MKULTRA

APP

TeamRocket
Charlie Gerhardus s3050009
Sanne Boumans s3031926

18 mei 2012

Wat veranderd is na de eerste indiening:

- **De requirements zijn SMART gemaakt.**
- **Een actie toegevoegd in het Use case diagram**
- **Use cases zijn aangepast en uitgebreid**
- **Planning aangepast en wat specifieker gemaakt (voor zover dat kan)**
- **Spelling door iemand anders na laten kijken**

Inhoudsopgave

1. Inleiding

Het door ons ontworpen spel gaat zich afspelen in de ruimte. De gebruiker speelt in dit geval een vliegtuig wat als doel heeft zo lang mogelijk te overleven. 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.
Gedurende het spel zullen er verschillende moeilijkheidsgraden zijn en zal de gebruiker een beperkt aantal levens hebben om het spel zo uitdagender te maken. Tevens zullen er verschillende 'power-ups' te verkrijgen zijn (denk hierbij bijvoorbeeld aan raketten of extra levens).
Een van de dingen die dit spel uitdagend en onderscheidend maakt is het feit dat het de moeilijkheidsgraad waarin de speler speelt gaat onthouden waardoor de speler voor het volgende spel weer op dit niveau kan beginnen. Dit voorkomt dat de speler eerst dingen moet doen die te makkelijk zijn en zo wordt verveling dus tegen gewerkt.

Het spel zal met name bedoeld zijn voor personen die het leuk vinden om af en toe even snel een spelletje te spelen zonder dat ze gelijk vast zitten aan een bepaalde spelduur.
Het spel zal geschikt zijn voor alle leeftijden.

Uitgebreide spel uitleg

De gebruiker kan zijn vliegtuig in een bepaald gedeelte van het scherm bewegen. Hier kan hij zowel vooruit, achteruit en zijwaarts bewegen. Deze beweging is nodig om eventuele vijanden (en hun kogels) te ontwijken of om op vijanden te mikken.
Vijanden verschijnen telkens boven aan in het scherm en komen op de gebruiker af, het is aan de gebruiker om deze vijanden en hun kogels te ontwijken. Vijanden worden vernietigd als ze door de gebruiker neergeschoten zijn of als ze de gebruiker gepasseerd zijn (dus de onderkant van het scherm bereikt hebben).

In het spel zijn er ook verschillende moeilijkheidsgraden, oftewel 'ranks'. Nadat de gebruiker een bepaald aantal vijanden heeft vernietigd zal er één groot schip, het moederschip, verschijnen. Dit schip is moeilijker te verslaan dan de standaard vijanden omdat het meerdere levens heeft.
Als dit moederschip eenmaal verslagen is zal de gebruiker door gaan naar de volgende rank. (in totaal zijn er 4 ranks)

Per rank zijn er telkens twee verschillende wapens te verkrijgen. Deze wapens verschijnen in het spel in de gedaante van kisten die door de gebruiker opgepakt kunnen worden. Als de speler een leven verliest zal hij zijn huidige wapen verliezen en daarvoor het minst goede wapen van zijn huidige rank terug krijgen. Ook zijn er in de ranks kisten te vinden met doelzoekende raketten die het raken van een doel gemakkelijker maken.

Voor elke vijand die de speler vernietigd zal de speler punten krijgen (je krijgt meer punten als je in een hogere rank zit). Deze punten worden dan gebruikt om de highscore te berekenen.

De speler heeft in het begin van het spel standaard 3 levens, maar ook deze zijn op te hogen door extra levens die in kisten te vinden zijn. Uiteraard verliest de gebruiker een leven als hij geraakt wordt door een vijand.

2. Requirements

Hieronder volgen de 'requirements', oftewel eisen, waaraan het systeem moet voldoen. Als het systeem uiteindelijk niet aan deze voorwaarden voldoet dan is het spel niet klaar voor produceren.

