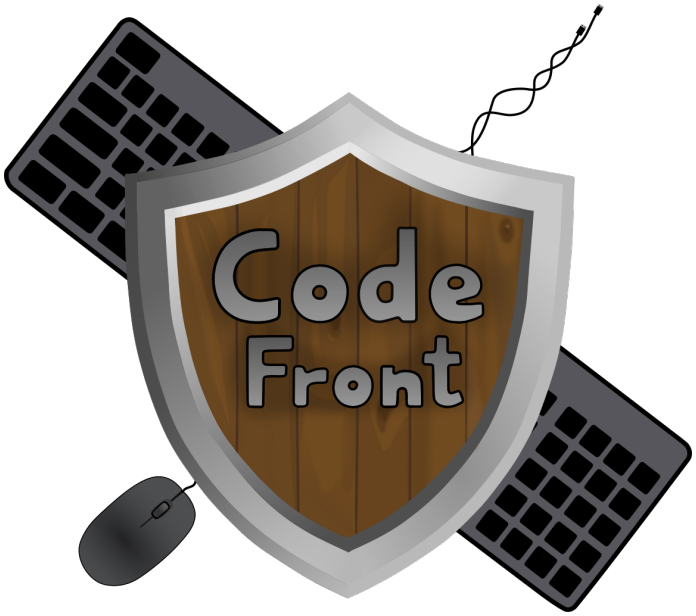
CHALMERS | GÖTEBORGS UNIVERSITET



codeFront

Utveckling av ett spel för att främja högstadieelevers intresse för programmering

Kandidatuppsats i data- och informationsteknik

LUDVIG ANDERSSON
LISA KARLSSON
WILLIAM LEVÉN
AGNES MÅRDH
LUCAS RUUD
KARL WIKSTRÖM

Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2019

Kandidatuppsats 2019

codeFront

Utveckling av ett spel för att främja högstadieelevers intresse för programmering

Ludvig Andersson
Lisa Karlsson
William Levén
Agnes Mårdh
Lucas Ruud
Karl Wikström

Institutionen för data- och informationsteknik
Chalmers tekniska högskola
Göteborg, Sverige 2019

codeFront
Utveckling av ett spel för att främja högstadieelevers intresse för programmering

Ludvig Andersson
Lisa Karlsson
Agnes Mårdh
Lucas Ruud
William Levén
Karl Wikström

Handledare: John Hughes, Institutionen för data- och informationsteknik
Examinator: Nils Anders Danielsson, Institutionen för data- och informationsteknik

Kandidatuppsats 2019
Institutionen för data- och informationsteknik
Chalmers tekniska högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Omslag: Spelets logotyp (Skapad av Ludvig Andersson)

Typeset in LaTeX
Printed by Chalmers Reproservice
Göteborg, Sverige 2019

codeFront
Designing a game promoting young teenagers' interest in programming
Ludvig Andersson
Lisa Karlsson
Agnes Mårdh
Lucas Ruud
William Levén
Karl Wikström
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

A prototype for a serious game has been developed in order to investigate if, and how, serious games can promote an interest in programming for pupils in Swedish compulsory schools aged 13-15. The development of the prototype has been inspired by methods for gamification. The prototype has been developed and evaluated on the target group in two iterations. Evaluations were done with semi-structured interviews after the participants had played the game.

Results from the evaluations show that most of the participants thought it was fun to play the game. What the participants thought was fun was to complete levels and control the characters, they also enjoyed writing code. Programming in our game differed from the pupils' earlier experiences and from many of the games on the market. Ours consists of defining functions that run several times per second and, for example, determine the characters' movement. The programming the pupils had done before consisted of defining a sequence of instructions to solve a problem. Compared to that kind of programming, the programming in our game can be seen as more general problem-solving. The pupils saw the programming in codeFront as "real" programming but also thought it was harder, compared to the programming they had done previously. There were few participants in the evaluations so no definite conclusions can be drawn. Further investigation of what kind of programming is most enjoying and educational needs to be done.

Keywords: serious games, gamification, learning programming, programming interest, programming in school, Sweden

Sammanfattning

I detta projekt har en prototyp av ett seriöst spel utvecklats för att undersöka om, och hur, seriösa spel kan främja högstadieelevers intresse för programmering. Utvecklingen av prototypen har tagit inspiration från metoder för spelifiering. I två iterationer har prototypen framtagits och utvärderats på målgruppen. Utvärderingarna genomfördes med semistrukturerade intervjuer efter att eleverna testat prototypen. Resultatet från utvärderingarna visar att majoriteten av eleverna tyckte att spelet var roligt.

Det eleverna tyckte var roligt var att klara nivåer och att kontrollera karaktärerna. Eleverna uppskattade även att skriva kod. Programmeringen i vårt spel skiljde sig från deras tidigare erfarenheter och från många av de spel som finns på marknaden. Vårt spels programmering består av att definiera funktioner som körs flera gånger per sekund, som exempelvis avgör karaktärens rörelser. Det eleverna gjort innan var att definiera en sekvens av instruktioner som löser ett problem. Jämfört med den sortens programmering kan programmeringen i vårt spel ses som mer generell problemlösning. Eleverna såg programmeringen i codeFront som mer ”riktig”, men också svårare, än den programmering de tidigare använt. Det var däremot få elever i utvärderingarna så ingen definitiv slutsats kan dras. Det behövs ytterligare undersökning kring vilken typ av programmering som är roligast och mest lärorik.

Nyckelord: spelifiering, seriösa spel, programmeringsintresse, programmering i skolan, högstadieelever, lära sig programmering

Förord

Först vill vi tacka vår handledare John Hughes för all hjälp och stöttning. Vi vill även tacka Agneta Nilsson som hjälpt oss vid förberedning och sammanställning av intervjuer. Dessutom vill vi tacka Staffan Björk och Alex Gerdes som gett råd kring spelets design och utformning av intervjufrågor. Slutligen, tack till de som har hjälpt oss att testa och utvärdera vårt spel eller gett återkoppling på rapporten.

Ludvig Andersson, Lisa Karlsson, William Levén, Agnes Mårdh, Lucas Ruud & Karl Wikström, Göteborg, Juni 2019

Innehåll

Figurer

1

Inledning

I och med digitaliseringen och den snabba teknikutvecklingen som sker inom samhällets alla områden ökar medborgarens behov av att kunna använda och förstå tekniska verktyg. Det behövs även fler yrkesprogrammerare. EU-kommissionen rapporterar en förväntad avsaknad av cirka 825,000 programmerare år 2020 [1].

För att kunna hantera digitaliseringen och den snabba teknikutvecklingen har Sverige infört förändringar i läro- och kursplaner för att förbättra elevernas digitala kompetens [2]. Numera är programmering en del av utbildningen i grundskolan inom ämnen såsom matematik och teknik [2]. För att kunna lära ut digitala kompetenser till eleverna krävs en viss kompetens hos lärarna, vilket inte tidigare varit ett krav.

Generellt har behovet av kompetensutveckling för svenska lärare minskat, men inte inom programmering [3]. Sju av tio lärare har behov av kompetensutveckling inom programmering [3]. Ett sätt att öka undervisarnas programmeringskompetens är Skolverkets webbkurs [4] som bland annat vänder sig till lärare för årskurs 4-9. Lärarna får i kursen prova på programmeringsspråken Scratch, Python och Micro:bit.

Förutom att öka kunskapen inom programmering i skolan är det viktigt att öka intresset. Eftersom det kan få fler att se programmering som ett framtida yrke och därmed bidra till att minska bristen av programmerare. Det kan vara speciellt viktigt att öka intresset hos högstadieelever innan deras gymnasieval, vilket kan ses som en inriktning inför arbetslivet.

Användandet av spel, eller spelifierade (gamified) applikationer, skulle kunna öka intresset för programmering. Att använda sig av seriösa spel (serious games), spel som ämnar att lära användaren någonting, motiverar spelaren och kan öka deras förmåga att nå skolans lärandemål [5]. En granskning av 49 seriösa spel visar att de flesta av spelen är omtyckta av sina spelare [6]. Spelen som granskats i studien visade sig främst vara riktade mot universitetsstudenter, snarare än barn och ungdomar [6]. Dessutom var endast hälften tillgängliga för att laddas ner eller spelas på internet [6].

CodeCombat [7] och CodeMonkey [8] är exempel på spel som är välanvända och används i skolmiljö. Båda lär ut programmering i form av textbaserade språk. I dessa spel skriver användaren en sekvens av instruktioner för att lösa ett specifikt problem. Sekvensen är ofta så pass specifik att den inte kan användas direkt för att lösa ett annat liknande problem. Spel som inte har denna typ av programmering utan använder sig mer av ett algoritmiskt tänkande och en mer generell problemlösning riktar sig istället till mer vana programmerare. Den sortens programmering kan även ses som mer reaktiv då programmet reagerar på saker, till exempel hinder, i spelet.

Ett problem med att använda CodeCombat eller CodeMonkey i svenska skolor är att de inte är på svenska. På svenska finns till exempel Swift playgrounds [9], Julkalendern: Selmas saga [10], Coda Game [11] och ScratchJr [12]. Dessa riktar sig i flera fall till yngre barn, är inte textbaserade och har typen av programmering som består av sekvenser av instruktioner.

Förutom att seriösa spel ofta utvecklas utifrån den amerikanska utbildningen skrivs det även relativt få vetenskapliga artiklar om hur den digitala kompetensen kan utvecklas i Sverige. En term relaterad till digital kompetens är datalogiskt tänkande (computational thinking), vilket innebär att kunna formulera problem och lösningar till dessa på ett sätt som en dator kan förstå och utföra [13]. En granskning av studier inom datalogiskt tänkande visar att 63 artiklar kommer från USA jämfört med en artikel från Sverige [14]. Urvalet gjordes med hjälp av en databas och nationaliteten baserades på artikelns förstnämnda författare. Resultatet av granskningen med avseende på nationalitet kan därför vara missvisande till viss del men kan indikera på hur många artiklar som publicerats inom ämnet i Sverige relativt andra länder.

På grund av bristen av svenska artiklar är det svårt att veta hur datalogiskt tänkande och programmering ska implementeras i de svenska skolorna på ett bra sätt. Det finns inga kända studier om vilken typ av programmering som fungerar bäst. Därför är det svårt att veta till exempel om den mer specifika problemlösningen med en sekvens av instruktioner eller den mer algoritmiska eller generella problemlösningen fungerar bäst för nybörjare. Det kan finnas ytterligare typer av programmering som skulle kunna fungera bättre men även det är svårt att veta säkert och det är inget som undersökts närmare i detta arbete. Det finns få spel som är på svenska som har en mer generell och algoritmisk programmering och som använder sig av ett textbaserat programmeringsspråk.

Denna rapport presenterar ett spel som har använts för att utforska hur högstadieelevers intresse för programmering kan ökas. Resultatet visar att vårt spel upplevdes som ”riktig” programmering och mer lärorik jämfört med elevernas tidigare erfarenheter av programmering. Det var mycket som var nytt och eleverna hade svårt att följa syntaxen, vilket var problematiskt. I övrigt tyckte eleverna om spelet och uppskattade att få skriva kod.

Båda iterationerna av spelet kan hittas här: https://codefrontgame.github.io/codeFrontDemo

1.1 Syfte och målsättning

Projektets syfte är att undersöka hur intresset för programmering kan främjas. Syftet med rapporten är att beskriva projektet och presentera resultat från utvärderingar för att hjälpa framtida utveckling av liknande projekt. Målet med projektet är att ta fram och utvärdera en prototyp av ett spel. Prototypen skall användas för att utforska på vilket sätt spel kan främja intresset för programmering hos högstadieelever.

1.2 Avgränsningar

Detta projekt riktar sig mot högstadieelever i Sverige. Därför kommer spelet huvudsakligen att vara på svenska, med undantag för att koden användaren skriver kommer vara på engelska. Spelet kommer inte vara speciellt anpassat för att användas i klassrumsundervisning, så ingen läroplan eller materiel för hemuppgifter kommer att tas fram. För att hantera den grafiska delen av spelet samt inmatning av användarkod kommer befintliga lösningar att användas, istället för att implementera egna i projektet. Spelet kommer inte vara anpassat för att användas på andra enheter än en dator, delvis för att spelet är tänkt att användas med ett fysiskt tangentbord. Vi kommer inte heller att hantera någon form av distribution. Detta inkluderar att skicka programmet till användare samt hantering av de immateriella rättigheterna.

Rapporten fokuserar på spelets design och resultatet från utvärderingarna snarare än utvecklingsprocessen eller de tekniska aspekterna. Resultatet från utvärderingarna kommer inte mäta om programmeringsintresset förändras. Detta då vi inte har ett utgångsläge för programmeringsintresse som vi kan utgå från. Dessutom är det svårt att mäta om ett intresse ökar och huruvida det är ihållande över tid.

1.3 Samhälleliga och etiska aspekter

Då projektet syftar till att skapa nytta i både ett akademiskt och samhälleligt sammanhang är det viktigt att etiska problem hanteras på ett bra sätt. Som tidigare nämnt är målet med detta projekt att utveckla en prototyp till ett spel för att utforska hur det kan främja programmeringsintresset hos högstadieelever. Eftersom det är ett spel som ska utvecklas och projektets målgrupp är högstadieelever tillkommer en del etiska aspekter som bör tas hänsyn till.

Att helt undkomma de etiska problem, till exempel beroende och oönskad tävlingsinriktning, som kan uppstå i vårt arbete är något som är mycket svårt. Det är det mänskliga belöningssystemet vi vill nyttja, för att skapa en uppmuntrande miljö och få personer intresserade av programmering. Dock är detta samma belöningssystem som ligger till grund för spelberoenden.

Vi har i projektet försökt undvika att ha med element som ger oönskade sidoeffekter i vårt spel. Vilka dessa är, och hur vi gått till väga, diskuteras mer i avsnitt 2.3 och avsnitt 2.2. Hur vi tagit högstadieelevernas integritet i beaktning kan läsas om i avsnitt 3 och avsnitt 3.5.

