Afstudeerverslag

Onderzoek naar de mogelijkheden om platformonafhankelijke mobiele applicaties te maken en het uitbreiden van de Mobile Flight Assistant Applicatie op basis van de resultaten uit het onderzoek

Versie:	1.0
Auteur:	Alvin Nutbeij
Studentnummer:	50128
Examinatoren:	Gerard Tuk, Robin Collard
Bedrijfsmentor:	Jorrit van der Ven
Datum:	31 mei 2012

Referaat

Alvin Nutbeij, Afstudeerverslag onderzoek platformonafhankelijke mobiele applicaties en uitbreiden van de Mobile Flight Assistant applicatie, NCIM Groep, Leidschendam, 2012

Dit afstudeerverslag beschrijft de aanpak, de werkzaamheden en de daaruit voortgekomen resultaten van Alvin Nutbeij bij het uitvoeren van de afstudeeropdracht, het onderzoeken naar de mogelijkheden voor platformonafhankelijke applicaties en het uitbreiden van de Mobile Flight Assistant applicatie. Het project is uitgevoerd in opdracht van de NCIM Groep gedurende de periode van 6 februari 2012 tot en met 31 mei 2012.

Descriptoren:

Mobiele applicaties
Luchtvaart
Experimenten
Platform Onafhankelijkheid
Onderzoek
Virtuele wereld
3D Engine
Landkaarten

Voorwoord

Voor u ligt het afstudeerverslag van Alvin Nutbeij. Dit verslag is de tekstuele beschrijving van de uitgevoerde activiteiten bij het onderzoeken naar de mogelijkheden om platformonafhankelijke mobiele applicaties te ontwikkelen en het uitbreiden van de Mobile Flight Assistant applicatie voor de NCIM-Groep. Tijdens dit project heb ik tijdens mijn opleiding, Informatica aan de Haagse Hogeschool, opgedane kennis toegepast in de praktijk.

Ik wil het management en medewerkers van de NCIM-Groep bedanken voor de gelegenheid om deze opdracht uit te voeren en voor de prettige omgeving waarin ik terecht kwam. In het bijzonder wil ik Jorrit van der Ven bedanken voor de begeleiding vanuit de NCIM-Groep. Daarnaast wil ik in het bijzonder Gijs Pannebakker en Arjan Kodde bedanken voor het mij wegwijs maken binnen de NCIM-Groep. Ook wil ik Gijs Pannebakker bedanken voor het inwerken in het Mobile Flight Assistant project. De medewerkers van de NCIM-Groep die tijdens mijn afstudeerperiode in het hoofdkantoor aanwezig zijn geweest wil ik allen bedanken voor het meedenken bij problemen. Tevens wil ik Gerard Tuk en Robbin Collard bedanken voor hun begeleiding vanuit school.

Delft, 31 mei, Alvin Nutbeij

Inhoudsopgave

1 Inleiding 5
2 De opdrachtgever 6
3 De opdracht 7
3.1 Kader van de opdracht 7
3.2 Probleemstelling 7
3.3 Doelstelling 8
3.4 Methoden en technieken 8
3.5 Op te leveren producten 9
3.6 Planning 9
4 Onderzoek platformonafhankelijke ontwikkeling 10
4.1 Vooronderzoek 10
4.2 Onderzoeksplan 14
4.2.1 Probleemstelling 14
4.2.2 Doelstelling 14
4.2.3 Onderzoeksontwerp 15
4.2.4 Deelvragen 15
4.2.5 Tijdspad 16
4.3 Uitvoer van het onderzoek 17
4.3.1 Hypothese 17
4.3.2 Deelvragen 18
4.3.3 Experimenten 21
4.4 Conclusie en aanbevelingen 24
4.4.1 Conclusie 24
4.4.2 Aanbevelingen 24
4.5 Gebruik nieuwe kennis 26
5 Uitbreiden van de MOFLAS applicatie 27
5.1 Huidige status 27
5.2 Ontwikkelmethode 28
5.2.1 Requirements 28
5.2.2 Ontwikkelen 28
5.2.3 Sprint bord 28
5.2.4 Evaluatie 28
5.3 Requirements 29

5.4 Planning 30
5.4.1 3D Engine Selectie 30
5.4.2 Kaart tegels 30
5.4.3 Weergegevens & Flight planning 30
5.5 3D Engine Selectie 31
5.5.1 3D engines 31
5.6 Kaart selectie 34
5.7 Klassendiagram 36
5.8 Use Case diagram 36
5.9 Opbouw van de uitbreiding 37
5.9.1 Singleton Klassen 37
5.9.2 Singleton implementatie 39
5.9.3 Object Pool klasse 41
5.9.4 Object Pool implementatie 41
5.10 Knelpunten 43
5.10.1 Kaart in 3D 43
5.10.2 Geheugenproblemen 45
5.10.3 Render problemen 47
6 Evaluatie 51
6.1 Productevaluatie 51
6.2 Procesevaluatie 53
7 Bewijs beroepstaken 54
7.1 Uitvoeren analyse door definitie van requirements 54
7.2 Ontwerpen systeemdeel 54
7.3 Bouwen applicatie 54
7.4 Overig 55
7.4.1 Voorbereiden en opstarten softwareontwikkeltraject 55
7.4.2 Ontwerpen, bouwen en bevragen van een database 55
8 Bronnen 56
Lijst van figuren 58
Lijst van tabellen 58

1 Inleiding

Dit verslag is geschreven voor mijn examinatoren van de Haagse Hogeschool en de gecommitteerde op de examenzitting. Daarnaast kan dit verslag gebruikt worde door studenten met een vergelijkbaar project.

Met dit verslag wil ik aantonen dat ik dat wat ik tijdens mijn opleiding heb geleerd kan toepassen op een praktijksituatie en daarmee dat ik voldoende opgeleid ben om de stap naar de beroepspraktijk te nemen.

Het bedrijf waar ik de opdracht heb uitgevoerd, de NCIM Groep, houdt zich sinds kort bezig met de ontwikkeling van mobiele applicaties. De opdrachtgever wil weten of deze applicaties platformonafhankelijk kunnen worden ontwikkeld. Daarnaast wil de opdrachtgever ook dat een van de projecten, de Mobile Flight Assistant wordt uitgebreid.

Hieronder worden de hoofdstukken van dit verslag kort beschreven.

In hoofdstuk 2 wordt het bedrijf, de NCIM Groep, waarbinnen ik de opdracht heb uitgevoerd beschreven.

In hoofdstuk 3 worden de belangrijkste onderdelen van de opdrachtomschrijving en het Plan van Aanpak beschreven.

In hoofdstuk 4 wordt het onderzoek naar de mogelijkheden om platformonafhankelijke applicaties te ontwikkelen beschreven.

In hoofdstuk 5 wordt het uitbreiden van de Mobile Flight Assistant applicatie beschreven.

In hoofdstuk 6 evalueer ik de uitvoer en de resultaten van de opdracht.

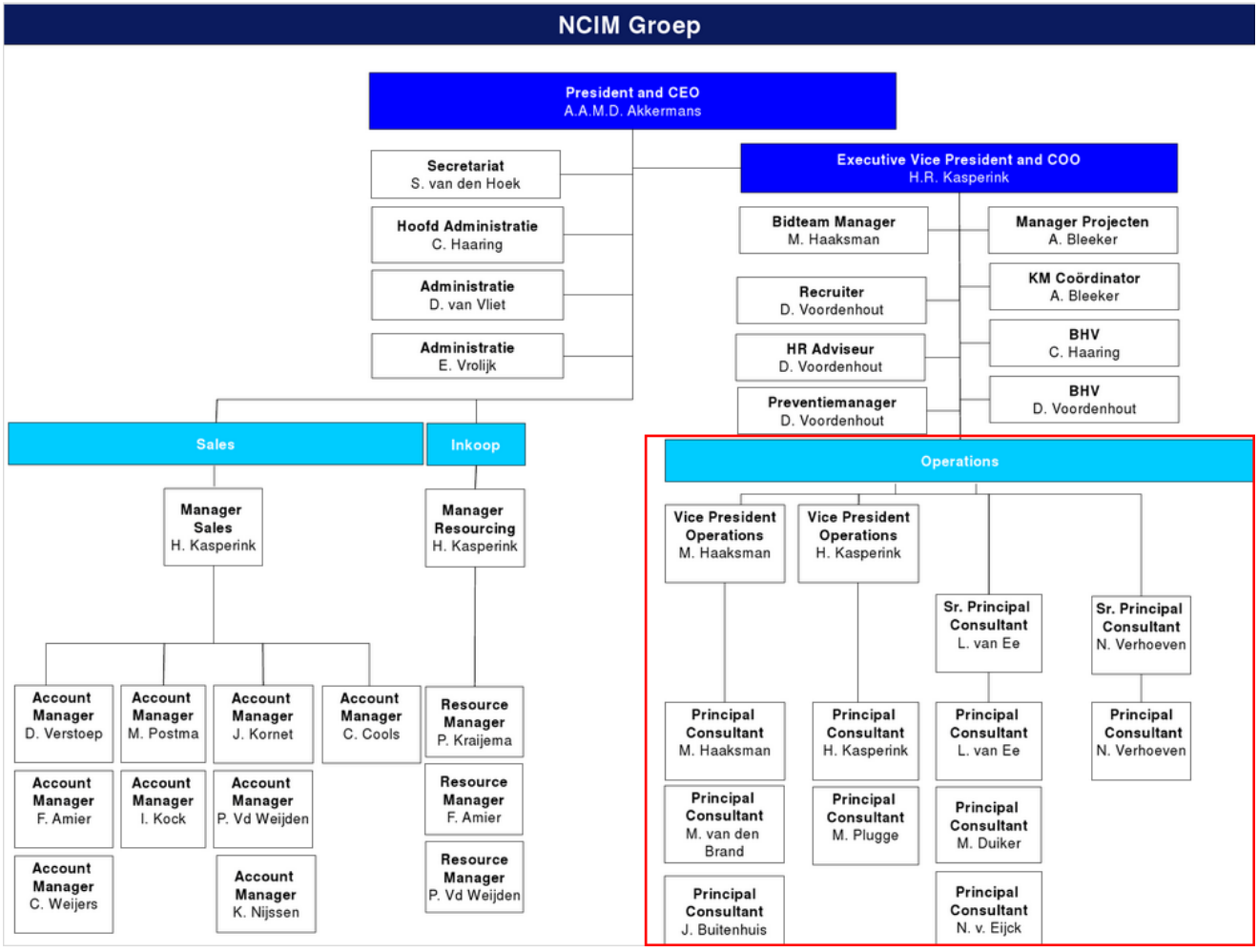
In hoofdstuk 7 vat ik de behaalde beroepstaken samen.

2 De opdrachtgever

De NCIM Groep is gespecialiseerd in het detacheren van professionals voor projectmanagement, consulting, softwareontwikkeling en system management. De NCIM Groep is begonnen in 1988 en heeft vestigingen in Leidschendam, Eindhoven, Brussel, Camberly (Verenigd Koninkrijk) en Fort Worth (Verenigde Staten). Het hoofdkantoor is gevestigd in Leidschendam.

Naast detachering lopen binnen de NCIM Groep ook interne projecten. Deze projecten zijn bedoeld voor professionals die tussen twee detacheringsklussen zitten. Daarnaast kunnen ook afstudeerders aan deze projecten meewerken.

De NCIM Groep is ingedeeld in drie afdelingen; verkoop, operations en overige. Onder overige worden onder andere de administratie, secretaresse en recruiters gezien. De operations afdeling is de grootste afdeling, hieronder vallen alle professionals die gedetacheerd worden. De verkoop afdeling is aanzienlijk kleiner, hieronder vallen de medewerkers die zorgen dat professionals gedetacheerd worden bij bedrijven. Dit is terug te zien in het organogram uit Figuur 2.1.



Figuur 2.1 Organogram NCIM Groep

3 De opdracht

In dit hoofdstuk wordt de afstudeeropdracht beschreven.

3.1 Kader van de opdracht

De opdrachtgever en algemeen directeur van de NCIM Groep, Toon Akkermans, bezit een sportvliegtuig waar hij regelmatig in vliegt. De handboeken, met daarin frequenties en dergelijke, zijn zeer groot. Daarom kwam hij op het idee om deze te vervangen door een iPad applicatie.

Nadat een demo applicatie, genaamd Mobile Flight Assistant (MOFLAS), gemaakt was heeft de NCIM Groep hiermee op een beurs gestaan waar deze applicatie enthousiast werd ontvangen. Zodoende wil de NCIM Groep zich meer gaan richten op het ontwikkelen van mobiele applicaties en willen ze meer demonstratie materiaal hebben om te laten zien wat ze kunnen.

3.2 Probleemstelling

De NCIM-Groep wil zich gaan richten op de mobiele applicatie markt. Om de mogelijkheden binnen deze markt te onderzoeken zijn ze bezig met de ontwikkeling van de iPad applicatie “Mobile Flight Assistant”. Dit is een applicatie die sportpiloten helpt tijdens de vlucht.

Deze applicatie bevat momenteel slechts beperkte functionaliteiten en is geschreven in native Objective-C code. Deze applicatie is al eerder als demo op een beurs getoond en heeft veel positieve response gehad. De NCIM-Groep wil deze applicatie dus verder ontwikkelen.

Om aan de wensen van klanten te kunnen voldoen moeten applicaties echter op verschillende platforms werken. Om de applicaties voor elk platform te maken kost veel tijd en daarmee geld. De NCIM-Groep is daarom op zoek naar een mogelijkheid om applicaties zo veel mogelijk platform onafhankelijk te maken.

De Mobiel Flight Assistant is momenteel nog niet uitgebreid genoeg om daadwerkelijk klanten binnen te kunnen halen. Daarom moet de applicatie verder uitgebreid worden.

3.3 Doelstelling

Om applicaties platformonafhankelijk te kunnen ontwikkelen moet een geschikt framework gezocht worden. Ook moeten de minimale eisen aan dit framework nog worden vastgesteld. In eerste instantie is de NCIM Groep geïnteresseerd in het gebruik van HTML5 als vervanging van native code, mocht dit geen uitkomst bieden kunnen ook andere methoden onderzocht worden.

Om de bevindingen van het onderzoek te ondersteunen dient een Proof of Concept applicatie gemaakt te worden om de mogelijkheden te laten zien. Dit Proof of Concept bestaat slechts uit een kleine demo applicatie die geen werkelijke functie hoeft te hebben.

De NCIM Groep wil naast de kennis over platformonafhankelijke ontwikkeling ook de MOFLAS applicatie uitgebreid zien worden met als doel deze in te kunnen zetten op beurzen en zodoende klanten te overtuigen voor de NCIM Groep te kiezen. Daarnaast wil de opdrachtgever de applicatie ook gebruiken tijdens het vliegen.

De functionaliteiten die geïmplementeerd moeten worden zijn:

- Flight planning:
  De gebruiker moet middels het invoeren van waypoints een route kunnen vaststellen. Vervolgens moet de opgegeven route in 3D worden weergegeven.
- Live weergegevens:
  De bedoeling is dat de applicatie realtime weergegevens laat zien. Deze worden uitgezonden door een ADS-B (Automatic Dependent Surveillance-Broadcast) grondstation. Het signaal moet dus opgevangen, gedecodeerd en weergegeven worden.

3.4 Methoden en technieken

Tijdens de uitvoer van de opdracht worden de volgende methoden en technieken gebruikt:

- Literatuur onderzoek, gebaseerd op de methode van Nel Verhoeven [1],
- Experimenteel onderzoek,
- Scrum,
- UML.

De NCIM Groep gebruikt nog geen methodieken voor onderzoek, daarom heb ik de methoden zoals ik die op school geleerd heb gebruikt. De NCIM Groep gebruikt al Scrum voor het software ontwikkeltraject en UML voor het maken van diagrammen. Daarom heb ik deze methoden overgenomen.

3.5 Op te leveren producten

Aan het einde van de opdracht worden de volgende producten opgeleverd aan de NCIM Groep:

- De resultaten van het vooronderzoek,
- Het onderzoeksrapport,
- Het platformonafhankelijke Proof of Concept en de daarbij behorende code,
- De documentatie behorende bij de uitbreidingen, waaronder:
    - UML Diagrammen,
    - Bijgewerkte versie van het product backlog,
    - Sprint backlogs.
- De uitbreiding van de applicatie en de daarbij behorende code.

3.6 Planning

In deze paragraaf wordt de planning omtrent de afstudeeropdracht besproken. Omdat de opdracht uit twee delen bestaat zijn binnen deze delen specifieke planningen gemaakt. De planning, zoals te zien in Bijlage A, bevat een aantal mijlpalen. De mijlpalen zijn te zien in Tabel 3.1.

Oplevering	Mijlpaal	Omschrijving
Week 5	Onderzoeksrapport	Het onderzoeksrapport met daarin mijn bevindingen en aanbevelingen omtrent de ontwikkeling van platformonafhankelijke mobiele applicaties.
Week 11	3D Flightplanning	De 3D Flight planning uitbreiding is geïmplementeerd in de MOFLAS applicatie.
Week 15	Live Weergegevens	De live weergegevens uitbreiding is geïmplementeerd in de MOFLAS applicatie.
Week 17	Afstudeerdossier	Het afstudeerverslag wordt samen met alle overige documentatie ingeleverd .

Tabel 3.1 Mijlpalen

4 Onderzoek platformonafhankelijke ontwikkeling

Het eerste deel van de afstudeeropdracht was het onderzoeken naar de mogelijkheden om platformonafhankelijk mobiele applicaties te ontwikkelen. In dit hoofdstuk worden de aanpak en de resultaten uit dit onderzoek besproken.

4.1 Vooronderzoek

Voordat ik begonnen ben aan het onderzoek heb ik mijzelf eerst georiënteerd op het onderwerp. Op basis van deze oriëntatie kwam ik er achter dat er een aantal onderwerpen waren waar ik meer duidelijkheid over wilde hebben voordat ik aan het onderzoek begon. Omdat deze informatie heb nuttig kon zijn om een betere planning te kunnen maken heb ik besloten de vragen te beantwoorden middels een vooronderzoek in plaats van deze op te nemen als deelvragen van het onderzoek. De volgende vragen worden in het vooronderzoek beantwoord:

1. Welke mobiele platformen spelen een rol?
2. Welke methoden bestaan er om platformonafhankelijke applicaties te ontwikkelen?
3. Welke tools & frameworks zijn beschikbaar?
4. Welke eisen stellen de applicatiewinkels van de platformen aan applicaties?

Op de volgende pagina's wordt voor elk van de bovenstaande vragen de totstandkoming hiervan besproken. Daarnaast worden de resultaten ook kort samengevat.

Vraag 1: Welke mobiele platformen spelen een rol?

Momenteel zijn er een groot aantal mobiele platformen beschikbaar. Omdat bestaande tools, of eventueel zelf gemaakte tools, mogelijk niet voor alle platformen kunnen ontwikkelen wilde ik eerst bepalen wat de prioriteiten zijn. Het antwoord op deze vraag geeft mij dus inzicht welke platformen in het onderzoek als criteria meegenomen worden.

Het doel van platform onafhankelijke applicaties ontwikkelen is het bereiken van zoveel mogelijk mensen vanuit één code base. Daarom heb ik in eerste instantie gekeken naar de marktaandelen van de grootste platformen.

Van de platformen die momenteel het grootst zijn valt vooral op dat zowel BlackBerry als Symbian in een dalende lijn zitten. Terwijl BlackBerry OS nog volop ontwikkeld wordt, is Nokia gestopt met de ontwikkeling van Symbian en stopt vanaf 2016 met de ondersteuning ervan. Daarnaast is ook opvallend dat het nieuwe platform van Microsoft, Windows Phone, nog vrijwel geen marktaandeel heeft. Dit was voor mij aanleiding om verder te zoeken naar de verwachtingen voor de komende jaren.

Top-3 Smartphone Operating System Market Share Ranking Forecast (Share of Shipments)

2015 Rank	Operating System	2011	2012	2013	2014	2015
1	Android	47.4%	53.9%	55.9%	57.8%	58.1%
2	Windows Phone	1.9%	9.0%	15.3%	16.1%	16.7%
3	iOS	18.0%	18.0%	17.3%	16.8%	16.6%
	Others	32.7%	19.1%	11.5%	9.3%	8.6%
	Total	100.0%	100.0%	100.0%	100.0%	100.0%

Source: IHS iSuppli January 2012

Figuur 4.1 Voorspelling marktaandeel Mobiele Platformen t/m 2015

In Figuur 4.1 zijn de voorspellingen van IHS iSuppli te zien. Deze voorspellingen laten zien dat Windows Phone zeer waarschijnlijk een grotere rol gaat spelen in de toekomst. Het is daarom ook van belang om daar tijdens dit onderzoek al rekening mee te houden.

