Ejento 148  
Technisch Verslag

|  |  |
| --- | --- |
| Naam | Studentnummer |
| Aydin Biber | 1666849 |
| Jessy Visch | 1661709 |
| Kevin Damen | 1660811 |

**Versie 1.0**

Team: S A D B O Y S  
Laatst gewijzigd op: 05-11-2015

# Versiebeheer

|  |  |  |
| --- | --- | --- |
| **Versie** | **datum** | **Wijzigingen** |
| 0.1 | 22-09-2015 | Eerste opzet. Voorblad, Inhoudsopgave, Achtergronden en Doelstellingen toegevoegd. |
| 0.2 | 05-11-2015 | Koppen toevoegen conform voorbeeld technisch verslag. |
| 0.3 | 06-11-2015 | Invullen 2, 2.1, 2.2, 4.1, 4.3, 5 |
| 0.4 | 09-11-2015 | Invullen 3.1, 7 |
| 1.0 | 09-11-2015 | Spellings en grammatica controle |

Inhoudsopgave

[Versiebeheer 2](#_Toc434851362)

[1 Managementsamenvatting 4](#_Toc434851363)

[1.1 Inleiding 4](#_Toc434851364)

[1. 2 Ontwikkeling ontwerpen 4](#_Toc434851365)

[1.3 Resultaten en conclusies 4](#_Toc434851366)

[1.4 Aanbevelingen 4](#_Toc434851367)

[2 Inleiding 5](#_Toc434851368)

[2.1 Motivatie 5](#_Toc434851369)

[2.2 Aanleiding 5](#_Toc434851370)

[2.3 Leeswijzer 5](#_Toc434851371)

[3 Functioneel Ontwerp 6](#_Toc434851372)

[3.1 Game Design Document 6](#_Toc434851373)

[3.2 Gemaakte keuzes 6](#_Toc434851374)

[4 Technisch Ontwerp 7](#_Toc434851375)

[4.1 Architectuur systeem 7](#_Toc434851376)

[4.2 Klassendiagram 8](#_Toc434851377)

[4.3 Beschrijving Klassendiagram 9](#_Toc434851378)

[5 Realisatie 11](#_Toc434851379)

[5.1 Problemen realisatie 11](#_Toc434851380)

[5.1.1 Probleem met AnimatedSprite 11](#_Toc434851381)

[5.1.2 Probleem met Git 12](#_Toc434851382)

[5.2 Testrapporten 12](#_Toc434851383)

[6 Evaluatie 14](#_Toc434851384)

[7 Conclusies en aanbevelingen 15](#_Toc434851385)

[7.1 functionele aanbevelingen 15](#_Toc434851386)

[7.2 technische aanbevelingen 15](#_Toc434851387)

[8 Bronnenlijst 16](#_Toc434851388)

# 1 Managementsamenvatting

## 1.1 Inleiding

Dit verslag gaat over de ontwikkeling van het spel Ejento 148 in opdracht van de Hogeschool Utrecht. Het spel moest is ontwikkeld in de programmeer taal C++ met gebruik van de SFML library.

Voor de ontwikkeling van het project werd er gebruik gemaakt van Git en SCRUM. Dit zijn methodes om het werken in een project beter te laten te verlopen wanneer er gebruik wordt gemaakt van de Agile ontwikkel methode.

## 1. 2 Ontwikkeling ontwerpen

Het spel werd ontwikkeld aan de hand van een zelf bedacht concept. Er werd gekozen voor een simpel design met mogelijkheid tot diepgang zodat het mogelijk was voor de ontwikkelaars om vrije keuzes te maken in het ontwerpen.

Als genre game werd er gekozen voor een 2D-sidescroller. Dit omdat een 2D setting perfect is voor het toepassen van de door de ontwikkelaars verkregen kennis uit de voorgaande projecten. Het sidescroller element werd gekozen omdat dit toegang biedt tot snelle gameplay binnen een level zonder dat de speler moet wachten op laadschermen voor de volgende ruimte.

Om het spel een unieke twist te geven bestuurd de speler een “bad-guy”. Dit personage heeft een donkere achtergrond en hiervoor is een bijpassend thema gekozen voor de Sprites. Het spel en alle elementen er binnen reflecteren dit.

