Linköpings universitet | Insitutionen för teknik och naturvetenskap
Kandidatarbete | Medieteknik
Vårterminen 2020 | LIU-ITN-TEK-G-19/013—SE

Potion Party

Fleranvändarspel med fysiska reglage

Anna Furugård
Carl Håkansson
Adam Lindfors
Alexander Uggla
Carmen Yu

Examinator: Daniel Jönsson

LINKÖPINGS UNIVERSITET

Linköpings universitet
SE-581 83 Linköping
013–28 10 00, www.liu.se

Sammanfattning

Den här rapporten beskriver arbetet för ett datorspel som använder sig av en specialgjord kontroller istället för en konventionell kontroller som till exempel datormus och tangentbord. I projektet skapades ett fleranvändarspel som använder sig av icke-konventionella kontroller. Syftet med arbetet var att undersöka och jämföra vilka tekniska principer som utvecklingsgruppen skulle tillämpa i projektet, samt studera hur ett spel utvecklas med ett agilt arbetssätt.

Den agila arbetsmetoden som användes i arbetet var scrum. Projektgruppen delades upp i olika scrum-specifika roller samt ansvarsområden för att arbetet skulle fortskrida enligt den framtagna tidsplanen. Samtliga utvecklare har arbetat med alla ansvarsområden inom projektet. Projektgruppen har haft sprintplaneringsmöten, dagliga scrum-möten och sprintutvärderingsmöten enligt scrum-modellen.

Det undersöktes också hur en icke konventionell kontroll borde utformas och hur den påverkar spelupplevelsen. Eftersom spelupplevelsen inte skulle påverkas negativt av den kontroll som tagits fram lades ett stort fokus på hur den skulle utformas så att den hade god affordans. Detta innebar att kontrollens design följde konvention i största utsträckning och ett försök att ge kontrollens reglage god affordans till deras funktioner i spelet gjordes. Spelet har skapats i Unity som vidare integrerats med Arduino för att koppla de fysiska reglagen till systemet. Spelarna skulle med användning av reglagen kunna interagera med spelmiljön samt styra spelkaraktärerna. Spelarna skulle utan genomgång kunna spela spelet och vid första anblick förstå hur reglagen skulle användas.

Slutprodukten är ett spel för två spelare som använder kontroller bestående av fysiska reglage som bestod av knappar, spakar och en ratt. Spelets mål går ut på att de två spelarna genom samarbete skapar trolldrycker av ingredienser som finns placerade i nivån. Spelarna ska gemensamt skapa så många trolldrycker som möjligt på bestämd tid.

För att ta reda på om de mål som satts upp blivit uppnådda analyserades resultatet. De teorier som undersöktes jämfördes också med de upptäckter som gjordes under arbetet för att se om de stämde överens. Vidare diskuterades hur frågeställningarna besvarats utifrån projektets utvecklingsarbete och resultat.

Användartester visade att en kontrollpanel för icke-konventionella kontroller bör utformas så att den tar hjälp av konventioner om styrdon och av användares tidigare erfarenheter. Genom att utforma reglagen specifikt efter funktioner i spelet förstår användaren lättare hur reglagen ska användas.

Innehåll

Sammanfattning **i**

Figurer **v**

Tabeller **vi**

1 Inledning **1**

1.1 Bakgrund 1

1.2 Syfte 1

1.3 Spelidé 2

1.4 Frågeställningar 2

1.5 Avgränsningar 3

2 Relaterat arbete **4**

2.1 Fysiska reglage och kontroller 4

2.2 Arduino 6

2.2.1 Programmering och kommunikation med Arduino-kretskort 6

2.2.2 Koppla reglage 7

2.3 Icke-konventionella kontrollers effekt på spelupplevelsen 7

2.4 Lärande genom spel 7

3 Utvecklingsprocessen för fleranvändarspel som använder fysiska reglage **9**

3.1 Utvecklingsmetodik 9

3.2 Tidsplan 9

3.3 Mötesprinciper 11

3.3.1 Scrum-möten 11

3.3.2 Arbetsmöten 12

3.4 Dokumentationsprinciper och rutiner 13

3.5 Organisation 13

3.5.1 Ansvarsområden 13

3.5.2 Organisativa roller 14

3.6 Kravhantering och -spårning . . . 15
3.6.1 Användartester . . . 15
3.6.2 Kodgranskning . . . 16
3.7 Versionshantering, -system och rutiner . . . 16

4 Tekniska lösningar 17
4.1 Grundläggande, initiala krav och systembegränsingar . . . 18
4.2 Standarder . . . 18
4.3 Målplattform . . . 18
4.4 Fysiska reglage . . . 18
4.5 Arduino till Unity . . . 19
4.6 Spelgrafik . . . 19
4.7 Utvecklingsmiljö . . . 19
4.7.1 Spelmotor . . . 20
4.7.2 Modellering- och bildbehandlingsprogram . . . 20
4.7.3 Arduino IDE . . . 20

5 Resultat 21
5.1 Tekniska principer . . . 21
5.2 Spelet . . . 21

6 Analys och diskussion 24
6.1 Metod . . . 24
6.2 Resultat . . . 25
6.3 Arbetet i ett vidare sammanhang . . . 25

7 Slutsatser 26
7.1 Hur icke-konventionella kontroller kan användas för att göra ett spel engagerande . . . 26
7.2 Hur en kontrollpanel för icke-konventionella kontroller bör utformas för att användarna ska förstå deras funktionalitet . . . 26
7.3 Hur det säkerställs att användare vet hur de ska kombinera ingredienser för att skapa trolldrycker . . . 27
7.4 Förbättringsmöjligheter . . . 27

Litteraturförteckning 29

A Användartester 30

B Arbetsfördelning 31
B.1 Alexander Uggla . . . 31
B.2 Anna Furugård . . . 32

B.3 Carl Håkansson 32
B.4 Carmen Yu 32
B.5 Adam Lindfors 32

Figurer

1.1 Enkel visualisering av spelmiljö och viktiga spelobjekt 2

2.1 En Xbox-kontroller 4
2.2 En knapp av märket Adafruit 5
2.3 En styrspak av märket Adafruit 5
2.4 Ett rotationsreglage 6
2.5 Ett Arduino Nano-kretskort 6
2.6 Färgblandningen CMY och RGB 8

3.1 Tidsplanen för projektet. 11
3.2 Kravhanterning- och spårning med användning av Trello 15

4.1 Spelets systemarkitektur och användarnas relation till systemets komponenter 17

5.1 En skärmdump av spelet 22
5.2 Kontrollpanel av skokartong, fysiska reglage och Arduino. 22
5.3 Insida av kontrollpanel där reglagen är kopplade till Arduino 22
5.4 Receptplansch för att brygga trolldrycker 23

Tabeller

3.1 Utvecklarnas namn, kontaktuppgifter och roller 13

4.1 Hårdvara för reglage och Arduino 19

4.2 Programvaror som användes under utvecklingsfasen. 20

Kapitel 1

Inledning

Ett fleranvändarspel med fysiska reglage ska utvecklas för att ställas ut på ett museum för visualisering och programmering. Spelidéen tas fram av arbetsgruppen tillsammans med en intern kund på ITN, Linköpings Universitet med målet att resultatet ska bli ett utställningsprojekt. Arbetet ska organiseras med en projektplan.

Rapporten beskriver utvecklingsförloppet till resultatet och avslutas med att diskutera samt motivera valen som har gjorts. För att förstå innehållet i rapporten förutsätts att läsaren har förkunskaper likvärdiga med en student som läser sjätte terminen på Civilingenjörsprogrammet för Medieteknik på Linköpings Universitet.

1.1 Bakgrund

Teknik kan används för att skapa engagerande och lärande spel för barn och unga vuxna. Spel kan utöver vara en nöjesprodukt även ha ett inlärningssyfte. Visualiseringscenter C har emellanåt utställningar som riktar sig till åldersgruppen barn och unga vuxna. Spelet ska skapas för att kunna förmedla kunskap på ett engagerande sätt till denna åldersgrupp.

Kundens inledande krav är att spelet ska använda sig av fysiska reglage samt vara ett fleranvändarspel. Spelet ska köras på en persondator som kopplas ihop med reglagen. Reglagen består av fem knappar, två styrspakar och två rattar. Spelet ska styras med användning av reglagen och minst två spelare ska behövas för att klara spelmål och vinst.

Arbetsgruppen har som mål att spelet ska lämpa sig för en utställningsmiljö och det är därför viktigt att spelet är lättförstått. Utifrån kraven kan en spelidé och projektplan tas fram. I systemutveckling är det viktigt att det finns en tydlig plan för hur projektet ska genomföras för att få en bra slutprodukt.

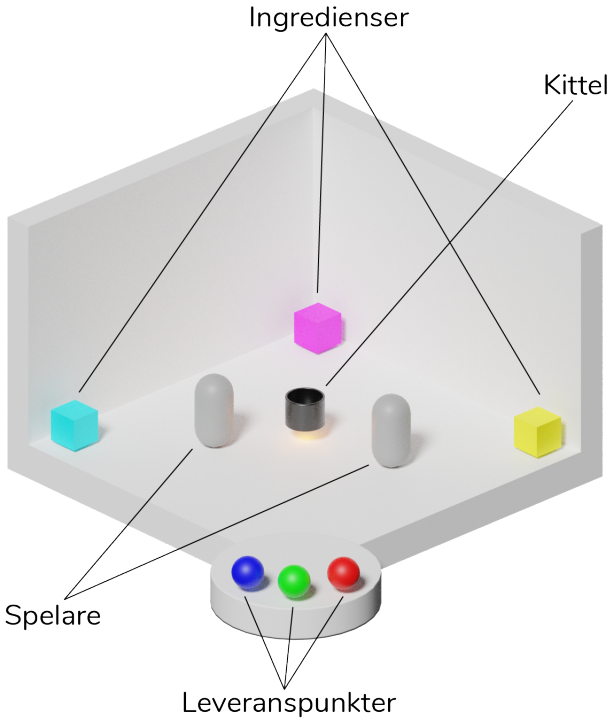
1.2 Syfte

Syftet med arbetet är att undersöka och jämföra vilka tekniska principer som ska tillämpas för att integrera icke-konventionella reglage i ett spel, samt studera hur ett spel utvecklas på ett agilt arbetssätt.

Slutprodukten ska bli ett samarbetsspel för två personer och ska presenteras på Visualiseringscenter C. Spelkaraktärerna i spelet ska vara beroende av varandra och behöva hjälpas åt för att lösa vissa problem. Spelet ska visa vikten av samarbete och kommunikation, samt lära ut kopplingen mellan *RGB*- och *CMYK-färger*.

