Project Zuul

Een text based adventure game

Instituut voor Communicatie, Media & IT

Hanzehogeschool Groningen
University of Applied Sciences

Inhoudsopgave

1. Inleiding **3**

2. Werkwijze **4**
2.1. Teams . . . 4
2.2. Bijeenkomsten . . . 4
2.3. Versiebeheer . . . 5

3. Requirements **5**
3.1. Functionaliteit . . . 5
3.2. Uitbreidingen . . . 6
3.3. Code . . . 7
3.4. Restricties . . . 7

4. Oplevering **7**
4.1. Documentatie . . . 7
4.1.1. One-Page Quick Start Guide . . . 8
4.1.2. Technical Notes . . . 8
4.2. Git bundle . . . 8
4.3. Reviews . . . 8

5. Beoordeling **9**
5.1. Code . . . 9
5.2. Documentatie . . . 9
5.3. Reviews . . . 9
5.4. Demo . . . 9

6. Planning **10**
6.1. Week 1 . . . 10
6.1.1. Tutoraat . . . 10
6.2. Week 2 . . . 10
6.2.1. Activiteiten . . . 11
6.2.2. Tutoraat . . . 11
6.3. Week 3 . . . 11
6.3.1. Activiteiten . . . 11
6.3.2. Tutoraat . . . 11
6.4. Week 4 . . . 12
6.4.1. Activiteiten . . . 12

6.4.2. Tutoraat 12
6.5. Week 5 12
6.5.1. Activiteiten 12
6.5.2. Tutoraat 12
6.5.3. Deadline 13
6.6. Week 6 13
6.6.1. Activiteiten 13
6.6.2. Tutoraat 13
6.6.3. Deadline 13

A. Bijlage **13**
A.1. Git bundles 13
A.2. Quick Start Guide 14
A.2.1. Dungeon Generator 14
A.2.2. Trizbort.io 15
A.3. Technical Notes 15
A.3.1. Indeling 16

1. Inleiding

De komende weken ga je in tweetallen hoofdstuk 8 (Klassen ontwerpen) uit het boek „Programmeren in Java met BlueJ" doorlopen waar je een text based adventure game in stappen opbouwt.[1] Het spel kent een lange historie en is gebaseerd op het klassieke spel Colossal Cave Adventure uit 1976, in figuur 1 zie je hoe dit er in 1977 uitzag en jouw implementatie zal hier heel erg op gaan lijken!

Het doel van dit project is om een aantal principes van software ontwerp toe te passen met betrekking tot afhankelijkheid, cohesie en het scheiden van verantwoordelijkheden. Daarnaast zal je de basisimplementatie die het boek jou geeft uitbreiden met een eigen verhaallijn en spelelementen.

Naast een correct werkend spel zal je ook documentatie moeten opleveren die specifiek gericht is op de eindgebruiker (in de vorm van een (one-page) quick start guide) en vakgenoten (technical notes). De beoordeling van het werk vindt plaats door de docent en *drie* andere projectgroepen (peer review) waar zal worden gekeken naar de uitwerking van het spel, documentatie en ontwikkelhistorie (door middel van Git).

