

14

**Technisch Verslag**

**- Shoot The Bully**

**Arnout Reitsma**

**Ilias Ameziane**

**Tim Hasselaar**

**Hylco Uding**

**Daniël van den Berg**

**Inhoudsopgave**

[Opdrachtomschrijving - 2 -](#_Toc403298608)

[Ons product - 3 -](#_Toc403298609)

[Inleiding - 4 -](#_Toc403298610)

[Ontwikkelomgeving - 5 -](#_Toc403298611)

[1. Visual Studio 2013 - 5 -](#_Toc403298612)

[2. SFML 2.1 - 5 -](#_Toc403298613)

[3. Github - 5 -](#_Toc403298614)

[4. Google Drive - 5 -](#_Toc403298615)

[5. Microsoft Office - 5 -](#_Toc403298616)

[6. Paint.net - 5 -](#_Toc403298617)

[7. Software Ideas Modeler - 5 -](#_Toc403298618)

[Shoot The Bully - 6 -](#_Toc403298619)

[1. Regels - 6 -](#_Toc403298620)

[2. Core-mechanics - 6 -](#_Toc403298621)

[3. Wapens - 7 -](#_Toc403298622)

[54 Design van de game - 7 -](#_Toc403298623)

[De map - 7 -](#_Toc403298624)

[Menus - 8 -](#_Toc403298625)

[Gamebesturing - 8 -](#_Toc403298626)

[Geluid - 8 -](#_Toc403298627)

[Gebruikersmodellen - 10 -](#_Toc403298628)

[1. Game sequence - 10 -](#_Toc403298629)

[2. Klassendiagram - 11 -](#_Toc403298630)

[3. Activity diagram - 12 -](#_Toc403298631)

[Testrapporten - 13 -](#_Toc403298632)

[Conclusie - 13 -](#_Toc403298633)

[Bronvermelding - 13 -](#_Toc403298634)

# Opdrachtomschrijving

In opdracht van de Hogeschool Utrecht is ons, Scrumbag Games, gevraagd om een 2D game te maken. Om deze applicatie te programmeren moeten wij gebruik maken van de programmeer taal C++, en de SFML 2.1 libraries. De game moet ongeveer 10 minuten speeltijd leveren, en voor het onderwerp van de game zijn we volledig vrij gelaten.

## Ons product

Wij hebben als team gekozen om de game 'Shoot The Bully' te maken. De speler bestuurd in deze game een character in een afgesloten rechthoekige ruimte. Het spel bestaat uit een aantal rondes van twee minuten. Gedurende deze twee minuten zullen er oneindig veel vijanden op de speler af komen. De speler moet met verscheidene wapens de vijanden van zich af zien te houden. Aan het eind van elke ronde van twee minuten komt er een baas van de ronde, welke de speler moet verslaan om verder te komen in het verhaal.

# Inleiding

In dit technische verslag worden alle aspecten van onze game gedocumenteerd. We zullen beginnen met welke programma's we hebben gebruikt voor het ontwikkelen van de game. Daar na gaan we alle eigenschappen van de game zelf uitlichten. Hier bij zullen we het bijvoorbeeld ook hebben over alle items uit de game, de opbouw van de map, en de besturing. We zullen ook nog de use-cases, het klassendiagram en het flow-diagram bespreken. Verder komen nog alle testrapporten en onze conclusie.

# Ontwikkelomgeving

Hier zullen we alle programma's uitlichten die we hebben gebruikt bij het ontwikkelen van onze game. Hier zal vooral worden besproken waar het programma vandaan komt, en waarvoor het primair is gebruikt.

## 1. Visual Studio 2013

Visual Studio is een programmeer omgeving voor C++. Via onze school hebben wij een gratis versie kunnen bemachtigen. In dit programma is alle code van onze game geschreven.

## 2. SFML 2.1

De SFML 2.1 libraries zijn open-source, en dus gratis, libraries. Deze libraries kunnen worden gecombineerd met C++ in Visual Studio om op een simpele en effectieve manier het grafische deel van de game te programmeren.

