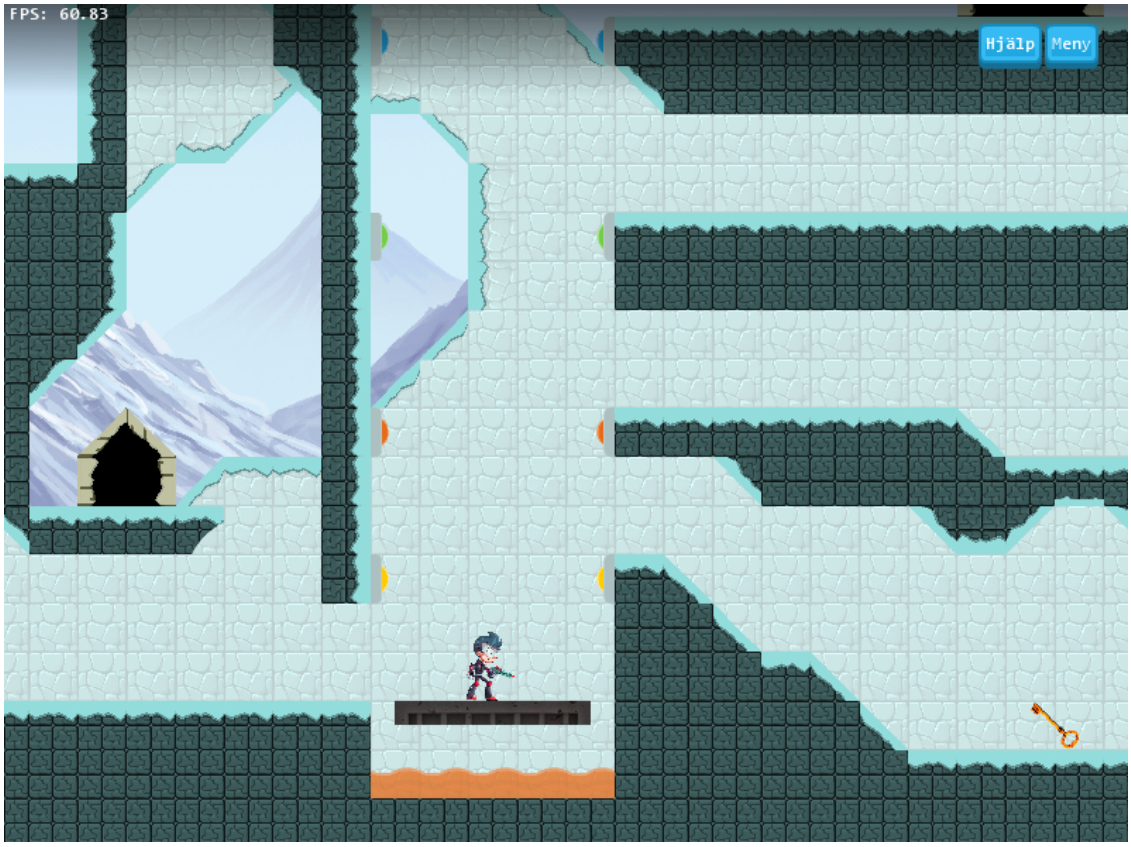
CHALMERS



JACE

- ett seriöst spel för programmeringsstudenter, inspirerat av spelifiering

Kandidatarbete inom Data- och Informationsteknik

Anton Annenkov
Andreas Hagesjö
Oskar Kärrman
Rafael Mohlin
Theodor Åstrand

CHALMERS TEKNISKA HÖGSKOLA
Institutionen för Data- och Informationsteknik
Göteborg, Sverige, Juni 2015

JACE
Ett seriöst spel för programmeringsstudenter, inspirerat av spelifiering

Anton Annenkov
Andreas Hagesjö
Oskar Kärrman
Rafael Mohlin
Theodor Åstrand

Examiner: Jan Skansholm

Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2015

Förord

Rapporten genomfördes i ett kandidatarbete under våren 2015 vid Instutionen för Data- och Informationsteknik, Chalmers tekniska högskola. Författarna till rapporten vill uttrycka ett stort tack till Erland Holmström för den tid han ägnade åt att handleda gruppen. Ett stort tack riktas även till Ludwig Lindberg, som deltog i projektet men inte hade möjlighet att fortsätta sitt deltagande i projektets slutskede. Vi vill även tacka Eric Leskinen, student vid Göteborgs Universitet, för den tid han lade ner på att ge återkoppling på rapportens innehåll och utformning. Slutligen vill vi tacka Kristian Knudsen, för formgivning av affisch till utställningen.

Abstract

This paper describes the development and evaluation of a serious game, inspired by gamification, which has the purpose of increasing students' incentive to study object-oriented programming. This particular subject is considered to be difficult to grasp by many students and recent studies implies that a need for enhancing study incentive, and learning efficacy to study object-oriented programming, exists. This background constitutes the basis of our development of a game containing engaging game mechanics such as feedback, progress and achievements. These game mechanics constitutes the core of gamification. A prestudy was conducted in order to finalize a theoretical framework, which was analyzed in a following phase. This analysis was followed by the decision that our game should be a video game, which could be used as an aid while learning and introducing object-oriented programming. In addition to these game mechanics, the problem based learning method inspired the content of the game. This resulted in a number of game-based problems implemented into the game, as well as a quiz related to these problems. In addition to these implementations, an achievement system was added. A survey was conducted on 24 persons to evaluate whether the goal of the game to motivate and educate had been fulfilled. The result of the survey shows that 75 percent experienced increased incentive to study object-oriented programming. Furthermore, 46 percent responded that they would consider using this game as an aid while studying object-oriented programming. Game mechanics such as feedback and achievements, in combination with difficult problems related to programming, were found to be effective methods to increase the students' incentive to study object-oriented programming. However, further implementation of game content and additional surveys are considered necessary in order to come to a conclusion regarding the influence of the game, considering study incentive and learnability among students.

Sammanfattning

Denna rapport syftar till att beskriva utvecklingen och utvärderingen av ett seriöst spel, vilket är inspirerat av spelifiering, som har syftet att öka motivationen hos studenter inom objektorienterad programmering. Objektorienterad programmering anses vara problematiskt för högskolestudenter och det har visats att ett behov existerar att förbättra inlärning och motivation hos studenterna. Detta har föranlett vår utveckling av ett datorspel med fokus på motivationshöjande spelmekanik såsom återkoppling, framsteg och belöningssystem, vilka är centrala för spelifiering. En förstudie genomfördes för att fastställa en teoretisk referensram. Denna referensram analyserades och det beslutades att ett motivationshöjande datorspel skulle utvecklas, där spelet kan användas som ett hjälpmedel vid studier inom objektorienterad programmering på en introduktionsnivå. Utöver motivationshöjande spelmekanik inspirerades spelet även av pedagogikmetoden problembaserat lärande. Detta resulterade i programmeringsrelaterade problem och frågor i form av en frågesport i spelet. Utöver detta implementerades ett belöningssystem i spelet. En enkätundersökning genomfördes på 24 testpersoner för att utvärdera huruvida spelets mål att motivera och undervisa hade uppnåtts. Resultatet av enkätundersökningen visar att 75 procent av testpersonerna upplevde ökad motivation till studier inom objektorienterad programmering, och 46 procent svarade att de kunde tänka sig använda spelet som ett hjälpmedel vid egenstudier av objektorienterad programmering. Återkoppling och belöningssystem i kombination med krävande programmeringsrelaterade problem visades vara effektiva metoder för att höja studenternas motivation till studier inom objektorienterad programmering. Implementation av fler kunskapsmoment i spelet och utförligare tester anses dock vara nödvändigt för att dra generella slutsatser om spelets inverkan på motivation och inlärningsförmåga hos studenterna.

Innehållsförteckning

Ordlista

API	*Application Programming Interface*, ett gränssnitt för hur en viss programvara ska kommunicera mot en annan programvara.
Enspelarläge	Anger att användaren spelar ensam på nivåerna i spelet.
FPS	*Frames per second*, ett mått på antalet bilder per sekund.
Färgdjup	Ett mått på antalet bitar som används för att representera färger.
JSON	*JavaScript Object Notation*, ett textbaserat datautbytesformat.
Kategori	Ett samlingsnamn för delar av spelet som innehåller viktig funktionalitet.
Ljudläge	Anger om ljud är aktiverat eller inte.
OpenGL	*Open Graphics Library*, ett plattformsoberoende API med stöd för att skapa applikationer med två- eller tredimensionell grafik.
Speldata	Utgörs av data som sparas i JSON filer.
Sprint	En arbetsperiod i detta projekt som varar i sju dagar.
Uppdateringsfrekvens	Ett mått på antalet bilder som kan ritas upp på en skärm per sekund.
Visningsläge	Anger om fullskärmsläge eller fönsterläge används.

1. Inledning

Spel av olika slag har existerat under en lång tid i mänsklighetens historia. Schack, hasardspel, kortspel och mer traditionella sällskapsspel har med tiden utvecklats till datorspel och TV-spel. Den senare sorten har blivit allt annat än ett överflödigt element i dagens samhälle och kultur, utan har snarare blivit ett givet inslag i var mans hem (Zackariasson & Wilson, 2010).

Spel har traditionellt sett existerat i fysisk form – såsom brädspel och kortspel – under en mycket lång period, medan TV- och datorspel har ett kort förflutet i jämförelse. Trots detta är denna bransch en snabbväxande sådan, inte minst spel till mobila enheter och onlinespel (De Prato et al., 2014). En populär genre inom TV- och datorspel, som har existerat sedan dess uppkomst, är pusselspel: Myst, Tetris och Lemmings är några av många klassiker inom denna genre. På senare tid har även Portal, Fez och The Swapper anslutit sig till skaran av klassiska pusselspel. Dessa spel inriktar sig mot underhållning, men i spelhistoriens tidiga skede hade spel ett mer bildande syfte – särskilt inom militära sammanhang där olika strategiska spel existerade. Dessa undervisande krigsspel i form av brädspel användes så sent som under andra världskriget, men de lever kvar i dagsläget i form av krigssimulationsspel för datorer (Smith, 2010).

Ett syfte med de tidigare brädspelen var alltså av en mer bildande karaktär snarare än en underhållande. Undervisande spel börjar återigen bli allt mer vanligt i dagsläget, i form av att alldagliga sysslor omvandlas till ett spel eller en lek för att öka motivationen och inlärningen hos användaren (Buckley & Doyle, in press). Detta fenomen, vanligen kallat för spelifiering, har till och med kallats för ”the next big thing in marketing” (Luminea, 2013, s.13). Ett antal närbesläktade begrepp, såsom seriösa spel och spelbaserat lärande, har även ökat i popularitet inom såväl industriella som akademiska sammanhang (Liu et al., 2014).

Införandet av element som ökar inlärningsförmåga och motivation hos studenter är något som eftertraktas i dagsläget av många lärare. Inom ämnet programmering finns ett tydligt behov av att bemöta de problem som finns i dagsläget, framför allt inom högskoleväsendet (Mow, 2008). Bland förstaårsstudenter tycks det dessutom vara en vanlig åsikt att programmering är något som är problematiskt och svårt att lära sig (Tan et al., 2009). I programmeringskurser finns därtill ett behov av interaktiva lärandemiljöer (ibid). Mer specifikt finns det problem rörande objektorienterad programmering där studenterna framför allt har svårt att förstå koncept som konstruktorer, klasser och hjälpfunktioner – begrepp som är centrala inom objektorienterad programmering (Biju, 2013). Steget från procedurell programmering till objektorienterad programmering tycks även vara problematiskt, något som tyder på att den sistnämnda programmeringsparadigmen i synnerhet kräver mer fokus på ökad motivation (ibid). Uppfattningen om att programmering generellt sett är något av ett berg av kunskap att bemästra är en av många faktorer som resulterar i att eleverna misslyckas med att bli godkända på skrivningarna, eller väljer att hoppa av sin utbildning (Tan et al., 2009). Inlärning av programmering är något som är problematiskt för studenter, i synnerhet i introduktionskurser (Jerinic, 2014). Det finns därmed ett tydligt behov av att förändra synen på – samt lärandet av – programmering för elever inom högskolor och universitet.

1.1. Problemdiskussion

Programmering är ett nytt ämne i jämförelse med traditionella ingenjörsämnen på högskolenivå. Som tidigare har nämnts i detta kapitel finns det tydliga behov av att förbättra såväl lärandet som motivationen hos studenter som studerar programmering. I dessa sammanhang är spelifiering ett begrepp som används för att öka motivationen hos en eller flera användare med hjälp av spelelement och spelmekanik (Arnold, 2014). Spelifiering definieras vanligen som användandet av spelelement för att uppnå ett önskvärt beteende hos användaren och har bland annat använts i motivationshöjande syften (Deterding et al., 2011). Ytterligare ett begrepp för att förbättra inlärning hos studenter är seriösa spel, vilket definieras som spel vars syfte är skilt från underhållning (Deterding et al., 2011), och har använts inom bland annat datavetenskapliga ämnen (Hakulinen, 2011). En förekommande uppfattning är att seriösa spel i undervisande sammanhang är en bättre metod än traditionell undervisning, något som stundtals har varit svårt att styrka (Wouters et al., 2013). Det finns dock bevis för att kunskap och lärande kan genomföras bättre i seriösa spel än med hjälp av traditionella pedagogiska verktyg (ibid). Gemensamt för många studier rörande seriösa spel är att positiva effekter har observerats, även om mer forskning krävs för att bekräfta dessa observationer (Connolly et al., 2012).

Till skillnad från seriösa spel är spelifiering ett begrepp som blev vedertaget inom akademiska sammanhang kring 2010 (Deterding et al., 2011). Det finns i dagsläget lite forskning kring såväl begreppets definition och kritik kring begreppet, trots begreppets ökade användning inom industrin. I synnerhet saknas forskning kring skillnaden mellan spelifiering och seriösa spel, vilket motiverar ett akademiskt granskande och jämförande av dessa två begrepp. Nyttjandet av dessa begrepp inom utbildningssammanhang hos högskolestudenter är i dagsläget en relativt ny och outforskad nisch – inte minst användandet av begreppen för att skapa ett bildande och motivationsökande datorspel. Detta föranleder frågan om ett sådant spel kan skapas och om syftet med ökad motivation för ämnet programmering kan uppnås med hjälp av detta spel.

1.2. Syfte

Syftet med detta projekt är att skapa och utvärdera ett seriöst spel med inspiration från begreppet spelifiering. Spelet ska vara ett datorspel, vars syfte är att förbättra motivationen hos studenter som har liten eller ingen tidigare erfarenhet av programmering. Därtill ska spelet verka som ett hjälpmedel vid studier inom objektorienterad programmering.

1.3. Avgränsningar

Spelet som ska utvecklas i detta projekt ska vara ett datorspel. Med anledning av detta ska spelet enbart stödja persondatorer (Windows, Mac och Linux). Därmed ska stöd för mobila enheter och webbplatser utelämnas, vilka anses vara utanför projektets ramar. Innehållsmässigt ska spelet begränsas till att innehålla ett *enspelarläge* med anledning av att fokusera på ökning av motivation på individuell nivå. Därtill ska spelet inte innefatta samtliga kunskapsmoment inom objektorienterad programmering, med anledning av att spelet ska vara syftat till nybörjare inom programmering. Tekniska begränsningar i projektet innefattas av att spelet ska skapas med tvådimensionell grafik istället för tredimensionell grafik. Detta ska göras med anledning av att den sistnämnda tekniken oftast är komplex och stundtals medför svårigheter vid implementation (Johnson, 2008). Vi anser att tvådimensionell grafik är tillräcklig för att uppnå projektens mål, med anledning av att spelets innehåll är mer relevant än dess utseende. I enlighet med projektets syfte ska komplexa programmeringstekniker utelämnas i spelet, med anledning av att målgruppen har inga eller få kunskaper inom programmering sedan tidigare.

2. Metod

Projektet delades in i fyra faser där somliga löpte parallellt med varandra under projektets genomförande: planeringsfas, förstudiefas, implementationsfas och utvärderingsfas. Planeringsfasen utgjordes av en tidig planering av arbetet och föranledde förstudiefasen. Vi ansåg att genomförandet av förstudiefasen krävde erkända vetenskapliga metoder för att genomföras och låg till grund för utvecklingen av spelet i implementationsfasen. Utvecklingen av spelet genomfördes i implementationsfasen, vilket krävde vedertagna metoder, ramverk och verktyg för att underlätta utvecklingen av spelet. Slutligen utvärderades spelet i utvärderingsfasen, där bland annat en enkätundersökning genomfördes. Metoderna som användes i projektet beskrivs och motiveras nedan.

2.1. Vetenskaplig metod och datainsamling

Kvantitiva ansatser användes huvudsakligen i projektets genomförande. Därtill präglas projektet genomgående av begreppet *operationalisering*, något som innebär framtagning av relevanta begrepp ur tillgänglig teori (Eliasson, 2013). Operationalisering utnyttjades i projektet genom att formulera en teoretisk referensram, där begrepp såsom spelifiering, seriösa spel och problembaserat lärande ingår. Därefter användes dessa begrepp för att bilda frågor vilka ingick i enkätundersökningen. Dessa begrepp ansågs vara relevanta att fokusera på med anledning av projektets syfte. Begreppen nyttjades även vid utformningen av spelet.

De slutsatser som drogs i kapitel 4 bygger på vår tolkning och analys av insamlat material i den teoretiska referensramen. Utvärderingen av spelet genomfördes med hjälp av enkätundersökningar, vilken är en kvantitativ metod för att analysera erhållen data (ibid), och våra analyser. Vid insamling av data i enkätundersökningen formulerades frågor vilka var relevanta för att undersöka spelets inverkan på den valda målgruppen. Enkätundersökningen kompletterades även med kvalitativa frågor för att besvara delar av rapportens syfte. Detta hänvisas ofta som *triangulering*, vilket innebär en kombination av kvantitativa och kvalitativa ansatser (ibid). Denna enkätundersökning utgjordes av ett frågeformulär, vilket kan ses i Bilaga A.

