Den virtuella världen består av en statisk bakgrund samt enkla objekt och texturer för att undvika för låg bilduppdateringsfrekvens. Gruppen har även följt Googles riktlinjer för hur en VR-applikation bör byggas [11], framförallt gällande övergångar mellan scener och logiken kring navigation.

Kapitel 4

Utvecklingsprocessen

Innan arbetet med applikationen påbörjades satte projektgruppen upp en projektplan. Denna plan behandlade en tidsplan, men även principer, rutiner, krav och mallar som gruppen skulle följa. Projektplanen uppdaterades allt eftersom projektet fortskred och nya kunskaper inhämtades. Detta kapitel beskriver den slutgiltiga projektplanen med de mallar, krav och rutiner som projektet har följt under utvecklingen.

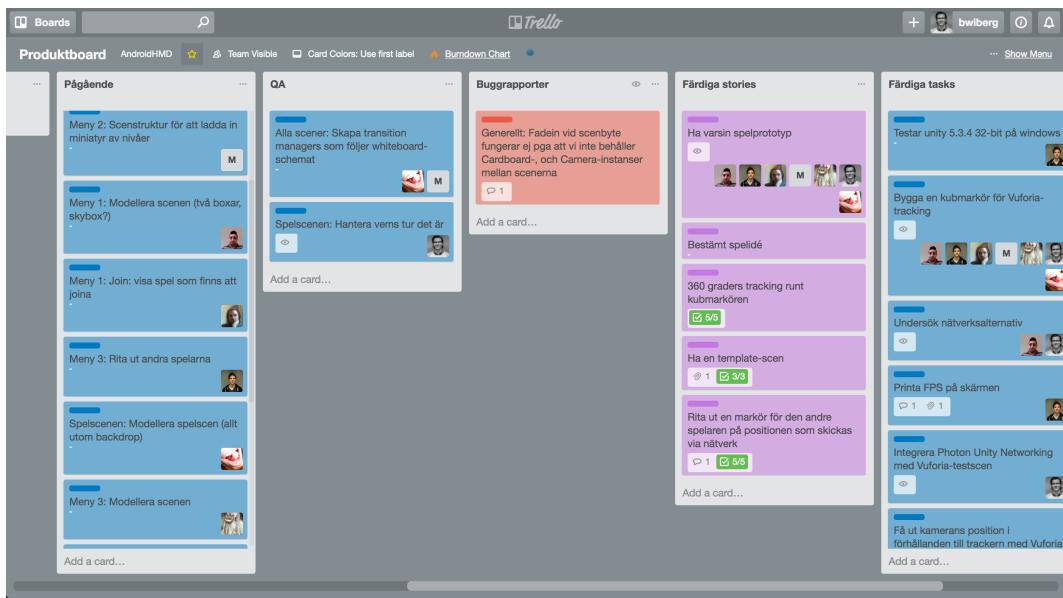
4.1 Systemutvecklingsprinciper

En projektplan bör beskriva hur projektgruppen ska arbeta med systemutvecklingen. För detta krävs ett antal principer och rutiner gällande kravhantering, dokumentering, versionhantering och liknande. Detta avsnitt beskriver dessa systemutvecklingsprinciper.

4.1.1 Utvecklingsmetodik

Projektet följe en agil utvecklingsmetodik, mer specifikt metoden *Scrum*. Alla uppgifter i projektet beskrivs genom berättelser, *stories*, där det specificerades vem i projektgruppen som skulle göra vad, när, var och varför. Alla berättelser sparades i en *product backlog*, produktlista. Själva arbetet utfördes under sprintar. Dessa beskrivs utförligare i avsnitt 4.2. Inför en sprint togs de berättelser som ligger överst i produktlistan och flyttades till en temporär *sprint backlog*, sprintlista. Sedan delades varje berättelse upp i flera *tasks*, uppgifter, som beskrev vad som behövde göras för att implementera berättelsen. Applikationen Trello användes för dokumentation av produkt- och sprintlistan samt kontinuerlig uppdatering av alla uppgifter under en sprint. Figur 4.1 visar en överblick av Trello under en av utvecklingssprinterna. En fullständig dokumentation över projektets berättelser och uppgifter finns i Bilaga B.

I Trello låg produktlistan som en egen flik (ej synlig i Figur 4.1). När en berättelse flyttades från produktlistan till sprintlistan fyllde projektgruppen gemensamt på en checklista med uppgifter på berättelsens kort. När dessa uppgifter sedan var avklarade var även berättelsen avklarad. Under sprinten kunde därefter en projektmedlem välja en uppgift, göra om det till ett uppgifts-kort, och lägga det under fliken "Pågående". När uppgiften sedan var klar flyttades kortet till fliken "QA" tills en annan projektmedlem hade testat koden (se avsnitt 4.1.6). Ifall koden blev godkänd flyttades kortet till "Färdiga tasks". När alla uppgifter på en berättelse var implementerade flyttades berättelsekortet till "Färdiga stories". Ifall en bugg stöttes på under sprintens gång dokumenterades den enligt 4.1.8 samt lades in som ett kort under fliken "Bugrapporter".



Figur 4.1: Trello - överblick

4.1.2 Mötesrutiner

Varje sprint inleddes med ett uppstartsmöte där sprinten planerades, berättelser fylldes på med uppgifter och ansvarsområden delades ut (se Bilaga A). Varje arbetspass inleddes därefter med ett sprintmöte där alla projektmedlemmar berättade status på sin uppgift och vad de hade tänkt arbeta med under dagen. Som avslutning på sprinten hölls ett reflektionsmöte där gruppen diskuterade hur det gått under sprinten, vilka berättelser som var avklarade och vilka som eventuellt behövde tas med till nästa sprint. Gruppen analyserade även projektplanen och uppdaterade den ifall det fanns behov. En utförlig beskrivning av projektets sprintar samt vilka berättelser som avklarades eller ej återfinns i Bilaga B.

Ett kundmöte hölls i början av projektet för att diskutera initiala krav och möjliga tekniska lösningar. Nästa kundmöte hölls efter att gruppen bestämt spelidé och denna diskuterades då med kunden. Kunden gav även förslag på fler potentiella framtida utvecklingsområden. Avstämningar däremellan har endast skett via mejl.

Utvecklingsarbetet skedde mest individuellt men gruppen satt ändå gemensamt och arbetade. Detta dels eftersom det endast fanns en fysisk markör som gruppen använde för spårning men även för att kunna ställa frågor till varandra och diskutera problem under gemensamma pauser. I perioder tillämpades även parprogrammering för att lösa vissa specifika uppgifter.

4.1.3 Kravhantering

Krav på produkten fanns för att driva projektet framåt. En viktig aspekt var att dessa krav skulle vara mätbara. Nya interna krav sattes upp innan varje sprint. Testning av kraven dokumenterades i tjänsten Google Drive och i Trello för kontinuerlig uppdatering för övriga i projektgruppen samt för kund. Kraven specificerades dels utifrån samtal med kunden samt dels utefter övrig dokumentation inom VR-området.

Krav från kund

- Spelet ska ha stöd för minst två spelare
- Varje spelare ska kunna se sina motspelare
- Spelet ska implementeras till Google Cardboard
- Spelarens verkliga position och rotation ska spåras och reflekteras i den virtuella världen

Krav från projektgruppen

- Spelet ska gå att spelas av både en och flera spelare
- Spelet ska ej vara bundet till stationär hårdvara
- Uppdateringsfrekvens på minst 50 bilder per sekund (FPS) eller att max 25% av testanvändare svarade ja på frågan ”Tycker du att spelet känns för segt eller efterdröjande när du tittar runt, för att det ska vara spelbart och underhållande?”

4.1.4 Versionshantering

Versionshanteringsverktyget Git användes med GitHub som central plats för *repositories*. Ett *repository* är ett arbetsarkiv för ett projekt som versionshanteras med Git. Alternativ till GitHub är BitBucket, GitLab eller en privat Git-server, men då projektmedlemmarna använt GitHub tidigare föll valet på GitHub. En organisation skapades på GitHub med alla projektmedlemmar som bidragande medlemmar. Organisationen användes som en samlingsplats där medlemmarna kunde skapa och hantera olika *repositories*, se Bilaga E.1.

Spelets *repository* hade en mängd *branches*, grenar. I detta stycke innebär *kursiv stil* namnet på en *branch*.

- *master*

Denna *branch* gick alltid att köra och var så stabil som möjligt. När spelet demonstrerades var det denna *branch* som användes. Grenen användes även för att sammanfoga ändringar och ny funktionalitet som implementerats på mer specifika *branches*.

- *feature-branch*

Ny funktionalitet, refaktorering av moduler samt eventuella enhetstester implementerades på specifika *branches*. En sådan *branch* földe ett specifikt arbetsflöde:

1. En ny *branch* skapades med *master* som utgångspunkt. Exempel: *dual-wield-weapons*.
2. Funktionaliteten implementerades enligt planerna på *dual-wield-weapons*. Ifall *master* hade utvecklats markant under tiden kunde utvecklarna på *dual-wield-weapons* göra en *merge* från *master* in i *dual-wield-weapons* för att redan i det stadiet se till så att de nya funktionerna fungerade tillsammans. När utvecklarna var klara med förändringen skapades en *pull request* från *master* att integrera in de nya funktionerna från *dual-wield-weapons*.
3. Ett team (som kunde vara samma utvecklare) granskade nämnda *pull request* genom att se över kodändringarna samt testa så att denna *branch* samt *master* fungerade tillsammans. Om funktionaliteten fungerade enligt utsago avslutades denna *pull request* med en *merge* och *dual-wield-weapons* stängdes. Viktigt var att *branches* som var färdiga aldrig återanvändes utan att nya unika *feature-branches* skapades.

4.1.5 Kodkonventioner

Spelets skriptspråk var C#. Även om Unity har stöd för skript med Javascript-likt syntax användes endast C#, med filändelsen ”.cs”.

Projektet följe Microsofts kodkonventioner för C# [12] till en mån som ansågs tillräcklig. Detta inkluderade konventioner för namnsättning, layout och kommentarer. Variabler deklarerades implicit (dvs ”var <namn> = ...;”) om det var uppenbart vid initialisering vilken datatyp det var, annars – om variabeln initialiseras exempelvis genom ett anrop till en otydlig funktion – deklarerades variablen explicit då det inte var direkt uppenbart vilken datatyp det handlade om. En ytterligare stilregel var att ”squiggly brackets” (dvs ”{” och ”}”) skrevs på ny rad för att synliggöra kodens omfattning.

Vidare definierades en mängd andra konventioner för gränssnitt, *interfaces*, samt abstrakta- och implementerande klasser:

- Vid beroenden mellan moduler var beroendet alltid dolt bakom en abstraktion; antingen via ett *interface* eller en abstrakt basklass.
- Ett *interface* hade ett ”I” som prefix. Exempel: ”IActor” eller ”IDestructible”.
- Abstrakta basklasser hade ej något prefix eller suffix utan beskrev klassen i sin enklaste form. Exempel: ”List” eller ”Weapon”.
- Implementerande klasser hade ett beskrivande prefix alternativt ett helt specifikt namn om namnet entydigt antydde att den var av samma typ som basklassen. Exempel: ”LinkedList” eller ”ArrayList” alternativt ”Sword” eller ”Fist”.

Ett exempel på projektets C#-kod syns i Figur 4.2.

```
/***
 * Randomly generates a player name from the prefix, main name and suffix arrays.
 */
public static string GenerateName()
{
    string prefix = GetRandomInArray(prefixes);
    string main = " " + GetRandomInArray(mainNames);

    // generate a suffix 50% of the time
    var suffix = "";
    if (Random.Range(0, 2) == 1)
    {
        suffix = GetRandomInArray(suffixes);
    }

    return prefix + main + suffix;
}
```

Figur 4.2: Kodexempel för C#

4.1.6 QA, testningsprinciper och rutiner

Quality assurance (QA) handlade om att säkerhetsställa att projektet höll en hög kvalitetsnivå genom hela utvecklingen. En hög kvalité för detta projekt var exempelvis att positionsbestämningen fungerade kontinuerligt, att bilduppdateringsfrekvensen låg på den önskade nivån och att spelet skulle kunna vidareutvecklas med fler nivåer och banor. Detta uppnåddes genom att spelet utvecklades med en objektorienterad struktur där de olika delarna smidigt utvecklades och testades separat.

Tester genomfördes när en ny komponent hade implementerats samt på hela systemet efter varje sprint. Gruppen genomförde interna tester för att hitta buggar och läckor, men anordnade även tester för användare utanför projektgruppen för att få återkoppling kring spelupplevelsen. Användartesterna beskrivs mer i avsnitt 7.2.1.

