EscApe – Final release

##### Iteratie 3 – revisie A

19-06-2015 Chris Ros, Aaik Oosters

Inhoud

[Opleverdata 4](#_Toc422512833)

[Introductie tot spel 4](#_Toc422512834)

[Verandering t.o.v iteratie 2 5](#_Toc422512835)

[Highscore Systeem 5](#_Toc422512836)

[Extra item: Cape 5](#_Toc422512837)

[Eisen van de opdrachtgever 5](#_Toc422512838)

[Spel veld 5](#_Toc422512839)

[Valsspeler 5](#_Toc422512840)

[Bazooka 5](#_Toc422512841)

[Helper 5](#_Toc422512842)

[Spellogica 5](#_Toc422512843)

[Toevoegingen door ontwikkelaar 6](#_Toc422512844)

[Cheatcodes 6](#_Toc422512845)

[Cape 6](#_Toc422512846)

[Highscores 6](#_Toc422512847)

[Eisen bedrijf ForFun 6](#_Toc422512848)

[Minimale eisen: 6](#_Toc422512849)

[Eerste iteratie 6](#_Toc422512850)

[Tweede iteratie 6](#_Toc422512851)

[Derde iteratie 6](#_Toc422512852)

[Spelregels en richtlijnen 7](#_Toc422512853)

[Definities van bestanden 8](#_Toc422512854)

[Classes 8](#_Toc422512855)

[Level data 8](#_Toc422512856)

[Images 8](#_Toc422512857)

[Testbevindingenrapport 9](#_Toc422512858)

[Fysiek testen op het bewegen 9](#_Toc422512859)

[Testen: Bewegen van de Held 10](#_Toc422512860)

[Logische testgevallen aan de hand van test maat 2 11](#_Toc422512861)

[Padencombinatie test 11](#_Toc422512862)

[Fysieke testgevallen 11](#_Toc422512863)

[Gebruikt grid tijdens tests 11](#_Toc422512864)

[Testen: Doorvoeren van beweging 12](#_Toc422512865)

[Logische testgevallen aan de hand van test maat 2 13](#_Toc422512866)

[Padencombinatie test 13](#_Toc422512867)

[Fysieke testgevallen 13](#_Toc422512868)

[Gebruikt grid tijdens tests 13](#_Toc422512869)

[Testen: Fire van bullet 14](#_Toc422512870)

[Logische testgevallen aan de hand van test maat 2 15](#_Toc422512871)

[Padencombinatie test 15](#_Toc422512872)

[Fysieke testgevallen 15](#_Toc422512873)

[Gebruikt grid tijdens tests 15](#_Toc422512874)

[Samenvattingen testen 16](#_Toc422512875)

[Testen: Bewegen van de Held 16](#_Toc422512876)

[Testen: Doorvoeren van beweging 16](#_Toc422512877)

[Testen: Fire van bullet 16](#_Toc422512878)

[Use cases & Beschrijvingen 17](#_Toc422512879)

[Functionele eisen 17](#_Toc422512880)

[Use cases 18](#_Toc422512881)

[Analysis diagram 21](#_Toc422512882)

[Design Diagram 22](#_Toc422512883)

[Sequence Diagram: Bewegen van Held 23](#_Toc422512884)

[Sequence Diagram: Fire of bullet 24](#_Toc422512885)

[Applicatie preview 25](#_Toc422512886)

[Afbeeldingen Legenda 27](#_Toc422512887)

[Instructie level configuratie 28](#_Toc422512888)

# Opleverdata

- Iteratie 1 week 5 week van: 25 mei

- Iteratie 2 week 7 week van: 8 juni

- Iteratie 3 week 8 (vrijdag\*) week van: 15 juni. Opleveren op de 18e

\*uitstel vanwege hulp introductie middag (meet & match)

# Introductie tot spel

In het spel in kwestie, EscApe is het doel de held zo snel en zo efficient mogelijk door het doolhof te manouveren tot het einde, gemarkeerd met een ‘vriend’ , is bereikt. Het spel bevat enkele ‘items’ welke door de speler te gebruiken zijn. Dit gebeurt wanneer de speler zich in het zelfde vakje als het voorwerp bevindt. Er zijn 4 items: de helper, de cheater, de cape en de bazooka. De helper tekent een pad vanaf de held naar het einde van het doolhof, de cheater veranderd de score van de held. Na het oppakken van de cape is de speler in staat op en van muren te springen. En de bazooka, wanneer afgevuurd, zal de eerste verwoestbare muur in de richting waar de speler voor het laatst bewogen is vernietigen. Het huidige spel bevat 3 levels, als je alle rondes hebt voltooid krijg je je totale score te zien, en de mogelijkheid om het spel opnieuw te starten.

# Verandering t.o.v iteratie 2

In de final reale versie van het spel zijn enkele extra functionaliteiten toegevoegd om de fun-factor van het spel te verhogen. Hieronder een beknopte opsomming van de toevoegingen.

### Highscore Systeem

Aan het startscherm is de mogelijkheid toegevoegd om te kijken naar de 10 beste spelers.   
aan het einde van het spel is het mogelijk om d.m.v het invoeren van je naam jouw score te bewaren . Dit is bereikt door gebruik te maken van een centrale database in MySQL.

### Extra item: Cape

De cape is net als de overige items op te pakken, en zal na het oppakken er voor zorgen dat de speler in staat is niet alleen over paden, maar ook over de niet buitenste muren te wandelen. De cape verlaagt je score niet.

