Technisch ontwerp Bubble trouble

Door: Kachung Li (studentnummer 642552) en Max Lodders (studentnummer 635277)

Docentnaam: Bart van der Wal

Coursenaam: OOPD

HAN Arnhem 25-03-2020

# Inhoud

[1. Inleiding 2](#_Toc36647026)

[2. Klassen 3](#_Toc36647027)

[2.1 Inleiding 3](#_Toc36647028)

[2.2 Terrain 3](#_Toc36647029)

[2.3 Player 4](#_Toc36647030)

[2.4 Bubble 5](#_Toc36647031)

[2.5 Powerup 6](#_Toc36647032)

[2.5.1 PowerupProjectileSpeed 6](#_Toc36647033)

[2.5.2 PowerupMoveSpeed 6](#_Toc36647034)

[2.6 Projectile 7](#_Toc36647035)

[2.7 Projectiletrail 7](#_Toc36647036)

[2.8 HUD 7](#_Toc36647037)

[2.9 Score 8](#_Toc36647038)

[2.10 Lives 8](#_Toc36647039)

[2.11 Timer 8](#_Toc36647040)

[3. Conclusie 9](#_Toc36647041)

[4. Bijlage 10](#_Toc36647042)

# 1. Inleiding

In het vorige document hebben wij het functioneel ontwerp gemaakt van het spel Bubble Trouble. In dit document lichten we nog een aantal begrippen toe die wij vanuit het functioneel ontwerp hebben verwezen. Daarna maken we de klassendiagram van ons spel en lichten we onze keuzes van alle objecten binnen de diagram toe. De klassendiagram is bijgevoegd in Hoofdstuk 4 Bijlage.

Na aanleiding van het laatste feedback rondje hebben wij het gebruik van polymorfie toegelicht bij de klasse Powerup. Ook hebben wij de hoofdstukken opnieuw gesorteerd zodat het meer in lijn is met het FO.

# 

# 2. Klassen

## 2.1 Inleiding

In dit hoofdstuk beschrijven wij de klassen en hoe ze met elkaar samenhangen.

Optionele klassen worden in hun kopje met (optioneel) aangeduid.

## 2.2 Terrain

Terrain erft over van view omdat het de spelwereld weergeeft Terrain implementeert IalarmListener omdat hij dit nodig heeft om te weten wanneer de leveltimer afgelopen is. In terrain slaan we informatie over het level op, specifiek:level: een numerieke waarde die bepaalt op welk level de speler zit.

* Bubbles: een arraylist van alle bubbels die in het huidige level bewegen. Dit doen we om te weten wanneer er geen bubbels meer over zijn.
* Players: een arraylist van alle objecten van de klasse Player. dit attribuut wordt gebruikt om te kijken of er nog spelers met levens over zijn
* checkLevelComplete: kijkt of de lengte van de arraylist bubbels 0 is. als dit zo is returned hij true
* checkGameOver: kijkt of bij alle spelers of lives op 0 staat. als dit zo is returned hij true
* toNextScreen: kijkt naar condities en voor afhankelijk daarvan acties uit

- als checkLevelComplete true returned verhoogt hij level met 1.

- als checkGameOver true returned gaat hij naar het eindscherm

## 2.3 Player

Een player is de karakter binnen het spel die bestuurbaar is door de speler. Player erft over van AnimatedSpriteObject binnen de GameEngine pakket, omdat de player een animatie heeft als de player beweegt. De player heeft de volgende eigenschappen:

* De player kan naar links en naar rechts bewegen. De richting van de player past zichzelf aan op basis van welke kant de speler laatst heeft bewogen.
  + Binnen het GameObject klasse van de GameEngine zit er een keyPressed methode. Hierbinnen luistert de game naar de input van de gebruiker (Zie FO voor de specifieke inputs).
* De player kan een projectiel afvuren
  + De player krijgt een canFire boolean toegewezen die standaard op true staat.
  + Zodra een player een projectiel afvuurt, staat de canFire boolean op false, totdat het projectiel verdwenen is. Dan staat canFire weer op true.
* De player start met 3 levens(Lives) en de player verliest een leven bij aanraking van een bubbel
  + Dit gebeurt door de functie subtractLives binnen de Lives klasse aan te roepen.
* De player kan met andere objecten binnen het spel colliden
  + Afhankelijk van het object die de player aanraakt, triggert er een bepaald event. Het object waarmee de player collide houdt de klasse Player bij met een arrayList attribuut.
    - gameObjectCollisionOccurred voert dan de methode uit op basis van de GameObjecten die er in de lijst zitten.
  + Zodra de event met het object is uitgevoerd, haalt de klasse Player het object waarmee er is gecollide, weer weg van de arraylist.

