|  |  |
| --- | --- |
| Pacman  APPLICATIE DOCUMENT | Abstract  “Waka waka waka waka “ ~ Pacman  Denzel Bendt (student) |

# 1. Inleiding

## 1.1 Doel van het Document

In dit document neem ik u graag mee in het ontwikkelingsproces van mijn Pacman-applicatie. Het doel is om u niet alleen te informeren over de werking en de technische aspecten van de applicatie, maar ook over de keuzes en stappen die ik heb gemaakt tijdens de ontwikkeling. De applicatie is gericht op belanghebbenden die een affiniteit hebben met gamen, en in het bijzonder op liefhebbers van klassieke arcade-games zoals Pacman.

Pacman is een tijdloos spel dat velen herkennen, en het bood een mooie uitdaging om deze ervaring opnieuw vorm te geven. Door mijn passie voor zowel programmeren als klassieke spellen te combineren, heb ik een interactieve en authentieke game-ervaring gecreëerd. In dit document behandel ik elk aspect van de ontwikkeling, van de eerste conceptfase tot de technische implementatie en de visuele keuzes die ik heb gemaakt om de retro-look en -feel te behouden.

## 1.2 Doel van de Applicatie

Mijn Pacman-applicatie is ontworpen om de nostalgie van het klassieke spel te herbeleven, terwijl ik tegelijkertijd mijn vaardigheden als softwareontwikkelaar toon. De applicatie is gericht op mensen die houden van het oplossen van problemen en het ontdekken van de technische processen achter een spel. Door u als lezer mee te nemen in mijn ontwikkelingsreis, hoop ik mijn passie voor game-ontwikkeling over te brengen en inzicht te geven in wat er allemaal komt kijken bij het creëren van een interactieve applicatie.

Of u nu een gameliefhebber bent of simpelweg geïnteresseerd in de technieken achter klassieke games, ik hoop dat dit document u een inspirerende en informatieve kijk biedt op de ontwikkeling van mijn Pacman-applicatie.

## 1.3 Scope

Zie de weergaven van de scope en no scope van de applicatie, hieronder.

### 1.3.1 Belangrijkste Functionaliteiten

De game moet de volgende basisfunctionaliteiten bevatten:

* **Beweging en besturing**: Pacman kan door de gebruiker met pijltjestoetsen worden bestuurd en vrij bewegen door het doolhof.
* **Doolhofstructuur:** Statische doolhoven met paden en muren waar Pacman en de spookjes doorheen bewegen.
* **Pellets:** Punten (pellets) worden door het doolhof verspreid en kunnen door Pacman worden opgegeten, wat punten toevoegt aan de score.
* **Spookjes:** AI-gestuurde spookjes die Pacman door het doolhof achtervolgen. Als Pacman wordt gevangen, is het game over.
* **Score en level:** Een scoreboard dat de huidige score en level toont
* **Highscore manager:**  Een functie binnen de app die de highscores van de speler bijhoudt in overzicht(een lijst met ranknummers).
* **Menu:** De optie voor de gebruiker om te kiezen tussen het spelen van het spel, highscores te bekijken en het spel af te sluiten.

### 1.3.2 Uitgesloten Functionaliteiten (Out of Scope)

Om de complexiteit en projectomvang beheersbaar te houden, worden de volgende functies uitgesloten:

* **Power-pellets:** Specifieke pellets die Pacman tijdelijk de mogelijkheid geven om spookjes te “eten” voor extra punten.
* **Levens:** De speler meerdere kansen geven om een level te behalen nadat de speler geraakt is door een spookje.
* **Multiplayer-functionaliteit:** De game is ontworpen voor één speler en bevat geen co-op- of PvP-opties. Door de tijd heen, werd dit populairder onder de gamers.
* **Meerdere niveaus:** Er wordt slechts drie doolhoven gebruikt.
* **Geavanceerde AI voor spookjes:** De AI van de spookjes zal eenvoudig zijn (bijvoorbeeld willekeurige bewegingen of eenvoudige achtervolgingslogica) en niet de complexiteit van het originele Pacman-spel benaderen.
* **Geluids- en achtergrondmuziek:** De focus ligt op het spelontwerp en de functionaliteit; geluidseffecten en muziek worden niet toegevoegd in deze versie.

### 1.3.3 Technische Specificaties

* **Programmeertaal en omgeving:** Het project wordt gebouwd in C#, als conole window app.
* **Platform:** De applicatie is gericht op pc-gebruik.
* **Resolutie:** Een vaste schermresolutie om visuele consistentie te behouden.(60x30, breedte x hoogte)

### 1.3.4 Projecttijdlijn

Zie hier een overzicht van de tijdlijn van dit project:

* **Sprint 1**: Ontwerp van het doolhof en basisbeweging van Pacman.
* **Sprint 2**: Implementatie van de punten/pellets en scoretelling.
* **Sprint 3**: Toevoegen van spookjes, eenvoudige AI-logica.
* **Sprint 4**: Integratie van levels, highscore, menu en bijwerken van de scorelogica.
* **Sprint 5:** Testen, debugging en visuele verfijning.