2.1. Functionele eisen

De functionele eisen geven aan wat het systeem allemaal moet kunnen. De functionele eisen zijn zichtbaar voor de gebruiker en hebHet logboek bij elkaar voegen helemaal en in verslag vorm opschrijvenben dan ook direct invloed op het gebruik van het systeem.
Het systeem moet het volgende kunnen doen:

Weergave

- Vliegtuig van speler wordt weergegeven als het prototype klaar is (05-06-2012).
- Meerdere vliegtuigen van de vijanden worden weergegeven als het prototype klaar is (05-06-2012).
- Vliegtuig van eventuele eindbaas wordt weergegeven als de complete app af is (± 20-05-2012).
- De achtergrond wordt weergegeven waarvan random wordt bepaald wat waar getekend wordt als de uiteindelijke app af is (± 20-05-2012).
- Eventuele power-ups worden weergegeven als het prototype klaar is (05-06-2012).
- Afgevuurde kogels worden weergegeven als het prototype klaar is (05-06-2012).
- Standaard het aantal levens in het scherm weergeven als het prototype klaar is (05-06-2012)
- Standaard de moeilijkheidsgraad in het scherm weergeven als het prototype klaar is (05-06-2012).
- Standaard de score van de gebruiker in het scherm weergeven als het prototype klaar is (05-06-2012).
- Standaard het aantal munitie van de gebruiker in het scherm weergeven als het prototype klaar is (05-06-2012).

Besturing

- Vliegtuig van gebruiker kan bewegen als het prototype klaar is (05-06-2012).
- Gebruiker kan raket afschieten als de uiteindelijke app af is (± 20-05-2012).
- Gebruiker moet kunnen schieten als het prototype klaar is (05-06-2012).
- Er is een vaste ruimte waarin de gebruiker kan vliegen als de uiteindelijke app af is (± 20-05-2012).

Gebruiker

- Het spel moet op pauze kunnen als het prototype klaar is (05-06-2012).
- High-score moet bewaard worden als het prototype klaar is (05-06-2012).
- Moeilijkheidsgraad moet bewaard worden als de uiteindelijke app af is (± 20-05-2012).
- Spel kan vervolgen op de bewaarde moeilijkheidsgraad als de uiteindelijke app af is (± 20-05-2012).
- Spel moet op ieder moment afgesloten kunnen worden door de gebruiker als het prototype

klaar is (05-06-2012).
- Moeilijkheidsgraad kan aangepast worden door de gebruiker als de uiteindelijke app af is (± 20-05-2012).

In-game

- Power-ups kunnen worden opgepakt als de uiteindelijke app af is (± 20-05-2012).
- Aantal levens kan opgehoogd worden met +1 als de uiteindelijke app af is (± 20-05-2012).
- Vijand kan schieten als het prototype klaar is (05-06-2012).
- Gebruiker kan geraakt worden waardoor er levens af gaan als het prototype klaar is (05-06-2012).
- Vijand kan geraakt worden en van scherm verdwijnen als het prototype klaar is (05-06-2012).
- Eindbaas kan kanonnen afvuren als de uiteindelijke app af is (± 20-05-2012).
- Eindbaas heeft meerdere levens als de uiteindelijke app af is (± 20-05-2012).
- Eindbaas verdwijnt van scherm (als zijn levens op zijn) als de uiteindelijke app af is (± 20-05-2012).
- Vijand verdwijnt van scherm als hij de gebruiker gepasseerd is als de uiteindelijke app af is (± 20-05-2012).
- Eindbaas blijft bovenin het scherm totdat hij vernietigd is als de uiteindelijke app af is (± 20-05-2012).

Menu

- Speler kan een nieuw spel beginnen als het prototype klaar is (05-06-2012).
- Speler kan high-scores bekijken als de uiteindelijke app af is (± 20-05-2012).
- Speler kan verder gaan met een gepauzeerd of opgeslagen spel als het prototype klaar is (05-06-2012).
- Uitleg over spelbesturing in beschikbaar voor gebruiker als de uiteindelijke app af is (± 20-05-2012).