Till följd av brist på en enhetlig definition av spelifiering (Hamari et al., 2014) samt en vedertagen skillnad mellan spelifiering och närbesläktade begrepp (Kapp, 2012) ansågs bildandet av ett teoretiskt ramverk som en väsentlig del i rapporten. Litteratur användes huvudsakligen i två syften: inledningsvis för att bilda en uppfattning om problemet, dess förutsättningar och obekanta begrepp, för att därefter analysera dessa begrepp. Efter att den inledande skrivprocessen genomfördes, vilken innefattar bakgrund och formulering av syfte, konstaterades det att en begreppsutredning var nödvändig i den teoretiska referensramen. Det insamlade materialet utgjordes huvudsakligen av sekundärkällor i form av vetenskapliga publikationer och artiklar rörande spelifiering, seriösa spel och pedagogik inom programmering. Insamling av material till den teoretiska referensramen utfördes med hjälp av kända bibliografiska databaser och sökmotorer såsom Scopus, Proquest och ACM Digital Library. Vetenskapliga publikationer användes till stor del, i synnerhet föredrogs metastudier, något som underlättade ställningstagandet till källornas pålitlighet.

Urval vid enkätundersökning
Urvalsramen utgjordes av blivande, nuvarande och tidigare studenter vid Chalmers tekniska högskola och Göteborgs Universitet. Dessa lärosäten valdes med anledning av att enkätundersökningens genomförande krävde att vi närvarade för att svara på frågor. Med anledning av projektets tidsbegränsningar ansågs det inte vara rimligt att genomföra en djupare undersökning av studenter från flera orter och lärosäten. Huvudsakligen valdes individer vilka hade ingen till måttlig erfarenhet av programmering, med anledning av att spelet är inriktat mot nybörjare inom nämnda ämne.

2.2. Agil mjukvaruutveckling och Scrum

I dagsläget är agil mjukvaruutveckling den dominerande typen av arbetsmetod, något som är ett resultat av ett ökat behov av varaktig återkoppling och bearbetning av produkten (Mukker et al., 2014). Med anledning av de agila arbetsmetodernas vedertagna användning ansåg vi att denna typ av metod var ett lämpligt alternativ vid utveckling av spelet. Agil mjukvaruutveckling är en utvecklingsmetod där produkten skapas iterativt (Dybå & Dingsøyr, 2008). Utvecklingen sker därmed i små steg, där varje steg lägger till nya funktioner eller förbättrar tidigare arbete (ibid). Inom agil mjukvaruutveckling existerar ingen detaljerad planering för projektet, snarare utförs detaljplanering parallellt med projektets genomförande vilket ofta benämns som en ickelinjär arbetsprocess (Williams & Cockburn, 2003). I kontrast till en definierad arbetsprocess, vilket kan liknas vid löpandebandprincipen, sker denna ickelinjära process inkrementellt och tillåter ändringar under arbetets gång (ibid). Vi ansåg att den iterativa arbetsprocessen var önskvärd i projektet, med anledning av att stegvis kunna utvärdera spelet. Detta var något som inte kunde genomföras med hjälp av en definierad arbetsprocess och med anledning av detta valdes istället den iterativa arbetsprocessen.

De agila arbetsmetoderna har därefter resulterat i mer specifika arbetsmetoder. En av dessa är arbetsmetoden Scrum vilken betonar det inkrementella arbetet i form av *sprints* kombinerat med kontinuerlig återkoppling på arbetet (Schwaber & Beedle, 2002). Scrum innefattar huvudsakligen tre arbetsroller: *scrum master*, produktägare och produktutvecklare (ibid). Scrum master ansvarar för administrativa uppgifter medan produktägaren ansvarar för vilka funktioner som ska vara med i produkten, vilket bildar produktens *backlog* (ibid). Sprints består av den veckoliga eller månatliga planeringen för produktens utveckling och innehåller därmed funktioner ur produktens backlog (Paasivaara et al., 2009). Scrum är därmed användbart i situationer då en långsiktig planering inte kan fastställas, exempelvis när ett tydligt slutmål för produkten saknas (Schwaber & Beedle, 2002).

Vi valde därmed att nyttja arbetsmetoden Scrum. Detta beslut föranleddes av att ett tydligt slutmål saknades i planeringsfasen, samt att vi hade tidigare erfarenhet av denna arbetsmetod. Utvecklingen av produkten utfördes genom att tilldela arbetsroller, såsom scrum master och produktägare. Arbetsrollerna växlades halvvägs i projektet för att involvera projektets medlemmar inom flera delar i utvecklingen av spelet. Veckoliga möten hölls under projektet, vilka användes för att planera nästkommande vecka i form av sprints. Därtill genomfördes dagliga möten, eftersom

vi ansåg att det var nödvändigt att uppdatera samtliga av projektets medlemmar med projektets fortskridande. Vidare nyttjades de slutsatser vilka härleds i kapitel 4 för att upprätta en kravspecifikation, som sedan användes för att skapa projektets backlog.

2.3. Utvecklingsverktyg

Vid utvecklingen av spelet fanns ett flertal delmoment, såsom animation av huvudkaraktären och skapande av nivåer, vilka ansågs vara tidskrävande att genomföra utan hjälp av utvecklingsverktyg. Som ett kompletterande stöd valdes följande verktyg: Git, Waffle, Spine och Tiled, i syfte att underlätta utvecklingen av spelet. Verktygen beskrivs och motiveras nedan:

- Mjukvaruutveckling i en grupp bestående av flera utvecklare underlättas av koordination och möjligheter att arbeta med samma kod parallellt. Git är ett versionshanteringssystem som tillåter att flera utvecklare arbetar samtidigt på ett projekt. Detta möjliggörs genom att Git sparar tidigare versioner av filer, historik, upphovsman och innehåll (Loeliger, 2009). För att göra användandet av Git mer anpassligt användes tjänsten GitHub, med anledning av att GitHub tillhandahåller en decentraliserad versionshantering och säkerhetskopiering av data (GitHub, 2015).

- För att underlätta hantering av projektets sprints användes verktyget Waffle, vilket är ett verktyg för projektledning som är integrerat med GitHub (Waffle, 2015). Waffle valdes, till skillnad från liknande verktyg såsom Trello och Wrike, huvudsakligen med anledning av dess integrering med GitHub.

- Med anledning av att en trovärdig spelkänsla var en viktig del vid utvecklingen av spelet beslutades det att huvudkaraktären i spelet skulle animeras. Detta gjordes i samverkan med verktyget Spine, som är ett animationsverktyg med stöd för LibGDX. Genom användandet av Spine skapades och hanterades animationer i spelet, där animationerna utgjordes av en skelettstruktur innehållandes bilder (Esoteric Software, 2015).

- Vid utveckling av nivåer användes verktyget Tiled, vilket är en *baneditor* som används för att placera ut objekt och bilder vid skapande av nivåer (Cowan & Kapralos, 2011). Tiled valdes med anledning av att den valda spelmotorn LibGDX stödjer denna baneditor, samt att dokumentation tillhandahålls för hur nivåer kan skapas och implementeras (Badlogic Games, 2015).

3. Teoretisk referensram

Kapitlet syftar till att redogöra för den teori som ligger till grund för projektet och dess genomförande. Inledningsvis presenteras en genomgång av spelifiering och seriösa spel. Därefter redogörs för teori kring svårigheter inom inlärning av programmering, vilket berör objektorienterad programmering huvudsakligen. Slutligen berörs den teori som finns kring spelelement och spelmekanik, vilka är nödvändiga moment för att skapa spelet.

3.1. Spelifiering

Spelifiering, eller *gamification* på engelska, är ett relativt nytt begrepp vilket myntades år 2002 av Nick Pelling – en brittisk programmerare som skapade spelifieringstjänster i sitt yrke (Pelling, 2011). Begreppet blev dock inte erkänt och vedertaget förrän år 2010; först efteråt granskades begreppet i form av vidare forskning och begreppsutredning (Deterding et al., 2011). Detta föranledde de definitioner som finns i dagsläget, vilka skiljer sig åt markant i vissa fall. En vanligt förekommande definition för spelifiering är: ”the use of game design elements in non-game contexts” (Deterding et al., 2011, s. 10). Detta innebär att användandet av spelelement i sammanhang som inte berör spel utgör spelifiering. Värt att notera är dock att denna definition varken utreder begreppet spel eller spelsammanhang. Det finns samtidigt andra definitioner som snarare riktar sig mot syftet med spelifiering: ”Gamification is using game-based mechanics, aesthetics and game thinking to engage people, motivate action, promote learning, and solve problems” (Kapp, 2012, s. 10). Denna definition betonar användandet av spelmekanik och spelestetik i syftet att engagera, motivera och uppmuntra lärande hos användaren. I stark kontrast till den definition som har skapats av Deterding et al. (2011) utesluts inga sammanhang, utan fokuserar snarare på syftet med begreppet.

Spelifiering har nyttjats inom industriella sammanhang, vilket har föranlett ett flertal definitioner från företag vilka specialiserar sig på spelifierade produkter. Ett av dessa företag är Bunchball, som har definierat spelifiering som: ”Gamification applies the mechanics of gaming to non-game activities to change people’s behavior” (Bunchball, 2010). Återigen poängteras nyttjandet av spelmekanik som en viktig faktor inom spelifiering och spelelement anses dessutom vara ett centralt begrepp inom spelifiering. Buckley & Doyle (in press) betonar just spelelement som kärnan inom spelifiering och de sammanfattar de viktigaste spelelementen som: specifika regler, belöningssystem, återkoppling och tävlande element. Dessa moment redgörs för i delkapitel 3.3.

Begreppet spelifiering är i dagsläget relativt omstritt och blandas ofta ihop med begrepp som *seriösa spel* och *spelbaserat lärande*. Enligt Kapp (2012) kan i själva verket det sistnämnda begreppet anses vara en variant av spelifiering. Detta motiveras av att dessa spel har ett syfte som inte är att underhålla, utan snarare att utbilda eller förbättra (ibid). Deterding et al. (2011) är dock noga med att skilja spelifiering från spel, i synnerhet då det senare oftast förknippas med lek. Även i de fall då spel utformas i ett seriösare syfte än att underhålla är det svårt att påstå att detta är spelifiering av ett spel, eftersom detta snarare rör sig om speldesign och inte spelifiering (ibid). Datorspel kan även ses som ett komplett nyttjande av spelelement, medan spelifiering identifierar och använder relevanta spelelement för

att engagera användaren (Landers, 2015).

3.2. Seriösa spel

Begreppet *seriösa spel*, eller *serious games* på engelska, har till skillnad från spelifiering existerat under en relativt lång tid. Likt spelifiering har seriösa spel definierats av såväl forskare som företag, där somliga definitioner har blivit mer vedertagna än andra. En av de mest vedertagna definitioner av seriösa spel lyder: ”The simplest definition of serious games, then, is games that do not have entertainment, enjoyment or fun as their primary purpose” (Chen & Michael, 2005, s.21). Denna definition betonar därmed att seriösa spel är spel vars huvudsyfte inte är att underhålla, vilket även konstateras av Deterding et al. (2011). Ytterligare en definition, vilken används av företaget Serious Games Initiative, för seriösa spel lyder: ”Any meaningful use of computerized game/game industry resources whose chief mission is not entertainment” (Sawyer, 2007, s.12). Här beskrivs ett seriöst spel som ett meningsfullt nyttjande av datorspel eller spelresurser, vars huvudsyfte inte är att underhålla användaren. Det bör dock uppmärksammas att seriösa spel kan upplevas som underhållande, trots att detta inte är huvudsyftet (Arnab et al., 2014). Därtill krävs troligen en noggrann övervägning mellan underhållning och seriöst lärande för att framgångsrikt utveckla ett seriöst spel (ibid).

Spel vars syften är skilda från underhållning har framhävts som en förbättrande komponent i utbildningssammanhang, eftersom spel kan vara en engagerande läromiljö och har möjlighet att ge studenterna återkoppling på erhållna kunskaper (Kiili, 2005). Dessutom anses fullföljandet av läromål som det mest centrala momentet för att studenter ska kunna uppnå goda studieresultat, något som utbildande spel kan uppnå (ibid). Det har även framhävts att datorspel kan ge tydlig återkoppling på studenternas framsteg och misslyckanden, eftersom återkoppling sker visuellt och genomförandet av ett problem eller pussel i spelet belönar studenten omedelbart (Kazimoglu et al., 2012). Det har även konstaterats att datorspel i undervisande eller bildande syfte kan förbättra såväl motivation som inlärningsförmåga, även om det i dagsläget finns ett tydligt behov av mer forskning kring detta (Kim et al., 2009; Connolly et al., 2012). Skapandet av ett seriöst spel kan ses som en uppgift vilken kräver ett tydligt fokus på engagerande spelkänsla, skapande av lärande verktyg samt visualisering av läromålen (Kelly et al., 2007). Det kan även konstateras att utveckling av ett seriöst spel bör involvera samarbete mellan personer med bakgrund från olika discipliner, med anledning av att många aspekter ska täckas i spelet (ibid).

3.3. Spelelement, spelmekanik och speldynamik

Game mechanics och *game design element*, vilket hädanefter hänvisas som spelmekanik och spelelement, har tidigare inkluderats i definitioner för såväl spelifiering som seriösa spel, såsom i definitionen av Deterding et al. (2011). Spelelement innefattar en rad olika begrepp rörande funktionalitet, utformning och metoder vilka utgör ett spel (ibid). De fundamentala spelelementen kan sammanställas i två kategorier: spelmekanik och speldynamik (Bunchball, 2010).

Spelmekanik
Spelmekanik berör grundläggande spelfunktionalitet, vilken nyttjas inom spelifiering för att öka motivation hos användaren (Zichermann & Linder, 2013). Dessutom kan spelmekanik ses som en konkretisering av de regler som finns för spelet, vilket medför att spelmekanik omvandlar spelinstruktioner till spelfunktioner (Koster, 2013). I delkapitel 3.1 nämns spelelement vilka är centrala inom spelifiering. Werbach & Hunter (2012) sammanfattar dessa spelelement som olika sorters spelmekanik: poäng- och emblemsystem, nivåer, topplistor och belöningar.

Poängsystem ger direkt återkoppling till användaren vid framsteg i spelet, med anledning av att poäng delas ut vid genomförandet av en uppgift (Zichermann & Linder, 2013; Domínguez et al., 2013). Enligt Bunchball (2010) kan poängsamlande ses som ett moment där användaren har möjlighet att uppnå ett mål och det konstateras även att poängsystem är ett vanligt förekommande moment inom spelifierade produkter. I nära anslutning till poängsystem existerar *achievement systems*, som hädanefter hänvisas emblemsystem, vilket kan liknas vid utdelning av medaljer eller emblem (Bunchball, 2010). Utdelning av emblem är huvudsakligen ämnat till att väcka positiva känslor hos användaren, eftersom emblem är utformade som direkta belöningar (Domínguez et al., 2013). Emblemsystem kan även ge användaren direkt återkoppling på de prestationer som vederbörande har uppnått (ibid). Därtill kan emblemsystem användas för att åskådliggöra de mål som användaren ska uppnå och uppmuntrar även användaren att utforska efter flera delmål (ibid). Det har även visats att emblemsystem har stor potential att motivera användarna till slutförande av sina uppgifter (Fitz-Walter et al., 2011). Därtill upplevs emblemsystemet som underhållande och uppskattat av användarna (ibid). Samlandet av emblem är även något som kan uppmuntra användare att återvända till spelet (Zichermann & Linder, 2013).

Ett exempel på ett emblemsystem visas i figur 3.1, där användaren har låst upp emblemet ”Den givmilde”. Villkoret för att låsa upp detta emblem går att läsa under emblemets titel, i detta fall: ”Ge 100 vapen till dina lagkamrater”. Emblem som är upplåsta är färglagda, medan emblem som inte är upplåsta är gråmarkerade. Användaren kan därmed följa sina framsteg och se vilka emblem som är möjliga att låsa upp. Emblemsystem har likheter med *nivåsystem*, vilket är ett system där användaren kan följa sina framsteg (ibid). Nivåsystem kan liknas vid en titel, vilket kan jämföras med bältesfärger i kampsporter, vilken erhålls genom användarens deltagande och representerar till skillnad från emblemsystem inte nödvändigtvis en direkt visuell belöning eller återkoppling (Bunchball, 2010).

Figur 3.1: Skärmdump från Counter-Strike: Global Offensive som visar ett emblemsystem (Valve Corporation, 2012).

Leaderboards, vilket hädanefter hänvisas som topplistor, syftar till att införa ett tävlingsmoment inom datorspel och spelifiering (Domínguez et al., 2013). Användarna kan med hjälp av topplistor se hur deras prestationer förhåller sig till andra användare, något som medför en känsla av tävlande (Bunchball, 2010). Utöver detta har topplistor potential att skapa ett önskvärt beteende hos en användare, något som kan nyttjas för att ändra beteenden hos studenter (Singer & Schneider, 2012). Viss tvivel kring denna form av tävlande finns dock, eftersom somliga användare upplever att tävlande och jämförande moment motverkar motivation (Domínguez et al., 2013). Ett exempel på en topplista från spelet Left 4 Dead kan ses i figur 3.2. Användarna rangordnas i denna topplista efter genomförd tid för en nivå, där en längre tid medför en högre placering i topplistan.

Figur 3.2: Ett exempel på en topplista från spelet Left 4 Dead. Gjord av (mrwynd, 2009). CC-BY 2.0.

Utöver dessa typer av spelmekanik finns det även ett flertal andra typer, vilka anses vara relevanta för spelifiering, såsom: handling, tydliga mål, återkoppling, belöningar, framsteg och utmaningar (Hamari et al., 2014). Dessa typer av spelmekanik och dess motiverande effekt har inte studerats i samma omfattning som poängsystem, emblemsystem och topplistor. Spelmekaniken återkoppling har dock granskats i de fall då frågesporter har använts för att ge återkoppling på användarens kunskaper (Cheong et al., 2013). Denna variant av återkoppling har visat sig vara uppskattad av studenter som har nyttjat frågesporter inom utbildningssammanhang (ibid). Återkoppling kan även ses som ett övergripande tema för spelifiering, då många typer av spelmekanik erbjuder just detta (Muntean, 2011).