### 1.3.5 Kwaliteitscriteria en Acceptatievoorwaarden

* **Functionaliteit:** De game moet vrij van fouten draaien en reageren op gebruikersinput zonder vertraging.
* **Correcte scoreweergave:** Punten worden correct bijgehouden en weergegeven wanneer Pacman pellets eet.
* **Duidelijke visuele weergave:** Het doolhof, Pacman en spookjes zijn goed zichtbaar en bewegen vloeiend.
* **Spelafloop:** Het spel eindigt wanneer Pacman geen levens meer heeft, en een eindscherm geeft de eindscore weer.

# 2. Functionele Vereisten

In hoofdstuk 1.3.1 , is een overzicht van de belangrijkste functionaliteiten weergegeven van de applicatie. Binnen dit hoofdstuk wordt er dieper ingegaan, door de acceptatiecriteria, de functionele en niet functionele vereisten te beschrijven.

## 2.1 Overzicht Functionaliteiten

Zie de overzicht wat eerder is weergegeven weer:

* **Beweging en besturing**: Pacman kan door de gebruiker met pijltjestoetsen worden bestuurd en vrij bewegen door het doolhof.
* **Doolhofstructuur:** Statische doolhoven met paden en muren waar Pacman en de spookjes doorheen bewegen.
* **Pellets:** Punten (pellets) worden door het doolhof verspreid en kunnen door Pacman worden opgegeten, wat punten toevoegt aan de score.
* **Spookjes:** AI-gestuurde spookjes die Pacman door het doolhof achtervolgen. Als Pacman wordt gevangen, is het game over.
* **Score en level:** Een scoreboard dat de huidige score en level toont
* **Highscore manager:**  Een functie binnen de app die de highscores van de speler bijhoudt in overzicht(een lijst met ranknummers).
* **Menu:** De optie voor de gebruiker om te kiezen tussen het spelen van het spel, highscores te bekijken en het spel af te sluiten.

## 2.2 Gedetailleerde Functionele Vereisten

De gedetailleerde functionele vereisten per functionaliteit:

**1. Beweging en besturing**

* **Functionele vereisten**:
  + Pacman beweegt op basis van pijltjestoetsen.
  + Pacman stopt met bewegen als hij een muur raakt.
  + Pacman kan door openingen aan de randen van het doolhof gaan om naar de andere kant te teleporteren.

**2. Doolhofstructuur**

* **Functionele vereisten**:
  + Het doolhof bevat statische muren en paden, die Pacman en de spookjes sturen in hun beweging.
  + Het doolhof bevat doorgangen aan de randen om teleportatie mogelijk te maken.
  + Doolhofstructuur moet objecten zoals pellets en spookjes correct positioneren bij het starten van een nieuw spel.

**3. Pellets**

* **Functionele vereisten**:
  + Pellets worden correct weergegeven in het doolhof bij het starten van elk level.
  + Pacman kan pellets "opeten" door eroverheen te bewegen, wat punten toevoegt aan de score.
  + Power-pellets moeten afzonderlijk worden weergegeven en geven Pacman tijdelijk de mogelijkheid om spookjes te eten.

**4. Spookjes**

* **Functionele vereisten**:
  + Spookjes navigeren door het doolhof en volgen Pacman op basis van AI-patronen.

**5. Score en level**

* **Functionele vereisten**:
  + Een scoreboard toont de huidige score en het level.
  + De score wordt bijgewerkt wanneer Pacman pellets eet.
  + Het level wordt verhoogd zodra alle pellets in een level zijn opgegeten.

**6. Highscore manager**

* **Functionele vereisten**:
  + Highscores worden bijgehouden en opgeslagen, met een overzicht van de beste scores op volgorde van hoogste naar laagste.
  + De speler kan highscores bekijken in een aparte sectie van de app.

**7. Menu**

* **Functionele vereisten**:
  + Het menu bevat opties om het spel te starten, highscores te bekijken en het spel af te sluiten.
  + De gebruiker kan met het toetsenbord of de muis door het menu navigeren.

## 2.3 Niet-functionele Vereisten

De gedetailleerde niet-functionele vereisten per functionaliteit:

**1. Beweging en besturing**

* **Niet-functionele vereisten:**
  + Responsieve besturing, met minimale vertraging tussen de invoer van toetsen en beweging.
  + Bewegingsanimaties moeten vloeiend en consistent zijn bij alle framerates.

**2. Doolhofstructuur**

* **Niet-functionele vereisten**:
  + De doolhofstructuur moet snel geladen worden bij het starten van een spel.
  + Het ontwerp van het doolhof moet visueel aantrekkelijk en duidelijk zijn.

**3. Pellets**

* **Niet-functionele vereisten**:
  + Pellets moeten visueel te herkennen zijn voor de speler.

**4. Spookjes**

* **Niet-functionele vereisten**:
  + Spookjes moeten een vloeiende, niet-schokkerige beweging hebben.
  + De AI van spookjes moet voorspelbare prestaties leveren zonder veel vertraging.

