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](#_Toc35522470)

[2. Acties en objecten 3](#_Toc35522471)

[2.1 Inleiding 3](#_Toc35522472)

[2.2 Acties 3](#_Toc35522473)

[2.2 Objecten 3](#_Toc35522474)

[2.2.1 Bubbels 3](#_Toc35522475)

[2.2.2 Powerups 3](#_Toc35522476)

[2.2.3 Projectiel 3](#_Toc35522477)

[2.2.4 Overige elementen 3](#_Toc35522478)

[3. Schermen/werking spel 4](#_Toc35522479)

[3.1 Inleiding 4](#_Toc35522480)

[3.2 Beginscherm 4](#_Toc35522481)

[3.3 Speelscherm 5](#_Toc35522482)

[3.4 Eindscherm 6](#_Toc35522483)

[4. MoSCoW prioriteiten 7](#_Toc35522484)

[4.1 Inleiding 7](#_Toc35522485)

[4.2 Must have 7](#_Toc35522486)

[Should have 7](#_Toc35522487)

[Could have 7](#_Toc35522488)

[(probably) Won't have 7](#_Toc35522489)

# 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. Het spel is te zien vanaf de zijkant. 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 gebruiker 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 de level. Een level is een ander woord voor niveau, waarin er bijgehouden wordt hoe ver de speler komt. Hoe hoger de 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 gebruiker 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 gebruiker 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 gebruiker 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 grootte van 32x32 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 16x16 pixels groot, die dan twee verschillende richtingen nemen, 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 8x8 pixels groot.. Zodra deze bubbel neergeschoten is, dan is de bubbel definitief vernietigd.

### 3.4.2 Powerups

Als er een bubbel neergeschoten is, is er een kans dat er een powerup naar beneden valt. Een powerup is een object die een speler helpt om bubbels makkelijker neer te schieten. Bijvoorbeeld een powerup dat ervoor zorgt dat een projectiel sneller naar boven gaat zodat of dat de speler sneller beweegt. Een powerup zou nooit nadelig moeten zijn voor de speler.