# Eisen van de opdrachtgever

Om aan de eisen van de opdrachtgever te voldoen moet het spel de volgende onderdelen bevatten, deze moeten aan de daaronder genoemde specificaties voldoen

### Spel veld

- basisspel is een rechthoek

- bevat vriend (is eindpositie)

- bevat speler (heeft startpositie)

- bewegen door pijltjestoetsen

- puntentelling op basis van stappen/tijd

### Valsspeler

- verschillende waarde

- timer / stappen terug

- geraakt? Van het veld

### Bazooka

- oppakken

- afschieten

- muur weg schieten

- bazooka moet verdwijnen als je deze hebt afgeschoten

### Helper

- zichtbaar snelste route

-Hoeft niet te wijzigen als je gat maakt in muur na vragen op helper komt

### Spellogica

- Minimaal 3 doolhoven

- Opnieuw kunnen starten

- Willekeurig verschijnen van levels (geen vaste volgorde van levels)

# Toevoegingen door ontwikkelaar

### Cheatcodes

- door bepaalde woorden uit te typen is het mogelijk power-ups te verkrijgen

### Cape

- Oppakken, springen over muren door er tegen aan te lopen

### Highscores

**-**10 beste scores worden weergeven in scherm  
 -na eind elk spel mogelijkheid score op te slaan

# Eisen bedrijf ForFun

## Minimale eisen:

- minimale koppeling

- maximale cohesie

- scheiding van Model en View

- Goed versie beheer bijhouden

- Module testen

- Iteratieve systeemontwikkeling

- Bij iteratie: demo geven, eventueel eisen bij stellen

### Eerste iteratie

- proof of concept opstellen en uitvoeren

- Laten testen of het ontwerp geschikt is voor project

- belangrijk eerst Requirements opstellen

- Analysis klassendiagram en desingklassendiagram en squence diagram van proof of concept

### Tweede iteratie

- bassispel ontwerpen en opleveren

- design klassengram uitbreiden

- indoen nodig: extra squence diagrammen opleveren

- minimale één module in JUnit getest . Ook test rapport opleveren

- afsluiten met een demo

### Derde iteratie

- Extra uitbreidingen, functionaliteiten in overleg

- functionaliteiten kunnen leiden tot uitbreiding van het design klassendiagram, use cases en sequence diagrammen

## Spelregels en richtlijnen

Het spel start als de knop ‘start’ ingedrukt wordt

De speler kan met de pijltjestoetsen de held naar zijn vriend(eindpunt) begeleiden

Een level start met een bepaald aantal punten, en geen tijdslimiet

Bij elke stap worden er 10 punten van de punten afgetrokken (tot een minimum van 1/10e van de initiële score)

bij elke keer dat de solver wordt aangeroepen (d.m.v. cheatcode ‘help’ of helper in het speelveld) wordt de helft van de initiële punten van de score afgetrokken

bij het bereiken van het eindpunt wordt indien beschikbaar het volgende level geladen, na het spelen van alle levels krijgt de speler een score overzicht.

Na het oppakken van een bazooka kan een speler één keer een muur veranderen in een pad.  
  
Na het oppakken van de cheater wordt de score van de speler verandert naar een willekeurig getal tussen de 100 en 2000.

# Definities van bestanden

## Classes

|  |  |
| --- | --- |
| **Naam** | **Functie** |
| Startscreen | Main class van de applicatie, start de applicatie, tevens startscherm |
| EndScreen | Na voltooien spel wordt dit scherm weergeven |
| Spel | Belangrijke class die de hele runtime van het spel bestaat, start volgende levels |
| Maze | Een representatie van het doolhof, met alle benodigde variabele |
| MazeSolved | Overerving van maze, heeft een snelste route en een hoeveelheid benodigde stappen |
| Frame | De GUI bevat een panels waarin het doolhof, het scoreboard, en de knoppen gedrawed worden |
| Getkey | KeyListener class leest cheatcodes en parsed KEYVALUE naar move methode Held |
| ScoreBoard | Class verantwoordelijk voor het updaten en berekenen van de score |
| Solver | Berekent de snelste route van start tot eind, maakt een mazeSolved object |
| LevelLoader | Class verantwoordelijk voor het omzetten van level bestanden naar objecten |
| Held | Het poppetje wat de speler bediend |
| Block | Representatie van een ‘vakje’ in het maze |
| KEYVALUE | ENUM die de mogelijk bruikbare toetsen representeert |
| Item | Parent van helper, cheater en bazooka, representeert een op pakbaar item |
| Cheater | Veranderd de score van de held naar een willekeurig getal |
| Helper | Tekent een oplossing in het doolhof |
| Bazooka | Bij oppakken kan de speler een muur vernietigen |
| Cape | Na het oppakken is het voor de speler op en over muren heen te springen |

## Level data

|  |  |
| --- | --- |
| **Naam** | **Functie** |
| Level[0-2] | Textbestand met XML opmaak voor het bewaren van levels |

## Images

(voor grafische toelichting zie pagina 17)

|  |  |
| --- | --- |
| **Naam** | **Functie** |
| held | Het personage wat bediend wordt door de speler |
| pad | Het pad wat bewandeld kan worden |
| finish | Het einde wat bereikt moet worden in het spel |
| start | Het vakje waar het spel begint |
| wall | Onbewandelbare muren in het spel |
| solvedpad | Hetzelfde als pad, maar dan gemarkeerd indien onderdeel van de snelste route |
| solver | Weergeven dat een blok een solver bevat |
| Helper | Weergeven dat een blok een helper bevat |
| Cheater | Weergeven dat een blok een cheater bevat |
| Cape | Weergeven dat een blok een cape bevat |