1.4 Rapportöversikt

Rapporten är strukturerad enligt följande: Nästkommande kapitel redogör och motiverar beslut som tagits i projektet samt presenterar tidigare forskning inom projektets ämnesområde. Tredje kapitlet beskriver och motiverar metoden för de utvärderingar som genomförts. Kapitel 4 och 5 beskriver resultatet från projektets två itera-

tioner, vilket inkluderar både prototypens design och resultatet från utvärderingen som genomfördes. Sjätte kapitlet diskuterar prototypens design, utvärderingarnas resultat och projektets etiska aspekter. Slutligen drar kapitel 7 slutsatser och listar potentiella möjligheter för vidare forskning.

2

Spel för inlärning

I detta kapitel beskrivs olika designmöjligheter och aspekter av spelet codeFront samt de beslut som tagits. Dessa beslut är förankrade i teori och tidigare forskning kring pedagogik, spelifiering, seriösa spel samt gränssnittsdesign. Även de etiska aspekterna som behöver tas hänsyn till beskrivs i detta kapitel. Slutligen presenteras och motiveras vilken genre och plattform som valts för spelet.

2.1 Pedagogik inom programmering

Att lära sig programmering kan vara svårt och tidskrävande. Den aspirerande programmeraren behöver både besitta kunskap om programmering, samt strategier för att kunna applicera denna kunskap. Dessutom behöver hen både kunna förstå och producera program, och dessa förmågor skiljer sig åt [15].

Det finns flera skillnader mellan nybörjare och vana programmerare. En skillnad är att nybörjare letar lokalt efter förståelse och läser rad för rad snarare än programmet i sin helhet [15]. De spenderar mindre tid på planering jämfört med vana programmerare, misslyckas med att applicera relevant kunskap och saknar mentala modeller [15]. Nybörjare tenderar att lägga stort fokus på syntax när de lär sig, vilket kan leda till att de har svårigheter att se mönster och därmed inte själva kan skapa bra strategier [16]. Nybörjare har dessutom oftare fel på *for-loopar* och *while-loopar*, vilket kan bero på att de har svårt att förstå hur de fungerar och att de gör en feltolkning baserat på de naturliga språken [15]. Ett exempel är att nybörjare inte uppfattar att villkoret i *while-loopen* testas i varje iteration, vilket kan resultera i fel. Det som framför allt bör fokuseras på vid utlärning av programmering är bra strategier snarare än syntax [15, 16].

För vårt spel som riktar sig till nybörjare finns viktiga egenskaper som programmeringsspråket användaren löser problem med bör ha. Några av de viktiga egenskaperna är lättförståelighet, enkel och tydlig syntax, att felmeddelanden är enkla att förstå samt att språket skrivs i en hjälpsam utvecklingsmiljö [17]. Tanken är att användaren, efter att ha spelat spelet, ska kunna applicera sina kunskaper för att kunna programmera mer självständigt. Det är därför en fördel om språket är, eller liknar, ett välanvänt språk, då det skulle bli en mindre drastisk övergång till att programmera på egen hand.

Det är vanligt att unga blir introducerade till programmering genom blockbaserade språk, vilka har både likheter och skillnader med textbaserade språk. Intresse och attityd påverkas inte av språkets typ, och uppgifter uppfattas som lika svåra [18]. En skillnad är dock att textbaserat språk kan ge större självförtroende gällande programmeringskunskaper [18]. Blockbaserade språk är ofta riktade mot barn, till

exempel Scratch som riktar sig till barn i åldrarna 8-16 år [19].

Det finns fler kategorier än språktyp som också behöver övervägas vid val av språk. Vi valde ett språk som krävde en väldigt liten mängd kod för att komma igång så att större fokus hamnar på problemlösning. I en skolklass upptäcktes det att de elever som lärde sig programmera i Python istället för det mer omständliga språket Java hade färre syntax- och semantikfel i sina program [20]. Dessutom var program skrivna i Python oftare kör- och kompileringsbara [20], men det betyder inte att de är korrekta och fungerar som tänkt. Semantikfel kan finnas som kan resultera i att koden kör men inte beter sig som förväntat. Att programmet kör innebär dock att man kan se hur programmet beter sig, vilket kan hjälpa vid felsökning mer än vad felmeddelanden från kompilering gör. Dessa är ofta svåra för nybörjare att förstå.

Det är svårt att dra en slutsats om vilken typ av språk som främjar högstadieelevers intresse för programmering bäst. De blockbaserade språken har fördelen att de är enkla och motverkar syntaxfel. Å andra sidan kanske eleverna gynnas mer av det självförtroende de kan få av att skriva kod i textform, till skillnad från att placera block. Dessutom kan det vara en fördel att det är ett språk som används i verkligheten. För att minska inlärningskurvan väljs med fördel ett språk som kräver en liten mängd kod för att komma igång. Dessutom skulle det kunna främja elevernas intresse att programmera utan att stöta på de kompilatorfel som hårdare typade språk kan ge. Det beslutades att användarna skulle skriva i Python då det används i stor utsträckning utanför inlärning, är textbaserat och inte är lika omständligt som andra språk.

2.2 Etiska aspekter gällande spel

Detta projekt försöker nyttja människans belöningssystem för att få ungdomar mer intresserade av programmering, samt att lära ut enkla koncept inom programmering. Det positiva med att använda sig av spel, och därmed belöningssystemet, är att ungdomar kan bli mer engagerade i att lära sig, och kanske tar till sig mer av kunskapen. Det kan också leda till att det känns roligare att lära sig, och att det då känns lättare att fortsätta utforska ämnet på egen hand.

Det negativa med att nyttja spelarnas belöningsystem är att de kan utveckla ett beroende [21]. Ett beroende kan leda till negativa aspekter som att ungdomar ”fastnar” i spelet och fokuserar mindre på den lärande aspekten. Det kan också leda till andra negativa konsekvenser, till exempel att de spelar för mycket, vilket också kan påverka personer i deras omgivning negativt. I nästkommande del behandlar vi hur människor kan påverkas av olika spelelement, och vilka som kan leda till negativa sidoeffekter. Dessa element kommer att undvikas i projektet.

2.3 Spelifiering

Att lära sig nya färdigheter kan i många fall kännas repetitivt, långsamt och till och med tråkigt. Hur kan en process som denna bli roligare och mer engagerande?

En erkänd lösning på detta problem, som ofta används i industrin, är spelifiering (gamification) [22].

Spelifiering handlar om att, i en icke-spelrelaterad process, använda sig av spelelement som syftar till att förhöja användarupplevelsen. Exempel på spelelement är topplistor (high scores) och prestationer (achievements). Ett annat vanligt element är en förloppsindikator (progress bar) som visar hur långt användaren har kommit i processen.

En av anledningarna till att spelifiering fungerar är att det utnyttjar människans belöningssystem, men utnyttjandet kommer dock inte utan oönskade sidoeffekter [23]. Dessa sidoeffekter kan skapa beroende, ofrivilliga tävlingsmoment eller uppmuntra överanvändning. Trots effektiviteten hos dessa element [23] strävar vi mot att begränsa deras förekomst i vårt spel, då vi anser att sidoeffekterna överstiger fördelarna.

Istället för beroendeframkallande element används element som främjar lärandet utan de tidigare nämnda sidoeffekterna. Exempel på denna typ av element är att bygga upp uppgifter som uppdrag och att ge återkoppling vid genomtänkta tillfällen för att användaren ska känna att hen får den hjälp som behövs, och får beröm vid framgång.

Spelifiering kan gynnsamt användas inom flera olika områden, ett av dessa områden är utbildning. Använt rätt skapar spelifiering en miljö som motiverar och möjliggör samarbete mellan elever [24]. Vanliga spelelement som får elever att känna att det går framåt och att de lyckas är poäng, nivåer, märken och en förloppsindikator [24]. Användande av poängtavla kan skapa en bra tävlingssituation mellan elever och har rankats som en hög motivationsfaktor [24]. Däremot kan det skapa en oönskad tävlan samt ta fokus från lärandet. Att ha en berättelse förbättrar elevers inlärning, samt ger sammanhang till lärandet och problemlösandet [24].

Vi valde att göra nivåerna i spelet till uppdrag och att återkopplingen skulle ske som en del av spelet. Återkopplingen sker dels genom en kompanjon som hjälper till och dels genom att ge beröm när användaren klarar av en nivå.

2.4 Seriösa spel

Ett koncept som är nära besläktat med spelifiering är seriösa spel (serious games). Definitionen av seriösa spel är inte helt tydlig, men en bred definition är: kompletta spel vars huvudsyfte inte är underhållning [25, 26]. Som tidigare nämnts fungerar spelifiering på så sätt att spelelement används i en icke-spelrelaterad process. Seriösa spel och spelifiering verkar inom samma område men har olika innebörd. Det finns ingen tydlig gräns mellan ett spel och en spelifierad applikation, vilket lätt kan skapa förvirring [26]. Gränsen mellan dessa kan även vara olika för olika personer [26].

Likt spelifiering ämnar seriösa spel att nyttja spelelement till att förhöja användarupplevelsen och göra uppgifter roligare, men på ett annorlunda sätt än som

tidigare nämnts. Tillvägagångssättet seriösa spel har visat sig vara väldigt effektivt, ett exempel på detta är Foldit [27]. Foldit utmanar spelare att vecka proteinkedjor så perfekt som möjligt genom att använda de funktioner som finns inbyggda i spelet, och ger sedan poäng baserat på hur bra de är veckade. Genom att använda Foldit kunde spelare i vissa fall göra lika bra, eller bättre, ifrån sig än algoritmer designade för att lösa samma problem [28]. Foldit är ett konkret exempel på otydligheten mellan spelifiering och seriösa spel då spelet kan tänkas passa in i båda dessa kategorier. Det kan ses som ett pusselspel som används för att lära ut proteinveckning, men det kan också ses som en applikation för att manuellt vecka proteinkedjor med spelelement, till exempel poäng.

2.5 Gränssnitt

Ett bra gränssnitt kan både förhöja och förenkla användningen av en applikation, om det utvecklas på ett bra sätt. En motsatt effekt kan dock uppstå om applikationens beståndsdelar inte utformas rätt. Användare kan då få en negativ upplevelse på grund av gränssnittet. Det kan exempelvis vara att för många visuella element gör det svårt för användaren att fokusera och hitta rätt, att det är irriterande att behöva bekräfta sina handlingar eller att en van användare inte behöver all hjälp en ny användare behöver [29, kap. 11]. Dessa är problem som behöver undvikas, till exempel genom att anpassa hjälpen efter användarens kunskapsnivå. Microsoft använde sig av vänliga agenter (Friendly agents) på 90-talet för att hjälpa nybörjare och få dem att känna sig mer bekväma med att använda mjukvaran [30, s. 140]. De uppfattades istället som barnsliga och gulliga, eller som påträngande [30, s. 141]. Det är därmed viktigt att användaren får tillräckligt med hjälp, utan att den upplevs som irriterande.

Att hjälp finns, och inte upplevs som irriterande, är viktigt även i vårt spel. Därför ska det finnas flera sätt att få hjälp på, samt att denna hjälp ges stegvis. För att den inte ska upplevas som irriterande är hjälpen frivillig och utgår från att användaren ber om den. Gränssnittet ska vara enkelt och inte innehålla stora mängder text, då detta kan överväldiga användaren.

Vi har designat flera olika hjälpsystem för att hjälpa spelaren i sin inlärning. Ett av de enklare sätten användaren får hjälp är genom att det ovanför varje funktionsdefinition finns en text som beskriver funktionen och dess möjliga returvärden. Texten ska hjälpa användaren att förstå hur funktionen bör definieras. Det finns även ledtrådar för varje nivå som kan hjälpa spelaren om denna inte kommer vidare. Ledtrådarna ger olika mycket hjälp, vilket innebär att användaren inte behöver ta mer hjälp än den behöver. Användaren ber om en ledtråd i taget, via en knapp. Förklaringar som kräver mer ingående text eller exempel finns i en bok som kan öppnas och stängas med ett knapptryck. Boken förklarar de programmeringskoncept som hittills förekommit i spelet. En tanke med denna bok är att uppmuntra till sökande efter information, då det är något som är viktigt att kunna som programmerare. Slutligen finns en anka, en kompanjon som hjälper användaren och fungerar som berättare. Ankan har vissa likheter med en vänlig agent då tanken är

att den ska hjälpa och ge trygghet till användaren. För att den inte ska upplevas som irriterande går det att hoppa över ankans dialog.

2.6 Val av genre

Spelet är ett tower attack-spel, alltså ett omvänt tower defence-spel, där spelaren kontrollerar anfallarna. Målet är att ta alla anfallare till toppen av spelplanen utan att de dör. För att lyckas med detta skriver spelaren kod som modifierar beteendet hos anfallarna, till exempel hur de rör sig.

Tower defence-spel har både pussel- och strategielement, vilket gör problemlösning till en naturlig del i dessa spel. Denna genre går bra ihop med programmering då båda handlar mycket om problemlösning. Tower defence är en väletablerad genre och de allra flesta vet vad den innebär. Detta kan medföra att spelarna inte känner lika stort driv i början av spelandet, om det är en genre som inte uppskattas eller om den är överspelad. För att spelet ska passa alla, samt för att motverka förutfattade meningar om genren, vände vi på konceptet och valde istället att skapa ett tower attack-spel. Denna genre kan ses som ny och intressant, och därmed skapa engagemang. Ett tower attack-spel erbjuder också samma möjligheter som ett tower defence-spel gällande att lära ut programmering.

2.7 Plattform

Det viktigaste när det gäller plattformen är att den är lättillgänglig så att fler kan spela spelet. För att ett spel ska få så stor spridning som möjligt bör det kunna spelas på internet, då internet är tillgängligt för nästan hela Sveriges befolkning. Enligt SCB hade över 90% av den svenska befolkningen tillgång till internet i hemmet år 2018 [31]. Dessutom användes internet dagligen av över 80% av befolkningen [32], medan färre än 10% aldrig använde internet alls [33]. Utöver tillgängligheten till och användandet av internet har 93% av befolkningen över tolv år tillgång till en dator [34], vilket även detta underlättar spridningen. Om ett spel finns lättillgängligt för målgruppen ökar chansen att fler personer hittar och testar det, vilket kan leda till att fler blir intresserade av programmering.

I skolan är tillgången till datorer också stor. Enligt Skolverket fanns det 1,9 elever per dator 2015, datortillgången har sedan dess ökat till 1,3 elever per dator 2018 [3]. Det finns alltså större möjligheter att integrera fler digitala verktyg i undervisningen än tidigare, vilket betyder att spelet även skulle kunna användas i undervisningssyfte i skolan.

