Technisch Ontwerp

Diving for Treasure: The Sequel

2 April 2021

Studenten
Laurens van Brecht (640101)
Alex Cheng (634967)

Docent
Mark van der Maas

Inhoudsopgave

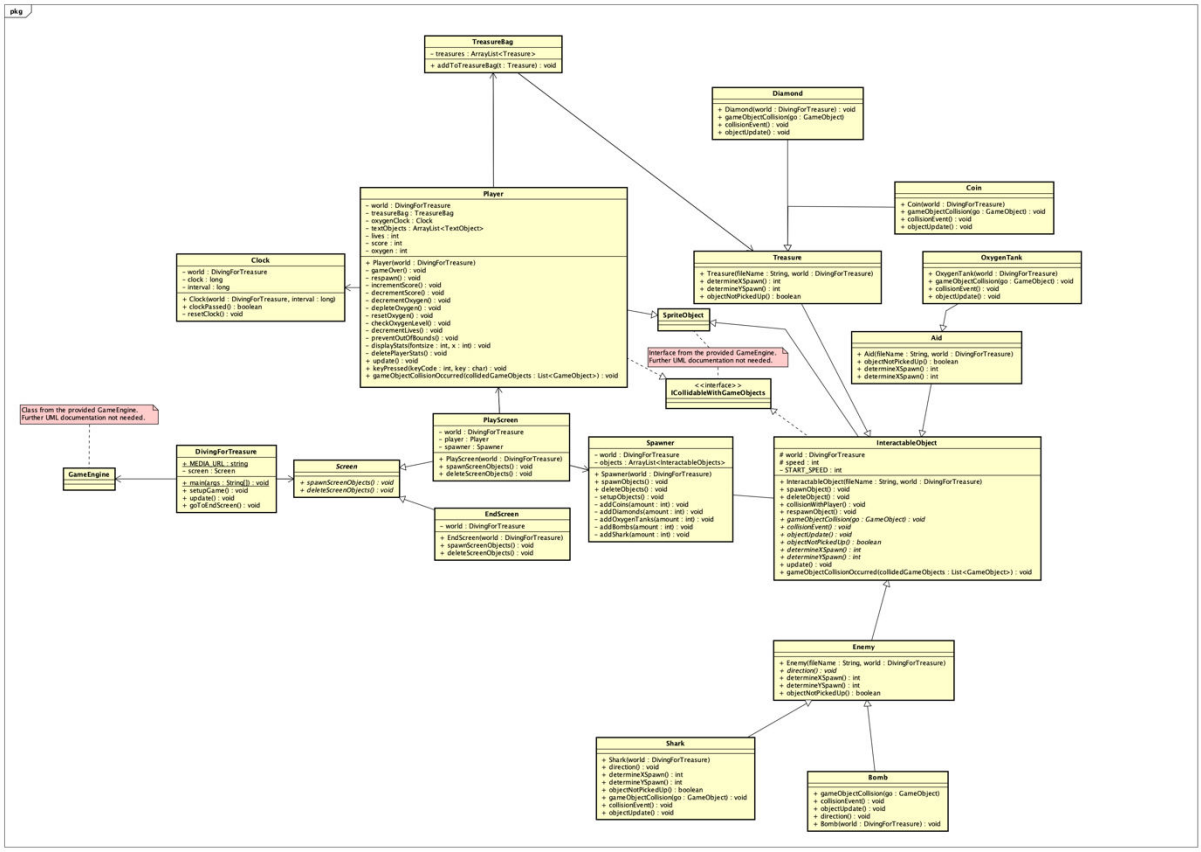
Inleiding

Zoals al bij het functioneel ontwerp is gezegd, is dit spel, Diving for Treasure: The Sequel, gebaseerd op het spel Diving for Treasure dat wij hebben 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 we zouden gaan programmeren, is er ook een technisch ontwerp opgesteld. In dit document 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

Het volgende klassendiagram is opgesteld voor ons spel, Diving for Treasure: The Sequel:



Afbeelding 1: Het volledige klassendiagram van ons spel.

(Voor een duidelijker beeld bekijk het aangeleverde .asta bestand)

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 klasse met de abstracte functies spawnScreenObjects() en deleteScreenObjects(). 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

De player kan items in het spel oprapen, deze worden bijgehouden in de TreasureBag klasse, dit wordt simpelweg gedaan doormiddel van een array list en met hierbij een addToTreasureBag functie.

Spawner

Deze Spawner klasse is erg belangrijk voor het spel, hierdoor worden alle objecten daadwerkelijk in het spel geplaatst. Doormiddel van de InteractableObject klasse krijgt de spawner een lijst met objecten. Daarbij wordt in de setupObjects() bepaald hoeveel van iedere object er toegevoegd moet worden aan het spel, en is er een aparte functie voor elk object zelf om deze toe te voegen.

InteractableObject

De InteractableObject extend SpriteObject en implementeert ICollidableWithGameObjects. Zoals boven al werd beschreven regelt de InteractableObject de lijst met objecten. Daarbij heeft deze abstracte klasse allerlei abstracte functies zoals het bepalen van de X&Y spawn met determineXSpawn() / determineYSpawn(). Daarnaast een handler voor als er collision plaatst vindt. In deze gameObjectCollisionOccured() functie wordt dan de collisionWithPlayer() functie aangeroepen die het object zal respawnen indien het object in aanraking is gekomen met de player.

Treasure

Ook de Treasure klasse is een abstracte klasse. Deze extend van de InteractableObject klasse. In deze klasse wordt simpelweg de eigenschappen van een schat bepaald. De X spawn wordt gezet in determineXSpawn() gebaseerd op een willekeurig getal tussen 0 en de world width. En de Y spawn wordt gezet in determineYSpawn() en deze is gelijk aan de world height. Ook wordt er nog bepaald op welke hoogte de objectNotPickedUp() functie moet gebeuren, in ons geval willen we dat wanneer een schat bovenuit het scherm gaat.

Coin

Een van de op te pakken schatten is de coin, deze Coin klasse wordt extend door Treasure en hierbij wordt eigenlijk alleen de betreffende image geïnitialiseerd.

Diamond

Net zoals coin is dit ook een schat waarbij de klasse extend wordt door Treasure. Ook hierin wordt de image geïnitialiseerd.

Enemy

Enemy is een abstracte klasse die InteractableObject extend. Deze klasse is vergelijkbaar met de Treasure klasse. Hierin worden ook de X & Y locatie van de spawn bepaald en wordt er een hoogte gezet wanneer objectNotPickedUp() aangeroepen wordt.

Bomb

De Bomb klasse is een van de enemies van de player. Deze klasse extend daarom ook vanuit de Enemy klasse. Hierin wordt de image geïnitialiseerd en ook de direction bepaald waar het object naartoe gaat.

Shark

De Shark klasse is in principe hetzelfde als de Bomb klasse, deze klasse extend ook Enemy. Het grootste verschil tussen deze twee klasses is dat shark op een andere positie in het spel gezet wordt. Hierbij is de determineYSpawn() gebaseerd op een willekeurige positie ergens tussen het midden van het spelscherm en onderaan het spelscherm.

Aid

Ten slotte is er nog een abstracte klasse genaamd de Aid klasse. Deze klasse heeft extend weer InteractableObject en overschrijft weer de determineXSpawn(), determineYSpawn()en objectNotPickedUp() functies

OxygenTank

Deze OxygenTank methode extend de Aid klasse en initialiseerd de correcte afbeelding van het object.

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