|  |
| --- |
| ­­­­­­­­­Inception rapport |
| Projectopdracht Informatica 2 |
| Blok I2 – Peter-Pim Baken |

16-12-2012 Docent: Tim Cocx

Inhoudsopgave

[1. Inleiding 1](#_Toc346795277)

[2. Omschrijving en doelstelling project 1](#_Toc346795278)

[1. Opdrachtomschrijving 1](#_Toc346795279)

[2. Probleembeschrijving 2](#_Toc346795280)

[3. Doelstelling van de opdracht 2](#_Toc346795281)

[3. Projectorganisatie 2](#_Toc346795282)

[1. Randvoorwaarden 2](#_Toc346795283)

[4. Benodigde mensen en middelen 2](#_Toc346795284)

[1. De relevante mensen zijn 2](#_Toc346795285)

[2. De relevante spullen zijn 2](#_Toc346795286)

[5. Functional and non-functional requirements 3](#_Toc346795287)

[1. Functional requirements 3](#_Toc346795288)

[2. Non-functional requirements: 4](#_Toc346795289)

[6. Use Cases 5](#_Toc346795290)

[1. Use case diagram 5](#_Toc346795291)

[2. Use case descriptions 5](#_Toc346795292)

[7. Planning 1. Globale Planning 6](#_Toc346795293)

# Inleiding

Dit rapport gaat over de manier waarop we de opdracht voor het maken van een Pacman spel aanpakken.

# 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 van toepassing.

## 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 waar de speler het spel in speelt.
5. Het speelveld moet Positions hebben, waarop de spelobjecten kunnen worden geplaatst.
6. Er moet een object “Wall” zijn, waar de speler niet doorheen kan bewegen .
7. Er moet een object “Player” zijn, die door de speler word bestuurt.
8. Er moet een object “Monster” zijn.
9. Er moet een object “Box” zijn.
10. Het speelveld word gemaakt op basis van een level, die van te voren zijn gemaakt.
11. Het spel wordt gestart wanneer de speler de applicatie opstart.
12. Zodra het spel wordt gestart, moet er een speelveld worden aangemaakt, dat vervolgens word gevuld met de spelobjecten voor dat level.
13. Zodra een level wordt gestart, word de score op nul gezet.
14. Zodra een level wordt gestart, begint het monster de bewegen.
15. Alle objecten moeten dezelfde grootte hebben, ze nemen dus allemaal een gelijke hoeveelheid Positions in van het speelveld.
16. Het moet altijd mogelijk zijn het Monster in te sluiten, zodat de speler het spel kan winnen.
17. Het spel moet een melding “Game Won” , wanneer het Monster is ingesloten en geen beweging meer kan maken. Het level dat wordt gegenereerd is het volgende level of hetzelfde, afhankelijk van de speler keuze.
18. Het Monster moet om de seconde een beweging maken, dichter naar de speler toe.
19. Wanneer de Monster de speler niet kan bereiken, beweegt hij in willekeurige directies.
20. Het Monster kan niet door kisten of muren bewegen.
21. Er moet een melding worden gegeven “Game Over” wanneer de speler is gevangen(aangeraakt) door de Monster; daarna wordt het level opnieuw gegenereerd of gestopt, afhankelijk van de keuze van de speler.
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(het word dus mogelijk een rij kisten achter elkaar te verplaatsen).
25. Kisten kunnen niet worden verschoven wanneer er een of meerdere kisten achter staan, tenzij Childmode aanstaat.
26. Naarmate de speler verder komt in het spel, moeten de levels moeilijker worden, dus het word steeds moeilijker het monster in te sluiten.
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)

