Functioneel ontwerp Bubble Trouble

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

Docentnaam: Bart van der Wal

Coursenaam: OOPD

HAN Arnhem 16-03-2020

# Inhoudsopgave

[1. Inleiding 2](#_Toc35617034)

[2. Eisen spel 3](#_Toc35617035)

[2.1 Inleiding 3](#_Toc35617036)

[2.2 Eisen 3](#_Toc35617037)

[3. Over het spel 5](#_Toc35617038)

[3.1 Inleiding 5](#_Toc35617039)

[3.2 Wat is Bubble Trouble? 5](#_Toc35617040)

[3.3 Acties en bewegingen 5](#_Toc35617041)

[3.4 Objecten 5](#_Toc35617042)

[3.4.1 Bubbels 5](#_Toc35617043)

[3.4.2 Powerups 6](#_Toc35617044)

[3.4.3 Projectiel 6](#_Toc35617045)

[3.5 Overige elementen 6](#_Toc35617046)

[3.5.1 Score 6](#_Toc35617047)

[3.5.2 Level 6](#_Toc35617048)

[3.5.3 Levens 7](#_Toc35617049)

[3.5.4 Timer 7](#_Toc35617050)

[4. Schermen/werking spel 8](#_Toc35617051)

[4.1 Inleiding 8](#_Toc35617052)

[4.2 Beginscherm 8](#_Toc35617053)

[4.3 Speelscherm 9](#_Toc35617054)

[4.4 Eindscherm 11](#_Toc35617055)

[4. MoSCoW prioriteiten 12](#_Toc35617056)

[4.1 Inleiding 12](#_Toc35617057)

[4.2 Must have 12](#_Toc35617058)

[Should have 12](#_Toc35617059)

[Could have 12](#_Toc35617060)

[Won’t have 13](#_Toc35617061)

[5. Conclusie 14](#_Toc35617062)

[6. Verwijzingen 15](#_Toc35617063)

# 1. Inleiding

In dit document beschrijven wij het functioneel ontwerp van het spel Bubble Trouble.

Het spel dat we maken is gebaseerd op een flash game uit 2004 genaamd “Bubble struggle”. Het doel van het spel is om alle bubbels te vernietigen binnen de tijdslimiet en daarmee zo veel mogelijk punten te verdienen. In dit document zetten we eerst de minimum eisen die het spel nodig heeft op een rij om dit spel succesvol af te ronden. Daarna beschrijven we de aanwezige acties en objecten binnen het spel. Daarna beschrijven we de aanwezige elementen binnen elk scherm en de werking van het spel. Vervolgens prioriteren wij de ontwikkeling van het spel in de vorm van de MoSCoW methode. Als laatst schrijven we de conclusie voor dit document.

# 2. Eisen spel

## 2.1 Inleiding

In dit hoofdstuk zetten we de minimale eisen op een rij om dit spel succesvol af te ronden en beschrijven we kort hoe we dit kunnen bereiken.

