Polar Palooza

Hoe een hongerige ijsbeer de wereld zal veroveren

Maflaroon

Matthias Ghering - s4395727
Flip van Spaendonck - s4343123
Lars Kuijpers - s4356314
Toon Lenaerts - s4321219

27 juni 2014

Inhoudsopgave

Hoofdstuk 1

Voorwoord

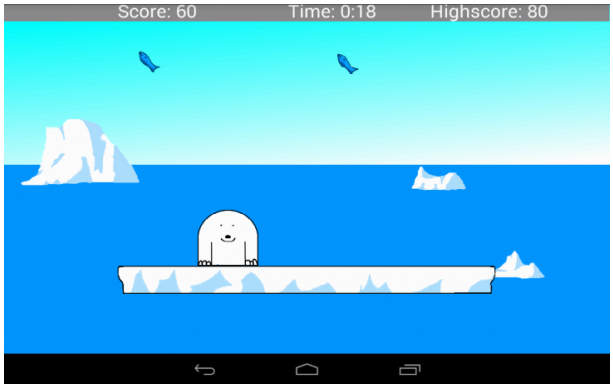
Voor het Research & Development Project moesten wij met een groep van 4 personen een app maken voor Android telefoons. Het doel hiervan was om in groepsverband samen van de grond af een software ontwikkelings project op te zetten en onderzoek te doen naar usability. Dit document zal alleen maar over het eerste deel gaan, de ontwikkeling van onze app. We zullen zowel het eindproduct als de weg er naartoe bespreken.

We beginnen met een beschrijving van onze app, Polar Palooza. Deze beschrijving bestaat uit een korte inleiding over wat de app is, gevolgd door een verantwoording over waarom ons spel uniek op de markt is en tot slot een gedetailleerde specificatie over wat onze app allemaal kan. Daarna volgt een hoofdstuk over het ontwerp, waarin we ingaan op de globale onderdelen van de app en hoe ze met elkaar samenwerken, hoe we de belangrijkste van deze onderdelen hebben onderverdeeld in Java-klassen en welke attributen en methoden deze klassen hebben, en een korte verantwoording waarin we enkele van onze ontwerpkeuzes uitlichten. We eindigen met een reflectie over hoe wij dit project en het maken van onze app hebben ervaren, met zowel goede punten als slechte punten, en wat we hiervan hebben geleerd voor de toekomst.

Hoofdstuk 2

Beschrijving

2.1 Inleiding



Onze app, Polar Palooza, is een game gemaakt voor Android. Het is een simpel spelletje dat makkelijk te begrijpen is en iedereen kan spelen. Het idee is als volgt: je speelt als een hongerige ijsbeer op een ijsschots op een onbekende locatie. Er springen vissen uit het water en het doel van het spel is om zoveel mogelijk van deze vissen te vangen voordat ze terug in het water vallen. De speler is af als de tijd om is. Naast dat de tijd natuurlijk wegtikt kan de speler extra tijd krijgen door vissen te vangen en verliest deze tijd door vissen van het scherm te laten vallen zonder ze te vangen.

2.2 Productverantwoording

Maar wat voegt Polar Palooza nou eigenlijk toe? Er zijn waarschijnlijk honderden spelletjes in de playstore waar je iets moet vangen, waarom is die van ons zo uniek? Hoewel het principe van al deze spellen hetzelfde is, zoals hierboven beschreven, kan de uitvoering nogal verschillen.

Het eerste essentiële verschil in de uitvoering is de besturing. De speler moet op de een of andere manier het karakter aan de onderkant van het scherm naar links en naar rechts kunnen bewegen om de vallende dingen te vangen. Deze besturing wordt over het algemeen op drie verschillende manieren gerealiseerd:

1. De speler klikt links en rechts op het touchscreen om het karakter die kant op te laten bewegen (zie bijvoorbeeld "Papi Catch")
2. De speler sleept het karakter naar een plek om het daar naartoe te bewegen (zie bijvoorbeeld "Catch the Eggs")
3. De speler kantelt de telefoon naar links en rechts om het karakter die kant op te laten bewegen (zie bijvoorbeeld "Andy's Apples")

