Arcade Game

Documentsoort:	Behoeftespecificatie
Versie:	2.0
Datum:	20 Mar 2015
Auteur:	Quinten Soetens
Status:	Afgeleverd

Samenvatting

Dit document bevat de specificaties voor de basis van een arcade game. Het is geschreven in het kader van het vak "Inleiding software Engineering" (1ste bachelor informatica - Universiteit Antwerpen).

Legende

Een typische use-case bevat de volgende onderdelen.

Refertenummer & titel:

Wordt gebruikt om naar een bepaalde use-case te verwijzen.

Prioriteit:

De specificatie van een systeem vraagt meer dan wat binnen de voorziene tijd op te leveren is. Vandaar dat we per use-case aangeven in hoeverre die functionaliteit belangrijk is. In volgorde van belangrijkheid kan hier staan: VERPLICHT (deze use-case moet opgeleverd worden), BELANGRIJK (niet essentieel maar bij voorkeur toch opleveren), NUTTIG (interessant maar kan weggelaten worden).

Doel:

Summiere beschrijving van het waarom van de use-case, t.t.z. wat de use-case bijdraagt tot de gehele functionaliteit.

Preconditie:

Summiere beschrijving van de uitgangspunten bij aanvang van de use-case.

Succesvol einde:

Summiere beschrijving van wat opgeleverd zal worden als er niks fout is gegaan.

Stappen:

Een sequentiële beschrijving van hoe de use-case precies zal verlopen als alles goed gaat (het zogenaamde "happy day scenario"). De stappen zijn genummerd en kunnen controle instructies (WHILE, IF, ...) bevatten.

Uitzonderingen:

Een lijst van mogelijke probleemgevallen en hoe die behandeld zullen worden. Een probleem geval (a) verwijst naar het nummer van de stap waar het probleem kan

optreden, (b) bevat een conditie die aangeeft wanneer het probleemgeval optreedt, (c) omschrijft heel kort (een lijn) hoe het probleem behandeld zal worden.

Voorbeeld:
Een voorbeeld van wat in- of uitgevoerd kan worden.

Soms is een use-case een uitbreiding van een andere use-case, en dan zijn volgende onderdelen relevant.

Uitbreiding:
Een referte naar de use-case waarvan deze een uitbreiding is.

Stappen:
Een lijst van extra en/of aangepaste stappen t.o.v de use-case waarvan deze een uitbreiding is.
Een uitbreinding (a) verwijst naar het nummer van de stap die uitgebreid wordt, (b) zegt of de uitbreiding voor, na of tijdens de normale stap zal gebeuren, (c) omschrijft wat precies in de uitbreiding zal gebeuren.

Behoeftes

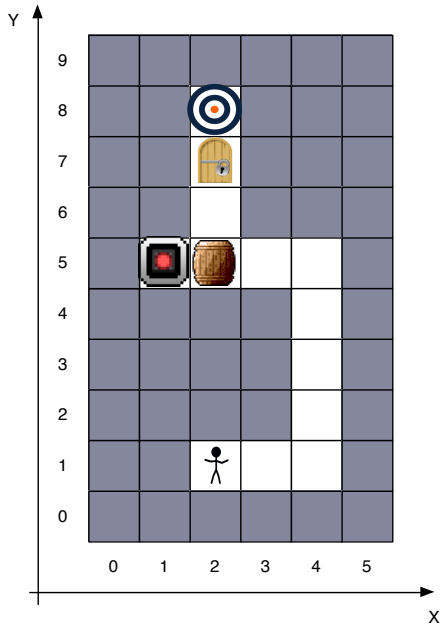
Hieronder volgt een opsomming van alle use-cases inclusief hun prioriteit.

Use-case	**Prioriteit**
1: Invoer	
1.1.b. Speelveld inlezen	VERPLICHT
1.2.b Acties inlezen	BELANGRIJK
1.1.c. Speelveld inlezen (met monsters)	BELANGRIJK
1.1.d. Speelveld inlezen (met Water)	BELANGRIJK
1.1.e. Speelveld inlezen (met Boobytraps)	NUTTIG
2: Uitvoer	
2.1.b. Huidig speelveld wegschrijven	VERPLICHT
2.2.b. Resterende acties wegschrijven	BELANGRIJK
2.1.c. Speelveld met monsters wegschrijven	BELANGRIJK
2.1.d. Speelveld met water wegschrijven	BELANGRIJK
2.1.e. Speelveld met boobytraps wegschrijven	NUTTIG
2.3. Simpele Grafische Impressie	VERPLICHT
2.3.c. Simpele Grafische Impressie (met monsters)	BELANGRIJK
2.3.d. Simpele Grafische Impressie (met water)	BELANGRIJK
2.3.e. Simpele Grafische Impressie (met boobytraps)	NUTTIG
2.4. HTML Uitvoer	NUTTIG
2.5. XML Uitvoer (Save Game)	NUTTIG
2.6. Integratie met Computer Graphics	NUTTIG