## 2.4 Bubble

Een bubble is een bal die rond stuitert binnen het spel. Bubble erft over van SpriteObject binnen het GameEngine pakket, omdat Bubble een object is waarmee een player kan colliden. De bubble class heeft geen animatie, dus erft bubble niet over van AnimatedSpriteObject. Een bubble binnen het spel heeft de volgende eigenschappen:

* Aan het begin van het level gebruiken we setGravity om de gravity op 2 te zetten.
* Een bubbel stuitert binnen het spelwereld(Terrain) rond.
  + Door de bounce methode blijft een bubbel constant bewegen. Op contact met een ground tile veranderd hij de ySpeed dmv setySpeed naar -ySpeed. op contact met een muur tile doet hij hetzelfde met de xSpeed.
* Een bubbel kan zichzelf naar twee nieuwe bubbels splitten
  + Het splitten van een bubbel gebeurt alleen als een projectile een bubbel raakt.
    - Tijdens het splitten maakt het spel twee nieuwe bubbels aan die een grootte hebben van de helft van de orginele bubbel. Vervolgens haalt het spel binnen de terrain klasse de bubbel weg van de ArrayList binnen de Terrain.
    - Beide bubbels nemen twee verschillende richtingen nadat de twee nieuwe bubbels zijn toegevoegd in het lijst. Als de eerste bubbel dan bijvoorbeeld een richting neemt van 270 graden bij het splitten, dan neemt de tweede bubbel het tegenovergestelde richting, dus 90 graden.
  + Het aantal score van een bubbel is al beschreven in het Functioneel Ontwerp. Deze score is opgeslagen binnen het bubbel object, en krijgt de speler toegevoegd bij zijn huidige score wanneer de bubbel vernietigd is.

## 2.5 Powerup

Powerup is een abstracte klasse en erft over van SpriteObject, omdat het een object is waarmee de player kan interacteren, en in het terrain plaatsvindt. In deze klasse maken wij gebruik van polymorfie. Er zijn namelijk meerdere soorten powerups, en het kan altijd zijn dat er in de toekomst meerdere powerups in het spel ontwikkeld wordt. Elk powerup heeft ook maar één doel, dat is namelijk om de speler te helpen met het vernietigen van bubbels.

Elk powerup doet dit op een ander manier.Als we een powerup aanmaken geven gebruiken we setGravity om het te laten vallen. Het verdwijnen van een powerup doen we met behulp van een timer op het moment dat deze de grond aanraakt. Er moet 3 seconden afgelopen zijn nadat de speler de powerup niet oppakt voordat deze verdwijnt. Het aanraken van de grond doen we m.b.v. de interface ICollidableWithTiles van de GameEngine. De speler behoudt elke powerup totdat het huidige level afgelopen is.

### 2.5.1 PowerupProjectileSpeed

PowerupProjectileSpeed is een powerup die de speler kan pakken om de snelheid van het projectiel kan verhogen. PowerProjectileSpeed erft over van powerup. Deze classes heeft dezelfde attributen en methoden als Powerup, omdat de superclass abstract is. Daarnaast heeft deze powerup een ander methode, wat per klasse verschilt.

In dit geval heeft deze klasse de methode enhanceProjectileSpeed, die de snelheid van het projectiel omhoog haalt door de speedMultiplier van de speler te pakken, en deze op te tellen met 0.5, waardoor het projectiel sneller beweegt. De klasse Player zet de boolean canFire sneller om, waardoor de speler sneller projectielen kan afvuren.

### 2.5.2 PowerupMoveSpeed

Komt op hetzelfde neer als PowerUpProjectileSpeed, maar in plaats van dat de projectile speed omhoog gaat, gaat de snelheid van de speler omhoog. De speed haalt de Powerupmovespeed klasse van de Player klasse in plaats van de Projectile klasse.

## 2.6 Projectile

Een projectile is een object die de speler af kan vuren. Projectile erft over van SpriteObject binnen de GameEngine pakket, want een projectile is een object binnen het spel die geen animatie heeft. Een projectile heeft de volgende eigenschappen.

