**UITLEG GAMEPLAY**

Het concept van het spel is dat je bent gecrashed met een spaceship op een onverkende planeet en je op zoek bent naar een deel van het schip dat zorgt voor de energietoevoer. Je komt onbekende stenen wezens tegen die gedeeltelijk zijn opgebouwd uit hetzelfde blauwe material in ertsvorm dat jouw ruimteschip doet werken. Zo vind je het schipdeel samengesmolten met een rockgolem terug.

Het menu bestaat uit een texture en een lijst van plaatsbare objecten, 2 geanimeerde, en 2 statische buttons. Eens op start gedrukt ga je naar het volgende level. De player kan links en recht bewegen, springen en bukken en ook sluipen door bukken en bewegen te combineren. Elke staat van de speler heeft een animatie.

De player wordt door colliders en zwaartekracht op de grond gehouden. De camera volgt de speler mee zodat de levels niet beperkt worden door enkel het scherm. De grond bestaat uit meerdere blocks gegenereerd door een tilemap, ook het gras en plantjes op de achtergrond zijn blokken, maar zonder colliders.

Elk menu en level heeft een eigen deuntje. Ook de boss heeft er 1.

De achtergrond beweegt vertraagd mee met de camera om een parallax effect te krijgen. Eveneens de enemies beschikken over verschillende states van bewegen en animaties. De tweede enemy verschilt van de eerste in dat hij kan zweven.

De enemies kunnen damage krijgen, maar de speler ook. De enemies beschikken over harmful colliders, jouw wapen ook als je slaat.

Als je hp op 0 komt dan komt er een game over scherm en wordt je herleidt naar het hoofdmenu om terug opnieuw te beginnen.

Eens je door het eerste level bent ga je naar het volgende en kom je terecht in een donkere grot. Hier staat de boss te wachten met het deel van het schip in zijn arm.

Om het level uit te spelen moet je de hp-balk van de boss tot 0 verlagen door de boss te slaan met je zwaard. De boss beukt op de grond, wat veel damage kan doen. Eens verslagen loop je naar het einde van de grot en is het spel gedaan. Je komt terrecht op het game over menu waar je terug naar het hoofdmenu kunt navigeren.

Ik wil nog even laten weten dat ik alle textures zelf heb getekend buiten de achtergrond met het bos en de grot. Ook de muziek heb ik niet zelf gemaakt, maar is wel copyright free.

**UITLEG MONOGAME CODE SHOWCASE**

GAMENOVA

In de GameNova klasse gebeurt alles

Hierin zitten scenes die gemanaged worden door de scenemanager

SCENE, MENU, LEVEL

Een scene is een abstracte klasse dat zowel een level of een menu kan zijn.

Ze zijn grotendeels hetzelfde. In beide kan aan de hand van een List, positioneerbare objecten worden toegevoegd, dit kan een Static Item, Dynamic Item of een button zijn, alles dat voortkomt van de interface IPositionable.

Het grootste verschil tussen een menu en een level is dat in een level een controleerbare player en een spritemap aanwezig moet zijn.

TERRAIN

De tilemap van de terrain klasse genereerd solide blokjes waarop de player kan staan door collision en gravity. Ook genereerd het void blokjes waar de speler kan doorwandelen voor bijvoorbeeld estetische elementen toe te voegen zoals gras of een achtergrond van blokjes. De tilemap genereerd ook enemies. De terrain klasse beschikt ook over een achtergrond die zo geporgrammeerd is om een parallax effect te geven.

ALIVE, DEAD - SCENEMANAGER

Players en enemies erven over van de interface ICanDie en beschikken over de states Alive en Dead. Dit wordt gebruikt in de Scenemanager om de GameOver scene weer te geven wanneer de speler sterft. Ook wordt door het bereiken van het einde van een level, de volgende scene gespeeld. Na het laatste level is het sowieso gedaan en kom ook terrecht op het game over menu. Vanuit de gameover menu kun je terug gaan naar het hoofdscherm waar je kunt kiezen om te starten of te stoppen.

CREATURE, PLAYER, ENEMY - ENTITY

Een creature, de abstracte klasse van een player en een enemy, beschikt over heel veel capaciteiten tegenover een Entity wat slechts ergens gepositioneerd en getekend kan worden. Deze kan bewegen, geanimeerd worden, botsen met andere objecten, beschikken over health en damage krijgen en zelfs doodgaan. Dit komt omdat het beschikt over meerdere interfaces. Het neemt de interfaces over van een entity nl. IVisible en daarnaast IPolyMorph, IRigid, IMovable, IAnimatable, ICanRunAnimated, ICanCrouchRunAnimated, ICanAttackAnimated, ICanCrouchAnimated, ICanJumpAnimated. Deze hebben allemaal een aparte functie en kunnen dan zodanig ook gecombineerd worden om andere samengestelde Klassen te vormen. Ik ga niet dieper ingaan op elke interface want dan kom ik niet rond met 10 minuten. Ze zouden normal gezien wel voor zich spreken.

ALLCOLLIDERS

Wat wel belangrijk om te vermelden is dat er een static class is voor de colliders van alle objecten die colliders hebben te verzamelen. Zo kan elke instantie van wat dan ook dat beschikt over colliders steeds een andere collider uit de lijst terug vinden en alle variabelen lezen van dat object.

Wat een speler verschilt van een enemy, is de soort inputreader. Een player maakt gebruik van een keyboard reader en een enemy van een scriptreader.

CAMERA

De camera klasse is zo geprogrammeerd dat het een Ipositionable kan volgen, in dit geval de player.

**SOLID**

**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