**SINGLE RESPONSIBILITY PRINCIPLE**

Graphical user interface, text, application, chat or text message

Description automatically generated

Beide Health- en DeathManager regelen dingen in verband met de HP van een Creature, toch zijn deze opgesplitst in het regelen van de **hoeveelheid** HP en het regelen van de **status** van een creature adhv de hoeveelheid HP. (zie hieronder)

Text

Description automatically generated

Text

Description automatically generated

Dit principe is ook terug te vinden in Gravity en Movement. Beide hebben ze te maken met het bewegen van een Creature. Toch zijn ze opgesplitst omdat ze beide andere functies hebben en ze niet altijd samen gebruikt moeten worden.

Graphical user interface, text, application, chat or text message

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

Screenshots terug te vinden in ClassDiagram.cd, Classes/HealthManager.cs, Classes/DeathManager.cs, Classes/Movement.cs en Classes/Gravity.cs

**OPEN/CLOSED PRINCIPLE**

Doorheen gans het project maak ik gebruik van het open/closed principe. Dit in de vorm van Interfaces die combineerbaar zijn, abstracte klassen waarvan reeds overgeërft is geweest en de mogelijkheid biedt om extra subklassen toe te voegen. Deze abstracte klassen kunnen dan ook wel eens polymorfisch genoemd worden.

Overzicht klassendiagram:

Graphical user interface

Description automatically generated

Screenshots terug te vinden in ClassDiagram.cd

**INTERFACE SEGREGATION PRINCIPLE**

Zoals hieronder te zien is (zie afb.1) heb ik een hele hiërarchie van interfaces gemaakt om er voor te zorgen dat ik classes kan maken enkel met de hiervoor nodige variabelen en methodes. Een visueel duidelijk voorbeeld is Entity en Creature (zie afb.2). Entity erft slechts over van **één** interface terwijl Creature van **negen** interfaces overerft. Een goed voorbeeld om het principe van ISP aan te tonen is dat ItemStatic en ItemDynamic slechts met één interface verschilt met elkaar. Deze interface, genaamd IAnimatable, zorgt ervoor dat een item niet gewoon een texture vertoont, maar een animatie vertoont.

Diagram

Description automatically generated with medium confidence

Diagram

Description automatically generated Screenshots terug te vinden in ClassDiagram.cd

**DEPENDENCY INVERSION PRINCIPLE**

Ik maak ook gebruik van het DIP-principe. Ik maak allemaal instanties aan van de klasse Entity. Pas achteraf wordt adhv de constructor bepaalt welke subklasse van Entity het gaat zijn.

Text

Description automatically generated

Text

Description automatically generated

Screenshots terug te vinden in Classes/NovaGame.cs