## 1.3 Resultaten en conclusies

Tijdens het ontwikkelen van Ejento 148 zijn we tegen een aantal dingen aangelopen, waaronder het animeren van de Sprites. We kwamen erachter dat SFML geen ondersteuning biedt voor bewegende animaties. Dit hebben wij kunnen oplossen door gebruik te maken van de library genaamd “AnimatedSprite”. Deze library maakt gebruik van de functionaliteiten die SFML zelf bied maar past ze toe om Sprites te animeren.

Een ander punt waar wij tegen aan liepen tijdens het ontwikkelen was dat we veel tijd kwijt waren aan merge conflicten binnen Git. Hierdoor moesten wij vaak een handmatige merge uitvoeren wat veel tijd inneemt. Uiteindelijk hebben wij dit opgelost door de bestanden handmatig aan de Git commit toe te voegen, maar op de lange termijn kan dit beter worden opgelost door een goed .gitignore bestand toe te passen.

## 1.4 Aanbevelingen

Een laatste punt, dat wij streng aan bevelen voor een toekomstig project, is het opbouwen van het level. Momenteel is het complex om een level op te bouwen binnen het bestand waaruit het geladen wordt. Dit bestand is moeilijk om snel aan te passen, en het kost tijd om het spel opnieuw op te starten na iedere aanpassing. Wij willen dan ook streng aanbevelen om te beginnen met het ontwikkelen een level builder. Hierdoor kan het level vanuit het spel worden aangepast waardoor aanpassingen makkelijker kunnen worden gecontroleerd.

# 2 Inleiding

Dit document beschrijft het technische gedeelte van het project dat is geïnitieerd voor de ontwikkeling van het spel Ejento 148. Het doel van dit document is het toelichten van de beslissingen en risico’s die zijn genomen in het project.

Wanneer er een referentie naar code wordt gemaakt in het verslag zal deze worden onderstreept met een grijze streep, zoals volgt:

std::cout << “Hello World!”;

2.1 Motivatie  
Het bedrijf “S A D B O Y S” wilt een spel ontwikkelen dat simpel is om te spelen, maar uitdagend/moeilijk is om te meesteren. Het doel van deze opdracht is om C++, Git en Scrum te leren beheersen.

## 2.2 Aanleiding

Te veel spellen worden gemaakt met als doel winst in plaats van het bieden van een uitdaging aan de speler. Hierdoor worden spellen tegenwoordig te simpel gemaakt zonder een uitdaging te bieden waar de speler daadwerkelijk van kan genieten.

“S A D B O Y S” heeft als doel genomen een spel te ontwikkelen wat toegankelijk is voor alle spelers, maar een uitdaging biedt die de speler nog lang bezig zal houden.

## 2.3 Leeswijzer

Dit verslag is ingedeeld in hoofdstukken.

Op pagina 3 is een inhoudsopgave te vinden met daarin de koppen van elk hoofdstuk.

# 3 Functioneel Ontwerp

In dit hoofdstuk worden de design keuzes beschreven. Tevens worden er in dit hoofdstuk verwijzingen gemaakt naar het Game Design Document (“GDD”).

## 3.1 Game Design Document

De speler is een huurmoordenaar genaamd “Ejento 148”. Deze huurmoordenaar is ingehuurd om de onschuldige zakenman “Walter E. Aboo” om te brengen en zijn uitvinding te stelen.

Dit doet de speler door vijanden te vermoorden of ontwijken op zijn weg naar Walter. De vijanden die Ejento 148 zal tegenkomen zijn bodyguards die Walter heeft ingehuurd om hem te beschermen.

Het vermoorden van de vijanden kan de speler doen door middel van een zwaard. Dit zwaard kan sterker worden gemaakt door middel van upgrades die door het level zijn verspreid.

Het ontwijken van de vijanden kan de speler doen door middel van platformen. Deze platformen zijn door het level verspreid en de speler zal moeilijke sprongen moeten doen om deze succesvol te benutten.

De vijanden verdedigen zichzelf met een pistool. Wanneer een vijand Ejento 148 ziet zal hij zichzelf verdedigen met een salvo aan schoten. Wanneer Ejento 148 te vaak is geraakt zal deze overlijden en is het spel voorbij. Ejento 148 kan zichzelf ook helen door middel van “health items” die door het spel zijn verspreid.

Het spel speelt zich af binnen een 2-dimensionale omgeving.