1.3 Spelidé

Spelidén är att en häxa och en trollkarl äger en butik gemensamt där de säljer hemmagjorda trolldrycker. Spelkaraktärerna ska genom samarbete hämta ingredienser, blanda dessa till en trolldryck och sedan leverera dem. Varje typ av ingrediens har en grundfärg i CMY-färgrymden, och varje trolldryck har en grundfärg i RGB-färgrymden. Par av CMY-grundfärger kan blandas till grundfärger i RGB-färgrymden, och samma färgkombinationer används för att skapa trolldryckerna i spelet. I spelnivån finns hela tiden tre stycken leveransugglor, som alla har en RGB-grundfärg. För att få poäng ska spelarna para ihop trolldrycker och ugglor med samma färg. När en uggla får en trolldryck med rätt färg flyger den iväg med trolldrycken och en ny uggla med en slumpmässig RGB-grundfärg flyger in. För att få mycket poäng skall dessa steg upprepas så många gånger som möjligt innan tiden är ute. För att se hur lång tid som har gått och hur lång tid som är kvar finns en klocka i spelmiljön med en enda visare. Då denna visare roterat ett helt varv är tiden ute och spelet är över.



Figur 1.1: Enkel visualisering av spelmiljö och viktiga spelobjekt

Figur 1.1 visar en konceptuell visualisering av spelmiljön och de viktigaste spelobjekten. Ingredienser förs av spelare i lämpliga kombinationer till kitteln där de kombineras till trolldrycker. Då trolldrycken är färdig tas den från kitteln till en uggla av samma färg och lämnas av vid den leveranspunkten. Fler trolldrycker kan tillverkas om spelare samarbetar vilket leder till att de får mer poäng. Detta uppmuntrar till samarbete mellan spelare.

1.4 Frågeställningar

Följande frågeställningar besvaras i rapporten:

- Hur kan icke-konventionella kontroller användas för att göra ett spel engagerande? Spelupplevelsen ska förhöjas av de specialgjorda kontrollerna och får inte påverka användarna på ett negativt sätt.

- Hur bör en kontroller utformas så att en användare förstår dess funktionalitet? Användaren ska förstå vilken kontroller som styr vilken karaktär vid första anblick och utan instruktioner.
- Hur säkerställs det att användare vet hur de ska kombinera ingredienser för att skapa trolldrycker? Användarna ska förstå hur ingredienser kombineras för att skapa trolldrycker på mindre än en minut utan att tidigare ha spelat spelet.

1.5 Avgränsningar

- Arduino-kretskort och reglage är givna av kunden och ska användas i projektet.
- Hela projektet ska utvecklas inom tre månader, där fem utvecklare arbetar tre dagar i veckan, åtta timmar per dag.
- Spelet ska utvecklas för två spelare.
- Spelet ska utvecklas specifikt för att köras på en Windows-dator.
- Målgruppen för spelet kommer vara 13 till 30 år gamla personer.

Kapitel 2

Relaterat arbete

Kapitlet beskriver tidigare arbete och forskning som är relevant för utvecklingsgruppens arbete. Utvecklingsgruppen har valt att fokusera på hur icke-konventionella kontroller påverkar spelupplevelser och hur spel kan användas i pedagogiska syften.

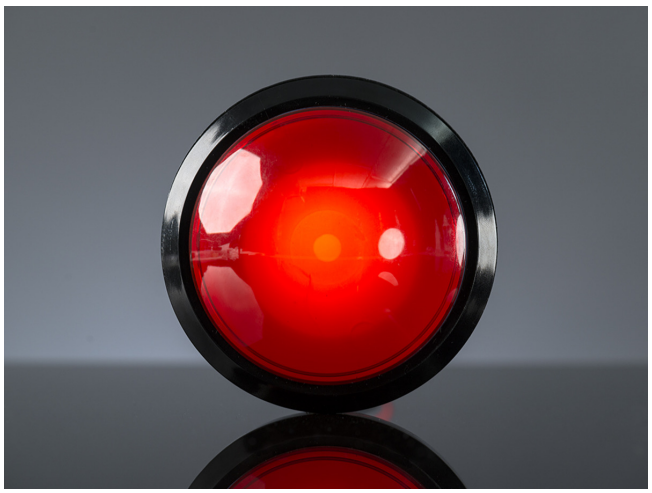
2.1 Fysiska reglage och kontroller

I projektet används fysiska reglage för att utföra gärningar i spelet. Med fysiska reglage menas knappar, spakar, rattar eller liknande styrdon. Flera fysiska reglage kan kombineras till en kontroller. Ett exempel på en konventionell kontroller är en Xbox-kontroller som visas i figur 2.1. Denna kontroller definieras vara konventionell eftersom den används av många spelare för många olika spel[20] och har liknande utformning som många andra spelkontrollers på marknaden. En icke-konventionell kontroller definieras vara en kontroller specifikt designad för ett program med reglage valda specifikt för programmets syften.

En kontroller är ett användargränssnitt och kan därför använda sig av designprinciperna inom användbarhetsteori som beskrivs av Norman D. i *Design of Everyday Things* [9]

Figur 2.1: En Xbox-kontroller

Figur 2.2 visar ett knappreglage. Knappen är digital och består av en strömbrytare som aktiveras när den är intryckt och har en diod som kan få den att lysa.

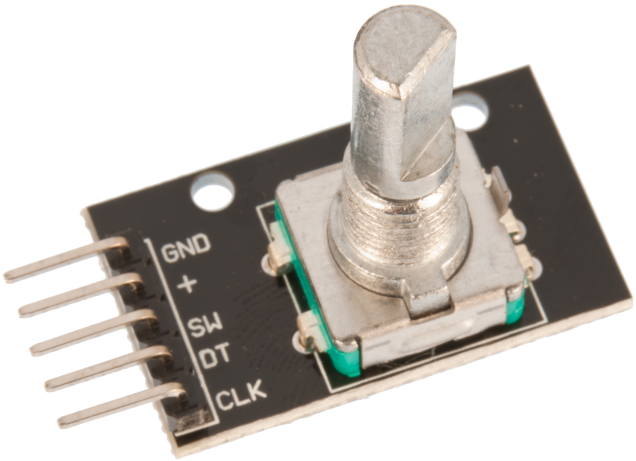


Figur 2.2: En knapp av märket Adafruit

Figur 2.3 visar ett analogt styrspaksreglage. Spaken består av fyra strömbrytare som aktiveras när spaken förs i de fyra olika riktningarna. Spaken kan roteras fritt, så det är möjligt att aktivera två närliggande strömbrytare samtidigt.

Figur 2.3: En styrspak av märket Adafruit

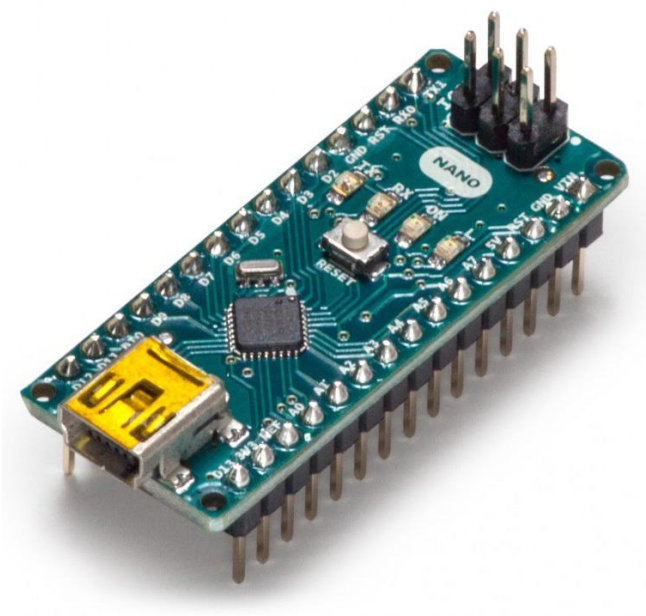
Figur 2.4 visar ett rotationsreglage. Reglaget skickar digitalt en positiv signal när den vrids åt ena hållet, och en negativ signal när den vrids åt andra hållet. Reglaget kan också tryckas in som en knapp som består av en strömbrytare.



Figur 2.4: Ett rotationsreglage

2.2 Arduino

Arduino är en produktserie av kretskort, kretskortstillbehör och mjukvara som utvecklas av Arduino AG [19]. Ett arduino-kretskort kan programmeras för att läsa av signaler, utföra beräkningar och skicka ut signaler. Det finns olika typer av Arduino-kretskort som har olika utseende, prestanda och funktionalitet. Ett Arduino Nano-kretskort kan ses i figur 2.5.



Figur 2.5: Ett Arduino Nano-kretskort

2.2.1 Programmering och kommunikation med Arduino-kretskort

För att programmera och kommunicera med kretskortet så måste det vara kopplat till en dator via kretskortets USB-port (porten med en gul film på sig längst ner i figur 2.5). Kretskortet försörjs också med ström över denna koppling.

Det finns flera verktyg som kan användas för att programmera Arduino-kretskort, men Arduino IDE

är rekommenderad av Arduino AG [15]. Arduino-kretskort kodas i ett Arduinospecifikt språk som har liknande syntax som C++ men som kompileras till Java när det körs på kretskortet.

För att få information från Arduino-kretskort kan datorn läsa av kretskortets konsollogg.

2.2.2 Koppla reglage

Arduino-kretskort kan hantera både analoga och digitala signaler. Externa komponenter kopplas med sladdar till kretskortets stift, vilket är metalbitarna som sticker ut under kretskortet (se figur 2.5). Analoga komponenter kopplas mellan ett analogt stift (A0-A7) och ett jordstift (GND). Digitala komponenter kopplas mellan ett strömkällestift (V5), ett jordstift (GND) och ett digitalt stift (D2-D12). Flera sladdar kan kopplas till samma jord-/ strömkällestift. En resistor bör användas för digitala komponenter mellan komponenten och jord- eller strömkällestiftet för att undvika att kretskortet överbelastas. Om kretskortet överbelastas tar det inte skada men ett brus kommer till i signalerna.

2.3 Icke-konventionella kontrollers effekt på spelupplevelsen

En spelupplevelse beror på många olika faktorer som skärmstorlek, ljudkvalité och speldesign[8]. För att kontrollen inte ska påverka spelupplevelsen negativt är det viktigt att kontrollen känns naturlig och inlärningskurvan är så liten som möjligt[10]. Kontrollens knappar måste därför ha god *affordans* och vara väl *mappade* till funktioner i spelvärlden. Det innebär att det är tydligt vilken knapp som gör vad genom att bara titta på den. Till exempel används kontrollens spak till rörelse på grund av konvention.