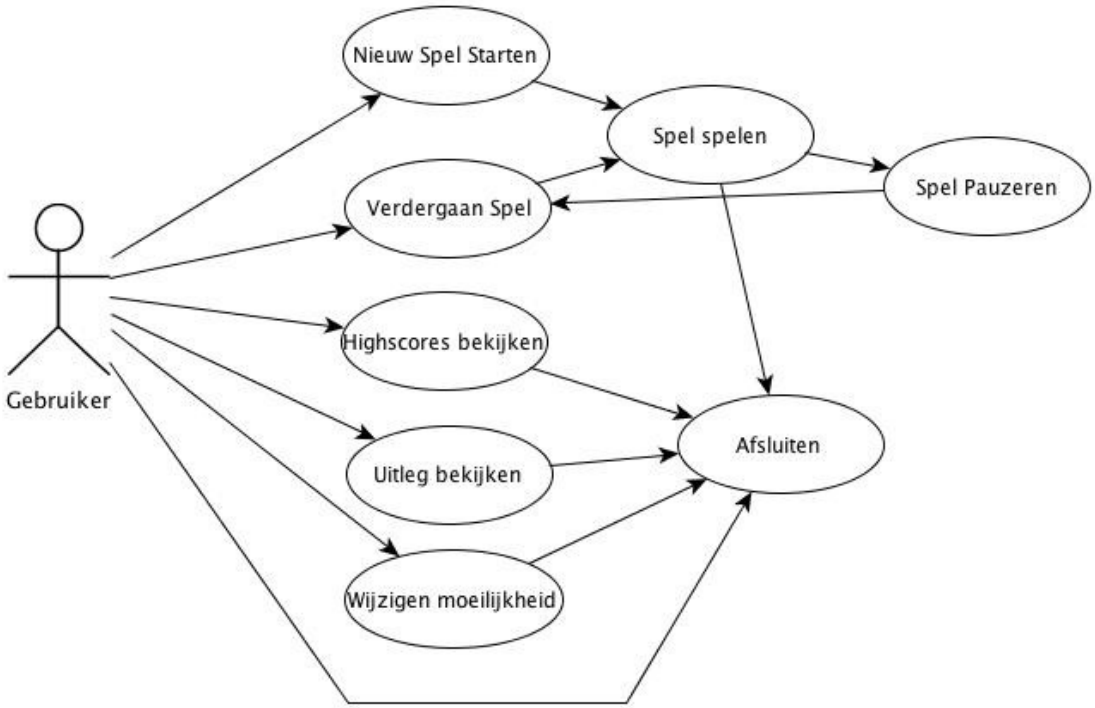
2.2. Niet-Functionele eisen

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.

- Spel is aan het eind van het hele proces (± 20-05-2012) ontworpen op zelf gemaakte game-engine.
- Aan het eind van het proces (± 20-05-2012) zijn alle vliegtuigen in de game zelf ontworpen en getekend.
- Aan het eind van het hele proces (± 20-05-2012) is het lettertype zelf ontworpen en getekend.
- Aan het eind van het hele proces (± 20-05-2012) is de achtergrond zelf ontworpen en getekend.
- Binnen 35 ms na aanraking van de gebruiker reageert het spel in de uiteindelijke app (± 20-05-2012).

- Onderhoud van het systeem moet makkelijk zijn en updates kunnen door de gebruiker worden gedaan zonder dat dit langer dan 2 minuten duurt.

2.3. Use Case Model



2.4. Use Cases

Hieronder staan de mogelijke acties van de gebruiker beschreven.
Use cases 5 t/m 8 zijn niet heel uitgebreid. Dit komt omdat onze app relatief weinig mogelijke acties heeft. Zo kan de gebruiker maar enkele dingen in het menu doen (te zien in de use case) en die dingen hebben vaak geen sub-acties.