## 2.2 Eisen

* Het FO is verzorgd (zie ook de ICA-controlekaart) en bevat voldoende informatie om het TO en de realisatie op te baseren.
  + Om het Functioneel ontwerp te verzorgen, kijken wij naar de Controlekaart van de ICA en moeten we aan de punten voldoen die daarin staan.
  + Om de TO en de realisatie hierop te baseren, zullen wij de basis van dit spel al in dit document uit moeten denken. Ook zullen beslissingen al hierin opgenomen moeten worden
* Het TO is verzorgd (zie ook de ICA-controlekaart), bevat voldoende informatie om de realisatie op te baseren en is in lijn met het FO.
  + Het verzorgen van de TO gebeurt op dezelfde manier als de FO
  + In de FO zijn een aantal beslissingen al genomen, maar het kan zijn dat het niet mogelijk is om bepaalde functies te realiseren. In het TO omschrijven we dan hoe het anders zou kunnen, of dat het helemaal niet mogelijk is om een functie te implementeren.
* Alle zelfgebouwde publieke methoden en zelf toegevoegde publieke attributen zijn voorzien van documentatie (bij voorkeur met Javadoc, zie bijv. <https://nl.wikipedia.org/wiki/Javadoc>)
  + Er moet documentatie gegenereerd worden met behulp van een generator uit de Javadoc. Zie Technisch ontwerpdocument voor meer informatie.
* Er zijn minimaal acht eigen klassen gerealiseerd die voldoende verschillend zijn en die op z'n minst enige functionaliteit bevatten (klassen met vrijwel identiek gedrag dat ook met andere attribuutwaarden te realiseren zou zijn, tellen niet mee, en ook interfaces of volledig abstracte klassen tellen niet mee).
  + Naast de klassen die wij gebruiken vanuit de GameEngine waar wij gebruik van maken, moeten wij ook 8 klassen zelf ontwikkelen. zie Technisch ontwerpdocument voor meer informatie.
* Er wordt minimaal één van de interfaces uit de engine succesvol toegepast.
  + Een soort van een klasse met abstracte methodes erin. Wij moeten 1 interface van de GameEngine in onze spel toepassen. Zie Technisch ontwerpdocument voor meer informatie.
* Er wordt overerving toegepast binnen de zelfgebouwde klasses (dus niet alleen overerven van objecten uit de engine).
  + Er moeten dus child classes erin zitten die methodes van de parent classes kunnen overnemen. Zie Technisch ontwerpdocument voor meer informatie.
* Er wordt minimaal eenmaal polymorfie toegepast binnen zelfgebouwde klassen (dus: er is de mogelijkheid van polymorfie ingebouwd, en deze wordt daadwerkelijk gebruikt).
  + Een object moet meerdere vormen kunnen hanteren waardoor klassen flexibel zijn. Zie Technisch ontwerpdocument voor meer informatie.
* Klassen en methoden hebben duidelijke verantwoordelijkheden:
  + Naam komt overeen met de taak
  + Namen van attributen dekken de lading
  + Zo weinig mogelijk dubbele code
  + Geen static variabelen, tenzij daar een goede reden voor is
* Alle studenten uit het groepje begrijpen alle code en kunnen deze tijdens het assesment toelichten.

(Han, sd)

# 3. Over het spel

## 3.1 Inleiding

In dit hoofdstuk beschrijven wij algemene informatie van het spel zelf. Hierin beschijven wij wat het spel is, de omgeving binnen het spel, de acties en bewegingen die er binnen het spel gedaan kan worden en de objecten binnen het spel.

## 3.2 Wat is Bubble Trouble?

Bubble Trouble is een spel dat is gebaseerd op een flash game uit 2004 genaamd “Bubble struggle”. De speler is met zijn karakter binnen het spel ingesloten in een ruimte met een bepaald aantal bubbels. Het aantal bubbels wat in de ruimte zit, hangt af van het level. Een level is een ander woord voor niveau, waarin er bijgehouden wordt hoe ver de speler komt. Hoe hoger het level is, hoe moeilijker het spel over het algemeen wordt. Het doel van Bubble Trouble is om alle bubbels te vernietigen binnen de tijdslimiet en daarmee zo veel mogelijk punten te verdienen. De speler ziet alle objecten en zijn karakter van de zijkant, omdat er een aantal objecten binnen het spel naar boven stuiteren of omhoog geschoten worden. Dat is lastig te zien vanuit de bovenkant.

## 3.3 Acties en bewegingen

De speler bestuurt het karakter met de toetsenbord. De linker- en rechterpijltjes van het toetsenbord gebruikt de speler om het karakter horizontaal aan te sturen en de spatiebalk gebruikt de speler om via zijn karakter een projectiel af te vuren. Het projectiel wordt afgevuurd op de huidige positie van de karakter van de speler.

## 3.4 Objecten

Een object is een voorwerp binnen het spel. De speler kan met zijn karakter met een object interacteren, waardoor er een bepaald aantal condities gebeuren, op basis van met welk object er geïnteracteerd wordt.