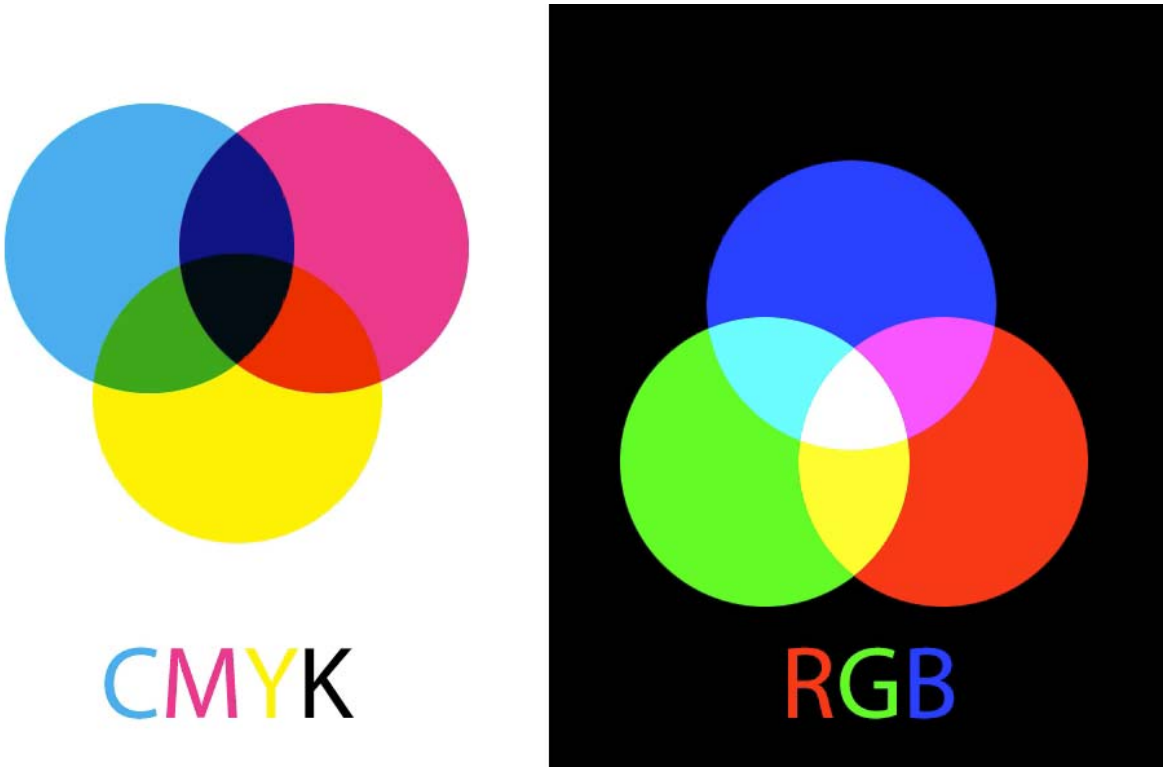
Genom att designa en kontroll som har god affordans och designad så att den endast innehåller de funktioner som krävs för spelet kan spelupplevelsen förbättras[10]. Eftersom det är en kontroller som är designad för just detta spel innebär det också att ingen användare har tidigare erfarenhet av kontrollen.

2.4 Lärande genom spel

I *Video games and learning: Teaching and Participatory Culture in the Digital Age* förklarar Kurt Squire [6] om hur vissa spel kan lära ut kunskaper på ett effektivt och engagerat sätt. Spel är mestadels kreativa och underhållande, vilket gör att spelaren blir mer engagerad och absorberar in mer information. Spel är inte bara lärorik, utan ger även motivation för framtida jobb och mål. Exempel på sådana spel är: *Guitar Hero, Simcity, Cuckoo Time* och *Brain Age*. Idag finns många applikationer och spel med underhållning och lärorika ändamål. Det är alltifrån simpla applikationer som lär ut multiplikationstabellen till spel med svåra pussel. I gruppens spel skapar spelarna olika trolldrycker beroende på vad för sorts kombination av ingredienser som används. Ingredienser har färger som representerar *CMY* och trolldryckerna i sin tur har färger som representerar *RGB*. CMY står för: **C**yan, **M**agenta, Gul (**Y**ellow). RGB står för: **R**ött, **G**rön och **B**lå.

RGB och CMY är varandras motsatser. RGB är en additiv färgblandning, vilket betyder att ju mer färg som läggs till, desto ljusare blir det[5]. CMY är en subtraktiv färgblandning, vilket betyder att ju mer färg som läggs till, desto mörkare blir det.

Spelets mål är att vara kul för spelarna samtidigt som de lär sig kopplingen mellan RGB och CMY. Eftersom färgblandningarna är varandras motsatser så kan färger från RGB blandas för att få färger från CMY och tvärtom. Till exempel om röd och blå kombineras så fås färgen magenta, samt om magenta och gul kombineras så fås färgen röd. Figur 2.6 visar kopplingen mellan CMY och RGB.



Figur 2.6: Färgblandningen CMY och RGB

Kapitel 3

Utvecklingsprocessen för fleranvändarspel som använder fysiska reglage

För varje projekt ser utvecklingsprocessen olika ut. Ett projektets framgång beror till stor del på dess processer är väl beskrivna och tydliga för utvecklarna. Detta kapitel går igenom utvecklingsprocessen för ett fleranvändarspel med icke-konventionell kontroller.

3.1 Utvecklingsmetodik

I utvecklingen av spelet användes den agila arbetsmetoden *scrum* för att organisera projektet. Fördelarna med scrum är enligt Schwaber K. och Sutherland J. i *The Scrum Guide* att:

- Arbetsprocessen är adaptiv, vilket gör det enkelt att ändra i utvecklingen eller produkten utifrån tester.
- Ramverket är lätt att förstå och kräver inte mycket förarbete innan projektet kan starta.
- Arbetsprocessen är iterativ, vilket gör det möjligt att bestämma resursförelning till olika utvecklingsmoment under projektets gång. Ifall ett område kräver mer utveckling är det lätt att delegera resurser till det.

Clint K. skriver i *Agile Game Development with Scrum* [3] att Scrum är en ”*near perfect match*” för spelutveckling. Han menar att fokuset på att alltid ha en spelbar version av spelet gör det lättare att förstå vilka effekter som speldesign och implementation har på upplevelsen. Dessutom menar han att den iterativa processen är nödvändig inom spelutveckling eftersom ett färdigt spel sällan liknar dess initiala koncept. Spelet måste itereras fram för att nå ett bra resultat.

Arbetet delades upp i två veckor långa *sprintar* där varje sprint hade sina egna mål och uppgifter. Dessa mål och uppgifter bestämdes i början av sprinten och berodde på hur långt projektet hade kommit efter föregående sprint.

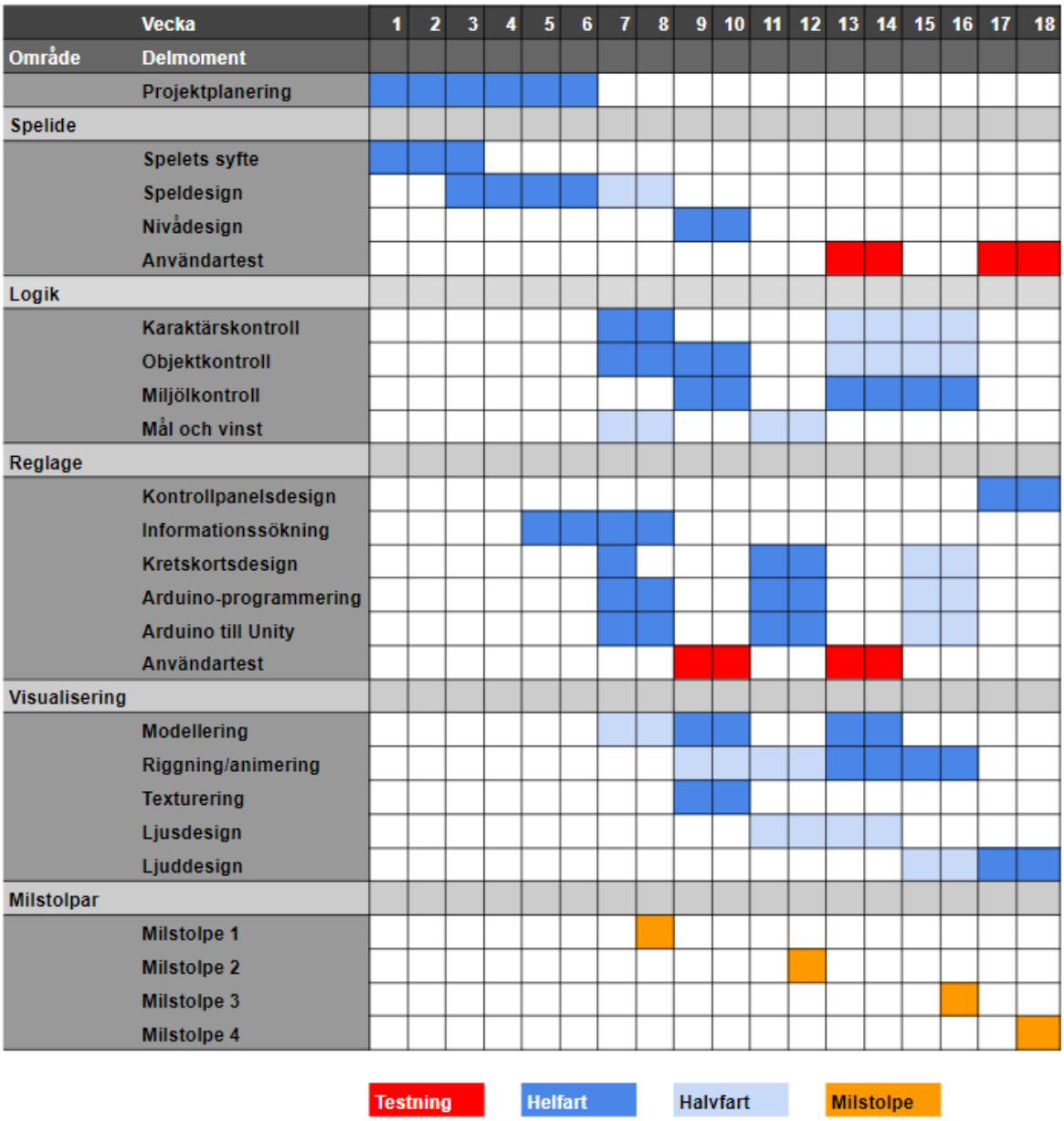
3.2 Tidsplan

Projektet utvecklades under 18 veckors tid. De första sex veckorna hanterade endast planering och informationssökning. Under denna 18 veckors period arrangerades möten med examinator och kund,

vilket beskrivs i 3.3.2. Tidsplanen beskriver utvecklingsgruppens arbetsplanering. Gruppen hade utöver ordinarie slutdatum för inlämning av projektplan och projektapport som finns i kursen definierat milstolpar i projektets utveckling. Milstolpar och tidsplanen kan ses i figur 3.1. Vid milstolparnas datum skulle specifika funktioner vara implementerade för att projektet inte skulle bli försenat.

