Projektplan SIRTET

Av: Safir Najafi, Ziad Salam & Daniel Cserhalmi

1. Programbeskrivning

DEL A:
Zombiegame, som är influerat av Namco's Pac-Man, är ett labyrintspel skrivet i Java(till desktop) som går ut på att överleva en zombieapokalyps så länge som möjligt.
Spelaren styr huvudkaraktären genom en labyrint där det ständigt dyker upp zombies som man ska undvika för att överleva.
På spelplanen dyker det upp olika bonussaker/items som förändrar spelegenskaperna, t.ex. att spelaren bli odödlig under en kort stund.
Spelet slutar när huvudkaraktären dör var efter speltiden skrivs ut, vilket motsvarar spelarens poäng.
Om tiden räcker till kommer vi även göra en version till Android.

DEL B:
Tetris är ett klassiskt spel där det på en tvådimensionell spelplan faller klossar i olika former från toppen av skärmen till botten. Klossarna faller med en viss hastighet som ökar under spelets gång. Klossarna går att rotera 90° åt höger oändligt många gånger, fyra rotationer ger alltså klossens ursprungsläge. Spelaren får poäng genom att fylla rader med klossar. När en rad fylls försvinner den och spelaren har då mer yta att lägg klossar på. Om klossarna travas upp till toppen förlorar spelaren spelet och den befintliga poängen blir spelarens slutgiltiga poäng. Spelet går alltså ut på att få så hög poäng som möjligt och tar inte slut förrän man "förlorar".

DEL C:
Istället för att skriva Zombiegame skrev vi tetris. Detta för att vi insåg att Zombiegame skulle ta lite för mycket tid och inte bli så bra som vi tänkt oss. Istället valde vi att lägga vår energi på att skriva ett bra tetris som tillfredställer förväntningarna vi har på vårt eget spel. Att skapa spelet tog dessutom bra mycket längre tid än vad vi trodde och när alla detaljer spelade in blev det ganska mycket mer arbete än förväntat. Alltså är detta tetris endast till desktop då vi inte hann konfigurera så det fungerade i Android.

2. Användarbeskrivning

DEL A:
Spelet ska vara så pass intuitivt att alla, med grundläggande datorkunskaper, ska kunna spela det. Vår målgrupp inkluderar således alla åldrar med villkoret att grundläggande datorkunskaper finns.

DEL B:
Även fast vi ändrade oss och utvecklade ett tetris liknande spel förändrades varken målgruppen eller de förutsättningar som krävs för att använda vår applikation.

DEL C
Vår målgrupp förändrades inte under projektets gång.

