

Medusa - 3D-rekonstruktion

Projektrapport, TNM094

Caroline Gard
Isabell Jansson
Cecilia Lagerwall
Johan Reimann
Erik Sandréen
Pelle Serander
Carl Englund

3 juni 2015

Sammanfattning

Denna rapport behandlar arbetet inom ett projekt i kursen Medietekniskt Kandidatprojekt, TNM094 på Linköpings universitet. Projektets mål var att utveckla ett 3D-scannerbås där en användare har möjlighet att scanna sig själv för att sedan erhålla en solid 3D-modell med färgtextur. Rapporten omfattar huvudsakligen projektets utvecklingsprocess, metodval, verktyg, planering samt resultat.

Projektet har utvecklats agilt med arbetsmetoden *scrum*. Scrum är en flexibel arbetsmetod där utvecklingsteamet arbetar i *sprintintervall*. Varje *sprint* har pågått under två veckor där utvecklingsteamet har arbetat med planerade *tasks* som bygger på kundens krav på systemet. Varje sprint utvärderades vid avslut för att kontinuerligt kunna förbättra arbetsprocessen och lösa eventuella problem.

Scanningen använder sig av tre stycken *Microsoft Kinect-kameror för Xbox 360* och styrs av användaren via en enkel *webbapplikation* som är placerad inuti båset. Webbapplikationen kommunicerar med systemet via en *websocketserver*. Systemet är konstant aktivt och inväntar anrop från webbapplikationen som sker när användaren startar scanningen från användargränssnittet. Den färdiga modellen sparas i en databas. För att komma åt sin modell kan användaren välja att få ett e-postmeddelande innehållande en länk till sin modell. För att säkerställa att användaren endast kommer åt sin egna modell innehåller filnamnet en krypterad sträng.

Utvecklingsteamet har uppfyllt kundens krav för systemet, att skapa en 3D-scanner där användaren erhåller en solid modell av sig själv. Användaren kan även starta systemet via ett enkelt gränssnitt, vilket också var ett krav. Systemet är fortfarande en prototyp och innehåller ett flertal utvecklingsmöjligheter för att betraktas som ett färdigt system.

Arbetsprocessen och verktygen har utvecklingsteamet tagit fram tillsammans efter vad teamet har kompetens för och vad som anses vara mest lämpligt för projektet. Arbetet har lagts upp på det sättet att kundens krav uppfylls under den utsatta tidsperioden.

Innehåll

Sammanfattning	i
Figurer	iv
Tabeller	v
Typografiska konventioner	vi
1 Inledning	1
1.1 Bakgrund	1
1.2 Syfte	1
1.3 Frågeställning	1
1.4 Avgränsningar	2
2 Relaterat arbete	3
2.1 Microsoft Kinect för XBox 360	3
2.1.1 Kalibrering av färgkamera och IR-kamera hos Kinect	3
2.1.2 Positionskalibrering av Kinect-kamerorna	3
2.2 Ytrekonstruktion	4
2.3 Gränssnitt	5
2.3.1 Websockets	5
2.3.2 MongoDB	6
2.3.3 DART	6
3 Arbetssätt	7
3.1 Arbetsmetoden Scrum	7
3.1.1 Sprints och möten	9
3.2 Kravhantering	9
3.3 Dokumentation	10
3.3.1 Textdokument och bildfiler	10
3.3.2 Koddokumentation	10
3.4 Versionshantering	10

3.5 Testning	11
4 Redogörelse för arbetet	12
4.1 Systemarkitektur	12
4.2 Systemets delar	13
4.2.1 Scanning	13
4.2.2 Bildbehandling	13
4.2.3 Gränssnitt	15
4.2.4 Websockets	15
4.2.5 MongoDB	15
5 Resultat	16
5.1 Scanning	16
5.2 Gränssnitt	16
5.3 Bildbehandling	17
6 Analys och diskussion	21
6.1 Metod	21
6.1.1 Scanningsprocessen	21
6.1.2 Ytrekonstruktion	22
6.1.3 Gränssnittet	22
6.2 Resultat	22
6.3 Arbetet i ett vidare sammanhang	22
6.3.1 Samhälleliga aspekter	22
6.4 Problem som har uppstått under projektet	23
7 Slutsatser	24
7.1 Svar på frågeställningar	24
7.2 Framtida förbättringar	25
7.3 Konsekvenser för berörd målgrupp	25
Litteraturförteckning	26
A Involverade parter	27
B Statistik från github	28
C Gantt schema	29

Figurer

2.1	Processflödet vid ytrekonstruktion	4
3.1	Planering av scrum med hjälp av Trello	8
4.1	Ursprungliga systemarkitekturen	12
4.2	Den nya och utvecklade systemarkitekturen	13
4.3	Kinect-kamerornas position.	14
5.1	Gränssnittet när modellen presenteras för användaren	18
5.2	Gränssnittet när modellen presenteras för användaren	19
5.3	Bilder från ytrekonstruktion	20

Tabeller

1	Typografiska konventioner	vi
3.1	Roller inom utvecklingsteamet	8

Typografiska konventioner

Nedan listas olika typografiska konventioner, se Tabell 1.

Tabell 1: Typografiska konventioner

Teckensnitt	Förklaring	Exempel
ABCabc123	Tekniska termer, först förekommande	Projektet utvecklades med en buildserver
ABCabc123	Programvarubibliotek	För kalibreringen användes OpenCV
<i>ABCabc123</i>	Engelsk term	Inför varje <i>sprint</i> hölls ett möte

Kapitel 1

Inledning

Möjligheten att återskapa objekt från verkliga världen i en virtuell värld har varit ett intresseområde de senaste 30 åren i både underhållningssyfte och för informationsvisualisering[1]. Det är en process som tidigare krävt kompletteringar av mänskligt arbete. Med hjälp av ny teknik och mer datorkraft, har denna process kunnat automatiseras till den grad att det som förut skulle kunna ta en vecka nu kan ske på några sekunder. Hårdvaran som har använts har utvecklats genom åren och är nu tillgänglig för alla. Detta, i kombination med utvecklingen av *3D-skrivare*, har lett till att intresset för att återskapa objekt blivit ett populärt ämne.

1.1 Bakgrund

I kursen “Medietekniskt kandidatarbete”, TNM094, fick ett utvecklingsteam bestående av sju personer i uppgift att utveckla ett mjukvarusystem för ett 3D-scannerbås för att skapa en modell av användaren i färg. Båset skulle innehålla flera djupkameror för att scanna användaren. Den erhållna punktdatan skulle användas för att skapa en solid tredimensionell modell. Användaren skulle även ha möjlighet att interagera med systemet via ett enkelt gränssnitt. Gränssnittet skulle ha som användningsområde att starta scanning, spara- och kasta modell och göra om scanningsprocessen. En stor del av uppgiften var att arbeta med en agil utvecklingsmetod. Fokus för projektet har legat på att utveckla programvaran till ett fysiskt scanningsbås.

1.2 Syfte

Syftet med rapporten är att framlägga målet med projektet och beskriva arbetsmetoder och verktyg som har behandlats under projektets gång. Syftet är även att ge en tydlig reflektion över metodval för arbetsprocessen och undersöka dessa metoder i verlig miljö. Samtliga medlemmar i projektet ska utveckla en förståelse för teoretiska metoder inom systemutveckling.

1.3 Frågeställning

Nedan listas relevanta frågeställningar berörande projektet:

- Hur kan ett användande av andra generationens Kinect skapa en 3D-modell med tillräckligt god kvalité, utan komplicerade instruktioner och avancerade inställningar?

- Vilka krav kan ställas på systemets justerbarhet, för att anpassa systemet efter varje individuell användare, och därmed kunna scanna och rendera en korrekt bild?
- Vilken eller vilka metoder ska systemet använda sig av för att reducera mängden brus i den insamlade punktdatan?
- Hur kan utvecklingsteamet, utifrån undersökningar av alternativa sätt för insamling av data, finna den bästa lösningen med avseende på kvalitén hos 3D-modellen och vad är för- respektive nackdelarna med de framtagna lösningarna?

1.4 Avgränsningar

Under planeringen inför projektet diskuterades vilka avgränsningar för systemet som skulle göras. En begränsning som bestämdes var att scanningen endast avser en person åt gången och är inte anpassad till flera objekt. Detta på grund av att risken för felaktiga tolkningar under scanningen ökar och därmed kan hål i modellen uppkomma. Detta skulle i sin tur förhöja risken för deformation av den resulterande solida modellen.

En annan avgränsning för systemet var att ingen automatisering av 3D-utskrift för modellen skulle implementeras i systemet. Dagens 3D-skrivare har inte tillräcklig snabb utskrift för en acceptabel väntetid. Däremot ska modellen ha ett format för att användaren själv ska ha möjlighet att skriva ut modellen på egen hand.

Kapitel 2

Relaterat arbete

2.1 Microsoft Kinect för XBox 360

En *Kinect-kamera* består av en färgkamera, en IR-kamera¹ samt en projektör. Projektorn sänder ut punkter som fångas upp av IR-kameran. Punkterna jämförs sedan med ett referensmönster som erhålls genom att Kinect-kameran skapar ett plan på ett känt avstånd från kamerasensorn. Planet sparas sedan i Kinect-kamerans sensorminne. Genom att jämföra om varje punkt är placerad framför eller bakom planet kan Kinect-kameran avgöra vilka punkter som är synliga eller gömda bakom olika föremål. Denna process utförs automatiskt av Kinecten. För bästa resultat bör det inscannade objektet befina sig inom ett område av en till tre meter från sensorn, vilket resulterar i mindre brus och bättre upplösning av djupdatan[2].

