MKULTRA

Stop TeamRocket from taking over the world!

Door: Charlie Gerhardus en Sanne Boumans
Studentnummers: 3050009 en 3031926
Datum: 29-06-2012

Voorwoord

Op 17 April 2012 ging het vak research en Development van start. Het doel van deze cursus was om te leren over hoe een ontwikkelingstraject van een bepaald product verloopt. Om dit het beste te leren hebben wij een app van begin af aan moeten ontwikkelen. Hierbij werden alle stappen van de ontwerpfase en de ontwikkelfase tot aan uiteindelijk de laatste test fase door ons zelf uitgevoerd.
Dit document is er dan ook voor bedoeld om het proces kort te belichten en de uiteindelijke resultaten te tonen. Om dit te doen zullen wij ons focussen op vier globale onderwerpen die elk op hun beurt toegelicht worden.
Ten eerste volgt er een beschrijving. Hierin wordt kort vertelt over wat de door ons gemaakte app precies is en waarom we voor deze app hebben gekozen. Tevens worden de specificaties van de app kort belicht.
Vervolgens komt het ontwerp aan bod. In het ontwerp wordt er eerst kort vertelt over het globale ontwerp van de app, om vervolgens daar in bepaalde opzichten wat dieper op in te gaan. Ook wordt er nog even kort stil gestaan bij een aantal problemen waar wij tegen aan liepen en bij welke ontwerp beslissingen wij toen hebben genomen.
Het derde globale onderwerp waar we naar gaan kijken is de evaluatie. In deze evaluatie wordt verteld over de eindtest die gedaan is om de uiteindelijke app te testen en wat daar de uitslagen van waren.
Tenslotte zullen wij vervolgens een korte reflectie van het gehele leerproces geven die zowel groepsgewijs als individueel gedaan is.

Inhoudsopgave

1. Beschrijving

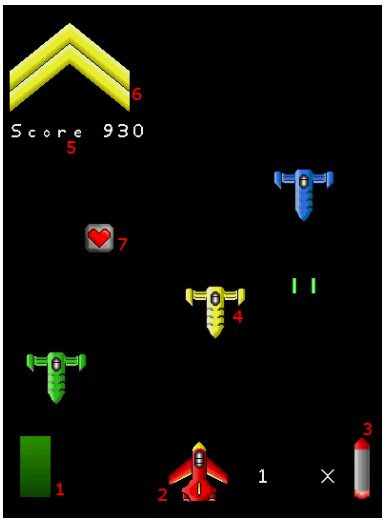
1.1 Inleiding

Spelbeschrijving

De MKULTRA app is een schietspel waarin de gebruiker een vliegtuig bestuurt met als doel zo lang mogelijk in leven te blijven en zo veel mogelijk punten te verzamelen. De gebruiker vliegt in zijn vliegtuig door de ruimte en komt voortdurend een stroom van vijanden tegen. Het is dan ook de bedoeling dat deze vijanden uitgeschakeld worden door middel van de schietfunctie die het vliegtuig van de gebruiker heeft. Voor elke vijand die de gebruiker uitschakelt verdient hij punten die nodig zijn om naar een volgende (moeilijker) rang over te gaan. In deze nieuwere rangen zullen er verschillende, betere, wapens verkrijgbaar zijn en zullen er sterkere vijanden te voorschijn komen. Om de gebruiker enigszins bij te staan zullen er verschillende 'power- ups' te verkrijgen zijn in het spel (denk hierbij bijvoorbeeld aan raketten of extra levens).

Scherm uitleg

Hieronder volgt een screenshot van het spel in actie met daarbij een korte uitleg over de betekenis van de verschillende dingen die op het scherm te zien zijn.



1. De levens balk. Deze toont hoeveel leven de speler over heeft. Als de balk leeg is is de speler game-over.
2. De speler. Dit vliegtuig wordt door de gebruiker bestuurd en moet heel zien te blijven.
3. De doelzoekende raket. Hier staat hoeveel doelzoekende rakketten de gebruiker heeft (in dit geval 1x). Voor meer informatie zie het kopje 'wapens'
4. De vijand. Dit is een van de mogelijke vijanden die op de speler af komen. Voor meer informatie zie het kopje 'vijanden'.
5. De score. Hier ziet men de score, die verkregen is door vijanden te vernietigen staan.
6. De rang. Hier wordt aangegeven in welke rang de speler zich bevindt. In dit geval is dat rang 2 (te zien door de twee strepen).
7. Een power-up. Deze dozen vliegen regelmatig door het spel. Voor meer informatie zie het kopje 'power-ups'.

Besturing

Om het vliegtuig te verplaatsen moet de gebruiker het vliegtuig over het scherm slepen. Om een aantal seconden durend salvo van kogels af te vuren moet de gebruiker klikken. En om de doelzoekende raket af te vuren klikt de gebruiker op de doelzoekende raket die standaard rechts onderin het scherm staat.

Wapens

In het spel zijn er verschillende wapens te vinden en te gebruiken, te weten:


Wapen 1. Stelt de gebruiker in staat om een salvo van 1 kogel af te vuren.


Wapen 2. Stelt de gebruiker in staat om een salvo van 2 kogels af te vuren.


Wapen 3. Stelt de gebruiker in staat om een salvo van 3 kogels af te vuren.


Wapen 4. Stelt de gebruiker in staat om een salvo van 4 kogels af te vuren.

Doelzoekende raket. Stelt de gebruiker in staat om een doelzoekende raket af te vuren die uit zichzelf op de vijand die de meeste punten waard is richt.

Vijanden

In het spel zijn er tevens verschillende vijanden te vinden die elk een bepaald aantal punten waard zijn en die elk een andere manier van aanvallen hebben. In de eerste rang zijn vijand 1 en 2 al in het spel. Voor de overige vijanden geldt dat er bij elke nieuwe rang ook een nieuwe vijand bij komt.


Vijand 1. Probeert de speler af te laten gaan door tegen hem aan te botsen. Deze vijand kan alleen rechtdoor vliegen.


Vijand 2. Probeert de speler af te laten gaan door 1 kogel op hem af te sturen. Deze vijand kan alleen rechtdoor vliegen.


Vijand 3. Probeert de speler af te laten gaan door 2 kogels op hem af te sturen. Deze vijand kan alleen schuin vliegen.


Vijand 4. Probeert de speler af te laten gaan door 4 kogels af te vuren die allemaal verschillende kanten op gaan. Deze vijand kan alleen rechtdoor vliegen.


Vijand 5. Probeert de speler af te laten gaan door 3 kogels op hem af te sturen. Deze vijand kan alleen rechtdoor vliegen.


Eindbaas. Verschijnt aan het einde van elke rang en vuurt meerdere kogels af. Hij verdwijnt pas van het scherm als hij verslagen is.

Power-ups

Zoals al eerder gezegd verschijnen er regelmatig verschillende power-ups in het spel. Deze zijn door de gebruiker op te pakken en hebben een effect op bepaalde dingen in het spel.

 Levens kist. Als de gebruiker deze kist op pakt wordt zijn levens-balk gevuld.

 Salvo kist. Als de gebruiker deze kist op pakt wordt de duur van zijn kogel salvo verlengt.

 Raket kist. Als de gebruiker deze kist op pakt krijgt hij er een doelzoekende raket bij.

1.2 Productverantwoording

Het doel van onze app was een real-time spel te maken voor de android telefoon. Hiermee bedoelen wij een spel waar het scherm minimaal 20x per seconden wordt ververst (in onze app 30x) zodat er op een voor het menselijk oog vloeiende manier gespeeld kan worden. Daarnaast moet input van de gebruiker snel genoeg verwerkt worden om de voortgang van het spel niet te hinderen. Het mag niet zo zijn dat een gebruiker af gaat omdat zijn of haar invoer niet snel genoeg verwerkt kon worden, afgaan mag uitsluitend door het eigen falen. Om te onderzoeken of ons doel (een real-time spel voor de android maken) haalbaar was hebben wij er dus voor gekozen om een relatief simpele app te maken en ons wat meer op de research kant te richten.
Daarnaast zijn wij niet echt op zoek naar een unieke app maar meer naar een vermakelijke spel. Ons spel is het beste te vergelijken met een Steven Segal film, deze kijkt men niet voor het diepgaande verhaal, de aangrijpende karakter ontwikkeling of het uitmuntende acteer werk, nee, het is vermaak in de meest pure vorm. Het spel kan gespeeld worden wanneer men: op de bus staat te wachten, in de wachtkamer van de doctor, tijdens een hoorcollege, op de wc en op nog veel meer onverwachte plaatsen waar men geen toegang heeft tot een Steven Segal film.