## 3.2 Gemaakte keuzes

*Waarom Ninja thema?*

Wij wilden een spel maken wat voor de typische speler makkelijk te begrijpen is. Als je een ninja ziet denkt men aan een huurmoordenaar met een zwaard. Dit is het idee wat wij aan de speler willen overbrengen, omdat de speler eigenlijk de “bad-guy” speelt.

*Waarom een 2d sidescroller?*

Dit spel wilden wij neerzetten als een actiespel. De speler kan er voor kiezen om snel te gaan of langzaam, maar uiteindelijk wordt wachten niet beloond. Een level systeem zoals “The Legend of Zelda” (denk aan elkaar gekoppelde kamers) zou de actie te veel limiteren tot een enkel scherm per kamer. Wij willen ook dat de speler rekening moet houden met wat achter hem is wanneer deze er voor heeft gekozen om een vijand te ontwijken i.p.v. te vermoorden.

*Waarom health items?*

Wij willen een spel maken wat makkelijk is om te spelen, maar moeilijk is om te bemeesteren. Wanneer het onmogelijk is voor een speler om zichzelf te helen, worden de fouten die een speler maakt te zwaar gestraft.

Hiernaast kan het ook een extra moeilijkheid toevoegen, wanneer een speler lage levens heeft en een aantal gevaarlijke sprongen zal moeten maken om een health item te pakken.

*Waarom bodyguards met pistolen?*

De bodyguards die Walter heeft ingehuurd zijn er om Walter te beschermen. Deze bodyguards zijn niet ingehuurd om iemand te vermoorden tenzij hun eigen leven of dat van Walter in gevaar is.

De bodyguards zijn om deze reden dan ook aan het patrouilleren rond hun aangeven locatie, in plaats van zoeken naar de huurmoordenaar.

# 4 Technisch Ontwerp

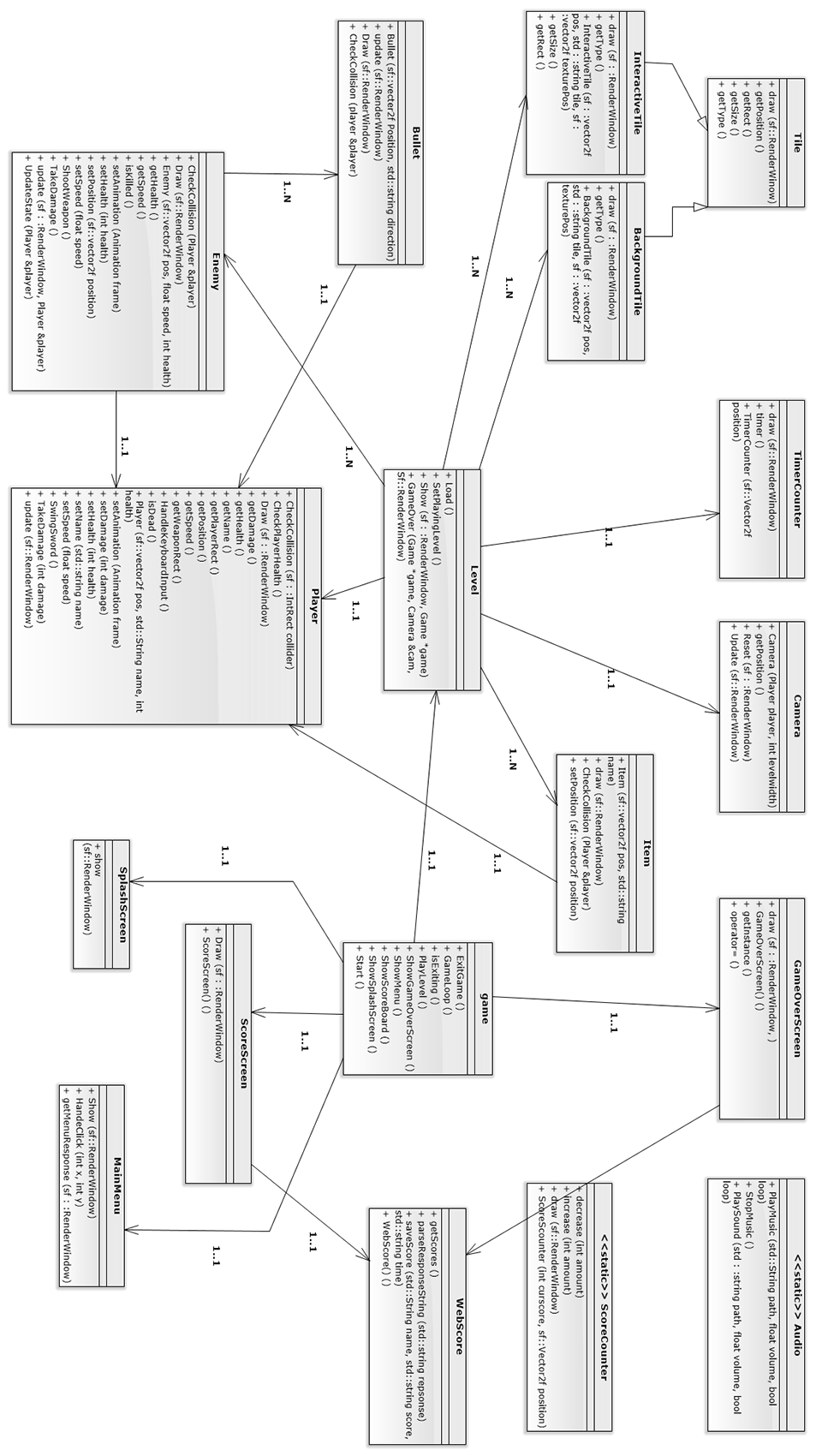
In dit hoofdstuk worden de specificaties van de het spel besproken en de keuzen die we hebben gemaakt bij het bouwen van het spel.