4.1.7 Profilering

För att hitta flaskhalsar i applikationens *pipeline*, den tekniska körgången från början till slut, användes Unity Profiler som är ett inbyggt profilingsverktyg i Unity. Verktyget användes under körning i Unitys redigerare och sparade information om exempelvis hur lång tid det tog för spelscenen att renderas, information om hur effektiv spellogiken var etc. Sedan ritades informationen ut längs en tidslinje så att användaren enkelt kunde se var eventuella prestationsslukande processer befann sig. Detta hjälpte projektgruppen att hitta de delar som behövde optimeras.

4.1.8 Dokumentationsprinciper

Det ansågs att odokumenterad kod inte var färdigskriven, vilket innebar att all egenskriven kod skulle förklaras. Projektet använde dokumentationsverktyget Doxygen för att generera överskådlig dokumentation över klasser och funktioner samt variabler och konstanter. Doxygen-dokumentation genererades utifrån källkoden genom att programmet analyserade kommentarerna i koden. Kommentarerna följer Doxygens syntax [13]. Ett exempel på Doxygen-dokumentation av en klass kan ses i Figur 4.3. För en översikt över alla klasser, se Bilaga F.1.

TowerVR.TowerGameManagerImpl Class Reference

Inheritance diagram for TowerVR.TowerGameManagerImpl:

```

graph TD
    MB[MonoBehaviour] --> TGM[TowerVR.ITowerGameManager]
    TGM --> TGMImpl[TowerVR.TowerGameManagerImpl]
    TGMImpl --> MTGM[TowerVR.MasterTowerGameManagerImpl]
  
```

Public Member Functions

- virtual void `notifyIsReady ()`
- virtual void `tryStartGame ()`
- virtual void `selectTowerPiece (TowerPieceDifficulty difficulty)`
- virtual void `placeTowerPiece (float positionX, float positionZ, float rotationDegreesY)`

Public Attributes

- TowerGameManager parent

Protected Member Functions

- virtual void `Awake ()`
- virtual void `OnDestroy ()`
- virtual void `onEvent (byte eventCode, object content, int senderID)`
- virtual void `handleGameStateChangedEvent (int gameState)`
- virtual void `handleTurnStateChangedEvent (int turnState)`
- virtual void `handleTowerStateChangedEvent (int towerState)`
- virtual void `handleNextPlayerEvent (int nextPlayerID)`
- virtual void `handleScoreChangedEvent (int playerID, Score score)`
- virtual void `handlePlayerLostEvent (int playerID)`
- virtual void `handlePlayerWonEvent (int playerID)`

Static Protected Member Functions

- static void `LogMalformedEventContent (string eventName, int senderID)`

Detailed Description

An `ITowerGameManager` that represents a remote instance.

Do not use this class directly in the game logic.

Member Function Documentation

`virtual void TowerVR.TowerGameManagerImpl.Awake ()` protected virtual

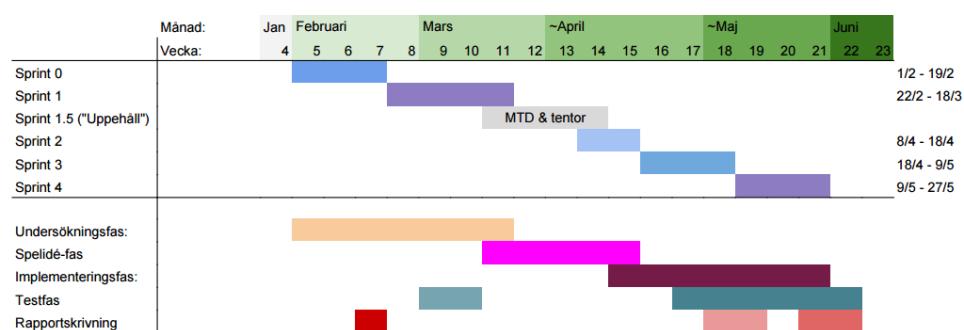
Reimplemented in `TowerVR.MasterTowerGameManagerImpl`.

Figur 4.3: Doxygen-dokumentation av en C#-klass

Dokumentation över använda algoritmer, designval i utvecklingsprocessen samt klassdokumentation, som inte automatiskt inkluderades vid körning av Doxygen, skrevs i separata filer. Projektgruppen skrev även övrig dokumentation om projektets utförande såsom mötesprotokoll och diverse information relaterad till utvecklingsmoment. Detta lagrades och organiserades, tillsammans med olika filer för utveckling, i en gemensam mapp genom tjänsten Google Drive. Även bugrapporteringar dokumenterades på Drive. Applikationen Slack användes för all kommunikation mellan gruppmedlemmar rörande projektet. Google Drive, Trello och GitHub integrerades med Slack för att förmedla notiser om uppdateringar och ändringar, vilket bidrog till att Slack blev ett heltäckande kommunikations- och planeringsverktyg.

4.2 Tidsplanering/Sprintplanering

Eftersom projektmedlemmarna inte hade särskilt mycket tidigare erfarenhet av spelutveckling eller VR bestämdes det att en väsentlig del av projektet behövde bestå av undersökningfaser. Det behövde undersökas vilka resurser som krävdes för att kunna implementera en teknisk lösning av kraven samt vilka typer av spelidéer som skulle passa inom VR-principerna (se kapitel 3). Figur 4.4 visar hur den slutgiltiga sprintplaneringen såg ut. Enligt Pfleeger & Atlee är det viktigaste inom agil utveckling tankesättet att projektets mål ändras över tid och därför måste projekthanteringen vara dynamisk [5]. Detta har gjort att gruppen kontinuerligt uppdaterat tidsplaneringen och dess sprinter under projektets gång.



Figur 4.4: Sprintplanering

Projektet hade fyra fullständiga sprints, plus två stycken som kan ses som ”halva” sprints. Alla sprints hade berättelser och uppgifter som sattes upp i början av sprinten, se avsnitt 4.1.1. Alla sprints med respektive berättelser är dokumenterade i Bilaga B. Sprintplaneringen kan sammanfattas med en undersökningsfas för tekniska lösningar, en undersökningsfas för spelidéer samt en implementeringsfas.

4.2.1 Undersökningsfas 1 - teknik

Arbetet inleddes med en första undersökningsfas för de tekniska lösningarna, framförallt gällande nätverk och positionsbestämning. Då projektgruppen inte hade tidigare erfarenheter av spelutveckling bestämdes det att spelprogrammering på hög nivå med en spelmotor var att föredra. För positionsbestämning behövdes en teknisk lösning, och där ansågs ett färdigt bibliotek vara att föredra då positionsbestämning vore för komplext att arbeta med på låg nivå. Projektgruppen ägnade Sprint 0 till att undersöka olika alternativ, däribland OpenCV, det därpå baserade ramverket Aruco samt Vuforia. Det hölls även ett kundmöte i början av fasen där det resonerades kring tekniker och idéer. I slutändan bestämdes det att Vuforia skulle användas då biblioteket var mer väl dokumenterat och har flera inbyggda komponenter i Unity. Vuforia redogörs för i avsnitt 5.1. Som nätverkslösning valdes Photon Unity Networking, som även det har flera inbyggda funktioner i Unity. En mer utförlig beskrivning av nätverkslösningen finns i avsnitt 5.2. Unity valdes som spelmotor då applikationen dels har en hel del inbyggda funktioner som hjälper till med portning till smarttelefoner men också för att det finns inbyggt stöd för Google Cardboard SDK.

Efter att de tekniska lösningarna hade spikats inleddes Sprint 1. Under den här perioden testades den tekniska integrationen av lösningarna. Mycket arbete gick ut på att anpassa Vuforia för en VR-applikation. Detta för att Vuforia i grund och botten är byggt för AR-applikationer [2]. Bland annat behövde kamerans bildrendering stängas av samt lösningar för vad som händer när spårningen av markören tappas undersökas och implementeras.

Olika versioner av spelmotorn gav olika resultat. Flera tester med olika versioner av Unity och biblioteken genomfördes för att optimera prestandan. Här användes profilering för att hitta flaskhalsar. För att få 360-graders positionsbestämning runt ett bestämt origo tog projektgruppen fram en kub med spårbara bilder på alla sidor. Ett test där ett objekt som representerar den andre spelaren ritas ut på korrekt position, skickad via nätverket, genomfördes. En körbart exempel scen där alla integrationer fungerade togs fram med avsikt att fungera som mall för utvecklingen av mindre prototyper samt det slutgiltiga spelet.

4.2.2 Undersökningsfas 2 - spelidé

Fasen inleddes med en tidsperiod där projektgruppen hölls upptagen utav många åtaganden ej relaterade till projektet. Under denna tid ansattes flexibelt individuellt arbete där olika spelidéer och spelprototyper togs fram. Projektgruppen kallade perioden för Sprint 1.5. Arbetet med spelprototyper fortsatte även under Sprint 2. En fullständig redogörelse av dessa prototyper återfinns i avsnitt 6.1.

Under ett internt möte vid slutet av perioden beslutade projektgruppen om vilken spelidé som skulle vidareutvecklas. Spelidén beskrivs utförligare i avsnitt 6.2.1. Ett kundmöte där spelidén diskuterades avslutade sprinten.

4.2.3 Implementeringsfas

Med tekniska lösningar för nätverk och positionsbestämning färdigställda, samt en bestämd spelidé, kan resterande sprintar sammanfattas som en implementeringsfas. I Sprint 3 sattes ett nytt *repository* upp för spelet och projektets slutgiltiga *sprint backlog* fylldes med *stories* och *tasks* specifika för spelidén. Arbetet delades sedan upp i mindre moduler för att gruppmedlemmarna skulle kunna arbeta parallellt. Exempel på moduler var modellering av scener, menyhantering, nätverk och spellogik. Under Sprint 3 avsattes även tid för rapportskrivning. Under Sprint 4 sammanfogades de olika delarna, vilket ledde till en del temporära buggar, och implementeringen slutfördes. Tid avsattes även för felsökning, opponering, användartester samt rapportskrivning.

Kapitel 5

Teknisk beskrivning

Projektarbetet har många olika tekniska komponenter. Dessa komponenter förklaras mer ingående i detta kapitel. Som det nämntes i avgränsningar samt i kravspecifikationen (se avsnitt 1.4 & 4.1.3) är målplattformen Android-smarttelefoner med Google Cardboard som VR-glasögon. Ett av tredjepartsbiblioteken är därför Google Cardboard SDK för att få Cardboard-funktionaliteten att fungera på smarttelefoner.

Det finns vissa systembegränsningar för att applikationen skall fungera på en viss enhet. Skärmen skall enligt riktlinjerna för Google Cardboard ligga mellan 4 - 5,5 tum i storlek för optimal VR-upplevelse. I dessa storlekar är upplösningen på smarttelefonerna betydligt mindre än i dedicerade VR-plattformar som exempelvis Rift [14] [15]. Grafiken i applikationer till Cardboard kan därmed inte återges i samma upplösning som hos applikationer till andra VR-plattformar.

Androidversionen hos användaren bör vara 4.1 (*Jelly bean*) eller högre. Telefoner måste även ha ett fungerande gyroskop för att kunna läsa av den lokala rotationen.

5.1 Spårning

För positionsbestämning används tredjepartsbiblioteket Vuforia, som har ett tillägg till Unity. Vuforia använder mobilenhetens kamera för att lokalisera och spåra intressepunkter i bilder. Med hjälp av Vuforias API kan användare smidigt designa en markörkub med unika bilder/mönster på varje sida. De olika sidorna används för att Vuforia ska kunna avgöra var användaren befinner sig 360-grader relativt kuben. Detta innebär att kuben agerar som en referenspunkt för spelarna i spelets tredimensionella värld, och spårningen med Vuforia anger kontinuerligt spelarnas position. Figur 5.1 visar markörkuben som används i utvecklingen av detta projekt.

Markörkubens exakta mått behövs för att Vuforias API ska kunna generera en korrekt representation av kuben, som sedan importeras i Unity för vidare utveckling. Om fel dimensioner matas in i Vuforias API så finns det risk att kameran inte lyckas spåra intressepunkter korrekt relativt varandra.

Rotationsbestämningen beräknas av den inbyggda funktionaliteten i Google Cardboard SDK. Den använder telefonens inbyggda gyroskop för att beräkna den lokala rotationen och tillämpa den på kameran i den virtuella världen. Denna rotation tillämpas på den virtuella kameran enbart om Vuforia inte har lyckats spåra markörkuben. När positionsbestämningen förloras fortsätter rotationen fortfarande att spåras. Cardboard SDK möjliggör även funktionen att kunna slå av och på stereoskopisk vy under körning.