- Milstolpe 1 : De fysiska reglagen skulle vara kopplade och fungera. En spelkaraktär skulle vara skapad. Spelkaraktären skulle kunna gå runt, plocka upp en låda och slänga lådan i ett hål för att "vinna". Karaktären skulle ha en klar men simpel modell och grundläggande texturer.
- Milstolpe 2 : En andra spelkaraktär skapad med en simpel modell och grundläggande textur. Simpla miljömodeller och bastexturer skulle vara gjorda för spelnivån. Animation skulle vara klar för spelkaraktären såsom att gå, trolla och dylikt. Kontrollern skulle fungera.
- Milstolpe 3 : Spelnivån skulle vara implementerad med modeller och texturer. Karaktärerna skulle ha mer detaljrika texturer och graciösa rörelser i animationen. Vissa objekt i miljön skulle vara animerade. Kontrollpanelen skulle ha ett mer polerat användargränssnitt.
- Milstolpe 4 : Spelet skulle ha en ljudimplementering. Spelet skulle även vara finslipat och klart. En gemensam projektrapport skulle skrivigts klart.

Utvecklingsarbetet började med informationssökning om samtliga delar i systemet. Vidare utveckling av projektet vägleddes med hjälp av projektets milstolparn.



Figur 3.1: Tidsplanen för projektet.

3.3 Mötesprinciper

Projektet hade två huvudsakliga typer utav möten; *Arbetsmöten* och *scrum-möten*. Dessa möten bestämdes och togs fram i enlighet med scrum modellen[2].

3.3.1 Scrum-möten

Scrum-mötena utfördes i enlighet med scrum-modellen i syfte underlätta det organisatoriska arbetet i projektet.

Dagligt scrum-möte

Varje dag började med ett scrum-möte där alla utvecklare berättade vad som gjordes föregående dag och vad som skulle arbetas på under dagen. Detta ledde till att alla utvecklare alltid hade bra förståelse om hur projektet gick framåt. Det var också ett bra tillfälle att fråga resten av gruppen frågor om problem som uppstått.

Sprintplaneringsmöte

Inför varje sprint genomfördes ett sprintplaneringsmöte. Syftet med sprintplaneringsmötena var att sätta upp tydliga sprint-mål och bestämma vilka arbetsområden som skulle täckas. Uppgifter genomfördes från en produkt-backlogg som sedan delas upp i mindre uppgifter och lades in i sprint-backloggen. Det skulle finnas tydliga beskrivningar på uppgifterna så att samtliga gruppmedlemmar visste vad som skulle göras. Sprintplaneringsmötena beräknades ta cirka 2 timmar.

Utvärderingsmöte

Efter varje sprint genomfördes ett utvärderingsmöte. Utvecklingsgruppens arbete och resultat diskuterades och analyserades. Arbetet granskades kritiskt och väl utfört arbete uppmärksammades. Viktiga insikter från den genomförda sprinten togs med in i nästa sprintfas. Utvärderingsmötena beräknades ta cirka 2 timmar.

Kundmöte

Projektets kund och externa intressent var *Alexander Bock*, universitetsadjunkt på Linköpings Universitet.

Under kundmötena visade utvecklingsgruppen vad de genomfört sedan senaste mötet. Kunden fick till största möjliga mån testa produkten. Kunden fick möjlighet att kommentera projektet och även ställa nya krav. Om nya krav ställs lades dessa in i projektets backlog. Kundmötena skulle inte ta mer än 2 timmar och genomfördes minst en gång per milstolpe.

3.3.2 Arbetsmöten

Arbetsmötena täckte de möten som inte är scrum-specifika, här diskuteras utvecklingen av projektet.

Utvecklingsmöte

Mötena där det huvudsakliga arbetet genomfördes var under utvecklingsmötena. Hur arbetet delades upp och vilka uppgifter som genomfördes baserades på vad som bestämts tidigare under scrum-mötena.

Programmering utfördes till en början med endast parprogrammering för att senare övergå till enskild programmering. Övergången från parprogrammering till enskild programmering skedde när utvecklarna hade bra förståelse över systemets struktur. Övergången till enskild programmering skedde så att flera utvecklare inte satt på samma problem. Utveckling av grafik gjordes enskilt. För att underlätta arbetet och kommunikationen genomfördes utvecklingsmötena alltid gemensamt, utvecklarna befann sig i närheten av varandra för att gemensamt kunna lösa eventuella problem/frågetecken. Utvecklingsmötena beräknades ta cirka fyra timmar och skedde två gånger per dag.

Examinatorsmöte

Examinatorsmöten påkallades av examinatorn. Utvecklingsgruppen fick under dessa möten vägledning av examinatorn.

3.4 Dokumentationsprinciper och rutiner

Möten dokumenterades av den dokumentansvarige i *Google Docs* och sparades i en gemensam *Google Drive* map. Google Drive användes så att samtliga utvecklare alltid hade tillgång till dokumenten och för att det underlättade fildelning mellan utvecklingsgruppen. De dagliga scrum-mötena dokumenterades inte eftersom de var informella. Utvärderingsmöten och sprintplaneringarna dokumenterades dock mer flitigt eftersom det var viktigt att samtliga utvecklare skulle veta vad nästa sprint skulle innehålla och vad som hade gått fel i den föregående sprinten.

Eftersom projektgruppen bestod flera utvecklare var det viktigt att koden som skrevs var lättläst och välstrukturerad. Detta löstes genom att alltid kommentera och testa kod innan den lades till i Unity Collaborate.

Kod kommenterades på engelska. Högst upp i varje kodfil skrevs kommentarer gällande kodens syfte och vad koden gör. Dessutom kommenterades koden med förklaringar över vad som gjorts i varje steg. Koden som skrevs följde *C#-standarder* [11].

3.5 Organisation

Projektet utvecklades av totalt fem utvecklare. Utvecklarna arbetade flexibelt med det som behövdes för att uppnå sprintmålen. Samtliga utvecklare hade dock ansvarsområden för att säkerställa att samtliga delar i projektet blev genomfört. Vissa utvecklare hade även administrativa roller. I tabell 3.1 visas samtliga utvecklares namn, kontaktuppgifter och roller. I stycke 3.5.1 och 3.5.2 beskrivs ansvarsområdena och de organisativa rollerna. De externa intressenterna var kunden. Ett Gantt-schema utvecklades för att visa projektets tidsplan.

Tabell 3.1: Utvecklarnas namn, kontaktuppgifter och roller

Namn	**Kontaktuppgifter**	**Ansvarsområde**	**Organisativ roll**
Adam Lindfors	adali414@student.liu.se	Spellogiksansvarig	-
Alexander Uggla	aleug359@student.liu.se	Arduino-ansvarig	scrum-mästare
Anna Furugård	annfu090@student.liu.se	Spellogiksansvarig	Dokumentansvarig
Carl Håkansson	carha719@student.liu.se	Grafikansvarig	Produktägare
Carmen Yu	caryu376@student.liu.se	Arduino-ansvarig	-

3.5.1 Ansvarsområden

Utvecklarna hade specifika ansvarsområden vilket säkerställde att samtliga delar i projektet blev genomförda. De olika ansvarsområdena var *Arduino-ansvarig*, *grafikansvarig* och *spellogiksansvarig*. Uppdelningen säkerställde att alla tre områden kunde utvecklas parallellt vilket underlättade för integreringen av reglagen med spelet.

Arduino-ansvarig

Två utvecklare var Arduino-ansvariga. Utvecklarna hade ett större ansvar över implementation av de fysiska reglagen även om samtliga utvecklare var med i implementationen. Arduino-ansvariga var huvudsakligt ansvariga för implementation av de fysiska reglagen till huvudprogrammet samt var huvudansvariga för integreringen av reglagen i utställningsrummet.

Grafikansvarig

En utvecklare hade större ansvar än de andra när det kommer till spelets grafik vilket innebar spelets 3D-objekt, texturer och animationer. Utvecklaren hade även ansvar över att de andra i gruppen följde samma grafiska stil när de skapade objekt så att allt såg enhetligt ut.

Spellogiksansvarig

Två utvecklare var spellogiksansvariga, vilket innebar att de hade störst ansvar över spelets utveckling i *Unity*. De var huvudansvariga över att spellogiken som implementerades fungerade korrekt. De spellogiksansvariga hade därmed bäst förståelse över spelets uppbyggnad.

3.5.2 Organisativa roller

De organisativa rollerna var: produktägare, scrum-mästare och dokumentansvarig. Dessa roller fanns för att projektet skulle kunna gå så smidigt som möjligt enligt scrum. De utvecklare som inte hade några organisativa roller var endast med i utvecklarteamet.

Produktägare

Produktägaren var ansvarig över produktens kravlista. Gruppen kom överens om vad som skulle läggas till, men för att det skulle kunna läggas till behövde produktägaren vara närvarande. Produktägaren såg till att gruppen prioriterade rätt arbetsuppgifter under rätt tid. I det här fallet var produktägaren även ansvarig över kontakten med kund.

Scrum-mästare

Scrum-mästaren var ansvarig för att lära ut scrum till utvecklingsteamet samt se till att ramverket för scrum följdes. Scrum-mästaren såg även till att kunden inte ställde orimliga krav på utvecklingsgruppen angående hur utvecklingen av produkten skedde. Scrum-mästaren startade scrum-möten där gruppen gick igenom vad de gjort sedan föregående arbetspass samt vad de skulle göra nuvarande arbetspass. När alla hade beskrivit vad de gjort och skulle göra så uppdaterade scrum-mästaren uppdragsposterna i *Trello* (se kapitel 3.6 för uppdragsposter och Trello).

