Individuell rapport, TNM094

Daniel Olsson
Flerreglarsspel

5 februari 2018

Sammanfattning

En sammanfattning ska kort och koncist beskriva och motivera det studerade problemet, metoden samt resultat och slutsatser. Arbetets bidrag till huvudområdet ska tydligt framgå. Vad är det rapporten säger om huvudområdet som vi inte visste tidigare? Sammanfattningens längd växer med längden på rapporten. I en rapport av denna typ kan den vara tre stycken lång: ett inledande motiverar arbetet och beskriver bakgrund, ett beskriver redogörelsen och ett beskriver analys och slutsatser. Sammanfattningen innehåller inga referenser eller ekvationer

Innehåll

Figurer

Tabeller

Kapitel 1

Inledning

Processen för att utveckla ett system är ofta komplex och lång. Det finns därför många olika metodiker som utvecklare kan jobba enligt för att få processen att bli mer överblickbar, redundant och anpassningsbar.

1.1 Bakgrund

Olika arbetsprocesser lämpar sig för olika sorters av projekt och det är därför viktigt att analysera varje projekt var för sig för att se vad som lämpar sig bäst. En arbetsprocess som inte lämpar sig för ett projekt kan få ödesdigra effekter i form av overhead[1] och kanske till och med ett nedlagt projekt.

Förutom arbetsprocessen så är systemets arkitektur viktigt för hur effektivt ett system är och hur lätt det är att underhålla det efter flera år.

1.2 Syfte

Syftet med detta arbete är att analysera och rekommendera en passande utveckling plan för ett projekt. Detta innebär att ge förslag på utvecklingsmetodik, systemarkitektur och projekthantering. Planen är en rekommendation och kan ändras under projektets gång om planen inte fungerar eller om ett bättre plan uppstår.

Projektet innebär att slutprodukten skall vara ett spel där flera externa knappar och spakar används för att styra och kontroller olika aspekter av spelmiljön. Detta är för att öka integration mellan barn i en fysisk miljö. Spelet kommer gå ut på att ta sig från punkt A till punkt B med hjälp av olika spakar och knappar som förändrar miljön som möjliggör en passage för spelarobjektet.

1.3 Frågeställning

För att kunna analysera vilken arbetsprocess som passar utvecklingsteamet bäst måste det definieras vad som är önskvärt. Det som eftersträvas är minskad tidsåtgång, högre kvalitet och att all funktionalitet på slutprodukten fungerar. Detta kan formuleras med följande frågeställning:

- Vilken utvecklingsmetodik ger minst overhead?

[1] Overhead - Tid som går åt till administrativa uppgifter eller andra indirekta kostnader.

- Finns det en spelmotor som kan sammankopplas med externa knappar och spakar?
- Vilka skäl finns det att utnyttja refaktorering[2] istället för planering?

1.4 Avgränsningar

Inga avgränsningar har gjort för detta arbetet.

[2]Refaktorering - Förbättra redan skriven kod så att funktionaliteten är samma fast koden fungerar bättre[1].

Kapitel 2

System och tekniska lösningar

I detta kapitel diskuteras tekniska lösningar för projektet och tredjepart programvara.

2.1 Grundläggande, initiala krav och systembegränsningar

Slutprodukten skall vara ett spel som skall vara lärande för barn men samtidigt kul och utmanande. Spelet skall stå i en utställning på visualations center i Norrköping och detta ställer en del krav på systemet. I samarbete med kunden tog en kravlista fram över slutprodukten vilket kan ses i sitt fullo i bilaga A. De viktigaste punkterna som definerar slutprodukten kan ses nedan.

Slutprodukten skall:

- minst använda sig av 3 externa knappar och 1 extern spak.
- kunna brukas av en sexåring.
- kräva minst två spelare för att vara spelbart.
- ha rymd eller programmeringstema.
- gå att montera för en normalteknisk person på ca 2h.

Visa av kraven ställer ett undre krav men inget övre krav och detta är för att ge utvecklingsteamet lite friheter för implementationen samt möjligheten för skalning av slutprodukten.