Figur 5.1: Markörkub

5.2 Nätverk

I den inledande fasen undersöktes olika nätverksalternativ, bland annat *Google Play Games for Unity*, *peer-to-peer*, *NugServer* och Unitys egna API *NetworkManager*. Den lösning som gav bäst och enklast resultat tillsammans med Unity var dock Photon Unity Networking (PUN). Därför valdes PUN som slutgiltig nätverkslösning till spelet.

PUN har flera inbyggda funktioner till Unity och är gratis att använda för Unity 5.0 och högre. Position och rotation hos specificerade objekt synkroniseras automatiskt mellan anslutna klienter via PUN. Utöver det tillhandahåller PUN servrar för att starta spelrum och synkronisera scener i Unity. Genom ett enkelt API kan klienter skapa och ansluta sig till spelrum. Det går även att namnsätta sina rum så de lätt går att hitta samt sätta maxantal för hur många spelare som får vara med. Vidare kräver PUN ej att klienterna är i samma nätverk då all kommunikation går via deras servrar.

Kapitel 6

Implementation

När de tekniska lösningarna hade bestämts och testats så var det dags att börja själva implementeringen. Under undersökningsfas 2 implementerades olika spelprototyper och under implementationsfasen implementerades det slutgiltiga spelet.

6.1 Prototyper

Som tidigare nämnts i avsnitt 4.2.2 arbetade gruppmedlemmarna under en tid individuellt med att ta fram spelprototyper. Syftet med att utveckla prototyper var att utforska och utvärdera olika spelidéer för att skapa underlag till ett grundat beslut om en slutgiltig idé, men även att alla gruppmedlemmar skulle få erfarenhet av att utveckla i Unity. Prototyp innehåller i det här sammanhanget antingen ett fungerande litet spel eller en genomblikkt idé med skisser. De olika prototyperna som utvecklades var:

- **Arga björnar**

Spelaren ska försvara sig mot björnar som kryper upp ur ett hål i marken och rör sig mot spelarna. Spelarna kan få björnarna att försvinna genom att titta på dem och sända ut en laserstråle i synens riktning. Problematiken med prototypen var att då björnarna rör sig mot spelarna riktas spelets fokus bort från markörkuben och problematik kring positionsbestämning förvärras, så som att spelarnas positioner inte uppdateras. Det fanns inte heller någon särskild tanke kring flerspelarläget och hur det skulle vara intressant att tävla eller samarbeta.

- **Bowlingspel**

Ett spel för flera spelare där uppgiften är att träffa de andra spelarna med bowlingklot. Kloten sänds ut i synens riktning genom att trycka på knappen som sitter på sidan av Cardboard. I spelvärlden finns också hinder som spelarna kan gömma sig bakom för att undvika att bli träffade av bowlingkloten. Problematiken med spelet var, likt tidigare, att spelarens fokus ofta drogs bort från markörkuben och på så vis orsakade instabilitet i positionsbestämningen. Med en annan typ av positionsbestämning (exempelvis *Valve's Lighthouse Tracking Technology* [16]) hade denna spelidé kunnat bli en unik VR-upplevelse.

- **Skjuta på klossar**

Spelet går ut på att skjuta ner ett torn av klossar som är placerat på en plattform i spelvärldens centrum. Olika typer av klossar ger olika mycket poäng. Skjutmekanismen fungerar på samma sätt som i bowlingspelet. Problematiken med spelet var att det var svårt att utvidga till flera spelare utan att ändra spelidén för mycket.

- **Backpacker/Frågesport**

Spelaren startar spelet genom att välja en av flera möjliga destinationer (länder) som spelaren

sedan förflyttas till. Uppgiften är sedan att besvara frågor om den valda destinationen och frågorna ger olika mycket poäng. Problematiken med spelet var att det var svårt att hålla spelarens fokus kring markörkuben vilket resulterade i problem med spårningen.

- **Bygga torn**

Ett spel för flera spelare där spelarna står på en plattform som omsluter spelvärldens centrum. Spelarna ska turas om att placera olika typer av objekt i mitten av banan för att gemensamt bygga ett torn. Om tornet rasar åker spelaren som placerade byggklossen som orsakade raset ut. Denna spelidé visade sig ha i princip allt som de uppsatta kraven efterfrågade såsom: alternativ för flera spelare, fokus kring markörkuben för att upprätthålla en stabil positionsbestämmelse och en spelidé som ger en bra VR-upplevelse. Detta var anledningen till att spelidén blev segraren bland prototyperna och även den idé gruppen valde att satsa på i det fortsatta arbetet.

- **Labyrint**

Flera spelare har ett bräde var med en labyrint på. Spelarnas uppgift är att leda en boll till labyrintens utgång. När detta utförts får spelaren ut bollen ur labyrinten och kan då avfyra bollen på sina motspelare. En spelare som blir träffad av en boll åker direkt ut ur spelet.

- **Jenga**

En spelidé som fungerar ungefär som det klassiska sällskapsspelet Jenga. Flera spelare ska turas om att ta bort klossar från ett torn utan att tornet rasar. Den spelare som får tornet att rasa åker ut. Denna idé bidrog med inspiration till den slutgiltiga spelidén (Bygga torn).

6.2 Slutgiltiga spelet

Efter att alla gruppmedlemmar prövat på att utveckla en spelprototyp började implementationen av det slutgiltiga spelet. Denna process beskrivs i detta avsnitt.

6.2.1 Spelidé

Den slutgiltiga spelidén bygger mest vidare på Bygga torn-prototypen, vilket innebär att spelet går ut på att bygga ett så högt torn som möjligt av olika typer av klossar. Om tornet raseras när spelaren lägger en kloss åker sagda spelare ut. Tornet som byggs i 3D-världen är placerad i origo av ytan som spelarna befinner sig på, och navigering runt tornet fungerar så att spelarna går runt mittpunkten för att emulera att spelaren rör sig i en virtuell verklighet. Positionsbestämningen av varje spelare sker med hjälp av markörkuben som är placerad i mitten av rummet.

Spelet går i omgångar, där en spelare i taget placerar en kloss på tornet. Första fasen under en tur är valfasen. Spelaren får då se tre alternativ av klossar i svårighetsgraderna lätt, mellan och svår. Under valfasen visas klossarna endast lokalt på klienten. När spelaren valt en kloss påbörjas placeringsfasen och då instansieras klossen även hos övriga klienter.

Under placeringsfasen ser alla spelarna en projicering av klossen på den plats där aktuelle spelare tittar för stunden. Det är även på den platsen klossen placeras när spelaren trycker på Cardboards magnetknapp. En tidsbegränsning ser till att spelare inte kan vänta för länge på att placera sin kloss. Om en spelare inte har placerat en kloss när tiden tar slut så placeras blocket automatiskt på spelarens aktuella fokus på tornet. En liknande tidsbegränsning finns under valfasen. Då väljs den lättaste biten automatiskt ifall spelaren inte hunnit välja innan tiden tar slut.

En ytterligare stor inspiration för spelidén kommer från brädspelet **Bandu**, som går ut på att spelare ska lyckas stapla olika typer av objekt på höjden. När en spelares hög med objekt raseras så åker

denne ut ur spelet. Exempel på idéer som kom från **Bandu** var att inkludera mer udda objekt, som spikar, stjärnor och halvsärer, samt att spelaren skall kunna välja mellan olika svårighetsgrader på objekt och därmed få olika mycket poäng.

6.2.2 Grafisk profil

Spelet följer en futuristisk design med fokus på ett begränsat antal färger i menyerna. Dessa färger är främst mörkgrå, blå och vit. Projektgruppen har tagit inspiration av diverse *science fiction*-filmer och spel, såsom filmen **Tron** och spelet **Mirror's Edge**, som även dessa följer ett futuristiskt tema. Färgerna har även valts ut med hänsyn till färgblindhet, vilket inneär att en kombination av färgerna i menyerna samt spelscenerna inte resulterar i konflikter för användare med färgblindhet. Den vanligaste typen av färgblindhet är röd-grön färgblindhet vilket innebär att personen inte kan eller har svårt att se skillnad mellan rött och grönt. Därför är det främst kombinationen av rött och grönt som gruppen unvikt att använda.

Spelets olika scener är designade på det sättet att ge spelaren en känsla av en oändlig värld, vilket innebär att spelaren aldrig är innesluten i ett rum i VR-världen. Detta sker genom att spelaren står på en plattform i spelvärlden som omsluts av en förbestämd bakgrundsbild som ger en illusion av att vara placerad långt borta i alla riktningar.

Spelare kan välja vilket tema spelscenen kommer att ha innan spelomgången sätts igång, vilket är ett sätt att minska risken att spelet känns repetitivt. Dessa olika teman har unika färgscheman i spelscenen samt ändrar bakgrundsbilden som omsluter spelvärlden.

6.2.3 Modellering

De modeller som kan ses i spelet har modellerats i Blender och 3ds Max för att sedan exporteras till Unity. För att inte tappa prestanda under spelandet av slutprodukten har antalet polygoner hela tiden tagits hänsyn till då alltför detaljrika modeller tenderar att göra att spelet får en förminskad bilduppdateringsfrekvens. De texturer som används har antingen skapats i Photoshop CC 2015, hämtats från nätet med fria användarrättigheter eller tagits från Unity's Asset Store. För att addera mer detaljer utan att använda fler polygoner har en normalmap sedan genererats till varje textur via Photoshop CC 2015.

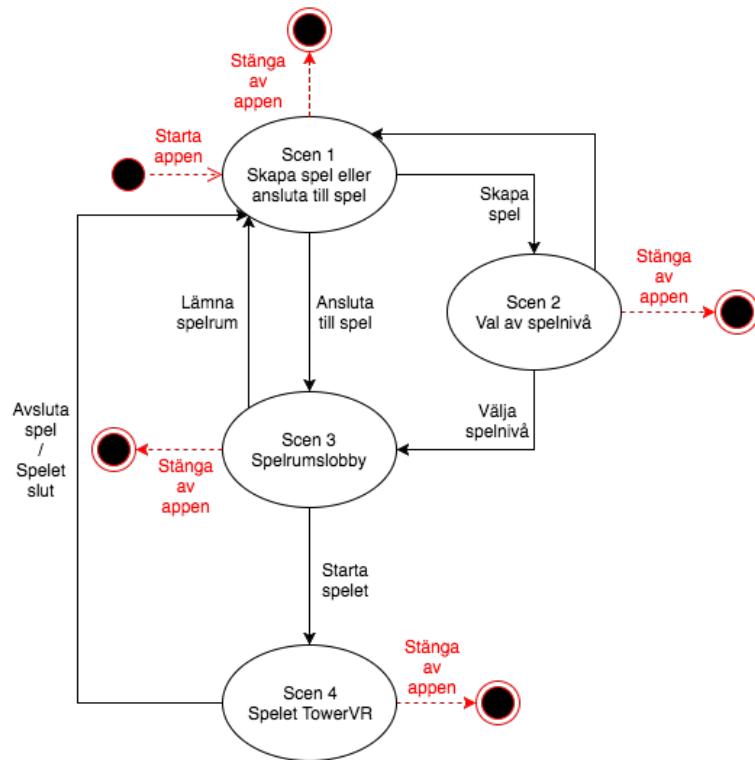
6.2.4 Menyer

Innan spelarna kommer till spelscenen så måste tre menyscener gås igenom. I dessa menyer gör spelarna val om hur många samt vilka spelare som ska spela mot varandra i en viss omgång.

Flödeschema

Spelets menyer följer en struktur som beskrivs i Figur 6.1. När användaren först sätter igång applikationen så befinner sig denne i Scen 1, vilket är huvudmenyscenen. Användaren har som val att antingen skapa ett nytt spelrum, och bli spelrummets värd, eller ansluta sig till ett befintligt spelrum.

Om användaren väljer att skapa ett spelrum så tas denne till Scen 2, där användaren har som val att ändra på diverse inställningar inför spelet. Dessa inställningar inkluderar begränsningar för maximalt antalet spelare samt val av nivå i spelrummet. När valen är gjorda flyttas användaren till Scen 3, vilket är en spelrumslobby. Väljer användaren att istället ansluta sig till ett befintlig spelrum så tas denne direkt till Scen 3.



Figur 6.1: Flödeschema över scener i spelet

I spelrumslobbyn får användaren se vilka som är anslutna till spelrummet samt vänta på att värden för spelrummet startar spelet. Här har även användaren som alternativ att lämna spelrumslobbyn och återvända till Scen 1. Som tidsfördriv tills alla spelare anslutit sig har en enkel version av ”Skjuta på klossar”-prototypen implementerats. I denna scen kan spelarna även se varandra som *wisps*, små svävande bollar av ljus, i olika färger.

