

Animal Approach

Förstärkt verklighets-sällskapsspel över nätverk

Jacob Nyman
Viktor Sandberg
Lucas Palnén Rung
Erik Nilsson
Mikael Lundell Vinkler
Julius Kördel

Examinator: Karljohan Lundin Palmerius

Sammanfattning

Rapporten beskriver hur ett projekt utförts för att utveckla ett sällskapsspel till mobila Androidenheter. Syftet med projektet är att utveckla spelet på ett sätt som motiverar användaren att röra på sig. Detta för att få med den fysiska aspekten av klassiska sällskapsspel i ett virtuellt spel med förstärkt verklighet. Enheterna agerar som fönster till en viss del av en gemensam spelplan och deras fysiska position bestämmer vilken del av spelplanen som ska visas.

Projektet består av tre huvuddelar: nätverk, spårning och spelutveckling. Nätverk behövs för att det ska vara möjligt att spela med flera andra enheter. Spårning behövs för att kunna visa en viss del av den gemensamma spelplanen beroende på en enhets fysiska position. Spelutveckling är väsentligt för att skapa ett underhållande spel som människor vill spela mer än vid ett tillfälle.

Spelmotorn *Unity* användes för att utveckla spelet vilket kommer med ett högnivå nätverks API (HLAPI) som används i projektet. Nätverket använder sig utav en server/klient-struktur där en enhet agerar både klient och server och resterande enheter är klienter. *Unity* har även stöd för en del spårningsverktyg som t ex *Vuforia* vilket är verktyget som valdes. *Blender* användes för att 3D modellera alla objekt i spelet. *Scrum* användes som utvecklingsmetodik under projektets gång där projektiden var uppdelat i fem sprintar om tre veckor. Fördelarna med att *Scrum* användes var att projektets planering och arbete var adaptivt vilket innebar att utvecklingen av systemet kunde justeras för att uppfylla nya krav från kunden och korrigera delar av systemet som kunden var missnöjd med. Nackdelar med den agila utvecklingen har varit att strukturen för det generella projektet har blivit sämre. Sämre i form av att sammankopplandet av systemets olika delar blev komplicerade att göra.

Spelplanen är designad för en slumpmässigt genererad terräng med träd och stenar som blockerar synfältet från att se hela planen vid en godtycklig position. Detta innebär att användaren behöver röra sig fysiskt för att se olika delar av spelplanen. Med spårningsverktyget *Vuforia* används en markör som registreras av användarens kamera på sin mobila enhet. I enheten sker beräkningar för att bestämma dess position i relation till markören. Spelplanen skapas virtuellt på markörens position och genom en viss skalning mellan fysiska koordinater i världen och koordinater i spelet, kan en översättning göras från enhetens beräknade positionen till spelets koordinater. De spelbara karaktärerna designades som djur där varje karaktär har en unik förmåga som är relaterat till djuret. *Low Poly*-grafik användes i utvecklingen av spelet både för att öka spelets prestanda och för att vara tilltalande till alla åldrar, speciellt den yngre målgruppen.

Slutprodukten är ett ”fånga flaggan”-spel som har stöd för upptill åtta spelare med max fyra olika lag. Spelet innehåller fyra olika spelbara karaktärer där varje karaktär har en gemensam och en unik förmåga. Det lag som har mest poäng när speltiden är över, vinner.

De slutsatser som kan dras efter detta projektarbete är att det är möjligt med hjälp av virtuell terräng, motivera en användare att förflytta sig fysiskt runt en spelplan. Med hjälp av *Vuforia* är det möjligt att skapa en relation mellan spelplan och spelare, genom denna relation översätts den fysiska translationen till translation av kameran i spelet. För att ett fånga flaggan-spel ska upplevas balanserat behövs en genombrottande design av karaktärer och deras förmågor vilket är svåruppnått. Detta projekt visar även att det krävs genombrottande strukturer och rutiner för att effektivisera projekt av denna storlek.

Innehåll

Sammanfattning	i
Figurer	v
1 Inledning	1
1.1 Bakgrund	1
1.2 Syfte	1
1.3 Frågeställning	1
1.4 Avgränsningar	2
2 Relaterat arbete	3
2.1 Nätverk	3
2.2 Spårning	4
2.3 Inspiration	4
2.3.1 SpaceTrackers	4
2.3.2 Tanks!!!	4
3 Utvecklingsprocesser och rutiner	5
3.1 Utvecklingsmetodik	5
3.2 Kundkontakt	6
3.3 Organisation	6
3.4 Tidsplan	7
4 Rutiner och principer	9
4.1 Versionshantering, -system och rutiner	9
4.2 Dokumentationsprinciper	9
4.3 Kvalitetssäkring	9
4.4 Användartester	10
5 Teknisk Implementation	11
5.1 Tekniska verktyg	11
5.2 Systemarkitektur	11

5.3	Spårning	13
5.3.1	Vuforia	13
5.3.2	Kudan	13
5.3.3	Förbättringar av förstärkt verklighet	13
5.4	Nätverk	14
5.5	Spelmekanik	15
5.5.1	Grundläggande spelmekanik	15
5.5.2	Power Ups	16
5.5.3	Poängsystem	16
5.5.4	Styrning	16
5.5.5	Automatisk generering av spelplan	17
5.6	Karaktärer, modeller och animation	19
5.6.1	Karaktärer	19
5.6.2	Förmågor	20
5.6.3	Modellering & Animering	20
6	Resultat	21
6.1	Spelet	21
6.2	Lobby och GUI	22
7	Analys och diskussion	24
7.1	Metod	24
7.1.1	Arbetsprocess	24
7.1.2	Möten	25
7.1.3	Versionshantering	25
7.2	Resultat	25
7.3	Spelets begränsningar	26
7.4	Problem och motgångar	26
7.5	Arbetet i ett vidare sammanhang	27
8	Slutsatser	28
Litteraturförteckning		30
Appendices		31
A	Arbetsfördelning	32
A.1	Jacob	32
A.2	Lucas	32
A.3	Erik	32

A.4	Viktor	32
A.5	Mikael	33
A.6	Julius	33
B	Användartester	34

Figurer

2.1	Visualisering av peer-to-peer för fyra enheter	3
3.1	Överblick av sprint 3	7
3.2	Preliminär plan vecka 4-21	7
3.3	Reviderad plan vecka 4-21	8
4.1	Profileringsverktyget när spelaren är i spelvärlden.	10
5.1	Abstrakt aktivitetsdiagram	12
5.2	Abstrakt klassdiagram	12
5.3	Markör från Vuforia med Vuforias spårningspunkter	14
5.4	Markör från Kudan med Vuforias spårningspunkter	14
5.5	Fyra spelare placerade bredvid sitt lags bas, sett från två olika klientfönsters perspektiv	15
5.6	Återkoppling att det lila laget just har lämnat en flagga i sin bas	16
5.7	Virtuella joysticken från Animal approach	17
5.8	Slumpgenerering av terräng	18
5.9	Spelbara karaktärer i Animal approach	19
5.10	Animeringsprocess i blender	20
6.1	Karaktärsmeny	21
6.2	Det sammanställda spelet	22
6.3	<i>Unitys Lobby-exempel</i>	23
6.4	Redigerad lobbymeny	23

Kapitel 1

Inledning

Rapporten redogör hur arbetet med det virtuella sällskapsspelet Animal Approach gick till. Den går igenom hur utvecklingsprocessen och den tekniska implementationen av applikationen utfördes. Arbetet med projektet utgick ifrån den agila utvecklingsmetoden *Scrum*.

1.1 Bakgrund

I kursen Medietekniskt kandidatarbete, TNM094, utförs ett projektarbete inom systemutveckling. Projektgruppen består av en grupp på sex personer som med hjälp av teorier och dokumentation kring systemutveckling ska utveckla en produkt. Den produkt som ska utvecklas är ett virtuellt sällskapsspel på förfrågan av kund. Tekniken som används för att utveckla detta spel är förstärkt verklighet. Det är en teknik som växt fram de senast åren och har möjliggjort en stor variation av applikationer. Tekniken bygger på att datorgenererade objekt kan renderas i realtid på en bild av verkligheten med hjälp av kameran på en surfplatta eller mobiltelefon. Anledningen till att denna teknik lämpar sig för att använda till ett sällskapsspel är att den skapar en immersiv miljö som speglas på verkligheten. Detta leder till att det går att skapa spel som kräver att spelaren rör på sig för att se olika delar av spelplanen. Det kommer på så sätt göra att det inte enbart är ett spel för de som är vana vid mer traditionella datorspel utan även för de med mindre intresse av dessa.

1.2 Syfte

Syftet med detta projekt är att utveckla ett sällskapsspel med förstärkt verklighet som skapar en social interaktion mellan spelarna och motivera dem att interagera fysiskt samtidigt som de har roligt. Spelet använder ett verkligt bord som spelplan där enheterna agerar fönster in till den virtuella världen där spelet äger rum. Syftet med rapporten är att呈现出 produkten, dess begränsningar, problem och motgångar under utvecklingstiden samt produktens användargränssnitt.

1.3 Frågeställning

Genom denna rapport kommer följande frågeställningar att besvaras:

- Hur kan design av ett AR-spel motivera en spelare att röra sig i rummet där spelet befinner sig, för att på så sätt få dem att socialt interagera mer med varandra?

- Hur kan en handhållen enhets fysiska koordinater översättas till koordinater i en virtuell spelplan, för att på så sätt visa en viss del av spelplanen beroende på enhetens position?
- Hur ska karaktärerna designas så att alla är unika med olika styrkor och svagheter så att de passar i ett fånga flaggan- spel i en AR-miljö, för att på så sätt tilltala barn från 7 år och uppåt?

1.4 Avgränsningar

För detta projekt gjordes två avgränsningar, en i teknologi och en i målgruppen. Den tekniska avgränsningen är att utveckling endast är riktad mot mobila enheter som använder Android som operativsystem. Avgränsningen gällande målgrupp är att målgruppen för denna produkt är familjer eller kompisgäng men miljön och modellerna ska vara lämpliga för en yngre användare, i detta fall barn från 7 år och uppåt [1].

Kapitel 2

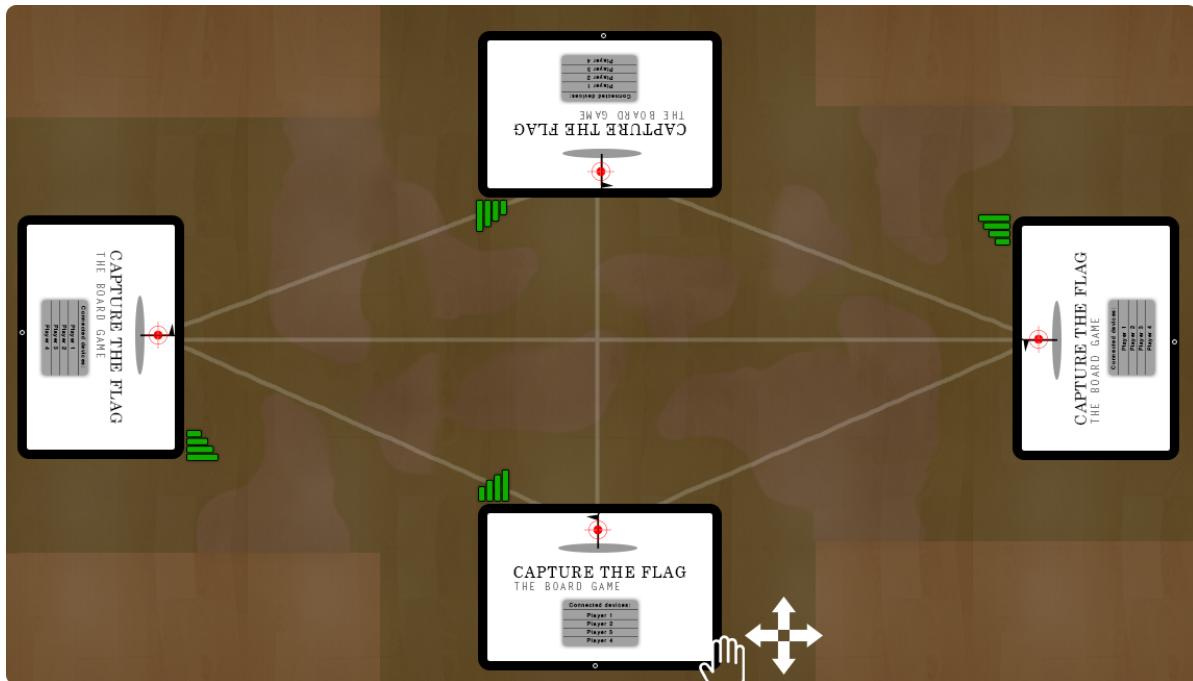
Relaterat arbete

Detta projekt innehåller modeller och tekniker för spårning och nätverk. Det finns olika alternativ och användningssätt, exempel på sådan går att finna i kursen TNM094.