Speldynamik
Speldynamik och spelmekanik växelverkar med varandra, då spelmekanik påverkar speldynamik (Hunicke et al., 2004). Därmed innehåller spelmekanik de funktioner som utgör spelet, vilka används för att skapa speldynamik (ibid). Speldynamik utgörs således av drivkrafter och behov, vilka föregås av spelmekanik (Bunchball, 2010). Dessa drivkrafter kan därefter nyttjas för att skapa känslor hos användaren och dessa känslor utgör begreppet *spelestetik* (Hunicke et al., 2004).

3.4. Intelligens och motivation
Begreppet intelligens har länge varit ett ämne för diskussion i akademiska sammanhang, något som har resulterat i ett flertal diskussioner kring vad som kan betraktas som intelligens eller intelligent agerande (Deary, 2013). Forskaren och psykologen Robert Sternberg definierar intelligens som ”one’s ability to learn from experience and to adapt to, shape, and select environments” (Sternberg, 2012, s.501). Detta beskriver intelligens som en människas förmåga att lära från erfarenhet och att anpassa, forma samt välja miljöer. Även om intelligens delvis är något som tycks vara ärftligt så är det vedertaget att såväl miljö som yttre faktorer styr intelligens hos en individ (Deary, 2013). Det har argumenterats för att intelligens är en avgörande faktor vid akademiska studier, men såväl motivation som lärandestrategier har också visat sig vara viktiga faktorer (Murayama et al., 2013; Rosander et al., 2011). Motivation är därmed ett centralt begrepp inom utbildning: ett flertal försök har genomförts för att öka motivationen bland studenter i form av olika slags belöningar, endera konkreta eller symboliska (Deci et al., 2001). Dessutom kan denna yttre motivation vara av värde för studenter – i synnerhet yttre belöningar som resulterar i något form av certifikat eller liknande bevis på uppnådd utbildning eller förvärvad kunskap (Hubackova & Semradova, 2014).

3.5. Pedagogik inom programmering
Som tidigare nämnts i kapitel 1 existerar svårigheter inom ämnet programmering, vilket upplevs som ett abstrakt ämne av somliga studenter. Detta har föranlett skapandet av olika simulatorer, spelliknande miljöer och program för att underlätta förståelsen för studenterna (Jayasinghe & Dharmaratne, 2013). Mow (2008) har genomfört en övergripande studie som beskriver många av dessa svårigheter och menar att detta problem existerar över ett brett spektrum av nybörjare, där även andraårsstudenter vid högskolor upplever programmering som problematiskt. Gällande val av programmeringsspråk för nybörjare har såväl Java som C++ valts av många

lärare (Lahtinen et al., 2005). De moment inom programmeringskurser vilka har upplevts som problematiska är bland annat begreppen *loopar*, hänvisas hädanefter som slingor, *villkorliga satser* och *datatyper* (ibid). Klasser och objekt har även visat sig vara begrepp vilka upplevs som svårförståeliga av studenter (Milne & Rowe, 2002). Det har även visats att studenter har svårt för att kunna applicera kunskap i praktiken, formulera strategier för att lösa ett problem och lära sig programmeringssyntax (Corral et al., 2014).

Olika metoder och strategier har nyttjats för att effektivt lära ut objektorienterad programmering till studenter, men det existerar ingen vedertagen och optimal metod (Vihavainen et al., 2011). Traditionellt sett har det funnits två olika metoder gällande utlärning av objektorienterad programmering: *objects first*, där objekt lärs ut först, eller *objects later*, där traditionella element – såsom variabler, villkorliga satser och tilldelningar – lärs ut först (Ehlert & Schulte, 2009). Det finns en rådande uppfattning om att ”objects first” är en bättre metod än ”objects later”, vilket motiveras av att objekt är centrala inom objektorienterad programmering (Uysal et al., 2012). ”Objects first” kan dock medföra vissa svårigheter för studenter, eftersom begrepp såsom klasser, konstruktorer och inkapsling lärs ut samtidigt som villkorliga satser, typer och variabler (Cooper et al., 2003). Det har även påstått att val av ”objects first” eller ”objects later” inte påverkar lärandeförmågan nämnvärt hos studenter (Ehlert & Schulte, 2009). Det finns ytterligare en metod som istället fokuserar på objektorienterade problem som ger visuell återkoppling, vilket kan utföras i ett pedagogiskt datorspel (Leutenegger & Edgington, 2007). Därmed introduceras objekt och traditionella kunskapsmoment samtidigt i denna metod och studenterna kan få en uppfattning om att ändringar i koden påverkar objekt (ibid). Denna metod benämns av Leutenegger & Edgington (2007) som *Games first*. Det finns även spelifierade produkter som är ämnade att engagera programmeringsstudenter, såsom frågesportsapplikationen ”Quick Quiz” (Cheong et al., 2013). Denna applikation använder sig av återkoppling och topplistor för att motivera studenter till egenstudier inom programmering, vilket har visat sig vara ett uppskattat upplägg (ibid).

En annan metod, vilken används inom såväl industriella som akademiska sammanhang är *pair programming*, vilket innebär att två personer arbetar tillsammans med samma programmeringsuppgift (Beck, 2000). Metoden utgörs av att den ena programmeraren arbetar med implementation av uppgiften medan den andra programmeraren analyserar och utvärderar uppgiften (ibid). Denna metod har visat sig vara användbar inom akademiska sammanhang, där metoden har potential att höja studenternas studieresultat (Salleh et al., 2011). Den upplevda känslan av nöje hos studenter, vilka nyttjade metoden ”pair programming”, var därtill högre jämfört med studenter som programmerade självständigt (ibid). Det poängteras även att nyttjandet av metoden bör anpassas efter kunskapsnivå hos studenterna, genom att bilda par av studenter med likadan kunskapsnivå inom programmering (ibid).

En annan metod som har använts inom dessa sammanhang är nyttjandet av datorspel, i form av att studenterna lär sig programmering genom att spela ett datorspel

(Muratet et al., 2009). Ett antal sådana spel har skapats och visat sig vara uppskattade av studenter och studenterna upplevde att deras motivation ökades med hjälp av dessa spel (Kazimoglu et al., 2012). Det har även identifierats att återkoppling i form av emblem eller poäng är önskvärt i dessa spel, eftersom belöningar har potential att motivera studenten att fortsätta lära sig (ibid). Det har även konstaterats att denna typen av spel har haft en positiv inverkan på studenter, vilka har förbättrat sin problemlösningsförmåga genom att spela ett pedagogiskt spel (Liu et al., 2014).

Problembaserat lärande är en metod vilken har använts inom utbildningssammanhang under en lång tid (Wood, 2003). Problembaserat lärande beskrivs som ”not about problem solving per se, but rather it uses appropriate problems to increase knowledge and understanding” (Wood, 2003, s.1), vilket visar att problembaserat lärande inte handlar om problemlösning i sig, utan mer handlar om att använda relevanta problem för att öka kunskap och förståelse. Därtill fokuserar problembaserat lärande kring ett uppmuntrande av att självständigt söka efter information som kan lösa uppgiften (Kilroy, 2004). Somliga definitioner av problembaserat lärande fokuserar kring att lösa ett svårlöst problem, men studenternas analyserande av problemet anses vara det centrala inom problembaserat lärande (Nuutila et al., 2005; Kilroy, 2004). Problembaserat lärande har använts inom programmeringskurser och en motivationshöjande effekt har kunnat ses som en följd av detta nyttjande (Nuutila et al., 2005). Denna metod har även nyttjats för att minska oro hos studenter som upplever svårigheter med att lösa problem (ibid). Gällande upplevd oro hos studenter som bemöter svåra problem har det konstaterats att fel som begås av studenten under lärandeprocessen är något som är konstruktivt – så småningom leder dessa misstag till framgång, vilket stärker den kunskap som erhålls (Jerinic, 2014). Tvärtemot vad somliga lärare anser bör inte felfritt lärande vara i fokus – felbetingade problem medför en djupare förståelse för ämnet jämfört med direkt instudering av rätt svar (ibid).

3.6. Redan existerande produkter

Det finns en uppsjö av spel med fokus på problemlösande och pussel: FEZ, Braid, Limbo och The Swapper är exempel på sådana spel. Om problemlösning istället begränsas till sådant som involverar programmering blir skaran genast mindre. Glitchspace är ett spel där programmering används för att lösa problem och dessa problem är utformade i form av objekt som kan manipuleras, till exempel genom att ändra egenskaper hos objektet (Budgie, 2015). Glitchspace är dock, tillsammans med övriga spel som listats ovan, ämnat att användas i underhållande syfte. I dagsläget finns det ett fåtal spel vars syfte är att motivera ett ökat intresse för, eller förbättra lärandet inom, programmering och datavetenskapliga ämnen. Det finns tydligare exempel på spel som har använts direkt i undervisande syfte: ett exempel på ett sådant spel är Math Blaster, vilken har använts tidigare för att förstärka lärandet av matematik (Rodrigo & Baker, 2011). En rad olika företag har även valt att fokusera på spelifiering: företaget Bunchball lanserade år 2012 en *gamification engine* vid namn ”FlameThrower”, vilket möjliggör skapandet av spelifierade applikationer för företag (Technology News Focus, 2012). Detta kan liknas vid spelmotorer, såsom Unreal Engine, men med ett tydligt fokus mot spelifiering.

4. Analys av teoretisk referensram

I detta kapitel presenteras den analys som har gjorts med hjälp av den teoretiska referensramen. Kapitlet inleds med en analys av målgrupp och vilket kunskapsomfång som spelet ska innefatta. Därefter analyseras de pedagogiska metoder samt typer av spelmekanik som ska användas i spelet. Slutligen analyseras spelet: hur det ska skapas, vilken spelmotor som ska användas samt vilket dataformat som ska nyttjas. Dessa analyser resulterade därefter i en kravspecifikation för spelet, vilken beskrivs i Bilaga B.

4.1. Målgrupp och kunskapsomfång

I kapitel 1 konstateras det att somliga studenter upplever att det existerar svårigheter vid inlärning av ämnet programmering. En studie genomförd av Mow (2008), som beskrivs i delkapitel 3.5, visar att problem vid inlärning existerar både bland nybörjare samt andraårsstudenter vid högskolor. Denna studie styrks även av Tan et al. (2009) som menar att det är en vanlig åsikt bland förstaårsstudenter att programmering upplevs som ett problematiskt ämne. I enlighet med det som nämnts ovan, valde vi att begränsa målgruppen till studenter med ingen eller liten tidigare erfarenhet av objektorienterad programmering. Det är dessutom relevant att identifiera vilka kunskapsmoment som upplevs problematiska av studenter. I delkapitel 3.5 såg vi att de begrepp som vanligtvis upplevs som problematiska är: slingor, villkorliga satser och datatyper, samt att klasser och objekt. Med anledning av detta anser vi att spelet bör innehålla dessa moment. Det är därtill värt att diskutera hur dessa kunskapsmoment ska inkluderas i spelet samt i vilken ordning kunskapsmomenten ska presenteras.

De kunskapsmoment som spelet ska innehålla samt i vilken följd de ska presenteras i, beslutades huvudsakligen enligt ”Learning the Java Language” vilken är Oracles handledning för Java. Oracles handledning valdes då vi anser att nybörjare inom programmering förmodligen kommer att söka sig till den officiella handledningen för Java. I handledningen beskrivs inledningsvis konventionella objektorienterade koncept – såsom objekt och klasser – följt av språkets byggstenar i följande ordning: datatyper, tilldelningar, fält, operatorer, villkorliga satser och slingor (Oracle, 2015). Handledningen innehåller även ett flertal övriga koncept som kan inkluderas i spelet, dock beslutades det med avseende på projektets tidsbegränsningar att utesluta dessa. Med anledning av att spelet inte kan inkludera samtliga koncept beslutades det även att spelet enbart ska användas som ett hjälpmedel vid utbildning, snarare än ett substitut för en hel programmeringskurs. Det beslutades även att spelet ska delas in i nivåer för att uppnå samma följd av kunskapsmoment som Oracles handledning beskriver. Totalt ska minst tre nivåer skapas: den första nivån ska fokusera på datatyper och tilldelningar. Den andra nivån ska inrikta sig på villkorliga satser och operatorer. Den tredje nivån ska fokusera på slingor. Övriga nivåer ska inkludera begrepp såsom klasser och arv. Det bör även poängteras att nivåerna i spelet ska bero på föregående nivåer och således inkluderas tidigare kunskapsmoment i nästkommande nivåer.

4.2. Pedagogisk metod

I delkapitel 3.5 nämndes huruvida ”objects first”- eller ”objects later”-metoden ska nyttjas inom objektorienterade programmeringskurser. Ehlert & Schulte (2009) anser dock att det inte existerar någon större skillnad mellan dessa metoder vad gäller studenters lärandeförmåga. Det måste dock framhävas att objekt är centrala inom objektorienterad programmering, något som styrks av Kölling (1999) och Uysal et al. (2012). Därtill konstaterar Leutenegger & Edgington (2007) att problem relaterade till objekt är nödvändigt i pedagogiska spel som syftar till att lära ut objektorienterad programmering. Vi anser därmed att det är relevant att introducera objekt tidigt i spelet. Tilldelningar, variabler och villkorliga satser är dock moment vilka har nämnts som problematiska att förstå för studenter, vilket beskrivs i delkapitel 3.5. Detta föranleder vår slutsats om att objekt bör introduceras parallellt med begrepp såsom tilldelningar, variabler och villkorliga satser. Dessa moment kan utföras i objekt, vilket medför att användaren kan få en uppfattning om att objekt har egenskaper vilka kan ändras. Objektorienterad programmering innehåller dock fler moment som är centrala för ämnet. Somliga av dessa har nämnts i delkapitel 3.5 och spelet bör därmed innefatta dessa begrepp.

Liu et al. (2014) hävdar att pedagogiska spel kan öka studenternas problemlösningsförmåga. Dessa spel bör dock innehålla relevanta och väl formulerade problem för att en ökad problemlösningsförmåga ska kunna uppnås, enligt Wood (2003). Problembaserat lärande, som nämndes i delkapitel 3.5, uppmuntrar studenter till att självständigt söka egen information för att lösa uppgifter. Med anledning av detta anser vi att problembaserat lärande är en metod som bör nyttjas vid utformningen av problemen i spelet. Därtill ska spelets programmeringsrelaterade problem utformas med inspiration från boken ”Java direkt med Swing”, vilken skrevs av Jan Skansholm (2013). Boken valdes med anledning av att den nyttjas i en introduktionskurs för objektorienterad programmering på Chalmers samt att projektets medlemmar har positiva tidigare erfarenheter av boken. Vidare ska uppsökning av relevant information eller kurslitteratur uppmuntras i spelet. Såväl ”pair programming” som problembaserat lärande nyttjar gruppaktiviteter för att förbättra studenternas studieresultat, vilket föranleder rekommendationen om att vårt spel bör spelas i grupper av två studenter.

4.3. Spelmekanik

Teorin i delkapitel 3.1 beskriver att datorspel innehåller ett komplett användande av spelelement och spelmekanik. Det finns dock vissa typer av spelmekanik, vilka är centrala för spelifiering, som har potential att öka motivationen hos användaren. Den mest vedertagna definitionen av Deterding et al. (2011) utesluter spelifiering av ett datorspel. Spelifiering visar däremot att utvald spelmekanik såsom belöningssystem och återkoppling kan ha en motivationshöjande effekt. Detta föranleder tanken om att en eller flera av dessa typer av spelmekanik bör betonas i spelet, eftersom motivation har lyfts fram som en viktig komponent vid akademiska studier i delkapitel 3.4.

Återkoppling nämns som en viktig komponent i spelifiering av Buckley & Doyle (in press), Cheong et al. (2013) och Muntean (2011). Spelmekanik såsom poängsystem,

emblemsystem och topplistor får anses tillgodose återkoppling, eftersom dessa spelmekaniker ger återkoppling i form av belöningar. Ett alternativ som vi diskuterade var att implementera ett poängsystem där användaren får poäng baserat på hur bra denne har löst programmeringsrelaterade problem i spelet, men detta ansågs vara problematiskt att implementera. Problematiken uppstår till följd av att spelet i så fall måste innehålla en bedömande funktionalitet för att avgöra hur bra användaren har lyckats genomföra spelets olika moment. Detta ska dessutom värderas rättvist för att poängsystemet ska ge korrekt återkoppling. Denna variant av återkoppling sågs förvisso som önskvärd, men det hade i praktiken varit tidskrävande att implementera.

Topplistor har tidigare nämnts i delkapitel 3.3 som ett tävlande moment inom datorspel för att öka motivationen hos användarna. Därtill kan användaren bilda ett statusvärde med hjälp av topplistor, för att motivera användaren att förbättra sina resultat (Bunchball, 2010). I delkapitel 3.3 nämns det dock att tävlande moment kan upplevas som hindrande och riskerar därmed att motverka det motiverande syftet med denna spelmekanik. Det valdes dock att diskutera huruvida det var möjligt att implementera en topplista, som inte medför prestationskrav för användaren, i spelet. Ett förslag som framfördes var att införa möjligheten för användarna att inte medverka i topplistan. Det var dock svårt att se något syfte med att implementera en topplista i detta fall, eftersom somliga användare inte motiveras av denna spelmekanik. Topplistor får därmed anses vara olämpligt att nyttja i utvecklingen av spelet, med anledning av att det finns studenter som upplever att programmeringsrelaterade problem ger prestationsångest (Nuutila et al., 2005).

Emblemsystem beskrivs i delkapitel 3.3 som en spelmekanik vilken kan förtydliga användarmål och motivera användaren att slutföra målen. Detta styrks även av Bunchball (2010), som menar att emblem nyttjas optimalt om dessa kan uppvisas för andra användare. Vi ansåg att ett emblemsystem skulle implementeras i spelet med anledning av att detta möjliggör skapandet av tydliga mål för användaren. Utdelning av belöningar vid genomförande av målen ansågs även vara en önskvärd effekt för att motivera användaren. Det finns också relativt få nackdelar med detta system i de artiklar som har inkluderats i delkapitel 3.3.