[1] Als een klas een oneven aantal studenten telt zal één team uit drie leden bestaan.

```
.run adven


WELCOME TO ADVENTURE!!  WOULD YOU LIKE INSTRUCTIONS?

yes

SOMEWHERE NEARBY IS COLOSSAL CAVE, WHERE OTHERS HAVE FOUND FORTUNES IN
TREASURE AND GOLD, THOUGH IT IS RUMORED THAT SOME WHO ENTER ARE NEVER
SEEN AGAIN.  MAGIC IS SAID TO WORK IN THE CAVE.  I WILL BE YOUR EYES
AND HANDS.  DIRECT ME WITH COMMANDS OF 1 OR 2 WORDS.  I SHOULD WARN
YOU THAT I LOOK AT ONLY THE FIRST FIVE LETTERS OF EACH WORD, SO YOU'LL
HAVE TO ENTER "NORTHEAST" AS "NE" TO DISTINGUISH IT FROM "NORTH".
(SHOULD YOU GET STUCK, TYPE "HELP" FOR SOME GENERAL HINTS.  FOR INFOR-
MATION ON HOW TO END YOUR ADVENTURE, ETC., TYPE "INFO".)
                                - - -
THIS PROGRAM WAS ORIGINALLY DEVELOPED BY WILLIE CROWTHER.  MOST OF THE
FEATURES OF THE CURRENT PROGRAM WERE ADDED BY DON WOODS (DON @ SU-AI).
CONTACT DON IF YOU HAVE ANY QUESTIONS, COMMENTS, ETC.

YOU ARE STANDING AT THE END OF A ROAD BEFORE A SMALL BRICK BUILDING.
AROUND YOU IS A FOREST.  A SMALL STREAM FLOWS OUT OF THE BUILDING AND
DOWN A GULLY.

east

YOU ARE INSIDE A BUILDING, A WELL HOUSE FOR A LARGE SPRING.

THERE ARE SOME KEYS ON THE GROUND HERE.

THERE IS A SHINY BRASS LAMP NEARBY.

THERE IS FOOD HERE.
```

Figuur 1: Terminal output van het spel Colossal Cave Adventure op een PDP-10 (1977)

2. Werkwijze

2.1. Teams

Dit project wordt door tweetallen uitgevoerd en tijdens de *eerste* bijeenkomst worden teams gevormd. Je bent vrij om jouw teamlid te kiezen hoewel de docent kan ingrijpen als bijvoorbeeld het niveau van de teamleden teveel verschilt.

2.2. Bijeenkomsten

Elke week heb je met jouw team een bijeenkomst met de projectdocent. Tijdens deze bijeenkomst wordt de voortgang besproken en eventuele problemen. Mogelijke zal je voor bijeenkomsten ook werk moeten voorbereiden, zie de planning voor details per week. De bijeenkomsten zijn verplicht

en aanwezigheid is voorwaarde voor een eindbeoordeling. Mocht je vanwege omstandigheden niet aanwezig kunnen zijn dan laat je dit jouw projectdocent tijdig en met opgaaf van reden weten.

2.3. Versiebeheer

Versiebeheer ga je toepassen door middel van Git en gebruik GitHub om onderling te synchroniseren. De project repository op GitHub mag privé zijn want de code die je ter beoordeling gaat opleveren deel je door middel van een Git *bundle* (een repository archiefbestand).

BlueJ is geschikt als ontwikkelomgeving voor dit project (maar niet verplicht) en biedt ook Git ondersteuning, zie de handleiding op https://www.bluej.org/tutorial/git/.

3. Requirements

Je zal een spel opleveren dat alle functionaliteit bevat die in de opdrachten van het boek jou wordt gevraagd uit te werken. Daarnaast zal je jouw implementatie moeten aanvullen met een aantal (verplichte) uitbreidingen.

3.1. Functionaliteit

- het spel heeft meerdere locaties en/of ruimtes (minimaal 6)
- de speler kan zich tussen locaties verplaatsen
- elke ruimte kan een willekeurig aantal items bevatten
- sommige items kunnen door de speler worden opgepakt, anderen niet
- elk item heeft een gewicht en de speler kan maar een beperkt gewicht dragen
- de speler moet kunnen winnen: er moet een situatie zijn die herkend wordt als het einde van het spel waarbij de speler geïnformeerd wordt dat hij of zij heeft gewonnen
- voeg minstens vier nieuwe commando's toe
- één van deze commando's is `back` die de speler naar de vorige locatie brengt
- de speler kan `back` meerdere keren invoeren, er zal dus een historie moeten worden bijgehouden
- alle locaties hebben duidelijke beschrijvingen en bevatten hints voor de speler
- het spel heeft een menu inclusief een *Quit* and *About* optie
- het spel moet buiten BlueJ kunnen worden gespeeld door middel van een `main` methode (zie opgave 8.49 en verder bijlage E in het boek)