**5. Score en level**

* **Niet-functionele vereisten**:
  + De scoreboardweergave moet altijd zichtbaar en goed leesbaar zijn.
  + Score en level moeten direct bijgewerkt worden zonder vertraging.

**6. Highscore manager**

* **Niet-functionele vereisten**:
  + Highscores moeten persistent zijn, zelfs na het afsluiten van de applicatie.
  + De highscorelijst moet visueel georganiseerd en makkelijk leesbaar zijn.

**7. Menu**

* **Niet-functionele vereisten**:
  + Het menu moet gebruiksvriendelijk en eenvoudig te navigeren zijn.
  + Overgangen tussen menu-opties en het spel moeten vloeiend zijn en zonder vertraging verlopen.

## 2.4 Acceptatiecriteria

De acceptatiecriteria per functionaliteit:

**1. Beweging en besturing**

* **Acceptatiecriteria**:
  + Pacman reageert direct op invoer van de pijltjestoetsen en stopt bij contact met een muur.
  + Pacman teleporteert correct door het doolhof via randopeningen.
  + Beweging is vloeiend en toont geen haperingen.

**2. Doolhofstructuur**

* **Acceptatiecriteria**:
  + Doolhofstructuur wordt correct weergegeven en blijft consistent tijdens het spel.
  + Pellets en spookjes zijn op de juiste posities geplaatst bij start van elk level.
  + Het doolhof laadt in minder dan 2 seconden bij het starten van een nieuw spel.

**3. Pellets**

* **Acceptatiecriteria**:
  + Pacman kan alle pellets in een level opeten, en de score wordt correct bijgewerkt.

**4. Spookjes**

* **Acceptatiecriteria**:
  + Spookjes volgen Pacman volgens gedefinieerde AI-patronen.
  + Beweging van de spookjes is vloeiend en consistent.

**5. Score en level**

* **Acceptatiecriteria**:
  + Score en level worden correct bijgewerkt in de scoreboardweergave tijdens het spel.
  + Scoreboard is goed zichtbaar en bijgewerkt zonder vertraging na elke actie.
  + Level-up vindt plaats zodra alle pellets in een level zijn opgegeten.

**6. Highscore manager**

* **Acceptatiecriteria**:
  + Highscores worden correct opgeslagen en zijn persistent na afsluiten van de applicatie.
  + Speler kan op elk moment het highscoreoverzicht bekijken in het menu.
  + Highscorelijst toont scores van hoogste naar laagste en is goed leesbaar.

**7. Menu**

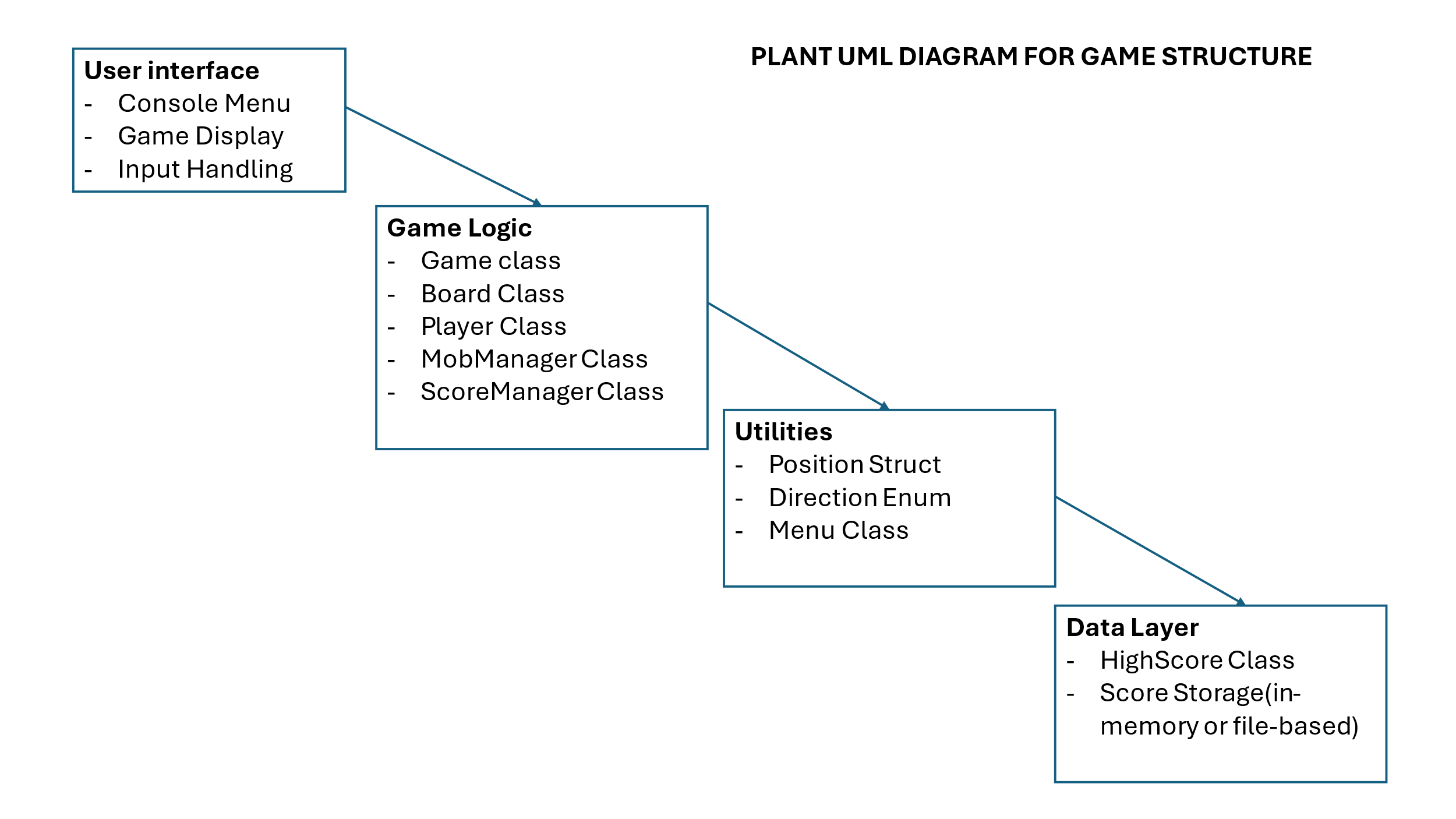
* **Acceptatiecriteria**:
  + Menu biedt de opties om het spel te starten, highscores te bekijken, en de applicatie af te sluiten.
  + Navigatie door het menu werkt correct met toetsenbord.
  + Overgangen van menu naar spel en terug zijn vloeiend en zonder vertraging.