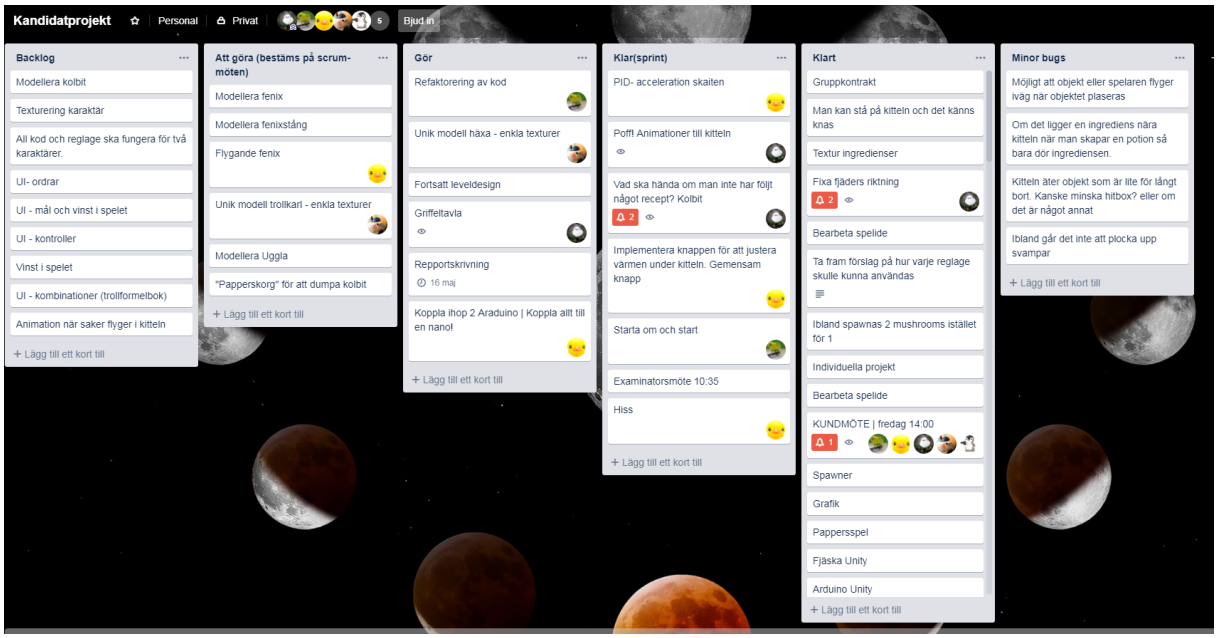
Dokumentansvarig

Dokumentansvarig hade hand om alla dokument och såg till att alla i gruppen hade tillgång till dessa dokument. Dokumenten den dokumentansvariga var ansvarig över låg på Google Drive och *Overleaf*. Dokumentansvarig skrev även ner anteckningar under mötena mellan gruppen, samt kundmötena och examinatorsmötena

3.6 Kravhantering och -spårning

Trello användes för hantering av poster där posterna placerades under tillhörande kategori, se figur 3.2. Som figuren visar är en post en kort text som beskriver en uppgift, krav eller ett problem som ska lösas. Postkategorierna i Trello var *Backlog*, *Att göra*, *Gör*, *Klar(sprint)*, *Klart* och *Minor bugs*. Produktägaren ansvarade för att sätta upp och presentera kraven och dessa ses i *Backlog*. I början av varje sprint flyttades poster från *Backlog* till *Att göra*. När en post arbetades på flyttades den från *Att göra* till *Gör*, samt att utvecklarens namn specificerades på posten. När en post slutfördes flyttades den till *Klar(sprint)*. Efter slutförd sprint flyttades alla poster från *Klar(sprint)* till kategorin *Klart*. I postkategorin *Minor bugs* rapporterades mindre buggar som upptäckts under arbetets gång.

Dokumentansvarig ansvarade för att dokumentera alla krav som uppfyllts. Under utvärderingsmöten utvärderade gruppen gemensamt ifall kraven följts.



Figur 3.2: Kravhanterning- och spårning med användning av Trello

3.6.1 Användartester

Användartesterna genomfördes för att utvecklingsgruppen skulle få in icke-partisk kritik om spelet. Det var också nödvändigt att se hur målgruppen interagerade med systemet då spelet använde sig av icke-konventionella kontroller. Om endast utvecklarna testat internt fanns risken att brister i systemet inte upptäcktes då utvecklarna redan hade kunskap i hur systemet fungerade. Mer specifikt var utvecklingsgruppen intresserade av att se hur intuituvt spelet var i helhet, alltså om testare klarade av att spela utan att utvecklingsgruppen behövde förklara spelmomenten. Vidare bad utvecklingsgruppen om konstruktiv kritik från testarna, vad de hade velat ändra, ta bort, lägga till eller lägga mer vikt på.

Användartesterna genomfördes under sprint 3 på Visualiseringscenter C där personer från vald målgrupp fick testa spelet. Testanvändarna fick ingen information om hur spelet spelades eller hur reglagen skulle användas tillsammans med systemet. Testanvändarna fick själva försöka lista ut målet och vinsten. Utvecklarna hjälpte endast till om testanvändarna fastnade på en uppgift. Efter att testanvändarna spelat klart fick de komma med förslag på vad de ansåg kunde förbättras eller om de upplevde någon funktion som svårförstådd. Allting dokumenterades och kan ses i bilaga A.

Efter att användartesterna genomförts utvärderade gruppen resultatet. De förändringar och förbätt-

ringar som utvecklarna ansåg nödvändiga lades till i *backlog*.

3.6.2 Kodgranskning

Kodgranskning var nödvändigt för att kvalitétssäkra ny kod som implementerades i projektet. För att säkerställa att kodkvaliteten blev så hög som möjligt refaktorerades därför koden löpande under projektets gång. Syftet med refaktorering är att förbättra och öka kvalitén på redan existerande kod.

Parprogrammering användes till en början i projektet för att genomgående öka kodkvaliteten. Utvecklarna testade att koden fungerade önskvärt innan den lades till i systemet. Efter slutförd sprint granskades all ny kod gemensamt i utvecklingsgruppen och extra tid lades på refaktorering. Det var även efter slutförd sprint som utvecklingsgruppen gemensamt granskade spelet för att se hur väl det uppfyllde kundens kravspecifikation.

I ett projekt med fem utvecklare var det viktigt att koden var lättläst. Vid refaktorering var det därför nödvändigt att granska koden utifrån läsbarhet, och då granskades till exempel variabelnamn och kodkommentarer.

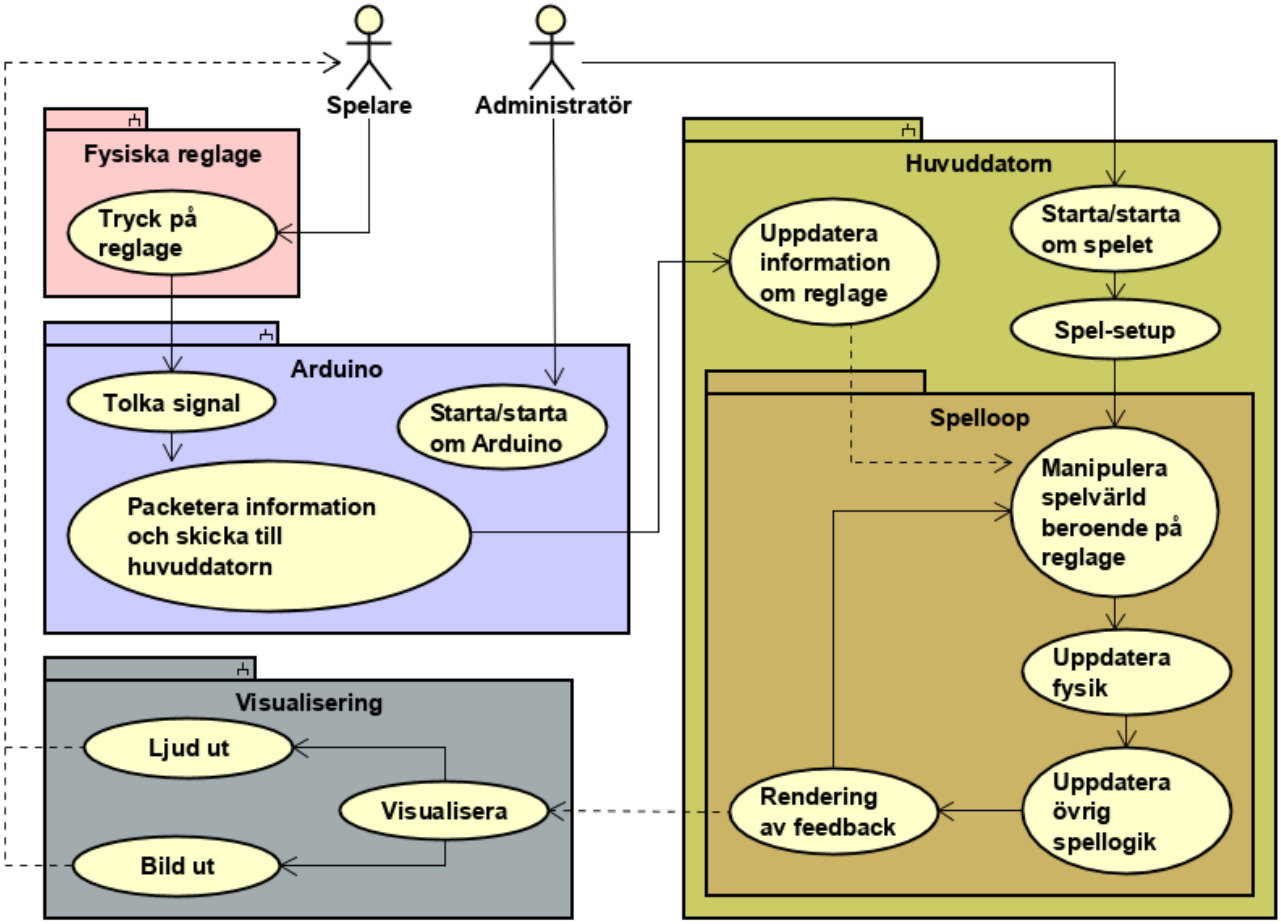
3.7 Versionshantering, -system och rutiner

Unity Collaborate användes för att dela projektfiler samt kod för Unity vilket är ett effektivt sätt att dela filer som ej är textbaserade vilket var nödvändigt. Unity Collaborate användes eftersom det är standard för Unity och kan åstakomma samma sak som alternativa programvaror. *GitLab* användes för delning av kod till Arduino. GitLab användes eftersom det tillhandahålls av Linköpings Universitet.

Kapitel 4

Tekniska lösningar

För att utveckla produkten var det nödvändigt att samtliga utvecklare hade förståelse för hur systemet skulle byggas upp. Utvecklarna började i tidigt skede att göra en informationssökning för att kunna avgöra vilka tekniska lösningar som kunde användas och efter det kunde programmerings- och modelleringsarbetet börja. Figur 4.1 visar hur produktens olika komponentera relaterade till varandra, vad varje komponent ansvarade för och vad användare kunde göra i systemet.



Figur 4.1: Spelets systemarkitektur och användarnas relation till systemets komponenter

4.1 Grundläggande, initiala krav och systembegränsingar

Ett initialt krav var att spelet skulle vara intuitivt. Spelarna skulle utan genomgång eller beskrivning kunna spela spelet. Målet var att 90% av målgruppen skulle kunna slutföra spelet utan vägledning eller genomgång. De fysiska reglagen låg som grund till spelidéen. Spelarna skulle endast kunna styra spelkaraktärerna med användning av reglagen. Reglagen var begränsade i antal och funktionalitet vilket speldesignen behövde anpassas efter. För att användarna skulle få en bra spelupplevelse var det viktigt att spelet inte hade för lång svarstid. Spelet skulle ha en bild-frekvens på över eller lika med 60 bilder per sekund 90% av körtiden.