3.2. Uitbreidingen

Implementeer minstens **twee** van de volgende uitbreidingen:

- **Trapdoor**

  Voeg een „valluik" toe, waar bij betreden de speler naar een een willekeurig andere ruimte wordt verplaatst. Misschien verliest de speler onderweg ook één of meerdere items uit zijn of haar collectie? Of zou het commando `up` kunnen betekenen dat de speler in een eindeloze ruimte verdwijnt … ?

- **Restricted**

  Voeg afgesloten ruimtes toe. De speler zal de sleutel moeten vinden (of op een andere manier kunnen verdienen) om toegang te krijgen.

  Een variant is het introduceren van verborgen ruimtes. Deze kunnen alleen zichtbaar worden met een bijzondere *access pass* die de speler in een andere ruimte zal moeten vinden.

- **Timelimit**

  Voeg een tijdbegrenzing toe (opgave 8.41). Laat de speler ook bij elke stap weten dat de klok tikt en het spel op elk moment afgelopen kan zijn!

- **Beamer**

  De speler kan zichzelf teleporteren naar een eerder bezochte ruimte (opgave 8.43).

- **Actors**

  Voeg figuranten of actoren toe (zie opgaven 8.47 & 8.48) en breidt de command parser uit om commando's van drie woorden te herkennen, bijvoorbeeld dat de speler brood aan een elf kan geven (`give elf bread`).

- **Configuration**

  Lees de spelspecificatie uit een bestand in plaats van een harde codering zodat het spel meerdere scenarios kan spelen.

- **State**

  Implementeer een *Save* en *Load* commando dat de spelstatus opslaat en laadt.

- **Custom**

  Mocht je zelf een idee hebben voor een uitbreiding, bespreek deze dan met jouw docent om te bepalen of het voldoende complexiteit bevat.

3.3. Code

- gebruik Git voor versiebeheer
- de code is correct geformatteerd[2]
- de code is gedocumenteerd met JavaDoc, documenteer zowel klassen als methodes
- in JavaDoc neem je ook informatie op wie aan welke functionaliteit heeft gewerkt door middel van de `@author` tag
- voeg meerdere `@author` tags toe als je aan hetzelfde hebt gewerkt

3.4. Restricties

Jouw team mag een implementatie opleveren die afwijkt of verder gaat dan de text based versie van het spel, bijvoorbeeld door het toevoegen van een grafische interface. Dit mag echter geen afhankelijkheden introduceren en het zal moeten kunnen worden uitgevoerd door een standaard Java JDK (OpenJDK versie 11) installatie.

4. Oplevering

Je zal een viertal producten gaan opleveren:

- Code
    1. een correct werkend spel
- Documentatie
    2. een Quick Start Guide voor gebruikers
    3. Technical Notes voor vakgenoten
- Review
    4. beoordelingen van drie andere teams

4.1. Documentatie

Tijdens de opleiding zal je vaak volledige documentatie moeten opleveren waar jij jouw keuzes met betrekking tot het analyseren van een problemen en het ontwerpen en realiseren van een oplossing beschrijft en verantwoordt. In mindere mate wordt aandacht besteed aan communicatie richting de eindgebruiker die jouw software zal gaan gebruiken.

[2] Gebruik de Ctrl-Shift-i optie in BlueJ

Maar ook technische documentatie zal niet altijd even uitgebreid moeten zijn. In dit project communiceer je met vakgenoten (medestudenten) die aan hetzelfde probleem werken als jij, zij zullen eerder geïnteresseerd zijn in waar je afwijkt van de standaardimplementatie of een oplossing voor een specifiek probleem hebt gevonden.

