



Een collaboratief platform voor het efficiënt reconstrueren van fresco's

Nicolas Hillegeer

Thesis voorgedragen tot het behalen van de graad van Master in de ingenieurswetenschappen:
computerwetenschappen

Promotor:

Prof. dr. ir. P. Dutré

Assessoren:

Ir. N/A
N/A

Begeleider:

Dr. ir. B.J. Brown

© Copyright K.U.Leuven

Zonder voorafgaande schriftelijke toestemming van zowel de promotor als de auteur is overnemen, kopiëren, gebruiken of realiseren van deze uitgave of gedeelten ervan verboden. Voor aanvragen tot of informatie i.v.m. het overnemen en/of gebruik en/of realisatie van gedeelten uit deze publicatie, wend u tot het Departement Computerwetenschappen, Celestijnenlaan 200A bus 2402, B-3001 Heverlee, +32-16-327700 of via e-mail info@cs.kuleuven.be.

Voorafgaande schriftelijke toestemming van de promotor is eveneens vereist voor het aanwenden van de in deze masterproef beschreven (originele) methoden, producten, schakelingen en programma's voor industrieel of commercieel nut en voor de inzending van deze publicatie ter deelname aan wetenschappelijke prijzen of wedstrijden.

Voorwoord

Dit is mijn dankwoord om iedereen te danken die mij bezig gehouden heeft. Hierbij dank ik mijn promotor, mijn begeleider en de voltallige jury. Ook mijn familie heeft mij erg gesteund natuurlijk.

Nicolas Hillegeer

Inhoudsopgave

Voorwoord	i
Samenvatting	iv
Lijst van figuren en tabellen	v
1 Inleiding	1
2 Overzicht van het thera project	3
2.1 De opdelingen van het thera project	3
2.2 Reconstructie, de bestaande oplossingen	4
3 Doelen & Motivatie	9
3.1 Aspecten waarop de thesis tracht te verbeteren	10
4 Ontwerp van het project	15
4.1 De grote lijnen	15
4.2 Modulariteit	17
4.3 De eenheid van informatie: het fragmentenpaar	17
4.4 Het beheer van de data	18
4.5 Visualisatie, een manier om met de data te werken	19
4.6 Integratie in thera project	19
4.7 Uitbreidbaarheid	20
5 Bespreking van het database ontwerp	21
5.1 Vereisten	21
5.2 Alternatieven	21
5.3 Externe database traag, interne database snel	21
5.4 Slimme client of slimme server?	22
5.5 Benchmarking	23
6 Modules	25
6.1 MatchTileView	25
6.2 Proof of concept: GraphView	25
7 Tests & Vergelijkingen	27
7.1 Use cases	27
7.2 Snelheid (objectieve tests)	28
7.3 Gebruiksgemak (subjectieve tests)	29
8 Besluit en toekomstige toepassingen	31

INHOUDSOPGAVE

8.1 Toekomst	31
A Numerieke resultaten	35
Bibliografie	37

Samenvatting

Deze **abstract** moet nog geschreven worden.

Lijst van figuren en tabellen

Lijst van figuren

2.1	Een voorbeeld Griphos tafelblad, 1 voorgesteld paar en 2 aparte fragmenten zijn reeds ingeladen	4
2.2	Griphos kan gebruikt worden om de posities van fragmenten in hun opslagplaats te onthouden en vervolgens snel terug te vinden. Het kan ook een foto van de bak waarin ze liggen onder het virtuele tafelblad projecteren. De brokstukken die buiten de bak staan weergegeven zijn verplaatst geweest naar een andere bak. (afbeelding met toestemming gebruikt uit [7])	6
2.3	Browsematches in werking, de balken boven de paren duiden een validatie door de gebruiker aan. Rood betekent bijvoorbeeld “dit voorstel is zeker niet juist”	7
3.1	Het huidige proces met de aanvullingen van de thesis in het blauw	10
4.1	Het abstracte model van de applicatie, links staat de <i>View/Controller</i> en rechts het <i>Model</i> . De controller stuurt een verzoek naar het model voor een bepaalde (sub)set van de data — al dan niet gesorteerd — en het model antwoordt met alle paren die voldoen aan de criteria	16
4.2	De manier van weergeven uit Browsematches werd gekopieerd naar het nieuwe platform, met uitbreidingen	17
4.3	Een vereenvoudigde kijk op de componenten van de visualisatielaag, het hoofdscherm en het model. De componenten in het wit behoren tot de rest van het thera project.	18

Lijst van tabellen

Hoofdstuk 1

Inleiding

Het reconstrueren van fresco's waarvan in opgravingen fragmenten gevonden worden is een moeilijke taak. Men kan het vergelijken met het oplossen van een enorme puzzel waarvan de stukken arbitraire vormen hebben, de meesten hun originele kleur zijn verloren en er vele anderen ontbreken. Daarbovenop komt nog eens dat er heel wat stukken over de eeuwen heen een zekere vorm van slijtage hebben ondervonden, waardoor ze niet meer perfect op elkaar passen en dus confirmatie nog moeilijker maken.

[afbeelding van enkele opgegraven stukken]

De stukken van de rand van het fresco met recht afgelijnde kanten of die nog een voldoende zichtbaar geometrisch patroon bevatten, zijn in vergelijking met de anderen eenvoudig met elkaar te verbinden. De overige fragmenten die minder informatie bevatten zijn echter een nachtmerrie om aan elkaar te koppelen, gezien het menselijke visuele systeem niet gemaakt is om dat soort grillige randen te vergelijken en aan elkaar te zetten. Om deze reden is het normaalgezien noodzakelijk om vele mogelijke kandidaten visueel te onderscheiden en ze over elkaar te laten glijden om te zien of het past en waar precies.

In deze context situeert zich het **thera**¹ project, dat probeert om het werk van de archeoloog gemakkelijker te maken door middel van een software platform. De redenering achter het project is dat wat voor een mens zeer tijdsabsorberend en saai zou zijn, geautomatiseerd zou kunnen worden m.b.v. een computer. Dergelijk systeem werd in 2007 aan de *Princeton* universiteit in Amerika geconciepeerd. Sindsdien is er door verschillende onderzoekers van over de hele wereld aan gewerkt om alle noodzakelijke delen te ontwerpen, implementeren en integreren. [vermeld hoofdonderzoekers]. De werking van het systeem wordt in het volgende hoofdstuk nader toegelicht.

¹Thera is de oude naam voor het huidige griekse eiland genaamd Santorini, waar het project voor het eerst in de praktijk werd toegepast.

1. INLEIDING

Deze thesis draait rond het maken van een uitbreiding op één van deze delen. De uitbreiding moet de gebruikers van het systeem in staat stellen om de beschikbare data op nieuwe manieren te kunnen gebruiken, visualiseren, aanpassen en delen met medeonderzoekers.

// TODO: vorige stuk gaat niet zozeer over samenwerking meer, herschrijf dit mss?
De gangbare methode van manueel puzzelen van vóór het thera project leende zich wel tot een zekere vorm van samenwerking, op voorwaarde dat men op dezelfde site aanwezig was. Verder moest alle betreffende informatie van commentaren tot classificaties in een centrale plek bewaard worden zodat iedereen erbij kon. Dit ging van memobriefjes op de fragmenten zelf tot manueel aangepaste elektronische documenten. Dankzij het thera systeem kon deze manier van werken veranderd worden, alle relevante kennis kan elektronisch opgeslagen worden. Er was echter tot op heden nog geen gemakkelijke manier om deze kennis met andere archeologen te delen en al zeker niet om de bevindingen van verschillende onderzoekers te combineren. Dat is het kernprobleem dat deze thesis zal proberen uit de wereld te helpen. Kortom: de nodige aanpassingen en nieuwe implementaties zullen gedaan worden om het oude systeem collaboratief te maken, zodat het beheren, delen en combineren van data gemakkelijk en intuitief wordt.

Hoofdstuk 2

Overzicht van het thera project

Zoals eerder vermeld bouwt deze thesis verder op een reeds bestaand project. Om ze beter te kunnen situeren in het geheel is het handig om eerst even het thera project bekijken om zo te kunnen zien waar dit project in past.

2.1 De opdelingen van het thera project

Ruwweg gezien kan het reconstrueren van een fresco met behulp van de computer opgedeeld worden in 3 fasen.

Acquisitie Alle gevonden fragmenten worden ingescand met behulp van 3D en 2D-scanapparatuur om zo een virtueel model van elk stuk te bouwen, zie [1].