# 3. Technische Specificaties

## 3.1 Architectuur

In dit hoofdstuk is beknopt beschreven uit welke lagen en componenten de app bestaat, ook is er een diagram weergegeven, die het overzicht biedt.

### Beschrijving van de lagen en componenten



*Een diagram die aangeeft hoe de app gebouwd is.*

**User Interface (UI)**

* **Console Menu**: Behandelt de navigatie en interactie met de gebruiker via de console.
* **Game Display**: Toont de huidige status van het spel, inclusief de speler, mobs, en het bord.
* **Input Handling**: Verwerkt gebruikersinvoer, zoals toetsenbordinvoer voor beweging en menu-opties.

**Game Logic**

* **Game Class**: Beheert de spelstatus, start en reset het spel, en coördineert de interactie tussen andere componenten.
* **Board Class**: Beheert de spelomgeving, inclusief het doolhof, muren, en het verzamelen van punten.
* **Player Class**: Beheert de status van de speler, zoals positie en score.
* **MobManager Class**: Beheert de beweging en logica van de vijanden (mobs) in het spel.
* **ScoreManager Class**: Beheert de scores van de spelers en de highscorelijst.

**Data Layer**

* **HighScore Class**: Bevat informatie over de high scores, inclusief de naam van de speler, score en datum.
* **Score Storage**: Beheert de opslag van scores, dit kan in-memory zijn of in een bestand.

**Utilities**

* **Position Struct**: Bevat de X- en Y-coördinaten voor de positie van de speler en mobs.
* **Direction Enum**: Definieert de mogelijke richtingen (bijv. omhoog, omlaag, links, rechts) voor beweging.
* **Menu Class**: Behandelt de logica voor het weergeven van menu's en het vragen om gebruikersinvoer.

## 3.2 Datamodel

Per Entiteit zijn de relaties met andere attributen entiteiten kort beschreven met afbeeldingen als ondersteuning.

### 1. Entiteit: Player

**Attributen**:

* PlayerName (string): De naam van de speler. Dit is de naam die de speler invoert en die wordt weergegeven in de highscorelijst.
* Score (int): De huidige score van de speler tijdens de sessie. Dit kan worden verhoogd door het verzamelen van punten in het spel.
* Position (Position struct): De huidige positie van de speler op het bord, met X- en Y-coördinaten.
* CurrentLevel (int): Het huidige level waarin de speler zich bevindt.

**Relaties**:

* Een speler kan meerdere high scores hebben, maar in de context van het spel is er meestal slechts één actieve score per sessie.

### 2. Entiteit: HighScore

**Attributen**:

* PlayerName (string): De naam van de speler die de score heeft behaald.
* Score (int): De behaalde score.
* Date (DateTime): De datum en tijd waarop de score is behaald.
* LevelAchieved (int): Het hoogste level dat de speler heeft bereikt bij het behalen van deze score.

**Relaties**:

* Een high score is gekoppeld aan één speler, maar een speler kan meerdere high scores hebben (bijvoorbeeld in verschillende speelsessies).

### 3. Entiteit: Board

**Attributen**:

* Maze (char[,]): Een 2D-array van karakters die het doolhof voorstelt. Dit bevat muren (#) en open ruimtes ( ).
* Width (int): De breedte van het bord.
* Height (int): De hoogte van het bord.
* CurrentLevel (int): Het huidige level van het spel.
* TotalDots (int): Het totale aantal punten dat verzameld kan worden in het level. Dots worden aangegeven met een punt (‘ . ’).
* Dots (bool[,]): Een 2D-array die bijhoudt welke punten zijn verzameld.