1.3 Specificaties

1.3.1 Functionele eisen

De functionele eisen geven aan wat het systeem allemaal moet kunnen. De functionele eisen zijn zichtbaar voor de gebruiker en hebben dan ook direct invloed op het gebruik van het systeem. Het systeem moet het volgende kunnen doen:

Weergave

- Vliegtuig van speler wordt weergegeven.
- Meerdere vliegtuigen van de vijanden worden weergegeven.
- Vliegtuig van eventuele eindbaas wordt weergegeven.
- Eventuele power-ups worden weergegeven.
- Afgevuurde kogels worden weergegeven.
- Het aantal levens wordt standaard in het scherm weergeven.

- De moeilijkheidsgraads/rang wordt standaard in het scherm weergeven.
- De score van de gebruikers wordt standaard in het scherm weergeven.
- Het aantal munitie van de gebruikers wordt standaard in het scherm weergeven.

Besturing

- Vliegtuig van gebruiker kan bewegen.
- Gebruiker kan schieten.
- Gebruiker kan een doelzoekende raket afschieten.

Gebruiker

- Het spel kan op pauze.
- High-scores worden bewaard.
- Moeilijkheidsgraad kan worden bewaard.
- Spel kan op bewaarde moeilijkheidsgraad vervolgen.
- Spel moet op ieder moment afgesloten kunnen worden door de gebruiker.

In-game

- Power-ups kunnen worden opgepakt.
- Aantal levens kan worden opgehoogd met een power-up.
- Vijand kan schieten.
- Vijand kan bewegen.
- Gebruiker kan geraakt worden waardoor er levens af gaan.
- Vijand kan geraakt worden en van het scherm verdwijnen.
- Vijand verdwijnt van scherm als hij de gebruiker gepasseerd is.
- Eindbaas kan kanonnen afvuren.
- Eindbaas heeft meerdere levens.
- Eindbaas verdwijnt van scherm (als zijn levens op zijn).
- Eindbaas blijft bovenin het scherm totdat hij vernietigd is.

Menu

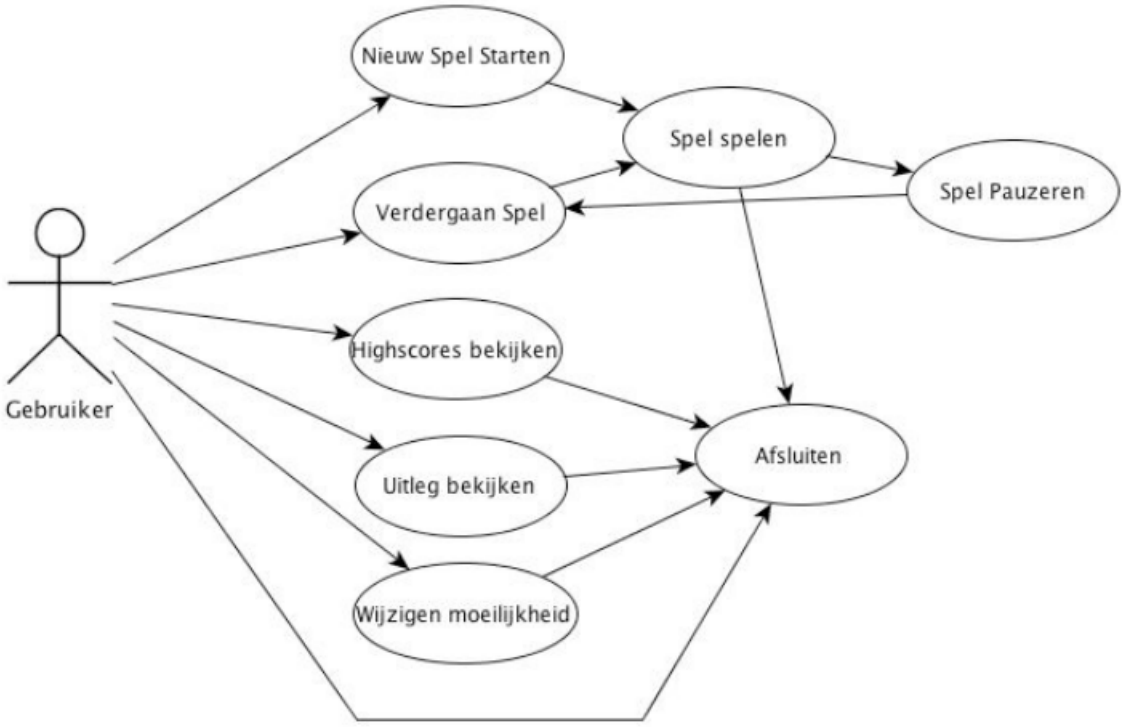
- Speler kan een nieuw spel beginnen.
- Speler kan high-scores bekijken.
- Speler kan verder gaan met een gepauzeerd spel.
- Uitleg over de spelbesturing is beschikbaar voor de gebruiker.

1.3.2 Niet-functionele eisen

Niet-functionele eisen zijn eisen die niet direct zichtbaar zijn voor de gebruiker maar die wel bestaan. Het gaat met name om randvoorwaarden die voor de ontwikkelaar van het systeem van belang zijn.

- Het spel is ontworpen op een zelf gemaakte game-engine.
- Alle vliegtuigen in het spel zijn zelf ontworpen en getekend.
- Alle power-ups en wapens in het spel zijn zelf ontworpen en getekend.
- Het lettertype is zelf ontworpen en getekend.
- Binnen 35 ms na de aanraking van de gebruiker reageert het spel.
- Onderhoud van het systeem moet makkelijk zijn en updates kunnen door de gebruiker worden gedaan zonder dat dit langer dan twee minuten duurt.

1.3.3 Het use case diagram



1.3.4 Use case 'Spel Spelen'

UC numer	3
Naam	Spel Spelen
Omschrijving	De gebruiker speelt het spel
Actor	Gebruiker
Trigger	De gebruiker heeft ervoor gekozen verder te gaan met een spel of een nieuw spel te beginnen
Basisflow	
stap	**actie**
1	Het spel is gestart en de gebruiker kan het vliegtuig besturen
2.1.1	De gebruiker wordt geraakt door een vijand en raakt levens kwijt, gaat verder met stap 1 of als hij geen levens meer heeft met stap 2.1.2
2.1.2	De gebruiker is al zijn levens kwijt en wordt terug gestuurd naar het hoofdmenu
2.2.1	De gebruiker schiet op een vijand, gaat verder met stap 1 of met stap 2.2.2 als de vijand geen levens meer heeft
2.2.2	De vijand heeft geen levens meer en verdwijnt van het scherm, gebruiker gaat verder met stap 1.
2.3.1	De gebruiker pakt een speciaal wapen op, gaat verder met stap 1
2.3.2	De gebruiker schiet een speciaal wapen af, gaat verder met stap 2.2.1
2.4	De gebruiker pakt een power-up op en krijgt extra levens, gaat verder met stap 1.
Alternatieve flow	
stap	**actie**
3.1	Onderbreking van het spelen omdat de gebruiker naar het hoofdmenu terug gaat
3.2	UC 4 gaat van start
precondities	UC1 of UC2 zijn gestart
postcondities	De gebruiker heeft een spel gespeeld
opmerkingen	-

2. Ontwerp

2.1 Globaal ontwerp

Het project bestaat uit drie hoofd componenten: android specifieke platform code, spelmotor en het spel zelf. Spelmotor gebruikt de verschillende standaard api's om een nieuwe omgeving te creren die alle functionaliteit bevat die een spel nodig heeft zoals: het beheren verversen/tekenen van alle objecten, het laden van afbeeldingen en het verkrijgen van invoer van de gebruiker. Het spel maakt gebruik van spelmotor en bevat enkele aanroepen naar spelmotor.

Android platform code
Spelmotor zelf wordt gestart door een activity, deze bevat Java en C code om tussen Java en C++ te communiceren. Het Java deel is waar de applicatie in begint, er wordt een SurfaceView gecreëerd waarna er wordt overgegaan naar het C++ deel. Daarnaast zijn er enkele voor C++ zichtbare statische methoden.

methode	**Omschrijving**
JavaLink.opengl_start()	Initialiseert OpenGL ES voor de huidige thread.
JavaLink.opengl_stop()	Deinitialiseert OpenGL ES voor de huidige thread.
JavaLink.opengl_toon()	Wisselt de back en front buffer.
JavaLink.afsluiten()	Sluit de applicatie op een nette manier.
JavaLink.lees(string naam)	Leest een raw resource met 'naam' en returned een unsigned char* met de inhoud (of NULL).

De reden waarom OpenGL in java word geinitialiseerd is omdat de EGL interface pas vanaf api level 9 in C++ beschikbaar is. Aangezien een groot aantal android devices nog een besturingssysteem met api level 8 bevatten willen wij geen functionaliteit gebruiken die niet op deze devices ondersteund wordt.
Het C++ deel wordt verbonden met het Java deel via een aantal C functies. Het is nodig deze functies in C te schrijven omdat C++ de neiging heeft namen van functies te vervormen.