2.2 Målplattform

Projektet utvecklas för Windows 10 och ingen vikt kommer ges till att få slutprodukten att bli bakåt kompatibelt. Detta är för att slutprodukten är en del av en utställning och kommer därmed levereras med tillhörande hårdvara inklusive dator.

Utvecklingsteamet kommer använda sig av en extern spelmotor. Detta är på grund av projektets storlek inte gör det hållbart att utveckla en spelmotor och utvecklingsteamet har inga framtida planer på fler spel så kostnaden för utvecklingen av en extern spelmotor kan inte delas på flera projekt. De spelmotorer som är intressanta är:

- Unreal Engine 4

- Unity3D

Valet mellan spelmotorerna grundar sig i en fundamental grundpelare som måste uppfyllas vilket är att motorn kan hantera externa input från hårdvara. Då båda motorerna klarar detta så valdes Unity3D då dess inlärningskurva är mycket bättre och det genererar en snabbare start i projektet.

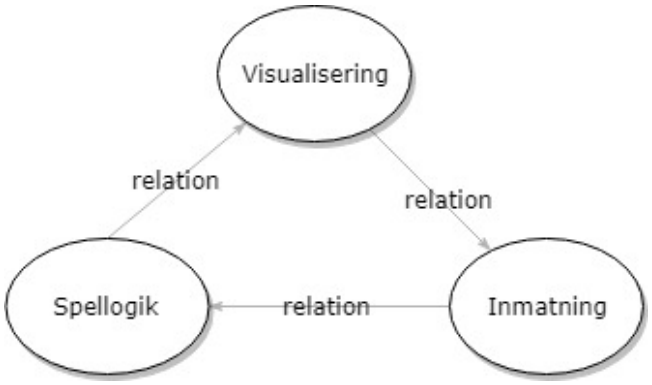
2.3 Grundläggande system-arkitektur

Systemet som skall utvecklas kan grovt delas in i 3 delar:

- Visualisering
- Inmatning
- Spellogik

Visualerings objektet är allt som generar en bild på skärmen, inmatnings objektet hanterar inputs från användaren och spellogiks objektet bestämmer om inmatningen skall hanteras eller kastas.

Flödet som sker mellan dessa är sker alltid framåt. Visualisering ger data till användaren som ger nya kommandon via inmatningen som sedan behandlas av spellogiken och därefter uppdaterar visualiseringen. Detta system liknar strukturen för modell view controller(MVC)[2] och det är den strukturen som används för detta system på den högsta nivån.



Figur 2.1: Systemets tre objekt och deras relationer till varandra

Fördelar med denna struktur är att det är enklare att hitta fel i systemet och programkoden blir enklare att läsa då koden blir mer linjär och flödet är alltid riktat åt ett håll så som ses i 2.1.

Visualiserings och spellogik objekten byggs upp i Unity3D och skrivs i C-sharp. Inmatnings objekten kommer handskas både i Untiy3D och en arduino. På arduinon skrivs koden i C och kommer endast hantera de externa kontrollernas

2.4 Standarder

Det finns inga speciella standarder som behövs följas.

2.5 Utvecklings-miljö

I projektets utvecklingsfas kommer flera olika verktyg användas för att bygga och hantera slutprodukten. Dessa verktyg och dess användingsområde är samlade i tabell 2.1.

Tabell 2.1: All programvara som används för att utveckla systemet

Programtyp	**Programnamn**	**Användningsområde**
Spelmotor	Unity3D	Använd som grafikmotor och nivåskapare.
3D Modellering	3DS Max	Används till att skapa alla 3D modeller
Bildbehandlings-program	3DS Max	Används till att skapa texturer
Utvecklingsmiljö	Arduino Software	Används som utvecklingsmiljö när program skrivs till Arduinon
Versionshantering	Git	Används till att hålla koll på tidigare versioner och underlätta samkodning
Projekthantering	Hack n plan	Används till att hantera projektets olika delar och moment.

Kapitel 3

Projekthantering

Olika projekthanteringsmetoder ger olika fördelar och nackdelar.