Voor beide vormen van communicatie geldt „*less is more*", oftewel de kunst van het weglaten en dit is niet altijd even eenvoudig! Je zal daarom ter begeleiding van het spel ook een Quick Start Guide voor de eindgebruiker en Technical Notes voor vakgenoten opleveren.

4.1.1. One-Page Quick Start Guide

Je gaat een Quick Start Guide van maximaal één pagina opstellen. Je zal hier goed moeten nadenken over de representatie, is tekst voldoende om een snel overzicht te krijgen of is een grafische representatie meer op zijn plaats (of een combinatie)? Met andere woorden, hoe weet de gebruiker met minimale moeite wat het spel is, hoe het gespeeld wordt en en wat het einddoel is? Zie verder de bijlage voor een aantal ideeën.

4.1.2. Technical Notes

Naast een Quick Start Guide schrijf je een samenvatting van maximaal twee pagina's gericht aan vakgenoten, dat wil zeggen medestudenten die aan exact hetzelfde probleem werken als jij. De basisimplementatie is voor iedereen gelijk (deze krijg je van het boek), wat je in de Technical Notes (of „aantekeningen") zal moeten gaan beschrijven zijn de *bijzonderheden* van jouw implementatie waar het gaat om de uitbreidingen die je hebt gekozen en specifieke keuzes die je hebt moeten maken.

Waar de Quick Start Guide bedoeld is als een navigatieinstrument voor de gebruiker zijn Techical Notes van belang voor een vakgenoot om snel jouw implementatie te kunnen begrijpen en doelgericht de code te kunnen lezen. Zie de bijlage voor meer informatie over het opstellen van Technical Notes.

4.2. Git bundle

Code lever je op door middel van en Git bundle, een archiefbestand. Dit bestand bevat alle code en een volledige *commit* geschiedenis. Zie de bijlage voor meer informatie over het maken en gebruiken van Git bundles.

4.3. Reviews

Je gaat het werk van *drie* teams reviewen en beoordelen (zowel code als documentatie). Deze beoordelingen maken deel uit van het op te leveren werk.

5. Beoordeling

5.1. Code

- de code compileert
- een duidelijk spelscenario is uitgewerkt
- de minimale set aan requirements is geïmplementeerd
- kamers bevatten duidelijke beschrijvingen en handelingen
- minstens twee uitbreidingen zijn toegevoegd
- de code is correct gedocumenteerd met JavaDoc
- minstens 60% van klassen en methoden is gedocumenteerd (coverage)
- elk teamlid heeft een evenredig aantal commits geplaatst tot maximaal 60% van alle commits[3]

5.2. Documentatie

- de Quick Start Guide is duidelijk en direct bruikbaar
- de Technical Notes zijn duidelijk en volledig
- de Technical Notes corresponderen met de Quick Start Guide met betrekking tot uitbreidingen en een „shortest path to exit” (zie bijlage).

5.3. Reviews

Jouw werk zal worden beoordeeld door een drietal andere teams. Deze beoordelingen maken deel uit van het eindcijfer.

5.4. Demo

Ter afsluiting van het project zal je in een demo van ~20 minuten jouw werk samen met de projectdocent doorlopen en dit is het moment waar de docent een oordeel over jouw werk zal vormen. Deze beoordeling, samen met reviews van andere teams zal resulteren in een eindbeoordeling.

[3] Deze voorwaarde is opgenomen om er voor te zorgen dat *beide* teamleden Git gebruiken, er worden verder geen kwalitatieve gegevens uit afgeleid die het eindcijfer beïnvloeden. Gebruik `git shortlog -nse --no-merges` om snel een verhouding van commits te zien.