När ett spel har satts igång hamnar spelarna i Scen 4, vilket är själva spelscenen. För att börja spelet måste värden starta en nedräkning, som syns hos alla klienter, innan spelomgången börjar. När ett spelrum tagit slut får alla spelare ett val mellan att gå tillbaka till huvudmenyn eller att spela en omgång till.

Det förekommer inte något menyalternativ för att stänga av spelet och återvända till telefonens huvudmeny, utan detta görs genom att användaren stänger av spelet manuellt i telefonen, alternativt minimerar spelet för att temporärt använda en annan app. Det är inte rekommenderat att minimera spelet under en spelsession eftersom nätverket då kopplas bort.

Representation av menyvalsknappar

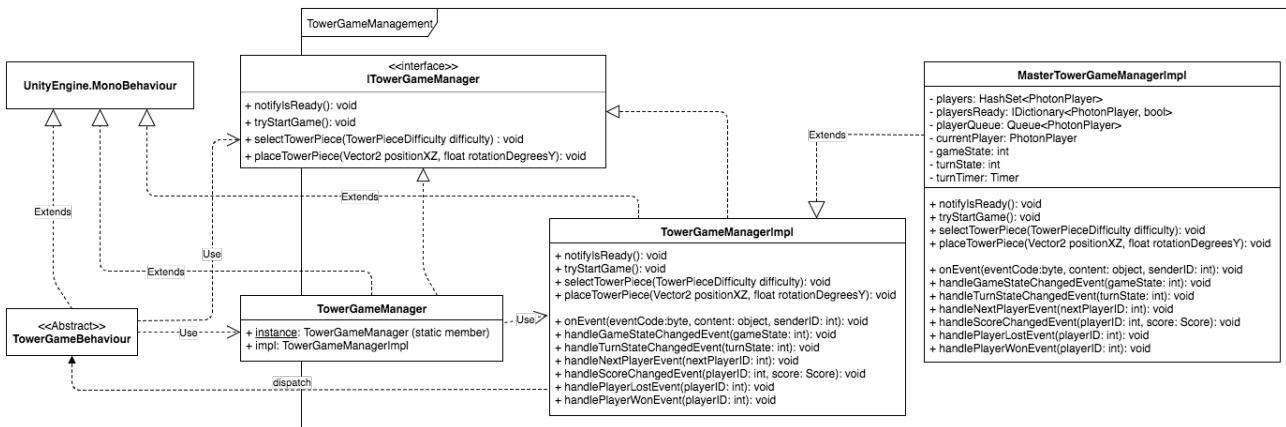
Menyvalen som spelaren presenteras inför representeras av tredimensionella knappar som är statiskt placerade i mitten av den virtuella världen. Dessa knappar är designade så att informationstext för knapparna förekommer på både fram- och baksida, vilket innebär att användaren kan gå runt i VR-världen och alltid se vad knapparna har för funktionalitet. Spelaren utför ett menyval genom att ha fokus på en knapp i VR-världen och sedan klicka med den magnetknapp som finns på Cardboard. Designen på en knapp i spelet visas i figur 6.2.



Figur 6.2: Exempel på en tredimensionell knapp från spelet

6.2.5 Implementation av Photon Unity Networking

Utöver högnivåfunktionerna tillhandahåller PUN funktioner för att skicka nätverksmeddelanden mellan klienterna. Dessa meddelanden definieras av en eventkod, mottagningsgrupp samt innehåll. Innehållet kan vara heltal, flyttal, strängar samt arrayer av dessa. Dessa meddelanden har använts för att implementera kommunikationen kring spellogik mellan klienterna. PUN identifierar den klient som skapade servern som *master client* (värdklient). Gruppen valde att implementera en struktur där beslut kring spelets tillstånd fattades av värdklienten. För att underlätta implementationen i ett senare skede valdes att abstrahera denna värdklients ansvar bakom ett gränssnitt, enligt Figur 6.3. Fördelen med denna implementation är att funktioner kan använda samma kod på alla klienter, även då värdklienten är den klient som mottar alla nätverksmeddelanden.



Figur 6.3: Klassdiagram

Det yttre gränssnittet i nätverksstrukturen definieras av **ITowerGameManager**, ett gränssnitt som definierar de funktioner som ska vara tillgängliga för varje klient. Den implementerande klassen är en **TowerGameManager**, vilket är en *singleton*-klass som även ärver från **UnityEngine.MonoBehaviour** för att kunna placeras som komponent i spelscenen. *Singleton* innebär att det endast kan finnas en instans av objektet i en viss scen. Denna instans refererar i sig till ett objekt som implementerar funktionerna olika beroende på om klienten är värdklient eller inte. Den yttersta delen i strukturen är **TowerGameBehaviour** som andra klasser kan ärvä ifrån för att skapa spelrelaterad funktionalitet på klienten. Klassen exponerar **TowerGameManager**-instansen och möjliggör även för att registrera funktioner som ska anropas då speltillståndet uppdateras på något sätt, exempelvis då det är nästa spelares tur eller när en spelare förlorat.

6.2.6 Spellogik

Eftersom Cardboard endast har en knapp har en standard för hur val genomförs tagits fram. En cirkel renderas hela tiden rakt fram i användarens synfält. När cirkeln svävar över ett objekt som är valbart ökar cirkelns omkrets och det valbara objektet byter färg temporärt. Ifall användaren då ger input via Cardboards magnetknapp väljs det aktuella objektet. Vilket objekt som ska väljas beräknas genom att en stråle sänds ut från den virtuella kameran som krockar med alla objekt i scenen som är aktiva i kollisionshanteringssystemet. När knappen aktiveras väljs det närmsta objekt som strålen krockar med.

För att projicera den valda klossen på tornet har en liknande metod använts. Istället för en stråle sänds en kub ut från kamerans position. Kubens dimensioner omsluter helt det valda objektet. Ifall kuben krockar med ett objekt som är definierat som ”torn” projiceras klossen strax ovanför positionen som kuben fastnade på. Hur objekt definieras hanteras genom lager som kan filtreras bort av strålen eller kuben som sänds ut. Under tiden som användaren funderar på var klossen ska placeras är gravitationen och kollisionshanteringen på klossen avstängd. När klossen placeras aktiveras dessa egenskaper igen. Att klossen placeras något ovanför ytan ställer högra krav på spelarens precision för att tornet inte skall rasa.

Innan metoden ovan valdes testades andra principer för placering av klossarna. En metod gick ut på att följa en stråle från kameran som var vinklad snett uppåt och hitta skärningspunkten i ett plan ovanför tornet. Klossen projicerades sedan ner på tornet. En annan metod, som använde samma kub-kastande som ovan, undersökte ifall ytan som den utsände kuben kolliderat med hade en normal som pekade uppåt. Ifall så inte var fallet projicerades klossen nedåt till en yta som klarade detta kriterium. Med båda dessa metoder medföljde ett flertal problem i själva placeringen, samt att beräkningarna tog längre tid, vilket var anledningen att den första metoden valdes.

När tornet nått en viss höjdkonstant flyttas spelarna och plattformen de står på upp med samma konstant. Detta för att spelarna alltid ska kunna placera en kloss på ovansidan av tornet. Ett horisontellt plan ovanför tornets plattform känner av ifall det är dags att öka plattformens höjd. För att undvika ökning under valfasen eller placeringsfasen detekterar planet endast kollisioner under en kort tidsperiod när tornet är stabilt efter att en kloss har placerats.

Slumpgeneration av namn på spelare och spelrum

Som nämntes i avsnittet ovanför är inputmöjligheterna från användaren begränsade till en enda knapp. För att spelarna ska kunna identifiera sig själva tilldelas varje spelare ett slumpmässigt namn. Namnet genereras utifrån en fördefinierad mängd namnprefix, huvudnamn samt namnsuffix. Alla genererade namn har prefix samt huvudnamn, och med 50% sannolikhet har de även ett suffix. Då det finns drygt 20 alternativ i varje namnklass blir namnvariationen stor. Några exempel på genererade spelarnamn är ”*Drunken Muffin, PhD*”, ”*Masochistic Moose*” och ”*Fierce McBoatface the Third*”. I spelets första meny kan användaren välja att slumptgenerera ett nytt namn, detta görs genom att fokusera på sitt namn och klicka på magnetknappen.

För att en användare som ska ansluta till ett specifikt rum ska kunna identifiera det bland alla aktiva spelrum måste unika namn tilldelas varje spelrum. Detta görs på ett liknande sätt som spelnamnen, men enbart två namnklasser används - adjektiv samt substantiv. Några exempel på genererade servernamn är ”*Medieval Fortress*”, ”*Ancient Sandbox*” eller ”*Roman Planet*”.

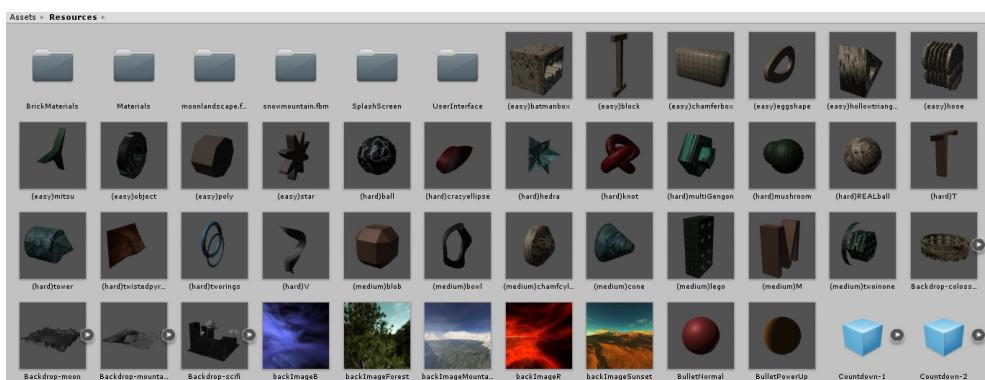
Kapitel 7

Resultat

Efter knappt ett halvårs arbete har ett resultat av projektarbetet börjat ta form. Avsnitten som följer tar upp de olika delarna av detta resultat, framförallt presenteras skillnaderna på slutresultatet från vad som beskrivits i tidigare kapitel. Det framgår även ifall kravspecifikationen (se avsnitt 4.1.3) har blivit uppfylld eller ej.

7.1 Slutprodukt

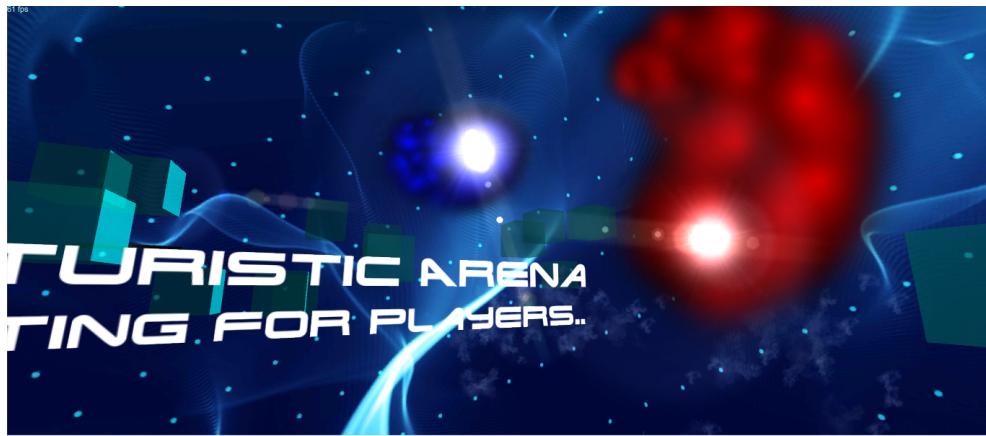
Den sluttgiltiga produkten blev mycket lik den speldé som beskrivits i avsnitt 6.2.1. Spelarna kan i turordning bygga ett torn av olika klossar. I början av sin tur kan spelaren välja mellan tre svårighetsgrader, som alla ger olika poäng. I nuläget finns det cirka 10 olika klossar för varje svårighetsgrad, se Figur 7.1.