3.1 Utvecklingsmetodik

Från kravspecifikationen som ses i bilaga A kan det ses att kunden inte har satt speciellt hårda krav på vad som slutprodukten skall innehålla vilket genererar att utvecklingsteamet har stor möjlighet att påverka vad innehållet kommer vara. Då bra idéer kan uppstå när som så är det bra och ha en utvecklingsmetodik som är flexibel. Därför har agil utveckling valts till detta projekt.

3.1.1 Agil utveckling

Agil utveckling kallas ofta för lättrörliga metoder eller iterativa metoder och menas att varje projekt bryts ner i mindre delar som kan utvecklas var för sig. Detta gör att det alltid finns fungerande programvara efter varje iteration. Inom agil utveckling följer man 4 st riktlinjer[3]:

- Värderar individer och interaktion över processer och verktyg.
- Värderar fungerande programvara över omfattande dokumentation.
- Värderar kundsamarbeten över kontraktsförhandlingar.
- Värderar att svara på förändring över att följa en plan.

Detta kan sammanfattas med att agil utveckling är värdedrivande, dvs att utvecklingsteamet skall prioritera det som ger värde till slutprodukten.

En teknik som förespråkas inom agil utveckling är par programmering[3]. Par programmering går ut på att två programmerare sitter tillsammans och skriver kod. Den som skriver kallas föraren och den andra programmeraren kallas navigatören. Under tiden föraren skriver kod så granskar navigatören och pekar ut fel och ger råd till föraren. Detta generarar att fel kan upptäckas tidigt i processen och att koden blir mer läsbar direkt. Par programmering kommer att användas vid de viktigaste processerna så som koden för att kontrollera de externa kontrollerna.

3.1.2 Scrum

Beskriv och motivera här den övergripande strukturen, utvecklingsmetodiken eller metodikerna, som valts för projektet. Här ska bara grundläggande principer och motivering diskuteras. Detaljer diskuteras och motiveras under respektive avsnitt.

3.2 Organisation

Detta projekt har tillgång till 20st utvecklare under 12 månaders tid för att färdigställa slutprodukten enligt kravspecifikationen i bilaga A. Förutom dessa 20st utvecklare så finns även en projektansvarig som är överst ansvarig för alla utvecklarna.

Dessa 20st utvecklare är indelade i 4 grupper med olika arbetsuppgifter. Uppdelningen kan ses i tabell 3.1.

Tabell 3.1: Fördelningen mellan utvecklare i projektet och dess arbetsuppgifter

Utvecklingsområde	**Antal**	**Förklaring**
Spellogik	8st	Ansvarar för att all spellogik finns och att de externa kontrollerna fungerar med spelet.
Modellering och nivåeditering	6st	Ansvarar för att skapa alla 3D modeller och texturer till spelet. Sätter även ihop alla nivåer.
Ljud	3st	Ansvarar för alla
Manus	3st	Ansvarar för att skapa manus.

Projektansvariga roll är att se till att alla utvecklare samarbetar och ligger i fas med varandra samt hålla möten med kunden och fördela krav vidare till arbetsgrupperna.

3.3 Tidsplan

Beskriv, diskutera och motivera tidsplan för hela projektet, sprints, möten och milestones, inklusive tid för planering och leverans.

3.4 Milestones och leverabler

Mer detaljerad beskrivning och motivering av milestones och leverabler, såsom rapporter, prototyper och färdigt system.

Kapitel 4

Rutiner och principer

För att öka produktiviteten och samarbetet mellan alla utvecklare sätts rutiner upp. Dessa skall hjälpa att all kod och dokument följer ett särskilt mönster så att det inte blir misskommunikation mellan utvecklare.

4.1 Mötesprinciper och rutiner

Det kommer ske tre olika sorters möten och de är veckomöte, sprintmöte och scrummöte.

Veckomöte är ett möte som infaller varje måndag klockan 08:15 för att säkerställa att all vet vad de ska göra den kommande veckan samt att påminna om eventuella kundmöten. Detta mötet skall ta ungefär 30 min.