6. Planning

Hoofdstuk 8 (Klassen ontwerpen) geeft jou een basisimplementatie van het spel en in een aantal iteraties zal je het ontwerp gaan aanpassen, verbeteren en uitbreiden. Weersta de verleiding om stappen over te slaan door bijvoorbeeld met project `zuul-better` te beginnen zonder eerst `zuul-bad` te hebben doorlopen:

- tussen de twee projecten door begin je gelijk al met functionaliteit toevoegen;
- met het stap voor stap doorlopen van de projecten krijg je het probleem „in de vingers", het helpt jou het probleem en de oplossing te begrijpen;
- het probleem goed begrijpen maakt het gemakkelijker de verplichte uitbreidingen te implementeren.

6.1. Week 1

9 december – 13 december

Teams, spelscenario, een eerste blik op code en voorbereiding van de ontwikkelomgeving

6.1.1. Tutoraat

Onderwerpen:

- teams vormen
- indeling van teams per tutoraat

Activiteiten:

- dit document volledig lezen
- boek 8.1 – 8.3: spelscenario en plattegrond
- boek 8.4: kennismaken met code en direct een eerste verbetering!
- eventueel aanmaken projectrepository op GitHub

6.2. Week 2

16 december – 20 december

De eerste uitbreidingen: kamers maken, rondkijken, bewegen en meer verbeteringen

In het project zal je een aantal datatypes tegenkomen die pas deze en volgende week tijdens OOP1 worden behandeld waaronder `HashMap`, een structuur die je in PHP kent als een associatief array.

Tijdens het tutoraat is beperkt ruimte om vragen over deze en andere types te stellen, gebruik hier de OOP1 practica voor.

6.2.1. Activiteiten

Afronden van activiteiten week 1 en verder met:

- boek 8.5 – 8.7: bewegingsrichtingen `up` en `down` toevoegen
- boek 8.8 – 8.10: rondkijken in de ruimte met `look` en alvast vooruitdenken
- eventueel start activiteiten week 3

6.2.2. Tutoraat

Onderwerpen:

- bespreking spelscenario en plattegrond
- implementatie van bewegingen (`up` en `down`) en handelingen (`look`)
- bespreken verplichte uitbreidingen: welke ga je kiezen?

6.3. Week 3

6 januari – 10 januari

Het toevoegen van items aan ruimtes, naar ruimtes terug kunnen keren, de klasse `Player` en verdere verbeteringen

6.3.1. Activiteiten

Afronden van activiteiten week 2 en verder met:

- boek 8.11 – 8.12: items toevoegen, `back` commando en de klasse `Player`
- boek 8.13: maak het spel taalonafhankelijk
- start werk aan uitbreidingen

6.3.2. Tutoraat

Onderwerpen:

- bespreken van spelhistorie (`back` commando)
- bespreken aanpak uitbreidingen

6.4. Week 4

13 januari – 17 januari

Uitbreidingen en voorbereiden van documentatie

6.4.1. Activiteiten

Afronden van activiteiten week 3 en verder met:

- uitbreidingen
- uitwerken en vormgeven van de Quick Start Guide
- opstellen van Technical Notes

6.4.2. Tutoraat

Onderwerpen:

- voortgang uitbreidingen
- bespreken aanpak en uitwerking van de Quick Start Guide
- bespreken invulling Technical Notes

6.5. Week 5

20 januari – 24 januari

Afronding en oplevering

6.5.1. Activiteiten

- afronding Quick Start Guide & Technical Notes
- afronding uitbreidingen

6.5.2. Tutoraat

Onderwerpen:

- bespreking voortgang afronding en oplevering

6.5.3. Deadline

Inleveren van de Quick Start Guide, Technical Notes en Git bundle vóór 24 januari 12.00. De projectdocent zal deze dan verspreiden onder de projectteams voor de reviews in week 6.

6.6. Week 6

27 januari – 31 januari

Team reviews en demo

6.6.1. Activiteiten

- review en beoordeling van *drie* teams

6.6.2. Tutoraat

Onderwerpen:

- demo spel

6.6.3. Deadline

Inleveren van team reviews uiterlijk 31 januari, 23.59.