### 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

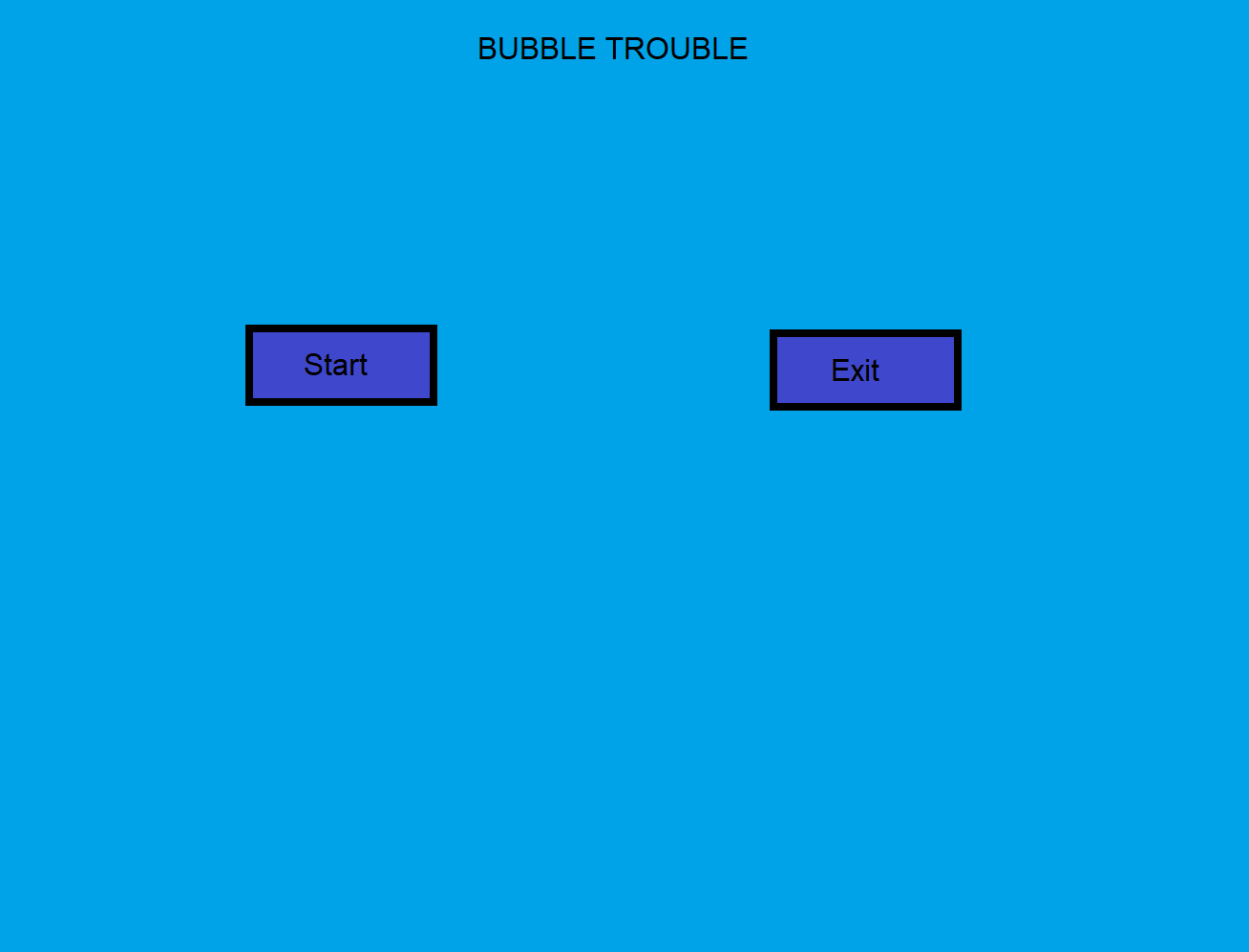
je huidige score en het aantal resterende levens worden onderin het scherm weergegeven. ook wordt de resterende tijd gevisualiseerd door middel van een balk die langzaam leegloopt.