UC nummer	1
Naam	Nieuw Spel Starten
Omschrijving	Het starten van een nieuw spel
Actor	Gebruiker
Trigger	De gebruiker drukt op de knop voor een nieuw spel
Basis flow	
Stap	**actie**
1. 1	De gebruiker wordt gevraagd of hij zeker weet dat hij met een nieuw spel wilt beginnen en niet met het opgeslagen spel verder wilt gaan. Als er voor wordt gekozen om een nieuw spel te starten dan wordt stap 2 uitgevoerd.
2	Het spel start op en use case 3 wordt gestart.
	Alternatieve flow
stap	**actie**
1.2	De gebruiker kiest er toch niet voor om een nieuw spel te starten en wil verder met het opgeslagen spel, use case 2 wordt gestart.
3	Onderbreking van het opstarten omdat de gebruiker naar het hoofdmenu terug gaat
precondities	Gebruiker is in het hoofdmenu van de app
postcondities	Er is een spel begonnen
opmerkingen	-

UC nummer	2
Naam	Verdergaan Spel
Omschrijving	Opgeslagen spel wordt opgestart
Actor	Gebruiker
Trigger	De gebruiker drukt op de knop voor verdergaan met spel
Basis flow	
Stap	**actie**
1.1	Het spel start op en use case 3 wordt gestart.
	Alternatieve flow
stap	**actie**
1.2	Er is geen opgeslagen spel en dit wordt aan de gebruiker verteld
1.3	Er wordt gevraagd of de gebruiker een nieuw spel wil starten
1.3.1	De gebruiker kiest ja en er wordt een nieuw spel gestart en verder gegaan met use case 3

1.3.2	De gebruiker kiest nee en wordt terug gestuurd naar het hoofdmenu van de app
2	Onderbreking van het opstarten omdat de gebruiker naar het hoofdmenu terug gaat
precondities	De gebruiker is in het hoofdmenu of in stap 1.1 van UC 1
postcondities	Er is een spel begonnen
opmerkingen	-

UC nummer	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
Basis flow	
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	-

UC nummer	4
Naam	Spel Pauzeren
Omschrijving	Het spel wordt gepauzeerd
Actor	Gebruiker
Trigger	Tijdens een spel drukt de gebruiker op de hoofdmenu knop
Basis flow	
Stap	**actie**
1.1.1	Het spel wordt opgeslagen en de gebruiker gaat terug naar het hoofdmenu
1.2	De speler kan verder gaan met dit spel door op de goede knop te duwen. UC2 gaat van start
	Alternatieve flow
stap	**actie**
1.1.2	Spel kan niet worden opgeslagen door te weinig geheugen. Gebruiker krijgt hiervan een notificatie
1.1.3	Tijdens het spel opslaan wil de gebruiker de app sluiten
1.1.3.1	Gebruiker krijgt keuze te zien of hij het zeker weet
1.1.3.1.1	Gebruiker kiest ja, de app wordt afgesloten en er wordt niks opgeslagen
1.1.3.1.2	Gebruiker kiest nee en stap 1.1.1 wordt uitgevoerd
precondities	De speler zit in een spel
postcondities	Het spel is opgeslagen
opmerkingen	-

UC nummer	5
Naam	Higscores bekijken
Omschrijving	De highscores van de gebruiker worden weergegeven
Actor	Gebruiker
Trigger	De gebruiker drukt op de knop om higscores te laten zien
Basis flow	
Stap	**actie**
1.	De hoogste 10 scores van de gebruiker worden weergegeven
	Alternatieve flow
stap	**actie**
2	Onderbreking van het bekijken omdat de gebruiker naar het hoofdmenu terug gaat
precondities	Gebruiker is in het hoofdmenu van de app

postcondities	Lijst met highscores is weergegeven
opmerkingen	Deze UC heeft weinig stappen omdat er niet veel acties ondernomen kunnen worden

UC nummer	6
Naam	Uitleg bekijken
Omschrijving	De uitleg van het spel wordt weergegeven
Actor	Gebruiker
Trigger	De gebruiker drukt op de knop om de uitleg te laten zien
Basis flow	
Stap	**actie**
1.	De uitleg van het spel wordt weergegeven
	Alternatieve flow
stap	**actie**
2	Onderbreking van het bekijken omdat de gebruiker naar het hoofdmenu terug gaat
precondities	Gebruiker is in het hoofdmenu van de app
postcondities	Uitleg van het spel is weergegeven
opmerkingen	Deze UC heeft weinig stappen omdat er niet veel acties ondernomen kunnen worden

