SL2A & ADP document  
Joel, Rik & Emma

Sprint documentatie:   
  
Wij hebben als groepje 4 (0-3) sprints gedaan om tot het eindproduct te komen. Hier hebben wij per sprint een aantal “issues” in de sprintplanning mee gemaakt.

**Sprint 0:**

In de eerste sprint zijn wij begonnen met brainstormen. Hieruit is gekomen dat wij de game PONG willen maken, welke toetsen gebruikt zullen worden als je de game speelt.   
Hiervoor was het nodig om de volgende opdrachten te maken:  
User stories, Product backlog, Sprint backlog, Design, Wireframes verder uitwerken.  
  
In deze sprint waren weinig “issues” ontstaan aangezien het enkel het bedenken van een plan was.  
  
**Sprint 1:**

In deze sprint zijn de opdrachten die wij hadden gepland in de vorige sprint uitgevoerd. Zo hebben wij de volgende dingen uitgewerkt:  
User stories, Product backlog, Sprint backlog, Design, Wireframes verder uitwerken.  
A black rectangular object with a dotted line

Description automatically generatedA screen shot of a game

Description automatically generated

A screen shot of a game

Description automatically generatedA black rectangular object with white text

Description automatically generatedA grey rectangular object with text

Description automatically generated

In deze sprint was er ook nog geen spraken van veel “issues”. De enige “issue” was dat we niet een hele strakke planning hadden gemaakt voor de eerst volgende sprint.  
  
**Sprint 2:**  
  
Bij deze sprint is de sprint reviews met de andere groepjes gemaakt.  
En de rollen verdeeld binnen ons groepje, wie wat gaat oppakken.  
  
De “issue” van deze sprint is voornamelijk dat we ons niet volledig hebben gefocust op dit project aangezien wij hiernaast nog een project hebben lopen.  
Doordat de deadline van onze game PONG ook in de buurt komt is dit niet ideaal en zouden wij dat in de laatste sprint moeten oplossen.  
  
**Sprint 3:**  
  
Deze sprint is belangrijk omdat het einde inzicht komt en er nog een aantal stappen ondernomen moeten worden.  
In deze sprint hebben we gefocust op het daadwerkelijk coderen van het spel, en de hiervoor nodige taken onder elkaar verder te verdelen. Op deze manier zullen we het eindproduct volledig afmaken en inleveren.  
  
“Issues” die bij deze sprint zijn voorgekomen waren dat wij niet precies wisten hoe groot de opdracht was en dus ook niet precies wisten hoeveel tijd we hiervoor kwijt zouden zijn.

User stories

Start pagina

Als speler wil ik een start pagina zodat ik niet meteen begin met spelen

Criteria: Er moet een begin pagina zijn met uitleg hoe je de game start en een knop voor info

=============================

Info pagina

Als speler wil ik een info pagina zodat ik kan zien met welke knoppen ik moet spelen

Criteria: Er moet een pagina zijn waarop staat welke knoppen speler 1 & 2 gebruiken om de paddles te bewegen

=============================

Game

Als speler wil ik pong kunnen spelen in de console zodat ik dit makkelijk op mijn computer kan spelen

Criteria: Console app geschreven in C#

=============================

Scoreboard

Als gebruiker wil ik een score bord zodat ik mijn score kan zien

Criteria: Scoreboard met de score van beide spelers

=============================

Code in aparte bestanden

Als developer wil ik dat alle aparte klassen in een eigen bestand staan zodat ik deze makkelijk kan vinden

Criteria: Elke klasse in een eigen bestand met de klasse naam als bestandsnaam

=============================

Clean code

Als developer wil ik duidelijk leesbare code zodat ik de code makkelijk kan begrijpen en aanpassen

Criteria: Er is gebruik gemaakt van KISS, DRY & YAGNI

=============================

Comments

Als developer wil ik duidelijke comments zodat ik snel snap wat en waarom iets iets doet

Criteria: Bij (bijna) elke regel een comment plaatsen

Backlog items

Backlog 1: Start pagina

Userstory: Als speler wil ik een startpagina zodat ik niet meteen begin met spelen.