# 3. Schermen/werking spel

## 3.1 Inleiding

## 3.2 Beginscherm

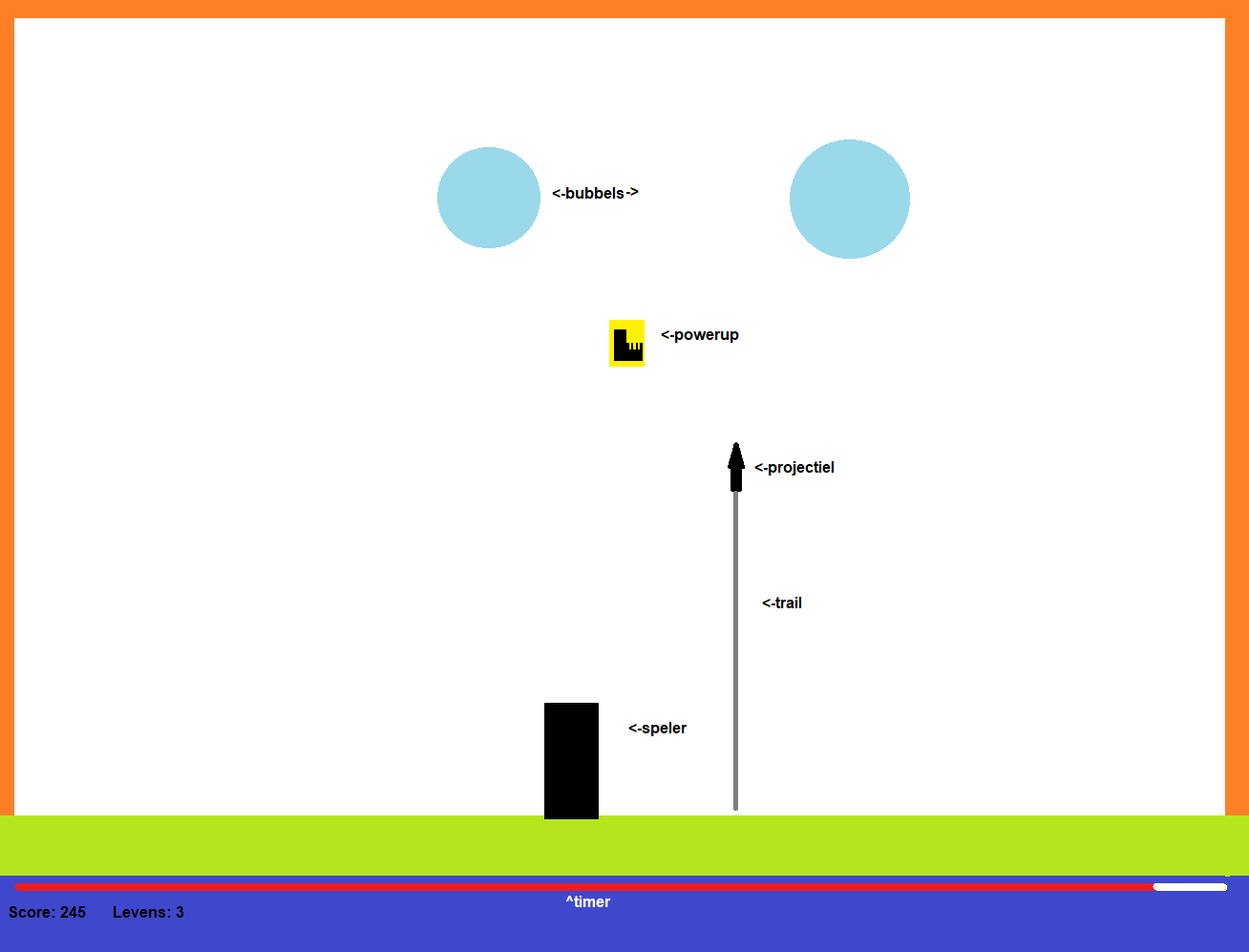
De speler start het spel op en komt bij het beginscherm. Op het beginscherm kan de speler het spel starten door op de startknop op het scherm te klikken. Zodra de speler dit doet, wordt de speler naar het speelscherm gezet (?). Ook kan de speler ervoor kiezen om het spel af te sluiten door op de exit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



