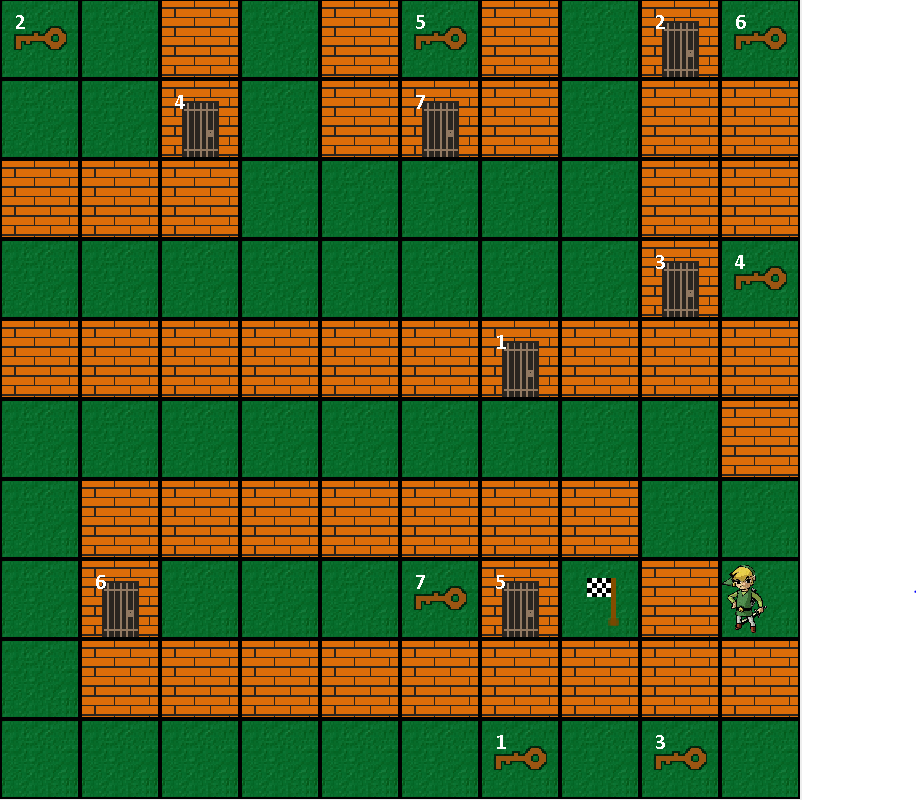
Ontwikkel document



Raymond Lee, 11106298

Fabio Waljaard, 15086836

Joris Willig, 15077179

04-07-2016, den Haag

Inhoud

[Applicatie eisen 3](#_Toc447832705)

[De basis 3](#_Toc447832706)

[De uitbreiding 3](#_Toc447832707)

[UML 4](#_Toc447832708)

[Analyse diagram 4](#_Toc447832709)

[Design diagram basis 4](#_Toc447832710)

[Design diagram met uitbreiding 5](#_Toc447832711)

[Tests 6](#_Toc447832712)

[Character movement 6](#_Toc447832713)

[KeyTile 7](#_Toc447832714)

[BoardCoordinate 8](#_Toc447832715)

[Bijlagen 9](#_Toc447832716)

# Applicatie eisen

In het spel moet de speler doormiddel van het oppakken van sleutels en het daarmee openen van barricades de finish bereiken.

## De basis

De eisen:

Er zijn verschillende onderdelen in het spel:

* Het karakter, deze wordt bestuurd door de speler.
* Tegels, hiervan zijn er meerdere soorten die straks zullen worden toegelicht.
* Sleutels, deze kunnen barricades (een soort tegel) weghalen.

Het spelboord bestaat uit de volgende soorten tegels:

* Muren, deze staan vast in het spel en het karakter kan hier niet door heen lopen.
* Barricades, deze hebben een bepaalde pincode en kunnen alleen door een sleutel met dezelfde pincode geopend worden. Het karakter kan hier niet door heen lopen.
* De finish, wanneer het karakter op deze tegel komt te staan is het level voorbij en komt er een popup box tevoorschijn die zegt dat je gewonnen hebt.
* Lege tegels, deze vullen de ruimtes op die nog niet gevuld zijn met de andere soorten tegels.