**Relaties**:

* Het bord is verbonden met de speler en de mobs, omdat het de omgeving is waarin zij zich bevinden.

### 4. Entiteit: Mob

**Attributen**:

* Position (Position struct): De huidige positie van de mob op het bord.
* Type (string): Het type mob (bijv. 'ghost', 'enemy'). Dit kan invloed hebben op hun gedrag.
* Speed (int): De snelheid waarmee de mob beweegt.
* Direction (Direction enum): De huidige bewegingsrichting van de mob.

**Relaties**:

* Mobs zijn verbonden met het bord en de speler, omdat ze zich in dezelfde omgeving bevinden en interactie kunnen hebben met de speler.

### 5. Entiteit: ScoreManager

**Attributen**:

* HighScores (List<HighScore>): Een lijst van HighScore objecten die de high scores bijhouden.
* MaxHighScores (int): Het maximum aantal high scores dat kan worden opgeslagen.

**Relaties**:

* De ScoreManager beheert de HighScore entiteiten en is verantwoordelijk voor het toevoegen, sorteren en weergeven van scores.

### 6. Entiteit: Game

**Attributen**:

* CurrentPlayer (Player): De huidige speler die het spel speelt.
* Board (Board): Het huidige bord dat wordt weergegeven.
* MobManager (MobManager): Beheert de mobs in het spel.
* ScoreManager (ScoreManager): Beheert de scores en high scores.

**Relaties**:

* De Game entiteit coördineert de interacties tussen de Player, Board, MobManager, en ScoreManager.

### Relaties Overzicht

1. **De Player** heeft een één-op-veel relatie met **HighScore**: Een speler kan meerdere high scores hebben, maar elke high score is gekoppeld aan één speler.
2. **Board** heeft een relatie met **Player** en **Mob**: Het bord is de omgeving waarin zowel de speler als de mobs zich bevinden.
3. **Mob** heeft een relatie met **Board**: Mobs bewegen binnen de grenzen van het bord.
4. **ScoreManager** beheert de **HighScore** entiteiten en zorgt ervoor dat scores correct worden opgeslagen en weergegeven.
5. **Game** coördineert de interacties tussen alle entiteiten en beheert de spelstatus.

### Functionaliteit en Interacties

* **Spelstart**: De speler start het spel, en de Game class initialiseert een nieuwe Player, Board, en ScoreManager.
* **Spelverloop**: De speler beweegt op het bord, verzamelt punten en interacteert met mobs. De score wordt bijgehouden in de Player class.
* **Level Voltooiing**: Wanneer een level is voltooid, wordt de score opgeslagen in de ScoreManager, en de speler kan hun naam invoeren voor de highscorelijst.
* **Game Over**: Bij game over wordt de score weergegeven en kan de speler opnieuw beginnen of terugkeren naar het menu.

---

# 4. Gebruikersinterface (UI) Ontwerp

## 4.1 Overzicht van Schermen

In dit deel van het hoofdstuk is met afbeeldingen weergegeven hoe de verschillende schermen van de app eruit zien. Bij de levels had ik de mobs uitgeschakeld om te voorkomen dat ik verhinderd werd tijdens het maken van een screenshot.

#### Startscherm/menu

Afbeelding met tekst, Lettertype, schermopname, ontwerp

Automatisch gegenereerde beschrijving

#### Level 1

Afbeelding met schermopname, tekst, plein

Automatisch gegenereerde beschrijving

#### Level 2

Afbeelding met schermopname, tekst, scherm, ontwerp

Automatisch gegenereerde beschrijving

#### Level 3

Afbeelding met schermopname, tekst, scherm, plein

Automatisch gegenereerde beschrijving

#### Overgang naar next level

Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijving

#### Alles gehaald

Afbeelding met tekst, schermopname, scherm

Automatisch gegenereerde beschrijving

#### Highscore list

Afbeelding met tekst, schermopname, Lettertype, ontwerp

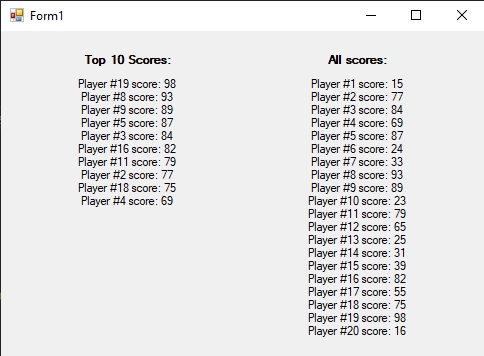
Automatisch gegenereerde beschrijving

## 4.2 Mockups/Wireframes

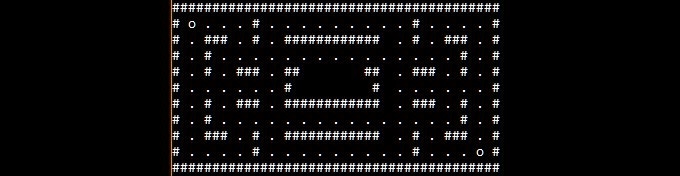
Voor het design front heb ik afbeeldingen van het internet gebruikt om te kunnen realiseren wat ik nu heb in de game zelf.