Identificatie Vervolgens worden deze virtuele fragmenten aan een zogenaamde *matcher* gegeven, die voor elk fragmentenpaar gaat kijken of ze mogelijk op elkaar passen. Er zijn verschillende types ontwikkeld. Eén ervan (genaamd *RibbonMatcher* [1]) kijkt enkel naar de randen van de brokstukken om te zien of ze in elkaar sluiten. Hierdoor is het in staat om passende fragmenten te vinden die voor mensen moeilijk te vinden zijn wegens te weinig distinctieve attributen zoals tekeningen. Het onderzoek gaat verder, in 2010 werd er een nieuw type ontwikkeld dat zijn analyse baseert op een combinatie van verschillende eigenschappen, zoals de sporen die een borstel kan nalaten bij het kleuren van een fresco of zelfs de beoordeling van de eerder genoemde *RibbonMatcher* [6].

Classificatie & Reconstructie De derde en laatste stap bestaat uit het classificeren van wat de vorige stappen produceren. Elk voorgesteld paar moet gecontroleerd worden op validiteit. Verschillende statussen kunnen toegekend worden aan een paar, zoals: *geconfermeerd*, *misschien*, *nee*, *in conflict met een ander paar*, et cetera. De *matcher* produceert in de regel zeer veel mogelijk passende paren, waarvan slechts een klein deel correct kan zijn. De drempel voor het beslissen van wat een paar kan zijn en wat niet wordt in de vorige stap zo laag ingesteld omdat men wil vermijden dat twee fragmenten die toch passen genegeerd worden. Anders gezegd, de kost van “*false negatives*” wordt

2. OVERZICHT VAN HET THERA PROJECT

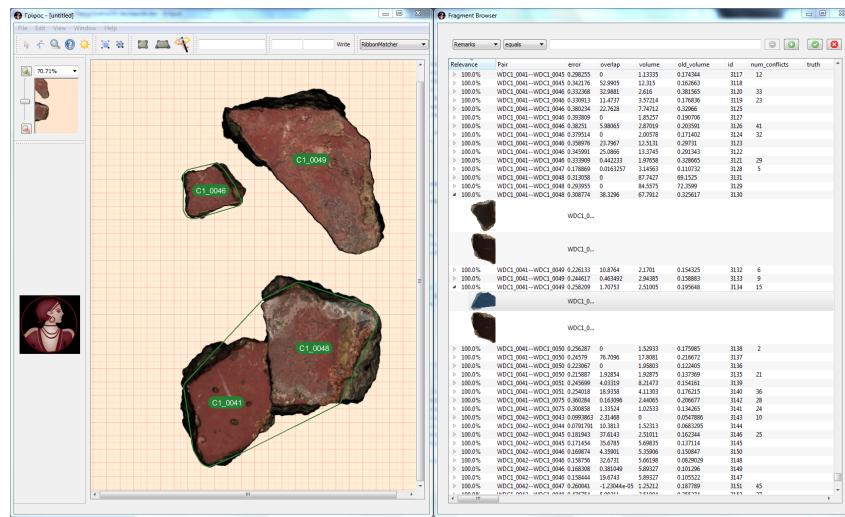
hoog ingeschat. Uiteindelijk zal hieruit een gereconstrueerde fresco moeten ontstaan.

2.2 Reconstructie, de bestaande oplossingen

Het thesisproject besproken in deze tekst tracht de reeds bestaande hulpmiddelen van de reconstructiefase aan te vullen. Hievoor werden reeds programma's ontwikkeld, namelijk *Griphos* en *Browsmatches*. Hierop volgt een bondige bespreking van beide programma's, wat ze kunnen en niet kunnen, en eventuele voor- en nadelen. Deze factoren hebben een belangrijke rol gespeelt bij het bepalen van de richting van deze thesis.

2.2.1 Griphos

Griphos was de eerste applicatie gericht op het weergeven en beoordelen van de resultaten van de voorgaande stappen. Het werd ontworpen met het doel een centraal zenuwstelsel te zijn voor alle informatie en met deze uiteindelijk een (zo goed mogelijk) gereconstrueerd fresco te maken. Het is mogelijk om zowel aparte fragmenten als automatisch herkende fragmentparen op een virtueel tafelblad te plaatsen en manipuleren (zie figuur 2.1). De idee achter een dergelijke voorstelling komt voort uit een rechstreekse vertaling naar de computer van wat een archeoloog op een werkelijk tafelblad doet.



FIGUUR 2.1: Een voorbeeld Griphos tafelblad, 1 voorgesteld paar en 2 aparte fragmenten zijn reeds ingeladen

Griphos wordt echter geplaagd door een paar problemen: het is traag en biedt geen goede manier aan om de talloze gegenereerde paarvoorstellingen na te kijken. De gebruikte metafoor van het virtuele tafelblad waarin gepuzzeld kan worden stelt een belangrijk deel van het reconstructieproces voor, maar schiet tekort als men de resultaten van automatische fragmentpaarontdekking op vloeiente wijze in het proces wil betrekken. Het probleem lijkt te zijn dat er te veel informatie is en Griphos geen goede manier aanbiedt om door de bomen het bos te zien. De aanzienlijke traagheid van sommige delen van het programma komen vooral voort uit de manier van dataopslag voor paren (XML bestanden) en de complexiteit van de visualisatie (afbeeldingen van hoge kwaliteit en/of gedetailleerde 3D-modellen). Dit zorgt voor een soms onaangename werkervaring zelfs indien men er toch in slaagt de juiste fragmentparen te lokaliseren. Desalniettemin is het een krachtig programma dat veel functionaliteit biedt voor de detailinspectie van brokstukken.

Het heeft zijn nut al op verschillende vlakken bewezen, behalve detailinspectie is het bijvoorbeeld ook in staat om de posities van fragmenten in een bak te onthouden. Dit betekent een grote snelheidswinst wanneer men bijvoorbeeld denkt een goed paar te hebben gevonden en men wil dit met echte fragmenten verifiëren. Als beide fragmenten in dezelfde bak liggen kan het correcte tafelblad ingeladen worden, Griphos kan vervolgens de gezochte fragmenten laten oplichten. Dit is veel efficiënter dan fragmenten opsporen door vormen te vergelijken, zeker omdat de brokstukken vaak moeilijk te onderscheiden zijn zoals te zien valt in figuur 2.2.

// te technisch voor dit hoofdstuk, verplaats naar iets later: [TODO: remove] Het eerste probleem wordt vooral veroorzaakt door trage datatoegang en te complexe visualisaties. Het programma slaagt alles betreffende paren op in een (enorm) XML bestand. Dit is nog doenbaar als men slechts een paar duizend voorstellen heeft maar ondervindt reeds snel onacceptabele vertragingen eens men er meer probeert in te laden. Ter voorbeeld, een kleine voorstellenverzameling van een bepaalde opgraving heeft er reeds 50000. Gekoppeld hieraan laadt Griphos ook steeds een hoge-kwaliteits afbeelding of zelfs een volledig 3D-model in voor elk paar. De combinatie van het gebruik van grote XML-bestanden om informatie over de paren in op te slaan, en het steeds inladen van hoge kwaliteits-afbeeldingen en 3D-modellen, om alle info in op te slaan. Dit is bijzonder inefficiënt en geeft ook bitter weinig mogelijkheden tot uitbreiding. Een ander deel is het gebrek aan zoekmogelijkheden binnen de voorstellen.

2.2.2 Browsematches

Een eerste prototype om de in Griphos ontbrekende delen aan te vullen, werd Browsematches genoemd. Het gebruikt eerst de visualisatiecapabiliteiten van Griphos om kleine afbeeldingen te nemen van elk bestaand paar met aan de linkerkant de doorsnede van hun raakvlak. Vervolgens toont het een scherm vol paren en kan men met de pijltoetsen navigeren tussen schermen.