Het oppakken van sleutels werkt als volgt:

De speler kan een sleutel oppakken door over een sleutel in een lege tegel heen te lopen. Wanneer het karakter al een sleutel bij zich droeg wordt deze verwijderd en vervangen door de net opgepakte sleutel. Opgepakte sleutels zijn niet neer te leggen.

Het karakter beweegt door middel van de pijltjestoetsen. Deze beweging wordt zoals eerder vermeld beperkt door de barricades en muren maar ook door de randen van het spel.

Tijdens het spel is het mogelijk om het level opnieuw te starten en om terug te gaan naar het menu om andere levels te kiezen.

## De uitbreiding

Wij hebben als uitbreiding een level editor toegevoegd waarin je je eigen levels kan maken en opslaan om te spelen.

Hier zijn de volgende eisen aan gesteld:

Wanneer de editor wordt geopend, wordt er een scherm geopend dat een speelveld laat zien, gevuld met lege tegels. Naast het speelveld zijn knoppen aanwezig waar de gebruiker een tegel kan selecteren om die vervolgens te plaatsen door op tegels te klikken met de muis. In een tekst veld boven deze knoppen kan een pincode ingevoerd worden die toegepast wordt op barricades en sleutels die daarna geplaatst worden. Wanneer hier een ongeldige invoer in staat (iets dat geen integer is of geen invoer) wordt de pincode 0 toegepast. De gebruiker kan het gemaakte level opslaan onder een zelfgekozen naam. Wanneer deze naam al bestaat onder de bestaande levels krijgt de gebruiker een popup te zien die dit laat zien.

# UML

## Analyse diagram

Hier onder spreken wij over figuur 1, te vinden in de bijlagen.

Een gameboard bestaat uit Tiles. In een Tile kan een barricade, sleutel, Finish of een Wall liggen. Deze tiles verschillen van elkaar en zijn daarom aparte klasses.

De gameboard bestuurt de GameCharacter.

Een GameCharacter kan een sleutel vasthouden en een GameCharacter staat op een Tile.

De GameCharacter kan met een Key, een Barricade openen.

## Design diagram basis

Hier onder spreken wij over figuur 2, te vinden in de bijlagen.

Uit het design diagram is te zien dat de taken van het lezen van een bestand tot het bouwen van het spel gedelegeerd zijn naar andere klassen. Een klasse die een bestand inleest, een klasse die de level bouwt. Een klasse die het hele spellogica reguleert met behulp van coördinatie. Een abstracte klasse waar subklassen van de superklasse overerven. Door de taken te delegeren zorgt ervoor dat polymorfisme mogelijk is.

Game class  
Hierin staan de constante variabelen die het hele spel gebruikt.   
De Game class gebruikt de levelReader om levels in te lezen uit een bestand, wij hebben gekozen om dit binnen Game te doen omdat deze klasse ook het GameBoard beheerd.

LevelReader class  
LevelReader leest in uit een bestand en maakt Level aan.

Level class  
Level class maakt Tiles aan, heeft een BoardCoordinate als startPositie van een character.

GameBoard class  
GameBoard gebruikt Level als parameter en heeft een associatie met Level omdat currentLevel wordt bijgehouden.  
GameBoard gebruikt BoardCoordinate om met GameCharacter bewegingen om te gaan.  
Hiervoor wordt MoveDirection als parameter gebruikt.  
GameBoard bestaat uit Tiles en een GameCharacter.

Tile class  
Tile gebruikt GameCharacter als parameter. Tile is een abstracte class omdat er specifiekere klasses gebruikt worden.

EmptyTile, Finish en Wall classes  
Deze klassen overerven van Tile class.

KeyTile class  
KeyTile overerft ook van Tile class en bevat altijd een Key, daarom is er gekozen voor een aggregaat.

