Project 3:

Game physics

Glenn Latomme  
2 ICT

Inhoud

[Background 2](#_Toc354484603)

[Objectives 2](#_Toc354484604)

[Overview 2](#_Toc354484605)

[Progress 2](#_Toc354484606)

[Engine 2](#_Toc354484607)

[How the engine works 2](#_Toc354484608)

[Tick 2](#_Toc354484609)

[Paint 2](#_Toc354484610)

[Game Panel 3](#_Toc354484611)

[Game Objects 3](#_Toc354484612)

[Main program 3](#_Toc354484613)

[REFLECTIE 3](#_Toc354484614)

[Bibliografie 4](#_Toc354484615)

# Background

Voor game phyiscs kon ik redelijk wat gebruik maken van mijn kennis van vorige school, deze was game development, en hier draaide zowat alles in programeren en wiskunde rond dit onderwerp. Ook wat oude projecten kwamen hier van pas om alles nog eens goed op te frissen door dat alles deftig gecommentarieerd was.

# Objectives

## Overview

Het doel was om een werkend voetbal spel te maken, zonder taktieken. Gewoon basic gameplay en alles aan de praat krijgen.

## Progress

Voor de development van het project ben ik begonnen met de design, en dan die verder uitgewerkt naar code. De recommendation is het laatst uitgewerkt geweest zodanig dat deze niet zou worden tegengehouden door code dat er nog niet was en waardoor er niet in meerdere files tegelijk zou moeten gecodeerd worden, wat niet zo’n mooie code zou geven.

### Engine

Elke game heeft een ondersteunende engine nodig om alles te kunnen draaien. Soms zit deze verwerkt in het project zelf, soms zit deze er bij. Voor dit project hebk ik gekozen om deze appart te laten draaien. En een zeer simpele basis enginge te maken die alles kon doen wat er in dit project van me verwacht werd

#### How the engine works

Mijn engine bestaat uit 2 grote delen: namelijk een Tick en een paint, deze zijn 2 elementen dat er voor zorgen dat alles geordend blijft.

Een tick word meestal vaker uitgevoerd dan een paint, om prezere berekeningen te hebben, zonder overbodig te moeten tekenen, want een mens zal alles sneller dan 50 frames per seconde niet meer waarnemen. En dan zou dat verloren rekenkracht zijn

##### Tick

Hierin worden alle berekeningen gedaan, zoals welke nieuwe posities de objecten heben, of er een botsing is, als er met sprites gewerkt word, de volgende sprite.

Af en toe word hier ook de toetsen afhandeling gedaan, soms word dit ook appart gedaan. Maar aangezien dit niet relevant was zit dit er ook niet in.

##### Paint

Hier in worden enkel en alleen alle elementen op het scherm gezet, dit draaid vaak aan 60 FPS omdat zoals net vermeld mensen het verschil niet meer zien boven de 50, en om wat speling te hebben voor als er eens een zwaardere bereknging zit deze niet direct visueel zichtbaar is

#### Game Panel

Als je een nieuw spel maakt die deze engine maakt, dan moet je gewoon in je form een panel maken en deze laten overerven van de gamePanel voorzien in de engine, daarin moet je een FPS, Height en Width meven en dan kan je gewoon gebruik maken van de Tick en Paint zonder enige andere berekeningen te moeten maken

#### Game Objects

Dit heb ik onderverdeel in 3 delen, een interface, statisch en dynamisch game object.

Elk element dat je visueel wilt tonen is een game object daarvan heeft mijn interface gewoon een paint als enige functie. Maar het nut hiervan is dat je alles dat je wilt painten theoretisch in 1 list kan steken zonder dat dit enige problemen geeft. Maar dit geeft wel iets minder overzicht op code, dus dat doen we dan niet

Het statisch Game object erft dus over van de interface en voegt wat meer functies toe aan het object, deze heeft een rectangle waarin deze zich bevind, wat dus een positie breedte en hoogte heeft, dus er kunnen hier al botsingen op uitgevoert worden

Dan als laatste het dynamisch game object, deze is zoals de naam zegt dynamsich dus beweegt rond in de wereld, adhv een velocity.

### Main program

Door mijn engine was er neit veel meer dat het programma moest voorzien, buiten de collisions.

Een botsing tussen 2 cirkels moet je maar een paar dingen controleren, eerst en vooral of ze botsen, dit is wanneer de afstand tussen de 2 middelpunten (stelling van pythagoras) kleiner is dan de som van de 2 stralen

Wanneer dat zo is kan je hoek tussen de 2 circels berekenen, en daarmee kan je de cosinus en de sinus verkrijgen

Nu je deze 2 hebt kan je de nieuwe richtings vectoren berekenn, en toepassen op de bollen

## REFLECTIE

Dit was niet de meest uitdagende opdracht doordat ik meeste dingen al gezien had, waardoor ik snel verveelt graakte en daardoor deze opdracht niet kon afwerken.

# Bibliografie

latomme, G. (2010). Projects from DAE.