2. OVERZICHT VAN HET THERA PROJECT

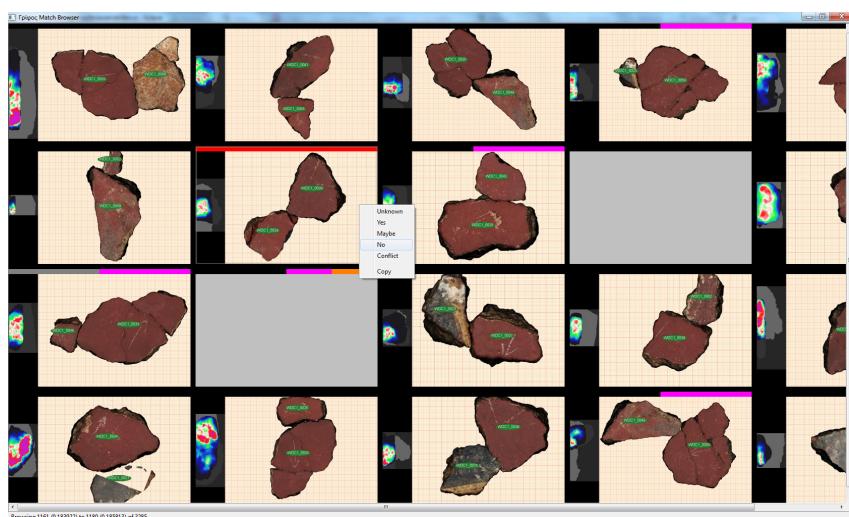


FIGUUR 2.2: Griphos kan gebruikt worden om de posities van fragmenten in hun opslagplaats te onthouden en vervolgens snel terug te vinden. Het kan ook een foto van de bak waarin ze liggen onder het virtuele tafelblad projecteren. De brokstukken die buiten de bak staan weergegeven zijn verplaatst naar een andere bak.
(afbeelding met toestemming gebruikt uit [7])

Dit kleine programma groeide uit noodzaak: het valideren van paarvoorstellingen is zo omslachtig met Griphos dat Browsematches in korte tijd werd gemaakt en succesvol gebruikt om het proces te stroomlijnen. Bij het begin van de thesis, om kennis te maken met het Thera project, werd een verbinding gemaakt met Griphos zodat voorstellen die interessant waren in detail bestudeerd konden worden.

Browsematches erfde echter ook enkele van de nadelen van Griphos. Het inladen van de data is traag en vergt veel geheugen zelfs voor gereduceerde datasets. Daarbij is het moeilijk om de informatie uit te breiden of deze gemakkelijk te combineren met de bevindingen van andere onderzoekers. Daarenboven is Browsematches een prototype en niet bedoeld voor algemeen gebruik. Het simpele maar succesvolle concept diende als inspiratie en basis voor deze thesis.

2.2. Reconstructie, de bestaande oplossingen



FIGUUR 2.3: Browsematches in werking, de balken boven de paren duiden een validatie door de gebruiker aan. Rood betekent bijvoorbeeld “dit voorstel is zeker niet juist”

Hoofdstuk 3

Doelen & Motivatie

Het hoofddoel van het thera project is om het volledige proces van fresco's reconstrueren zo gemakkelijk mogelijk te maken. Er moet een manier gevonden worden om een waardevolle contributie te maken aan het thera-ecosysteem zodat het zijn doel beter kan vervullen. Dit is onder andere mogelijk door delen die ontbreken aan de vorige oplossingen aan te maken of bestaande componenten veelzijdiger te maken.

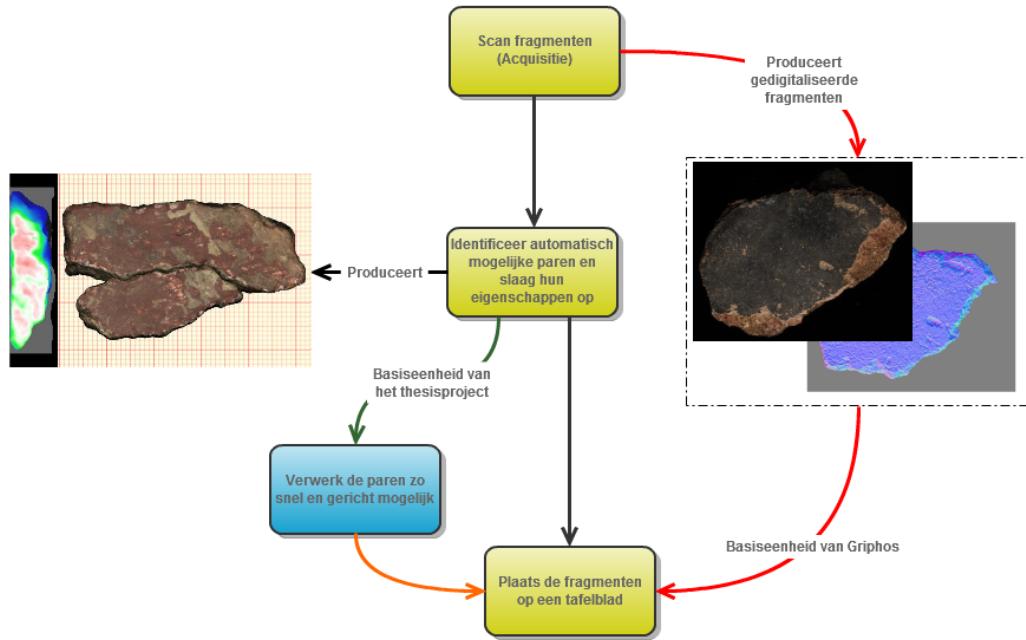
Gezien de huidige stand van zaken besproken in hoofdstuk 2, valt het op dat er nog behoorlijk wat dingen kunnen toegevoegd worden op het gebied van informatie en het visualiseren ervan. De ongelooflijke hoeveelheid data die het thera project reeds heeft geproduceerd en blijft produceren kan op een betere manier behandeld worden zodat het ware potentieel ervan naar de oppervlakte komt. De laatste stap in het sterk geautomatiseerde virtuele reconstructie proces wordt gekenmerkt door een nood aan interactie met de mens die elk soort hulpmiddel moet krijgen om een beslissing zo correct en snel mogelijk te maken. Natuurlijk gaat er niets boven de feitelijke fragmenten fysisch vastnemen en ze aan elkaar te proberen zetten. Maar gezien dit een zeer tijdsrovende bezigheid is, moet dit zo veel mogelijk beperkt worden.

Er werd gesproken over Griphos en diens weinig ontwikkelde ondersteuning voor het behandelen van automatisch voorgestelde fragmentparen, alsook over de eerste poging om dit recht te trekken: Browsematches. Deze thesis tracht de lijn van Browsematches verder te zetten, vertrekende van de goede ideeën en ervaring waaraan het zijn ontstaan te danken heeft. Dit houdt in dat de focus niet meer zozeer ligt op het actief puzzelen en brokstukken op een tafelblad plaatsen, maar eerder op het beoordelen van de omvangrijke verzameling voorstellen, waarvan slechts een zeer klein deel correct kan zijn. De hoop en verwachting is dat dit zeer kleine deel meteen ook een significante hoeveelheid van de werkelijk overblijvende paren voorstelt.

Daarmee valt te benadrukken dat dit project geen vervanging probeert te zijn van de bestaande software, het is eerder bedoeld als een complement. Merk op dat gevalideerde paren niet kunnen conflicteren en dus rechtstreeks in een groep op een tafelblad geplaatst kunnen worden. In de limiet, wanneer alle mogelijke

3. DOELEN & MOTIVATIE

paren correct geklassificeerd zijn vloeit hieruit op natuurlijk wijze een (zo compleet mogelijk) fresco voort. Daarom valt de implementatie in de thesis te zien als een extra tussenstap, chronologisch vóór het plaatsen van de fragmenten op een tafelblad en na het uitvoeren van de herkenningsalgoritmen. Figuur 3.1 stelt dit visueel voor.



FIGUUR 3.1: Het huidige proces met de aanvullingen van de thesis in het blauw

Een van de meest fundamentele verschillen tussen de centrale filosofie die Griphos vooropstelt en die van dit project is dus dat terwijl bij Griphos de focus ligt op aparte geplaatste fragmenten, het thesisproject eerder de automatische gevonden paren behandelt. Het is in Griphos wel degelijk mogelijk om voorstellen van de automatische paarherkenning in te laden maar de naald in de hooiberg vinden is moeilijk.