# Testbevindingenrapport

## Fysiek testen op het bewegen

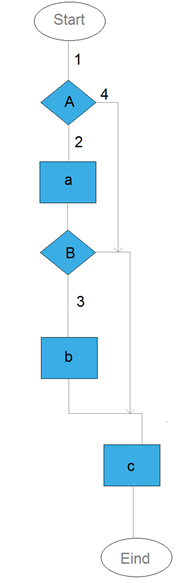
Onze verwachtingen bij het bewegen naar boven, beneden, links en rechts waren dat bij de toets omhoog de held een plek omhoog beweegt. Bij toets naar beneden de held een plek naar beneden beweegt. Wanneer de toets naar links of naar rechts word ingedrukt de held een plek naar links of rechts beweegt.

We hebben dit getest aan de hand van de code en door middel van het te testen terwijl het programma liep. Onze verwachtingen kwamen uit. We hadden ook verwacht dat bij een willekeurige andere toets er niks zou gebeuren en ook dit kwam uit het resultaat.

Wanneer de held ergens tegen een muur aan liep ging de held niet door de muur heen.

# Testen: Bewegen van de Held

In de loop van het project is door de opdrachtgever gevraagd of er een test kon worden opgesteld van de code verantwoordelijk voor het bewegen van de held. De methode verantwoordelijk voor het bewegen van de speler bevindt zich in de klasse Held en is hieronder weergeven (Fig. 1)

Naast de code staat een abstracte weergave van de methode (Fig. 2) met daarin alle verschillende paden genummerd , deze zullen later worden gebruikt om een test te bedenken die aan de dekkingseisen van de gebruikte test maat voldoet.

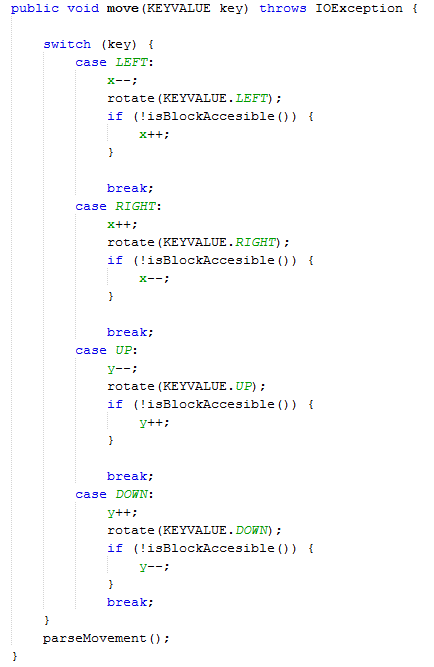


Fig. 1 Fig. 2

**a** = bewegen ,roteren ;  
**b** = beweging terug zetten;  
**c** = beweging doorvoeren ;  
**A** = is het een key : true->2; false-> 4;  
**B** = is het is het block begaanbaar: true->4; false-> 3;

Voor alle gemaakte testen is een doolhof (geinitialiseerd met een grid) met een start punt nodig. Wij hebben ervoor gekozen zo’n klein mogelijk grid te gebruiken om de zo min mogelijk omgevingsvariabelen te creëren.

Doordat de move methode gebruik maakt van een ENUM (een variabele wat maar enkele waardes kan hebben) hebben we hier niet te maken met grenswaardes. Na alle tests met behulp van JUnit te hebben getest zijn wij tot de conclusie gekomen dat de verwachte uitvoer is behaald en daarmee alle opgestelde tests zijn geslaagd. Dit geld voor alle gemaakte testen.

|  |  |  |  |
| --- | --- | --- | --- |
| **Beslissingspunten** | **IN** | **OUT** | **UITKOMST** |
| A | 1 | 2,4 | 1-2; 1-4; |
| B | 2 | 3,4 | 2-3; 2-4; |

### Logische testgevallen aan de hand van test maat 2

|  |  |
| --- | --- |
| **Testgeval** | **Paden** |
| 1 | 1-2-3 |
| 2 | 1-4 |
| 3 | 1-2-4 |

### Padencombinatie test

|  |  |  |  |
| --- | --- | --- | --- |
| **Pad combinatie** | **TG 1** | **TG 2** | **TG 3** |
| 1-2 | X |  | X |
| 1-4 |  | X |  |
| 2-3 | X |  |  |
| 2-4 |  |  | X |
| **MOGELIJK?[[1]](#footnote-1)** | JA | NEE | JA |

### Fysieke testgevallen[[2]](#footnote-2)

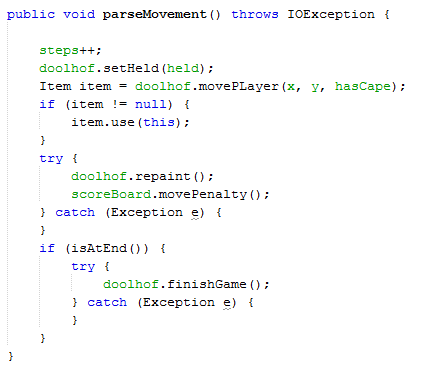
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test** | **Invoer Key** | **Invoer Toegankelijk** | **Verwachte uitvoer** | **Startpositie testen** | **Result** |
| 1 | LEFT | JA | (2, 1) | (3, 1) | (2, 1) |
| 2 | DOWN | JA | (3, 3) | (3, 2) | (3, 3) |
| 2 | SPACEBAR | - | (3, 2) | (3, 2) | (3, 2) |
| 3 | LEFT | NEE | (2, 2) | (2, 2) | (2, 2) |