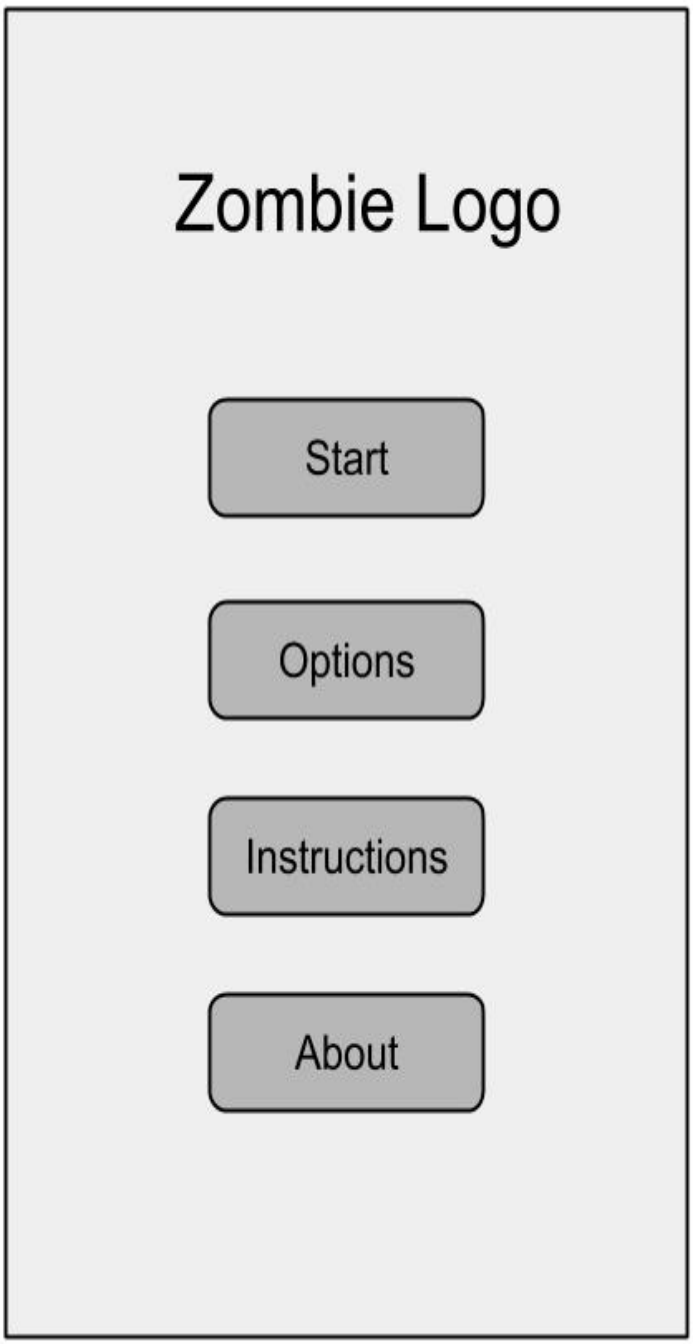
3. Användarscenarier

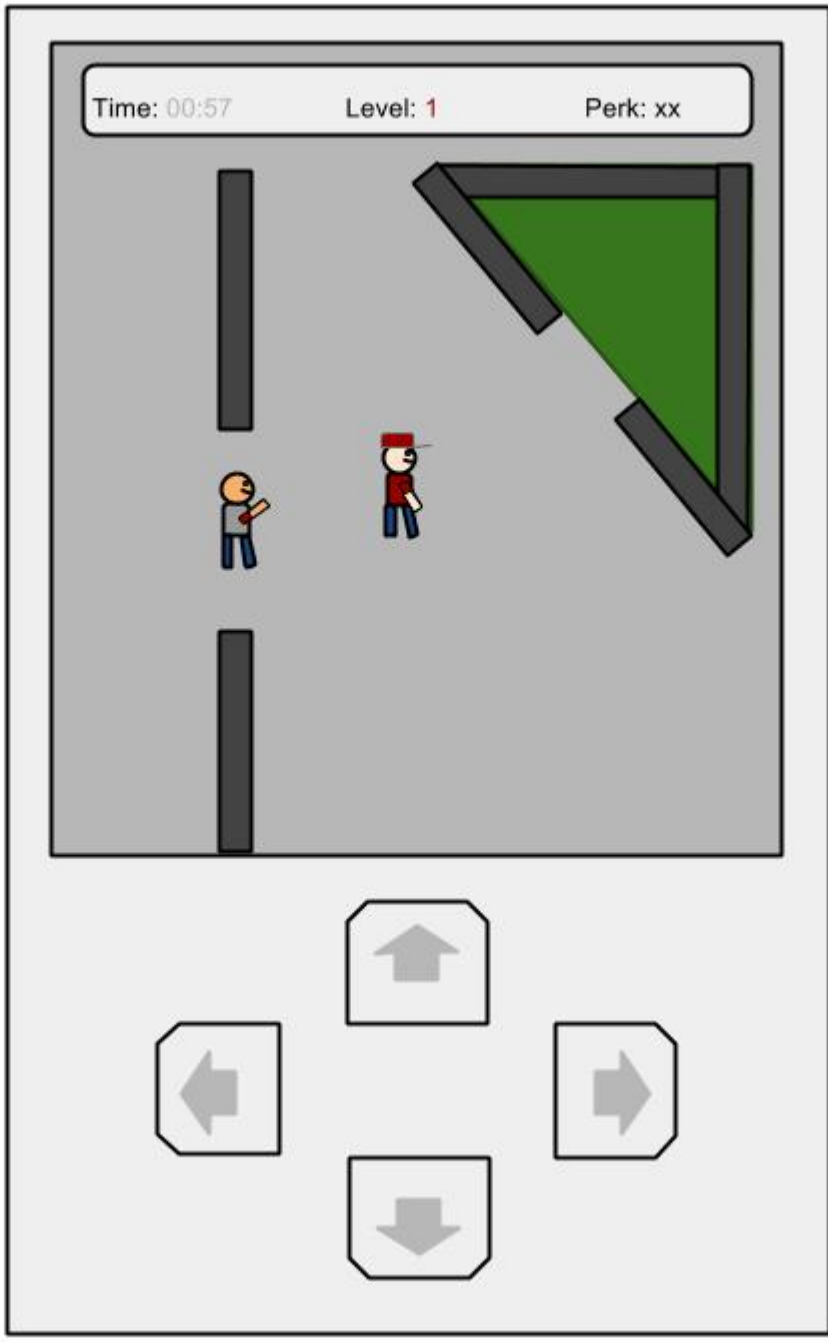
DEL A

Scenario 1(erfaren datoranvändare och datorspelare):