Het antwoord op deze vraag leverde de volgende criteria op voor mijn onderzoek:

- De ontwikkelmethode moet iOS en Android ondersteunen.
- De ontwikkelmethode moet iOS en Android ondersteunen, of van plan zijn dit in de toekomst te gaan doen.

Vanwege de verwachtte daling van het marktaandeel heb ik BlackBerry niet opgenomen als eis aan de ontwikkelmethode die door het onderzoek bepaald gaat worden. Wel kan de mogelijkheid tot het ontwikkelen van BlackBerry applicaties meewegen in de uiteindelijke beslissing.

Vraag 2: Welke methoden bestaan er om platformonafhankelijke applicaties te ontwikkelen?

Tijdens mijn oriëntatie kwam ik aantal categorieën waarin platformonafhankelijke applicaties kunnen worden opgedeeld tegen. De categorieën waren opgedeeld per methode waar op de applicatie tot stand kwam. Tijdens mijn oriëntatie kwamen een aantal categorieën naar voren [2]:

- Mobiele web applicatie,
- Hybride applicatie,
- Geïnterpreteerde applicatie,
- Gegenereerde applicatie.

Tijdens het vooronderzoek heb ik voor elk van deze methode vastgesteld wat de methode exact inhoudt. Hierbij heb ik voor elk van de methodes de voor –en nadelen bepaald. De resultaten hiervan zijn samengevat in Tabel 4.1.

Het antwoord op deze vraag gaf inzicht in de methodes waarop platformonafhankelijke applicaties tot stand kunnen komen. Op basis van het resultaat konden vervolgens de tools uit vraag 3 ingedeeld worden in categorieën zodat deze directer met vergeleken konden worden. Daarnaast konden de voor –en nadelen gebruikt worden om later, in het onderzoek, beslissingen te kunnen nemen over de geschiktheid van de tools die binnen de methode vallen.

Methode	Voordelen	Nadelen
Mobiele web applicatie	• Werken met bekende web standaarden. • Updates hoeven niet gedistribueerd te worden.	• Geen native gebruikersinterface. • Wordt niet opgenomen in de app stores. • Geen lokale dataopslag. • Geen gebruik van de functionaliteiten van de smartphone. • Gecompliceerde animaties trager dan native applicaties.
Hybride applicatie	• Werken met bekende web standaarden. • Distributie via app stores. • Gebruik van smartphone functionaliteiten.	• Geen native gebruikersinterface. • Animaties trager dan native animaties.
Geïnterpreteerde applicatie	• Native gebruikersinterface. • Distributie via app stores. • Native animaties.	• Afhankelijk van API.
Gegenereerde applicatie	• Native gebruikersinterface. • Applicatie specifieke code. • Complexe applicaties mogelijk.	• Gedeeltelijk afhankelijk van API. • Mogelijk een nieuwe programmeertaal leren. • De tools kosten veel geld.

Tabel 4.1 Overzicht voor -en nadelen van methoden voor platformonafhankelijke ontwikkeling

Vraag 3: Welke tools & frameworks zijn beschikbaar?

Voorafgaand aan de afstudeeropdracht was het voor zowel mij als de opdrachtgever nog niet duidelijk of er al tools bestonden om platformonafhankelijke applicaties te ontwikkelen. Mochten deze nog niet bestaan moest gekeken worden naar een eigen oplossing. Al tijdens de oriëntatie bleek dat deze tools ruimschoots beschikbaar zijn.

Met behulp van deze vraag wilde ik inzicht krijgen in welke tools er beschikbaar zijn. Tijdens het vooronderzoek wilde ik nog niet te diep in gaan op individuele tools. Daarom heb ik voor het vooronderzoek slechts bepaald in welke categorie (vraag 2, pagina 12) de tools ingedeeld kunnen worden. Daarnaast heb ik ook voor elke tool ook vastgesteld met welke programmeertalen gewerkt kan worden en voor welke platformen de tool applicaties kan ontwikkelen.

Vraag 4: Welke eisen stellen de applicatiewinkels van de platformen aan applicaties?

Doordat ik eerder zelf applicaties heb uitgebracht via de AppStore van Apple ben ik eerder in aanraking gekomen met de eisen die Apple aan applicaties stelt. Daarnaast heb ik tijdens een introductie dag in Windows Phone van InforSupport geleerd dat ook Microsoft dergelijke eisen aan applicaties stelt. Om te bepalen of een tool geschikt is, is een belangrijk criteria dat de applicatie die met de tool ontwikkeld wordt ook door het review proces van de applicatiewinkel komt.

Bij het beantwoorden van deze vraag heb ik de eisen van de applicatiewinkels op een rijtje gezet. Ik heb hierbij de nadruk gelegd aan de eisen op gebied van User Interface omdat hierin grote verschillen zitten tussen de verschillende platformen. Zo werken Android en iOS met een tab bar in de navigatiestructuur, maar staat deze bij iOS onderin het scherm [3] terwijl deze bij Android juist weer bovenin staat [4]. Windows Phone gebruikt daarentegen geen tab bar maar een ‘pivot menu’ [5]. Daarom is het belangrijk om te weten in hoeverre deze design keuzes als richtlijn gelden of als eis worden gesteld.

Zoals verwacht hadden Microsoft en Apple uitgebreide voorwaarden voor applicaties die aan hun applicatiewinkel toegevoegd worden. In beide gevallen was hierin ook een sectie gericht op de User Interface. Apple stelt daarin een aantal eisen waaraan de User Interface moet voldoen, maar is vaag in de definitie van een slechte user interface. Zo lijken ze een slag om de arm te houden om in principe elke applicatie te weigeren door voorwaarden op te nemen als “If your user interface is complex or less than very good, it may be rejected” [6]. Vooral “less than very good” is zeer subjectief.

Microsoft heeft meer in detail uitgestippeld aan welke eisen de applicatie moet voldoen. Maar zowel in het geval van Microsoft als dat van Apple is moeilijk vast te stellen in hoeverre applicaties geweigerd worden wanneer van deze voorwaarden afgeweken wordt. BlackBerry geeft alleen aan dat ze het recht hebben applicaties te controleren en eventueel tegen te houden, terwijl Android elke applicatie toe staat en slechts in gevallen van malware en oplichting het recht behoudt om applicaties achteraf te verwijderen uit hun applicatiewinkel.

4.2 Onderzoeksplan

Op basis van de resultaten uit het vooronderzoek heb ik het onderzoeksplan opgesteld. Het onderzoeksplan is gebaseerd op de methode van Nel Verhoeven [1]. In de volgende paragrafen wordt de totstandkoming van de verschillende onderdelen van het onderzoeksplan besproken.

4.2.1 Probleemstelling

De NCIM Groep wil in de toekomst ook mobiele applicaties aanbieden aan hun klanten. De Mobile Flight Assistant applicatie is gemaakt om een beeld te krijgen hoe het maken van een mobiele applicatie in zijn werk gaat. Daarnaast is begonnen met de RestaurantApp, deze applicatie stelt de gebruikers in staat om via de applicatie bestellingen te doen in een restaurant wanneer ze uit eten gaan.

Beide applicaties zijn momenteel slechts voor het iOS platform ontwikkeld, waardoor slechts een beperkte doelgroep bereikt kan worden. Vooral in het geval van de RestaurantApp is het bereiken van een grote doelgroep zeer belangrijk. De opdrachtgever verwacht ook dat veel klanten het belangrijk vinden een grote doelgroep te bereiken. De opdrachtgever wil dus weten hoe applicaties voor alle platformen gemaakt kunnen worden zonder deze voor elk platform opnieuw te schrijven.

Op basis van het vooronderzoek bleken meerdere tools te bestaan die mogelijk aan de wens van de opdrachtgever voldoen. Daarom wil de opdrachtgever weten wat de beste tool is om platformonafhankelijke applicaties te ontwikkelen. Zodoende ben ik tot te volgende probleemstelling gekomen:

> *"Wat is de beste manier om platformonafhankelijke mobiele applicaties te ontwikkelen?"*

4.2.2 Doelstelling

Uit gesprekken met de opdrachtgever bleek dat hij niet alleen geïnteresseerd was in een theoretisch antwoord op de probleemstelling. De opdrachtgever gaf ook aan een voorbeeld applicatie zien om te bewijzen dat de methode ook werkelijk geschikt is. De voorbeeld applicatie kan dan tevens gebruikt worden om aan anderen te demonsteren hoe de applicatie gemaakt wordt.

Met behulp van de resultaten van het onderzoek wil de opdrachtgever in de toekomst kunnen beslissen op welke wijze nieuwe opdrachten uitgevoerd worden. Daarnaast is hij ook geïnteresseerd in wat de oplossing voor de huidige projecten kan betekenen. Op basis hiervan heb ik de volgende doelstelling benoemd:

> *"De opdrachtgever wil advies hebben over de beste manier om platformonafhankelijke mobiele applicaties te maken. Naast dit advies wil de opdrachtgever ook, doormiddel van een Proof of Concept, zien hoe de applicaties gemaakt moeten worden."*

4.2.3 Onderzoeksontwerp

Voordat ik aan het onderzoek begon had ik een aanpak gepland waarbij eerst op basis van theorie onderzocht wordt welke methode het meest geschikt is voor het ontwikkelen van platformonafhankelijke applicaties, om dit vervolgens te bekrachtigen met een Proof of Concept. Op basis van het vooronderzoek heb ik echter gekozen voor een andere aanpak.

Uit het vooronderzoek kwam naar voren dat er verschillende methodes zijn om platformonafhankelijke applicaties te ontwikkelen. Binnen een methode zijn tools op basis van specificaties één op één te vergelijken. Tools vanuit verschillende methodes kunnen echter niet één op één vergeleken worden omdat ze een verschillende aanpak hebben. Vooral het uiteindelijke resultaat verschilt.

Daarom heb ik gekozen om voor elk van de methodes de meest geschikte tool te selecteren, om met behulp van de gekozen tool vervolgens een Proof of Concept te bouwen. Zo ontstaat voor elk van de een aparte Proof of Concept applicatie. Op deze manier kunnen de tools ook vergeleken worden op de leercurve en het uiteindelijke resultaat.

4.2.4 Deelvragen

Omdat ik gekozen heb om het Proof of Concept met meerdere tools te bouwen, moest ik eerst bepalen welke tools ik ging gebruiken. De specificaties die ik in het vooronderzoek heb opgesomd zijn echter niet voldoende om een afweging te kunnen maken. Daarom moet dieper om de tools in worden gegaan, dit gebeurd in de volgende deelvraag:

"Met welke tools/frameworks wordt geëxperimenteerd?"

Het antwoord op deze deelvraag geeft mij inzicht in welke tools het beste zijn per categorie. De tools die hieruit naar voren komen gebruik ik vervolgens om het Proof of Concept mee te bouwen.

In het Proof of Concept moeten een aantal functionaliteiten die vaak in mobiele applicaties voorkomen. Aan de hand van deze functionaliteiten kan dan beoordeeld worden in welke mate de tool geschikt is om gebruikt te worden voor het ontwikkelen van applicaties. De functionaliteiten worden vastgesteld middels het beantwoorden van de deelvraag:

"Welke functionaliteiten moet het Proof of Concept bevatten?"

Het antwoord op deze deelvraag levert de requirements op voor het Proof of Concept. Doordat meerdere tools gebruikt worden om het Proof of Concept te bouwen heb ik geen gedetailleerd ontwerp gemaakt, maar ben ik puur uit gegaan van de requirements die bij het beantwoorden van deze deelvraag naar voren kwamen.

Tijdens het experiment met het gekozen web applicatie framework merkte ik dat transities tussen schermen erg traag en schokkerig waren. Ditzelfde gold voor het scrollen in een lijst. Persoonlijk zou dit voor mij een reden zijn om een applicatie niet te gebruiken, echter wilde ik ook vaststellen hoe dit in het algemeen geldt. Zodoende kwam ik tot de extra deelvraag:

"Hoe belangrijk is gebruikerservaring?"

De uitkomst van deze deelvraag heb ik gebruikt bij het vormen van de uiteindelijke conclusie. Tegelijkertijd met deze deelvraag ontstond ook nog een vierde deelvraag. Doordat het framework beter werkte op nieuwere telefoons was het duidelijk dat de hardware op dit moment een grote rol speelde in de slechte prestatie van het framework. Dit leverde de volgende deelvraag op:

"Welke toekomstige ontwikkelingen worden verwacht?"

Deze deelvraag richt zich naast de ontwikkelingen in de hardware vooral op de ontwikkelingen rond HTML5 en Javascript. Het antwoord op deze deelvraag wordt gebruikt bij het vormen van de uiteindelijke conclusie.

Uiteindelijk zijn dus de volgende deelvragen gesteld:

1. Met welke tools/frameworks wordt geëxperimenteerd?
2. Welke functionaliteiten moet het Proof of Concept bevatten?
3. Hoe belangrijk is gebruikerservaring?
4. Welke toekomstige ontwikkelingen worden verwacht?

4.2.5 Tijdspad

Als onderdeel van het onderzoeksplan is ook een globale planning, het tijdspad, gemaakt. In Figuur 4.2 is het tijdspad te zien zoals dit in eerste instantie gepland was. Na het beantwoorden van deelvraag 1 is dit uitgebreid door "Proofs of Concept bouwen" te vervangen door de gekozen tools. Na het bouwen is de planning nogmaals bijgewerkt om de werkelijke ontwikkeltijd weer te geven, de laatste versie van het tijdspad is te zien in Figuur 4.3.

	Week 1	Week 2	Week 3	Week 4	Week 5
Tools/Frameworks selecteren					
Proof of Concept ontwerpen					
Proofs of Concept bouwen					
Onderzoeksrapport opstellen					

Figuur 4.2 Tijdspad onderzoek

	Week 1	Week 2	Week 3	Week 4	Week 5
Tools/Frameworks selecteren					
Proof of Concept ontwerpen					
PhoneGap + jQuery Mobile PoC					
Appcelerator Titanium PoC					
MoSync PoC					
Onderzoeksrapport opstellen					

Figuur 4.3 Bijgewerkt tijdspad onderzoek

4.3 Uitvoer van het onderzoek

In dit hoofdstuk wordt de uitvoer van het onderzoek naar platformonafhankelijke applicaties besproken. Tijdens de uitvoer van het onderzoek heb ik eerst de gestelde deelvragen beantwoord en daarna een Proof of Concept met elk van de gekozen tools gebouwd.

4.3.1 Hypothese

Nog voordat ik aan het beantwoorden van de deelvragen ben begonnen heb ik een hypothese opgesteld.

> *"De verwachting is dat gegenereerde applicaties het beste resultaat op zullen leveren. Doordat deze applicaties code werkelijk omzetten naar native code kan hierbij waarschijnlijk het beste resultaat gehaald worden op het gebied van performance en de native look & feel."*

Zoals al aangegeven in de hypothese wordt de code van gegenereerde applicaties daadwerkelijk omgezet naar native code, daardoor komt het resultaat het dichts bij een native applicatie. Ook de code van geïnterpreteerde applicaties wordt omgezet naar native code. Echter de geïnterpreteerde applicaties zijn sterker afhankelijk van API dan gegenereerde applicaties. Mobiele web –en hybride applicaties worden niet omgezet naar native code, daardoor zal de performance lager zijn dan bij gegenereerde en geïnterpreteerde applicaties.

4.3.2 Deelvragen

De eerste stap in de uitvoer van het onderzoek was, na het stellen van de hypothese, het beantwoorden van de in paragraaf 4.2.4 gestelde deelvragen. Hieronder worden de antwoorden en de totstandkoming van de antwoorden per deelvraag beschreven.

Deelvraag 1: Welke tools & frameworks worden onderzocht?

Tijdens deze deelvraag heb ik op basis van een aantal criteria bepaald welke tools en frameworks ik gebruikt heb tijdens het onderzoek, hierbij heb ik naar de volgende criteria gekeken:

- Welke applicaties zijn met deze tool gemaakt?
  Door eerder uitgebrachte applicaties te downloaden en uit te proberen kon ik een beeld vormen van wat mogelijk was met de tools. De applicaties heb ik geselecteerd vanuit de showcases van de websites van de tools, zodat deze representatief zijn voor de tool.
- Heeft de tool een actieve community?
  Wanneer de tool een actieve community heeft betekent dit dat meer mensen deze tool gebruiken. De community kan dan weer geraadpleegd worden bij eventuele problemen.
- Voor welke type applicaties is de tool bedoeld?
  Sommige van de tools zijn gericht op het maken van games, andere zijn juist niet gericht op games. Omdat de NCIM Groep zich niet richt op het maken van games worden de tools die dat wel doen niet meegenomen in dit onderzoek.
- Kan met de tool een applicatie met een native UI gemaakt worden?
  In het vooronderzoek kwamen een aantal eisen naar voren die de applicatiewinkels stellen aan de User Interface van de applicaties. Wanneer applicaties geen native user interface hebben kan dit betekenen dat de applicaties niet opgenomen worden in de applicatiewinkels die de eisen stellen. Het is daarom belangrijk dat de tool dus zoveel mogelijk aan de eisen voldoet.

Voor elk van de categorieën heb ik de tools vergeleken op basis van de hierboven gestelde criteria. De tools die het beste aan de criteria voldeden binnen hun categorie waren:

- jQuery Mobile (Mobiele web applicatie),
- Phonegap (Hybride applicatie),
- Appcelerator (Geïnterpreteerde applicatie),
- MoSync (Gegenereerde applicatie).

Deelvraag 2: Welke functionaliteiten moet het Proof of Concept bevatten?

Om te bepalen welke functionaliteiten het Proof of Concept moest bevatten heb ik eerst gekeken naar mijn eigen ervaring met het programmeren van mobiele applicaties. Tijdens mijn werk als programmeur bij mijn vorige werkgever bevatten vrijwel alle applicaties de volgende functionaliteiten:

- (Google) Map met een aangepaste *overlay,*
- Tabbladen,
- Menu navigatie,
- GPS,
- Transitie animaties,
- *TableView,*
- *ScrollView,*
- Pop up dialoog,
- Invoervelden / formulieren,
- Lokale database,
- Communicatie met een externe server (PHP),
- Push notificaties.

Deze functionaliteiten heb ik overlegd met Gijs Pannenbakker en Arjan Kodde. Zij waren op het moment dat ik bij de NCIM Groep kwam ook bezig met het maken van een mobiele applicatie.

Omdat ik wilde garanderen dat het framework met de sensoren van de telefoon kan praten heb ik ook bewegingsdetectie als eis toegevoegd. Dit is een sensor die alle smartphones bezitten en daardoor is deze functionaliteit goed te testen.

Om de applicatie te ondersteunen heb ik een database gemaakte bestaande uit twee tabellen. Ik heb deze gemaakt in MySQL, deze werd vervolgens benaderd door de applicatie via PHP. De database bestond uit twee tabellen:

- Mailinglist:
  Deze tabel bevat slechts één kolom waarin de e-mailadressen worden opgeslagen.
- Notifications:
  Deze tabel bevat twee kolommen. In de ene kolom wordt de device token, ter identificatie van het apparaat, opgeslagen. In de tweede kolom wordt een code voor het besturingssysteem van het apparaat opgeslagen, zodat bepaald kan worden van bij welke push service de device token hoort.

Deelvraag 3: Hoe belangrijk is gebruikerservaring?

Tijdens het vooronderzoek kwam al naar voren dat de applicatie winkels eisen stellen aan de gebruikerservaring die de applicaties leveren. Hoe belangrijk deze gebruikerservaring is en hoe dit belangrijk is voor mobiele applicaties komt uitgebreid naar voren in het artikel “Thoughts on Mobile UI Design” [7] van Johannes Fahrenkrug.

Gebruikerservaring is bij mobiele applicaties zeer belangrijk. Dit heeft te maken met de verwachting van de gebruiker. De gebruiker verwacht dat nieuwe applicaties er in grote lijnen net zo uitzien als alle andere applicaties. Standaard knoppen, zoals navigatieknoppen, zitten op standaard plekken binnen de applicatie. Wanneer hiervan afgeweken wordt bestaat de kans dat gebruikers de applicatie als slecht ervaren, ongeacht hoe goed de applicatie verder werkt.

Deelvraag 4: Welke toekomstige ontwikkelingen worden verwacht?

Smartphones maken momenteel een razendsnelle ontwikkeling door. De verschillende merken komen constant met nieuwe modellen uit, welke dan weer uitgerust zijn met een snellere processor, meer RAM en betere grafische prestaties.

