Technisch Ontwerp

Diving for Treasure: The Sequel

**2 April 2021**

**Studenten**

Laurens van Brecht (640101)

Alex Cheng (634967)

**Docent**

Mark van der Maas

# Inhoudsopgave

[**Inhoudsopgave**](#_m5k5kghq3woo) **2**

[**Inleiding**](#_1fob9te) **3**

[**Klassendiagram**](#_3znysh7) **4**

[**Klassen**](#_2et92p0) **5**

[DivingForTreasure](#_c4zb9154srnh) 5

[Player](#_1ksv4uv) 5

[Bomb](#_3dy6vkm) 5

[Coin](#_2s8eyo1) 5

[OxygenTank](#_lnxbz9) 6

[BombSet](#_1t3h5sf) 6

[CoinSet](#_17dp8vu) 6

[OxygenTankSet](#_35nkun2) 6

[Shark](#_itg04karcygp) 7

[Clock](#_4d34og8) 7

[PowerUp](#_44sinio) 7

[ResistanceField](#_2ntfdglzfm65) 7

[Boat](#_x2kkobdusqwx) 8

[EndScreen](#_a8lc9ohp84um) 8

[GameObject](#_4y6b24utkok9) 8

[**Interfaces**](#_1y810tw) **9**

[Object Interaction](#_9zz6h8dnb6sb) 9

[Object Score](#_gysxkyqdgae3) 9

[ObjectSet](#_alxal6wrpsmb) 9

[OneOfAKingObject](#_ycov0ll5r6bi) 9

[**Gebruikte bronnen**](#_2xcytpi) **10**

# Inleiding

Zoals al bij het functioneel ontwerp is gezegd, is dit spel, Diving for Treasure: The Sequel, gebaseerd op het spel Diving for Treasure dat ik heb moeten maken tijden het vak Structured Programming Development (SPD).

Naast het functionele ontwerp met een uitleg en analyse van ons gewenste spel, waarin u overtuigd werd wat voor spel ik wou gaan programmeren, is er ook een technisch ontwerp opgesteld. In dit document zullen is te zien hoe het spel is ontwikkelt door middel van een klassendiagram en daarbij een korte uitleg wat er precies in de klassendiagram weergeven is, met daarbij de verbanden tussen elke klasse / interface.

# Klassendiagram

TO DO:

UML KLASSEN DIAGRAM

# Klassen

## DivingForTreasure

Deze klasse is in principe het hart van het gehele spel. In deze klasse zal de main staan van het spel. De DivingForTreasure klasse wordt extend door de klasse GameEngine, waaruit functies zoals setupGame() en update() worden overgenomen en overschreven. Verder wordt de state van het spel scherm bijgehouden in de Screen klasse die hieruit geïnitialiseerd wordt.

## Screen

Dit is een abstracte functie die spawnScreenObjects() en deleteScreenObjects() aanroept. Deze klasse maakt gebruik van een zogenaamde Strategy pattern, dit houdt in dat deze schermen eenvoudig vervangen kunnen worden door een andere wanneer het programma gewoon draait.

## Playscreen

Dit is de klasse met het eerste scherm dat aangeroepen en getoond wordt. Deze klasse erft alle methoden van Screen en initialiseert daarna de speler en spawner. Wanneer hij de spawnScreenObject() functie aanroept zal er een speler geplaatst worden met daarbij alle betreffende spel objecten door de spawner. Ten slotte heeft deze klasse ook nog een deleteScreenObjects() functie die alle spawner objecten kan verwijderen van het spel.

## Endscreen

Ook de Endscreen klasse erft alle methoden van Screen. Verder bevat deze klasse eenvoudig wat er getoond moet worden wanneer het spel eindigt, in ons geval is dit niets.

## Player

De Player klasse is erg essentieel voor ons spel, aangezien dit object door de speler zelf bestuurd wordt. Deze klasse wordt extend door de klasse SpriteObject en implementeerd de interface ICollidableWithGameObjects. Een aantal belangrijke punten die bijgehouden wordt in de Player klasse zijn:

* Zuurstof wordt bijgehouden doormiddel van incrementOxygen(), decrementOxygen(), depleteOxygen(), resetOxygen() en checkOxygenLevel().
* De speler moet natuurlijk kunnen bewegen, dit wordt geregeld door de keyPressed() functie.
* Object collision wordt geregeld met de geïmplementeerde interface, deze checkt of een object in aanraking komt met een andere object in gameObjectCollisionOccurerred(), deze krijg de gehele lijst met objecten mee om te controleren met welke objecten de Player in aanraking kan komen.

## Treasure Bag

## Spawner

## InteractableObject

## Treasure

## Coin

## Diamond

## Enemy

## Bomb

## Shark

## Aid

## OxygenTank

# Gebruikte bronnen

LucidChart. (z.d.). Tutorial UML-klassendiagram. Geraadpleegd op 18 mei 2020, van <https://www.lucidchart.com/pages/nl/tutorial-klassendiagram>

Wikipedia-bijdragers. (2020, 11 mei). Unified Modeling Language. Geraadpleegd op 18 mei 2020, van <https://nl.wikipedia.org/wiki/Unified_Modeling_Language>