Spelarna behövde samarbeta för att slutföra spelet och tävlade inte mot varandra. Kraven justerades inte under utvecklingsprocessen.

4.2 Standarder

Formatet .FBX användes för alla 3D-modeller. Formatet användes eftersom det kan lagra all data som är nödvändiga för att representera 3D-modeller i en spelmiljö, till exempel *vertexdata*, texturer och animationer.

I Unity kunde tre olika programmeringsspråk användas: C#, *Boo* och *UnityScript*. Unity gick 2017 ut med att de skulle ta avstånd från både Boo och UnityScript och istället fokusera mer på C# [12]. Detta gjorde det naturligt att använda C# som programmeringsspråk i projektet.

Koden som kördes på Arduino-kretskortet liknar *C++*. Det var möjligt att använda andra programmeringsspråk som t.ex. *JavaScript* med användning av externa bibliotek [13]. Efter informationssökning beslutades det att programmering i JavaScript skulle försvåra utvecklingsprocessen eftersom utvecklarna var mer bekväma med C++.

4.3 Målplattform

Spelet utvecklades för en dator med Windows som operativsystem. Windows är idag det mest använda operativsystemet [14], vilket gjorde spelet lättillgängligt för störst mängd användare. Utvecklarnas respektive datorer körde Windows som operativsystem och det fanns inga ytterligare resurser i projektet vilket också motiverade valet av Windows som operativsystem.

4.4 Fysiska reglage

Tabell 4.1 visar den hårdvaran som användes för de fysiska reglagen. Reglagen kopplades till Arduino Nano-kretskortet som i sin tur kopplades till datorn där spelet kördes. Spakreglagen är analoga och består vardera av fyra strömbrytare. Varje spakreglage har en port som kopplas till jord och resterande fyra portar kopplas till analoga portar på kretskortet. Knapparna kopplas till digitala portar. I ena ändan kopplades de till en strömkälla på kretskortet och i andra ändan till jord och en digital port. Rotationsreglaget är digitalt och ger signaler när den roteras motsols, roteras medsols och när den trycks in. Reglagens jord och ström kopplades till respektive portar på kretskortet. Övriga tre kopplingar kopplades till kretskortets digitala portar.

Avläsning av signalerna från reglagen gjordes med kod från exempelprogram som finns i Arduino IDE.

Tabell 4.1: Hårdvara för reglage och Arduino

Hårdvara	**Antal**	**Syfte**
Arduino Nano [19]	1	Sammanställa signaler från reglage till huvuddator
Spakreglage (Adafruit) [18]	2	Styrdon
Knappreglage (Adafruit)[18]	2	Styrdon
Rotationsreglage (Adafruit)[18]	1	Styrdon

4.5 Arduino till Unity

Unity har inget inbyggt system för att läsa av information från ett Arduino-kretskort. För att göra detta användes *Ardity* [16] vilket är ett bibliotek till Unity för att läsa av ett Arduino-kretskorts konsol och sedan lägga in raderna i Unity i form av strängar.

Under varje uppdatering av koden på Arduino-kretskortet läses först reglagens status av och sparas som *booleanska* värden. Sedan konverteras dessa värden till en sträng. För knapparna representerar ’0’ av och ’1’ på. Varje spak har ett värde för horisontell- och vertikalled respektive. För dessa värden representerar ’0’ bak, ’1’ stilla och ’2’ fram. Rotationsreglaget har endast ett värde. För det värdet representerar ’0’ bak, ’1’ stilla och ’2’ fram.

I Unity konverteras strängarna från Arduino-kretskortet till en *array* av *integers*.

4.6 Spelgrafik

För spelet skapades två spelkaraktärer, en häxa och en trollkarl, och flera spelobjekt såsom möbler och ingredienser till trolldrycken.

De viktiga modellerna var spelkaraktärerna, en kittel, ingredienserna till trolldryckerna och en objekt spelkaraktärerna kunde ge trolldrycken till. Spelkaraktärerna fick en gå-animation och en animation när de plockar och lägger ned ingredienserna och trolldryckerna. Kitteln fick en animation när spelkaraktärerna lade in två ingredienser för att indikera att en trolldryck tillagades.

4.7 Utvecklingsmiljö

I tabell 4.2 visas program som användes för projektet.

Tabell 4.2: Programvaror som användes under utvecklingsfasen.

Programtyp	Namn	Användningsområde
Spelmotor	Unity	Skapandet av spelets level
Programutvecklingsmiljö	Visual Studio	Programmering av spelfunktioner
Modelleringsprogram	3ds Max och Blender	Skapandet av karaktär och objekt
Bildbehandlingsprogram	Photoshop	Skapandet av texturer
Utvecklingsmiljö	Arduino IDE	Utvecklingmiljö för kodning
Versionshantering	Unity Collaborate	Underlätta för utvecklare att programmera i samma filer
Projekthantering	Google drive, Trello och overleaf	Hantering av dokumentation, att-göra-listor samt för den gemensamma rapportskrivningen

4.7.1 Spelmotor

Som tabell 4.2 visar användes Unity som spelmotor och spelet programmerades i C#. Unity användes eftersom det är en gratis spelmotor och arbetsgruppen hade bra förståelse av Unitys gränssnitt. C# valdes som programmeringsspråk eftersom gruppen hade tidigare kunskaper i språket.

4.7.2 Modellering- och bildbehandlingsprogram

Texturerna skapades i bildbehandlingsprogrammet *Photoshop* och hämtades även på vissa webbsidor med gratis texturer. Båda spelkaraktärerna modellerades och animerades i modelleringsprogrammet *Blender* och spelobjekten modellerades i både *Blender* och *3ds Max*.

4.7.3 Arduino IDE

Arduino IDE används för att skriva och kompilera kod till Arduino samt för att ladda upp koden på Arduino-kretskortet. Programmet valdes eftersom det utvecklas och rekommenderas av samma företag som bygger Arduino-kretskorten [15]. Koden liknar C++, vilket underlättar för utvecklingen då utvecklarna har tidigare erfarenheter i språket.

Kapitel 5

Resultat

Slutprodukten av utvecklingsarbetet är ett spel för två personer som använder sig utav en spelkontroll som skapats med hjälp av Arduino komponenter.

5.1 Tekniska principer

Som nämnts i 2.1 togs inspiration för kontrollernas utformning från *Design of Everyday Things* [9], där Norman diskuterar hur affordans och användarvänlighet kan maximeras genom att utnyttja konventioner och användares tidigare erfarenheter av användargränssnitt. Detta hjälpte utvecklingsgruppen att uppnå syftet att integrera icke-konventionella kontroller i ett spel.

Reglagen kopplades till spelet med hjälp av det existerande kodbiblioteket Ardity vilket beskrivs i kapitel 4.5.

Användartesterna som beskrivs i kapitel 3.6.1 resulterade i värdefull information om vilka förbättringar och förändringar som kunde göras i spelet. Kontrollerna kunde ändras för att göra dem mer engagerande och lättare att förstå.

5.2 Spelet

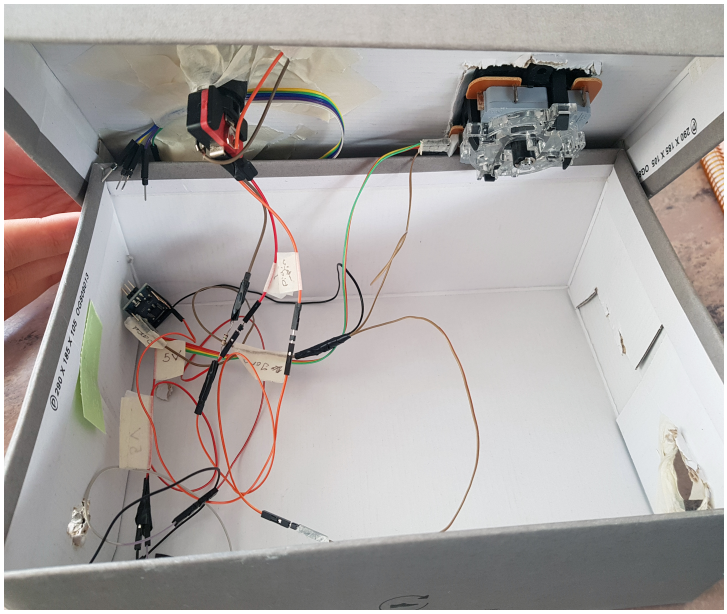
Spelet går ut på att kombinera olika ingredienser som finns i spelnivån för att skapa trolldrycker. Trolldryckerna ska sedan lämnas till en uggla med samma färg som då ger spelarna poäng beroende på hur snabba spelarna var på att leverera drycken. Figur 5.1 visar en skärmdump av spelet.

Figur 5.1: En skärmdump av spelet

Alla 3d-modeller, texturer och animeringar i spelet - förutom texturerna till mattorna och tunnan - är gjorda av projektgruppen. Eftersom allt är gjort av utvecklingsgruppen själva har vi bättre kontroll på hur krävande spelet är. På en bärbar dator från 2016 utan diskret grafikkort har spelet ungefär 80 bilder per sekund.

Kontrollerna som skapats består utav varsin skolåda, knapp och spakreglage. Kontrollerna är inte helt symmetriska utan en av dem har också en ratt som används för att få fram ingrediensen slem. Svarstiden mellan spelkontrollerna och spelet är tillräckligt snabbt för att testanvändarna kunde spela utan att de upplevde hinder. Figur 5.2 och figur 5.3 visar kontrollernas utseende och upplägg.

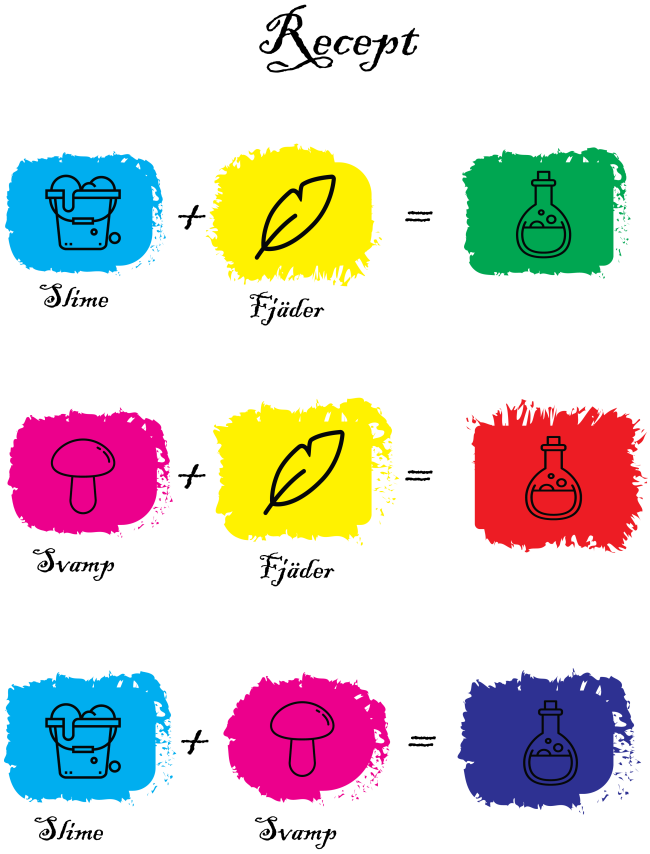
Figur 5.2: Kontrollpanel av skokartong, fysiska reglage och Arduino.