2.1 Nätverk

För att flera spelare samtidigt skall kunna interagera i spelet krävs någon form av nätverkskommunikation. Det finns två vanliga modeller för nätverk, peer-to-peer och server-klient.

Peer-to-peer är ett icke hierarkiskt nätverk vilket medför att alla enheter kan skicka data och interagerar med varandra utan någon mellanhand, en visualisering av detta ses i Figur 2.1. Detta kan vara en snabbare metod för att skicka data i jämförelse med server-klient. En nackdel för denna implementation inom spel är att det kan uppstå osynkroniserat beteende vilket påverkar alla spelarna. Synkroniseringen kan medföra att vissa spelare får en fördel eftersom de får informationen markant före de andra spelarna. Motsatsen till detta kan även förekomma, alltså att en spelares information kommer långt efter de andras.



Figur 2.1: Visualisering av peer-to-peer för fyra enheter

Server-klient är ett hierarkiskt nätverk vilket medför att alla klienter är kopplad mot en enhet, servern. En klient skickar information till servern vilken agerar som mellanhand och distribuerar den till övriga klienter. Eftersom en mellanhand används kan utsändningen av utdata lättare synkroniseras över klienterna vilket är fördelaktigt för ett rätvist spel. Nackdelen för denna struktur är att den behöver en server och om det uppstår problem hos den påverkar det samtliga klienter [2].

2.2 Spåring

För att skapa ett sällskapsspel med gemensam spelkarta krävs teknik som översätter en enhets fysiska koordinater i världen till koordinater i spelet för att kunna visa olika delar av spelkartan beroende på en enhets fysiska position. Spåring är ett aktuellt forskningsområde i både förstärkt verklighet och virtuell verklighetsutvecklingssammanhang[3] (AR och VR) som gör denna översättning mellan fysiska koordinater och spelkoordinater. Flera spårningsverktyg har skapats för att underlätta utveckling till AR applikationer, t ex *Vuforia*, *Kudan* och *MAXST* där spåring kan hanteras med eller utan markör. En markör är en speciell bild som används för att beräkna en kameras position i förhållande till markören. Vissa verktyg har stöd för spåring utan markör där en kamera läser in omgivningen och försöker beräkna dess position i förhållande till omgivningen.

2.3 Inspiration

Projektet har funnit inspiration från olika projekt och från dessa projekt har koncept överförts till den egna spelidén.

2.3.1 SpaceTrackers

SpaceTrackers är ett sällskapsspel med markörbaserad spåring för mobila enheter som utvecklades i kursen TNM094 år 2017. Spelet har en gemensam spelplan där en spelare kan se olika delar av planen med hjälp av en markörbaserad spåring. Nätverket som används är klient/server-nätverk där servern är en enhet som placeras i mitten av spelplanen.

Från detta projekt har inspiration tagits angående hur markörer och spåring skulle implementeras samt även en del idéer kring hur spelutvecklingen och speldesignen skulle vara.

2.3.2 Tanks!!!

Tanks är ett av *Unitys* egna projekt framtaget 2015. Detta var ett spel med stöd för flera spelare på samma tangentbord. I spelet styr spelarna en egen pansarvagn och försöker förstöra de andras pansarvagnar genom att spränga dem. Detta projekt är tillgängligt för alla och år 2017 kompletterades detta projekt med stöd för nätverk och flerspelarläge. Med både versionen från 2015 och 2017 finns det steg för steg instruktioner och förklaringar för det som är inkluderat i projektet [4].

Detta projekt kunde användas som en referens för implementation av nätverk och hur det var menat att en struktur ska byggas upp ett spel i *Unity*. Det gav även utvecklarna inspiration av detaljer vilket förstärker helhetsintrycket.

Kapitel 3

Utvecklingsprocesser och rutiner

Utvecklingsprocessen och dess rutiner är grunden för ett projektarbete. Dessa behöver anpassas för de uppgifter och de förutsättningar vilket varje projektet innehållar.

3.1 Utvecklingsmetodik

Välet av utvecklingsmetod till projektet grundade sig i egenskaperna kontroll och flexibilitet. Kontroll i form av att veta vad som ska göras, när det ska göras och hur det ska göras. För att ha kontroll över projektet behövs även mycket kunskap i ett tidigt skede. Nackdelen med kontroll är att det kan bli svårt när en projektgrupp utan mycket erfarenhet eller kunskap ska använda sig av det. Det blir då sällan bra att värdera kontroll över t.ex flexibilitet. Flexibilitet var det som värderades i detta projektet på grund av att det redan i ett tidigt skede stod klart att anpassning av utvecklingen var nödvändig. Då projektgruppen inte hade tillräcklig kunskap av det som skulle utvecklas blev flexibilitet ännu viktigare på grund av att arbete ger ny kunskap och information löpande [5].

När tidigare nämnda för- och nackdelar lagts fram stod det klart att flexibilitet var det viktigaste för utvecklingen och då kunde en utvecklingsmetod väljas. En agil utvecklingsmetod valdes då de fungerar bra vid förändringar i projekt. Fördelarna med en agil utvecklingsmetod är att projektplanen går att ändra under projektets gång samt att det passar mindre utvecklingsteam. Den agila utvecklingsmetod som valdes till detta projekt var *Scrum*. *Scrum* utgår ifrån en empirisk processtyrning som bygger på tre egenskaper: transparens, granskning och anpassning.

Inom *Scrum* delas arbetet upp i sprintar som är en begränsad tidperiod på högst en månad där utvecklingsteamet ska lyckats utveckla en leverabel. I detta projektet valdes sprintarna till att vara tre arbetsveckor långa. Inom dessa sprintar utfördes ett antal *scrumaktiviteter*. Dessa var: *sprintplanering*, *dagligt scrummöte*, *sprintgranskning* och *sprintåterblick*. Under *sprintplanering* som skedde i början av en sprint bestämdes målet med sprinten och vad som skulle utföras under den. Det skapades även en *sprintbacklogg* med de poster som skulle utföras under varje sprint. Posterna skapades utifrån produktbackloggen som är en lista över allt som behövs till produkten. Posterna ska kunna ta en dag att utföra. Under det *dagliga scrummötet* planerades vad som skulle göras under dagen och arbetet som gjorts sedan förra mötet granskades. Det *dagliga scrummötet* var tidsbegränsat till 15 minuter för att maximera effektiviteten. I slutet av varje sprint hölls en *sprintåterblick* där projektgruppen granskade sig själva och förbättringar som borde göras inför kommande sprintar. En typ av *sprintgranskning* hölls också och där presenterades vad som åstadkommits under sprinten [6].

3.2 Kundkontakt

Möten med kunden har skett vid två tillfällen under projektet. Först hölls ett möte i starten av projektet för att diskutera kundens krav. Det innebar att kunden nämnde de krav som eftersöks och sedan ställdes följd frågor till kunden för att säkerställa att utvecklarna förstod vad kraven innebär. Kunden satte en del krav men inga hårda krav utan utvecklarna fick fria händer till att skapa ett sällskapspel där en enhet agerar som fönster till en gemensam spelplan. Kraven som var eftersökta var att spelet ska vara engagerande, lätt att lära sig men svårt att bemästra. Spelet ska även ha skicklighetsaspekter men även turaspekter så att en nybörjare ska ha möjlighet att kunna vinna. Spelet ska ha en definierad start och slut där en spelomgång pågår med en tidslängd på 3-5 minuter.

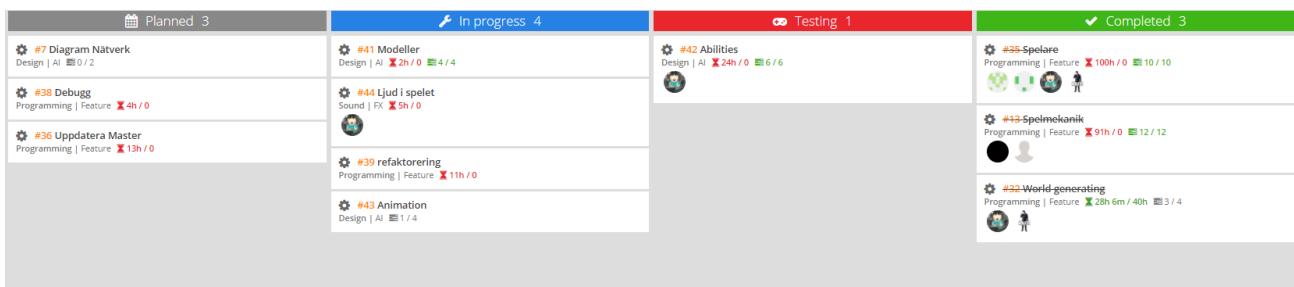
Det andra kundmötet hölls halvvägs in i projektet där syftet var att visa upp implementerade delar och diskutera om ändringar och även diskutera planerade implementation. På mötet diskuterades en del om spelmekanik och design som ledde till ändringar av vissa implementerade delar. Ett krav som kunden prioriterade var att användaren ska behöva röra sig fysiskt för att spela spelet vilket senare påverkade hur spelplanen blev designad. Kundmötet gav en bättre förståelse mellan vad kunden ville ha i systemet och hur utvecklarna hade tänkt utveckla systemet.

3.3 Organisation

Projektgruppen har arbetat med kravhanteringen genom *elicitering*. Detta innebär att de olika synsätt som finns på projektet förenas för att bilda en enhetlig bild av de kraven som sedan leder till kravhanteringen. De krav som fastställts för projektet har brutits ner och förfinats för att sedan läggas in i *produktbackloggen*, vilket har gjorts av produktägaren. Inom varje sprint har en *sprintbacklogg* skapats där de posterna som varit högst prioriterade i *produktbackloggen* hamnat. Posterna från *produktbackloggen* delades upp i mindre beståndsdelar vilket har blivit posterna i *sprintbackloggen*. På detta sättet kommer kraven för systemet att delas in i mindre beståndsdelar som utvecklarna kan arbeta med.

Det verktyg som har använts för att skapa både produkt- och *sprintbackloggen* är *hacknPlan*. Detta är ett verktyg där alla krav och poster kan listas och delas upp mellan varje utvecklare. I *hacknPlan* går det att spåra vilken utvecklare som gjort vad och när detta gjordes. Detta gör att det går att få en tydlig överblick över statusen på varje post. I *hacknPlan* finns det fyra olika statusar vilket en post kan befina sig i. Dessa är planerad, i utveckling, testning och slutförd.

De poster som hamnat i *sprintbackloggen* har delats upp mellan utvecklarna, vilket har medfört att det alltid är någon som har ansvar för varje enskild post. Ansvaret tilldelades i början av varje sprint. För att underlätta överblicken av vad alla utvecklare arbetar med tilldelades uppgiften i *hacknPlan* till utvecklaren. När en post är avklarad försvinner den från *hacknPlan*, det leder till att en post inte görs två gånger och en utvecklare kan se vad som finns kvar att göra. Varje post i *sprintbackloggen* förväntas vara möjlig att genomföras under en arbetsdag men avvikelse förekommer. Hur det såg ut vid slutet av en arbetsdag under sprint 3 ses i Figur 3.1.