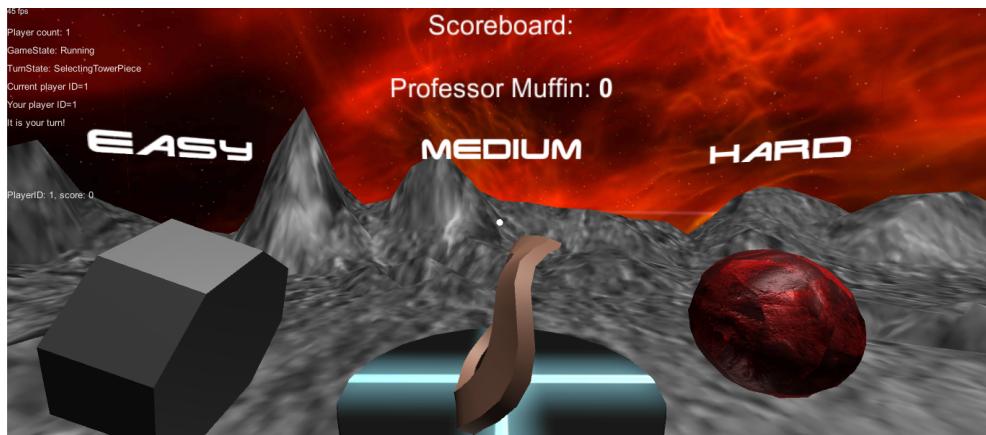
Figur 7.1: Byggklossar

När en spelare ska välja kloss slumpas en kloss ifrån varje svårighetsgrad, och dessa tre klossar presenteras som alternativ framför användaren. Spelarna placerar klossarna som det är beskrivet i avsnitt 6.2.6. I spelscenen och i Scen 3 kan spelarna se varandra som *wisps*, för att ge en bättre känsla av att spelarna spelar tillsammans, detta syns i Figure 7.2. Detta uppfyller även kravet att en spelare ska kunna se sina motståndare. Spelscenen, med ett valt tema (redogörs för längre ner), kan se ut som i Figur 7.3.

Under spelomgångarna finns det en tidsbegränsning på 20 sekunder innan aktuella tillstånd byter till det nästföljande, exempelvis om användaren tar för lång tid på sig för att välja en bit. Detta för att få ett bättre flyt i spelet. Vad som händer när tiden tar slut beskrivs i avsnitt 6.2.1. För att få spelarna mer engagerade och inte tycka det går för långsamt var även tanken att implementera *power-ups*,



Figur 7.2: Spelarnas avatarer - "wisps"

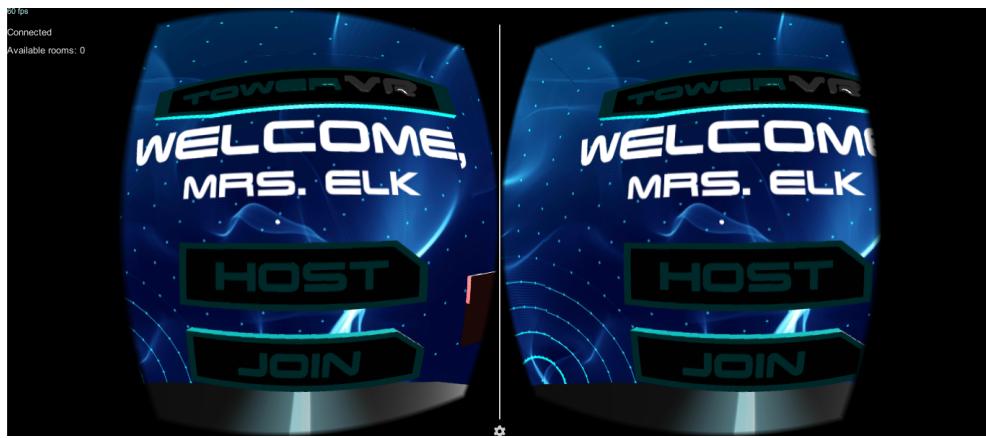


Figur 7.3: Byggklossar

upgraderingar, som påverkar spelet. Detta hänns inte med men förslag på hur dessa upgraderingar borde implementeras undersöktes i användartesterna, se avsnitt 7.2.1.

När spelarna startar applikationen hamnar de i Scen 1 (se Figur 7.4), som är huvudmenyn. Här finns två knappar, en för att skapa ett spelrum (Host) och en för att ansluta till ett redan befintligt spelrum (Join), samt en roterande logotyp. Det finns även en mindre knapp bredvid i form av en röd ruta, som stänger av eller sätter på stereo-läget som visar två bilder. Denna knapp inkluderades temporärt i utvecklingssyfte. Att kunna byta mellan visningslägen är användbart för testning eftersom att det blir lättare att se spelet utan att behöva använda Cardboard. Häданefter används inte stereoläget för bilder på scenerna.

I Scen 2 finns det, förutom val för max antal spelare och en lämna-knapp, fyra olika spelbanor att välja mellan. Dessa presenteras som miniatyrer på ett bord i centrum av scenen (se Figur 7.5). När användaren tittar på en av banorna laddas dess objekt och skybox temporärt in i den virtuella världen för att användaren ska få en uppfattning av hur banan ser ut. En skybox är en samling bilder som läggs på som bakgrundsbilder på kameran och är sedan det första som renderas, de täcker därmed inget i scenen utan tillför endast en känsla av djup. När en bana är vald sparas den och laddas in fullständigt när spelscenen startas i ett senare skede. De olika banorna har olika teman, som även ändrar textur på de klossar som går att välja mellan, för att få en större variation i spelet. De teman som finns är en historisk/antik värld med Colosseum som bas, rymden med fokus på månen, en futuristisk stadsmiljö samt snö där världen omsluts av snötäckta berg.



Figur 7.4: Scen 1, huvudmeny, visat i stereo-läge. Här syns även ett slumpvist genererat namn för en spelare, vilket redogörs för i avsnitt 6.2.6



Figur 7.5: Scen 2, rumsinställningar

Spelarna som anslutit sig till ett spelrum tillbringar väntetiden i Scen 3 (se Figur 7.6), och de kan välja mellan att stanna i denna väntescen eller lämna spelrummet. Om spelrummets skapare startar spelet transporteras alla spelare till spelscenen. Ifall värden inte väljer att starta ett spel och ingen lämnar scenen kommer inget att hänta, så länge ingen förlorar nätverket och blir utkastad ur rummet.



Figur 7.6: Scen 3, spelrumslobby

Spelet går att spela både som ensam spelare samt i flerspelareläge, vilket uppfyller ett av de initiala kraven.

7.2 Spelupplevelse

Spelet är i enlighet med kravspecifikationen implementerat till Google Cardboard och använder sig av positions- (Vuforia) och rotationsbestämning (inbyggt gyroskop). Designprinciperna för VR och för Cardboard som nämns i kapitel 3 har följts för implementering av menyer och navigation.

I avsnitt 4.1.6 förklaras det vad som anses krävas för att projektet ska hålla en hög kvalitetsnivå. Positionsbestämningen skall vara kontinuerlig, bilduppdateringsfrekvensen ska ligga enligt kravspecifikationen och spelet ska byggas modulärt för att lätt kunna vidareutvecklas. Positionsbestämningen för slutprodukten sker inte helt kontinuerligt utan ibland kan den tappas. Spelarens senaste position sparas då och endast rotation tillämpas tills markörkuben hittas igen. Beroende på om spelaren har flyttat på sig eller ej kan det då bli ett ”hopp” i den virtuella världen. Spelet har följt en objektorienterad struktur och går lätt att vidareutveckla med fler banor och funktioner. Bilduppdateringsfrekvensen resoneras det kring i avsnitt 7.4.

7.2.1 Användartester

Under de sista veckorna under projektarbetet utförde gruppen användartester. Detta gjordes genom att först sammanställa ett frågeformulär med frågor som gruppen ville få besvara och sedan utföra användartester med användare i grupper om två till fyra personer. Användarna fick först en kort introduktion till spelet, VR och Google Cardboard. Därefter fick de testa spelet utan fler instruktioner medan projektgruppen observerade. Då projektgruppen ej hade tillräckligt många fungerande Cardboards kunde inte alla användare spela i huvudstyrda stereo-vy samtidigt utan några fick delta med handhållen mono-vy istället. En fråga om hur detta upplevdes lades till i frågeformuläret. Efter att alla användare testat minst en omgång med Cardboard fick de besvara frågorna i formuläret. Användaren svarade muntligt och en projektmedlem antecknade svaren. Det fanns även rum för användaren att komma med egen respons utöver frågorna. De resulterande svaren visas i Bilaga C. Efter användartesterna kunde gruppen fortsätta utveckla de delar som enligt testerna ansågs vara bristfälliga.

På frågan ”Tycker du att spelet känns för segt eller efterdröjande när du tittar runt, för att det ska vara spelbart och underhållande?” har endast 17% svarat ja, vilket uppfyller kravet på max 25%.

7.3 Nätverk

Nätverk implementerades med tillägget Photon Unity Networking (PUN) till Unity. Med låg prestandapåverkan kan ett flertal spelare medverka i en scen; projektgruppen testade med upp till sju spelare utan märkbar skillnad i prestanda. Detta uppfyller kravet på att ha stöd för minst två spelare.

7.4 Prestanda

Slutprodukten har en bildfrekvens som varierar mellan 30 till 60 FPS i alla scener, men den ligger ofta vid det senare vilket är i nivå med kravspecifikationen. Prestandan påverkas bland annat av vilken hårdvara smarttelefonen har och ifall fler applikationer körs i bakgrunden. Även med en bra hårdvara och utan konkurrerande applikationer kan dock bildfrekvensen sjunka temporärt i vissa scener. Anledningen till detta är ännu inte helt klarlagt utan verkar bero på att Unity, till synes slumpmässigt, ibland lägger mer beräkningskraft på vissa delar.

Kapitel 8

Analys och diskussion

Det har varit relativt smidigt att arbeta med Unity, Vuforia och PUN vilket har lett till att gruppen hunnit med mycket under projektets gång. I nuläget är en första version av spelet färdig och gruppen är nöjd med vad som åstadkommits. Nedan analyseras arbetets gång och resultatet mer detaljerat och arbetet sätts i ett vidare sammanhang. Sedan beskrivs även de problem som gruppen har stött på under projektets gång och hur dessa tacklades på bästa sätt under rådande förutsättningar.

8.1 Arbetssätt

Som redogörs för i resultatet har projektets initiala krav uppfyllts. Utvecklingsmetoden har fungerat bra då *Scrum*-principerna har följts och versionshantering tillsammans med verktygen Slack och Trello har varit till god nytta för projektgruppen. Dessa faktorer har bidragit till att kommunikation och arbete fungerat smidigt. Det anses också givande då dessa verktyg används av många företag vilket har gett en inblick av utvecklingsmetodik i arbetslivet.

Projektgruppen höll sig överlag bra till de planerade kodkonventionerna, vilket resulterade i en bra helhetsstruktur. I vissa fall kunde utvecklarna sätta sig in i obekant kod under projektets gång genom att referera till Doxygen-dokumentationen. Profilering som har gjorts har hjälpt för att hitta flaskhalsar i mjukvaran.

Replikerbarhet

När det gäller QA och testning kunde arbetet ha haft ett tydligare förhållningssätt från början för att få replikerbarhet. Struktur på dokumentationen hade kunnat gagnas av en granskning och omformatering halvvägs igenom projektets gång då det blev tämligen rörigt mot slutet. Inställningar och framsteg har samlats i dokument men det blev således lite ostrukturerat och svårt att hitta rätt.

Tidsplaneringen som lades upp i början av projektet har följts till större del. Vissa ändringar skedde mot slutet då uppstartsmötet för Sprint 3 och 4 flyttades något. Undersökningsfasen var viktig för att få en bra överblick över vad som var möjligt att implementera i spelet. Implementeringsfasen borde inletts något tidigare för att ha mer tid till spellogiken men tentamina och gruppmedlemmars olika engagemang vid sidan av skolan gjorde att spelidé-fasen drog ut på tiden. Gruppen ansåg att det var viktigt att alla medlemmar testade på att göra en egen spelprototyp så en extra vecka lades på det istället för att börja med slutprodukten.

Reliabilitet

Eftersom en stor del av tiden lades på undersökningsfaser anser projektgruppen att arbetet håller en hög reliabilitet, åtminstone för tidpunkten för arbetet. Då VR-området fortfarande växer exponentiellt uppkommer det hela tiden nya tekniker som hade påverkat slutresultatet. Under projektarbetets gång har det framkommit ett flertal tekniker som hade gagnat projektet ifall de funnits tillgängliga från start. Exempelvis hade Univrses Playground [17] gjort positionsbestämningen mycket stabilare. Detta innebär att ifall en projektgrupp om ett års tid får exakt samma uppgift kommer de antagligen få ett annat resultat, men ifall de följer samma metod som denna projektgrupp, med liknande programvaror, borde de få ungefär samma.