### **Gebruikt grid tijdens tests**

**0 0 0 0 0   
0 1 1 1 0   
0 1 1 1 0   
0 1 1 1 0   
0 0 0 0 0**

# Testen: Doorvoeren van beweging

In het vorige test verslag is ter sprake gekomen dat na elke beweging de beweging door gevoerd dient te worden, onder andere om het scherm de beweging te laten ‘tekenen’ dit werd door ons als essentieel onderdeel van het ‘bewegen van de speler’ gezien en daarvoor hebben wij dit ook meegenomen in de uitgevoerde tests. Ook deze methode bevindt zich in de klasse held, en ook hier hebben we de code (Fig. 3) en het flow diagram (Fig. 4) meegenomen in het opstellen van de test.



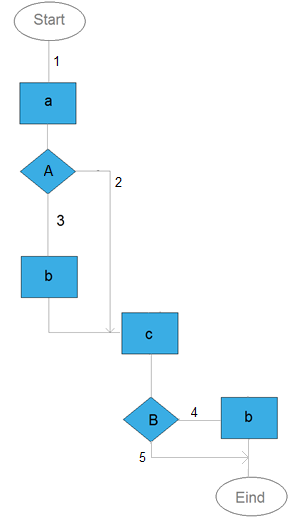


Fig. 3

Fig. 4

|  |  |  |  |
| --- | --- | --- | --- |
| **Beslissingspunten** | **IN** | **OUT** | **UITKOMST** |
| A | 1 | 2,3 | 1-2; 1-3; |
| B | 2,3 | 4,5 | 2-4; 2-5; 3-4; 3-5; |

### Logische testgevallen aan de hand van test maat 2

|  |  |
| --- | --- |
| **Testgeval** | **Paden** |
| 1 | 1-2-4 |
| 2 | 1-3-5 |
| 3 | 1-2-5 |
| 4 | 1-3-4 |

### Padencombinatie test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Pad combinatie** | **TG 1** | **TG 2** | **TG 3** | **TG 4** |
| 1-2 | X |  | X |  |
| 1-3 |  | X |  | X |
| 2-4 | X |  |  |  |
| 2-5 |  |  | X |  |
| 3-4 |  |  |  | X |
| 3-5 |  | X |  |  |
| **MOGELIJK?** | JA | JA | JA | JA |

### Fysieke testgevallen

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Invoer item** | **Invoer einde** | **Verwachte uitvoer** |
| 1 | NEE | JA | Einde zonder item |
| 2 | JA | NEE | Zonder einde met item |
| 3 | NEE | NEE | Zonder einde zonder item |
| 4 | JA | JA | Einde met item |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test** | **Invoer item** | **Invoer einde** | **Held locatie** | **Verwachte uitvoer** | **Resultaat** |
| 1 | false | (2, 2) | (2,2) | End level | End level |
| 2 | true | (3, 2) | (2,2) | Cape | Cape |
| 3 | false | (3, 2) | (2,2) | No cape, no End | No cape, no End |
| 4 | true | (2, 2) | (2,2) | Cape, End level | Cape, End level |

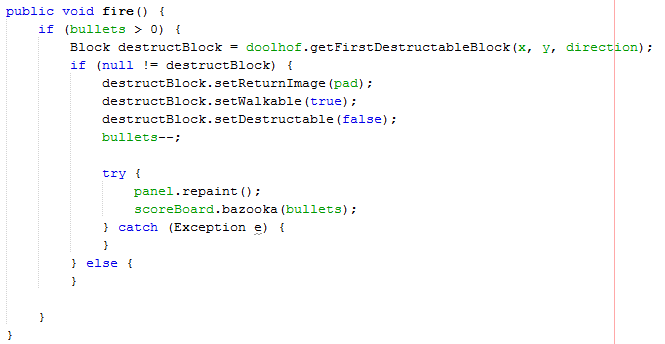
### **Gebruikt grid tijdens tests**

**0 0 0 0 0   
0 1 1 1 0   
0 1 1 1 0   
0 1 1 1 0   
0 0 0 0 0**

# Testen: Fire van bullet

Deze test is ontstaan uit het feit dat we zeker willen weten dat het gebruik van een bazooka goed werkt. Dit is een belangrijk onderdeel van het spel en moet dan ook goed werken.

Op het moment dat fire word aangeroepen door de spatiebalk word er gekeken of er genoeg kogels zijn. De uitkomst hiervan zijn de paden 2 of 3 in Fig. 7. Wanneer er genoeg kogels zijn word er via ‘pad’ drie verder gekeken of het blok waarop geschoten is verwoest baar is. Zo ja dan worden de laatste handelingen doorgevoerd. Denk hierbij aan het verminderen van de kogels en de muur verwijderen. De code van de methode fire is terug te vinden in Fig. 6 en het gebruikte flowdiagram is te vinden in Fig. 7.