Under sprintmötet bestämts vad som skall ske under nästa sprint och vad som skall färdigställas. Arbetsuppgifter delas ut bland de ansvariga utvecklarna som sedan delar ut de till sina utvecklargrupper. Detta mötet är beräknat till ca 60 min.

Scrummöterna sker i varje utvecklingsgrupp och där tas problem upp och diskuteras hur långt alla har kommit med sprinten. Detta mötet är beräknat till ungefär 5 min.

4.2 Kravhantering och -sårning

Beskriv hur gruppen arbetar med kravhantering och -spårning och vilket teknikstöd som kommer att användas. Här beskrivs också hur projektet säkerställer synkronisering mellan intressenternas behov och genomförandet.

4.3 Versionshantering, -system och rutiner

Versionhantering är en viktigt pelare i ett projekt för att enklare kunna se vem som gjorde vad och gå tillbaka till gamla versioner. I detta projekt kommer Git användas som versionhanteringsprogram tillsammans med en extern server. Den externa serven kommer ligga på Gitlab. Valet att ha den externa servern på Gitlab grundar sig på tidigare licenser.

För att undvika konflikter och trasig kod så skall några få riktlinjer följas:

- Alla commitments skall kommenteras kort. Nya tillägg skall kommenteras vad de gör och änd-

ringar på gammal kod kommenteras vad de har ändrats.

- Master branchen skall alltid innehålla fungerande kod, alla ändringar görs på en egen branch som sedan mergar in till mastern när den fungerar.
- Alla namn på branches skall innehålla i namnet vad som utvecklas och namn på ansvarig utvecklare. Stilen på namnet är *Process_Namn* och ett exempel kan se ut *WrapperInput_Daniel*.
- När en branch är fungerande skall alla test tas bort och branchen mergas in i mastern och sedan skall alltid branchen raderas om allt arbete på den är klar.
- Branches på branches behöver inte ha en ansvarig utvecklare i namnet.

Dessa punkter är riktlinjer och skulle ett special dyka upp så skall detta diskuteras på ett möte.

4.4 Arkitektur- och programdesign, standarder och rutiner

Beskriv hur gruppen arbetar med arkitektur och programdesign.

4.5 Dokumentationsprinciper och rutiner

Dokumentation är en viktig del för att underlätta förståelse mellan utvecklare. Nedan beskrivs riktlinjerna för dokumentation.

4.5.1 Programkod

Programkod skall kommenteras på två olika ställen. En kommentar i toppen på varje fil och en över varje funktion. Kommentaren i toppen på varje fil skall innehålla:

- Namnet på ansvarig utvecklare. Det menas med utvecklaren som är ansvarig för området och inte utvecklaren som har skrivit koden.
- Eventuella licenser
- Evuentuella tredjeparts-APIer som används.
- Kort sammanfattning vad filen gör.

All kommentering skall ske i Engelska.

4.5.2 Möten

Alla veckomöten skall dokumenteras enligt en fördefinerad mall som kan ses i bilaga B. De dagliga scrum mötena dokumenteras kortfattat i en textfil som lägg på samma server som all kod och hanteras av git. All dokumentation från möten sker på svenska.

4.5.3 Slutproduktion

Slutprodukten som är ett spel kommer dokumenteras på två sätt. En dokumentation kommer ske direkt i spelet iform av en introduktion till spelet och den andra dokumentationen är en manual hur slutprodukten skall kopplas samman.

Indroduktionen i spelet kommer skrivas direkt i Unity3D och skall fokusera på att ge en grafisk handledning istället för text. Informationen ges på de språket som användaren har valt att spela i.

Manualen kommer skrivas i Latex och kommer innehålla en stegvis beskrivning på hur alla tillbehör sätt tillsammans samt hur slutprodukten installeras. Felsökning kommer även inkluderas i manualen. Manualen skrivs på Engelska.

4.6 Kvalitetssäkring

Beskriv hur gruppen arbetar med kvalitetssäkring av programkod (granskning och testning) och vilket teknikstöd som kommer att användas.

