LINKÖPINGS UNIVERSITET

TDP005 Projekt: Objektorienterat system

Reflektionsdokument

Författare

Mohammad Omar, mohom373@student.liu.se

INSTITUTIONEN FÖR DATAVETENSKAP

Höstterminen 2019
Version 0.1

2019-12-17

Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
0.1	Skapade reflektionsdokumentet	191217

1 Introduktion

1.1 Syfte

Syftet med kursen är att tillämpa den objektorienterade teorin man lärde sig i kursen TDP004. Detta gjorde man genom att skapa ett spel i språket C++ där fokus låg på bland annat programutvecklingsmetodik och design av större program med hjälp av UML som verktyg.

2 Erfarnheter

2.1 Tekniker och verktyg

Diverse tekniker och verktyg användes under projektets gång. Grafik biblioteket SMFL gjorde det enkelt att bland annat skapa olika Sprites eller hantera kollision med objekt. Detta gjorde att mer tid kunde läggas på implementationen av klasserna i designspecifikationen istället för att behöva uppfinna bland annat kollisionhantering på nytt. Verktyg som Clion och cmake gjorde redigering och kompilering av kod smidigt. Båda dessa verktyg kräver dock tid att lära sig men med hjälp av bland annat de olika labbpassen under projektets gång var det relativt enkelt att komma igång.

Modelleringen av spelets design gjordes med Unified Modeling Langunge (UML) som är ett objektorienterat generellt språk för modellering av olika typer av system. Med hjälp av UML kunde man enkelt och tydligt skapa en design över klassheirarkin, samt enkelt se en översikt över bland annat datamedlemmarna och metoderna som klasserna skulle ha. En bra designad UML-diagram kan underlätta implementationen av spelet då man i princip bara behöver bygga klasserna baserat hur de är i diagrammet.

Projektets största fokus är att ge studenter möjligheten att kunna använda kunskapen de har om den objektorienterade paradigmen i ett större projekt. Spel är en av de vanligaste mjukvarorna som byggs med denna designparadigm eftersom det är enkelt att föreställa sig de olika objekten på spelplanen som instanser av diverse klasser som har implementerats.

Objektorientrad programmering har många styrkor så som att man får modlär kod med klasser som ofta är återanvändbara. Pågrund av denna moduläritet kan koden också vara mer hanterbar och risken för buggar minskas betydligt mycket.

2.2 Svårigheter

Som i många projekt är det i princip oundvikligt att inte stötta på problem, men det viktigaste är att inte ge upp när man stötter på dem. De två största problemen jag hade var tidsbrist på grund av studier på mer än 100% studietakt, samt att jag arbetade själv. Men pågrund av min erfarenhet med denna kurs sedan förra året samt ett extra års programmeringserfarenhet än förväntat ledde till att jag klarade mig bra ändå.

Implementationsmässigt hade jag mest problem med kollisionshanteringen av objekten på spelplanen. Särskilt kollision mellan bollen och brädan och kollision mellan bollen och de olika poängzonerna. Jag stötte på vad verkade vara ett klassiskt problem för de som skapar en pong/breakout klon för första gången. Det enklaste man kan göra när en kollision sker mellan en boll och bräda är att ändra y-velocity till -y-velocity för att bollen ska kunna studsta och traverera i motsatt håll.

Det som ibland kan ske är att bollen inte hinner traversera utanför korsningspunkten mellan den och brädan. Detta räknas då som en kollision igen och leder till att y-velocity blir motsatt värde igen. Effekten blir då att bollen ser ut att åka igenom brädan. Detta ledde till större problem vid kollision mellan boll och poängzon då man kunde få massor med extra poäng vilket inte var ett önskvärt beteende. Efter diskussion med bland annat handledaren valde jag att lösa detta problem genom att förflytta bollen ett antal pixlar motsvarande bollens längd direkt för att inte vara kvar i korsningspunkten. Efter detta kunde jag ändra på y-velocity för att få den önskade effekten av en studs utan problem.

3 Sammanfattning

För att kort sammanfatta min erfarenhet med projektet i sin helhet skulle jag säga att jag är nöjd med min insats och resultatet. Nu har jag en fungerande bas som bör vara enkel att utveckla vidare på grund av de tekniker som användes. Den objektorienterade strukturen kommer göra så att det enkelt kan läggas till fler power-ups eller power-downs för att göra spelet mer utmanande och intressant. Spelplanen byggs genom filinläsning vilket också leder till att man enkelt kan bygga flera olika banor utan större hinder eller några ändringar i koden.

Meny och highscore hann jag tyvärr inte med, men dessa saker kan nog läggas till utan några större modifikationer av nuvarande kod. Det som jag kommer ta med mig mest från denna kurs är vikten av en bra design. Om man inte lägger den tid som behövs för att designen ska hålla kommer själva implementationen av mjukvaran vara fylld med problem som man lätt kunde ha undvikat ifall mer fokus lades på att ha en bra bas med sin design.