Vi valde att utveckla spelet som en webbapplikation. Om det var menat att spelet skulle utvecklas som en skrivbordsapplikation istället för en webbapplikation, skulle det vara möjligt att utforma spelet i exempelvis språken Java eller C, då dessa är standard för denna typ av plattform. Om språkvalet var Java hade det varit möjligt att spela spelet på alla datorer med Java installerat, men förmodligen även krävt nedladdning och installation. Med C hade det kunnat bli ännu mindre tillgängligt då

det även skulle kräva ett specifikt operativsystem för att vara körbart. Begränsningarna på en skrivbordsapplikation var grunden till att vi valde webben som plattform, då det gav störst chans att kunna sprida spelet till så många som möjligt.

3

Metod för utvärdering

För att utvärdera och utveckla projektet har intervjuer, samt test av spel, genomförts med personer från den tänkta målgruppen. Deltagarna har även svarat på ett formulär angående deras bakgrund och inställning till programmering. Tre tillfällen planerades in i början av projektet, en träff med en fokusgrupp i ett tidigt skede, en första utvärdering ungefär i mitten av projektet, och en andra utvärdering mot slutet av projektet.

I början av projektet tog vi kontakt med en skola i Göteborgsförorten Mölndal för att se om några av eleverna där var intresserade av att ställa upp i våra tester. Vi fick kontakt med årskurs 7 och skickade genom läraren ut ett missiv (se bilaga B) till föräldrarna.

På grund av tidsbrist, då det var svårigheter med kontakten med lärare och föräldrar, blev tillfället med fokusgruppen inställt. Fokusgruppen var tänkt att hjälpa till med idéer och riktning för projektet. Till detta tillfälle hade det arbetats fram en spelidé, ett tema för spelet och bilder för hur spelet var tänkt att se ut. Fokusgruppen skulle hjälpa oss att få reda på vilka delar av konceptet som verkade intressanta.

3.1 Intervjumetod

Den intervjumetod vi valde att använda var semistrukturerade intervjuer. Denna typ av intervjuer utgår från öppna frågor eller i förtid bestämda ämnen och lämnar rum för att deltagarna ska kunna prata fritt inom ämnet. Då mycket av det vi ville veta handlar om upplevelser, tankar och känslor ansåg vi att denna metod var passande. Om strukturerade intervjuer använts hade vi gått miste om möjligheten att ställa följdfrågor baserat på det deltagarna svarade, och ostrukturerade intervjuer skulle inte gett oss möjlighet att styra diskussionen.

Då intervjudeltagarna är relativt unga utfördes intervjuerna i grupper tillsammans med en intervjuledare. Efter diskussion med vår handledare kom vi fram till att intervju med grupp troligtvis skulle leda till diskussion mellan deltagarna, och därmed mer information för oss. Att de möjligen skulle känna sig säkrare om de hade varandra under intervjun togs också i beaktning. Något negativt som kan komma av att intervjuerna utfördes i grupp är att någon i gruppen kan vara dominerande och att de andra då inte vågar säga emot eller komma med egen åsikt. För att minska en sådan effekt kan den som håller i intervjun styra samtalet och uppmuntra alla att göra sin röst hörd.

3.2 Möjliga felkällor

Då enbart en relativt liten grupp deltagare (sex respektive 19 deltagare) varit med i våra utvärderingar är det viktigt att vi ser på vilka sätt denna grupp skiljer sig från en genomsnittlig grupp med svenska högstadieelever, och på vilket sätt detta kan påverka vårt resultat. Vi vill kunna se om vår testgrupp skulle kunna vara representativ för den tänka målgruppen: högstadieelever. Dessa potentiella avvikelser kommer vi möjligtvis kunna hitta i formuläret med elevernas bakgrundsinformation. Resultatet kommer sedan analyseras för att hitta eventuella skillnader, även om det är en liten testgrupp.

En av frågorna vi valde att ställa berörde deltagarnas kön. I den senaste iterationen av Statens medieråds undersökning ”Ungar & Medier” som gjordes 2017, en undersökning av medieanvändning hos barn mellan 9 och 18 år, fanns en tydlig skillnad mellan könen vad gäller spelande av elektroniska spel [35]. I åldrarna 13 - 18 är det 30% - 47% av pojkar som spelar mer än 3 timmar varje dag, jämfört med 3% - 9% av flickor i samma åldersspann [35]. Det finns också en skillnad i intresse för programmering mellan könen på högstadiet [36]. I åldersgruppen 13 - 18 är tjejer mindre intresserade av programmering än killar i samma åldersgrupp [36]. På grund av skillnaden i spelande och i intresse för programmering ville vi ha med kön som bakgrundsinformation. Denna skillnad skulle kunna påverka hur pojkar respektive flickor uppfattar vårt spel.

Vi valde att även ställa frågor om var deltagarnas föräldrar var födda och vilken utbildningsbakgrund deras föräldrar hade. Då barn med utlandsfödda föräldrar har något lägre genomsnittsbetyg än de med inrikesfödda föräldrar när de går ut nionde klass [37] är det viktigt att se om dessa var representerade i vår testgrupp. Detta gäller även barn utan någon högskoleutbildad förälder, då även de i genomsnitt har sämre betyg än de som har minst en högskoleutbildad förälder [38]. Om inte de grupper som har det svårare i skolan är representerade i testet kan det vara svårt att veta om spelets svårighetsgrad är korrekt satt.

Ytterligare en aspekt som kan påverka resultatet är hur intervjuerna utfördes. Till exempel kan ledande frågor ge felaktiga svar. Vi har försökt undvika dessa i största möjliga mån, men kan inte garantera att samtliga frågor utformats och ställts på korrekt sätt då intervjuernas öppna struktur gör detta nära omöjligt. Vi har inte heller kunnat garantera att deltagarna vågat vara kritiska gentemot oss som intervjuar och det vi skapat, men har försökt, i största möjliga mån, uppmuntra helt ärlig kritik.

Eftersom tiden inte tillät testning på flera skolor blev testgruppen liten och det är då svårt att dra några säkra slutsatser. Projektet och utvärderingarna är till för att utforska området och se vilka delar som kan vara intressanta att utforska vidare. Detta är något som kan indikeras även av en liten grupp deltagare.

3.3 Första utvärderingen

Under första utvärderingen träffade vi sex personer i årskurs 7 som fick spela en enkel prototyp av vårt spel. Dessa sex personer fick lottas ut då det var många i klassen som uttryckt intresse av att vara med vid detta tillfälle. Spelet bestod av två nivåer där spelaren fick testa att ändra i en *if-sats* som styrde en karaktär. Denna nivå av programmering ansågs vara ett lagom stort steg för användaren under ett första test. Det vi ville se var om de förklaringar som gavs var tillräckliga för att hjälpa spelaren att ta sig vidare i spelet, vilka delar som fungerade bra och vilka som fungerade mindre bra.

Efter att deltagarna spelat spelet genomfördes intervjuer i grupper om två. De frågor som ställdes under intervjuerna handlade dels om deras tidigare erfarenheter av och inställning till programmering, och deras upplevelse av spelet (se bilaga D). Efter att deltagarna var klara med intervjun fick de svara på ett kort formulär om deras bakgrund (se bilaga E). Frågor kring programmering ställdes då det ansågs viktigt att veta vad de hade för tidigare relation till programmering, vad de gjort innan och vad de tyckte om det. Om de var vana vid den typen av programmering som spelet använder sig av skulle detta kunna göra att de hade enklare för spelet än vad som kan förväntas av den tänkte genomsnittsspelaren. Frågor kring intresse ställdes för att få en uppfattning om ingående intresse. Även frågor om vad programmering kan användas till ställdes, då vi tror att chansen för intresse är större om en person vet vad programmering är användbart till.

De frågor som ställdes om spelet handlade om vad som var roligt eller tråkigt, och varför. Svaren på dessa frågor skulle ge en bild av vad som skulle arbetas vidare med och vad som behövde ändras. Vi frågade också om de kände att de hade kontroll över karaktären i spelet och om den gjorde som de ville. Här ville vi se om de haft svårighet med att få komma framåt och få spelet att göra som de ville, samt om aspekten av kontroll över karaktären upplevdes som något positivt. Utöver detta frågade vi om deltagarna upplevde det de hade gjort som programmering. Slutligen tillfrågades deltagarna vad de skulle vilja se och göra i spelet i framtiden, för att se om det fanns några idéer vi kunde använda oss av i nästa fas av utvecklingen.

3.4 Andra utvärderingen

Under vår andra utvärdering fick 19 elever ur en sjundeklass på samma skola testa en mer utvecklad version av spelprototypen och sedan vara med i en intervju, och därefter svara på ett kort formulär (se bilaga I). Spelet under denna utvärdering bestod av tolv nivåer där deltagarna fick använda *if-satser* på olika sätt för att ta sig förbi hinder, och därmed klara nivåerna. Efter att de spelat en viss tid blev de uppdelade i grupper om 3-5 personer för intervjuer. De som deltagit i vår föregående utvärdering placerades i en egen grupp. Då detta tillfälle var ett mer utvärderande tillfälle än det föregående ställdes fler frågor, och frågor som var mer underbyggda av tidigare forskning, utöver de frågor från det tidigare tillfället som gett intressanta svar och diskussioner (se bilaga H).

De frågor som tillkom till denna utvärdering var främst baserade på sexton aspekter hos ett bra spel, enligt James Paul Gee [39]. Vi ville se om dessa upplevdes i vårt spel för att kunna dra slutsatser om vilka områden det finns mycket utrymme för förbättring och inom vilka vårt spel fungerar bra som det såg ut vid testtillfället. Att ha fler frågor, och att ha frågor som var mer specifikt inriktade på olika upplevelseområden, gjorde det enklare att styra diskussionen när intervjuerna hölls. Utöver detta frågade vi även om deltagarna uppfattade programmet som ett spel och vad i det som i så fall gjorde att det var, respektive inte var, ett spel. Detta för att se om vårt program mer sågs som ett spelifierat lärande eller mer som ett seriöst spel.

3.5 Etiska aspekter av intervjuer

Vid utvärderingar med skolelever uppkommer viktiga integritetsfrågor vi bör beakta. För att vem som är vem under arbetet samlades personuppgifter in, vilka vi sedan behövde hantera på ett bra sätt. Efter transkription av intervjuerna anonymiserades alla svar och inspelningarna raderades. Vi inhämtade samtycke för deltagande i utvärderingen från samtliga elevers vårdnadshavare, genom ett missiv (se bilaga B), och från eleverna själva precis innan intervjun. Vi fick även tillstånd att spela in.

4

Iteration 1: En enkel prototyp med grundläggande funktionalitet

Under iteration 1 utvecklade vi en första, grundläggande version av spelet. Vi designade delvis spelet med hjälp av teorin vi läst, men också mycket från våra tidigare erfarenheter. I detta kapitel beskrivs först spelets design i sin första iteration, och motivation för de beslut och prioriteringar som resulterade i spelets utformning. Sedan presenteras och analyseras resultatet från första utvärderingen.

4.1 Design

I första iterationen, inför första utvärderingen, skapades spelet med de mest nödvändiga delarna implementerade. Det var viktigt att se hur gränssnittet i stort fungerade samt om spelet började på en nivå anpassad för målgruppen.

Figur 4.1: Skärmdump från spelet codeFront, så som det såg ut inför den första utvärderingen. Bilden visar den första nivån i vilken användaren behöver skriva kod för att förflytta en zombie över en spelplan.

Gränssnittet är uppdelat i två delar (se fig. 4.1). Den vänstra delen består av själva spelet och spelplanen. Den högra delen består av det spelaren behöver för att skriva och köra koden. Längst till höger finns en meny för att välja karaktär samt en knapp för att starta eller avbryta körning av kod. För varje karaktär finns sedan en editor

där användaren kan skriva kod till funktionskroppen för metoden *move*. Vid detta tillfälle finns dock endast en karaktär.

För editorn beslutades det att funktionshuvudet inte ska kunna editeras av användaren för att undvika att användaren ändrar på något som inte ska ändras och har svårt att återgå till kodens ursprungliga läge. Den kod som användarna skriver följer syntaxen för Javascript. Anledningen till att användaren inte skrev i Python i denna iteration var att användningen av Python krävde översättning till vårt utvecklingsspråk: Javascript. Annan funktionalitet och design prioriterades därför över användandet av Python.

Koden som skrivs körs enbart då användaren trycker på knappen markerad ”starta”. Det valdes framför att köra koden direkt när den kan köras för att garantera att koden inte ändras genom nivån. För att visualisera och tydliggöra detta blir editorn grå då koden körs och inga ändringar i koden tillåts.

De två första nivåerna av spelet implementerades för första utvärderingen. Första nivån har hållits enkel och kräver ingen kod från användaren. Främsta syftet med första nivån är att introducera spelet och gränssnittet. Målet för samtliga nivåer är att ta karaktärerna till toppen av spelplanen. Att ha samma mål för samtliga nivåer gör spelet enkelt och konsekvent. I första nivån ska karaktären som finns, en zombie, ta sig till toppen av spelplanen med den redan implementerade funktionen *move*. Spelets användare kan åstadkomma detta genom att trycka på startknappen. Andra nivån har samma karaktär men med en sten på spelplanen, vilket innebär att samma implementation av funktionen inte längre kommer att fungera. För att klara nivån behöver spelaren därmed genomföra en mindre ändring i funktionens implementation. Konceptet *if-sats* introduceras för att kunna gå runt stenen. Syftet med andra nivån är att få spelaren att börja skriva kod, samt att introducera en *if-sats*.

Iterationen av spelet som användes vid första utvärderingen kan hittas här:
https://codefrontgame.github.io/codeFrontDemo/eval1

4.2 Utvärdering

Utvärderingen genomfördes med sex elever i sjunde klass där de först fick prova spelet och sen delta i en intervju samt svara på en enkät. För att skydda respondenternas integritet publiceras inte de transkriberade intervjuerna. Vid starten av utvärderingen introducerade vi oss och förklarade anledningen till vår undersökning. Innan vi började förklarade vi varför vi var där och vilka vi var. Vi tryckte på att de skulle fråga om hjälp om de inte förstod, att detta skulle hjälpa oss att göra spelet bättre i framtiden. För att kompensera att spelet ännu inte gav så mycket hjälp gav vi dem både ett papper som förklarade hur en *if-sats* fungerar (se bilaga C) och chansen att ställa frågor. Under tiden eleverna spelade spelet fanns det två personer i rummet de kunde ställa frågor till om de behövde. Det fanns även en person som antecknade de frågor som ställdes, vad de behövde hjälp med samt hur

de interagerade med varandra och den förklarande text de fått. När de spelat klart spelet intervjuades de parvis kring deras intresse och erfarenhet av programmering samt vad de tyckte om spelet. Frågorna som ställdes finns i bilaga D.