## 3. Github

Github is een service waarmee we online al onze code kunnen delen. Door middel van de online synchronisatie techniek van Github kunnen we makkelijk op meerdere computers aan dezelfde code werken. Deze manier is veel handiger dan bijvoorbeeld elke wijziging via de mail door te sturen naar elkaar.

## 4. Google Drive

Net als Github is Google Drive een manier om online bestanden te delen. We gebruiken Google Drive vooral om onze documentatie te delen. Google Drive is hier wat handiger voor omdat meerdere gebruikers op hetzelfde moment aan hetzelfde document kunnen werken.

## 5. Microsoft Office

Word en Excel zijn de twee voornaamste programma's die we gebruiken uit het microsoft office pakket. Word wordt bijvoorbeeld gebruikt om onze documenten als het Game Design te schrijven. Excel is bijvoorbeeld gebruikt om al onze items en attributen van deze items op een overzichtelijke manier op te slaan.

## 6. Paint.net

Paint.net is in principe een uitgebreidere versie van het alom bekende paint. Het voegt verschillende functies toe die handig zijn bij het bewerken van afbeeldingen. Het is dan wel niet net zo uitgebreid als bijvoorbeeld photoshop, maar het is ideaal voor onze doeleinden en bovendien een gratis applicatie

## 7. Software Ideas Modeler

Dit programma is gebruikt om bijvoorbeeld onze use-cases en klassendiagrammen te maken. Het is een gratis programma gespecialiseerd in deze taken.

# Shoot The Bully

In dit hoofdstuk worden alle eigenschappen van de game belicht. We zullen beginnen met de verhaallijn, en de core-mechanics en regels van de game. We zullen ook alle enemies, bullies en powers uit de game beschrijven, samen met alle bijbehorende attributen.

## 1. Regels

De regels van het spel houden in wat de speler wel en/of niet mag doen. De regels die wij voor onze game hebben vastgesteld zijn:

* De speler mag maar 3 wapens bij zich hebben, een (ge-upgrade) mes en 2 wapens die de speler in de shop heeft gekocht.
* Als de speler dood gaat moet hij de laatste wave over doen.
* De speler kan niet meer geld uitgeven dan hij heeft.
* De speler kan niet het spel starten zonder dat hij de tutorial heeft gezien.
* De speler kan niet specifieke levels opnieuw doen, alleen weer bij level 1 beginnen.
* De speler kan alleen schieten met wapens als hij daar genoeg ammunitie voor heeft.
* De speler kan niet door muren heen lopen.
* De speler kan alleen vanaf een bank op een tafel klimmen.

## 2. Core-mechanics

Het spel moet zich ook aan een aantal regels houden. Deze regels noemen we de core-mechanics. De core-mechanics van onze game zijn:

* Het spel is een top down shooter. De speler bevind zich in een afgesloten rechthoekige ruimte (een kantine van een middelbare school),
* De speler kan met de muis zijn wapen mikken en met de WASD, of pijltjestoetsen, rondlopen met zijn character.
* De map heeft meerder aparte objecten die vernietigd kunnen worden (denk aan tafels die door midden breken of borden die kapot vallen in scherven).
* Bullies spawnen net buiten het scherm.
* Bullies lopen altijd recht op je af. Als er zich een obstakel bevind tussen de bully en de speler, dan loopt de bully een aantal pixels opzij. Net zo lang tot dat er zich geen obstakel meer tussen hem en de speler zit.
* Elke ronde duurt 2 minuten. In deze twee minuten spawnen oneindig veel bullies, met een maximum aantal tegelijk op het scherm.
* Het maximum aantal bullies tegelijk op het scherm word bepaald door de ronde waar in de speler zich bevind, en de moeilijkheid waar op de speler het spel speelt.
* Elke ronde komen er sterkere bullies (dit houdt in: meer levenspunten en een hogere loopsnelheid).
* Sommige bullies laten een random gekozen power vallen als ze dood gaan.
* Als een bully wordt geraakt door een kogel neemt zijn hp af.
* Een wapen mag alleen worden herladen als het magazijn van het wapen niet meer vol zit.
* Aan het einde van een wave kan je in de shop nieuwe wapens kopen of bestaande wapens upgraden.
* Als je een bully neer schiet (of slaat) krijg je jouw gestolen lunchgeld terug ( willekeurig tussen de 50 en 100 euro).
* De speler kan niet door obstakels heen lopen.
* De speler kan op lage obstakels (zitbanken) klimmen.
* De speler kan via een bankje op een tafel klimmen.