De grote vraag voor platformonafhankelijke applicaties is echter hoe de softwarematige ontwikkeling zich door zet. Momenteel ondersteunen de mobiele browsers nog lang niet alle HTML5 functionaliteiten. HTML5 is daarnaast nog niet volledig gespecificeerd en zal naar verwachting pas in 2014 ‘klaar’ zijn [8]. Ook zijn de tools om met HTML5 te ontwikkelen nog lang niet zo ver als de tools om native applicaties te maken.

Windows (Phone) 8 gaat zich voornamelijk richten op web applicaties, deze stap zal er mogelijk voor zorgen dat meer grote ontwikkelaars de stap naar HTML5 nemen. Deze overstap zorgt er voor dat nieuwe tools beschikbaar zullen worden waardoor ook ontwikkeling voor andere platformen mogelijk meer deze kant op gaat [9].

Strategy Analytics heeft onderzoek gedaan naar de verwachte verkoop van telefoons die overweg kunnen met HTML5 [10]. Dit rapport is helaas niet vrij inzichtelijk maar de samenvatting geeft aan dat in 2013 naar verwachting een miljard telefoons met HTML5 ondersteuning verkocht gaan worden. Het ziet er dus naar uit dat HTML5 in de toekomst een zeer grote rol kan gaan spelen bij de ontwikkeling van mobiele applicaties.

4.3.3 Experimenten

In dit hoofdstuk worden de uitvoer en de resultaten van de experimenten beschreven. In dit document worden slechts een aantal belangrijke beslissingen en resultaten behandeld, meer over de experimenten is terug te vinden in het onderzoeksrapport.

PhoneGap & jQuery Mobile

Voor het eerste experiment heb ik PhoneGap en jQuery Mobile samengenomen. jQuery Mobile is een javascript framework dat gebruikt kan worden om een webpagina met een mobiele look & feel te maken. PhoneGap is een framework dat een website verpakt in een mobiele applicatie en daarbij, middels een eigen javascript API, het gebruik van native telefoonfunctionaliteiten aanbiedt. Ik heb ze daarom samengenomen in één experiment, waarbij ik een alternatieve versie van de jQuery Mobile applicatie online draaide om te kunnen testen of PhoneGap invloed had op de snelheid van de applicatie. Omdat PhoneGap gebruik maakt van een browser die door het OS geleverd wordt waren de prestaties nagenoeg gelijk aan de browser versie.

Met behulp van PhoneGap konden alle functionaliteiten uit deelvraag 2 geïmplementeerd worden door slechts Javascript, HTML en CSS te gebruiken. Slechts de push berichten konden niet geïmplementeerd worden. Het resultaat zag op er op iOS en op Android hetzelfde uit, zoals te zien in Bijlage B.

Appcelerator

Het tweede experiment heb ik uitgevoerd met Appcelerator. Net als PhoneGap werkt ook Appcelerator met Javascript. Het grootste verschil is dat Appcelerator deze code omzet naar Native code, Java voor Android en Objective-C voor iOS. Daardoor zien de applicaties er op beide platforms anders uit, zoals te zien in Bijlage C.

Omdat de Javascript code omgezet wordt naar native code van het platform was het in sommige gevallen wel nodig stukken code aan te passen afhankelijk van het platform om de verschillen tussen deze platformen op te vangen. Het is daarnaast wel mogelijk om push berichten te implementeren op iOS, echter is dit nog niet mogelijk op Android.

MoSync

MoSync is een zeer uitgebreid pakket met vele mogelijkheden om platform onafhankelijke applicaties te ontwikkelen. Op dit moment zijn de volgende mogelijkheden beschikbaar:

- C Basic Project
- C Newlib Project
- C NativeUI Project
- Empty Project
- C++ Moblet Project
- C++ STL Project
- C++ NativeUI Project
- C++ OpenGL Project
- HTML5/JS/C++ Hybrid Project
- HTML5 JS NativeUI Project
- HTML5/JS WebUI Project

Vanwege de beperkte tijd voor dit onderzoek kon slechts één van deze mogelijkheden gebruikt worden. Omdat een al een web based oplossing gebruikt is, en omdat zoveel mogelijk naar een native UI gestreefd wordt, heb ik gekozen om voor “C++ NativeUI Project” te kiezen.

Het MoSync framework bevat een eigen datatype voor strings. Dit datatype wordt voor veel functies gebruikt als input parameter of als return type, dus om het framework te gebruiken moet ook het

datatype gebruikt worden. Wanneer een enkele string variabele vaker gebruikt wordt geeft dit echter een probleem. Bijvoorbeeld de loop waarin namen aan de namenlijst worden toegevoegd.

Deze loop gaf meerdere problemen. Het eerst probleem is een vreemd teken dat over de spatie tussen de voor en achternaam wordt gezet, zoals te zien in Figuur 4.4. In eerste instantie leek een verborgen karakter achter de voornaam te staan. Toen echter de lengte van de voornaam aan de totale string werd toegevoegd (Figuur 4.5) verdween het vreemde teken. Wanneer dit aantal weer werd verwijderd kwam het teken weer terug.

```
List<Person *> people = db->getAllPeople();
List<Person*>::ListIterator it = people.begin();
Person *currentPerson;
ListViewItem *item;
String text;
String lastName;
while(it.hasNext()){
    currentPerson = it.next();
    item = new ListViewItem();
    text = currentPerson->getFirstName();
    lastName = currentPerson->getLastName();
    text.append(" ",1);
    text.append(lastName.c_str(), lastName.length());
    item->setText(text);
    lijst->addChild(item);
}
```

Figuur 4.4 MoSync vreemd teken bij strings in een loop code & resultaat

```
List<Person *> people = db->getAllPeople();
List<Person*>::ListIterator it = people.begin();
Person *currentPerson;
ListViewItem *item;
String text;
String lastName;
while(it.hasNext()){

    currentPerson = it.next();
    item = new ListViewItem();
    text = currentPerson->getFirstName();
    lastName = currentPerson->getLastName();
    text.append(" ",1);
    text.append(lastName.c_str(), lastName.length());

    text.append(
        integerToString(
            currentPerson->getFirstName().length()
        ).c_str(),
        integerToString(
            currentPerson->getFirstName().length()
        ).length()
    );

    item->setText(text);
    lijst->addChild(item);
}
```

Figuur 4.5 MoSync strings in loop code & resultaat met voornaam lengte

Naast de vreemde tekens is in Figuur 4.4 en Figuur 4.5 nog een fout te zien, de laatste letters van de eerste achternaam staan ook bij de rest van de achternamen. Het ziet er naar uit dat dit ontstaat doordat bij het ophalen van de naam de string eerst leeg wordt gemaakt, maar gewoon wordt overschreven. Wanneer een nieuwe string dus korter is dan de oude blijft het restant van de oude string gewoon staan. De oplossing hiervoor leek simpel, de string leeg maken voordat deze opnieuw een waarde wordt toegekend. De code hiervan is te zien in Figuur 4.6. De clear functie van het string datatype veranderd echter niets aan de uitvoer.

```
List<Person *> people = db->getAllPeople();
List<Person*>::ListIterator it = people.begin();
Person *currentPerson;
ListViewItem *item;
String text;
String lastName;
while(it.hasNext()){
    lastName.clear();
    text.clear();
    currentPerson = it.next();
    item = new ListViewItem();
    text = currentPerson->getFirstName();
    lastName = currentPerson->getLastName();
    text.append(" ",1);
    text.append(lastName.c_str(), lastName.length());
    item->setText(text);
    lijst->addChild(item);
}
```

Figuur 4.6 MoSync reset string waarde in loop code

Op basis van de eerder genoemde problemen in combinatie met de gebrekkige documentatie is besloten het experiment met MoSync te stoppen.

4.4 Conclusie en aanbevelingen

Op basis van de beantwoorde deelvragen en de experimenten heb ik een conclusie getrokken. Daarnaast heb ik een aantal aanbevelingen gedaan.

4.4.1 Conclusie

Op basis van de experimenten kon ik concluderen dat de tools op verschillende situaties en behoeften gericht zijn. PhoneGap in combinatie met een framework als jQuery Mobile wordt vooral gebruikt om op een gemakkelijke manier veel mensen te bereiken. Een PhoneGap applicatie is echter niet meer dan een website verpakt in een applicatie en dat merkt de gebruiker ook.

Appcelerator is de beste tool om een applicatie te maken wanneer deze alleen op iOS en Android gericht moet zijn. De applicatie krijgt een native look & feel en werkt net zo vloeiend als een echte native applicatie. Het grote nadeel van Appcelerator is echter de afhankelijkheid van het Javascript framework. Wanneer telefoons nieuwe functionaliteiten aanbieden kunnen deze pas gebruikt worden als Appcelerator deze overneemt. Zo blijven Appcelerator applicaties constant achter lopen.

Mijn conclusie op basis van dit onderzoek is dus ook dat het op dit moment het beste is om de applicaties in native code te schrijven. Wanneer een applicatie voor Android en iOS moet verschijnen is het wel de moeite waard eerst te kijken of de beperkingen van Appcelerator een belemmering zijn, zo niet dan is dit een goed alternatief voor native code.

4.4.2 Aanbevelingen

Op basis van het onderzoek heb ik een aantal aanbevelingen gedaan. Ik heb een algemene aanbeveling gedaan en ook specifieke aanbevelingen voor de applicaties die momenteel ontwikkeld worden.

Algemeen

De conclusie van het onderzoek was dat applicaties momenteel nog niet platform onafhankelijk ontwikkeld moeten worden. Echter ontwikkelen de tools en frameworks hiervoor zich razendsnel, net zoals de telefoons zelf. Tijdens het uitvoeren van het onderzoek kwamen PhoneGap, MoSync, Sencha Touch en jQuery Mobile met grote updates. Daarnaast kondigde Appcelerator en Marmalade een grote nieuwe release aan en is Codename One naar een open beta fase overgegaan.

Bij het beantwoorden van deelvraag 4 kwam naar voren dat de HTML5 ondersteuning steeds beter gaat worden. Daarnaast wordt ook de hardware meer geschikt om hiermee om te gaan. Ook worden nog steeds nieuwe functionaliteiten aan HTML5 toegevoegd, om bijvoorbeeld sensoren van de telefoon aan te spreken. Dit kan in de nabije toekomst dus een nieuw onderzoek waard zijn.

Huidige applicaties

Op het moment van het onderzoek was de NCIM Groep bezig met twee mobiele applicaties. Dit waren MOFLAS en de Restaurant App. Voor beide heb ik een aparte aanbeveling gedaan, omdat de applicaties erg verschillend zijn.

MOFLAS

Op basis van het onderzoek is het voorlopig aan te bevelen om MOFLAS in native-code te blijven schrijven. Naast het gebruik van native gebruikersinterface elementen worden ook zware bewerkingen uitgevoerd. Met een 3D functionaliteit in de planning is het simpelweg niet mogelijk om een web applicatie te maken.

Ook is de MOFLAS applicatie in eerste instantie gericht om te laten zien wat mogelijk is met een mobiele applicatie. Door naar een platformonafhankelijke oplossing over te stappen wordt ten aller tijden ingeleverd op native functionaliteiten. Om het maximale uit de applicatie te halen is het advies dus native code te schrijven.

Restaurant App

Op basis van het onderzoek zijn goede mogelijkheden te zien om de restaurant app platformonafhankelijk te ontwikkelen. Momenteel is de applicatie geschreven in native Objective-C code. Daarnaast is een variant geschreven met behulp van jQuery Mobile, deze is echter nog niet volledig up-to-date met de native applicatie.

Het doel van deze applicatie is het bestellen in restaurants te vergemakkelijken. Om hiermee zoveel mogelijk gebruikers aan te spreken is het belangrijk deze applicatie op meerdere platformen uit te brengen. Doordat de applicatie al gedeeltelijk in jQuery Mobile gemaakt is kost het relatief weinig moeite om deze variant uit te breiden en om te zetten naar een native applicatie met behulp van PhoneGap.

Voor iOS kan dan de native applicatie gebruikt blijven worden terwijl alle andere platformen de web variant draaien. Daarnaast maakt de jQuery Mobile versie het ook mogelijk om een online versie te draaien, waardoor gebruikers slechts een browsers nodig hebben om te bestellen. De online versie kan dan onder andere als back-up dienen in het geval dat Microsoft de PhoneGap variant weigert op te nemen in de Windos Phone Marketplace.

4.5 Gebruik nieuwe kennis

Al vrij snel na het de uitvoering van het onderzoek startte een nieuw project waarin een mobiele applicatie werd gemaakt; SocialSOS. Voor deze applicatie is het zeer belangrijk dat het multi platform werkt en daarbij snel en native aan voelt. Aan de hand van mijn onderzoek is toen gekozen de ontwikkeling met Appcelerator te doen. De klant had echter de voorkeur om, om demonstratie redenen, eerst een iOS specifieke applicatie te maken. Daarom is de projectgroep overgestapt op de drag & drop editor van Apple.

Op 10 mei 2012 werd de applicatie gedemonstreerd op een beurs met als doel geldschieters voor het project te vinden, gezien de klant slechts bedenker is en dus niets financiert. Daarop werd van bovenaf besloten dat de applicatie wel degelijk multi platform moest zijn, en tevens bij voorbijgangers geïnstalleerd moest kunnen worden. Daarop heeft de projectgroep wederom naar mijn onderzoek gegrepen. De applicatie is nu, voor demonstratie doeleinden, gemaakt met jQuery Mobile.

5 Uitbreiden van de MOFLAS applicatie

Het tweede onderdeel van de opdracht was het uitbreiden van de Mobile Flight Assistant (MOFLAS) applicatie. Op basis van de bevindingen tijdens het onderzoek uit hoofdstuk 4 moest dit eventueel platformonafhankelijk gedaan worden. Tijdens het onderzoek heb ik echter geconcludeerd dat de tools nog niet geschikt waren om deze applicatie platformonafhankelijk te kunnen maken.

5.1 Huidige status

Aan de huidige applicatie is duidelijk te zien dat dit de eerste applicatie van de NCIM Groep is geweest. De applicatie is gebaseerd op een (zeer) verouderde SDK. Een aantal functionaliteiten zijn dusdanig verander dat deze code opnieuw geschreven moet worden, zoals het Geocoden (het omzetten van een adres naar latitude en longitude). Bij het builden van het project worden direct meer dan 120 warnings gegeven. De applicatie crasht dan ook direct bij het opstarten. Ook ontbreekt een sqlite database in de SVN repository. De applicatie is, zoals door Apple ook wordt voorgeschreven, wel volgens een MVC pattern opgebouwd. De bestanden staan echter door elkaar waardoor het project onoverzichtelijk wordt.

Om bovenstaande redenen heb ik aangeraden dat de hele applicatie opnieuw geschreven wordt. Hiermee is door de NCIM Groep ingestemd, in het geval dat ik tijd over zou houden zou ik dit ook gaan uitvoeren. Tot die tijd heb ik mijn uitbreiding in een apart project gemaakt, waarbij ik wel rekening heb gehouden dat de uitbreiding gemakkelijk geïntegreerd kan worden met de bestaande applicatie zodra die is herschreven. Ik bleek uiteindelijk niet genoeg tijd te hebben om de bestaande applicatie te herschrijven waardoor de uitbreiding nu nog niet geïntegreerd is.

5.2 Ontwikkelmethode

De projecten die intern door de NCIM Groep worden uitgevoerd, maken allemaal gebruik van de Scrum methode. Ook de MOFLAS applicatie was tot nu toe ontwikkeld volgens deze methode. Zodoende werd van mij verlangd ook op deze manier verder te gaan. Om werkelijk volgens Scrum te werken, is echter een team nodig [11]. Normaalgesproken bestaat het Scrum team uit de Product Owner, het team van ontwikkelaars en de Scrum Master. In mijn geval waren er slechts een Product Owner en ikzelf als ontwikkelaar. Om toch volgens de methode te kunnen werken zoals deze gebruikt wordt door de NCIM Groep heb ik een aantal delen van de Scrum methode wel gebruikt, en een aantal delen niet.

5.2.1 Requirements

Dit hield voor mij in dat ik de requirements op heb gesteld als user stories. Normaal gesproken schrijft de opdrachtgever zelf deze user stories, echter voor interne projecten heeft de opdrachtgever daar geen tijd voor. Daarom heb ik deze zelf opgesteld en vervolgens ter goedkeuring naar de opdrachtgever gestuurd.

5.2.2 Ontwikkelen

Het ontwikkelen wordt uitgevoerd in sprints van een week. Tijdens een sprint worden één of meer user stories geïmplementeerd. Welke user stories tijdens de sprint geïmplementeerd worden, wordt bepaald door de opdrachtgever. Bij interne projecten wordt dit echter bepaald door de projectgroep zelf, eventueel op basis van de prioriteiten die vooraf aan de user stories zijn gegeven. Normaal gesproken wordt tijdens de Daily Scrum besproken wat de vorige dag goed en fout ging, daarnaast wordt besproken wat op de huidige dag gedaan gaat worden. Deze Daily Scrum meetings heb ik niet gehouden omdat ik alleen aan het project werkte.

5.2.3 Sprint bord

De gekozen user stories worden vervolgens door de programmeurs opgedeeld in taken, die nodig zijn om tot het gewenste resultaat te komen. Aan deze taken wordt gewicht gegeven door middel van punten. Punten worden bepaald door taken te vergelijken met eerder uitgevoerde taken. De taken met bijbehorende punten worden vervolgens opgenomen in een sprint backlog. Daarnaast worden de taken op opgehangen op een Scrum bord met behulp van post-its. De Scrum borden in de kamer waar ik werkte waren echter al in gebruik door andere teams. Ik heb daarom een online Scrum bord gemaakt bij Trello.

5.2.4 Evaluatie

Dagelijks worden de afgeronde taken als afgerond gemarkeerd in de sprint backlog. Aan het eind van de sprint is dan een burndown chart te zien, een grafiek waarvan het verloop van de sprint afgelezen kan worden. Na afloop van de sprint wordt deze geëvalueerd en worden de bevindingen van de evaluatie samen met de burndown chart opgenomen in een sprint progress report.

Op vrijdagen werden Sprint Reviews gehouden. Normaalgesproken zou dit met het Scrum team zelf zijn, maar bij de NCIM Groep worden de Sprint Reviews gezamenlijk gehouden door alle teams. Tijdens deze Sprint Reviews vertelt elk team hoe de afgelopen week was verlopen. Vervolgens kunnen de andere teams hier dan feedback en tips op geven. Op deze manier leren teams niet alleen van hun eigen fouten, maar leren ze ook van de foutendie door de andere teams gemaakt zijn.

5.3 Requirements

Voordat ik kon beginnen aan de uitbreidingen moest eerst de opdracht nog verder verduidelijkt worden. Ik heb daarom een afspraak gemaakt met de opdrachtgever. Hij bleek echter net op reis te zijn waardoor de afspraak op zich liet wachten. Ik heb een aantal andere betrokkenen aangesproken, maar geen van hen kon de uitbreidingen verder specificeren.

In afwachting van de afspraak ben ik zelf al op onderzoek uitgegaan naar de mogelijkheden om real-time weergegevens op te vangen. In de opdracht werd gesproken over een ADS-B ontvanger. De ADS-B ontvanger vervangt de traditionele, op radar gebaseerde, systemen. Het kan gezien worden als een groot draadloos netwerk, waarbij grond stations als access points dienen en de vliegtuigen als clients. Weergegevens worden door de grond stations gestuurd door middel van een zogenaamd FIS-B signaal. [12] Dit signaal wordt echter niet ontvangen door de ADS-B ontvanger waar de NCIM-Groep over bezit.

Tijdens het gesprek kwam naar voren dat het inderdaad niet mogelijk is om real-time weergegevens te ontvangen via onze ADS-B ontvanger. Het blijkt dan ook dat deze gegevens momenteel alleen in de Verenigde Staten worden uitgezonden, in Europa zijn dus geen real-time weergegevens beschikbaar. Tijdens het gesprek zijn we wel tot een tijdelijke oplossing gekomen, het KNMI heeft een feed waarin de laatste weergegevens gezet worden. Voorlopig kan een van deze berichten hard-coded in de applicatie worden gezet om zo weergegevens te kunnen gebruiken. Later, buiten de scope van mijn opdracht, kan dan een server worden gemaakt die de gegevens van het KNMI ophaalt en vervolgens via satelliettelefoon verstuurt naar het vliegtuig om ze zo real-time bij de applicatie te krijgen.

Tijdens het gesprek kwam tevens naar voren dat de applicatie twee doelen dient, namelijk het assisteren van piloten tijdens hun vlucht en het binnen halen van nieuwe klanten middels demonstraties op beurzen. Dit leverde twee type gebruikers op voor de applicatie. Omdat de applicatie gebruik maakt van de werkelijke GPS gegevens die de iPad binnenkrijgt heb ik gesuggereerd de gebruiker de mogelijkheid te geven deze te overschrijven met zelf ingevoerde waarden. Hiermee ging de opdrachtgever akkoord. Ook de sales representatives van de NCIM Groep waren enthousiast over die suggestie.