UC nummer	7
Naam	Wijzigen moeilijkheid
Omschrijving	De opgeslagen moeilijkheidsgraad wordt gewijzigd
Actor	Gebruiker
Trigger	De gebruiker drukt op de knop om de moeilijkheidsgraad te wijzigen
Basis flow	
Stap	**actie**
1.	De gebruiker krijg te zien in welke moeilijkheidsgraad hij nu zit
2.1	De gebruiker kiest een andere moeilijkheidsgraad
2.1.1	De app vraagt of de gebruiker het zeker weet. Zo ja dan wordt de gebruiker terug geleid naar het hoofdmenu. Zo nee dan start UC 7 opnieuw
	Alternatieve flow

stap	**actie**
3	Onderbreking van het wijzigen omdat de gebruiker naar het hoofdmenu terug gaat
precondities	Gebruiker is in het hoofdmenu van de app
postcondities	Moeilijkheidsgraad is gewijzigd
opmerkingen	Deze UC heeft weinig stappen omdat er niet veel acties ondernomen kunnen worden

UC nummer	8
Naam	Afsluiten
Omschrijving	De app wordt afgesloten
Actor	Gebruiker
Trigger	De gebruiker drukt op de menu knop van zijn telefoon
Basis flow	
Stap	**actie**
1.1	Het huidige spel wordt opgeslagen en de app wordt afgesloten
	Alternatieve flow
stap	**actie**
1.2.1	Spel kan niet worden opgeslagen door te weinig geheugen. Gebruiker krijgt hiervan een notificatie
1.2.2	Speler kan kiezen of hij toch de app wilt afsluiten. Zo ja wordt de app afgesloten en het spel opgeslagen. Zo nee dan wordt de gebruiker terug gestuurd naar het hoofdmenu
precondities	-
postcondities	App is afgesloten
opmerkingen	Deze UC heeft weinig stappen omdat er niet veel acties ondernomen kunnen worden

3. Ontwerp

3.1. 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 creëren 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()	Initialiseerd OpenGL ES voor de huidige thread.
JavaLink.opengl_stop()	Deinitialiseerd OpenGL ES voor de huidige thread.
JavaLink.opengl_toon()	Wissel 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. Zo gauw 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 spelobjecten 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 spelobjecten 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 overgeërfd, 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, creëert alle spelobjecten en maak deze zichtbaar.
spelobjecten	knoppen, speler, vijanden, laser stralen...

3.2. Gegevens ontwerp

Voor spelobjecten beheert spelmotor de standaard gegevens zoals positie, rotatie, snelheid... De hier genoemde gegevens zijn de voor het object specifieke gegevens.

Gegevens op het scherm:
- levens
- munitie doelzoekende raket
- rank (moeilijkheidsgraad)
- score

Gegevens voor de speler:
- levens
- wapen

Gegevens per vijand:
- levens
- wapen

Gegevens voor kisten:
- inhoud

Gegevens voor wapens:
- soort munitie

Gegevens voor doelzoekende raketten:
- doel om te zoeken

3.3. Detail-ontwerp per component

Klasse-omschrijving

Spelmotor

Klasse	Omschrijving
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. Zo gauw 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 overgeërfd 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 start() aangeroepen worden. Daarna zal elke stap ververs() aangeroepen worden en wanneer de applicatie gesloten wordt zal stop() aangeroepen worden.
motorafbeelding	Een afbeelding die door gegeven 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 vertegenwoordigd. Deze klasse kan als vader opgegeven worden voor spelobjecten om ze zo aan de wereld toe te voegen.
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 geïmporteerde methode zijn te bereiken via deze klasse.
cpp_link	Bevat alle geëxporteerde C functies.