### 3.4.1 Bubbels

Bubbels zijn de tegenstanders in dit spel. Een bubbel begint met een diameter van 64 pixels. Bubbels stuiteren over het veld heen. Ze proberen niet actief de speler uit te schakelen, maar als de speler met zijn karakter een bubbel aanraakt verliest hij een leven en herstart het huidige level. De speler kan een bal vernietigen of verkleinen door een projectiel af te vuren.

Als de speler een bubbel aanraakt met een projectiel of de spoor van een projectiel splitst de bubbel zich op. Er komen dan twee bubbels met de helft van de grootte van de orginele bubbel op het veld, dus 32 pixels in diameter, die dan naar de buitenste richting stuiteren, en dan verder stuiteren. Die bubbel kan dan nog een keer splitsen, waardoor er twee bubbels komen die weer de helft van de grootte van de vorige bubbel is, dus 16 pixels in diameter. Zodra deze bubbel neergeschoten is, dan is de bubbel definitief vernietigd.

Als een bubbel een ander bubbel aanraakt, dan stuiteren ze beide door elkaar heen. Het maximale aantal bubbels die er op het scherm kunnen, is 12, op level 3.

### 3.4.2 Powerups

Als er een bubbel neergeschoten is, is er 50% kans dat er een powerup naar beneden valt. Een powerup is een object die een speler helpt om bubbels makkelijker neer te schieten. Er zijn twee powerups in dit spel, namelijk een powerup dat ervoor zorgt dat een projectiel sneller naar boven gaat zodat, en een powerup dat ervoor zorgt dat de speler sneller beweegt. Het is niet de bedoeling dat een powerup nadelig is voor de speler. Als de powerup de grond aanraakt, dan blijft de powerup voor 3 seconden op de grond liggen, voordat deze uiteindelijk verdwijnt.

### 3.4.3 Projectiel

De speler vuurt een projectiel met zijn karakter af als de speler op de spatiebalk drukt. Als de projectiel een bubbel aanraakt, dan verdwijnt het projectiel, en wordt de bubbel gesplitst of vernietigd, zoals beschreven in Hoofdstuk 3.4.1. Het projectiel verdwijnt ook als het projectiel het plafond raakt. Ook laat het projectiel een spoor achter, die de bubbel vernietigd of verkleind. De spoor blijft staan totdat het projectiel verdwijnt. De speler kan niet nog een projectiel schieten totdat de oude projectiel verdwenen is. De speler kan wel een powerup oppakken die ervoor zorgt dat een projectiel sneller tegen het plafond aankomt, waardoor de speler sneller kan schieten.

## 3.5 Overige elementen

Hierin beschrijven wij welke overige elementen er in het spel zitten naast de acties en objecten. Zie afbeelding 2 van hoofdstuk 3.3 voor een afbeelding van het spel waarin de overige elementen er in voorkomen.

### 3.5.1 Score

De speler start met een score van 0. De speler krijgt voor het neerschieten van de eerste bubbel 250 punten. Voor elke subsequente bubbel wordt er 50 punten afgetrokken van de 250, dus krijgt de speler bij de tweede bubbel 200 punten, en bij de laatste bubbel 150 punten. Als de speler het level gehaald heeft, dan krijgt de speler het aantal score dat uitgerekend is op basis van de resterende tijd. De formule voor het berekenen van de extra score is (huidige tijd / max tijd \* 100 = percentage van de tijd \* 20).

### 3.5.2 Level

De speler begint na het opstarten van het spel op het eerste level. Bij de eerste level is er 1 bubbel. Nadat de speler alle bubbels heeft vernietigd, mag de speler door naar het volgende level. Per level komt er een extra bubbel op het scherm, dus er zijn twee bubbels op level 2. Er zijn maximaal 3 levels in het spel. Als de speler al zijn levens is verloren, dan is de speler klaar met het spel en wordt het huidige level op het eindscherm weergegeven zodat de speler kan zien hoe ver de speler is gekomen.

### 3.5.3 Levens

De speler begint met 3 levens. Zodra een bubbel de speler aanraakt verliest de speler een leven en wordt het level gereset naar zijn orginele staat, waar de speler het huidige level opnieuw moet spelen.

