TDDC76 - Programmering och Datastrukturer

Erfarenhetsrapport

Aqua Invaders

Version 1.0

Anmar Karmush, anmka989@student.liu.se

Nils-Ruben Skogemyr, nilsk758@student.liu.se

Sigge Rystedt, sigry751@student.liu.se

Isak Wernroth, isawe536@student.liu.se

Clara Ekbäck, claek609@student.liu.se

Emil Eriksson, emier604@student.liu.se

Institutionen för datavetenskap (IDA)
Programvara och System (SAS)
Linköpings Universitet

December 12, 2023

Contents

1. Tidsplan och upplägg

På första mötet började vi muntligt diskutera ett gruppkontrakt, där information så som medlemmarnas tidigare erfarenheter och ambitionsnivå togs upp. Vi pratade även om hur samtliga medlemmar arbetar mest idealt, om det är tillsammans med med någon eller självständigt. Det här visade sig vara väldigt bra då vi snabbt kunde få en överblick angående vilken nivå vårt spel ska vara på och hur vi ska arbeta för att komma till en accepterar nivå. Efter en diskussion om gruppkontraktet började vi som grupp att diskutera en tidsplan. Vi bestämde oss snabbt för att det är lättast att vi anpassar oss till kursens hårda deadlines för att fastställa vår tidsplan. De första två veckorna la vi mycket tid på att få en tydlig och väldefinierad kravspecfikation samt objekt orienterad analys (OAA). Samtliga i gruppen kom väldigt snabbt överens under gruppkontrakts diskussionen att en välplanerad kravspecfickation och OOA kommer göra arbetet betydligt lättare. Vi la också fokus på att hitta bra program för att skriva rapporter och göra klassdiagram. Vi bestämde oss att använda Overleaf för dokumentskrivande och Lucidchart för att skapa vårt klassdiagram. Dessa program var nya för vissa medlemmar men hela gruppen men höll med att det var värt inlärningskurvan. Produktiviteten ökade också betydande när alla hade blivit familjära med programmen. Exempelvis var det väldigt lätt att lägga till och ta bort krav från kravspecfikationen och ändra i klassdiagrammet.

Efter att vi hade bestämt oss på en spelidé så började diskussionen angående vilka krav vi behöver samt även vilka klasser som vi vet vi kommer att behöva. OOA:n var lite svårare att göra då det inte var lätt att redan vid projektets start bestämma vilka klasser vi skulle behöva. Vi utgick mest från vad vi själva trodde vi skulle behöva samt från projektföreläsningen. Exempelvis så visste vi från start att vi skulle behöva en virtuell *States* klass som gör att vi lätt kan byta mellan olika states. Detsamma var sant för en virtuell *Object* klass som ska innehålla allt gemensamt som de olika objekten kommer att ha. Det finns definitivt stora skillnader mellan versionerna för OOA:n. Men själva grundiden är samma och det var framförallt mer detaljer som behövde ändras. Exempelvis var inte helt säkra på vad skulle finnas i vår Context klass. Mer information om OAA processen kan läsas i sektion 2.2.

Efter dokutmentskrivande, som tog ungefär en och en halv vecka så började vi programmera. Vi bestämde under dokuemntskrivandet att jobba i grupper av två skulle vara det bästa upplägget för oss. Eftersom vi var säkra att vi behövde States och Object klasserna, så blev detta en naturlig uppdelning. Två personer jobbade på States klasserna, och fyra personer jobbade på Object klasserna då vi kände att mer tid skulle behövas för dessa klasser. I början gick också en hel del fokus på lära känna SFML och vad bibliotektet hade att erbjuda.

Allt som allt så funkade detta väldigt bra för oss. Genom att hålla oss till projektets hårda deadlines så kände vi att vi höll vår tidsplan väldigt bra. Det var aldrig någon period under projektet där vi kände att vi inte skulle hinna klart. Och eftersom vi hade lagt en mycket tid och energi på tydliga krav och en välplanerad OOA så var det lätt att komma igång. Det var dessutom relativt enkelt att fortsätta bygga vidare på klasser och idéer.

2. Projektets olika faser

I det här kapitlet kommer vi diskutera projektets olika faster.