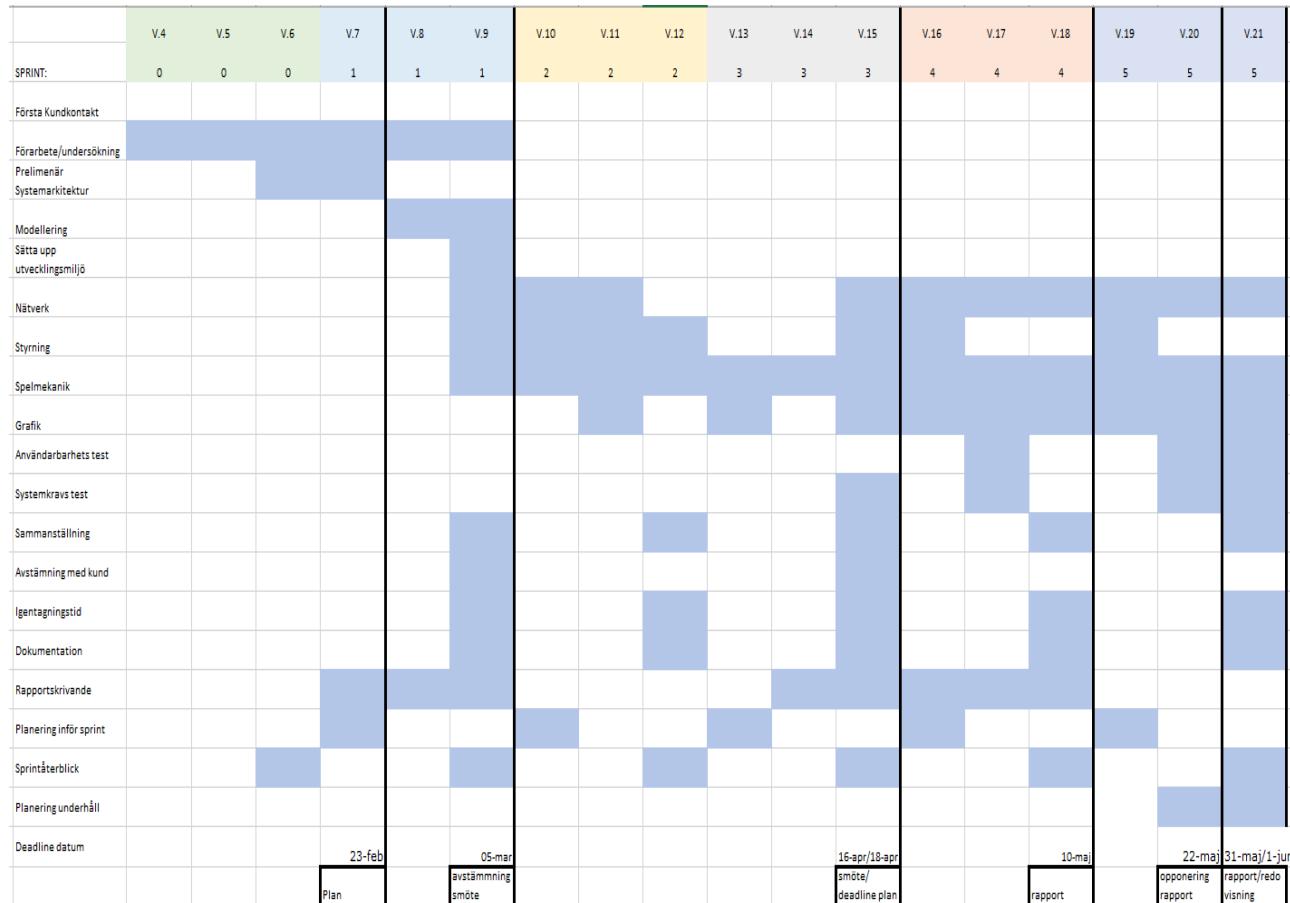
Figur 3.1: Överblick av sprint 3

3.4 Tidsplan

Vid projektarbetets början gjordes en uppskattning av de olika momenten vilket arbetet skulle innehållt. Tidsspannet för det totala arbetet är fyra månader från första kundkontakt till sluttid för leverabel. Det beslutades att bedriva arbetet i sprintar om tre veckor vilket medför att de fyra månaderna blev uppdelade i fem sprintar. Planen var skapad med rum för förändring och tänkt att möjliggöras med hjälp av de rutinbaserade mötena enligt *Scrums* ramar [6]. Målbilden för den preliminära planen var att skapa en gemensam förståelse inom projektet för den tillgängliga tiden och att prioritera uppgifter efter deadlines och milstolpar. Den preliminära planen ses i Figur 3.2.

Figur 3.2: Preliminär plan vecka 4-21

Uppgifter tog sällan den tid de blivit uppskattade att ta. Detta, tillsammans med att uppgifter har omprioriterats vid *sprintplanering*, vilket medför att planen har reviderats. En flexibel tidsplan är användbart då arbetet inte varit låst vid den ursprungliga planen. Arbetet har kunnat anpassats sig efter utökad förståelse inför uppgifter och förändringarna i prioriteringslistan. En reviderad version av planen ses i Figur 3.3.



Figur 3.3: Reviderad plan vecka 4-21

Förändringen mot den ursprungliga tidsplanen var i form av att testning bortprioriterades i det tidiga skedet av projektarbetet. Dessutom genomfördes moment för nätverk och styrning snabbare än uppskattat. Då lades fokus och resurser på andra områden istället vilket ses i den reviderade tidsplanen i Figur 3.3.

Kapitel 4

Rutiner och principer

Projektet byggs på rutiner och principer för att säkerställa ett fungerande samarbete och rätt kvalitet på utveckling och slutprodukt.

4.1 Versionshantering, -system och rutiner

Den mjukvara som används för att hantera versionshanteringen var till en början *Git*, men eftersom *Git* inte kan hantera den valda spelmotorn *Unitys META-filer* behövde versionshanteringsverktyget byteas. På grund av detta valdes en annan mjukvara för att hantera detta. Det versionshanteringverktyget som istället valdes att användas är *Unity Collaborate*. Eftersom en grenstruktur inte finns i *Collaborate* är det viktigt att inte arbeta i samma filer då det resulterar i att en fil måste väljas över den andra. Därför blev rutinen sådan att de utvecklare som skulle sammanfoga sitt arbete var tvungna att sitta tillsammans för att minimera risken för felaktig sammanställning.

4.2 Dokumentationsprinciper

Dokumentation under projektets gång skedde i form av att varje gruppmedlem i slutet av varje dag dokumenterade i *hacknPlan* vad som gjorts under dagen och hur lång tid det tagit. Utöver detta kommenterades all kod av den som skrivit den. Dessa kommentarer finns i början av varje kodfil som beskriver vad kodens syfte är, samt vem eller vilka som har arbetat med koden. Större funktioner har också kommenterats med dess uppgift och funktion.

För att kodningen för gruppen ska vara enkel att påbörja har *Unified modelling language*-diagram används. UML-diagrams syfte är att hjälpa teamet att förstå en komponents struktur, uppbyggnad och hur den hänger ihop med andra komponenter, med hjälp av grafik. I projektet underlättade abstrakta diagram för beskrivning av komponenter. Det valdes abstrakta eftersom konkreta diagram tar lång tid att göra samt att det tar extra tid att hålla de uppdaterade utan att direkt resultera i någon användbar artefakt [7].

4.3 Kvalitetssäkring

Kvalitetssäkring genomfördes genom att kodgranskning utnyttjades. I synnerhet användes partnergranskning där en annan gruppmedlem läser igenom koden som skrivits och bedömer om den är förståelig och därmed kan godkännas. Godkänns den inte måste koden *refaktoreras*. För att optimera

och identifiera effektiv kod har *Unity*s profileringsverktyg använts. Detta verktyg tillåter användaren att se vilka funktioner som anropas, hur stor del av den totala användningen den funktionen står för samt hur många gånger respektive funktion anropas. Profileringsverktyget har använts när funktionsliten hos en komponent är uppnådd. Utvecklaren kan då utifrån denna statistik som presenteras av profileringsverktyget få en indikator av vilka delar som behöver genomgå *refaktorering* eller omskrivas för att optimera produkten. Hur profileringsverktyget kunde se ut när produkten kördes ses i Figur 4.1.



Figur 4.1: Profileringsverktyget när spelaren är i spelvärlden.

Det finns även ett annat verktyg som användes när rutinerna för partnergranskning var svåra att efterleva. Det verktyget och rutinerna som ersatte var att istället utföra en kodpresentation. Under en kodpresentation presenterar alla medlemmar den kod eller komponent utvecklaren har arbetat med sedan senaste granskning, hur dessa fungerar och nödvändig kunskap för att kunna använda dem.

4.4 Användartester

När produkten var färdig utfördes användartester. Dessa tester gick ut på att se hur andra personer upplevde spelet och hur väl återkopplingen fungerade. Testpersonen fick navigera ostört genom spelets menyer och starta en lobby, där personer från projektgruppen anslöt sig till lobbysektionen för att möta personen i spelet. Efter att spelet var färdigt, fick testpersonen skriva i ett formulär om hur upplevelsen var.

Den feedback projektgruppen fick från användertesterna hjälpte till att skapa nya visioner över vad som behövde uppdateras med produkten. Dessa funktioner kunde dock inte implementeras till slutprodukten på grund av att det inte fanns tid, utan har istället lagts som ett framtida projekt. I Bilaga B finns resultatet från användertesterna.

Kapitel 5

Teknisk Implementation

För att översätta vision till ett färdigt spel behövs tekniska verktyg. Det finns många att välja mellan och oftast har de för och nackdelar vilket kommer påverka artefakten olika mycket beroende på sammanhanget.

5.1 Tekniska verktyg

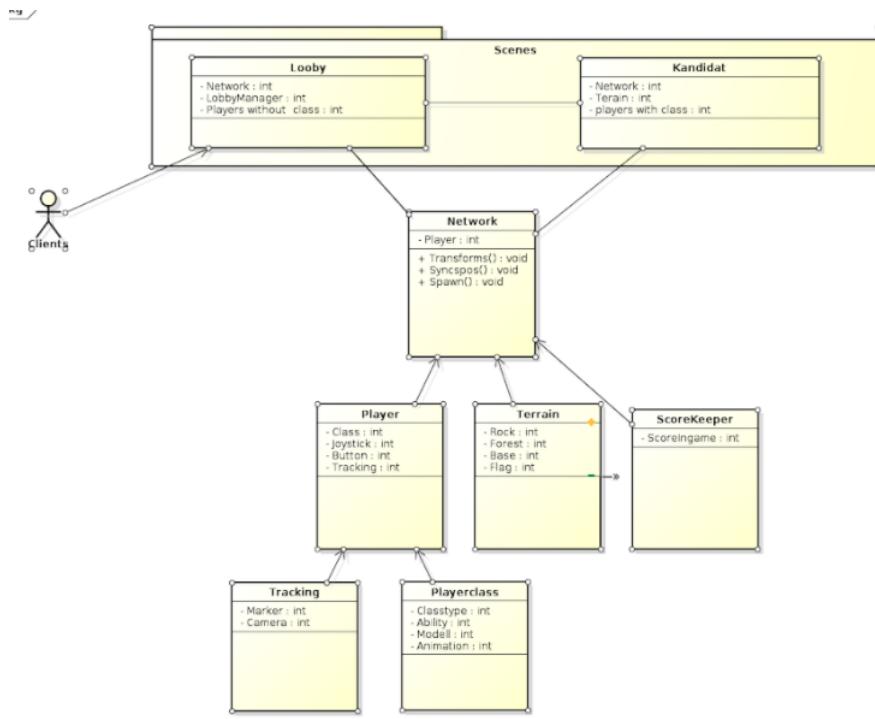
Detta projekt valdes att utvecklas med en spelmotor. Det finns flera olika att välja på, några av de alternativ som övervägts är *Unity*, *Unreal engine* och *Ogre3D*. Beslutet blev att använda *Unity* för detta spel då det var den spelmotor flest utvecklare var bekant med och att genom denna bekantskap var tanken att minska inlärningskurvan. *Unity* är även den största och mest använda spelmotorn för mobilspel [8]. *Unity* möjliggör för utvecklarna att komma igång med utvecklingen tidigt och låta dem fokusera på att förbättra och anpassa komponenter vilket redan är skapade i *Unity*. Detta för att minska den behövda tiden för att behöva återuppfinna funktionaliteter som redan fungerar i *Unity*. Istället kunde de främmande momenten som spårning och nätverk fokuseras på.

5.2 Systemarkitektur

Spelet är gjort för flera spelare. För att öka den allmänna förståelse för hur nätverket samverkar mellan spelarna och deras inmatningar skapades ett aktivitet- samt klassdiagram. Båda diagrammen är av abstrakt typ och har använts som bas för diskussioner och förklaring men ändå har de inte varit förklaringen i sig. Aktivitetsdiagrammet ses i Figur 5.1 och klassdiagrammet i Figur 5.2.



Figur 5.1: Abstrakt aktivitetsdiagram



Figur 5.2: Abstrakt klassdiagram

5.3 Spårning

Ett förstärkt-verklighetsspel behöver ett spårningsverktyg för att systemet ska veta var användaren befinner sig med enheten i den virtuella världen.

5.3.1 Vuforia

Spårningen som används i spelet skapades via *Vuforia*. *Vuforia* är ett förstärkt-verklighets API som är specifikt utvecklat för AR-applikationer i *Unity* [9]. Det är därför det implementeras då spelet är skapat i *Unity*. Spårningen fungerar med hjälp av en markör som placeras på ett bord som sedan agerar spelplan. Detta sker när enheten som spelet spelas på hittar markören med hjälp av enhetens kamera. Spelarens enhet blir då ett fönster till spelplanen och måste därför flytta sig för att kunna se alla delar. När spelaren flyttar enheten, flyttas även den virtuella kameran som är riktad mot spelplanen. Det vill säga att translation i verkligheten översätts till translation i spelet. Detta sker med hjälp av vad enhetens kamera identifierar tillsammans med hjälp av enhetens accelerometer.

