Project_

_Pacman

Functioneel Ontwerp en Technisch Ontwerp - OOPD

Versie: 1.4

Onderdeel: SEB OGP

Datum: 25-04-2021

David Bartenstein
(Studentnr. 403851)

HAN_UNIVERSITY
OF APPLIED SCIENCES

Titel:	Functioneel Ontwerp en Technisch Ontwerp - OOPD
Datum:	20 april 2021
Auteur:	David Bartenstein
Onderwijsinstelling:	HAN University of Applied Sciences
Opleiding:	HBO-ICT Software Development
Begeleider:	Peter Cornelissen

Voorwoord

Als onderdeel van de SEB module Object Georienteerd Programmeren heb ik – in eerste instantie samen met Marielle Gerrits – gekozen om een object georienteerde versie van Pacman te programmeren. Hoewel Marielle halverwege het project tot de conclusie kwam dat zij deze opleiding niet verder zou gaan volgen dit jaar ben ik toch verder gegaan om Project Pacman te voltooien en ik hoop dat ik hiermee een volwaardig ontwerp in heb kunnen leveren.

Inhoudsopgave:

FUNCTIONEEL ONTWERP ‘Project_Pacman’

1. Globale omschrijving van het spel:

In deze object georienteerde remake van het originele spel uit 1980 speel je als Pacman en is het je taak om monstertjes te ontwijken en alle snoepjes op te eten. Pacman wordt bestuurd door middel van de pijltjestoetsen en alle snoepjes waar hij mee in aanraking komt eet hij automatisch op. Het spel is afgelopen wanneer alle snoepjes opgegeten zijn of wanneer Pacman door een monstertje geraakt wordt en zijn laatste leven daardoor verliest. Het speelveld van de game wordt in gangetjes onderverdeeld door middel van muur-‘tiles’ waar Pacman en de monstertjes niet doorheen kunnen, en in deze gangen liggen de snoepjes die Pacman moet opeten om punten te verzamelen, maar daarin bewegen de monstertjes die Pacman moet ontwijken zich ook willekeurig voort.

Voor de speler is het dus een kwestie van inschatten welke kant de monstertjes op gaan en hoe snel hij de snoepjes voor hun neus op kan eten.

2. Perspectief

Als speler bekijk je het speelveld in vogel-perspectief. Je ziet dus het hele veld van bovenaf, al zie je de spelkarakters van de zijkant waardoor het geheel een soort pseudo-perspectief wordt.

3. Doel van het spel

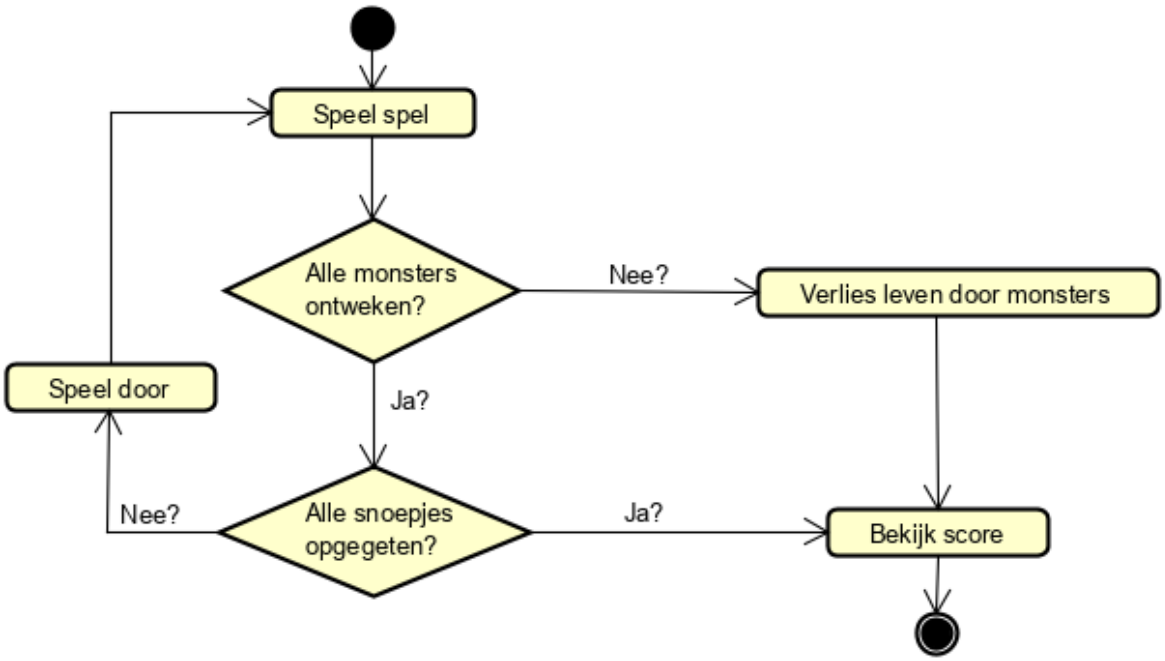
De speler moet zoveel mogelijk punten verzamelen door snoepjes op te eten en daarbij proberen om niet door een monstertje opgegeten te worden.