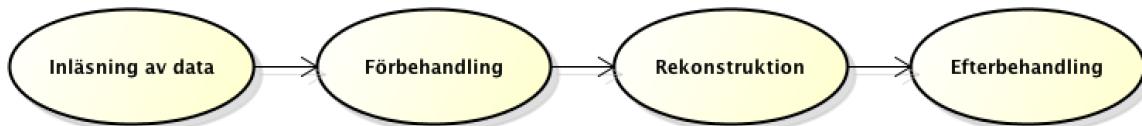
2.1.1 Kalibrering av färgkamera och IR-kamera hos Kinect

Eftersom färgkameran och IR-kameran inte har exakt samma position i en Kinect-kamera behöver dessa kameror kalibreras. Utan kalibrering kommer texturen att projiceras på modellen inkorrekt[2]. Båda kamerorna i Kinecten kalibreras samtidigt genom att använda en bild av ett schackmönster. Båda kamerorna måste sedan extrahera hörnpunkterna i schackmönstret, vilket till exempel kan utföras med hjälp av biblioteket **OpenCV**. Dessa punkter sparas sedan för att ta fram en *translationsmatris* som beskriver sambandet mellan djupbilden från IR-kameran och bilden från färgkameran. Translationsmatrisen används för att förflytta färgbilden innan den projiceras som textur på objektet. Utöver translationsmatrisen erhålls även koefficienter som beskriver den radiella och tangentella distorsionen. [3].

2.1.2 Positionskalibrering av Kinect-kamerorna

För att flera kameror skall kunna användas behöver deras position utifrån ett gemensamt origo vara känt. Genom att använda den schackrutiga bilden som ovan för att placera ut origo kan translationsmatrisen och distortionskoefficenterna användas för att skapa en vymatris för kameran. Denna matris kan användas för att transformera bilder och punkter mellan världskoordinater och kamerakoordinater. Positionskalibreringen möjliggör att projiceringen av färg-bilden sker utifrån korrekt vinkel och position på modellen[4].

¹Infraröd-kamera



Figur 2.1: Processflödet vid ytrekonstruktion

2.2 Ytrekonstruktion

För att rekonstruera en 3D-modell utifrån inläst punktdata används flera processer, se Figur 2.1. Det första steget innebär att systemet skapar eller läser in punkter, där inläsningen utförs till exempel med en IR-kamera. Sedan reduceras datan för att minska komplexiteten och brus filtreras bort. En yta rekonstrueras utifrån de kvarvarande punkterna. En utförligare förklaring av stegen följer nedan.

1. Inläsning av punktdata

Inläsning av de punkter som finns kan ske med till exempel en Kinect-kamera. Processen sker då enligt kapitel 2.1.

2. Förbehandling: Reducering av punktdata

Vid inläsning av punktdata via ett flertal Kinect-kameror erhålls punkter i en stor mängd, vilket medför att programmet blir beräkningstungt för datorn. Punkter placerade nära varandra kommer inte att tillföra någon märkbar kvalitetsförändring av modellen. Alla dessa punkter är därför inte nödvändiga för att erhålla en tillräckligt noggrann tredimensionell modell för ett bra utseende. Reducering av punktdata kommer att underlätta senare beräkningar för datorn. Det finns flera metoder för att utföra reducering av data och det är nödvändigt att avväga vad som viktigast, prestanda eller kvalité.^[5]

Voxelgrid-filter är en metod i biblioteket **PCL**² som används för reducering av punktdata. Med denna metod skapas ett tredimensionellt rutnät över den inlästa punktdatan vilket bildar kuber, voxels, utmed ytan där samtliga punkter är uniforma och har samma massa. Punkter som befinner sig i samma voxel används sedan för att uppskatta en ny punkt baserat på tyngdpunkter hos varje punkt i voxeln. Punkterna nedsampelas därför i samtliga voxels och reduceras efter en approximation beroende på varje punkts placering. Metoden är snabb och ger en relativt hög upplösning av ytan hos modellen. Genom att minska storleken på varje voxel kan även noggrannheten förbättras, men programmet blir dock mer beräkningstungt. ^[6]

3. Förbehandling: Reducering av brus

Vid scanningen kan det uppstå felaktiga punkter med extremvärden som uppkommer på grund av mätningfel vid inläsningen. Detta beror på att IR-kameran registrerar felaktig data som exempelvis kan inträffa vid inläsning av partiklar i luften. Felaktiga eller fristående punkter tolkas som brus och måste avlägsnas för att senare kunna skapa en solid modell.

StatisticalOutlierRemoval filter är en metod för att hantera brusreducering och är en del av **PCL**. Brusreduceringen analyserar den statistiska sannolikheten för att en punkts position är relevant i förhållande till sin omgivning. För att identifiera skräppunkter, beräknas det genomsnittliga avståndet för varje punkt till de närmast intilliggande punkterna. Detta genomförs genom att använda både ett globalt medelavstånd samt en lokal standardavvikelse. Det globala medelavståndet har en förbestämd konstant definition medan standardavvikelsen beräknas med radien

²Point Cloud Library

för varje punkt. De punkter som antar ett större avstånd än beräknat medelvärde kan tolkas som skräppunkter och därmed tas bort från punktdatan[5].

4. Rekonstruktion

För att sammanfoga punkter till en solid modell, används triangulering[7]. Triangulering innebär att att punkterna omvandlas till vertexpunkter och att kanter och ytor bildar en solid polygonmodell. Det finns flera metoder för rekonstruktion som kan appliceras på punktdatan, dessa med både för- och nackdelar.

Vid rekonstruktionen kan vissa problem uppstå. Metoderna kan till exempel ha svårt att förutse normalriktning om datan har en hög brusnivå. Det kan även finnas områden i datan, exempelvis skuggade områden, som saknar information vilket medför problem hurvida modellen blir solid eller ej. För att hantera problemen finns två kategorier med metoder, globala metoder samt lokala metoder[8].

Globala metoder innebär att hela datamängden behandlas direkt vilket medför att rekonstruktionen för objektet sker på en gång. Globala metoder innehåller ofta implicita funktioner. Funktionerna är växande vilket skapar tät matriser, vilket betyder att de flesta värdena i matrisen är skilda från noll. Dessa matriser har även dåliga villkor som till exempel kan ge felaktiga resultat[8].

Lokala metoder behandlar rekonstruktionen för mindre delar av datan för sig. Hur delarna sedan byggs ihop är en uppskattning av ytan. Lokala metoder ger ett detaltrikare resultat men har en högre känslighet för brus [8].

Poisson Surface Reconstruction är en global metod där en ytterlinje skapas baserat på gradienten för en 3D-indicatorfunktion, som fås genom en integralrelation med punkterna. Funktionen är bitvis konstant vilket medför att ytan får obegränsat stora värden. För att undvika det appliceras ett *smoothing filter* på funktionen och grader beräknas för den nya funktionen. Användning av Poisson medför därför att följande steg, efterbehandling, inte behöver appliceras då de redan är inkluderat i metoden[5, 8].

5. Efterbehandling

Vid inläsning av punktdata via Kinect-kamerorna uppstår det små distansfel mellan vissa av punkterna. Dessa fel orsakar ojämnhet hos modellens yta, vilket måste korrigeras för att modellen ska se tillräckligt bra ut. En ojämnn yta kommer att ha normaler med spretiga riktningar, medan en jämnare yta har normaler som förhåller sig bättre till varandra.

En metod som kan användas till att jämma ut ytan, som består av XYZ-koordinater, är marching cubes. Med *marching cubes* sätts de punkter som är placerade längst ut till ett referensplan. Metoden grundar sig i att en vertexpunkt väljs ut och att man därifrån stegar sig fram till de närmaste vertexpunkter med hjälp av *smallest edge detection*[7].

2.3 Gränssnitt

2.3.1 Websockets

Websockets är en teknik som möjliggör tvåvägskommunikation mellan klient och serverapplikationer. Genom en websocket kan klienten skicka och ta emot information när som helst från en serverapplikation.

2.3.2 MongoDB

MongoDB är en *no-sql* dokumentdatabas som använder *BSON*, som är en binär variant av *JSON*³.

2.3.3 DART

DART är ett programmeringsspråk som är **open-source**, skalbart och anpassat för webbapplikationer.

³JavaScript Object Notation, en dokumenstruktur för javascript objekt

Kapitel 3

Arbetssätt

3.1 Arbetsmetoden Scrum

Projektet har utvecklats agilt med arbetsmetoden *scrum*. En agil arbetsmetod var att föredra eftersom det är en iterativ och flexibel arbetsprocess och scrum är en bra metod för arbete mellan utvecklingsteam och kund. Idén med scrum är att leverera bästa möjliga resultat genom att arbeta kreativt och effektivt för att attackera olika typer av problem. Utvecklingsteamet ska vara av den storlek att arbetet kan utföras effektivt med minsta möjliga antal utvecklare men ändå resultera i ett system av hög kvalité. Samtliga medlemmar i utvecklingsteamet ska arbeta under samma titel, det vill säga en platt hierarki. I och med detta kan varje enskild person i utvecklingsteamet hålla sina arbetsuppgifter rörliga mellan arbetsområdena.