När användaren startar programmet möts hen av en meny(se figur nedan) som man navigerar med hjälp av musen:

Eftersom vår användare är en erfaren datorspelare trycker hen på “Start” och möts då av en ny vy där spelet startar:





I desktop varianten styr användaren huvudkaraktären med piltangenterna och försöker sedan överleva så länge som möjligt. I menubaren(en meny belägen högst upp på skärmen) finns det möjligthet att avsluta spelet under Arkiv → Avsluta(ctrl q), eller Paus/Start under Arkiv → Paus(ctrl p) resp. Arkiv → Start(ctrl p).
Spelaren kan även välja att starta om spelet och börjar då på nivå 1, Arkiv → Reset.

Scenario 2(Oerfaren datorspelare):
Till skillnad från den erfarne datorspelaren väljer hen att först läsa igenom instruktionerna(“Instructions”) som leder till en ny vy där spelregler och kontroller beskrivs. Sedan väljer hen att starta spelet vilket leder till en ny vy, själva spelvyn.

Spelaren spelar tills hen dör och kan sedan välja att starta om spelet eller avsluta(precis som i scenariot innan)

DEL B
Vi valde att köra med liknande upplägg när vi testade tetris. I vår användartestning hade vi fem testare varav två oerfarna och tre erfarna. De erfarna testerna kunde spela på direkten och fann våra kontroller enkla att förstå. De oerfarna tyckte strynings kontrollerna var självklara men kunde inte gissa fram våra genvägar. Eftersom vi inte hade en meny i tetris under användartestningen kunde vi inte testa hur de skulle navigera sig i huvudmenyn

DEL C
Under scenariot för erfarna personer förutsåg vi att de skulle kunna spela som tänkt på direkten och inte ha några problem med kontrollerna vilket de även gjorde under vår användartestning. Vi nämnde även att en oerfaren skulle först välja att läsa om spelet och förstå knapparna, men eftersom vi saknade en huvudmeny där vi förklarade hur man spelade så fick de oerfarna testarna problem med att hitta genvägar.

4. Testplan

DEL A:
1. Intern användartestning
Först och främst kommer vi att testa applikationen internt i projektgruppen. Vi se till att spelets funktioner fungerar som tänkt, och komplettera med test klasser som kontrollerar alla metoder enskild.

2. Extern användartestning
I den externa användartestningen är målet främst att testa spelets användbarhet och utifrån testresultaten analysera lösningar på existerande problem och ineffektiv design.
Användarna kommer få i uppdrag att navigera sig igenom spelet på egen hand. Genom think aloud testning kommer vi snabbt se vad problemen med designen är och vad användaren förväntar sig av spelets GUI. Går användartesterna bättre än förväntat kan vi ge klara instruktioner som vi vill att användarna ska utföra för att göra testerna svårare och mer konkreta.

Vi kommer i de externa användartesterna använda två olika scenarion, ett med en erfaren datoranvändare och ett med en i stort sätt helt okunnig datoranvändare. Detta i enlighet med våra tidigare specificerade användarscenarion.

Vårt mål är att göra fem användartester och sedan revidera spelet om behovet finns.