Figur 5.3: Insida av kontrollpanel där reglagen är kopplade till Arduino

För att förmedla hur trolldrycker blandas skapades en plansch som kan ses i figur 5.4. Denna plansch

ska finnas tillgänglig när spelet spelas.



Figur 5.4: Receptplansch för att brygga trolldrycker

Kapitel 6

Analys och diskussion

I det här kapitlet diskuteras projektets resultat. De teorier som tagits upp i kapitel 2 analyseras och jämförs med resultatet och arbetet undersöks i ett vidare sammanhang. Därefter diskuteras huruvida spelet är etiskt acceptabelt.

6.1 Metod

En del av syftet med arbetet var att studera hur ett spel kan utvecklas med en agil utvecklingsmetodik. Eftersom dagliga möten hölls var det alltid tydligt hur projektet gick framåt. När någon gruppmedlem haft problem med något moment har det tagits upp på dessa möten och hjälp har kunnat fås. Detta har medfört att projektet har utvecklats i en miljö där frågor och funderingar uppmuntras vilket är positivt.

Under sprintplaneringarna kunde det ibland vara svårt att uppskatta hur lång tid de olika uppgifterna skulle ta att slutföra. Detta medförde att uppgifter ibland inte blev slutförda i tid och fick ligga kvar till sprinten efter.

Det fanns också moment som endast ett fåtal i gruppen hade kunskap nog att slutföra. Det var till exempel svårt för alla i gruppen att få bra förståelse för hur Arduino och de fysiska reglagen fungerade vilket resulterade i att de Arduino-ansvariga utförde de uppgifterna. Detta medförde då att de Arduino-ansvariga fick bättre förståelse medan de andra hamnade mer bakom med just den kunskapen.

Parprogrammering användes vid största delen av arbetet. Det finns både positiva och negativa aspekter med detta. I projektets uppstart var det positivt med parprogrammering eftersom det möjlighetgjorde att flera personer kunde sätta sig in och förstå spelets grundarkitektur. Det var även bra med parprogrammering i detta skede eftersom vissa baskomponenter och funktioner kunde vara svåra att programmera då de är stora och komplexa. Ett exempel på en sådan funktion är möjligheten för spelaren att plocka upp objekt i världen. Detta är mer komplicerat än vad det först låter eftersom rätt objekt alltid ska plockas upp och objekt kan läggas till och försvinna från världen.

När spelets grundstruktur var färdigbyggd och alla i gruppen hade bra förståelse över den borde gruppen slutat med parprogrammeringen. Detta pågrund av att funktionerna som var kvar att arbeta på var relativt små och kunde klaras av en programmerare. Gruppen fortsatte trots det med parprogrammeringen för länge vilket slösade tid eftersom två programmerare satt med samma problem.

6.2 Resultat

Slutprodukten blev ett samarbetsspel för två personer som även är lärorikt för målgruppen. Spelet lär ut kopplingen mellan RGB- och CMYK-färger. Det främsta som lärs ut i spelet är färglära vilket tas upp i kapitel 2. Under användartestet var det tydligt att användarna funderade på hur de olika ingredienserna skulle kombineras för att skapa rätt trolldryck. Det är dock oklart om de faktiskt tänkte på att ingredienserna och trolldryckerna var färgkodade. Det hade kunnat förtydligats att färger på ingredienser och trolldryck var kopplade. Trots detta är det möjligt att användaren lär sig om färglära undermedvetet.

Användandet av fysiska reglage som kontroller istället för konventionella kontroller visade sig vara lyckat. Användarna hade en positiv spelupplevelse. De fysiska reglagen gjorde att spelet var svårare än om det hade använt konventionella kontroller eftersom kontrollen inte var responsiv nog. Detta gjorde att användarna ville fortsätta spela för att försöka bli så bra som möjligt med kontrollen. Det var också positivt för spelupplevelsen eftersom alla användare var ungefär lika bra på spelet. Det spelade alltså inte så stor roll om användaren hade stor vana med datorspel tidigare eller inte. Detta medförde att användarna kände att de alla kunde tävla om att få så bra poäng som möjligt.

Eftersom utvecklingsgruppen inte fick många krav på spelet från kunden var det utvecklingsgruppen som fick bestämma hur spelet skulle vara. På grund av detta blev det lätt att utvecklingsgruppen satte mål som inte var rimliga inom tidsramen. Det var därför nödvändigt att arbetsgruppen arbetade med att undersöka och jämföra vilka tekniska principer och lösningar som skulle tillämpas i projektet. Utvecklingsgruppen valde tekniska lösningar som lätt kunde integrera hårdvara och mjukvara.

Scrum var därför till stor nytta eftersom gruppen kunde ändra de mål som satts upp och ändå ha ett spel som gick att spela. Att studera hur ett agilt arbetssätt kan användas för att utveckla ett spel var därför nödvändigt för att kunna slutföra spelet.

6.3 Arbetet i ett vidare sammanhang

Eftersom spelet kommer ställas ut i en utställningsmiljö är det möjligt att det kommer spelas av användare utanför målgruppen. Att barn under 13 år skulle spela spelet är därför troligt. Barn vid unga åldrar är väldigt godtrogna och det är därför viktigt att det som lärs ut genom spelet är både etiskt och faktisk korrekt. En ingrediens i spelet fås genom att ta en fjäder från en fågel. Detta kan få barn att tro att det är okej att skada djur i verkligheten. För att undvika detta har fågeln gjorts till en fenix vilket gör kopplingen till verkligheten mindre.

För att både killar och tjejer skulle känna sig inkluderade skapades en karaktär av vardera kön. Eftersom tjejer ofta är underrepresenterade inom spelkultur så är det viktigt att se till att det finns kvinnor representerade i spel för att tjejer ska känna sig välkomna.

Kapitel 7

Slutsatser

Det här kapitlet avslutar rapporten med detaljerade svar på frågeställningar, förbättringsmöjligheter, samt vad projektgruppen har för planer med projektet i framtiden.

7.1 Hur icke-konventionella kontroller kan användas för att göra ett spel engagerande

Vid användning av icke-konventionella reglage finns utmaningen att reglagen ska skapa engagemang och inte frustration hos användaren. Det var därför önskvärt för utvecklingsgruppen att testanvändarna upplevde det spännande att använda kontroller som inte var konventionella. För att säkerställa att spelet upplevdes engagerade genomfördes användartester, syftet diskuteras i 3.6.1. Idag finns en stor mängd av spel som använder konventionella kontroller som datormus och tangentbord. Det medförde att testanvändarna vid första anblick av kontrollpanelen upplever att det är spännande och någonting nytt.

Det var positivt för användarnas engagemang att de inte hade tidigare erfarenhet av kontrollen. Detta gjorde att även personer som inte hade stor erfarenhet av datorspel kunde uppskatta spelet eftersom alla var på samma nivå. Resultatet från användartesterna som lade grund för slutsatsen kan ses i bilaga A.

Kontrollerna som tagits fram har utvecklats för göra spelet så engagerande som möjligt. Ena kontrollen har till exempel en miniatyrversion av ett valv som finns i spelet som kan vridas för att få fram ingrediensen slem. Användarna hade enligt användartesterna svårt att hitta kranstationen men trots detta var spelupplevelsen positiv hos användarna under användartestet. Även de knappar som inte hade tydlig affordans till sin funktion uppskattades eftersom användarna tyckte det var kul att ta reda på knappens funktion genom testning.

7.2 Hur en kontrollpanel för icke-konventionella kontroller bör utformas för att användarna ska förstå deras funktionalitet

Som det togs upp i kapitel 2.3 är det viktigt att en icke konventionell kontroll har god affordans till det den gör i spelet för att användaren ska förstå vad den gör. Resultatet som lade grund för kontrollernas utformning kan ses i avsnitt 5.1. Spakarna har tydlig affordans till att röra spelkaraktärerna då spakar konventionellt används för att styra saker i ett tvådimensionellt plan. De stora knapparna som används för att plocka upp och släppa ingredienser har inte lika bra affordans då en knapp utan någon signal

inte har någon specifik konvention för vad den bör göra. Ratten har inte heller god affordans eftersom den inte liknar ratten i spelet tillräckligt mycket. Genom att göra den mer lik den ratten skulle den troligen få bättre affordans.

De olika reglagelådorna har inga signaler för att symbolisera vilken låda som styr vilken spelkaraktär. Ifall de hade det skulle det vara enklare för användare att separera vilken karaktär som styrs av vilken kontroll. Lådorna är dock mappade till spelkaraktärernas startpositioner, det vill säga att lådan till höger styr karaktären som startar till höger, så användare får ändå en chans att förstå vilken kontroll som styr vilken karaktär. Resultatet av reglagelådorna kan ses i figur 5.2 och figur 5.3.

Därmed bör en kontrollpanel för icke-konventionella kontroller utformas så den tar hjälp av konventioner om styrdon och så den tar hjälp av användares tidigare erfarenheter. Kontrollpanelen bör ha en eller flera signaler till vad de styr.

7.3 Hur det säkerställs att användare vet hur de ska kombinera ingredienser för att skapa trolldrycker

I utvecklingsgruppen har samtliga utvecklare kunskap i hur CMY(K) och RGB kan kombineras, vilket användes för att skapa ett spel som också vara lärande. En utmaning var att få en novis användare att också förstå hur de ska kombinera ingredienserna för att skapa trolldrycker. I kapitel 2 diskuteras tidigare arbete och forskning inom området som var relevant för utvecklingsgruppens arbetet. Att undersöka relaterat arbetet var nödvändigt för projektets resultat vilket kan ses i avsnitt 5.1.