Functie	**Statische Java code**	**Omschrijving**
start()	CppLink.start()	Maakt een motorthread object en start deze.
stop()	CppLink.stop()	Stopt motorthread en geeft het object vrij.
ververs_dimensie (int b, int h)	CppLink.ververs_dimensie (int b, int h)	Als de scherm resolutie veranderd (de telefoon wordt een kwart slag gedraaid) wordt deze methode aangeroepen.
bericht_vinger (int type, int x, int y)	CppLink.bericht_vinger (int type, int x, int y)	Als er een onTouch event plaatsvindt wordt deze methode aangeroepen, type=vinger_druk, vinger_sleep_begin, vinger_sleep_eind, vinger_sleep_ververs.

Nadat motorthread is gestart en OpenGL geinitialiseerd zullen de volgende stappen worden herhaald totdat de applicatie stopt.

1. motorhoofd.teken()
2. motorhoofd.ververs()
3. java_link.opengl_toon()
4. wacht tot er 33ms zijn verstreken sinds stap 1 (30 beelden per seconden)

De reden waarom er eerst wordt getekend en daarna pas ververst heeft te maken met de manier waarop OpenGL werkt. Als we iets tekenen wordt de tekenopdracht asynchroon naar de GPU verstuurd welke daar dan mee aan de gang gaat. Zodra we opengl_toon() aanroepen wordt er eerst gewacht totdat de GPU klaar is met het verwerken van alle tekenopdrachten. Door het verversen na het tekenen uit te voeren wordt de tijd die wij op de GPU moeten wachten geminimaliseerd.
We gaan niet verder in op het Java, C en motorthread deel van de applicatie. Deze code vormt een heel klein deel van de applicatie daarom zullen we ons in de rest van dit document focussen op de werking van spelmotor en de implementatie van het spel.

Spelmotor

Spelmotor bevat enkele klassen die gebruikt worden om met behulp van OpenGL afbeeldingen weer te geven. Het voordeel aan OpenGL is dat het eigenlijk in 3 dimensies werkt. Al zijn er maar twee dimensies zichtbaar voor de gebruiker de derde dimensie wordt gebruikt om diepte aan te geven. Hierdoor hoeven we onze spel-objecten niet van achter naar voren te tekenen. Alle geladen afbeeldingen worden in een globale lijst bewaard en als men een nieuwe afbeelding wil laden zal eerst deze lijst doorzocht worden, is de afbeelding al geladen dan word deze gegeven. Hierdoor zal een afbeelding nooit twee keer in het geheugen voorkomen. Het laatste deel van spelmotor is het deel waar het spel mee in aanraking komt. Allereerst moet elk spel de klasse spelwereld overerven. Deze klasse is verantwoordelijk voor het beheren van de spel toestand (menu, help...) en het aanmaken van spel-objecten. Alle spel-objecten moeten de klasse motorobject overerven, hierdoor krijgen ze meteen een set aan standaard functionaliteit mee (positie, afmeting, rotatie, richting, snelheid....). Daarnaast kan een motorobject andere motorobjecten als kinderen hebben, deze hebben dan een positie relatief aan hun vader. Om gebruikers invoer te verkrijgen moet de klasse motorinvoer worden overgerfd, deze klasse zal dan automatisch bij spelmotor worden geregistreerd. Het is mogelijk om een klasse te maken die zowel motorobject als motorinvoer overerft.

Spel

Het spel bestaat uit een klasse wereld welke spelwereld overerft, klassen voor de verschillende toestanden en een verzameling aan klasse die motorobject en eventueel motorinvoer overerven. Klasse die motorobject overerven hebben een methode 'ververs' die, bij elke stap dat zij onderdeel zijn van de motorwereld, aangeroepen wordt. Omdat spelmotor op een vaste 30 beelden per seconden draait hoeft hier geen rekening gehouden te worden met de verstreken tijd.

Klasse(n)	**Taak**
wereld	Het beheren van de verschillende toestanden.
toestand_x	Laadt alle afbeeldingen die voor deze toestand nodig zijn, creert alle spelobjecten en maak deze zichtbaar.
spelobjecten	knoppen, speler, vijanden, laser stralen...

2.2 Detail ontwerp

Hier volgt door de use-case 'Spel Spelen' een uitleg over hoe deze gerealiseerd wordt in termen van volledig uitgewerkte klassediagrammen.

spelmotor

motorhoofd	Dit is de basis klasse voor spelmotor, via deze klasse kan een spel alle functionaliteit van spelmotor bereiken. Hij beheert alle verschillende onderdelen van spelmotor (tekenaar, afbeelding lader...).
motortekenaar	Deze klasse wordt gebruikt om afbeeldingen te tekenen. Het is niet nodig voor een spel deze klasse te gebruiken. De klasse wordt intern gebruikt door motorfiguur en is alleen nodig als men de tekenmethode van een spelobject overschrijft.
motorkunst	Beheert alle geladen afbeeldingen en zorgt ervoor dat elke afbeelding maar één keer geladen wordt. Per afbeelding wordt het aantal referenties bijgehouden. Zodra er geen referenties meer naar een afbeelding zijn zal deze uit het geheugen worden verwijderd.
motorletters	Bevat een bitmap font dat door motortekst gebruikt wordt om tekst weer te geven.
motorinvoer	Een basis klasse die overerft kan worden om gebruikers invoer te ontvangen. De constructor zal hem bij motorhoofd registreren en de deconstructor zal hem verwijderen.
spelwereld	Elk spel moet een klasse bevatten die deze klasse overerft. Als spelmotor gestart wordt zal de start() methode aangeroepen worden. Daarna zal elke stap ververs() aangeroepen worden en wanneer de applicatie gesloten wordt zal stop() aangeroepen worden.
motorafbeelding	Een afbeelding die doorgegeven kan worden aan een figuur. De constructor is protected en een afbeelding is alleen te verkrijgen met behulp van de motorkunst klasse.
motorfiguur	Het tekenbare onderdeel van een object. Als motorfiguur een afbeelding ontvangt zal hij zelf een eigen referentie naar deze afbeelding verkrijgen en deze vrijgeven als het object vernietigd wordt, de afbeeldingen moet dus nog steeds door de gebruiker vrijgeven worden. Een animatie kan worden gemaakt door een zegel grootte in te stellen en elke stap de gewenste zegel te selecteren. Als men slechts een gedeelte van een afbeelding wil weergeven kan dit door een selectie op te geven.
motorwereld	Een subklasse van motorobject dat de oorsprong van de hiërarchische object boom vertegenwoordigt. Deze klasse kan als vader opgegeven worden voor spelobjecten om ze zo aan de wereld toe te voegen. Alle objecten die nog steeds onderdeel zijn van de wereld boom wanneer de applicatie gesloten wordt zullen automatisch uit het geheugen verwijderd worden.
motorobject	Een basis klasse voor alle spelobjecten. Deze klasse bevat een hoop standaard functionaliteit die door veel spelobjecten gebruikt word. Zoals een snelheid/richting, rotatie, positie... Elke stap dat een object onderdeel is van de object boom zal ververs worden aangeroepen. Deze MOET door alle subklassen worden geïmplementeerd.
java_link	Alle vanuit java geimporteerde methode zijn te bereiken via deze klasse.
cpp_link	Bevat alle geëxporteerde C functies die door java worden gebruikt om informatie door te geven die niet vanuit C++ beschikbaar is.