3.1 Aspecten waarop de thesis tracht te verbeteren

Hieronder beschreven staan verschillende deelaspecten waarop dit thesisproject tracht verbeteringen (op Browsematches) te maken. Bij elk aspect staat een beschrijving van wat het resultaat allemaal zou moeten mogelijk maken. Men kan deze desgewenst onafhankelijk bekijken maar vaak steunen ze op elkaar om werkelijk tot hun recht te komen. Wat is bijvoorbeeld een visualisatiemethode zonder een manier om de zaken die men wil zien te kiezen en aan te reiken? Wat is een uitgebreid databaseheersysteem zonder een mogelijkheid om eigenlijk iets met deze data te doen? Om de verbeteringen

3.1. Aspecten waarop de thesis tracht te verbeteren

en ideeën te bundelen en te testen is er ook een applicatie gemaakt die dient om reeds een voorproef te geven van wat er mogelijk is met de ontwikkelde technologie.

3.1.1 Integratie

Zoals eerder vermeld staat dit thesisproject niet alleen maar maakt het deel uit van een groter geheel. Binnen de grenzen van het mogelijke zou er moeten rekening gehouden worden met de integratie van de geschreven code met die van de andere onderzoekers. Dit verhoogt de kans dat het werk aanvaard wordt en ingang vindt in andere subprojecten.

3.1.2 Collaboratie

Ideaal gezien zouden archeologen steeds toegang moeten krijgen tot hun project op eender welk moment vanop eender welke plaats. Dit kan een gedeelde collectie zijn die over het internet beschikbaar wordt gesteld, of een lokale kopie. Een mogelijk scenario hierbij is dat een ervaren archeoloog in de Verenigde Staten gevraagd wordt om zijn opinie te geven over de huidige stand van zaken (reeds geïdentificeerde correcte paren, moeilijke gevallen, ...). Alle aanpassingen en commentaren die hij maakt worden automatisch ingevoegd en centraal beschikbaar gesteld voor de onderzoekers ter plekke. Op termijn moet het bijvoorbeeld zelfs mogelijk worden om amateurs te laten kijken naar de voorstellen en hun beoordeling te gebruiken om de nog na te kijken voorstellen te rangschikken.

Van cruciaal belang is dat alle verzamelde data (zoals de classificatie van voorstellen) op een robuuste manier opgeslagen, gedeeld en geïncorporeerd kan worden. De toegang naar en manipulatie van deze informatie moet efficiënt zijn. De huidige oplossingen zijn hiervoor ontoereikend en traag, zoals besproken in hoofdstuk 2. De mogelijkheid van meerdere en zelfs lokale kopieën impliceert ook de nood om te kunnen synchroniseren¹. Dit is ook nodig omdat er reeds verschillende huidige verzamelingen bestaan die eventueel in het nieuwe systeem geïmporteerd en gecombineerd moeten worden. Dergelijk systeem is ook nuttig voor het maken van (kleinere) pocketversies en in gebieden waar de internetconnectiviteit niet adequaat of onbestaande is. Belangrijk is dat er steeds een manier is om waardevolle data te combineren en aan te vullen zodat niets verloren gaat.

3.1.3 Schalering

De bestaande oplossingen voor het valideren van voorgestelde paren werken noodgedwongen met een sterk gereduceerde verzameling. Eén van de redenen hiervoor is het gebruik van een groot XML bestand om alles in op te slaan. Deze techniek werkt goed voor bijvoorbeeld het opslaan van tafelbladen — waar er bijvoorbeeld 50 fragmenten en hun locaties moeten opgeslagen worden — maar schiet tekort voor

¹Naar het model van de zogenaamde *Distributed Version Control System (DCVS)* systemen zoals Git, Mercurial, ...

3. DOELEN & MOTIVATIE

paarvoorstellen. Een snelle rekensom geeft de reden aan: een laag aantal fragmenten voor een specifieke opgraving is bijvoorbeeld 1500. De meeste paarherkenners gaan alle brokstukken een keer met elk andere brokstuk vergelijken en zodoende het meest waarschijnlijke aankoppelingspunt vinden. Het is zelfs mogelijk dat een fragment meerdere keren een plausibel paar vormt met eenzelfde stuk. Naar onder afferond komen uit deze stap reeds 2 miljoen configuraties gerold, de ene wat waarschijnlijker dan de andere. Dit nummer stijgt kwadratisch in het aantal fragmenten en zelfs met hogere drempels om volgens een bepaalde maatstaf paren automatisch te verwijderen stijgt het aantal configuraties snel.

3.1.4 Gebruiksvriendelijkheid

De uiteindelijke gebruikers van de applicatie zijn niet de ontwikkelaars van het project zelf, maar de archeologen. Om deze reden is het belangrijk dat er rekening gehouden wordt met de noodzaak van een visueel aangename en intuïtieve gebruikservaring. Om deze intuïtiviteit te bereiken moet in het programma steeds de aandacht gevestigd blijven op hetgeen het belangrijkst en meest herkenbaar is: de paren en hun fragmenten. Bij voorkeur moet elke operatie gemakkelijk ontdekbaar zijn in de context waar ze kan gebruikt worden, met eventueel een woordje uitleg erbij.

Volgens vele studies op het gebied van gebruikersinterfaces [2, 4, 3] geraken gebruikers gefrustreerd vanaf een operatie een zekere tijd duurt. Deze frustratierempel hangt af van de aard van de operatie (het resultaat), de frequentie waarmee die uitgevoerd wordt, of er visuele tekenen van voortgang zijn en of er tijdens het wachten (steeds) iets anders kan uitgevoerd worden. Om deze reden is het belangrijk dit aspect in acht te nemen bij het ontwikkelen van de applicatie. Dit is vooral zo omdat veel van de acties die mogelijk moeten zijn het potentieel hebben traag te lopen en dus de gebruiker te frustreren. Een algemene vaststelling: de tijd die een operatie mag innemen is omgekeerd evenredig met de frequentie waarmee deze operatie moet uitgevoerd worden. Deze regel in acht nemend is het duidelijk dat bijvoorbeeld het inladen van een scherm vol voorstellen zoals bij Browsematches, het veranderen van een attribuut van een paar, het filteren en sorteren en dergelijke meer acties zijn die met de grootst mogelijke snelheid moeten worden uitgevoerd. Hoe functioneel ook, een programma dat stoom reageert en elke computer op z'n knieën dwingt zal zo weinig mogelijk gebruikt worden [5].

// hoort mss bij [DESIGN] De gebruikersinterface van de oude programma's was goed en kon efficiënt gebruikt worden mits enige training. Maar voor het ondersteunen van collaboratief werken moeten er natuurlijk allerhande nieuwe operaties toegevoegd worden. Tijdens het implementatieproces werd het duidelijk dat de reeds bestaande code van het Browsematches niet uitbreidbaar genoeg was en die van Grifos te complex en belangrijk. Daardoor werd de beslissing genomen om de basisinterface van Browsematches over te nemen maar alle onderliggende code te herschrijven zodat die uitbreidbaar zou zijn. Bij het opnieuw construeren van dit alles zijn er een aantal verbeteringen gebeurd die niet meteen te maken hebben met het collaboratie

3.1. Aspecten waarop de thesis tracht te verbeteren

aspect maar wel met de workflow van het classificeren. Dit werd gedaan om het hele programma gebruiksvriendelijker en krachtiger te maken in functie van het hoofddoel van het project.

3.1.5 Uitbreidbaarheid

Het samenstellen van werken uit de oudheid is een zeer vakkennis- en ervaringsintensief proces. Hoewel men luistert naar wat de archeologen hierover te vertellen hebben — wat ze graag zouden zien of kunnen doen — zijn er vele zaken die nu nog niet duidelijk zijn maar in de toekomst zeker aan het licht zullen komen. Dit kan bijvoorbeeld zijn omdat de onderzoekers in kwestie niet goed kunnen uitleggen waar ze naar kijken of hoe ze zoeken: na zovele jaren vertrouwen ze op hun moeilijk te definiëren intuïtie. Anderzijds is het vertalen van het proces om fresco's samen te stellen naar de computer nog niet zo vaak geprobeert in het verleden. Dit betekent dat een goede werkwijze in de realiteit misschien niet zonder meer de efficiëntste is als men de transitie naar virtueel reconstrueren maakt (zoals bij Graphos). Daarbovenop zijn er nog vele kansen om innovatieve nieuwe technieken aan te wenden die niet werkbaar zijn als men enkel over fysische fragmenten beschikt.

Het is dus onwaarschijnlijk dat het laatste woord over de ideale metafoor reeds gezegd is waardoor er een grote kans is dat het platform zal moeten vervangen of herbouwd worden als het niet met uitbreidbaarheid in gedachte ontworpen wordt. Er moeten moeiteloos nieuwe delen aan de applicatie en de onderliggende lagen kunnen toegevoegd worden om snel nieuwe ideeën te incorporeren. Dit alles moet best mogelijk zijn zonder de ervaring van gebruikers met oudere versies of andere gebruikersinterfaces te degraderen.