Återkoppling nämns genomgående i de för spelifiering centrala typer av spelmekanik, men enligt Hamari et al. (2014) är återkoppling en separat spelmekanik. I delkapitel 3.3 exemplifieras återkoppling i form av en frågesport för studenter. Vi anser att återkoppling i form av en frågesport är önskvärt att inkludera i spelet för att användaren ska få återkoppling på erhållen kunskap. Implementation av en frågesport kan dessutom medföra att studenter analyserar krävande problem, vilket är centralt inom nyttjandet av problembaserat lärande. Frågesporten bör därtill innehålla frågor om de kunskapsmoment som nämns i 4.1. Övrig återkoppling, såsom ljud och visuella effekter, bör även inkluderas enligt delkapitel 3.2 för att nyttja alla typer av spelmekanik. Det är även viktigt att poängtera att fler typer av spelmekanik såsom spelfysik ska ingå i spelet med anledning av att det är ett seriöst spel som ska skapas. Dessa typer är dock inte centrala för spelifiering och betonas därmed

inte i spelet.

4.4. Programmeringsspråk, ramverk och format

I projektets planeringsfas ansågs val av programmeringsspråk, ramverk och format vara en central del av projektet. Likaså ansågs valet av spelmotor vara betydelsefullt, då detta påverkade hur utvecklingen av spelet skulle genomföras. Flertalet förslag på eventuella spelmotorer lades fram, däribland: Slick2D, Unreal Engine, IdTech2, Unity och Ogre. Dessa valdes dock bort, då de antingen fokuserade på tredimensionell grafik, nyttjade ett programmeringsspråk vilka vi inte var bekanta med eller hade begränsningar såsom ofullständig dokumentation. Därmed beslutades det att använda LibGDX för att utveckla spelet. Detta beslut genomfördes med anledning av att LibGDX är baserat på det objektorienterade programmeringsspråket Java, något som vi hade tidigare erfarenhet av. Därtill hävdar Lahtinen et al. (2005) att Java är ett programmeringsspråk som har valts av många lärare i introduktionskurser inom programmering. Ytterligare en faktor vilken bidrog till valet av spelmotor var att LibGDX nyttjar grafikbiblioteket *Open Graphics Library*, vilket hädanefter hänvisas som OpenGL. Detta är ett plattformsoberoende *Application Programming Interface*, vilket hädanefter hänvisas som API, med stöd för att skapa applikationer med två- eller tredimensionell grafik (Silicon Graphics, 2014). Därutöver har LibGDX ett aktivt forum och är väl dokumenterat, något som ansågs vara önskvärt för att underlätta utvecklingen av spelet.

För att läsa in och skriva *speldata*, vilket är data som till exempel används för att spara användarens framsteg i spelet, ska dataformatet *JavaScript Object Notation* (JSON) användas. Detta är ett språkoberoende och textbaserat datautbytesformat (JSON, 2015). JSON valdes huvudsakligen med anledning av att det använder syntaxkonventioner som är välkända och användarvänliga för programmerare samt att det är enkelt för en dator att tolka och generera detta utbytesformat (Nurseitov et al., 2009).

5. Implementation

Följande kapitel beskriver hur spelet, dess kategorier och viktiga funktioner implementerades. En kategori kan ses som ett samlingsnamn för delar av spelet som innehåller viktig funktionalitet, såsom huvudmeny, frågesport eller prestationer. Inledningsvis presenteras de grafiska resurser och datastrukturer som nyttjades för att utveckla spelet. Därefter redogörs för spelets kategorier i samma ordning som kategorierna återfinns i spelets huvudmeny.

5.1. Grafiska resurser och hantering av speldata

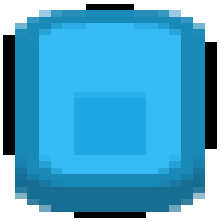
För att skapa ett genomgående tema i spelet upprättades ett ramverk för utseende av kategorier, innehållandes regler som presenteras nedan:

- Vid utvecklingen av samtliga kategorier, med undantag för kategorin ”Spelläge”, användes tabeller för att upprätta en struktur över de grafiska elementen. Överst i tabellen placerades en *Label*, som hädanefter hänvisas som en etikett. Etiketten innehöll en beskrivande titel för kategorin.

- Vid implementation av knappar i spelet används genomgående en *NinePatch-bild* som bakgrundsbild för knapparna, vilket är en bild med fördefinierade och tänjbara ytor (Badlogic Games, 2015). Ett exempel på en sådan bild illustreras i figur 5.1. Bilden omringas av en ram, bestående av svarta streck, som är en pixel bred. Det vänstra och det översta strecket representerar hur bilden ska tänjas. Det högra och det nedersta strecket representerar hur text ska placeras på bilden.

- Under formgivning av spelets utseende användes *Skin*, hädanefter skal, som beskriver hur element såsom textfält eller knappar skulle se ut. Mindre byggstenar, såsom teckensnitt, fördefinerade färger och bilder, byggde upp dessa grafiska element och behövde därför inte definieras upprepade gånger (Badlogic Games, 2015).

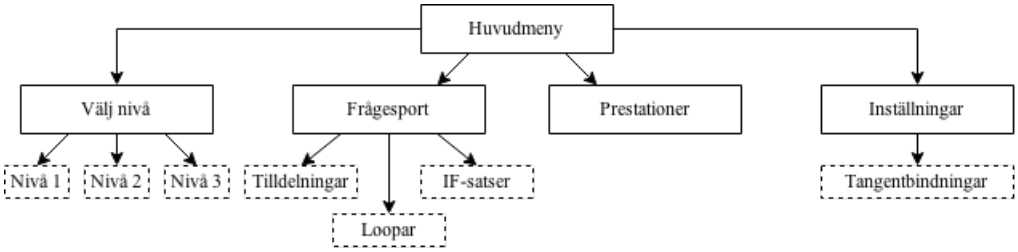
- För att läsa in, skriva och lagra speldata, i enlighet med delkapitel 4.4, användes datautbytesformatet JSON. Ett flertal övergripande exempel av hur JSON nyttjas demonstreras i Bilaga C.



Figur 5.1: Ett exempel på hur NinePatch användes i kombination med bilder. De svarta strecken representerar hur bilden ska tänjas samt hur text ska placeras i den.

5.2. Huvudmeny

När spelet startas bemöts användaren av huvudmenyn, vars huvudsakliga uppgift är att navigera användaren genom spelet. Spelet har en trädliknande struktur, där huvudmenyn kan ses som roten i trädet. Från huvudmenyn grenas spelet ut i olika kategorier, vilket kan ses i figur 5.2.



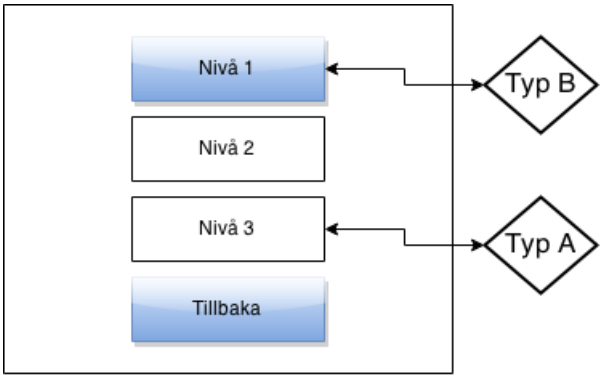
Figur 5.2: Spelets trädliknande struktur. Rutor med heldragen ram representerar kategorier i spelet och rutor med streckad ram motsvarar möjliga val inom valda kategori.

Huvudmenyn består i enlighet med delkapitel 5.1 huvudsakligen av en tabell och en skiss på dess preliminära utseende kan ses i figur 5.3. I tabellen placerades olika valbara alternativ, vilka visar sig i form av klickbara knappar: ”Nytt spel/Återuppta nivå”, ”Välj nivå”, ”Frågesport”, ”Prestationer”, ”Inställningar” och ”Avsluta spel”. Om användaren väljer ett alternativ förutom det förstnämnda eller sistnämnda startas den valda kategorin. Om användaren väljer det förstnämnda alternativet, i fallet då knappen innehåller texten ”Nytt spel”, startas en ny instans av kategorin ”Spelläge”. Om knappen innehåller texten ”Återuppta nivå” återupptas den redan påbörjade instansen av kategorin ”Spelläge”. Om användaren väljer det det sistnämnda alternativet avslutas spelet.

Figur 5.3: En skiss på spelets huvudmeny.

5.3. Välj nivå

I denna kategorin får användaren en möjlighet att välja bland redan avklarade nivåer i spelet samt nästa oavslutade nivå. Syftet med den här kategorin är att användaren ska få en möjlighet att förbättra eller förnya sina kunskaper genom att repetera redan avklarade moment. ”Välj nivå”-kategorin utformades på samma sätt som kategorin ”Huvudmeny” enligt delkapitel 5.1. Således består den här kategorin av en tabell och i tabellen har tre knappar placerats, vilka representerar de nivåer som finns i spelet. Kategorin har två skilda typer av knappar, vilka illustreras i figur 5.4. En knapp av typ A används när en nivå inte är tillgänglig (med andra ord har inte nivån avklarats, med undantag för nästkommande nivå) och är således inte klickbar. Knapp B indikerar att knappen är klickbar och används i det motsatta fallet då nivån är tillgänglig för användaren. Om användaren klickar på en knapp av typ B startas den valda nivån.



Figur 5.4: En designprototyp på kategorin ”Välj nivå”.

5.4. Spelläge

När användaren har valt en nivå i spelet startas kategorin ”Spelläge”. Kategorin består av tre tillstånd: i det första tillståndet spelar användaren spelet och detta tillstånd startas automatiskt när kategorin har valts. I det andra tillståndet är spelet fryst, vilket innebär att man inte kan flytta på huvudkaraktären, och användaren kan modifiera somliga objekt i spelet. I det tredje tillståndet presenteras en hjälpruta för användaren. Hjälprutan innehåller information som användaren kan nyttja för att klara nivån.

Längst upp i ”Spelläget” placerades ett användargränssnitt, som har en bakgrund bestående av en gradient bild. I användargränssnittet placerades två klickbara knappar: den första knappen startar det tredje tillståndet. Den andra knappen öppnar en meny som innehåller knapparna: ”Återuppta”, ”Inställningar” och ”Avsluta spel”. Därutöver placerades även en etikett i användargränssnittet, innehållandes information om *frames per second*, hädanefter fps, som är ett mått på antalet bilder per sekund.

Tillstånd 1: Spelläge

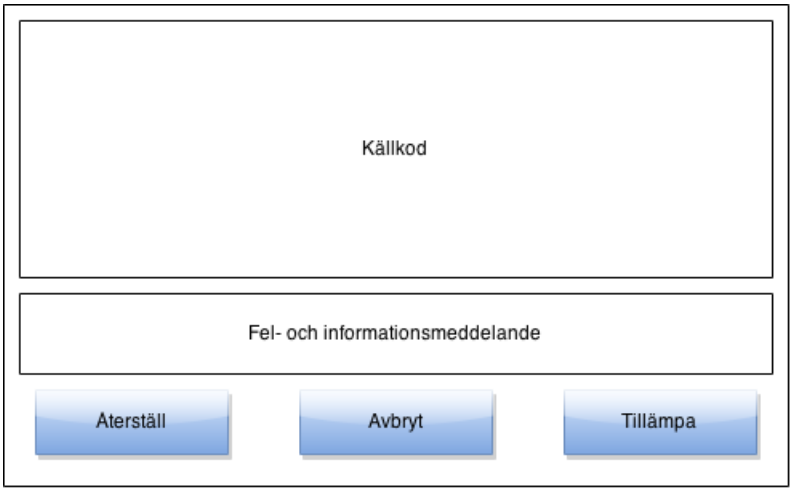
I detta tillstånd kan användaren spela de nivåer som finns i spelet. För att skapa nivåer i spelet användes Tiled, vilket är i enlighet med delkapitel 4.4. En nivå i Tiled

är uppbyggd av ett rutnät av kvadratiska fält, där fälten antingen består av bilder eller objekt. Rutnätet omfattas av olika lager, där ett lager beskriver hur fälten ska tolkas av spelet. Exempelvis skapas kvadratiska rutor som huvudkaraktären kan gå på vid lagernivå ”ground”. I lagernivå ”background” skapas rutor som huvudkaraktären inte kan interagera med. Om ett objekt har placerats i ett fält utgör detta fält en specifik funktion, såsom startposition för huvudkaraktären, dödszon eller kontrollpunkt. Dessa objekt presenteras och förklaras mer utförligt nedan:

- Objektet ”PlayerSpawn” används för att ge huvudkaraktären en position att starta på när en nivå inleds. I fallet då en nivå återupptas utnyttjas koordinater, som matas in till kategorin, för att förflytta huvudkaraktären till specificerad position.
- Objektet ”Checkpoint”, som hädanefter hänvisas som kontrollpunkt, används för att ge användaren en möjlighet att spara sitt spel efter avklarad etapp i spelet. På så vis bestraffas inte användaren vid eventuella misslyckanden, exempelvis när huvudkaraktären dör, något som hade medfört att huvudkaraktären istället hade återuppstått i början av spelet. När huvudkaraktären kommer i kontakt med en kontrollpunkt sparas dess aktuella position och används därefter som startposition för huvudkaraktären.
- Objektet ”KillZone” används för att döda huvudkaraktären. Detta inträffar då spelaren befinner sig i en förbjuden zon såsom under vattenytan. Huvudkaraktären återuppstår därefter vid senaste kontrollpunkt, med anledning av att huvudkaraktären varken har *livmätare* eller *extraliv* i spelet.
- I slutet av varje nivå placerades objektet ”EndZone”. Detta objekt avslutar den aktuella nivån och startar även nivåns tillhörande frågesport, vars funktionalitet förklaras i sektion 5.5.

Tillstånd 2: Modifiera objekt
Kategorins andra tillstånd startas när användaren högerklickar på ett objekt. För att ge användaren grafisk återkoppling, och en insikt om vilka objekt som kan modifieras, ger dessa objekt ifrån sig ett blått sken. När användaren högerklickar på ett objekt fryser spelet och en tabell presenteras. Om objektet innehåller en parameter med namnet *CodeEnv* genereras en *kodmiljö*, vilken är en tabell som innehåller källkod. Kodmiljön lokaliserar därefter en fil som specificerats i Tiled. Spelet läser sedan in filens innehåll och därefter kan användaren ändra på kod i kodmiljön, vilken interagerar med spelet under körtid och påverkar därmed spelet dynamiskt. Kodmiljön kan därmed nyttjas av användaren för att ändra på kod i objekt. Med hjälp av denna funktionalitet kan användaren lösa problem i spelet och på så vis ta sig vidare.

Kodmiljön illustreras i figur 5.5 och består huvudsakligen av två textfält: i det första textfältet placeras källkoden. Det andra textfältet går inte att interagera med då den enbart är till för att visa fel- och informationsmeddelanden som uppstår under kompilering eller exekvering av användarens kod. Utöver detta placeras tre knappar i kodmiljön: ”Återställ”, ”Avbryt” samt ”Tillämpa”.



Figur 5.5: En skiss på spelets kodmiljö.

Eftersom källkod kan innehålla mycket semantik som inte är relevant för användaren, såsom klass- och variabledeklarering, implementerades ett textbaserat system. Detta möjliggjorde notering av vilka delar i källkoden som ska visas för användaren samt vilka av dessa delar användaren kan ändra på. Systemet läser igenom källkoden och letar efter speciella nyckelord i slutet av varje rad, vilket medför att dessa noteringar kan ändras med enkelhet jämfört med om dessa hade varit statiskt angivna i spelet.

Därtill implementerades säkerhetsåtgärder för att kontrollera att användarens inmatning inte medför en oönskad inverkan på spelet. Dessa åtgärder innebar att de klasser som skapas av användaren separeras från spelets egna klasser. Dessutom garanterade åtgärderna att exekvering av användarens kod sker på ett sådant sätt att eventuella fel eller undantag fångas innan detta medför att resten av spelet krashar.

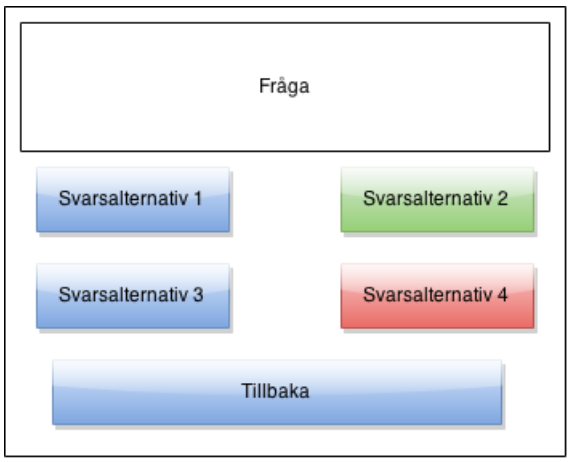
Tillstånd 3: Hjälpruta
Varje nivå i spelet har en förutbestämd textfil innehållandes information, vilken är ämnad att hjälpa användaren att genomföra nivån. När det tredje tillståndet startas, vilket görs via en tangentbindning eller en klickbar knapp i användargränssnittet, läses den bestämda textfilen in och placeras inuti en etikett. Etiketten placeras sedan inuti en tabell och presenteras därefter för användaren.

5.5. Frågesport
Kategorin ”Frågesport” är uppdelad i tre tillstånd: i det första tillståndet kan användaren välja mellan tillgängliga frågesporter. I det andra tillståndet får användaren svara på frågor som ingår i frågesporten, vilken valdes i det första tillståndet. I det tredje och sista tillståndet har användaren slutfört frågesporten och får därmed sitt resultat förmedlat.

Tillstånd 1: Val av frågesport
Varje nivå i spelet har en tillhörande frågesport vars innehåll utgörs av grundläggande frågor inom ett visst område, såsom tilldelningar och villkorliga satser. Om användaren klarar en nivå i spelet så startar den tillhörande frågesporten. Som

nämnts ovan har användaren även en möjlighet att starta frågesporten utan att ha klarat av tillhörande nivå tidigare. Tillståndet följer rekommendationerna som beskrivs i delkapitel 5.1 och består därmed av en tabell. I tabellen har klickbara knappar placerats, där en knapp representerar en nivå.