Voor het menu na het opstarten heb ik de dash letter art gebruikt als inspiratie, zodat de speler toch het gevoel heeft dat een spel is opgestart en het spel gespeeld kan worden.



Ook voor de highscorelist wilde ik een zo simpel mogelijke UI interface. Net als de highscore list die hiernaast is weergegeven.

En tot slot voor de doolhoven heb ik het volgende gebruikt om mijn versie van het speelveld te verwerkelijken.

# 5. Testplan

## 5.1 Teststrategie

**5.1.1Doel**

De teststrategie richt zich op het verifiëren van de basisfunctionaliteit en gebruikservaring van een eenvoudige Pacman-app. De belangrijkste functies die worden getest, zijn de bewegingen van Pacman, interacties met objecten (punten en geesten), scoreberekening en eindcondities van het spel.

**5.1.2 Testniveaus en -typen**

* **Integratietests:**  
  De interactie tussen verschillende componenten, zoals Pacman en de spelfunctionaliteiten (bijvoorbeeld muren, geesten en punten) worden getest. En door te controleren of alle onderdelen goed samenwerken en dat het spel de interacties correct verwerkt.
* **Functionele testen:**  
  Verifieert of de basisfunctionaliteiten (zoals beweging, het verzamelen van punten, en scoreberekening) voldoen aan de acceptatiecriteria.
* **User Interface (UI) Testing:**  
  Controleer de lay-out en zichtbaarheid van elementen zoals het spelbord, de scoreweergave, en het huidige level.
* **Acceptatietests:**  
  Door te verifiëren of de app voldoet aan de eisen van de gebruikers. De acceptatietests richten zich op het testen van de volledige speelervaring, zoals beschreven in de acceptatiecriteria.

**5.1.3 Testscenario's en Testgevallen**

In Hoofdstuk 5.2 worden meerdere testgevallen beschreven. Hier zijn enkele testscenario's die moeten worden uitgevoerd:

* **Start van het Spel:**  
  Test of het spel correct begint wanneer op de startknop wordt gedrukt.
* **Bewegingscontrole van Pacman:**  
  Verifieert dat Pacman in alle vier de richtingen kan bewegen en stopt bij muren.
* **Verzamelen van punten:**  
  Test dat Pacman punten kan verzamelen en dat de score met elk verzameld punt toeneemt.
* **Botsing met Geesten:**  
  Test dat het spel eindigt wanneer Pacman een spook raakt en het “Game Over”-scherm verschijnt.
* **Winconditie:**  
  Verifieer dat het spel correct eindigt met een overwinning wanneer alle punten zijn verzameld.

**5.1.4 Testomgevingen**

De Pacman-app wordt getest op:

* Verschillende resoluties (voor consistentie van het UI-design).
* Verschillende apparaten of platforms, indien de applicatie bijvoorbeeld bedoeld is voor zowel desktop als mobiele apparaten.

**5.1.5 Risico's en Beperkingen**

* **Risico’s:**  
  Het niet goed testen van randgevallen, zoals botsingen in de hoek van het speelveld, kan leiden tot bugs in de bewegingen van Pacman.
* **Beperkingen:**  
  Aangezien dit een kleine app is, wordt de teststrategie beperkt gehouden tot basisfunctionaliteiten zonder uitgebreide load- en performance testing.

**5.1.6 Testrapportage**

In de sprintdocumentatie komt naar voren wat succesvol gegaan is, wat mislukt is tijdens elke sprint, wat meer aandacht nodig had in het proces en wat nog openstaat.