A. Bijlage

A.1. Git bundles

In sommige gevallen is het niet mogelijk om met Git over een netwerk (HTTPS, SSH, etc.) te synchroniseren en in dit soort situaties kan je een Git repository delen door middel van een Git *bundle*, een archiefbestand.

Het werk gaat worden beoordeeld door medestudenten en in plaats van hen toegang te geven tot jouw project op GitHub zal je de code en ontwikkelhistorie gaan delen door middel van een bundle.

De werking is eenvoudig, in jouw project directory voer je het volgende commando uit om een bundle te creëeren van de `master` branch. Het resultaat is een bestand met de naam `zuul.bundle`.

```
git bundle create zuul.bundle HEAD master
```

Het bestand `zuul.bundle` kan je versturen of op een andere manier delen, de ontvanger kan jouw Git repository vervolgens lokaal met `git clone` importeren:

```
git clone zuul.bundle
```

Zie Git Bundling voor meer informatie en opties voor het exporteren en gebruiken van bundles.

A.2. Quick Start Guide

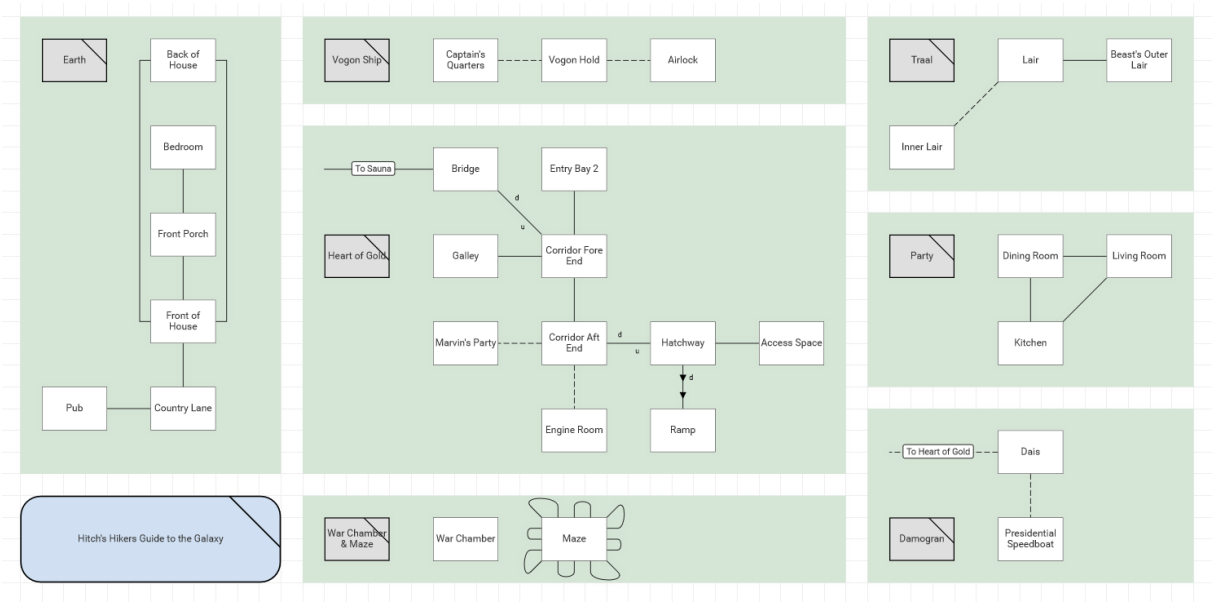
Voor een Quick Start Guide bestaat geen recept, je zal zelf de beste vorm voor jouw spel moeten bedenken. Een tekstuele beschrijving van het spel ligt voor de hand maar vaak zeggen afbeeldingen meer dan woorden. We noemen hier een tweetal (online) toepassingen die jou daarbij kunnen helpen, misschien weet je zelf andere hulpmiddelen?