Projekt Sirtet
Grupp: YAM Prodcutions
Kurs: Introduktion till datalogi

DEL B

TestProtokoll

Uppgift:	J	A	S	L	S
Starta spelet	“ Utan problem”	“Inga probem”	“.. lätt”	“Var ju bara att trycka.”	“ Gick bra”
Spela spelet	“Spelar som man ska”	“Tog lite tid att förstå”	“Ja, det är ju tetris”	“Stor plan.”	“Enkelt”
Pröva att navigera de fallande blocken	“Självklara knappar”	“Självklara knappar”	“Lite konstigt med acceleration när man trycker in knapparna.”	“Ja, de rör sig som man tror lätt styrning.”	“Bra navigering”
Rotera fallande block	“Gick bara bra”	“ Utan problem”	“Roterar som de ska..”	“Inga problem.”	“Enkelt”
Pausa spelet och sedan unpause	“Inga konstigheter”	“Lyckades pausa”	“Tryckte på F3 först, men P är också självklart.”	“Det var svårt när man inte visste hur man gör.”	“Enkelt med menu bar”
Pausa spelet med hotkey	“Funkade bra”	“Hitta inte hotkey”	-||-	“Var lätt när man sett tangenten i menyn.”	“Gissade fram genväg”
“Förlora” i spelet.	“Enkelt”	“Slutar som det brukar”	“Som väntat.”	“Ja.. det tog slut.”	“Fungerade som det skulle”
Avsluta spelet efter en session	“Också självklart”	“Tryckte på kryss”	“Försökte med hotkeys, men gick inte.”	“När jag använt menyn en gång var det lätt att navigera den.”	“Med den tydliga menyn var det enkelt”
Avsluta spelet mitt i en session	“Funkade bra”	“Tryckte på fönster kryss”	“Använde menyn.”	“Samma som efter att jag förlorat.”	“Enkel meny”
Starta om spelet	“Lätt med meny knappen.”	“ Tog lite tid att hitta meny”	“Samma, finns ingen hotkey så använde menyn.”	“Också i menyn.”	“Gjordes enkelt med menyn”
Övriga kommentarer:	“Inget övrigt att tillägga”	“Förlåt är inte så van”	“Det är tetris, enkelt och bra.”	“Tänk på att det är en nästan-pensionär som testar.”	“Bra gjord samt lätt att spela”

DEL C

Under den interna användartestningen fick fann vi de väsentliga buggarna genom att själva spela spelet och progressivt fixa till problem som uppstod. Vi fick bland annat problem med rotationen där block kunde roteras in i väggen, rörelsen av block i sidled samt utritningen av spel frame (layout). Den interna användartestningen ledde till att vi inte fann några buggar under den externa testningen men fick istället feedback på vad vi kunde ändra/förbättra. Majoriten av användartesterna kunde klara test protokollet utan märkvärdiga problem (se tabellen), vi fick feedback på att någonstans skriva ut controls och dess genvägar vilket vi löste med hjälp av en "Start Meny" där man välja att läsa om vårt spel innan man startar spelet.

5. Programdesign

DEL A:

Vi försöker arbeta enligt MVC, det vill säga att dela upp projektet i lager som var och en ansvarar för sitt separata område. De tre lagren kommer vara Model, View och Controller.

Model ansvarar för spellogiken och i den processeras och beräknas alla olika element inom spelet.
Models klasser:

1. **Board:** Denna klass kommer innehålla och skapa spelplanen och allt som ska finnas på den samt ha hand om dess förändringar. Display kommer kunna hämta information ur Board för att kunna visa spelplanen i spelets fönster.

Metoder i Board:

- makeBoard(): Skapar spelplanen med hjälp av en tvådimensionell vektor.
- makePlayer(): Skapar spelaren och placerar denna på spelplanen.
- makeZombies(): Skapar zombies och placerar dessa på spelplanen.
- makeItems(): Skapar items och placerar dessa på spelplanen.

2. **Enum:** Spelplanen kommer bestå av en tvådimensionell vektor av enums och för att man ska kunna komma åt dessa enums från flera klasser har vi valt att skapa en egen klass för dessa.

View ritar upp modellen och visar denna för användaren för att interaktion med spelet ska bli så enkel och logisk som möjligt.

Views klasser:
Display: Denna klass kommer se till att spelplanen visas på skärmen med rätt layout.

Metoder i Display:

- Render(): Ser till att bilder buffras och visas i spelets fönster.

Controller ansvarar för att lyssna på och svara på användarinteraktion, så att huvudkaraktären förflyttar sig åt höger när användaren trycker på högerpil, t.ex.

Controllers klasser:
UserInput: Processerar användarens input och utför en lämplig handling som svar på detta.
Metoder i UserInput:

- receiveInput(): Tar emot användarens knapptryck och utför en lämplig handling genom att interagera med Model.

DEL B:

Vi försökte arbeta enligt MVC, det vill säga att dela upp projektet i lager som var och en ansvarar för sitt separata område. De tre lagren kommer vara Model, View och Controller.