Spel

wereld	De spel wereld, hier beheren we de verschillende spel-toestanden.
spelinfo	Bevat de informatie over het spel die opgeslagen wordt alvorens af te sluiten. De volgende variable worden onthouden: de rang, het wapen en de laatst ingevoerde naam.
toestand_spel	De daadwerkelijke spel toestand. Deze toestand beheert de speler en zijn tegenstanders.
kist	Een basis klasse voor alle opraapbare objecten.
kist_xxx	Een opraapbaar item dat het spel beinvloed.
Kogel	Een basis klasse voor objecten die de speler of vijanden schade kunnen toebrengen.
laser, raket	Verschillende subklassen van kogel die de daadwerkelijke munitie simuleren.
speler	De speler, dit is zowel een spel object als een invoer klasse zodat hij kan reageren op invoer van de gebruiker. Bevat een referentie naar spelinfo zodat waar het huidige wapen en rang worden bewaard.
vijand	Klasse voor vijandige objecten, deze klasse bevat standaard functionaliteit die er voor zorgt dat de vijand geraakt kan worden en wat voor ai deze vijand heeft.
vijand_ai_x	Deze klasse bestuurt een vijand, verschillende vijanden hebben verschillende wapens en aanvlieg routes. Wanneer deze klasse aan een vijand gekoppeld wordt zal hij zijn uiterlijk aanpassen.
vijand_moederschip	Alvorens we een rang omhoog gaan moeten we een vijand van dit type vernietigen, het wapen en de hoeveelheid levens van het moederschip zijn afhankelijk van de rang waarvoor we strijden.
wapen	Basis klasse voor wapens. De schiet methode maakt een kogel object of recycled een kogel die niet langer iemand kan raken.
wapen_1, 2, 3...	De individuele wapens, deze gebruiken de schiet methode om een bepaald aantal kogels in een bepaald patroon te maken.
rang	De rang die op het scherm wordt weergegeven. Welke afbeelding er weergegeven word word bepaald aan de hand van spelinfo.
leven	De balk die aangeeft hoeveel leven we nog hebben. Om dit te kunnnen doen heeft leven een referentie naar het speler object nodig.
raketwerper	Toont het aantal doelzoekende raketten dat de speler nog heeft. Als er op dit object gedrukt word en er is 1 raket beschikbaar zal deze op de vijand die de meeste punten waard is gevuurd worden.
score	Toont de huidige score van de speler.

2.3 Ontwerpverantwoording

Tijdens het ontwerpen en ontwikkelen van onze app zijn wij tegen een aantal dingen aangelopen. Sommige van deze dingen waren kleine problemen (zoals bijvoorbeeld hoe gaan we de objecten zelf tekenen zonder dat het er onprofessioneel uitziet) en sommige van deze dingen waren een wat groter probleem. Omdat het uitwerken van al deze problemen een stap voor stap verslag zou worden van wat er allemaal gedaan is werken wij hier enkel twee ontwerp beslissingen uit. Wij hebben hier voor beslissingen gekozen die in ons opzicht de app in grote mate verbeteren en wat uiteindelijk goede beslissingen zijn geweest.

Beslissing 1
Tijdens het ontwerpen van de app kwamen wij erachter dat er verschillende manieren waren om objecten te tekenen in Tekenen OpenGL ES. Deze waren:
• SurfaceView.onDraw()
• SurfaceView.lockContext()
• GLSurfaceView.Renderer.onDrawFrame()
• Eigen Surface view met OpenGL ES

Methode 1 en 3 vielen meteen af. Deze methoden werken met behulp van een asynchrone thread die bestuurd word vanuit de main thread. Aangezien onze engine in een eigen thread draait zou deze niet direct invloed kunnen uitoefenen op het geen dat getekend wordt en wanneer dat getekend word. Hierdoor zouden we alle spel objecten moeten synchronizeren om te voorkomen dat er tijdens het tekenen iets aan het object veranderd.
Methode 2 is enkel vanuit java te gebruiken en doet veel berekeningen in de game thread, zoals bijvoorbeeld rotaties. OpenGL ES bied de mogelijkheid om deze berekeningen op een graphische processor te doen (indien deze aanwezig is) het nadeel is wel dat de getransformeerde coördinaten niet makkelijk terug te lezen zijn.
Eerdere ervaring met OpenGL en de mogelijkheid om de volledige cappaciteit van een android device te benutten hebben ons doen besluiten methode 4 te gebruiken. Met behulp van een (in java geïmplementeerde) eigen SurfaceView en EGL wordt er een OpenGL ES 1.1 context verkregen deze context wordt daarna verbonden met de game thread zodat het verversen van het spel en het tekenen in de zelfde thread gebeurd. Hierdoor maken we dus gebruik van een eventuele grafische processor zonder dat alle spel objecten door meer dan een thread gebruikt worden.

Het is dus wel van belang dat we goed registreren welke objecten er in botsing kunnen komen en of er rekening gehouden moet worden met rotaties, want om de absolute coordinaten van een object te verkrijgen zullen we deze handmatig moeten berekenen. Daarom hebben we de volgende beperkingen opgelegt aan botsingen:
• Alle objecten worden als een cirkel beschouwd met als straal de helft van de kortste zijde van het object.
• Als een object om zijn middelpunt roteert is het niet nodig om zijn rotatie mee te nemen tijdens het detecteren van een botsing.
• Het is mogelijk een object andere straal te geven dat beter aansluit op de vorm van het object.
• Omdat wortel trekken een van de duurdere operaties is word de precieze afstand tussen twee botsende objecten niet berekend, hierdoor hoeven we alleen het kwadraad van de lengte van de verschil vector (positie object 1 - positie object 2) te vergelijken met de som van de twee stralen in het kwadraad.

In pseudo code ziet dit er zo uit:

```
bool botst(motorobject* a, motorobject* b)
{
        positie pa = a->bereken_absolute_positie()
        positie pb = b->bereken_absolute_positie()
        positie vv = pa-pb
        int     afstand_kwadraad  = vv.x*vv.x+vv.y*vv.y
        int     stralen_kwadraad  = a->straal()+b->straal()
                stralen_kwadraad *= stralen_kwadraad
```

```
        return stralen_kwadraad >= afstand_kwadraad
}
```

bereken_absolute_positie() geeft de absolute positie van een object, als deze getransformeerd is zal hij deze berekenen.

Beslissing 2
Omdat niet alle telefoons een FPU (Floating-Point Unit) hebben worden op deze telefoons berekeningen met variable van het type float of double geëmuleerd. Dit is veel langzamer en in een spel dus niet gewenst. Om te voorkomen dat ons spel langzaam wordt op android devices zonder FPU is het gebruik van floating point variable zo veel mogelijk terug gedrongen. Dit levert problemen op wanneer we een object een snelheid en een richting willen geven. Normaal gesproken zou dit op de volgende manier geïmplementeerd kunnen worden:

In de objectklasse:

```
//de positie
double x, y;
//de snelheid
double sx, sy;
```

In de ververs methode:

```
void object::ververs()
{
        .....
        x += sx;
        y += sy;
        .....
}
```

Het instellen van de richting en snelheid kan dan als volgt gedaan worden:

```
void object::richting_snelheid(float r, float s)
{
        sx = cos(r)*s;
        sy = sin(r)*s;
}
```

Wij gebruiken integers om de positie en snelheid aan te geven, hierdoor verliezen we een hoop precisie en kan het zelfs zo zijn dat door het afronden van de doubles sx of sy 0 wordt terwijl deze bijvoorbeeld 0.9 had moeten zijn. Daarom hebben wij dit op de volgende manier in onze app geimplementeerd:
• Alle snelheden worden opgegeven in verplaatsing per seconde.
• In elk object wordt er een teller bijgehouden die telkens van 0 tot 30 telt.
• Een beweging wordt beschreven met behulp van 4 intergers en 2 booleans.

De 4 intergers zijn:
• int stap_x, stap_y; (Deze integers geven aan hoeveel eenheden een object elke keer als hij ververst word verplaatst.)
• int mod_x, mod_y; (Als (teller%mod_x) of (teller%mod_y) gelijk is aan 0 zal er niet met respectivelijk stap_x of stap_y bewogen worden maar met (stap_x+1), (stap_x-1), (stap_y+1)

of (stap_y-1). Of er -1 of +1 wordt gebruikt is afhankelijk van de variable inv_x en inv_y. We kunnen hier niet simpelweg kijken of stap_x of stap_y ¡ 0 of ¿ 0 omdat deze ook 0 kunnen zijn.)

De 2 booleans zijn:
• bool inv_x, inv_y; (Als deze variable true zijn moet er op (teller%mod_x,y) met (stap_x,y-1) bewogen worden, zijn ze false dan word er met +1 bewogen.

We kunnen nu dus met minder als 1 pixel per verversing bewegen zonder gebruik te maken van floating point variable. Wel maken we nog gebruik van deel operaties (% is een bijproduct van een deel operatie) en is het nog steeds nodig om doubles te gebruiken wanneer we stap_x,y en mod_x,y uitrekenen, maar dit word niet elke stap gedaan.

In de bijlage is de code gegeven voor het berekenen van de 6 variable en de berekening die elke stap word uitgevoerd.