A.2.1. Dungeon Generator

Figuur 2: Maze of the Dread Witch

Dungeons and Dragons is een rollenspel maar vooral ook een variant op en uitbreiding van adventure games als Zuul. Ruimtes en het zich kunnen verplaatsen tussen ruimtes is ook hier mogelijk en een gereedschap als een Dungeon Generator kan jou helpen een spelscenario te visualiseren, zie figuur 2 voor een voorbeeld.

A.2.2. Trizbort.io



Figuur 3: Hitchhiker's Guide to the Galaxy

Misschien ben je op zoek naar een meer schematische weergave van jouw spelscenario voor ruimtes, paden en handelingen? Trizbort.io is een tool waar je deze zou kunnen combineren en zelfs verder, zie bijvoorbeeld figuur 3 voor een schematische weergave van The Hitchhiker's Guide to the Galaxy.[4]

Trizbort.io kan schema's exporteren naar verschillende formaten, zou dit een manier kunnen zijn voor het laden van verschillende spelscenario's (configuratie) in jouw implementatie?

A.3. Technical Notes

Ook voor Technical Notes bestaat geen vast formaat, maar we geven je hier een voorbeeld hoe je dit zou kunnen aanpakken. Bedenk, de lezer is heel goed bekend met de context (zij of hij werkt immers

[4] The Hitchhiker's Guide to the Galaxy van Douglas Adams heeft sporen nagelaten in onder andere informatica: de kans is groot dat je in voorbeelden of uitwerkingen het getal 42 tegenkomt. Maar ook in het gewone leven wanneer je op 25 mei mensen met een handdoek ziet lopen …

aan hetzelfde probleem) en is vooral geïnteresseerd in details of bijzonderheden. Schrijf kort en bondig, in dit project mag het document niet langer dan *twee* pagina's zijn.

A.3.1. Indeling

Begin het document in met basisinformatie over de auteurs en de uitbreidingen die voor deze oplossing van belang zijn, bijvoorbeeld:

Project	Zuul
Titel	Deceitful Elfs and a Great Escape
Auteur	Auteur 1 / studentnummer <email_1@st.hanze.nl>
	Auteur 2 / studentnummer <email_2@st.hanze.nl>
Uitbreidingen	Trapdoor, Actors
Datum	20 januari 2020

Naast de basisinformatie neem je in ieder geval de volgende onderdelen op:

- **Samenvatting**

  Hier beschrijf je in het kort waar het in jouw implementatie om gaat. Schrijf dit als laatste nadat je de volgende onderdelen volledig hebt uitgewerkt.

- **Uitbreidingen**

  Hier beschrijf je per uitbreiding bijzonderheden met betrekking tot de implementatie. Met welke klassen heb je het spel moeten uitbreiden, wat is de rol van deze klassen en hoe heb je gezorgd voor cohesie en heb je afhankelijkheden weten te beperken?

- **Bijzonderheden**

  Zijn er situaties die je moet uitleggen die niet specifiek met uitbreidingen te maken hebben? Heb je misschien de basisstructuur die het boek jou heeft gegeven volledig veranderd? Is dit een oplossing die misschien in de volgende editie van het boek zou moeten worden opgenomen?

  Dit zou ook een plek kunnen zijn om uitleg te geven over de organisatie van jouw code als deze afwijkt van wat gebruikelijk is.

- **Voorbeeldgebruik**

  Noteer hier hoe de uitbreidingen eenvoudig kunnen worden gespeeld zodat de werking snel zichtbaar is.

Beschrijf hier ook een „shortest path to exit”, dat wil zeggen hoe een gebruiker het snelst het spel doorloopt en daarbij alle uitbreidingen tegenkomt (dit is dus niet de optie Quit uit een menu!).

Wat je hier beschrijft zou voor een gebruiker „cheats” zijn om het spel te helpen winnen. Voor de lezer van deze aantekeningen zijn ze echter bedoeld om snel en efficiënt jouw implementatie te kunnen doorlopen en beoordelen.