Data

De eerder besproken uitbreidbaarheid die nodig is manifesteert zich op het niveau van de data bijvoorbeeld bijvoorbeeld op deze manier: een onderzoeker vindt een nieuw algoritme om fragmentparen te rangschikken op “goedheid” of men wil informatie over het dikteverschil tussen twee fragmenten opslaan. Idealiter zouden deze zaken als een attribuut bij een paar moeten kunnen toegevoegd worden zodat elke gebruiker er op kan zoeken en sorteren zonder iets extra te hoeven doen.

Enkel data die op geen enkele manier om te vormen valt naar een attribuut van een enkel paar zal een speciale module vereisen om te kunnen gebruiken. Een vereiste is natuurlijk wel dat dit geen effect mag hebben op de delen van het platform die hier geen weet van hebben.

Visualisatie

Er zijn vele visualisatiemaniéren denkbaar die elkaar kunnen aanvullen bij het volbrengen van het zoek- en classificeerprocess. Zo stelt men zich bij de uitdrukking “fresco's samenstellen” waarschijnlijk een grote puzzel voor waar men ten alle tijde

3. DOELEN & MOTIVATIE

het overzicht kan behouden en stukken proberen te passen.² Deze puzzel visualisatie is visueel aantrekkelijk en biedt het menselijke patroonherkenningsvermogen³ de mogelijkheid om zich van zijn beste kant te laten zien. Echter, door de grote hoeveelheid aan fragmenten en dus mogelijke paren is het moeilijk om hier aan te beginnen. Maar, hoe meer reeds geconfermeerde paren er zijn, hoe duidelijker het globale beeld kan worden. Dit staat toe om een overzicht te krijgen van de vooruitgang en gerichter te zoeken naar stukken die nog ontbreken. Dit is een voorbeeld van een macro-perspectief.

Een alternatief is te beginnen door te kijken naar waar de computer wél goed in is: fragmenten aan elkaar passen en rangschikken naar kans/overeenkomst/et cetera. Door een gemakkelijk navigeerbare lijst op te stellen van alle voorstellen die de reconstructie algoritmes hebben gedaan, kunnen er snel op elkaar passende fragmenten geïdentificeerd worden. Dit kan men zien als een micro-perspectief of *bottom-up* manier om fresco's te reconstrueren. Het is ook de aanpak die in Browsematches gebruikt wordt. Nadat men bijvoorbeeld met deze manier een deel acceptabele paren heeft geïdentificeerd, kunnen deze bijvoorbeeld weergegeven worden als een grote puzzel en dienen zij als beginpunt om verder te puzzelen. Dit maakt het globale beeld veel informatiever: stel dat duidelijke "gatenontstaan in een resem goede fragmentparen, dan kan er gericht gezocht worden naar een fragment dat erin past door te zoeken naar een fragment dat met elk van deze insluiters past (indien het gevonden werd bij de opgraving).

Kortom, het is duidelijk dat een visualisatie die in alle gevallen de meest geschikte is niet bestaat. De beschikbare informatie moet soms gewoon op andere manieren worden weergegeven. Om deze reden is het wenselijk om het mogelijk te maken snel nieuwe visualisaties in te bouwen die kunnen communiceren met andere delen van de applicatie en eventueel de informatie manipuleren.

²Het Grifos programma gaat uit van het idee van kleine beheersbare stukken van de reuzenpuzzel (elk tafelblad zou een verzameling kunnen zijn van stukken die gerelateerd waren, bijvoorbeeld door hun vindplaats)

³Iets waar computers de mens nog altijd niet in evenaren. Een ander therapie subproject onderzoekt wel de mogelijkheid om zogenaamde 'clusters' te identificeren en te gebruiken voor reconstructie.

Hoofdstuk 4

Ontwerp van het project

Eenmaal de belangrijkste vereisten gekend zijn kan er een ontwerp opgetekend worden. Daarom even de grote lijnen die te concluderen vallen uit hoofdstuk 3:

- De data en het visuele aspect van de applicatie moeten zo ontkoppeld mogelijk zijn, elk moet apart uitbreidbaar zijn
- Zeer grote hoeveelheden data moeten vlot kunnen behandeld en genavigeerd worden: zoeken, filteren, sorteren, aanpassen, ...
- De data moet zowel centraal als decentraal toegankelijk zijn en synchronisatie toestaan
- Compatibiliteit met de reeds geschreven onderdelen van het project moet indien mogelijk bewaard blijven
- Om het ontwerp te verifiëren moet een werkend programma gemaakt worden, gebruiksvriendelijkheid, functionaliteit en snelheid zijn hierbij belangrijk

4.1 De grote lijnen

Van alle vereisten die rechtstreeks invloed kunnen uitoefenen op de architectuur, is de splitsing van de data en de visualisatie waarschijnlijk de meest fundamentele. Een beproefde aanpak om dit te realizeren is het ontwerppatroon genaamd *Model-View-Controller (MVC)*¹.

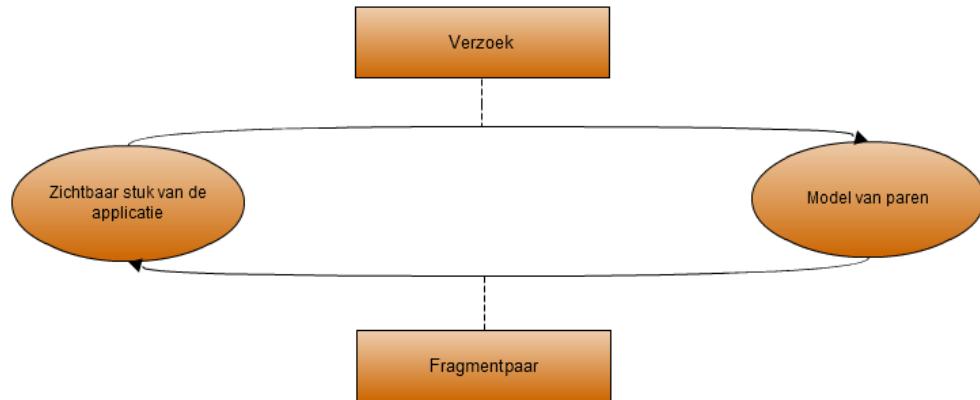
Het doel is een ontwerp te maken waarin we een visualisatiemethode kunnen verbinden aan de juiste databron. Enkele zaken moeten echter nog vastgelegd worden, namelijk: wat is de basiseenheid van data? Waar zal een visualisatiemodule achter vragen? Zoals in figuur 3.1 te zien is, ligt de focus bij dit project op koppelingen van fragmenten in plaats van fragmenten op zich. Er zijn op het eerste zicht

¹Een algemeen overzicht kan men bijvoorbeeld bekomen op <http://en.wikipedia.org/wiki/Model-view-controller>

4. ONTWERP VAN HET PROJECT

twee alternatieven om dit te modelleren. Een eerste mogelijkheid is een soort van *MatchedFragment* die een fragment beschrijft plus een lijst met alle fragmenten die er potentieel aan gekoppeld kunnen worden en op welke locatie. De tweede mogelijkheid is om elk paar een apart object te laten voorstellen (bvb. genaamd *FragmentPair*). Het eerste alternatief lijkt het programmatische voordeel te hebben dat het gemakkelijk is om na te kijken of een fragment reeds “bezet” is. Ook zou het dan mogelijk zijn om bijvoorbeeld een grafe op te stellen door van brokstuk naar brokstuk te springen. Echter, dit soort opstelling bevat veel redundantie, een fragment zal op die manier een verwijzing met attributen naar een fragment bevatten, en dit fragment zal op zijn beurt een identieke omgekeerde verbinding hebben. De redundantie vermijden en een verwijzing als een apart object voorstellen waar beide fragmenten naar kunnen verwijzen is eigenlijk niets anders dan de tweede optie (een *FragmentPair*). Daarbij kan het probleem van hoe de “bezetting” van een object te weten te komen (alsook de grafe, zoals later zal aangetoond worden) opgelost worden door de vereiste zoekfunctionaliteit van het datamodel te benutten.

Eenmaal deze basiseenheid van informatie gekozen, valt de kern van de applicatie volgens MVC uit te beelden als in figuur 4.1.