## 3. Wapens

Voor de wapens hebben wij gekozen uit een selectie melee wapens en geweren. We hebben ervoor gekozen 6 verschillende wapens in de game te stoppen. Elk wapen heeft zijn eigen voor- en nadelen.

Voor de melee wapens geldt bijvoorbeeld dat ze oneindige ammunitie hebben en veel damage doen, maar ze kunnen alleen gebruikt worden op een hele korte afstand. Tussen de twee melee wapens zit ook nog het verschil dat de knife de kortste afstand heeft, maar sneller attackt. Het zwaard heeft een iets langere range maar slaat langzamer.

Voor de geweren is er wat meer diversiteit. Je hebt bijvoorbeeld het pistool, welke weinig damage doet maar oneindige ammunitie heeft. Als andere wapen in dit slot kun je kiezen voor de assault rifle, welke een all round wapen is. Hij heeft een hoge fire rate, redelijke damage en veel kogels. Als de speler comfortabel is met het spelen met een wapen zonder oneindig ammunitie is de assault rifle een heel sterk wapen.

Voor het derde wapen slot kan de speler kiezen uit twee wapens met hoge damage. Deze zijn de shotgun en de sniper. De sniper zoomt het beeld van de speler uit en heeft de hoogste damage, en de shotgun heeft meer kogels per magazijn. Dus ook al zijn de wapens beide geschikt om veel damage te doen in weinig schoten, toch is er een redelijke afweging tussen de twee.

## 4. Design van de game

Wij hebben gekozen om de game weer te geven op een scherm van 640 bij 480 pixels. De standaard grootte van game objecten zijn 32 bij 32 pixels. Deze grootte geld bijvoorbeeld voor het character van de speler en vijandelijke bullies.

### De map

De map van het spel zal de kantine van een middelbare school zijn. Het is een rechthoekige kamer die 4 maal zo groot is als het scherm waar op de speler het spel speelt, dus twee maal de hoogte en twee maal de breedte.

Elke ronde word op dezelfde map gespeeld. Toch wilde we dat er enige variatie zat in het spel met betrekking tot de map. Daarom hebben we besloten elke ronde de map kleine beetjes te veranderen.

De eerste ronde zal er een rode gloed over de map hangen die de woede van Marcel weergeeft. Alle objecten zoals bijvoorbeeld de tafels zullen in het eerste level ook heel erg scheef staan. Hoe verder de speler komt met de rondes, hoe rustiger de map word. Dit houdt in dat de rode gloed weg trekt en dat alle objecten netter worden neergezet. In de laatste ronde zullen dus alle tafels en bankjes volledig recht staan.

De map word omringd aan alle vier de kanten door muren, en er zal in elke hoek een pilaar staan. De zaal zal gevuld worden met tafels en bankjes en dergelijke objecten die je in een kantine kunt vinden. Het midden van de map zal vrij worden gehouden, zodat de speler daar elke ronde kan beginnen.

### Menus

Als de speler zich in een menu van de game bevind zal de achtergrond bestaan uit de map, waar de camera willekeurig over heen beweegt. Ook zullen er een aantal knoppen worden weer gegeven afhankelijk van in welk menu de speler zich bevind.

De menu's zullen bestaan uit:

* Het main menu; waar de speler naar de opties kan, en het spel kan starten..
* Het opties menu; waar de speler bijvoorbeeld het volume kan veranderen, of het bloed aan/uit kan zetten.
* Het in-game menu; in het in-game menu word het spel gepauzeerd, en kan de speler het spel opnieuw starten, of terug naar het main menu.

### Gamebesturing

Voor de besturing van onze game hebben we gekozen voor een vrij standaard schema voor top-down shooters. De speler kan zich in het spel rond bewegen met beide de W, A, S en D knoppen, of gebruik maken van de pijltjes toetsen.