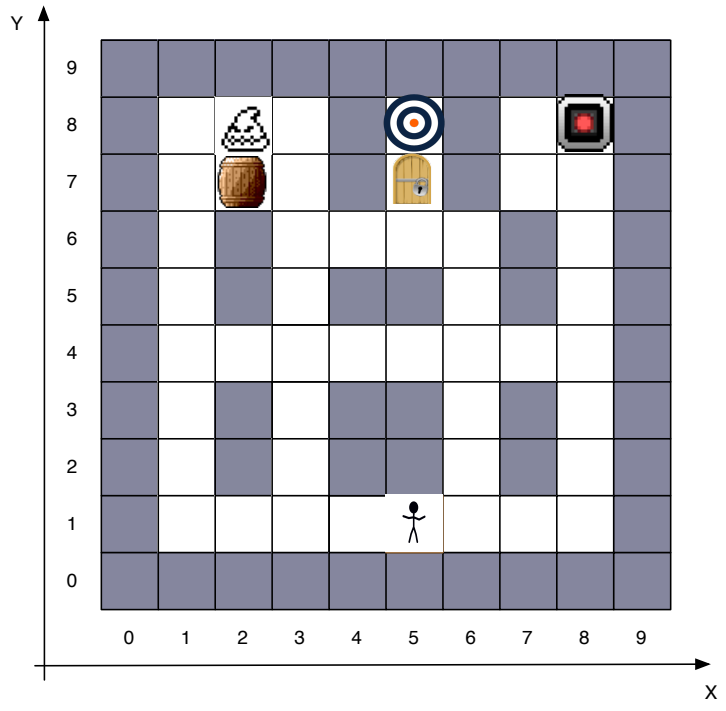
3: Simulatie	
3.1.b. Bewegen over het speelveld	VERPLICHT
3.1.c: Monsters bewegen over het speelveld	BELANGRIJK
3.3.b. Automatisch uitvoeren van scenario	BELANGRIJK
3.4. Aanval uitvoeren	BELANGRIJK
3.5. Druknopen openen weg naar poorten	VERPLICHT
3.6. Speler wint door doel te bereiken	VERPLICHT
3.7. Monsters doden speler	BELANGRIJK
3.8. Entiteit verdrinkt in water	BELANGRIJK
3.9. Entiteit bouwt brug over water	BELANGRIJK
3.10 Entiteit vernietigt Boobytrap / Boobytrap vernietigt Entiteit	NUTTIG
4: Gebruikersinterface	
4.1. Simpele UI voor spel	VERPLICHT
4.2. GUI voor spel	NUTTIG

