

Kortspelsaktivverad strid

Evaari - Datorspel i VR

Adrian Andersson, adran117
Anton Cevey Tärnholm, antta334
Anneli Nilsson, annni204
Gustaf Wallström, gusan112
Jonas Långdahl, jonla677
Tobias Olsson, tobol101
Tobias Ryttlinger, tobry437

Examinator: Daniel Jönsson

Sammanfattning

Följande rapport behandlar utvecklandet av ett spel i kursen *TMN094 - Medietekniskt Kandidatprojekt*. Rapporten tar upp hur en grupp om sju personer arbetat med agil systemutveckling genom ramverket *scrum* för att skapa ett spel. Spelet i fråga är ett VR-spel med kortspelsaktiverad strid i realtid. Spelet tillåter två spelare som i realtid spelar mot varandra. Eftersom spelet utspelar sig i en virtuell miljö så måste samtliga spelare använda *HMD*, närmare bestämt *HTC Vive*. För att effektivt kunna utveckla spelet har gruppen arbetat med *UML*. Flera krav ställs på systemet från en kund vilket innebär att kravspårning och kravhantering har använts för att säkerställa att dessa uppfylls. Diverse frågeställningar som hör till utvecklandet och spelet i fråga presenteras tidigt i rapporten och svaras på under rapportens gång. Det förekommer även en sammanfattande diskussion om både frågorna och svaren som hittats.

I spelet utvecklades även ett åskådarläge i *AR* som dock inte förekommer i slutprodukten. Tanken var att tillåta fler spelare att kunna åskåda en pågående spelsession som utspelar sig i en virtuell värld. Detta skulle möjliggöras genom att skanna en fysisk bild med en mobiltelefons kamera. Modellerna i spelet är skapade efter en lågpolygonstil och till flera av dessa modeller finns animationer. Allt detta beskrivs utförligt i rapporten.

I den virtuella miljön finns två baser samt en spelplan. Spelarna har möjlighet att plocka upp kort samt lägga ut dem på spelplanen. När ett kort spelas så skapas en figur som har möjlighet att attackera motståndarens figurer eller att försvara mot dem. När ena spelaren tar sönder sin motspelares bas avslutades matchen och spelaren vinner. Olika figurer har unika värden på sin attackförmåga samt andra egenskaper. Inom genren realtidsstrategi är ett obalanserat spel särskilt utsatt då spelare kan utnyttja detta för att få en orättvis fördel. För att skapa en bra användarupplevelse har gruppen även arbetat mot att balansera spelet så att matcherna blir så rättvisa som möjligt.

Innehåll

Sammanfattning	i
Figurer	v
Tabeller	vi
1 Inledning	1
1.1 Syfte	1
1.2 Frågeställning	1
1.3 Avgränsningar	2
2 Bakgrund	3
2.1 Relaterat arbete	3
2.1.1 Cross World Portals - ett inlägg av Alan Stafford	3
2.1.2 Clash Royale - mobilspel med kortspelaktiverad strid	4
3 Projekthantering	5
3.1 Rutiner och principer	5
3.2 Utvecklingsmiljö	5
3.3 Tidsplan	6
3.4 Administration Trello	6
3.5 Versionshantering	8
4 System och tekniska lösningar	9
4.1 Grundläggande, initiala krav och systembegränsningar	9
4.2 Målplattform	10
4.3 Grundläggande system-arkitektur	10
4.4 Standarder	11
4.5 Kvalitetssäkring	11
5 Utvecklingsprocessen	12
5.1 Spelet - VR	12

5.1.1	Bordsplacering	12
5.1.2	Rutnät	13
5.1.3	Speltimer - Mana	13
5.1.4	Kort	13
5.1.5	Monster	15
5.1.6	NavMesh och NavMeshAgents	15
5.1.7	Byggfas	16
5.1.8	Spelomgång och huvudmeny	16
5.1.9	Spela ut kort	17
5.1.10	Animationer och Ljud	18
5.2	Figurer, landskap och byggnader	18
5.3	Nätverk	20
5.3.1	Photon Unity Networking	20
5.3.2	Unity UNet	20
5.4	Augmented reality	20
5.4.1	Försök 1	21
5.4.2	Försök 2	21
5.4.3	Försök 3	21
5.4.4	Felsökning	21
6	Resultat	23
6.1	Slutprodukten	24
6.2	Krav	24
7	Analys och diskussion	25
7.1	Metod	25
7.1.1	Projekthantering	25
7.1.2	System och tekniska lösningar	25
7.1.3	Utvecklingsprocessen	26
7.1.4	Källkritik	26
7.2	Resultat	26
7.3	Arbetet i ett vidare sammanhang	26
8	Slutsatser	28
8.1	Svar på frågeställningar	28
8.2	Slutsats	29
Litteraturförteckning		29

A Tidsplan	32
B Ansvarsområden	34
B.1 Summering av enskilt arbete	34

Figurer

3.1	Översikt av planeringen av arbetsuppgifter i Trello.	7
3.2	Översikt av färdiga arbetsuppgifter och arkivet i Trello.	7
4.1	Översikt av spellogiken av korten.	9
4.2	Översiktlig arkitektur av systemet.	10
5.1	Spelplan med rutnät.	13
5.2	Spelarens korthand med speltimer (mana).	14
5.3	Spelets olika kortklasser	14
5.4	Monsters prioritet och handlingar i spelet.	15
5.5	Bild på <i>NavMeshen</i> som användes i spelet.	16
5.6	Aktivitetsdiagram över kortblockning.	17
5.7	Resultatet av första försöket med att ta upp ett kort.	18
5.8	Exempel på modeller och karakterer som förekommer i spelet.	19
5.9	Landskapet som används i spelet.	19
5.10	Tre olika stadier i åskådarapplikation.	21

Tabeller

5.1 Tabell över fördelningen av kortklasser	14
B.1 Projektets medlemmar med ansvarsområden	34

Kapitel 1

Inledning

Följande projektrapport behandlar utvecklandet av ett spel till kursen TNM094, Medietekniskt Kandidatarbete. Spelet i fråga kategoriseras som *VR*, virtuell verklighet, och *AR*, *augmented reality*, samt inom genren realtidsstrategi. Vidare så spelas spelet med *HTC Vive* på *PC* och på *Android*-telefoner.

1.1 Syfte

Syftet med projektet är att undersöka hur *VR*- och *AR*-tekniker kan användas till spelutveckling, med hjälp av agil utveckling med ramverket *scrum* som underlag. Att arbeta agilt i grupp är ett förberedande moment inför många utmaningar som de flesta utvecklare stöter på i arbetslivet. Vidare får alla utvecklare ett unikt tillfälle att kunna jobba med hårdvara som annars kan vara dyr eller svåråtkomlig. Under projektets gång ska samtliga utvecklare öka sin förståelse och kunskap om moderna *AR*- och *VR*-tekniker som används inom spelutveckling. Även testa nya tekniker som är mindre utvecklade inom industrin som att ha ett åskådarläge i *AR* på mobiltelefoner som visar den aktiva *VR*-scenen. Utöver det tidigare nämnda, ämnar gruppen att undersöka hur sten-sax-påse modellen kan modifieras för att passa ett spel i realtid.

Projektet utmanar samtliga utvecklarens programmeringsfärdigheter och förståelse för speldesign såväl som kommunikation och *scrum*. Slutprodukten ämnar uppfylla de krav som kunden ställer inom den givna tidsramen. Vidare liknar projektet till stor del de verkliga uppdrag utvecklare stöter på i arbetslivet inom mjukvaruutveckling.

1.2 Frågeställning

- Hur kan *VR* och *AR* kopplas samman i samma spelvärld för att fördjupa spelupplevelsen med hjälp av en extra plattform i form av mobiltelefoner?
- Hur kan sten-sax-påse modellen modifieras för att anpassas till ett realtidsspel mellan flera spelare för att ge en balanserad spelupplevelse?
- Hur kan den virtuella världen synkroniseras med den verkliga världen för att ge användarna återkoppling kring position av bord och motspelare i den virtuella världen?

1.3 Avgränsningar

En medveten avgränsning som gjorts är att inte utforska andra metodiker än *scrum*. Spelet utvecklas mot *HTC Vive* och *PC* med operativsystemet *Windows 7* eller senare samt *Android*-telefoner med version 7.0 eller högre som är ett krav för *ARCore* som är ett API för AR utveckling. Målgruppen är personer med spelintresse, men även har ett intresse för strategi, *VR*, och ny teknik. Spelet kommer inte tillåta spelare att möta varandra över internet och det kommer inte skapas ett enspelarläge. Det kommer inte gå att spela spelet utan *VR*, dock kommer mobiltelefoner ej krävas för att spela spelet.

Användare med daltonism tas inte i åtanke under utvecklingen av spelet.

Kapitel 2

Bakgrund

Uppdraget som tilldelades till gruppen var att utveckla ett spel i VR inom genren realtidsstrategi. Genom att placera olika kort skapar spelaren figurer och placerar byggnader på en virtuell spelplan. Kraven är att två spelare ska möta varandra i samma virtuella rum, se varandras rörelser och interagera med den andres figurer. Figurernas attacker och försvar balanseras enligt sten-sax-påse strategi.

I spelteorin finns ett uttryck som kallas *Nash Equilibria*, sv. Nashjämvikt, myntat av John F. Nash. Nashjämvikt hävdar att om två lika skickliga spelare möter varandra finns det till slut ingen strategi som en ensam spelare kan förbättra för att öka sina chanser att vinna. Ett jämviktsläge uppstår [1]. Vid ett spel som sten-sax-påse blir detta tydligt då två spelare spelar otaliga runder mot varandra och hela tiden anpassar sin strategi efter sin motståndares drag.

Det ska finnas möjlighet för ytterligare spelare att ansluta med mobiltelefon till rummet som åskådare. Åskådaren ser endast spelplanen som AR, dvs att den virtuella planen med figurerna syns i den verkliga världen.

Augmented reality, eller på svenska, förstärkt verklighet myntades 1992 av Thomas Caudell & David Mizel [2]. På senare år blev det populärt genom mobilspelet *Pokémon Go*. AR innebär att använda den riktiga världen i kombination med en eller flera virtuella, datorgenererade, objekt. Detta skiljer sig från en virtuell verklighet som enbart består av datorgenererade objekt och miljöer [3]. Användningsområdena för AR sträcker sig utanför spelindustrin och används idag inom flera stora industrier som till exempel möbel- och sportindustrin [4].

2.1 Relaterat arbete

Under projektets gång har gruppen tagit inspiration och lärdom av följande produkter och texter. Dessa texter tar upp delar av vad projektet handlar. Att använda AR i samband med VR är tillräckligt unikt och utforskat för att antalet relaterade arbeten ska vara få. Därför ämnar denna rapport att kunna bidra med sitt innehåll för att hjälpa gemenskapen.

2.1.1 Cross World Portals - ett inlägg av Alan Stafford

Alan Stafford skriver i ett inlägg att det är möjligt att med hjälp av AR åskåda ett pågående spel som utspelar sig i VR. I sitt inlägg framgår det att även han har använt spelmotorn Unity och ARCore [5]. Inlägget i sig är dock kortfattat och beskriver mer hur telefonerna kan synliggöras i den virtuella miljön än tvärtom.

2.1.2 Clash Royale - mobilspel med kortspelaktiverad strid

Ett spel som också baserar sig på kortspelsaktiverad strid är *Clash Royale* till mobila enheter. I spelet möter två spelare varandra och spelar kort som framkallar figurer som i sin tur strider mot varandra på en spelplan. Korten har olika attribut såsom hur mycket skada de gör på varandra och hur mycket skada de kan ta emot med mera. Spelet går på tid och är över när någon av spelarnas huvudbas går sönder eller när tiden är slut. I *Clash Royale* har varje spelare en huvudbas samt två torn som är färdigt utsatta på spelplanen, detta skiljer sig emot detta projekt där tornen sätts ut på valfri plats av spelplanen utav spelarna [6].

Kapitel 3

Projekthantering

I detta kapitel redovisas tidsplanen, utvecklingsprinciper och verktyg som användes under projektet.

3.1 Rutiner och principer

Projektet följde agil utvecklingsmetodik enligt *scrum*, som gjorde att varje krav från kunden betraktades som en post. Dessa poster hamnade i en *backlog* som gav upphov till sprintar. Från projektets start identifierades alla sprintar och detta var en utmaning som krävde planering. Sprintplaneringen rangordnade sprintarna efter hur mycket resurser de krävde. Efter detta kom utvecklingslaget överens om vilka utvecklare som tog sig an vilka arbetsuppgifter.