Barricade class  
Barricade overerft ook van Tile en gebruikt Key om te kijken of de speler over de Tile mag lopen.

MoveDirection class  
MoveDirection werkt alleen met BoardCoordinate, maar de relatie is van korte duur. Daarom is er een dependency gekozen met use, parameter en create.

## Design diagram met uitbreiding

Hier onder spreken wij over figuur 3, te vinden in de bijlagen.

### Editor

Editor gebruikt Level als parameter, bestaat uit Tiles en heeft twee associaties met BoardCoordinate voor de start positie van GameCharacter en voor het geselecteerde vakje.

Het opslaan van levels in bestanden wordt gedaan in een aparte klasse LevelWriter, om de logica te scheiden. Deze klasse wordt kort gebruikt door Editor, daarom is er een dependency tussen Editor en LevelWriter. Om de verschillende types Tile bij te houden wordt de enum TileType gebruikt. Omdat dit voor een lange tijd wordt bijgehouden is er een associatie tussen Editor en TileType.

# Tests

## Character movement

### testMoveCharacterToBarricade

De GameCharacter staat links van twee barricades met keyCode 4, eerst word getest of de speler naar links kan gaan als hij de sleutel met keyCode 4 bezit. Er wordt verwacht dat dit mogelijk is, omdat de barricade geopend word. Daarna word getest of de speler naar links kan gaan als hij de sleutel met keyCode 5 bezit. Er wordt verwacht dat dit niet mogelijk is omdat de sleutel niet past op de barricade. Als laatste wordt getest of de speler naar links kan gaan als hij geen sleutel bezit. Er wordt verwacht dat dit niet mogelijk is omdat de barricade niet geopend wordt.

### testMoveCharacterToWall

De GameCharacter staat onder een muur, er wordt getest of de speler naar boven kan bewegen. Er wordt verwacht dat dit niet kan omdat spelers niet door muren heen mogen lopen.

### testMoveCharacterOutOfGameEdge

De GameCharacter staat rechtsonder op het bord. Er wordt getest of de speler op het bord blijft wanneer hij rechts of naar beneden probeert te bewegen. Er wordt verwacht dat de speler niet uit het bord komt.

De GameCharacter staat linksboven op het bord. Er wordt getest of de speler op het bord blijft wanneer hij links of naar boven probeert te bewegen. Er wordt verwacht dat de speler niet uit het bord komt.

### Coverage

Deze tests voldoen aan decision coverage ten aangaande van het bewegen van het karakter omdat alle mogelijke scenarios getest worden: het openen van barricades, het lopen tegen muren, het blijven op het bord en het lopen tegen gesloten barricades.

## KeyTile

Bij creatie van de KeyTile werd een key met code 2 gebruikt.

### testIsPassable

Er wordt getest of de KeyTile true teruggeeft bij isPassable(). Er wordt verwacht dat de KeyTile true teruggeeft omdat de speler er op mag gaan staan.

### testGetKey

Er wordt getest of een KeyTile de juiste Key teruggeeft bij getKey. Er wordt verwacht dat de getKey methode een Key teruggeeft met code 2.

### testGetReplacement

Er wordt getest of een KeyTile de juiste vervangende Tile teruggeeft bij getReplacement(). Er wordt verwacht dat de vervangende Tile een EmptyTile is.

Er wordt getest of de vervangende Tile dezelfde positie heeft. Er wordt verwacht dat de positie van het origineel en de vervanging gelijk blijft.

### testOnCharacterEnter

Er wordt getest of een KeyTile zijn Key afgeeft aan GameCharacter wanneer onCharacterEnter(char) aangeroepen wordt. Er wordt verwacht dat de GameCharacter na onCharacterEnter een Key bezit met code 2.

### Coverage

Deze test voldoen aan code coverage omdat alle methodes binnen deze klasse volledig getest worden.

## BoardCoordinate

### testGetX / testGetY

Bij deze tests word verwacht dat de methoden de juiste X en Y waarden teruggeven. Deze waarden moeten gelijk zijn aan de waarden die gebruikt worden tijdens de creatie van het object.