Under projektutvecklingen testades *Vuforia* och det visade en sämre AR-upplevelse än förväntat. Spårningen var fördröjd och földe inte med enhetens rörelser trots att kameran var riktad mot markören.

5.3.2 Kudan

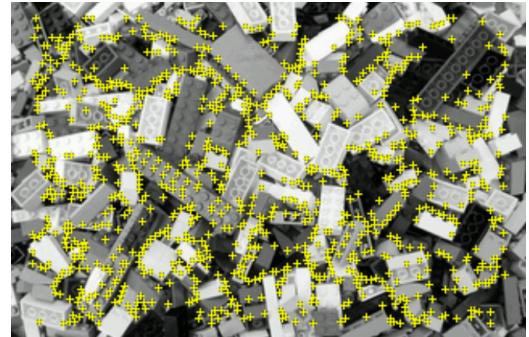
Gruppen beslutade att kolla på ett alternativ till *Vuforia*, *Kudan AR*. *Kudan* är ett utvecklat AR-ramverk med en stabil och avancerad typ av spårning [10]. I *Kudans* utvecklingsmiljö är det markören som flyttas beroende på hur enheten styrs. Det betyder att kameran har en låst position i rummet. *Kudans* låsta kameraposition påverkade färdigutvecklade delar negativt vid implementation. Det som slutade fungerade var placering av objekt vid start av spelet, styrning av karaktär och gravitationen. Det gör att implementering av *Kudan* kräver att färdigutvecklade funktioner behövdes skrivas om från grunden. Istället för att implementera *Kudan* valdes det att förbättra spårningen av *Vuforia*.

5.3.3 Förbättringar av förstärkt verklighet

En viktig del av markör-spårning är hur markören ser ut. Markörspårningen har använts för att alla spelare ska se kartan på samma ställe. Utan markör är det lätt att spelarna har sin karta bredvid motståndarens. Markören som användes först var *Vuforias* egna som antogs vara bra och pålitlig. Markören byttes ut till en som används av *Kudan*, det gav ett positivt resultat på spårningen. Den nya markören har många fler spårningspunkter, vilket ger kameran mer punkter att utgå ifrån och spåra. I *Vuforias* utvecklingsverktyg är det möjligt att se spårningspunkter på en markör. I Figur 5.3 och 5.4 nedan ses skillnaden i spårningspunkter mellan den första markören från *Vuforia* och *Kudans* markör.



Figur 5.3: Markör från Vuforia med Vu-
forias spårningspunkter



Figur 5.4: Markör från Kudan med Vu-
forias spårningspunkter

För att förbättra spelupplevelsen och stabiliteten på den förstärkta verkligheten används *Extended Tracking*, vilket innebär att enhetens kamera tar upp vad som visas i bakgrunden och använder det för att spåra när markören inte syns. Det medför att markören inte alltid måste vara i bild för att spelplanen ska synas, vilket förbättrar spelupplevelsen genom att spelaren inte alltid måste ha fokus på att se markören och kan tappa anslutningen till markören utan att spelet avbryts som det hade gjorts annars.

5.4 Nätverk

Till en början ansågs peer-to-peer vara den optimala nätverkslösningen där planen var att undvika att ställa krav på användaren att ha en extra enhet som enbart agerar server vilket krävs i klient-server nätverk. Därefter upptäcktes att *Unity* har ett högnivå API som kallas “High Level API” där en enhet kan agera både klient och server[11]. Av den anledningen valdes *Unity*’s API som nätverkslösning för systemet. Ytterligare en anledning är att högnivå API ofta leder till snabb implementation. Detta ledde till att systemet tidigt hade ett fungerande nätverk. Nackdelen med att använda *Unity*’s högnivå API är att det tar mycket tid att ändra viss funktionalitet eftersom att full förståelse i flera sammankopplade C# klasser/skript krävs.

Alla spelobjekt som behöver vara synkroniserade över nätverket ges en komponent som kallas *NetworkIdentity* som ger objektet ett id över nätverket och möjliggör synkronisering. Spelobjekt som rör på sig behöver ytterligare en komponent som kallas *NetworkTransform* som skickar över data om hur objektets position och rotation förändras.

För att kommunicera med servern från klienter genom skript ska funktioner markeras som en kommando-funktion `emphCommand`. Ifall servern behöver uppdatera alla klienter så kallas funktioner markerade `ClientRPC`.

Ett färdiggjort lobbygränssnitt av *Unity* hämtades från *Unity*’s “Asset Store”. I gränssnittet kan användare skapa eller söka efter matcher genom *Unity*’s flerspelarmolntjänst (multiplayer cloud service).

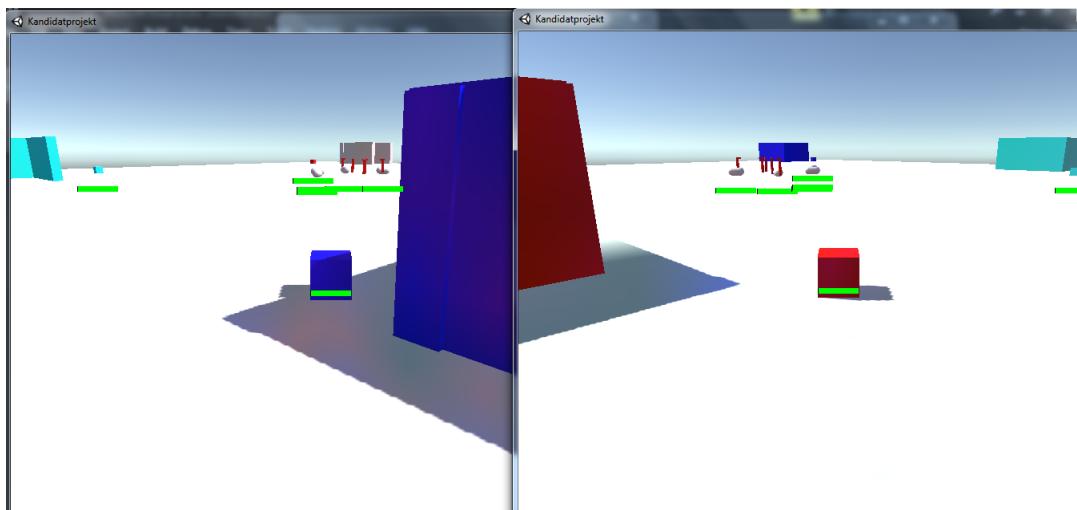
5.5 Spelmekanik

Även den bästa tekniska implementationen av spårning och nätverk skulle vara överflödig ifall spelet i sin kärna skulle sakna element vilket gör det underhållande. Spelmekaniken är vad som får den tekniska implementationen att upplevas ännu bättre och är anledningen till att spelare återkommer till artefakten.

5.5.1 Grundläggande spelmekanik

Den första spelmekaniken som implementerades var att en spelare ska kunna ta upp flagg-objekt vilket är essentiellt då sällskapspelet planerades att bli ett fånga/erövra flaggan spel (*capture the flag*). Spelaren var i början en kub som styrdes med tangentbordet. Därefter skapades ett bas-objekt där flaggor kan lämnas. Spelet har därmed de grundläggande spelmekanikerna: flaggor skapas och placeras i mitten av spelkartan och kan plockas upp av spelare och lämnas i en bas. En flagga kan stjälas både från andra spelare och baser. Därefter implementerades att antalet baser som skapas i en match beror på antalet spelare så att varje spelare har en bas att lämna flaggor till. Varje spelare sparar ett ID till en bas för att säkerställa att flaggan enbart kan lämnas i spelarens bas vilket är väsentligt för att senare kunna ge poäng.

Spelarna kan i lobbyn välja en färg som bestämmer vilket lag de vill vara och därefter ändrades att en bas skapas per lag istället för per spelare. En ”ordlista” (kallad *Dictionary*[12] i C#) används för att lagra basID för olika färger. Ordlistor fungerar med en nyckel-och-värderelation där ett värde lagras per nyckel. Nyckeln är i det här fallet färg och värdet är ett basID. På så sätt kan spelare i samma lag, det vill säga, spelare med samma färg få ID till samma bas att lämna flaggor till. Baserna färgas i lagens färger för att kunna åtskilja var en spelare ska lämna flaggor. Spelarna, som fortfarande är kuber, färgas även till lagets färg som valdes i lobbyn. Spelet har stöd för max fyra lag och åtta spelare vilket innebär att baser ska skapas på max fyra olika ställen på spelkartan. Därför skapas en lista som innehåller fyra olika koordinater i spelkartan, varje bas får unika koordinater där de ska skapas. Ytterligare en ordlista skapas för att lagra vilken bas som fick vilka koordinater där nyckeln återigen är färg. Med hjälp av den listan kan spelare placeras i närheten av sitt lags bas i början av matchen, se Figur 5.5. Senare implementerades olika karakterer med olika egenskaper som kunde spelas istället för en primitiv kub.



Figur 5.5: Fyra spelare placerade bredvid sitt lags bas, sett från två olika klientfönsters perspektiv

5.5.2 Power Ups

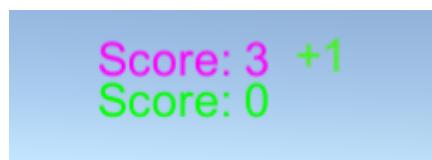
I spelet finns det objekt som kan plockas upp av spelaren. Detta objekt ger spelaren olika förmågor beroende på vilket sorts objekt som plockats upp. Förmågan är oftast tidsbaserad och försvinner där-för efter ett par sekunder. Dessa skapades genom att skapa objekt som har en kollisionsarea och när ett objekt som är markerat som spelare kommer innanför den arean får spelarens karaktär en tillfällig förmåga och sedan förstörs power-upen. Av de implementerade Power-upen finns det en hastighets-ökning som gör att spelaren rör sig snabbare i en kort stund och en power-up som teleporterar alla spelare till en slumpmässigt vald plats på kartan. Dessa är implementerade så att det skapas två powerups när spelet startar och om en plockas upp av en spelare tar det en kort stund innan en ny power-up skapas på nytt. Denna nya power-up är slumpmässigt vald bland de power-ups som skapas. Förmågorna är till för att kunna få ett tillfälligt övertag i spelet och för att dessa inte ska kunna användas hela tiden behöver de begränsas. Begränsningen sker genom att varje förmåga kan endast användas en gång inom en viss tidsram.

5.5.3 Poängsystem

Poängsystemet baseras helt på antalet flaggor som finns i ett lags bas. Spelet pågår som minst under ett förutbestämt antal sekunder t ex. 300 sekunder och laget med flest flaggor i sin bas när tiden är slut vinner. Varje gång en spelare tar en flagga från någon annans bas läggs några sekunder till på den kvarstående speltiden. Ifall ingen vinnare kan avgöras på grund av att t ex två lag har lika många poäng/flaggor så pågår matchen tills det enbart finns ett lag med fler flaggor i sin bas än alla andra lag.

För att visa poäng under spelets gång så skapas i början av matchen, likt hur baser skapas, ett textobjekt per lag där textobjektet sparar i en ordlista med färg som nyckel. Ytterligare en ordlista används för att spara poängvärdet för varje färg-nyckel.

Alla spelare kan se alla lags poäng under spelets gång, texterna som visar poäng är i lagets färger. Ett lags poäng ökas med ett när en flagga lämnas i lagets bas och minskas med ett ifall ett annat lag tar en flagga från dess bas. Poängtexterna uppdateras vid dessa två tillfällen vilket kan påverka spelarnas taktik att t ex försöka ta flaggor från det vinnande lagets bas. Utöver att texterna uppdateras så används ytterligare återkoppling som visar hur ett lags poäng blev påverkat, ifall en flagga lämnades i en bas visas en text bredvid poängtexten, se Figur 5.6. Liknande återkoppling sker när ett lag förlorar poäng.



Figur 5.6: Återkoppling att det lila laget just har lämnat en flagga i sin bas

5.5.4 Styrning

För att kontrollera karaktärens rörelse används en virtuell joystick. En virtuell joystick är ett element som finns på skärmen som interageras med genom att dra spaken åt olika håll. Det finns andra alternativ, ett sådant är knappar i alla riktningar likt en joystick eller att användaren lutar sin enhet. I Animal approach används en joystick som ses i Figur 5.7.