Usability 1
Wat opviel aan de eerste usability test was dat mensen het spel snel genoeg vonden reageren dit was vreemd omdat wij zelf wel een vertraging opmerkten als er met de vinger over het scherm gesleept werd. Hoewel dit dus niet naar voren kwam in de eerste usability test hebben wij toch de snelheid aangepast.
Verder kwam naar voren dat de leesbaarheid van de tekst in het menu en tijdens het spelen is volgens iedereen geen probleem veroorzaakte, wel waren ze het er over eens dat het lettertype mooier gemaakt mag worden. Hier zijn we niet aan toe gekomen aangezien wij dit een minder belangrijk probleem vonden dan sommige andere problemen, het lettertype is geen 'dealbreaker in ons opzicht. Ook de speluitleg werd als lelijk bevonden maar ook hier geldt dat dit geen prioriteit was voor ons.
De groep jongeren had geen problemen met de gekozen spel besturing, de andere groep had vooral in het begin moeite het spel te besturen. Hieraan hebben wij helaas niks kunnen doen, dit omdat het enkel een kwestie van oefening is.
Wat opviel was dat beiden doelgroepen vonden dat na enige tijd spelen het spel te makkelijk word. Hier hebben wij wel wat aan gedaan. We hebben meer vijanden ingebouwd die verschillende wapens hebben en op verschillende manieren vliegen om zo variatie en uitdaging in het spel te brengen.

Wat opviel aan de eerste usability test was dat de dingen die naar voren kwamen vooral esthetische dingen waren waar wij helaas niet veel meer aan hebben kunnen doen wegens beperkte tijd en andere hogere prioriteiten.

3. Evaluatie

Vergeleken met vijftien jaar geleden hebben tegenwoordig veel meer mensen een mobiele telefoon, zo bleek uit onderzoek van het CBS dat er in vergelijking met 1995 acht keer meer mobiele telefoons verkocht waren in 2006. En dat niet alleen, uit verder en meer recent-onderzoek van gfk (een consumentenpanel bedrijf) blijkt dat eind 2011 45 procent van de bevolking in het bezit was van een smartphone. Bij dit toenemende gebruik van smartphones komt als gevolg het toenemende aantal apps. In de huidige app-stores zijn er zodoende dus miljoenen verschillende apps te vinden.
Met dit grote aantal apps is de keuze zeer uitgebreid en iemand die een app ontwikkelt moet dus veel competitie verwachten en een app zal dus alleen goed verkocht/gedownload worden als deze goed in elkaar ziet en gebruiksvriendelijk is. Als dit namelijk niet het geval is zal de gebruiker simpelweg een van de, betere, alternatieven kiezen.
Dit gezegd te hebben is het dus ook voor onze app belangrijk dat hij goed in elkaar is en gebruiksvriendelijk is. Kort om: hij moet een goede 'usability' hebben. Om deze usability te testen kan men natuurlijk het beste rechtstreeks naar de potentiële gebruiker gaan en aan hem vragen wat hij van de app vindt. Dit geeft een goed inzicht in wat de eventuele goede punten en verbeterpunten van de app zijn in de ogen van de gebruiker wat erg handig is want uiteindelijk is de gebruiker degene die ervoor zorgt dat de app verkocht wordt.

3.1 Testdoelen

Om de zojuist genoemde usability te testen werden er van te voren een aantal testdoelen opgesteld. Dit om de test bepaalde kanten op te sturen zodat wij de informatie die wij nodig hadden konden verkrijgen. De testdoelen die wij kozen waren voornamelijk gebaseerd op de keuzes die wij tijdens ons ontwikkel proces hebben genomen en waar wij graag feedback over wilden hebben.
Zo hebben wij in het begin van het proces lang nagedacht over hoe de besturing in elkaar zou moeten zitten, over hoe we alles grafisch gingen vormgeven en over welke functies wij in de app wilden bouwen. Na deze punten op een rij te zetten kwamen wij op een aantal testdoelen uit die wij vervolgens naast de 5 E's van usability testing hebben gelegd om zo tot duidelijk geformuleerde testdoelen te komen.
Met de vijf E's wordt hier het volgende bedoeld:
Effective: How completely and accurately the work or experience is completed or goals reached
Efficient: How quickly this work can be completed
Engaging: How well the interface draws the user into the interaction and how pleasant and satisfying it is to use
Error Tolerant: How well the product prevents errors and can help the user recover from mistakes that do occur
Easy to Learn: How well the product supports both the initial orientation and continued learning throughout the complete lifetime of use.

Na deze zorgvuldig te hebben bekeken kwamen wij op de volgende testdoelen uit:
• Er achter komen wat de gebruiker van de besturing vindt en of deze ervoor zorgt dat het spel fijn te spelen is. (Effective)
• Er achter komen wat de gebruiker van het uiterlijk van de app vindt. (Engaging)
• Er achter komen of de menu's en de knoppen duidelijk zijn in wat zij doen. (Easy to Learn)
• Er achter komen of het duidelijk is wat de bedoeling van het spel is. (Easy to Learn)
• Er achter komen wat de gebruiker van de moeilijkheidsgraad van het spel vindt en of het

uitdagend is/blijft. (Engaging)
• Er achter komen wat de gebruiker vindt van de tijd die de app er over doet om te reageren op bepaalde commando's. (Effective)
• Er achter komen of de app genoeg indruk heeft gemaakt op de gebruiker dat hij de app nog een keer zou willen gebruiken. (Efficient)

Zoals te zien is focussen wij ons in dit onderzoek vooral op de Effective en Engaging kanten van de app. Dit is bewust gedaan omdat de error tolerant gedeelten in onze app in ons oogpunt nog niet voldoende aan bod komen om vaar vragen over te stellen. Zo hebben wij in de vele keren dat wij de app gebruikten nooit meegemaakt dat hij vast liep of dat er een andere fout tevoorschijn kwam. Ook als wij dit actief proberen (door knoppen herhaaldelijk in te drukken etc.) gebeurt dit niet.

Deze testdoelen staan direct met de functionele en niet-functionele eisen in verband. Zo gaat het eerste punt direct over het kopje 'Besturing' uit de eisen, het uiterlijk over het kopje 'Weergave', de duidelijkheid gaat over de speluitleg die in het kopje 'Menu' wordt genoemd, de moeilijkheidsgraad gaat in het kopje 'Gebruiker' en de tijd staat bij de niet-functionele eisen.

3.2 Gebruik(sub)groepen

Voor onze gebruikers hebben wij onderscheid gemaakt tussen twee verschillende groepen mensen. De eerste groep was een groep mensen die ervaring had met smartphones en bekend was met games op dit toestel. De tweede groep was een groep mensen die weinig tot geen ervaring hadden met smartphones en games hier op. Dit onderverdelen in groepen hebben wij gedaan om te kijken hoe goed de app aangepast is op bepaalde leeftijds groepen en om te kijken of het eventueel mogelijk is om een app te maken die zowel voor beginnende als ervaren smartphone gebruikers leuk is.

Uiteindelijk kwamen wij door deze scheiding in groepen op een totaal aantal van 6 mensen waarvan elke groep er 3 had. De mensen in de eerste groep (met ervaring) waren drie personen, waarvan allen van het mannelijk geslacht waren. Zij vielen allen in de leefstijdscategorie van 17 tot 21 jaar. De tweede groep mensen (zonder ervaring) waren drie personen, waarvan één van het mannelijk geslacht en twee van het vrouwelijk geslacht. Hun leeftijdscategorie varieerde van 48 tot en met 54 jaar.

Alle deelnemers hebben wij verwerft uit onze eigen omgeving (denk hierbij aan ooms, tante's, vrienden en buren).

3.3 Scenario's

Om de eerder genoemde testdoelen te testen werden er enkele scenario's bedacht die als ware de gebruiker door de app heen leidden. Het aantal scenario's was in ons geval erg mager. Dit komt omdat onze app relatief simpel is qua menu's en dus weinig mogelijke acties heeft. De door ons gekozen scenario's waren:
• De gebruiker wilt de app op starten
• De gebruiker wilt de scores bekijken
• De gebruiker wilt de spel-uitleg bekijken

• De gebruiker wilt een nieuw spel beginnen

3.4 Methoden voor gegevensverzameling

De methoden van dataverzameling waar wij voor gekozen hebben was een vragenlijst. Deze vragenlijst bestond uit 56 verschillende vragen van zowel open als gesloten aard en is in de bijlagen te vinden. Tevens onderdeel van het materiaal was de app zelf, deze werd afgespeeld op een android emulator op de computer. Dit omdat, hoewel wij wel mensen met een smartphone kennen, niemand hem voor een dag wilde uit lenen. Uiteraard hebben wij de app wel op de smartphone getest en wij kwamen tot de conclusie dat er geen verschil in het spel is tussen de emulator en de smartphone.

3.5 Procedure

Na de deelnemers te hebben gevraagd of zij mee wilden doen aan ons onderzoek legden wij ze de vragenlijst voor met daarbij een korte uitleg waarover dit onderzoekje ging en waarom wij het afnamen. Tevens werd de deelnemer verteld dat de uitleg op de test zelf stond en dat hij of zij altijd tijdens de test ons vragen mocht stellen.
Nadat de speler de gehele vragenlijst had doorlopen (wat ongeveer 15 minuten duurde ivm. het spel spelen) bedankten wij hun voor hun medewerking en vroegen we of zij nog verdere vragen hadden en of ze de vragen uit de test duidelijk en begrijpelijk vonden (post-test). Deze verdere vragen waren er niet en ook waren er geen opmerkingen over de begrijpelijkheid van de test zelf.

