|  |
| --- |
| SPAD OPLEVERDOCUMENT\_  Versie 1 – |

Brian Rijkers (student) – HBO-ICT 1C 2130167 – Herman Telman (docent)

1 november 2024

INHOUDSOPGAVE

[1 Inleiding 3](#_Toc181367577)

[2 Algoritmen 4](#_Toc181367578)

[2.1 Algoritme 1- Bewegingsalgoritme 4](#_Toc181367579)

[2.2 Algoritme 2- Aanrakingsalgoritme 4](#_Toc181367580)

[2.3 Algoritme 3- Willekeurige plaatser algoritme 4](#_Toc181367581)

[3 Software architectuur 5](#_Toc181367582)

[3.1 Globale variablen 5](#_Toc181367583)

[3.2 ASetup 5](#_Toc181367584)

[3.3 BDraw 5](#_Toc181367585)

[3.4 CControls 6](#_Toc181367586)

[3.5 DObjects 6](#_Toc181367587)

[3.6 EWin 6](#_Toc181367588)

[4 Reflectie op codekwaliteit 7](#_Toc181367589)

[5 Testrapportage 8](#_Toc181367590)

[5.1 FR01- 8](#_Toc181367591)

[5.2 FR02- 8](#_Toc181367592)

[5.3 FR03- 8](#_Toc181367593)

[5.4 FR04- 8](#_Toc181367594)

[5.5 FR05- 9](#_Toc181367595)

[5.6 FR06- 9](#_Toc181367596)

[5.7 FR07- 9](#_Toc181367597)

[5.8 FR08- 9](#_Toc181367598)

[5.9 FR015- 9](#_Toc181367599)

[6 Conclusie 10](#_Toc181367600)

[7 Bronnenlijst 11](#_Toc181367601)

# Inleiding

De laatste paar weken heb ik van docent Herman Telman de opdracht gekregen het spel Space Raider te maken in processing Java. In dit verslag worden de 3 belangrijkste algoritmes, de software architectuur, een reflectie op de codekwaliteit en de testrapportage besproken van de code van het spel Space Raiders.

# Algoritmen

## Algoritme 1- Bewegingsalgoritme

Het bewegingsalgoritme heb ik gecodeerd door een boolean functie te maken voor elke richting die speler op kan (boven, onder, links en rechts). Elke keer wanneer de speler een toets indrukt zal de bijbehorende boolean true worden en alle andere 3 false, zodat de speler niet schuin beweegt.

Speler drukt bewegingstoets in -> spel zorgt ervoor dat speler in die richting beweegt (en stopt speler van andere beweging maken totdat de beweging klaar is)

## Algoritme 2- Aanrakingsalgoritme

Ik heb een aanrakingsalgoritme boolean functie gecodeerd zodat elke keer wanneer de speler een object aanraakt het spel weet dat de speler dit object aanraakt. Bij muren wordt de boolean moving naar false gezet als de speler dit object aanraakt in de richting waar de speler wil bewegen, zodat de speler niet door muren beweegt.

Speler raakt object aan -> spel ziet dat speler object aanraakt -> spel activeert bijbehorende activiteit van het object

## Algoritme 3- Willekeurige plaatser algoritme

Ik heb 2 verschillende willekeurige plaatsingssystemen gemaakt, eentje voor slechte overvallers en eentje voor schatten. Het slecht overvaller algoritme zorgt ervoor dat een 50% kans is dat er slechte overvaller komt op een plaats die uitgekozen is door mij (zodat er altijd een manier is om het spel te voltooien zonder een slechte overvaller aan te raken). Bij het schatten algoritme heb ik het zo gemaakt dat er 20% kans is dat er een 1 punt schat komt en een 8% kans is dat er 5 punt schat komt op elke plaats waar een lucht blokje normaal staat.

Speler laad spel -> spel zorgt ervoor dat er een 50% kans is dat een slechte overvaller kan kom elke plek waar een slechte overvaller kan komen -> spel zorgt ervoor dat er een 20% kans is dat er een 1 punt schat komt en een 8% kans is dat er 5 punt schat komt op elke plaats waar een lucht blokje normaal staat.