## 5.2 Acceptatietesten

Zie hieronder een lijst van de testcases en de status daarvan.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Testcase ID | Omschrijving | Input | |  | | --- | | **Verwachte Uitkomst** |  |  | | --- | |  | | Status |
| |  | | --- | | **TC-001** |  |  | | --- | |  | | |  | | --- | | Startscherm wordt correct weergegeven |  |  | | --- | |  | | Start spel | Start scherm met een menu, waaruit start spel, highscore en afsluiten gekozen kan worden | |  | | --- | | Pass/Fail |  |  | | --- | |  | |
| TC-002 | Het doolhof wordt succesvol gegenereerd | Spel starten via het menu | Het doolhof wordt goed weergegeven. | |  | | --- | | Pass/Fail |  |  | | --- | |  | |
| |  | | --- | | **TC-003** |  |  | | --- | |  | | |  | | --- | | Pacman beweegt in de juiste richting |  |  | | --- | |  | | Alle pijltoetsen na elkaar indrukken | De richting van de pijltoets is ook werkelijk de richting van pacman | |  | | --- | | Pass/Fail |  |  | | --- | |  | |
| |  | | --- | | **TC-004** |  |  | | --- | |  | | |  | | --- | | Pacman stopt bij botsing met muur |  |  | | --- | |  | | In de richting van een muur bewegen | Pacman stopt tegen een muur als dat zo is | |  | | --- | | Pass/Fail |  |  | | --- | |  | |
| |  | | --- | | **TC-005** |  |  | | --- | |  | | |  | | --- | | Pacman eet een punt bij aanraking |  |  | | --- | |  | | In de richting van een punt bewegen | Pacman eet een punt bij aanraking | |  | | --- | | Pass/Fail |  |  | | --- | |  | |
| |  | | --- | | **TC-006** |  |  | | --- | |  | | |  | | --- | | Spel eindigt wanneer Pacman door spook wordt geraakt |  |  | | --- | |  | | Zodanig bewegen met pacman dat het in contact komt met een spook | Huidige speelsessie eindigt in aanraking met een spook/mob | |  | | --- | | Pass/Fail |  |  | | --- | |  | |
| |  | | --- | | **TC-007** |  |  | | --- | |  | | Spook kan niet door een muur heen | Tijdens een speel sessie controleren of de mobs/spoken buiten of in de muren kunnen gaan | De mobs blijven tussen de muren | |  | | --- | | Pass/Fail |  |  | | --- | |  | |
| |  | | --- | | **TC-008** |  |  | | --- | |  | | Score wordt correct bijgehouden | Door bij te houden of het punten aantal toeneemt, als een punt wordt gegeten door pacman | De score wordt correct bijgehouden | |  | | --- | | Pass/Fail |  |  | | --- | |  | |
| |  | | --- | | **TC-009** | | Game gaat succesvol naar het volgende level na het behalen | Door te controleren of het spel, de speler neemt naar het volgende level wanneer alle punten verzameld zijn | Game gaat succesvol naar het volgende level na het behalen | Pass/Fail |
| |  | | --- | | **TC-0010** | | Speler kan succesvol door het menu navigeren | Door met de pijltoetsen en enter te checken of het mogelijk is | Speler kan succesvol door het menu navigeren | Pass/Fail |
| |  | | --- | | **TC-011** | | Highscores worden keurig gerangschikt | Door te navigeren naar de highscores, nadat de highscores zijn opgeslagen | Highscores worden keurig gerangschikt | Pass/Fail |
| TC-012 | Na elke gamesessie worden de naam en highscore van de speler opgeslagen en in de highscore list gezet met datum. | Door te checken of het naam in de input overeenkomt met de naam in de highscore list, gevolgd door de highscore en datum | Na elke gamesessie worden de naam en highscore van de speler opgeslagen en in de highscore list gezet met datum. | Pass/Fail |
| TC-013 | Spook en pacman kunnen teleporteren van de ene kant naar de andere kant van het veld als daar een opening voor is | Door een opening te creeren daar door heen te gaan en te controleren of pacman werkelijk naar de andere kant wordt geteleporteerd | Spook en pacman kunnen teleporteren van de ene kant naar de andere kant van het veld als daar een opening voor is | Pass/Fail |
| TC-014 | De highscores worden opgeslagen en kunnen altijd bekeken worden ook na het afsluiten van het spel | De highscores bekijken na een sessie, het spel afsluiten en weer controleren of dezelfde highscores te zien zijn na het opstarten van het spel | De highscores worden opgeslagen en kunnen altijd bekeken worden ook na het afsluiten van het spel | Pass/Fail |

---

# 6. Planning en Implementatie

## 6.1 Mijlpalen en Leverdata

Zie de belangrijke mijlpalen en bijbehorende leverdata in de onderstaande tabel weergegeven:

|  |  |  |
| --- | --- | --- |
| Mijlpaal | Beschrijving | Leverdata |
| |  | | --- | | **Mijlpaal 1: Projectinitiatief** |  |  | | --- | |  | | |  | | --- | | Definitieve planning en specificaties van basisfunctionaliteiten (beweging, punten verzamelen, geesten, spelregels). |  |  | | --- | |  | | 06/09/2024 |
| |  | | --- | | **Mijlpaal 2: Basisbewegingsmechanica** |  |  | | --- | |  | | |  | | --- | | Implementatie van Pacman's beweging en botsingsdetectie met muren en doolhofomranding. |  |  | | --- | |  | | 13/09/2024 |
| |  | | --- | | **Mijlpaal 3: Doolhofontwerp** |  |  | | --- | |  | | |  | | --- | | Ontwerp en bouw van het spelbord met vaste muren, op basis van het klassieke Pacman-doolhof, en plaatsing van punten. |  |  | | --- | |  | | 20/09/2024 |
| |  | | --- | | **Mijlpaal 4: Puntensysteem** |  |  | | --- | |  | | |  | | --- | | Implementatie van punten die Pacman kan verzamelen en een scoremechanisme dat punten bijhoudt. |  |  | | --- | |  | | 27/09/2024 |
| |  | | --- | | **Mijlpaal 5: Basisgeestgedrag** |  |  | | --- | |  | | |  | | --- | | Basisfunctionaliteit voor geesten, inclusief beweging door het doolhof en interactie (verliesconditie wanneer Pacman wordt geraakt). |  |  | | --- | |  | | 04/10/2024 |
| |  | | --- | | **Mijlpaal 6: Spelcondities** |  |  | | --- | |  | | |  | | --- | | Toevoegen van “Game Over” en “Win”-schermen, afhankelijk van verlies- of wincondities (alle punten verzameld). |  |  | | --- | |  | | 11/10/2024 |
| |  | | --- | | **Mijlpaal 7: UI en Menuscherm** |  |  | | --- | |  | | |  | | --- | | Implementatie van een startscherm, een menu dat bestaat uit start spel, highscores en afsluiten en mogelijkheid om opnieuw te spelen na een verlies. |  |  | | --- | |  | | 18/10/2024 |
| |  | | --- | | **Mijlpaal 8: Testen en Verbeteringen** |  |  | | --- | |  | | |  | | --- | | Uitvoeren van acceptatietests en oplossen van bugs; verfijnen van bewegingslogica en visuele elementen op basis van testresultaten. |  |  | | --- | |  | | 25/10/2024 |
| |  | | --- | | **Mijlpaal 9: Eindoplevering** |  |  | | --- | |  | | |  | | --- | | Definitieve oplevering van de werkende Pacman-game, inclusief documentatie en alle ontwikkel- en testnotities. |  |  | | --- | |  | | 01/11/2024 |

## 6.2 Risicoanalyse

De mogelijke risico’s zijn bepaald in de tabel hieronder weergegeven, ook wordt er beschreven in die zelfde tabel hoe die behouden of verminderd kunnen worden.

|  |  |  |
| --- | --- | --- |
| Risico | Beschrijving | Behouden of verminderen |
| |  | | --- | | **Complexiteit van botsingsdetectie** |  |  | | --- | |  | | |  | | --- | | Pacman moet correct stoppen bij muren en reageren op geesten en punten. Fouten in botsingsdetectie kunnen leiden tot een onspeelbaar spel. |  |  | | --- | |  | | |  | | --- | | Door betrouwbare, geteste algoritmes voor botsingsdetectie te gebruiken. Ook door te beginnen met eenvoudige botsingstests en deze geleidelijk uit te breiden. En tot slot deze functionaliteit grondig te testen met verschillende test zoals een integratietest. |  |  | | --- | |  | |
| |  | | --- | | **Onvoorspelbaar gedrag van geesten** |  |  | | --- | |  | | |  | | --- | | Als geesten niet correct bewegen of voorspelbare patronen volgen, kan het spel te moeilijk of te gemakkelijk worden. |  |  | | --- | |  | | |  | | --- | | Door een test te ontwikkelen op basis van AI voor de geesten met gecontroleerde bewegingen. En de moeilijkheidsgraad te testen en patronen aan te passen op basis van feedback. |  |  | | --- | |  | |
| |  | | --- | | **Prestatieproblemen bij complexe bewegingen** |  |  | | --- | |  | | |  | | --- | | Bij toenemende objecten of animaties kan het spel traag worden, vooral op minder krachtige apparaten. |  |  | | --- | |  | | |  | | --- | | Door de code te optimaliseren, de zware grafische elementen te minimaliseren, en de prestatie op apparaten met verschillende verwerkingscapaciteiten te testen. |  |  | | --- | |  | |
| |  |  |  | | --- | --- | --- | | |  | | --- | | **Spelcondities werken niet zoals verwacht** |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | | Win- en verliescondities moeten foutloos functioneren. Fouten hierin kunnen de speelbaarheid beïnvloeden. |  |  | | --- | |  | | |  | | --- | | Door Definieer en test specifieke scenario’s te definiëren en te testen voor elke spelconditie (bijvoorbeeld botsing met een spook, alle punten verzamelen). En Acceptatietest uitvoeren om te controleren of deze condities consistent functioneren. |  |  | | --- | |  | |
| |  |  |  | | --- | --- | --- | | |  | | --- | | **Onvoldoende testdekking** |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | | Gebrek aan tests kan ervoor zorgen dat bugs in productie terechtkomen. |  |  | | --- | |  | | |  | | --- | | Door het ontwikkelen van een teststrategie met duidelijke testcases en uitvoeren van integratie-, en acceptatietests uit. Het door documenteren van tests en resultaten, en het herhalen van tests na elke belangrijke update. |  |  | | --- | |  | |
| |  |  |  | | --- | --- | --- | | |  | | --- | | **Slechte documentatie en overdraagbaarheid** |  |  | | --- | |  | |  |  | | --- | |  | |  | |  | |  | | |  | | --- | | Slechte documentatie kan het moeilijk maken om het project later aan te passen of door te geven aan een andere ontwikkelaar. |  |  | | --- | |  | | Door behoud van code gedocumenteerd en duidelijke geschreven projectdocumentatie met uitleg van logica, functies en eventuele installatie-instructies. |