De speler kan zijn wapen mikken met de muis, en het wapen gebruiken door met de muis te klikken. Ook kan de speler zich door de menu's navigeren met behulp van de muis.

Ook kan de speler tussen zijn gekozen wapens kiezen met de knoppen 1, 2 en 3. Voor het pauzeren van het spel kan de speler de escape knop, of de letter 'P' gebruiken.

### Geluid

Voor onze game gebruiken we achtergrond muziek, welke vanaf het opstarten vanaf de game zal draaien. Ook zullen we schietgeluiden gebruiken voor alle geweren, en melee wapens. Ook word er een geluidje geimplementeerd voor het moment dat de speler gewond raakt.

# Code opbouw

We hebben geprobeerd onze code zo overzichtelijk mogelijk te houden, en zullen daarnaast ook nog in dit hoofdstuk een aantal keuzes toelichten. Ook zullen we documenteren wat de verantwoordelijkheden van bepaalde klassen zijn.

## Klassenverantwoordelijkheden

Voor het opstarten van de game zijn vooral de volgende klassen belangrijk:

* Main
  + In de Main worden de instances van de LevelController, SoundController en Hudcontroller aangemaakt. Ook wordt de instance van de GameController aangemaakt, en wordt door het aanroepen van een functie hierin de game gestart. Op deze manier is het gehele spel in één keer geladen.
* LevelController
  + In de LevelController worden alle levels geladen en worden alle GameObjecten bijgehouden. Dezen worden vanaf hier dus ook geupdated, gemoved en gedrawt. Door dit bij elkaar te houden kunnen er gemakkelijk objecten en features worden toegevoegd.
* Logo
* GameController
  + In de GameController wordt het spel bestuurd. Dit betekend dus dat hierin het gameWindow en de FPS bijgehouden en bestuurd. Er is gekozen om deze controller aan te maken omdat deze een laag vormt tussen het spel (de gameObjecten) en de desktop.

Daarna, in het main menu, komen de volgende klassen aan bod:

* Checkbox
  + ToggleGore
  + ToggleMusic
  + ToggleSound
  + De Checkbox klasse is een GameObject voor op de HUD waarop geklikt kan worden om een setting aan of uit te zetten. Deze Checkbox laad zijn waarde uit en slaat zijn waarde op via de SettingsController.
* Clickable
  + De clickable klasse is de superklasse voor alle GameObjecten waarop geklikt kan worden. Door een virtual function click() te overriden kan een child acties uitvoeren als er op de clickable geklikt wordt. De clickable handelt het berekenen van de positie van de muis en de input van de muisknoppen af.
* MainMenu
  + MainMenu is een button waarop geklikt kan worden om naar het Main Menu level te gaan.
* Options
  + Options is een button waarop geklikt kan worden om naar het options level te gaan.
* Tutorial
  + Tutorial is een button waarop geklikt kan worden om naar het tutorial level te gaan.
* TutorialScreen
  + In het tutorial screen word door middel van tekst het verhaal van onze game weergegeven. Ook word de besturen van de game aan de speler uitgelegd.
* TextureManager
  + De textureManager is de klasse die er voor zorgt dat alle textures ingeladen worden, en dat ze ook maar één keer ingeladen moeten worden. Hij slaat alle textures op in een map om deze later, indien nodig, aan gameObjecten door te geven.
* Slider
  + SoundSlider
  + MusicSlider
* SettingsController
  + De settingsController is de controller die er voor zorgt dat alle settings ingeladen en opgeslagen kunnen worden. Deze controller zit dus tussen de gameObjecten en de savefile. Er is voor gekozen om de settings op te slaan in een plaatje, omdat dit, in tegenoverstelling tot een tekstfile, geen zoekfuncties vereist. In plaats daarvan wordt een bepaalde setting met een bepaalde pixel geassociëerd.
* Play
  + Play is een button waarop geklikt kan worden om het spel te beginnen.