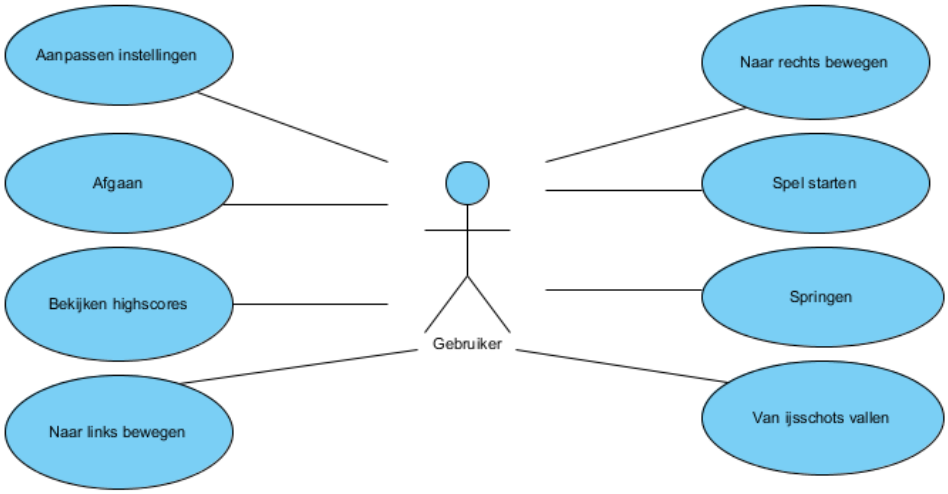
Alle apps die wij hebben gevonden kiezen echter maar één van deze drie mogelijkheden uit, terwijl onze app spelers de mogelijkheid geeft om te kiezen welk besturingssysteem, de eerste of de derde, ze willen gebruiken.

Een ander aspect van het originele idee wat wij anders benaderen dan andere apps is hoe Game Over gaan wordt geregeld. Bij het overgrote deel van de andere apps is het idee om zoveel mogelijk vallende dingen te vangen zonder ze te missen (wat inhoudt dat ze de onderkant van het scherm raken zonder door de speler geraakt te zijn). De speler verliest een leven als deze een van de vallende objecten mist en als de speler alle levens verloren is, is het Game Over. Andere spellen, zoals bijvoorbeeld "Catch the Drops", hebben een level-gebaseerd systeem, die vaak op tijd werkt. Hierbij is de bedoeling om in elk level een minimale hoeveelheid score te halen binnen een bepaalde tijd. Elk level wordt dit moeilijker doordat de speler minder tijd krijgt óf doordat meer score nodig is. Hoe wij het regelen in Polar Palooza, met tijd als een variabel iets dat omhoog of omlaag kan, is ook iets dat we niet bij andere apps tegen zijn gekomen.

2.3 Specificaties

2.3.1 Functionele eigenschappen

Onze functionele eigenschappen staan allemaal in het volgende schema:



We beginnen met een korte beschrijving van al deze use cases en werken daarna de drie interessantste (aanpassen instellingen, bekijken highscores en spel starten) uit als volledige use cases.

- **Aanpassen instellingen -** De gebruiker verandert de instellingen van de app.
- **Afgaan -** De gebruiker verliest het spel.
- **Bekijken highscores -** De gebruiker bekijkt zijn highscores.
- **Naar links bewegen -** De gebruiker beweegt de ijsbeer naar links
- **Naar rechts bewegen -** De gebruiker beweegt de ijsbeer naar rechts.
- **Spel starten -** De gebruiker start het spel.
- **Springen -** De gebruiker springt met de ijsbeer.
- **Van ijsschots vallen -** De ijsbeer valt van de ijsschots af.

Use Case	Aanpassen instellingen
Description	De gebruiker verandert de instellingen van de app.
Actor	Gebruiker
Basic course of events	1. De gebruiker drukt op de knop met "Settings" in het hoofdmenu. 2. De app laat een menu zien met twee schuifbalken, naast de eerste balk staat "Sound" en naast de tweede balk staat "Music". Onder de tweede balk staat een checkbox met de tekst "Gyro" ervoor. 3. De gebruiker drukt op de terug-knop van zijn android apparaat. 4. De app laat het hoofdmenu zien.
Alternate paths	**Sound aanpassen** 3a. De gebruiker drukt op een plaats op de balk met de tekst "Sound" ervoor. 4. De app laat opnieuw het settingsmenu zien maar met de aanpassingen van de gebruiker verwerkt in de schuifbalk. Ga verder bij 3. **Besturing aanpassen** 3b. De gebruiker drukt op de checkbox onderaan. 4b. De app laat opnieuw het settingsmenu zien maar dit keer is de kleur van de checkbox veranderd van rood naar groen of andersom. Ga verder bij 3.
Preconditions	De speler bevindt zich op het moment in het hoofdmenu.
Postconditions	De app heeft de aanpassingen in de instellingen opgeslagen.