Under testets gång märktes det att eleverna inte läste på pappret de fått utan istället började testa sig fram och hellre frågade oss eller en klasskamrat om hjälp när de inte förstod.

4.2.1 Sammanställning av svar angående programmering

När eleverna svarade på frågor om tidigare programmeringserfarenheter framkom det att de tidigare testat programmering både i skolan och i viss utsträckning på fritiden. De hade i skolan programmerat med Lego, Micro:bit och i spelet Minecraft. Några av eleverna hade även programmerat i samband med kursverksamhet utanför skolan, och en av dem hade programmerat för att fuska i ett spel.

De flesta respondenterna trodde att programmering var användbart för dem, men var inte nödvändigtvis själva intresserade av programmering. De kände att programmering kommer vara användbart för dem då det är en stor del av samhället, men att det också skulle bero på vad de vill göra i framtiden. En av eleverna sa också att programmering inte kommer att vara användbart då hen inte använder datorn så mycket. Vid frågan om intresse fick vi mer varierande svar såsom ”ja”, ”lite” och ”ibland”. En av de svarande uttryckte att den vet många som håller på med programmering.

Eleverna sa också att mycket kan göras med programmering, men beskrev huvudsakligen fysiska gränssnitt när vi frågar om exempel på vad man kan göra. De nämnde exempelvis att programmera robotar och bygga maskiner. En elev nämnde också att man kan programmera spel.

När vi frågade om hur de skulle gå tillväga för att lära sig programmera fick vi blandade svar. Material såsom böcker, spel och videor nämndes, men även aktiviteter såsom programmeringsläger och kurser. Två av eleverna uttryckte också att de skulle tycka det var svårt att lära sig själv och det är bra med hjälpen man får när man lär sig programmera i skolan.

4.2.2 Sammanställning av svar angående spelet

Alla elever tyckte att det var roligt att testa spelet och uttryckte att de gärna skulle ha spelat mer. De sa specifikt att det var roligt att kontrollera zombien och att man själv fick välja hur man skulle lösa uppgiften. Ett par uttryckte också att det var kul för att man kände att man behövde lösa problemen som presenterades, och att det var bra att ha en kompis i närheten så att man kunde hjälpas åt. En av eleverna uttryckte sig så här: ”Det kändes inte riktigt som ett spel, eller jo på ett sätt, men det var såhär: man ville försöka lösa det, hur man tar sig fram, [det är det] som gör [att] det blir mer spännande.”

När vi frågade om vad som var tråkigt påpekade de svarande att de skulle velat ha fler nivåer och en längre spelupplevelse. De sa också att det var lite svårt att förstå i början, innan de läst instruktionerna ordentligt, men att det blev roligt när man väl förstod. De sa också att den hjälp som fanns var bra men att en video hade kunnat hjälpa dem att förstå bättre. Det togs också upp att väderstrecken var svåra att förstå på engelska. Slutligen uttryckte eleverna ändå att spelet var lagom svårt.

När vi frågade om upplevelsen av spelat sa alla deltagare att det var roligt, att de förstod och att zombien gjorde som de ville. En elev gjorde också tillägget att hen kände sig mäktig och smart för att hen kunde kontrollera zombien. Alla deltagare tyckte också att det de fick göra i spelet var riktig programmering, varav en gjorde tillägget att det var enkel programmering.

Deltagarna uttryckte under intervjuerna en förväntan om att zombien skulle följa rutnätet och gå ett visst antal steg framåt eller åt sidan. De berättade även att några av de tidigare erfarenheter de haft av programmering har baserats på att följa ett rutnät eller gå ett visst antal steg i en riktning.

På frågan om vad som saknas i spelet fick vi huvudsakligen utökning av spelet, snarare än ny funktionalitet, som svar. Eleverna ville ha fler hinder, fler nivåer, fler karaktärer och ljudeffekter. Ett par elever ville även kunna skapa egna nivåer och kunna spela varandras nivåer.

4.2.3 Analys

Då eleverna uttryckte att spelet var lagom svårt bör det ha en första nivå som är av samma svårighetsgrad, eller enklare, än den nivå som eleverna testade. Efter den första nivån bör svårighetsgraden långsamt öka eftersom de flesta av koncepten verkade vara nya för eleverna. Eleverna uttryckte även att spelet var väldigt svårt i början men de verkade gilla utmaningen som följde med svårighetsgraden. Vi behöver se till att det finns tillräckligt med hjälp i början av spelet, och att användaren inte känner att det är omöjligt och ger upp. En av de aspekterna vi vill försöka bevara är att spelet känns utmanande, att det är lite svårt men motiverande och roligt.

Att de knappt använde sig av det hjälppapper de fått tror vi kan bero på att de ansåg det enklare att fråga oss, eller varandra, än att läsa på pappret. Pappret innehöll mycket text och det kan ha varit svårt att hitta precis det de behövde. För att spelaren ska ha tillräckligt mycket hjälp om det inte finns någon att fråga vill vi se till att informationen finns lättillgänglig i spelet.

Från det vi såg i undersökningen verkade det som att eleverna helst inte läste längre text om de kunde undvika det. Vi tror därför att det kan vara bra om den text som finns i spelet kommer i kortare stycken. Att bara visa användaren information som är relevant just för tillfället tror vi också kan hjälpa användaren att inte bli överväldigad av texten.

Eleverna verkade förknippa rutnätet med att flytta karaktärerna i diskreta steg, och det verkade även vara denna typ av programmering med diskreta steg som de flesta av eleverna gjort tidigare. Då vårt spel inte fungerar på detta sätt bör denna diskrepans adresseras på något sätt. Antingen bör vi ändra så att karaktärerna rör sig i diskreta steg utefter rutnätet eller så bör rutnätet tas bort. Om karaktärerna ska fortsätta att programmeras så att de rör sig kontinuerligt bör detta förklaras för spelaren på något sätt, då detta sätt att tänka verkade vara nytt för eleverna. Möjligtvis med en video som föreslagits av några av eleverna.

5

Iteration 2: En komplett prototyp med utökad funktionalitet

Under iteration 2 utvecklades spelet vidare till en mer färdig prototyp och en ny utvärdering genomfördes, denna gång med en större omfattning. För att vidareutveckla spelet använde vi oss av den respons vi fått i föregående iteration och vi använde även denna respons för att förbättra intervjufrågorna inför utvärderingen. I detta kapitel beskrivs först hur vi arbetat med designen av den nya versionen av spelet och sedan hur vi gick tillväga under utvärderingen. Därefter sammanställs intervjuerna och en kort analys av svaren presenteras.

5.1 Design

Figur 5.1: En skärmdump av nivå tio från spelet med två olika typer av hinder och två karaktärer som användaren kontrollerar med koden till höger. Ankan hjälper användaren förstå nivån.

I iteration 2 har spelbrädet utvecklats genom att fler karaktärer, fiender och hinder har lagts till (se fig. 5.1). Fler karaktärer och hinder var något som efterfrågades i föregående utvärdering. Själva bakgrunden till spelplanen har också utvecklats genom att rutnätet tagits bort då det skapade förvirring och stilen på grafiken ändrats för att passa ihop bättre med karaktärer och hinder. Många av de spelelement som ut-

vecklats har dock inte inkluderats i de nivåer som används vid andra utvärderingen då fokuset har varit på att göra en lagom ökning av svårighetsgrad mellan nivåerna istället för att visa upp alla nya element.

Utseendet på resten av spelet har också ändrats för att passa ihop med spelplanen. Anpassningen av spelets utseende har gjorts genom att spelplanen ramats in med en bakgrund som sträcker sig över hela skärmen. Även färger har ändrats och nya ikoner har tillkommit för att göra gränssnittet mer enhetligt och tilltalande.

Nu ger även programmet tydligare respons på vad användaren gör så att användaren ska känna sig uppmuntrad att fortsätta och känna att spelet reagerar på vad hen gör. Den tydligare responsen har åstadkommits genom att lägga till grafisk respons då användaren klarar en nivå och genom att knappar samt ikoner ändrar utseende när de vidrörs. Det finns också en berättarkaraktär i form av en anka som ger beröm och hjälper användaren på alla banor.

Det finns förutom ankan flera andra element som hjälper användaren. Användare kan få ledtrådar till hur uppgiften skall lösas genom att trycka på en knapp och har tillgång till en informationsbank i spelet i form av en bok som redogör för nödvändiga programmeringskoncept så att det inte finns något behov av att leta information från externa medier. Den förklarande text som fanns till varje funktion i iteration ett finns även kvar och har utvecklats för att vara tydligare och mer precis eftersom den uppskattades vid första utvärderingen. Det finns även en återställa-knapp för varje funktion för att återställa koden till hur den såg ut i början av nivån. Knappen ska hjälpa användaren att hitta tillbaka om hen ändrat för mycket och inte vet hur funktionen ska se ut längre.

Det har också ändrats hur användaren kontrollerar karaktärerna. I den här versionen används programmeringsspråket Python istället för Javascript som användes i första iterationen. Användaren behöver även kontrollera flera karaktärer samtidigt på vissa nivåer för att påvisa hur mångsidig den kod de producerar kan vara.

En planerad följd av nivåer har också tillkommit (se bilaga G). Nu finns tolv nivåer med en successivt ökande svårighetsgrad. Nivåerna har designats för att lära ut programmeringskonceptet *if-sats*. Att lära ut ett enda programmeringskoncept bedömdes som lagom i omfattning för den begränsade utvärdering som genomfördes.

Iterationen av spelet som användes vid andra utvärderingen kan hittas här: https://codefrontgame.github.io/codeFrontDemo/eval2

5.2 Utvärdering

Utvärderingen genomfördes med 19 elever från samma årskurs som för förra utvärderingen. Även denna gång fick de även spela spelet och sedan delta i en semistrukturerad intervju, som denna gång var i grupper av tre till fem elever. Frågorna som ställdes under denna utvärdering finns i bilaga H. Efter intervjun fick de också svara

på en enkät gällande deras bakgrund och intresse för spel- och programmeringsintresse.

Innan eleverna fick börja spela spelet så höll vi i en kort presentation om hur spelet fungerade eftersom vi under den förra utvärderingen märkte att vissa tyckte att det var svårt att förstå. Vi hade tänkt att ha en video som gjorde den förklaringen i spelet istället eftersom vi i förra utvärderingen lärde oss att de gärna lär sig genom att kolla på videor. Eftersom vi inte hann med att spela in denna video så gjorde vi istället den tidigare nämnda presentationen. Vi uppmuntrade eleverna att använda sig av de hjälpsystem som fanns i spelet men gjorde det även tydligt att vi fanns tillgängliga för att hjälpa till med saker som kändes svåra även efter att man konsulterat spelets hjälpsystem. Under tiden eleverna fick testa spelet gick vi runt i rummet och gav ledtrådar till de som behövde det.

Inför denna utvärdering hade vi utvecklat fler hjälpsystem i spelet som eleverna skulle kunna använda sig av i respons till det stora antalet frågor som vi fick under första utvärderingen. Hjälpsystemen i spelet verkade dock inte ha hjälpt så mycket som vi hade hoppats på då vi fortfarande fick många frågor. Det uppdagades även snabbt att eleverna inte skulle hinna med alla nivåer i spelet och vi hjälpte dem därför lite mer med nivåerna för att se till att de i alla fall hann en tredjedel av nivåerna. Vi märkte dessutom att en av eleverna gjorde annat istället för att delta men eftersom alla deltog frivilligt så såg vi inte någon mening med att försöka få personen att delta mer aktivt.

5.2.1 Sammanställning av svar angående programmering

Eleverna hade programmerat i skolan men bara ett fåtal hade programmerat hemma. De nämnde Bee-bot, Micro:bit och programmering av LEGO-figurer som den programmering de tidigare gjort i skolan. De som hade programmerat på fritiden hade använt code.org, varit på läger eller programmerat med en förälder.

Vad gäller intresset av programmering var det blandat. Det som vissa tyckte var intressant var att få styra figurer eller att skapa kod. För de som inte tyckte att det var intressant handlade det mer om att de tyckte programmering är svårt.

De såg programmering som användbart. En elev nämnde att det inte riktigt fanns gränser för den kod man kunde skriva. En annan uttryckte att det behövs i fabriker. En tredje nämnde att man kan göra datorer och utveckla tekniken med programmering.

Även om eleverna ser det som användbart tror de inte nödvändigtvis att de kommer använda sig av programmering i framtiden. Vissa påstod att det berodde på vilket jobb man hade medan andra menade på att det inte gjorde det. Några elever menar att man kommer behöva viss kunskap för att samhället kommer vara så tekniskt. Andra trodde inte att de skulle använda sig av programmering. En del som trodde att de skulle programmera i framtiden kunde inte motivera varför.

För att lära sig mer programmering uttryckte många att de skulle utbildat sig vidare på gymnasial och eftergymnasial nivå. Andra nämnde att de kunde lära sig av personer i sin omgivning. En elev nämnde codeFront som tillvägagångssätt. I skolan tycker en elev att man ska ha mer roliga uppgifter som programmeringen av LEGO de gjort innan eller liknande det vi gjorde. Eleven tyckte det var roligt att en grupp kom och hälsade på och att de fick testa någonting gruppen gjort.

Deltagarna beskriver en programmerare som någon som skriver massa kod, kan hacka och är smart. En beskriver programmerare som någon som är som Einstein. Två elever beskriver en programmerare som någon som kan kod utantill.

5.2.2 Sammanställning av svar angående spelet

De flesta deltagarna tyckte att spelet var roligt, några tyckte det var svårt och ett par tyckte det var både och. Några elever svarade att de kände sig dumma för att de inte förstod medan andra sa att de kände sig smarta när de klarade att lösa problemen. Ett par elever sa att de kände sig som programmerare men de flesta gjorde inte det. Resten meddelade att de inte hade en känsla åt det ena eller det andra hållet.