2.1 Krav

När vi började projektet var vår plan att skriva en utförlig kravspecifikation där vi beskrev vad alla delar ska göra. Detta gjorde vi genom att för varje klass skriva ner ett antal mätbara krav och rangordna dem som absoluta, bör och kanske krav. Där absolut-kraven var dem som slutprodukten minst skulle uppfylla, bör-kraven som borde uppfyllas men slutprodukten skulle fungera utan dem och kanske-kraven skulle uppfyllas i mån av tid. Överlag höll vi oss till vår kravspecfikation genom hela projektet och att vi gjorde den tydlig från början förenklade arbetet under projektets gång märkbart. Istället för att skriva kod med endast en slutprodukt som mål, så kunde vi ta ett absolut-krav i taget och när dessa var uppfyllda hade vi en fungerande produkt. Sedan blev det självklart så att vi fick revidera kravspecifikationen under projektets gång. Detta berodde på att när vi kodade märkte att vi behövde ändra vissa saker för en bättre speldesign och fick därmed ändra i våra krav. När vår slutprodukt var färdig och vi gick igenom vår kravspecifikation så uppfyllde vi alla våra absolut-krav, de flesta bör-kraven och några kanske-krav. Sammanfattningsvis kan man säga att arbetet vi lade i början på att göra en utförlig kravspecifikation gjorde att arbetet gick smidigt och kunde håla oss till den genom hela projektet. Det som eventuellt skulle kunna gjorts bättre var att när vi behövde ändra något i koden som påverka ett krav, se till att direkt ändra det i kravspecifikationen så att alla vet vad som gäller.

2.2 Objektorienterad analys (OOA)

Vi satte upp ett UML-diagram i början av projektarbetet, vilket gjorde att övergången till OOA:n gick smidigt, då vi endast överförde det mesta vi gjort. Vi hade en gemensam idé, hur vi ville att programmet skulle se ut vilket gjorde att vi fick en bra start och följde tidsplanen. Vi blev dock tvungna att ändra UML-diagrammet och därmed OOA:n med mindre detaljer under implementeringen, för att få projektet att flyta på. Efterhand upptäckte vi delar av spelet som kunde göras smartare. Exempelvis ändrades hur hajen rörde sig och dess kollision med avataren, samt slog ihop klasserna powerup och powerdown till en gemensam klass, då de hade snarlik funktionalitet och egentligen kunde separeras enklare med en variabel.

Många av dessa små ändringar var väldigt rimliga och gjordes tidigt, vilket gjorde att det inte blev problematiskt att göra dessa ändringar. Vi förlorade inte heller någon tid på det. Denna fas flöt på bra, vilket antagligen är resultatet av att vi diskuterade mycket kring detta och planerade väl, då hela gruppen var samlad.

2.3 Implementation

I början av implementeringen jobbade vi mycket i par. Det fungerade bra på flera sätt. Det var givande att ha någon att diskutera med när klasserna skulle implementeras då det ofta fanns flera vägar att gå. Det var också ett ganska effektivt sätt att jobba på eftersom man inte kör fast på samma sätt som man kan göra själv, samtidigt som vi kunde angripa flera delar av projektet samtidigt. Slutligen fungerade det bra rent logistiskt då det är enklare att hitta tid att arbeta om man bara är två personer jämfört med att ses alla sex i gruppen.

Det svåra med vårt upplägg var sedan att få ihop allt till ett sammanhängande projekt. Något som fungerade bra då var att byta par så två som hade bra koll på olika klasser tillsammans kunde få klasserna att fungera med varandra som önskat. Under några veckor i mitten av projektet tvingades vi arbete mycket gemensamt som grupp. Det var inte särskillt effektivt men nödvändigt för att göra projektet mer sammanhängande. Det

var också viktigt för att alla skulle känna att de hade koll på projektets olika klasser för att därefter kunna forsätta.

Ungefär vid halvtidsavstämningen hade vi ett sammanhängande projekt och majoriteten av våra ansolut-krav var uppfyllda. Därefter har vi jobbat med att snygga till koden, fixa buggar, och jobba vidare med bör/kanske-krav.

För vår grupp har det fungerat bra att ses ofta. Det har förenklat kommunikationen och gjort att vi kunnat ligga i bra fas tidsmässigt eftersom man inte tänker att något kan vänta tills nästa vecka "när man ändå ska ses hela gruppen". Det har varit nyttigt att programmera i olika konstellationer; i olika par, som större grupp, och på egen hand.