Use Case	Bekijken highscores
Description	De gebruiker bekijkt zijn highscores.
Actor	Gebruiker
Basic course of events	1. De gebruiker drukt op de knop met "Highscores" in het hoofdmenu. 2. De app laat een lijst zien met de top 10 highscores van de speler met een knop met "Clear highscore".
Alternate paths	**Het verwijderen van de highscores** 3a. De speler drukt op de knop "Clear highscore". 4. De app laat opnieuw een lijst zien met de top 10 highscores maar dit keer staat alles op 0. Onder de lijst laat hij een knop zien met "Clear highscore".
Preconditions	De speler bevindt zich op het moment in het hoofdmenu.
Postconditions	De top 10 highscores van de gebruikers zijn zichtbaar op het scherm. Als de gebruiker het Alternate path "Het verwijderen van de highscores" heeft gekozen zijn de highscores definitief verwijderd.

Use Case	Een spel starten
Discription	De gebruiker start het spel.
Actor	Gebruiker
Basic course of events	1. De gebruiker drukt op de knop met “Play” in het hoofdmenu 2. De app geeft de uitleg van het spel, met op de achtergrond de ijsbeer op een ijsschots en een aantal decorative ijsschotsen daar omheen. 3. De Gebruiker drukt op het scherm. 4. De app geeft nu alleen de ijsbeer op een ijsschots en de blauwe achtergrond.
Alternate paths	
Preconditions	De gebruiker bevindt zich in het hoofdmenu.
Postconditions	Het spel is nu gestart.

2.3.2 Niet-functionele eigenschappen

Bij het ontwikkelen van de app is er ook rekening gehouden met enkele niet-functionele eigenschappen. De volgende niet-functionele eigenschappen vonden wij belangrijk voor het ontwikkelen van onze app:

- **Userfriendly:** Polar Palooza combineert een duidelijk en makkelijk te begrijpen design met intuïtieve besturing.
  - **Duidelijk:** De navigatie en functies van knoppen wordt goed aangegeven door middel van tekst en/of plaatjes.
  - **Intuïtief:** Acties moeten niet te ingewikkeld zijn en logisch zijn om op de gekozen manier uit te voeren. Bijvoorbeeld om naar links te bewegen moet de speler links klikken en niet ergens anders op het scherm. Alles wat niet meteen duidelijk is moet worden uitgelegd aan de speler.
- **De Flappy Bird-factor:** De "Flappy Bird-factor" houdt in dat een app entertainend, verslavend, fast-paced en uitdagend tot frustrerend is. Door de app de "Flappy Bird-factor" te geven, heeft de app meer potentieel om een rage te worden. We zien deze factor verschijnen in veel andere populaire spellen, zoals bijvoorbeeld Tiny Wings, Doodle Jump en Fruit Ninja.
  - **Entertainend:** Het spel moet leuk zijn om te spelen.
  - **Verslavend:** De speler moet na een ronde het gevoel hebben dat hij snel nog een ronde moet spelen. Dit wordt gedaan door de rondes relatief kort te houden. Het kort houden van de rondes heeft als gevolg dat de speler niet te snel verveeld raakt met het spel en zorgt dat de speler kan spelen, zelfs als deze maar kort de tijd heeft.
  - **Fast-paced:** Fast-paced houdt in dat het spel een snel veranderend klimaat heeft. In Polar Palooza is dat duidelijk te zien aan de toename in het aantal vissen, wat het spel steeds moeilijker maakt.
  - **Uitdagend:** Het moet moeilijk zijn om hoge scores te halen in het spel. Dit wordt bereikt door de hoeveelheid vissen, naarmate de speler een hogere score haalt, toe te laten nemen. Hierdoor zorgen wij ervoor dat als de speler een bepaald speelniveau heeft bereikt, en daarmee een bepaalde score, hij vervolgens een hoger speelniveau moet bereiken om een hogere score te kunnen halen.

Hoofdstuk 3

Ontwerp

3.1 Globaal Ontwerp

Onze app kan onderverdeeld worden in verschillende componenten, die zelf ook weer onderverdeeld kunnen worden.