## 4.1 Architectuur systeem

Het systeem roept een hoofdklasse aan die de beeldschermen managet. Wanneer er een event gebeurt op het scherm wordt het resultaat hiervan teruggegeven en wordt er een handeling uitgevoerd gebaseerd op het resultaat.

Het afhandelen van welk scherm er getoond moet worden wordt geregeld door middel van een variabele genaamd gameState. Een recursieve functie binnen de hoofdklasse controleert wat de huidige staat is van deze variabele en roept aan de hand hiervan een functie aan. Binnen deze functie wordt een nieuwe instantie van een scherm gecreëerd en van dit scherm wordt de show() functie aangeroepen, die het beeldscherm tekent. Binnen deze functie worden de elementen van deze klasse zelf getekend.

## 4.2 Klassendiagram



## 4.3 Beschrijving Klassendiagram

**Tile**

Klasse Tile is de superklasse van InteractiveTile en background Tile. Deze Tile klasse is verantwoordelijk voor het aansturen van zijn subklassen.

**InteractiveTile**

De klasse InteractiveTile is verantwoordelijk voor het tekenen, het bijhouden van de positie, de grootte, de rectangle en de Texture van een InteractiveTile. Een interactieve Tile is een Tile in het level die interactie heeft met andere dingen in het level. Denk hier bij aan spelers en enemies.

**BackgroundTile**

De klasse BackgroundTile is verantwoordelijk voor het tekenen en bijhouden van de positie, de grootte en de Texture van een interactieve Tile. Een background Tile is een Tile die er puur is om een deel van het level te tekenen. Deze Tiles hebben geen interactie met andere objecten in het level.

**Level**

De klasse level is verantwoordelijk voor het laden, tekenen en updaten van het level. In de level klasse wordt alles samengebracht. In functie level::show wordt onder andere de speler, de enemies, en de items aangemaakt. Ook wordt er hier gekeken of de speler en enemies collision hebben met platformen of elkaar.

**Player**

De Player klasse is verantwoordelijk voor het tekenen en bijhouden van alle gerelateerde dingen aan een Player. Zo worden in deze klas de animaties getekend van het karakter, zijn levens geüpdatet en zijn snelheid bijgehouden. Ook wordt hier de input van de speler afgehandeld en tot slot wordt hierop aangeven van de Level klasse collision detection uitgevoerd.

**Enemy**

De Player klasse is voor het tekenen en bijhouden van alle gerelateerde dingen aan een Enemy. Zo worden in deze klas de animaties getekend van de vijand, zijn levens geüpdatet, zijn snelheid bijgehouden en zijn keuzes gemaakt op basis van AI. Ook worden er in deze klasse instanties van Bullet aangemaakt die de Enemy richting de speler schiet.