Wanneer de game op start beginnen de volgende klassen te werken:

* Animation
  + De animation klasse zorgt ervoor dat animaties in onze game mogelijk worden. Hij laad alle benodigde plaatjes voor een bepaalde animatie af, en laat deze op volgorde afspelen als het hem word gevraagd.
* Bench
  + De bench klasse is voor alle banken in ons spel. Ze doen eigenlijk niet heel veel bijzonders naast het staan in de map. De speler kan de banken gebruiken om op tafels te klimmen.
* Collision
  + De collision klasse kan worden gebruikt om te kijken of bepaalde game objecten elkaar raken. Hij kan tussen bijvoorbeeld twee objecten, of een object en een lijn uitrekenen wat de twee dichtst bij elkaar liggende punten zijn, en of deze objecten elkaar raken.
  + Dit word bijvoorbeel gebruikt om te kijken of een speler beide een bankje en een tafel raakt als hij een tafel op wil klimmen.
* Exception
  + De exception klasse word gebruikt om onze eigen exceptions te kunnen gooien in onze console. Dit werd vooral gebruikt bij het playtesten om te kijken waar mogelijk fouten in onze game zaten.
* Factory
  + De factory klasse zorgt ervoor dat alle objecten uit de .level file in de game geladen worden. Dit is een standaard en bekend systeem.
* GameObject
  + De gameObject klasse word eigenlijk gebruikt voor elk object in de game. Elk object, van button tot bully in onze game is een game object. Op deze manier kunnen we alles bijvoorbeeld in dezelfde vector stoppen om makkelijk onze hele game te kunnen updaten en te drawen.
* GameObjectManager
  + De GameObjectManager zorgt ervoor dat er een gameObject opgevraagd kan worden aan de hand van een string. Hiervoor is een aparte klasse aangemaakt om een te lange switch in de Factory te voorkomen.
* HudController
  + De HudController zorgt ervoor dat alle GameObjecten die op de hud getekend moeten worden geupdated, gemoved en gedrawt worden. Dit wordt in een andere klasse dan de LevelController gedaan om de gameObjecten goed gescheiden te kunnen houden. Een alternatief zou twee aparte lijsten in de LevelController zijn geweest, maar dit zou zorgen voor een onoverzichtelijke LevelController.
* NextRound
* Player
  + De player klasse is de codering van de character die de speler bestuurd in de game. Deze klasse draait bijvoorbeeld de character mee met de muis van de speler en laat hem ook naar de besturing van de speler lopen. Ook houd deze klasse bij welke wapens de speler momenteel allemaal gebruikt en zorgt hij ervoor dat het huidig geselecteerde wapen afgevuurd word als de speler klikt.
* Powerup
  + De powerup klasse is verantwoordelijk voor alle power ups in de game, en voor het geld. Op het moment dat een bully dood gaat is er een kans dat er een willekeurige power up spawnt. Wat deze power up doet, en wat er gebeurt als de speler hem op pakt word allemaal geregeld in deze klasse.
* Resume
* SoundController
  + De SoundController zorgt ervoor dat alle geluiden afgespeeld kunnen worden, en maar één keer ingeladen worden. Ook zorgt deze controller voor het afspelen van de background muziek.
* Table
  + De tafel is een object dat een groot deel van de game vult. Naast in de game wereld staan doet de tafel niet veel. De speler kan niet bovenop een tafel klimmen, tenzij hij dit via een bankje probeert.
* Trashcan
  + De trashcans zijn pure obstakels in de game. De speler kan niet door ze heen lopen, en kan ook niet via andere obstakels boven op ze klimmen.

Voor de shop, samen met de wapens, worden de volgende klassen gebruikt:

* Upgrade
  + De upgrade klasse representeerd de upgrade button in de shop. Elke button word toegewezen aan een bepaald wapen, en een bepaald attribuut van dit wapen. Als de speler op deze knop klikt zal dat bepaalde attribuut van het wapen worden geupgrade, als de speler daar genoeg geld voor heeft.
* switchWeapon
  + De switchweapon is een button die er voor zorgt dat de speler in de shop kan wisselen tussen de beschikbare wapens op een bepaald slot in zijn inventory
* Weapon
  + De weapon klasse is de parent klasse van alle wapens. Dit stelt ons in staat om alle verschillende wapens in één lijst te plaatsen, en ook makkelijk te wisselen tussen wapens in de inventory van de speler.
* WeaponCard
  + De weapon card is wat je ziet in de shop. Het lijstje met het plaatje van het wapen, de upgrade buttons en de statistieken van een wapen. Deze klasse zorgt er voor dat al deze elementen goed worden weer gegeven.
* WeaponManager
  + De weapon manager zorgt voor de wapens die speler bij zich heeft en dat alle upgrades behouden blijven bij het wisselen van een level. Ook kan de weapon manager alle upgrades resetten bij het restarten van de game.
* Shop
* ShopCard
* Knife
* Gun
* Bullet