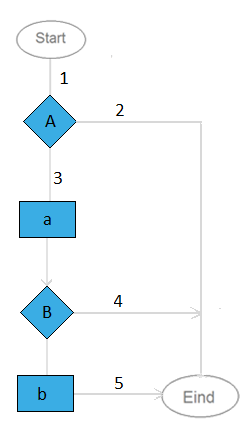


Fig. 6

Fig. 6

Fig. 7

|  |  |  |  |
| --- | --- | --- | --- |
| **Beslissingspunten** | **IN** | **OUT** | **UITKOMST** |
| A | 1 | 2, 3 | 1-2; 1-3; |
| B | 3 | 4, 5 | 3-4; 3-5 |

### Logische testgevallen aan de hand van test maat 2

|  |  |
| --- | --- |
| **Testgeval** | **Paden** |
| 1 | 1-2 |
| 2 | 1-3-4 |
| 3 | 1-3-5 |

### Padencombinatie test

|  |  |  |  |
| --- | --- | --- | --- |
| **Pad combinatie** | **TG 1** | **TG 2** | **TG 3** |
| 1-2 | X |  |  |
| 1-3 |  | X | X |
| 3-4 |  | X |  |
| 3-5 |  |  | X |
| **MOGELIJK?** | JA | JA | JA |

### Fysieke testgevallen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Invoer bullet** | **Invoer destructableBLock** | **Verwachte uitvoer** | **Result** |
| 1 | 0 | - | 0 bullet | 0 bullet |
| 2 | 2 | (1,4) | 2 bullets | 2 bullets |
| 2 | 2 | (2,4) | 2 bullets | 2 bullets |
| 3 | 3 | (4,4) | 2 bullets | 2 bullets |

### **Gebruikt grid tijdens tests**

**0 0 0 0 0   
0 1 1 1 0   
0 1 1 1 0   
0 1 1 0 0   
0 0 0 0 0**

## Samenvattingen testen

Bij elke van de drie geteste methodes hebben we algoritmisch getest. Ook hebben we statement en decision coverage toegepast en hebben we gebruik gemaakt van beslissing tabellen testen.

### Testen: Bewegen van de Held

Dit is een ingewikkelde test geweest. We hadden een goed werkende test tot we de code moesten herschrijven. Hierdoor is de move methode zo veranderd dat de JUnit test en het testontwerp moesten veranderen.

De gehele test is ontstaan door het vanuit de code een flow diagram te schrijven. We hebben twee beslissingspunten zonder loops. We zijn begonnen met het opschrijven van alle mogelijke paden. Dit resulteerde in drie testen. In deze drie testen hebben we alle mogelijke uitkomsten gecheckt. Met de drie testen hebben we gezorgd dat elk mogelijke uitkomst gecheckt wordt.

Eén van de drie testen was niet mogelijk. Door het gebruik van een ENUM word het intoetsen van een foutieve key code afgevangen en zal er verder niks gebeuren.

### Testen: Doorvoeren van beweging

Deze test is gebaseerd op het bewegen van de held. In deze test hebben we gekeken of een held een cape heeft en of de held op het einde is. Zo is elke waarde een keer bepalend voor de uitkomst.

Ook deze test is weer begonnen bij een flow diagram. Uit het flow diagram is gebleken dat er zes mogelijke wegen zijn naar het einde. Dit resulteerde in vier testen. Waarvan elke test mogelijk is.

### Testen: Fire van bullet

De test is gebaseerd op het afvuren van een kogel bij het indrukken van de spatiebalk. Er word gekeken of er genoeg kogels zijn, zo ja dan word er gekeken of het eerst volgende blok verwoest baar is.

Dit resulteerde in drie logische testen. Tijdens het omzetten van de logische testen naar fysieke testen zijn we tot de conclusie gekomen dat testgeval 2 tweemaal uitgevoerd kan en moest worden. Dit komt omdat er moet gekeken worden of de buitenste muren verwoest baar zijn en of je kan schieten op een begaanbaar pad.

# Use cases & Beschrijvingen

## Functionele eisen

* Als speler wil ik bewegen
* Als speler wil ik een bazooka oppakken
* Als speler wil ik de kortste weg vinden
* Als speler wil ik opnieuw beginnen
* Als opdrachtgever wil ik drie spel levels hebben
* Als speler wil ik bewegen met een cape
* Als speler wil ik een cheater gebruiken

## Use cases

|  |  |
| --- | --- |
| **Use case: Bewegen over veld** | |
| ID | 1 |
| Samenvatting | De actor beweegt zich door het doolhof en vind zijn vriend |
| Primaire Actors | Speler |
| Secondaire Actors | Geen |
| Precondities | Geen |
| Main Flow | 1. De use case start als de actor te kennen geeft te willen bewegen. 2. De actor beweegt over het veld[1] 3. De actor vind de vriend 4. Het systeem geeft te kennen dat het level is gehaald en start volgend level [2] |
| Postcondities | De speler heeft het level gehaald en kan nu een level verder |
| Alternative Flows | [1] De actor kan niet de gewenste richting op bewegen |

|  |  |
| --- | --- |
| **Use case: Bewegen over veld met cape** | |
| ID | 1.1 |
| Samenvatting | De actor beweegt zich door het doolhof en vind zijn vriend |
| Primaire Actors | Speler |
| Secondaire Actors | Geen |
| Precondities | Speler heeft cape opgepakt |
| Main Flow | 1. De use case start als de actor de cape heeft opgepakt en te kennen geeft te willen bewegen. 2. De actor beweegt over het veld[1] 3. De actor vind de vriend 4. Het systeem geeft te kennen dat het level is gehaald en start volgend level [2] |
| Postcondities | De speler heeft het level gehaald en kan nu een level verder |
| Alternative Flows | [1] De actor kan niet de gewenste richting op bewegen |

