Chalmers tekniska högskola  
TDA367 Objektorienterat Programmeringsprojekt

Robin Lilius-Lundmark

Lina Blomkvist

Andreas Carlsson

Felix Engelbrektsson

2016-05-09

Vaporwave

(Undertitel här: namnet på projektet är heller inte klubbat än)



Sammanfattning

Denna rapport kommer beskriva hur vi tagit oss an ett objektorienterat projekt i en grupp på fyra personer, i kursen TDA367, från idé till färdig produkt, och hur vi har tagit oss dit. Vi har, föga oväntat, stött på förhinder men har även till vår bästa förmåga tagit oss förbi dessa. Vi beskriver här hur vi gjort och vad som gått bättre eller sämre.

Denna rapport kommer att beskriva hur vi har gått tillväga för att utveckla ett pixel-baserat spel för flera personer. I metod går vi igenom hur vi rent praktiskt har gjort medan teori förklarar de mer tekniska bitarna. Vi avslutar sedan med en presentation utav vårt resultat och sedan en diskussion runt omkring det.

Innehållsförteckning

1. Inledning

1.1 Bakgrund

1.2 Syfte

1. Metod

2.1 Tekniska lösningar

2.2 Arbetsflöde

2.3 Avgränsning

1. Teori
2. Resultat
3. Diskussion

5.1 Slutsatser

1. Slutsats

Källor

Bilagor

1 Inledning

1.1 Bakgrund

I slutet på första året av IT-linjen på Chalmers har vi i kursen TDA367 Objektorienterat Programmeringsprojekt, där uppgiften är att använda det vi hittills lärt oss för att i grupp driva ett projekt att programmera någon typ av applikation i språket java.

1.2 Syfte

Vi beslöt oss för att programmera ett datorspel från grunden, som bygger på idén från det gamla Bomberman, med stöd för flera spelare samtidigt. Vi har också bestämt oss för en väldigt speciell grafisk approach, som kallas “vaporwave”.

Precis som med majoriteten av spel som finns på marknaden så utvecklade vi vår produkt i rent underhållningssyfte. Det skall vara enkelt att köra igång och spela en liten stund i mån av tid, men även ha tillräckligt med djup för att man skall kunna fastna timmar i sträck.

2 Metod

2.1 Tekniska lösningar

Genomgående i projektet har vi valt att arbeta med egna lösningar framför att använda ramverk. Vår utgångspunkt har varit att det är lättare att förstå kod som vi själva skrivit.

Vi har också som ambition att skriva koden så säker och felfri som bara möjligt, och för att göra detta har vi använt oss utav STAN, vilket är ett program som tittar hur de olika klasserna har beroenden till varandra. Målet är att få ett så lågt antal beroenden som möjligt. Vi har också använt oss utav Find Bugs som letar upp felaktig, osäker och dålig kod.

Vi har designat vår applikation enligt Model-view-controller-mönstret. Då delar man upp alla klasser som modeller, vyer och controllers. Det finns ett antal olika varianter av mönstret. I vår applikation har vi valt en vanlig lösning där modellen och vyn är helt oberoende av varandra. Controllern känner till både modellen och vyn och skicka information från modellen till vyn.

Modell-klasser använder vi för att modellera verkliga saker. Detta innebär i vårt fall karaktär, bomb, fiende och liknande.

Vyer är klasser som används för att visa saker på skärmen. Dessa ska inte innehålla någon logik utan ska endast få till sig information och rita upp detta på skärmen.

Controllers är länken mellan modeller och vyn

2.2 Arbetsflöde

Vår strategi för projektet har varit att bygga en ordentlig grund, och sedan koppla på mer delar utifrån det. Vi har haft väldigt många iterationer utav spelet, där varje iteration haft ytterligare funktionalitet.

Vi hade redan från början en vag vision om hur spelet skulle se ut och hur det skulle fungera innan vi startade projektet. I början var det mycket diskussioner om vad för funktionalitet vi skulle ha, hur spelet skulle fungera och framförallt hur spelet skulle se ut.

Efter att vi hade pratat igenom det ordentligt och ritat upp skisser på spelet började vår vision bli mycket tydligare.

De första veckorna spenderades tiden mestadels med att rita diagram och flöden för hur spelet skulle vara uppbyggt och vad som skulle hända när en spelare gjorde ett val.