Validitet

Då projektgruppen gjort en omfattande undersökning av VR-principer (se kapitel 3) och sedan implementerat produkten efter dessa ansåg projektgruppen att åkommor som *cybersickness* borde minimerats. Under användartesterna (Se Bilaga C) svarade två tredjedelar att de inte upplevde några obehag alls när de använde VR-glasögonen vilket anses vara godkänt. Detta resultat hade blivit mer trovärdigt ifall fler användartester hade utförts, något tiden inte räckte till för. De obehag som upplevdes berodde mest på en instabil spårning av kuben. Det var även under de första användartesterna som dessa obehag framkom. Efter den omgången lärde sig projektgruppen att användarna behövde få en tydligare förklaring på hur Google Cardboard fungerar innan de testar spelet. Efter denna insikt upplevde inga fler testpersoner några obehag. För att motivera ifall projektet har en hög eller låg validitet behöver dock fler användartester genomföras.

Källkritik

Gällande källkritik har projektgruppen till stor del förhållit sig till vetenskapliga artiklar om VR-principerna, och andra utvecklare när det gäller teknikerna för nätverks- och positionsbestämning. De vetenskapliga artiklarna har till stor del hittats via Google Scholar eller liknande samlingar och har granskats för att hålla en hög nivå. Referenser till implementeringen har oftast hittats via olika utvecklarforum. Projektgruppen har även gjort flera egna inlägg i dessa forum. I de fall där det finns motsägande åsikter har dessa granskats noggrannare för att bekräfta vilket inlägg som har bäst underlägg.

Kundkontakt

Gruppen kunde ha haft en mer frekvent kundkontakt och därmed haft en klarare uppfattning om vad kunden ville ha implementerat i projektet. Exempelvis så gjorde gruppen en drastisk ändring i grafisk design efter att kunden inte samtyckte med tidiga skisser. Tid som lades ner på detta hade kunnat användas till annat och denna störning i arbetsflödet hade kunnat undvikas med bättre kommunikation. Med fler kundmöten kunde projektgruppen även fått mer input från kund innan en slutgiltig spelidé valts. Andra kundmötet hölls efter att en spelidé spikats av projektgruppen.

8.2 Analys av resultatet

Nedan följer en ingående analys av resultatet utifrån projektets initiala krav, med hänsyn till projektarbetets storlek och tidsram.

Spelupplevelse

Enligt de användartester som utfördes är spelet underhållande, men det upplevs inte som helt färdigt. Exempel på sådant som kan bidra till att spelet inte känns färdigt är att plattformen som spelarna står på är placerad för långt ner. Användarna känner helt enkelt inte att de står på plattformen. Byggklossarna behöver även analyseras och modifieras för att bli mer användarvänliga. Eftersom gruppen inte hann börja med användartester förrän relativt sent i projektet finns det flera liknande detaljer i spelet som skulle kunna omarbetas för en bättre spelupplevelse.

Nätverk

Photon Unity Networking (PUN) som används som nätverkslösning har fungerat stabilt i samtliga tester, och är samtidigt smidigt att implementera i Unity. Inget alternativ till PUN har varit aktuellt sedan gruppen testat detta tillägg, då tillägget har säkerställt att slutprodukten lever upp till nätverkskraven som gruppen bestämde tidigt i projektet. Denna implementering har även tillåtit gruppen att kunna lägga mer tid på andra områden i projektet än just nätverkslösningar.

Prestanda

Bilduppdateringsfrekvensen var tillräckligt bra för att användarna skulle uppleva bra immersion. Vidare angav 67% av användarna att deras rörelser i verkligheten stämde överens med de i spelvärlden till ett värde av 5, där 1 var inte alls och 7 var ”*real life*”. Detta baseras på den återkoppling som gavs via användartesterna som presenterades i sektion 7.2.1, och signalerar att gruppens mål gällande bra positions- och rotationsspårning bör anses tillräckligt.

Det finns dock vissa kvarstående faktorer som ger negativa konsekvenser för spelets prestanda, såsom att vissa telefoner ganska lätt tappar spårningen av kuben. Detta förvärras över att kuben inte är helt slät, se nästa avsnitt. Lösningen som implementerats enligt tidigare redogörelse är att koppla över till rotationsbestämningen från Cardboard SDK samt låsa spelarens position under tiden som markkökuben inte lyckas spåras. Detta resulterar dock alltid i ett hack i spelarens upplevda position i det ögonblick då Vuforia återigen lyckas spåra kubmarkören och sätter spelarens position direkt utifrån det. Detta hack i positionen upplevdes av ett fåtal testanvändare som jobbigt, och av enstaka som extremt jobbigt. Merparten av testanvändarna upplevde dock att deras verkliga position speglades väl i spelvärlden, och därmed har projektgruppens förundersökningar kring VR-principer som redogjordes i 3.2 lönat sig.

En mer avancerad lösning skulle kunna vara att mjukt interpolera spelarens position från senaste position till den som fås från Vuforia, för att minimera detta hack. Implementation av denna lösning påbörjades, men det konstaterades för svårt att omstrukturera scenhierarkin kring Cardboard SDK samt Vuforia under en rimligt kort tid.

8.3 Problem och hinder

Under projektets gång har ett flertal hinder stö�ts på. De flesta av dessa har projektgruppen löst på egen hand men ett fåtal andra kvarstår fortfarande. Detta avsnitt redogör för de problem och hinder som uppkommit.

8.3.1 Unity

I projektets tidiga skede hade utvecklingsteamet problem med Unity. Då flera tredjepartsbibliotek skulle implementeras i spelet samtidigt uppstod det problem. Vidare var initialt även bilduppdateringshastigheten sämre än de uppställda kraven, något som gjorde att arbetet inte fortskred i det tempo som var tänkt. Ett byte från Unity 5 till Unity 4 gjorde att prestandan blev mycket bättre, men i Unity 4 fungerade inte näverksbiblioteket Photon Unity Networking. Till slut lyckades teamet förbättra bilduppdateringsfrekvensen även i Unity 5. Profileringsverktyget i Unity var till hjälp då teamet hittade att en stor del av exekveringstiden gick åt till att rendera mellan olika *framebuffers*. Efter ytterligare undersökning kom teamet fram till att detta berodde på att en *deferred rendering path* användes. Då denna byttes till *forward rendering* höjdes prestandan märkbart till 60 FPS.

8.3.2 Smarttelefoner utan gyroskop

Gruppen bestämde sig tidigt för att utveckla spelet för Android-smarttelefoner och frågade därför projekthandledaren om det fanns möjlighet för universitetet att beställa in telefoner avsedda för testning. Detta godkändes och efter ett par veckor kunde gruppen hämta ut telefonerna. Det dröjde inte lång tid innan gruppen upptäckte att de lånade telefonerna saknade gyroskop vilket är ett verktyg för att mäta mobilens rotation. Detta verktyg är mycket viktigt för att ge en god immersion och används för nästan alla VR-applikationer. Om en mobil utan gyroskop används i en VR-applikation fryser bilden helt eftersom rotationen inte kan läsas av. I detta fall kunde mobilerna dock användas trots detta eftersom både position och rotation i den virtuella världen huvudsakligen bestämdes med hjälp spårning av en bestämd markör, men när mobilen av någon anledning tappar spårning fryser bilden på mobilen vilket ger ett mycket onaturligt intryck för användaren. När en mobil som har gyroskop tappar spårningen använder sig applikationen fortfarande av gyroskopet vilket gör att användaren fortfarande kan se sig om i spelvärlden genom att vrida på huvudet, även fast inte positionen ändras. Detta motverkar *cybersickness* och är därför att föredra.

8.3.3 Markörkuben

Tidigt i projektet skapades en kub med spårningsmarkörer (bilder som mobilkameran känner igen) på varje sida. Kuben användes för positionsbestämning vilket gjorde att spelarens rörelser i den verkliga världen speglades i den virtuella världen. För att skapa kuben användes de material som i stunden fanns tillgängliga och urvalet var mycket begränsat. Kuben skapades av kartong, bilder på utskrivet papper och tejp och därför blev kvalitén inte optimal. Kubens yta blev långt ifrån slät och detta ledde till brister i spårningen. Exempelvis hände det att kameran, på grund av ojämnheter i kubens yta, inte längre kände igen bilderna vilket ledde till att positionsbestämningen inte fungerade korrekt. Ibland läste den även av de böjda delarna av bilderna vilket gjorde att den virtuella världen blev helt förvridden. Gruppen upptäckte detta problem relativt tidigt och diskuterade eventuella förbättringsmöjligheter med projekthandledaren. Ett förslag var att tillverka en slät kub med hjälp av universitetets 3D-skrivare, men efter ett flertal försök visade det sig att skrivaren inte klarade av det. Gruppen försökte sedan att skriva ut en egen kub, men även detta försök misslyckades då vissa kanter hade böjts under utskrivandet (se Bilaga D). Ett annat alternativ var att beställa en kub från ett företag, men efter en del efterforskning visade det sig att detta skulle kosta för mycket pengar och blev därför inte aktuellt. Efter dessa motgångar bestämde sig gruppen för att fortsätta använda den ursprungliga kuben trots den begränsade kvalitén.

8.4 Arbetet i ett vidare sammanhang

VR har på senare tid fått en stor popularitetstillväxt, men även om tekniken nu har funnits ett tag har det inte till stor del lyckats etablera sig som en vardaglig del i människors liv. Detta grundar sig på att VR-enheter varit av det dyrare slaget, vilket har resulterat i en begränsad konsumentgrupp. Nu när billigare alternativ såsom Google Cardboard ökar i popularitet har VR-upplevelsen blivit mer tillgänglig till fler konsumenter i fler samhällsklasser. Detta är en drivande kraft för utbudets samt slutproduktens utveckling, och ger projektgruppen en chans att låna element från detta projekt för att utveckla andra VR-applikationer alternativt introducera nya spelelement i projektet för att nå ut till en bredare konsumentgrupp.

VR-applikationer är inte till för alla. Det finns många människor som mår psykiskt dåligt av att använda VR-glasögon. Även personer med synedsättning har inte samma möjlighet att ta till sig applikationen. På dessa grunder är en VR-applikation diskriminerande redan från grunden eftersom inte alla kan använda den utan att mår dåligt. Vidare har projektgruppen varit försiktig så att varken spelidé eller gränssnitt diskriminerar någon etnisk folkgrupp.

Genom att eventuellt kunna erbjuda alternativ i spelet för att avaktivera diverse spelelement kan gruppen minska risken för att vissa specifika konsumenter känner obehag av att använda VR-applikationen. Exempelvis skulle höjdaspekten i spelscenen kunna avaktiveras till konsumenter med fobi för höjder, vilket innebär att en omarbetad spelscen hade behövts utvecklas av projektgruppen.

Spelet bör även bara spelas i en kontrollerad miljö så varken användaren eller någon åskådare skadar sig. Grundtanken är att spelarna ska kunna röra sig runt i ett rum och då måste rummet vara fritt från hinder. När ett sinne byts ut beter sig kroppen annorlunda så en handledare som inte deltar i spelet bör alltid vara tillgänglig de första gångerna någon testar applikationen.

Kapitel 9

Slutsatser

Projektarbetet svarade på alla frågeställningar som ställdes inför arbetet. Här efter följer en återkoppling av frågeställningarna.

Vilka begränsningar för speldesign och interaktionsmöjligheter medför Google Cardboard som VR-plattform?

Till skillnad från andra VR-plattformer som finns på marknaden har Google Cardboard ingen egen hårdvara utan prestandan beror på den smarttelefon som används. För att en applikation ska klara de krav som ställs för att användaren inte ska bli illamående krävs ofta bättre processorer eller grafik än vad som finns i dagens smarttelefoner. Smarttelefonernas skärmar är även betydligt mindre med en lägre upplösning än de hos konkurrerande VR-plattformar (se kapitel 5). Detta gör att det är mycket lättare att må dåligt ifall man använder Google Cardboard då det helt enkelt inte går att få samma prestanda, upplösning eller synvinkel. Detta resulterar i att immersionen blir sämre.

De enda interaktionsmöjligheterna som finns till Cardboard är att röra på huvudet samt att använda magnetknappen. För en del mobiltelefoner fungerar inte alltid knappen effektivt heller då den sensorn som används för att upptäcka magnetrörelser kan sitta för långt bort ifrån magnetknappen. Eftersom det bara finns en ensam knapp, som ofta reagerar lite långsamt, går det inte att ha några avancerade kommandon eller insignaler utan alla val som användaren ska göra måste vara enkla och ge tydlig återkoppling. Vissa smarttelefoner saknar även gyroskop och kan då inte detektera rotationsrörelser vilket är grundläggande för en VR-applikation.

Hur kan biblioteket Vuforia integreras i ett Unity-projekt för att få positions- och rotationsbestämning till en VR-applikation?