Binnen de NCIM Groep worden alle projecten met de Scrum methodiek uitgevoerd. Requirements worden daarbij opgesteld middels user stories. Deze user stories worden normaal gesproken opgesteld door de opdrachtgever, echter heb ik deze in dit geval zelf opgesteld omdat de opdrachtgever daar geen tijd voor zag. De user stories heb ik vervolgens ter goedkeuring naar de opdrachtgever gestuurd. De goedgekeurde user stories zijn opgenomen in het product backlog in Bijlage D. In het product backlog zijn ook prioriteiten gegeven aan elk van de user stories. Naast de opdrachtgever heb ik deze ook naar de sales afdeling van de NCIM Groep gestuurd om te controleren of ook al hun wensen opgenomen waren in de applicatie.

5.4 Planning

Omdat de NCIM Groep met Scrum werkt heb ik elke week een sprint planning gemaakt naast de globale planning uit het Plan van Aanapak. Doordat het onderzoek op schema verlopen is, is de globale planning uit het Plan van Aanpak niet veranderd.

Nadat ik besloten had welke 3D engine ik zou gaan gebruiken ben ik begonnen met het doorlopen van sprints. Omdat ik slechts alleen in een project zit heb ik ook zaken omtrent het afstuderen opgenomen in de sprint planning, zoals het schrijven van dit verslag. Op deze manier bepaalde ik vooraf wat ik elke week ging schrijven zonder daar apart tijd voor vrij te hoeven maken. Omdat de Scrum borden op mijn kamer al in gebruik waren heb ik mijn Scrum bord online op Trello aangemaakt, een voorbeeld daarvan is bijgevoegd als Bijlage E. Aan het einde van elke dag voerde ik in het sprint backlog in welke taken afgerond waren. Aan het einde van de week ontstond dan een burndown chart, een voorbeeld hiervan is te zien in Bijlage F.

5.4.1 3D Engine Selectie

Ik was van plan het 3D deel van de applicatie zelf met OpenGL, ondersteund door het GLKit framework van iOS, te programmeren. Echter na een paar tutorials gevolgd te hebben bleek OpenGL meer tijd in te nemen dan ik vooraf had gedacht. Omdat hier niet genoeg tijd voor beschikbaar was heb ik besloten een 3D engine te zoeken om het teken werk voor mij te doen, dit wordt behandeld in paragraaf 5.5.

Het selecteren van een 3D engine was vooraf niet in de planning opgenomen en moest dus tussendoor gedaan worden. Tijdens het bespreken van de requirements kwam naar voren dat in Europa geen real-time weergegevens ontvangen kunnen worden.

5.4.2 Kaart tegels

Bij het inladen van de kaart in de 3D wereld liep ik tegen het probleem op dat de kaart tegels dezelfde texture kregen, de details hiervan worden beschreven in paragraaf 5.10.3. Het oplossen van dit probleem heeft in totaal vijf weken geduurd. Na de eerste week vast gezeten te hebben, heb ik gekozen met niet volledig op dit probleem te blijven richten. In de tweede en derde week van het probleem heb ik ongeveer 16 uur ingepland om aan het probleem te werken. In de vierde en vijfde week heb ik dit teruggeschroefd naar 8 uur.

Tijdens het maken van een sprint planning worden punten toegekend aan activiteiten, in plaats van dat deze in uren uitgedrukt worden. Het toekennen van uren is dus niet direct mogelijk bij het maken van de sprint planningen. Om dit toch te kunnen verwerken heb ik eerst de sprint planning van de week gemaakt om vervolgens het aantal punten per uur uit te rekenen. Vervolgens heb ik aan het oplossen van het probleem het aantal punten dat overeenkomt met de geplande uren toegekend.

5.4.3 Weergegevens & Flight planning

Doordat ik lang vastgezeten heb op het inladen van een kaart in 3D en het selecteren van de 3D engine heb ik in overleg met de opdrachtgever besloten de implementatie van de real-time weergegevens voorlopig te laten zitten. Door de real-time weergegevens niet te implementeren heb ik me beter kunnen richten op het navigeren tussen waypoints. Daardoor is het nu wel gelukt deze functionaliteit volledig, inclusief waypoint en route database, te implementeren.

5.5 3D Engine Selectie

Voordat ik aan het ontwerp van deze uitbreiding ben begonnen moest ik eerst leren hoe in 3D voor de iPad geprogrammeerd kan worden. In eerste instantie was ik van plan om gebruik te maken van GLKit, een onderdeel van het iOS SDK.

GLKit is een abstractie laag boven OpenGL, die recentelijk aan de iOS SDK is toegevoegd. Hierdoor worden een aantal render functionaliteiten makkelijker, zoals het aanroepen van de render loop. Ongeacht het gebruik van GLKit moeten echter vrijwel alle OpenGL communicatie zelf geprogrammeerd worden.

Om met pure OpenGL code een kubus op het scherm van de iPad te krijgen, en die rond te laten draaien, waren al tientallen regels code nodig. Om op deze manier de flight planning functionaliteit te schrijven was niet realistisch, vooral omdat het teveel tijd zou kosten te leren programmeren in OpenGL. Daarom heb ik gekozen hiervoor een 3D engine te zoeken.

5.5.1 3D engines

Het vinden van een 3D engine was lastiger dan ik vooraf had veracht. De meeste engines waren specifiek op games gericht, en kwamen ook vaak met een externe editor waardoor de code niet geïntegreerd kon worden in een bestaande applicatie. Voorbeelden hiervan zijn de Unreal Development Kit en Unity3D.

Ogre3D

Toen ik op zoek ging naar een 3D engine voor iOS viel mijn oog direct op Ogre3D. Tijdens mijn vorige opleiding heb ik namelijk eerder met deze engine, die geschreven is in c++, gewerkt. Het is voor iOS mogelijk om c++ code te integreren in Objective-C code. Echter bij het het builden van de meegeleverde demo applicaties liep ik direct al tegen poblemen aan. Intern in het framework wordt naar header files verwezen die niet op de juiste locatie staan, en daardoor niet goed gevonden worden. Deze moeten dus handmatig worden aangepast. Na de vele pogingen het werkend te krijgen heb ik besloten eerst verder te kijken voordat ik verder ging proberen de engine aan de praat te krijgen.

Cocos3D

Cocos3D is een veelgenoemde engine wanneer mensen vragen naar een 3D engine voor iOS. Het is een uitbreiding op het zeer populaire Cocos2D framework, wat veel gebruikt wordt voor 2D spelletjes op de iOS apparaten. Cocos3D is echter nog in een vroeg stadium van de bèta fase, bovendien wordt het ontwikkeld door slechts één persoon.

Cocos3D voegt aan Cocos2D de mogelijkheid toe om 3D spelletjes te maken. Daar zit dan ook direct al het eerste probleem, de engine is eigenlijk gericht op het maken van spelletjes. Toch was het wel mogelijk om een 3D weergave te integreren in een bestaande applicatie.

Ik heb daarom een kleine test applicatie gemaakt, waarin ik een vliegtuig inlaadde in een van de schermen binnen een standaard navigatiestructuur. Op de simulator ging dat prima maar op de werkelijke iPad werd het vliegtuig nooit ingeladen. Dit bleek uiteindelijk een timing probleem te zijn. Dit kon worden opgelost door een sleep te plaatsen in een van de functies die tijdens het laden worden aangeroepen. Dit is echter niet wenselijk gezien de verdeling van de processortijd bepaald wordt door het OS, en dus niet gegarandeerd kan worden hoe lang de sleep precies moet zijn. Net als bij Ogre3D ben ik daarom verder gegaan met zoeken naar andere oplossingen, met als optie om hierop terug te komen.

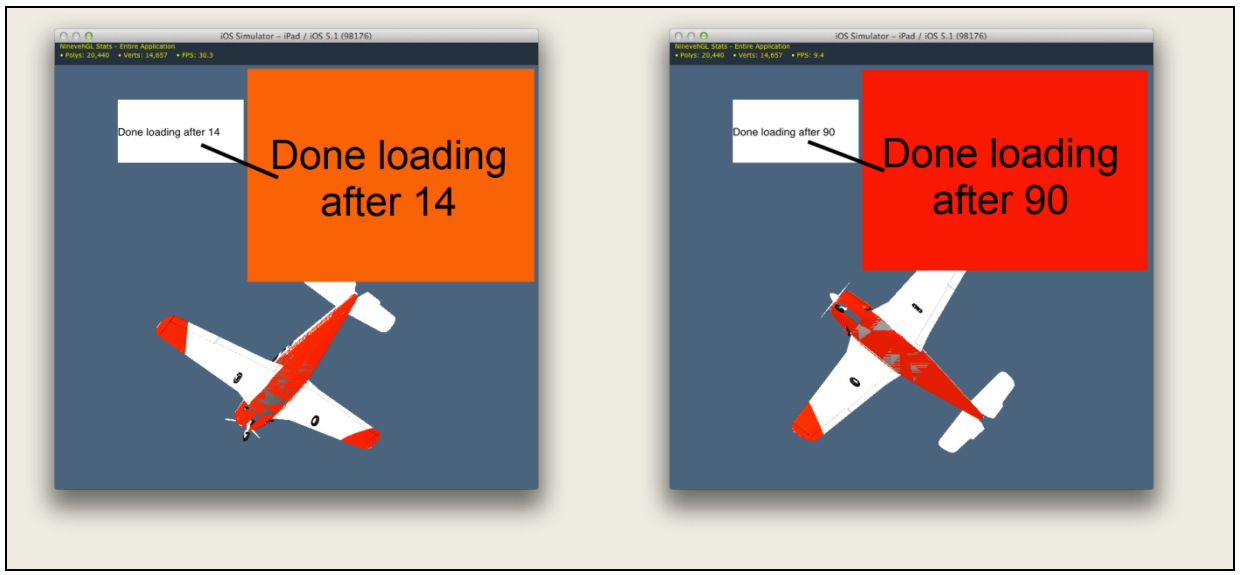
NinevehGL & iSGL3D

Net als Cocos3D is ook het NinevehGL framework nog in bèta fase. Waar Cocos3D nog in een vroeg stadium van de bèta fase (versie 0.6.5) zit, zit NinevehGL tegen het einde (versie 0.9.3) en wordt het tevens ontwikkeld door een team van vrijwilligers in plaats van slechts één vrijwilliger, zoals bij Cocos3D het geval is.

De gouden regel van NinevehGL is *"Keep it simple"* [13]. Dit is terug te zien in de korte tutorial video's, 10 stuks van 3 minuten per stuk. Ook in de opbouw van de voorbeeld applicaties is te zien dat het gemakkelijk is een 3D object in te laden, weer te geven en bijvoorbeeld om een van de assen te laten draaien.

iSGL3D is net als Cocos3D een veel genoemde 3D engine. iSGL is, in tegenstelling tot NinevehGL en Cocos3D, niet meer in de bèta fase. Net als NinevehGL is de iSGL3D engine vrij gemakkelijk in gebruik, ook is de engine erg volledig. De keuze is toch gevallen op NinevehGL, ondanks de bèta fase. Uiteindelijk is de keuze vooral gemaakt op het inladen van objecten. iSGL3D kan alleen .POD bestanden inlezen, terwijl NinevehGL meet .OBJ en .DAE bestanden om kan gaan. Om modellen om te zetten naar .POD bestanden moest een externe tool gebruikt worden, terwijl 3D tools als Autodesk 3ds Max en Blender zelf al .OBJ of .DAE bestanden kunnen exporteren.

Daarnaast is NinevehGL zo gemaakt dat objecten wanneer ze ingeladen worden omgezet worden naar een eigen bestandsformaat. Daardoor wordt de volgende keer dat het object wordt ingeladen het laden ongeveer 960% sneller gedaan dan als het originele bestand steeds wordt ingeladen. Dit geldt ook wanneer de applicatie wordt afgesloten en opnieuw wordt opgestart. In Figuur 5.1 is mijn eigen vergelijking tussen fast loading en original loading te zien. Hierin heb ik gemeten hoe vaak de load functie wordt aangeroepen. In het geval van fast loading was dit 14 keer, in het geval van original loading was dit 90 keer. Dat is dus in mijn geval ongeveer 643% sneller.



Figuur 5.1 NinevehGL fast loading tegen original loading

5.6 Kaart selectie

De iOS SDK bevat een API voor google maps. Hiermee is het gemakkelijk om een kaart weer te geven. Echter is het niet mogelijk om de standaard UIKit views, waar de google map view een subclass van is, in te laden in de 3D omgeving. Daarnaast heeft Google Maps ook een internet verbinding nodig, deze is in een vliegtuig niet aanwezig. Het is tegen de terms of service van Google om de kaart te cachen [14] dus moest ik op zoek naar een andere oplossing. Hierbij ben ik uitgekomen op OpenStreetMap.

Route-me

De OpenStreetMap kaart is opgebouwd uit losse tegels, elk 256 bij 256 pixels. De tegels zijn genummerd in 3 dimensies, namelijk:

- Z: Het zoomniveau van de tegel, dit varieert van 0 tot en met 19, waarbij 0 het verst uitgezoomd is en 19 het verst ingezoomd. Bij zoomniveau 0 valt de gehele wereld in 1 tegel, het aantal tegels loopt vervolgens op per zoom niveau. Zoomniveau 1 bevat 4 tegels, zoomniveau 2 bevat 16 tegels, zoomniveau 3 bevat 256 tegels etc.
- X: De nummering in de horizontale richting, dit begint bij 0 voor de meest linker tegel.
- Y: De nummering in de verticale richting, dit begint bij 0 voor de bovenste tegel.

De tegels kunnen via een web request opgehaald worden. Dit wordt gedaan middels de volgende url:

> "*http://tile.openstreetmap.org/zoom/x/y.png"*

Dus bijvoorbeeld voor het hoofdkantoor van de NCIM Groep op zoom niveau 13:

> *"http://tile.openstreetmap.org/13/4196/2703.png"*

Om te kunnen bepalen welke tegels opgehaald moeten worden, moet vanuit de huidige locatie uitgerekend worden welke tegels die locatie weergeven. Hiervoor moeten latitude en longitude dus omgerekend worden naar x en y coördinaten. Vanwege de bolling van de aarde is dat omrekenen erg lastig. Daarom ben ik op zoek gegaan naar een framework wat deze omrekening al doet en de juiste tegels teruggeeft.

Het route-me framework was in dit geval de meest logische oplossing. Het framework is gratis en open source. Het gebruikt uit zichzelf al OpenStreetMap, met als enige nadeel dat het zelf de kaart tekent in plaats van de tegels terug te geven.

Tile Database

Omdat tijdens de vlucht geen internet verbinding mogelijk is moet de kaart vooraf worden gecached. Het route-me framework biedt geen mogelijkheid om vooraf te cachen. Wel biedt het de mogelijkheid om een andere bron dan OpenStreetMap te gebruiken, bijvoorbeeld een eigen database.

Omdat het niet mogelijk is om kaart tegels van een groot gebied via de website van OpenStreetMap te downloaden heb ik hiervoor een perl script [15] gebruikt. Om deze tegels in een sqlite database te zetten heb ik gebruik gemaakt van de map2sqlite tool [16]. Deze kan vervolgens worden gebruikt door het route-me framework. Omdat de hogere zoomniveaus zeer veel tegels bevatten heb ik gekozen om voor nu alleen de Benelux te downloaden van zoomniveau 6 tot en met zoomniveau 13.

Deze tegels werden als BLOB datatype in de database opgeslagen. Ondanks dat de tabel slechts een paar duizend rijen bevatte werd de database hierdoor wel traag. Objecten worden namelijk opgehaald op basis van een combinatie van de x en y coördinaten en het zoomniveau. Het kostte de applicatie ruim 3 seconde om een tegel op te halen. Daarom heb ik secundaire indexen geplaatst op de x, y en zoom kolommen. Na het plaatsen van de indexen duurde het ongeveer 0,01 seconde om een tegel op te halen.

5.7 Klassendiagram

Om het klassendiagram te maken heb ik gekozen voor de tool UMLet. De keuze voor deze tool, boven andere tools komt voort uit de keuze voor de notatiewijze binnen het klassendiagram. Omdat Objective-C een afwijkende notatie heeft wat betreft methodes moest ik een keuze maken hoe de ik de notatie in het klassendiagram ging doen. Enerzijds kon ik kiezen voor de reguliere notatie (Figuur 5.2), anderzijds voor de Objective-C notatie (Figuur 5.3).

```
int multiplyValueAByValueB(int valueA, int valueB)
```

Figuur 5.2 Reguliere notatie vermenigvuldiging methode

```
-(int)multiplyValueA:(int)valueA withValueB:(int)valueB;
```

Figuur 5.3 Objective-C notatie vermenigvuldiging methode

Het gebruik van de reguliere notatie komt de begrijpbaarheid voor mensen buiten het project ten goede, gezien ze de notatie al kennen. Echter wanneer iemand in het project stapt en het klassendiagram als leidraad wilt gebruiken komen de methodenamen niet overeen. In het geval van het voorbeeld uit Figuur 5.2 en Figuur 5.3 is het verschil in naamgeving nog te overzien, maar hoe langer de naam hoe onoverzichtelijker dit wordt. Na overleg met mijn begeleider, waarin ik beide argumenten voorgelegd heb, vonden we het argument voor de Objective-C notatie zwaarder wegen en ben ik dus op zoek gegaan naar een tool waarin ik deze notatie aan kon houden.

UMLet wordt binnen de NCIM Groep al gebruikt voor het maken van klassendiagrammen. Doordat anderen de tool dus al geïnstalleerd hebben is het voor hun gemakkelijker om een klassendiagram in het formaat van UMLet bij te werken dan dat van een andere tool. Daarnaast was het goed mogelijk om een aangepaste notatie voor methoden te gebruiken, zoals te zien in Bijlage J.

5.8 Use Case diagram

Voor het Use Case diagra heb ik gebruik gemaakt van dezelfde tool, UMLet. In het Use Case diagram heb ik rekening gehouden met twee actoren, de piloot en de sales representative. Zowel de Use Cases al de actoren zijn gebaseerd op het Product Backlog dat is opgesteld op basis van de requirements. Het Use Case diagram is te vinden onder Bijlage I.

5.9 Opbouw van de uitbreiding

De uitbreiding heb ik opgebouwd volgens het Model View Controller (MVC) pattern. Het Cocoa framework, waar de iOS SDK gebruik van maakt, is ook opgebouwd volgens het MVC pattern [17]. Geïmporteerde classes passen dus direct in de MVC structuur van de uitbreiding. Daarnaast is ook de huidige applicatie in MVC opgebouwd. Daardoor kan de uitbreiding, die in een apart project gemaakt is, één op één overgezet worden zonder gevolgen voor de applicatie structuur.

5.9.1 Singleton Klassen

Tijdens het ontwerp van de uitbreiding heb ik voor een aantal klassen gekozen er een Singleton van te maken. De klassen waarvoor ik dit heb gedaan zijn:

- Settings,
- Tile (object) Pool,
- Database.

Een alternatief voor de Singleton klassen zijn static klassen. Deze zijn gemakkelijker te implementeren dan Singleton klassen, echter in geen van deze gevallen mogelijk. Elk van deze klassen hebben namelijk tenminste één object als property. Objecten moeten geïnitialiseerd worden, hetgeen nu gebeurd wanneer het Singleton object voor het eerst wordt aangeroepen. Bij een static klasse zou telkens wanneer het object gebruikt wordt eerst gecontroleerd moeten worden of het object al bestaat. In elke methode de benodigde controles schrijven om te controleren kost dus meer tijd dan een Singleton klasse, waarbij slechts gecontroleerd hoeft te worden of het Singleton object al bestaat.

Settings

De settings klasse bewaard alle instellingen die door de gebruiker zijn ingesteld. Bij het initialiseren van de applicatie worden deze opgehaald uit de zogenaamde user defaults, waarin key-value-pairs opgeslagen kunnen worden.

Een mogelijk alternatief voor de Singleton is een normale klasse maken. De uitvoer hiervan kan weer opgedeeld worden in twee richtingen. Enerzijds kan een object van de settings klasse worden aangemaakt in de constructor van het object dat de instellingen nodig heeft. Dit object kan dan als property van het object worden opgeslagen. Dit geeft echter een probleem dat als twee klassen beide een settings object bevatten, de wijzigingen bij de een niet door worden gevoerd bij de ander. Anderzijds kan ook steeds een nieuw settings object worden aangemaakt wanneer een instelling wordt opgehaald of gewijzigd, om vervolgens weer weggegooid te worden. Dit geeft het probleem dat steeds bij het aanmaken van het settings object de instellingen opgehaald moeten worden uit de user defaults, hetgeen meer tijd kost.

