|  |
| --- |
| ­­­­­­­­­Inception rapport |
| Projectopdracht voor het I2 blok: Sokoban |
| Blok I2 Projectgroep – Peter-Pim Baken |

30-4-2012

Inhoudsopgave

[1. Inleiding 1](#_Toc346367067)

[2. Omschrijving en doelstelling project 1](#_Toc346367068)

[1. Opdrachtomschrijving 1](#_Toc346367069)

[2. Probleembeschrijving 2](#_Toc346367070)

[3. Doelstelling van de opdracht 2](#_Toc346367071)

[3. Projectorganisatie 2](#_Toc346367072)

[1. Randvoorwaarden 2](#_Toc346367073)

[4. Benodigde mensen en middelen 2](#_Toc346367074)

[1. De relevante mensen zijn 2](#_Toc346367075)

[2. De relevante spullen zijn 2](#_Toc346367076)

[5. Functional and non-functional requirements 3](#_Toc346367077)

[1. Functional requirements 3](#_Toc346367078)

[2. Non-functional requirements: 4](#_Toc346367079)

[6. Use Cases 4](#_Toc346367080)

[1. Visual Paradigm diagram 4](#_Toc346367081)

[2. Use case descriptions 5](#_Toc346367082)

[7. Planning 1. Globale Planning 8](#_Toc346367083)

# Inleiding

Dit rapport gaat over de manier waarop we de opdracht voor het maken van een Sokoban spel aanpakken. Er komen verschillende onderwerpen aan de orde

# Omschrijving en doelstelling project

## Opdrachtomschrijving

De applicatie die moet worden gemaakt is een puzzelspel genaamd Happer. Happer is o.a. ontworpen in 1982 door Hiroyuki Imabayashi en is toen uitgegeven door Thinking Rabbit. Het is de bedoeling dat de speler dozen door een doolhof verschuift en het monster(zijn tegenstander) insluit met deze dozen. Tegelijkertijd mag de speler niet worden geraakt door het monster. De speler moet zo min mogelijk stappen doen, om een hogere score te behalen.

## Probleembeschrijving

Om het programma te kunnen schrijven moet ik de taal Java redelijk goed begrijpen en kennen. Ik heb de applicatie al een keer eerder ontworpen, maar toen een iets andere versie. Er zijn nog wel wat veranderingen aan te brengen waaronder de Dijkstra methode en het monster. Voor de rest is er vrij weinig verandering vantoepassing.

## Doelstelling van de opdracht

Het doel van de opdracht is om een spel, Happer, te ontwikkelen. Het doel van het spel is om mensen van verschillende leeftijden te entertainen.

# Projectorganisatie

## Randvoorwaarden

Bij het bouwen van de applicatie zijn er verschillende randvoorwaarden waaraan moet worden voldaan.

* Happer moet worden ontwikkeld in Netbeans, een IDE waarin met Java wordt geprogrammeerd.
* Het testen van de applicatie kan worden gedaan met JUnit in geval, dat dit nodig is. De handeling van de applicatie zijn vrij gemakkelijk, dus wellicht kan de gebruiker deze zelf testen m.b.v. opgestelde fysieke testgevallen.
* De UML diagrammen worden gemaakt in Visual Paradigm .
* Het moet mogelijk zijn om achteraf nog uitbreidingen te maken aan de applicatie, het moet dus object georiënteerd zijn, zodat het gemakkelijk is het programma te wijzigen en/of dingen toe te voegen.
* Het moet een standalone applicatie zijn, dus het hoeft niet samen te kunnen werken met andere systemen.

# Benodigde mensen en middelen

1. De relevante mensen zijn  
     
    \* Peter-Pim Baken  
    \* De tutor/expert, Vincent Broeren
2. De relevante spullen zijn  
     
    Computers met de benodigde software waaronder:

* SVN voor filesharing
* Visual Paradigm
* Netbeans IDE

# Functional and non-functional requirements

# 1. Functional requirements

Hieronder is een lijst weergegeven met de requirements waaraan de applicatie ‘Happer’ moet voldoen.