## 

## 3.3 Speelscherm

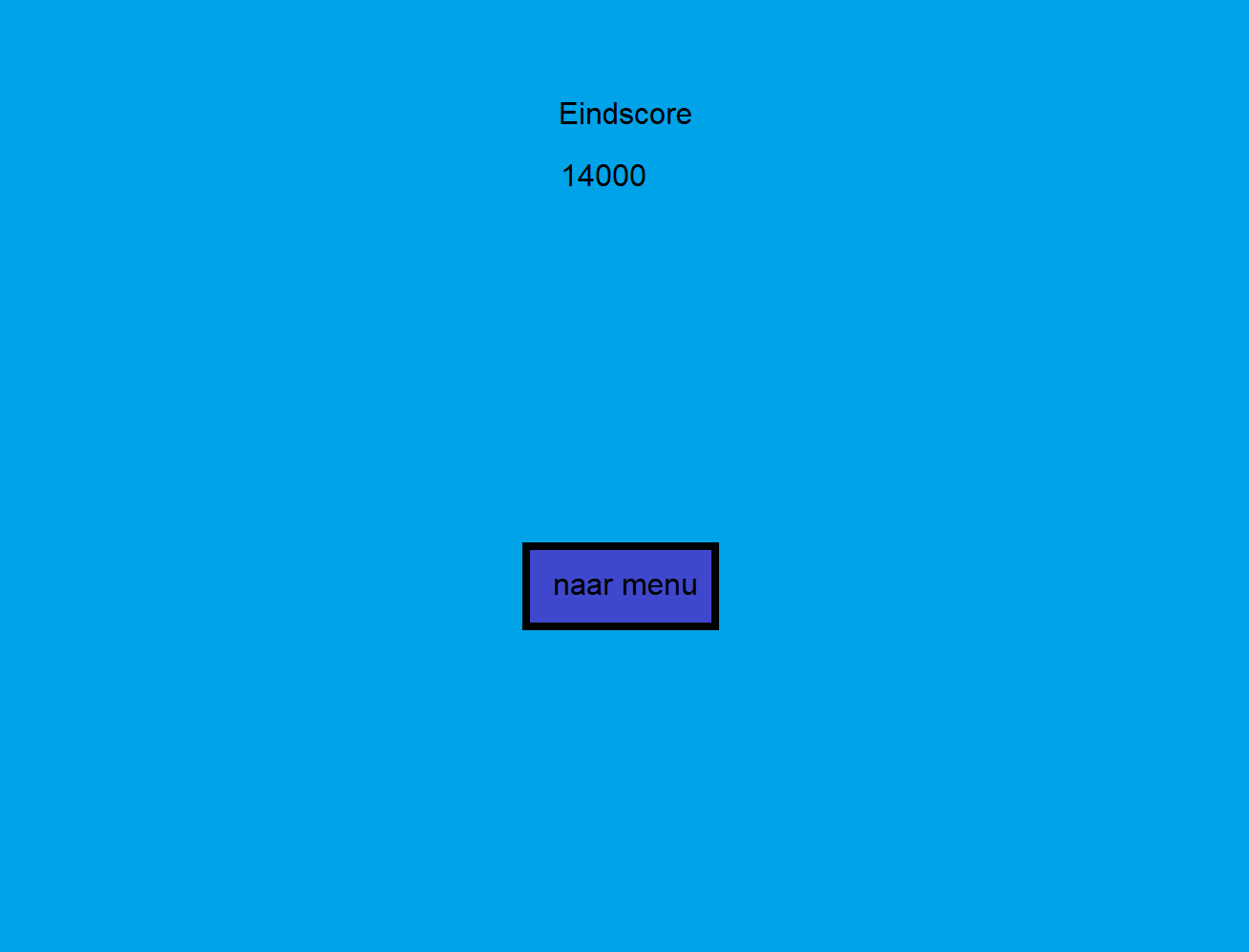
Het spel wordt nu? gestart en de speler ziet het speelscherm (verwijs naar figuur incl nr). Het karakter van de speler staat in het midden van het speelveld. In dit scherm kan de speler de levens en de score van zijn karakter zien. Ook ziet de speler een timer op het scherm staan. Deze timer gaat af op het moment dat het spel is gestart. De speler kan nu zijn karakter besturen. Er worden bubbels op het scherm geplaatst. De speler moet met zijn projectielen de bubbels vernietigen om verder te komen. De bubbel kan op de speler komen, waardoor de speler een leven verliest. Ook wordt het level teruggezet naar zijn oorspronkelijke staat. Als de speler een bubbel neerschiet, wordt de bubbel in twee delen gesplitst of vernietigd, en is er een kans dat er een powerup naar beneden valt. Als de speler deze oppakt, dan krijgt de speler een voordeel om het neerschieten van bubbels makkelijker te maken. Als de speler de powerup niet oppakt, dan verdwijnt deze na een bepaalde tijd. Zodra de speler alle bubbels heeft neergeschoten, wordt de speler naar het volgende level gestuurd. Dan wordt het hele proces nog een keer herhaald, maar komen er meer bubbels op het scherm om het niveau omhoog te halen. Er zijn een bepaald aantal levels, en als de speler de laatste level heeft gehaald, of de speler verliest al zijn levens, dan wordt de speler naar het eindscherm gestuurd.



## 

## 3.4 Eindscherm

Op het eindscherm ziet de speler de behaalde eindscore. Ook kan de speler op de knop op het scherm klikken om terug te gaan naar het startscherm.



# 

# 4. MoSCoW prioriteiten

## 4.1 Inleiding

## 4.2 Must have

* 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

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

## Could have

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

## Won't have

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

# 5. Conclusie

# 6. Verwijzingen

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