Kapitel 5

Analys och diskussion

Det är här ni analyserar och diskuterar arbetet. I diskussionskapitlet ska man explicit referera till både andra avsnitt i rapporten och externa källor som är relevanta för diskussionen.

5.1 Resultat

Finns det något i resultaten som står ut och behöver analyseras och kommenteras? Vad säger teorin om vad resultaten egentligen betyder? Finns det något i resultaten som är oväntat baserat på teori och andra källor, eller stämmer det bra överens med vad man teoretiskt kunde förvänta sig?

5.2 Arbetet i ett vidare sammanhang

Det ska ingå ett stycke med en diskussion om etiska och samhälleliga aspekter relaterade till arbetet. Detta är viktigt för att påvisa professionell mognad samt för att utbildningsmålen ska kunna uppnås. Om arbetet av någon anledning helt saknar koppling till etiska eller samhälleliga aspekter ska detta explicit anges i stycket Avgränsningar i inledningskapitlet. Exempel på samhälleliga eller etiska aspekter är människors liv och hälsa, samhällets funktionalitet, demokrati, rättssäkerhet och mänskliga fri- och rättigheter, miljö och ekonomiska värden samt nationell suveränitet. Inom planering av/och systemutveckling kan det exempelvis handla om arbetsbelastning, ekonomisk kompensation, uppdelning mellan arbetstid och fritid, könsdiskriminering eller annan form av diskriminering, programvarans användning i terrorism, kärnvapen, miljökopplad industri eller demokratisk utveckling.

Kapitel 6

Slutsatser

I detta kapitel ska en återkoppling till syfte och frågeställningar ske. Har syftet uppnåtts och vad blev svaret på frågeställningarna? Varje frågeställning kan få ett eget avsnitt för att tydliggöra strukturen. Här ska också arbetets konsekvenser för berörd målgrupp och eventuellt för forskare och praktiker beskrivas. Man kan också ha ett stycke eller avsnitt om framtida arbete där man beskriver vad man skulle vilja göra om man hade mer tid eller som rekommendationer för framtida studier eller exjobb. Om man har ett sådant stycke är det dock viktigt att det är konkreta och väl genomtänkta förslag som presenteras, snarare än vaga idéer

Litteraturförteckning

[1] Fowler M, Beck K, Erich G. Refactoring : improving the design of existing code. Boston: Addison-Wesley; 2000. Available from: https://www.csie.ntu.edu.tw/~r95004/Refactoring_improving_the_design_of_existing_code.pdf.

[2] Gamma E. Design patterns : elements of reusable object-oriented software. Reading, Mass.: Addison-Wesley; 1995.

[3] Sewell L. Agile software development [Elektronisk resurs]. Delhi: Research World; 2012.

Bilaga A

Kravspecifikation

Krav nr 1	Systemet skall minst använda sig av 3 externa knappar och 1 extern spak.
Krav nr 2	Systemet skall vara engagerande.
Krav nr 3	Systemet skall kunna brukas av en sexåring.
Krav nr 4	Systemet skall kräva minst två spelare för att vara spelbart.
Krav nr 5	Systemet skall inte ge negativa responser vid fel utan ge konstruktiv kritik till användaren.
Krav nr 6	Systemet skall inte innehålla ett traditionellt poängsystem.
Krav nr 7	En användare skall kunna lära sig kontrollerna för systemet under 2 minuter.
Krav nr 8	Systemet skall öka svårighetsgrader så att det är svårt att bemästra systemet.
Krav nr 9	Systemets skall ha rymd eller programmeringstema.
Krav nr 10	Systemet får inte manipulera projektionsvägen.
Krav nr 11	Systemet skall använda grafiska hjälpmedel för att visa knappar och spakars funktionalitet.
Krav nr 12	Systemet skall gå att montera för en normalteknisk person på ca 2h.
Krav nr 13	Systemet skall kunna användas dagligen under 2 år utan att behöva repareras
Krav nr 14	Systemet skall kunna skalas om till rum i olika storlekar. Minsta storlek på rum är 2x2 meter.

Bilaga B

Protokoll mall

Bilaga C

Ordlista

- Overhead - Tid som går åt till administrativa uppgifter.