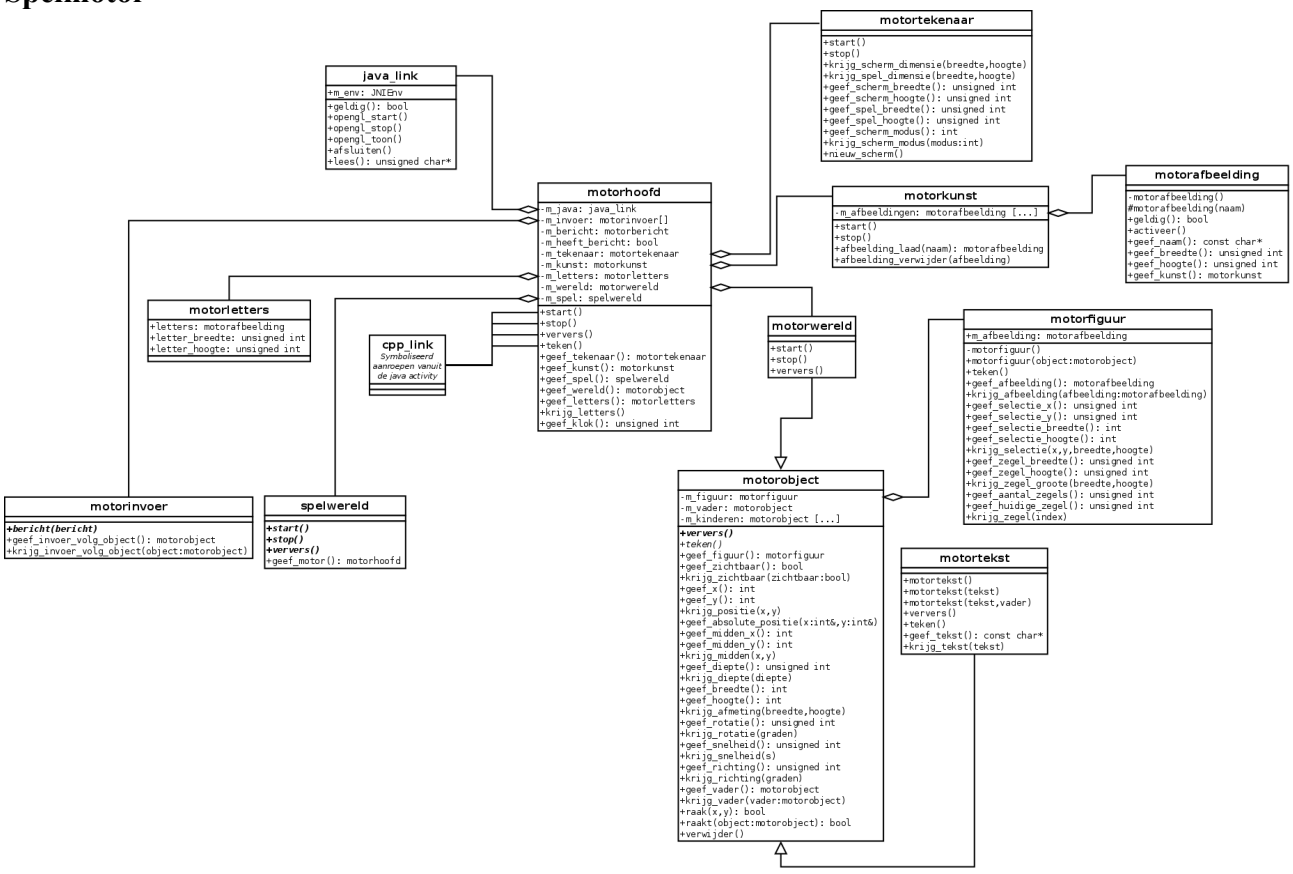
Spel

Klasse	Omschrijving
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.
toestand_menu	Het menu waar vanuit alle spel onderdelen toegankelijk zijn.
toestand_help	Toont informatie over het gebruik van de applicatie.
toestand_hiscore	Toont een lijst met de hoogst behaalde scores.
toestand_spel	De daadwerkelijke spel toestand. Deze toestand beheert de speler en zijn tegenstanders.
achtergrond	Dit spelobject tekent de achtergrond welke bestaat uit voorbij razende sterren. Deze achtergrond is in elke toestand zichtbaar maar enkel in de spel toestand zullen de sterren daadwerkelijk bewegen.
ster_1,2...	Verschillende sterren die in de achtergrond kunnen voorkomen.
knop	Een spel object waarop gedrukt kan worden waarna geef_was_ingedrukt() true returned, na deze