En *produktbacklogg* användes för att lista relevanta *user stories* tillhörande projektet. *Stories* beskriver krav från kund och utvecklingsteam och användes därmed som kravspecifikation för produkten. Projektet delades upp i mindre arbetsperioder, *sprints*, där varje sprint pågick under två veckors tid. Tidpunkten för dessa sprints planerades in i ett *gantt-schema* som utvecklingsteamet använde under projektets gång, se bilaga C. En viktig del för projektteamet var att längden av en sprint var tillräckligt lång för att kunna uppnå ett resultat, utan att sprintens mål blev för komplext. Inför varje sprint genomfördes en sprint-planering där olika user stories valdes till den kommande sprinten. Varje user story, delades sedan upp i *tasks*, där en task var ett arbetsmål med tidsbegränsningen på högst en dag. För att organisera produktbackloggen och för att planera uppsatta tasks och sprints använde arbetsteamet *Trello*¹. Alla tasks lades från början i en lista märkt med ”*To Do*”. Efter arbetets gång flyttades de sedan till ”*Doing*”, ”*Under Review*” och ”*Done*”, se Figur 3.1. En ofärdig eller ej påbörjad task flyttades till kommande sprint, förutsatt att den ansågs ha tillräckligt hög prioritet för att färdigställas.

Utvecklingsteamet bestod av sammanlagt sju personer men för att effektivisera arbetet delades utvecklingsteamet in i tre mindre ansvarsgrupper; scanning, bildbehandling och gränssnitt. En beskrivning av ansvarsgruppernas huvudområden följer nedan:

Scanning

Gruppen scanning ansvarade för att ta in data från kamerorna, både djup och färg. Gruppen ansvarade även för kalibrering av position och kalibrering av färg- och IR-kamera.

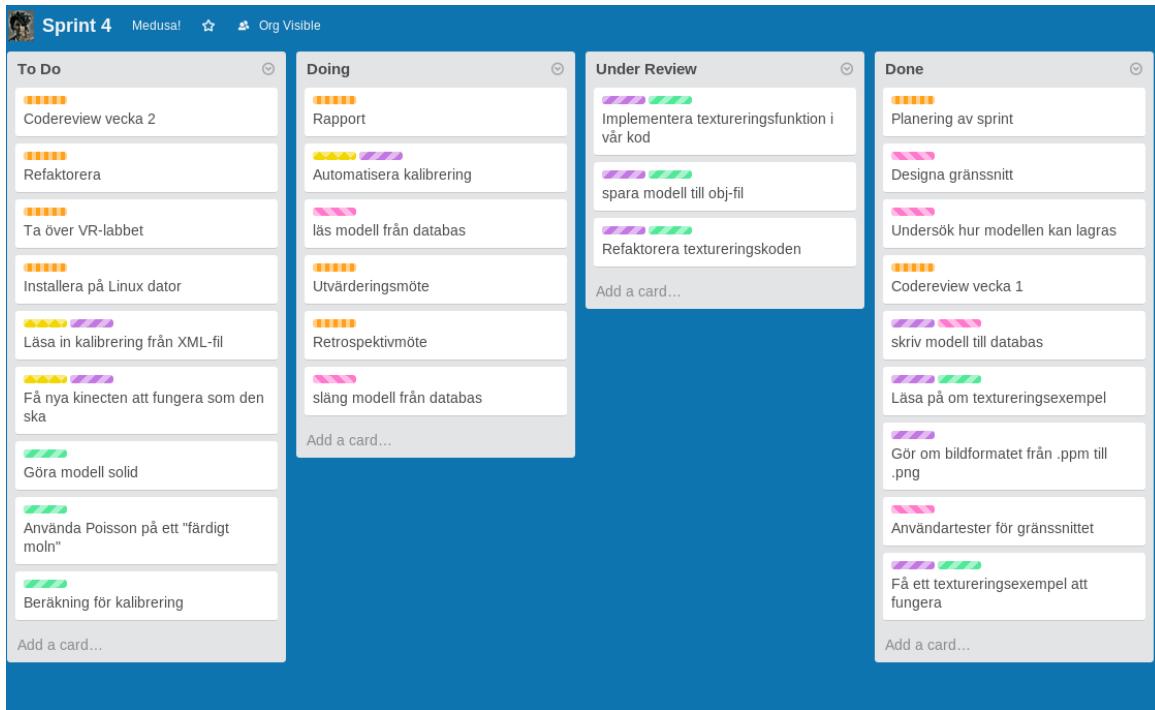
Bildbehandling

Gruppen bildbehandling ansvarade för att bygga en solid 3D-modell utifrån den data som kamerorna scannade in.

Gränssnitt

¹www.trello.com

Gruppen gränssnitt ansvarade för att bygga ett gränssnitt som användaren kan interagera med, samt att koppla ihop scanningssystemet med gränssnittet.



Figur 3.1: Planering av scrum med hjälp av Trello

Ansvarsgrupperna arbetade parallellt med olika tasks för att effektivt nå målen för projektet. Det var dock viktigt för utvecklingsteamet att vem som helst, oavsett ansvarsområde, skulle kunna sätta sig in i andra gruppars arbete.

Utvecklingsteamet tilldelades diverse roller enligt Tabell 3.1 nedan.

Tabell 3.1: Roller inom utvecklingsteamet

Produktägare/Kundrelation	Produktägarens ansvar var att produktbackloggen hölls uppdaterad och var prioriterad efter kundens önskemål. Produktbackloggen hade som krav att vara tydlig och förståelig för hela teamet. Som kundrelation var ansvaret att kontakta kunden vid behov.
Scrummästare	Scrummästaren såg till att arbetet följde scrum-metodens krav och riktsätt samtidigt som de dagliga stående mötena.
Gruppledare	Gruppledaren hade som uppgift att styra projektet i rätt riktning och såg till att samtliga gruppmedlemmar följde gruppkontraktet.
Rapportansvarig	Ansvarade för att rapporten påbörjades i god tid samt att den höll en hög standard.
Kodgranskningsansvarig	Ansvarade för att kodgranskning skedde löpande genom projektet.
Testansvarig	Ansvarig för att testning av gränssnittet utfördes och utvärderades.
Sekreterare	Sekreteraren ansvarade för att anteckna under varje möte samt att dokumentera uppkommen information.

3.1.1 Sprints och möten

Stående möte

Varje morgon inleddes med ett stående möte där hela utvecklingsteamet närvarade. Mötet hölls kort, ungefär tio minuter, och syftade på att sammanfatta gårdagens arbete samt dagens kommande arbete för varje enskild person i utvecklingsteamet. Mötet resulterade i att samtliga gruppmedlemmar fick en inblick i vad som förväntades utföras under dagen och en i de andras arbete. Det medförde även att de olika ansvarsgrupperna hölls uppdaterade om arbetet utanför den egna gruppen.

Utvärdering av sprint

Efter varje avslutad sprint hölls ett utvärderingsmöte. Under utvärderingsmötena diskuterade utvecklingsteamet resultatet av sprinten, vad som hinnits med respektive inte hinnits med, vad som kunde bortprioriteras och vad som kunde förbättras inför nästa sprint. Ett exempel på en åtgärd som utfördes efter ett utvärderingsmöte är att de tasks som hade varit för stora behövdes brytas ner till mindre tasks.

Retrospektivmöte

Retrospektivmöten skedde i slutet av varje sprint där det togs upp vad som hade gått dåligt respektive bra med arbetssättet under sprinten. Där kom utvecklingsteamet överens om det behövdes ändra någonting till nästkommande sprint och gav förslag på förbättringsmöjligheter som kunde utföras. Ett exempel på vad som förändrades tidigt i projektet var att de stående mötena flyttades bak från 09.00 till 08.30 för att teamet tidigare skulle kunna komma igång med arbetet för dagen och att alla skulle vara uppdaterade med vad andra gjorde.

Planeringsmöte

Ett planeringsmöte hölls en gång per sprint och syftade till att välja ut user stories från projektets produktbacklogg inför den kommande sprinten. Varje story delades sedan upp i mindre tasks och utvecklingsteamet diskuterade vad som skulle prioriteras. Utvecklingsteamet valde att planera flera tasks än vad som antogs hinna med och sedan arbeta med de högst prioriterade tasksen. Detta motsäger dock arbetsmetodiken *scrums* grundprinciper, som menar att en sprint ska innehålla de tasks som utvecklingssteamet tror sig hinna med.

Kodgranskning

Kodgranskningstillfället hölls i slutet av varje vecka, alltså två gånger per sprint. Målet med kodgranskningen var att säkerställa att all kod höll en hög standard, var välkommenterad och följde de riktlinjer om kodstandard som utvecklingsteamet hade beslutat om. Det var viktigt för utvecklingsteamet att alla ansvarsgrupperna närvarade oavsett vilken kod som granskades, eftersom koden skulle vara förståelig oavsett vem som skrivit den.

3.2 Kravhantering

Efter första mötet med kunden fick utvecklingsteamet krav som slutprodukten skulle uppnå.

- Systemet ska vara tillräckligt automatiserat för att en icke insatt användare ska ha möjlighet att använda systemet.
- Ska kunna se modellen i gränssnittet.
- Modellen ska vara solid, alltså ett objekt som är slutet.
- Det ska vara möjligt för användaren att antingen kunna spara eller kasta modellen.
- Modellen ska vara redo för att kunna skrivas ut på 3D-skrivare.
- Kunden hade som önskemål att systemet skulle fungera på en dator med operativsystemet Linux.

Ut efter dessa krav gjordes en kravspecifikation, vilket teamets produktbacklogg inkluderar, med krav som går att mäta.

3.3 Dokumentation

3.3.1 Textdokument och bildfiler

Projektet dokumenterades löpande på en gemensam **Google Drive** för att alla i teamet skulle ha tillgång till samtliga dokument. Dokumentationen innehöll anteckningar från utvärderingsmöten, retrospektivmöten, kundmöten etc. På Google Drive sparades även bilder. Mötesanteckningar storterades efter datum samt vilken mötestyp de tillhörde.