- **Engine**: Het systeem dat de IO regelt en het beeld dat de gebruiker te zien krijgt construeert.
- **Spel**: Het systeem dat de toestand van de app bijhoudt, de input verwerkt en regelt wat de gebruiker te zien krijgt. Kan onderverdeeld worden in:
  - **Hoofdmenu**: Het scherm waarop de gebruiker kan kiezen welke andere component van het spel deze te zien wil krijgen.
  - **Opties**: Het scherm waarop de gebruiker de instellingen van de app kan veranderen.
  - **Highscores**: Het scherm waar de hoogste behaalde scores in het spel te zien zijn.
  - **Spelscherm**: Het daadwerkelijke spel waar de gebruiker de ijsbeer controleert, visjes probeert te vangen en score haalt.

De eerste opdeling van componenten, in de engine en het spel PolarPalooza zelf, kunnen we maken omdat het spel niet afhankelijk is van de engine, maar gebruik maakt van interfaces om correct te werken. Als er een andere engine in de app gezet zou worden, werkt het spel nog steeds, zolang deze andere engine correct gebruik maakt van de interfaces. De twee componenten werken samen in een Model - View relatie; de engine vervult de rol van View en het spel de rol van Model. De engine krijgt input van de gebruiker, geeft deze door aan het spel, vervolgens verwerkt het spel de input en stuurt data terug naar de engine, waarna de engine deze data tot een beeld voor de gebruiker verwerkt.

De andere opdeling, die van het spel, is zo gedaan omdat dit de 4 schermen zijn die de gebruiker te zien krijgt tijdens het gebruik van de app. De componenten Opties, Highscores en Spelscherm hangen alledrie samen met het Hoofdmenu omdat deze drie schermen bereikt worden vanuit het hoofdmenu. Daarnaast hangen Opties en Highscores beiden samen met Spelscherm. De Opties omdat er in het optiescherm instellingen ingesteld kunnen worden die vervolgens in het Spelscherm te merken zijn. De Highscores omdat de scores die behaald worden in het Spelscherm hier te zien zijn.

De opsplitsing in engine en app komt overeen met de manier waarop we de app hebben gebouwd. Onze engine is namelijk gebaseerd op een engine uit "Beginning Android Games" door Mario Zechner en Robert Green, wij hebben het spel vervolgens ontworpen om te werken met de engine. De opsplitsing van het spel in vier delen komt overeen met de vier schermen die in de app te zien zijn, het komt dus ook overeen met de manier waarop we de app hebben gebouwd.

3.2 Detailontwerp

- **Engine:** de engine gebruikt een lijst met interfaces, en een lijst met klassen die deze interfaces implementeren. De volgende interfaces worden geïmplementeert:
  - **Audio:** de audioklasse implementeert twee methoden:
    - createMusic, met deze methode wordt muziek in het spel geladen.
    - createSound, met deze methode worden korte geluiden in het spel geladen.
  - **FileIO:** de IOklasse implementeert de volgende methoden:
    - readFile, met deze methode worden bestanden uit het vaste geheugen gelezen.
    - writeFile, met deze methode kan (in) een bestand op het vaste geheugen geschreven worden.
    - readAsset, met deze methode worden assets uit het vaste geheugen gelezen.
    - getSharedPreferences, met deze methode worden de shared preferences opgevraagd, die gebruikt kunnen worden om dingen zoals highscores en instellingen in op te slaan.

- **Graphics:** de graphicsklasse implementeert de volgende methoden:
  - newImage, met deze methode wordt een afbeelding uit het vaste geheugen gelezen.
  - clearScreen, met deze methode wordt het scherm volledig leeggemaakt.
  - drawLine & drawRect & drawImage, met deze methoden kan respectievelijk een lijn, rechthoek of plaatje op het scherm getekend worden.
  - getWidth & getHeight, met deze methoden kan de hoogte en breedte van het scherm worden opgevraagd.
  - drawARGB, met deze methode kan het scherm met een mogelijk transparante kleur gevuld worden.
- **Gyro:** de gyroklasse is een erg ingewikkelde klasse. Dit wordt veroorzaakt doordat er gebruikt wordt gemaakt van drie verschillende sensoren om de gyropositie te bepalen.
- **Image:** de imageklasse implementeert de volgende methoden:
  - getWidth & getHeight, met deze methoden kan de hoogte en breedte van het plaatje worden opgevraagd.
  - getFormat, met deze methode kan het opslagformaat van hey plaatje worden opgevraagd.
  - dispose, met deze methode wordt het plaatje uit het RAM-geheugen gegooid, zodat er meer ruimte vrijkomt.
- **Input:** met de inputklasse kan informatie worden opgevraagd van de touchsensor, deze informatie wordt opgeslagen als een "TouchEvent".
  - getTouchX & getTouchY, met deze methode kan de locatie van het "TouchEvent" worden opgevraagd.
  - getTouchEvents, met deze methode kunnen "TouchEvents" worden opgevraagd.
- **Music:** de muziekklasse wordt gebruikt om muziek af te spelen en op te slaan in het geheugen. De muziekklasse implementeert een grote hoeveelheid methoden, waarvan de volgende interessant genoeg zijn om te bespreken:
  - play, pause & stop, met deze methoden kan de muziek respectievelijk afgespeeld, gepauzeerd en gestopt worden.
  - setVolume, met deze methode kan het volume van de muziek aangepast worden.
  - dispose, met deze methode wordt de muziek uit het RAM-geheugen gegooid, zodat er meer ruimte vrijkomt.