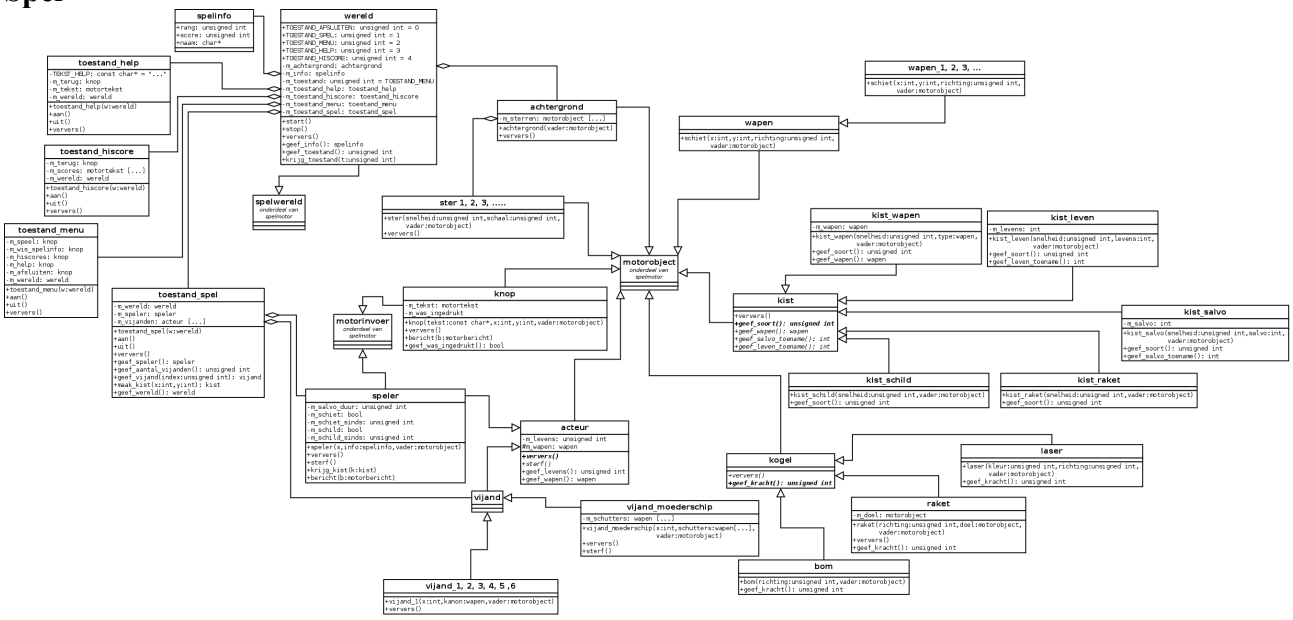
	aanroep zal de flag hersteld worden totdat er nogmaals op de knop gedrukt word.
kist	Een basis klasse voor alle opraap-bare objecten.
kist_xxx	Een opraap-baar item dat de speler beïnvloed.
kogel	Een basis klasse voor objecten die de speler of vijanden schade kunnen toebrengen.
laser, raket, bom	Verschillende 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.
vijand	Basis klasse voor vijandige objecten.
vijand_x	Een vijand, verschillende vijanden hebben verschillende wapens en aanvlieg routes.
vijand_moederschip	Alvorens we een rang omhoog gaan moeten we een vijand van dit type vernietigen.
acteur	Basis klasse voor alle 'levende' spel objecten.
wapen	Basis klasse voor wapens. De schiet methode hoort een kogel object te maken.
Wapen_1, 2, 3...	De individuele wapens.