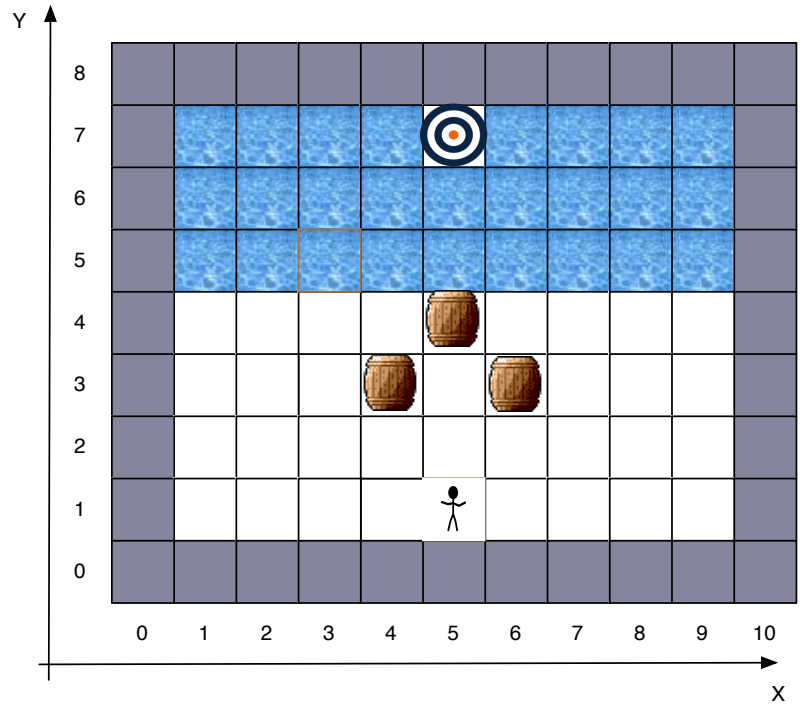
1. Invoer

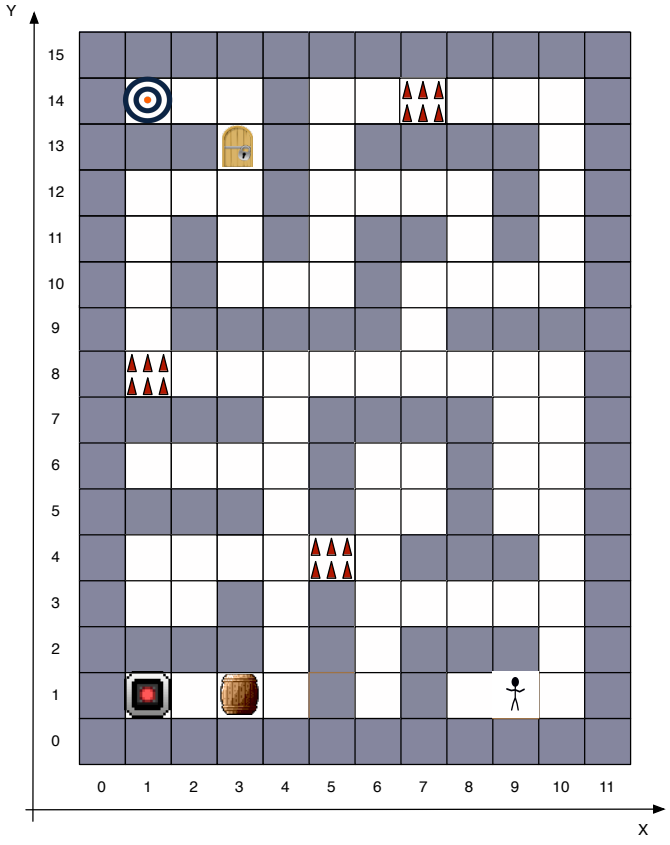
1.1.b. Speelveld inlezen	
Prioriteit	VERPLICHT
Doel	De begin-situatie van het systeem bestaat uit een speelveld waarop *meerdere obstakels* gepositioneerd zijn. Informatie over de vorm van het speelveld en de posities van de obstakels en de speler is beschikbaar in een XML-gestructureerde file. Doel van deze usecase is het inlezen van de XML-file en controleren of de beginsituatie fysisch haalbaar is. LET OP: de beschrijving van obstakels is nu veranderd. Het tag OBSTAKEL met een subtag TYPE is nu vervangen door typespecifieke Tags (MUUR, TON, …)
Preconditie	Een XML bestand met daarop een beschrijving van het slagveld. (Zie Appendix 1 voor meer informatie over het “aangepaste" XML formaat)
Succesvol einde	Het systeem bevat de beginsituatie van het slagveld.
Stappen	1. Open het invoerbestand 2. Parse het bestand met TinyXML. 3. WHILE Er zijn nog elementen in het geparste document 3.1. Herken het soort element (VELD, SPELER, DOEL, POORT, KNOP, MUUR, TON) 3.2. Lees de verder informatie voor dit element 3.3. IF Verifieer de geldigheid van het element 3.3.1. THEN voeg element toe aan de speelveld 4. Verifieer de consistentie van de speelveld
Uitzonderingen	3.1. [Onherkenbaar element] **Foutboodschap** + positioneer op volgende element in het bestand => verdergaan vanaf stap 3 3.3. [Ongeldige informatie] **Foutboodschap** + positioneer op volgende element in het bestand => verdergaan vanaf stap 3 4. [Inconsistent speelveld] **Foutboodschap**

1.1.b. Speelveld inlezen	
Voorbeeld	Een speelveld (zoals op onderstaande prent) staat beschreven in het bestand Level1.xml. 

1.2.b Acties inlezen	
Prioriteit	BELANGRIJK
Doel	Lees de verschillende acties in voor de speler. (Bewegingen en Aanvallen). Deze use case is enkel noodzakelijk indien ook de use cases ivm monsters worden behandeld (use cases 1.1.c.; 2.1.c; 2.3.c; 3.6.; 3.7.)
Preconditie	Een XML bestand met de beschrijvingen van de acties.
Succesvol einde	Het systeem bevat de acties die sequentieel in de volgorde van voorkomen in het bestand uitgevoerd kunnen worden.
Stappen	1. Open het invoerbestand met de bewegingen 2. Parse het bestand (met TinyXML) 3. WHILE Er zijn nog elementen in het geparste document 3.1. Herken het soort element (één van ACTIE of BEWEGING) 3.2. Lees de verder informatie voor dit element 3.3. IF Verifieer de geldigheid van het element 3.3.1. THEN voeg beweging toe aan de overige bewegingen
Uitzonderingen	3.1. [Onherkenbaar element] **Foutboodschap** + positioneer op volgende element in het bestand => verdergaan vanaf stap 3 3.3. [Ongeldige informatie] **Foutboodschap** + positioneer op volgende element in het bestand => verdergaan vanaf stap 3
Voorbeeld	Zie het bestand Level3Acties.xml

1.1.c. Speelveld inlezen (met monsters)	
Prioriteit	BELANGRIJK
Uitbreiding	Deze use case is een uitbreiding op use case 1.1.b
Doel	Het invoerbestand van het slagveld bevat nu ook informatie over monsters die op het veld staan.
Preconditie	Een XML bestand met daarop een beschrijving van het speelveld (inclusief de monsters). (Zie Appendix 1 voor meer informatie over het XML formaat)
Succesvol einde	Het systeem bevat de beginsituatie van het speelveld.
Stappen	Idem aan 1.1.b. 3.1. Herken het soort element (VELD, SPELER, DOEL, POORT, KNOP, MUUR, TON, **MONSTER**)
Uitzonderingen	Idem aan 1.1.b.
Voorbeeld	Een speelveld met monsters (zoals op onderstaande prent) staat beschreven in het bestand Level3.xml. 