Backlog:

Er moet een beginpagina zijn met uitleg over hoe de game te starten.

Een knop voor meer informatie moet beschikbaar zijn.

Backlog 2: Info pagina

Userstory: Als speler wil ik een infopagina zodat ik kan zien met welke knoppen ik moet spelen.

Backlog:

Een pagina met instructies over welke knoppen speler 1 & 2 gebruiken om de paddles te bewegen.

Backlog 3: Game

Userstory: Als speler wil ik Pong kunnen spelen in de console zodat ik dit makkelijk op mijn computer kan spelen.

Backlog:

Console-applicatie geschreven in C#.

Backlog 4: Scoreboard

Userstory: Als gebruiker wil ik een scorebord zodat ik mijn score kan zien.

Backlog:

Scorebord met de score van beide spelers.

Backlog 5: Code in aparte bestanden

Userstory: Als developer wil ik dat alle aparte klassen in een eigen bestand staan zodat ik deze makkelijk kan vinden.

Backlog:

Elke klasse in een eigen bestand met de klassenaam als bestandsnaam.

Backlog 6: Clean code

Userstory: Als developer wil ik duidelijk leesbare code zodat ik de code makkelijk kan begrijpen en aanpassen.

Backlog:

Gebruik van KISS (Keep It Simple, Stupid), DRY (Don't Repeat Yourself) & YAGNI (You Ain't Gonna Need It).

Backlog 7: Comments

Userstory: Als developer wil ik duidelijke comments zodat ik snel snap wat en waarom iets iets doet.

Backlog:

Bij (bijna) elke regel een comment plaatsen.  
Backlog 8: Game Attributes and Settings

Userstory: Als developer wil ik alle attributen van het Pong-spel op de juiste manier maken.

Backlog:

Definieer en implementeer attributen zoals de grootte van het speelveld, snelheid van de paddles en bal, scorelimieten, scoreboard en punten.

Teamreflectie:   
Onze reis door het C#-project was een leerzaam avontuur. Gelukkig verliep onze samenwerking helemaal goed, waarbij we open communiceerden en elkaar konden helpen waar nodig was.

In de eerste sprints lag de focus op het opstellen van een plan en documentatie. Hoewel we weinig problemen ondervonden, beseften we achteraf dat we wel te weinig tijd hadden besteed aan het maken van ons plan voor de volgende sprints en het daadwerkelijk beginnen met de game.

Later werden we afgeleid door andere projecten, waardoor onze focus op de game iets achterliep. Dit zorgde voor dat we wat minder werk gingen leveren aan dit project.

Ondanks deze opstakels hebben wij alsnog ons eindproduct optijd kunnen afmaken. We hebben geleerd dat een goede verdeling over de sprints belangrijk is voor succes en zullen dit meenemen naar mogelijke volgende projecten.

**1. Functionele Beschrijving**

De Pong game is ontworpen om een eenvoudige game-ervaring te bieden voor gebruikers. Het lijkt op het echte pingpongspel waarbij spelers peddels kunnen gebruiken om een bal heen en weer over de tafel te slaan.

**Errors:** Mochten er errors in de game onstaan tijdens het spelen, zal de game opnieuw beginnen

**Opstartprocedures:** Zodra het spel opstart moeten de spelers kiezen wie welke toetsen gaat besturen en dan kunnen de spelers de game starten.

**Gebruikersbeperkingen:** De game staat toe dat twee spelers tegelijk deelnemen en hun eigen toetsen kunnen gebruiken om de peddels de besturen.

**2. Gebruikersinterface**

De gebruikersinterface bestaat uit grafische elementen die het pong speelveld, rackets, bal en scorebord voorstellen(deze komt boven in het scherm te staan). De interface zal worden gemaakt om te lijken op een normale pingpongtafel om ping pong na te bootsen.