Många visste inte riktigt vad som gjorde spelet roligt, men vissa sa att något som var roligt var att klara nivåer. Många pratade också om att de inte tyckte att det var jättekul att klura på problemen eftersom de var svåra men att när de lyckades lösa dem så var det kul. Nästan alla som fick frågan svarade att de spelat mer om de hade fått, en visserligen bara i skolan.

Eleverna var överens om att det var ett spel de hade testat men hade väldigt svårt att besvara vad ett spel faktiskt är. Några motiverade uttalandet med att det fanns nivåer och hinder som man skulle överkomma. Andra motiverade det med att det fanns karaktärer.

När vi frågade om de tyckte att det fanns ett tydligt mål så svarade de allra flesta att de tyckte målet var att lära sig programmering och många menade att ta sig till slottet också var målet. Det var dock inte alla som visste att det var ett slott och kallade det istället för ”elden”.

Majoriteten av deltagarna menar att de hade kontroll över spelet och mer specifikt över karaktärerna och hur de rörde sig. De andra menade att de inte hade kontroll över spelet just eftersom karaktären inte gjorde som de ville eller för att de inte riktigt förstod hur vissa saker fungerade (exempelvis *else-nyckelordet*). Eleverna meddelande även att de märkte att de påverkade spelet genom att de såg karaktären röra sig på skärmen och att de ibland fick felmeddelanden. De meddelade även att de märkte att deras kod fungerat då de fick positiv respons från ankan, som berättade för dem att de gjort ett bra jobb.

De flesta av deltagarna tyckte inte att de skapade någonting i spelet förutom en elev som menade att de skapade vägen som karaktärerna tog.

När vi frågade om de fick den hjälp de behövde tyckte någon att användning av x och y i förklaringarna var förvirrande och förstod inte förrän en kompis berättade. Vissa läste boken och vissa sa att de inte riktigt förstod det som stod i boken. De som fick frågan tyckte att hjälpen kom när de ville ha den. En person nämnde att hen möjligtvis fått lite för mycket hjälp på första nivån, men att resterande hjälp varit på lagom nivå. Vissa av deltagarna tyckte ankan gav dem tillräckligt mängd hjälp. Andra nämnde att de använde sig av boken istället. En tyckte att ledtrådarna var svåra att förstå. Andra tyckte att de var hjälpsamma.

De flesta av eleverna tyckte att de kunde testa sig fram men några menade att de bara fick felmeddelanden och att det inte riktigt alltid hjälpte. De flesta av de som fick frågan trodde nog att det fanns mer än en lösning men de kunde inte alltid komma på exempel. Några tog upp exemplet med att karaktären kan gå höger eller vänster om hindret på andra banan. Vissa menade på att olika sätt att lösa problemen kunde vara att fråga en kompis eller kolla i boken.

De allra flesta tyckte att spelets nivåer var lagom svåra och att de blev svårare i lagom takt. Men några tyckte de blev lite för svåra lite för snabbt. De flesta som fick frågan om de kunde återanvända sig av kunskap från tidigare nivåer svarade nej, att de inte fick använda sig av det de lärt sig tidigare. Men några tyckte att de till exempel fick använda samma typ av parametrar som i tidigare nivåer och sedan klura ut vad den nya skulle heta.

De tillfrågades även om de tyckte att spelet hjälpte dem att förstå *if-satser*, vilket majoriteten svarade nej på. Dock svarade några lite eller typ och en eller två sa ja. Vissa sa att de kanske hade förstått om de hade haft mer tid på sig att spela. Många tyckte dock att det hjälpte att se karaktärerna röra på sig.

När vi frågade om hur det kändes att styra karaktärerna så svarade några att de kände att de hade makt, att det var ”najs” och några andra tyckte bara det kändes bra. Alla som svarade förutom en svarade att karaktärerna gjorde som de ville, den andre svarade att hen inte kunde skriva koderna och därför fungerade det inte. De allra flesta menade att de ”skrev koden” för att få karaktärerna att röra på sig. Någon menade att koden var tvungen att vara rätt också.

Vi frågade även hur det kändes att skriva instruktioner och några svarade att de tyckte det kändes bra och att det var skönt, och kanske nödvändigt, att ha en grund att stå på (den redan definierade koden). En annan beskrev att hen tyckte att det kändes bra när det gick bra och dåligt när det gick mindre bra. De som svarade på frågan om de tyckte detta var programmering tyckte att det var det men en tyckte att det var mer av en spelvariant. De som tyckte det var programmering motiverade det med att de skrev kod.

Vi frågade hur den här programmeringen skiljer sig från den deltagarna gjort tidigare. Vissa tyckte att man behövde vara smartare för vårt spel och att man behövde tänka längre. Andra tyckte att vårt program var ett spel man spelade medan det de provat innan var mer av att göra ett spel istället. Någon tyckte det var ”en massa

parenteser" i vårt spel jämfört med tidigare erfarenhet. Någon sa att vårt var "mer på riktigt" för att man fick skriva koder medan den tidigare erfarenheten var att trycka på knappar för att få saker att röra på sig. Någon tyckte det var mycket svårare. Några tyckte att deras tidigare erfarenhet med LEGO-gubbar var roligare än vårt spel, men att det vi lät dem testa var något man lärde sig mer av.

Vi frågade sen vad de tyckte var svårt att förstå. Några tyckte svårt att förstå koden till en början. Andra hängde inte riktigt med på hur *if-satser* fungerar och vad man ska skriva efteråt. En sa att det här med väderstrecken inte var jättetydligt förklarat men förstod när hen skrev fel.

Vi frågade vad de tyckte om att testa att programmera i spelet. Många tyckte att det var kul men att det var svårt på sina ställen, ibland många ställen. Vissa tyckte det var svårt att veta *vad* och *hur* man skulle skriva, antagligen tyckte de att det var svårt med syntax. Det var kul när man lyckades lösa problemet.

Sedan frågade vi vad de hade velat kunna göra i spelet, och fick som svar att några ville ha special-attacker. Någon önskade även saker (items) till karaktärerna och uppgraderingar. Det var även eftertraktat att spelet skulle ha musik och ljudeffekter. Vissa ville också kunna hoppa och rulla (som i Subway Surfers - mobilspelet).

Vidare på samma tema frågade vi vad mer de hade velat lära sig i spelet. Vissa hade velat lära sig mer om booleska uttryck ($x > 5$ till exempel) för de kände inte att de förstod det. Vissa ville lära sig mer än bara *if-satser*. En annan ville lära sig om det fanns "fler sätt att programmera [...] istället för att bara skriva *return*, *west*, *north* och sådär". Några ville lära sig att applicera det som de lärt sig nu utanför spelet. Någon ville lära sig om koordinater för att kunna styra en robot.

5.2.3 Analys

Intresset för programmering var varierat och de som inte var intresserade motiverade det med att de tyckte det var svårt. Bristen av intresse för programmering, på grund av uppfattningen att programmering är svårt, ser vi pekar på vikten i att ge ungdomar en bra plats att börja programmera på. Om det första de gör inom programmering är något som är för svårt kan detta möjligtvis hämma utvecklingen av ett intresse. Trots att inte alla var intresserade av det så instämde alla med att programmering var användbart men de flesta trodde att de inte kommer använda sig av det i framtiden.

En majoritet svarade att de inte kände att de förstod vad en *if-sats* var för någonting efter testet. Detta tror vi dock kan bero på att de inte hade tillräckligt med tid för att ta sig igenom alla nivåer som hanterade *if-satser*. Då ingen hann med ens hälften av nivåerna är det omöjligt att säga huruvida det material vi har med i spelet är tillräckligt för att ge en enkel förståelse av en *if-sats*. Dock är repetition bra för att befästa kunskaper, så användaren bör få skriva *if-satser* även i kommande nivåer.

Trots vårt genomtänkta val av programspråk för eleverna så tyckte vissa att det var svårt att veta *var* eller ens *hur* de skulle skriva sin del av koden. En bättre förklaring av programspråkets syntax kanske hade hjälpt dem att förstå bättre. En smart editor hade kanske också kunnat vara till hjälp med detta. De som var förvirrade kände också att spelet var mindre kul på grund av detta. Trots detta verkade alla vara överens om att det var kul när man löste ett problem och klarade en nivå. Detta tyder på att eleverna kände sig utmanade av spelet och att de fick en känslomässig belöning när de gjorde framsteg.

De allra flesta av eleverna tyckte att nivåerna de kom till var lagom svåra och att de blev svårare i lagom takt, trots att många beskrev att de haft problem i början. Vi tror att detta kan tyda på att eleverna inte upplevde att hindren i nivåerna var svåra utan att det var själva syntaxen och programmeringstänket som var svårt. Att problemet låg i att de inte förstod verktygen det var meningen att de skulle använda snarare än att hindren var för svåra. Det kan därför vara bra att lägga ännu mer tid till att se efter hur hjälpen kan utformas på bästa sätt så att eleverna både kan och vill ta till sig den. Spelet uppfattades som svårare men mer givande än de tidigare erfarenheter de haft med programmering. De tyckte dessutom att vårt spel var mer ”på riktigt”.

Som tidigare nämnt så försökte vi inkorporera frågor baserat på vad James Paul Gee anser finns i bra spel inför det sista intervjutillfället. De aspekter vi valde att fråga om var interaktivitet, skapande, risktagande, anpassningsbarhet, självbestämmande, identitetsskapande, utmaning, återanvändning av kunskap och att få hjälp vid behov. Från de svar som vi fick verkar det som att vi lever upp till några av dessa.

Några exempel på saker som kom upp under intervjuerna var ”Det var roligt att man fick bestämma helt själv”, ”När man väl knäckte vad man skulle göra så fick man lite adrenalin” och ”Jag kände mig smart när jag löste en uppgift”. Det första exemplet tyder på att spelet gav en känsla av självbestämmande, var interaktivt och anpassningsbart i den mån att det fanns flera sätt att lösa problemen. De andra två exemplen tyder på att spelet var utmanande, interaktivt och ingav en känsla av självbestämmande.

Eleverna tyckte att de fick hjälp när de ville ha den, även om hjälpen kanske var otillräcklig. De tyckte också att de kunde testa sig fram och därmed ta risker. De tyckte dock inte att de fick använda sig av det de tidigare lärt sig i spelet vilket är intressant då detta var tanken. Det skulle kunna bero på att de kanske inte riktigt kände att de lärde sig någonting varje nivå, eller att de inte hann spela de nivåer som specifikt var till för att spelaren skulle få återanvända sin kunskap. Endast några få kände sig som en programmerare när de spelade spelet, vilket var det vi hoppades att de skulle identifiera sig med. Detsamma gällde känslan av att skapa, vilken endast några få fick då de skrev kod.

Intervjuerna visade också att eleverna hade önskat ljudeffekter och/eller musik i spelet. De ville även ha specialattacker och att man kunde hoppa och rulla ”så som i

Subway Surfers”. Någon nämnde även att de ville ha saker (items) till karaktärerna. De hade också gärna velat lära sig mer programmering i spelet, till exempel ville någon lära sig mer om booleska uttryck.

Det framgick också att spelet upplevdes som roligt med respons som ”Ett roligt spel” och ”Det var roligt att man fick bestämma helt själv”. Det verkar som att eleverna tyckte att utvecklingen rörde sig åt rätt håll.

Enkäten visar att av de 15 som svarade så verkade det inte vara någon större skillnad på intresset av spel eller om de tyckte att programmering är användbart mellan killarna och tjejerna. Större skillnad var det däremot när det gäller intresset för programmering med ett medelvärde som skiljde de två med mer än 1 på en skala mellan 1 och 5. Denna skillnad är inte statistiskt säkerställd, men vi anser att den är värd att notera då den stämmer överens med annan forskning på området.

Nämnvärt är även att elva av de som svarade på enkäten rapporterade att de hade minst en förälder med högskoleutbildning. De andra fyra svarade att de inte visste eller att de inte ville svara.

6

Diskussion

Projektets syfte var att undersöka hur intresset för programmering kan främjas. Målet med projektet var att ta fram en spelprototyp för att utforska hur den kan främja intresset för programmering hos högstadieelever. De flesta av eleverna som testade spelet tyckte spelet var kul och utmanande. Det som var roligt var bland annat att kontrollera karaktärer, klara nivåer och lösa problem. Det är positivt och viktigt att spelet är roligt då det förmodligen bidrar till att de tycker att programmering är roligt, och även intressant. Med det sagt, det är svårt att mäta hur mycket spelet främjar intresset och hur bra utformat det är jämfört med andra tillvägagångssätt. Därför ställdes istället frågor i intervjuerna kring hur eleverna upplevde spelet. Detta för att undersöka om vår typ av spel kan fungera för att främja intresset.

Av de elever som intervjuades, som tyckte att programmering inte var intressant, berodde bristen av intresse på att de tyckte att programmering var svårt. Många av eleverna tyckte även att vårt spel var svårt, speciellt i början. Därför borde svårighetsgraden vara något vi fokuserat mer på, då det kan vara en anledning till att spelet inte skulle ha främjat intresse för programmering. Det vi hade kunnat göra för att anpassa svårighetsnivån bättre är att ge ännu mer hjälp till användaren, ha en mer hjälpsam editor eller göra nivåerna och nivåprogressionen enklare. Hur man designar hjälpsystem som hjälper användaren på ett effektivt sätt är något som kan utforskas ytterligare. De inbyggda hjälpsystemen var något som eleverna inte använde så mycket. De hade kanske kunnat fungerat bättre om hjälpen syntes mer eller till viss del visades även när användaren inte bett om hjälp. Det gäller åter igen att hitta balansen mellan vad användaren ser som hjälpsamt och vad som är irriterande.