Met betrekken tot de bullies hebben we de volgende klassen:

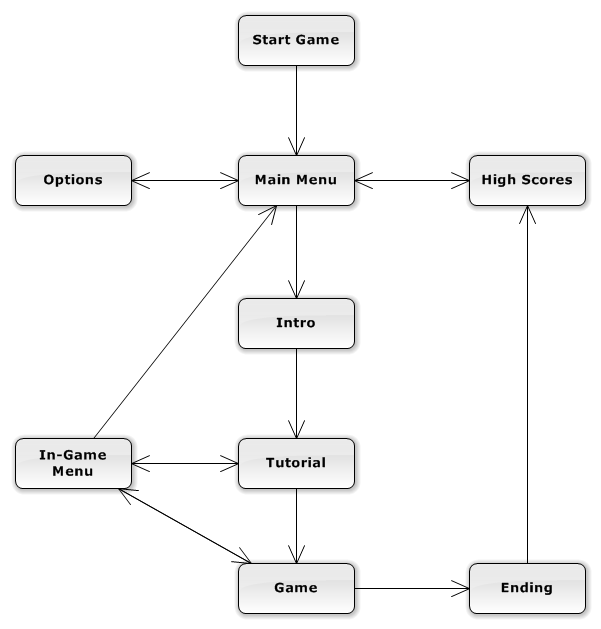
* bobbyBoss
* dunkenBoss
* Enemy
* ethanBoss
* parkerBoss
* zoeyBoss

Voor de particles word gebruikt gemaakt van de volgende klassen:

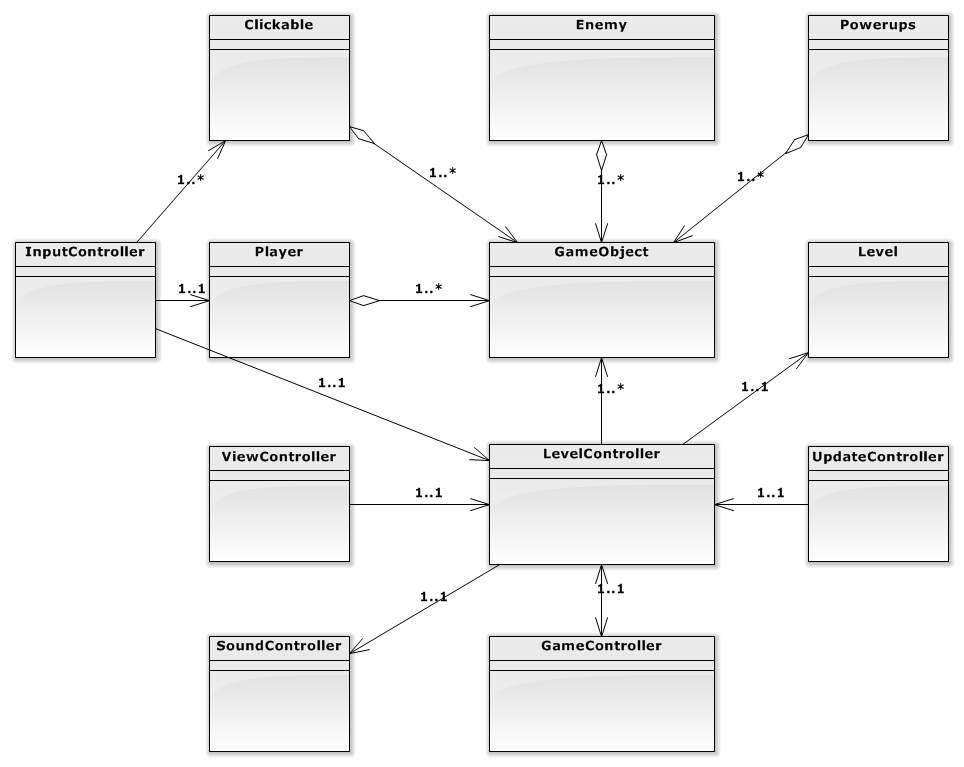
* ParticleManager
  + De particleManager is de klasse die alle lijsten met particles bij houdt. De level controller heeft een pointer naar dit object, welke hij door kan geven aan elk particleEmitter object. De particleManager zorgt ook voor het updaten, en tekenen van elke particle in de game. We laten dit door één object doen zodat we konden kiezen op welke laag van de game de particles worden getekend. We creëren de particle manager na alle objecten in de game, maar vóór de player en de enemies, zodat de particles bovenop alle objecten worden getekend, maar de player en de enemies er nog steeds overheen lopen.
* Particle
  + De particle is in principe een hele simpele class. Op het moment dat hij word gecreëerd krijgt hij een positie, een richting en een snelheid mee. Elke update zal hij zijn snelheid verminderen tot hij stil komt te liggen, op welk moment hij zichzelf weer zal verwijderen. Het klinkt niet als iets heel spectaculairs, maar op het moment dat je er genoeg tegelijk maakt in een game, ziet het er toch een stuk beter uit.
* ParticleEmitter
  + Alle objecten die particles moeten kunnen spawnen extenden de klasse particleEmitter. In deze klasse kunnen alle eigenschappen van de particles worden geset. Op het moment dat een game object dan particles wil spawnen hoeft hij maar één simpele methode van de emitter aan te roepen, en dan zoekt de particleEmitter contact met de particleManager en zal hij een aantal particles toevoegen aan de game.