# Software architectuur

## Globale variablen

int cols: Aantal kolommen

int rows: Aantal rijen

int cellSize: Grootte van elk blokje

int S: Slechte overvaller random replacer

boolean hitBadRaiderOnce: Controleer of score al gehalveerd is bij botsing

boolean[][] collectedTreasures: Houdt bij welke schatten zijn verzameld

int[][] grid: Positie van elementen op grid, 90 graden gedraaid

int gamePlayerSize: Grootte speler

int xGamePlayerPos: Begin x-positie speler

int yGamePlayerPos: egin y-positie speler

int gamePlayerSpeed: Snelheid van speler

boolean movingUp: Speler beweegt niet bij begin

boolean movingDown: Speler beweegt niet bij begin

boolean movingLeft: Speler beweegt niet bij begin

boolean movingRight: Speler beweegt niet bij begin

boolean moving: Houdt bij of de speler beweegt

int score1: Score van de speler

boolean checkCollision(int x, int y, int objectType): Checkt als speller iets raakt

int left: Linkerkant van de speler

int right: Rechterkant van de speler

int top: Bovenkant van de speler

int bottom: Onderkant van de speler

## ASetup

void replaceRandomSWithBadRaider(): Willekeurige slechte overvaller plaatsing

int badRaiderCount: Aantal slecht overvallers

int i: Cols

int j: Rows

void replaceRandom0WithTreasure(): Willekeurige schat plaatsing

int treasureCount: Aantal schatten

void setup(): Size

## BDraw

void draw(): Teken 60 fps

int i: Cols

int j: Rows

int playerCol: Controleer botsing met andere objecten col

int playerRow: Controleer botsing met andere objecten row

int score: Score van de speler

int score1T(int score1R): Score van de speler

## CControls

void keyPressed(): Controls

## DObjects

void drawGamePlayer(): Tekent speler

void drawWall(int i, int j): Tekent muur

void drawTreasure(int i, int j): Tekent schat1

void drawBadRaider(int i, int j): Tekent slechte overvaller

void drawPortal(int i, int j): Tekent portaal

void drawTreasure5(int i, int j): Tekent schat5

void drawScoreboard(int score): Tekent scorebord

## EWin

void showWinningScreen(): Win scherm

# Reflectie op codekwaliteit

De code kwaliteit is redelijk, maar niet perfect.

Voor elk codeblok heb ik een omkaderde taak uitvoert heb ik een methode gemaakt.

Elke methode doet maar 1 ding.

Informatie die een methode nodig heeft heb ik meegegeven via parameters.

Een methode geeft 1 keer informatie terug aan het hoofdprogramma via de returnwaarde.

De methodenamen heb ik altijd in lowerCamelCase getypt.

De naam van elke methode bevat een werkwoord.

Alle accolades heb ik goed toegepast.

Methodes worden wel in bij elkaar gebundeld in een module als ze bij elkaar horen, maar is niet altijd perfect toegepast wat alles een beetje rommelig kan maken.

Declaraties van variabelen heb ik wel gegroepeerd.

De modulenamen heb ik altijd in UpperCamelCase getypt.

Returnwaardes heb ik niet veel gebruikt en heb ik eigenlijk alleen toegepast in 1 situatie waar die eigenlijk niet echt nodig is. Dit heb ik gedaan aangezien ik er geen echt goede toepassing voor kon vinden, maar het moest per se volgens het beoordelingsformulier in de code.

Ik heb heel veel “comments” toegevoegd in mijn code, vooral zodat het in een keer makkelijk te begrijpen is, zelfs voor een beginner.

# Testrapportage

## FR01-

Een 1-player game, die voldoet aan de beschrijving in dit document, zonder de in dit hoofdstuk beschreven Should en Could haves.

|  |  |  |  |
| --- | --- | --- | --- |
| **Testcase** | **Actie** | **Verwachte resultaat** | **Geslaagd** |
| 1 | Alle toegepaste must haves testen | Spel speelt zoals verwacht. | Ja |

## FR02-

De speler kan de Raider besturen met het toetsenbord.