1. Er moet een score aanwezig zijn die de speler ten alle tijden kan zien.
2. De score wordt bijgehouden op basis van het aantal stappen dat het karakter heeft gedaan(de score wordt verhoogt per stap).
3. Door middel van de pijltjestoetsen moet de speler kunnen bewegen.
4. Er moet een speelveld zijn, groot genoeg om het speelbaar te houden en dat is afhankelijk van de andere spelverhoudingen.
5. Het speelveld moet Positions hebben, waarop de spelobjecten kunnen worden geplaatst.
6. Het speelveld word gemaakt op basis van een String die wordt ingeladen. Afhankelijk van de String worden er op de Positions spelobjecten geplaatst.
7. Het spel wordt gestart wanner de speler de applicatie opstart, een level opnieuw begint, het spel herstart.
8. Zodra het spel wordt gestart, moet er een speelveld worden aangemaakt, dat direct word gevuld met spelobjecten.
9. Zodra het spel wordt gestart, word de score op nul gezet.
10. Zodra het spel wordt gestart, begint het monster de bewegen.
11. Er moet een object “Wall” zijn, waar de speler niet doorheen kan bewegen .
12. Er moet een object “Player” zijn, die door de speler word bestuurt.
13. Alle objecten moeten dezelfde grootte hebben, ze nemen dus allemaal een gelijke hoeveelheid Positions in van het speelveld.
14. Er moet een object “Monster” zijn.
15. Het moet altijd mogelijk zijn het Monster in te sluiten, zodat de speler het spel kan winnen.
16. Het spel moet een melding “Game Won” geven en nieuw level genereren, wanneer het Monster is ingesloten en geen beweging meer kan maken.
17. Het Monster moet om de seconde een beweging maken, dichter naar de speler toe.
18. Wanneer de Monster de speler niet kan bereiken, beweegt hij in willekeurige directies.
19. Het Monster kan niet door kisten of muren bewegen.
20. Er moet een melding worden gegeven “Game Over” wanneer de speler is gevangen(aangeraakt) door de Monster; daarna wordt het level opnieuw gegenereerd.
21. Er moet een object “Box” zijn.
22. Kisten kunnen verschuiven door er tegen aan te lopen.
23. Kisten kunnen niet worden verschoven als er een muur achter staat.
24. Een Childmode, waarbij het mogelijk word om meerdere kisten tegelijk te verplaatsen(in totaal twee kisten).
25. Kisten kunnen niet worden verschoven wanneer er een of meerdere kisten achter staan, tenzij Childmode aanstaat.
26. Moeilijkheidsgraden per level. Iedere keer als de speler een level heeft behaald, moet er een nieuw(moeilijker) level worden gegenereerd.
27. Het spel moet een GUI hebben, zodat de speler makkelijk kan navigeren naar de functionaliteiten, zoals beschreven in de use cases.
28. Er moet een Menu in de GUI zijn, zodat de speler het spel kan beheren.

(Deze punten voor in het menu)

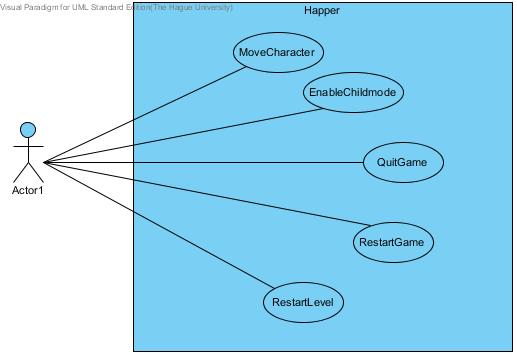
1. Het spel moet direct beginnen, nadat de speler de applicatie opstart.
2. Er moet een herstartSpel knop zijn, waarbij de moeilijkheidsgraad terug gaat naar het laagste niveau en er op dat niveau weer doolhoven worden gegenereerd, wanneer de speler erop drukt en bevestiging geeft, dat hij/zij het spel wilt resetten.
3. Er moet een RestartLevel knop zijn, waarbij de betreffende doolhof worden gereset, wanneer de speler erop drukt en bevestiging geeft, dat hij/zij de doolhof wilt resetten.
4. Er moet een enableChild knop zijn, waarbij de kindmodus wordt aangezet en het mogelijk word om twee dozen tegelijk te verplaatsen, wanneer de speler erop drukt en bevestiging geeft, dat hij/zij de Childmode wilt aanzetten.
5. Er moet een QuitGame knop zijn, wat de applicatie afsluit wanneer de speler erop drukt en bevestiging geeft, dat hij/zij het spel wilt afsluiten.
6. Zodra de speler op een van de vorige menuknoppen klikt, moet het spel gepauseerd worden terwijl er om een bevestiging word gevraagd. Dit houd in dat de monsters op dat moment niet bewegen.
7. Een handleiding moet met het spel meekomen, zodat de gebruiken deze kan doorlezen en weet hoe het spel werkt.
8. In de GUI moet een GameWindow zijn, waarin de spelactiviteiten zichtbaar zijn voor de speler.