2.4 Resultat

Det färdiga spelet är väldigt likt det vi planerade i början, även om en del har ändrats och lagts till. Det mesta kring funktionaliteten har förblivit oförändrad genom arbetet såsom hur highscore-listan fungerar och hur bläckfiskarna attackerar spelaren. Att vi lyckades hålla oss så nära den ursprungliga visionen är tack vare vår kravspecifikation. Den gjorde det väldigt enkelt att följa planen och prioritera vad som var viktigt att fokusera på. Dessutom kunde vi också implementera många bör-krav, vilket till stor det beror på att de oftast var enkla att lägga till när spelet redan fungerade. Vi la också till extra funktionalitet som en valbar svårighetsgrad. Många av kanske-kraven uppfylldes däremot aldrig men dessa var inte särskilt viktiga och vissa av de hade varit mer tidskrävande att implementera. Sammanfattningsvis är spelet väldigt nära vad vi ursprungligen planerade. Dessutom la vi till extra funktionalitet och kunde implemetera många bör-kraven, men inte alla kanske-krav.

3. Viktigaste personliga erfarenheten

I det här kapitlet får varenda gruppmedlem skriva om den alla viktigaste personliga erfarenheten/kunskapen som projektarbetet bidragit till.

3.1 Anmar

Eftersom detta är min andra iteration av projektet tar jag framförallt med mig att jag genomförde projektet på det sätt det faktiskt är tänkt att göras. Det vill säga att vi började tidigt med att lägga en stark grund för hur spelet skulle se ut och vad vi ville inkludera i det. Därefter, från ett tomt dokument, skrev vi sakta men säkert kod för att få ett fungerande spel. Det kändes väldigt skönt för mig och fick mig också att inse att projektet i sig inte var så svårt, vilket jag inte kände när jag gjorde projektet den första gången.
Då jag har tidigare erfarenheter från C++ och git från andra kurser så la jag mycket fokus på att hjälpa de i gruppen som inte var lika vana. Jag försökte också inta en ledande roll genom hela projektet, särskilt genom att hålla koll på hur det gick för de olika grupperna och se om det var något de behövde hjälp med. Eftersom jag hade en väldigt bra grupp i år var det även väldigt enkelt att fördela uppgifter och vara säker på att de skulle bli klara i tid. Dessa erfarenheter tar jag också med mig från projektet.

3.2 Ruben

Jag har under projektets gång lärt mig massor om C++, git, mm, men det viktigaste jag tar med mig är att jag lärt mig programmera mer strukturerat. Detta, genom att vi alla tidigt byggde en stark grund för hur projektet skulle se ut, samt delade upp oss i grupper som skötte olika objekt och klasser, vilket skiljde lite från tidigare, då jag bara hoppat in och hoppats på det bästa . Detta ledde till att projektet flöt på i ett bra tempo och de olika klasserna byggdes upp parallellt. En liten nackdel med detta, var just att man ibland kunde behöva andras klasser för att se att sina egna fungerade. Exempelvis, för att se att Octpous förlorade HP vid en kollision, behövdes klassen för projektiler för att ens kunna skjuta. Detta löste vi genom att fixa den grundläggande funktionaliteten för projectile-klassen , så att testerna gick att genomföra. Till nästa gång vet jag att ett sådant problem simpelt går att lösa genom att planera vilka klasser som bör fokuseras på tidigt, då de kan behövas för att få andra klasser att fungera ordentligt.

Jag tar även med mig lärdomen, att det ej går att planera ett projekt helt och hållet från början, oavsett hur erfaren man är, det kommer alltid krävas att man måste justera klasser eller liknande, det är därför viktigt att man är öppen för förändring även om man tycker sig gjort allt rätt från början. Detta skedde exempelvis vid designen av hur Power-klassen skulle aktivera sina effekter, där vi ändrade oss från att man sköt ned den med en projektil, till att man krockar med den för att aktivera den. Detta gjorde att användningen av ett power objekt blev roligare rent spelmässigt och kodmässigt, då de enda objekten som behövde inkluderas i proceduren var avatar och power, inte avatar,power och projectile.Hela gruppen var engagerade i projektet, och alla arbetade hårt, vilket gjorde att klasserna utvecklades i samma takt, parallellt. Det resulterade i att projektet blev klart, med god tidsmarginal. Detta är ett resultat jag är nöjd med.