**Bullet**

De Bullet klasse is verantwoordelijk voor het tekenen en bijhouden van de snelheid en de richting van de Bullet.

**Camera**

De camera klasse is verantwoordelijk van het tonen van het correcte scherm in het level aan de gebruiker. De camera krijgt via klasse Level de positie van de speler door en centreert hier het scherm op zodat het scherm meebeweegt met de speler.

**TimerCounter**

De TimerCounter klasse is verantwoordelijk voor het bijhouden en het tonen van de tijd die de speler in het level besteed.

**ScoreCounter**

De ScoreCounter klasse is verantwoordelijk voor het bijhouden en het tonen van de door de speler in het level behaalde punten.

**Item**

Klasse item is verantwoordelijk voor het tekenen en bijhouden van de positie en het type van het item.

**Game**

De klasse game is verantwoordelijk voor het tonen en bijhouden van de verschillende gameState’s. de gameState’s zijn bijvoorbeeld PlayingLevel, ShowingMenu, Showingscoreboard of ShowingGameOverScreen.

**MainMenu**

De klasse MainMenu is verantwoordelijk voor het tonen en het afhandelen van kliks op de verschillende menu items.

**Scorescreen**

De klasse Scorescreen is verantwoordelijk voor het ophalen en het tonen van de highscores.

**SplashScreen**

De klasse SplashScreen is verantwoordelijk voor het tonen van het SplashScreen.

**GameOverScreen**

De klasse GameOverScreen is verantwoordelijk voor het tonen van het scherm dat de speler ziet als hij dood is of het level heeft uitgespeeld. Op dit scherm kan de speler zijn naam invoeren voor de highscores.

**WebScore**

De klasse WebScore is verantwoordelijk voor het ophalen en wegschrijven van de highscores. Dit gebeurd door te communiceren meet een web API.

**Audio**

De klasse audio is verantwoordelijk voor het afspelen van de verschillende geluiden en muziekbestanden die de game bevat.

# 5 Realisatie

In dit hoofdstuk wordt besproken hoe de softwareontwikkeling in zijn werk gaat. Dit houdt in voor welk platform het spel gemaakt wordt en wat de software is die wij hebben gebruik om het spel te ontwikkelen. Tevens zal er een beschrijving zijn van hoe wij deze software hebben gebruikt om het product te realiseren.

Voor platform is er gekozen om Windows 10 te gebruiken. Wij hebben deze keuze gemaakt omdat het merendeel van spelers wereldwijd gebruik maken van Windows om games op te spelen. Tevens is het OS voor alle leden van het project toegankelijk.

De software die gebruikt wordt voor het maken van het spel is als volgt:

1. Windows 10, het Operating System
2. Windows Visual Studio 2015, de ontwikkelomgeving
3. SFML 2.3.2
4. AnimatedSprite, een library voor het tonen van geanimeerde Sprites
5. Adobe Photoshop, voor het aanpassen van de Sprites

Als eerste onderdeel hebben wij gewerkt aan het systeem voor de schermen. Dit zijn de Splash screen, MainMenu screen en Game screen. Het Game screen was nodig om andere functionaliteiten binnen het spel te kunnen testen zoals het tekenen van objecten en interactie tussen deze objecten.

Wanneer een aanpassing moest worden gemaakt werd eerst het spel afgesloten. Hierna hebben wij de aanpassing in de code doorgevoerd waarna we opnieuw het programma hebben gecompileerd. Tijdens het testen of de nieuwe aanpassing goed werkt hebben wij geen andere aanpassingen gedaan om het testen makkelijker te maken, zodat wij te allen tijde zeker weten dat de aanpassing binnen de realistische situatie goed werkt.

## 5.1 Problemen realisatie

### 5.1.1 Probleem met AnimatedSprite

Een probleem wat naar voren kwam bij de realisatie was dat SFML geen ondersteuning heeft voor bewegende animaties. Wij wilden deze bewegende animaties in het spel verwerken zodat de speler duidelijk kan zien welke acties worden uitgevoerd door zowel het speler karakter als de vijanden.

Uiteindelijk hebben wij dit opgelost door gebruik te maken van een library genaamd “AnimatedSprite”. Deze library maakt gebruik van de functionaliteiten die SFML zelf biedt en het design van de library volgt nauw het design van de sf::Sprite klasse die in SFML zit.