Användarens position och rotation har i spelet spårats med smarttelefonens kamera och en bestämd markör. Bilderna som kameran tar analyseras av Vuforia. Eftersom Unity har fått information om markörens verkliga storlek och rotation kan användarens position och rotation i spelet räknas ut utifrån Vuforias bildanalys. I detta projektet byggdes en markörkub för att spelarna skulle kunna röra sig 360-grader runt markören samt även förflytta sig vertikalt.

För att få en VR-applikation och inte AR, som Vuforia är rekommenderat för (se kapitel 2), stängdes vissa funktioner i Vuforia av för att inte kamerans insignal skulle renderas på skärmen. Kamerans vy omslöts istället av en helt digital verklighet där användaren kunde röra sig runt i. Även renderingen av markörkuben behövde deaktiveras manuellt då standardinställningen var att alltid rendera markörerna som spårades.

Vilka nätverksalternativ finns för smarttelefoner och Unity, och vilket är bäst för realtidsinteraktion mellan spelare?

Det nätverksalternativ som passade bäst för realtidsinteraktion i detta projekt var Photon Unity Networking. Andra alternativ var Unitys inbyggda, *peer-to-peer* m.fl.. I kapitel 5.2 förklaras det mer ingående varför PUN passar bäst.

Hur avancerad kan spelets grafiska stil vara (i form av 3D-modeller, visuella effekter osv.) samtidigt som en tillräckligt bra uppdateringsfrekvens bibehålls för att maximera *immersion* och undvika *cybersickness*?

Eftersom hårdvaran i en mobil är långt ifrån lika kraftfull som i en dator måste den grafiska stilen på exempelvis 3D-modeller vara betydligt lägre. Som tidigare nämnts i kapitel 3 är det extra viktigt med en tillräckligt hög bildfrekvens (50 FPS enligt uppsatta krav i 4.1.3) i VR-spel för att få en bra *immersion* och för att undvika *cybersickness*. För att nå upp till dessa krav krävs att grafikens nivå sänks ytterligare. För att mobiler säkert ska klara av att leverera den bildfrekvens som krävs för ett VR-spel används med fördel så enkla material och texturer som möjligt. Det är mycket vanligt att objekten i spelet är helt släta och enfärgade. I VR-upplevelsen är höjdpunkten själva känslan av att vara innesluten i den virtuella världen, därför är en något enklare grafisk stil tillräcklig.

9.1 Utvecklingsområden

Det finns mycket rum för vidare utveckling av spelet och dess funktionalitet. Det har lagts en grund till konceptet power-ups, som projektgruppen endast hade tid att implementera på en mindre skala i spelrumslobbyn. Tanken var att implementera det i spelet på ett sätt som kombinerade risk med belöning, förslagsvis att power-ups ligger som objekt i tornets ungefärliga bana, och att spelare kan riskera att lägga en mindre säker kloss för att nå och få en power-up. Eftersom denna funktionalitet inte var implementerad till användartesterna var det passande att inkludera en fråga om vad användarna hade för tankar kring integration av power-ups (se Bilaga C).

Ett viktigt utvecklingsområde är utformningen av klossarna som bygger upp tornet. Det var svårt att modellera klossar så att de inte blev för svåra att placera men ändå var en utmaning att placera. Ett annat problem var graderingen av klossarnas svårighetsgrader; det är oklart hur en kloss som är lätt att placera men svår för nästa spelare att placera på ska graderas, och sådana klossar exkluderades ur spelet.

En givande funktionalitet att tillägga vore *high-score*, vilket innebär att spelarnas poäng sparats och visas upp. Det skulle bidra till spelets livslängd, men skulle vara särskilt givande för scenariot att bara en spelare spelar, då en ensam spelare inte har något långsiktigt mål i den nuvarande versionen annat än att bygga ett så högt torn som möjligt varje gång.

Olika spellägen är ett annat område som skulle kunna utvecklas, vilket skulle öka spelets livslängd och göra det mer intressant. Detta genom att variera det vanliga händelseförfloppet för spelare.

Eftersom VR utvecklas snabbt skulle det inom en snar framtid kunna uppkomma framsteg som blir relevanta för spelets utveckling. Speciellt intressant är det om och när teknik kring positionsbestämning till VR för mobila enheter blir allmänt tillgängligt, som experimentet Univrses Playground [17], då det skulle kunna eliminera behovet av markörkuben och på så vis göra spelet mer tillgängligt.

Litteraturförteckning

- [1] Jeff Grubb. *Why positional tracking for mobile virtual reality is so damn hard.* <http://venturebeat.com/2016/02/24/why-positional-tracking-for-mobile-virtual-reality-is-so-damn-hard/>. Skriven: 2016-02-24, Hämtad: 2016-05-30.
- [2] Alexandro Simonetti Ibañez and Josep Paredes Figueras. *Vuforia v1.5 SDK: Analysis and evaluation of capabilities*, 2013. M.Sc. thesis, Universitat Politecnica De Catalunya.
- [3] Edward Andrukaniec, Carmen Franken, Daniel Kirchhof, Tobias Kraus, Fabian Schöndorff, and Christian Geiger. *Advances in Computer Entertainment: 10th International Conference, ACE 2013, Boekelo, The Netherlands, November 12-15, 2013. Proceedings*, chapter OUTLIVE – An Augmented Reality Multi-user Board Game Played with a Mobile Device, pages 501–504. Springer International Publishing, Cham, 2013.
- [4] Daniel Holst, Jens Jakobsson, Pontus Orraryd, Michael Sjöström, and Gabriella Rydenfors. *Splash, sällskapsspel för mobila enheter*, 2015. B.Sc. thesis, Linköpings Universitet.
- [5] Shari L Pfleeger and Joanne M Atlee. *Software Engineering, Fourth Edition, International Edition*. Pearson, 2010.
- [6] Mario Gutierrez, Frédéric Vexo, and Daniel Thalmann. *Stepping into virtual reality*. Springer Science & Business Media, 2008.
- [7] Xin Tong, Diane Gromala, Ashfaq Amin, and Amber Choo. *The Design of an Immersive Mobile Virtual Reality Serious Game in Cardboard Head-Mounted Display for Pain Management*. In *Pervasive Computing Paradigms for Mental Health*, pages 284–293. Springer, 2015.
- [8] Joseph J. LaViola, Jr. *A Discussion of Cybersickness in Virtual Environments*. *SIGCHI Bull.*, 32(1):47–56, January 2000.
- [9] Behrang Keshavarz, Bernhard E. Riecke, Lawrence J. Hettinger, and Jennifer L. Campos. *Vection and visually induced motion sickness: how are they related?* *Frontiers in Psychology*, April 2015.
- [10] LLC Oculus VR. *Oculus Best Practices*. https://developer.oculus.com/documentation/intro-vr/latest/concepts/bp_app_simulator_sickness/. Hämtad: 2016-05-09.
- [11] Google. *Designing for Google Cardboard*. <http://www.google.com/design/spec-vr/designing-for-google-cardboard/a-new-dimension.html#>. Hämtad: 2016-05-09.

- [12] Microsoft Developer Network. *C# Coding Conventions (C# Programming Guide)*.
<https://msdn.microsoft.com/en-us/library/ff926074.aspx>. Hämtad: 2016-05-06.
- [13] Doxygen. *Documenting the code*.
<https://www.stack.nl/~dimitri/doxygen/manual/docblocks.html>. Hämtad: 2016-05-30.
- [14] Digital trends Staff. *Spec Comparison: The Rift is less expensive than the Vive, but is it a better value?* <http://www.digitaltrends.com/virtual-reality/oculus-rift-vs-htc-vive/#:UkES8Kssn652QA>. Skriven: 2016-04-05, Hämtad: 2016-05-30.
- [15] Mobile Phones Spec. *Cell Phone Screen Resolution*.
<http://mobiphonespec.com/cellphone-screen-resolution.php>. hämtad: 2016-05-30.
- [16] Sean Buckley. *This is how Valve's amazing lighthouse tracking technology works*.
<http://gizmodo.com>this-is-how-valve-s-amazing-lighthouse-tracking-technol-1705356768>. Hämtad: 2016-05-09.
- [17] Univrses. *Univrses Playground*. <http://univrses.com/site/playground.php>. Hämtad: 2016-05-09.

Bilaga A

Gruppmedlemmar och ansvarsområden

Organisation		
Gruppmedlemmar	Ansvarsområde	Summering av enskilt arbete
Adam Alsegård	Projekthantering och planering	Stänga av rendering av markörkub, projicering och placering av klossar på tornet, låser användarens position när spårning förloras
Jonathan Grangien	Dokument och kontaktperson	Testande av plattformsversioner, nätverksrelaterad spellogik
Joakim Konac	Kravhantering	Design och modellering av 3D-objekt
Love Lidberg	QA och testing	Implementation av logik bakom uppstart och anslutning till servrar
Måns Lögdlund	Systemarkitektur	Implementation för att förhandsgranska spelnivå samt funktionalitet för att byta scen och ladda in en vald spelnivå till spelscenen.
Victoria Waldemarson	Planering och tid för projektrapport	Undersökning av VR-principer samt design och modellering av scener
Benjamin Wiberg	Kodgranskning	Implementation av nätverkskommunikation samt övergripande kodstruktur

Bilaga B

Stories

Här listas de övergripande *stories* som präglade respektive sprint. Efter varje *story* följer en lista på *tasks* som tillhör respektive *story*.

Sprint 1

- Som spelare vill jag att min position i den virtuella världen överensstämmer min verkliga position.
 - Konstruera markörkub för att användas till positionsbestämning
 - Integrera biblioteket Vuforia och Google Cardboard SDK med spelmotorn Unity3D
 - 360-grader tracking kring kubmarkören
- Som spelare vill jag se mina motspelare på deras verkliga position i den virtuella världen.
 - Implementera nätverkskommunikation via biblioteket PUN.
 - Synkronisera spelarnas positioner mellan enheterna via PUN.
 - Rita markör för de andra spelarna på positionerna som skickas via nätverk.
- Som utvecklare vill jag snabbt kunna skapa prototyper till spelidéer.
 - Skapa en scenmall i Unity som inkluderar ovan nämnda SDK:er och API:er.

Sprint 2

- Som kund vill jag ha ett underhållande spel som följer VR-principer.
 - Utveckla flera spelprototyper utifrån scenmallen från Sprint 1.
 - Bestämma spelidé utifrån de spelprototyper som utvecklats.

Sprint 3

- Som spelare vill jag kunna skapa ett spelrum som andra spelare kan ansluta sig till, eller gå med i ett existerande spelrum.
 - Implementera funktionalitet för att skapa spelrum med PUN.
 - Implementera funktionalitet för att visa vilka spelrum som finns på ett överskådligt sätt.

- Som spelare vill jag kunna variera spelupplevelsen mellan spelomgångarna.
 - Modellera 3D-modeller för spelets olika nivåer.
 - Implementera en meny där spelets skapare kan överblicka de olika spelnivåerna samt välja en som spelet utspelar sig i.
 - Designa olika spellägen (game modes) som varianter av grundspelet.

Denna story uppfylldes partiellt. Användaren kan välja mellan 4 spelbara nivåer, men enbart ett spelläge finns tillgängligt.

Sprint 4

- Som kund vill jag ha ett välpolerat spel som spelare finner underhållande.
 - Genomföra användartester för att få feedback på spelets underhållsfaktor samt implementationen av VR
 - Korrigera spelet efter användarnas önskemål.

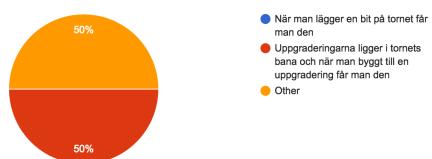
Bilaga C

Resultat från användartester

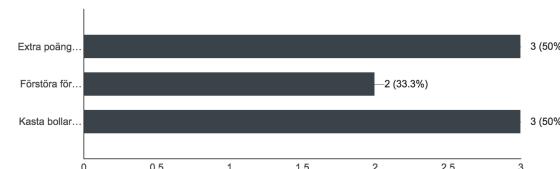
Vad tycker du om designen? (6 responses)

Cool, men opolerad. Det känns ambitiöst. Bra variation i miljöerna. Det är konsistent grafiskt språk, känns "killigt"
Gillade miljöerna, svårt att veta vad man skulle göra. Svårt att förstå hur cardborsten skulle användas, hur man styrs den. Olika storlekar på valbara knappar är oklart. "Power-off"/"Leave game", svårt att förstå vad de gör. Svårt att lägga ett objekt på rätt plats. Poängtavlan låg rakt i solen, såg inte vad det stod.
Designen var bra. Jag kände mig väldigt liten, behövde hålla mobilen högt för att det skulle fungera => planen längre ned.
Bra helhet på konceptet, allt gick i samma tema. Lagom storlek på knapparna. Bra med feedback.
Retro, spacat. Typ som Quake 3, (positivt)
Fräsch, tydliga knappar. Förstod vad alla gjorde (Förutom röda AR-knappen).