Tillstånd 2: Spela frågesport
Varje frågesport i spelet består av en fråga och fyra möjliga svar. Tillståndet implementerades med hjälp av en tabell: i översta raden i tabellen har en etikett placerats innehållandes frågan. Svarsalternativen synliggjordes med hjälp av klickbara knappar, där två knappar placerades per tabellrad. Om användaren klickar på ett svar uppstår en avsedd fördröjning på en sekund. Under denna sekund får användaren grafisk återkoppling på sitt svar genom att knapparnas bakgrund byts ut, vilket visas i figur 5.6.

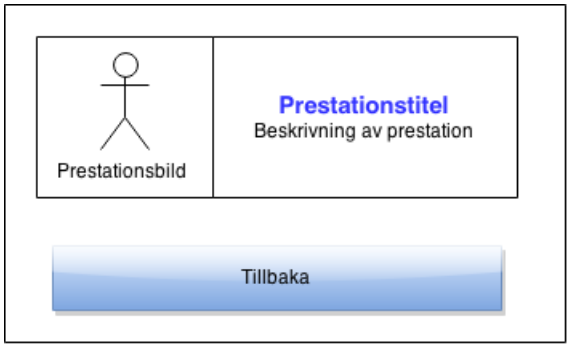


Figur 5.6: En skiss på spelets frågesport. Grön knapp motsvarar korrekt svar och röd knapp motsvarar felaktigt svar.

Tillstånd 3: Utvärdering av frågesport
När användaren har besvarat samtliga frågor i frågesporten startas det tredje tillståndet. Användaren får därmed omedelbar respons på sina svar i detta tillstånd. Resultatet från frågesporten åskådliggörs med hjälp av en etikett, vilken innehåller antalet korrekta svar som användaren svarade. I enlighet med det genomgående temat på spelet har etiketten placerats inuti en tabell.

5.6. Prestationer

I spelet kan användaren belönas med prestationer vid förtjänstfulla insatser, såsom att klara av en nivå eller slutföra en frågesport. Kategorin ”Prestationer” byggdes upp som en förteckning över alla prestationer i spelet. Kategorin består av en tabell, inom vilken alla prestationer placerades. Varje prestation består av en bild, en beskrivande förklaring och en titel. Ett exempel på en prestation kan ses i figur 5.7. Om användaren inte har låst upp prestationen kommer detaljer kring prestationen att vara oåtkomliga. I fallet då användaren låser upp en prestation animeras en notifikation i spelet och alla tillhörande detaljer blir därmed tillgängliga. Detta är därmed ett emblemsystem, vilket vi har valt att namnge som ”Prestationer”.



Figur 5.7: Ett utkast på kategorin ”Prestationer”.

5.7. Inställningar

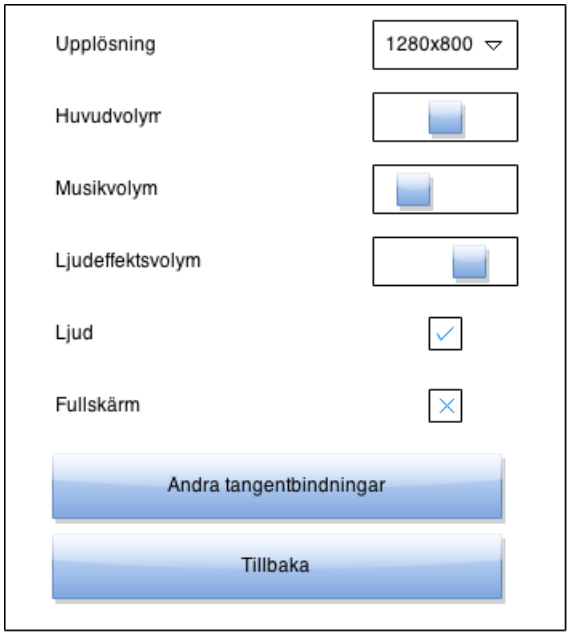
Kategorin ”Inställningar” består av två tillstånd: i det första tillståndet ges användaren en möjlighet att ändra på inställningar såsom upplösning och volym. I det andra tillståndet får användaren en möjlighet att ändra tangentbindningarna i spelet.

Tillstånd 1: Generella inställningar

När kategorin startas aktiveras automatiskt det första tillståndet. Likt andra kategorier, består detta tillstånd av en tabell. Tabellen består av två kolumner: i den vänstra kolumnen placerades en etikett, innehållande en beskrivande förklaring för inställningen i kolumnen till höger om etiketten. I den högra kolumnen placerade inställningarna. Inställningarna presenteras i följande ordning i spelet:

- Första inställningen i spelet ger användaren en möjlighet att ändra skärmupplösning, färgdjup och uppdateringsfrekvens. För att visa upp de olika valmöjligheterna användes en rullgardinslista.
- De tre nästkommande inställningarna hanterar ljudnivåer i spelet. Ljudnivåerna är indelade i tre olika delar: huvudvolym, musikvolym och ljudeffektsvolym. Dessa inställningar representeras med hjälp av ett reglage.
- De två sista inställningarna i spelet, *visningsläge* och *ljudläge*, implementerades med hjälp av en kryssruta. Kryssrutan för visningsläget anger om fullskärmsläge eller fönsterläge används. På samma sätt anger kryssrutan för ljudläget om ljud är aktiverat eller inte.

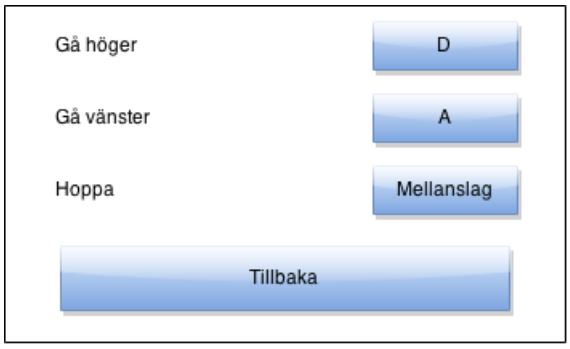
I nedersta raden i tabellen placerades en klickbar knapp. Om användaren klickar på knappen så startar det andra tillståndet. En skiss på de generella inställningarna i spelet kan ses i figur 5.8.



Figur 5.8: En skiss på spelets generella inställningar.

Tillstånd 2: Tangentbindningar
Det andra tillståndet är utformat likt det första tillståndet, med skillnaden att en etikett har placerats i den vänstra kolumnen, vilken beskriver tangentbindningen i kolumnen till höger. I den högra kolumnen återfinns alla tangentbindningar. Tangentbindningarna, som kan ses i figur 5.9, har implementerats med hjälp av ett textfält, som maximalt kan innehålla en bokstav. Vid undantagsfallet då användaren väljer en specialtangent, som inte kan representeras av en bokstav, kommer hela ordet att skrivas ut.

Om användaren klickar på ett av textfälten och väljer en ny tangentbindning kommer den tidigare tangentbindningen att ersättas. I fallet då användaren väljer en tangentbindning som redan är upptagen blir den tidigare tangentbindningen obunden.



Figur 5.9: En skiss på några av spelets tangentbindningar.

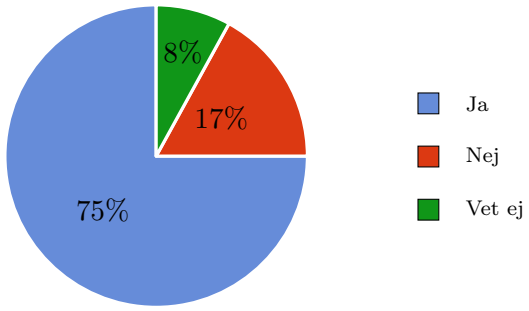
6. Resultat

Detta projekt har resulterat i ett seriöst spel som är inspirerat av begreppet spelifiering. Spelet nyttjar främst motivationsökande spelmekanik såsom återkoppling och emblemsystem, men nyttjar därtill pedagogikmetoden problembaserat lärande. Kunskapsinnehållet i spelet utgörs av traditionella programmeringsmoment såsom tilldelningar, datatyper och villkorliga satser, i enlighet med delkapitel 4.1. Somliga kunskapsmoment såsom arv och konstruktorer utgick dock i produkten med anledning av projektets tidsbegränsningar. En enkätundersökning genomfördes i slutskedet av projektet för att utvärdera spelet.

6.1. Enkätundersökning

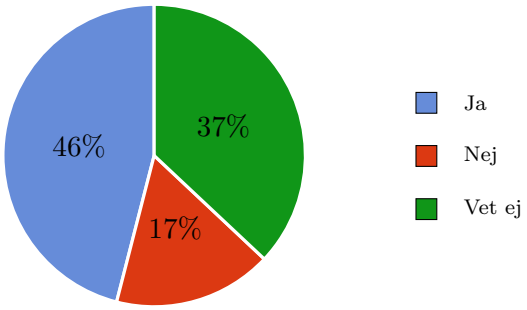
I utvärderingsfasen av projektet utformades en enkätundersökning vars syfte är att analysera spelet för den valda målgruppen. Enkätundersökningen genomfördes på 24 testpersoner och de resultat som är väsentliga för rapportens syfte presenteras nedan. Det fullständiga resultatet av enkätundersökningen kan ses i Bilaga A.2.

Resultatet av frågan: ”Upplever du att spelet ökade din motivation till studier av objektorienterad programmering?” presenteras i figur 6.1. Testpersonerna som upplevde att deras motivation hade ökat fick även möjligheten att ge synpunkter på vad som upplevdes vara motivationshöjande i spelet. En testperson nämnde följande: ”Roligt att se att något händer med objekten i spelet. Vissa pussel var kluriga och det var motiverande att faktiskt klara av dom”. Ytterligare en testperson sa: ”Kul med achievements som man kan samla på. Problemen var lagom svåra och det var roligt att klara av dem”. En annan testperson konstaterade även: ”Tycker att quizet var bra för att kolla om man har förstått saker, men det var lite svåra frågor ibland. Kanske vore bra att ha fler frågor”. Samtliga svar på denna fråga finns i Bilaga A.2.



Figur 6.1: Testpersonernas svar på frågan: ”Upplever du att spelet ökade din motivation till studier av objektorienterad programmering?”.

Resultatet av frågan: ”Kan du tänka dig att använda spelet som ett hjälpmedel vid egenstudier av objektorienterad programmering?” presenteras i figur 6.2. En stor andel svarade ”vet ej”, vilket utvecklas av en av testpersonerna: ”... beror lite på om det finns fler banor, hade kunnat spela det i så fall och använt mig av spelet för att förstå. Lite mer avslappnat än att läsa böcker och sånt.”. Samtliga svar på denna fråga finns i Bilaga A.2.



Figur 6.2: Testpersonernas svar på frågan: ”Kan du tänka dig att använda spelet som ett hjälpmedel vid egenstudier av objektorienterad programmering?”.

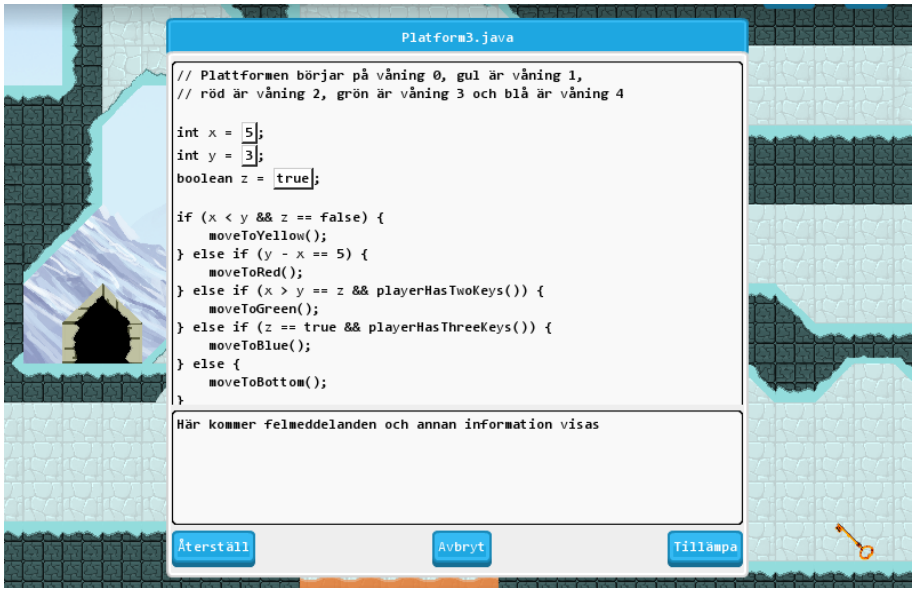
6.2. Spelet

Projektet resulterade i ett seriöst spel, vilket är inspirerat av spelifiering, som uppfyller nästintill samtliga krav i kravspecifikationen, som kan ses i Bilaga B. Spelet är utformat i olika nivåer, vilka representerar kunskapsmomenten: tilldelningar, datatyper, villkorliga satser och slingor. Objekt introduceras parallellt med dessa begrepp, i enlighet med delkapitel 4.2. Spelet är därtill inspirerat av begreppet spelifiering och nyttjar därmed de motivationsökande typerna av spelmekanik: återkoppling och emblemsystem. Huvudmenyn uppnår samtliga krav vilka upprättades i kravspecifikationen. En ögonblicksbild på huvudmenyn illustreras i figur 6.3.

Figur 6.3: Huvudmenyns utseende i den slutliga versionen av spelet.

I figur 6.4 visas ett exempel på ett programmeringsrelaterat problem i spelet. Dessa problem presenteras då användaren högerklickar på ett objekt. Problemen visas i form av ett kodskelett där användaren kan ändra på vissa delar av koden. I figuren har användaren högerklickat på ett objekt, som fungerar likt en hiss. Användaren kan därefter ändra på kod i kodskelettet för att få hissen att färdas till en annan

våning. Slutförandet av problemet medför att spelaren kan avancera vidare i spelet och slutligen klara av nivån.

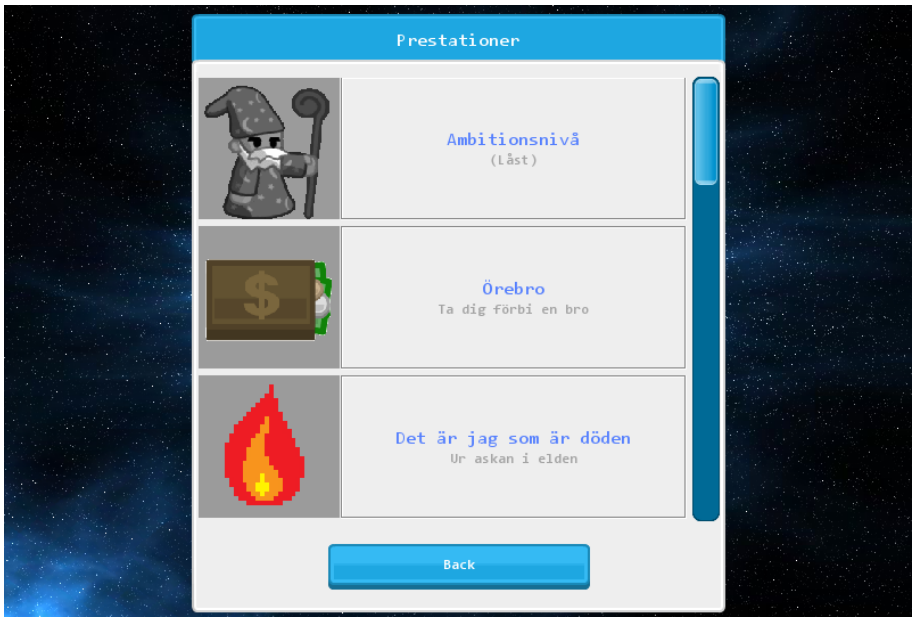


Figur 6.4: Ett programmeringsrelaterat problem i spelet. Bakgrunden i spelet är skapad av (Ren, 2015). CC-BY 3.0

En frågesport presenteras för användaren i slutet av varje nivå. Dessa frågor är relaterade till det kunskapsinnehåll som tillhörande nivå behandlar. Användaren får respons på om vederbörande har svarat rätt eller inte, och får i slutet av frågesporten sitt resultat presenterat. Ett exempel på en fråga från frågesporten visas i figur 6.5. I detta fall har användaren svarat fel på frågan och därmed visas det felaktiga svaret med hjälp av en röd bakgrund. Det korrekta svaret presenteras med hjälp av en grön bakgrund.

Figur 6.5: En av frågorna från frågesporten i spelet.

Emblemsystemet innehåller emblem vilka delas ut till användaren när vederbörande har uppnått ett mål eller delmål i spelet. Dessa emblem benämns som prestationer i spelet. Ett exempel på de prestationer som finns i spelet kan ses i figur 6.6. I detta fall har användaren tilldelats två prestationer och den översta prestationen har inte delats ut till användaren.



Figur 6.6: Spelets emblemsystem. Prestationer som inte har erhållits anges som ”låst”.

7. Diskussion

Kapitlet inleds med en diskussion som anknyter till resultatet av enkätundersökningen. Huvudsakligen diskuteras huruvida testpersonerna upplever att spelet ökade testpersonernas motivation till att studera objektorienterad programmering. Därtill diskuteras även testpersonernas åsikt till att nyttja spelet som ett hjälpmedel vid egenstudier av objektorienterad programmering. Därefter presenteras en diskussion angående arbetsmetoden som har nyttjats för projektets genomförande. Slutligen diskuteras de kunskapsmoment som spelet innefattar, val av pedagogisk metod samt möjlig funktionalitet för vidareutveckling av spelet.

7.1. Enkätundersökning

Den enkätundersökning som genomfördes visar att en majoritet av testpersonerna upplevde att spelet ökade deras motivation till att studera objektorienterad programmering. Ett flertal kommentarer lyfter fram spelets upplägg, vilket utgörs av lärande i form av ett datorspel, som en motivationshöjande och även underhållande faktor. Visuell återkoppling vid genomförandet av ett programmeringsrelaterat problem ansågs även vara mycket uppskattat av de flesta testpersonerna, enligt de kommentarer som kan ses i Bilaga A. Emblemsystemet var även uppskattat bland testpersonerna, vilka nämner att samlande av emblem upplevdes som motiverande. En testperson nämner till exempel: ”...ville spela om banor för att låsa upp alla achievements...”. Återkoppling och emblemsystem får därmed anses uppfylla spelets motivationshöjande syfte. Detta är något som även Kazimoglu et al. (2012) identifierar som viktiga typer av spelmekanik för att engagera och motivera studenter genom ett seriöst spel. Enkätundersökningen som genomfördes i detta projekt visar att detta påstående tycks gälla i praktiken. Detta resultat kan förklaras av att yttre belöningar i allmänhet är motivationshöjande (Hubackova & Semradova, 2014).