Varje sprint motsvarade i normala fall två veckor. Varje sprint innehöll flera arbetsuppgifter. Sprintarna påbörjades med en sprintgenomgång och avslutades med en sprintåterblick. Sprintgenomgången förberedde utvecklingslaget för vad sprinten innebar och vilka arbetsuppgifter som skulle genomföras för att kunna slutföra den. Sprintåterblicken gav utvecklingslaget en chans att utvärdera en nyligen slutförd sprint. I enlighet med *scrum* så började dagen med ett kort möte som styrdes av den utsedda *scrum*-mästaren. Under dessa möten berättade utvecklarna i tur och ordning vilken arbetsuppgift i sprinten de arbetade med och hur det gick [7]. För att inte överskrida tidsåtgången för dessa scrummöten hölls de stående. Det är på dessa möten som varje utvecklare har blivit uppdaterade på vad som görs i varje komponent och de kunde på så sätt vara delaktiga i varje beslut som gjordes.

För att underlätta det agila utvecklandet användes programvaran Trello som är en *online* kanbantavla. Anteckningarna i kanbantavlorna representerade arbetsuppgifter.

3.2 Utvecklingsmiljö

Verktyg som användes i implementationens- eller dokumentationssyfte:

- *Unity* är spelmotorn som användes då den har bra stöd för VR och gruppen har tidigare erfarenhet i programmet. API:er som användes tillsammans med Unity var *SteamVR* och *OpenVR*, för att hantera indata från VR-utrustning.
- *Visual Studio* är den integrerade utvecklingsmiljön (*IDE*) som användes. Den valdes baserat på dess väletablerade status och anpassningsbarhet.
- *Overleaf*, som är ett nätbaserat verktyg, användes för rapportskrivning i typsättningssystemet *LAT_EX*.

- *FL Studio* användes för utveckling av ljud. *FL Studio* valdes baserat på tidigare erfarenhet.
- *Blender* användes för modellering eftersom programmet är gratis och väletablerat samt tidigare känt av flera i utvecklingslaget.
- *Adobe*-program såsom *Illustrator* och *Photoshop* användes för 2D grafik och texturer. Dessa program användes eftersom båda är väldigt välimplementerade och användarvänliga med god dokumentation och möjlighet för support.
- *Trello* användes flitigt då det är ett program som underlättar agil utveckling. *Trello* innehåller projektets backlog.
- *Git* och *Github* användes till versionshantering i projektet, dels för att gruppen sedan tidigare har använt programmen och för att de är väl integrerade i både den valda IDE:n och i *Unity*.

3.3 Tidsplan

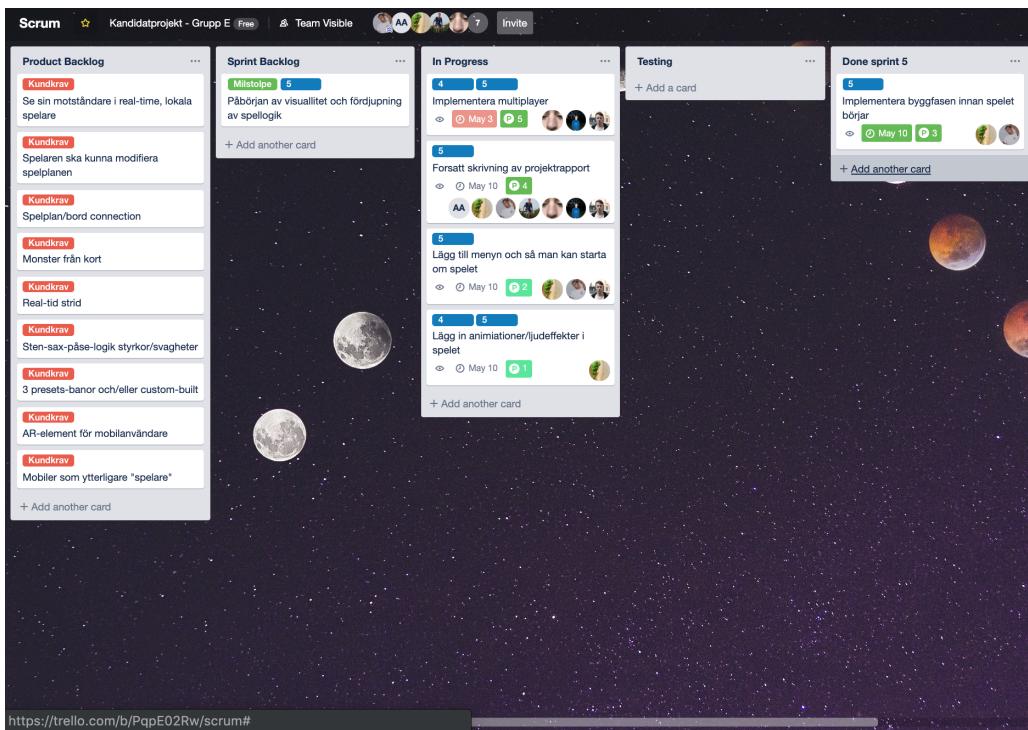
Tidsplanen är utformad efter olika sprintar som varar i två arbetsveckor med undantag av den första sprinten som varar i tre. I varje sprint finns det arbetsuppgifter som beskriver mer i detalj vad som är tänkt att genomföras under sprintens gång. I tidsplanen framgår det vilka utvecklare som ansvarar för vilka uppgifter och när de är tänkta att genomföras. Tidsplanen syns i bilaga A. Kolumnen "PCT FÄRDIGSTÄLLANDE" fungerar som en indikation på hur långt utvecklarna kommit med varje arbetsuppgift, men är främst där för att ingen uppgift ska glömmas. De ifyllda rutorna under sprintarna indikerar vilka dagar utvecklarna väntas arbeta.

3.4 Administration Trello

Administrationen kring utvecklingen sköttes med *Trello* [8] som är ett verktyg som visuellt visar varje utvecklares arbetsuppgifter. De kan användas för att planera framtida uppgifter, göra återblickar samt övervaka nuvarande arbeten.

I *Trello* kommer utvecklingen vara utformad efter *Scrum*, vilket innebär att utvecklingen kommer ske i såkallade *sprintar* som varar två arbetsveckor med start på måndag. I varje *sprint* kommer det finnas olika arbetsuppgifter som planeras i början av sprinten och som är utformade efter tidsplanen. Dessa uppgifter representeras i *Trello* i form av kort. På ett kort ska utvecklaren som arbetar med det taggas. Genom att tagga utvecklaren kan andra utvecklare enkelt se vem det är dom skall fråga om saker gällande den uppgiften/funktionen senare i relaterat arbete eller senare i utvecklingen. Korten som planeras i början av sprintarna är direkt kopplade till de uppgifter som står under tidsplanen i den sprinten. Om en uppgift i tidsplanen anses för stor delas den upp i flera kort. Korten används sedan för att visuellt visa vilka utvecklare som jobbar med vad och i vilket stadie av utvecklingen uppgiften är i, det vill säga utveckling, testning eller klart.

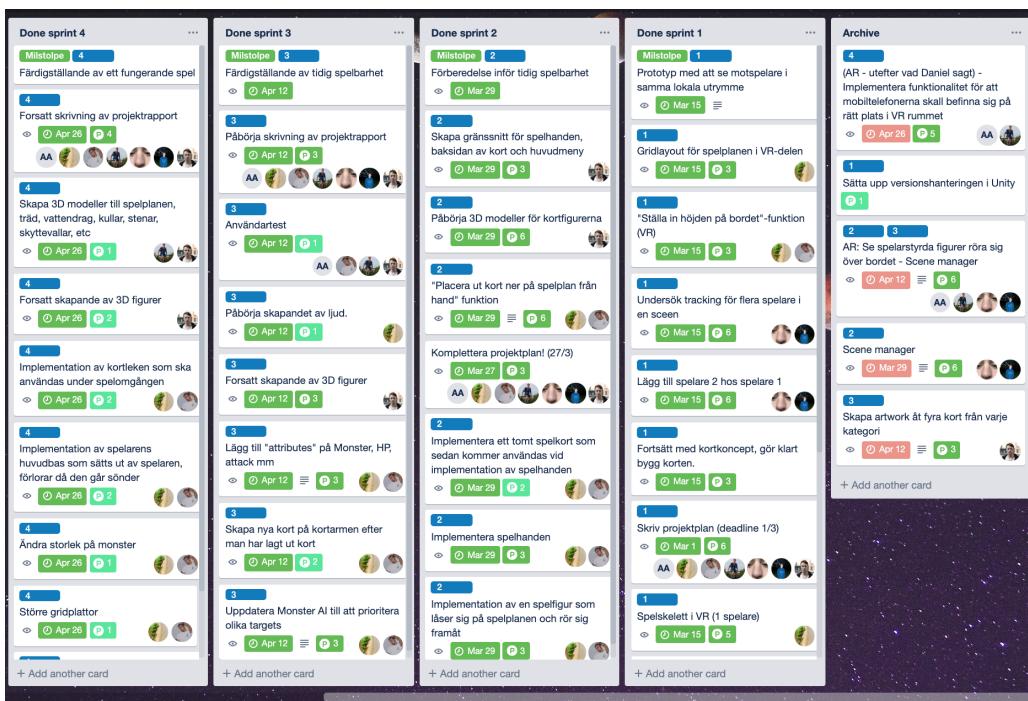
I figur 3.1 visas hur planeringen av korten i *Trello* ser ut. Längst till vänster finns kolumnen *Product Backlog* där kort med kundkrav är utformade, dessa skapades i början av projektet och förhålls statiska under hela utvecklingen. Alla sprintar och arbetsuppgifter är ämnade att vara utformade på så vis att de arbetar emot dessa kundkrav. *Sprint Backlog* är den kolumn där korten planerades. De fick en tagg beroende på vilken sprint de tillhörde samt hur många poäng de hade. Vilka utvecklare som ansvarade för att uppgiften skulle genomföras taggades även i kortet. Att en utvecklare taggades i ett kort gjorde det enkelt för de andra utvecklarna att veta vem de skulle fråga om de hade frågor kring olika funktioner. Detta tydliggjordes med hjälp av sprintåterblickarna i *Done Sprint* kolumnerna. Om



Figur 3.1: Översikt av planeringen av arbetsuppgifter i Trello.

en utvecklare skrev en kommentar på ett kort får även de utvecklarna som var taggade i kortet en notis. Poängen som korten fick indikerade hur lång tid uppgiften skulle förväntades ta att genomföra, där ett poäng var något som kunde göras på en halvdag och sex poäng tog en hel arbetsdag.

När korten var skrivna i *Sprint Backlog* så var de självbärande och utvecklarna som jobbar med kortet flyttade de till *In Progress* kolumnen där utvecklingen påbörjades. När de var klara med utvecklingen genomfördes kvalitetssäkring/testning och då flyttades kortet till kolumnen *Testing*. Minst två utvecklare gick igenom den kod som skrivits och testade funktionen som utvecklats. När testningen var klar



Figur 3.2: Översikt av färdiga arbetsuppgifter och arkivet i Trello.

lades kortet i någon av kolumnerna *Done Sprint* beroende på vilken tagg kortet hade. När kortet låg där indikerade det att kortet och arbetsuppgiften var helt färdig. Det finns olika kolumner för färdiga kort som är upplagda utefter sprintarna i tidsplanen. Detta underlättade sprintåterblickarna och gjorde det enkelt att se vad som gjorts och när. Se figur 3.2.

Det finns även en kolumn *Archive* där de arbetsuppgifter som av någon anledning inte gjorts lades. Detta kunde bero på tidsbrist, kravändringar från kunden eller då andra lösningar av uppgiften valdes. De taggar som används på korten är *Kundkrav*, *Milstolpe*, 1, 2, 3, 4 och 5. ”*Kundkrav*” i rött indikerade att kortet inte är en arbetsuppgift utan ett mål som de andra korten skulle utformas efter. ”*Milstolpe*” i grönt användes för att indikera en ny sprint, siffran i blått som finns på det indikerade vilken sprint de tillhörde.