3.3.2 Koddokumentation

Alla filer i projektet kommenterades enligt *Doxygen*² riktlinjer för dokumentation. Dokumentation autogenererades sedan av projektets **buildserver** för enkel åtkomst. Kommentarerna skulle vara tillräckligt tydliga för att en utomstående skulle kunna förstå syftet med bland annat funktioner. Alla funktioner kommenterades men även kommentarer för enskaka rader utfördes i funktionerna för att förtydliga användandet och syftet.

3.4 Versionshantering

Git användes för versionshantering under projektet. För att dela koden användes *Github* med välkomenterade *commits* om vilka ändringar som utförts vid varje *push*.

För att enkelt separera olika delar i projektet använde utvecklingsteamet olika grenar. Det fanns tre olika huvudgrenar förutom *develop* och *master*. Dessa var bildbehandling, scanning och DART-utveckling. Förutom dessa grenar skapades även separata grenar av alla utvecklare för att arbeta på mindre funktionaliteter. Dessa kunde sedan implementeras in i huvudgrenarna.

²www.doxygen.org

3.5 Testning

Testningen som har genomförts innefattar användartester på gränssnittet samt kodgranskningar för att säkerställa att en god kvalité hölls och koden följde de bestämda riktlinjerna. Kodgranskningarna underlättade även för olika ansvarsgrupperna att förstå varandras kod.

Jenkins³ har använts för *continuous integration* och har försökt bygga innehållet vid varje *push* till *develop*-, eller *master*-grenen. Om kompileringen misslyckades skickas ett felmeddelande till den ansvariga personen laddat upp filen på den påverkade grenen. Jenkins användes även för att autogenerera dokumentationen till projektet med hjälp av Doxygen.

För att säkerställa att gränssnittet var enkelt och intuitivt genomfördes användartester. Innan testet inleddes informerades testpersonerna med samma information som de hade fått om de hade varit på plats i båset. De blev alltså informerade om att de befann sig i ett 3D-scannerbås och att de hade en surfplatta framför sig. Testpersonerna fick sedan själva komma fram till vad som behövde göras för att utföra scanningen. Under tiden ombads de att beskriva sina tankar om både funktionaliteten och det grafiska. Även utvalda frågor ställdes till deltagarna. Totalt utfördes fem användartester på fem olika personer, vilket ansågs som tillräckligt. Gränssnittet uppdaterades sedan efter testpersonernas anmärkningar.

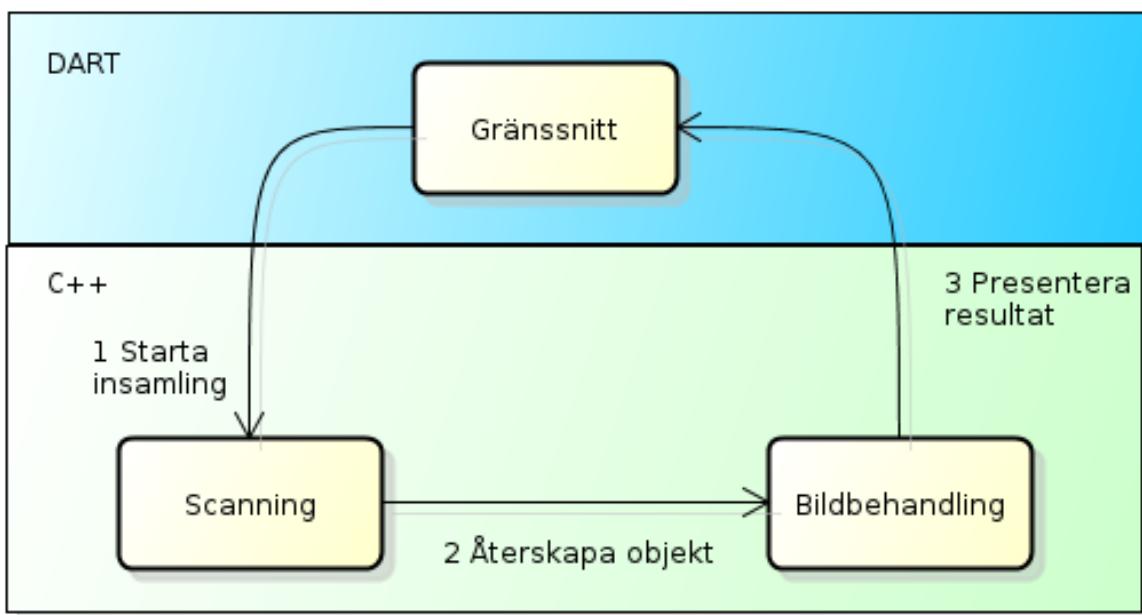
³www.jenkins-ci.org

Kapitel 4

Redogörelse för arbetet

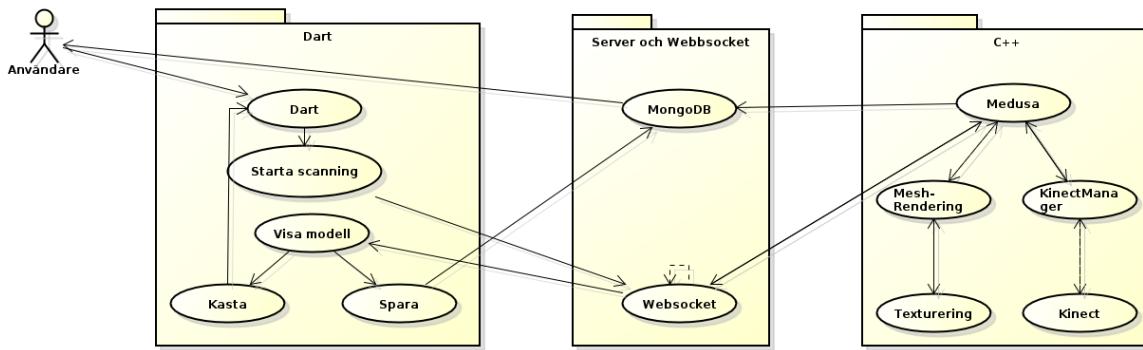
4.1 Systemarkitektur

Vid projektets start utfördes en enkel systemarkitektur för att säkerställa att utvecklingsteamet strävade mot ett gemensamt mål, samt för att på en abstrakt nivå veta hur systemet skulle se ut, se Figur 4.1.



Figur 4.1: Ursprungliga systemarkitekturen

Ovanstående systemarkitektur användes sedan som grund för projektet och utvecklades under processen och blev allt mer detaljerad, se Figur 4.2.



Figur 4.2: Den nya och utvecklade systemarkitekturen

4.2 Systemets delar

4.2.1 Scanning

Hårdvara

För att genomföra scanningen användes Microsofts Kinect-kameror för Xbox 360. En viktig fråga för utvecklingsteamet var hur insamlingen av data skulle hanteras där två möjliga alternativ fanns tillgängliga. Antingen kan kamerorna vara utplacerade på fasta bestämda platser. Metoden medför att flera kameror måste användas för en total täckning av det scannade objektet. Vid undersökning av att scanna en kropp visades att tre kameror, jämt utplacerade, var tillräckligt antal för denna metod. Det andra alternativet var att använda en kamera som sedan roterar runt objektet. Metoden avvisades då den ansågs svår att genomföra rent praktiskt. Eftersom det huvudsakliga målet inte var att skapa det fysiska systemet, valdes ett system med kameror med fasta positioner.

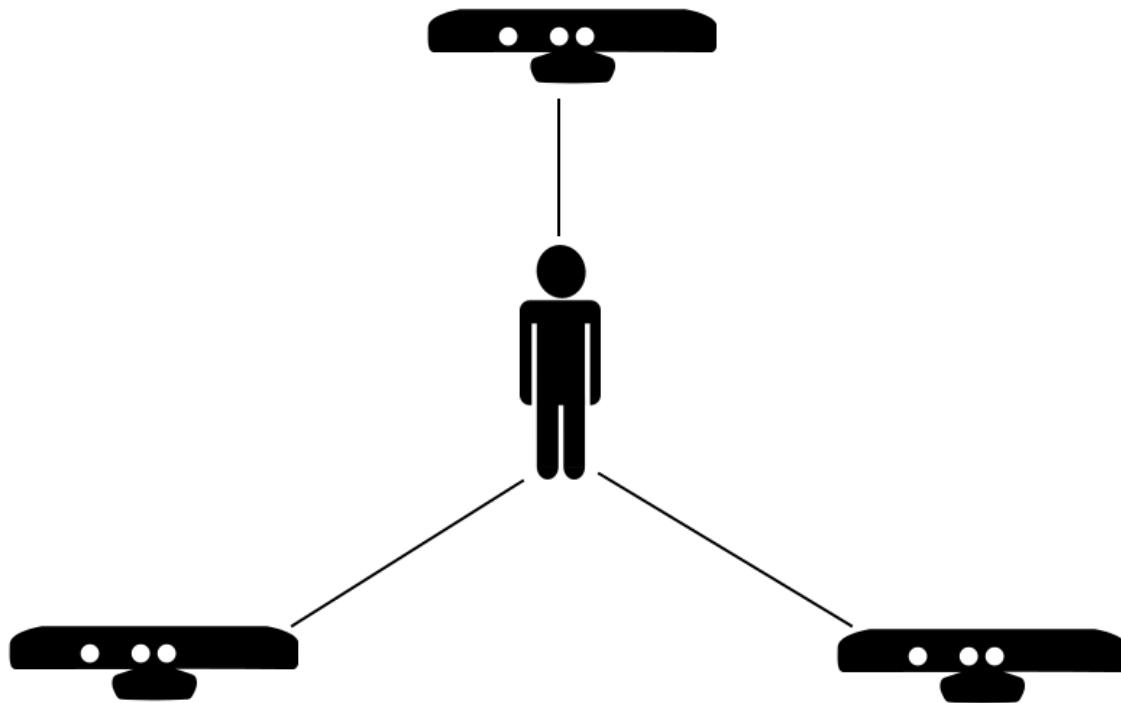
Kalibrering av kamerornas position

För att underlätta uppsättningen av det fysiska båset för kunden gjordes en kalibreringsfunktion av kamerornas position. Målet med kalibreringen var att kamerorna ska veta sin position i förhållande till ett gemensamt origo.