Enkätundersökningen visar även att spelets programmeringsrelaterade problem upplevdes som krävande av de testpersoner som inte hade tidigare erfarenhet inom programmering. Endast en testperson ansåg att svårighetsgraden i spelet var för enkel: ”Kan redan tillräckligt. Mer lämpat för nybörjare. Tråkiga frågor minskade motivationen”. Ett flertal testpersoner kommenterade även frågesporten som de spelade i slutet av nivån. Somliga testpersoner upplevde att frågesporten gav återkoppling på hur kursinnehåll har förvärvats och själva upplägget var uppskattat. Frågorna i frågesporten upplevdes dock som svåra och stundtals även tråkiga. Det kan därmed konstateras att återkoppling i form av en frågesport både lovordades och kritiserades av testpersonerna. Med anledning av detta bör frågesporten i spelet granskas och kompletteras vid framtida utveckling av spelet. Detta kan genomföras genom att införa fler belönande typer av spelmekanik såsom ett nivåsystem eller ett poängsystem, vilka exempelvis kan ge studenter yttre belöningar i form av bonuspoäng till skrivningar. Ett annat alternativ är att implementera en topplista i anslutning till frågesporten, där studenterna kan jämföra sina resultat med övriga studenter. Denna tävlande spelmekanik har nyttjats i anslutning till en frågesport av Cheong et al. (2013) och har resulterat i positiv respons av studenterna. Det kan därmed vara värt att implementera denna spelmekanik vid framtida utveckling i spelet. Testpersonernas kommentarer tyder även på en spridning gällande upplevd svårighetsgrad i spelet. Utformningen av problemen bör därmed omprövas och anpassas tydligare för målgruppen.

Testpersonernas svar på frågan ”Kan du tänka dig att använda spelet som ett hjälpmedel vid egenstudier av objektorienterad programmering?” gav blandade resultat. Många testpersoner upplevde att spelet kunde användas som ett hjälpmedel vid egenstudier och nämnde även under ”Övriga kommentarer om spelet” att frågesporten var ett uppskattat inslag i spelet. Samtidigt ställde sig många testpersoner tveksamma till spelets användning som ett hjälpmedel vid egenstudier. En testperson menade att: ”Vissa frågor i quizet var ganska kluriga. Tror att jag hade behövt läsa in mig mer på programmering för att klara quizet. Det hade varit bra med någon slags hänvisning till en kursbok eller sida där man kan läsa mer om programmering”. Flera testpersoner menar även att fler nivåer och kunskapsmoment i spelet är önskvärda att implementera framöver. Svårighetsgraden i frågesporten nämns av många testpersoner, något som tyder på att dessa frågor upplevdes som krävande. Dessutom upplevde somliga testpersoner att frågesporten var tråkig. Detta resultat skiljer sig från de slutsatser som drogs av Cheong et al. (2013) i deras artikel om frågesporten ”Quick Quiz”, där många studenter upplevde att frågesporten var ett uppskattat inslag. Vi anser med anledning av detta att frågesporten bör omarbetas vid en framtida utveckling av spelet och kompletteras med motiverande spelmekanik. En tydligare koppling mellan spelet och relevant kurslitteratur hade även kunnat underlätta testpersonernas genomförande av frågesporten. Vi observerade även att endast en av testpersonerna valde att nyttja hjälpmedel vid genomförandet av spelet, trots att detta uppmuntrades i enkätundersökningen. Det kan därmed konstateras att hänvisning till relevant kurslitteratur eller annat material bör förtydligas i spelet. Vi anser även att det finns potential att introducera användaren till populära och etablerade programmeringsforum där användaren kan ställa sina egna frågor och interagera med andra programmerare.

Det bör slutligen nämnas att det inte går att dra generella slutsatser om spelet och dess inverkan på den valda målgruppen, något som beror på att utförliga tester inte genomfördes. Detta var dock något som inte var möjligt att genomföra, med anledning av projektets tidsbegränsningar. Enkätundersökningen bör kompletteras för att förtydliga testpersonernas åsikter om spelet. En kommentar bland testpersonerna var bland annat ”Kul med achievements som man kan samla på. Problemen var lagom svåra och det var roligt att klara av dem”. Även om denna kommentar ger viktig respons på hur spelet upplevdes behövs detta utvecklas med olika kvalitativa metoder, såsom en strukturerad intervju med testpersonen. Dessa metoder bör nyttjas med anledning av att användarens åsikter är centrala vid utvärderingen av spelet. Vi hävdar därmed att den vetenskapliga metod som valdes i delkapitel 2.1 inte är lämplig för att dra generella slutsatser om spelets inverkan på studenter. Förvisso användes kvalitativa frågor i enkätundersökningen, men vi anser att en kombination av enkätundersökning och intervjuer kan ge en mer detaljerad analys av spelets inverkan på programmeringsstudenter. Den metodtriangulering som beskrivs i delkapitel 2.1 bör därför kompletteras med fler kvalitativa ansatser. Det måste också anses att den urvalsram som valdes till enkätundersökningen inte utgör en representativ bild av populationen och därmed krävs utförligare undersökningar för att få ett representativt resultat. Testpersonerna till enkätundersökningen valdes

huvudsakligen baserat på vilken erfarenhet dessa hade inom ämnet objektorienterad programmering. Vi ställer oss dock kritiska till detta urval vad gäller generella slutsatser som kan dras. Den enkätundersökning som har genomförts i detta projekt ger dock en uppfattning om den valda målgruppens intryck av spelet. Testpersonernas kommentarer i kombination med upplevd motivationshöjning gav värdefull respons, vilken är värd att ta hänsyn till vid en eventuell framtida utveckling av spelet.

7.2. Arbetsmetod

Genomförandet av detta projekt har präglats av kontinuerlig återkoppling under arbetsprocessen, eftersom nivåernas funktionalitet i spelet har krävt gemensamma analyser och diskussioner i gruppen. Arbetsmetoden Scrum har varit en effektiv arbetsmetod med detta i avseende. Dagliga scrummöten har möjliggjort kontinuerlig återkoppling bland gruppmedlemmarna gällande arbetets utveckling och eventuella problem har uppmärksammats i samband med dessa möten. Denna arbetsmetod måste dock kritiseras eftersom spelets funktionalitet utvecklades parallellt med formgivning av nivåerna i spelet. Formgivningen av nivåerna försvårades av Scrum, eftersom en tidsbegränsning fanns för varje del i utvecklingen. Vidare krävde formgivningen av nivåerna diskussioner om vilka grafiska resurser som skulle nyttjas, vilka programmeringsrelaterade problem som skulle ingå och hur emblemsystem och återkoppling skulle användas för varje nivå. Denna process kunde inte genomföras iterativt på samma sätt som spelets funktionalitet, vilket medförde att utformningen av nivåerna planerades separat. Detta berodde framförallt på att nivåerna skulle få en genomtänkt helhetsbild och välformulerade programmeringsrelaterade problem, i enlighet med den analys som gjordes i delkapitel 4.2. Spelets funktionalitet kunde därmed med hjälp av Scrum genomföras iterativt och analyseras under en sprint med få undantag, medan spelnivåernas utformning krävde mer tid och tydligare planering för att nå ett slutmål.

7.3. Spelet

Inledningsvis är det värt att poängtera att spelet är inspirerat av spelifiering och vi har valt att betona vissa varianter av spelmekanik som har en motivationshöjande effekt vid utvecklingen av spelet. Återkoppling och emblemsystem är centrala för spelifiering och används i motivationshöjande syfte, något som konstateras av Fitz-Walter et al. (2011 och Cheong et al. (2013). Dessa varianter av spelmekanik betonades under utvecklingen av spelet och uppskattades generellt av många testpersoner, men upplevdes även som motivationshöjande. Med anledning av att många testpersoner nämnde att återkoppling och emblemsystem var motivationshöjande får det anses att detta stämmer överens med det som påstås av Domínguez et al. (2013). Det nämns av Domínguez et al. (2013) att återkoppling och belöningssystem såsom emblemsystem är viktigt för att motivera studenter, något som även vi har kunnat konstatera i vår utvärdering av spelet.

Som tidigare nämnts i delkapitel 6.2 innehåller spelet följande kunskapsmoment: tilldelningar, datatyper, villkorliga satser och slingor. Fler kunskapsmoment planerades dock att implementeras i spelet, enligt delkapitel 4.1, vilka var: fält, arv och klasser. Dessa kunskapsmoment valdes dock att uteslutas från spelet, då vi ansåg att dessa kunskapsmoment skulle vara för tidskrävande att implementera. Detta konstaterades

huvudsakligen med anledning av att fler nivåer skulle behövt utformas i kombination med att fler frågor hade behövts formulerats till frågesporten. Dessutom ansåg vi att det fanns ett värde i att utforma genomtänkta nivåer av hög kvalitet istället för att fokusera på att spelet skulle omfatta så många kunskapsmoment som möjligt. Genomtänkta och väl formulerade problem är dessutom något som är centralt inom problembaserat lärande (Wood, 2003), vilket inspirerade utformningen av nivåerna i spelet. Det är dock värt att konstatera att de kunskapsmoment som utelämnades ur spelet är centrala inom objektorienterad programmering och bör därav inkluderas vid vidareutveckling av spelet.

De programmeringsrelaterade problem som innefattas i spelet är litet i antal jämfört med mängden problem som kan uppstå i en verklig situation som programmerare. Detta innebär att fler programmeringsrelaterade problem bör implementeras vid vidareutveckling av spelet. Samtidigt bör de redan existerande problemen i spelet utvärderas och eventuellt modifierats för att på så sätt göra spelet mer engagerande. Därtill bör programmeringsrelaterade problem med högre komplexitet implementeras. Detta hade dock varit svårt att genomföra i spelets nuvarande tillstånd beroende på vilken typ av problem som ska skapas. Detta beror på att spelet inte har funktionalitet för att stödja mer avancerade problem, såsom att låta användaren skapa egna virtuella objekt i spelvärlden eller ge användaren tillgång till alla objekt i spelet.

Vi uppmuntrar att spelet spelas i grupper av två studenter och att studenterna därmed nyttjar metoden ”pair programming”. Effekten av metoden ”pair programming” är dock något som inte har studerats i detta projekt. En observation gjordes dock som visade att somliga moment i spelet upplevdes som svåra av testpersonerna, såsom frågesporten och somliga programmeringsrelaterade problem. Vi anser att dessa moment hade kunnat underlättats med hjälp av ”pair programming”. Därmed bör ett flerspelarläge implementeras vid en fortsatt utveckling av spelet, där användarna bistår varandra för att på så sätt klara av de svåraste problemen i spelet. Det bör därtill genomföras fler undersökningar på hur denna metod påverkar studenternas upplevelse av spelet.

Kodmiljö
Kodmiljöns implementation i nuläget gör det möjligt för användaren att injicera potentiellt skadlig kod in i spelet. Åtgärder för att förhindra tillgång till spelets egna kod nämndes i delkapitel 5.4, men detta skydd är långt från fullständigt. Den nämnda separationen av klasser kan lätt förbises då användaren har tillgång till spelets *klassladdare*, vilket är ett objekt som har åtkomst till all funktionalitet i spelet. Åtkomsten till spelets klassladdare kan dock begränsas med hjälp av säkerhetsregler som avbryter användarens kod. Vidare kan säkerhetsreglerna också se till att användaren inte försöker använda sig av *reflektion* för att komma åt egenskaper och funktioner som denna egentligen inte har tillgång till. Användning av reflektion i Javakod gör det möjligt för programmeraren att bland annat komma åt fält, metoder, klasser och paket under körtid som inte finns tillgängligt under kompilering och är därmed ett mycket kraftfullt verktyg. Säkerhetsåtgärderna som finns implementerade i spelet berör inte reflektion och därmed är detta enbart en olägenhet för en

användare som redan är bekant med detta begrepp.

Önskemålet om en fullständigt skyddad miljö anser vi dock vara meningslös då användare med hög kunskapsnivå kan ändra spelets funktionalitet redan innan det startas. Detta betyder att ändringar görs innan säkerhetsreglerna har tillämpats. Med detta i åtanke bör syftet med dessa säkerhetsåtgärder istället rikta in sig på att minska risken för missförstånd hos användaren till följd av oavsiktligt avslöjande av bakomliggande kod. Enligt detta tankesätt uppfyller de redan implementerade åtgärderna sitt syfte.

8. Slutsats

Projektet resulterade i ett seriöst spel med inspiration från begreppet spelifiering. Spelet innehåller kunskapsmomenten villkorliga satser, tilldelningar, datatyper och slingor. Ytterligare kunskapsmoment var planerade att inkluderas i spelet, men med anledning av att nivåskapandet krävde mer tid än vad som hade förväntats kunde detta inte genomföras. Det finns dock goda förutsättningar att inkludera dessa moment vid en framtida utveckling av spelet, eftersom spelfunktionalitet för dessa moment finns.

Resultaten av enkätundersökningen visar att nyttjandet av, de för spelifiering centrala typerna av spelmekanik, emblemsystem och återkoppling upplevs som mycket positiva av den valda målgruppen. De programmeringsrelaterade problem, vilka inspirerades av problembaserat lärande, anses vara krävande men uppskattade av testpersonerna. Enkätundersökningen visar att en ökad motivation till att studera objektorienterad programmering uppvisades av en majoritet av testpersonerna. Spelet anses även av testpersonerna ha potential att kunna nyttjas som ett hjälpmedel vid egenstudier. Enkätundersökningen visar dock att ett flertal moment i spelet, såsom frågesporten, bör förbättras vid en framtida utveckling av spelet.

Slutligen kan det konstateras att den visuella återkopplingen vid problemlösning relaterat till objektorienterad programmering i kombination med spelmekaniken emblemsystem är en väl fungerande metod för att öka motivationen hos studenter. En omarbetning av frågesporten och införandet av fler motivationshöjande spelmekaniker är värt att beakta vid en framtida utveckling av spelet. Utförliga tester med enkätundersökningar och intervjuer är nödvändiga för att dra ytterligare slutsatser om spelets inverkan på studenter inom objektorienterad programmering.

Källförteckning

Arnab, S., Lim, T., Carvalho, M. B., Bellotti, F., Freitas, S., Louchart, S., Suttie, N., Berta, R., & De Gloria, A. (2014). Mapping learning and game mechanics for serious games analysis. *British Journal of Educational Technology*, 46(2), 391–411. http://dx.doi.org/10.1111/bjet.12113.

Arnold, B. J. (2014). Gamification in education. volume 21, page 32, San Diego. American Society of Business and Behavioral Sciences. Hämtad 2015-04-23, från `http://search.proquest.com/docview/1519057772?pq-origsite=summon`.

Badlogic Games (2015). *LibGDX Developer's Guide*. Hämtad 2015-04-24, från `https://github.com/libgdx/libgdx/wiki`.

Beck, K. (2000). *Extreme programming explained: embrace change*. Addison-Wesley Professional.

Biju, S. M. (2013). Difficulties in understanding object oriented programming concepts. I *Innovations and Advances in Computer, Information, Systems Sciences, and Engineering*, pages 319–326. Springer. http://dx.doi.org/10.1007/978-1-4614-3535-8_27.

Buckley, P. & Doyle, E. (In press). Gamification and student motivation. *Interactive Learning Environments*. http://dx.doi.org/10.1080/10494820.2014.964263.

Budgie, S. (2015). Glitchspace – a reprogrammable world. Hämtad 2015-04-20, från `http://www.glitchspace.com/`.

Bunchball (2010). Gamification 101: An introduction to the use of game dynamics to influence behavior. *White paper*. Hämtad 2015-04-10, från `http://www.asaecenter.org/files/HealthcareAssnConf/2011HAC/gamification101.pdf`.

Chen, S. L. & Michael, D. R. (2005). *Serious Games: Games That Educate, Train, and Inform*. Course Technology, Incorporated.

Cheong, C., Cheong, F., & Filippou, J. (2013). Quick quiz: A gamified approach for enhancing learning. I *Pacific Asia Conference on Information Systems*, page 206. Hämtad 2015-05-12, från `http://aisel.aisnet.org/cgi/viewcontent.cgi?article=1206&context=pacis2013`.

Connolly, T. M., Boyle, E. A., MacArthur, E., Hainey, T., & Boyle, J. M. (2012). A systematic literature review of empirical evidence on computer games and serious games. *Computers and Education*, 59(2), 661–686. http://dx.doi.org/10.1016/j.compedu.2012.03.004.

Cooper, S., Dann, W., & Pausch, R. (2003). Teaching objects-first in introductory computer science. *SIGCSE Bulletin*, 35(1), 191–195. http://dx.doi.org/10.1145/792548.611966.

Corral, J., Balcells, A., Estevez, A., Moreno, G., & Ramos, M. (2014). A game-based approach to the teaching of object-oriented programming languages. *Computers & Education*, 73, 83–92. http://dx.doi.org/10.1016/j.compedu.2013.12.013.

Cowan, B. & Kapralos, B. (2011). A simplified level editor. I *Games Innovation Conference (IGIC), 2011 IEEE International*, pages 52–54. IEEE. http://dx.doi.org/10.1109/IGIC.2011.6115130.

De Prato, G., Feijóo, C., & Simon, J.-P. (2014). Innovations in the video game industry: Changing global markets. *Communications & Strategies*. Hämtad 2015-04-12, från `http://search.proquest.com/docview/1545557204?pq-origsite=summon`.

Deary, I. J. (2013). Intelligence. *Current Biology*, 23(16), R673–R676. http://dx.doi.org/10.1016/j.cub.2013.07.021.

Deci, E. L., Koestner, R., & Ryan, R. M. (2001). Extrinsic rewards and intrinsic motivation in education: Reconsidered once again. *Review of Educational Research*, 71. http://dx.doi.org/10.3102/00346543071001001.

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: defining gamification. I *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, MindTrek '11, pages 9–15. ACM. http://dx.doi.org/10.1145/2181037.2181040.

