# Globale uitleg

Fables of Darkness (FoD) is een spel gebaseerd op het oude tekst-based adventure game genaamd Zuul. FoD heeft de basis van Zuul als een structuur gebruikt om hiermee een meer uitgebreid spel te maken. Enkele nieuwe functies zijn bijvoorbeeld:

* Deuren en sloten
* Enemies die bevochten kunnen worden
* Kunnen winnen
* Verschillende objecten oppakken en gebruiken
* Trap-doors
* En zo nog meer functies die origineel niet in Zuul zitten.

Om FoD te winnen, moet de speler zichzelf uit de diepe grot halen.

## Werking

Het spel bestaat uit 3 grote onderdelen, onderverdeeld in meerdere kleinere onderdelen. De 3 onderdelen werken met elkaar samen om een groot geheel te maken. De 3 onderdelen zijn:

* Game (Bestaat uit Start, Game, CommandParser, Randomizer en Battle)
* Command (Bestaat uit Parser, CommandWords, CommandWord en Command)
* Variabelen en game mechanics (Bestaat uit Levels, Item, Enemy, Room en Player)

Game

Zorgt voor de verwerking van alle commands en de juiste werking van het spel. Game kan vergeleken worden met het “brein” van het spel. Waar alle losse componenten worden aangestuurd op basis van de input van de speler.

Command  
Zorgt ervoor dat de rest van het spel de inputs van de gebruiker kan verstaan en gebruiken. Ook wordt hier gekeken naar valide inputs of als de speler eigenlijk een ander commando bedoelde (Zoals “north” in plaats van “go north”)

Variabelen en game mechanics  
hier wordt bepaald welk spel klaargezet wordt voor de speler. Op dit moment is er maar 1 level. Echter kan dit simpelweg uitgebreid worden zodra er een tweede level gemaakt wordt in de Level klasse. De rest van het spel zal zich dan aanpassen op wat er aangemaakt wordt in deze klasse. Verder worden de speler, kamers, vijanden en items klaargezet om het spel te gaan spelen.

# Game

Game zorgt voor het hoofdmenu en de grote “speel” loop. Deze loop houdt bij elke iteratie in de gaten of de speler gewonnen heeft, wil stoppen of game-over is. Is dit allemaal niet het geval dan start Game een nieuwe “ronde” op en wacht het spel weer op user input.

## CommandParser

CommandParser verwerkt alle commando’s die gegeven zijn. Zo wordt er gekeken wat het eerste woord is van het commando en vervolgens gekeken wat de speler wilt doen met dit commando op basis van het tweede woord.

## Battle

Battle zorgt voor een tweede spel, binnen het spel. Een soort van “minigame”. De speler heeft hier een kans te verliezen en hierbij het spel te beëindigen. De speler moet door spelen tot 1 van de 3 resultaten is behaald: De speler verslaat het monster / Het monster verslaat de speler / De speler rent weg van het monster (Kans % of dit lukt)

## Randomizer

De randomizer wordt gebruikt om verschillende componenten van het spel random te maken. Echter is het een vrij simpele randomizer die integers terug geeft op basis van een maximale waarde.

De randomizer wordt gebruikt bij de volgende onderdelen:

* Plaatsen van verschillende items in kamers / Bepalen van damage die gedaan wordt op de speler / Bepalen van damage die gedaan wordt op de monsters / Plaatsen van monsters in verschillende kamers

De randomizer zorgt ervoor dat elk spel als anders ervaren kan worden. Dit zorgt ervoor dat spelers een reden hebben op het spel opnieuw te spelen.

## Levels

Levels wordt gebruikt om een level klaar te zetten voor de speler. Hier worden de volgende dingen bepaald en vastgezet om het spel te spelen:

### Kamers

Kamers krijgen een korte descriptie, een lange descriptie en een standaard boolean om te bepalen of deze kamer items mag hebben.

Vervolgens wordt er gemeld welke kamer aan welke kamer ligt op basis van north / east / south / west. Zo kan de speler zich verplaatsen van kamer naar kamer.

Ook wordt er bepaald welke kamer een sleutel nodig heeft, welke kamer de winkamer is en er wordt genoteerd welk kamernummer het is om dat vervolgens later te gebruiken.

### Items

Items krijgen een naam, descriptie, categorie, minimale damage, maximale damage, armor-rating, healamount, gewicht, waarde en of ze opgepakt mogen worden.

Daarna wordt er bepaald met de randomizer waar ze komen te liggen. Enkele items worden niet gerandomized om de flow van het spel intact te houden. Als alles random was, konden er situaties ontstaan waar de speler niet kon winnen.

#### Key-items

Key items zijn nooit randomized. Deze items zijn gekoppeld aan een kamer waar de sleutel gebruikt kan worden. Zonder deze sleutels kan er niet verder gespeeld worden in het spel.

### Vijanden

Items worden gemaakt met een naam, descriptie en hoe veel min/max damage ze kunnen doen.

Daarna worden ze met behulp van de randomizer in verschillende blokken gezet. Aangezien de speler niet de baas al moet tegen komen in de eerste kamer, bijvoorbeeld.

## Player

De speler wordt aangemaakt. Met deze klasse wordt de speler gemaakt met alles wat hierbij nodig is. Zo krijgt de speler een beginwapen, begin-armor, health-points, maximaal gewicht dat de speler kan dragen en hoe veel damage de speler kan doen. Met al deze info kan het spel gespeeld worden.

## Room

Room wordt gebruikt om een kamer aan te maken en info op te roepen van deze kamer. Als de speler in de kamer staat, kan er gegeken worden wat alle aspecten zijn van deze kamer. Zoals of dit een sleutel-kamer is, vijand aanwezig is of een item op de grond ligt voor de speler. Ook wordt er gekeken naar alle uitgangen zodat de speler zich kan verplaatsen naar een andere kamer.

## Item

Item wordt gebruikt om items in het spel te krijgen. Ze worden aangemaakt met een aantal standaard waarden die daarna overschreven kunnen worden (zoals meer damage geven indien de speler sterken wordt). Ook kan elk item info aan het spel terug geven om ervoor te zorgen dat het spel weet waar dit item allemaal goed voor is. (Een sleutel bijvoorbeeld)