3.5 Versionshantering

För versionshantering användes *Git* och *Github*. Det som skapas på Github är ett så kallat *Repository* där alla projektets filer samlas på ett gemensamt ställe i molnet. Detta görs för att lättare kunna hantera filer och automatiskt få möjligheten att gå tillbaka till en tidigare version i de fall då något nyimplementerat inte fungerar som tänkt. Genom att ha möjligheten att falla tillbaka till någon tidigare version av projektet gavs gruppen en viss säkerhet under utvecklingen.

Gruppen använde *repositorys* vilket gav möjligheten att skapa personliga grenar för utveckling parallellt med varandra för att undvika konflikter arbetet. När utvecklarna färdigställde sina arbetsuppgifter sammanslogs grenarna till en gemensam gren.

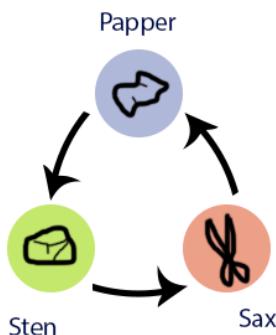
En stor del av alla konflikter kan automatiskt undvikas genom att i sin mapp ha en fil som specificerar filer, filtyper, och mappar som inte ska skickas med till Github. Dessa filer som inte skickas med är oftast personliga filer såsom inställningar till IDE och Unity. Dessa filer bör inte skickas med då de skulle orsaka konflikter varje gång.

Kapitel 4

System och tekniska lösningar

Systemet består av två HTC Vive med tillhörande kontroller kopplade till varsin dator och två basstationer för att spåra utrustningen i rummet. En basstation används för att spåra VR-utrustningen genom rummet. Spelet är gjort för två spelare som är tänkta att sitta vid ett bord mitt emot varandra. Då det är två Vive-enheter i samma utrymme leder det till att spelarna kan se varandra i VR på samma position som utanför spelet.

Spelet, namngivet *Evaari*, utvecklades till ett kortaktiviterat stridsspel där korten framkallas olika sorters figurer som strider för spelaren. Korten är indelade i olika klasser, dessa klasser har specifika egenskaper och förmågor som är starka eller svaga mot andra klasser. Tanken bakom var är att skapa en sorts sten-sax-påse balansering på spelplanen, se figur 4.1, för att få en mer strategisk spelupplevelse i realtid. Målet med spelet är att vinna mot sin motståndare genom att förstöra hens huvudbas.



Figur 4.1: Översikt av spellogiken av korten.

4.1 Grundläggande, initiala krav och systembegränsningar

Ett av huvudkraven för spelet var att utveckla som ett VR-spel som fungerar med *HMD*, i detta fall *HTC Vive*.

Eftersom spelet innehåller *VR* ställs det krav på antalet bilder per sekund, för att användaren ska få en bra spelupplevelse bör spelet kunna köras med mer än 90 bilder per sekund. Vidare kräver *HTC Vive* att *PC:n* som den används på är *VR ready*, vilket kräver ett modernt grafikkort. På mobiltelefonerna ställs inte samma krav då de inte är huvudmonterade på sina användare, det räcker att mobiltelefonerna klarar av att rendera grafiken i cirka 30 bilder per sekund [9].

Ytterligare krav på projektet var att spelets koncept skulle baseras på kortspelsaktivierad strid och att

det skulle spelas av minst två spelare. Utöver dessa krav fanns även kravet på att spelet skulle utgå från en sten-sax-påse-logik, det vill säga att alla kort i spelet ska vara starka mot ett slags kort och svaga mot ett annat. Det sista kravet som skulle tas hänsyn till är att striden mellan spelarna skulle ske i realtid.

4.2 Målplattform

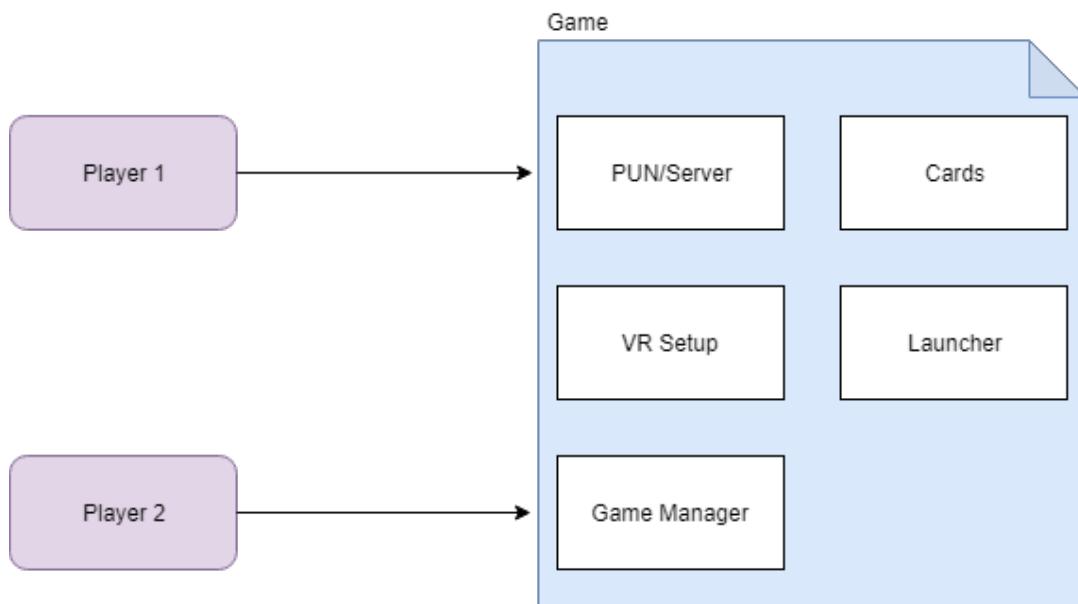
Målplattformen är *PC* med *Windows* samt *VR* hårdvaran *HTC Vive*. Utvecklingen sker även för *Android*-telefoner med stöd för *ARCore* [10]. Spelmotorn *Unity* kommer att användas för att utveckla och köra spelet.

4.3 Grundläggande system-arkitektur

Systemet består av två *HTC Vive* med tillhörande kontroller kopplade till varsin dator och två basstationer för att spåra utrustningen i rummet. En basstation används för att spåra *VR*-utrustningen genom rummet. Två spelare sitter vid ett bord mitt emot varandra. Spelarna ser varandra i *VR* på samma position som utanför spelet. Spelplanen matchar det verkliga bordet i storlek så att spelaren ska kunna luta armarna mot det.

Det skulle gå att köra ett sorts åskådarläge till spelet på en till två mobiltelefoner. Detta skulle senare kunna utvecklas till att mobilanvändarna också tar en aktiv del i spelet och kan spela ut kort.

Under utvecklingens gång användes mjukvaror varav den viktigaste var spelmotorn *Unity* som spelet är uppbyggt i. För att koda alla funktioner användes C# som programmeringsspråk. De modeller som användes i spelet modellerades samt riggades i *Blender*. För att skapa kortens design användes både *Adobe Photoshop* och *Adobe Illustrator*.



Figur 4.2: Översiktlig arkitektur av systemet.

Systemets arkitektur visas figur 4.2 i form av ett UML-diagram. Arkitekturen är indelad i ett antal olika komponenter som: spelarna, server, och själva spelet. Spelarna är användarna av systemet och använder sig av extern utrustning som *HTC Vive*. Servern är till för att koppla samman alla spelare i samma spelmiljö och spel-komponenten är själva spelet. För att koppla samman enheterna används

Photon Unity Networking (hädanefter *PUN*). Det är ett nätverksbibliotek som är väl integrerat med *Unity* och som är väl dokumenterat kring hur nätverksapplikationer skapas [11].

Spelkomponenten är även indelad i mindre moduler som: *Game Manager*, ljud, spelplan, *VR Setup*, och kort. *Game Manager* sköter spelets logik och att spelet håller samman, modulen samarbetar även med ljud-modulen som tar hand om alla ljud i spelet, *VR Setup* och spelplansmodulen. Spelplansmodulen sköter allt som har med spelets spelplan, exempelvis hur planen ser ut och vad för kort som kan modifiera den med mera. *VR Setup* är den del som har hand om spelarnas VR-utrustning och kortmodulen är den del som sköter spelkorten i spelet, med andra ord ingår kortens förmågor och tillhörande monster i denna modul.

4.4 Standarder

Spelet utvecklades i *Unity* där all kod skrevs i språket *C#*. *C#* användes då det är det mest optimerade programmeringsspråket i *Unity* och även det mest dokumenterade [12]. Samtliga utvecklare har tidigare erfarenhet av programmering i C-språk vilket gjorde *Unity* till ett väldigt bra val av spelmotor.

Som *API* för att enkelt kunna tolka rörelser och knapptryck från VR-utrustningen används *SteamVR* och *OpenVR*, som är utvecklat för just *HTC Vive* [13]. *SteamVR* finns även som tillägg till *Unity* vilket förenklar implementationen drastiskt.

4.5 Kvalitetssäkring

För att testa kod som skrivits för mobilapplikationen gjordes enbart systemtest. Av okänd anledning tillät *Unity* inte förhandsvisningar av applikationen direkt i gränssnittet via *USB*-kabel. Detta tvingade utvecklarna till att kringgå problemet för att utföra systemtest. För att kunna testa systemet behövde *Unity*-projektet byggas till en *apk*-fil som sedan laddades upp på *Google Drive*. Via mobiltelefonerna kunde sedan *apk*-filen installeras och därefter kördes applikationen direkt i telefonen vilket var en långsam process. För att kunna se konsolen med *debug*-meddelanden användes *Androids* egna *debug-tjänst logcat*.

För att garantera kodens kvalité användes parprogrammeringsprincipen där två utvecklare granskar koden medan den skrivs. Tester utfördes då uppgiften placerades i den kolumn i Trello som behandlade tester av arbetsuppgifter och befintlig kod.

Kapitel 5

Utvecklingsprocessen

I detta kapitel redovisas hur gruppen har arbetat under projektets gång. Gruppen har strukturerat arbetet utefter *Product Backlog* som beskrivs i 3.4 där följande rubriker utgjorde kort i *Sprint Backlog* som utvecklarna har arbetat utefter. Arbetsuppgifterna valdes i sådan ordning att en uppgift inte beror på en senare uppgift i utvecklingen, utan tvärtom. Det arbete som genomförs ska hjälpa senare uppgifter.

Utvecklingsprocessen av spelet var uppdelat i tre huvuddelar, *VR*, *AR* och nätverket mellan dem. Dessa delar arbetades mer eller mindre parallellt under hela projektets gång.

5.1 Spelet - VR

Projektets utveckling av *VR*-delen började med att skapa en grundscen i *Unity*. I scenen kunde en spelare se och interagera med objekt i en virtuell verklighet. Grundscenen utvecklades under projektets gång genom att gradvis lägga till funktioner och förändra utseendet i scenen.

5.1.1 Bordsplacering

Gruppen planerade för en funktion där användaren kunde placera ut det virtuella spelbordet för att matcha det fysiska bordet. Eftersom spelet är tänkt att användas i sittande position och vid ett bord skulle funktionen ge en haptisk återkoppling till användaren. Detta skulle kräva noggrann positionering av det virtuella bordet.

Det som först implementerades var att en användare kunde trycka på en av kontrollernas *hairtrigger* (en avtryckare på kontrollen), och sedan vänta en stund för att placera ut bordet på den platsen som kontrollen befann sig på. Funktionen fungerade, men efter undersökning av monster-målsökning på spelplanen upptäcktes det att bordsplacering i *runtime* komplicerade scenen. Målsökningen som monstren använder i spelet är en inbyggd funktion som kallas *NavMesh*. Den använder sig av en specifik yta. Denna yta kan inte flyttas i *runtime* vilket inte passade ihop med att bordet, som ytan var på, flyttades. När det fanns två spelare i scenen kunde det bli konflikter, eftersom användarna kan lägga ut bordet på olika ställen.

Gruppen bestämde sig att ta bort funktionen helt från spelet på grund av de komplikationer som uppstod. Det sparade mer tid att bara använda ett fast bord i scenen, och sedan instansiera spelarna på var sin sida av det virtuella bordet. Det virtuella bordet anpassades då efter måtten på ett rektangulärt bord.