Domínguez, A., Saenz-De-Navarrete, J., De-Marcos, L., Fernández-Sanz, L., Pagés, C., & Martínez-Herráiz, J.-J. (2013). Gamifying learning experiences: Practical implications and outcomes. *Computers and Education*, 63, 380–392. http://dx.doi.org/10.1016/j.compedu.2012.12.020.

Dybå, T. & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9), 833–859. http://dx.doi.org/10.1016/j.infsof.2008.01.006.

Ehlert, A. & Schulte, C. (2009). Empirical comparison of objects-first and objects-later. I *Proceedings of the fifth international workshop on Computing education research workshop*, pages 15–26. ACM. http://dx.doi.org/10.1145/1584322.1584326.

Eliasson, A. (2013). *Kvantitativ metod från början*. Studentlitteratur.

Esoteric Software (2015). *What is Spine?* Hämtad 2015-05-05, från `http://esotericsoftware.com/spine-in-depth`.

Fitz-Walter, Z., Tjondronegoro, D., & Wyeth, P. (2011). Orientation passport: using gamification to engage university students. I *Proceedings of the 23rd Australian Computer-Human Interaction Conference*, pages 122–125. ACM. http://dx.doi.org/10.1145/2071536.2071554.

GitHub (2015). *GitHub Security*. Hämtad 2015-05-09, från `https://help.github.com/articles/github-security/`.

Hakulinen, L. (2011). Using serious games in computer science education. pages 83–88. ACM. http://dx.doi.org/10.1145/2094131.2094147.

Hamari, J., Koivisto, J., & Sarsa, H. (2014). Does gamification work? – a literature review of empirical studies on gamification. pages 3025–3034. IEEE. http://dx.doi.org/10.1109/HICSS.2014.377.

Hubackova, S. & Semradova, I. (2014). Research study on motivation in adult education. *Procedia – Social and Behavioral Sciences*, 159, 396–400. http://dx.doi.org/10.1016/j.sbspro.2014.12.395.

Hunicke, R., LeBlanc, M., & Zubek, R. (2004). Mda: A formal approach to game design and game research. I *Proceedings of the AAAI Workshop on Challenges in Game AI*, volume 4. Hämtad 2015-04-15, från `http://www.cs.northwestern.edu/~hunicke/MDA.pdf`.

Jayasinghe, U. & Dharmaratne, A. (2013). Game based learning vs. gamification from the higher education students' perspective. pages 683–688. IEEE. http://dx.doi.org/10.1109/TALE.2013.6654524.

Jerinic, L. (2014). Teaching introductory programming. *International journal of advanced computer science & applications*, 5, 60–69. http://dx.doi.org/10.14569/IJACSA.2014.050611.

Johnson, S. (2008). 2d vs 3d – choosing the right camera system for your game. Hämtad 2015-04-20, från `http://search.proquest.com/docview/219049334?pq-origsite=summon`.

JSON (2015). *Introduktion till JSON*. Hämtad 2015-05-05, från `http://www.json.org/json-sv.html`.

Kapp, K. (2012). *The Gamification of Learning and Instruction: Case-Based Methods and Strategies for Training and Education*. Pfeiffer.

Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2012). A serious game for developing computational thinking and learning introductory computer programming. *Procedia-Social and Behavioral Sciences*, 47, 1991–1999. http://dx.doi.org/10.1016/j.sbspro.2012.06.938.

Kelly, H., Howell, K., Glinert, E., Holding, L., Swain, C., Burrowbridge, A., & Roper, M. (2007). How to build serious games. *Communications of the ACM*, 50(7), 44–49. http://dx.doi.org/10.1145/1272516.1272538.

Kiili, K. (2005). Digital game-based learning: Towards an experiential gaming model. *The Internet and higher education*, 8(1), 13–24. http://dx.doi.org/10.1016/j.iheduc.2004.12.001.

Kilroy, D. (2004). Problem based learning. *Emergency medicine journal*, 21(4), 411–413. http://dx.doi.org/10.1136/emj.2003.012435.

Kim, B., Park, H., & Baek, Y. (2009). Not just fun, but serious strategies: Using meta-cognitive strategies in game-based learning. *Computers & Education*, 52(4), 800–810. http://dx.doi.org/10.1016/j.compedu.2008.12.004.

Koster, R. (2013). *Theory of fun for game design*. O'Reilly Media, Inc.

Kölling, M. (1999). The problem of teaching object-oriented programming. *Journal of Object Oriented Programming*, 11(8), 8–15. Hämtad 2015-05-04, från `http://www.u-helmich.de/inf/BlueJ/oop1.pdf`.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. I *ACM SIGCSE Bulletin*, volume 37, pages 14–18. ACM. http://dx.doi.org/10.1145/1151954.1067453.

Landers, R. N. (2015). Developing a theory of gamified learning linking serious games and gamification of learning. *Simulation & Gaming*, pages 752–768. http://dx.doi.org/10.1177/1046878114563660.

Leutenegger, S. & Edgington, J. (2007). A games first approach to teaching introductory programming. *ACM SIGCSE Bulletin*, 39(1), 115–118. http://dx.doi.org/10.1145/1227504.1227352.

Liu, M., Rosenblum, J. A., Horton, L., & Kang, J. (2014). Designing science learning with game-based approaches. *Computers in the Schools*, 31(1-2), 84–102. http://dx.doi.org/10.1080/07380569.2014.879776.

Loeliger, J. (2009). *Version control with Git*. O'Reilly, Beijing. Hämtad 2015-05-05, från `https://www.dawsonera.com/readonline/9780596551032`.

Luminea, C. (2013). *Gamification*. Financial management. Hämtad 2015-02-03, från `http://search.proquest.com/docview/1321146653?pq-origsite=summon`.

Milne, I. & Rowe, G. (2002). Difficulties in learning and teaching programming—views of students and tutors. *Education and Information technologies*, 7(1), 55–66. http://dx.doi.org/10.1023/A:1015362608943.

Mow, I. C. (2008). *Issues and Difficulties in Teaching Novice Computer Programming*. Springer Netherlands. http://dx.doi.org/10.1007/978-1-4020-8739-4_36.

mrwynd (2009). Left 4 dead friends list leaderboards. Hämtad 2015-06-02, från `https://www.flickr.com/photos/mrwynd/3474664249/`.

Mukker, A., Singh, L., & Mishra, A. K. (2014). Systematic review of metrics in software agile projects. *Compusoft: International Journal of Advanced Computer Technology*, 3, 533–539. Hämtad 2015-04-14, från `http://ijact.in/systematic-review-of-metrics-in-software-agile-projects/`.

Muntean, C. I. (2011). Raising engagement in e-learning through gamification. I *Proc. 6th International Conference on Virtual Learning ICVL*, pages 323–329. Hämtad 2015-05-13, från `http://icvl.eu/2011/disc/icvl/documente/pdf/met/ICVL_ModelsAndMethodologies_paper42.pdf`.

Muratet, M., Torguet, P., Jessel, J.-P., & Viallet, F. (2009). Towards a serious game to help students learn computer programming. *International Journal of Computer Games Technology*, 2009, 3. http://dx.doi.org/10.1155/2009/470590.

Murayama, K., Pekrun, R., Lichtenfeld, S., & Vom Hofe, R. (2013). Predicting long-term growth in students' mathematics achievement: The unique contributions of motivation and cognitive strategies. *Child development*, 84(4), 1475–1490. http://dx.doi.org/10.1111/cdev.12036.

Nurseitov, N., Paulson, M., Reynolds, R., & Izurieta, C. (2009). Comparison of json and xml data interchange formats: A case study. *Caine*, 2009, 157–162. Hämtad 2015-05-04, från `http://www.cs.montana.edu/izurieta/pubs/caine2009.pdf`.

Nuutila, E., Törmä, S., & Malmi, L. (2005). Pbl and computer programming—the seven steps method with adaptations. *Computer Science Education*, 15(2), 123–142. Hämtad 2015-05-13, från `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.469.6559&rep=rep1&type=pdf`.

Oracle (2015). *Java Documentation*. Hämtad 2015-05-07, från `https://docs.oracle.com/javase/tutorial/java/`.

Paasivaara, M., Durasiewicz, S., & Lassenius, C. (2009). Using scrum in distributed agile development: A multiple case study. I *Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on*, pages 195–204. IEEE. http://dx.doi.org/10.1109/ICGSE.2009.27.

Pelling, N. (2011). The (short) prehistory of gamification. *Funding Startups (& other impossibilities). Haettu*, 7, 2013. Hämtad 2015-03-15, från `https://nanodome.wordpress.com/2011/08/09/the-short-prehistory-of-gamification/`.

Ren, Z. (2015). Backgrounds for 2d platformers. Hämtad 2015-03-19, från `http://opengameart.org/content/backgrounds-for-2d-platformers`.

Rodrigo, M. M. T. & Baker, R. (2011). Comparing learners' affect while using an intelligent tutor and an educational game. *Research and Practice in Technology Enhanced Learning*, 6, 43–66. http://dx.doi.org/10.1007/978-3-540-69132-7_9.

Rosander, P., Bäckström, M., & Stenberg, G. (2011). Personality traits and general intelligence as predictors of academic performance: A structural equation modelling approach. *Learning and Individual Differences*, 21(5), 590–596. http://dx.doi.org/10.1016/j.lindif.2011.04.004.

Salleh, N., Mendes, E., & Grundy, J. (2011). Empirical studies of pair programming for cs/se teaching in higher education: A systematic literature review. *IEEE Transactions on Software Engineering*, 37(4), 509–525. http://dx.doi.org/10.1109/TSE.2010.59.

Sawyer, B. (2007). The "serious games" landscape. I *The Instructional & Research Technology Symposium for Arts, Humanities and Social Sciences, Camden, USA*. Hämtad 2015-04-20, från `internet2.rutgers.edu/pres/speaker6-sawyer-final.ppt`.

Schwaber, K. & Beedle, M. (2002). *Agile software development with Scrum*. Pearson Education International.

Silicon Graphics (2014). *OpenGL Wiki*. Hämtad 2015-05-05, från `https://www.opengl.org/wiki/Main_Page`.

Singer, L. & Schneider, K. (2012). It was a bit of a race: Gamification of version control. I *Games and Software Engineering (GAS), 2012 2nd International Workshop on*, pages 5–8. IEEE. http://dx.doi.org/10.1109/GAS.2012.6225927.

Skansholm, J. (2013). *Java direkt med Swing*. Studentlitteratur, Lund.

Smith, R. (2010). The long history of gaming in military training. *Simulation & Gaming*, 41(1), 6–19. http://dx.doi.org/10.1177/1046878109334330.

Sternberg, R. (2012). Intelligence. *Wiley Interdisciplinary Reviews: Cognitive Science*, 3. http://dx.doi.org/10.1002/wcs.1193.

Tan, P. H., Ting, C. Y., Ling, S. W., & Society, I. C. (2009). Learning difficulties in programming courses: Undergraduates' perspective and perception. *Proceedings of the 2009 International Conference on Computer Technology and Development, Vol 1*, pages 42–46. http://dx.doi.org/10.1109/icctd.2009.188.

Technology News Focus (2012). Computers, software; bunchball launches first personalized gamification solution. Hämtad 2015-03-26, från `http://search.proquest.com.proxy.lib.chalmers.se/docview/929860004?OpenUrlRefId=info:xri/sid:summon`.

Uysal, M. P. et al. (2012). The effects of objects-first and objects-late methods on achievements of oop learners. *Journal of Software Engineering and Applications*, 5(10), 816. http://dx.doi.org/10.4236/jsea.2012.510094.

Valve Corporation (2012). Counter-strike: Global offensive. [PC game].

Vihavainen, A., Paksula, M., & Luukkainen, M. (2011). Extreme apprenticeship method in teaching programming for beginners. I *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, pages 93–98, New York, NY, USA. ACM. http://dx.doi.org/10.1145/1953163.1953196.

Waffle (2015). *Work better on GitHub issues*. Hämtad 2015-05-05, från `https://waffle.io/`.

Werbach, K. & Hunter, D. (2012). *For the win: How game thinking can revolutionize your business*. Wharton Digital Press.

Williams, L. & Cockburn, A. (2003). Agile software development: it's about feedback and change. *Computer*, 36, 39–43. http://dx.doi.org/10.1109/MC.2003.1204373.

Wood, D. F. (2003). Problem based learning. *BMJ*, 326(7384), 328–330. http://dx.doi.org/10.1136/bmj.326.7384.328.

Wouters, P., van Nimwegen, C., van Oostendorp, H., & van Der Spek, E. D. (2013). A meta-analysis of the cognitive and motivational effects of serious games. *Journal of educational psychology*, 105(2), 249–265. http://dx.doi.org/10.1037/a0031311.

Zackariasson, P. & Wilson, T. L. (2010). Paradigm shifts in the video game industry. *Competitiveness Review*, 20(2), 139–151. http://dx.doi.org/10.1108/10595421011029857.

Zichermann, G. & Linder, J. (2013). *The gamification revolution: how leaders leverage game mechanics to crush the competition.* McGraw-Hill Education, New York.

Bilaga A: Enkätundersökning

A.1. Enkät

Del 1

Tack för att du deltar! Börja med att besvara frågorna under del 1. Efter slutförande av speldemonstrationen kan du påbörja del 2 av enkätundersökningen. Notera att speldemonstrationen inte är ett test och därmed är alla hjälpmedel tillåtna.

*Obligatorisk

Kön? *

☐ Man
☐ Kvinna
☐ Vill inte svara

Hur gammal är du? *

☐ 16-20
☐ 21-25
☐ 26-35
☐ 36 eller äldre

Hur mycket tidigare erfarenhet har du av objektorienterad programmering i akademiska sammanhang? *

Med objektorienterad programmering åsyftas objektorienterade språk såsom Java, C++, Python et cetera.

☐ Ingen tidigare erfarenhet
☐ Liten erfarenhet (läst enstaka kurs inom objektorienterad programmering)
☐ Måttlig erfarenhet (läst ett flertal kurser inom objektorienterad programmering)
☐ Stor erfarenhet (avlagt examen med inriktning mot programmering)

Hur upplever du din motivation till att studera objektorienterad programmering? *

Svarsalternativen representerar din subjektiva åsikt.

☐ Ingen motivation
☐ Liten motivation
☐ Måttlig motivation
☐ Stor motivation

Del 2

Upplever du att spelet ökade din motivation till studier av objektorienterad programmering? *

☐ Ja
☐ Nej
☐ Vet ej

Om du har svarat "Ja" på frågan ovan, beskriv vilka delar av spelet som ökade din motivation:

Kan du tänka dig att använda spelet som ett hjälpmedel vid egenstudier av objektorienterad programmering? *

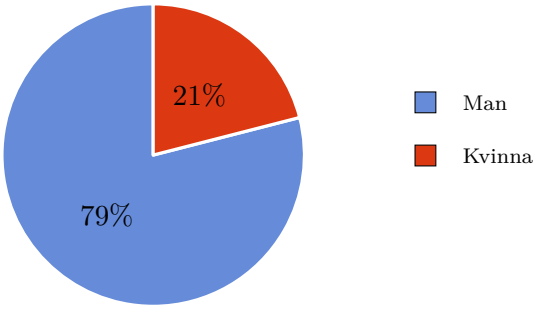
☐ Ja
☐ Nej
☐ Vet ej

Övriga kommentarer om spelet:
Här får du gärna förmedla dina egna åsikter om spelet.

A.2. Resultat av enkätundersökning

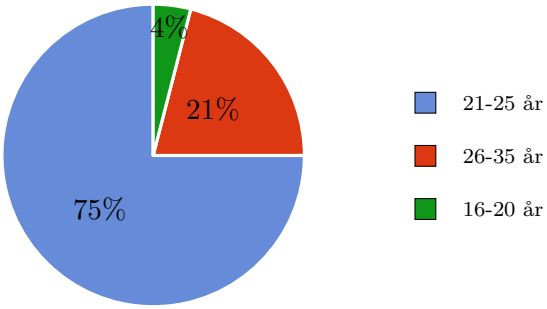
Nedan presenteras resultatet av enkätundersökningen. Totalt deltog 24 personer.

Kön?



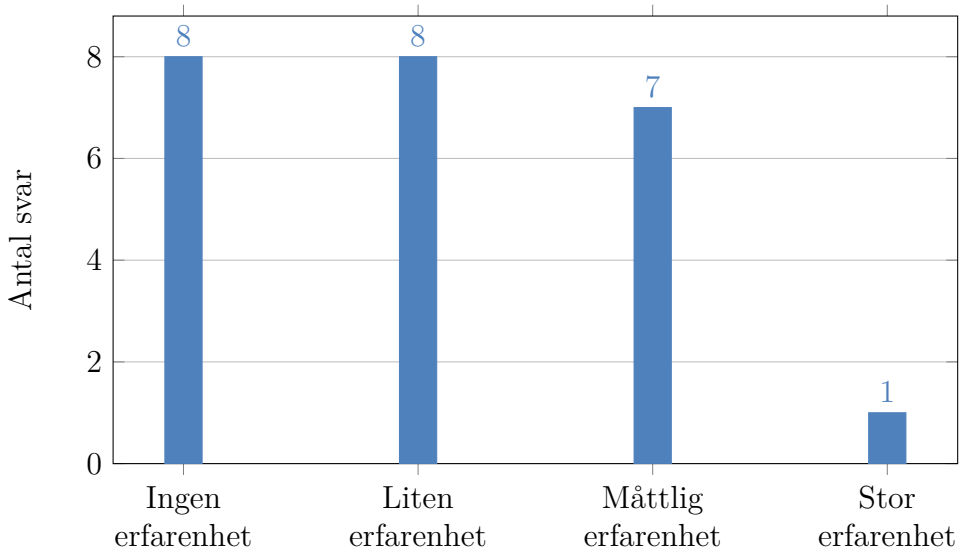
Figur A.1: Cirkeldiagrammet visar fördelning av kön hos testpersonerna i enkätundersökningen.

Hur gammal är du?