Na dit alles hebben wij de data geanalyseerd. Dit hebben wij ten eerste gedaan door de test op te delen in verschillende soorten vragen. De soorten vragen die we hebben onderscheden waren als volgt:
• Test vragen: vragen om te testen of de gebruiker een bepaald scenario heeft weten te voltooien.
• Opmerkingen: open vragen waarin de gebruiker werd gevraagd of hij of zij over bepaalde onderdelen opmerkingen had.
• Tijd vragen: vragen die over de snelheid/reactietijd van het programma gingen.
• Uiterlijk vragen: vragen die over het uiterlijk van de app ging.
• Begrip vragen: vragen die over de duidelijkheid/begrijpelijkheid van de app gingen.
• Besturing vragen: vragen die over de besturing van het spel gingen.
• Moeilijkheid vragen: vragen die over de moeilijkheidsgraad van het spel gingen.

Vervolgens hebben we voor de onderste vijf typen vragen de antwoorden bepaalde scores van 1 tot 5 gegeven waarbij het antwoord 'Slecht' 1 punt scoorde en het antwoord 'Goed' 5 punten scoorde.
Met deze scores hebben we vervolgens zowel het totale gemiddelde cijfer van de app kunnen uitrekenen als de individuele scores per type vraag om zo ook inzicht te krijgen in hoe de verschillende onderdelen (de 5 E's) in onze app scoren.

3.6 Resultaten

Hieronder is een tabel te vinden met per persoon de gemiddelde scores voor de onderdelen van de app. Onderaan is tevens het totaal gemiddelde te vinden.

	Tijd	**Uiterlijk**	**Begrip**	**Besturing**	**Moeilijkheid**	**Gehele app**
Volwassene1.	4.6	4.1	3.0	3.0	3.6	3.7
Volwassene2.	4.2	4.1	2.5	3.0	3.3	3.4
Volwassene3.	4.4	4.0	2.5	3.0	3.4	3.5
Jongere1.	3.8	3.7	3.5	5.0	3.9	4.0
Jongere2.	3.4	3.4	3.3	5.0	3.4	3.7
Jongere3.	3.6	3.5	3.5	4.0	3.8	3.7
totaal gem	4.0	3.8	3.0	4.0	3.6	3.7

Hieruit valt ook weer af te leiden wat de verschillen in gemiddelde per groep waren. Zo gaven de volwassenen de gehele app gemiddeld een 3.5 terwijl de jongeren de app een 3.8 gaven. Verder scoorden de groepen respectievelijk op de onderdelen Tijd, Uiterlijk, Begrip, Besturing en Moeilijkheid gemiddeld een 4.4/4.1/2.7/3.0/3.4 (voor de volwassenen) en gemiddeld een 3.6/3.5/3.4/4.7/3.7 (voor de jongeren).

De volwassenen behaalden voordat zij af gingen een minimale rang van 1 en een maximale rang van 2. Daarbij haalden zij met de groep een gemiddelde score van 1087 punten.
De jongeren behaalden voordat zij af gingen een minimale rang van 3 en een maximale rang van 4. Daarbij haalden zij met de groep een gemiddelde score van 5390 punten.

3.7 Conclusies

Voor de totale app hebben we van de gehele groep deelnemers gemiddeld een 3.7 gekregen (wat ruw vertaald een 7.4 is). Met dit cijfer zijn wij tevreden omdat het een ruime voldoende is.
Als we beter kijken naar de verschillende onderdelen die gescoord zijn valt het ons op dat het onderdeel 'Begrip' aan de lage kant is (als enigste onderdeel omgezet een 6). Dit is echter ook terug te zien in de open 'Opmerkingen' vragen, 5 van de 6 mensen gaven daar aan dat de speluitleg aan de korte kant en niet heel duidelijk was. Aan de overige gemiddelde valt naar onze mening in eerste opslag niets bijzonders te zien.

Als je echter per groep gaat kijken dan springen er wel meerdere verschillen uit. Wat ten eerste opvalt is dat de volwassen voor zowel het onderdeel 'Tijd' als het onderdeel 'Uiterlijk' gemiddeld best hoger scoren dan de jongeren. Na dit opgemerkt te hebben zijn we nog een keer in gesprek gegaan met de deelnemers en door wat meer door te vragen kwamen we erachter dat de jongeren onze app regelmatig vergeleken met andere app's. Wij verwachten dan ook dat deze jongeren qua snelheid en grafische vormgeving hogere eisen hebben dan de volwassenen simpel weg omdat ze meer ervaring hebben met app's en dus weten wat realistisch is om te verwachten. Wij denken dat dit redelijk aannemelijk is omdat in de test ook duidelijk een correlatie is tussen het antwoord op de vraag hoe vaak de deelnemer een telefoon gebruikt en tussen de groep.

Wat tevens opvalt is dat als het om 'Begrip' en 'Besturing' gaat het omgekeerde fenomeen te zien is. Hier scoren de jongeren juist duidelijk hoger op dan de volwassenen. Dit kwam

echter al in de test zelf naar voren. Het viel ons tijdens het afnemen van de test dan ook op dat we aan de volwassen groep vaker uitleg moesten geven over hoe de besturing werkte. Toen we aan een van de jongeren achteraf vroeg waarom hij dacht dat hij het spel beter begreep dan een van de volwassenen (in dit geval zijn vader) zei hij te denken dat het komt omdat mensen van zijn leeftijd vaker dit soort spelletjes spelen en dat deze spelletjes vaak op elkaar lijken en dat hij daarom misschien beter is. Ook dit leek ons wel een aardig logische aanname vooral aangezien er ook duidelijk in de rangen en scores terug te zien is welke groep beter in het spelletje was.

Natuurlijk blijft het op dit moment voornamelijk bij speculeren. Deze cijfers laten wel verschillen in de groepen zien maar vertellen niet duidelijk waar die verschillen nu aan liggen. Wel kunnen we concluderen dat het duidelijk is dat er voor verschillende doelgroepen verschillende eisen en wensen zijn en dat het lastig kan zijn om rekening te houden met een breed publiek.
Verder hebben wij kunnen concluderen dat wij onze vooraf gestelde doelen hebben behaald en dat ook de requirements stuk voor stuk zijn behaald. De verschillende onderdelen hebben stuk voor stuk gemiddeld een voldoende gekregen en hoewel er nog wel verbeterpunten zijn (uitgebreidere uitleg, een mooiere grafische vormgeving) zijn wij toch zeer tevreden met de app vooral aangezien een deel van onze deelnemers aangaf deze app best nog een keer te willen spelen.

4. Reflectie

Hieronder volgen zowel de groeps- als de individuele-reflecties over hoe wij het proces verloop vonden en over wat wij hebben geleerd.

4.1 Groep

Aan het begin van de cursus kwamen wij gelukkig al snel tot de conclusie dat wij beiden graag een spel wilde maken. Dit heeft de beslissing 'wat te maken?' heel erg vereenvoudigd. Sanne was bereid meer ontwerpende en documenterende taken over te nemen terwijl Charlie aan het werk ging met de eerste grote mijlpaal, spelmotor. Aanvankelijk wilde wij een op grand theft auto 2 gebaseerd spel maken waar de speler niet de rol van een crimineel aanneemt maar die van een agent met een werkwijze zoals Dirty Harry. Al snel werd ons duidelijk dat dit een veel te ambitieus plan was en hebben we besloten om een remake te maken van een oude favoriet: "Raptor". Charlie heeft het spel geprogrammeerd en Sanne heeft het spel getekend. Wanneer er een deadline naderde bekeken we welk onderdeel (tekenen/programmeren) het meeste tijd in beslag had genomen en aan de hand daarvan werden andere taken opgedeeld. Hierdoor vinden wij beiden dat we allebei even actief aan het project hebben bijgedragen en dit heeft op geen enkel moment een conflict opgeleverd. We zijn beiden trots dat het ons gelukt is binnen afzienbare tijd een game engine en een spel te produceren waarvan zowel de grafische kant als de programmeer code volledig door ons zijn gemaakt. Alle problemen die wij op onze weg tegen kwamen (en dat waren er nogal wat) hebben we helemaal zelf opgelost zonder gebruik te maken van werk dat door derden is vervaardigt. Hierdoor hebben wij wel een tweetal negatieve ervaringen opgedaan tijdens de cursus. Ten eerste vinden wij het jammer dat al dit harde werk in een demonstratie van enkele minuten met een 6 is beoordeeld. Grafisch zijn wij erg tevreden over het spel en de usability tests tonen ook dat deze mening niet alleen door ons gedeeld word. Daarnaast is het een van de weinige producten die in de emulator op een acceptabele snelheid draait, en dit is waar toch het meeste werk in heeft gezeten. Hierdoor hebben wij geprobeerd om tijdens de eindpresentatie een indruk te wekken van het grote aantal problemen dat wij hebben overwonnen om dit voor elkaar te krijgen, maar dat resulteerde uiteindelijk in tijdnood ondanks het feit dat wij al een groot deel buiten beschouwing hadden gelaten. Dit is ook meteen wat wij de volgende keer anders gaan doen, de demo niet beschouwen als slechts een 'bewijs dat er iets gemaakt is' maar als een onderdeel waar ook een cijfer aan vast zit, en we dus zelf ook duidelijker de aandacht naar onze app toe moeten trekken.