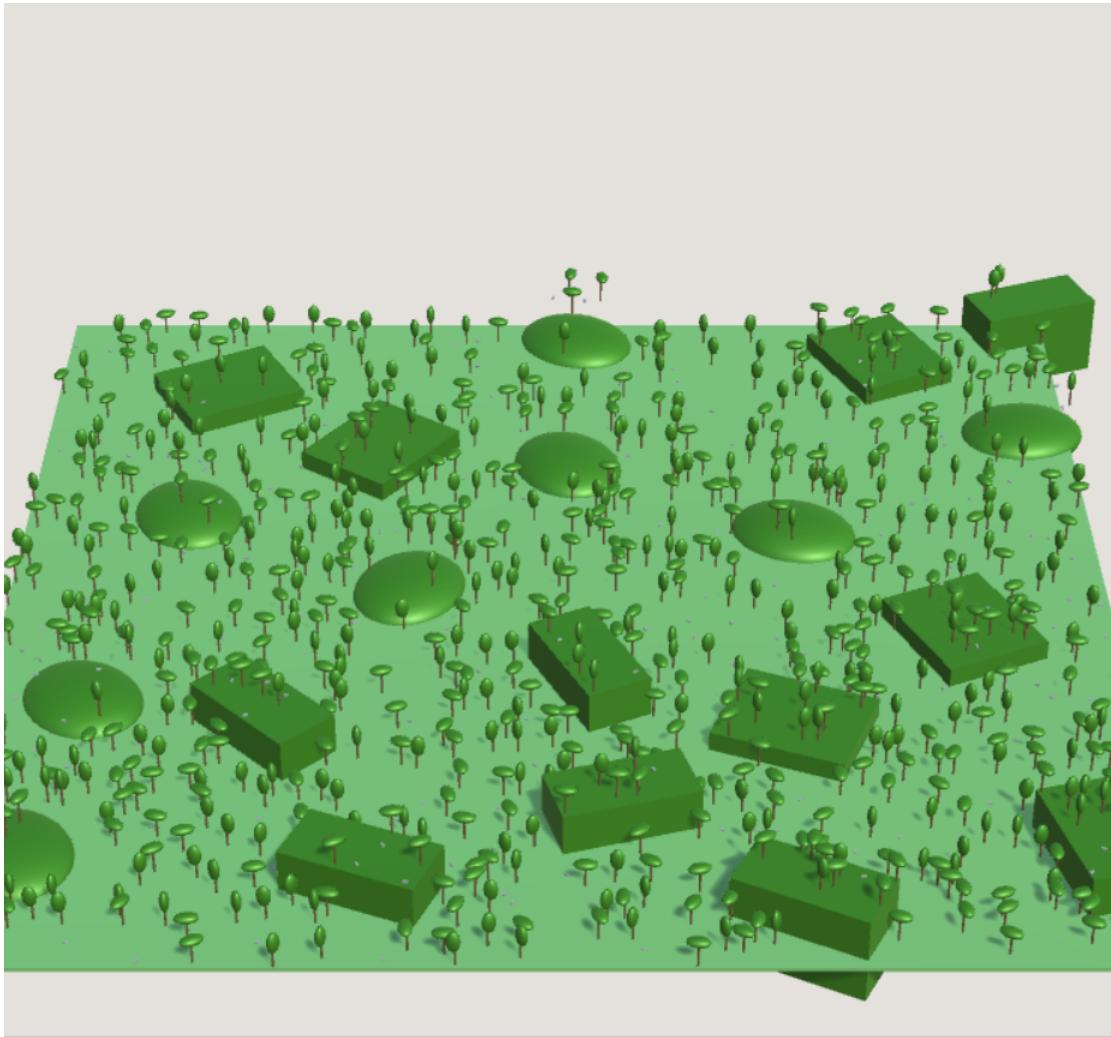


Figur 5.7: Virtuella joysticken från Animal approach

Flera olika versioner av styrningen diskuterades och utvecklades. En var att karaktären rörde sig beroende på hur karaktären var riktad. Ifall spelaren drar joysticken uppåt förflyttas karaktären åt det hålet den tittar mot. Dras joysticken istället åt vänster förflyttas karaktären i sidled till vänster av karaktärens riktning. En annan versionen av styrningen som testades var att använda fram och bak för translation och sidled för rotation. Implementation går ut på att joystickens vänster- och högerinmatning roterar karaktären åt varsitt håll. Dras joysticken upp rör sig karaktären framåt medan ner förflyttar karaktären bakåt. Ingen av dessa versioner som kan känna bra i andra spel kändes naturliga i en AR-miljö, utan var istället svåra att förstå och spela med. Det blev för svåra att förstå eftersom kameran rör på sig och karaktärens rörelser blir olika varje gång spelaren flyttar på sig fysiskt. Istället valdes det att utveckla en version som beror på hur spelaren riktar kameran mot spelplanen. Karaktären rör sig utifrån kamerans riktning, där upp på joysticken flyttar karaktären i riktning rakt fram sett från kameran. Joystickens höger- och vänsterinmatning rör karaktären höger och vänster med avseende på kamerans riktning. Detta ger en naturlig spelkänsla samtidigt som den är enkel att förstå.

5.5.5 Automatisk generering av spelplan

För att få spelplanen att känna levande och immersiv skapades ett system som skulle generera terräng till spelplanen. Detta system bygger på att spelplanen delas in i ett rutnät där terrängen kan placeras. Inom varje ruta kan sedan terrängen placeras slumpmässigt och sedan går systemet vidare till nästa ruta tills den gått igenom hela rutnätet. För att göra systemet flexibelt och ge det stor detaljrikedom har det även gjorts så att rutnätet kan ändra storlek och att det kan passera igenom fler gånger. När systemet går igenom rutnätet mer än en gång söker algoritmen i en cirkel runt objektet för att säkerställa att den inte kolliderar med ett annat objekt. Det har även gjorts så att olika typer av terrängobjekt kan placeras ovanpå varandra samt att de skapas olika ofta för att få den densiteten av objekt som passar. För att se till så att objekten skapas på själva spelplanen används en *raycast* som skjuts nedåt och då den träffar spelplanen vet scriptet om var den ska skapa objektet. I Figur 5.8 ses ett exempel på hur slumppgenereringen av spelplanen kan se ut.



Figur 5.8: Slumpgenerering av terräng

Den automatiska genereringen av spelplanen sker på servern och är krävande då det är många objekt som skapas samtidigt på kort tid. Detta medför att enheter med sämre prestanda kan uppleva att spelet är hackigt. För att motverka detta skapades därför *för-renderade* versioner av spelplanen. I dessa versioner gjordes alla beräkningar och objekt först för att sedan sparas ned till ett enda objekt som enkelt kunde läggas in i en lista. Ur denna lista slumpsades det sedan vilken av de olika spelplanerna som skulle användas. Detta gjorde att många beräkningar inte behöver göras vid start eller live i spelet. Det medförde dock även att spelplanen bara kan varieras lika mycket som det finns antal sparade objekt. I de automatgenererade versionerna av spelplanen är varje iteration av den unik. Här blir det därför ett val mellan spelupplevelse och prestanda. Eftersom båda sätten är implementerade är det enkelt att byta mellan för att matcha spelarens krav. Detta är dock inte implementerat som en funktion i produkten utan är endast möjligt att ändra via små ändringar i koden. För att förhindra att objekt hamnade i vägen eller helt blockerade spelaren flyttades de objekt som var placerade kring viktiga platser på kartan. Dessa platser är i huvudsak spelarens baser samt området där flaggan är. De automatgenererade objekten placeras innanför en kvadrat med bergsväggar som karaktären inte kan gå upp för. Bergsväggarna begränsar spelaren rörelse inom kartans område. I kartbasen finns det även en överhängande stenstruktur för att begränsa användarens synfält, för att spelaren inte ska få en överblick över hela kartan.

5.6 Karaktärer, modeller och animation

Gemensamt för alla modeller som har gjorts till spelet är att de har skapats enligt stilen *Low Poly*-grafik som innebär att modellerna medvetet är skapade med få polygoner. *Low Poly*-grafik lämpar sig bra för denna typen av applikation då det kräver mindre prestanda att rendera men ger även en minimalistisk känsla som passar för hela målgruppen. *Low Poly*-grafik är dessutom generellt inte skapad med texturer på ytan, utan materialen som används är enbart fasta färger för olika delar av modellen. På samma sätt är karaktärer och modeller som används i applikationen gjorda. Styrkan med att använda fasta färger som material på *Low Poly*-grafik är att när ljuset i scenen reflekterar mot modellerna ger det illusionen av att det är en textur som ligger på modellen, men att i själva verket är det skuggor som faller över modellen som ger upphov till färgändringar.

5.6.1 Karaktärer

I spelet finns det fyra olika spelbara karaktärer som är i motiv av en björn, en räv, en spindel och en struts där modellerna kan ses i Figur 5.9. Att de spelbara karaktärerna är djur är motiverat av att försöka nå ut till den målgrupp nämnd i sektion 1.4.

De olika karaktärerna som används i spelet har baserats på det klassiska spelet *Dungeons & Dragons* karaktärsdesign. Varje karaktär har olika attribut som motsvarar standardklasserna i spelet: Krigare (*Warrior*), Magiker (*Wizard*) och Tjuv (*Rouge*). [13]. Utifrån dessa karaktärsmodeller valdes 4 djur som skulle kunna representera dessa klasser bäst.

Som *Krigare* valdes en björn som representation då krigaren är en klass som inte har hög mobilitet men är stark i närrist. Som *Magiker* valdes en struts eftersom karaktären har förmågan i spelet att kunna lägga ägg som sprängs om en motståndare trampar på dem. *Tjuven* är en klass med hög mobilitet och smidiga rörelser, där valet av djur valdes till en räv. Den sista karaktären valdes utifrån att gruppen ville ha en karaktär med förmåga att kunna skjuta ut en projektil och dra sig till det område som projektilen träffar. För att hitta en bra representation för denna förmåga valdes en spindel som djur då förmågan kan relatera till en spindels spindelväv. Spindeln blir därför en karaktär med hög mobilitet och faller därför också under klassen *Tjuv*.

Eftersom att karaktärerna är baserade på olika klasser innebär det också att de är skapta för att ge olika spelkänslor för användaren. Med varje karaktärs olika förmågor och mobilitet gör det också att varje individuell karaktär har sin egen inlärningskurva och svårighetsgrad.



Figur 5.9: Spelbara karaktärer i Animal approach

5.6.2 Förmågor

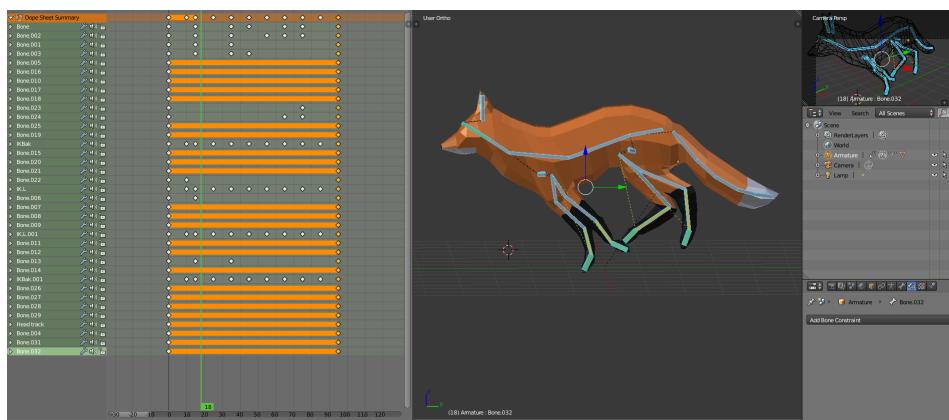
För att göra spelet mer engagerande och skicklighetsbaserat har unika förmågor skapats till varje karaktär. Alla karaktärer har en förmåga för att ta flaggor från de andra karaktärerna som kallas "Tag". Den använder sig av en kollisionsarea som aktiveras framför spelaren efter ett knapptryck och om en motståndare är inom kollisionsarean kommer den att förlora flaggan. Räven har en förmåga som kallas "Dash" och den translaterar spelaren i riktningen den är vänd åt. Björnen har en förmåga som kallas "Roar". När förmågan aktiveras skapas en stor kollisionsarea framför björnen. Alla motståndare som är inom den kollisionsarean förlorar förmågan att röra sig under en begränsad tid. Spindeln har en förmåga som kallas "Hook" som innebär att spindeln skjuter iväg ett objekt och om det objektet kommer i kontakt med en kollisionsarea kommer spindeln translateras till den kollisionsarean. Strutsen har en förmåga där den lägger ägg. När en spelare går på ägget, exploderas det och skjuter iväg närliggande karaktärer. Längden av karaktärens förflyttning beror på karaktärens avståndet till explosionen.

5.6.3 Modellering & Animering

Alla djur modellerades i *Blender* med hjälp av referensbilder av respektive djur. Detta hjälpte till att djuren kunde efterlikna de verkliga djuren, även fast få polygoner användes. Efter att karaktärerna var modellerade, byggdes skelett upp för respektive karaktär för animering. Skeletten bygger på djurens verkliga skelettuppbryggnad för att beteendet vid animation ska vara likt deras verkliga rörelser.