1.1.d. Speelveld inlezen (met water)	
Prioriteit	BELANGRIJK
Uitbreiding	Deze use case is een uitbreiding op use case 1.1.b
Doel	Het invoerbestand van het slagveld bevat nu ook informatie over water dat op het veld aanwezig is.
Preconditie	Een XML bestand met daarop een beschrijving van het speelveld (inclusief het water). (Zie Appendix 1 voor meer informatie over het XML formaat)
Succesvol einde	Het systeem bevat de beginsituatie van het speelveld.
Stappen	Idem aan 1.1.b. 3.1. Herken het soort element (VELD, SPELER, DOEL, POORT, KNOP, MUUR, TON, **WATER**)
Uitzonderingen	Idem aan 1.1.b.
Voorbeeld	Een speelveld met water (zoals op onderstaande prent) staat beschreven in het bestand Level2.xml. 

1.1.e. Speelveld inlezen (met boobytraps)	
Prioriteit	BELANGRIJK
Uitbreiding	Deze use case is een uitbreiding op use case 1.1.b
Doel	Het invoerbestand van het slagveld bevat nu ook informatie over valstrikken die op het veld aanwezig zijn.
Preconditie	Een XML bestand met daarop een beschrijving van het speelveld (inclusief de valstrikken). (Zie Appendix 1 voor meer informatie over het XML formaat)
Succesvol einde	Het systeem bevat de beginsituatie van het speelveld.
Stappen	Idem aan 1.1.b. 3.1. Herken het soort element (VELD, SPELER, DOEL, POORT, KNOP, MUUR, TON, **VALSTRIK**)
Uitzonderingen	Idem aan 1.1.b.
Voorbeeld	Een speelveld twee valstrikken (zoals op onderstaande prent) staat beschreven in het bestand Level4.xml. 

2. Uitvoer

2.1.b Huidig speelveld wegschrijven	
Prioriteit	VERPLICHT
Uitbreiding	Deze Use case is een uitbreiding op use case 2.1.
Doel	Volledige speelveld zoals hij op moment van uitvoeren is, wegschrijven naar een bestand. Dit bestand beschrijft dus a) hoe het speelveld er uit ziet, b) waar de speler en alle andere entiteiten zich bevinden.
Preconditie	Het systeem is correct geinitialiseerd Een speelveld werd ingelezen.
Succesvol einde	Het systeem heeft een tekstbestand (ASCII) uitgevoerd, waarin de informatie over het speelveld netjes is uitgeschreven.
Stappen	1. Creëer uitvoerbestand 2. Open uitvoerbestand 3. Schrijf gegevens uit voor het speelveld. 4. FORALL Entiteiten 4.1. Schrijf gegevens uit voor de entiteit 5. Sluit uitvoerbestand
Uitzonderingen	Zie Use case 2.1.

Voorbeeld	gegeven de input uit het voorbeeld van use case 1.1b, krijgen we volgende eenvoudige output:
	
	`Bestand: HuidigSpeelveld.txt`
	
	`Het huidige speelveld is Level 1:`
	`Eigenschappen van dit veld:`
	`-Breedte 6`
	`-Lengte 10`
	
	`Speler Chip bevindt zich in dit speelveld op positie (2,1).`
	`Er bevindt zich een doel op positie (2,8).`
	`Er bevindt zich een poort (met id gate1) op positie (2,7).`
	`Er bevindt zich een knop (gelinkt aan poort gate1) op positie (1,5).`
	`Er bevindt zich een ton op positie (2,5).`

2.2.b. Resterende acties wegschrijven	
Prioriteit	BELANGRIJK
Uitbreiding	Deze Use case is een uitbreiding op use case 2.2.
Doel	Alle nog niet gemaakte acties wegschrijven naar een bestand.
Preconditie	Het systeem is correct geinitialiseerd Een speelveld werd ingelezen.
Succesvol einde	Het systeem heeft een tekstbestand (ASCII) uitgevoerd, waarin de informatie over de resterende acties netjes is uitgeschreven.
Stappen	1. Creëer uitvoerbestand 2. Open uitvoerbestand 3. WHILE nog te maken acties aanwezig 3.1. Schrijf gegevens uit voor de actie (beweging of aanval) 4. Sluit uitvoerbestand
Uitzonderingen	Zie Use case 2.2.