### 3.5.4 Timer

De speler krijgt per level 1 minuut om een level te halen. De resterende tijd is gevisualiseerd door middel van een balk die langzaam leegloopt, te zien bij afbeelding 2 van hoofdstuk 3.3. De speler krijgt de tijd niet te zien zodat de speler de druk voelt om het level zo snel mogelijk uit te spelen.

# 4. Schermen/werking spel

## 4.1 Inleiding

In dit hoofdstuk leggen wij de werking van het spel en de aanwezige elementen uit. Dit leggen wij uit met behulp van een aantal schetsen.

## 4.2 Beginscherm

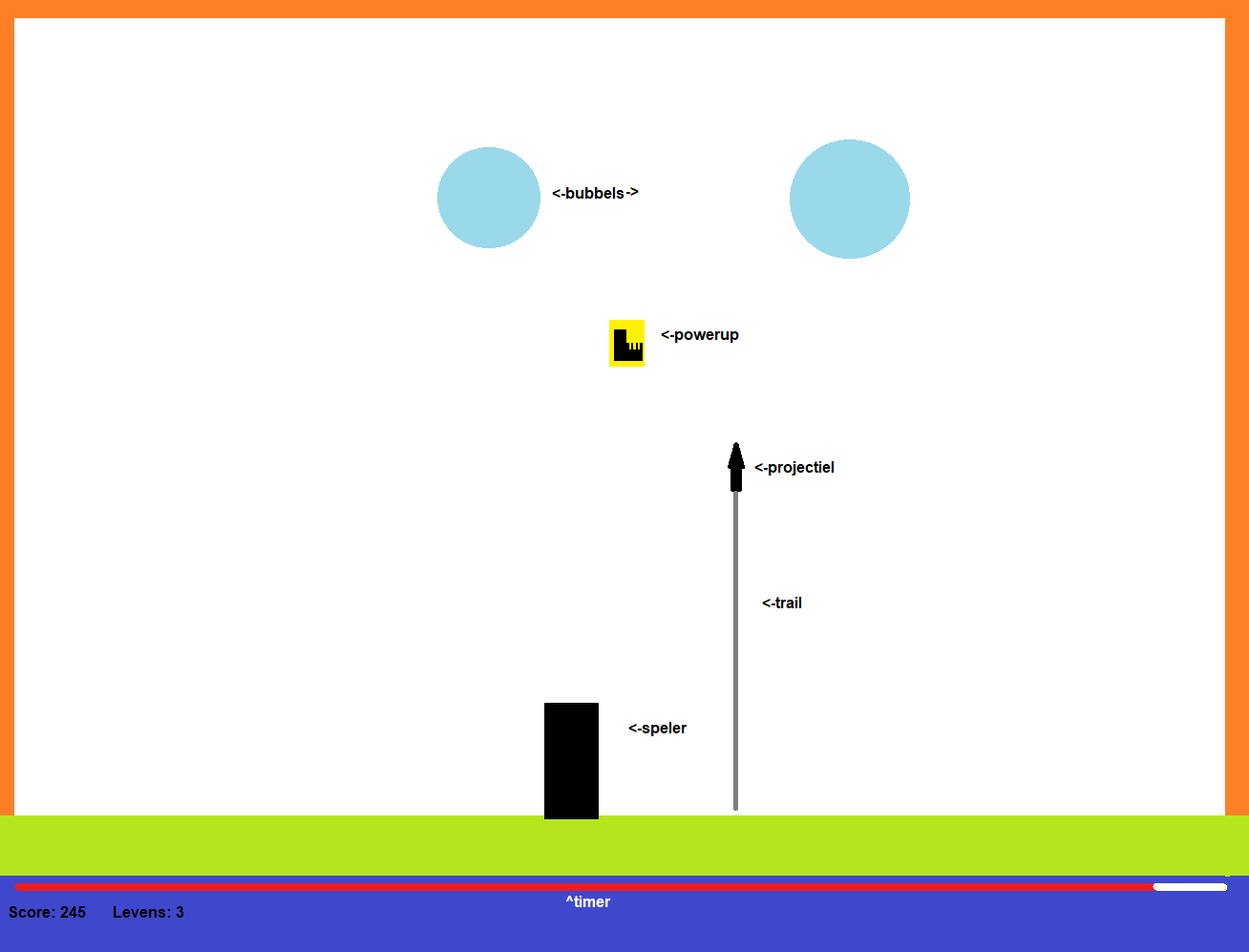


**Afbeelding 1 – Startscherm** (New Atlas, 2016)

Dit is het startscherm (Zie afbeelding 1 - Startscherm). De bedoeling van het startscherm is dat als speler het programma van dit spel opstart, niet meteen in het spel wordt gegooid. De speler kan op dit scherm op de Start knop te klikken. Zodra de speler dit doet, start het spel op en vindt er een transitie plaats van het beginscherm naar het speelscherm. Ook kan de speler ervoor kiezen om het spel af te sluiten door op de exit knop te klikken. Deze exit knop is meer bedoeld voor het geval dat er een fullscreen functionaliteit erin zit in dit spel.

De speler start het spel op en komt bij het beginscherm. Op het beginscherm kan de speler het spel starten door op de startknop op het scherm te klikken. Zodra de speler dit doet, wordt de speler naar het speelscherm gezet (?). Ook kan de speler ervoor kiezen om het spel af te sluiten door op de exit knop te klikken.

## 4.3 Speelscherm



**Afbeelding 2 – Speelscherm**

Het spel wordt gestart nadat de speler op de startknop heeft geklikt op het startscherm. Het scherm van het spel is 1080 pixels breed en 720 pixels hoog. De speler komt dan bij het speelscherm (Zie afbeelding 2 – Speelscherm). De speler komt met zijn karakter in het midden van het speelveld te staan. Het karakter van de speler is 32 pixels breed en 64 pixels hoog. Onderin het scherm staan de levens, de behaalde score van de speler en de timer in de vorm van een balk. Deze timer gaat af op het moment dat het spel is gestart.

