|  |
| --- |
| Centipedes\_  SPAD Beroepsproduct |

Job Zalmé

19 oktober 2022

INHOUDSOPGAVE

[INLEIDING 3](#_Toc117081460)

[1 Software-algoritmen 4](#_Toc117081461)

[1.1 collisionDetection() 4](#_Toc117081462)

[1.2 shortenCentipede() 4](#_Toc117081463)

[1.3 bananaHit() 4](#_Toc117081464)

[2 Architectuurplaatje 5](#_Toc117081465)

[3 Reflectie op codekwaliteit 6](#_Toc117081466)

[3.1 Goed gegaan 6](#_Toc117081467)

[3.2 Kan beter 6](#_Toc117081468)

[4 Testrapportage 7](#_Toc117081469)

INLEIDING

In dit bestand worden de drie belangrijkste software-algoritmes besproken, een architectuurplaatje, de testscenario’s en een reflectie over de geschreven code.

# Software-algoritmen

In dit hoofdstuk worden de drie belangrijkste software-algoritmen uit mijn uitwerking van de Centipedes casus besproken.

## collisionDetection()

In deze methode wordt er gecheckt of het hoofd van het staart op dezelfde positie wil staan als de staart zelf. Dit mag natuurlijk niet, omdat er anders door de staart gelopen kan worden. Door middel van deze methode zorgt het spel er nu voor dat deze actie niet mogelijk is. Het werkt als volgt:

Bijvoorbeeld, de speler beweegt een plek naar rechts.

De nieuwe positie van de speler wordt opgeslagen in een variabele genaamd testPosX.

collisionDetection kijkt of testPosX dezelfde positie heeft als dat een deel van de staart heeft.

Als dat zo is dan returnt de methode true.

Als collisionDetection true returnt wordt er vervolgens geen beweging gedaan.

Als collisionDetection false returnt kan de beweging gedaan worden.

## shortenCentipede()

shortenCentipede zorgt ervoor dat als deze methode wordt aangeroepen er een nieuwe array wordt aangemaakt met alle posities van de oude array behalve de laatste ofwel het einde van de staart. Deze methode werkt zo:

Bijvoorbeeld, de speler raakt met het hoofd van zijn Centipede een kameleon aan.

shortenCentipede wordt aangeroepen en maakt een lokale array met als waardes de lengte van het oude array min 1 en de waarde 2 voor de x en y positie.

shortenCentipede kopieert alle waarden behalve de laatste van de oude array met posities naar een nieuwe array.

De methode returnt de array met nieuwe waardes.

Dit nieuwe array wordt vervolgens gebruikt als array voor alle waarden.

## bananaHit()

bananaHit kijkt of het hoofd van de Centipede op dezelfde positie is als de banaan. Als dit het geval is wordt de banaan van het speelveld verwijdert en de score verhoogd met 10. bananaHit() werkt als volgt:

Bijvoorbeeld, het hoofd van de Centipede bevindt zich op dezelfde plek als een banaan.

bananaHit() maakt twee lokale variabelen die de offset van de linkerkant bevatten plus de positie van de banaan keer squareSize aangezien de locatie van de bananen zijn opgeslagen in grid-posities en niet scherm coördinaten. Deze moeten dus nog worden omgezet en dat wordt hier gedaan.

Nadat dat is gedaan wordt er gekeken of de x positie van het hoofd hetzelfde is als de x positie van de banaan. Hetzelfde wordt gedaan voor de y positie.

Als dat zo is wordt de banaan uit het speelveld gehaald en returnt de methode true.

De score wordt verhoogd met 10.

# Architectuurplaatje

Class Diagram Centipedes,

Job Zalmé

In dit architectuurplaatje zijn alle tabbladen weergegeven uit mijn Centipedes project. Een pijltje betekent “maakt gebruik van”. Zo is te zien dat niets gebruik maakt van de main method, CentipedeMain, maar deze maakt wel gebruik van de andere tabbladen om bijvoorbeeld informatie weer te geven.

# Reflectie op codekwaliteit

In dit hoofdstuk zal ik reflecteren op de geschreven code.

## Goed gegaan

Allereerst wil ik graag benoemen waar ik volgens mijzelf aan heb voldaan, daarna zal ik wat verder ingaan op wat de volgende keer beter kan en wat er juist erg goed ging.

1. De namen van de methodes zijn geschreven met lowerCamelCase.
2. De tabbladen zijn geschreven in UpperCamelCase.
3. Een methode heeft één taak.
4. Benodigde informatie voor methodes zijn meegegeven in de parameters.
5. Methodes geven informatie terug via returnwaarde zo dien nodig.
6. Namen van methodes bevatten werkwoorden.
7. Elk codeblok dat een taak uitvoert is in een methode geschreven.

Naast deze punten vond ik mijn vermogen tot op oplossingen komen erg goed. Iets wat mij enorm hierbij hielp was eerst de oplossing uitschrijven in pseudo-code en vervolgens dat omzetten naar normale code. Dit zorgde ervoor dat ik het probleem blokje bij blokje kon aanpakken. Ook het formuleren van de code zelf vond ik erg goed gaan, naar mijn mening is mijn code over het algemeen duidelijk. Daarbij hebben mijn notities ook goed geholpen. Er zijn echter wel een paar dingen die beter kunnen, deze onthoud ik dan ook voor de volgende keer.