De viktigaste klasserna:

1. **Board:** Ansvarar för själva spelplanen och tillhörande logik och spelelement.
Några viktiga metoder i Board:

- createEmptyBoard(): Skapar spelplanen och lagrar positionsdata för statiska block(fallna tetrominos) i en tvådimensionell vektor.
- playerMovePoly(): Hanterar alla input från spelaren och agerar sedan med hjälp av flera hjälpmetoder
- checkRows(): Kollar om en eller flera rader är fulla och tar sedan bort dessa och anpassar resten av spelplanen med hjälp av flera hjälpmetoder
- tick(): Ansvarar för att utföra flertalet operationer, med hjälp av flertarlet andra metdoer, under en spelrunda(en tick)

2. **TetrisComponent:** Är en "custom" JComponent som ansvara för att rita ut spelplanen och de fallande blocken(tetrominos)
Några av de viktigaste metoderna i TetrisComponent:

- paintComponent(): En ärvd metod som ansvara för att rita ut komponenten
- paintFalling(): Ritar ut de fallande blocken.

3. **TetrisFrame**: Ansvarar för att rita ut ramar och tillhörande menyer till spelplanen.
Några av de viktigaste metoderna:

- makeBoardPanel(): Skapar en ny panel som innestluter spelet m.m.
- makeSidePanel(): Skapar en ny panel som innehåller all spelinformation.

4. **Keyboard**: Tar hand om all input från tangentbordet.
Några av de viktigaste metoderna:

- keyPressed(): Ärvd metod(Overridden) som kollar vilka tangenter som är nedtryckta
- keyReleased(): Ärvd metod(Overridden) som kollar om en tangent är "släppt"

5. **NextBlockComponent**: Är en "custom" JComponent som ansvara för att rita ut kommande block(tetromino) och de fallande blocken(tetrominos)
Några av de viktigaste metoderna i TetrisComponent:

- paintComponent(): En ärvd metod som ansvara för att rita ut komponenten
- paintNextFalling(): Ritar ut nästa fallande block.

6. **TetrominoMaker**: Skapar alla block(tetrominos)
Några av de viktigaste metoderna:
- getPoly(): Skapar och returnerar en poly
- getNumberOfTypes():Returnerar antalet typer.

DEL C:
Eftersom vi ändrade oss och istället utvecklade tetris är det svårt att jämföra del A och B. Däremot liknar uppläggen varandra en hel del.

6. Tekniska frågor

DEL A:
En lista av tekniska frågor som ni måste hantera när ni bygger ert system. Var så detaljerad som möjligt. Ett viktigt steg mot en god design är att få ner så många frågor som möjligt på papper på ett organiserat sätt med så många förslag till lösningar som möjligt.

- Hur ska vi implementera karaktärens grafiska rörelser på ett bra sätt (Ben & Armar)?
- Hur kan vi få zombies att röra sig mot huvudkaraktären?
- Hur ska vi implementera algoritmen för att slumpmässigt placera perks på kartan?
- Hur ska man hantera när huvudkaraktären når en vägg, ska den byta riktning automatiskt?
- Hur kontrollerar vi att zombies inte skapas för nära spelaren?
- Vilken metod ska vi använda för att hantera tangentbordstryckningar?
- På vilket sätt ska vi få karaktären att förflytta sig på spelplanen?

DEL B:
- Hur ska vi representera spelplanen på ett sätt som gör logiken enkel att arbeta med?
- Hur ska vi representera de fallande klossarna på ett sätt som gör att klossarna lätt kan interagera med brädet och tar hänsyn till knapptryck, kollision och så vidare?
- Hur ska vi implementera klossarnas hastighet när de faller?
- Hur ska vi implementera rotationen av klossarna?
- Hur ska vi implementera att rader försvinner och allt ovanför faller ner ett steg i spelplanen?
- Vilken metod ska vi använda för att hantera tangentbordstryckningar?

DEL C
Självklart ändrades de tekniska detaljerna och frågeställningen då vi gjorde ett helt annat spel. Frågorna är samtidigt lika då de är väldigt logiskt orienterade vilket är det område där vi förväntade oss att stöta på problem. Vi har lyckligtvis lyckats lösa de logiska problemen och har med hjälp av användartester tagit bort alla buggar vi lyckats hitta.