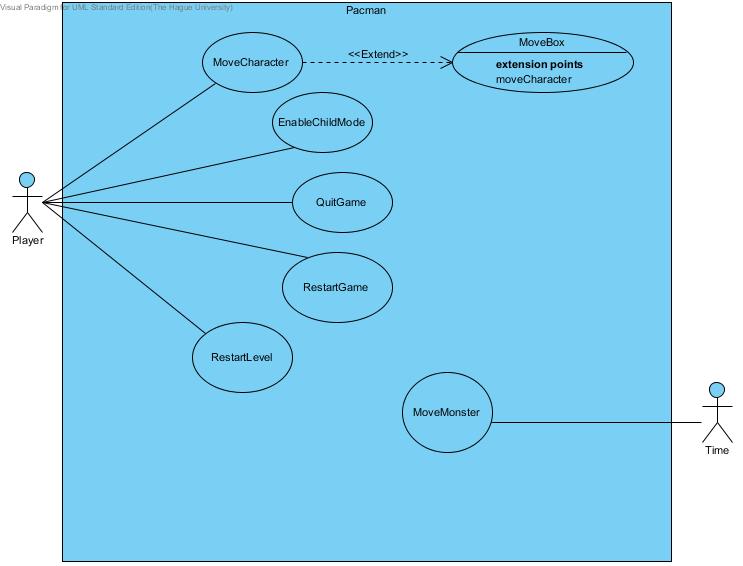
1. Het spel moet direct beginnen, nadat de speler de applicatie opstart.
2. Er moet een knop zijn om het spel te herstarten, waarbij de moeilijkheidsgraad terug gaat naar het laagste niveau en er op dat niveau weer doolhoven worden gegenereerd, wanneer de speler erop drukt en bevestiging geeft, dat hij/zij het spel wilt resetten.
3. Er moet een knop zijn om het level te herstarten, 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 knop zijn om de childmode aan te zetten, waarbij de childmode wordt ingeschakeld en het mogelijk word om een rij kisten tegelijk te verplaatsen.
5. Er moet een knop zijn om het spel af te sluiten, 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

## Use case 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. Het spel is niet gepauzeerd. |
| Main Flow | 1. De actor drukt op een pijltoets. 2. Het systeem verplaatst het karakter in de richting van de pijltoets. **[A] [B] [C] [D]** |
| Postcondities | Karakter heeft zich verplaatst. |
| Alternative Flows | **[A]**   1. Er bevindt zich een muur op de plaatst waar het karakter commando krijgt naartoe te verplaatsen. 2. Systeem verplaatst het karakter niet.   **[B]**   1. Er bevindt zich een monster op de plaatst waar het karakter commando krijgt naartoe te verplaatsen. 2. Het systeem beweegt het karakter niet.   **[C]**   1. Er bevind zich geen positie in de richting waar het karakter commando krijgt heen te verplaatsen. 2. Het systeem verplaatst het karakter niet.   **[D]**   1. Er bevind zich een kist in de richting waar het karakter commando krijgt heen te verplaatsen. 2. Ga verder naar use case ‘K.2’. |

|  |  |
| --- | --- |
| Naam | MoveBox |
| Id | K.2 |
| Samenvatting | Het bewegen van een kist, extensie op de use case MoveCharacter |
| Primairy actors | Speler |
| Secondairy actors | Geen. |
| Precondities | Use case K.1 is uitgevoerd. |
| Main Flow | 1. Het systeem verplaatst de kist en het karakter in de richting waar het karakter heen beweegt. **[A] [B] [D][E]** |
| Postcondities | Kist heeft zich verplaatst. |
| Alternative Flows | **[A]**   1. Er bevindt zich een muur op de plaatst waar de kist heen moet worden verplaats. 2. Het systeem verplaatst de kist niet.   **[B]**   1. Er bevind zich een kist, op de plaats waar de kist heen moet worden verplaatst. **[C]** 2. Het systeem verplaatst de kist niet.   **[C]**   1. De ChildMode staat aan. 2. Voer use case ‘K.2’ uit. 3. Indien use case ‘K.2’ succesvol is uitgevoerd, beweegt het systeem de kist in de directie waar het karakter heen beweegt.   **[D]**   1. Er staat een monster in de richting waar de kist heen moet worden verplaatst. 2. Het systeem verplaatst de kist niet.   **[E]**   1. Er bestaat geen positie in de richting waar de kist heen moet worden verplaatst. 2. Het systeem verplaatst de kist niet. |