5.1.2 Rutnät

Spelet använder sig av kort som i sin tur skapar monster som strider mot varandra. För att kunna placera ut korten behövdes det ett rutnät av något slag. Rutnätet gav användaren möjligheten att placera ut ett kort på ett specifikt ställe på sin halva av spelplanen. Hexagonerna som utgör rutnätet, är till för att ge en startposition för monstret. Det vill säga när monstret placeras följer den inte rutnätets struktur när den går mot sitt mål utan de går fritt längs spelplanen. Rutnätet genereras då spelscenen startas, och placeras sedan på rätt position på det virtuella bordet i scenen. I figur 5.1 visas hur spelets rutnät ser ut, i spelet fanns två olika rutnät, en för varje spelare.



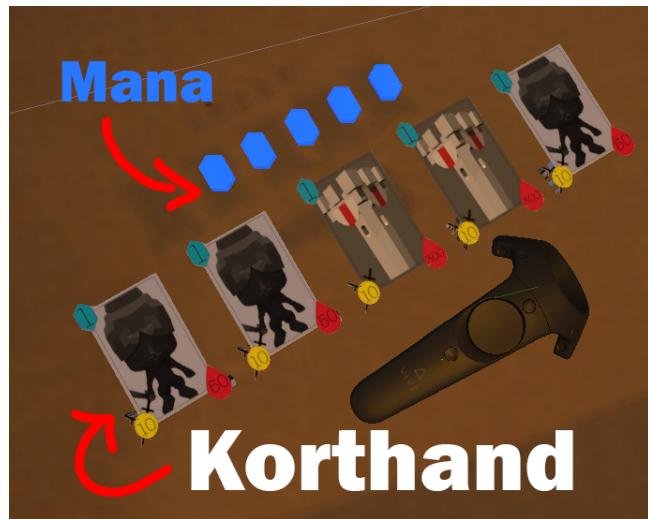
Figur 5.1: Spelplan med rutnät.

5.1.3 Speltimer - Mana

Ett av kundkraven i projektet är att en spelomgång ska spelas i realtid. Tidigt i utvecklingsprocessen fanns det ingen spärr på hur många kort en användare kunde lägga ut på spelplanen. Detta uppfyllde kravet för realsidsstrid, dock skapade detta problem med spelprestandan. Anledningen var att när för många objekt befann sig i scenen började spelprestandan att försämras och gjorde det näst intill omöjligt att spela vidare. Lösningen till problemet var att begränsa spelarens möjlighet att lägga ut kort. Begränsningen är skapad i form av en timer som räknar upp allt eftersom tiden går, och sänks när kort spelas ut. Timern är i form av fem olika *mana*-kristaller som lyser upp om de är tillgängliga, och när ett kort placeras på spelplanen släcker ett antal kristaller beroende på hur mycket kortet kostar. I figur 5.2 visas hur spelets timer ser ut, samt korthanden som spelaren har.

5.1.4 Kort

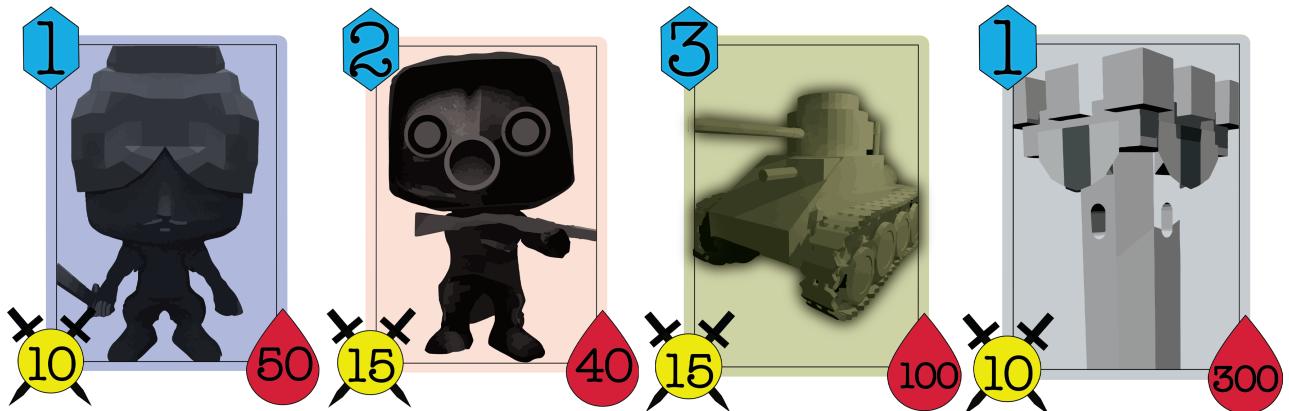
En viktig del av spelet är de fyra olika korten som spelaren kan spela ut. Korten representerar olika monster vars egenskaper visas i 5.1. Det är kortets textur som signalerar användaren vilket typ av kort det är, se figur 5.3. Räckvidd, hastighet och prioritet visas inte på korten, eftersom för mycket information på kortet skulle vara förvirrande och distraherande. Därför valdes bara tre attribut som skulle visas på kortet: liv, attack och manakostnad.



Figur 5.2: Spelarens korthand med speltimer (mana).

Klass	Färg	Monster	Attack	Liv (HP)	Manakostnad	Prioritet	Räckvidd	Hastighet
Papper	Blå	Figur med svärd	10	50	1	1	0.1	0.09
Sax	Röd	Figur med gevär	15	40	2	1	0.2	0.05
Sten	Grön	Stridsvagn	15	100	3	2	0.2	0.03
Torn	Grå	Torn	10	300	1	4	0.2	0

Tabell 5.1: Tabell över fördelningen av kortklasser.



Figur 5.3: Spelets olika kortklasser

I ett tidigt stadio av utvecklingen slumpsades egenskaperna fram för varje enskilt kort och texturen för kortet visade bara vilket typ av kort det var. Slumpningen av egenskaper gjorde det dock svårt att balansera kortleken rättvist mellan spelare, men gjorde det även svårt för utvecklarna att skapa möjligheten till spelaren att applicera olika spelstrategier. Därför fördefinierades egenskaperna hos korten i stället och sedan blandades korten i början av varje spelomgång.

I spelet genereras det 52 kort för varje spelare, 104 kort totalt, som innehåller två tornkort, tjugo papperkort, tjugo saxkort och tio stenkort. De två tornkorten finns alltid tillgängliga i början av en spelomgång och är bara tillgängliga en gång per spelomgång. De övriga korten är de spelkort som är kopplade till monster som kan förflytta sig över spelplanen mot sina mål. Spelaren har alltid fem kort på hand som hen kan välja mellan. När ett kort spelas ut skapas ett nytt utifrån de femtio kort (tornkorten kan bara komma upp en gång) som finns tillgängligt för den specifika spelaren.

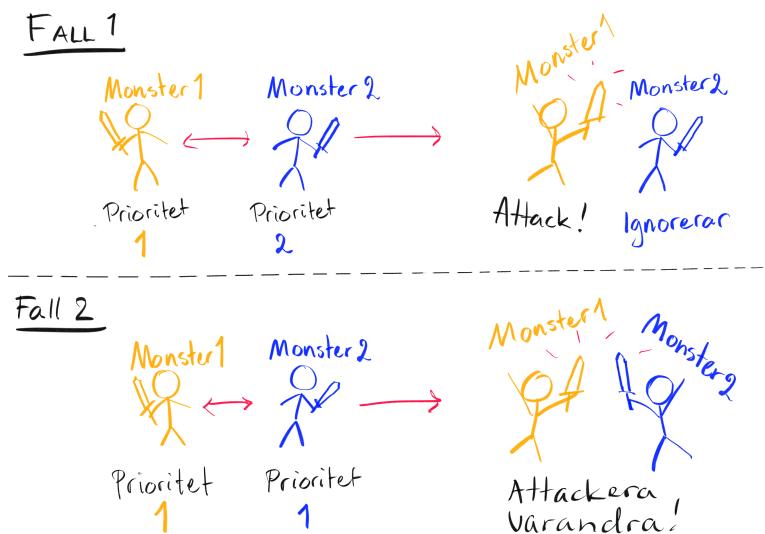
Korten är baserat på sten-sax-påse-modellen, vilket var ett kundkrav. Sten-sax-påse utgår från att ett

element är starkare mot ett annat element och svagare mot ett annat. I grunden handlar modellen om slump, men tillsammans med mänsklig faktor kan detta skapa olika strategier vilket var ett mål som projektet var ute efter. Dock är detta svårt att se i det som implementerades i spelet, eftersom det endast finns tre monsterkort i spelet, en av varje kategori, och monsternas prioritetsfunktion gör så att monster kan ignorera varandra. Stridsvagnen har mest liv och gör mest skada samt kostar flest *mana*-kristaller för att kompensera för det. Stridsvagnen kan endast skada byggnader. Svärdkortet kostar minst antal *mana*-kristaller och har minst attack samt femtio liv, detta eftersom den endast kan skada fiender som är precis intill den. Ett alternativ till implementation av sten-sax-påse modellen är att ge en bonusattack till monster när de attackerar svagare monster.

5.1.5 Monster

Till varje kort finns det ett monster kopplat som skapas när kortet placeras ut på spelplanen. Tornen beter sig som monster i fråga om förmågan att attackera, men kan inte förflytta sig. Övriga monster kan röra sig över spelplanen och attackera motståndarens bas och monster.

När ett monster skapas på spelplanen är dess första syfte att förflytta sig mot motståndarens bas. Ett monster kan avbryta förflyttningen om motståndarens monster är tillräckligt nära och har högre eller samma prioritet. I det fallet attackeras motståndarens monster istället. Dessa två fall illustreras i figur 5.4.



Figur 5.4: Monsters prioritet och handlingar i spelet.

5.1.6 NavMesh och NavMeshAgents

NavMesh och *NavMeshAgents* är två inbyggda *Unity*-funktioner som används i spelet för att styra monster på spelplanen. En *NavMesh* är en specifik yta på ett objekt som *NavMeshAgents* kan navigera på. *NavMesh* i en spelomgång är osynlig för användaren, men ligger som ett lager på spelplanen. I figur 5.5 visas (med ljusblå färg) hur den navigerbara ytan ser ut för spelplanen.

För att kunna använda *NavMesh*-funktionerna behövs det *NavMeshAgents*, som kan navigera på ytan. Därför har varje monster i spelet en *NavMeshAgents*-komponent kopplad till sig.

NavMeshen kräver att den genereras innan (eller i samband med att) *Unity*-scenen startar, eftersom den behöver finnas i scenen innan dess agenter. Om den inte fanns i början skulle agenterna ge felmeddelanden och spelet skulle krascha. Denna aspekt förhindrade spelet att flytta ytan under körtid, vilket



Figur 5.5: Bild på *NavMeshen* som användes i spelet.

inte passade ihop med bordplaceringsfunktionen. *NavMesh* och *NavMeshAgents* var dock ett smidigt alternativ för att skapa monsterbeteendet för målsökning. Andra aspekter som också var användbara var hastigheten och stoppavståndet för agenterna. Dessa egenskaper kunde kopplas till monsternas hastighet och räckvidd, vilket bidrog till en bättre balansering av kortattribut. Bordsfunktionen valdes bort, eftersom det var enklare för arbetsgruppen att använda *NavMeshen* med tillhörande agenter till monsterimplementationen.

5.1.7 Byggfas

I början av utvecklingsprocessen planerades det att spelet skulle ha en sorts byggfas. Under denna fas skulle användaren bygga upp spelbanan med olika objekt som broar och torn. Denna aspekt av spelet hade inte lika stor prioritet som andra funktioner av spelet. I slutprodukten representeras byggfasen bara av de två tornkorten som spelaren har tillgång till i början av en spelomgång.

Anledningen till att byggfasen bara blev en liten del av spelet hade att göra med hur kort placeras ut. Kortplaceringsfunktionen baseras på att korten är vanliga monster med specifika attribut, medan byggkort hade andra attribut. Skillnaden mellan korten komplicerade funktionen för att plocka upp kort, eftersom den behövde skilja på dem.

Om varje spelare skulle tillåtas placera sin egen bas behöver nätverket veta om både baserna är placerade innan spelet kan börja, detta var svårt att implementera. En möjlig lösning till byggfasen hade varit att dela upp spelet i två scener, en byggfas-scen och en spelscen. Men med den tidsram som projektet hade skalades byggfasen ner till bara två tornkort som i grunden bara var stationära versioner av monster.