Vi fick också starta igång ett Github-repository för att kunna synkronisera koden mellan de olika programmerarna, samt sätta upp en utvecklingsmiljö som vi kunde arbeta i.

När vi hade nått ett UML-diagram som alla var nöjda med kunde vi börja med själva skapandet av spelet.

Första iterationens mål var enkelt, vi skulle få en karaktär att kunna röra sig på skärmen, och utvecklades därför snabbt till en lite mer utmanande uppgift - nämligen att kunna släppa bomber på en position också.

Vi fortsatte efter det med inläsning utav karta, och därmed också dess vy, man skulle kunna gå runt på vår karta, och bli stoppad av diverse hinder.

Nästa steg var självklart för oss - att kunna spränga de hinder som var förstörbara. Eftersom vi i vårt modell diagram redan hade förberett oss för det så gick det ganska snabbt att implementera.

Däremot så skulle det finnas en chans att “powerups” dök upp i dess ställe, och det var lite mer tidskrävande. Vi har fyra olika typer, och de skulle inte ha samma chans, då vissa var bättre än andra. Dessutom ville vi ha med estetiskt tillfredställande animationer för detta.

Pararellt med detta arbetade andra delar utav gruppen med att utveckla en artificiell intelligens eller algoritm för att få datorstyrda karaktärer att röra sig runt om på skärmen.

Detta krävdes inte särskilt mycket kodande för att implementera, men däremot så var det mycket planering, funderande och testande kring detta.

Det visade sig vara svårare än väntat, och i första iterationen av våra datorstyrda spelare använde vi oss utav ett evalueringsystem som hade ett konstant värde runt omkring våra karaktärer.

Vi insåg ganska kvickt att detta skulle behövas göras om rekursivt, för att det skulle fungera bättre.

Det var inte förrän i fjärde iterationen som vi implementerade en första prototyp utav menysystemet. Tanken var att man skulle kunna välja hur många spelare som skulle vara med, vilken bana man skulle spela, vilka karaktärer spelarna skulle vara och diverse andra småinställningar.

Precis som tidigare så började vi dock med en grund, och utvecklade därefter, vilket gjorde att först kunde man enbart starta ett spel eller avsluta det.

Eftersom vi har så mycket bilder och grafiska element så tog det väldigt lång tid att ladda spelet. Speciellt efter att vi hade implementerat ljudeffekter också.

Delar av vår grupp fick ta till sig uppdraget att fixa detta på ett snyggt sätt vilket gjorde att vi fick utveckla en laddningsskärm som visar hur mycket av innehållet som laddas in,

och visar upp en bild samtidigt.

Vi fortsatte dessutom att försöka ordna evalueringsystemet för de datorstyrda karaktärerna på ett snyggare sätt, utan något ordentligt genombrott.

Vårt största hinder fann sig nu, där vi under en hel vecka försökte implementera stöd för spel över internet. Då våra kunskaper inom detta område är väldigt begränsat, så gick det långsamt fram. Efter att det tagit alldeles för mycket tid, utan några riktiga framsteg så var vi tvungna att dra i nödbromsen.

Vi var helt enkelt tvungna att lägga ner stöd för online-spel helt och hållet.

Vårt fokus lades nu istället på att göra så att flera spelare kan spela på en dator, och försöka hitta på bra lösningar för det istället.

Det dröjde inte lång tid innan två spelare kunde spela samtidigt, då vi hade förberett för detta redan innan.

I sjätte iterationen tyckte vi det var viktigt att börja fokusera på extra funktionalitet som skulle förhöja spelupplevelsen.

Vi utvecklade en skärm som visade hur spelarna låg till jämfört med varandra, med ett slags poängsystem som kan visas upp vid ett knapptryck.

En annan viktig funktion var givetvis att kunna pausa spelet. Tanken var då att en meny skulle visas, där man kunde anpassa inställningar efter eget tycke.

Detta fick dock slopas då vi började få ont om tid, och istället visas bara en indikator att spelet är pausat.

Arbetet fortsatte med essentiella delar såsom menysystem, stöd för ännu fler spelare och en förbättrad evalueringsalgoritm.

Vi fick tänka om lite på tillvägagångssättet gällande evakueringsalgoritmen och det ledde oss till en rekursiv version utav den.