Vid en kalibrering måste först Kinect-kamerorna kalibreras enskilt för att ta reda på deras interna distorsionskoefficienter och kameramatris. Detta sker genom att låta alla kameror ta bilder kontinuerligt och försöka hitta schackbrädet, som tidigare nämnts, i tio stycken individuella bilder. Utifrån dessa tio bilder kan koefficienter och en kameramatris tas fram för att sedan kalibrera Kinect-kamerornas position. En funktion för att sedan få alla Kinect-kameror att ta en bild samtidigt på schackbrödet skapades. Dessa bilder kunde användas för att skapa ett gemensamt origo för alla Kinect-kameror [4].

4.2.2 Bildbehandling

För bearbetning av djupdata användes **PCL**. **PCL** är en samling av moderna algoritmer och metoder för att hantera 3D-data för både företag och privatpersoner. Bibliotekets uppbyggnad är modulbaserat vilket innebär att det är ett relativt enkelt att hantera metoder för önskad funktionalitet [9].



Figur 4.3: Kinect-kamerornas position.

Scanningsprocessen

För att minska risken för förbisedda punkter och få en tillräckligt stor täckning av objektet användes tre stycken Kinect-kameror, positionerade enligt Figur 4.3. Punkterna hos det scannade objektet registrerades med Kinect-kamerornas IR-kameror, enligt kapitel 2.1. Bruspunkter i punktdatan rensades bort med metoden *StatisticalOutlierRemoval filter*, en metod i **PCL**.

En nedsampling av punktdatan kunde sedan utföras genom att applicering av voxelgrid filter, även denna en metod i **PCL**, och en stor del av punktdatan kunde reduceras. Denna metod verkställde en reducering på ungefär 90 procent av punktdatan, detta utan någon märkbar skillnad av kvalité hos den resulterande modellen. Däremot gav metoden ingen betydande skillnad i beräkningshastigheten hos systemet. Poissons metod för ytrekonstruktion applicerades sedan med hjälp av **PCL** på punktdatan för att skapa en solid modell[5].

Texturering

Till textureringen av 3D-modellen användes färgkamerorna för att ta bilder som projiceras på modellen. Eftersom varje Kinect-kamera ger en bild kommer flera texturer att projiceras på objektet, men från olika vinklar. När kamerornas position kalibreras erhålls en matris som beskriver hur varje färgbild behöver transformeras för att projiceras på modellen på en korrekt plats. Texturen projiceras på modellen med hjälp av **PCL**.

4.2.3 Gränssnitt

Gränssnittet utvecklades som en webbapplikation med målet att därifrån kunna styra scanningsystemet. Det var viktigt att gränssnittet var enkelt och intuitivt att använda och användartester har därför genomförst, se stycke 3.5. Webbapplikationen visas på en touchskärm där användaren enkelt startar scannern. För att gränssnittet ska kunna kommunicera med scanningsystemet har websockets implementeras på klientsidan med hjälp av **DART**.

4.2.4 Websockets

Websockets användes för att kommunicera mellan webbapplikation och den stationära datorn där själva programmet körs, för detta användes biblioteket **libwebsockets** på serversidan. Detta bibliotek valdes för att det var enkelt och avskalat vilket passade bra då det bara behövdes skickas små korta meddelanden. När ett meddelande skickas till servern tas detta emot och servern distribuerar ut uppgifter till de olika delmodulerna i systemet.

4.2.5 MongoDB

För att lagra filerna används MongoDB, vilket möjliggör lagring av *obj-filerna* direkt i databasen med ett *MD5*-krypterat datum samt ett nyckelord som id. Nyckeln kan sedan skickas till användaren via epost om användaren bestämmer sig för att spara modellen samt användas för att radera modellen i annat fall. Klassen *GridFS* användes för att undvika problem med filernas storlek. Stora filer delas då upp i flera delar i databasen.

Kapitel 5

Resultat

5.1 Scanning

Projektarbetet resulterade i ett system som utför en scanning med ett flertal sammankopplade Kinect-kameror. Hårdvarans prestanda avgör hur många Kinect-kameror som kan användas samtidigt. Dock är minsta möjliga antal kameror för en total scanning tre stycken för total täckning. Systemet är utformat för att sättas upp i ett bås med en skärm på insidan som via ett gränssnitt går att interagera med.

5.2 Gränssnitt

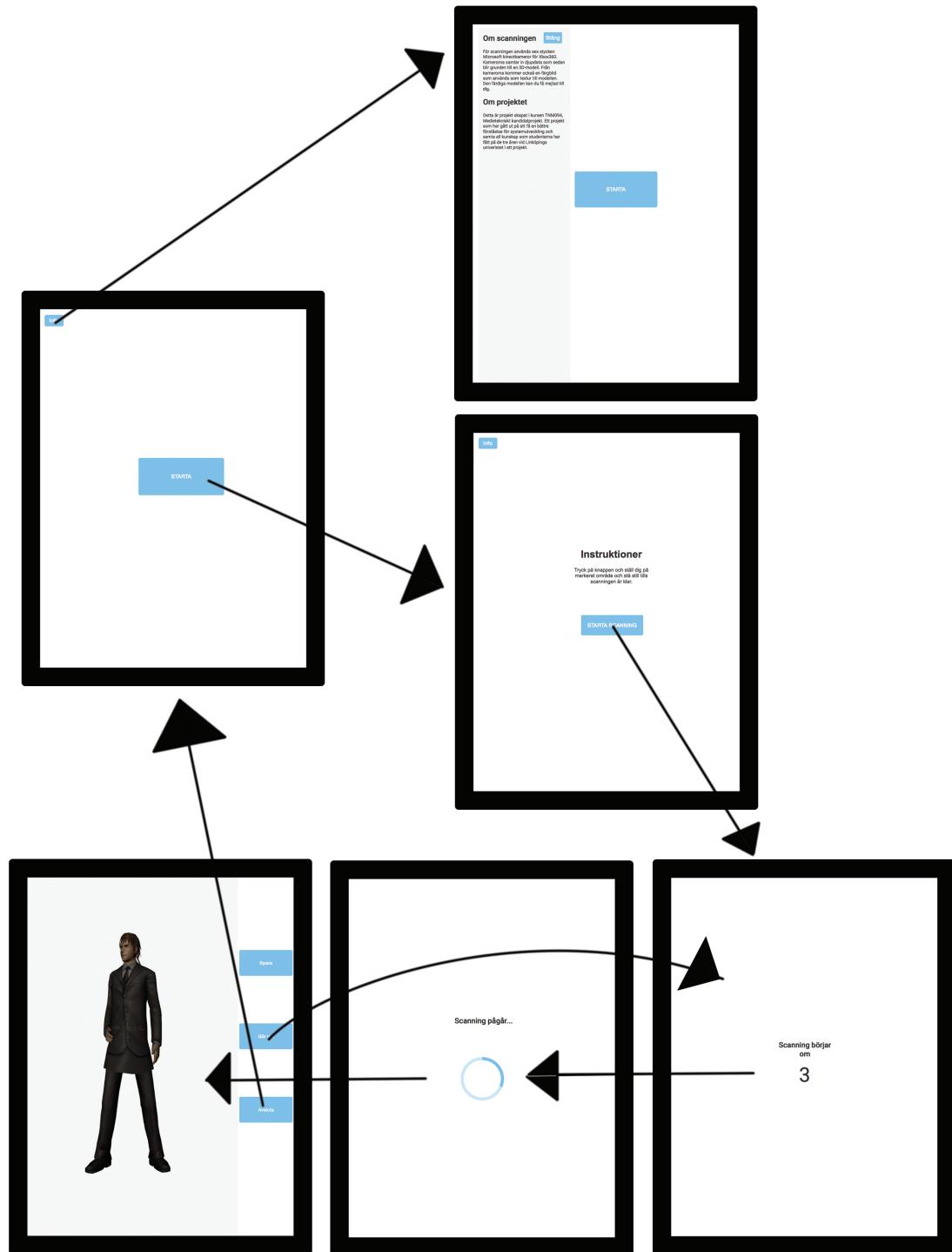
Bilderna nedan, se Figur 5.1 och 5.2 representerar gränssnittets utseende och vad som händer när användaren interagerar med de olika knappalternativen. Användaren möts först av sidan som visas på skärmen högst upp till vänster. Där finns det två alternativ en knapp med texten "Info" som leder till skärmen högst upp till vänster. Där får användaren information om projektet och om systemet. Andra knappen med texten "START" på leder vidare i scannningsprocessen. En kort information om vad användaren ska göra för att starta scanningen och en tillhörande knapp under informationen som leder till sidan längst ner till höger. Den sidan visar en nedräkning på när scanningen startas och användaren hinner ställa sig i den position den vill bli scannad i. När nedräkningen når noll kommer användaren vidare till nästa sida som visar en laddningssymbol på hur långt scanningen har kommit. När den är fylld kommer användaren till sista sidan och 3D-modellen visas och får då tre alternativ.