* De projectile komt alleen in het spel wanneer de speler op de spatie toets heeft gedrukt. Dit werkt op hetzelfde manier als het bewegen van een Player (Zie hoofdstuk 2.4).
* De projectile beweegt naar boven zodra deze afgevuurd is. De projectile kan maar 1 richting op, namelijk het noorden.
* De snelheid van de projectile hangt deels af van de multiplier die erin zit.
  + De multiplier van de snelheid start op 1.0, en de multiplier krijgt een waarde van 0.5 opgeteld als er een powerup is opgepakt.
* De projectile verdwijnt van het spel als de projectile de bovenkant van het terrain heeft geraakt. De detectie hiervan bepaalt de klasse met behulp van de interface ICollidableWithTiles.

## 2.7 Projectiletrail

De projectiletrail is een lijn die de projectile achterlaat. Projectile erft over van SpriteObject binnen de GameEngine pakket.

* Projectiletrail beweegt op hetzelfde snelheid als projectile
  + Dit betekent dat de speed op hetzelfde snelheid ligt als die van projectile. De speed van de projectile krijgt de projectiletrail door getSpeed van projectile aan te roepen.

## 2.8 HUD

HUD erft over van Dashboard omdat het een dashboard moet tekenen in de onderkant van het scherm. HUD is de klasse die alle informatie die de speler nodig heeft weergeeft onder in het scherm (optioneel) hij doet dit voor elke speler. HUD heeft de volgende eigenschappen:

- een paar pixels af van de bovenkant van de hud tekenen we het tijdslimiet weergegeven als rode balk die langzamerhand leegloopt

- hieronder tekenen we de score van de speler aan de linkerkant voor speler 1 (optioneel) en aan de rechterkant voor speler 2

- de levens numeriek/ (optioneel) als sprites (per Player) worden rechts van de score getekend (optioneel) en links van de score voor speler 2

## 2.9 Score

Score implementeert IPersistence, zodat de score opgeslagen kan worden in een datastring. Dit is vooral bedoeld om highscores bij te houden (optioneel). Anders erft score nergens van over.

We gebruiken de klasse score om bij te houden wat de score van de speler is.

Ook onthouden we wat de score van de speler was toen hij het level begon zodat we de score naar dat cijfer kunnen veranderen als de speler een leven verliest en het level herstart. Verder gebruiken we score om de high score bij te houden en extra punten te geven voor de resterende tijd als een level voltooid is.

* previousScore: de score die de player had voordat hij aan het huidige level begint. Zoals net beschreven, houden we dit bij voor het geval dat een Player door een bubbel geraakt wordt.
* currentScore: de huidige score van de speler
* multiplier: Dit is de multiplier van de clear bonus, dus wanneer de player de level heeft afgemaakt. In het FO hebben wij de multiplier de getal 20 gegeven. Deze multiplier gebruiken wij om het bonus score te berekenen als de speler alle bubbels heeft vernietigd.

## 2.10 Lives

Lives is het aantal levens die een speler heeft. Lives erft nergens van over. Dit is een klasse die alleen bestaat zodat andere classes er gebruik van kunnen maken. Tot nu toe maakt alleen de klasse Player gebruik van de klasse Lives.

* currentLives attribuut slaat het aantal huidige levens op van het object die er gebruik van maakt. In dit geval is dat de Player klasse.
* subtractLives geeft het aantal levens terug met 1 afgetrokken, dus 3 - 1 = 2 levens bijvoorbeeld.

## 2.11 Timer

Timer erft over van Alarm binnen de OOPG engine, omdat er binnen de timer naar een bepaalde tijd moet luisteren. Dit meet de timer in seconden, en de class Alarm ondersteund dit m.b.v. de methode getSeconds.

pastTime is de additie hier. in deze variabele slaat de timer op wanneer hij aangemaakt is. Dit is nodig om de balk van de timer te laten leeglopen op de HUD.

# 3. Conclusie

In dit document hebben wij toegelicht hoe de klassen binnen het klassendiagram beschreven. Daarnaast hebben wij ook de samenhang binnen de klassen beschreven. Ook hebben we per klasse de eigenschappen beschreven. In de bijlage hebben wij de volledige klassediagram erin gezet. Over de implementatie kunnen we op dit moment nog geen conclusie trekken.

# 4. Bijlage