Zodra de timer is gestart, kan de speler zijn karakter besturen. Er is een bubbel op de bovenkant van het scherm getekend, die naar beneden valt. Het aantal bubbels die er worden getekend, is beschreven in Hoofdstuk 3.5.2, maar er is altijd minimaal 1 bubbel op het scherm bij een level. Daarna stuiteren de bubbels heen en weer. De speler moet ervoor zorgen dat de bubbels vernietigd worden zonder dat hij geraakt wordt door een stuiterende bubbel. De bubbel kan de speler van alle kanten raken, behalve via de onderkant, want de bubbel kan niet door de grond heen zakken. De werking van de bubbel is beschreven in Hoofdstuk 3.5.3. De speler moet dus de bubbel neerschieten om te voorkomen dat diegene geraakt wordt. Als de speler dat heeft gedaan, is er een 50% kans dat er een powerup naar beneden valt. De werking van de powerup is beschreven in Hoofdstuk 3.4.2.

Als de speler alle bubbels heeft vernietigd, dan krijgt de speler extra score zoals beschreven in Hoofdstuk 3.5.1. Daarna gaat de speler door naar het volgende level. Het spel voert dan de hele proces zoals hierboven omschreven nog een keer uit, maar er komt dan een extra bubbel op het scherm. Als de speler het laatste level heeft gehaald zoals beschreven in hoofdstuk 3.5.2, of de speler verliest alle levens, zoals beschreven in hoofdstuk 3.5.3, dan stuurt het spel de speler naar het eindscherm toe.

## 

## 4.4 Eindscherm



**Afbeelding 3 – Eindscherm** (New Atlas, 2016)

Op het eindscherm ziet de speler de uiteindelijke behaalde eindscore en de level die de speler heeft behaald. Als de speler het spel gehaald heeft, dan ziet de speler op de titel bovenin in plaats van game over gefeliciteerd en print het spel het behaalde level niet. Als de speler op de start opnieuw knop klikt, dan begint het spel opnieuw. Als de speler op de naar menu knop klikt, dan gaat het spel weer terug naar het startscherm.