Tile (object) Pool

Om de kaarttegels aan te maken heb ik gekozen voor het gebruik van een Object Pool, hierover wordt meer uitgelegd in paragraaf 5.9.3. Deze Object Pool is, zoals gebruikelijk [18], tevens een Singleton. In de huidige situatie wordt slechts vanaf één plek objecten uit de pool gehaald. Een normaal object zou in dit geval dus ook mogelijk zijn geweest. Echter is het, met oog op de toekomst, beter om hier toch een Singleton te gebruiken zodat later eventueel andere soorten kaarten toegevoegd kunnen worden met de zelfde pool. Op deze manier hoeft dan niet voor elke kaart apart tiles opgeslagen te worden.

Database

Doordat dezelfde database op meerdere plaatsen in de applicatie geraadpleegd wordt is het qua geheugen voordeliger om de benodigde gegevens in één Singleton object op te slaan, dan om op elke plaats waar de database nodig is een nieuw database object te initialiseren.

5.9.2 Singleton implementatie

In Objective-C is het niet mogelijk om het standaard Singleton Pattern te implementeren. Het is namelijk niet mogelijk om een private of protected constructor te maken, zoals bij een Singleton gebruikelijk is [19]. Om het verschil tussen een Singleton in Objective-C en in andere talen duidelijker te maken, gebruik ik in dit hoofdstuk twee voorbeelden. Het eerste voorbeeld is de Singleton implementatie in C# (Figuur 5.4), het tweede voorbeeld is de Singleton implementatie in Objective-C (Figuur 5.5).

```
class Singleton{
    private static Singleton instance;

    protected Singleton(){
    }

    public static Singleton Instance(){
        if (instance == null){
            instance = new Singleton();
        }

        return instance;
    }
}
```

Figuur 5.4 Singleton geïmplementeerd in C#

Het belangrijkste verschil tussen de Singleton in C# en in Objective-C is dat in C# de constructor protected is gemaakt, zodat andere objecten geen instanties van de Singleton klasse kunnen maken. In Objective-C is het echter niet mogelijk om een constructor protected te maken, dus heb ik op een andere manier moeten zorgen dat er slechts één object aangemaakt kan worden.

De *allocWithZone* methode wordt door de super klasse gebruikt om geheugen te reserveren voor het object. In plaats van de super aan te roepen return ik hier het Singleton object. In de methode die het Singleton object terug geeft wordt daarentegen wel de allocatie methode van de super klasse aangeroepen, zodat wanneer nog geen instantie bestond geheugen vrijgemaakt wordt om de instantie in op te slaan. Vervolgens kan dan de eventuele initialisatie van het Singleton object worden aangeroepen.

Daarnaast moeten in verband met het geheugenmanagement van Objective-C nog een aantal andere methoden overschreven worden. De methode *copyWithZone* geeft normaal gesproken een kopie van het huidige object terug, omdat van de Singleton slechts één object mag bestaan moet deze methode overschreven worden.

De *retainCount* methode geeft aan hoeveel objecten een referentie naar het huidige object hebben. Wanneer dit aantal 0 is wordt het geheugen van het object weer verijgegen. De retain count van het Singleton object is altijd *NSUIntegerMax* zodat deze nooit vernietigd wordt. De *retain* methode verhoogd normaal gesproken de retain count, om vervolgens het huidge object terug te geven. Omdat de retain count altijd op het maximum ligt is geeft deze methode nu slechts het huidige object terug.

De *release* methode is het tegenovergestelde van de *retain* methode en verlaagd de retainCount dus. Normaal wordt een object gereleased wanneer het niet meer nodig is. Wanneer het object dan zovaak gereleased is dat de retain count 0 wordt, wordt het object gedealloceerd.

```
static Singleton *sharedSingleton = nil;

+(Singleton *)sharedSingleton{
    if(sharedSingleton == nil){
        sharedSingleton = [[super allocWithZone:NULL] init];
        [sharedSingleton initialise];
    }

    return sharedSingleton;
}

+(id)allocWithZone:(NSZone *)zone{
    return [self sharedSingleton];
}

-(id)copyWithZone:(NSZone *)zone{
    return self;
}

-(id)retain{
    return self;
}

-(NSUInteger)retainCount{
    return NSUIntegerMax;
}

-(oneway void)release{
    //do nothing
}

-(id)autorelease[
    return self;
}

-(void)dealloc{
    //do nothing
}
```

Figuur 5.5 Singleton geïmplementeerd in Objective-C [20]

5.9.3 Object Pool klasse

De overstap naar losse tegels voor het opbouwen van de kaart, zie 5.10.1 Kaart in 3D, zorgde ervoor dat steeds nieuwe objecten 3D objecten werden aangemaakt om de texture van de kaart op te leggen. Hierdoor moest dus steeds wanneer een nieuwe rij van tegels aan de kaart werd toegevoegd voor elke tegel geheugen gealloceerd worden, en moest het object geïnitialiseerd worden. Tijdens een vlucht wordt zeer veel afstand afgelegd, waardoor vaak nieuwe tegels ingeladen moeten worden. Wanneer hogere zoomniveaus worden gebruikt, waardoor de tegels gedetailleerder worden maar minder meters per pixel bevatten, gebeurd het toevoegen van tegels nog vaker. Om te zorgen dat de allocatie en initialisatie niet steeds opnieuw uitgevoerd hoeft te worden heb ik een Object Pool gemaakt waarin de kaart tegels bijgehouden worden.

5.9.4 Object Pool implementatie

Voordat ik de Object Pool geïmplementeerd had, had ik de methode die tegels aan de kaart toevoegt en van de kaart verwijdert al geschreven. Ik wilde bij het maken van de Object Pool dus ook zorgen dat de bestaande code, inclusief het geheugen beheer, zo min mogelijk aangepast hoefde te worden. Aan de hand van een aantal code voorbeelden wil ik laten zien hoe de Object Pool is opgebouwd.

```
-(void)initPool{
    _pooledTiles = [[NSMutableArray alloc] init];
}
```

Figuur 5.6 Object Pool Initialisatie

Zoals beschreven in paragraaf 5.9.1 is de Object Pool een Singleton. Tijdens het ophalen van de instantie wordt eenmalig de *initPool* methode (Figuur 5.6) aangeroepen. Deze methode initialiseert een array waarin de 3D objecten voor de kaart tegels bijgehouden worden.

```
-(NCIMTileMesh *)dequeueReusableTileMesh{
    NCIMTileMesh *tileMesh = [self findFreeTile];

    if(!tileMesh){
        tileMesh = [[NCIMTileMesh alloc] init];
        [_pooledTiles addObject:tileMesh];
        [tileMesh release];
    {

    return tileMesh;
}
```

Figuur 5.7 Object ophalen uit de Object Pool

Om een object uit de Object Pool op te halen wordt de *dequeueReusableTileMesh* methode (Figuur 5.7) aangeroepen. Deze methode geeft een *NCIMTileMesh,* een 3D object waarop de kaart texture getekend kan worden, terug. In eerste instantie probeert de methode een ongebruikte tegel te vinden met behulp van de *findFreeTile* methode (Figuur 5.8). Wanneer geen ongebruikte tegels gevonden zijn, wordt een nieuwe aangemaakt. Deze wordt vervolgens toegevoegd aan de array met tegels.

```
-(NCIMTileMesh *)findFreeTile{
    for(NCIMTileMesh *tile in _pooledTiles){
        if(tile.retainCount == 1){
            [tile setTileInfo:nil];
            return tile;
        }
    }
    return nil;
}
```

Figuur 5.8 Vrij object in de Object Pool zoeken

In de *findFreeTile* methode (Figuur 5.8) wordt de array met tegels doorlopen op zoek naar een ongebruikte tegel. Wanneer de retain count van de tegel 1 is, is het enige object wat nog referentie naar de tegel heeft de array van tegels. In dit geval is de tegel nog vrij en wordt deze teruggegeven. Wanneer de retain count groter is dan 1 heeft ook een ander object nog een referentie naar de tegel, in dat geval is de tegel dus niet vrij. Wanneer geen vrije tegels gevonden worden, wordt een NULL waarde teruggegeven. Hierdoor weet de *dequeueReusableTileMesh* methode uit Figuur 5.7 dat er geen vrije tegels gevonden zijn, en dat er dus een nieuwe tegel aangemaakt moet worden.

Op deze manier hoeft bij het gebruik van tegels alleen de allocatie methode vervangen te worden door de *dequeueReusableTile* methode. Het vrijgeven van geheugen kan dus hetzelfde blijven als zonder Object Pool.

Alternatieve implementatie

Voordat ik de Object Pool heb geïmplementeerd heb ik eerst gezocht op voorbeeld implementaties in Objective-C. Ik kon echter geen voorbeelden vinden van Object Pools, echter vond ik wel een alternatieve implementatie.

De klasse uit de alternatieve implementatie moet gebruikt worden in plaats van de standaard superklasse *NSObject*. De klasse zorgt ervoor dat wanneer de retain count van een object 0 bereikt de deallocatie niet wordt uitgevoerd. In plaats daarvan wordt het object bewaard. Naast het object zelf is een *PoolInfo* object nodig waarin het laatste object van de objecten in de pool wordt opgeslagen. Alle objecten bewaren vervolgens weer het vorige object.

Wanneer een object vrijgegeven wordt, wordt dit als laatste object van de *PoolInfo* geplaatst. Daarbij wordt het voormalige laatste object als vorig object van het nieuwe laatste object opgeslagen. Wanneer vervolgens weer een object uit de pool gehaald wordt, wordt het laatste object gebruikt als dit vrij is, anders wordt een nieuw object aangemaakt. Als het laatste object gebruikt wordt, wordt vervolgens het een-na-laatste object opgeslagen als laatste object in de *PoolInfo*.

Deze klasse doet eigenlijk precies wat ik van de Object Pool wilde, inclusief dat de het gebruik van de objecten niet veranderde. Echter is het helaas niet mogelijk om in Objective-C van twee klassen te overerven. Omdat de tile klasse al overerft van de *NGLTileMesh* klasse kon ik dit alternatief in dit geval niet toepassen.

5.10 Knelpunten

Tijdens het programmeren kwamen een aantal knelpunten naar voren. In dit hoofdstuk worden de grootste en meest ingrijpende knelpunten behandeld.

5.10.1 Kaart in 3D

Nadat het probleem uit hoofdstuk 5.6, de mogelijkheid tot offline kaarten, was opgelost kwam ik tot een volgend probleem. Namelijk weergeven van een 2D view in een 3D wereld. In eerste instantie wilde ik de kaart kantelen om hier perspectief in te krijgen, en vervolgens de 3D wereld over de kaart heen projecteren. De 3D wereld zou dan een transparante achtergrond hebben. Wanneer een OpenGL view een transparante achtergrond krijgt, heeft dit echter grote gevolgen voor de performance [21]. Ik heb daarom gekozen een manier te zoeken om de 2D view in de 3D wereld te krijgen. Hiervoor had ik twee opties:

1. De kaart tekenen als texture op een 3D object.
2. De kaart zelf opbouwen met 3D objecten.

Wanneer ik de kaart zelf moet opbouwen, moet ik zelf de latitude en longitude omzetten naar x en y coördinaten. Bij het natekenen van de kaart is dit al door het route-me framework berekend en hoef ik dit dus niet zelf te berekenen. In eerste instantie ben ik dus ook voor deze oplossing gegaan.

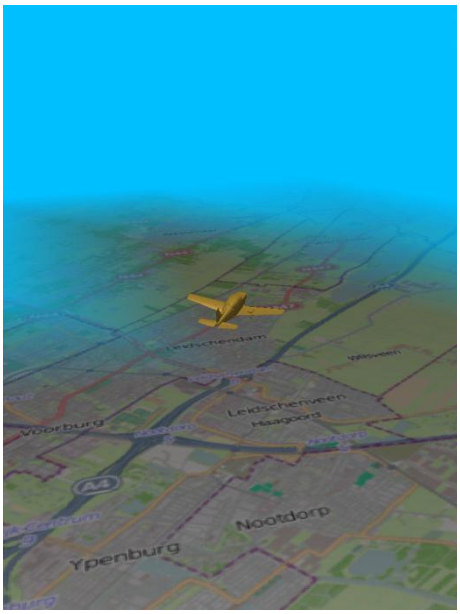
View omzetten naar texture

Binnen het iOS framework is het mogelijk een view te tekenen in een plaatje (Figuur 5.9). Dit plaatje kan vervolgens gebruikt worden om als texture aan een 3D object toegevoegd te worden. Door dit steeds te doen wanneer de locatie, en daarmee de kaart, wordt geüpdatet blijft lijkt het vliegtuig over de 3D kaart te verplaatsen, terwijl in feite steeds de texture veranderd.

```
//Create a graphics context, based on the view's size
UIGraphicsBeginImageContext(_mapView.bounds.size);
//Render the map into the context
[_mapView.layer renderInContext:UIGraphicsGetCurrentContext()];
//Create an image from the context
UIImage *mapImage = UIGraphicsGetImageFromCurrentImageContext();
//End the context
UIGraphicsEndImageContext();
```

Figuur 5.9 Image object vanuit een view object

De kaart wordt buiten het scherm getekend, zodat de gebruiker deze niet ziet. Om een plaatje te tekenen is het echter wel nodig dat de 2D view, in dit geval de kaart, geplaatst wordt in de actieve applicatie. Dat deze niet zichtbaar is heeft geen effect.



Figuur 5.10 3D kaart vanuit een 2D view

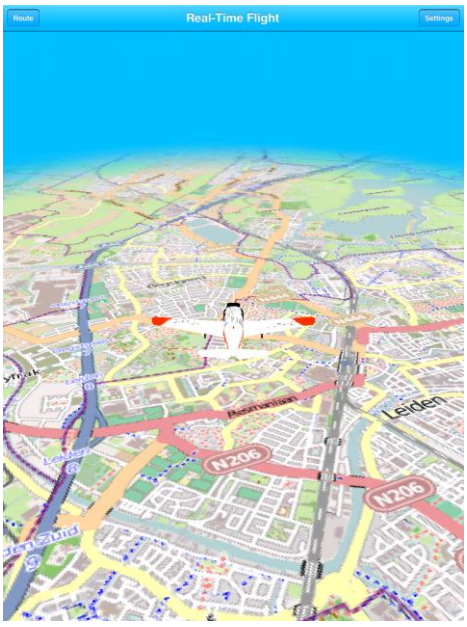
Door op de randen van de 3D kaart een mist toe te voegen zijn de randen van de kaart niet zichtbaar, hierdoor lijkt de kaart in de verte door te lopen, zoals te zien in Figuur 5.10 Om de kaart plat te kunnen leggen en toch de randen buiten beeld te houden heb ik de 2D kaart twee keer zo groot gemaakt als het scherm van de iPad.

Het nadeel van deze oplossing is echter dat de bewerking om de 2D view in een plaatje te tekenen zeer veel tijd kost. Deze bewerking duurde op de iPad1 ongeveer 2,5 seconden. Om dit toch sneller te laten gaan heb ik de 2D kaart weer dezelfde grootte gemaakt als het iPad scherm, maar liet ik ondertussen wel het oppervlak waarop deze getekend werd even groot. Op deze manier werd de kaart dus een klein beetje opgerekt. Dit was niet storend wat betreft hoe het er uit zag, echter was de snelheid winst die het opleverde nog niet voldoende. Het duurde namelijk nog steeds ongeveer 0,8 seconde op de iPad1 en 0,3 seconde op de iPad2.

Kaart opbouwen met losse tegels

Omdat de performance van de vorige oplossing, het natekenen van de kaart op een 3D object, niet voldoende was heb ik besloten zelf de kaart op te bouwen vanuit 3D objecten. Dit leverde echter een volgend probleem op; de huidige GPS locatie moet omgezet worden naar x en y coördinaten. De losse tegels zijn opgeslagen in de database op basis van een x, y en z positie. De x en y zijn oplopend van vanaf het linker boven punt op een 2D kaart. De z positie geeft het zoomniveau aan, en bepaald daarmee hoeveel tegels er totaal zijn. Ik moest dus op basis van de locatie (latitude en longitude) en het zoomniveau bepalen wat de x en y van de zichtbare tegels is.

Om de tegels te downloaden van OpenStreetMap heb ik de Geo::OSM::Tiles tool [15], geschreven in Perl, gebruikt. Deze tool downloadde de tegels op basis van latitude, longitude en zoomniveau, dus wist dit om te zetten naar x en y coördinaten. Na flink zoeken vond ik de methodes waarin dit gedaan werd en ik heb de benodigde formules er uit gehaald. Wanneer ik met deze formules zelf de latitude & longitude omzette naar x en y coördinaten kwam hier steeds een verkeerde uitkomst uit. Omdat dit na een aantal dagen, en de hulp van collega's met een wiskunde achtergrond, nog niet gelukt was heb ik weer teruggegrepen naar het route-me framework.



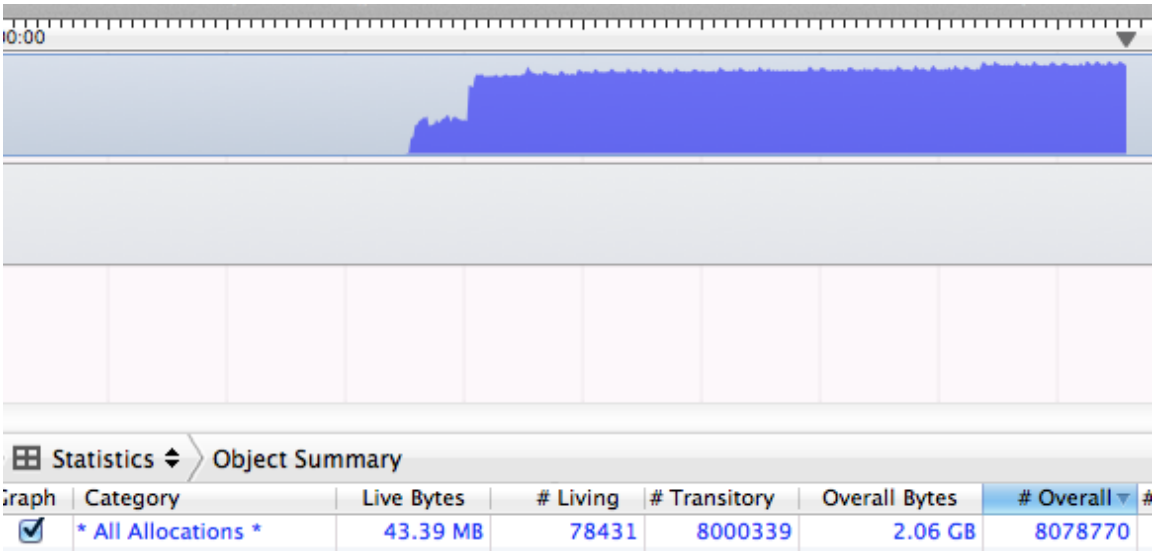
Figuur 5.11 3D Kaart vanuit losse tegels

Binnen het route-me framework wordt een ander framework gebruikt om de werkelijke berekening uit te voeren. Het route-me framework heeft dan weer methodes die de uitkomst omzetten naar gemakkelijk te gebruiken objecten. Omdat het route-me framework erg groot is heeft het nog wel een aantal dagen geduurd voordat ik alle benodigde methodes had gevonden. Het route-me framework bood nog een voordeel; de goede plaatjes worden direct uit de database gehaald. Hiervoor hoefde ik dus niet zelf nog aparte methodes te schrijven.

De plaatjes die uit de database kwamen heb ik vervolgens als textures op losse tegel objecten toegevoegd. Van deze tegelobjecten heb ik een raster gemaakt van 11 bij 11 tegels (Figuur 5.11). De middelste tegel wordt hierbij gebaseerd op de huidige locatie van de gebruiker, terwijl de tegels er omheen relatief aan de middelste tegel geplaatst en opgehaald worden.