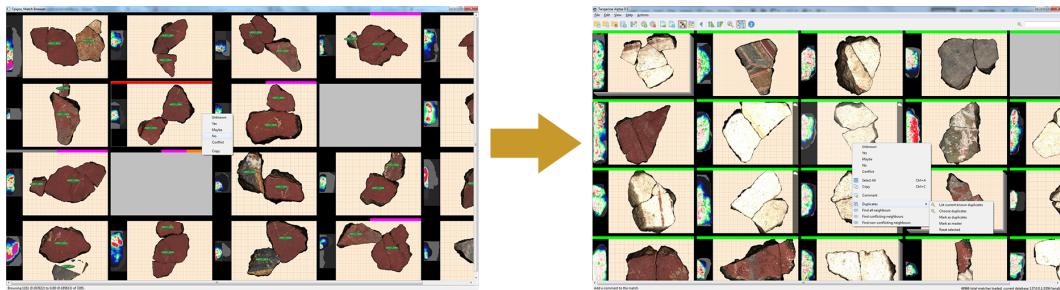


FIGUUR 4.1: Het abstracte model van de applicatie, links staat de *View/Controller* en rechts het *Model*. De controller stuurt een verzoek naar het model voor een bepaalde (sub)set van de data — al dan niet gesorteerd — en het model antwoord met alle paren die voldoen aan de criteria

4.2 Modulariteit

Om het geheel uitbreidbaar te maken is een pluginsysteem gemaakt. Er is een hoofdapplicatie (codenaam “Tangerine”) die de connectie maakt met de databasebeheerlaag en verschillende visualisatieplugins kan laden. Het basissysteem zonder modules bestaat uit een manier om een fragmenten en paren-database in te laden en te kiezen welke module op te starten.

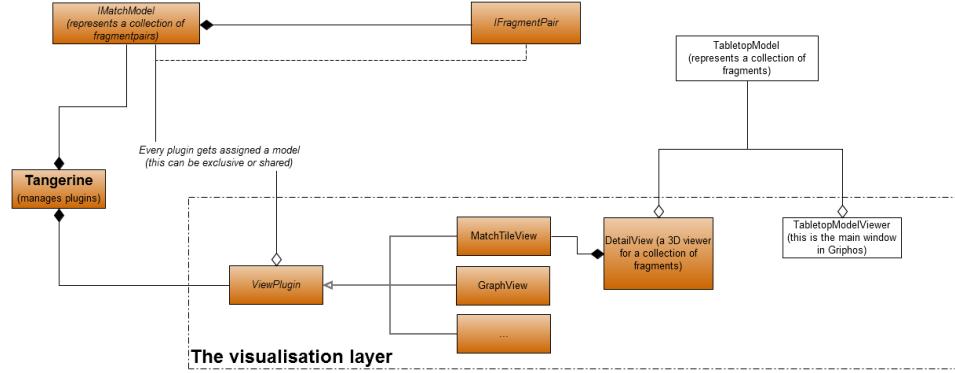
De eerste en meest uitgewerkte daarvan werd *MatchTileView* gedoopt. Het geeft de paren op dezelfde manier weer als het Browsematches prototype, maar is natuurlijk uitgebreid qua mogelijkheden. De bespreking van de nieuwe functionaliteiten komt aan het bod in het hoofdstuk over modules.



FIGUUR 4.2: De manier van weergeven uit Browsematches werd gekopieerd naar het nieuwe platform, met uitbreidingen

Elke module krijgt van de applicatie een model toegewezen, dit kan een blanco model zijn zonder criteria of een gedeeld model. Een gedeeld model betekent bijvoorbeeld dat als er een module beslist om te sorteren op een attribuut zoals “het verschil van de dikte tussen twee fragmenten”, alle modules die gebruikmaken van ditzelfde model opeens over een gesorteerde dataset beschikken. Via speciale signalen worden zij hiervan op de hoogte gebracht, zodat ze kunnen beslissen of het nodig is een actie te ondernemen. Indirecte communicatie via het model is (voorlopig) de enige manier waarop modules elkaar kunnen beïnvloeden. De structuur van de componenten ziet er uit als in figuur 4.3. Merk op dat er een plugin is (*DetailView*) die geen gebruik maakt van fragmentparen maar eerder van een virtueel tafelblad net als Griphos. Zoals eerder aangehaald kunnen fragmentparen automatisch op een tafelblad gezet worden. Dit tafelblad kan dan in 3D weergegeven worden door *DetailView*, waarover later meer.

4. ONTWERP VAN HET PROJECT



FIGUUR 4.3: Een vereenvoudigde kijk op de componenten van de visualisatielaag, het hoofdscherm en het model. De componenten in het wit behoren tot de rest van het thera project.

4.3 De eenheid van informatie: het fragmentenpaar

Speciale eigenschappen, write-through, ...

4.4 Het beheer van de data

Redenering: Alles vloeit voort uit de paren, het is voorspelbaar (simpel voor te stellen) en ondubbelzinnig

Het zelf kunnen maken van paren is van secundair belang (volgens de thesis), zij kunnen door de HumanMatcher ingevoerd worden in de grote database

Griphos is zeer nuttig -> identificatie locatie van fragmenten in de bak, etc. -> Tangerine is het ontbrekende middendeel!

Dit betekent echter niet dat beide perspectieven elkaar uitsluiten, integendeel.
[TODO: verhuizen naar ontwerp van database]

4.4.1 Het oude systeem: XML-bestanden

Eén van de belangrijkste stappen op weg naar een collaboratieve applicatie, is het omvormen van

Het ontwerp van het database model en de implementatie details zijn groot genoeg om hun eigen hoofdstuk te verdienen, zij worden later in hoofdstuk [...] besproken.

4.4.2 Ontleding van de data

...relationeel van aard => SQL database

Paren

Attributen van paren

Complexe informatie die niet in een simpel attribuut past

4.4.3 De vereisten van een database ontwerp

Zoals in sectie [doelen] omschreven moet de applicatie in staat zijn om data op een uitbreidbare manier uit te lezen en te veranderen, liefst zonder

4.4.4 Complexe informatie voorstellen als een attribuut

Meta-attributen, VIEWS

4.4.5 Verschillende iteraties

4.4.6 Databases samenstellen

Geschiedenis bijhouden

4.5 Visualisatie, een manier om met de data te werken

4.5.1 Model-View-Controller

4.5.2 UML diagramma

[maak UML diagramma]

4.5.3 GUI

Met behulp van de Qt toolkit [link]...

Multi-threading voor responsiviteit

4.6 Integratie in thera project

Gebruik van bibliotheken, ... Extensie door refactoring: FragmentConf -> IFragmentConf ==> SQLFragmentConf | FragmentConf Hierdoor is het nodig om de reeds bestaande code van het thera project om te zetten naar het gebruik van IFragmentConf waar mogelijk maar het aanmaken van FragmentConf anderzijds of FragmentConf ==> SQLFragmentConf Alternatieve oplossing, geen veranderingen in thera code maar SQLFragmentConf sleept dan veel onnodige ballast mee van FragmentConf

4.7 Uitbreidbaarheid

[afbeeldingen invoegen]

Een module die ingeladen wordt weet niets over de hoofdapplicatie of andere modules behalve de huidige database. Deze informatie zit vervat in een object dat een model van een aantal paren voorstelt. Dit werd mogelijk gemaakt door een zogenaamde Model-View(-Controller)² [citaat] aanpak. Een module kan vragen om een applicatie-wijd of een nieuw model te krijgen. Met een applicatie-wijd model kan invloed uitgeoefend worden op andere modules die hetzelfde model gebruiken. Dit is handig bijvoorbeeld als men in één module een verzameling paren filtert en selecteert en men een andere module deze selectie wil visualizeren. applicatie-modellen zijn dus bijzonder nuttig voor visualisatie plugins. Om de werkbaarheid van het systeem uit te testen werd een voorbeeldmodule ontwikkeld die alle huidige paren in een grafe plaatst en deze met een ontwarringsalgoritme probeert te plaatsen, zodat men een globaler beeld kan krijgen van de huidige selectie.

[afbeelding invoegen, selectie naar grafe transit!]

²De controller functionaliteit zit in de applicatie verdeeld bij het model en de view

Hoofdstuk 5

Bespreking van het database ontwerp

5.1 Vereisten

Gebruikers moeten een externe database server kunnen aanspreken om zo te werken op de meest actuele data. Het moet ook steeds mogelijk zijn om op een eigen kopie van de data te werken

5.2 Alternatieven

NoSQL (Cassandra, ...) SQL XML database (oud)
Redenering!