# 4. MoSCoW prioriteiten

## 4.1 Inleiding

“MosCoW is een samenstelling van beginletters die ergens voor staan. De 2x o zijn ingevoegd om het woord ‘moscow’ leesbaar te maken, zonder dat ze betekenis hebben. De M staat voor **Must-haves**, de S staat voor **Should-haves**, de C staat voor **Could-haves** en de W staat voor **Won’t-haves** of **Would-haves**.” (toolshero, sd)

## 4.2 Must have

“Het gaat hier om het minimale eisenpakket dat vooraf gesteld wordt en waaraan het eindresultaat moet voldoen. Zonder deze eis, faalt het project en is het product niet meer bruikbaar. Het is vereist voor een werkbaar product en er is geen alternatief. De Must-haves zijn essentieel. MUST wordt ook wel uitgelegd als het acroniem dat staat voor Minimum Usable SubseTs.” (toolshero, sd)

* Speler kan bewegen en projectielen afvuren
* Bubbels bewegen door het scherm en stuiteren van muren af
* Powerups spawnen en vallen naar de grond, blijven daar een tijdje en verdwijnen dan
* Score wordt weergegeven en is cumulatief over levels
* Timer werkt

## Should have

“Dit zijn aanvullende en zeer gewenste eisen, die wel een hoge prioriteit hebben, maar geen vereiste zijn voor een bruikbaar eindproduct. Zonder dat deze eisen worden ingewilligd is het product evengoed bruikbaar en met het voldoen aan de eisen krijgt het product alleen maar meerwaarde. Afhankelijk van de tijd kan er altijd later nog aandacht aan deze eisen worden gesteld.” (toolshero, sd)

* Icoontjes voor levens
* Bubbels vallen alsof de zwaartekracht de bubbels naar beneden trekt

## Could have

“Als er tijd voor is kunnen deze eisen altijd nog aan bod komen. Zo niet, dan is het geen probleem en heeft het geen nadelig gevolg voor het eindresultaat. De could-haves hebben minder prioriteit dan de should-haves. De optie wordt alleen meegenomen als er echt ruim tijd voor is om het te realiseren. Het wordt ook wel ‘nice to have’ genoemd en het is eerder een wens dan een harde eis.” (toolshero, sd)

* Geluid aka sound effects en evt achtergrondmuziek
* Particle effects op collision
* High scores
* Karakters met verschillende projectielen
* Full screen mogelijkheid

## Won’t have

“Het gaat hier om toekomstwensen die vaak niet mogelijk zijn of heel veel tijd kosten om te realiseren. Indien niet mogelijk, dan is het verstandig om geen energie hierin te stoppen. Is realisatie wel mogelijk dan zal er veel tijd (en geld) geïnvesteerd moeten worden en is er sprake van een would-have. Vaak worden would-haves in een vervolgtraject meegenomen.” (toolshero, sd)

* 2 speler modus
* Verschillende moeilijkheidsgraden (als in kleinere timers, snellere bubbels of meer levens)

# 5. Conclusie

In dit document hebben wij beschreven welk spel wij ontwikkelen voor dit beroepsproduct. Wij hebben de eisen op een rij gezet waaraan het spel moet voldoen. Daarna hebben wij een aantal elementen binnen het spel beschreven. Vervolgens hebben wij schetsen van een aantal schermen laten zien waarin deze elementen samen bij elkaar komen. Als laatst hebben wij de functies in een prioriteitenlijst gezet om het spel werkend te krijgen.

# 6. Verwijzingen

Han. (sd). *Onderwijsonline*. Opgehaald van Han.nl: https://onderwijsonline.han.nl/elearning/lesson/dNwlWL7y

New Atlas. (2016, 09 05). *Bubble-like "Memzyme" increases speed and efficiency of carbon capture*. Opgehaald van newatlas.com: https://newatlas.com/bubbles-memzyme-co2-capture/45253/

toolshero. (sd). *MoSCoW Methode*. Opgehaald van toolshero.com: https://www.toolshero.nl/project-management/moscow-methode/