Door gebruik te maken van deze library werd het voor ons moeilijker om de positie van de speler en vijand te gebruiken voor berekeningen. In plaats van de positie die wij initieel meegeven aan deze objecten, moest gebruik gemaakt worden van de positie van de animatie. Dit zag er als volgt uit:

animation.getPosition().x; en animation.getPosition().y;

Dit was een probleem omdat de positie van de speler en vijanden wordt meegegeven bij het aanmaken van deze objecten. Hierdoor moesten wij later nogmaals de positie van de animatie aanpassen aan de hand van de positie die wij meegaven, zodat de speler en vijand verscheen waar wij deze wilden hebben. Daarna moesten wij in de update() van het object de positie zelf gelijk zetten aan de positie van de animatie zodat wij deze konden gebruiken om in andere klassen de positie op te vragen van deze objecten.

### 5.1.2 Probleem met Git

Er werd tijdens het ontwikkelen van het spel veel tijd verloren aan merge conflicten van Git. Omdat .obj en .suo bestanden niet waren toegevoegd in het .gitignore bestand, werden deze geüpdatet in de origin op github.com. Dit had als gevolg dat er vaak merge conflicten ontstonden die handmatig moesten worden opgelost.

Uiteindelijk hebben wij dit opgelost door de bestanden waar wij aan hebben gewerkt handmatig toe te voegen aan de git commit. Hierdoor werden geen merge conflicten meer gecreëerd.

## 5.2 Testrapporten

|  |  |
| --- | --- |
| **Testrapport** | Level |
| **User Stories** | als speler wil ik het level kunnen zien om te weten waar mij karakter zich bevind.  Als speler wil ik kunnen lopen zodat ik vooruit kan in het spel. |
| **Doelstelling** | De doelstelling van deze test is kijken of een speler het level kan zien en hier door heen kan lopen. |
| **Methode** | Eerst ga ik kijken of het level en de speler zichtbaar zijn.  Vervolgens ga ik kijken of de speler niet door het level zakt maar op de zichtbare blokjes kan lopen. |
| **Resultaten** | De speler ziet het level en kan hier doorheen lopen. |
| **Conclusie** | Deze twee user stories zijn correct uitgevoerd en afgerond |

|  |  |
| --- | --- |
| **Testrapport** | Hoofdmenu |
| **User Stories** | Als een speler wil ik op de stop knop kunnen drukken zodat het programma stopt.  Als een speler wil ik op de start knop kunnen drukken zodat het spel start.  Als een speler wil ik op de leaderboards knop kunnen drukken zodat ik de wereldwijde scores kan bekijken |
| **Doelstelling** | Met deze tests willen wij aantonen dat de knoppen hun verwachte taken uitvoeren zonder foutmeldingen.  Verwachte taken zijn het starten van het spel met de start-knop, het stoppen van het programma met de stop-knop en het tonen van de leaderboards met de leaderboards knop. |
| **Methode** | **Testen start knop**  De start knop test ik door in het hoofdmenu op de start knop te drukken.  **Testen leaderboards knop**  De leaderboards knop test ik door in het hoofdmenu op de leaderboards-knop te drukken.  **Testen stop knop**  De stop knop test ik door in het hoofdmenu op de stop knop te drukken. |
| **Resultaten** | **Resultaten start knop**  Na het klikken op de start knop in het hoofdmenu werd het spel gestart. Dit kon ik waarnemen doordat het scherm kort zwart werd, waarna een speler karakter verscheen.  **Resultaten leaderboards knop**  Na het klikken op de leaderboards knop in het hoofdmenu werd het leaderboards scherm getoond. Dit kon ik waarnemen doordat het scherm de scores van voor mij onbekende spelers toont.  **Resultaten stop knop**  Na het klikken op de stop knop werd het scherm gesloten. Dit kon ik waarnemen doordat het proces van het spel niet meer in taakbeheer stond. |
| **Conclusie** | De start, leaderboards en stop knop zijn functioneel en werken correct zonder foutmeldingen te genereren. |