5.3 Externe database traag, interne database snel

Het grote probleem dat duidelijk werd na het rechtstreeks werken met externe databases, is dat zelfs op een lokaal netwerk de verzoeken een zeer grote vertraging opleverden.

De redenen hiervoor waren natuurlijk de vele queries die elk object kan versturen om zijn attributen op te halen of te veranderen.

Nog een groot voordeel bij het maken van dergelijk systeem is dat er gebruik kan gemaakt worden van niet-desctructieve veranderingen. Op het einde van een werkssessie kan men de volledige lokale database ofwel in de hoofd database invoegen ofwel gewoon verwijderen.

Het design moest transparant zijn, zowel een rechstreekse connectie als een gebufferde connectie zouden voor de eindgebruiker en de ontwikkelaar aan de buitenkant hetzelfde lijken.

5. BESPREKING VAN HET DATABASE ONTWERP

Idee: Vervang object ID met object hash voor snellere match-to-match identificatie zonder een globale id nodig te hebben

5.4 Slimme client of slimme server?

Het programma bevat alle nodige functionaliteit en de server is in dat opzicht gewoon de specifieke database waarnaar het een verwijzing heeft.

Voor het gebruik op kleine apparaten (zoals tablets) zijn de mogelijkheden voor computationeel zware activiteiten niet zo uitgebreid. Daarenboven moet ervoor gezorgd worden dat de batterijduur van dit apparaat niet te hard beknot wordt door het gebruik van de applicatie. Daarom zal de transitie naar zeer mobiele platformen enkel mogelijk worden indien er een simpele client applicatie kan ontwikkeld worden die op de server vertrouwd om de juiste berekeningen te maken.

In de thesisperiode is geen tijd gevonden om dit concept uit te werken maar er zijn wel plannen gemaakt waardoor dergelijke applicatie op een efficiënte manier zou kunnen ontwikkeld worden. De gebruikersinterface Tangerine voert alle functies met betrekking tot de achterliggende data uit met behulp van een grote bibliotheek van klassen die op zich niets met de interface te maken hebben. Er zou dus een alternatief programma gebouwd kunnen worden die in plaats van met een grafische interface kan bediend worden via een zelfgemaakte protocol. Dit soort ontwerp wordt vaak aangetroffen in de UNIX wereld, waar er een enkele server-variant van een programma bestaat en meerdere mogelijke clients die ervan gebruik maken. Het bekendste is misschien wel de X server.

Materialized views!!!

Model change batching!!! (startBatch/endBatch)

Incompatibiliteiten tussen *SQL

Analyse van datatoegangspatroon: vooral SELECT, ORDER BY, GROUP BY —> veelvuldig gebruik van indexes

Metadata preloading (na het testen van de snelheid van het ophalen van metadata over een internetconnectie, werd besloten om...)

De gevaren van een stale cache! (en ook de gevaren van een stale window, ook een soort cache)). Oplossing altijd een request sturen om te dubbelchecken?! —> te traag

optimize voor fast reads —> inserts kunnen lokaal gedaan worden, updates hopelijk niet zo veel of lokaal

Recheck om de X seconden: doenbaar indien materialized views met indices! (eigenlijke window query < 1 msec). Systeem zou 1000'en gebruikers kunnen ondersteunen, zeker met een grotere cache

Cache = write-through

Vele disciplines van de computerwetenschappen

dynamisch zoekopdrachten genereren

SQLite formaat maakt database gemakkelijk te delen (USB-stick) <- geen internet connectie

Detecteren van incompatibiliteiten EN veranderingen met reguliere expressies:

Aanpakken voor synchronization: moeilijk: changelogging functionaliteit, triggers, ... (mogelijk maar moeilijk cross-db en error-prone) "gemakkelijk": Maak gewone queries zeer goedkoop

Vereisten: Scaleerbaarheid (XML schaalt NIET) Voordeel van XML is echt wel dat het een mooi outputformaat is voor matchers... import capaciteit voorzien ==> import naar temp SQLite db en merge

Ondersteuning voor "meerdere snelheden", minder modules hebben is niet erg, elke additie is uitbreiding. Zo ook minder/geen versie van db-schema problemen dependency scanning!

5.5 Benchmarking

De query cache staat af Om de variabiliteit van de filesystem (nederlands) cache een beetje buiten spel te zetten zijn al deze routines "opgewarmd"

Pagination Late row lookup

Ideaal = minder dan een seconden voor elke gegeven query vanuit een gebruikersinteractie standpunt (System Response Time and User Satisfaction pagina 5)

Effect van DB configuratie (veel geheugen...)

Suggested workaround voor het text probleem -> sphinx, restrict fragment names (niet ZO gemakkelijk), string + nummer Suggested workaround voor het indexing probleem (zoals gezien voor status IN (...)) -> force een index?! dunno... hij pakt in ieder geval de verkeerde!

High Performance MySQL, Second Edition, O'REILLY, ISBN: 978-0-596-10171-8
MySQL Reference Manual for version 5.1

<http://nlp.stanford.edu/IR-book/html/htmledition/permute-and-indexes-1.html> (dit is hoe wildspeed werkt) (LIKE performance lijkt niet zo slecht in Postgres, het is de sorting eerder...) <http://www.slideshare.net/techdude/how-to-kill-mysql-performance>

[http://stackoverflow.com/questions/1540590/how-to-speed-up-like-operation-in-sql-postgres-preferably -- use trigrams \(fail\)](http://stackoverflow.com/questions/1540590/how-to-speed-up-like-operation-in-sql-postgres-preferably--use-trigrams-fail), MAAR BETER IN 9.1 (future research) <https://cgsrv1.arrc.csiro.au/blog/201> views-for-postgresql/

Hoofdstuk 6

Modules

Hier komen de modules!!!!

6.1 MatchTileView

6.2 Proof of concept: GraphView

Hoofdstuk 7

Tests & Vergelijkingen

Een goede manier om te weten te komen of het project zoals dit nut heeft gehad is om tests uit te voeren. Er zal worden vergeleken tussen de reeds bestaande oplossingen voor het classificeren van fragmentparen en de nieuwe applicatie waar mogelijk. In sommige gevallen zijn er nieuwe zaken toegevoegd die vroeger op geen of slechts manuele wijze mogelijk waren, hier zal worden vergeleken met (evt. fictieve) alternatieve implementaties. Voor het geval er helemaal geen vergelijking tussen alternatieven mogelijk is, zal een zo goed mogelijke maatstaf bedacht worden zodat er toch een beeld ontstaat van hoe de applicatie het doet.

7.1 Use cases

Elke gebruiker wenst met een applicatie een bepaald doel te bereiken, in dit geval is dat het classificeren van fragmentparen. Hiervoor zullen de mogelijkheden die het programma aanbiedt hopelijk zowel voldoende als gemakkelijk in gebruik zijn.

7. TESTS & VERGELIJKINGEN

- 7.1.1** Vind uit bak ONGESORTEERD het paar dat het meeste ruimte inneemt
- 7.1.2** Vind alle mogelijke paren die conflicteren met een waarschijnlijk paar
- 7.1.3** Vind alle mogelijke paren die grenzen aan het geselecteerde paar maar niet conflicteren ermee
- 7.1.4** Voeg een andere database bij de huidige, prefereer de veranderingen van de andere als er conflicten zijn
- 7.1.5** Gebruik conflict detectie & 3D voorstelling om correcte paren te identificeren
- 7.1.6** Duid aan dat een verzameling mogelijke paren eigenlijk hetzelfde paar zijn (duplicaten)
- 7.1.7** Bekijk alle duplicaten van een paar
- 7.1.8** Vind alle paren die geklassificeerd staan als 'misschien'/'maybe', schrijf bij een paar je mening ('comment')
- 7.1.9** Exporteer een database naar een leesbaar formaat (XML)
- 7.1.10** Importeer een database van een leesbaar formaat (XML)
- 7.1.11** Kopiëer een aantal paren naar Griphos
- 7.1.12** Selecteer een aantal paren om te groeperen en in 3D weer te geven
- 7.1.13** Vindt wat je weet zijn: gebruik de zoek- en sorteervoorzieningen om snel een gewenst deel van de verzameling paren te zien
- 7.1.14** Vindt wat je niet weet zijn: gebruik de zoek- en sorteervoorzieningen om snel veelbelovende paren te vinden
- 7.1.15** Werken op een externe database (gebruiksgemak)
- 7.1.16** Databases samenstellen: conflicten oplossen

7.2 Snelheid (objectieve tests)

- 7.2.1** Opstartsnelheid
- 7.2.2** Inladen van fragmenten

Meet het inladen van fragmenten, verschil tussen browsermatches en tangerine (intern en extern)

7.2.3 Navigeren tussen schermen

Meet het navigeren tussen schermen, sorteren, filteren

Teruggaan naar eerder schermen

Door een caching mechanisme...