Utvärderingen visar även att den programmering eleverna fick göra i spelet skiljde sig från den programmering de gjort tidigare i skolan. Programmeringen i spelet kan ses som att funktioner körs flera gånger per sekund där nästa beslut för karaktärerna ska tas. Den tidigare programmeringen eleverna har gjort innebar att ge en sekvens av instruktioner som ska lösa hela problemet. Sekvensen är ofta så pass specifik att den löser just det specifika problemet men inte andra liknande problem. Typen av programmering i vårt spel kan därför ses som mer generell och algoritmisk, då samma kod kan lösa flera problem. Exempel och ytterligare förklaring kan läsas i bilaga A. Vissa beskrev programmeringen de använde i vårt spel som mer ”riktig” än den programmering de använt tidigare. De nämnde även att de behövde bry sig mer om syntax. Det högre kravet på korrekt syntax kan vara en anledning till att några av eleverna tyckte att den programmering de gjorde i spelet var svårare och krävde att man var smartare och tänkte längre. Några tyckte att programmeringen de gjort med LEGO-figurer i skolan var roligare, men att man lärde sig mer från vårt spel. Eleverna skiljde även på att de tidigare har skapat spel snarare än att spela

spel. Att programmeringen i vårt spel skiljde sig en del från vad de gjort innan kan ha bidragit till att det var svårt. Det kan också vara något som gör spelet intressant, då de får lösa problem på ett annat och mer generellt sätt samt att de får se flera olika programmeringssätt och tillämpningar.

Att både introducera en ny typ av programmering och att använda ett språk som är textbaserat utan en hjälpsam editor kan ha orsakat frustration och svårigheter för eleverna. Det skulle kunna ge ett annat, och kanske mer tydligt, resultat om man endast undersökte effekten av att introducera en ny sorts programmering eller endast effekten av att använda ett nytt textbaserat språk. Mycket kan bero på att editorn inte hjälpte till med felmeddelanden, så om det hade fungerat bättre kanske det som var nytt inte hade varit ett stort problem.

Det var oväntat att resultatet från utvärderingarna säger att eleverna tyckte att det de gjorde i spelet definitivt var programmering, att de värderade att skriva kod och att de tyckte att problemlösningen var unik. Resultatet visar att man med små medel kan få exempelvis problemlösning att kännas unik och rolig. Ett exempel på vad som upplevdes som unika lösningar från en nivå är att bestämma om karaktären skulle svänga åt vänster eller höger när den närmar sig en sten. Det är även intressant med tanke på att det innebär att man kan presentera enkla och liknande problem som upplevs lagom svåra, intressanta och unika, snarare än repetitiva och tråkiga.

Något som är värt att notera är att nästan alla i testgruppen svarade att de har minst en förälder med högskoleutbildning och att det inte fanns någon som svarade att ingen av deras föräldrar har högskoleutbildning. Då barn med högskoleutbildade föräldrar generellt har det enklare i skolan skulle det kunna innebära att vårt spel skulle uppfattas som ännu svårare av en genomsnittlig 14-åring. Gruppen innehöll både tjejer och killar, och barn med inrikes- och utrikesfödda föräldrar. Vi kunde inte se någon större skillnad mellan tjejer och killar i intresse av spel, och om de tyckte att programmering är användbart. Men vi såg en skillnad mellan de tjejer och killar som svarade i huruvida de anser sig vara intresserade av programmering. Ett spel, eller annat program, som är speciellt riktat mot tjejer vad gäller genre och design skulle därför kunna ha större inverkan gällande intresse för programmering. Att ha ett spel i en genre som det är vanligare att tjejer spelar och som använder sig av programmering skulle kanske kunna hjälpa tjejer att se programmering som ett intresse.

Utvärderingarna hade kunnat utföras på ett annat sätt, vilket hade kunnat ge ett bättre resultat. Eftersom eleverna inte hann spela hela spelet under utvärderingarna fick vi inte möjlighet att testa hela spelupplevelsen. Att vi var med i klassrummet och svarade på elevernas frågor innebär att vi inte vet fullt ut hur spelets hjälp fungerar, eller om den påverkade elevernas inställning till spelet. Ett alternativ hade kunnat vara att eleverna fick testa spelet hemma över en viss tid och sedan prata med oss. Fördelen med detta hade varit att de hade varit själva med spelet, vilket är mer troget till hur det är tänkt att spelet ska spelas. En nackdel är att vi inte hade kunnat hjälpa till när eleverna stötte på problem, vilket skulle kunna leda till extra frustration och att de då slutade i ett tidigt skede av spelet.

Att intervjuerna genomfördes i grupp kan också ha påverkat resultatet. Vi upplevde att det mestadels fungerade bra, men att det var något färre diskussioner än vi förväntat oss, samt att det var svårt att höra allas åsikt. Mängden diskussion kan bero på nervositet, då de elever som intervjuades vid båda tillfällena diskuterade mer och verkade mindre nervösa. Vi ser inget bättre alternativ till gruppintervjuerna, då individuella intervjuer hade varit mer tidskrävande. Indelningen av grupper kan också påverkat. Grupperna delades upp efter hur de satt eller vilka som blev klara med spelet samtidigt. Det hade kunnat vara intressant att exempelvis dela upp dem så att alla i gruppen känner varandra väldigt väl, alternativt att de i gruppen inte vanligtvis pratar med varandra.

Ytterligare en aspekt som påverkade resultatet är att frågorna för första utvärderingen inte var grundade i teori. Istället baserades frågorna på det vi ville veta, och tänkte skulle vara relevant, samt vad som skulle hjälpa oss i den fortsatta utvecklingen. Att vi inte baserade frågorna på teori inom området kan ha lett till att vi till exempel missat att fråga viktiga frågor. Eftersom frågorna för andra utvärderingen baserades på teori innebär det även att frågorna från första och andra utvärderingen skiljer sig åt, och det blir därmed svårare att jämföra resultaten.

Det finns flera svårigheter i projektet som kan ha påverkat resultatet. En utmaning med utvärderingarna var planeringen av dessa. På grund av tidsbrist och svårigheter med planering blev den första tänkta utvärderingen, med en fokusgrupp, inställd. En annan svårighet var att veta vilken kunskapsnivå högstadieelever är på gällande programmering. Där hade det första tillfället med en fokusgrupp kunnat hjälpa till, alternativt att en förstudie hade gjorts.

Det behövs även en bred kompetens för att genomföra ett projekt som berör så många områden. Det behövs kunskap inom bland annat pedagogik, spelutveckling och intervjumetodik. När det kommer till utveckling av spelet var en utmaning att editorn vi använde inte var tillräckligt smart och inte heller gav så mycket hjälp. Det var inget som fokuserades på i projektet, men det kan ha varit en anledning till att spelet upplevdes vara svårt.

6.1 Samhälleliga och etiska aspekter

Redan i början av projektet insåg vi att etiska aspekter skulle vara något som vi behövde ta hänsyn till. Både på grund av att vi hade som mål att utvärdera vår produkt på högstadieelever, men även då spel kan ha en del negativa sidoeffekter. Detta medförde alltså att vi hade etiska aspekter i åtanke under hela utvecklingsprocessen.

Under utvecklingen av spelet lästes mycket material som handlade om negativa och positiva aspekter av olika spelelement. Detta är något som har legat till grund för vad som har och inte har implementerats och har hjälpt till att undvika negativa designelement i spelet. Till exempel har vi undvikit att implementera en topplista då den kunde leda till ett oönskat tävlingsmoment. Användningen av ett element

som en topplista kan ha effekten att spelare som hamnar högt upp på listan arbetar mer för att bibehålla sin position, vilket i sin tur kan leda till att spelare lär sig mer av materialet som presenteras. Dock kan det också få effekten att spelare långt ner på listan förlorar motivation till att lära sig, eftersom det känns som en lång uppförsbacke att klättra upp i topplistan. Att fokusera på att behålla eller klättra i positioner på en topplista är också något som kan bidra till ett beroende på ett dåligt sätt och är något som vi vill undvika. Därför valde vi att inte inkludera en topplista då vi inte ville bidra till att spelare tappar lusten att lära sig samtidigt som vi vill undvika att spelare fastnar i ett beroende.

Vi har implementerat tydlig respons på när man gör något bra, för att uppmuntra spelaren att fortsätta. Denna respons är i form av beröm när en spelare klarar en nivå. Dock är prototypen fortfarande ganska enkel och det har därför inte varit allt för stora problem med att undvika negativa element, och samtidigt implementera positiva element. Om mer arbete skulle lagts ner på utvecklingen skulle dock detta vara något som skulle behöva tas mer hänsyn till.

Eftersom vi utförde intervjuer var det också viktigt att vi såg till att deltagarnas personliga integritet skyddades, och detta blev särskilt viktigt på grund av att det handlade om högstadieelever. Därför togs flera åtgärder för att skydda den personliga integriteten. Inför intervjutillfällena skickades ett missiv ut till skolan och vårdnadshavarna för att se till att vi fick tillstånd att intervjua eleverna. Intervjuerna blev inspelade med elevernas tillåtelse och transkriberade. Efter transkriptionen var klar raderades dessa inspelningar. Transkriptionerna publiceras inte heller i denna rapport, utan endast sammanställningar av dessa, för ytterligare en grad av anonymitet. Att vi har fått återkoppling från vår tänkta målgrupp under projektets gång har gett mycket bra information om hur både vi och andra kan arbeta vidare. Samtidigt så är det, som sagt, väldigt viktigt att den personliga integriteten hos de som ger respons bevaras. Som påpekats lade vi ner mycket fokus på att hantera den information som samlas in på ett ansvarsfullt sätt. Vid vidare arbete på detta projekt skulle fler intervjusessioner vara viktiga att utföra, och det skulle då också innebära ett fortsatt fokus på personlig integritet.

Vi konstaterade att detta projekt hade möjligheter att bidra med både nytta och skada. Den samhälleliga nyttan utforskades samtidigt som den möjliga skadan minimerades så mycket som möjligt. När tidigare material inom detta område studerades fick vi en överblick över vilka saker som kan vara problematiska, vilket gav oss en riktning under projektets gång.

7

Slutsats

Spelet codeFront som skapades i projektet är ett seriöst spel (serious game) som tagit inspiration från metoder som används inom spelifiering (gamification). Hur spel skulle kunna användas för att främja intresse för programmering undersöktes genom två utvärderingar vid olika tillfällen i utvecklingsprocessen.

Resultatet från utvärderingarna indikerar att ett spel som codeFront kan vara ett roligt sätt att öka intresse för programmering. Att skriva kod, lösa problem och att klara nivåer var det eleverna tyckte var roligast när de testade spelet. Resultatet visar att den typ av programmering som använts i spelet upplevdes som ”riktig” programmering och mer lärorik jämfört med elevernas tidigare erfarenheter av programmering. Exempel och ytterligare förklaring av skillnaden mellan programmeringen i vårt spel och programmeringen eleverna tidigare gjort kan läsas i bilaga A. Däremot har utvärderingarna gjorts med få deltagare, vilket betyder att ingen definitiv slutsats kan dras kring vilken typ av programmering som bäst främjar programmeringsintresse. Dessutom har ingen rigorös mätning av intresse för programmering gjorts i projektet. Dels för att vi inte har haft något basläge att utgå från, och dels för att det är svårt att mäta hur vår spelprotoyp påverkar intresset för programmering.

För att utveckla den här typen av program och spel behövs en bred kompetens inom bland annat pedagogik, grafik och spelutveckling. Det finns flera komponenter, som alla behöver fungera bra, för att användaren ska få en bra upplevelse. Den kanske viktigaste komponenten är editorn, som behöver hjälpa användaren att följa syntax. Dessutom behövs kännedom om målgruppens kunskapsnivå för att undvika att spelet blir för svårt och frustrerande, eller för lätt och därmed tråkigt. Det krävs dessutom mycket tid till intervjuer och användartester för att genomförandet av dessa ska ske på ett bra sätt.

De positiva resultaten från utvärderingarna visar att det finns mer att utforska inom området. Vi ser därför att mer forskning behöver göras om spelifiering och seriösa spel när det kommer till att lära sig, och främja intresset för, programmering. Vi tycker även att det behövs fler studier som undersöker om den mer algoritmiska typen av programmering kan få mer positiva effekter i jämförelse med programmering baserat på sekvenser av instruktioner, specifikt för seriösa spel. En liknande studie i större omfattning skulle kunna undersöka de positiva indikationerna som vi noterat från detta projekt.

Källförteckning

[1] The Directorate-General for Communications Networks, Content and Technology. (2018). Coding - the 21st century skill. [Online], Tillgänglig: https://ec.europa.eu/digital-single-market/en/coding-21st-century-skill (hämtad 2019-02-12).

[2] Regeringskansliet. (2017). Stärkt digital kompetens i läroplaner och kursplaner. [Online], Tillgänglig: https://www.regeringen.se/pressmeddelanden/2017/03/starkt-digital-kompetens-i-laroplaner-och-kursplaner/ (hämtad 2019-02-12).

[3] Skolverket. (2019). Fler datorer i skolan men teknikkrångel skapar problem. [Online], Tillgänglig: https://www.skolverket.se/om-oss/press/pressmeddelanden/pressmeddelanden/2019-02-20-fler-datorer-i-skolan-men-teknikkrangel-skapar-problem (hämtad 2019-04-01).

[4] ——, (2019). Att programmera - webbkurs. [Online], Tillgänglig: https://www.skolverket.se/skolutveckling/kompetensutveckling/att-programmera--webbkurs (hämtad 2019-04-26).

[5] S. Watson och H. R. Lipford, "Motivating students beyond course requirements with a serious game", i *SIGCSE 2019 - Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, [Online], 2019, s. 211–217. DOI: 10.1145/3287324.3287364.

[6] M. A. Miljanovic och J. S. Bradbury, "A review of serious games for programming", i *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, [Online], 11243 LNCS, 2018, s. 204–216. DOI: 10.1007/978-3-030-02762-9_21.

[7] CodeCombat. (2019). About. [Online], Tillgänglig: https://codecombat.com/about (hämtad 2010-05-10).

[8] CodeMonkey Studios Inc. (2019). CodeMonkey. [Online], Tillgänglig: https://www.playcodemonkey.com/ (hämtad 2010-05-10).

[9] Apple Inc. (2019). Swift Playgrounds. [Online], Tillgänglig: https://www.apple.com/se/swift/playgrounds/ (hämtad 2010-05-10).

[10] Sveriges Television AB. (2016). Julkalendern: Selmas saga. [Online], Tillgänglig: https://itunes.apple.com/se/app/julkalendern-selmas-saga/id1178555993?mt=8 (hämtad 2010-05-10).

[11] Filimundus AB. (2016). Coda Game - Bygg dina egna spel. [Online], Tillgänglig: https://itunes.apple.com/se/app/coda-game-bygg-dina-egna-spel/id1031841567?mt=8 (hämtad 2010-05-10).

[12] ScratchJr. (2019). Om ScratchJr. [Online], Tillgänglig: https://www.scratchjr.org/about/info (hämtad 2010-05-10).

