

# Verslag Roadfighter

Project gevorderd programmeren

**Stijn Rosaer**  
20172195



Januari 2019

# 1 Idea

Het idee is om een spel te creëren waarbij men goed mogelijk de finish moet bereiken met een zo hoog mogelijk puntensaldo. Dit kan door auto's neer te schieten, speciale autos aan te rijden en het vermijden om andere voertuigen te raken.

# 2 Game

De game bestaat uit een klasse Game die alles zal controleren en in een goede volgorde gaan aanroepen. Dit vormt dus een centrale plaats om controles uit te voeren op de toestand van het spel waardoor ik besloten heb om hier de distance, score, background, factory, world, ... bij te houden.

De world bezit alle objecten die tijdens het spel voorkomen. Deze komen daar door middel van abstract factoring die de objecten terug geeft als een shared entity pointer. Wanneer een object verwijderd moet worden, zal dit ook van hieruit gebeuren.

Een update van alle elementen gebeurt eveneens vanuit world. Door composition design pater zal ik van hieruit voor elke bijgehouden entity de update aanroepen, idem. voor draw.

Om het spel "speelbaar" te maken en een constante snelheid te hebben over alle computers start in in het begin van elke loop over mijn spel een timer, op het einde wacht ik dan de tijd die nog resteert om een frame rate te bekomen van 30 fps..

Het genereren van passing cars gebeurt random op een locatie bepaald met mijn Random singleton. Wanneer er een gegenereerd wordt is afhankelijk van een functie die ik uit voer nl. wanneer je verder zit in het spel, zullen er meer auto's komen, als je trager rijdt zullen er minder frequent gegenereerd worden om zo een evenwicht te bekomen. Elke keer dat een passing car aangemaakt wordt zal deze met 10% kans een special passing car zijn

Door het gebruik van observer pattern kan ik doorgeven vanuit mijn observables (PlayerCar, Bullet en PassingPointsCar) wordt mijn observer (Game) aangeroepen die ervoor zorgt dat telkens de juiste actie ondernomen wordt.

### **3 Objects**

Elk object bezit de nodige eigenschappen op de meest voor de hand liggende plaats in de minst afgeleide klasse. Zo bevindt zich de structuur boundaries in de entity welke een top left, bottom right, center, width en height bevat om zo gemakkelijk door te kunnen geven. Voor het controleren van collisions heb ik er voor gekozen om deze functie te implementeren in Entity omdat het me logisch leek omdat een entity botst. Hiervoor geef ik een vector mee van entities omdat ik dan zelf kan kiezen welke botsingen effectief effect hebben. Elk object wordt volledig vanuit de logische kant gecontroleerd. Dit betekent dat beslissingen en bewegingen van hieruit gestuurd worden en dat de grafische kant enkel voor het displayen is. In de grafische kant zal dus ook de transformation singleton gebruikt worden om deze omzetting te doen.

### **4 Background & movement**

Om een perceptie van snelheid te creëren heb ik de achtergrond als een lus door laten lopen aan de snelheid van de speler. Elk ander object krijgt zelf een snelheid en in de update functie zal de snelheid van de speler mee doorgegeven worden waaruit de relatieve snelheid wordt berekend en hoeveel het object dus relatief moet verplaatsen.

### **5 Racing car & special passing car**

#### **5.1 Racing car**

Voor de racing car heb ik gekozen om een simpele autonomie te implementeren. Deze heb ik verkregen door elke keer dat de auto een object raakt een 90% kans te hebben dat hij dit object zal ontwijken met een gelijke verdeling links en rechts wat overeen komt met een normaal persoon.

Doordat de racing car geen rekening houdt met de speler, heb ik de keuze genomen dat deze eigenlijk dezelfde race rijdt als de speler, maar zonder een effect hier op te hebben. We kunnen dus eigenlijk zeggen dat de racing car een ghost player is.

#### **5.2 Special passing car**

Als tweede passing car heb ik beslist om een auto te maken die, wanneer geraakt, zorgt voor extra punten. Voor de rest heeft deze exact dezelfde eigenschappen als een normale passing car.

## 6 Interaction

Hieronder is een overzicht te vinden van de inheritace tussen de voornaamste klassen. Extra schemas's en documentatie zijn te vinden in de DOC folder gegenereerd door doxygen.