|  |  |
| --- | --- |
| **Alternative Flow: Bewegen over veld: loopt vast** | |
| ID | 1.1.1 |
| Samenvatting | Het systeem geeft de actor kennis dat de opgegeven richting niet bewandeld kan worden |
| Primaire Actors | Speler |
| Secondaire Actors | Geen |
| Precondities | De actor heeft aangegeven een richting op te gaan |
| Main Flow | 1. De alternative flow start na stap 2 van de main flow. 2. De speler beweegt niet 3. De use case begint weer bij stap 1. |
| Postcondities | De actor is niet bewogen |

|  |  |
| --- | --- |
| **Alternative Flow: Bewegen over veld: level gehaald** | |
| ID | 1.1.2 |
| Samenvatting | Het systeem geeft de actor kennis dat de er geen nieuwe levels meer mogelijk zijn |
| Primaire Actors | Speler |
| Secondaire Actors | Geen |
| Precondities | De speler heeft een level gehaald |
| Main Flow | 1. De alternative flow start na stap 4 van de main flow. 2. Het systeem stopt het spel |
| Postcondities | Het spel is uitgespeeld |

|  |  |
| --- | --- |
| **Use case: Bazooka gebruiken** | |
| ID | 2 |
| Samenvatting | De actor pakt een bazooka op en vuurt |
| Primaire Actors | Speler |
| Secondaire Actors | Geen |
| Precondities | De speler is bewogen door het veld |
| Main Flow | 1. De use case start als de actor te kennen geeft een bazooka te willen oppakken. 2. Het systeem geneerd een aantal kogels 3. De actor beweegt over het veld 4. De actor geeft te kennen te willen schieten. 5. Het systeem verwijdert een deel van de muur 6. Het systeem creëert een nieuw pad. 7. Het systeem vermindert het aantal kogels [1] |
| Postcondities | De actor heeft een muur verwijderd |
| Alternative Flows | - |

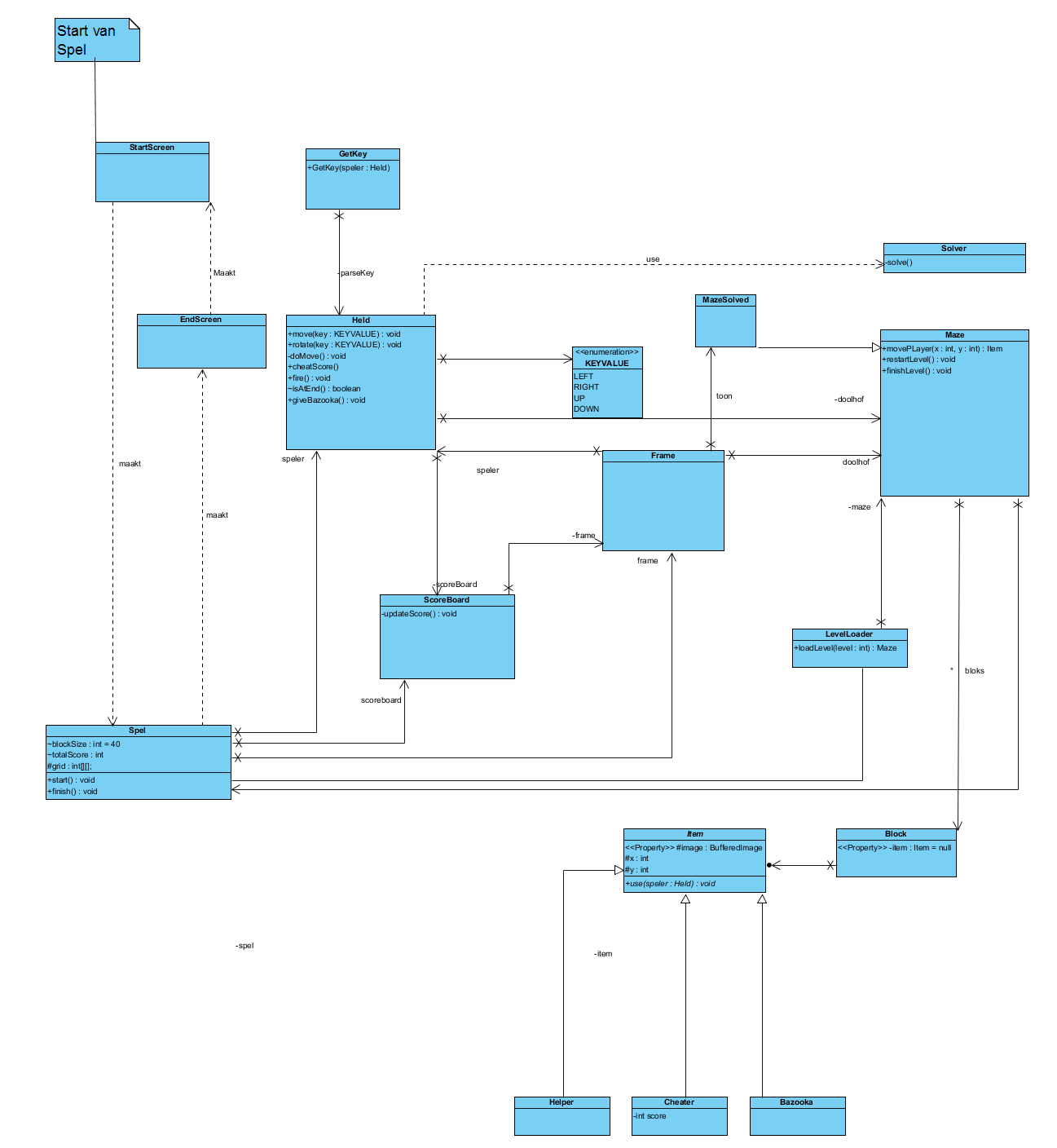
|  |  |
| --- | --- |
| **Alternative Flow: Bazooka gebruiken** | |
| ID | 2.1 |
| Samenvatting | De kogels zijn op en bazooka word verwijdert |
| Primaire Actors | Speler |
| Secondaire Actors | Geen |
| Precondities | De speler heeft een kogel afgevuurd |
| Main Flow | 1. De alternative flow start na stap 7 van de main flow. 2. Het systeem verwijdert de bazooka |
| Postcondities | De actor heeft een muur verwijderd |