[13] J. M. Wing. (2014). Computational Thinking Benefits Society. [Online], Tillgänglig: http://socialissues.cs.toronto.edu/index.html%3Fp=279.html (hämtad 2019-05-10).

[14] T.-C. Hsu, S.-C. Chang och Y.-T. Hung, ”How to learn and how to teach computational thinking: Suggestions based on a review of the literature”, *Computers and Education*, vol. 126, s. 296–310, 2018. DOI: 10.1016/j.compedu.2018.07.004.

[15] A. Robins, J. Rountree och N. Rountree, ”Learning and Teaching Programming: A Review and Discussion”, *Computer Science Education*, 2003, [Online], ISSN: 0899-3408. DOI: 10.1076/csed.13.2.137.14200.

[16] M. C. Linn och M. J. Clancy, ”The case for case studies of programming problems”, *Communications of the ACM*, vol. 35, nr 3, s. 121–132, mars 1992, [Online], ISSN: 00010782. DOI: 10.1145/131295.131301.

[17] M. Ivanovia och Z. Budimac, ”First programming language - Never-ending story”, i *AIP Conference Proceedings*, [Online], 2013, s. 353–356, ISBN: 9780735411845. DOI: 10.1063/1.4825496.

[18] C. M. Lewis, ”How programming environment shapes perception, learning and goals”, i *Proceedings of the 41st ACM technical symposium on Computer science education - SIGCSE '10*, [Online], 2010, s. 346–350, ISBN: 9781450300063. DOI: 10.1145/1734263.1734383.

[19] Lifelong Kindergarten Group at the MIT Media Lab, *Scratch*, [Online], 2019. Tillgänglig: https://scratch.mit.edu/ (hämtad 2019-02-13).

[20] L. Mannila, M. Peltomäki och T. Salakoski, ”What about a simple language? Analyzing the difficulties in learning to program”, *Computer Science Education*, vol. 16, nr 3, s. 211–227, 2006, ISSN: 0899-3408. DOI: 10.1080/08993400600912384.

[21] M. D. Griffiths och A. Meredith, ”Videogame Addiction and its Treatment”, *Journal of Contemporary Psychotherapy*, vol. 39, nr 4, s. 247–253, maj 2009, [Online], ISSN: 1573-3564. DOI: https://doi.org/10.1007/s10879-009-9118-4.

[22] D. Basten, ”Gamification”, *IEEE Software*, vol. 34, nr 5, s. 76–81, aug. 2017, [Online]. DOI: 10.1109/MS.2017.3571581.

[23] F. R. H. Andrade, R. Mizoguchi och S. Isotani, ”The Bright and Dark Sides of Gamification”, i *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer International Publishing, 2016, s. 176–186, ISBN: 978-3-319-39583-8. DOI: 10.1007/978-3-319-39583-8_17.

[24] F. F.-H. Nah, Q. Zeng, V. R. Telaprolu, A. P. Ayyappa och B. Eschenbrenner, ”Gamification of Education: A Review of Literature”, vol. 8527 LNCS, s. 401–409, 2014, ISSN: 16113349. DOI: 10.1007/978-3-319-07293-7_39.

[25] S. Deterding, R. Khaled, L. Nacke och D. Dixon, ”Gamification: Toward a Definition”, i *CHI 2011 Gamification Workshop*, Vancouver, Kanada, 2011, s. 1–4. Tillgänglig: http://gamification-research.org/wp-content/uploads/2011/04/02-Deterding-Khaled-Nacke-Dixon.pdf (hämtad 2019-04-28).

[26] S. Deterding, D. Dixon, R. Khaled och L. Nacke, ”From Game Design Elements to Gamefulness: Defining “Gamification””, i *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments, MindTrek 2011*, 2011, s. 9–15, ISBN: 9781450308168. DOI: 10.1145/2181037.2181040.

[27] UW Center for Game Science et al, *Solve Puzzles for Science*, [Online], 2019. Tillgänglig: https://fold.it/portal/ (hämtad 2019-04-29).

[28] J. Markoff, "In a Video Game, Tackling the Complexities of Protein Folding", *New York Times*, 2010, [Online]. Tillgänglig: https://www.nytimes.com/2010/08/05/science/05protein.html (hämtad 2019-04-29).

[29] A. Cooper, R. Reimann och D. Cronin, *About Face 3: The Essentials of Interaction Design*, 3:e. Wiley, 2007, ISBN: 9780470084113.

[30] J. Preece, H. Sharp och Y. Rogers, *Interaction Design: Beyond Human-Computer Interaction*, 4:e. Wiley, 2015, ISBN: 9781119020752.

[31] Statistiska centralbyrån. (2018). Andel personer som har tillgång till internet i hemmet. [Online], Tillgänglig: https://www.scb.se/hitta-statistik/statistik-efter-amne/levnadsforhallanden/levnadsforhallanden/befolkningens-it-anvandning/pong/tabell-och-diagram/andel-personer-som-har-tillgang-till-internet-i-hemmet/ (hämtad 2019-04-01).

[32] ——, (2018). Andel personer som använt internet i stort sett varje dag. [Online], Tillgänglig: https://www.scb.se/hitta-statistik/statistik-efter-amne/levnadsforhallanden/levnadsforhallanden/befolkningens-it-anvandning/pong/tabell-och-diagram/andel-personer-som-anvant-internet-i-stort-sett-varje-dag/ (hämtad 2019-04-01).

[33] ——, (2018). Andel personer som aldrig använt sig av internet. [Online], Tillgänglig: https://www.scb.se/hitta-statistik/statistik-efter-amne/levnadsforhallanden/levnadsforhallanden/befolkningens-it-anvandning/pong/tabell-och-diagram/andel-personer-som-aldrig-anvant-sig-av-internet/ (hämtad 2019-04-01).

[34] Internetstiftelsen. (2018). Svenskarna och internet 2018. [Online], Tillgänglig: https://2018.svenskarnaochinternet.se/allmant-om-internetutvecklingen/ (hämtad 2019-04-01).

[35] Statens medieråd, "Ungar och medier 2017", Skolverket, Stockholm, Sverige, tekn. rapport, 2017, [Online], s. 29. Tillgänglig: https://www.statensmedierad.se/publikationer/ungarochmedier/ungarmedier2017.2344.html (hämtad 2010-04-30).

[36] E. B. Witherspoon, C. D. Schunn, R. M. Higashi och E. C. Baehr, "Gender, interest, and prior experience shape opportunities to learn programming in robotics competitions", *International Journal of STEM Education*, vol. 3, s. 1–12, nov. 2016, [Online], ISSN: 21967822. DOI: 10.1186/s40594-016-0052-1.

[37] H. Grönqvist och S. Niknami, "Ankomst och härkomst - en ESO-rapport om skolresultat och bakgrund", Regeringskansliet Finansdepartementet, Stockholm, Sverige, tekn. rapport, 2017, [Online], s. 53. Tillgänglig: https://eso.expertgrupp.se/wp-content/uploads/2014/12/ESO-2017_3.pdf (hämtad 2019-05-02).

[38] P. Fredriksson och J. Sandqvist, "Analyser av familjebakgrundens betydelse för skolresultaten och skillnader mellan skolor", Skolverket, Stockholm, Sverige, tekn. rapport 467, 2018, [Online], s. 13–17. Tillgänglig: https://www.skolverket.se/getFile?file=3927 (hämtad 2019-05-02).

[39] J. P. Gee, *Good Video Games and Good Learning*, [Online], Madison. Tillgänglig: https://academiccolab.org/resources/documents/Good_Learning.pdf (hämtad 2019-05-02).

Bilagor

A

Exempel och förklaring av olika typer av programmering

Här beskrivs de två olika typerna av programmering som övervägdes i projektet.

Programmeringen som utförs i spelet består av att definiera funktioner som beskriver en karaktärs beteende. Funktionen körs flera gånger per sekund. Varje körning kan ses som ett beslut av nästa drag för karaktären. Följande är ett exempel från nivå 1 i spelet där karaktären ska gå runt en sten.

```
def move(isInFrontOfRock):
        if isInFrontOfRock:
                return Response.EAST
        else:
                return Response.NORTH
```

Nedan är ett exempel på en vanlig typ av programmering i seriösa spel som lär ut programmering. Exemplet löser samma problem som kodstycket ovan löser, det vill säga att ta karaktären framåt men runt en sten. Användaren av spelet får definiera en sekvens av instruktioner som löser ett problem. Värt att notera att den här typen av programmering i stort sett kräver att det är tydligt för användaren vad ett steg är och att det förmodligen behövs ett rutnät av något slag för att visualisera koordinatsystemet.

```
def move():
        moveForward()
        moveForward()
        moveForward()
        moveLeft()
        moveForward()
        moveForward()
```

En skillnad mellan dessa typer av programmering är att den första sorten, som också används i spelet, enligt oss oftare blir mer generell problemlösning då samma kodstycke kan lösa flera olika problem. En annan skillnad är att den mer sekvensbaserade programmeringen kan vara lättare att förstå då det förekommer i fler spel och har ett högre krav på visualisering.

B

Missiv

Vi är sex studenter som läser IT på Chalmers. Vi gör nu ett kandidatarbete där vi utvecklar ett spel som ämnar att öka intresset och kunskaper i programmering hos högstadieelever.

I vårt arbete ingår det också att testa och utvärdera vårt spel. Detta görs i tre steg:

1. Diskutera vår idé med en mindre grupp, 4-6 personer

2. Utvärdera spelet halvvägs med samma mindre grupp

3. Utvärdera slutprodukten med en större grupp, gärna en hel klass

Vår målgrupp är som sagt högstadieelever, och för att bättre kunna göra ett spel som uppfyller vårt syfte för målgruppen önskar vi att ditt barn får delta i denna utvärdering.

De frågor vi kommer ställa handlar dels om intresset för programmering och dels om hur de upplever spelet. Vid första och andra tillfället samlas data in genom intervjuer. Vid tredje tillfället sker det genom en enkät.

Intervjun kommer dokumenteras skriftligt. Vi vill även göra ljudupptagning på tillfälle 1 och 2 som sparas tills dess att det har transkriberats. Resultatet kommer användas för att förbättra vårt spel samt utvärdera spelet. Citering kan förekomma men alla barn som deltar kommer garanteras anonymitet.

Dokumentationen kommer vara en del av den rapport som är en del av vårt arbete. Det kommer endast användas akademiskt. Rapporten, med dokumentationen, kommer publiceras under Chalmers tekniska högskola.

För frågor och funderingar är du välkommen att kontakta mig (Agnes) på agnesma@student.chalmers.se

Hälsningar Ludvig Andersson
Lisa Karlsson
William Levén
Agnes Mårdh
Lucas Ruud
Karl Wikström

C

Spelmanual som förklarar en *if-sats*

codeFront

Spelet består just nu av två nivåer. Målet är att få zombien att gå från längst ner på fältet och hela vägen upp, längst norrut.

Tips för nivå 1

Kör koden genom att trycka på knappen längst ner till höger. Försök också förstå vad som händer.

Tips för nivå 2 (Den med en sten i mitten)

Här kommer lite tips för att klara nivå 2.

För att klara den här nivån behöver du också använda dig av en **if-sats** (if som i om på engelska).
En if-sats ser ut så här:

```
if (villkor) {
   // kod 1 här
} else {
   // kod 2 här
}
```

Villkor är i detta sammanhang ett villkor för att kod under if-satsen ska köras. Den används alltså om man bara vill använda viss kod när något speciellt villkor uppfylls. Villkoret uppfylls när det är sant. Annars (om villkoret är falskt) körs koden som står under else.

Man kan läsa en if-sats som "**Om** villkoret är sant vill jag att kod 1 ska köras, **annars** vill jag att kod 2 ska köras".

Exempel på villkor:

- 5 > 3 (läses "5 är större än 3", detta är alltså också alltid sant)
- x > 3 (läses "x är större än 3", det är sant när x är 4 eller mer)
- isMoving (detta är en variabel som beräknas någon annanstans, på så vis kör vi bara koden när något rör sig)

Exempel på if-sats:

```
if (isTalking) {
      return response.north;
    } else {
      return response.south;
    }
```

Detta betyder alltså: **om** någon pratar kommer den gå norrut **annars** söderut .

D

Frågor för intervju vid utvärdering 1

Följande frågor användes som bas vid intervjuerna för utvärdering 1. Då det var semi-strukturella intervjuer kanske inte alla frågor ställdes vid varje intervju och ytterligare frågor kan ha ställts.

- Har ni programmerat innan / på vilket sätt har ni kommit i kontakt med programmering?
- Tycker ni programmering är intressant?
- Hur kändes det att få testa på?
- Vad tänker ni att man kan göra med programmering?
- Tycker ni att programmering känns användbart för er?
  - Varför?
- Tror ni att ni kommer använda er av programmering i framtiden?
- Vad skulle ni göra för att fortsätta lära er programmering?
- Var det roligt att spela spelet?
  - Vad?
  - Varför?
  - Vill ni göra mer?
  - Vad var tråkigt?
    - Varför?
- Hur kändes det att kunna skriva kommandon till zombien och att den gjorde som du skrev?
  - Gjorde den det du ville?
- Förstod du hur du skulle göra för att få zombien att röra sig?

- Är detta programmering?
  - Varför?
- Var något svårt att förstå?
- Vad skulle du vilja kunna göra i spelet?

E

Formulärfrågor vid utvärdering 1

Frågor efter testning av spelet CodeFront

Ditt namn används för att koppla samman dina svar på intervjun med de i formuläret. Du kommer att vara helt anonym (ditt namn och andra identifierande uppgifter kommer att tas bort) i rapporten.

1. **Namn**
2. **Har dina föräldrar en högskoleutbildning?**
    - Båda har en högskoleutbildning
    - En av dem har en högskoleutbildning
    - Ingen av dem har en högskoleutbildning
    - Vet ej / Vill ej svara
3. **Är dina föräldrar födda i Sverige**
    - Båda är födda i Sverige
    - En av dem är född i Sverige
    - Ingen av dem är född i Sverige
    - Vet ej / Vill ej svara
4. **Kön**
    - Tjej
    - Kille
    - Icke binär
    - Vill ej svara
5. **Jag tycker att programmering är intressant**
    - 1. Håller inte med
    - 2.