Voorbeeld	gegeven de input uit het voorbeeld van use case 1.2b, krijgen we volgende eenvoudige output:
	
	`Bestand: ResterendeActies.txt`
	
	`Monster monster1 zal volgende beweging nog uitvoeren:`
	`Rechts`
	`Speler Chip zal volgende beweging nog uitvoeren:`
	`Links`
	`[…]`
	`Speler Chip zal volgende aanval nog uitvoeren:`
	`Omhoog`
	`Speler Chip zal volgende beweging nog uitvoeren:`
	`Omhoog`
	`[…]`
	`Speler Chip zal volgende beweging nog uitvoeren:`
	`Omhoog`

2.1.c Speelveld met monsters wegschrijven	
Prioriteit	BELANGRIJK
Uitbreiding	Deze Use case is een uitbreiding op use case 2.1.b.
Doel	Volledige speelveld zoals hij op moment van uitvoeren is, wegschrijven naar een bestand. Dit bestand beschrijft dus a) hoe het speelveld er uit ziet, b) waar de speler en alle andere entiteiten zich bevinden. **De uitbreiding ligt in het feit dat monsters ook hun informatie moeten weergeven.**

Voorbeeld	gegeven de input uit het voorbeeld van use case 1.1c, krijgen we volgende eenvoudige output:
	
	`Bestand: HuidigSpeelveld.txt`
	
	`Het huidige speelveld is Level 3 (With a monster):`
	`Eigenschappen van dit veld:`
	`-Breedte 10`
	`-Lengte 10`
	
	`Speler Chip bevindt zich in dit speelveld op positie (5,1).`
	`Er bevindt zich een monster op positie (2,8).`
	`Er bevindt zich een doel op positie (5,8).`
	`Er bevindt zich een knop (gelinkt aan poort gate1) op positie (8,8).`
	`Er bevindt zich een poort (met id gate1) op positie (5,7).`

2.1.d Speelveld met water wegschrijven	
Prioriteit	BELANGRIJK
Uitbreiding	Deze Use case is een uitbreiding op use case 2.1.b.
Doel	Volledige speelveld zoals hij op moment van uitvoeren is, wegschrijven naar een bestand. Dit bestand beschrijft dus a) hoe het speelveld er uit ziet, b) waar de speler en alle andere entiteiten zich bevinden. **De uitbreiding ligt in het feit dat de output ook moet weergeven waar er water is.**

Voorbeeld	gegeven de input uit het voorbeeld van use case 1.1c, krijgen we volgende eenvoudige output:
	
	`Bestand: HuidigSpeelveld.txt`
	
	`Het huidige speelveld is Level 2 (with water):`
	`Eigenschappen van dit veld:`
	`-Breedte 11`
	`-Lengte 9`
	
	`Speler Chip bevindt zich in dit speelveld op positie (5,1).`
	
	`Er bevindt zich water op positie (1,7)`
	`Er bevindt zich water op positie (2,7)`
	`Er bevindt zich water op positie (3,7)`
	`Er bevindt zich water op positie (4,7)`
	`Er bevindt zich een doel op positie (5,7).`
	`Er bevindt zich water op positie (6,7)`
	`[…]`
	`Er bevindt zich water op positie (9,5)`
	`Er bevindt zich een ton op positie (5,4).`
	`Er bevindt zich een ton op positie (4,3).`
	`Er bevindt zich een ton op positie (6,3).`

2.1.e Speelveld met boobytraps wegschrijven	
Prioriteit	BELANGRIJK
Uitbreiding	Deze Use case is een uitbreiding op use case 2.1.b.
Doel	Volledige speelveld zoals hij op moment van uitvoeren is, wegschrijven naar een bestand. Dit bestand beschrijft dus a) hoe het speelveld er uit ziet, b) waar de speler en alle andere entiteiten zich bevinden. **De uitbreiding ligt in het feit dat boobytraps ook hun informatie moeten weergeven.**

Voorbeeld	gegeven de input uit het voorbeeld van use case 1.1e, krijgen we volgende eenvoudige output:
	
	`Bestand: HuidigSpeelveld.txt`
	
	`Het huidige speelveld is Level 4 (Maze with boobytraps):`
	`Eigenschappen van dit veld:`
	`-Breedte 11`
	`-Lengte 16`
	
	`Speler Chip bevindt zich in dit speelveld op positie (9,1).`
	`Er bevindt zich een doel op positie (1,14).`
	`Er bevindt zich een valstrik op positie (7,14).`
	`Er bevindt zich een poort (met id gate1) op positie (3,13).`
	`Er bevindt zich een valstrik op positie (1,8).`
	`Er bevindt zich een valstrik op positie (5,4).`
	`Er bevindt zich een knop (gelinkt aan poort gate1) op positie (1,1).`
	`Er bevindt zich een ton op positie (3,1).`

2.3. Simpele Graphische Impressie	
Prioriteit	VERPLICHT
Uitbreiding	Deze Use case is een uitbreiding op use case 2.1.(b.)
Doel	Het speelvled wordt graphisch weergegeven aan de hand van ASCII symbolen.
Preconditie	Het systeem bevat een geldig speelveld
Succesvol einde	Het systeem heeft een tekstbestand (ASCII) uitgevoerd, waarin de situatie van het speelveld graphisch staat beschreven.
Stappen	1. Creëer uitvoerbestand 2. Open uitvoerbestand 3. Teken de gegevens uit voor de toestand van het speelveld 4. Sluit het uitvoerbestand
Uitzonderingen	1. Creatie mislukt: **Foutboodschap** & Schrijf uit naar console ipv. bestand.