# Gebruikersmodellen

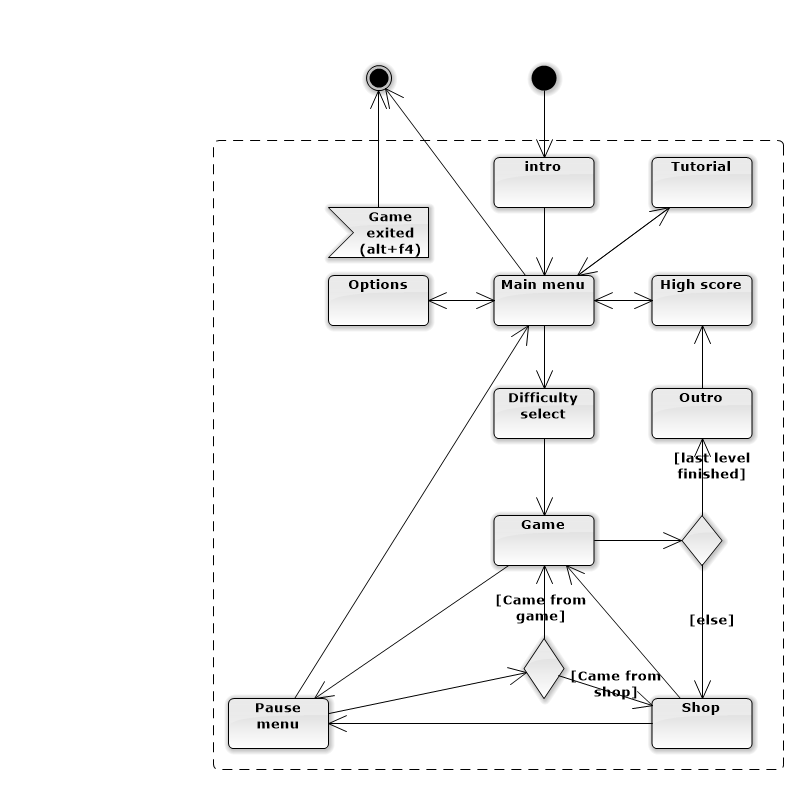
## 1. Game sequence



## 2. Klassendiagram



## 3. Activity diagram



# Testrapporten

# Conclusie

# Bronvermelding