Det slutade med att den delade ut ett lämpligt värde så att de datorstyrda karaktärerna jagar de spelarstyrka karaktärerna och delar ut skada då de kommer ifatt dem.

Eftersom vi skulle ha möjligheten att spela mot en datorstyrd spelare, som skulle vara lika duktig som iallafall en medioker spelare så var det viktigt att man skulle kunna ställa in vad för typ av spelare man vill möta.

Det gav oss en del problem i utvecklingen utav menysystemet, då vi inte hade planerat ordentligt för detta sedan innan.

2.3 Avgränsning

Genom projektets gång har vi prioriterat olika spelfunktioner enligt en skala 1 - 4 där 1 är mest prioriterat. Till exempel så var karaktärsrörelse en etta på skalan och multiplayer en 4.

Det vi kände var av högsta prioritet var att få ett enkelt bomberman spel med bomber, powerups och fiender. Därefter var planen att bygga på med det vi ville ha med men som inte var fullt nödvändigt för ett fungerande spel, till exempel flera spelare över nätverk.

För att inte fortsätta bygga på projektet i all oändlighet var vi också noggranna med att göra en lista från början, med allt vi ville ha med.

3 Teori

3.1 Model-view-controller

Model-view-controller är ett av de vanligaste designmönstrena som används inom objektorienterad programmering. Det förkortas oftast MVC, och så kommer det att benämnas i vår rapport.

Det är ett designmönster för att skilja modeller från vyer, genom att låta en eller flera klasser agera mellanhand, en så kallad controller.

Modellerna skickar information till controllern när något händer, i vårt fall, på spelfältet. Controllern säger sedan till vyerna att uppdateras så att vyerna stämmer överrens med informationen som precis skickades.

Detta gör man för att förhindra att objekt känner till information som de egentligen inte behöver ha tillgång till, och på så sätt kan de inte förändra informationen på något sätt.

3.2 Domändriven design

    En domändriven design innebär att kodens design utvecklats genom att utgå från en kärndomän, diskutera fram delar att koppla på den. De delar som man väljer att ha med i domänmodellen skall vara funktionella modeller som användaren utav mjukvaran ser, eller använder.

3.3 Testdriven utveckling

    Testdriven utveckling är helt enkelt vad det låter som. Man testar varje modell man skapar, dess funktioner och hur de fungerar tillsammans med varandra. Utvecklaren skall kunna köra koden lokalt, och dessutom skall det finnas tester för koden som skall godkännas innan koden sammanförs med det gemensamma projektet.

4. Resultat

Resultatet av vårt arbete har blivit en bra arbetsgång med en seriös planering så att det alltid funnits något att göra. Däremot så har det visat sig att det var viktigt att alla vet vad dem andra håller på med, delvis för att inte koda på samma grejer och delvis för att ingen ska känna att den inte hänger med.

Vår kodning har gått väldigt väl då vi har varit väldigt noggranna med att koda efter ett sant MVC koncept med många klasser och paket indelningar.

5. Diskussion

Det har sammanfattat gått väldigt bra genom projektets gång, med kodning såväl som gruppsammanhållning. Någonting som skulle kunna förbättras är att alla får veta vad de andra håller på med så att ingen kodar på samma och så att alla kan känna att de hänger med. De kan också ha varit en bra idé att dela in vad var och en skulle jobbat med från början så att alla får göra det de brinner mest för samt så att alla fått lika stora uppgifter att jobba med. Detta kan också vara viktigt för att alla ska känna att de behövs genom projektet.

Kodmässigt så skulle vi kunna varit ännu mer hårda med namngivning av variabler samt från början varit mer säkra på vad ett MVC mönster är från början då vi var tvungna att refraktorera en hel del i mitten av projektet. Vi kunde också varit mer noggranna med att inte få med några cykliska beroenden då detta ledde till ännu mer refraktorering i slutändan av projektet.

6. Slutsats

Vi har gjort ett bomberman spel med powerups, karaktärer och fiender. Det går alltid att göra mer än vad en har gjort, och det går alltid att förbättra sånt här, men vi är fullkomligt nöjda med resultatet.

Källor   

Introduction to Test Driven Development - http://agiledata.org/essays/tdd.html

Evans, Eric - Domain-Driven Design, 2003