|  |  |
| --- | --- |
| **Test rapport** | Animaties speler |
| **User Stories** | Als een speler wil ik mijn karakter kunnen zien bewegen zodat ik weet wat voor actie wordt uitgevoerd. |
| **Doelstelling** | Met deze tests willen wij controleren of de animaties van de speler correct worden uitgevoerd na het drukken op de knoppen voor beweging |
| **Methode** | **Testen animatie links**  De animatie links test ik door op de A knop te drukken.  **Testen animatie rechts**  De animatie rechts test ik door op de D knop te drukken.  **Testen animatie vallen**  De animatie vallen test ik door op de spatie knop de drukken.  **Testen animatie dood gaan**  De animatie springen test ik door mij aan te laten vallen door vijanden tot mij alle levens verlies. |
| **Resultaten** | **Resultaten animatie links**  Na het klikken op de A knop werd er een animatie getoond van een karakter wat naar links loopt.  **Resultaten animatie rechts**  Na het klikken op de D knop werd er een animatie getoond van een karakter wat naar rechts loopt.  **Resultaten aanvallen**  Na het klikken op de spatie knop werd er een animatie getoond van een karakter wat met een zwaard slaat.  **Resultaten animatie dood gaan**  Na het verliezen van alle levens werd een animatie afgespeeld van een karakter wat naar viel. |
| **Conclusie** | De animaties worden correct uitgevoerd zonder foutmeldingen van het programma. |

# 6 Evaluatie

In 4.1.1 hebben wij aangegeven dat een probleem werd opgelost door middel van de “AnimatedSprite” library en dat deze oplossing op zich weer problemen veroorzaakte. Wij hadden deze problemen kunnen voorkomen door gebruik te maken van een andere methode om de animaties af te beelden, of een slimmere implementatie van het instantiëren van de speler en Enemy objecten.

Door de gelimiteerde ervaring die de programmeurs hebben met animaties werd er gekozen voor deze AnimatedSprite klasse omdat deze snel toe te passen was. Achteraf gezien was er niet veel wat kon worden veranderd, maar deze kennis kan in de toekomst worden toegepast om een betere methodiek te vinden voor het animeren van Sprites.

In 4.1.2 gaven wij aan dat er problemen waren ontstaan bij het gebruik van Git.

Dit probleem werd veroorzaakt omdat er bij het begin van het project geen .gitignore bestand is gemaakt, waardoor .obj en .suo bestanden niet werden genegeerd bij het committen van de code.

Dit probleem had beter kunnen worden opgelost door aan het begin van het project een goed opgezet .gitignore bestand te maken waardoor alleen .cpp en .hpp bestanden worden toegevoegd.

Het spel maakt geen gebruik van een vaste updatefrequentie. Dit betekent dat de game afhankelijk is van FPS (Frames Per Second) om dingen zoals zwaartekracht en snelheid van de speler te bepalen.

We hebben dit opgelost door de Frame rate van de game te limiteren naar 200 frames per seconden. Helaas laat deze oplossing nog veel te wensen over. In de toekomst is het nodig om gebruik te maken van een vaste updatefrequentie om raar gedrag te voorkomen indien een bepaald aantal frames niet behaald word.

# 7 Conclusies en aanbevelingen

## 7.1 functionele aanbevelingen

Eén van de grote punten waar wij tegenaan liepen op het gebied van functionaliteit is het feit dat de levels erg complex zijn om met de hand te maken.

De levels worden aan de hand van een tekstbestand ingelezen en getekend. Deze tekstbestanden bevatten combinaties van tekens en cijfers die een bepaalde Tile op een bepaalde plek te zetten, dit handmatig doen is een erg tijdrovende klus.

Wij bevelen hier voor aan om te beginnen met het maken van een level builder. In deze level builder kan je snel en gemakkelijk met een GUI een mooi level maken. Dit is niet alleen erg handig voor de developer, maar spelers vinden het zelf ook erg leuk om hun eigen level te maken.

## 7.2 technische aanbevelingen

Zoals reeds besproken bij de evaluatie in hoofdstuk 5 is er voor de game absoluut een vaste updatefrequentie nodig. In de huidige game hangt het af van frames per seconde af en dit is niet ideaal.

# 8 Bronnenlijst

* Foaly. (2015). AnimatedSprite Geraadpleegd op 24 september 2015, van <https://github.com/SFML/SFML/wiki/Source:-AnimatedSprite>
* Henry, P. (2014). SFML *Made Easy* Geraadpleegd op 1 oktober 2015, van <https://www.youtube.com/playlist?list=PL0249162D5D6FA5D1>