Utvecklingsgruppen hade som ambition att implementera en guide i spelet som skulle vägleda spelarna. Det hann inte implementeras utan en plansch användes för att visa hur ingredienserna kombinerades till trolldrycker, se figur 5.4. Genom användartester visades att detta var ett effektivt sätt att kommunicera vilka trolldrycker olika kombinationer av ingredienser skulle resultera i. Planschens receptbeskrivning var tydligt färgkodade och stämde nära överens med ingrediensernas och trolldryckernas färger i spelet vilket gjorde det lätt för användare att koppla ihop dem. Vidare är det mycket svårt ta miste på ingredienser och trolldrycker då de har färger som ligger på olika sidor av deras respektive färgspektran, alltså CMY(K) respektive RGB. Detta innebär att färgerna i princip är så olika som färger kan bli avseende färgnyans, vilket underlättar när ingredienser och trolldrycker ska särskiljas [5].

7.4 Förbättringsmöjligheter

Eftersom arbetet var under tidspress och höga krav sattes av projektgruppen finns det funktioner som tyvärr inte kom med i slutprodukten. En förbättringsmöjlighet är att utveckla kontrollerna till att använda sig utav fler fysiska reglage. Om dessa fysiska reglagen implementeras på ett sätt som ökar kontrollens affordans kan spelupplevelsen öka.

Spelet är för närvarande enkelt och saknar djup. Det innebär att användaren kan se och göra allt spelet har att erbjuda på en spelomgång. För att göra spelet mer komplicerat skulle fler ingredienser och kombinationer läggas till. Det skulle även kunna läggas till fler steg för att få en trolldryck från ingredienser. Till exempel skulle användaren kunna behöva koka ingrediensen slem innan den går att använda.

Ett annat sätt att göra spelet mer komplicerat är att sätta in händelser i spelet som introducerar nya spelmoment. Exempel på en sådan händelse skulle kunna vara att trolldrycksbutiken angrips av datorstyrda fiender, vilket skulle kräva att spelarna samarbetar för att besegra fienderna. Ett annat exempel

på en händelse är en jordbävning som separerar marken så spelarna måste stå på varsin del av spelplanen och eventuellt kasta saker till varandra för att klara nivån.

Majoriteten av modellerna hade mörkare färger, vilket ledde till att scenen upplevdes som mörkt. Därför var det svårt för spelarna att röra runt då de inte kunde se vart spelkaraktärerna var. En förbättringsmöjlighet är därför att höja ljusintensiteten i ljussystemet.

Något som utvecklingsgruppen tyvärr inte hunnit implementera är ett spelläge för färgblinda. Då de viktigaste spelmomenten kretsar kring att blanda färger blir spelet så gott som ospelbart för en färgblind person. Till exempel skulle en person med röd-grön färgblindhet ha mycket svårt att avgöra om de ska göra en röd eller grön trolldryck.

De fysiska reglagen hade kunnat implementeras på ett mer unikt och nytänkande sätt, se figur 5.2. De fysiska reglage som använts i detta projekt är visserligen ett ovanligt sätt att interagera med ett spel, men de har enligt projektgruppen använts på ett förutsägbart sätt i detta projekt vilket också kan ses i resultatet från användertesterna i bilaga A. Styrspakar används för att förflytta karaktärerna och knappar används för att ta upp objekt och lägga ned dem igen. Detta har gjorts i spel förut och bjuder inte på vidare överraskningar. Hade utvecklingsgruppen fått göra om detta projekt igen hade den implementerat reglagen på sätt som var mer oväntade, till exempel för att manipulera miljön.

Litteraturförteckning

https://teams.microsoft.com/?lm = deeplinklmsrc = homePagecmpid = SignIn/conversations/AllmKeith
SchwaberK., SutherlandJ., *The Scrum Guide*, scrum.org, november2017, hmtad : 2019−02−13
https : //www.scrumguides.org/docs/scrumguide/v2017/2017−Scrum−Guide−US.pdfKeith, C., *Agile*
M.Comstedt., *Grunderna i Scrum - roller, begrepp och aktiviteter*, Onbird, 2017,
https : //onbird.se/grunderna−i−scrum/McGavinD., StukenborgB., WitkowskiM., *Color Figures in BJ*
KurtSquire, *Video games and learning: Teaching and Participatory Culture in the Digital Age*, TeachersColle
DonaldA.Norman*The Design of Everyday Things*, BasicBooks, 2013RoryMcGloin, KirstieFarrar&Mari
//doi.org/10.1080/15213269.2012.752428*C# Coding Conventions*, hmtad : 2019 − 05 − 31
https : //docs.microsoft.com/en − us/dotnet/csharp/programming − guide/inside − a −
program/coding−conventionsRichardFine*UnityScript's long ride off into the sunset*, Unity, 2017−
08 − 11, hmtad : 2019 − 02 − 26
Johnny-Five: JavaScript for Arduino, hmtad : 2019 − 02 − 26
http : //johnny−five.io/StatCounter, *Operating System Market Share Worldwide*, hmtad : 2019−
02 − 27
http : //gs.statcounter.com/os−market−share/desktop/worldwideArduino, *Arduino IDE*, Arduino, u.., h
2019 − 05 − 13
https : //www.arduino.cc/en/Main/SoftwareDanielWilches, *Ardity: Arduino + Unity communication mad*
2019 − 02 − 19
https : //assetstore.unity.com/packages/tools/integration/ardity−arduino−unity−communication−
made−easy−123819ArguelloD., *Why are game studios run like sweat shops? The human toll of 'crunch time'*,
10 − 21., hmtad : 2019 − 02 − 20
https : //www.digitaltrends.com/gaming/how−crunch−affects−game−developers/Adafruit, *The inter*
2019 − 02 − 20
https : //io.adafruit.com/Arduino, *What is Arduino?*, u.., hmtad : 2019 − 02 − 19
https : //www.arduino.cc/JeffGrubb, *783 dance pads and 27.2 million Xbox 360 controllers are connected t*
09 − 25, hmtad : 2019 − 05 − 31
https : //venturebeat.com/2018/09/25/steam − controller − stats/

Bilaga A

Användartester

Användartester genomfördes den 11 April 2019 på Visualiseringscenter C. Målgruppen var personer mellan 13 och 30 år gamla.

Upplevda problem i spelupplevelsen:

- [redacted] Att lägga märke till klockan som visade speltiden.
- Användarna hade svårt att hitta kranstationen i spelmiljön.
- Användarna råkade av misstag kasta färdiga trolldrycker ut för balkongen. Användarna hade önskat att det fanns en osynlig vägg som hindrade att det kunde hända.

Testanvändarnas förslag på förbättringar:

- Ifall spelkaraktären hamnar vid kitteln ska elden under påverka spelkaraktärens liv.
- Till det fysiska reglagen ta fram en ratt som kan fästas på reglagen.
- Belöna spelaren med att utöka speltiden om spelaren är tillräckligt bra.
- Att klockan som visar speltiden blinkar till i början samt när det är lite speltid kvar.
- Att spelkaraktärerna accelererar upp till en hastighet när de börjar springa istället för att direkt röra sig med en konstant hastighet.

Bilaga B

Arbetsfördelning

Nedan följer de delar i projektet som respektive utvecklare har bidragit till. Samtliga utvecklare har tillsammans skapat spelideén och spelflödet.

B.1 Alexander Uggla

Arduino-ansvarig och scrum-mästare

Alexander har arbetat med följande:

- Informationssökning, design och implementering av flödet från spaktryck till Unity. (Tillsammans med Carmen Yu)
- Kod för att läsa av signaler från reglagen. (Tillsammans med Carmen Yu)
- Kod för att packetera signalerna från reglagen på arduino-kretskortet. (Tillsammans med Carmen Yu)
- Kod för att avpacketera signalerna från reglagen inuti Unity.
- Design och implementation av recept/kombinationssytemet, det vill säga hur spelet förstod vilken trolldryck den skulle skapa beroende på vilka ingredienser som lades i kitteln.
- Design och implementation av arvshierarkin för objekt i spelet.
- Koppling av sladdar mellanreglage och Arduino-kretskortet. (Tillsammans med Carmen Yu)
- Design och konstruktion av reglagelådorna.
- Ledde de dagliga scrummötena och såg till så att alla följde scrummetoden.
- Höll koll på Trello; la till poster under möten och rensade i slutet av varje sprint.
- Skapade en vacker anka.

B.2 Anna Furugård

Spellogiksansvarig och dokumentansvarig

Anna har varit spellogiksansvarig och främst arbetat med kodning i Unity, större delen av arbetet genomfördes med parprogrammering. Arbetet har innefattat bland annat funktioner för hur spelkaraktären interagerar med spelobjekt i miljön, spelkaraktärens rörelse och arbetet med mål- och vinst. I arbetet som spellogiksansvarig tillsammans med Adam ingick att ha en god förståelse hur spelet är uppbyggt och spelutvecklingen i Unity.

Som dokumentansvarig har Anna haft hand om alla dokument och sett till att dessa varit tillgängliga för resterande gruppmedlemmar. Som dokumentansvarig ingick även att dokumentera under sprintutvärdering, sprintplanering, kundmöten och examinatorsmöten.

Anna har även arbetat med modellering av fjäder, pergamentrulle och bokhylla till miljön. Anna lade till animation för kitteln.

B.3 Carl Håkansson

Produktägare och grafikansvarig

Carl har arbetat med att producera assets till spelet med allt vad det innebär. Modeller, UV-layout, texturer, riggning, animering och exportering till Unity, följt av implementationen av dessa assets i Unity. Detta inkluderar alla karaktärer och omgivningar förutom några få objekt. Carl har också programmerat hur alla animationer (förutom till kitteln) beter sig, en fåtal funktioner i spelmiljön och funktioner relaterade till tidsåtgång i spelet och musik m.m. Carl utvecklade också en mycket tidig prototyp av spelet, med några få centrala spelmekaniker som sedan utvecklades vidare.

B.4 Carmen Yu

Arduino-ansvarig

Carmen har varit Arduino-ansvarig och främst arbetat med kopplandet av sladdar mellan de fysiska reglagen och Arduino-korten. Carmen har programmerat in funktioner för varje fysiska reglage. Varje funktion fick en viss signal som skickades till Unity för att kunna kontrollera spelkaraktärerna. Allt detta gjordes tillsammans med Alexander.

Utöver programmering av Arduino har Carmen även designat häxan, ingredienserna och trolldrycken, samt modellerat simpla objekt till spelnivån som tunnor och lådor.

B.5 Adam Lindfors

Spellogiksansvarig

Adam har främst arbetat med att koda i Unity, större delen av arbetet genomfördes med parprogrammering. Adam har arbetat med en mängd olika scripts som tex hur spelarna plockar upp objekt och funktionen att pausa och starta om spelet. Adam har även lagt ner tid på att lösa buggar i systemet. Adam har även arbetat med hur trolldrycker och ingredienser ska fungera. Hur kameran ska fungera

så att den följer de båda spelarna samtidigt har implementerats av Adam och Anna. Adam har också kodat hur ugglorna ska komma till