## Kan beter

Zo kwam ik er al snel achter dat mijn globale variabelen niet altijd ideaal verdeeld waren over de tabbladen. Dit kwam, omdat soms een variabele gebruikt moest worden in een ander tabblad maar daar nog niet geïnitialiseerd was. Daarom moest ik de variabele op een andere positie deponeren. Zo werden sommige variabelen op plekken neergezet waar ik het niet helemaal mee eens ben, maar ik weet zeker dat zodra we met classes gaan werken dit al een stuk minder gaat gebeuren.

Ook moet ik goed letten op de taal die ik gebruik door mijn code heen. Ik heb namelijk op het einde de namen van variabelen en tabbladen moeten aanpassen, omdat ik Nederlands en Engels door elkaar heen gebruikte. Dit zorgde voor onnodig tijdsverlies. Gelukkig had ik nog een klein beetje tijd over en heb ik dit kunnen fiksen.

Ten derde moet ik naar mijn mening goed letten op mijn spacing in de code. Processing heeft hier een handig trucje voor genaamd ctrl + t, maar ik wil mijzelf hier niet afhankelijk van maken. Ik vind het belangrijk dat ik zelf de juiste structuur van de code leer zodat deze goed leesbaar en duidelijk is voor anderen. Op het einde van het project begon ik zelf ook al veel meer na te denken over wanneer ik tab moest gebruiken en wanneer niet. De leesbaarheid werd hierdoor al een stuk beter!

# Testrapportage

|  |  |  |  |
| --- | --- | --- | --- |
| Testcase | Actie | Verwacht resultaat | Geslaagd |
| FR01 | M  Een 1 speler spel, waarbij de speler zoveel mogelijk punten moet proberen te  scoren voordat alle segmenten van de duizendpoot zijn verdwenen. | De speler selecteert 1 speler in het beginscherm. | Er wordt een spel gegenereerd dat door één speler speelbaar is. | JA |
| FR02 | M  Het spel wordt met het toetsenbord bestuurd. Bij elke beweging van het hoofd van  de duizendpoot neemt elk segment de oude positie aan van het segment dichter  bij het hoofd. De segmenten volgen dus het hoofd. | De speler klikt op een pijltje op het toetsenbord. | De Centipede beweegt in de richting van de toets waarop is gedrukt. | JA |
| FR03 | M  Het speelveld bestaat uit 16 bij 8 vakjes. Alle velden zijn bij aanvang ‘verborgen’,  behalve die van de duizendpoot. | Er wordt een speelveld van 16 bij 8 gemaakt waarop geen dingen te zien zijn behalve de Centipede. | Er wordt een spel vak gegenereerd van 16 bij 8 hokjes waarop alleen de Centipede te zien is. | JA |
| FR04 | M  Er zijn binnen het speelveld op willekeurige posities 20 bananen verborgen. | Er worden 20 bananen op random posities geplaatst. Deze zijn verborgen voor de speler. | Er zijn 20 bananen geplaatst op willekeurige posities. Deze bananen zijn niet te zien door de speler. | JA |
| FR05 | M  Op 10% van de velden waar geen fruit ligt, zijn kameleons verborgen. Deze zijn  willekeurig verdeeld. | Er wordt berekent hoeveel kameleons er moeten komen en deze worden willekeurig verdeeld op posities waar geen bananen liggen. | Er worden 2 kameleons gemaakt bij 20 bananen en deze worden geplaatst op willekeurige posities waar geen bananen of andere kameleons liggen. | JA |
| FR06 | M  De duizendpoot verliest het laatste segment van zijn staart bij het aanraken van  een kameleon. | Als de duizendpoot een kameleon aanraakt wordt het laatste segment van zijn staart verwijderd. | Het laatste segment van de duizendpoot wordt verwijderd als hij een kameleon aanraakt. | JA |
| FR07 | M  Bananen leveren 10 punten op | Als een banaan wordt geraakt komen er 10 punten bij de score. | De banaan is geraakt en levert 10 punten op voor de speler. | JA |
| FR08 | M  Tijdens het spel staan de scores continu in beeld. | Zodra het speelscherm wordt gestart staat de score en beeld en verdwijnt deze niet. | De score blijft op het speelveld continu in beeld staan en wordt geüpdatet wanneer nodig. | JA |
| FR10 | S  Een startscherm waarin het aantal bananen en het speelveld in te stellen is. Het  speelveld bestaat uit een raster van minimaal 16 vakjes breed bij 8 vakjes hoog  en maximaal 32 vakjes breed bij 16 vakjes hoog. | Als de speler een andere grootte speelveld selecteert wordt deze ingeladen in plaats van de standaard 16 bij 8 grootte. | Het speelveld veranderd aan de hand van de ingegeven waarde door de speler. | JA |
| FR11 | S  Er is een eindscherm met de uitslag en de mogelijkheid om opnieuw te beginnen. | Het spel is voorbij en de eindscore wordt weergegeven aan de speler. Ook is er een restart button die je naar het startscherm brengt en de score op 0 zet. | Zodra het spel voorbij is wordt de score laten zien en kan ik terug naar het startscherm. | JA |

# Bijlagen

Bijlagen zijn bijgevoegd bij dit bestand. De bijlagen bestaan uit:

CD Centipedes.png

Class Diagram Centipedes.asta

Centipedes.zip