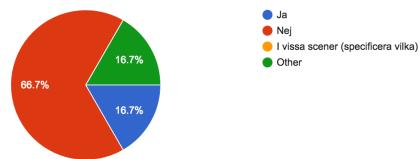
Hur skulle du vilja få tag i power-ups? (6 responses)



Vad skulle du vilja ha för power-ups? (6 responses)



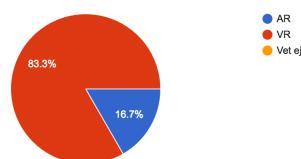
Tycker du att spelet känns för segt eller efterdröjande när du tittar runt, för att det ska vara spelbart och underhållande? (6 responses)



Upplever du något obehag när du har på dig VR-glasögonen? (hur?) (6 responses)

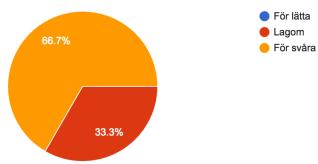
Ja, lite konstigt med höjdskillnaden, känns som man tittar rakt fram men tittar ned i vr-världen
Ja. Småryck i början. Texten kom rakt i ansiktet på en. Hoppade fram och tillbaka. Gick inte att se lädan i glasögonen, omkalibrering var jobbigt. Mot slutet skakade det massvis, väldigt obehagligt.
Nej, inte alls.
Visste inte vart man var. Annars inget.
Upplevde inget.
Nej

Skulle du heller spela spelet i AR eller VR? (6 responses)

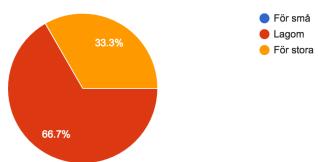


Figur C.1: Användartester

Vad tycker du om svårighetsgraden på bitarna man bygger med? (6 responses)



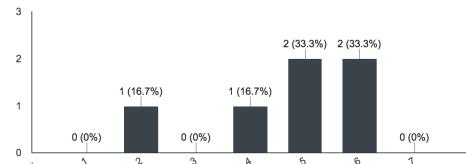
Vad tycker du om storleken på bitarna? (6 responses)



Ger bitarna lagom med poäng? (6 responses)

- Ja
- Såg inget poägsystem, svårt att säga därmed.
- Easy/Medium/Hard, såg inte poägen.
- Såg inte hur mycket poäng de gav
- Såg inte poängen-
- Såg inte.

Hur väl stämmer dina rörelser i verkligheten överens med de i spelvärlden?
(6 responses)



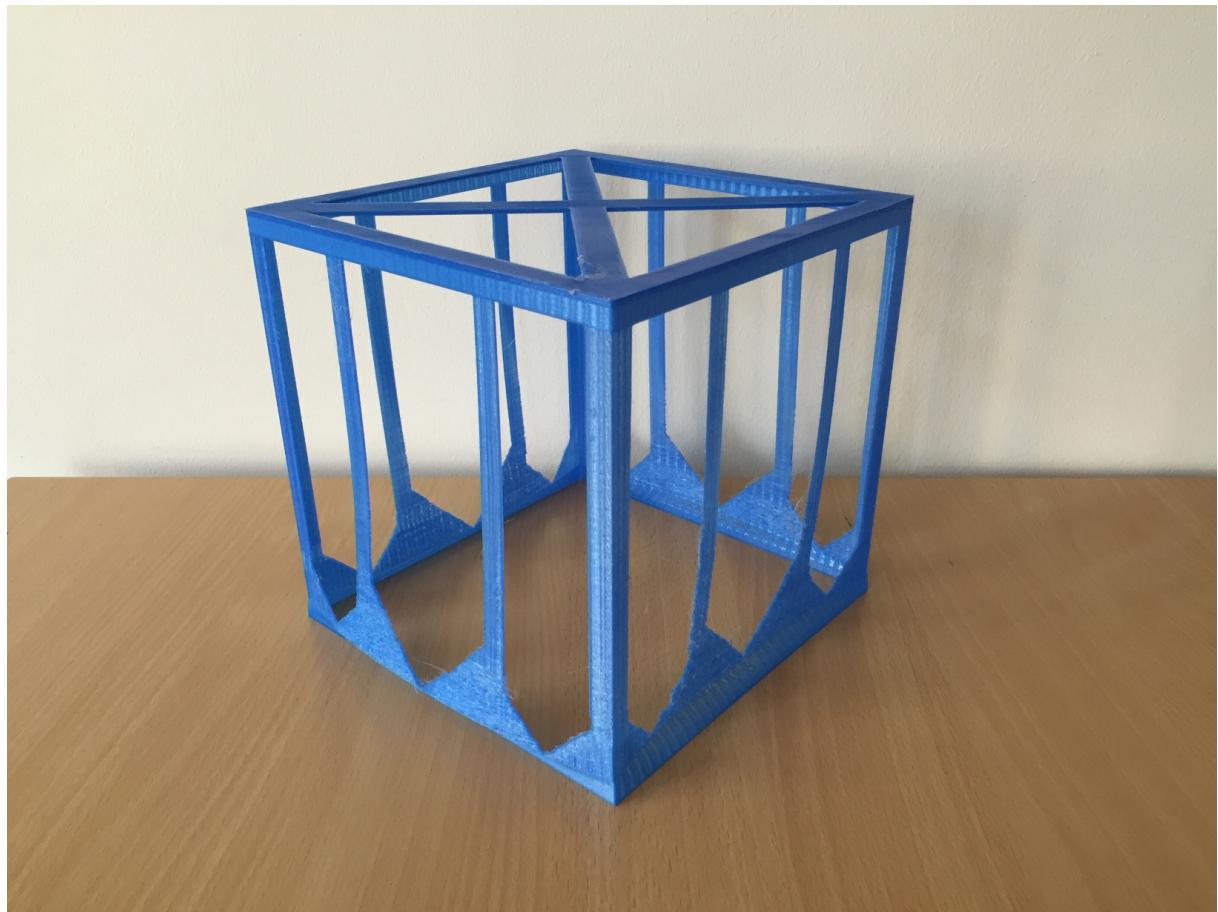
Har du några förbättringsförslag till spelet? (6 responses)

- Text i lobbyscenen för "start game" eller "leave game". Gör också de ikonerna olika stora, startknappen bör vara större. "Waiting for players" to join
- Power-up: extra liv. Progression i svårighetsgraden på bitarna, lättare först. Bordet var läspunkt, därmed 2 "poäng".
- Att biten inte roterar med mig när jag går runt, den ska ha statisk rotation. Timern för att placera bit bör visas på något sätt.
- Några av bitarna var lite för svåra! Kanske inte byta till röd färg på startknappen när man ska starta spelet. Gillart!
- Lite otydligt när man skulle starta spelet, borde följa samma tema. Leave-knappen i backdrop var alldeles för högt. Gör scenerna lite jussare (sci-fi)
- Röligt att jävelas med motståndarna. Det var svårt att tracka högt upp/lägga högt upp.

Figur C.2: Användartester

Bilaga D

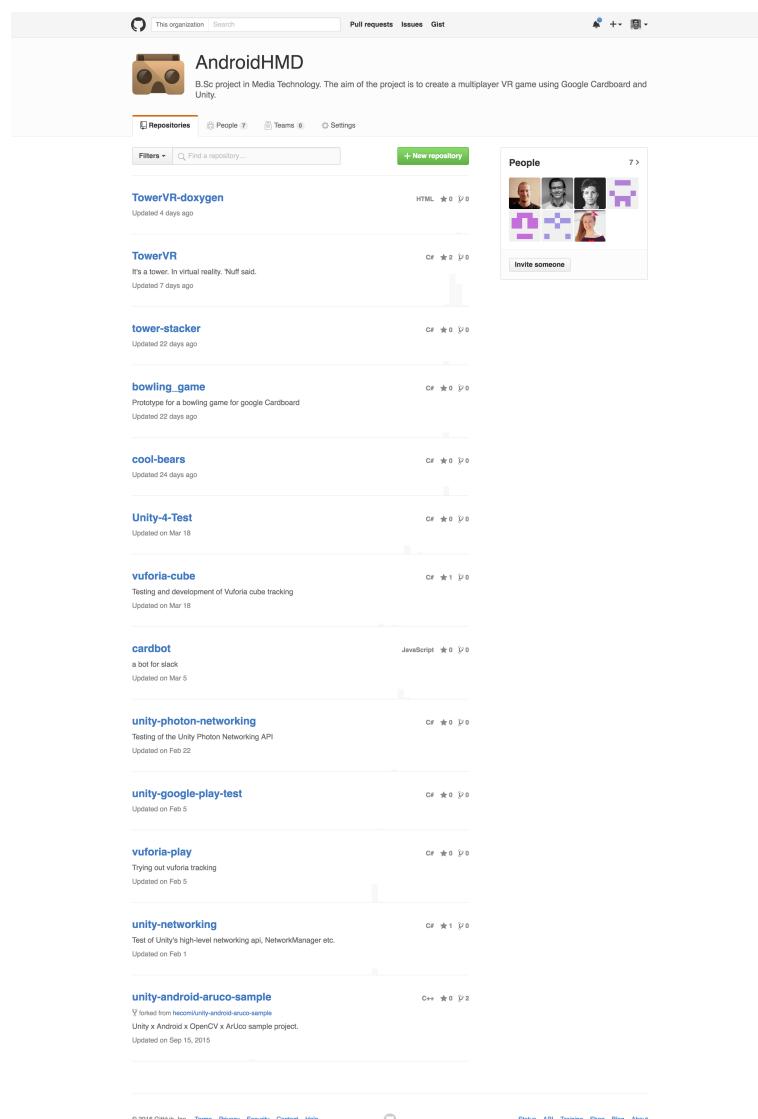
Misslyckat försök till utskriven kub



Figur D.1: 3D-printad kub som böjdes under utskrivningen. Den gick inte att använda till projektet.

Bilaga E

GitHub - organisationen AndroidHMD



Figur E.1: GitHub - organisationen AndroidHMD

Bilaga F

Doxxygen - klassindex

TowerVR 1.0

The screenshot shows the Doxygen Class Index for the TowerVR 1.0 project. The interface has a header with tabs for Main Page, Packages, Classes, Files, Class List, Class Index (which is selected), Class Hierarchy, and Class Members. A search bar is at the top right. The main content area is titled "Class Index" and contains a grid of class names. Each class name is preceded by a small icon representing its type (e.g., a person for Player classes, a gear for NetworkedBehaviour). The classes are grouped by letter:

C	H	N	S	Torus
ComponentPreviewer	HandleEndGame (TowerVR)	NetworkedBehaviour	RetrieveAndSpawnPlayers	Torus
ConnectAndCreateTowerManager	HandleStartGame (TowerVR)	NetworkEventCodes (TowerVR)		TorusGeneral
ConnectAndJoinTest		NextPlayerEvent (TowerVR)		TowerConstar
CreateNewGame	I		Score (TowerVR)	TowerGameB
	IncreaseHeight (TowerVR)		ScoreChangedEvent (TowerVR)	TowerGameM
D	ITowerGameManager (TowerVR)		ScreenLog	TowerPiece (T
DestroyBullet	J		SelectionPieceHovering	TowerState (T
F	JoinServer	P	SelectTowerPieceEvent (TowerVR)	TowerStateCh
Fading	L	PhotonNetworkEvent	SetMaxPlayers	TransformDet
FallingTowerDetection (TowerVR)	LifetimeDebugger	PlaceTowerPieceEvent (TowerVR)	Shoot	TriggerDebug
FPSDisplay	LostTracking (Vuformia)	PlacingBricks (TowerVR)	ShowServers	TryStartGame
G	M	PlayerCameraMovement	Singleton	Tuple
GameState (TowerVR)	MasterClientOnlyBehaviour (TowerVR)	PlayerLostEvent (TowerVR)	SpawnPowerUps	TurnState (To
GameStateChangedEvent (TowerVR)	MasterClientOnlyBehaviour	PlayerNameChooser	SpawnSelectedLevel	TurnStateCha
GazeOver	MasterTowerGameManagerImpl (TowerVR)	PlayerNameGenerator	TestMoveBehaviour	TurnTimeLim
GazeOverPreview		PlayerReadyEvent (TowerVR)	Timer	r
		PlayerTurnObserver (TowerVR)	ToggleVR	removeTarget
		PlayerWispCameraMovement		
		PlayerWonEvent (TowerVR)		
		PowerUpCollision		
	R	RandomizeServerName		

Figur F.1: Doxygen - klassindex