5.1.8 Spelomgång och huvudmeny

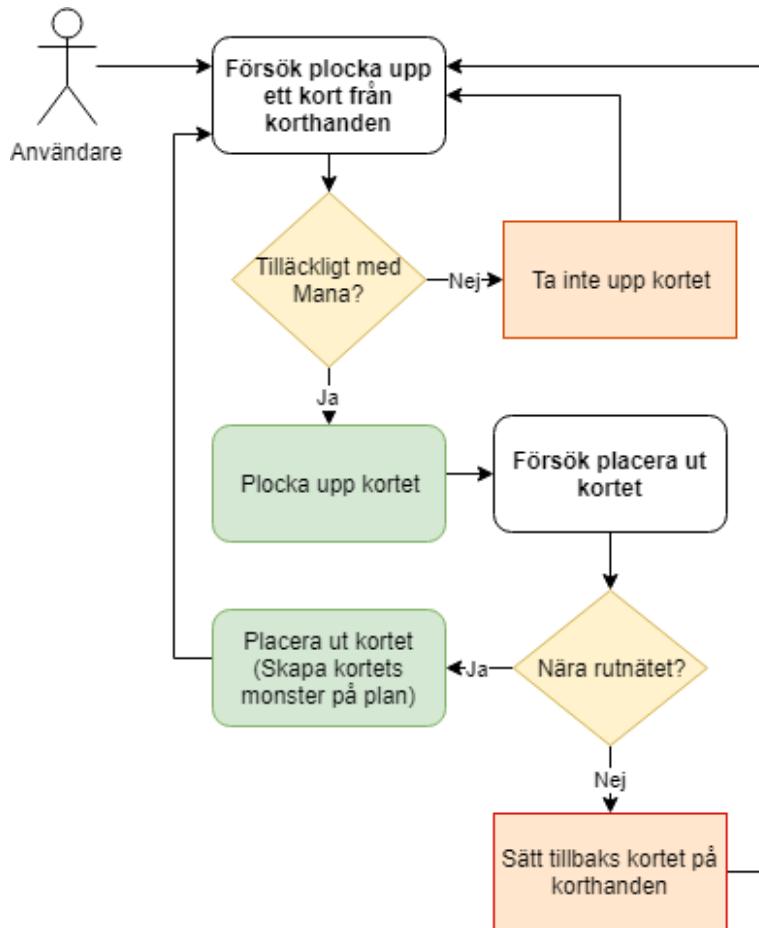
Spelet består av två olika *Unity*-scener, en huvudmeny och en spelscen. Huvudmenyn är till för att skicka spelaren till spelscenen, genom att skapa spelsessionen, eller att avsluta spelet. När användaren skickas till spelscenen påbörjas en spelomgång med en motståndare. I början av en spelomgång finns det två baser med 1000 liv var, en för varje spelare. Varje spelare har även varsin korthand med fem kort och en manatimer som är synliga för motspelaren. Spelarna kan på så sätt lokalisera varandra. Genom att hämta transformen för kontrollerna för varje *HMD* kan ett objekt placeras på exakt den plats och med samma rotation som kontrollerna besitter.

Användarna spelar en match genom att placera ut sina kort, och när en av baserna förstörs gömmer

spelet korten och visar istället en avslutningsknapp. Knappen är till för att för att skicka tillbaka användaren till huvudmenyn när användaren rör den.

5.1.9 Spela ut kort

Spelet behövde en funktion för att spela ut kort på spelplanen. Denna funktion behövde gå igenom ett antal steg innan användaren kan lägga ut ett kort, och i figur 5.6 visas hur stegen ser ut. Stegprocessen påbörjas när användaren trycker och håller in kontrollens *hairtrigger* vid ett kort.

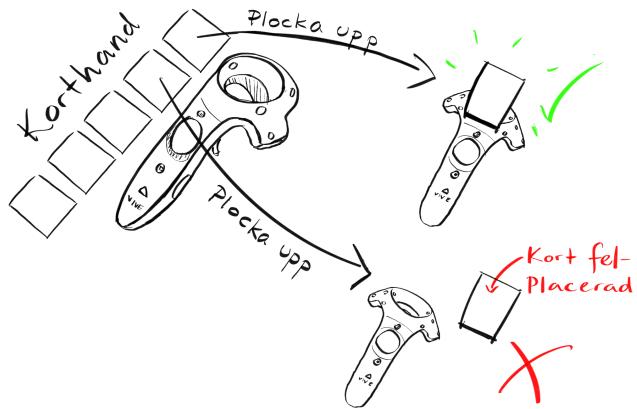


Figur 5.6: Aktivitetsdiagram över kortplockning.

När en spelare försöker ta upp ett kort från korthanden kontrollerar spelet om det finns tillräckligt med mana för att placera kortet. Om kortet inte kan plockas upp är kortet i fråga mer genomsiktig i jämförelse med de tillgängliga korten. Detta används som en visuell indikation till användaren att kortet inte är tillgängligt.

I situationen då spelaren plockar upp ett kort visas kortet i den andra handen, och försinner från korthanden. Denna funktion hade tre olika implementationer under utvecklingsprocessen. Det första försöket som gjordes var att försöka flytta det upplockade kortet från korthanden till den andra handen. Problemet med denna logik var att beroende på vilket kort användaren plockade upp blev den flyttad till olika positioner på den andra handen. I figur 5.7 illustreras problemet.

För att lösa positioneringen av det upplockade kortet ändrades metoden till att skapa ett nytt kort i den andra handen istället. Denna metod ändrade dock kortets egenskaper. Med andra ord, när användaren plockade upp ett kort från korthanden visades inte samma kort i andra handen. Detta problem berodde på vilken ordning kort fick sina egenskaper och att skriva över egenskaperna inte fungerade alla gånger. Detta ledde till att metoden ändrades en tredje och sista gång. Lösningen var att alltid ha ett



Figur 5.7: Resultatet av första försöket med att ta upp ett kort.

kort på handen, som hämtar det upplockade kortets egenskaper och sedan visar det till spelaren. I korthanden göms det upplockade kortet, och om kortet placeras på spelplanen hämtas nya egenskaper från kortlistan och sedan visas kortet igen. Med andra ord hade en spelare alltid sammanlagt sex kortobjekt, fem kort i korthanden och ett temporärt osynligt kort i andra handen som visas när spelaren tar upp ett kort.

Det spelet kontrollerar efter ett kort har plockats upp är om kortet är i närheten av rutnätet. Det finns en viss gräns för hur långt ifrån en hand-kontroll (kontrolpen som håller det upplockade kort) kunde vara från rutnätet. Gränsen är till för att ge en starkare känsla till användaren att kort läggs ner på spelplanen. Om det upplockade kortet är nära rutnätet och användaren släpper kontrollens *hairtrigger* skapas ett monster från kortet på spelplanen. Monstret börjar därefter attackera motståndarens bas eller monster.

5.1.10 Animationer och Ljud

För att göra spelet mer verklighetstroget och intressant skapades animationer till de olika spelkaraktärerna. Bortsett från stridsvagnen har varje spelkaraktär en rörelseanimation och en attackanimation. Stridsvagnens animation på banden syntes inte tillräckligt tydligt i VR och togs därför bort. Alla animationer skapades i *Blender* med hjälp av *keyframing* där start och slutpositionen på karaktärens ben bestäms för att sedan interpoleras mellan de två positionerna för att få en mjuk övergång. Genom att göra flera sådana interpolationer undviks en ryckig animation.

De ljud som finns i spelet är tre olika instrumentala låtar samt ljudeffekter till olika händelser i spelet, exempelvis låter det när en mana-kristall lyser upp. De instrumentala låtarna är bakgrundsmusik i spelet, en låt finns i huvudmenyn och de två andra i spelscenen. När spelscenen startas slumpas ett värde som bestämmer vilken av de två möjliga låtarna som ska spelas, för att ge omväxling i bakgrundsmusiken. De ljudklipp som används i spelet, som inte är gjorda i FLStudio, används enligt *creative commons*-licensen.

5.2 Figurer, landskap och byggnader

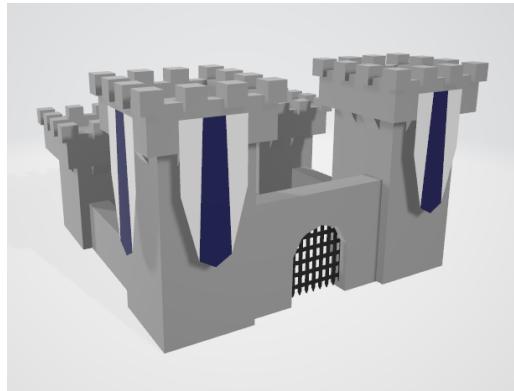
De 3D-modeller som används i spelet är skapade och animerade i *Blender*. Alla modeller är anpassade efter de krav som mobiltelefonerna ställer. Dessa krav är bland annat att varje objekt inte bör överskrida 5000 trianglar. Karaktärerna är skapade för att passa in i temat *steampunk*. Exempel på modeller visas i figurerna 5.8a och 5.8b. Samtliga modeller följer en lågpolygonstil, som bidrar till låg renderingskostnad.

Modellerna skapades först i högupplösta modeller med flera hundra tusen trianglar för att kunna få modellen detaljerad. För att nå projektets gräns med femtusen trianglar gjordes en *retopology* där detaljerna skalas ned och alla trianglar ritades om för att få önskad mängd. Till att börja med var tanken att göra en *retopology* och sedan använda sig av en normal-map men eftersom figurerna var så pass små samt att modellens normal-map inte fungerade helt som planerat förkastades den idén och enbart *retopology* användes.

Modellerna har också olika färgscheman för att signalera vilken spelare de tillhör, till exempel varierar färgerna på karaktärernas kläder och flaggorna på slotten.



(a) En av de stridande karaktärerna som finns att spela ut i spelet med tydligt *steampunk*-tema med *trenchcoat* och *gasmask*.

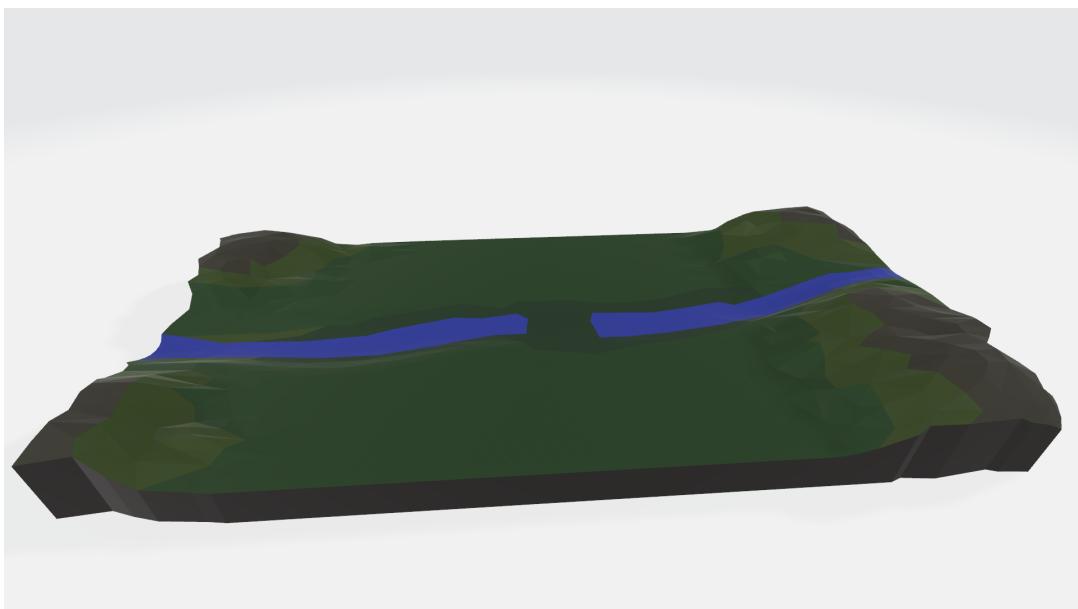


(b) Modell av ett slott som representerar en spelares bas.

Figur 5.8: Exempel på modeller och karaktärer som förekommer i spelet.

Landskapet som presenteras i figur 5.9 utformades med hänsyn till var spelfigurerna skulle röra sig. Utgångsläget var två plana ytor med en smal övergång mellan dem. Höjdskillnader (bergen och floden) är därav av estetiska skäl samt för att visa spelaren vilka startpositioner hen kan välja åt sina figurer. Färgskillnaderna i texturerna gör det extra tydligt var höjdskillnaderna är.

För att på bästa sätt kunna representera den fysiska miljön vid vilken användarna befinner sig i har det virtuella bordet modellerats efter det fysiska bord som testandet skett på.



