# Technisch Verslag

Dit is het technisch verslag dat bijgeleverd wordt bij het product/project “GUNOUT”. Hierin gaan wij iets verder, verhaalend, in op de code, als bijlage voor de doxygen.

# Entity

Entity class is de base klasse voor alle “voorwerpen” in het spel die interactable zijn, daarom hebben we gekozen voor een virtuele functie.

# EntityController class

De entityController class is de klasse die verantwoordelijk is voor de interactie tussen alle entiteiten. De entity controller bevat daarom ook alle geërfde klassen van entity. De entiteit doet ook alle collision detectie tussen de entiteiten. Bijvoorbeeld als een bullet een crate overlapt wordt er gezorgd dat de crate stuk gaat en er een item wordt gedropt. De entityController bevat std::vectors waar de entiteiten worden ingestopt. De entityController zorgt er ook voor dat bepaalde entiteiten worden destruct tijdens het spelen van een level als ze niet meer nodig zijn (dus geheugen management). De entityController zorgt ook voor het aanroepen van de update en draw functie van de entiteiten. De entityController haalt alle entiteiten in het begin op van de Map klasse. De map klasse leest de kleurwaarden van een PNG bestand om te bepalen waar welke entiteit moet en geeft de objecten terug aan de entityController.

Graphic

De graphic klasse is om de plaatjes in te laden.

circular independence

De entityController behandeld de controller input van de Player en niet de player object zelf om circular independence tegen te gaan.

De hit functie in de Entity klasse returns een Entiy object. De twee klassen die een daadwerkelijk een object teruggeven in plaats van een null pointer zijn enemy en crate klasse. Alle twee geven ze de geërfde entity klasse item terug, echter is er niet gekozen om te definiëren dat de hit functie een item moet teruggeven want de item klasse is al een erfenis van entity. Hiermee voorkomen we een circular independence. De teruggeven pointer van een entity word weer naar een item gecast om de functies van een item object aan te roepen.

# Camera class

De camera class veranderd de positie van de view zodat de player altijd in het midden is.

PlayerStats  
De playerStats is een struct waar maar 1 van is die de player altijd heeft. De player object heeft maar één playerStats. De playerStats is verantwoordelijk om de player info die continu tijdens een level veranderd bij te houden.

# Timer

De timer klasse is een klasse die afhankelijk van de frame rate bijhoudt hoeveel tijd voorbij is gegaan. De Frame rate in het spel is op 60 fps gezet.

LevelManager

De level manager houdt bij welke staat de game is. De level manager bevat een GameLoop object die die de entity controller aanroept om aan te geven welk level ingeladen moet worden. De GameLoop bevat Playing state objecten voor elke level, Daarnaast zijn er ook Menu, titel screen, high score en credit klassen.

HUD

De hud klasse is een klasse die verantwoordelijk is voor informatie te displayen tijdens het spelen van de levels. De HUD klasse is vanwege tijd niet heel abstract gehouden. Als men verder wilt ontwikkelen aan de game wordt aanbevolen om een hud controller te maken die hud objecten bevat.

BUGLOG

-geluid speelt raar af als de speler wisselt van state en tegelijk schiet.  
-geluid stopt met spelen tijdens de game (Niet zeker als het een een sfml dll zit)

-Als de speler tijdens het spelen in een level naar het main menu gaat dan is er eerste een transitie waar je de speler voor één seconde aan het begin van de level ziet