3.3 Isak

Den främsta erfarenheten jag tar med mig från projektet är hur man programmerar effektivt i en större grupp. Vi delade ganska tidigt i projektet upp oss i två gruppper, en med ansvar för Obcjects och en med ansvar för States. Överlag fungerade detta bra och gjorde så att vi kunde skriva klart respektive del snabbare. Vi som jobbade med Object-delen hade inte speciellt mycket erfarenhet med git sedan tidigare, vilket gav oss problem då vi ofta jobbade på samma filer, vilket gav en hel del mergekonflikter att reda ut. Under projektets gång lärde vi oss att arbeta mer effektivt med git och tror att detta kommer spara en hel del tid i senare projekt. När vi skulle kombinera States och Object delarna hade vi lite problem då vi i grupperna hade skrivit kod som inte var helt kompatibel, vilket resulterade i att vi fick spendera mer tid en förväntat på att skriva om kod för att det skulle fungera tillsamman. Detta är troligtvis svårt att undvika fullständigt, men kunde troligtvis minskats om vi hade diskuterat hur delarna skulle hänga ihop kontinuerligt under tiden vi skrev koden. Sammanfattningsvis tycker jag att vi lyckades bra för att vara relativt oerfarna på att programmera i större grupp och tycker att vi hanterat problemen som uppstått på ett bra sätt.

3.4 Emil

En av de viktigare erfarenheterna för mig i projektet har varit att ha en tydlig planering över vad alla funktioner och klasser ska göra, vilket vi hade i ett UML-diagram. Då kan man lätt se vilka funktioner som behövs i varje klass och vilka funktioner som behövs i ett flertal klasser och därför kan vara lämpliga att ärva. Dock när vi programmerade insåg vi flera gånger att vi behövde ändra i UML-diagrammet för att vi la till funktionalitet eller förändrade vad klassen borde ha hand om. Detta fungerade bra då vi kommunicerade med varandra och förklarade vad vi ville ändra så att alla kunde skriva sin kod så det fungerde bra med helheten av programmet.

3.5 Sigge

Jag har lärt mig väldigt mycket i alla steg av projektet, hela vägen från planeringen till genom hela programmeringen av spelet. En stor del av vad jag tar med mig till framtida projekt är hur man planerar kravspec, hur man gör ett OOA och UML-diagram. Speciellt kravspecen är något som verkligen har hjälp oss under projektets gång och som jag tidigare inte trott varit så viktigt. Den gjorde det väldigt tydligt hur man skulle prioritera under projektets gång; absolut-kraven som krävs för att spelet ska fungera, bör-kraven som borde vara med men är inte nödvändiga och kanske-kraven som vi lägger till om tid finns. I framtiden är det inte så mycket jag vill ändra på hur vi gjorde det, då det som beskrivet innan fungerade väldigt bra. Dessutom lärde jag mig mycket om c++; både syntax och hur man ska strukturera kod. Det huvudsakliga med c++ som var nytt för mig var hela idén hur man ska strukturera programmet med olika state:s. Det var ett sätt väldigt annorlunda sätt för mig att tänka och tog ett tag innan jag vande mig. Det var dock värt insattsen att lära sig och jag kan inte tänka mig hur vi skulle gjort det här projektet utan att strukturera koden så. Sammanfattningsvis har jag lärt mig väldigt mycket om hur man arbetar i grupp, planerar projekt och programmerar i c++. Jag tycker att projektet har gått väldigt bra och att jag har lärt mig oerhört mycket av det.

3.6 Clara

Under projektet har jag lärt mig mycket nytt och jag tycker att jag utvecklat min programmering. Såklart har jag lärt mig mer om C++, git och att jobba mer fritt med ett projekt, men det har också varit nyttigt att programmera med nya personer. Om det är något jag ska försöka ta med mig så är det att ta lite mer initiativ och att jag kan mer än vad jag tror. Även om jag tänker att någon annan hade skrivit bättre kod eller löst ett problem snabbare så ska jag försöka ge mig själv tiden att själv fundera över vad jag tror hade blivit bra.

4. Avslutande ord

Sammanfattningsvis tycker vi att projektet gick smidigt från start till mål. Vi lyckades uppfylla alla våra absolut krav och hann även med några extra krav. Dessutom höll vi alla våra deadlines och det fanns aldrig någon risk att vi inte skulle hinna bli klara. Det har fungerat bra att arbeta i grupp och vi har inte haft några konflikter. Alla i gruppen bidrog med sina unika erfarenheter och arbetade effektivt tillsammans, oberoende av vem man jobbade med. Det resulterande i att spelet uppfyllde våra förväntningar, och alla är nöjda över slutprodukten samt det arbete som samtliga personer har lagt ner.