Voorbeeld	`#: muur`
	`Y: speler`
	`X: doel`
	`O: ton`
	`=: gesloten poort`
	`[spatie]: leeg / open poort`
	
	De startsituatie van het speelveld uit Level1.xml ziet er dan uit als volgt:
	
	`######`
	`##X###`
	`##=###`
	`#.O###`
	`#### #`
	`#### #`
	`#### #`
	`##Y  #`
	`######`

2.3.c Simpele Graphische Impressie (met monsters)	
Prioriteit	BELANGRIJK
Uitbreiding	Deze Use case is een uitbreiding op use case 2.3.
Doel	Het speelveld wordt graphisch weergegeven aan de hand van ASCII symbolen. Maar nu is er ook een symbool voorzien voor monsters.

Voorbeeld	`#: muur`
	`Y: speler`
	`X: doel`
	`O: ton`
	`@: Monster`
	`=: gesloten poort`
	`[spatie]: leeg / open poort`
	
	De startsituatie van het speelveld uit Level3.xml ziet er dan uit als volgt:
	
	`##########`
	`# @ #X# .#`
	`# O #=#  #`
	`# #    # #`
	`# # ## # #`
	`#        #`
	`# # ## # #`
	`# # ## # #`
	`#    Y   #`
	`##########`

2.3.d Simpele Graphische Impressie (met water)	
Prioriteit	BELANGRIJK
Uitbreiding	Deze Use case is een uitbreiding op use case 2.3.
Doel	Het speelveld wordt graphisch weergegeven aan de hand van ASCII symbolen. Maar nu is er ook een symbool voorzien voor water. Indien er een brug wordt gemaakt met een ton, dan wordt deze ook als spatie weergegeven.

Voorbeeld	`#: muur`
	`Y: speler`
	`X: doel`
	`O: ton`
	`~: Water`
	`=: gesloten poort`
	`[spatie]: leeg / open poort / brug`
	
	De startsituatie van het speelveld uit Level3.xml ziet er dan uit als volgt:
	
	`##########`
	`#~~~~X~~~~#`
	`#~~~~~~~~~#`
	`#~~~~~~~~~#`
	`#    O    #`
	`#   O O   #`
	`#         #`
	`#    Y    #`
	`##########`

2.3.e Simpele Graphische Impressie (met boobytraps)	
Prioriteit	NUTTIG
Uitbreiding	Deze Use case is een uitbreiding op use case 2.3.
Doel	Het speelveld wordt graphisch weergegeven aan de hand van ASCII symbolen. **Echter voor boobytraps wordt geen visualisatie voorzien (deze moeten voor de speler onzichtbaar zijn!)**

Voorbeeld	`#: muur`
	`Y: speler`
	`X: doel`
	`O: ton`
	`=: gesloten poort`
	`[spatie]: leeg / open poort`
	
	De startsituatie van het speelveld uit Level4.xml ziet er dan uit als volgt:
	
	`############`
	`#X  #      #`
	`###=# #### #`
	`#   #    # #`
	`# # # ## # #`
	`# #   #    #`
	`# ##### ####`
	`#           #`
	`#### ####  #`
	`#    #  #  #`
	`#### #  #  #`
	`#      ### #`
	`#  # #     #`
	`#### # ### #`
	`#. O # # Y #`
	`############`

2.4. HTML Uitvoer	
Prioriteit	NUTTIG
Uitbreiding	Deze Use case is een uitbreiding op use case 2.2.(a.)
Doel	Het speelveld wordt nu uitgeprint als een html bestand dat kan geopend worden in een internet browser. De toestand van het speelveld wordt daarin grafisch weergegeven.
Preconditie	Het systeem bevat een speelveld
Succesvol einde	Er is een HTML bestand uitgevoerd dat kan geopend worden in een browser en waarin een grafische impressie is getoont van de toestand van het speelveld.

2.5. XML Uitvoer	
Prioriteit	NUTTIG
Uitbreiding	Deze Use case is een uitbreiding op use case 2.2.(a.)
Doel	Het speelveld wordt nu uitgeprint als een xml bestand gebruik makende van de XML tags die ook gebruikt worden voor het inlezen van het systeem. Dit is als het ware een “save game” functionaliteit, het inlezen van een speelveld wordt dan de “load game” functionaliteit.
Preconditie	Het systeem bevat een speelveld
Succesvol einde	Er is een XML bestand uitgevoerd.

2.6 Integratie met Computer Graphics	
Prioriteit	NUTTIG
Doel	Een 3D visualisatie van de simulatie. Hiervoor kan je je graphics engine gebruiken. Maak hiervoor gebruik van je 3D engine uit het vak Computer Graphics.
Preconditie	Het systeem bevat een speelveld
Succesvol einde	Het volledige speelveld inclusief uitvoeren van de acties wordt weergegeven in een 3D omgeving.