- 3.
- 4.
- 5. Håller med helt

6. **Jag tycker att programmering är användbart**
    - 1. Håller inte med
    - 2.
    - 3.
    - 4.
    - 5. Håller med helt

F

Sammanställning av formulärsvar vid utvärdering 1

Efter att eleverna deltagit i intervjun fick de även svara på en enkät för att tillhandahålla bakgrundsinformation. Här presenteras en sammanställning av svaren från den enkäten.

Antal: Sex personer.

Könsfördelning: Tre killar och tre tjejer svarade på formuläret.

Föräldrar med högskoleutbildning: Fem av de svarande har minst en förälder med en högskoleutbildning. En svarar att hen inte vet.

Föräldrar födda i Sverige: Fyra av eleverna svarar att båda deras föräldrar är födda i Sverige och två av dem svarar att ingen av deras föräldrar är födda i Sverige.

Jag tycker att programmering är intressant:[1] Fyra svarande valde 4 och två valde 3. Medelvärdet för killar var 4 medans medelvärdet för tjejer var $\approx 3{,}3$

Jag tycker att programmering är användbart:[1] Fyra svarande valde 5 och två svarande valde 4. Medelvärdet för både killar och tjejer var $\approx 4{,}7$.

[1] På den här frågan fick den svarande ett påstående och en skala med fem steg där 1 var ”Håller inte med” och 5 var ”Håller med helt”.

G

Nivåer i spelet vid utvärdering 2

Nivåerna vid utvärdering 2 är konstruerade för att lära ut programmeringskonceptet *if-sats*. Nivåerna gör detta genom att stegvis introducera spelaren för mer avancerade problem som kan lösas med hjälp av en *if-sats*.

Under de första nivåerna får spelaren mycket hjälp med att lösa problemen av spelets berättarkaraktär men under senare delar av spelet behöver hen lösa samma problem igen fast utan hjälp från berättarkaraktären. I de sista nivåerna introduceras nya spelelement som skulle kunna användas för att skapa mer komplexa nivåer i framtiden.

Nivå 0

Första nivån kräver ingen programmering utav användaren. Den ämnar att presentera spelet genom dialog och genom att spelaren får utforska gränssnittet. För att klar nivån och gå vidare till nästa behöver användaren trycka på starta.

Nivå 1

På andra nivån finns ett hinder i form av en sten som karaktären behöver ta sig runt. Den kod som behövs för att ta sig runt stenen finns redan i editorn men spelaren behöver modifiera nyckelordet *NORTH* till *WEST* eller *EAST* för att karaktären skall undvika stenen. Här behöver alltså användaren förstå vilken del av koden som körs när karaktären är framför stenen.

Nivå 2

På den här nivån har stenen bytts ut mot en stock. Användaren behöver ändra villkoret som kollar ifall karaktären är framför en sten så att den istället kollar om karaktären är framför en stock. Detta gör användaren genom att byta ut den boolska variabeln *isInFrontOfRock* mot en annan fördefinerad variabel som heter *isInFrontOfLog*.

Nivå 3

På fjärde nivån får spelaren för första gången kontrollera två karaktärer. Båda karaktärerna kontrolleras av samma kod men har olika hinder framför sig. Användaren behöver alltså kolla båda variablerna *isInFrontOfRock* och *isInFrontOfLog* för att avgöra om karaktären skall svänga. På den här nivån introduceras också nyckelordet *or*

av berättar karaktären så användaren kan använda det för att lösa uppgiften genom att byta ut villkoret som i förra uppgiften blev *isInFrontOfLog* mot *isInFrontOfLog or isInFrontOfRock*

Nivå 4

På denna nivå är hinderna utlagda så att karaktären måste gå åt olika håll beroende på vilken typ av hinder som är framför. Användaren behöver alltså först komma på lösningen på problemet genom att studera hindernas position och sen implementera den. För att hjälpa med implementationen introducerar berättarkaraktären Python-nyckelordet *elif*.

Nivå 5

På nivå fem är det två karaktärer som skall ta sig igenom en liknande konstruktion av hinder som den i nivå 4. På den här nivån behöver användaren inte skriva någon ny kod då koden från föregående nivå borde fungera för även detta problem. Nivån finns för att visa spelaren att den kod de skriver kan lösa flera liknande problem.

Nivå 6

När användaren kommer till den här nivån har hen fått en ny karaktär och all kod från tidigare är försvunnen. Användaren behöver på den här nivån rekonstruera lösningen från nivå 1 på egen hand.

Nivå 7

På den här nivån behöver användare rekonstruera lösningen från nivå 3. Denna gång utan den hjälp som gavs av berättarkaraktären i nivå 3.

Nivå 8

Nivå åtta bygger på samma problem som i nivå 4. Men användaren behöver den här gången konstruera *elif* konstruktionen utan hjälp från berättarkaraktären.

Nivå 9

Den här nivån introducerar den ena karaktärens förmåga att flyga. När karaktären flyger behöver den inte gå runt stenar och stockar. Användaren behöver bara få karaktären att flyga framåt genom att återskapa lösningen från nivå 0.

Nivå 10

På nivå 10 får spelaren för första gången kontrollera två olika typer av karaktärer samtidigt. Kontrollerna för att växla mellan karaktärerna introduceras av berättar-karaktären. På den här nivån är det alltså olika karaktärer och olika kod som styr de olika typerna av karaktärer.

Problemet som skall lösas är samma som på nivå 5. Dock så behöver inte användaren skriva någon ny kod då den första karaktären redan har rätt kod för att lösa problemet och den andra karaktären kan flyga.

Nivå 11

På sista nivå introduceras torn som ett nytt hinder. Den andra karaktären kan inte flyga över tornen så den behöver nu kod för att undvika torn. Den första karaktärens kod behöver också modifierats så att den undviker torn.

H

Frågor för intervju vid utvärdering 2

Följande frågor användes som grund vid intervjuerna för utvärdering 2. Då det var semi-strukturella intervjuer kanske inte alla frågor ställdes vid varje intervju och ytterligare frågor kan ha ställts.

- Har ni programmerat något i skolan innan?
- Har ni programmerat något på fritiden?
- Tycker ni programmering är intressant?
  - På vilket sätt är det intressant/inte intressant?
- Vad tänker ni att man kan göra med programmering?
- På vilket sätt är programmering användbart för er, om det är det?
- Tror ni att ni kommer använda er av programmering i framtiden?
  - Varför/varför inte kommer ni använda er av programmering i framtiden?
- Vad skulle ni göra för att fortsätta lära er programmering?
- Hur skulle ni beskriva programmet ni testade?
  - Vad är ett spel?
  - På vilka sätt är detta som ett spel, och på vilka sätt är det annorlunda?
- Vad i spelet var roligt?
  - Varför?
- Hade ni velat spela vidare?
- Vad var tråkigt?
  - Varför?
- Fanns det ett tydligt mål?

  - Vad var detta mål?
- Hur kände ni er när du spelade?
  - Kände ni er som en programmerare?
  - Vem är (en) programmerare?
- Påverkade det ni gjorde spelet?
  - Hur såg ni det i spelet?
  - Fick ni veta när ni gjorde något bra, eller mindre bra?
- Var ni med och skapade i spelet?
  - Vad var ni med och skapade?
- Kunde ni testa er fram?
  - Var det okej att göra fel?
  - Vad hände om man gjorde fel?
  - Kunde ni testa saker utan att helt veta vad som skulle hända?
- Fanns det flera sätt att lösa problemen?
  - Vilka olika sätt fanns?
  - Kunde ni testa dessa olika sätt?
- Kände ni att ni hade kontroll över spelet?
  - På vilket sätt?
- Var nivåerna lagom svåra?
  - Kändes det utmanade?
  - Kändes det görbart (att det gick att klara av)?
  - Blev nivåerna svårare i en lagom takt (gick det för fort eller långsamt)?
  - Fick ni använda er av det ni lärt er i tidigare nivåer?
- Fick ni den hjälp ni behövde?
  - Kom den när ni ville ha den?
  - Gav ankan er tillräckligt för att du skulle kunna börja med nivåerna?

  - Var ledtrådarna hjälpsamma?
  - Fick ni för mycket hjälp?
- Hjälpte spelet er att förstå hur en if-sats fungerar?
  - Hjälpte det att ni kunde se hur figurerna rörde sig?
- Hur kändes det att styra karaktärerna?
  - Gjorde dem det du ville?
  - Hur gjorde ni för att få figurerna att röra sig?
- Hur kändes det att skriva instruktioner?
  - Är detta programmering?
    * Varför?
  - Hur skiljer sig den här programmeringen från den ni gjort tidigare?
    * Vad är roligast?
    * Vad lär man sig bäst på?
- Vad var svårt att förstå?
- Vad tyckte ni om att få testa på att programmera i spelet?
- Vad mer skulle du vilja kunna göra i spelet?
- Vad mer skulle du vilja lära dig i spelet?

I

Formulärfrågor vid utvärdering 2

Frågor efter testning av CodeFront

Ditt namn används för att koppla samman dina svar på intervjun med de i formuläret. Du kommer att vara helt anonym (ditt namn och andra identifierande uppgifter kommer att tas bort) i rapporten.

1. **Namn**
2. **Har dina föräldrar en högskoleutbildning?**
    - Båda har en högskoleutbildning
    - En av dem har en högskoleutbildning
    - Ingen av dem har en högskoleutbildning
    - Vet ej / Vill ej svara
3. **Är dina föräldrar födda i Sverige?**
    - Båda är födda i Sverige
    - En av dem är född i Sverige
    - Ingen av dem är född i Sverige
    - Vet ej / Vill ej svara
4. **Kön**
    - Tjej
    - Kille
    - Ickebinär
    - Vill ej svara
5. **Jag är intresserad av spel**
    - 1. Håller inte med
    - 2.

- 3.
- 4.
- 5. Håller med helt

6. **Vad för spel spelar du? Du kan ge exempel på spel (ex. Fortnite, Minecraft, osv.) eller spelgenre (ex. RTS, mobilspel, FPS, osv.)**
7. **Jag är intresserad av programmering**
    - 1. Håller inte med
    - 2.
    - 3.
    - 4.
    - 5. Håller med helt
8. **Jag tycker att programmering är användbart**
    - 1. Håller inte med
    - 2.
    - 3.
    - 4.
    - 5. Håller med helt

J

Sammanställning av formulärsvar vid utvärdering 2

Efter att eleverna deltagit i intervjun fick de även svara på en enkät för att tillhandahålla bakgrundsinformation. Här presenteras en sammaställning av svaren från den enkäten.

Antal: 19 deltog vid utvärderingen men en grupp svarade inte på enkäten så det finns endast 15 svarande.

Könsfördelning: Sju killar och åtta tjejer svarade på formuläret.

Föräldrar med högskoleutbildning: Elva av de svarande har minst en förälder med en högskoleutbildning. Fyra svarar att de inte vet eller inte vill svara.

Föräldrar födda i Sverige: Nio av eleverna svarar att båda deras föräldrar är födda i Sverige, en att hen har en förälder som är född i Sverige och fyra av dem svarar att ingen av deras föräldrar är födda i Sverige. En vet inte eller vill inte svara på frågan.

Jag är intresserad av spel:[1]

Alla, medelvärde ≈ 4,1					
Svar	1	2	3	4	5
Antal	0	1	1	8	5

Killar, medelvärde ≈ 4,2					
Svar	1	2	3	4	5
Antal	0	0	0	5	2

Tjejer, medelvärde 4					
Svar	1	2	3	4	5
Antal	0	1	1	3	3

Jag tycker att programmering är intressant:[1]

Alla, medelvärde ≈ 2,9					
Svar	1	2	3	4	5
Antal	1	4	6	3	1

Killar, medelvärde ≈ 3,6					
Svar	1	2	3	4	5
Antal	0	1	2	3	1

Tjejer, medelvärde ≈ 2,4					
Svar	1	2	3	4	5
Antal	1	3	4	0	0

Jag tycker att programmering är användbart:[1]

Alla, medelvärde ≈ 4,3					
Svar	1	2	3	4	5
Antal	0	1	2	4	8

Killar, medelvärde ≈ 4,4					
Svar	1	2	3	4	5
Antal	0	0	1	2	4

Tjejer, medelvärde ≈ 4,1					
Svar	1	2	3	4	5
Antal	0	1	1	2	4

[1] På den här frågan fick den svarande ett påstående och en skala med fem steg där 1 var ”Håller inte med” och 5 var ”Håller med helt”.

K

Teknisk beskrivning av CodeFront

Spelet som har utvecklats är gjort för att köras i webbläsare och är därför skrivet i JavaScript. Ramverket Vue användes för att tillhandahålla grundläggande funktionalitet och strukturera upp applikationen. Vue användes också för att hantera informationsflödet så att vi lättare kunde koppla ihop användarens kod med vad som händer i spelet.

Det har under utvecklingen lagts ett stort fokus på att spelet skall vara lätt att utöka. Därför kan nya levlar, karaktärer och hinder läggas till genom att skapa filer som beskriver det aktuella elementet. Resten av kodbasen behöver inte anpassas efter nya element ifall de inte vill utnyttja funktioner som ännu inte finns i spelet.

Länk till källkoden för projektet: https://github.com/codefrontgame/codeFront

K.1 Hantering av användarkod

En utav de största utmaningarna för projektet var att på ett säkert sätt kunna köra användarens kod och använda resultatet för att kontrollera spelet. För att åstakomma detta användes biblioteket Esper. Esper exekverar, bland annat JavaScript och Python, i en virtuell miljö och returnerar sedan resultatet.

Biblioteket Ace, en textbehandlare (editor), används för att ge visuellt stöd och automatisk rättning när användarna skriver sin kod. Ace tillhandahåller en editor lik många moderna IDE's med syntaktisk överstrykning (syntax highlighting), hantering av indentering och förgranskning av syntax (linting).

K.2 Grafik

Eftersom karaktärerna skall röra sig framåt i en simulerad 3D värld behöver de bli mindre vid slutet av banan. För att lätt kunna skala alla element skapades de med vektorgrafik. För att hantera utmålningen i spelet valde vi att använda oss utav biblioteket p5js. p5js tillhandahåller hlälpmetoder för att måla ut bilder och geometrisk former på en *HTML-canvas*. Tyvärr stödjer inte biblioteket vektorgrafik och därav skalar inte elementen på spelbrädet så bra och blir lätt suddiga.