Användaren kan välja mellan att spara, gör om och avsluta systemet. Varje alternativ visar ett pop-up fönster enligt figuren ovan. Väljer användaren att spara kommer det upp ett fönster där en e-postadress kan skrivas in. Här valideras e-postadressen till den grad att användaren behöver skriva in rätt format på adressen, alltså kontrolleras inte att adressen finns. Stämmer inte formatet på adressen måste användaren skriva om adressen igen. Stämmer adressen får användaren en återkoppling om att modellen är sparad och skickad till den angivna adressen. Användaren kommer då tillbaka till sidan där modellen visas men det är nu inte längre möjligt att spara om modellen på nytt. Väljer användaren att göra om modellen kommer det upp ett fönster som frågar om användaren är säker på att användaren vill göra om modellen och talar om att nuvarande modell kommer att kastas, detta visas i figuren ovan på den mellersta skärmen. Nu hamnar användaren tillbaka till det steget där användaren ska ställa sig i position för scanningen. Sista alternativet, "Avsluta", fungerar som "Gör om" men användaren kommer nu komma tillbaka till första skärmen.

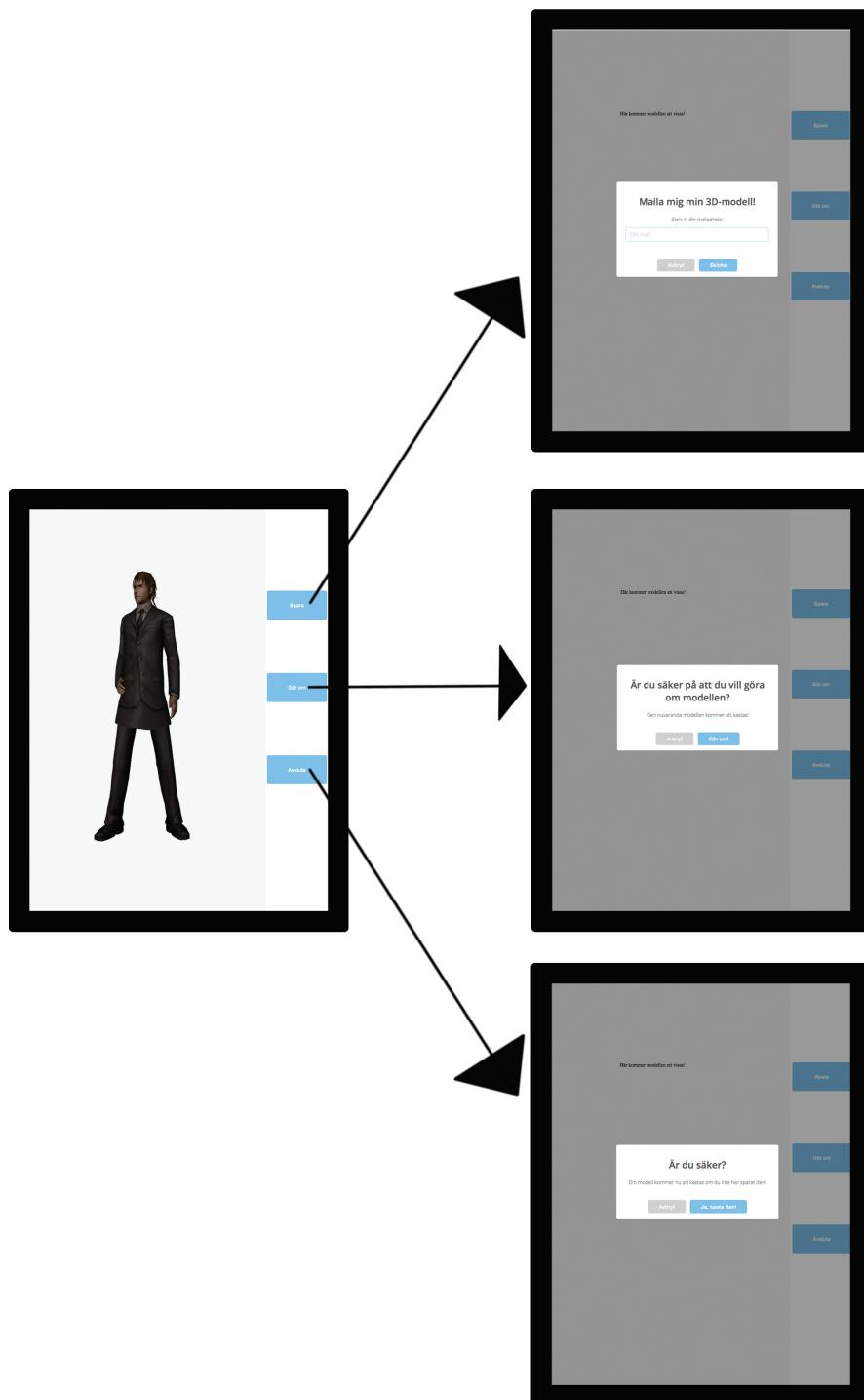
5.3 Bildbehandling

Enligt teorin kan en IR-kamera användas till att samla in punkter som sedan kan användas till att skapa en solid 3D-modell. Ett voxelgrid filter kunde användas för att sampla ner insamlad data till en mer lämplig mängd. Vidare kunde metoden StatisticalOutlierRemoval filter användas för att reducera onödigt brus. Som rekonstruktionsmodell användes **PCLs** klass Poisson för att applicera en rekonstruktionsmodell.

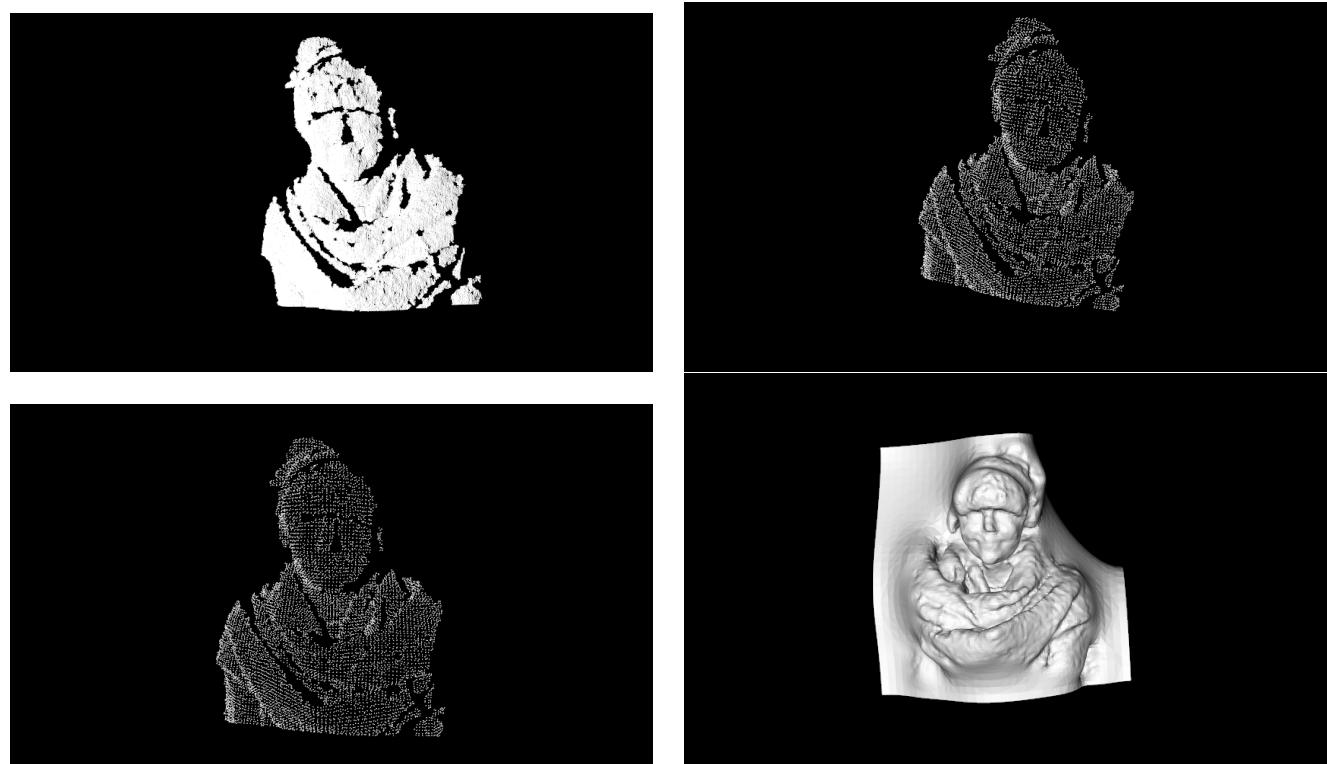
I bildserien nedan, se Figur 5.3, visar en stegvis behandling av data för scanning utförd av en Kinect-kamera. Scanningen är utförd med endast en kamera vilket resulterar i ett oönskat resultat efter Poissons metod för ytrekonstruktion. Metoden vet inte hur objektet ska byggas ihop då täckning saknas.