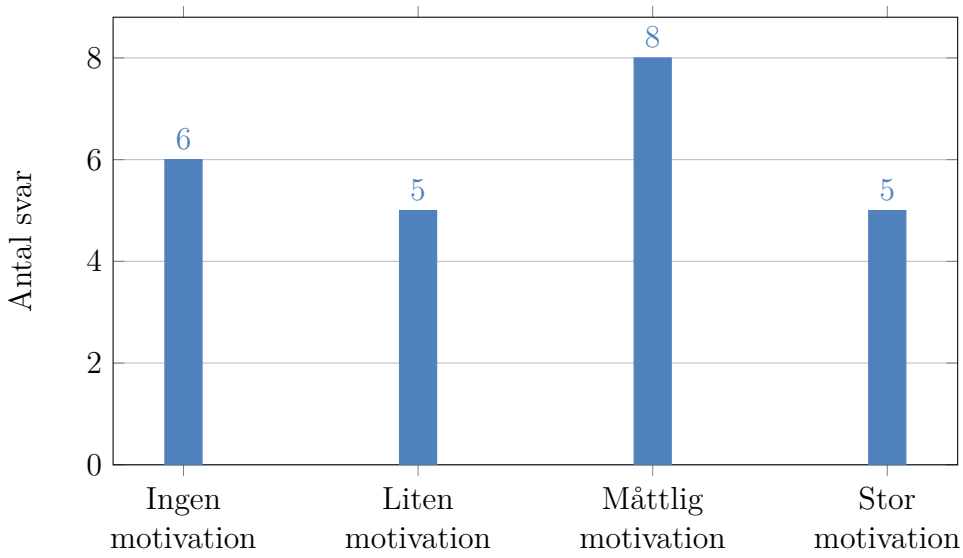
Figur A.2: Cirkeldiagrammet visar fördelning av åldersgrupper i enkätundersökningen.

Hur mycket tidigare erfarenhet har du av objektorienterad programmering i akademiska sammanhang?
Med objektorienterad programmering åsyftas objektorienterade språk såsom Java, C++, Python et cetera.



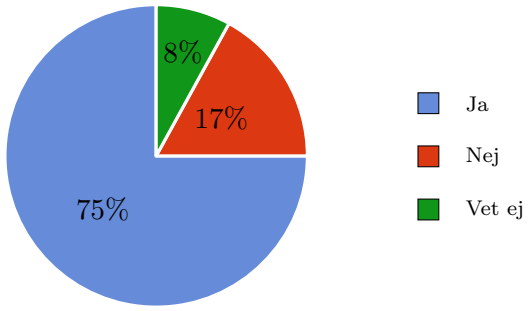
Figur A.3: Stapeldiagrammet visar fördelning av testpersonernas tidigare erfarenheter av objektorienterad programmering i akademiska sammanhang.

Hur upplever du din motivation till att studera objektorienterad programmering?
Svarsalternativen representerar din subjektiva åsikt.



Figur A.4: Stapeldiagrammet visar fördelning av testpersonernas motivation till att studera objektorienterad programmering.

Upplever du att spelet ökade din motivation till studier av objektorienterad programmering?



Figur A.5: Cirkeldiagrammet visar testpersonernas motivation till studier av objektorienterad programmering, efter genomförande av testnivå i spelet.

Om du har svarat "Ja" på frågan ovan, beskriv vilka delar av spelet som ökade din motivation:

"Det var intressant att se hur föremålen ändrar sig om man ändrar på egenskaperna! Bra med en hjälpruta så att man förstår lättare! Ville gärna spela om banorna för att låsa upp alla achievements."

"Det var kul att lösa problem med programmering."

"Roliga achievements! Det var roligt att lära sig lite om hur objekt fungerar."

"Kul med achievements som man kan samla på. Problemen var lagom svåra och det var roligt att klara av dem."

"Det var roligt att klara svåra pussel"

"Roligt att se att något händer med objekten i spelet. Vissa pussel var kluriga och det var motiverande att faktiskt klara av dom."

"Roligare än att läsa en massa böcker"

"Nivå 1 var lite för enkel. Nivå 2 var lagom svår och då blev spelet mer motiverande."

"Skönt att det faktiskt går att koppla ändringar i objektet med något visuellt. Det var motiverande att komma vidare i spelet. Roligt att få feedback på det man gör också! Såg ett tydligt resultat av det jag gjorde!"

"Roliga achievements var faktiskt bra, ville spela om banor för att låsa upp alla achievements. Lagom svårighetsgrad också."

"Spelet var ganska bra på att förklara hur objekt kan ha egenskaper. Det var ett pussel med olika if-satser som var väldigt hjälpsamt för att förklara att programmet kollar om något stämmer och sedan gör något med föremålet."

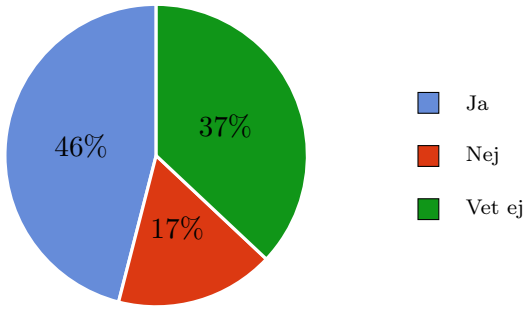
"Roligt att spela sådana spel! Roligare i spelform än att programmera."

"Jag vill också kunna göra sådana här häftiga spel!"

"Det var roligt att spela spelet och samtidigt interagerar med spelet i form av kod för att komma vidare och stöta på allt svårare problem som man behöver tänka till på."

"Gillar upplägget med programmering i ett spel!"

Kan du tänka dig att använda spelet som ett hjälpmedel vid egenstudier av objektorienterad programmering?



Figur A.6: Cirkeldiagrammet visar testpersonernas subjektiva åsikt till att använda spelet som ett hjälpmedel vid egenstudier av objektorienterad programmering.

Övriga kommentarer om spelet:
Här får du gärna förmedla dina egna åsikter om spelet.

"Vissa frågor i quizet var ganska kluriga. Tror att jag hade behövt läsa in mig mer på programmering för att klara quizet. Det hade varit bra med någon slags hänvisning till en kursbok eller sida där man kan läsa mer om programmering."

"Vissa frågorna i slutet av spelet var svåra. Jag är i och för sig helt obekant med programmering så det kanske inte var så konstigt"

"Tycker att quizet var bra för att kolla om man har förstått saker, men det var lite svåra frågor ibland. Kanske vore bra att ha fler frågor."

"Quizet var ganska bra på att kolla om man har förstått variabler, tilldelningar och sånt. Vore roligt att se fler banor!"

"Bra spel, skulle vilja ha mer story bara. Jag skulle gärna vilja att ett skott skjuts iväg när man skjuter på ett föremål. Det är lite svårt att veta vilka föremål som man kan ändra på. Quizet borde gå att skippa om man inte orkar göra det."

"Hade velat ha lite mer story, kändes som att det fanns ett tema i spelet men det förklarades aldrig."

"Gör fler banor, fixa småbuggar"

"Kan redan tillräckligt. Mer lämpat för nybörjare. Tråkiga frågor minskade motivationen"

"Spelet är inte tillräckligt djupt. Mer kunskapsmoment hade vart bra."

"Ser väldigt bra ut. Det var väldigt mycket booleans."

"Quizet kan vara bra att ha för att stämma av att man faktiskt har förstått kursinnehållet. Det är ett bra upplägg! Saknade lite större exempel på hur objekt fungerar. Men det kanske ska vara med senare i spelet i så fall."

"Roligt spel överlag, skönt med bra musik som inte stör en mitt i tänkandet."

"Ni borde förklara hur spelet fungerar lite bättre med kontroller osv. nivån på quizet motsvarar inte riktigt vad man lär sig i spelet."

"Första nivån var lite lätt, men den andra var mer klurig och bra. Vid fortsatt utveckling av spelet

så tror jag att det har potential till att bli ett bra spel för inlärning”

“Gillade att lösa pussel i spelet. Quizet kändes inte särskilt roligt.”

“Bra idé, kul med ett nytt koncept. Lite för enkelt bara.”

“Är helt ny till programmering och tyckte att quizet var lite svårt. Vissa achievements var lite enkla att få.”

“Ganska roligt spel, vet inte om jag är särskilt intresserad av programmering alls tyvärr. Tyckte att det var ett bra sätt att lära sig saker mha quizet. Sista frågan beror lite på om det finns fler banor, hade kunnat spela det i så fall och använt mig av spelet för att förstå. Lite mer avslappnat än att läsa böcker och sånt.”

Bilaga B: Kravspecifikation

B.1. Funktionella krav

Nedan presenteras spelets funktionella krav. I fältet prioritet anges, om kravet är ett *ska-krav* eller ett graderat *börkrav*.

- Ett ska-krav måste definitivt genomföras innan projektets slut.
- Ett börkrav graderas efter prioritet från (1-5), där 1 innebär låg prioritet och 5 hög prioritet.

Huvudmeny

Ref. nr.	Kravbeskrivning	Prioritet	Status
1.1	Huvudmenyn ska ha en etikett med spelets namn.	**Ska**	**Avklarad**
1.2	Huvudmenyn ska innehålla en tabell med sex stycken klickbara knappar: Nytt spel, Välj nivå, Frågesport, Prestationer, Inställningar och Avsluta spel.	**Ska**	**Avklarad**
1.3	Huvudmenyn bör ha en bakgrund av atmosfärisktyp, för att passa in i spelets tema	**5**	**Avklarad**
1.4	Knappen “Nytt spel“ bör starta en ny instans av kategorin “Spelläge“. Om ett spel redan har påbörjats av användaren, bör knappen istället ha texten “Återuppta spel“ och starta den påbörjade versionen av “Spelläge“.	**4**	**Avklarad**
1.5	Knappen “Välj nivå“ ska starta starta en ny kategori, där användaren kan välja bland existerande nivåer i spelet.	**Ska**	**Avklarad**
1.6	Knappen “Frågesport“ ska starta starta en ny kategori, där användaren kan spela frågesport.	**Ska**	**Avklarad**
1.7	Knappen “Prestationer“ ska starta starta en ny kategori, där användaren kan se Avklarade och inte Avklarade prestationer.	**Ska**	**Avklarad**
1.8	Knappen “Inställningar“ ska starta starta en ny kategori, där användaren kan ändra inställningar och tangentbindningar i spelet.	**Ska**	**Avklarad**
1.9	Knappen “Avsluta spel“ ska avsluta spelet.	**Ska**	**Avklarad**

Välj nivå

Ref. nr.	Kravbeskrivning	Prioritet	Status
2.1	Kategorin “Välj nivå“ ska ha en klickbar knapp för varje nivå i spelet, som startar kategorin “Spelläge“. Om nivån inte är Avklarad ska detta markeras med hjälp av en annan bakgrundsbild på knappen.	**Ska**	**Avklarad**
2.2	Användaren bör inte kunna starta nivåer, med undantag för nästkommande nivå, som denne inte redan har klarat av.	**5**	**Avklarad**

Spelläge

Ref. nr.	Kravbeskrivning	Prioritet	Status
3.1	Kategorin “Spelläge“ bör ha ett användargränssnitt, som visar fps, menyknapp och hjälpknapp.	**4**	**Avklarad**
3.2	Huvudkaraktären ska animeras.	**Ska**	**Avklarad**
3.3	Användaren bör få återkoppling med hjälp av ljud vid händelser av betydelsefull karaktär.	**4**	**Avklarad**
3.4	Användaren ska kunna spara och återuppta sitt spel.	**Ska**	**Avklarad**
3.5	Bakgrunden i kategorin “Spelläge“ bör ha en “parallax effekt“.	**2**	**Oavklarad**
3.6	Om användaren högerklickar på specifika objekt i spelet, ska användaren få en textruta presenterad för sig. I textrutan ska användaren kunna skriva kod för att manipulera objektet i fråga eller dess variabler.	**Ska**	**Avklarad**
3.7	Användaren ska kunna få en hjälpruta presenterad för sig. Hjälprutan ska innehålla information, som kan nyttjas för att klara av nivån.	**Ska**	**Avklarad**
3.8	En nivå bör kunna ha unik bakgrundsmusik.	**3**	**Avklarad**
3.9	Artificiell intelligens bör förekomma i spelet	**2**	**Oavklarad**
3.10	Användaren bör få visuell återkoppling då dennes markör svävar över objekt som kan manipuleras.	**3**	**Avklarad**
3.11	Huvudkaraktären ska kunna: förflytta sig, hoppa samt interagera med objekt, i spelet.	**Ska**	**Avklarad**

Frågesport

Ref. nr.	Kravbeskrivning	Prioritet	Status
4.1	Kategorin “Frågesport“ ska innehålla en frågesport för varje nivå i spelet.	**Ska**	**Avklarad**
4.2	En frågesport ska bestå av en fråga och fyra möjliga svar.	**Ska**	**Avklarad**
4.3	Användaren bör få visuell återkoppling då denne svarar på en fråga i frågesporten	**4**	**Avklarad**
4.4	Tillhörande frågesport bör automatiskt starta då en nivå avklaras.	**3**	**Avklarad**
4.5	Efter avklarad frågesport, ska användaren bli presenterad med resultatet.	**Ska**	**Avklarad**

Prestationer

Ref. nr.	Kravbeskrivning	Prioritet	Status
5.1	I kategorin “Prestationer“ ska samtliga av spelets prestationer presenteras. Oupplåsta prestationer ska ha dold beskrivning.	**Ska**	**Avklarad**
5.2	Vid upplåsning av en prestation bör en animerad ruta presenteras.	**5**	**Avklarad**

Inställningar

Ref. nr.	Kravbeskrivning	Prioritet	Status
6.1	Användaren ska kunna ändra upplösning och visningsläge i spelet.	**Ska**	**Avklarad**
6.2	Användaren bör kunna justera ljudnivåer och ljudläge i spelet.	**5**	**Avklarad**
6.3	Användaren bör kunna ändra tangentbindningar i spelet.	**4**	**Avklarad**
6.4	Användaren bör kunna ändra musbindningar i spelet.	**2**	**Oavklarad**

B.2. Icke-funktionella krav

För att ställa krav på projektets genomförande upprättades ett par kvalitetskrav: tillförlitlighet, kapacitet, effektivitet, användarvänlighet och kompabilitet. Vidare upprättades krav på vilka kunskapsmoment som spelet bör innefatta och dessa kommer presenteras nedan:

- Användaren ska ges möjligheten att mata in kod till spelet under körtid. Detta innebär att spelet måste kunna hantera *Exceptions*, hädanefter undantag, som skapas av användarens kod. Spelet ska även kunna hantera kompileringsfel och framföra felmeddelandet till användaren. För att användaren inte ska kunna modifiera spelets klasser under körtid, bör användarens kod befinna sig i en miljö avgränsad från spelets kodmiljö.
- Spelet ska enbart ha stöd för ett enspelarläge.
- Spelet ska prestera åtminstonde trettio fps vid användandet av en modern, inköpt inom det senaste året, persondator.
- Spelet ska ha en genomtänkt struktur, det vill säga att all funktionalitet ska nås via ett mindre antal klick.
- Spelet ska stödja Windows, Mac och Linux. Därtill ska spelet kräva att användaren har installerat Java Development Kit version 7.
- Spelet bör innefatta kunskapsmomenten: primitiva datatyper, tilldelningar, fält, operatorer, villkorliga satser, slingor, arv och klasser.

B.3. Definition av klart

För att underlätta utveckling av spelet har ett par kriterier upprättats. Kriterierna kommer presenteras nedan och ska ses som praxis för att underlätta utvecklingen av spelet i grupp.

- Innan kod laddas upp på GitHub, måste koden vara: körbar och testad i simulator eller accepterad enhet.
- All kod ska följa ordinära kodningskonventioner för Java.
- Alla metoder ska innehålla Javadoc-kommentarer.
- All kod ska revideras - för att undvika redundant kod.
- Betydande prestandaförluster vid nya tillägg är ej acceptabelt.

Bilaga C: JSON

C.1. Sparfil

Nedan presenteras ett exempel på hur JSON användes för sparfilen i spelet. Sparfilens uppgift är att innehålla information om användarens framfart genom spelet. För varje nivå i spelet sparas användarens senaste x- och y-koordinat, om nivån är avklarad samt antal korrekta svar på den tillhörande frågesporten. Vid fallet då användaren inte har nått någon kontrollpunkt på nivån tilldelas x- och y-koordinaten värdet -1. Utöver detta så sparar sparfilen även den senaste spelade nivån samt de prestationer som användaren har låst upp.

```
{
mostRecentLevel: level2.tmx
levels: {
        level2.tmx: {
                playerX: 30.30314
                playerY: 11.032643
                finished: false
        }
        level1e.tmx: {
                playerX: -1
                playerY: -1
                finished: true
        }
        level3.tmx: {
                playerX: -1
                playerY: -1
                finished: false
        }
}
unlockedAchievements: [
        complete_level
        secret_spot
        penny_bridge
]
quizzez: {
        level1.tmx: {
                numCorrectAnswers: 4
        }
}
}
```

C.2. Inställningar

Nedan presenteras ett exempel på hur JSON användes för att spara information om inställningarna som förekommer i spelet.

```
{
width: 800
height: 600
bitsPerPixel: 32
refreshRate: 0
fullscreen: false
vSync: true
masterVolume: 1
sfxVolume: 1
musicVolume: 1
audioEnabled: true
keyBindings: {
        Hoppa: 62
        Höger: 32
        Vänster: 29
        Kamera vänster: 21
        Kamera höger: 22
        Kamera upp: 19
        Pausa/återuppta spel: 131
        Växla kameraläge (spelare/friläge): 54
        Kamera ner: 20
        Visa/dölj hjälptext: 48
}
}
```

C.3. Frågesport

Nedan demonstreras ett exempel på hur JSON har nyttjas i samband med kategorin "Frågesport". Som tidigare nämnt, i delkapitel 5.5, har samtliga nivåer i spelet en egen frågesport och med anledning av detta identifieras frågesporterna med hjälp av namnet på nivån. Därtill har också varje frågesport ett tema, som i det här fallet är tilldelningar. Då varje nivå kan innehålla flera frågor utnyttjas ett fält, där varje element i fältet motsvarar en fråga i frågesporten.

```
{
quizzes: [
            {
        map: level1.tmx
        subject: Tilldelningar
        questions: [
                        {
                         question: Vad är A?
                         a1: 1
                         a2: 2
                         a3: 3
                         a4: 4
                         correctAnswer: a3
                        }
                    ]
            }
    ]
}
```