## Non-functional requirements:

1. Het spelinterface(GUI) moet er mooi en strak uitzien.
2. Het spel moet snel reageren op een actie van de speler (gewenst < 75 ms).
3. Er moet een teller zijn voor het level waarin de speler zich bevind, die zichtbaar moet zijn voor de speler.
4. Monsters schieten LaserBeams af. Als de speler wordt geraakt door een beam begint het level opnieuw na de melding “Game Over”.
5. Eventuele uitbreidingen op het spel
6. De structuur van de levels moet kunnen worden opgehaald uit XML files die worden verwerkt in de applicatie.
7. De .XML files moeten los staan van het project.
8. Het spel bevat geavanceerde graphics en ook animaties.
9. Het spel bevat audio.
10. De speler moet de levels uit kunnen kiezen die hij heeft gehaald, om te spelen.

# Use Cases

## Visual Paradigm diagram



## Use case descriptions

|  |  |
| --- | --- |
| Naam | MoveCharacter |
| Id | K.1 |
| Samenvatting | Het bewegen van het karakter. |
| Primairy actors | Speler |
| Secondairy actors | Geen. |
| Precondities | Er is een level gestart. |
| Main Flow | 1. Actor drukt op een pijltoets. 2. Karakter beweegt in de richting van het ingedrukte pijltoets. 3. Het systeem verplaatst het karakter in de richting van de pijltoets. **[A] [B] [D] [G]** |
| Postcondities | Karakter heeft zich verplaatst. |
| Alternative Flows | **[A]**   1. Er bevindt zich een muur op de plaatst waar het karakter naartoe wordt verplaatst. 2. Systeem verplaatst het karakter niet.   **[B]**   1. Er bevinden zich twee kisten achter elkaar. **[C]** 2. Systeem verplaatst het karakter niet.   **[C]**   1. Kindmode staat aan. 2. Het systeem verplaatst het karakter en de twee kisten in de richting van de pijltjestoets.**[F]**   **[D]**   1. Er bevindt zich één kist op de plaatst waar de karakter naartoe wilt bewegen. 2. Het systeem verplaatst de kist en het karakter in de richting van de pijltoets. **[E]**   **[E]**   1. Er staat een monster of muur in de weg. 2. De kist en het karakter worden niet verplaatst.   **[F]**   1. Voor de rij met kisten staat een muur of monster. 2. Het systeem verplaatst het karakter en de kisten niet.   **[G]**   1. Er bevindt zich een monster op de plaatst waar het karakter naartoe wordt verplaatst. 2. Het karakter beweegt tegen het monster en er word een melding gegeven, dat het spel over is. 3. De actor kiest optie RestartGame. **[H]** 4. Het level wordt opnieuw gegenereerd. 5. De actor begint met spelen.   **[H]**   1. De actor kiest optie QuitGame 2. Het systeem sluit zichzelf af.   **[I]**  1. Er is geen positie meer in de directie waar het karakter heen wilt bewegen.  2. Het systeem verplaatst het karakter niet. |

|  |  |
| --- | --- |
| Naam | RestartGame |
| Id | G.1 |
| Samenvatting | Actor herstart het spel. |
| Primairy actors | Speler |
| Secondairy actors | Geen. |
| Precondities | Er is een level gestart. |
| Main Flow | 1. Actor drukt op knop Menu. 2. Actor kiest optie HerstartSpel. 3. Systeem vraagt om een bevestiging van de actor. 4. De actor kiest voor ja. **[A]** 5. Systeem zet het level attribuut op 1. 6. Systeem verwijdert oude spelcomponenten en maakt nieuwe aan. 7. Actor kan beginnen met spelen. |
| Postcondities | Spel is herstart. |
| Alternative Flows | **[A]**   1. De actor kiest voor nee. 2. Systeem hervat het actuele level. |