4. Globale flow van het spel

Zodra het spel start zijn er maar twee mogelijkheden waarop het spel beeindigd wordt:

1. Als Pacman geraakt wordt door monsters en zijn laatste leven verliest
2. Als alle snoepjes opgegeten zijn

Zolang dat niet het geval is moet er doorgespeeld worden. Hieronder staat dit schematisch weergegeven in de vorm van een flowdiagram:



5. Spelonderdelen

Het spel moet in ieder geval de volgende elementen bevatten:

- Pacman (het door de speler te besturen karakter)
- Monstertjes (willekeurig bewegende vijanden – bewegen zich voort door dezelfde gangen als Pacman)
- Een speelveld (afgebakend terrein binnen het scherm met een achtergrond)
- Een Scorebord (gedeelte van het scherm waar de score en levenspunten zichtbaar zijn)
- Snoepjes (statische voorwerpen - door de gangen verspreid – leveren punten op)
- Muurtjes (statische voorwerpen – bepalen de bewegingsruimte van zowel Pacman als monstertjes)

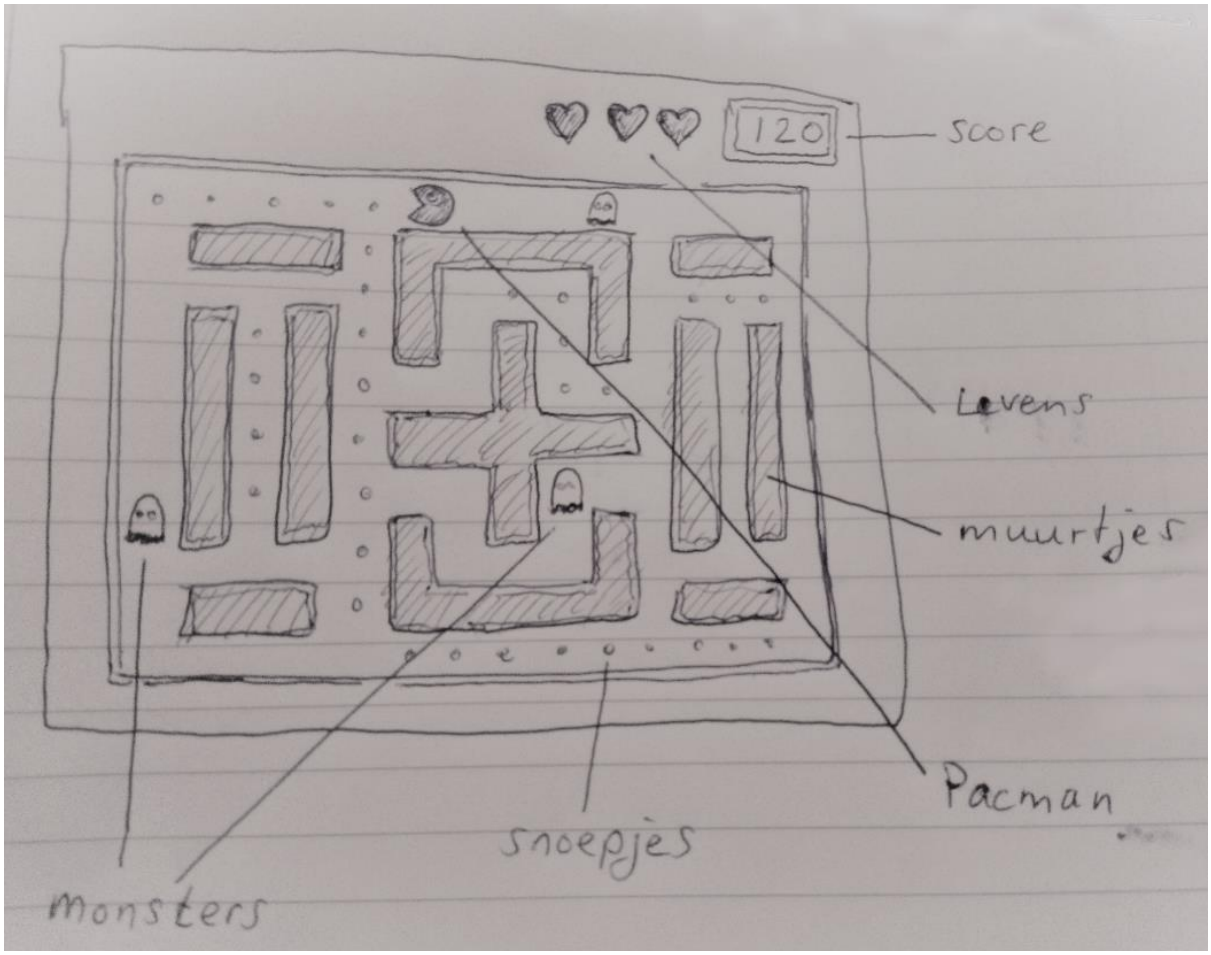
6. Besturing

De speler kan Pacman besturen met behulp van de pijltjestoetsen; zolang de speler een pijltjestoets ingedrukt houdt zal Pacman in die richting blijven bewegen. Wanneer Pacman tegen een muurtje aan gemanouvreerd wordt stopt hij met voortbewegen en komt pas weer in beweging als er een andere richting gekozen wordt.

De monstertjes zijn voortdurend in beweging. Ze bewegen in een random richting en kiezen een nieuwe random richting wanneer ze tegen een muurtje aan botsen. Monstertjes kunnen dus niet bestuurd worden.

7. Schermontwerp

Naast de monsters, de snoepjes, de muurtjes, en Pacman zelf, is er in het speelveld ook een scorebord zichtbaar waarop het aantal punten de hele tijd bijgehouden wordt. Dit zal rechtsbovenaan in het scherm in beeld gebracht worden. Hieronder zie je alles in beeld zoals het tijdens het spel eruit zal moeten gaan zien:



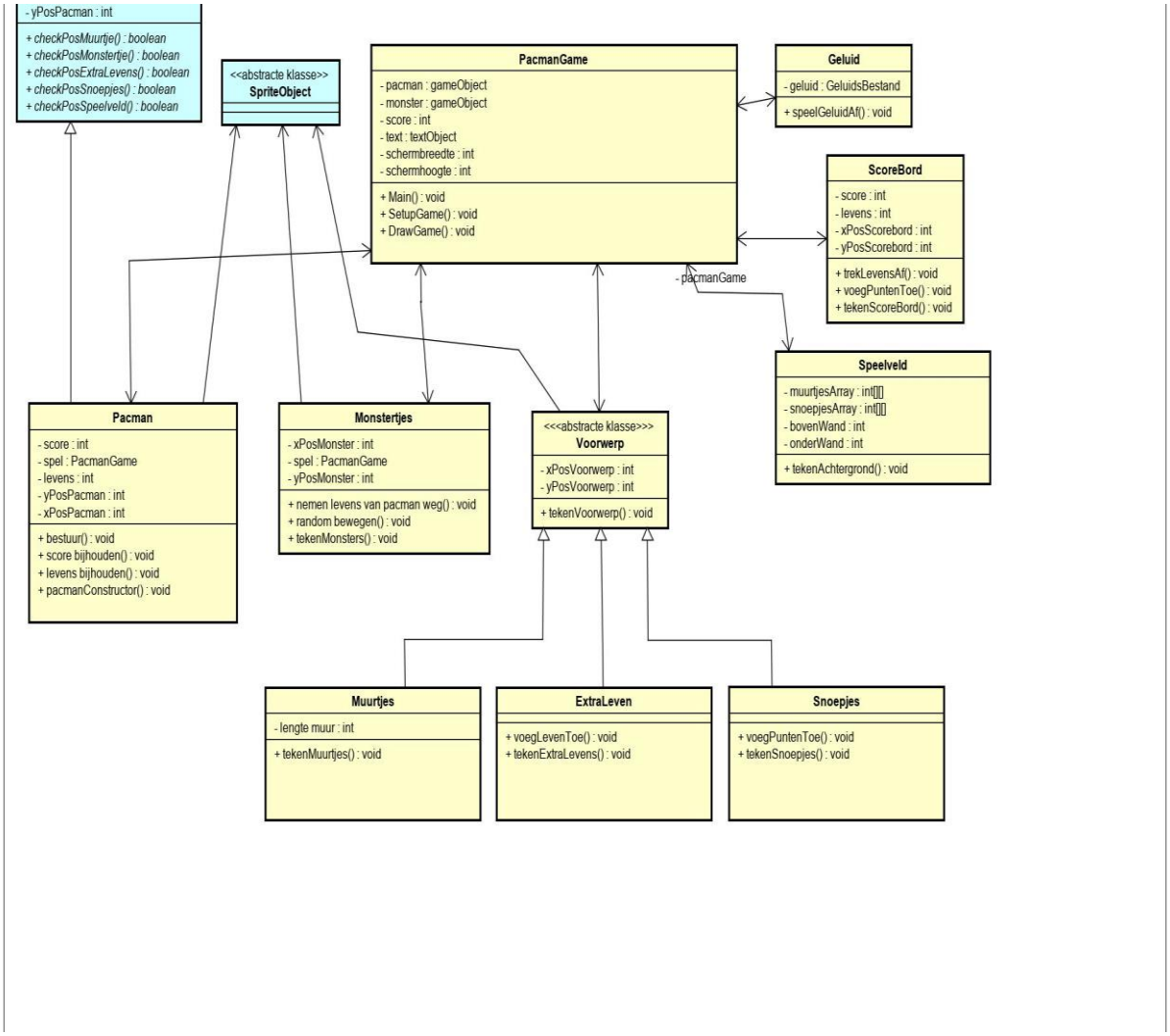
8. Vereisten in het game ontwerp:

De eisen die wij aan ons spel stellen geprioritiseerd volgens de MoSCoW methode:

	Het speelveld bestaat uit een doolhofje van muurobjecten waarbinnen pacman en de monstertjes kunnen bewegen maar waar ze niet doorheen kunnen	M
	De monstertjes bewegen zich in willekeurige richtingen door het speelveld	M
	Pacman kan punten verdienen door snoepjes op te eten die door de gangen van het speelveld verspreid liggen	M
	Als Pacman door een monstertje geraakt wordt of alle snoepjes opgegeten heeft is het spel afgelopen	M
	score + levens continu in beeld	M
	Pacman heeft meerdere levens waarvan er elke keer dat hij geraakt wordt door een monster eentje verdwijnt	S
	Als Pacman alle snoepjes op heeft gegeten begint er een nieuw level	S
	Pacman kan extra levens verdienen door speciale snoepjes op te eten die maar korte tijd verschijnen	C
	Pacman kan teleporteren naar een andere plek door op een teleporteer platformpje te gaan staan	W
	Het doolhof van muurtjes wordt elke keer willekeurig gegenereerd	W

TECHNISCH ONTWERP ‘Project_Pacman’

9. Klassendiagram:



Toelichting bij het klassendiagram:

Het spel zal onder andere de volgende klassen moeten bevatten met bijbehorende relaties en implementaties en overerving:

De klasse PacmanGame is de hoofdklasse die de main() methode bevat. Deze klasse extends de gameEngine. Deze klasse bevat een aantal variabelen, zoals:

- Een pacman object
- Een of meer monster objecten
- Een score
- Een schermbreedte
- Een schermhoogte
- De tekst voor in het scorebord

En methodes zoals:

- Een setupGame() methode. Daarin worden onder andere de afmetingen van het scherm, de plaatsing van de muurtjes-tiles, en het scorebord bepaald en geinitialiseerd.
- Een drawGame() methode. Hierin worden het speelveld, de muurtjes-tiles, en het scorebord getekend en daarna worden Pacman, de monstertjes, en de overgebleven voorwerpen via constructors aangemaakt.

De klasse Pacman is de spelersklasse die een bestuurbaar karakter aanmaakt. Deze klasse extends spriteObject en implementeerd collision. Deze klasse bevat een aantal variabelen, zoals:

- Een x positie
- Een y positie
- Een snelheid

En methodes zoals:

- Een pacman() methode. Dat is de constructor.
- Een bestuur() methode. Hiermee wordt bepaald welke richting pacman zich op beweegt.

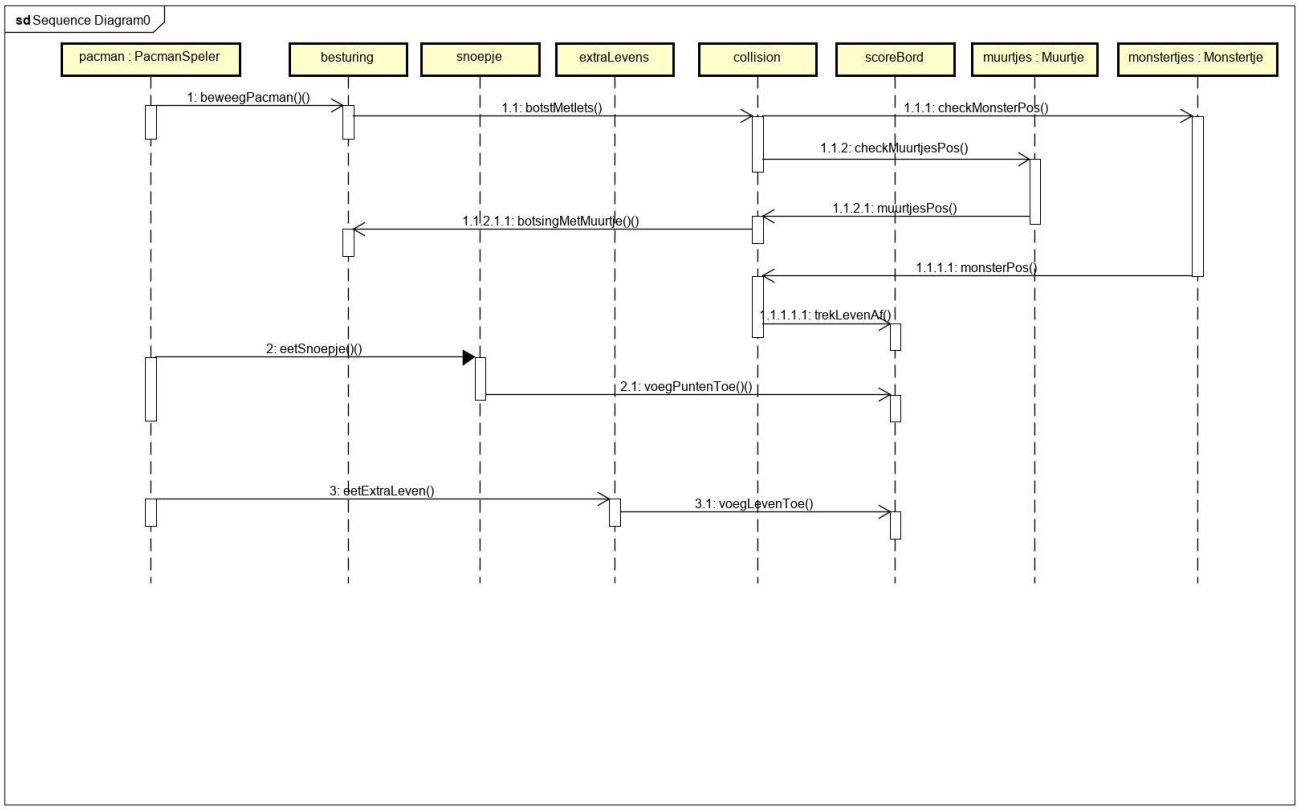
De klasse Monster is een non-playable karakter die in willekeurige richtingen voortbeweegt. Deze klasse extends spriteObject en implementeerd collision. Deze klasse bevat een aantal variabelen, zoals:

- Een x positie
- Een y positie
- Een snelheid
- Een richting

En methodes zoals:

- Een monster() methode. Dat is de constructor.
- Een neem-levens-weg() methode waarmee iedere keer dat pacman met het monster in aanraking komt een leven van pacman verwijderd wordt.
- Een beweegRandom() methode. Dit zorgt ervoor dat de monsters in willekeurige richtingen bewegen.

Sequentie Diagram:



Toelichting bij het sequentiediagram:

In dit sequentiediagram is in kaart gebracht wat er moet gebeuren wanneer de speler het pacman karakter voortbeweegt met behulp van de pijltjestoetsen. Uiteindelijke kunnen er levens afgetrokken worden of toegevoegd worden en kan de score verhoogd worden door het eten van snoepjes.

Collision speelt hier een essentiele rol in doordat alles afhangt van waar pacman mee in aanraking komt.

- Een botsing met een muurtje dwingt je als speler een andere kant op te sturen.
- Een snoepje verhoogt de score
- Een monster verlaagt het aantal levens / beeindigt het spel
- Een extra leven kan het aantal levens weer verhogen

Conclusie

Tijdens het in de praktijk uitwerken van dit ontwerp kwam ik erachter dat ik heel veel moest aanpassen. Onder meer het volgende is aangepast:

- Het speelveld is middels een tilemap in de PacmanGame klasse geplaatst
- Ik heb geen aparte besturingsinterface nodig gehad maar kon gewoon een keyPressed en keyReleased methode gebruiken binnen de pacman klasse
- Ik heb de status van Pacman en van de Game op een gegeven moment bij willen houden om mee te bepalen wanneer het spel over was en heb daar enums voor gebruikt in combinatie met switch case statements

Nawoord

Het heeft me bloed zweet en tranen gekost om de game engine te leren begrijpen en elke keer dat er vanalles gecrashed was alles weer aan de praat te krijgen. Ik vind het ontzettend interessant en als ik meer tijd had gehad had ik het ook nog heel erg leuk gevonden. Al met al ben ik behoorlijk tevreden met het eindresultaat maar als ik deze zomer wat tijd over heb zou ik er graag nog wat dingen aan verfijnen en stroomlijnen.

Literatuurlijst & Bronnen

Literatuurlijst

Craig Larman. (2010). *Applying UML and patterns* (3de editie).
Upper Saddle River: Prentice Hall PTR.
Heijmink, B. at al. 2014. *OOPD Processing Game Engine (OOPG)* Geraadpleegd in maart 2021 op https://github.com/HANICA/oopg

Notities

Algemene kennis voor het uitvoeren van dit applicatie ontwerp is opgedaan met behulp van onderstaande bronnen.

Pluralsight - https://app.pluralsight.com
Youtube - https://www.youtube.com/
Onderwijsonline - https://onderwijsonline.han.nl/elearning/content/5qv6gaqw

Beeldmateriaal

"background.png" voor het gamescherm - Photo by Sandro Gonzalez on Unsplash title: "green leaves with water droplets"

"pacman1.png" voor het spelerscharacter - https://www.cleanpng.com/png-pac-man-computer-icons-clip-art-pac-man-1214502/download-png.html

"pacman2.png" voor het spelerscharacter - https://www.subpng.com/png-299t07/