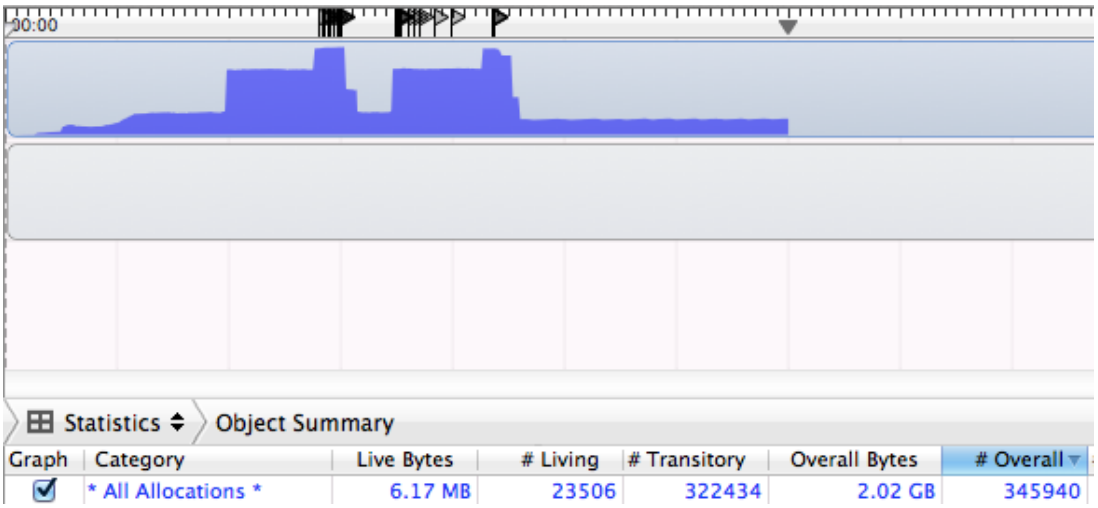
5.10.2 Geheugenproblemen

Voordat ik van aanpak wisselde omtrent het inladen van de kaart in de 3D werd de kaart zowel in 2D als 3D geladen, echter werd de 2D kaart buiten beeld gehouden. Omdat de kaart in de 3D weergave gekanteld was moest de kaart ook groter zijn dan het iPad scherm, waardoor de off screen kaart groter getekend moest worden dan gebruikelijk. De applicatie werd op de iPad bij het testen steeds gesloten door het OS vanwege geheugenoverschrijdingen. Hierin lag ook een groot verschil tussen de simulator en de werkelijke iPad.



Figuur 5.12 Geheugengebruik route-me op de iPad simulator

Zoals te zien in Figuur 5.12 wordt ruim 43 MB RAM geheugen gebruikt, zonder dat dit wordt vrijgegeven. In het geval van de iPad simulator is dit echter geen probleem, omdat de simulator het gehele geheugen van de Mac gebruikt zonder de limieten die de iPad wel heeft toe te passen.



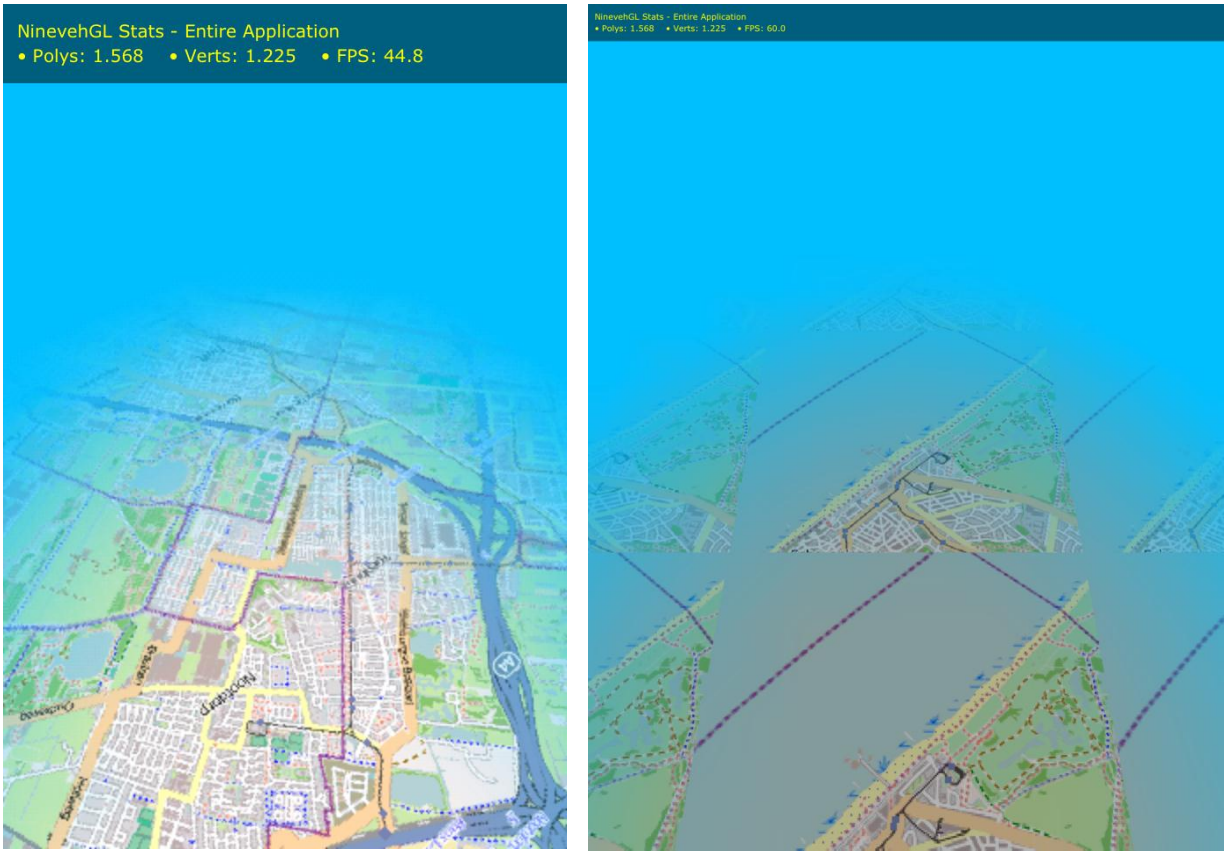
Figuur 5.13 Geheugengebruik route-me op de iPad

In figuur Figuur 5.13 is te zien dat het geheugengebruik op de iPad er heel anders uitziet dan op de iPad simulator. Na het inladen wordt slechts 6 MB geheugen gebruikt, tegen de 43 MB bij de simulator. Echter zijn ook hier de grote allocaties te zien.

De vlaggetjes boven de geheugengrafiek geven aan wanneer een geheugenwaarschuwing wordt afgegeven. Wanneer bij een geheugenwaarschuwing niet snel genoeg gereageerd wordt, door meer geheugen vrij te geven, kan het OS de applicatie killen. Daardoor crasht de applicatie vaak al tijdens het opstarten. Doordat ik nu, zoals beschreven in paragraaf 5.10.1, zelf de kaart direct naar de 3D wereld inlaad wordt het off screen tekenen overgeslagen.

5.10.3 Render problemen

Het grootste probleem tijdens de opdracht was het probleem dat kaarttegels de verkeerde texture toegewezen kregen. Opmerkelijk hierbij was dat de problemen alleen voorkwamen op de iPad, terwijl de iPad simulator geen problemen gaf (Figuur 5.14).



Figuur 5.14 Kaart tegels gaan goed op de simulator (links) maar fout op de iPad (rechts)

Het oplossen van dit probleem heeft meerdere weken geduurd. Vandaar dat de gevolgen voor het project zeer groot waren. De vertraging die ik door dit probleem heb opgelopen heb ik niet meer in kunnen halen waardoor niet alle afgesproken functionaliteiten zijn geïmplementeerd. Dit wordt uitgebreider besproken in hoofdstuk 5.4. Gedurende de periode dat ik met dit probleem bezig was ben ik tot de volgende mogelijke oplossingen gekomen:

- Object formaat van NinevehGL bevat een fout,
- Fouten in de loop waarin kaart tegels worden aangemaakt,
- Gebrek aan resources op de iPad,
- Verschil tussen OpenGL en OpenGL ES,
- Fout in tegel model.

In deze paragraaf worden de oplossingen één voor één besproken, gevolgd door de werkelijke oplossing.

Object formaat van NinevehGL

Zoals beschreven in paragraaf 5.5.1 zet NinevehGL ingeladen 3D objecten om naar een eigen formaat, zodat deze in het vervolg sneller ingeladen kunnen worden. Hoe dit eigen object formaat in elkaar zit is echter niet bekend. In eerste instantie dacht ik daarom dat het probleem met de kaarttegels werd veroorzaakt doordat hetzelfde object steeds hergebruikt werd, inclusief de texture. Door het eigen object formaat van NinevehGL uit te schakelen hoopte ik dat het probleem opgelost zou zijn, dit bleek echter niet het geval. Het probleem ligt dus niet bij het object formaat van NinevehGL.

Kaart tegel loop

Nadat ik het object formaat van NinevehGL had uitgesloten heb ik de loop waarin de kaart tegels worden aangemaakt bekeken. De code hiervan is te vinden in Bijlage G. De code was geschreven voordat een Object Pool was toegepast op de kaart tegels, de *NCIMTileMesh* objecten worden dus nog apart gealloceerd en geïnitialiseerd. Het eerste onderdeel van de loop dat ik bekeken heb waren de geheugenadressen van de objecten die worden aangemaakt. Ik heb elk van de geheugenadressen naar een log geschreven om te controleren of niet bijvoorbeeld alle textures naar hetzelfde adres wijzen. Alle objecten hadden echter een eigen geheugenadres dus dat was niet het geval. Om te garanderen dat het geheugen van de objecten ook werkelijk wordt vrijgegeven wanneer dat nodig is heb ik een aantal aanpassingen gedaan aan de loop, waarvan de voornaamste het toevoegen van een *AutoReleasePool* om te garanderen dat geheugen opgeruimd wordt aan het einde van de loop.

Om te kijken of de plaatjes ook echt verschillend zijn heb ik deze omgezet naar een hexadecimale waarde, deze heb ik vervolgens naar een log geschreven. Net als de geheugenadressen verschilde ook deze waarden. Om uiteindelijk zeker te zijn dat de plaatjes in de loop echt verschilde heb ik ze los toegevoegd aan de view, en zodoende een 2D kaart opgebouwd. Deze kaart was precies zoals verwacht, met verschillende tegels op de juiste plaats.

Resource gebrek

Nadat de plaatjes in een 2D weergave wel aan de verwachtingen voldeden heb ik contact gezocht met de maker van de 3D engine. Omdat de maker in Brazilië woont was contact alleen via e-mail mogelijk. Ook zorgde het tijdsverschil voor nogal vertraging in de antwoorden die we elkaar konden geven. In mijn eerste mail heb ik het probleem uitgelegd en de code van de loop (Bijlage H) toegevoegd. De maker herkende dit probleem, omdat ze het al eerder hadden opgelost. Daarom stuurde hij me naast een paar tips ook nog de laatste versie van het Framework, waarin het probleem zeker opgelost was. Dit loste mijn probleem echter niet op.

De maker van de engine vroeg me vervolgens het gehele project op te sturen, zodat hij zelf kon kijken waar het probleem lag. Ik kreeg al vrij snel antwoord, inclusief een aantal screenshots, waarin hij liet zien dat het bij hem wel werkte. Daardoor bedacht ik me dat het ook aan iPad1, de enige iPad die binnen de NCIM Groep beschikbaar was, kon liggen. De iPad1 heeft namelijk aanzienlijk minder RAM dan de nieuwere varianten. De maker van de engine heeft me, ook nadat ik hiernaar gevraagd heb, nooit verteld welke iPad variant hij gebruikt, maar vertelde me wel dat het een stuk moeilijker is om een 3D applicatie voor de iPad 1 te maken. Doordat de opdrachtgever zelf de Nieuwe iPad kocht schoof hij zijn iPad2 door naar ons. Helaas was ook het overstappen naar deze iPad niet de oplossing.

Verschil tussen OpenGL en OpenGL ES

Tijdens het wachten op antwoord van de maker van de engine heb ik onder andere veel gelezen over 3D programmeren. Eén van de artikelen was geschreven door de maker van de engine en ging over OpenGL. Hierin werden onder andere de verschillen tussen de gewone en de mobiele versie beschreven [22]. Een groot verschil zit in de opbouw van modellen. Modellen kunnen worden opgebouwd uit driehoeken (triangles) en vierkanten (quads). De mobiele versie van OpenGL,OpenGL ES, ondersteund deze quads niet.Ik heb het tegel model geopend en zag dat het was opgebouwd uit quads. Dit kon mogelijk het probleem zijn omdat de simulator de OpenGL versie van de Mac gebruikt, dus niet de mobiele versie, terwijl de iPad wel met de mobiele versie werkt.

Ik heb dit daarom naar de maker van de engine gestuurd, met de vraag of dit het probleem kon zijn. Vervolgens heb ik ook het tegel model omgezet naar triangles. Ook dit had geen effect. De maker van de engine antwoordde dan ook dat het inderdaad klopt dat OpenGL ES geen quads kan renderen, maar dat de engine dit al afvangt.

Fout in het tegel model

Gedurende de periode dat ik met dit probleem zat heb ik regelmatig met collega's gesproken over wat het probleem kan zijn. De meeste ideeën die ze opperde had ik zelf ook al gehad. Eén van de collega's zei dat het ook kan liggen aan het soort model dat ik gebruik voor de kaart tegels. Hiervoor heb ik namelijk een plane gebruikt. Een plane is een 2D object maar wel in een 3D wereld, het heeft dus een lengte en een breedte, maar geen hoogte. Dit object is ook alleen vanaf de bovenkant zichtbaar, wanneer naar de onderkant gekeken wordt zou het object doorzichtig moeten zijn maar de collega had zelf al eens ervaren dat dit weergave problemen kon veroorzaken.

Ik heb daarop twee dingen gedaan om te garanderen dat de plane goed gebruikt werkt. Allereerst heb ik de planes op hun kop getekend, zodat de onderkant zichtbaar zou zijn. Toen werden de kaarttegels inderdaad doorzichtig, en was dus niets meer te zien. Daarnaast heb ik ook kubussen gemaakt van de kaart tegels, zodat hoe deze ook geplaatst worden een deel van de textures zichtbaar moest zijn. Nog steeds werd steeds dezelfde texture herhaald.

Fout in de engine

Op verzoek van de maker van de engine had ik een nieuwe versie van mijn project opgestuurd, zodat hij er nog een keer naar kon kijken. In afwachting van zijn bevindingen heb ik zelf de textures weggelaten en vervangen door kleuren. Ik heb in deze in de loop twee kleuren om en om aan de tegels toegewezen, waardoor een soort schaakbord ontstond. Hierbij ging de volgorde wel goed. Daarna heb ik het met 3 kleuren geprobeerd en ook dat ging goed. Het probleem lag dus zeker bij de textures.

Wanneer een nieuw 3D project aangemaakt wordt volgens het template van NinevehGL wordt automatisch een kubus aan het project toegevoegd. Deze kubus heeft een aparte texture voor elke kant. Ik heb met deze textures hetzelfde gedaan als de kleuren, twee textures om en om plaatsen. Net als bij de kleuren werd steeds de goede texture getekend. Ook bij 3, 4, 5 en 6 verschillende textures bleef de volgorde goed gaan. Zodoende kwam ik op de oplossing.

Oplossing

Het verschil tussen de meegeleverde textures en de kaart tegel textures is dat de meegeleverde textures is de manier waarop deze worden geïnitialiseerd. De meegeleverde textures worden

geïnitialiseerd vanuit een bestand (Figuur 5.15), terwijl de kaart tegel textures geïnitialiseerd worden vanuit een *UIImage* object (Figuur 5.16).

```
tempTileTexture = [[NGLTexture alloc] init2DWithFile:tempTile.imageFile];
```

Figuur 5.15 Texture vanuit een bestand

```
tempTileTexture = [[NGLTexture alloc] init2DWithImage:tempTile.image];
```

Figuur 5.16 Texture vanuit een *UIImage*

Om te testen of dit inderdaad het geval was heb ik de meegeleverde textures eerst ingeladen in een *UIImage* object en vervolgens aan de hand daarvan de texture geïnitialiseerd. Dat bleek inderdaad het probleem, want ook de meegeleverde textures werden fout getekend wanneer vanuit een *UIImage* object gewerkt werd.

Gedurende de tijd dat ik aan het probleem gewerkt heb, heb ik ook de Object Pool voor de kaart tegels geïmplementeerd. Aan de hand daarvan was dus ook de loop al enigszins veranderd. Om de texture initialisatie op basis van een *UIImage* te ontwijken heb ik een aantal aanpassingen gemaakt. Wanneer tegel objecten uit de database gehaald worden, wordt het plaatje direct opgeslagen in de cache map van de applicatie. De locatie van dit plaatje wordt vervolgens in het tegel object opgeslagen. Vervolgens wordt de initialisatie op basis van een bestand gebruikt om de texture te maken. In de Object Pool worden de plaatjes uit de cache map verwijderd wanneer de tegel klaar gemaakt wordt voor hergebruik. Bij het opstarten van de applicatie wordt de gehele tegel cache geleegd, zodat plaatjes die niet gebruikt worden niet onnodig ruimte op de iPad in blijven nemen.

Communicatie over de oplossing

Op het oment dat ik dacht de oplossing gevonden te hebben, heb ik deze oplossing naar de maker van de engine gemaild en gevraagd of hij dat kon controleren en verifiëren. In afwachting van zijn antwoord ben ik zelf al begonnen met het implementeren van de oplossing die ik had bedacht. Toen dit de oplossing bleek te zijn heb ik dit, inclusief mijn code, opgestuurd met wederom de vraag of hij kon verifiëren of dit echt het probleem oplost.

In zijn antwoord negeerde hij echter compleet mijn oplossing en zei dat hij me de oplossing een paar dagen eerder al had gestuurd. Toen ik zei dat ik iets had ontvangen kreeg ik te horen dat hij het nog eens gestuurd had, wederom had ik niets ontvangen. Ik heb hem toen nog gevraagd het naar een ander e-mail adres te sturen, sinds toen heb ik niets meer gehoord. Het ziet er dus niet naar uit dat het probleem opgelost gaat worden.

6 Evaluatie

Tijdens de afstudeeropdracht heb ik veel kunnen leren, in dit hoofdstuk bespreek ik wat ik heb geleerd en wat beter of anders had gemoeten zowel op product als op procesniveau.

6.1 Productevaluatie

Uit de afstudeeropdracht zijn uiteindelijk twee belangrijke eindproducten gekomen; het onderzoeksrapport en de iPad applicatie. Daarnaast zijn een aantal tussenproducten opgeleverd. In deze paragraaf evalueer ik de (tussen)producten individueel.

Vooronderzoek

Het vooronderzoek heeft mij geholpen met het afbakenen van het onderzoek en heb ik op basis van het vooronderzoek de deelvragen kunnen opstellen. Het gaf mij inzicht in de verschillende methodes om platformonafhankelijke mobiele applicaties te ontwikkelen en in de tools die daarvoor gebruikt kunnen worden. Daarnaast heeft het uitzoeken van de eisen van de applicatiewinkels ook zeker geholpen bij het bepalen van de geschiktheid van de tools, omdat een applicatie die niet toegelaten wordt niet nuttig is. Daarnaast heb ik tijdens het uitvoeren van het vooronderzoek veel gelezen over platformonafhankelijke mobiele applicatie ontwikkelen.

Onderzoeksrapport

Ook over het resultaat van het onderzoeksrapport ben ik tevreden. Het beantwoord de vraag van de opdrachtgever. De resultaten uit het onderzoeksrapport zijn tevens al gebruikt bij het bepalen hoe de SocialSOS applicatie ontwikkeld ging worden.

Over de deelvragen die ik heb opgesteld ben ik echter niet tevreden. Zowel de deelvragen zelf als de volgorde waarin ze staan had beter gekund. In deelvraag 1 werd nu bepaald met welke tools geëxperimenteerd ging worden, terwijl daarna pas werd bepaald welke functionaliteiten tijdens dit experiment geïmplementeerd moesten worden. In de praktijk heb ik dit nu opgelost door mijn antwoord op deelvraag 1 bij te stellen na het beantwoorden van deelvraag 2.

Deelvraag 2 was nu “Welke functionaliteiten moet het Proof of Concept bevatten?”, terwijl in werkelijkheid antwoord wordt gegeven op de vraag “Welke functionaliteiten komen het meest voor in mobiele applicaties?”. Achteraf gezien had dat een betere vraag geweest, het antwoord daarop zou dan alsnog gebruikt kunnen zijn bij het uitvoeren van de experimenten het bepalen van de tools & frameworks.

Klassendiagram

Bij het maken van het klassendiagram heb ik, in overleg met mijn begeleider, de keuze gemaakt om de Objective-C notatie aan te houden zodat toekomstige programmeurs die op dit project worden gezet de applicatie sneller kunnen begrijpen. Ik ben echter achteraf niet tevreden over het resultaat. Ik moest een afweging maken tussen het klassendiagram overeen laten komen met de code en het volgen van de standaarden in UML diagrammen. Achteraf gezien denk ik dat ik beter de UML standaarden had kunnen volgen zodat externen het diagram beter kunnen begrijpen en het ook gemakkelijker om te zetten is naar een ander platform.

iPad applicatie

Doordat vertraging opgelopen is met het in 3D weergeven van een kaar zij niet alle functionaliteiten die vooraf zijn afgesproken geïmplementeerd. Wel is de iPad app zoals deze nu is werkend opgeleverd.