### testGetLeft / testGetRight / testGetDown / testGetUp

Bij deze tests word verwacht dat de methoden het juiste coördinaat teruggeven, het coördinaat wordt verwacht links/rechts/boven/onder het origineel te liggen.

### testCompareTo

Er wordt getest wat de compareTo(pos) methode teruggeeft bij twee coördinaten. Bij gelijke coördinaten wordt een 0 verwacht. Wanneer het object groter is dan de parameter wordt een getal groter dan 0 verwacht. Wanneer het object kleiner is dan de parameter wordt een getal onder 0 verwacht.

### testConstructorLimit

Er wordt getest of BoardCoordinates kunnen worden aangemaakt met onjuiste coördinaten. Bij negatieve coördinaten wordt verwacht dat het coördinaat op de minimale waarde ( 0 ) wordt gezet. Bij coördinaten die buiten het spel liggen wordt verwacht dat het coördinaat op de maximale waarde ( het aantal kolommen - 1, het aantal rijen - 1 ) wordt gezet.

### Coverage

De tests op deze klasse voldoen aan code coverage omdat elke methode met diverse parameters getest wordt.

# Bijlagen

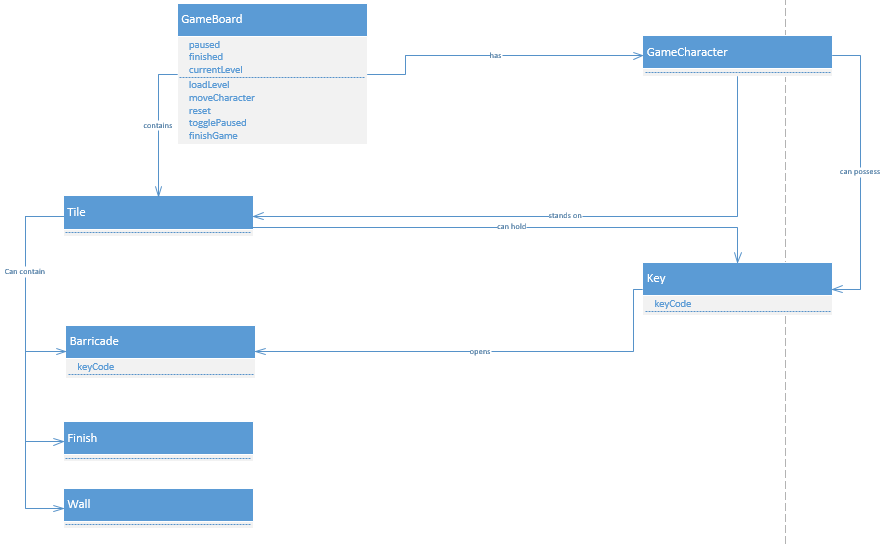


Figure 1 Analyse diagram

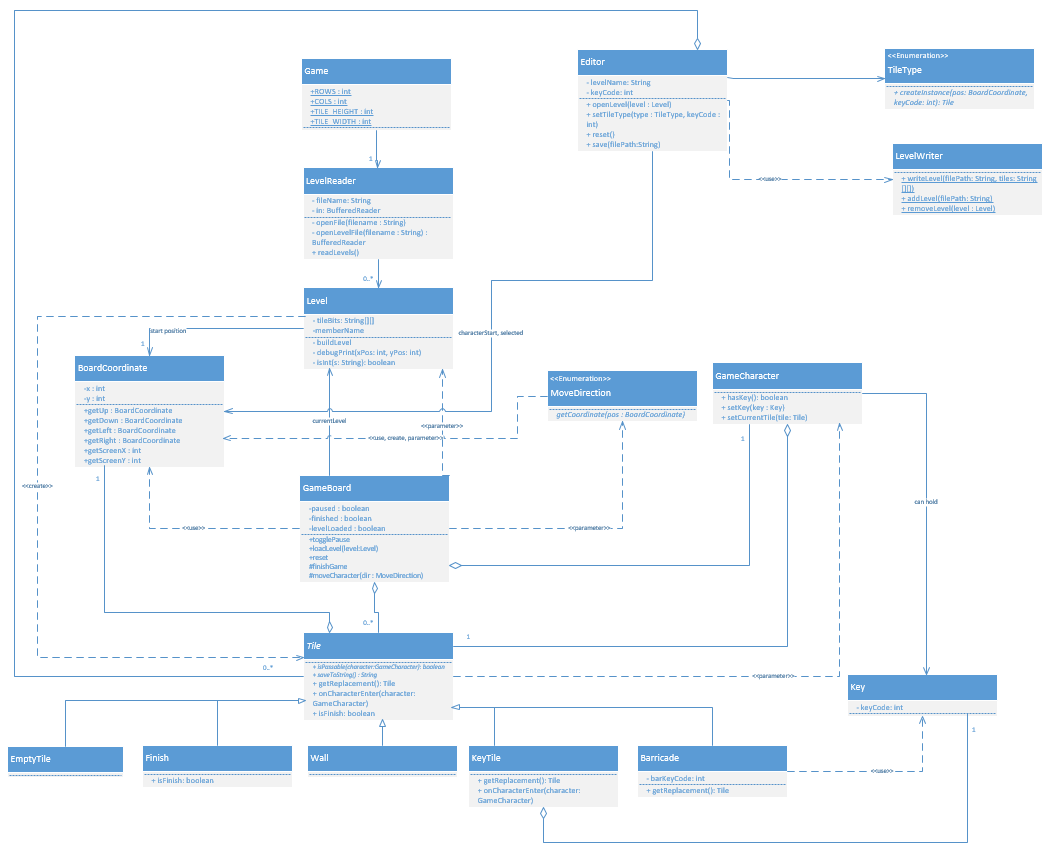


Figure 2 Design diagram basis

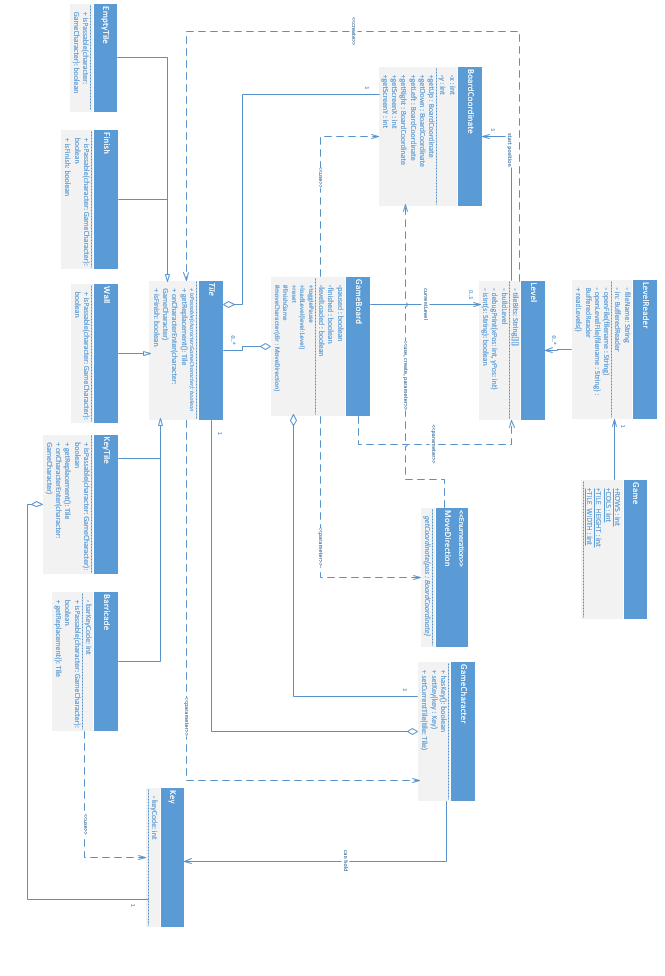


Figure 3 Design diagram met uitbreiding