7.3 Gebruiksgemak (subjectieve tests)

Subjectieve waardeschattingen van gebruikers

7.3.1 Intuitiviteit

Het doelpubliek van de applicatie bestaat in de nabije toekomst uit archeologen en eventueel later uit amateurs die helpen met de classificatie. Qua gebruikersprofiel passen archeologen in het plaatje van een gebruiker met veel domeinkennis (classificeren), maar weinig kennis of ervaring met het gebruik van geavanceerde computerapplicaties. Anders gezegd: de gebruikers weten wat ze willen en moeten doen, maar niet noodzakelijk hoe ze het moeten doen in de applicatie.

Voor elk programma kan training voorzien worden zodat er uiteindelijk optimaal gebruik van kan gemaakt worden, maar het doel is om deze training tot een minimum te beperken. Dit houdt in dat de functies ontdekbaar

Informatie-popups (eenmalig/meermalig)

7.3.2 Responsiviteit

// MOVED NAAR DOELEN.TEX Voor het dagdagelijkse gebruik van een applicatie is het van groot belang dat de gebruiker het niet hinderlijk vindt om ermee te werken. In enkele werken over gebruikersinterface ontwerp [aan elkaar?] [citatie] wordt erop gewezen dat de snelheid waarmee een applicatie reageert een sleutelfactor is voor het gebruiksgemak. Een algemene vaststelling: de tijd die een operatie mag innemen is omgekeerd evenredig met de frequentie waarmee deze operatie moet uitgevoerd worden. Deze regel in acht nemend is het duidelijk dat bijvoorbeeld het inladen van een scherm, het veranderen van een attribuut, het filteren en sorteren en dergelijke meer acties zijn die met de grootst mogelijke snelheid moeten worden uitgevoerd.

Voor al deze operaties is sinds het eerste ontwerp van de applicatie rekening gehouden met de implicaties van elke beslissing op de snelheid. Hierdoor is de snelheid van de applicatie op elke vlak verbeterd tegenover zijn voorgangers, soms op dramatische wijze. Om dit objectief vast te stellen zijn er een paar tests uitgevoerd:

Hoofdstuk 8

Besluit en toekomstige toepassingen

Net zoals een gebroken fresco in feite een puzzel is, kan men het thera project zien als de som van vele delen die in elkaar passen. Dit thesisproject is bedoeld als een stuk dat een ander perspectief biedt op het geheel en het in staat moet stellen om meer en sneller resultaten te boeken. Het complementeert de bestaande aanpakken en zorgt ervoor dat de resultaten van de automatische paarherkenning nog nuttiger gebruikt kunnen worden. Omdat het validatiewerk noodzakelijk door mensen moet gebeuren, kan dit niet zonder meer opnieuw door een algoritme gedaan worden indien er iets misloopt. De in dit thesisproject geproduceerde componenten proberen er onder andere voor te zorgen dat dit zo min mogelijk voorkomt.

Op het vlak van ontginnung van nuttige informatie met nieuwe visualisaties en nieuwe manieren om de juiste patronen te ontdekken zijn er natuurlijk nog steeds vele opportuniteiten. Want — zoals opgemerkt in een recente paper over het thera project [citaat: siggraph submission 2011] — het vinden van de juiste paren is zoals zoeken naar een naald in een hooiberg. Met elke nieuwe toevoeging aan de mogelijkheden van het platform is er de kans dat deze een manier is om de hooiberg te verkleinen, door te lichten met X-stralen of gewoonweg op een grote krachtige magneet in een windtunnel te plaatsen. Naar deze laatste methode is iedereen natuurlijk op zoek. Tot dan is het zeker belangrijk dat men gemakkelijk kan experimenteren alsook bijhouden en opvragen welk deel van de berg reeds doorkamt is, waar de gevonden naaldrijke aders zitten en wat hun eigenschappen zijn. Wie weet zijn de naalden immers niet van metaal...

8.1 Toekomst

Met dit project is ook een verdere stap gezet naar mobiele toepassingen. Het data-luik staat bijvoorbeeld toe om applicaties voor tablets te schrijven die beroep kunnen doen op een externe database om een groot deel van het zware werk over te nemen. Deze soort programma's kunnen het manueel verifiëren van paren sneller en aangenamer

8. BESLUIT EN TOEKOMSTIGE TOEPASSINGEN

maken. De huidige werkwijze is als volgt: nadat er een resem waarschijnlijke paren zijn geïdentificeerd, kan men de kisten met fragmenten uit de opslagruimte halen en nakijken welke er écht passen. Gezien de grote hoeveelheid brokstukken is het niet mogelijk om ze allemaal bij de hand te houden. Daardoor duurt het altijd even voor de gewenste fragmenten gevonden worden. In het slechtste geval wordt er gewerkt op een (krachtige) desktop. Hierdoor is het nodig is om ofwel de namen en locaties van de fragmenten te onthouden, of een heleboel afbeeldingen af te drukken. Na het fysisch testen van de fragmenten moet men dan terug naar de desktop om de bevindingen in te geven. Gelukkig behoort een laptop ook tot de mogelijkheden hoewel applicaties als Browsematches en Grifhos niet bepaald licht zijn. Het performantieprobleem wordt natuurlijk reeds een deel verholpen door het invoegen van een externe database. Ook kan men nu gemakkelijker met meerdere mensen en laptops tegelijkertijd werken aan de validaties wegens de automatische synchronisatie. Een stap verder zou zijn om een tablet te gebruiken. Er zijn reeds in het verleden experimenten geweest binnen het thera project om aanraakgevoelige omgevingen te maken en de hoop is dat tesamen met de resultaten van deze thesis er in de toekomst iets concreets van gemaakt kan worden.

Bijlagen

Bijlage A

Numerieke resultaten

Nog niets hier

Bibliografie

- [1] B. J. Brown, C. Toler-Franklin, D. Nehab, M. Burns, D. Dobkin, A. Vlachopoulos, C. Doumas, S. Rusinkiewicz, and T. Weyrich. A system for high-volume acquisition and matching of fresco fragments: Reassembling Theran wall paintings. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 27(3), Aug. 2008.
- [2] J. A. Hoxmeier, P. D, and C. D. Manager. System response time and user satisfaction: An experimental study of browser-based applications. In *Proceedings of the Association of Information Systems Americas Conference*, pages 10–13, 2000.
- [3] J. Nielsen. *Usability Engineering*. Morgan Kaufmann, San Francisco, 1994.
- [4] B. Shneiderman. Response time and display rate in human performance with computers. *ACM Comput. Surv.*, 16:265–285, September 1984.
- [5] J. Spolsky. User interface design for programmers. URL: <http://www.joelonsoftware.com/uibook/fog000000249.html>, laatst nagekeken op 25/05/2011.
- [6] C. Toler-Franklin, B. Brown, T. Weyrich, T. Funkhouser, and S. Rusinkiewicz. Multi-feature matching of fresco fragments. In *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, Dec. 2010.
- [7] C. Toler-Franklin, B. Brown, T. Weyrich, T. Funkhouser, and S. Rusinkiewicz. Needles in a haystack: Finding matches among fresco fragments. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, Feb. 2011.

Fiche masterproef

Student: Nicolas Hillegeer

Titel: Een collaboratief platform voor het efficiënt reconstrueren van fresco's

Engelse titel: Een collaboratief platform voor het efficiënt reconstrueren van fresco's

UDC: 621.3

Korte inhoud:

Hier komt een heel bondig abstract van hooguit 500 woorden. L^AT_EX commando's mogen hier gebruikt worden. Blanco lijnen (of het commando \par) zijn wel niet toegelaten!

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Thesis voorgedragen tot het behalen van de graad van Master in de ingenieurswetenschappen: computerwetenschappen

Promotor: Prof. dr. ir. P. Dutré

Assessoren: Ir. N/A
N/A

Begeleider: Dr. ir. B.J. Brown