3. Simulatie

3.1.b: Bewegen over het speelveld	
Prioriteit	VERPLICHT
Uitbreiding	Deze Use case is een uitbreiding op use case 3.1.
Doel	Het uitvoeren van een beweging door de speler. De uitbreiding ligt in de interactie met: knop, poort en doel
Preconditie	Het systeem is correct geinitialiseerd. Een speelveld werd ingelezen. Een beweging werd ingelezen.
Succesvol einde	De speler waarvoor de beweging werd gedefineerd bevindt zich op de juiste eindpositie.
Uitzonderingen	1.1.1,1.2.1, 1.3.1, 1.4.1 [**Speler komt op doel**]: zie Use Case 3.5 1.1.1,1.2.1, 1.3.1, 1.4.1 [**Speler raakt poort**]: zie Use Case 3.4 1.1.1,1.2.1, 1.3.1, 1.4.1 [**Speler komt op knop**]: zie Use Case 3.4

3.1.c: Monsters bewegen over het speelveld	
Prioriteit	BELANGRIJK
Uitbreiding	Deze Use case is een uitbreiding op use case 3.1.
Doel	Het uitvoeren van een beweging door **monsters**.
Preconditie	Het systeem is correct geinitialiseerd. Een speelveld werd ingelezen. Een beweging van een monster werd ingelezen.
Succesvol einde	Het monster waarvoor de beweging werd gedefineerd bevindt zich op de juiste eindpositie.
Uitzonderingen	LET OP: Monsters kunnen evengoed interageren met beweegbare obstakels, knopen en water. maw. - Zolang een monster op een knop staat, is de gelinkte poort geopend. **(zie use case 3.5)** - Een monster kan een ton verplaatsen **(uitbreiding van use case 3.2)** - Een monster verdrinkt in water **(zie use case 3.8)** - Een monster wordt gedood door een boobytrap waardoor ook de boobytrap verdwijnt. **(zie use case 3.10)**

3.3.b. Automatisch uitvoeren van scenario	
Prioriteit	BELANGRIJK
Doel	Een volledig scenario van meerdere acties (bewegingen/aanvallen) wordt uitgevoerd.
Uitbreiding	Deze Use case is een uitbreiding op use case 3.3 **door nu ook automatisch aanvallen uit te voeren.**
Preconditie	Het systeem is correct geinitaliseerd. Een slagveld werd ingelezen. Acties werden ingelezen.
Succesvol einde	Er zijn geen onafgewerkte acties. Alle entiteiten bevinden zich in de juiste eindpositie.
Stappen	1. WHILE nog niet afgewerkte acties 1.1. IF actie is beweging 1.1.2 beweeg speler/monster volgens beweging (zie use-case 3.1) 1.2. ELSE IF actie is aanval 1.2.2 voer aanval uit (zie use case 3.3)

3.4. Aanvallen uitvoeren	
Prioriteit	VERPLICHT
Doel	Een speler kan een gegeven aanval uitvoeren.
Preconditie	Het systeem is correct geinitaliseerd. Een speelveld werd ingelezen. Acties werden ingelezen. De eerstvolgende actie is een aanval van de speler.
Succesvol einde	De speler blijft op dezelfde positie staan en heeft een aanval in een gegeven richting uitgevoerd. **Indien op de positie naast de speler in de richting van de aanval een monster stond, is het monster gedood.**

3.5. Druknopen openen weg naar poorten	
Prioriteit	VERPLICHT
Doel	Zolang er een entiteit op een druknop staat, is de poort die aan deze knop gelinkt is geopend.
Uitbreiding	Deze Use case is een uitbreiding op use case 3.1.b en 3.1.c (en 3.2)
Succesvol einde	Zolang er een entiteit (**speler, monster of ton**) op een druknop staat, is de poort die aan deze knop gelinkt is geopend.
Uitzonderingen	Geen

3.6. Speler wint door doel te bereiken	
Prioriteit	VERPLICHT
Doel	Als de speler op een doel komt is het spel over en heeft de speler gewonnen.
Uitbreiding	Deze Use case is een uitbreiding op use case 3.1.b.
Succesvol einde	Het spel is afgelopen.
Uitzonderingen	Geen

3.7. Monsters doden speler	
Prioriteit	BELANGRIJK
Doel	Indien een monster op een locatie komt waar er reeds een speler staat, wordt de speler gedood en is het GAME OVER.
Uitbreiding	Deze Use case is een uitbreiding op use case 3.1.c.
Succesvol einde	De speler werd gedood en uit het spel verwijdert.
Uitzonderingen	Geen

3.8. Entiteit verdrinkt in water	
Prioriteit	BELANGRIJK
Doel	Indien een speler of monster beweegt naar een locatie waar er zich water bevindt, wordt de speler of het monster gedood. Indien de speler stierf, is het game over.
Uitbreiding	Deze Use case is een uitbreiding op use cases 3.1.b en 3.1.c
Succesvol einde	De entiteit die verdonk is uit het spel verwijdert. Indien de speler stierf werd het spel prober afgesloten.
Uitzonderingen	Geen

3.9. Entiteit bouwt brug over water	
Prioriteit	BELANGRIJK
Doel	Als een ton (of een ander beweegbaar obstakel) verplaatst wordt naar een locatie waar er zich water bevindt, wordt het water vervangen door een brug.
Uitbreiding	Deze Use case is een uitbreiding op use case 3.2
Succesvol einde	De ton is uit het systeem verwijdert. De locatie waar voorheen water was, is nu leeg.
Uitzonderingen	Geen