|  |  |
| --- | --- |
| Naam | MoveMonster |
| Id | K.3 |
| Samenvatting | Het bewegen van het monster. |
| Primairy actors | Time |
| Secondairy actors | Geen. |
| Precondities | Er is een level gestart. Het spel is niet gepauzeerd. |
| Main Flow | 1. Er gaat een seconde voorbij in het spel. 2. Het systeem kiest een directie waar het monster heen moet bewegen om zo dicht mogelijk bij de speler te komen.**[A][C]** 3. Het monster beweegt in de besloten directie. |
| Postcondities | Karakter heeft zich verplaatst. |
| Alternative Flows | **[A]**   1. De speler is onbereikbaar voor het monster. 2. Het monster beweegt in mogelijke willekeurige directie.**[B]**   **[B]**   1. Er is geen mogelijkheid voor het monster om te bewegen. 2. Het systeem geeft een melding dat de speler het level heeft gewonnen.   **[C]**   1. Op de plaats waar het monster commando krijgt om heen te bewegen, staat de speler. 2. Het systeem geeft een melding dat de speler het level heeft verloren. |

|  |  |
| --- | --- |
| Naam | RestartGame |
| Id | G.1 |
| Samenvatting | Actor herstart het spel. |
| Primairy actors | Speler |
| Secondairy actors | Geen. |
| Precondities | Het spel is gestart. |
| Main Flow | 1. De actor kiest optie HerstartSpel. 2. Het systeem vraagt om een bevestiging van de actor. 3. De actor kiest voor ja. **[A]** 4. Het systeem zet het levelniveau op 0. 5. Het systeem verwijdert alle objecten en initialiseert nieuwe spelobjecten. 6. Het systeem zet de speler score op 0. 7. Het system hertekent het level. 8. Actor kan beginnen met spelen. |
| Postcondities | Spel is herstart. |
| Alternative Flows | **[A]**   1. De actor kiest voor nee. 2. De actor gaat verder met spelen. |

|  |  |
| --- | --- |
| 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. De actor kiest optie QuitGame. 2. Het systeem vraagt om bevestiging. 3. De actor kiest voor ja. **[A]** 4. Het systeem sluit zichzelf af. |
| Postcondities | Systeem bevindt zich in het hoofdmenu. |
| Alternative Flows | **[A]**   1. De actor gaat verder met spelen. |

|  |  |
| --- | --- |
| 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 kiest optie RestartLevel. 2. Het systeem vraagt om bevestiging. 3. De actor kiest voor ja **[A]** 4. Het systeem verwijdert alle spelobjecten en initialiseren nieuwe spelobjecten. 5. Het systeem zet de spelerscore op 0. 6. De 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: | EnableChildmode |
| ID: | R.1 |
| Samenvatting: | Het aanzetten van de Childmode |
| Primary actors: | Speler |
| Secondary actors: | Geen |
| Precondities: | Geen. |
| Main Flow: | 1. De actor kiest optie EnableChildmode. 2. Het systeem zet de Childmode aan.**[B]** 3. Het systeem geeft een melding dat de Childmode is aangezet. |
| Post condities: | Childmode is aangezet. |
| Alternative Flows: | **[B]**   1. De Childmode staat al aan. 2. Het systeem zet de Childmode uit. 3. Het systeem geeft een melding dat de Childmode is aangezet. |

# 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 * Logisch en fysiek testontwerp van het prototype * Ook maken deze week: * Architectuur proof of concept (prototype) * UML Sequentie Diagram van de use cases moveCharacter, QuitGame, RestartGame |
| Week 6 | * Iteratie 1 **Maken**: * 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 Diagrammen van use case moveCharacter en methode draw() |
| Week 6 | * Doelgroepanalyse * Logische en fysieke testontwerpen |
| Week 8 | * Construction rapport van iteratie 1 en 2 * Applicatie met eventuele uitbreidingen en verbeteringen * Testrapport |
| Week 8 | * Individueel procesverslag |