Klasse-diagrammen

Spelmotor

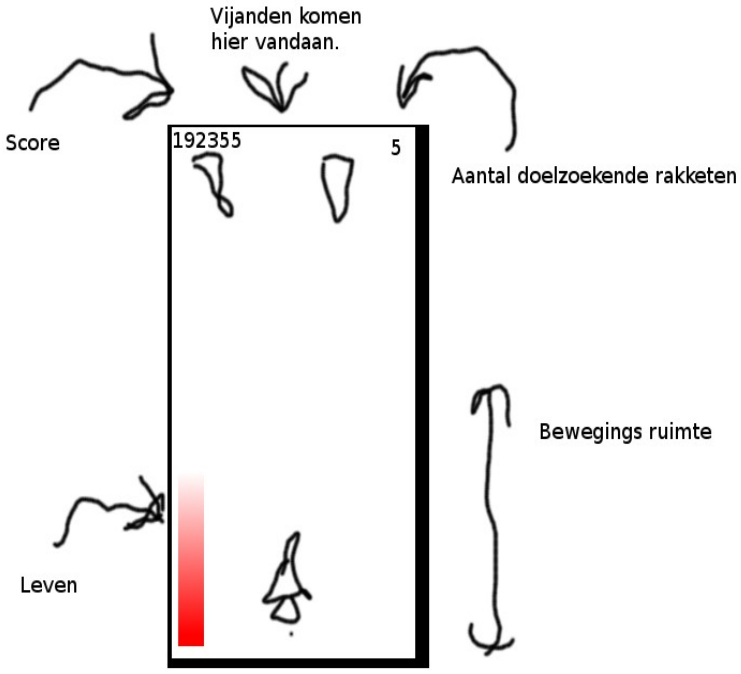


Spel



3.4. Gebruikers interface

In bepaalde rangen komen bepaalde wapens beschikbaar. Je kunt dan van wapen wisselen door andere kisten te pakken die op het scherm verschijnen. Als je een leven verliest verlies je ook je huidige wapen en krijg je daar het laagste wapen van de rang waarin je zat voor terug (dus niet het laagste wapen van het hele spel).



Besturing:

- speler slepen – bewegen
- speler drukken – schiet voor bepaalde tijd afhankelijk van verkregen kisten.
- Vijand drukken – indien voldoende munitie schiet doelzoekende raket op tegenstander.

Plaatjes:

- Speler
- Vijanden
- Eindbaas (moederschip)
- Lasers
- Raket
- Sterren
- Explosie
- Kisten
- Letters
- Planeten
- Schild
- Moeilijkheidsgraden/ranking

4. Planning

Zondag 3 Juni	Moederschip is getekend Score wordt berekend In het logboek is over het tekenen wat geschreven
Maandag 4 Juni	10.30uur afspreken om de presentatie van het prototype in elkaar te zetten
Dinsdag 5 Juni	Presentatie van het prototype
Woensdag 6 Juni	10.30uur afspreken om usability-testen te maken
Donderdag 7 Juni	Mensen usability test laten uitvoeren
Vrijdag 8 Juni	Mensen usability test laten uitvoeren
Zaterdag 9 Juni	Alle power ups en extra wapens zijn getekend
Zondag 10 Juni	Extra wapens en power ups kunnen gebruikt worden
Maandag 11 Juni	10.30 uur afspreken om samen te vatten wat in usability tests naar voren kwam en een lijst maken met wat veranderd moet worden. Zet dit in het logboek
Dinsdag 12 Juni	De rangen werken helemaal goed en zijn aanpasbaar
Woensdag 13 Juni	Uitslag van usability tests verwerken en aanpassen
Donderdag 14 Juni	Uitslag van usability tests verwerken en aanpassen Kijken wat van t ontwerpdocument behaald is en dit opschrijven
Vrijdag 15 Juni	Uitslag van usability tests verwerken en aanpassen Het logboek bij elkaar voegen helemaal en in verslag vorm opschrijven
Zaterdag 16 Juni	Eindverslag maken
Zondag 17 Juni	Eindverslag maken
	Alles hierna is extra ruimte voor de laatste aanpassingen