Figur 5.1: Gränssnittet när modellen presenteras för användaren



Figur 5.2: Gränssnittet när modellen presenteras för användaren



Figur 5.3: (A) Bild på rå punktdata. (B) Reducering av punktdata. (C) Reducering av brus. (D) Yta med Poissons metod för ytrekonstruktion

Kapitel 6

Analys och diskussion

6.1 Metod

De valda metoderna i projektarbetet är bestämda efter de kompetenser som teamet besitter och resurser som ledningen har kunnat bistå med. Det finns delar som teamet hade utfört annorlunda om projektet skulle genomföras en gång till eller om förhållanden hade varit annorlunda, som till exempel att teamet hade haft en längre tidsperiod på sig. Metodernas för- och nackdelar kommer i det här stycket att diskuteras och analyseras.

6.1.1 Scanningsprocessen

Teamet valde att scanna med Microsofts Kinect-kameror för Xbox 360 som är utvecklat och anpassat framförallt för Windows. Eftersom teamet valde att utveckla ett system som passar till Linux, OS X och till Windows valdes biblioteket **libfreenect**. Biblioteket är öppen källkod som har utvecklats av personer med eget intresse för att skapa system till Kinect-kameror. Alternativet hade varit att enbart utveckla till Windows och då kunnat använda Kinect-kamerornas tillhörande bibliotek **Kinect for Windows SDK** som har finansiellt stöd från Microsoft och då större möjligheter till att utvecklas. Med **Kinect for Windows SDK** finns det större möjligheter och det är lättare att utveckla systemet då biblioteket har fler färdiga funktioner än **libfreenect**.

För projektet valdes det att använda tre Kinect-kameror som tillsammans scannar in en människa. För att inte missa viktig information om punktdatan behövs det minst tre kameror, detta gäller om Kinect-kamerorna är statiskt placerade. Fördelen med att använda minsta möjliga antal Kinect-kameror är att detta minskar mängden data som tas in i systemet, utan att påverka kvalitén för den resulterande 3D-modellen. Detta bidrar till mindre beräkningar för datorn. Nackdelen är att kamerorna är placerade runt objektet och det är därför möjligt att missa information om vissa delar är täckta av andra och missar även informationen som är på objektet, till exempel huvudets topp. Detta gör att systemet gissar hur objektet ser ut bakom dolda delar. En lösning hade varit att ha fler kameror för att ta in mer information om objektet. Risken blir då att systemet blir trögt och i värsta fall inte uppfyller kravet att användaren ska uppleva systemet tillräckligt snabbt. En annan lösning på problemet, som tidigare har nämnts i rapporten, är att ha en rörlig kamera. Detta var inte praktiskt möjligt med de resurser som teamet hade. En rörlig kamera som snurrar kräver en mer komplicerad anordning och eftersom teamet inte hade som krav att faktiskt bygga det fysiska båset blev det en ytterligare faktor till att inte ha en rörlig kamera.

Kalibrering av djup- och färgkamera

Kalibering av djup- och färgkamera undersöktes, men prioriterades senare bort. Resultatet skulle hy-

potetisk bli ännu bättre med en kalibrering, men ansågs ändå tillräckligt bra för att lämnas till senare arbete.

6.1.2 Ytrekonstruktion

Reduceringen av data som användes av systemet är effektiv och tillräckligt snabb för att köras inom en rimlig tid. Användaren behöver endast vänta en kortare stund innan processen är klar. Ned-samplingsmetoden tar bort cirka 90 procent av punkterna i scenen. Det finns flera olika metoder för själva ytrekonstruktionen. Metoden som valdes, Poisson, gav en bra blandning av jämma och solida ytor. Andra metoder som testades var *greedyprojectiontriangulation*, *marching cubes*, *marching cubes hoppe* och *moving least squares*. Ingen av dessa metoder skapade en solid yta, vilket var ett av kraven på projektet, och därför valdes de bort.

6.1.3 Gränssnittet

Målet med gränssnittet var att göra det lätt för användaren att hantera systemet utan direkta instruktioner och att det skulle vara snabbt och enkelt att använda. Detta eftersom systemet har ett större användningsområde vid allmänna platser snarare än för privat bruk. Gränssnittet visas på en touchskärm eftersom detta är enkelt och snabbt att interagera med och förstå.

Gränssnittet utvecklades som en webbapplikation då utvecklingsteamet hade mer erfarenhet inom webbutveckling jämfört med gränssnittsutveckling inom exempelvis C++. Det var även enklare att få igång gränssnittet och bedriva användartester på en touchskärm.

6.2 Resultat

Resultatet förhåller sig bra till vad som kunde förväntas. Det kan dock variera beroende på hur kamerorna är placerade samt i vilken position användaren står i. Det kan uppstå situationer där vissa ytor inte uppfattas av IR-kameran vilket gör att rekonstruktionsmodellen uppskattar hur ytan borde byggas ihop för att skapa en solid modell.

Eftersom resultatet har en estetisk synvinkel finns vissa svårigheter med att mäta hur bra resultatet är. Utvecklingsteamets mål var att att modellen skulle vara solid, vilket har uppnåtts. En färdig 3D-modell är lik den scannade människan och kan därför ses som ett bra resultat.

6.3 Arbetet i ett vidare sammanhang

I ett vidare sammanhang skulle en 3D-skrivare möjligen kunna kopplas direkt till systemet. Med detta skulle användaren direkt kunna få en fysisk modell efter scanningprocessen istället för en digital 3D-modell via e-post. Om modellen skulle skrivas ut behöver det vara inom en rimlig tid, något som inte finns tillgängligt idag med dagens 3D-skrivarteknik. Skrivaren skulle även behöva skriva ut modellen med tillhörande textur för bästa resultat.

6.3.1 Samhälleliga aspekter

Vill samhället att en person ska ha möjlighet att skapa en ny identitet i ett spel som sig själv? Detta är en fråga som kanske inte har ett rätt svar. Om den här tekniken överfördes på spel, skulle det kunna

tänkas att personer som redan lever sina liv på Internet blir ännu mer isolerade genom att faktiskt överföras till en fysisk kropp. Detta skulle i sin tur generera en större grad av passivitet från denna grupp av människor.

Hur ska sparad data hanteras? Om produkten blir kommersiell på exempelvis Microsoft och det där börjas scannas människor, vilka lagar gäller då? Skulle företaget exempelvis kunna använda modellerna utan personens vetskap genom att personerna skulle godkänna en policy.

6.4 Problem som har uppstått under projektet

Utvecklingsteamet fick sent i projektet tillgång till en Kinect för Xbox One. Det bestämdes dock att fortsätta fokusera på de första kamerorna eftersom all kod hade skrivits för dessa och att göra om kodbasen för den nya kameran skulle ta onödig tid. Prestandaskillnaden mellan de båda kamerorna är inte märkbar för projektets användningsområde.

Någonting som har varit tidskrävande var att länka alla bibliotek på gruppens datorer. Då det har utvecklats på olika plattformar har installationsstegen för projektet varit olika. Detta har lett till att mycket tid har lagts på att konfigurera och förbereda för att överhuvudtaget kunna utveckla.

Utvecklingsteamet hade från början planerat att använda Googletest samt DARTs inbyggda testsystem. Efter hand upptäcktes dock att den typen av test inte var lämpade för projektet. En stor del av projektet handlade om att få modellen tillräckligt estetiskt tilltalande vilket inte är mätbart med dessa testmetoder.

Kapitel 7

Slutsatser

Rapporten syftade till att framlägga målet med projektet samt beskriva arbetsmetoder och verktyg som har använts under projektets gång. Rapporten har även reflekterat över projektets metodval och gett en förståelse för hur valda systemutvecklingsmetoder har fungerat i en verlig miljö.

7.1 Svar på frågeställningar

- **Hur kan ett användande av andra generations Kinect skapa en 3D-modell med tillräckligt god kvalité, utan komplicerade instruktioner och avancerade inställningar?**

Utvecklingsteamet beslutade att använda första generationens Kinect då det var det som fanns tillgängligt från början. Det ansågs sedan inte tillräckligt prioriterat att byta till andra generationens Kinect eftersom prestandaskillnaden mellan dessa två kameror inte motiverade de ändringarna som skulle behöva göras i kodbasen. Kinect-kamerorna används för att läsa in punktdata med hjälp av den inbyggda IR-kameran. Punktdatan kan sedan användas för att skapa en polygonmodell. Genom att använda flera Kinect-kameror erhålls tillräckligt med täckning för att hela kroppen ska bli scannad och inga inställningar behöver ställas in under processen. Processen har gjorts helt automatiserad för att förenkla användningen och användaren kan starta scanningen genom ett enkelt webbaserat gränssnitt. Därmed har en 3D-modell av god kvalité kunnat skapas utan vidare invecklade inställningar och instruktioner.

- **Vilka krav kan ställas på systemets justerbarhet, för att anpassa systemet efter varje individuell användare, och därmed kunna scanna och rendera en korrekt bild?**

Kamerorna måste placeras med ett tillräckligt avstånd från användaren som ska scannas så att hela kroppen täcks av punkter från Kinect-kamerorna. Utvecklingsteamet kom fram till att tre kameror, utplacerade jämnt fördelat runt kroppen, behövs för total täckning. Systemet antar dock att användaren är placerad på angiven plats, vilket medför att kroppen alltid är placerad med samma avstånd från alla kameror. Om användaren inte följer anvisningarna kan fel uppkomma.

- **Vilken eller vilka metoder ska systemet använda sig av för att reducera mängden brus i den insamlade punktdatan?**

Utvecklingsteamet har använt sig av **StatisticalOutlierRemoval** filter för att reducera mängden brus. Metoden appliceras med hjälp av biblioteket **PCL** och syftar till att ta bort punkter som avviker för mycket från andra punkter.