Figur 5.9: Landskapet som används i spelet.

5.3 Nätverk

I följande del beskrivs de två olika metoderna som gruppen hade i åtanke när val av nätverks ramverk skulle göras.

5.3.1 Photon Unity Networking

Photon Unity Networking, eller *PUN*, är ett ramverk byggt för Unity som används för att integrera flerspelaraspекter i sitt spel. *PUN* har egna servrar som agerar värdar för alla spel som använder *PUN*. Utvecklarna behöver därför inte sätta upp egna servrar.

Spelet använder *PUN* för att skapa eller ansluta till ett rum i molnet som alla spelare ansluter till. Den första användarens spel skapar rummet när spelaren trycker på start i huvudmenyn, och den andra spelaren ansluter sig till första spelarens rum när spelet startas.

Det är genom rummet som all nätverkad information hanteras mellan spelarnas spelinstanser. För att synkronisera den informationen som har med spelobjektens rörelser eller ändringar på egenskaper måste de objekten instansieras för att synas hos alla spelare. Instansieringen innebär att de objekten nu är nätverkade och kan dela med sig av bland annat position och rotation. Instansieringen av objekt som ska finnas från start sker i *Game Manager*. Ett exempel på ett viktigt objekt att få nätverkat är korten, eftersom de styr vad för karaktär som ska dyka upp på spelplanen när ett kort placeras. När ett nytt kort skapas i korthanden måste de tilldelade egenskaperna hos kortet skickas till nätverket, för att den andra spelarens spelinstans också vet vad det är för kort.

5.3.2 Unity UNet

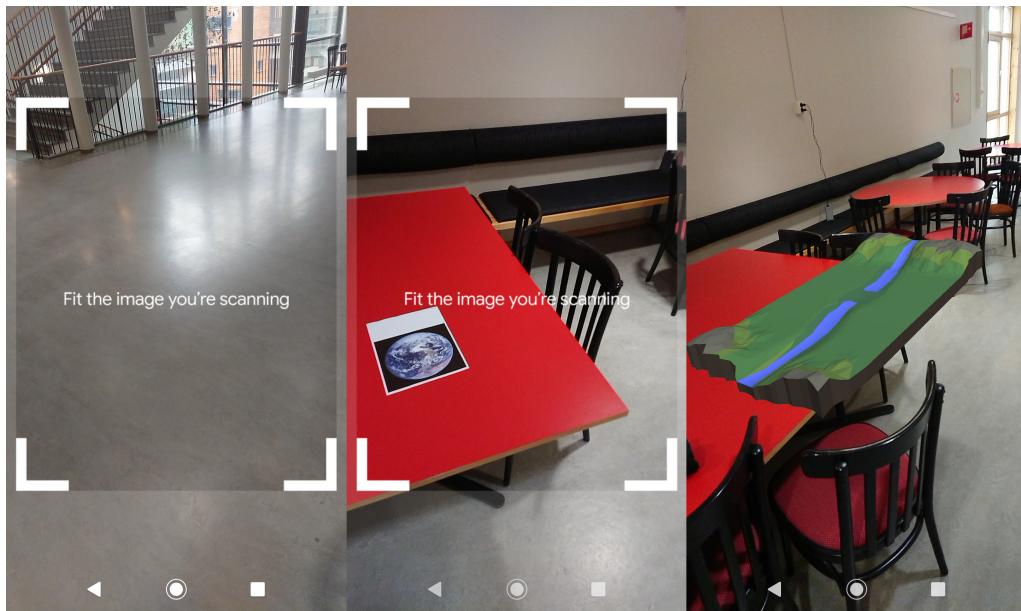
Unity har ett eget inbyggt system för att skapa flerspelarspel som heter *Unity UNet*, häданefter *UNet*. *UNet* valdes inte då det inom snar framtid kommer att förlora stöd då *Unity* i nuläget håller på att skapa ett nytt system för att skapa flerspelarspel [14, 15].

5.4 Augmented reality

För att låta ytterligare användare ta del av spelet gjordes flera försök att implementera en applikation vilken skulle låta användare åskåda spelet med hjälp av *AR*. Valet av *AR* skulle göra det möjligt att genom mobilkameran se miljön som spelarna befinner sig i, med ytterligare objekt som till exempel figurer, torn, och slott.

För att implementera *AR*-aspekten utgick utvecklingslaget ifrån ett exempel, *Augmented Images*, från *ARCores* paket till *Unity*. Här kunde en användare från början skanna en sedan tidigare känd bild och på dess position se ett 3D objekt. *ARCore* placerar ut ankare i samma tillfälle en fysisk bild igenkändes av kameran, ankaret behåller då en position i 3D rymden och relativt ankaret kan 3D objekt placeras. Över ankaret placerades en 3D modell av spelplanen. I figur 5.10 visar skärmdumpen till vänster och i mitten visar att bilden ännu inte är skannad korrekt medan skärmdumpen till höger visar att bilden skannats korrekt och därmed visar spelplanen. Ankarets transform (rotation, skalning och translation) sattes som förälder till resterande objekt i scenen. Detta innebär konceptuellt att världsorigo flyttas till ankarets position.

Kvar återstår att hitta diverse objekt från den virtuella scenen i *Unity* till *AR*-scen och visa dem på mobiltelefonerna. Här gjordes flertalet försök utan större framgångar.



Figur 5.10: Tre olika stadier i åskådarapplikation.

5.4.1 Försök 1

Här är scenerna uppdelade i *AR* respektive *VR*. Tanken i det här försöket var att hitta objekten i den virtuella scenen. Varje objekt behövde således ha egenskapen *DontDestroyOnLoad* eftersom den virtuella scenen laddas oavsett om användarplattformen. *AR*-scenen laddas sedan därför om användaren spelar på en androidtelefon, de objekt som då har *DontDestroyOnLoad* kommer inte att försvinna utan vara kvar i den nya scenen. Efter detta steg bestämdes att varje objekts transform var relativt ankarets transform. Problemet med det här tillvägagångssättet var att objektens transformer inte uppdaterades i *AR*-scenen då de transformeras i *VR*-scenen.

5.4.2 Försök 2

Här laddas scenerna additivt. För att detta skulle vara en framgångsrik metod måste *VR*-scenen placeras i *AR*-scenen så att de delar världsrymd. Istället för att visa objekt så behöver vissa objekt istället döljas om de inte ska vara synliga för åskådarna, till exempel avatarerna på spelarna och tältet som den virtuella världen utspelar sig inuti. Med det här tillvägagångssättet framstod ett liknande problem som i försök ett. Mobilanvändaren såg inte de transformationer som *VR*-spelaren orsakade på objekten.

5.4.3 Försök 3

Eftersom spelet använder sig av *API*:et *Photon*, ett applikationsprogrammeringsgränssnitt, för flerspelarläge och nätverk gjordes flera försök att införliva detta i de åskådbara objekten. Nätverkade objekt användes i kombination med de andra försöken. Ett nätverkat objekt kommunicerar med *Photon* och alla spelare som är kopplade till samma rum ska således kunna se objekten, oavsett vilken scen eller annan kontext de befinner sig i. Detta gjordes utan framgång.

5.4.4 Felsökning

I sektion 4.5 nämdes att *Unity* ska kunna användas för att testköra applikationer direkt i telefonen och samtidigt få varningar och felmeddelanden i konsolen på datorskärmen. Detta användes i tidigt

stadie, men slutade fungera efter en veckas utvecklande. Detta berodde troligen på ett internt fel hos *API*:et *ARCore*.

Under större delen av projektet felsöktes applikationen genom systemtester. Mot slutet av projektet gjordes felsökningar med hjälp av *logcat* i ett försök att öka effektiviteten.

Kapitel 6

Resultat

Projektet har använt sig av ett antal olika metoder för att skapa slutprodukten. Det ramverk som användes för att strukturera utvecklingsprocessen var *Scrum*. De aspekter som användes från *Scrum* var backloggar, sprintar och scrummöten. Backloggarna var produktbacklogg och sprintloggar som innehöll kundkraven för produkten, respektive uppgifter som skulle göras under en sprint. Kanban-tavlan *Trello* användes för att lägga upp backloggarna för arbetsgruppen och skapa en översiktlig vy på utvecklingsprocessens framsteg. De flesta sprintarna under projektet var två veckor långa och vid början av varje sprint samlades samtliga projektmedlemmar, för att diskutera den kommande sprintens uppgifter och den föregående sprintens framsteg.

De möten som projektgruppen hade var dagliga scrummöten som gav gruppmedlemmarna en idé på vad alla arbetade på. Andra möten som skedde under projektet var sprintplaneringsmöten, som användes till att planera sprintarnas uppgifter, och kundmöten. Kundmötena var då projektmedlemmarna hade möte med sin kund och diskuterade produkten och framstegen som hade gjorts. Kunden testade även prototyper av spelet om de var tillgängliga.

Den versionshantering som användes i projektet var genom Git och Github, med hjälp av olika grenar. Varje utvecklare i projektet hade en specifik gren som de kunde arbeta i, och dessa grenar sammnfogades vid senare tillfällen till huvudgrenen i projektet när funktioner i spelet var funktionsdugliga. Kvalitetssäkringen av produkten var i form av parprogrammering och systemtester av funktioner efter de hade implementerats.

Utvecklingsverktyg som användes i projektet var främst spelutvecklingsverktyget Unity för att skapa själva spelet, tillsammans med Microsoft Visual Studio som IDE. Andra verktyg som användes var: FLStudio för musik, Blender för 3D-modeller och Adobe Photoshop/Illustrator för 2D-grafik som texturer.

Samtliga utvecklare har bidragit till hela systemet trots att arbetet har skett på olika fronter, och därmed bidragit till god förståelse och delaktighet i produkten. Detta är resultatet av god kommunikation inom gruppen och att utvecklarna ofta arbetat på projektet i varandras närväro. Vidare har samtliga utvecklare ökat sin förståelse för både spelutveckling samt AR- och VR-tekniker.

Till AR-delen i spelet användes API:n *ARCore* för att visa delar av VR-världen genom en mobiltelefon. Att synkronisera två *HMD* med mobiltelefoner ska, enligt teorin, fungera genom att antingen göra ankaret i AR-scenen till förälder till alla övriga objekt i scenen eller att additivt ladda VR- eller AR-scenen ”ovanpå” varandra. Detta beskrivs mer utförligt i avsnitt 5.4. På grund av andra faktorer blev åskådarläget aldrig slutfört och hela funktionen valdes att inte ingå i det slutgiltiga systemet.

Vidare har det virtuella bordet modellerats för att passa det fysiska bordet vid vilket spelet ska spelas. Andra bord än detta kan medföra att den virtuella världen inte längre representerar den verkliga. Det gjordes försök att kunna ändra det virtuella bordet efter ett godtyckligt fysiskt bord vilket beskrivs i

sektion 5.1.1. Vid de rätta förhållandena kommer användaren att känna det fysiska bordet då den rör det virtuella. Motspelarens kortlek är synliga för användaren för att undvika eventuella kollisioner.

6.1 Slutprodukten

Slutprodukten är ett VR-spel med ett flerspelarläge som använder PUN, där två spelare möter varandra. VR-delen av spelet använder sig av API:erna SteamVR och *OpenVR*. Både spelarna kan i realtid spela ut flera olika kort för att försöka besegra sin motståndare. I spelet finns animerade figurer och flera modeller som exempelvis monster och byggnader. Figurerna är löst baserade på sten, sax, eller påse. I spelet finns det begränsningar i form av manatimer och rutnät för var spelare, för att matcherna ska vara rättvisa.

6.2 Krav

Projektet hade ett antal krav som skulle uppfyllas, och en del krav blev uppfyllda och en del inte. De krav som blev uppfyllda var främst kravet att spelet skulle vara ett flerspelarspel, vilket spelets grund baserades på. Flerspelaraspekten syns när två olika användare startar spelet från huvudmenyn och skickas till spelscenen. De två spelarna placeras på varsin sida av bordet och visar för varandra att de har en korthand och manatimer. När spelare placerar ut sina kort visas monsterna med olika färger beroende på vilken spelare det är. Ett annat krav i samband med spellogiken var att monster skulle skapas från kort när spelaren placerade ut dem.