Het maken van de iPad applicatie verliep helaas niet volgens planning. Ik heb een verkeerde inschatting gemaakt wat betreft de moeilijkheid van het programmeren in 3D. Het leren van programmeren met OpenGL zou nog meer tijd gekost hebben dan het selecteren van een 3D engine, maar ook dit kostte veel tijd. Ook het inladen van de kaart bracht grote problemen met zich mee. Hierdoor werd de vertraging dusdanig dat ik functionaliteiten heb moeten schrappen. In overleg met de opdrachtgever zijn dit de weergegevens geworden.

Het opgeleverde product is bruikbaar voor zowel piloten als de sales afdeling. De sales afdeling kan gebruik maken van de demonstratiemodus die ik ingebouwd heb. De piloten kunnen deze uitschakelen om de applicatie live gegevens te laten gebruiken. Helaas heb ik de applicatie nog niet in de lucht kunnen testen, dit ga ik mogelijk nog wel samen met de opdrachtgever doen.

6.2 Procesevaluatie

Over het proces dat ik tijdens het project doorlopen heb ben ik over het algemeen tevreden. Ondanks dat de opgeleverde applicatie nog niet alle functionaliteiten bevat heb de problemen wel op tijd gesignaleerd en met de opdrachtgever besproken om tot een bevredigend resultaat te komen.

Vooral het werken met de Scrum methodiek is mij goed bevallen en was ook zeer leerzaam. Door constant een week vooruit te plannen met kleine, specifieke taken merkte ik dat mijn productiviteit omhoog ging. Aan het begin was het maken van een sprint planning nog erg wennen, maar gaandeweg ging dit steeds beter. Op school heb ik weinig over Scrum geleerd, waardoor ik door het te gebruiken ook echt iets nieuws heb geleerd. In volgende projecten ga ik wel meer aandacht besteden aan het Product Backlog, dit heb ik nu pas één keer gemaakt maar het is de basis van de sprint planningen en daarmee ook zeer belangrijk. Om meer kennis over Scrum te vergaren ga ik vanuit de NCIM Groep dan ook een Scrum Master training volgen om Certified Scrum Master te worden.

Doordat de NCIM Groep steeds meer iOS projecten opstartte heb ik ook meer tijd besteed aan het helpen en opleiden van anderen. Dit vond ik erg leuk om te doen, maar kostte eigenlijk teveel tijd van mijn eigen proces. In een volgende opdracht ga ik dus ook minder tijd besteden aan het helpen van anderen zodat ik me beter op mijn eigen project kan concentreren.

Door vooraf met de opdrachtgever zijn eisen en wensen door te spreken en daarbij ook te kijken naar andere belanghebbende, in dit geval de sales afdeling, kwamen er tijdens de uitvoer van het project geen nieuwe eisen bij. Hier ben ik zeer tevreden over, omdat ik hier tot nog toe zowel tijdens school projecten als daarbuiten hiermee de fout ben ingegaan. In eerdere projecten kwamen er altijd tijdens het project nieuwe eisen bij waardoor eerder geschreven code aangepast moest worden of de planning werd overschreden. Vaak was een combinatie van die laatste twee het gevolg. Ik ga in de toekomst zeker door met veel tijd besteden aan het bepalen en vastleggen van de requirements zodat deze tijdens het traject niet meer veranderen.

Over het onderzoek ben ik voornamelijk tevreden over het vooronderzoek. Tijdens het vooronderzoek heb ik op basis van veel verschillende bronnen veel feiten over het maken van platformonafhankelijke applicaties, maar ook applicaties in het algemeen gevonden. Hierdoor had ik voldoende basis om het onderzoek op te stellen en uit te voeren.

Tijdens het daadwerkelijke onderzoek had ik liever vast willen houden aan mijn originele plan van één Proof of Concept applicatie in plaats van experimenten met verschillende tools. Door met meerdere tools te experimenteren kwam ik nu eigenlijk tijd te kort en kon ik het onderzoek minder grondig uitvoeren dan ik had gewild. Echter omdat de opdrachtgever vooral geïnteresseerd was in een oplossing met HTML5 en ik dacht dat dit niet de beste oplossing was ben ik toch voor de huidige aanpak gegaan. Zo kan de opdrachtgever namelijk ook de verschillen zien, in plaats van slechts een theoretische onderbouwing.

7 Bewijs beroepstaken

In dit hoofdstuk wordt per gekozen beroepstaak aangegeven waar in het verslag naar voren komt dat deze gehaald is. Daarnaast heb ik ook nog een aantal extra beroepstaken aangehaald die aan bod zijn gekomen tijdens de opdracht, maar niet in het afstudeerplan waren opgenomen.

7.1 Uitvoeren analyse door definitie van requirements

In hoofdstuk 5.3 Requirements op pagina 29 heb ik beschreven dat ik met de opdrachtgever de requirements heb besproken en zijn wensen vervolgens heb omgezet in User Stories binnen het template dat de NCIM Groep gebruikt (beoordelingscriterium 5). Deze heb ik vervolgens inclusief prioriteiten teruggekoppeld naar hem om te controleren dat ik zijn eisen goed heb geïnterpreteerd (beoordelingscriteria 1, 2 & 4). De requirements zoals deze besproken zijn komen terug in het Use Case diagram (beoordelingscriterium 3).

7.2 Ontwerpen systeemdeel

In hoofdstuk 5.7 Klassendiagram op pagina 36 heb ik beschreven hoe ik tot het klassendiagram ben gekomen. Hierin wordt onder andere de keuze voor de tool (beoordelingscriterium 1) beschreven. Daarnaast heb ik in hoofdstuk 5.8 Use Case diagram op pagina 36 het Use Case diagram beschreven. In het afstudeerplan was ik tevens van plan de User Interaction te beschrijven, deze is echter voortgeborduurd op de User Interface richtlijnen van Apple waardoor een aparte beschrijving niet nodig is (beoordelingscriterium 3).

Zowel het klassendiagram als het Use Case diagram komt overeen met de eisen van de opdrachtgever en de gebruikte design patterns zijn in het klassendiagram terug te zien (beoordelingscriterium 2).

7.3 Bouwen applicatie

In hoofdstuk 5.9 Opbouw van de uitbreiding op pagina 37 heb ik beschreven hoe ik de gebruikte design patterns geïmplementeerd heb en hoe ik tot deze design patterns ben gekomen.

In hoofdstuk 5.10 Knelpuntenop pagina 43 heb ik beschreven tegen welke problemen ik aan ben gelopen en hoe ik deze heb opgelost.

Bij het schrijven de applicatie heb ik rekening gehouden met de uitbreidbaarheid en aanpasbaarheid (beoordelingscriterium 1). Daarnaast heb ik de applicatie dusdanig opgebouwd dat deze zonder aanpassingen geïntegreerd kan worden in een andere applicatie.

Tijdens het bouwen heb ik gebruik gemaakt van SVN voor versiebeheer en heb ik ontwikkeld in de ontwikkelomgeving Xcode (beoordelingscriterium 2). Dit ontwikkelen is gedaan in Objective-C met het Cocoa Touch Framework van Apple (beoordelingscriterium 2).

De applicatie is werkend opgeleverd (beoordelingscriterium 3).

7.4 Overig

Tijdens de opdracht zijn nog een aantal andere beroepstaken bewezen, dit zijn de volgende beroepstaken:

- Voorbereiden en opstarten softwareontwikkeltraject,
- Ontwerpen, bouwen en bevragen van een database.

Daarnaast heb ik ook nog onderzoek gedaan, iets wat wel op school is behandeld maar niet terugkomt in de beroepstaken. Aan de hand van dit onderzoek is wel advies gegeven over te gebruiken tools & frameworks, iets wat wel terugkomt bij de beroepstaken *Selecteren van standaardsoftware* en *Selecteren methoden, technieken en tools*.

7.4.1 Voorbereiden en opstarten softwareontwikkeltraject

In het Plan van Aanpak heb ik de probleemstelling beschreven, dit komt terug in hoofdstuk 3.2 Probleemstelling op pagina 7 (beoordelingscriterium 1).

Ik heb de problemen met de huidige applicatie in kaart gebracht, dit wordt beschreven in hoofdstuk 5.1 Huidige status op pagina 27 (beoordelingscriterium 2).

De wensen van de opdrachtgever heb ik geformuleerd middels User Stories in een Product Backlog, zoals te zien in hoofdstuk 5.3 Requirements op pagina 29. In dat zelfde hoofdstuk heb ik ook aangekaart dat het niet mogelijk is om real-time weergegevens te ontvangen (beoordelingscriterium 3).

De applicatie is dusdanig ontworpen dat dit zowel aan de eisen vanuit de piloot als vanuit de eisen van de sales representative die de applicatie op beurzen moet aanprijzen voldoet (beoordelingscriterium 4). Dit wordt beschreven in hoofdstuk 5.3 Requirements op pagina 29.

7.4.2 Ontwerpen, bouwen en bevragen van een database

Voor de Mobile Flight Assistant applicatie heb ik een kleine database gebouwd waarin de kaart tegels, routes en waypoints worden opgeslagen. In de paragraaf Tile Database van hoofdstuk 5.6 Kaart selectie op pagina 35 wordt beschreven dat ik hierin gebruik heb gemaakt van secundaire indexen (beoordelingscriterium 2).

In de paragraaf Database van hoofdstuk 5.9.1 Singleton Klassen op pagina 38 heb ik beschreven dat ik een klasse gemaakt heb die de communicatie afhandelt tussen de applicatie en de database, deze doet dit middels SQL queries (Beoordelingscriterium 5).

8 Bronnen

[1] N. Verhoeven, Wat is onderzoek?, 4e red., Den Haag: Boom Lemma uitgevers, 2011.

[2] H. Behrens, „Cross-Platform App Development for iPhone, Android & Co - A Comparison | Presented at Mobile TechCon 2010," 11 oktober 2010. [Online]. Available: http://heikobehrens.net/2010/10/11/cross-platform-app-development-for-iphone-android-co-%E2%80%94-a-comparison-i-presented-at-mobiletechcon-2010/. [Geopend 9 februari 2012].

[3] Apple, „iOS UI Element Usage Guidelines," Apple, [Online]. Available: http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/UIElementGuidelines/UIElementGuidelines.html#//apple_ref/doc/uid/TP40006556-CH13-SW48. [Geopend 10 februari 2012].

[4] Google, „Tab Layout," Google, [Online]. Available: http://developer.android.com/resources/tutorials/views/hello-tabwidget.html. [Geopend 10 feburari 2012].

[5] Microsoft, „Application Tabs (Pivot Control) for Windows Phone," Micorosoft, [Online]. Available: http://msdn.microsoft.com/en-us/library/hh202890(v=vs.92).aspx. [Geopend 10 februari 2012].

[6] Apple, „App Store Review Guidelines," Apple, 2012. [Online]. Available: https://developer.apple.com/appstore/resources/approval/guidelines.html#user-interface. [Geopend 2 februari 2012].

[7] J. Fahrenkrug, „Thoughts on Mobile UI Design," Springenwerk, 12 september 2011. [Online]. Available: http://www.springenwerk.com/2011/09/thoughts-on-mobile-ui-design.html. [Geopend 27 februari 2012].

[8] W3C, „FAQs - HTML5 Wiki," W3C, 22 februari 2011. [Online]. Available: http://www.w3.org/html/wiki/FAQs#When_will_HTML5_be_done.3F. [Geopend 28 februari 2012].

[9] J. v. Oost, „Toekomst is aan HTML5-vormgevers/programmeurs," ZDNet, 21 februari 2012. [Online]. Available: http://www.zdnet.be/techzone/136580-1/-toekomst-is-aan-html5-vormgevers-programmeurs-/. [Geopend 28 februari 2012].

[10] N. Shah, „Global HTML5 Handset Sales Forecast," Strategy Analytics, 7 december 2011. [Online]. Available: http://www.strategyanalytics.com/default.aspx?mod=pressreleaseviewer&a0=5145. [Geopend 27 februari 2012].

[11] K. S. &. J. Sutherland, „The Scrum Guide," oktober 2011. [Online]. Available: http://www.scrum.org/storage/scrumguides/Scrum_Guide.pdf. [Geopend 15 mei 2012].

[12] FreeFlight Systems, „ADS-B For Beginners," 21 december 2011. [Online]. Available: http://www.google.nl/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CDAQFjAA&url=http%3A%2F%2Fwww.freeflightsystems.com%2Fdocs%2FADS-B_For_Beginners.pdf&ei=zjeiT_DEE-So4gS8zoiJCQ&usg=AFQjCNE_v6ZfjgwkT5Ch08tkxvIfy0B0bA. [Geopend 9 maart 2012].

[13] db-interactivly, „NinevehGL Tutorials," db-interactivly, 2011. [Online]. Available: http://nineveh.gl/docs/tutorials/. [Geopend 19 maart 2012].

[14] Google, „Google Maps Terms section 10.1.3 (b)," Google, [Online]. Available: https://developers.google.com/maps/terms?hl=nl-NL#section_10_1_3. [Geopend 15 maart 2012].

[15] R. Krahl, „Geo::OSM::Tiles," 6 april 2010. [Online]. Available: http://geo-osm-tiles.berlios.de/. [Geopend 22 maart 2012].

[16] F. Schröder, „map2sqlite tool 1.0," 13 juli 2009. [Online]. Available: https://groups.google.com/forum/?fromgroups#!topic/route-me-map/k0RZ3BNv_Sg. [Geopend 22 maart 2012].

[17] Apple, „Cocoa - Mac OS X Technologiy Overview - Apple Developer," Apple, [Online]. Available: https://developer.apple.com/technologies/mac/cocoa.html. [Geopend 14 maart 2012].

[18] S. Holzner, Design Patterns: Simply, Sourcemaking.

[19] E. F. B. B. K. S. Eric Freeman, Head First Design Patterns, O'Reilly Media, 2004.

[20] Apple, „Cocoa Fundamentals Guide: Cocoa Objects," Apple, 13 december 2010. [Online]. Available: http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/CocoaFundamentals/CocoaObjects/CocoaObjects.html#//apple_ref/doc/uid/TP40002974-CH4-SW32.

[21] Tark, „Make an EAGLView transparent?," Stackoverflow, 23 februari 2012. [Online]. Available: http://stackoverflow.com/questions/9410421/make-an-eaglview-transparent. [Geopend 14 maart 2012].

[22] D. Bomfim, „All about OpenGL ES 2.x," db-interactivly, 30 januari 2011. [Online]. Available: http://db-in.com/blog/2011/01/all-about-opengl-es-2-x-part-13/. [Geopend 19 april 2012].

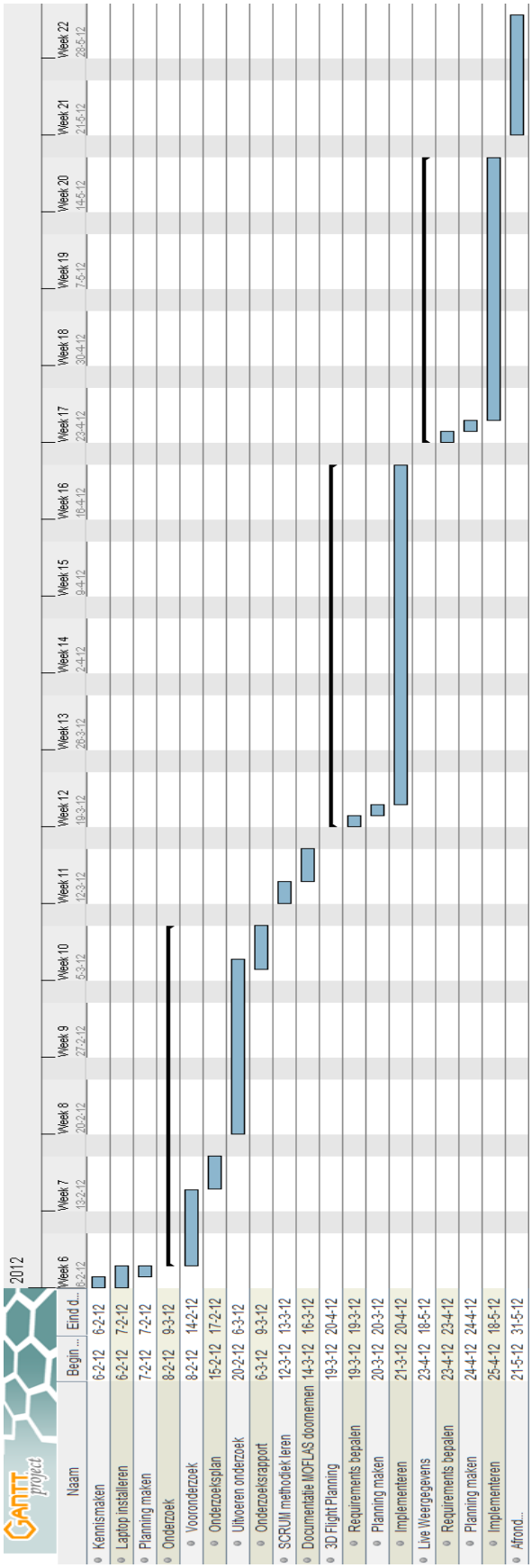
Lijst van figuren

Figuur 2.1 Organogram NCIM Groep 6
Figuur 4.1 Voorspelling marktaandeel Mobiele Platformen t/m 2015 11
Figuur 4.2 Tijdspad onderzoek 16
Figuur 4.3 Bijgewerkt tijdspad onderzoek 16
Figuur 4.4 MoSync vreemd teken bij strings in een loop code & resultaat 22
Figuur 4.5 MoSync strings in loop code & resultaat met voornaam lengte 23
Figuur 4.6 MoSync reset string waarde in loop code 23
Figuur 5.1 NinevehGL fast loading tegen original loading 33
Figuur 5.2 Reguliere notatie vermenigvuldiging methode 36
Figuur 5.3 Objective-C notatie vermenigvuldiging methode 36
Figuur 5.4 Singleton geïmplementeerd in C# 39
Figuur 5.5 Singleton geïmplementeerd in Objective-C [18] 40
Figuur 5.6 Object Pool Initialisatie 41
Figuur 5.7 Object ophalen uit de Object Pool 41
Figuur 5.8 Vrij object in de Object Pool zoeken 42
Figuur 5.9 Image object vanuit een view object 43
Figuur 5.10 3D kaart vanuit een 2D view 44
Figuur 5.11 3D Kaart vanuit losse tegels 44
Figuur 5.12 Geheugengebruik route-me op de iPad simulator 45
Figuur 5.13 Geheugengebruik route-me op de iPad 46
Figuur 5.14 Kaart tegels gaan goed op de simulator (links) maar fout op de iPad (rechts) 47
Figuur 5.15 Texture vanuit een bestand 50
Figuur 5.16 Texture vanuit een *UIImage* 50

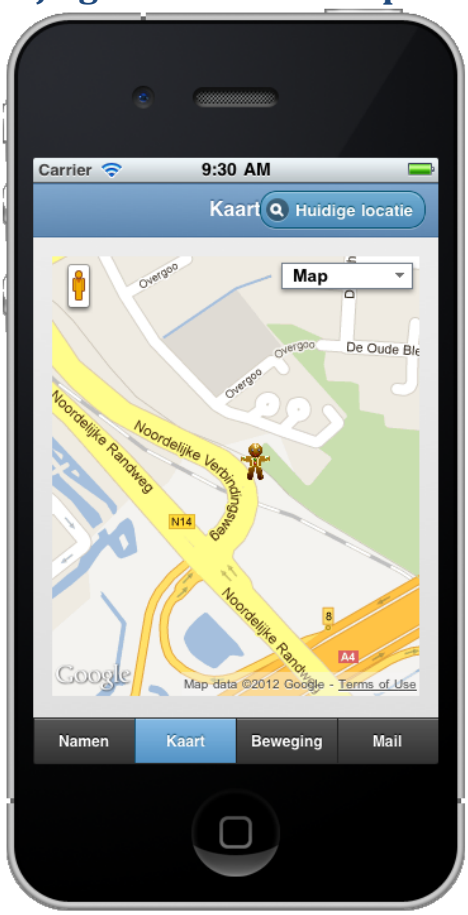
Lijst van tabellen

Tabel 3.1 Mijlpalen 9
Tabel 4.1 Overzicht voor -en nadelen van methoden voor platformonafhankelijke ontwikkeling 12