Varje karaktär animerades med 5 olika animationer. Den primära animationen som skapades var en springanimation för varje enskild karaktär. Animationerna är skapta utifrån djurens riktiga springcykler för att ge en trovärdig representation. Figur 5.10 visar animationsprocessen i *Blender* där rävens springcykel animeras. Karaktärerna fick även animationer för gång, två olika animationer för stillastående läge och en animation för karaktärens förmåga.

Efter att alla karaktärer var modellerade och animerade, exporterades filerna från *blender* som *.blend*-fil till *Unity*. I *Unity* kopplades sedan karaktärerna till en *Animation Controller* som hanterar vilka animationer från karaktären som visas. Joysticken som beskrevs i sektion 5.5.4 kopplades också till kontrollen och beroende på hur mycket joysticken har dragits, spelas antingen springanimationen eller gånganimationen upp. Om joysticken inte rörs, körs slumpmässigt någon av de två stillastående animationerna.



Figur 5.10: Animeringsprocess i blender

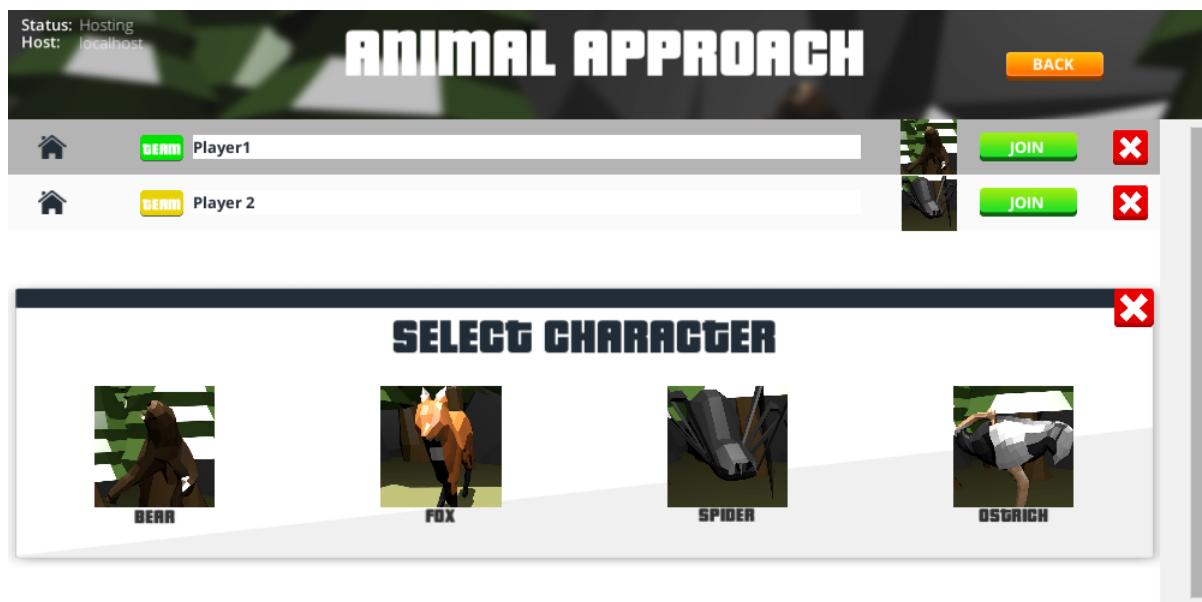
Kapitel 6

Resultat

I detta kapitel kommer resultatet av de olika moment av projektet att presenteras.

6.1 Spelet

Den resulterande produkten av projektarbetet är ett spel som spelas med hjälp av surfplattor eller mobiltelefoner och en spelmarkör. Spelmarkören läggs på ett bord eller på golvet och genom att användaren riktar enhetens kamera mot markören kan användaren se en del av spelplanen. I spelet kan användaren tävla mot andra spelare över ett nätverk. Användaren kan välja att ansluta till en existerande lobby eller att skapa en egen där varje lobby har kapacitet för upp till åtta olika spelare. I lobbyn kan sedan alla spelare välja en av fyra karaktärer som har olika förmågor och spelaren kan dessutom välja färg, vilket representerar vilket lag den spelaren tillhör. Detta presenteras i Figur 6.1.



Figur 6.1: Karaktärsmeny

I spelet skapas en bas för varje lag. Målet är sedan att förflytta sin karaktär från sin bas in till mitten där det finns en flagga som karaktären plockar upp om den kommer inom dess kollisionsarea. Spelaren ska sedan försöka ta sig tillbaka till sin bas. Motstående lag kan stoppa detta genom att använda sin förmåga på karaktären med flaggan vilket då gör att den får flaggan. Om spelaren kommer tillbaka

till sin bas utan att bli stoppad får det laget ett poäng och en ny flagga skapas i mitten av kartan. Motstående lag kan dock ta flaggor från andras baser och då förlorar det andra laget poäng men ingen ny flagga skapas av detta. Varje flagga kan enbart skapa en ny flagga när den lämnats i en bas för första gången. Detta innebär att enbart den nyaste flaggan som skapats kan i sin tur skapa nästa flagga. Det är därför möjligt att tappa bort den flaggan under spelets gång vilket motiverar användaren att även ta flaggor från varandras baser.

För att hjälpa spelaren finns i spelet olika *Power-Ups* som skapas var 15e sekund och ger spelaren som plockar upp den en tillfällig förstärkning, till exempel kan karaktären bli snabbare eller att spelaren teleporteras till en slumpmässig plats på spelplanen. För att vinna krävs det att ha mest poäng efter 300 sekunder. Denna tid ökar dock om ett lag tar en flagga från ett annat lags bas. Om flera lag skulle ha samma poäng när tiden tar slut vinner det laget som först får ett poäng till, eller det lag som först tar en flagga från ett annat lags bas.

Kartan i spelet är även designad med målet att objekt kommer skymma sikten och att spelaren därför måste flytta sin enhet kring spelplanen för att alltid ha översikt över sin karaktär. Det sammanställda spelet presenteras i Figur 6.2.

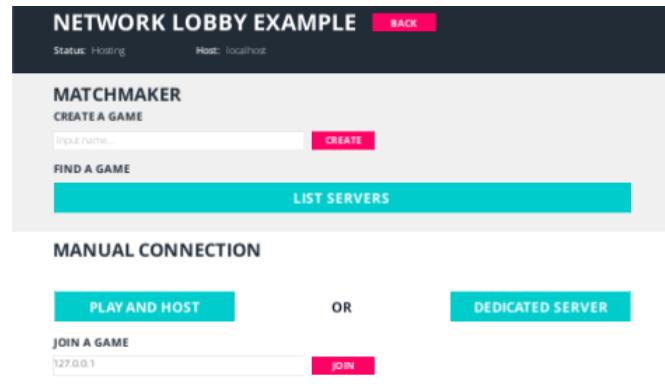
Spelet har även återkoppling i form av ljud och bilder. Olika ljud spelas upp för varje förmåga som används och när en flagga tas upp eller lämnas till sin bas. Det visas en bild för spelaren medan den har en flagga. En annan bild visas när spelarens karaktär blir förlamad och då spelas även ett ljud upp. Övrig återkoppling finns vid poäng förändring, se Figur 5.6 och när tid läggs till.



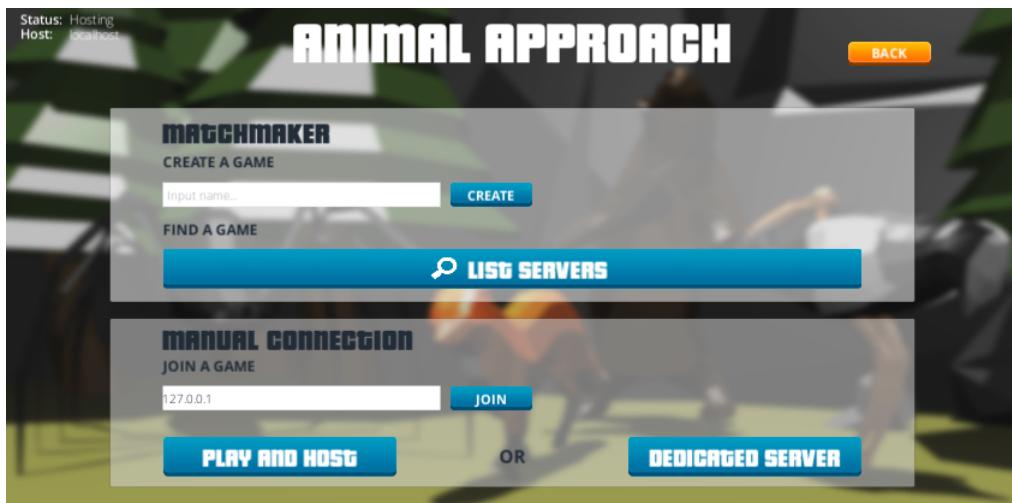
Figur 6.2: Det sammanställda spelet

6.2 Lobby och GUI

Unitys exempellobby användes som startpunkt för att bygga det slutgiltiga användargränssnittet som användes för flerspelarlägets menyer, se Figur 6.3. Menyn hade en praktisk layout och innehöll funktionaliteten gruppen ville ha, men behövde göras mer grafiskt tilltalande och passande till spelets tema.

Figur 6.3: *Unitys Lobby-exempel*

I Figur 6.4 visas den färdiga lobbyn som används i spelet. Panelerna har gjorts mindre och en bakgrundsbild har lagts till. Knapparna har skapats med en skugga som gör att användaren enklare förstår att de är tryckbara. Tysnitt har ändrats på titel, vissa knappar och dessutom har färgerna ändrats till att vara mer kontrastrika. Dessa ändringar gjorde att lobbyn relaterade mer grafiskt till speltemat och gav en bättre helhetskänsla.



Figur 6.4: Redigerad lobbymeny

Kapitel 7

Analys och diskussion

I detta kapitel kommer projektets olika moment att diskuteras och analyseras. Det som huvudsakligen kommer att analyseras är hur planeringen och de satta rutinerna har fungerat under projektets gång samt för och nackdelar med dessa som framkommit.

7.1 Metod

I denna kommer projektets utvecklingsmetoder att diskuteras. Arbetsprocessen för gruppen och hur det individuella arbetet har genomförts kommer också att diskuteras.

7.1.1 Arbetsprocess

Arbetet med utvecklingsmetoden *Scrum* har gått bättre än förväntat. Bättre i form av att det till en början av projekt kändes som *Scrum* var en metod som behövde mycket underhåll. Det visade sig senare att detta inte var fallet då det kom mycket naturligt efter någon veckas arbete med utvecklingsmetoden.

Arbetet i allmänhet skedde individuellt förutom vid mer komplicerade moment där parprogrammering förekom. Även fast arbete utfördes individuellt fanns ofta en kollega som arbetade inom samma del bredvid men med sitt eget moment. Resterande delen av gruppen arbetade dock i närheten vilket möjliggjorde att hjälpa med andra delar och komponenter fanns ifall det behövdes. Eftersom inte hela gruppen satt vid samma bord minskade irrelevanta frågor som utvecklaren kan lösa på egen hand. Utvecklaren frågade istället andra subgrupper om det var något som den inte kunde lösa på egen hand. Om en avvägning görs kring om det valda arbetssättet varit optimalt för att uppnå den bästa möjliga produkten är det svårt att komma till en slutsats. Detta på grund av att det har funnits många fördelar med att arbeta individuellt med kollegor i närheten över självständigt arbete. De fördelarna är framförallt kunskapsdelning och möjlighet att på ett enkelt sätt kunna fråga de andra gruppmedlemmarna om saker som relaterar till projektet. Problematik med detta arbetssätt var att dialoger inte alltid var effektiva. Detta i sig kanske inte berodde på logistiken utan rutinen kring hur kommunikation ska föras för utveckling. Slutligen väger dock fördelarna över nackdelarna då de kommer resultera i att mindre kommunikation kommer behövas via meddelandeapplikationer.