7. Arbetsplan

DEL A:

1. Prolog

Vi hade från början bestämt oss för att utveckla ett Tetris liknande spel och letade således efter en bra guide för att hjälpa oss att komma igång och strukturera en spelmotor.
Vi hittade en laboration från Linköpings universitet:
https://www.ida.liu.se/~TDDD78/labs/2014/lab4/ som passade oss perfekt då den beskrev de olika delarna generellt istället för att steg för steg hjälpa en skriva all kod.
Laborationen inspirerade oss till att skriva ett helt annat spel, men med utgångspunkt i det upplägg vi lärde oss.

2. Tidsplan

Inför varje etapp hålls ett kortare gruppmöte där vi diskuterar vad som ska utföras samt hur det ska delas upp mellan oss gruppmedlemmar.

- **Etapp 1(delvis avklarad):**
  Beskrivning: Slutföra ovan nämnda laboration för att ha en bättre förståelse för vilka komponenter som behövs för att bygga ett fungerande spel.
  Deltagar: Ziad, Safir & Daniel.
  Deadline: 4/5

- **Etapp 2:**
  Beskrivning: Utveckla en fungerande prototyp av spelet där, beroende på tid, en del funktioner är förenklade
  Deltagare: Ziad, Safir & Daniel.
  Deadline: 9/5

- **Etapp 3:**
  Beskrivning: Färdigutveckla spelet och utföra användartestning.
  Deltagare: Ziad, Safir & Daniel.
  Deadline: 13/5

- **Etapp 4:**
  Beskrivning: Bearbete feedbacken från användartestningen och förbättra spelet.
  Deltagare: Ziad, Safir & Daniel.
  Deadline: 16/5

DEL B

1. Prolog

Det blev till slut att vi skrev Tetris och vi använde oss av vissa implementationstips av laborationen (https://www.ida.liu.se/~TDDD78/labs/2014/lab4/), men vi lämnade sedan laborationen för att skriva spelet på egen hand, vilket ledde till att det tog ganska mycket längre tid än väntat.

2. Tidsplan

Inför varje etapp hålls ett kortare gruppmöte där vi diskuterar vad som ska utföras samt hur det ska delas upp mellan oss gruppmedlemmar.

- **Etapp 1:**
  Beskrivning: Alla implementerar grunden i ett tetris för att få en basal förståelse för logiken i spelet fungerar.
  Deltagare: Ziad, Safir & Daniel.
  Deadline: 7/5

- **Etapp 2:**
  Beskrivning: Utveckla en fungerande prototyp av spelet där, beroende på tid, en del funktioner är förenklade. Spelet ska vara helt spelbart och logiken ska vara färdig. Klasserna är nu uppdelade mellan gruppedlemmarna så att alla bidrar och håller koll på sin egen del av projektet.
  Deltagare: Ziad, Safir & Daniel.
  Deadline: 12/5

- **Etapp 3:**
  Beskrivning: Färdigutveckla spelet och utföra användartestning.
  Deltagare: Ziad, Safir & Daniel.
  Deadline: 14/5

- **Etapp 4:**
  Beskrivning: Bearbete feedbacken från användartestningen och förbättra spelet.
  Deltagare: Ziad, Safir & Daniel.
  Deadline: 16/5

DEL C

Planen gjordes om väldigt lite i och med bytet av spel, blev vi lite försenade och försköt många deadlines några dagar. Det var dessutom lättare att dela upp arbetet mellan oss och utförligt beskriva vad som skulle vara klart till när efter att vi fått en djupare förståelse för vad vi skulle skriva för program och hur det skulle komma att fungera.

8. Sammanfattning

Efter många tankar kring vilket spel vi skulle göra och hur implementationen för spelet skulle göras landade vi i alla fall på tetris. Vi skrev det med höga krav på att spelet potentiellt sett ska gå att utveckla vidare och fokuserade mycket på louse coupling och sammanhängande kod. Vi har försökt minimera hårdkodandet vilket har varit väldigt lärorikt och gett oss en struktur på koden vi är väldigt nöjda med.

Att koda logiken och algoritmerna för de olika metoderna har varit väldigt givande (stundtals frustrerande) och verkligen fått oss att tänka oss in i hur spelet faktiskt fungerar. Att implementera sin “egen fysik” i ett spel gör att man behöver tänka på de villkor man själv satt upp samt att man får lida konsekvenserna av sin implementation