Två andra krav som också uppfylldes var realtidsstrid och sten-sax-påse-balansering av kort. Realtidsstriden i spelet uppfylldes genom att spelare kan lägga ut kort på spelplanen, utan att behöva vänta på sin tur. Dock är utläggningen av kort begränsad av *manatimern* i spelet, för att främja strategitänkande. Balansen av korten gjordes i form av tre olika kortklasser: sten, sax och påse. Varje klass implementerades med specifika egenskaper som hade för- och nackdelar beroende på situation, för att bidra till olika spelstrategier. Exempelvis, ett stenkort har mer liv än sax och papper och lika mycket attack som sax, men är längsammare än de andra monsterna och kostar mer mana att lägga ut.

Ett krav som uppfylldes delvist var kravet om en byggfas i spelet eller fördefinierade spelbanor. Det som implementerades var en byggfas, som var i form av två tornkort som spelare kan lägga ut i början av en spelomgång. Byggfasen blev dock inte lika omfattande som planerat, utan blev bara en enklare version av den.

Det krav som inte uppfylldes i projektet var kravet att spelet skulle kunna kopplas ihop med mobiltelefoner, för att skapa ett sorts åskådarläge in i VR-världen med hjälp av *AR*.

Kapitel 7

Analys och diskussion

7.1 Metod

I detta avsnitt diskuteras projektets arbetsmetoder och hur väl de har fungerat.

7.1.1 Projekthantering

Projektets tillämpning av *Scrum* var i slutändan inte helt enligt ramverket. Sprintåterblickarna föll ur fokus under projektets gång. Efter varje sprint påbörjades nästa utan någon större diskussion kring den nyligen avslutade sprinten. Hade detta gjorts hade alla utvecklare fått en bättre förståelse kring de olika komponenterna, och kunna ge förslag på förbättringar.

Ett bristande moment under testerna var dokumentationen. Det hade varit önskvärt att samla information om vem eller vilka som utförde testet. Ett sätt att göra detta bättre vore att ta med utvecklare som i vanliga fall inte var involverade i den delen av spelet som funktionen behandlade och låta denne sitta med och kommentera all kod. Dessa kommentarer skulle då vara självbärande och skrivna på så vis att den person som ursprungligen skrivit koden inte är den enda som förstår den. Om den ursprungliga personen av någon anledning är borta kan andra utvecklare enkelt plocka upp dess arbete.

Att skriva en detaljerad tidsplan i början av projektet underlättade för att alla utvecklare ska veta vad de väntas göra och när. Det möjliggjorde en snabbare administration under projektets gång. Utvecklarna behövde på så vis inte planera om projektet i början av varje sprint. Tidsplanen gör det även tydligt ifall om utvecklarna ligger i fas, framför eller bakom planeringen. Det är mycket svårt att innan projektets start göra en detaljerad tidsplan med alla arbetsuppgifter för en lång utvecklingstid. I detta projekt har tidsplanen lyckats hållas mycket väl. Det fanns dock brister i den, huvudsakligen flerspelarläget och AR-delen där det under många sprintar står ”fortsatt utveckling”. Detta har visserligen gjort att tidsplanen gått att följa men det gav inte mycket information till utvecklarna om exakt vad de var tänkta att göra eller testa under den sprinten. ”Fortsatt implementering av flerspelarläget” skulle exempelvis kunna betyda både ”lägg in flerspelarläge i spelet” eller ”försök få två spelare att se varandra i en testscen”.

7.1.2 System och tekniska lösningar

ARCore utgjorde hinder då dokumentationen är bristfällig och otydlig. Vidare är *ARCore* relativt nytt och inte lika etablerat som till exempel *Vuforia*, detta leder till att det är svårare att hitta liknande arbeten. Att *ARCore* ändå användes var ett krav från kunden.

De andra *API*:erna som används har levererat de tekniska lösningar som systemet krävt. *PUN* har bra dokumentation och det finns gott om liknande arbeten att jämföra med. Med hjälp av en bra genomgång med exempel på *Photons* egna hemsida gick det relativt snabbt att förstå hur det fungerade. En sämre sak var att många funktioner i nätverkandet var autonoma, vilket gjorde att när avancerade lösningar behövdes var det svårt att förstå hur det skulle implementeras.

Vidare representeras den fysiska världen väl av den virtuella. Vid spelande känner användarna att det fysiska bordets dimensioner och positionering stämmer överens med det virtuella.

7.1.3 Utvecklingsprocessen

Under utvecklingen av *AR*-delen fokuserade gruppen minimalt på hur felsökningen genomfördes. Mot slutet upptäcktes ett fungerande sätt att felsöka med hjälp av *logcat*. Om detta hade upptäckts tidigare hade slutprodukten hunnit utvecklats mer effektivt. Även fast *logcat* är ett bra verktyg, var det svårt för utvecklarna att veta exakt hur verktyget kunde användas på bästa sätt. Om ett annat *API* används hade kanske det mer önskvärda alternativet med direkta förhandsvisningar i *Unity* fungerat. Detta ämnet togs även upp i sektion 5.4.4.

7.1.4 Källkritik

Eftersom projektet är byggt med ny teknik finns få bra källor som är granskade av andra utvecklare. Detta gör att utvecklare är tvungna att vända sig till källor *online* i form av foruminlägg och mindre artiklar dessa har granskats kritiskt innan de bedömts som pålitliga eller inte. Vad gäller underlag som handlar om *AR* och att kunna åskåda ett pågående *VR*-spel hittades få bra källor. En av de bättre källorna var ett inlägg skrivit av Alan Staffords [5]. Inlägget beskriver kortfattat att det är möjligt att åskåda en pågående spelsession genom *AR*, ändemot är inlägget mer fokuserat på att synliggöra mobiltelefonerna som avatarer för spelarna i den virtuella världen. Alan Stafford har inte heller några kontaktuppgifter.

7.2 Resultat

Något som sticker ut i resultatet är att åskådarläget inte ingår i slutsystemet. Ett antal faktorer som till exempel bristfällig dokumentation och felsökningsmetoder gjorde att utvecklingen var ineffektiv och därmed fanns det inte tillräckligt med tid att utveckla färdigt spelet.

Vidare fanns det höga ambitioner gällande spelet, det fanns till exempel önskemål att kunna välja mellan flera olika spelplaner och ännu fler kort att kunna spela ut. Än en gång fanns det inte tillräckligt med tid för att hinna med dessa ytterligare moment.

Den modifierade modellen av sten-sax-påse tillsammans med de blandade kortlekarna gör att spelare behöver både en bra strategi och lite tur för att ha ett övertag på sin motståndare.

7.3 Arbetet i ett vidare sammanhang

Det förs ständiga debatter om våldsamma digitala spel påverkar användare negativt. En del studier hävdar att speciellt barn tar efter spelen och själva agerar våldsamt samt blir mindre empatiska, men andra studier säger att detta inte stämmer [16] [17]. Eftersom spelet kan uppfattas som våldsamt i och med att det går ut på att besegra sin motståndare genom att förstöra dennes bas bör det diskuteras om

spelet ska ha en åldersgräns. I högre grad än andra spel kan VR-elementen i spelet ge en vekligare känsla och därmed kan väldet uppfattas som mer påtagligt. Ett sätt att reducera detta är att använda en lågpolygonstil och miniatyrer där tanken inte är att återge fotorealism och realism.

Spelarna blir representerade genom avatarer för sin motståndare. Det blir då viktigt att inte könsdiskriminera. Det vore då bra om spelarna själva får välja mellan flera olika avatarer som visar på olika fysiska attribut, alternativt används en könsneutral avatar för att representera alla spelare. Ovanstående gäller även för etnisk diskriminering. Olika spelare med olika etniska bakgrunder ska känna sig representerade om valbara avatarer används. Ett sätt att helt undvika ovannämnda former av diskriminering blir att använda icke-mänskliga avatarer. I det här spelet valdes att använda en kub för att representera alla spelare. Illustrationerna på korten är även gjorda med alla dessa tankar i åtanke.

Spelet spelas även sittandes för att tillåta en så stor mängd användare som möjligt att spela spelet oavsett om de har någon form av funktionsnedsättning. Exempelvis så ska rullstolsburna kunna spela spelet. Risken för att användare skadar sig då de använder en *HMD* i ett fysisk rum minskar också då de är sittande.

Kapitel 8

Slutsatser

Projektets syfte var att undersöka hur *VR*, *AR* och agil utveckling kan användas i spelutveckling. Slutprodukten som skapades var ett flerspelarspel inom genren kortspelsaktiviteter strid, samt utspelar sig i en virtuell värld och spelas med en *HMD* och tillhörande kontroller. Projektstrukturen baserades på agil utveckling och ramverket *scrum*, för att steg för steg skapa spelet *Evaari*.

8.1 Svar på frågeställningar

En av frågeställningarna som projektet hade var hur *VR* och *AR* kan kopplas samman med hjälp av en extra plattform, i detta fall mobiltelefoner, för att kunna fördjupa spelupplevelsen. Det som utvecklades under projektet var en sorts åskådarfunktion för mobiltelefoner i *AR*, dock kunde funktionen inte fullbordas i tid för att inkluderas i slutprodukten. Denna funktion löstes bara teoretiskt och inte i praktiken. Praktiskt var problemet svårloöst, dels på grund av bristande dokumentation och svårigheten kring felsökning. Felsökningen av *AR*-delen förhindrades till stor del av att spelet behövdes kompileras till en applikation innan mobiltelefonen kunde köra den. Problematiken med *AR*-delen var tillräckligt stor för att inte lösas inom projektets tidsram. Inga *AR*-element ingår i det slutgiltiga systemet, detta på grund av en ineffektiv arbetsmetod som dels beror på valet av *API*.

En annan frågeställning handlade om hur en slumpbaserad modell tillsammans med människans valmöjligheter kan bidra till olika spelstrategier när spelet körs i realtid. Den modell som valdes var sten-sax-påse utifrån kundens krav, som i spelet återspeglas i de olika kortklasserna. I spelet finns tre olika kortklasser som har olika egenskaper som liv, attack, manakostnad med mera. För att balansera de olika korten ytterligare, togs beslutet att hålla egenskaperna inom ett visst intervall. Vidare gäller att om en kortklass är bättre än de andra klasserna i en egenskapskategori så är klassen i fråga sämre i en annan kategori. Exempelvis är papperklassen snabbast av de tre klasserna men gör minst skada av de tre. Dessutom blandas kortleken i början av en spelomgång för att ta in slumpspekten av modellen och valmöjligheten är i form av de fem korten som spelaren kan välja mellan under spelomgången. Tillsammans kan användare tillämpa olika spelstrategier med både korten och placeringen av dem.

Den sista frågeställning angående hur synkronisering mellan den verkliga och virtuella världen kan utföras för att ge användaren en god förståelse kring var bord och motspelare har för position. Genom att visa användarens kort och manatimer för motspelaren ger det användaren en förståelse om var motspelaren är i både den verkliga och virtuella världen då de riktiga världskoordinater används i båda fallen. Att placera ut bordet med hjälp av handkontroller hade varit en bra lösning då det ger en korrekt position till bordet då den verkliga och virtuella världen delar koordinater. Då de komplikationer som nämns i [5.1.1](#), finns inte funktionen i nuvarande projekt. Principen är god dock, då det skulle ge användaren en bättre förståelse kring var bordet är placerat.

8.2 Slutsats

Utvecklingen av spelet har både varit lärorik och utmanande, men det finns möjlighet att utveckla vidare på produkten. En aspekt av spelet som kan vidareutvecklas är främst *AR*-delen med åskådarläget. Utvecklingen kan alltingen fortsätta med *ARCore* genom att lösa problemet med felsökningen och skillnaden i positioner eller byta till en annan *AR*-API, exempelvis *Vuforia*. I fråga om *VR*-delen av spelet kan flera spelbanor som spelaren kan välja mellan implementeras. Fler kort skulle kunna implementeras för att ge en större variation med mer möjlighet att tillämpa fler olika spelstrategier.

