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.

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

I kursen TDA367 beslöt vi oss för att programmera ett datorspel från grunden, som bygger på idén från det gamla Bomberman, men med twisten att få in online multiplayer. Vi har också bestämt oss för en väldigt speciell grafisk approach, som kallas “vaporwave”.

Syftet med rapporten är att redovisa för hur vi genomförde det, från idé till färdigt spel. Syftet innefattar även att evaluera hur väl denna projektformen fungerar för ett projekt av den här typen.

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 hålla nere antalet beroenden.

Vi har designat vår applikation enligt MVC-mönstret. Man kan grovt dela in 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

Innan projektarbetet kom igång på riktigt har det bollats idéer och ritats skisser för att komma fram till vår vision. Detta gjordes mestadels av ren arbetsglädje och inte för att förekomma någon form av planering, men det resulterade i detsamma; när kursen väl startade kände vi oss väl förberedda.

Arbetets gång har i princip bestått av två gruppmöten i veckan, och däremellan löst inplanerad tid då vi setts i skolan, vilket har fungerat väldigt bra då alla i gruppen har varit motiverade och generellt haft god närvaro.

På varje gruppmöte har vi kommit överens om vad som haft störst prioritering de kommande dagarna och delat upp uppgifter mellan oss inom detta prioriteringsområde, samt bestämt när vi skall ses kommande dagar och även bokat grupprum för detta etc.

Till en början handlade arbetet mest om planering och avgränsning (se nedan) och att förbereda oss inför att koda.

Efter två arbetsveckor satte vi igång med själva programmerandet, och då har vi använt oss av versionshanteringssystemet Git, via portalen github.com. Vi kikade lite på att använda oss av s.k branches, där en arbetar på olika uppgifter genom att spara ner separata versioner av projektet, men kom snabbt fram till att det innebar mer krångel än vi fick ut av det. Istället har vi, mestadels framgångsrikt, försökt kommunicera med varann innan vi tagit oss an uppgifter så att vi inte suttit och ändrat i samma kod. Detta har lett till en mycket snabbare arbetstakt, vilket i sin tur har uppfattats som väldigt positivt i gruppen.

Även om projektkursen i sig inte innefattat grafik och design, så har vi även försökt lägga stor vikt vid detta. All tid som har lagts ner på att generera snygg grafik till spelet syns inte, men där har vi lagt en hel del krut, när tid har funnits; på fritid och när annat arbete på spelet har förhindrat andra att lägga sig i för mycket.

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