- **Screen:** de screenklasse implementeert de volgende methoden:
  - update, deze methode wordt gebruikt om het spel te updaten.
  - paint, met deze methode wordt op het scherm getekend.
  - pause, met deze methode wordt het spel gepauzeerd.
  - resume, met deze methode wordt het spel hervat.
  - dispose, met deze functie wordt dit 'Screen' en alle onnodige data uit het RAM-geheugen gehaald, zodat er meer ruimte vrijkomt.
  - backButton, deze methode wordt gebruikt als op de terug-knop van de telefoon wordt gedrukt.
- **Sound:** 'Sound' wordt gebruikt om korte geluidjes af te spelen. De klasse implementeert twee methoden:
  - play, hiermee wordt het geluid afgespeeld.
  - dispose, hiermee wordt het geluid uit het RAM-geheugen gegooid, zodat er meer ruimte vrijkomt.
- **Game:** vanuit de gameklasse wordt het spel uitgevoerd. Dit gebeurt door middel van de volgende methoden:
  - getGyro, met deze methode kan het spel gebruik maken van de gyro sensor.
  - getAudio, met deze methode kan het spel gebruik maken van geluid.
  - getInput, met deze methode kan het spel gebruik maken van inputs. In het geval van Android is dit het touchscreen.
  - getFileIO, met deze methode kan het spel gebruik maken van de IOklasse en dus bestanden in het vaste geheugen van de telefoon gebruiken.
  - getGraphics, met deze methode kan het spel gebruik maken van de graphicsklasse en dus dingen op het scherm weergeven.
  - setScreen, met deze methode kan het spel van 'Screen' wisselen.
  - getCurrentScreen, met deze methode kan het spel de 'Screen' opvragen die op dat moment wordt getoond.
  - getInitScreen, met deze methode wordt aangegeven op welk 'Screen' de app moet starten.
  - updateHighscores, met deze methode kan het spel de highscores, die in de shared preferences staan, updaten.
  - getHighscores, met deze methode kan het spel de highscores vanuit de sharedpreferences opvragen.
  - clearHighscores, met deze methode kan het spel alle highscores resetten naar 0.

- **Game:** Het spel bevat een paar klassen die erg interesant zijn en daarom hier worden besproken:
  - **Assets:** hierin worden alle afbeeldingen, geluiden en muziekstukken opgeslagen.
  - **Settings:** hierin worden alle instellingen opgeslagen.
  - **Obj:** alle objecten in het spel (vissen en de ijsbeer), hebben een aantal overeenkomende attributen en methoden:
    * x & y, deze attributen houden de 2-dimensionale positie van het object vast.
    * update, met deze methode wordt het object geüpdate, hierdoor wordt zijn nieuwe positie berekend en andere belangrijke dingen.
  - **Screen:** het spel gebruikt een aantal verschillende schermen, die het 'Screen'-interface implementeren:
    * GameScreen, waarin het spel zelf zich afspeelt.
    * LoadingScreen, waarin alle instellingen en assets in de bijbehorende klassen geladen worden.
    * MainMenuScreen, waarin het hoofdmenu zit.
    * ScoreScreen, waarin de speler de highscores kan zien en verwijderen.
    * SettingScreen, waarin de speler de instellingen kan zien en aanpassen.
  - **TimeHandler:** de timehandler wordt gebruikt om tijd (tot op de centiseconde precies) bij te houden in het spel. De timehandler gebruikt de volgende methoden:
    * incr & decr, met deze methoden kan er tijd toegevoegd of afgetrokken worden van de teller.
    * reset, met deze methode wordt de teller gereset.
    * getTime & toString, met deze methoden kan de tijd opgevraagd worden als een getal of als een visuele weergave in tekst.