Litteraturförteckning

- [1] Patrick Honner. Why winning in rock-paper-scissors (and in life) isn't everything. <https://www.quantamagazine.org/the-game-theory-math-behind-rock-paper-scissors-20180402/>, Apr 2018. Hämtad : 2019-06-03.
- [2] Thomas P. Caudell & David W. Mizel. Augmented reality: An application of heads-up display technology to manual manufacturing processes. https://www.researchgate.net/publication/3510119_Augmented_reality_An_application_of_heads-up_display_technology_to_manual_manufacturing_processes, Feb 1992. Hämtad : 2019-05-30.
- [3] Joe Bardi. What is virtual reality? <https://www.marxentlabs.com/what-is-virtual-reality/>, Mar 2019. Hämtad : 2019-05-03.
- [4] Kerry Maxwell. Augmented reality. <https://www.macmillandictionary.com/buzzword/entries/augmented-reality.html>, Apr 2010. Hämtad : 2019-04-30.
- [5] Alan Stafford. Cross world portals: Using a smartphone to peer inside vr worlds. https://medium.com/@alan_stafford/cross-world-portals-using-a-smartphone-to-peer-inside-vr-worlds-b606d23fbclid=IwAR1bQtZCmyP4xqresLyu9rAjSdTm7b1YmDWYgLlU2_urwpKFGe88cebxA, Aug 2018. Hämtad : 2019-04-29.
- [6] Clash Royale. Clash royale. <https://clashroyale.com/>. Hämtad : 2019-05-02.
- [7] Michele Sliger. Agile project management with scrum. <https://www.pmi.org/learning/library/agile-project-management-scrum-6269>, Okt 2011. Hämtad : 2019-02-18.
- [8] Trello. Trello. <https://trello.com/en>. Hämtad : 2019-04-02.
- [9] IRIS VR. Importance of framerates. <https://help.irisvr.com/hc/en-us/articles/215884547-The-Importance-of-Frame-Rates>. Hämtad : 2019-02-26.
- [10] Google. Arcore. <https://developers.google.com/ar/>, Feb 2019. Hämtad : 2019-02-18.
- [11] Photon. Photon unity networking, pun. <https://www.photonengine.com/en-US/PUN>. Hämtad : 2019-02-25.
- [12] Unity. Unity documentation. <https://docs.unity3d.com/Manual/>. Hämtad : 2019-02-25.

- [13] Steam. Steamvr. <https://steamcommunity.com/>. Hämtad : 2019-02-27.
- [14] Evolving multiplayer games beyond unet. https://blogs.unity3d.com/2018/08/02/evolving-multiplayer-games-beyond-unet/?_ga=2.187045480.1718164529.1559204142-1078126610.1550839936. Hämtad: 2019-05-30.
- [15] Unet deprecation faq. <https://support.unity3d.com/hc/en-us/articles/360001252086-UNet-Degradation-FAQ>. Hämtad: 2019-05-30.
- [16] Malena Ivarsson. Unga pojkar som spelar våldsamma dataspel vänjer sig vid våldet. <https://www.su.se/om-oss/press-media-nyheter/pressmeddelande-arkiv/unga-pojkar-som-spelar-valdsamma-dataspel-vanjer-sig-vid-vadet-1.135381>, Maj 2013. Hämtad : 2019-03-05.
- [17] Peter Örn. Inte mindre empatisk av våldsamma datorspel. <http://psykologtidningen.se/2017/03/09/inte-mindre-empatisk-av-valdsamma-datorspel/>, Mar 2017. Hämtad : 2019-03-05.

Bilaga A

Tidsplan

TNM094 Kandidatarbete MT

PROJEKTTITEL	Kontaktperson/levererad Strid i VR		KONTAKTPERSON	Anton C. Tärnholm
	PROJEKTGRUPP	UPPGIFT		
M, T, O innebär vad vi arbetat dag	Grupp E			
1 Projektplanering och en första prototyp				
1.1 Projektplan	SAMTLIGA	2/25/19	VECKA 9	3/13/19
1.1.1 Anpassning av projektplan	SAMTLIGA	2/25/19	3/13/19	100%
1.2 Utvecklingsverksamhet	Tobias G	2/25/19	2/27/19	3
1.3 Skapa spelkarta i VR	Annele W	2/25/19	2/27/19	3
1.4 Skapa funktion för att släcka i borden	Annele W / Tobias G	3/4/19	3/6/19	80%
1.5 Undersök tracking för flera spelare i VR	Tobias G / Tobias O	2/25/19	3/6/19	60%
1.5.1 Logga ihop olika 3D modeller	Tobias G / Tobias O / Jonas L	3/4/19	3/13/19	6
1.6 Skapa en grönhet för spelplanen baserat av kort och kartor	Tobias R	3/11/19	3/13/19	3
1.7 Färdigställ spelpart för loppkort	Adrian CT	3/4/19	3/6/19	3
1.8 Undersökning av AR-Cave	Adrian A / Gustaf W	2/25/19	3/6/19	100%
1.8.1 Undersök möjligheten att placera verktyg för spelplanen via kameras kameran	Adrian A / Gustaf W	3/11/19	3/13/19	3
1.9 Skapa en individuell arbete	Tobias R	2/25/19	3/13/19	9
2 Förberedelser inför tredje sprinten				
2.1 Förberedelse för att skapa en synca scener mellan de två spelarna.	Tobias O / Tobias R	3/18/19	3/27/19	6
2.2 Skapa gränssnitt för spelplanen baserat på kort och kartor	Tobias R	3/18/19	3/27/19	6
2.3 Förberedelse för att göra spelplanen	Tobias R	3/25/19	3/27/19	3
2.4 Implementera ett torrt spelkort som sedan kommer användas vid spelplanens tillverkning	Annele W / Adrian CT	3/18/19	3/19/19	2
2.4.1 Implementera spelplanen	Annele W / Adrian CT	3/20/19	3/26/19	3
2.5 Implementering av en spelfil som baserar på spelplanen och nor sig	Annele W / Adrian CT	3/18/19	3/27/19	6
2.6 Implementering av spelplanen i 3D	Adrian A / Gustaf W	3/18/19	3/27/19	6
2.7 Tillsätt individuell arbete	SAMTLIGA	3/18/19	3/23/19	5
3 Färdigställande av tredje spelhälften				
3.1 Implementera funktionellitet för att kunna placera olika objekt	Annele W / Adrian CT	4/1/19	4/3/19	3
3.1.1 Implementera så att figuren är synlig för alla utöver de sittande på andra figure	Annele W / Adrian CT	4/9/19	4/9/19	1
3.1.2 Implementera så att figuren skjutas vid varje klick på den	Annele W / Adrian CT	4/9/19	4/10/19	2
3.2 Skapa funktion för fyra från varje kategori (lägg pris)	Tobias R	4/1/19	4/3/19	3
3.3 Försett skepnade av 3D figurer	Tobias R	4/1/19	4/10/19	6
3.4 Försett skepnade av kort	Annele W	4/1/19	4/10/19	6
3.5 Informering om kompletterad projektplan	SAMTLIGA	4/9/19	4/9/19	1
3.6 Informering om aktuella för spelplanen på mobiltelefonerna	Adrian A / Gustaf W	4/1/19	4/9/19	5
3.7 Informering om aktuella för spelplanen på mobiltelefonerna	Tobias O / Tobias R	4/1/19	4/10/19	6
3.8 Publik skrining av projektrapport	SAMTLIGA	4/9/19	4/10/19	2
3.9 Användarfester	Adrian A / Gustaf W / Tobias R	4/10/19	4/10/19	1
4 Färdigställande av ett fängende spel				
4.1 Avslutning med Daniel Jönsson	SAMTLIGA	4/16/19	4/17/19	2
4.2 Informering om att spelplanen har hämtats som sätts ut av spelplanen, tillverkning och leverans	Annele W	4/15/19	4/15/19	1
4.2.1 Implementering av funktionellitet för de första körten, tre från varje kategori	Annele W / Adrian CT	4/16/19	4/24/19	5
4.3 Försett skepnade av 3D figurer	Tobias R	4/15/19	4/24/19	6
4.4 Implementera funktionellitet för att placera olika objekt i spelplanen	Tobias R / Tobias O / Gustaf W	4/15/19	4/24/19	6
4.5 Skapa funktion för att placera kort, teknik, saker och annat	Jonas L	4/15/19	4/24/19	6
4.6 Försett skepnade av ljud, musik	Annele W	4/15/19	4/24/19	6
4.7 Försett skrining av projektrapport	SAMTLIGA	4/15/19	4/24/19	6
5 Fabriken visuellt och förståendig av spellogik				
5.1 Försett skepnade av arbets till kort	Tobias R	5/1/19	5/8/19	4
5.2 Försett implementation och skepnare av 3D figurer	Adrian A / Gustaf W / Tobias R	4/29/19	5/8/19	6
5.3 Implementation av 3D modeller till spelplanen	Annele W / Adrian CT	4/29/19	4/30/19	2
5.4 Implementera byggsatsen innan spellet börjar	Annele W / Adrian CT	5/1/19	5/1/19	1
5.4.1 Implementera byggsatsen	Annele W / Adrian CT	5/6/19	5/8/19	3
5.7 Försett implementation av närläck	Tobias G	4/29/19	5/8/19	6
5.8 Försettställande av speldrag till alla kort	Adrian CT	4/29/19	5/8/19	6
5.9 Försett skepnade av ljud	Annele W	4/29/19	5/8/19	6
5.10 Försett skrining av projektrapport	SAMTLIGA	4/29/19	5/8/19	6
6 Försettställande av spelkort och 3D objekt				
6.1 Försett skrining av projektrapport	SAMTLIGA	5/13/19	5/14/19	2
6.1.1 Informering om projektet är klar	SAMTLIGA	5/15/19	5/15/19	1
6.2 Skapa ett gränssnitt för handslängning	Tobias R	5/13/19	5/14/19	2
6.2.1 Implementera handslängning	Annele W / Tobias R	5/13/19	5/16/19	3
6.3 Implementera effekter	Annele W / Jonas L	5/15/19	5/27/19	2
6.4 Användarfester	Tobias G	5/13/19	5/22/19	6
6.5 Försett implementation av närläck	Tobias G	5/13/19	5/22/19	6
6.6 Försett implementation av kort	Annele W	5/13/19	5/22/19	6
6.7 Försett implementation av tidigare funktionellitet	Adrian A / Gustaf W / Tobias R / Jonas L	5/20/19	5/22/19	3
6.8 Försett implementation och skepnare av 3D figurer	Adrian A / Gustaf W / Tobias R / Jonas L	5/13/19	5/22/19	6
6.9 Skrining av opponenterrapport	SAMTLIGA	5/20/19	5/22/19	3
6.9.1 Informering opponenterrapport	SAMTLIGA	5/22/19	5/22/19	1
6.10 Försett skepnade och tillverkning	Annele W	5/13/19	5/22/19	6
7 Leverering av slutprodukt				
7.1 Skrining av kompletterad till produktopskrift	SAMTLIGA	5/27/19	6/2/19	4
7.2 Förberedelse inför leverering projektrapport	SAMTLIGA	6/4/19	6/4/19	1
7.3 Förberedelse inför sluttillmötet	SAMTLIGA	5/28/19	6/4/19	3
7.4 Slutsummanum	SAMTLIGA	6/5/19	6/5/19	1
7.5 Tillsätt individuell arbete	SAMTLIGA	5/27/19	6/5/19	5

Uppgifts ankringar behöver inte genomsökas uppiffran utan ankringar endast för att den här faktiskt

Bilaga B

Ansvarsområden

Tabell B.1: Projektets medlemmar med ansvarsområden

Namn	Ansvarsområde
Adrian Andersson	Scrummästare & AR
Anneli Nilsson	VR & Ljud
Anton Cevey Tärnholm	Produktägare & Spellogik
Gustaf Wallström	Dokumentationsansvarig & Modellering
Jonas Långdahl	Nätverk
Tobias Olsson	Versionshantering & Nätverk
Tobias Ryttinger	Modellering & Animering

B.1 Summering av enskilt arbete

- Adrian Andersson: Huvudansvar för utvecklingen av AR.
- Anneli Nilsson: Huvudansvar för utvecklingen av spelet i VR.
- Anton Cevey Tärnholm: Administrativt arbete och ansvarig av spellogik. Utveckling av spel-funktioner.
- Gustaf Wallström: Deltagit i utvecklingen av AR, modellerat landskap och byggnader samt fört protokoll under möten.
- Jonas Långdahl: Deltagit i utvecklingen av flerspelaraspekten i systemet och hjälpt till med AR.
- Tobias Olsson: Huvudansvar för versionshantering, i att sätta upp rutiner och att se till att de följs, och deltagit i utvecklingen av flerspelaraspekten.
- Tobias Ryttinger: Huvudansvarig för 3D-modellering av spelkaraktärer där riggning och animering och texturering ingår. Har också designat spelkorten och lite *koncept art* i projektets början.