|  |  |
| --- | --- |
| **Use case: Helper gebruiker** | |
| ID | 3 |
| Samenvatting | De helper wijst de korste weg |
| Primaire Actors | Helper |
| Secondaire Actors | Speler |
| Precondities | Gebruikt de helper |
| Main Flow | 1. De use case start als de actor te kennen geeft de helper te gebruiken. 2. Het systeem rekent de kortste weg 3. Het systeem deel de score door 2 4. Het systeem verwijdert de helper |
| Postcondities | De actor heeft doormiddel van de helper de kortste weg gevonden |
| Alternative Flows | Geen |

|  |  |
| --- | --- |
| **Use case: Cheater gebruiken** | |
| ID | 4 |
| Samenvatting | De cheater verandert de score |
| Primaire Actors | Cheater |
| Secondaire Actors | Speler |
| Precondities | Gebruik de cheater |
| Main Flow | 1. De use case start als de actor te kennen geeft de cheater te gebruiken. 2. Het systeem rekent een nieuwe score 3. Het systeem verwijdert de cheater |
| Postcondities | De speler heeft doormiddel van de cheater zijn score gemanipuleerd |
| Alternative Flows | Geen |

|  |  |
| --- | --- |
| **Use case: Opnieuw spelen** | |
| ID | 5 |
| Samenvatting | De actor is opnieuw begonnen |
| Primaire Actors | Speler |
| Secondaire Actors | Geen |
| Precondities | De actor is een spel begonnen |
| Main Flow | 1. De use case start als de actor te kennen geeft het spel opnieuw te willen spelen 2. Het systeem zet de speler terug naar start en de score terug naar de standaard waarde |
| Postcondities | De actor is het spel opnieuw begonnen |
| Alternative Flows | Geen |

# Analysis diagram

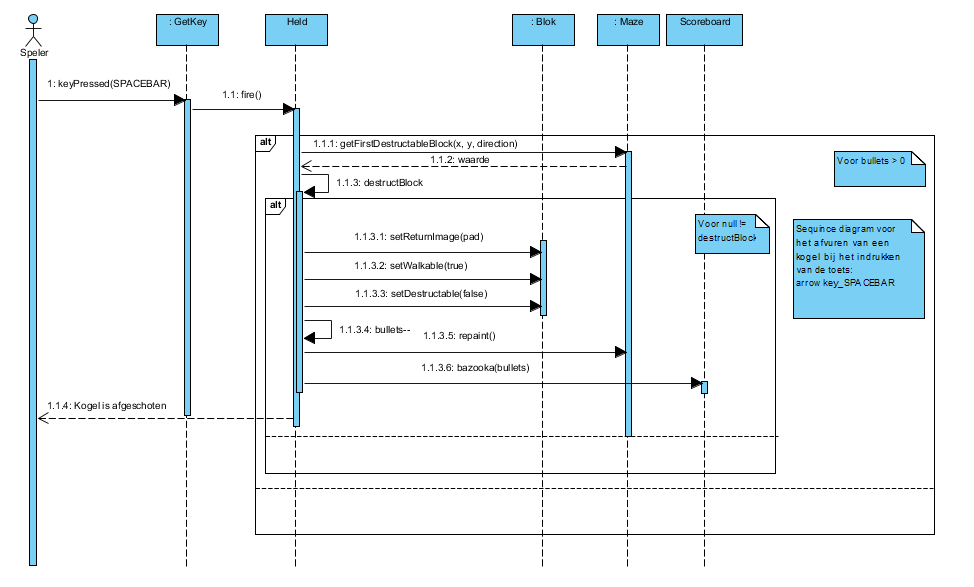


# Design Diagram

# Sequence Diagram: Bewegen van Held

1: De speler drukt de UP toets in.  
2: De wijziging in locatie wordt (tijdelijk) doorgevoerd.  
3: De afbeelding van de held wordt gedraaid.  
4: De bewandelbaarheid van het blok met de ingevulde coördinaten wordt opgevraagd.  
5: Als het blok een onbewandelbaar is wordt de locatie van de held terug gezet .  
6: Deze methode is verantwoordelijk voor het parsen van de beweging.  
 7: De hoeveelheid gezette stappen in het huidige level wordt vergroot met 1.  
 8: De afbeelding van de speler wordt geupdate (met mogelijk een geroteerde versie).  
 9: Het doolhof voegt de speler aan een nieuw blok toe .  
 10: Als er een item op het nieuwe blok is wordt deze gebruikt.  
 11: De GUI wordt geupdate.  
 12; Speler krijgt te zien dat held verplaatst is.  
 13: Het scoreboard wordt geïnformeerd over de beweging (het scoreboard zal de score bijwerken).  
 14: Controleert of de held op eindpunt is door de huidige locatie van de held met het eindpunt te vergelijken.  
 15: het eindpunt van het doolhof wordt opgehaald.  
 16: Als het einde bereikt is wordt het level afgerond en aan het spel om een nieuw level gevraagd.