4.2 Individueel Charlie

Tijdens RenD1 heb ik veel geleerd over de werking van een android device. Achter de schermen heeft google goed veel van Linux "geleend" en de documentatie van de C++ mogelijkheden van de android ndk is dan ook niet veel meer als een plat tekst bestandje met verwijzingen naar de man pagina's van enkele al dan niet geheel ondersteunde bibliotheken. Aan het begin van de cursus vond ik het niet erg om ontbrekende onderdelen zelf te schrijven, maar richting het einde werd het steeds vervelender om het idee te hebben bijna klaar te zijn om dan te ontdekken dat zoiets simpels als het opslaan van instellingen niet vanuit C++ kan en er nog weer een eigen formaat verzonnen moest worden. Binnenkort krijg ik een Windows telefoon en hoop dat het daar beter geregeld is maar tot dan blijf ik trouw aan Linux waar als iets niet werkt je het tenminste zelf kunt oplossen. Als plus punt hebben we wel een werkend spel

geproduceerd dat zelfs op de emulator met een respectabele snelheid draait. Het ontwerpen ging ook goed en er is uiteindelijk nagenoeg niet afgeweken van het oorspronkelijke spel ontwerp en de meeste problemen zijn opgelost in het spelmotor gedeelte waardoor de broncode van het spel zelf overzichtelijk is gebleven en (gemakkelijk) geport zou kunnen worden. Het presenteren vond ik ook goed gaan en de samenwerking heeft naar mijn mening ook geen problemen opgeleverd. Al met al vond ik het een leuke ervaring om een volledig programma te bouwen, iets wat bij de cursussen programmeren en OO ontbrak.

4.3 Individueel Sanne

Het voornaamste wat ik bij Research en Development geleerd heb is hoe android werkt en hoe er apps voor dit platform gemaakt moeten worden. Daarnaast heb ik ook geleerd hoe je usability tests moet uitvoeren, voordat dit vak begon wist ik überhaupt niet wat een usabilityonderzoek was, nu weet ik dat wel en weet ik dat het een van de belangrijkste tools is om je product te testen.
Wat mij ook heel erg opviel was dat er bij het maken van een app nog best veel komt kijken. In het begin had ik vooral de neiging om gelijk aan de slag te gaan met iets maken terwijl ik nu heb geleerd dat een ontwerp heel erg handig kan zijn (we hebben er regelmatig enige houvast aan gehad). Het tussendoor evalueren van de app heeft ook geholpen om ons op het goede pad te houden.
In het begin bij de eerste presentatie vond ik het niet fijn om te presenteren en ik kreeg dan ook als feedback dat ik best zenuwachtig over kwam (wat ik ook was). In de werkgroep waarin de presentaties besproken werden zijn mij een paar tips gegeven wat ik hier aan zou kunnen doen. Deze hebben mij geholpen, na de tweede presentatie kreeg ik goede feedback en werd mij verteld dat ik niet meer zenuwachtig overkwam.
Het samenwerken heeft geen problemen gegeven. Ik denk dat we elkaar goed hebben aangevuld.
Ik vond het leuk om een project te doen omdat ik zo langdurig met hetzelfde bezig was en zo dus ook de verschillende stappen van een proces heb meegemaakt.

Bijlage 1. De code

```
/* Snelheid instellen.
 *
 * Als de richting van een object veranderd word deze methode opnieuw
 * uitgevoerd met de huidige snelheid als parameter.
 * Waardoor deze berekening opnieuw word uitgevoerd.
 *
 * m_* variable zijn atributen.
 *
 * Zonder deze code zou de stap groote 1 zijn (30 stappen per seconden).
 * Nu is hij 1/30 (1 per seconden)
 */

void motorobject::krijg_snelheid(unsigned int s)
{
        //bewaar snelheid
        m_snelheid       = s;
        m_stappen_teller = 0;

        //bereken componenten
        double rad_hoek     = double(m_richting)*(2*PI/360.0);
        double x_component = cos(rad_hoek)*double(s);
        double y_component = sin(rad_hoek)*double(s);

        //maak positief
        m_snelheid_inv_x = false;
        if (x_component < 0)
        {
                x_component      *= -1;
                m_snelheid_inv_x  = true;
        }
        m_snelheid_inv_y = false;
        if (y_component < 0)
        {
                y_component      *= -1;
                m_snelheid_inv_y  = true;
        }

        //verplaatsing per stap
        m_snelheid_stap_x = 0;
        while (int(x_component) > 30)
         {m_snelheid_stap_x++;x_component -= 30.0;};
        m_snelheid_stap_y = 0;
        while (int(y_component) > 30)
         {m_snelheid_stap_y++;y_component -= 30.0;};

        //extra?
```

```
        m_snelheid_mod_x = 0;
        if (int(x_component) != 0)
        {
                m_snelheid_mod_x = 30/int(x_component);
        }
        m_snelheid_mod_y = 0;
        if (int(y_component) != 0)
        {
                m_snelheid_mod_y = 30/int(y_component);
        }
}

/* motorobject::intern_ververs is een niet virtueele methode van
 *  motorobject.
 *
 * naast ververs() (gebruiker implementeerd deze) roept hij ook
 * intern_ververs van al zijn kinderen aan.
 *
 * de methode word 30 keer per seconden aangeroepen.
 */
void motorobject::intern_ververs()
{
        //heeft een snelheid?
        if (m_snelheid != 0)
        {
                //vaste stap
                int x_bij = m_snelheid_stap_x;
                int y_bij = m_snelheid_stap_y;

                //precisie stap?
                //!= 0 omdat anders 'divide by zero, reboot universe'
                if (m_snelheid_mod_x != 0 && m_stappen_teller%m_snelheid_mod_x == 0)
                {
                        x_bij++;
                }
                if (m_snelheid_mod_y != 0 && m_stappen_teller%m_snelheid_mod_y == 0)
                {
                        y_bij++;
                }

                // sneller? -1*x_bij == ((1<<31)|x_bij)+1
                // maar dat weet g++ veel beter
                x_bij = m_snelheid_inv_x ? -1*x_bij : x_bij;
                y_bij = m_snelheid_inv_y ? -1*y_bij : y_bij;

                //verplaats!
                m_x += x_bij;
                m_y += y_bij;

                //verhoog stappenteller
```

```
                m_stappen_teller += 1;
                m_stappen_teller %= 30;
        }

        //verversen
        ververs();

        //ververs kinderen
        for (cgom::lijst<motorobject*>::loper l=m_kinderen;l.geldig();l++)
        {
                (*l)->intern_ververs();
        }
}
```

Bijlage 2. De usabilitytest

Vragenlijst Usability onderzoek MKULTRA app

Bedankt voor het mee doen aan het usability onderzoek voor onze smartphone app MKULTRA. Het doel van dit onderzoek is ons inzicht te brengen in de sterke kanten en de eventuele verbeterpunten van de door ons ontwikkelde app. De informatie die we met deze test verkrijgen zal gebruikt worden als eindevaluatie van onze app en zal bij dragen aan de beslissing of deze app goed genoeg is om daadwerkelijk door mensen in gebruik te laten nemen.

Dit onderzoek bestaat uit een korte vragenlijst met daarin een aantal scenario's. Deze scenario's beschrijven een bepaalde situatie die mensen mogelijk kunnen mee maken tijdens het gebruik van een smartphone app.
Een voorbeeld van zo een situatie is:

> *De gebruiker wilt de spel-uitleg bekijken*

Nu is het de bedoeling in dit onderzoek dat u zich als het ware verplaatst in de gebruiker waar de situatie om gaat en dus het doel van de gebruiker probeert te bereiken. (Wij vragen u aan de hand van het voorgaande voorbeeld dus om de spel-uitleg te gaan bekijken.) Het kan zijn dat het u niet lukt om een bepaald doel te behalen, dit is geen probleem. Wij vragen u dan wel de rede waarom het niet gelukt is op te schrijven.
Als u het scenario hebt uitgevoerd vervolgt het onderzoek zich met een aantal vragen over desbetreffend scenario die u, het liefst zo waarheidsgetrouw mogelijk, kunt invullen. Bij deze vragen bestaan er geen foute of slechte antwoorden, het is voor ons enkel belangrijk om uw mening over de app te krijgen.
Dit onderzoek bestaat uit 56 vragen die of gesloten of open kunnen zijn. Het gehele onderzoek duurt ongeveer 15 minuten.