3.3 Ontwerpverantwoording

Dat ons ontwerp een goed ontwerp is, komt niet door één specifieke keuze, maar doordat over elke beslissing goed is nagedacht. Natuurlijk zijn er enkele, specifieke keuzes die ons uniek maken en afzetten van andere ontwerpen, zoals de keuzes om verschillende control-schema's toe te staan en het gebruik van een custom engine, die we hier nader zullen toelichten.

3.3.1 De engine

We hebben gekozen voor het gebruik van een engine omdat we vonden dat de standaard structuur waarin je een app ontwikkeld niet gericht was op games, maar op applicaties zoals bijvoorbeeld de NS-reisplanner en socialmedia-apps.

Het gebruik van de engine bleek een verstandige beslissing te zijn. Andere groepen die ook een game maakten, maar geen engine gebruikten, schenen veel problemen te hebben met Android-specifieke constructies, die wij met onze engine omzeild hebben.

De engine heeft ons ook nog andere voordelen opgeleverd. Doordat de verbinding tussen de engine en de game volledig via interfaces wordt gedaan, kost het weinig moeite om het spel naar een ander platform om te zetten, zoals bijvoorbeeld van Android naar PC. Een ander voordeel vat het gebruik van een engine is dat engines van nature herbruikbaar zijn. Bij volgende games kunnen we, en zullen we, dus ook onze engine opnieuw gebruiken.

3.3.2 Besturing

Wat onze game afzet van gelijksoortige games is de mogelijkheid voor de speler om te kiezen welk besturingssysteem deze wilt gebruiken. Onze game staat de speler namelijk toe om te kiezen welke besturing deze wilt gebruiken, gyro of touchcontrol. Hierdoor kunnen spelers de besturing gebruiken die zij het fijnste vinden, en niet gedwongen zijn om één bepaalde vorm van besturing te moeten gebruiken. Als we dit niet zouden hebben gedaan zouden er spelers kunnen zijn die het idee van onze app heel leuk vinden, maar de besturing graag anders hadden gewild, en daarom een andere app gaan zoeken om hieraan te voldoen. Door ze hierin de keuze te geven zullen wij deze spelers niet kwijtraken aan concurrerende apps.

Hoofdstuk 4

Reflectie

Over het algemeen zijn we erg positief over zowel ons eindproduct als hoe het project als een geheel is verlopen. We vinden dat we een goed werkend eindproduct hebben neergezet, dat ook nog leuk is om te gebruiken. Wel vinden we het jammer dat er nog wat kleine fouten in zitten, zoals bijvoorbeeld het niet op kunnen slaan van de opties, en minder goed werkende dingen, zoals het springen.

Wij hebben ook veel geleerd van dit project. Hoewel we al eerder in groepsverband hebben gewerkt was het een interessante ervaring om dit ook echt met een softwareproject te doen van de grond af. Ook met Android specifiek hebben we nu ervaring opgedaan, wat we eerst geen van allen hadden. Met het maken van het usability onderzoek hebben we geleerd om kritisch te kijken naar zowel andermans als onze eigen producten. Ook over presenteren hebben we veel geleerd van de feedback die we hebben gekregen, wat ons zal helpen in toekomstige projecten.

Voor ons gevoel is er niets overdreven slecht gegaan, hoogstens hadden wat moeilijkheden met enkele onderdelen zoals de implementatie van de gyro. Dat het over het algemeen zo goed ging kwam mede doordat we goed hebben nagedacht over alle beslissingen die we hebben gemaakt. Ook heeft het gebruik van de engine, waar we in een vroeg stadium van het project al voor hadden gekozen, veel stress en problemen voorkomen, die anders misschien wel waren opgetreden tijdens het ontwikkelen van de app.

Bij toekomstige projecten zullen we zeker gebruik maken van de dingen die we bij dit project geleerd hebben. Zoals gezegd heeft de feedback over de presentaties ons erg geholpen, maar ook in het algemeen zijn we nu beter in staat om samen te werken met anderen. Bijvoorbeeld het maken van een goede planning en het behouden van goede communicatie in het team zijn dingen waar we zeker ook rekening mee zullen houden in toekomstige projecten, omdat het hier ons zo goed geholpen heeft.