7.1.2 Möten

De möten som hölls i gruppen var framförallt de som utgick från utvecklingsmetoden *Scrum*, men det hölls även möten för att diskutera hur utvecklingen av enskilda delar skulle göras. Det *dagliga scrummötet* sköttes smidigt och var sällan längre än 5 minuter långt. Detta medförde att alla i gruppen hade koll på vad som skulle göras under dagen och hur den tidigare dagens utvecklingen hade gått. För att hålla nere tiden på det mötet hölls istället andra möten om någon utvecklingsdetalj behövde diskuteras. Problemet med dessa möten var att de ofta var informella vilket medförde att de inte blev väldokumenterade eller hade någon god struktur. Ett annat möte som hölls kontinuerlig under projektet var *sprintåterblicken*. Dessa mötena gick effektivt till och projektgruppen kom ofta till bra slutsatser kring hur föregående sprint gått. Det dokumenterades även på detta mötet så den informationen alltid fanns tillgänglig, men de lärdomar som tagits från föregående sprint glömdes ofta av när arbetet på nästa sprint påbörjades. Något som skulle kunna gjorts för att motverka detta är att ha ett kortare möte varje vecka där hur arbetet genomförs går igenom. Detta hade kunnat medföra att projektgruppen ständigt har de tidigare nämnda lärdomarna med sig. *Sprintåterblicken* och *sprintgranskningen* sattes ihop till ett möte för att minska den administrativa tiden som gick åt till att planera möten. Detta var i efterhand inte det bästa beslutet utan om ett ordentligt möte hade hållits för att gå igenom funktionaliteten som utvecklats, hade det givit projektmedlemmarna mer kunskap kring det de själva inte utvecklade. *Sprintgranskningen* valdes till en början att inte hållas som ett enskilt möte då det kändes viktigare att utveckla något väsentligt innan detta behövde göras.

När det gäller hur möten hölls som helhet var det korta möten som genomsyrade projektet och att möten ofta valdes bort för att främja utveckling. Det har visat sig att de lett till att det funnits luckor kring projektet som helhet och vad som borde göras under ett visst skede av projektet.

7.1.3 Versionshantering

Versionshanteringen som hade planerats inför projektet kunde inte genomföras på det sätt som var tänkt. Detta då *Git* och *Unity* har problem att samverka eftersom *Git* vill slå ihop alla filer. Detta blir ett problem med *Unitys META-filer*, som är en fil som innehåller ett unikt ID. Varje fil har en sådan fil kopplad till sig och är unikt för varje version av den filen. Det innebär att om två personer har en likadan fil är den medföljande META-filen till den olika hos de båda parterna. Dessa går därför inte att slå ihop med utomstående versionshanterings-programmet som då stöter på problem som inte går att lösa. Lösningen på detta var att istället byta programvara och använda *Unitys egna versionshantering*, *Unity Collaborate*. Då detta är utvecklat specifikt för *Unity* fungerar det även för META-filerna. Problem som detta medförde var att endast tre personer i projektgruppen hade tillgång till nycklar till *Unity Collaborate*. Detta eftersom *Unity* har satt en gräns på tre nycklar per projekt för deras gratisversion. För att få fler än dessa tre måste användarna betala för den fulla versionen av *Unity*. Lösningen som projektgruppen kom fram till var att dessa tre nycklar tilldelades tre par, där varje par fick skicka filer mellan varandra med hjälp av *Google Drive* för att sedan ladda upp till huvudgrenen via *Unity Collaborate*.

7.2 Resultat

Den slutgiltiga produkten fungerar på det sätt som förväntats med undantag för ett par saker. Först och främst skulle det behövas optimera *Vuforia* över nätverket så att spårningen blir mer stabil och därför även ger en bättre användarupplevelse. Det är även mycket småsaker som skulle behöva fixas innan spelet lever upp till de förväntningar som sattes i början av projektet. Under testning framgick det att det var en obalans mellan karaktärerna där Räv-karaktären var avsevärt bättre än de andra.

För att lösa denna obalans behöver Räv-karaktären få sin rörelsehastighet förminskad samt att de andra karakterna måste ses över. Genom användartesterna framgick det att lobbys gränssnitt inte var logiskt och lätt att förstå. För att förbättra lobbyn måste det vara enklare att identifiera sig själv i lobbyn, genom bakgrundsfärg och att kunna ange sitt namn innan spelaren går med i sektionen. Det behöver också förtydligas hur spelaren byter lag och karaktär samt att det är möjligt att göra detta. Detta skulle kunna uppfyllas för genom förklarande pop-up fönster under sin första spelomgång och att knapparna i sig är större. Det har lagts till en överhängande stenstruktur över hela spelplanen som hade avsikt att tvinga spelaren att röra sig mer. Efter användartesterna framgick det att det var något som också fungerade.

I nuläget kan användare skapa och hitta en lobby genom *Unitys* molntjänst. Molntjänsten har dock en begränsning på att enbart 20 användare kan använda tjänsten samtidigt. Denna begränsning kan utökas till flera användare mot betalning. I och med att spelet ska spelas lokalt är det därför möjligt att istället implementera *local network discovery* där en användare kan skapa en lobby på det lokala nätverket och det är möjligt att hitta alla tillgängliga lobbys i nätverket.

Baserad på teorin som varit tillgänglig och den kunskapen som fanns sedan tidigare, uppfyller den färdiga produkten förväntningarna som satts.

7.3 Spelets begränsningar

Spelet är inte för alla. Det finns en fysisk begränsning eftersom användaren måste röra sig och kolla från olika vinklar för att se vad som sker i spelet. Människor med fysiska handikapp kan därför uppleva att spelet är svårt att genomföra då det kräver viss motorik och rörlighet. Spelare kan även uppleva trötthet efter att ha sprungit runt bordet för att få den bästa vyn för tillfället.

7.4 Problem och motgångar

Det visade sig tidigt att den befintliga bekantskapen var otillräcklig för de nya momenten som nätverket och spårningen. Det var smidigt att komma igång med nätverket och spårningen men något vilket var en större motgång var det faktum att *Unitys* dokumentation för dess HLAPI. Den hade tydlig dokumentation för samtliga funktioner vilket var möjliga att överskrida och ifall överskridning gjordes hade utvecklaren full kontroll över funktionaliteten. Det medför att det går att skräddarsy beteendet efter behov. Problemet var att exekveringsordningen ej var dokumenterad. Lobbyspelaren valde karaktär i lobbyn, sedan skapades matchspelaren innan dess rekommenderande funktion för att ändra spelares inställningar och attribut. Eftersom ordningen ej var tillgänglig lades tid på att undersöka ordningen med olika hjälpmeddelande, vilket resulterade i ett mer tidskrävande arbete.

Ett problem vid implementationen av *Kudan* var att det inte var uppbyggt som *Vuforia*. När användaren flyttar sin enhet så rör sig inte den virtuella kameran utan istället rör sig markören beroende på translationen. En teoretisk lösning för att få den virtuella kameran att röra sig istället för markören är att en inverstranslation av markören läggs på *Kudans* kamera. Det leder till att kameran rör på sig istället för markören och att *Kudan* fungerar likt *Vuforia*. Detta är inget som hittills med att testa i projektet utan det fokuserades istället på att förbättra *Vuforia*.

Ifall en markörlösspårning hade använts hade det varit väldigt svårt att få användarnas kartor att vara på samma ställe. Spelet vill uppmuntra till en social integration och därför är det viktigt att kartan är på samma ställe för alla användarna, inte att deras kartor ligger på varsin del av bordet. En stor del av spelet blir att enheterna behöver vara på samma plats för att se en del av kartan från samma riktning.

7.5 Arbetet i ett vidare sammanhang

Hur produkten kommer påverka samhället är svårt att vara säker på innan den släppts eller blivit testad på bredare publik. Spelet kräver fysisk närvaro för att spelas och skapar därför en interaktion mellan mäniskor, vilket är en bristande upplevelse på nuvarande mobila enhetsmarknaden. Detta tillsammans med att spelet är underhållande, skapar en social och annorlunda upplevelse i vardagen.

Det är idag vanligt att genom röst eller chattfunktioner få en form av social interaktion i vardagen. Man kan regelbundet träffa nya individer genom spel över nätet och det kan vara tillräckligt. Det finns samtidigt en grupp användare som behöver den fysiska sociala interaktionen, vilket uppfylls av spel som Animal Approach. Där användarna tvingas fysiskt röra sig runt den gemensamma spelplanen vilket kan ge upphov till fysisk kontakt samtidigt som du spelar ett digitalt spel. En unik del i spelet är att kommunikationen kring det digitala spelet sker direkt, då alla spelare är i samma rum.

Det som inte finns på marknaden nu är en kombination av de äldre generationernas brädspel och de moderna teknologierna för immersiva miljöer och förstärkt verklighet. En kombination av dess kan ge en ny form av socialt umgänge där mäniskor av alla åldrar kan fysiskt interagera med varandra genom sina mobila enheter. Detta kan vara ett framtida arbetsområde för immersiva miljöer och förstärkt verklighet.

Kapitel 8

Slutsatser

Detta spel utspelas på en gemensam spelplan som delas av alla spelare. Spelplanen kan ses genom spelarnas mobila enhet som använder sig av enhetens kamera för att identifiera en markör. På denna markör projekteras spelplanen och genom att fysiskt flytta enheten kommer olika delar av planen att ses.

- Hur kan design av ett AR-spel motivera en spelare att röra sig i rummet där spelet befinner sig, för att på så sätt få dem att socialt interagera mer med varandra?

Den genererade terrängen beskriven i sektion 5.5.5 fyller ut spelarvärlden men objekten agerar även som visuella hinder. Genom att ställa sig högt upp kan en överblick fås, men då täcker trädens överhängande grenar synfältet samt den överhängande stenstrukturen. En låg vinkel kan också tas, men då kan stenar och återigen träden blockera synfältet. En spelare motiveras alltså att justera sin synvinkel när den spelar ifall den ska se vad som sker. Detta kan vara små justeringar men det är även justering genom att gå runt spelplanens yta. De visuella hindren ökar behovet av att röra sig och det i sin tur leder till att spelarna i större utsträckning kommer behöva flytta sig och trängas med varandra, vilket leder till att de socialt interagerar mer med varandra. Svaret på frågan är att skapa objekt och spelfunktioner som belönar spelaren att röra sig istället för att stå still.

- Hur kan en handhållen enhets fysiska koordinater översättas till koordinater i en virtuell spelplan, för att på så sätt visa en viss del av spelplanen beroende på enhetens position?

En handhållen enhet kan bygga upp en virtuell-miljö på två olika sätt, genom spårning med markör eller utan. Utan markör kan miljön byggas upp när och var som helst då den inte behöver något mer än verkliga ytor. En spårning som utgår från en markör kräver att enhetens kamera ser en markör för att den virtuell-miljö ska skapas. Spårningen i projektet som har använts är en markörspårning. Markör används eftersom att alla spelare ska se världen på exakt samma position, det ska spelas på samma bord. När kameran hittar markören beräknas avståndet till markören i den verkliga världen och transformeras om till den virtuella med hjälp av *Vuforia*. *Vuforia* är ett AR-spårnings verktyg som är till för AR-spelutveckling i *Unity*. När markören är hittad och enhetens plats finns i den virtuella miljön styr enheten vad spelaren ser i spelet. Vad spelaren ser beror på hur enheten vinklas, roteras eller flyttas. I det virtuella rummet ligger kartan ovanpå en låst markör medan det är kameran som transformeras runt den, precis som i verkliga världen. När den virtuella kameran ser en del av markörer syns även en del av spelplanen vilket visas på spelarens skärm.

- Hur ska karaktärerna designas så att alla är unika med olika styrkor och svagheter så att de passar i ett fånga flaggan- spel i en AR-miljö, för att på så sätt tilltala barn från 7 år och uppåt?

I underrubriken [5.6.1](#) diskuterades om vilken karaktärsmodell som valdes att använda i spelet. Modellen bygger på en uppsättning karaktärer som de flesta spel idag, på ett eller annat sätt, har rötter till. På grund av att modellen har blivit en generalisering i spelindustrin och använd sedan 1974 [[13](#)], hjälper det användaren att intuitivt veta hur varje karaktärerna i spelet borde vara, innan karaktären ens har blivit testad. För att uppnå denna förutsägbarhet analyserades hur djuren är till sin natur, och förmågor, för-, och nackdelar kunde sättas utifrån detta.

Karaktärer och modeller är också något som har designats på ett sätt som ska passa in i helheten. Valet av *Low Poly*-grafik har hjälpt till att få en minimalistisk design som inte bara passar sig till den yngre publiken, utan även till resten av målgruppen.

Allt detta medför att en säker karaktärsmodell har valts och en design som är estetiskt tilltalande och passar för alla användare. Detta leder till att karaktärerna som har tagits fram kan ge en känsla av att de är lätt att förstå, men också svåra att bemästra.

De slutsatser som kan dras kring hur arbetet med projektet har gått grundar sig mestadels i att det är en projektgrupp med begränsad erfarenhet som har utfört projektet. Nu i ett slutskede av projektet är det lätt att förstå varför vissa utvecklingsmetoder är utformade som de är. I början av projektet var det svårt att förstå vikten av väl testade teorier och dokumentation kring systemutveckling. Med hjälp av de erfarenheter som fästs under projektets gång har projektgruppen förstått att ju fler delar ett projekt består av desto viktigare blir struktur och rutiner.