Standaardvragen

Naam: .....
Leeftijd: .....
Hoe vaak in uw leven hebt u een smartphone gebruikt? (kruis aan wat van toepassing is)

- ◯ nooit
- ◯ een enkele keer
- ◯ soms
- ◯ regelmatig
- ◯ elke dag

Scenario 1

De gebruiker wilt de app starten

1. Is het gelukt scenario 1 te volbrengen?
   - ◯ Ja
   - ◯ Nee, omdat: .....
     .....

2. wat vond u van de tijd die het duurde voordat het menu tevoorschijn kwam?
   - ◯ Slecht
   - ◯ Onvoldoende
   - ◯ Neutraal/Geen mening
   - ◯ Voldoende
   - ◯ Goed

3. Wat vond u van het uiterlijk van het menu?
   - ◯ Slecht
   - ◯ Onvoldoende
   - ◯ Neutraal/Geen mening
   - ◯ Voldoende
   - ◯ Goed

4. Wat vond u van de leesbaarheid van de tekst?
   - ◯ Slecht
   - ◯ Onvoldoende
   - ◯ Neutraal/Geen mening
   - ◯ Voldoende
   - ◯ Goed

5. Wat vond u van de duidelijkheid (qua betekenis) van de knoppen?
   - ◯ Slecht
   - ◯ Onvoldoende

◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

6. Leg kort uit waarvoor u denkt dat de knoppen dienen:

Speel knop: .....
Scores knop: .....
Help knop: .....
Afsluiten knop: .....

7. Heeft u nog tips/opmerkingen voor en op het menu?

.....
.....
.....
.....

Scenario 2

De gebruiker wilt de scores bekijken

8. Is het gelukt scenario 2 te volbrengen?

◯ Ja
◯ Nee, omdat: .....
.....

9. Wat vond u van de tijd die het duurde voordat de lijst met scores tevoorschijn kwam?

◯ Slecht
◯ Onvoldoende
◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

10. Wat vond u van het uiterlijk van deze lijst?

◯ Slecht
◯ Onvoldoende
◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

11. Wat vond u van de leesbaarheid van de tekst?

◯ Slecht
◯ Onvoldoende

◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

12. Heeft u nog tips/opmerkingen voor en op de score-lijst?

.....
.....
.....
.....

Scenario 3

De gebruiker wilt de spel-uitleg bekijken

13. Is het gelukt scenario 3 te volbrengen?
◯ Ja
◯ Nee, omdat: .....
.....

14. wat vond u van de tijd die het duurde voordat de uitleg tevoorschijn kwam?
◯ Slecht
◯ Onvoldoende
◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

15. Wat vond u van het uiterlijk van deze uitleg?
◯ Slecht
◯ Onvoldoende
◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

16. Wat vond u van de leesbaarheid van de tekst?
◯ Slecht
◯ Onvoldoende
◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

17. Wat vond u van de duidelijkheid van de uitleg?
◯ Slecht

◯ Onvoldoende
◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

18. Wat vond u van de volledigheid van de uitleg?
◯ Slecht
◯ Onvoldoende
◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

19. Wat vond u van het taalgebruik van de uitleg?
◯ Slecht
◯ Onvoldoende
◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

20. Heeft u nog tips/opmerkingen voor en op de spel-uitleg?
.....
.....
.....
.....

Scenario 4

De gebruiker wilt een nieuw spel beginnen en spelen

21. Is het gelukt scenario 4 te volbrengen?
◯ Ja
◯ Nee, omdat: .....
.....

22. wat vind u van de snelheid van het spel?
◯ Slecht
◯ Onvoldoende
◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

23. wat vind u van het uiterlijk van het spel?

- ◯ Slecht
- ◯ Onvoldoende
- ◯ Neutraal/Geen mening
- ◯ Voldoende
- ◯ Goed

24. Wat vind u van de gemakkelijkheid van de besturing?

- ◯ Slecht
- ◯ Onvoldoende
- ◯ Neutraal/Geen mening
- ◯ Voldoende
- ◯ Goed

25. Wat vind u van de moeilijkheid van het spel in het begin?

- ◯ Slecht
- ◯ Onvoldoende
- ◯ Neutraal/Geen mening
- ◯ Voldoende
- ◯ Goed

26. Wat vind u van de moeilijkheid van het spel na een tijdje spelen?

- ◯ Slecht
- ◯ Onvoldoende
- ◯ Neutraal/Geen mening
- ◯ Voldoende
- ◯ Goed

27. Tot welke rang bent u gekomen en welke score(s) heeft u behaald? (Als u het niet meer weet kunt u in de score lijst kijken)
Rang:

- ◯ 
- ◯ 
- ◯ 
- ◯ 

Score: .....

28. Bent u deze vijand tegen gekomen?

◯ Ja
◯ Nee

29. Wat vond u van de snelheid van deze vijand?

◯ Slecht
◯ Onvoldoende
◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

30. Wat vond u van de kracht van deze vijand?

◯ Slecht
◯ Onvoldoende
◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

31. Wat vond u van de moeilijkheid om deze vijand te ontwijken?

◯ Slecht
◯ Onvoldoende
◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

32. Heeft u nog tips/opmerkingen voor deze vijand?

.....
.....
.....
.....

33. Bent u deze vijand tegen gekomen?

◯ Ja
◯ Nee

34. Wat vond u van de snelheid van deze vijand?

- ◯ Slecht
- ◯ Onvoldoende
- ◯ Neutraal/Geen mening
- ◯ Voldoende
- ◯ Goed

35. Wat vond u van de kracht van deze vijand?
    - ◯ Slecht
    - ◯ Onvoldoende
    - ◯ Neutraal/Geen mening
    - ◯ Voldoende
    - ◯ Goed

36. Wat vond u van de moeilijkheid om deze vijand te ontwijken?
    - ◯ Slecht
    - ◯ Onvoldoende
    - ◯ Neutraal/Geen mening
    - ◯ Voldoende
    - ◯ Goed

37. Heeft u nog tips/opmerkingen voor deze vijand?

    .....
    .....
    .....
    .....

38. Bent u deze vijand tegen gekomen?

    - ◯ Ja
    - ◯ Nee

39. Wat vond u van de snelheid van deze vijand?
    - ◯ Slecht
    - ◯ Onvoldoende
    - ◯ Neutraal/Geen mening
    - ◯ Voldoende
    - ◯ Goed

40. Wat vond u van de kracht van deze vijand?
    - ◯ Slecht

◯ Onvoldoende
◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

41. Wat vond u van de moeilijkheid om deze vijand te ontwijken?
◯ Slecht
◯ Onvoldoende
◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

42. Heeft u nog tips/opmerkingen voor deze vijand?
.....
.....
.....
.....

43. Bent u deze vijand tegen gekomen?

◯ Ja
◯ Nee

44. Wat vond u van de snelheid van deze vijand?
◯ Slecht
◯ Onvoldoende
◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

45. Wat vond u van de kracht van deze vijand?
◯ Slecht
◯ Onvoldoende
◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

46. Wat vond u van de moeilijkheid om deze vijand te ontwijken?
◯ Slecht
◯ Onvoldoende

◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

47. Heeft u nog tips/opmerkingen voor deze vijand?

.....
.....
.....
.....

48. Bent u deze vijand tegen gekomen?

◯ Ja
◯ Nee

49. Wat vond u van de snelheid van deze vijand?

◯ Slecht
◯ Onvoldoende
◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

50. Wat vond u van de kracht van deze vijand?

◯ Slecht
◯ Onvoldoende
◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

51. Wat vond u van de moeilijkheid om deze vijand te ontwijken?

◯ Slecht
◯ Onvoldoende
◯ Neutraal/Geen mening
◯ Voldoende
◯ Goed

52. Heeft u nog tips/opmerkingen voor deze vijand?

.....
.....
.....
.....

53. Heeft u nog tips/opmerkingen voor en op het spel zelf?

.....
.....
.....
.....

54. Zou dit spel zelf willen gebruiken?

◯ Ja
◯ Nee

55. Zou u voor dit spel betalen? Zo ja, hoeveel?

◯ Ja, maximaal: .....
◯ Nee

56. Heeft u nog vragen/opmerkingen over deze vragenlijst?

.....
.....
.....
.....

Dit was het einde van de vragenlijst.
Bedankt voor uw tijd en aandacht!