- **Hur kan utvecklingsteamet, utifrån undersökningar av alternativa sätt för insamling av data, finna den bästa lösningen med avseende på kvalitén hos 3D-modellen och vad är förrespektive nackdelarna med de framtagna lösningarna?**

Utvecklingsteamet kom fram till att tre kameror var det minsta antalet kameror som behövdes för total täckning. En annan lösning som diskuterades var att endast en kamera skulle användas, men med en rotationsrörelse runt människan. Fördelen med den metoden var att endast en kamera behövdes. Utvecklingsteamet ansåg dock att anordningen för en roterande kamera skulle komplikera arbetet och uppta tid från mer relevant arbete, eftersom målet inte var att bygga det fysiska båset. Även en lösning för hur kameran skulle ta bilder under rotationsrörelsen skulle blivit ett tillkommande problem.

7.2 Framtida förbättringar

Framtida förbättringar för systemet som kan utvecklas är att koppla en 3D-skrivare till systemet för en direkt utskrift av modellen i samband med scanningen. Eftersom den färdiga modellen sparas som en standardiserad *obj-fil*, är en 3D-utskrift redan idag en möjlighet. Anledningen till att en 3D-skrivare inte är en del av systemet i dagsläget beror på att dagens 3D-skrivare inte har den utskriftshastighet som skulle behövas för att skriva ut modellen inom en godtycklig väntetid. Dagens 3D-skrivare saknar även den färgvariationen som behövs för att skriva ut en modell med den textur som skall appliceras. Men i och med att 3D-skrivare utvecklas och får snabbare utskrift, är en direkt utskrift en framtida möjlighet.

En annan utvecklingsmöjlighet är att användaren skulle kunna ha som val att spara sin modell i en gästbok där andra användare kan bläddra bland modeller. Det skulle ge användaren en insyn om vad scanningen handlar om och användaren kan därefter ta ställning till om man vill utföra en egen scanning. Det är dock viktigt att denna process är frivillig och användaren är inte tvingad till att göra modellen synlig för andra.

Utvecklingsteamet hade från början endast tillgång till en äldre version av Kinect-kamerorna. Då den nya versionen inte medförde en vesäntlig förbättring för ändamålet prioriterades inte en implementation av den nya versionen. Ytterligare ett steg skulle vara att göra programvaran kompatibelt med båda versionerna eller byta helt till den nyare versionen.

Projektet har utvecklats med **OpenKinect** eftersom flera utvecklingsplattformar har använts. Om utvecklingsteamet istället valt att endast utveckla på Windows med deras egna API hade det möjliggjort för systemet att skapa ett skelett till den inscannade kroppen. Det hade i sin tur kunnat leda till att utvecklingsteamet hade hunnit med att skapa en animation till modellen. En annan utvecklingsmöjlighet inom liknande område är om endast huvudscanning skulle möjliggöras. Huvudet skulle sedan kunna appliceras på andra färdiga modeller i spel. Det skulle ge användaren en känsla av närvaro.

7.3 Konsekvenser för berörd målgrupp

En bra scanningsprocess medför ett mindre behov av manuell 3D-modellering, detta kan leda till att jobb inom denna bransch försvinner.

Ett enkelt och billigt scanningssystem ger mindre företag och privatpersoner med lägre budget möjligheten att använda sig utav ett scanningssystem. Det kan medföra nya idéer och utvecklingsmöjligheter som underlättar för till exempel spelutvecklare som då inte behöver skapa en 3D-modell från grunden.

Litteraturförteckning

- [1] Craig J. Robert, *Establishing new boundaries for special effects* *Journal of Popular Film & Television*. Vol. 28 Issue 4 Winter 2001
- [2] Kourosh Khoshelham, Sander Oude Elberink, *Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications*, hämtad: 2015-05-05
<http://www.mdpi.com/1424-8220/12/2/1437/htm>
- [3] Jan smisek et al., *Consumer Depth cameras for computer vision*, springer 2013
- [4] Arturo de la Escalera & Jose María Armingol, *Automatic Chessboard Detection for Intrinsic and Extrinsic Camera Parameter Calibration*
<http://www.mdpi.com/1424-8220/10/3/2027/html> 2015-05-13
- [5] Hao Song, Hsi-Yung Feng *A global clustering approach to point cloud simplification with a specified data reduction ratio*, *Computer-Aided Design Volume 40, Issue 3* Elsevier 2008
- [6] Downsampling a PointCloud using a VoxelGrid filter
http://pointclouds.org/documentation/tutorials/voxel_grid.php
2015-05-13
- [7] Remondino Fabio *From pointcloud to surface: the modeling and visualization problem*, <http://www.isprs.org/proceedings/xxxiv/5-W10/papers/remondin.pdf> 2015-05-07
- [8] Michael Kazhdan et al. *Poisson Surface Reconstruction* <http://research.microsoft.com/en-us/people/hoppe/poissonrecon.pdf> 2015-05-11
- [9] Aitor Aldoma et al. *Point Cloud Library, Three-Dimensional Object Recognition and 6DoF Pose Estimation* http://www.inf.ethz.ch/personal/zeisl/publications/aldoma_2012jram_PCLTutorial.pdf 2015-06-07

Bilaga A

Involverade parter

Utvecklingsteamet

- Pelle Serander: Scrummaster, arbetat med bildbehandling, MD5-kryptering samt databasstöd från C++-sidan.
- Erik Sandrén: Kundkontakt, arbetat med de flesta delarna
- Cecilia Lagerwall: Arbetat med scanning(kalibrering av IR- och färgkamera) och gränssnittet.
- Johan Reimann: Gruppledare, har arbetat med scanning, kalibrering, texturering och hantering av filer.
- Isabell Jansson: Rapportansvarig, arbetat med gränssnittet, websocket och textureringen på båda DART- och C++-sidan.
- Carl Englund: Sekreterare, arbetat med gränssnitt och websocket samt databasstöd på DART- och C++-sidan.
- Caroline Gard: Kodgranskningsansvarig. Har arbetat med bildbehandling och gränssnitt.

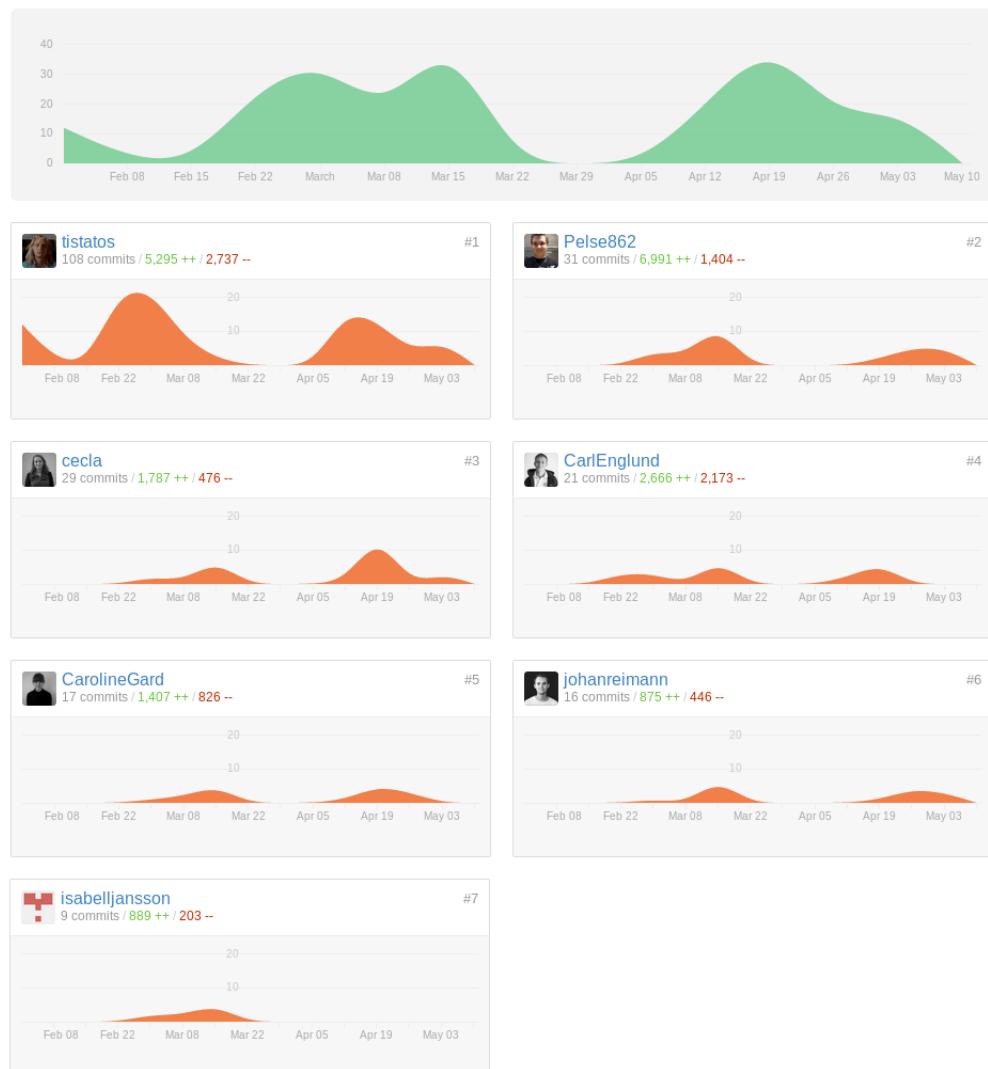
Bilaga B

Statistik från github

Feb 1, 2015 – May 11, 2015

Contributions to develop, excluding merge commits

Contributions: **Commits** ▾



Bilaga C

Gantt schema