Litteraturförteckning

- [1] PEGI, *Two levels of information as a guide: The pegi age labels*, PEGI, 2018, hämtad: 2018-05-20
<https://pegi.info/>
- [2] University of Michigan *Fast-Paced Multiplayer (Part I): Client-Server Game Architecture* Eric Cronin Burton ,Filstrup, Anthony Kurc , 2011-05-04, hämtad: 2018-09-03
<http://www.eecg.toronto.edu/~ashvin/courses/ece1746/2003/reading/cronin-umtr01.pdf>
- [3] Gao, Qing Hong, et al, *A Stable and Accurate Marker-less Augmented Reality Registration Method*, IEEE, 2017-12-01
<https://ieeexplore.ieee.org/abstract/document/8120297/>
- [4] Unity, *Tanks!!!*, Unity Technologies, 2017-07-20, hämtad: 2018-05-08
<https://assetstore.unity.com/packages/essentials/tutorial-projects/tanks-reference-project-80165>
- [5] Shari Lawrence Pfleeger och Joanne M. Atlee, *Software Engineering, Fourth Edition, International Edition*, Pearson 2010
- [6] Ken Schwaber och Jeff Sutherland, *Scrumguiden*, Scrum.org och Scruminc, 2013-07, hämtad: 2018-05-05
<http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-SE.pdf>
- [7] Kenji Hiranabe, *Modeling in the Agile Age*, change vision inc, 2014-10-07, hämtad: 2018-02-05
<http://cdn.astah.net/resources/modeling-in-the-agile-age.pdf>
- [8] Unity Technologies, *Dictionary*, Unity Technologies, 2018, hämtad: 2018-05-09
<https://unity3d.com/public-relations>
- [9] Unity, *Vuforia*, Unity, 2018-04-30, hämtad: 2018-05-07
<https://docs.unity3d.com/Manual/vuforia-sdk-overview.html>
- [10] Kudan, *KudanSLAM*, Kudan computer vision, 2018, hämtad: 2018-05-30
<https://www.kudan.eu/kudanslam/>
- [11] Unity, *The Multiplayer High Level API*, Unity technologies, 2018-04-30, hämtad: 2018-05-09
<https://docs.unity3d.com/Manual/UNetUsingHLAPI.html>
- [12] Microsoft, *Dictionary*, Microsoft, 2018, hämtad: 2018-05-09
[https://msdn.microsoft.com/en-us/library/xfhwa508\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/xfhwa508(v=vs.110).aspx)
- [13] Gerald Voorhees, *The Character of Difference: Procedurality, Rhetoric, and Roleplaying Games*, University of Waterloo , 2009-11-01, hämtad: 2018-05-02
<http://gamestudies.org/0902/articles/voorhees>

Appendices

Bilaga A

Arbetsfördelning

A.1 Jacob

Har agerat produktägare under projektet. Produktägarens uppgift har varit att hålla all kontakt med kund. Jacob har även arbetat med att modellera baser och miljömodeller till spelplanen. Dessutom har han skapat ett system som slumpmässigt genererar ut terrängen för spelplanen i början av varje omgång. Skapat förmågan "Hook" och ett system som ger varje förmåga en knapp och nedkylningstid. Gjorde även limiterad tidsförmåga som teleporterade karaktären till en slumpvald bas.

A.2 Lucas

Lucas har varit *Scrummästare* under projektet. *Scrummästarens* uppgift har varit att hålla i *Scrummöten* varje morgon. Lucas har arbetat med AR-spårning. Implementerat och optimerat *Vuforia* till-sammans med Erik. Testat och arbetat med *Kudan* som en alternativ spårning till *Vuforia*. Dessutom utvecklat styrning av karaktären samt vissa av karaktärernas specialförmågor.

A.3 Erik

Har i samverkan med Lucas jobbat med AR-spårning och därfor implementerat och optimerat *Vuforia*. Ansvarade för spelmarkörerna och att de fungerade under projektets gång samt databasen för dessa. Skapade även ett Power-Up system för tillfälliga uppdateringar i spelet och skapade en Power-Up som ökar spelarens hastighet i en viss tid. Har även ansvarat för sammanställningen av projektets alla delar, detta till viss del med hjälp av Julius. Var även versionshanterings-ansvarig under projektets gång.

A.4 Viktor

Arbetade med att modellera och animera alla karaktärer, samt skapandet av spelkartan. Har även arbetat med styrning av karaktärer och koppling mellan joystick och animation. Utvecklade även ett mer grafiskt tilltalande användargränssnitt i lobbyn genom att byta ut knappar och ändra layout. Viktor har också arbetet med skapandet av karaktärsvals-menyn och synkronisering av karaktärsval över nätverk tillsammans med Julius.

A.5 Mikael

Som utvecklare i projektet har jag arbetat med grundläggande spelmekanik som att kunna ta upp och lämna en flagga till en bas och har även skapat ett poängsystem. Jag har även arbetat med att hålla alla klienter synkroniserade över nätverk vilket innebär bland annat att spelkartan ska se lika ut för alla spelare och att poäng uppdateras och visas korrekt för alla spelare. Ytterligare har jag arbetat med björnens speciella förmåga och den karaktär-gemensamma förmågan samt när en match ska avslutas.

A.6 Julius

Primärt ansvar för att synkronisera funktionalitet vilket behöver fungera över nätverk. Alltså att överföra funktionalitet vilken fungerar i ensamt spelarläge samt utan nätverk till att dess beteende synkroniseras över nätverket och klienter för flera spelare samtidigt.

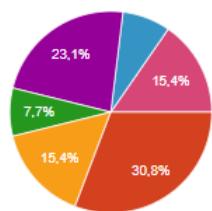
Bilaga B

Användartester

Detta är en sammanställning på användertesterna som gjordes i projektet. Sammanlagt var det 13 personer som testade spelet och fick svara på följande frågor:

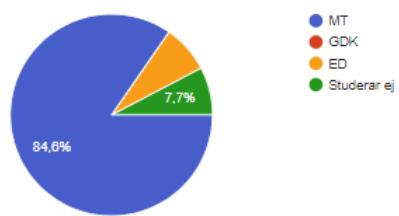
Vad har du för ålder?

13 svar



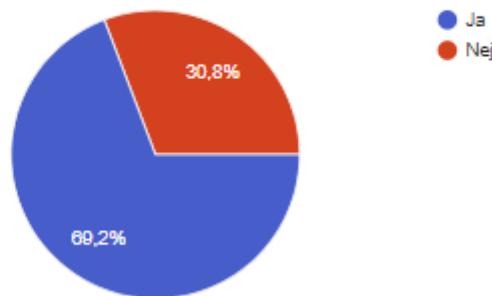
Vilket program går du?

13 svar



Har du någon erfarenhet av AR-spel tidigare?

13 svar



Vad är ditt första intryck av startskärmen(Lobbyn)?

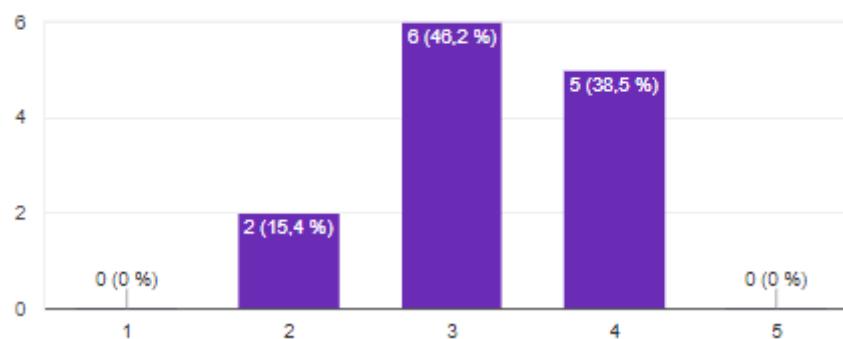
13 svar

Fin miljö, vet vad man ska göra
Tydligt var man ska trycka i början men i create game lobbyn så är det inte lika tydligt hur man byter lag samt byter karaktär.
Fin! Nice med en about-knapp. Förstår hyfsat vad man ska göra med.
Väldigt fin main menu bakgrund men musiken var inte det bästa
Lite rörig, men blev bättre när man fick häng på det.
Snyggt, ser proffsigt ut
Fin
Väldigt trevlig startskärm, mycket fina färger
Vacker, fin miljö.
Startskärmen imponerande. Lobby lite förvirrande vem man är och vad man har för lag samt inte självklart att det går att byta karaktär
Oklart hur man snabbt kommer igång till en match

Följande stapeldiagram visar hur användarna upplevde spelet på en 5-gradig skala. Siffran ett motsvarar här inget samtycke till upplevelsen eller en dålig upplevelse och siffran fem motsvarar fullt samtycke eller en mycket bra upplevelse. Siffran tre motsvarar i sin tur en neutral åsikt.

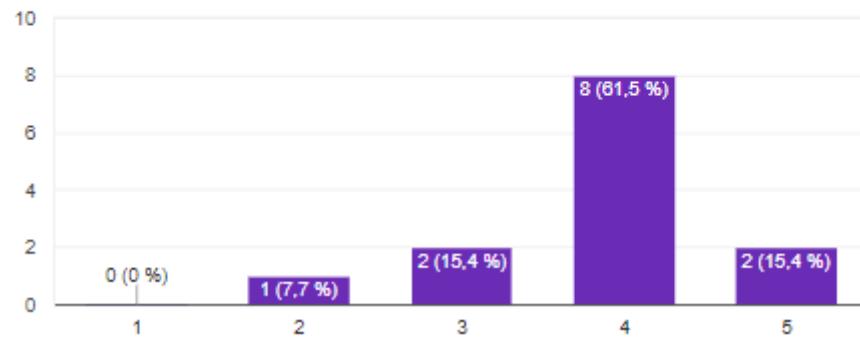
Hur tycker du kamerastyrningen är?

13 svar



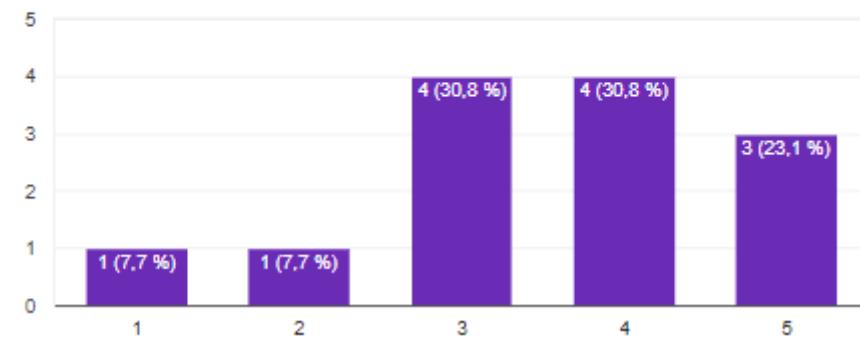
Hur tycker du styrningen av karaktären känns?

13 svar



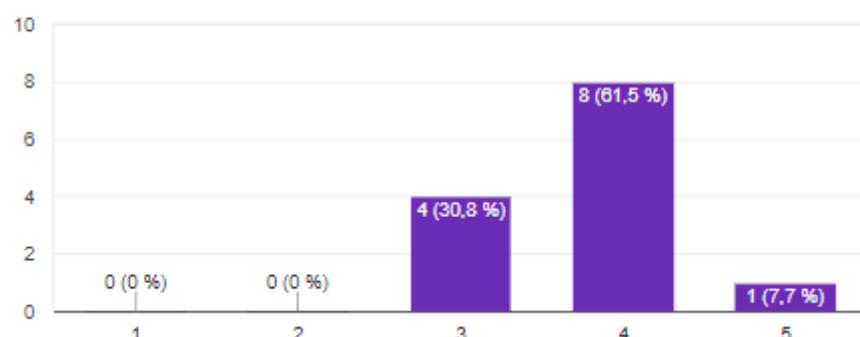
Kände du ett behov av att röra på dig fysiskt för att kunna spela spelet och se din karaktär?

13 svar



Var det ett kul spel?

13 svar



Har du något förbättringsförslag?

8 svar

Det var lite otydligt att man skulle ta flaggan i mitten och hur man kom upp på mittenplattformen. Knappen för power up var tydlig. Det var lite oklart hur man skulle göra för att ta flaggan av den andra spelaren.

Ingen grön lagfärg, inte hamna under marken

Kanske ta bort stenarna i "Taket" och minska antalet objekt i scenen.

Fixa touchen så man inte behöver klicka så mycket.

Svårt o veta vad uppdraget eller själva målet var.

Mindre hackigt

Ändra styrningen till en som känns mindre klumpig. Gör lobbyn tydligare.

Svårt för en nybörjare att sätta sig in i spelet och hitta sin karaktär även med hitta funktionen, skulle kunna implementera en ytterligare hitta funktion som är aktiv när karaktären ej är i synfältet.