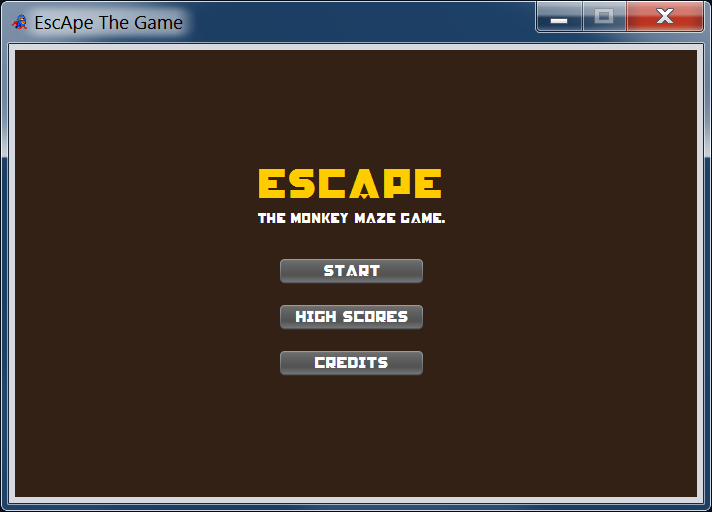
# Sequence Diagram: Fire of bullet



1: De speler drukt de spatiebalk toets in.  
1.1: De fire methode word aangeroepen, en gaat verder als er genoeg kogels zijn.  
1.1.1: Er word een blok opgehaald.  
1.1.2: block word teruggegeven.  
1.1.3: Er word gekeken of blok verwoest baar is en gaat dan verder met uitvoeren en anders is de methode klaar.  
1.1.3.1: Verandert de muur in een pad.  
1.1.3.2: Maakt het pad begaanbaar.  
1.1.3.3: Zet het pad op onverwoestbaar.  
1.1.3.4: Haalt er één kogel af.  
1.1.3.5: De GUI wordt geupdate.  
1.1.3.6:Er word meegegeven hoeveel kogels er nog over zijn aan de bazooka.

# Applicatie preview

Na het starten van de applicatie:



Na het lopen van enkele stappen:



Na het oppakken van de bazooka en de solver:



Na het voltooien van alle levels:



# Afbeeldingen Legenda

|  |  |
| --- | --- |
| C:\Users\Chris\Documents\GitHub\BLOKD\BLOK_D\src\images\pad.png | Pad (Bewandelbaar) |
| C:\Users\Chris\Documents\GitHub\BLOKD\BLOK_D\src\images\wall.png | Muur (Onbewandelbare) |
| C:\Users\Chris\Documents\GitHub\BLOKD\BLOK_D\src\images\heldRight.png | Held (bestuurd door speler) |
| C:\Users\Chris\Documents\GitHub\BLOKD\BLOK_D\src\images\cheater.png | Cheater |
| C:\Users\Chris\Documents\GitHub\BLOKD\BLOK_D\src\images\bomb.png | Bazooka |
| C:\Users\Chris\Documents\GitHub\BLOKD\BLOK_D\src\images\helper.png | Solver |
| C:\Users\Chris\Documents\GitHub\BLOKD\BLOK_D\src\images\solvedPad.png | Snelste route (wordt aangemaakt door solver) |
| C:\Users\Chris\Documents\GitHub\BLOKD\BLOK_D\src\images\start.png | start |
| C:\Users\Chris\Documents\GitHub\BLOKD\BLOK_D\src\images\finish.png | Eindpunt (vriend) |
| C:\Users\Chris\Desktop\BLOKD\BLOK_D\src\images\cape.png | Cape |

# Instructie level configuratie

EscApe is zo gebouwd dat het zeer gemakkelijk is om nieuwe levels toe te voegen, hier is geen enkele programmeer kennis voor nodig, de applicatie zorgt er zelf voor dat het level wordt ingeladen en weergeven.

*Voorbeeld level bestand:*

|  |
| --- |
| <start>  1  1  <end>  10  5  <height>  7  <width>  12  <cheater>  6  1  <bazooka>  4  5  <helper>  1  4  <grid>  000000000000  011110111010  001011101010  011010011010  010010010010  011111111110  000000000000  </grid> |

Alle specifieke blokken (start, end, cheater, bazooka, en helper) bevatten een x en een y coördinaat, dit is de locatie waar ze in het level zullen verschijnen, als er een 0 wordt ingevuld zal het item niet worden getekend (niet mogelijk met start en end)  
  
het grid is een representatie van het doolhof, waarin de nullen de muren zijn, en de enen de bewandelbare paden. Height en width zijn de hoogte en breedte van het grid.

Om een level leesbaar te maken voor het spel moet het een specifieke naam hebben: level[n].txt waarin [n] het eerstvolgende gehele getal is in de reeks met levelbestanden, te vinden in ~src/leveldata

1. De reden dat TG2 niet mogelijk is omdat keypress een ENUM is en niet anders kan zijn dan de 4 weergegeven waardes (LEFT, RIGHT, UP, DOWN). [↑](#footnote-ref-1)
2. Er is hier gekozen om een extra test te maken waarbij we controleren of het bewegen op de Y-as ook mogelijk is [↑](#footnote-ref-2)