|  |  |
| --- | --- |
| Naam | QuitGame |
| Id | G.2 |
| Samenvatting | Actor stopt het spel. |
| Primairy actors | Speler |
| Secondairy actors | Geen. |
| Precondities | Er is een level gestart. |
| Main Flow | 1. Actor drukt op knop Menu. 2. Actor kiest optie QuitGame. 3. Systeem vraagt om bevestiging. 4. De actor kiest voor ja. **[A]** 5. Systeem sluit zichzelf af. |
| Postcondities | Systeem bevindt zich in het hoofdmenu. |
| Alternative Flows | **[A]**   1. Systeem hervat het actuele level. |

|  |  |
| --- | --- |
| Naam: | RestartLevel |
| ID: | G.3 |
| Samenvatting: | Het opnieuw starten van een level. |
| Primary actors: | Speler |
| Secondary actors: | Geen |
| Precondities: | Er is een spel actief. |
| Main Flow: | 1. De actor drukt op knop Menu. 2. De actor kiest optie RestartLevel. 3. Systeem vraagt om bevestiging. 4. De actor kiest voor ja **[A]** 5. Systeem verwijdert oude spelcomponenten en maakt nieuwe aan. 6. Actor kan beginnen met spelen in het actuele level. |
| Post condities: | Level is herstart. |
| Alternative Flows: | **[A]**   1. De actor kiest voor nee 2. Systeem hervat het actuele level. |

|  |  |
| --- | --- |
| Naam: | EnableKindMode |
| ID: | K.2 |
| Samenvatting: | Het aanzetten van de Kindmode |
| Primary actors: | Speler |
| Secondary actors: | Geen |
| Precondities: | Geen. |
| Main Flow: | 1. De actor drukt op knop Menu **[A]** 2. De actor kiest optie EnableKindmode. 3. Systeem vraagt om een bevestiging van de actor. 4. De actor drukt op ja. **[B]** 5. Kindmode attribuut wordt op ‘true’ gezet. |
| Post condities: | Kindmode is enabled. |
| Alternative Flows: | **[A]**   1. De actor bevindt zich in een level. 2. De actor drukt op knop Menu. 3. De actor kiest optie EnableKindmode. 4. Systeem vraagt om bevestiging. 5. De actor drukt op ja. **[C]** 6. Kindmode attribuut wordt op ‘true’ gezet. 7. Systeem hervat het actuele level.   **[B]**   1. De actor drukt op nee. 2. Systeem keert terug naar hoofdmenu.   **[C]**   1. De actor drukt op nee. 2. Systeem hervat het level. |

# Planning 1. Globale Planning

|  |  |
| --- | --- |
| Week 2 | * Inception rapport **bevat**: * plan van aanpak * Requirements * UML Use case diagram * Use case beschrijvingen * **Ook maken deze week:** * UML Analyse Klassendiagram * UML Design Klassendiagram |
| Week 5 | * Elaboration rapport **bevat**: * UML Analyse Klassendiagram * UML Design Klassendiagram * Refined Requirements * JUnit testscripts van prototype * Ook maken deze week: * Architectuur proof of concept (prototype) * UML Sequentie Diagram |
| Week 6 | * Construction rapport van iteratie 1 **bevat**: * Applicatie van het (basis)spel * Logisch en fysiek testontwerp m.b.v. beslissingstabellen * Logisch en fysiek ontwerp met behulp van algoritme test * Handleiding van het spel * UML Sequentie Diagram |
| Week 6 | * Doelgroepanalyse * Logische en fysieke testontwerpen |
| Week 8 | * Construction rapport van iteratie 2 * Applicatie met eventuele uitbreidingen en verbeteringen * JUnit regressietest van de applicatie * JUnit testscripts van de applicatie * Testrapport |
| Week 8 | * Individueel procesverslag |