3.10. Boobytrap doodt Entiteit / Entiteit vernietigt Boobytrap	
Prioriteit	NUTTIG
Doel	Indien er een entiteit (speler, monster of ton) op een boobytrap komt, dan wordt (i) die entiteit gedood of vernietigd en (ii) de boobytrap verwdijnt. Indien de entiteit de speler was is het GAME OVER.
Uitbreiding	Deze Use case is een uitbreiding op use case 3.1.b en 3.1.c (en 3.2)
Succesvol einde	De entiteit die op de boobytrap kwam is uit het systeem verwijdert, de boobytrap is uit het spel verwijdert. Indien de speler stierf werd het spel prober afgesloten.
Uitzonderingen	GEEN

4. Gebruikersinterface

4.1. Simpele UI voor simulatie	
Prioriteit	VERPLICHT
Doel	De gebruiker wil een user interface waarmee hij interactief met de simulatie kan werken.
Preconditie	Er is een correct ingelezen speelveld in het systeem aanwezig.
Succesvol einde	Er zijn functies in de UI aanwezig die het uitvoeren van acties/ stappen in de simulatie toelaten: • Een invoerbestand dat een speelveld beschrijft laten inlezen. • Een invoerbestand dat acties beschrijft laten inlezen. • Eén of meerdere stappen van de simulatie laten lopen. • De simpele uitvoer laten uitvoeren. • De simpele graphische impressie tonen.
Uitbreidingen	Mogelijke uitbreidingen zijn dan nog: • Manueel een beweging invoeren en uitvoeren • Manueel een aanval invoeren en uitvoeren

4.2. GUI voor simulatie	
Prioriteit	NUTTIG
Doel	Er zijn functies in de GUI aanwezig die het uitvoeren van acties/ stappen in de simulatie toelaten: • Een invoerbestand dat een speelveld beschrijft laten inlezen. • Een invoerbestand dat acties beschrijft laten inlezen. • Eén of meerdere stappen van de simulatie laten lopen. • De simpele uitvoer laten uitvoeren. • De simpele graphische impressie tonen. Gebruik hiervoor een willekeurige GUI library voor C++.
Uitbreidingen	Mogelijke uitbreidingen zijn dan nog: • Manueel een beweging invoeren en uitvoeren • Manueel een aanval invoeren en uitvoeren

Appendix 1
Geldige informatie

We moeten nu nog de tags en hun waarde vastleggen die gelden voor ons probleem-domein. Dit bestaat uit het vastleggen van de mogelijke tags, Attributen en de verwachte inhoud (content).
De mogelijke tag-identifiers zijn:
veld, naam, lengte, breedte, speler, obstakel, type, bewegingen, beweging, spelernaam, richting
Coordinaten in het speelveld worden weergegeven met attribuut-identifiers X en Y.

Tags	Subtags	Attributen	Content
Veld	Naam		String
	Lengte		Integer
	Breedte		Integer
	Speler		Zie tag definitie hieronder
	Muur	Beweegbaar	boolean
		X	Integer
		Y	Integer
	Ton	Beweegbaar	boolean
		X	Integer
		Y	Integer
	Doel	X	Integer
		Y	Integer
	Knop	X	Integer
		Y	Integer
		id	String
	Poort		Zie tag definitie hieronder

	Water	Beweegbaar	boolean
		X	Integer
		Y	Integer
	Monster		Zie tag definitie hieronder
	Valstrik	X	Integer
		Y	Integer
Speler		X	Integer
		Y	Integer
	Naam		String
Monster		X	Integer
		Y	Integer
	ID		String
Poort		X	Integer
		Y	Integer
		Beweegbaar	boolean
	ID		String
Bewegingen	Beweging		Zie tag definitie hieronder
Acties	Beweging		Zie tag definitie hieronder
	Aanval		Zie tag definitie hieronder

Beweging	**Spelernaam** OF **ID** (indien ook met monsters wordt gewerkt)		String
	Richting		String
Aanval	ID		String
	Richting		String

(In)consistent speelveld

Het bestand met het in te lezen speelveld wordt met de hand geschreven. Om het ingelezen speelveld te kunnen simuleren moet de informatie consistent zijn.
Een speelveld is consistent als:

- Een entiteit (speler/muur/ton/water/doel/poort/knop/monster/valstrik) binnen de grenzen van het veld ligt.
- Een speelveld geen negatieve dimensies (lengte, breedte) heeft.
- Een entiteit (speler/muur/ton/water/doel/poort/knop/monster/valstrik) geen negatieve coordinaten (x,y) heeft.
- Een speler een naam heeft.
- Een monster een identifier heeft.
- Een poort een identifier heeft.
- Een knop met een bestaande poort wordt gelinkt.
- Er maar maximaal 1 speler/ton/monster per coordinatenpaar op het speelveld staat. Op dat coordinatenpaar mag dan weer wél een knop/poort/doel/water/valstrik staan.

Een beweging of aanval is consistent als:

- De richting “omhoog”, “links”, “omlaag” of “rechts is.
- De ID of spelernaam een bestaande entiteit op het veld is.