|  |  |  |  |
| --- | --- | --- | --- |
| **Testcase** | **Actie** | **Verwachte resultaat** | **Geslaagd** |
| 1 | Druk up key in wanneer er geen muur boven de speler staat | Speler beweegt omhoog | Ja |
| 2 | Druk down key in wanneer er geen muur onder de speler staat | Speler beweegt omlaag | Ja |
| 3 | Druk right key in wanneer er geen muur rechts van de speler staat | Speler beweegt naar rechts | Ja |
| 4 | Druk left key in wanneer er geen muur links van de speler staat | Speler beweegt naar links | Ja |
| 5 | Druk een beweging key in terwijl speler in het midden van een andere beweging zit | Speler blijft bewegen in de richting waar de speler eerst al op ging | Ja |
| 6 | Beweeg tegen een muur aan | Speler stop wanneer die de muur aanraakt | Ja |

## FR03-

Er is 1 type schat, namelijk het muntje. Deze levert 1 puntop.

|  |  |  |  |
| --- | --- | --- | --- |
| **Testcase** | **Actie** | **Verwachte resultaat** | **Geslaagd** |
| 1 | Beweeg met speler over 1 munt | Speler krijgt 1 punt | Ja |
| 2 | Beweeg met speler over 3 munten achter elkaar | Speler krijgt 3 punten | Ja |

## FR04-

De elementen Bad Raiders en schatten zijn willekeurig verdeeld in een onzichtbaar raster en blijven op die locatie.

|  |  |  |  |
| --- | --- | --- | --- |
| **Testcase** | **Actie** | **Verwachte resultaat** | **Geslaagd** |
| 1 | Druk 3 op start | Er zijn 3 verschillende scenario’s waar munten en slechte overvallers op andere plekken staan | Ja |

## FR05-

Er kunnen geen verschillende elementen op dezelfde plek staan

|  |  |  |  |
| --- | --- | --- | --- |
| **Testcase** | **Actie** | **Verwachte resultaat** | **Geslaagd** |
| 1 | Beweeg over munt heen | De munt zie je niet meer als je weg beweegt | Ja |
| 2 | Beweeg over slechte overvaller heen | De slechte overvaller zie je niet meer als je weg beweegt | Ja |

## FR06-

De score wordt bijgehouden en staat continu in beeld.

|  |  |  |  |
| --- | --- | --- | --- |
| **Testcase** | **Actie** | **Verwachte resultaat** | **Geslaagd** |
| 1 | Beweeg over munt heen | De score veranderd, maar scoreboard blijft staan | Ja |
| 2 | Beweeg over slechte overvaller heen | De score veranderd, maar scoreboard blijft staan | Ja |
| 3 | Beweeg over slechte portaal heen | Scoreboard blijft staan | Ja |

## FR07-

Aanraking met een Bad Raider deelt het aantal verdiende punten door 2.

|  |  |  |  |
| --- | --- | --- | --- |
| **Testcase** | **Actie** | **Verwachte resultaat** | **Geslaagd** |
| 1 | Beweeg over 1 munt heen | De score veranderd naar 0 (1 gedeeld door 2 is 0,5, maar er zijn geen halve punten in het spel) | Ja |
| 2 | Beweeg over 2 munten | De score veranderd naar 1 | Ja |

## FR08-

Bij aanraking van de portal is het spel afgelopen.

|  |  |  |  |
| --- | --- | --- | --- |
| **Testcase** | **Actie** | **Verwachte resultaat** | **Geslaagd** |
| 1 | Beweeg over portaal heen | Eindscherm komt in beeld en het spelveld is niet meer te zien | Ja |

## FR015-

Er zijn meerdere typen schatten die elk andere punten opleveren

|  |  |  |  |
| --- | --- | --- | --- |
| **Testcase** | **Actie** | **Verwachte resultaat** | **Geslaagd** |
| 1 | Beweeg over schat1 heen | 1 punt wordt toegevoegd aan de score | Ja |
| 2 | Beweeg over schat5 heen | 5 punten worden toegevoegd aan de score | Ja |

# Conclusie

Het spel Space Raiders gemaakt door mij heeft 3 belangrijke algoritmes, een bewegingsalgoritme, een aanrakingsalgoritme en een willekeurige plaatsingsalgoritme. Deze zijn fundamenteel voor het spel en waren ook het lastigst om te implementeren. In de code van het spel zitten 5 modules die elkaar aanroepen. De codekwaliteit is redelijk, maar niet echt heel goed. Alle test scenario’s zijn uitstekend getest. Alle must haves zijn voltooid en een should have ook. Dit zorgt voor al en al, een werkend spel.

# Bronnenlijst

Er zijn geen externe bronnen gebruikt.