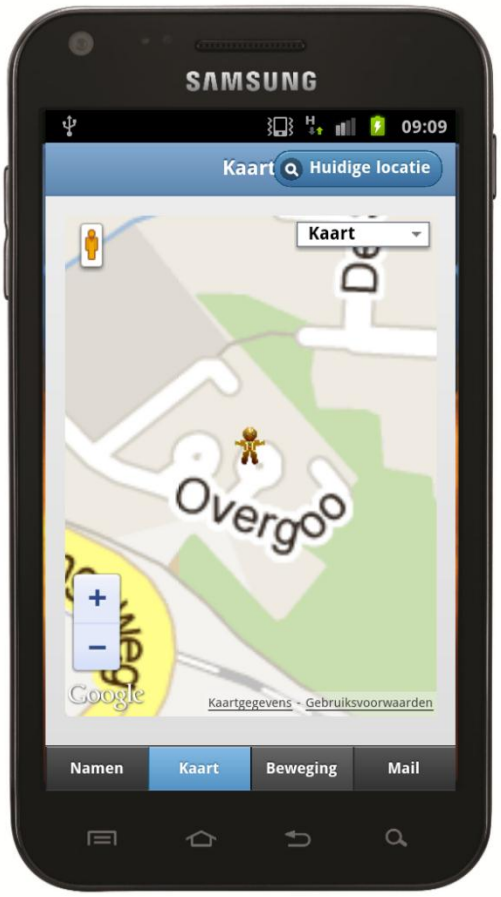
Bijlage A Globale planning



Bijlage B Reultaat experiment jQuery Mobile & Phonegap

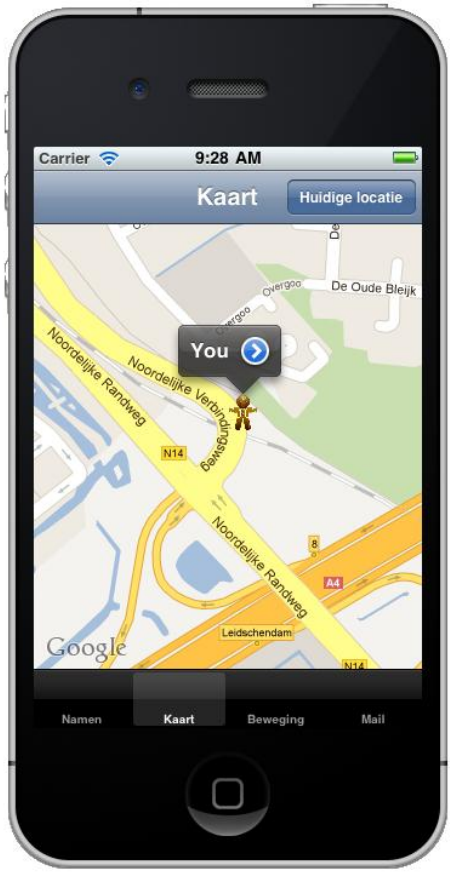


PhoneGap op iOS

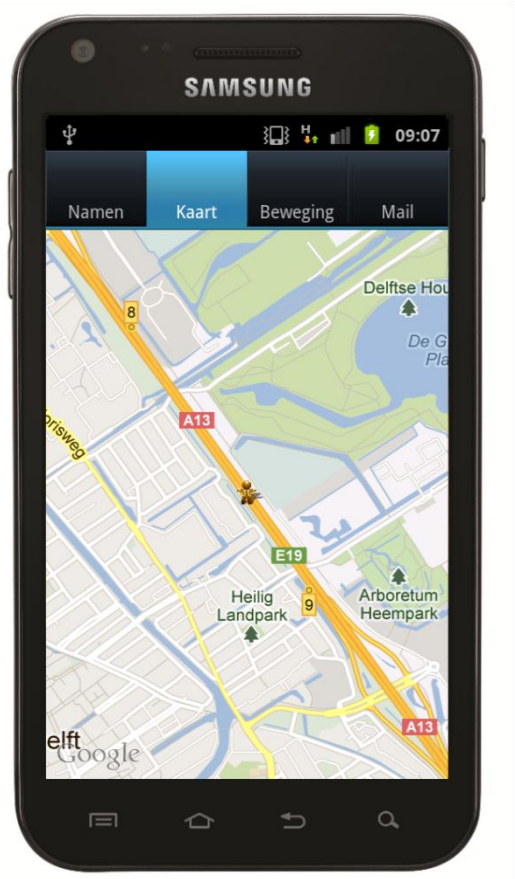


PhoneGap op Android

Bijlage C Resultaat experiment Appcelerator Titanium



Appcelerator Titanium op iOS

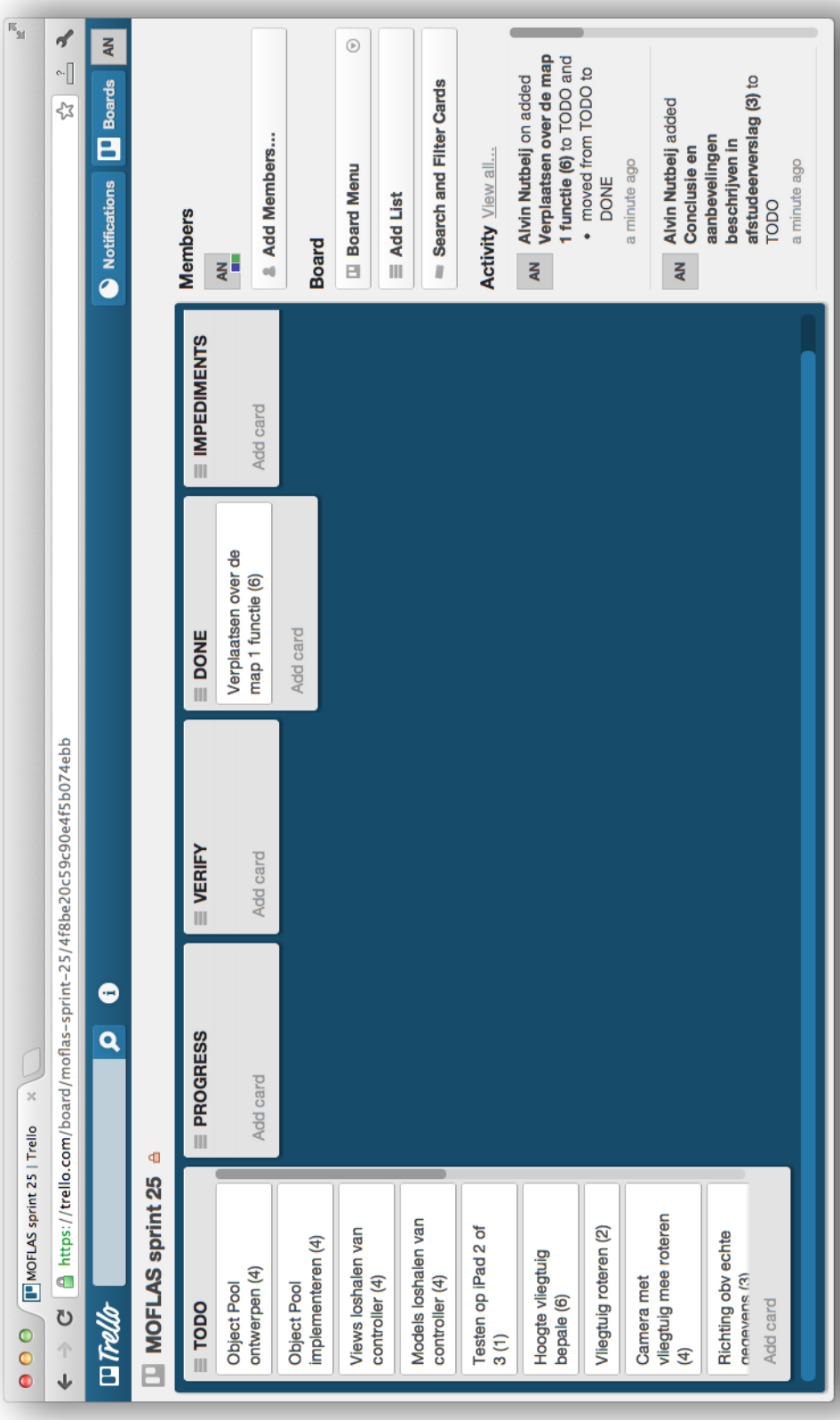


Appcelerator Titanium op Android

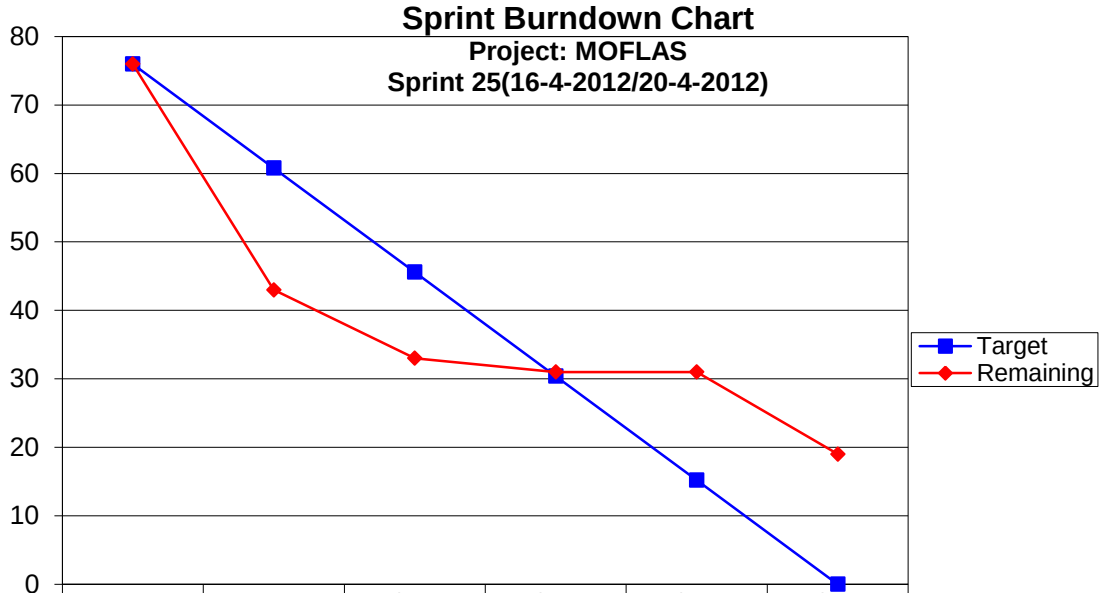
Bijlage D Product backlog

#	#	Type	Theme	As a	I want to	So that	Constraint / Note	Priority
1		0 Epic	Navigate	Pilot	view my route in 3D	I can see where I'm flying		Must
	1	1 Story	Navigate	Pilot	view my position on a map in a 3D enviroment	I can see my current location		Must
	2	2 Story	Navigate	Pilot	see my airplane move across the map	I can follow my flight route		Must
	3	3 Story	Navigate	Pilot	predetermine my route by inserting waypoints	I know what route to follow	Waypoints are added by inserting their name into a text box. When multiple waypoints with the same name are found an option box should appear	Must
	4	4 Story	Navigate	Pilot	see in what direction I should fly within the 3D enviroment	I know what heading to follow	The route should be visiualized by an arrow which the pilot should follow	Must
	5	5 Story	Navigate	Pilot	view my landing approach in a 3D enviroment	I can see exactly how I should approach the airstrip	Route should be displayed by hoops / squars through which the airplane must fly	Should
	6	6 Story	Navigate	Pilot	manage waypoints	I can alter or remove routes		Must
	7	7 Story	Navigate	Pilot	select previous routes	I don't have to create them each time I go flying		Must
2		0 Epic	Live weather	Pilot	receive live weather information	I know what I can expect		Must
	1	1 Story	Live weather	Pilot	see the weather conditions in a 3D enviroment	I can see the weather conditions around me		Must
	2	2 Story	Live weather	Pilot	see whether I should fly over, under or through the clouds	I won't make the wrongdecision		Should
	3	3 Story	Live weather	Pilot	see different type of clouds	I can distinguish thunder clouds from regular clouds	Thunder clouds should have a red color to indictate danger	Should

Bijlage E Scrumbord



Bijlage F Burndown chart



	Initial	After 1 day	After 2 days	After 3 days	After 4 days	After 5 days
Target	76	61	46	30	15	0
Remaining	76	43	33	31	31	19

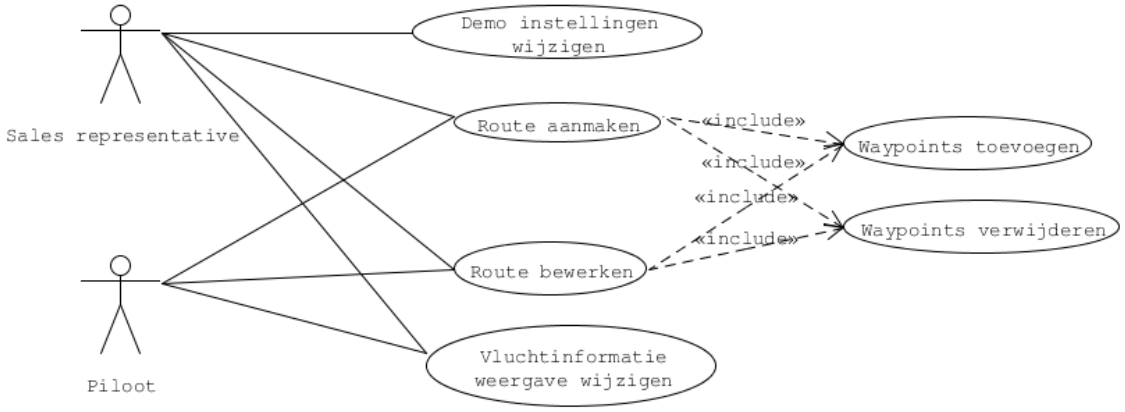
Bijlage G Kaart tegel loop versie 1

```
for(int i = topLeftX; i <= bottomRightX; i++){
    for(int j = topLeftY; j <= bottomRightY; j++){
        NCIMTile *tempTile = nil;
        NGLMesh *tempTileMesh = nil;
        NGLTexture *tempTileTexture = nil;
        NGLMaterial *tempTileMaterial = nil;

        tempTile = [NCIMTile tileWithX:i y:j andZoom:defaultZoom];

        tempTileMesh = [[NGLMesh alloc] initWithFile:@"tile_plane.obj"
settings:settings delegate:nil];
        tempTileMesh.tileInfo = tempTile;

        tempTileTexture = [[NGLTexture alloc]
init2DWithImage:tempTile.image];
        tempTileMaterial = [NGLMaterial material];
        [tempTileMaterial setDiffuseMap:tempTileTexture];
        [tempTileMesh setMaterial:tempTileMaterial];

        [tempTileMesh compileCoreMesh];

        tempTileMesh.x = i;
        tempTileMesh.z = j;

        [_tileArray addObject:tempTileMesh];
        [_camera addMesh:tempTileMesh];

        [tempTileMesh release];

        tempTileMesh = nil;
        tempTile = nil;
        tempTileTexture = nil;
        tempTileMaterial = nil;
    }
}
```

Bijlage H Kaart tegel loop versie 2

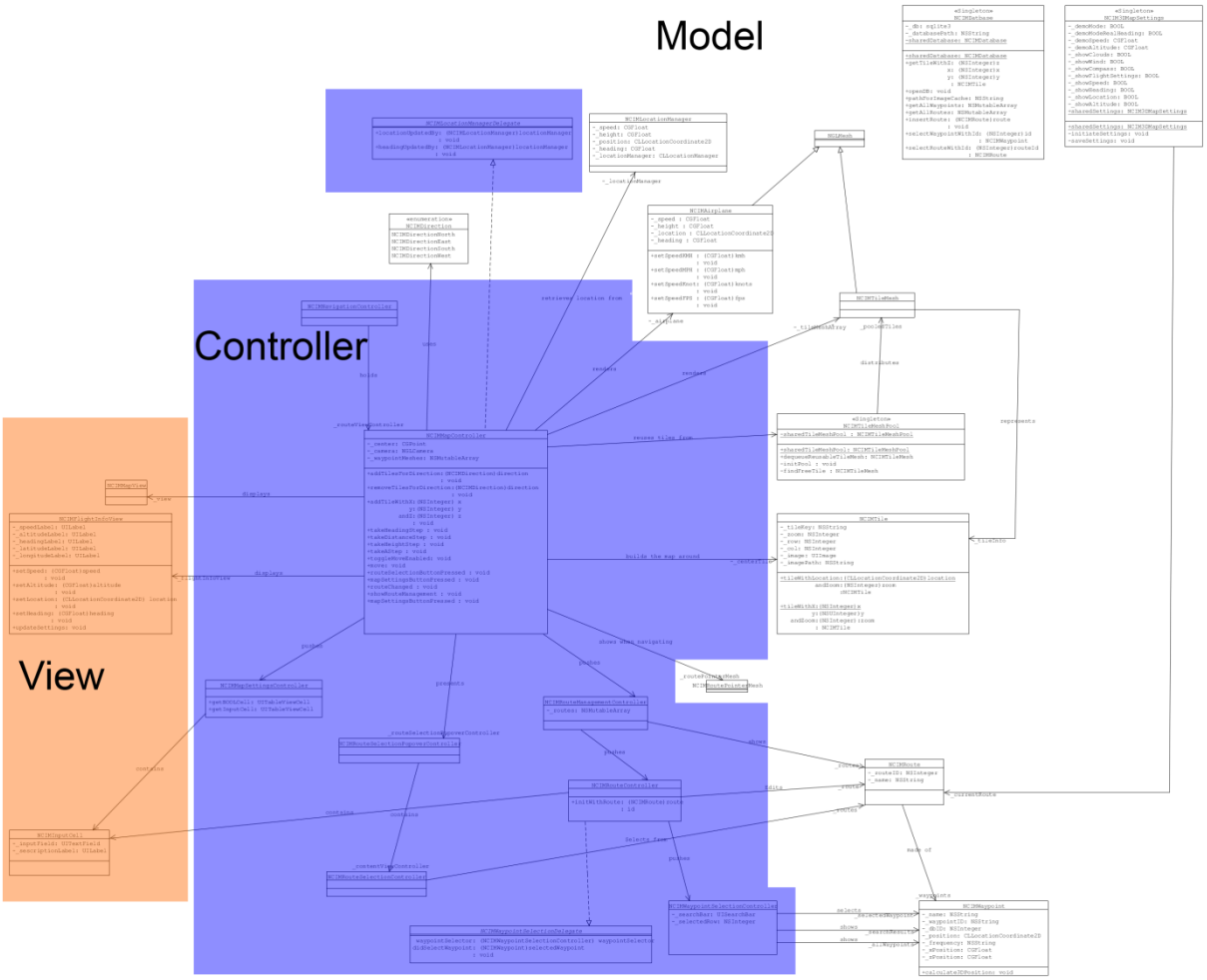
```
for(int i = topLeftX; i <= bottomRightX; i++){
    for(int j = topLeftY; j <= bottomRightY; j++){
         NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];

        //fetch the tile info (autoreleased object)
        tempTile = [NCIMTile tileWithX:i y:j andZoom:_centerTile.zoom];

        //create a mesh
        tempTileMesh = [[NGLMesh alloc] initWithFile:@"tile_plane.obj"
settings:settings delegate:nil];

        //create a material (autoreleased object)
        tempTileMaterial = [NGLMaterial material];

        //create a texture from the image in the tile info
        tempTileTexture = [[NGLTexture alloc]
init2DWithImage:tempTile.image];

        //make sure the texture fills out the mesh instead of repeating
        [tempTileTexture setRepeat:NGLTextureRepeatNone];

        //set the texture as diffuse map for the material
        [tempTileMaterial setDiffuseMap:tempTileTexture];

        //add the material to the mesh
        [tempTileMesh setMaterial:tempTileMaterial];

        //compile the mesh
        [tempTileMesh compileCoreMesh];

        //Set the position of the tile
        tempTileMesh.x = i;
        tempTileMesh.z = j;

        //Add to the array and to the camera
        [_tileArray addObject:tempTileMesh];
        [_camera addMesh:tempTileMesh];

        //Cleanup
        [tempTileMesh release];
        tempTileMesh = nil;
        [tempTileTexture release];
        tempTileTexture = nil;
        tempTileMaterial = nil;
        tempTile = nil;

        //Making sure the autoreleased objects are released
        [pool drain];
    }
}
```

Bijlage I Use Case Diagram



Bijlage J Klassendiagram

Het klassendiagram is te groot om op een A4 te passen, daarom is op deze pagina slechts een voorbeeld opgenomen. De volgende pagina kan opengeklapt worden zodat het diagram beter leesbaar is.



Model

Controller

View