**Wireframe Diagram:**

A black rectangular object with a dotted line

Description automatically generated A screen shot of a game

Description automatically generated

**3. Lean Methodologie**

Om low-fidelity wireframes voor de gebruikersinterface te maken, zullen het maken volgens de opdracht.

**4. Doelen en Mijlpalen**

**Doel:** Bied een simpele multiplayer game-ervaring die lijkt op het klassieke pingpong spel.

**Mijlpalen:**

* Maken van grafische elementen (peddels, bal, scoreboard).
* Laten lijken op klassiek pingpong (balbeweging, peddelbediening).
* Maken van multiplayer-functionaliteit.
* Laatste test- en debugfase voor het einde.

**5. Prioritisering**

Het belangrijkste van het spel is dat het gebruiksvriendelijk is, en duidelijk is.

**6. Huidige en Voorgestelde Oplossingen**

**Huidige Oplossing:** Klassiek pingpongspel met fysieke materialen.

**Voorgestelde Oplossing:** Virtueel pingpongspel dat toegankelijkheid en gemak biedt voor spelers. De rechtvaardiging omvat gemakkelijke toegang, mogelijkheid om op afstand te spelen en potentieel voor aanvullende functies zoals aanpasbare instellingen en online multiplayer.

**7. Tijdlijn**

De tijdlijn zal specifieke taken, deadlines en toegewezen teamleden voor elke fase van ontwikkeling bevatten, inclusief codering, testen en implementatie.

## UML

Een methode genaamd UML wordt gebruikt om ingewikkelde systemen te ontleden en duidelijk te beschrijven. Met UML kunnen we de structuur en het gedrag van software begrijpen en inzichtelijk maken door middel van visuele modellen.

* Klassendiagrammen: Deze diagrammen onthullen hoe klassen binnen een softwareprogramma met elkaar zijn verbonden en hoe ze samenwerken.
* Use Case Diagram: Dit type diagram biedt een visuele representatie van hoe gebruikers met een systeem kunnen omgaan. Het toont de functionaliteiten van het systeem en hoe gebruikers deze kunnen benutten.
* Activity Diagram: Een stroomschema dat de stappen en taken in een proces illustreert, weergegeven als afgeronde rechthoeken.
* Sequence Diagram: Dit diagram visualiseert hoe objecten binnen een systeem met elkaar communiceren en toont de interacties tussen deze objecten in de volgorde waarin ze plaatsvinden. Het biedt inzicht in welke objecten betrokken zijn, de sequentie van acties en de uitgewisselde informatie.
* State Machine Diagram: Ook bekend als toestandsdiagram, laat dit type diagram de verschillende toestanden en overgangen zien waarin objecten zich kunnen bevinden binnen een systeem.

UML is een krachtig hulpmiddel dat wordt gebruikt voor het verbeteren van het ontwerp en de ontwikkeling van software. Het stelt ons in staat om complexe systemen helder en gestructureerd te begrijpen en te communiceren. Door middel van verschillende diagrammen kan UML de relaties tussen objecten en klassen visualiseren, wat essentieel is voor een effectieve ontwikkeling en communicatie binnen teams, evenals tussen klanten en ontwikkelaars.

|  |
| --- |
| Game |
| * gameLock: object * border: Border * paddle1: Paddle * paddle2: Paddle * scoreboard: Scoreboard * ball: Ball * gameRunning: bool |
| + Start(): async Task  + Ball(): async Task  + AskAgain(): void  - InitializeGame(): void |

|  |
| --- |
| Ball |
| * velocityX: int * velocityY: int |
| + Move(): voids  + Reset(): void  + CheckBorderCollision(maxWidth: int, maxHeight: int, scoreboard: Scoreboard): void  + CheckPaddleCollision(paddle: Paddle): void |

|  |
| --- |
| Paddle |
| * upKey: ConsoleKey * downKey: ConsoleKey * Length: int * TopBoundary: int * BottomBoundary: int |
| + MoveUp(): void  + MoveDown(): void  + HandleInput(): void |

|  |
| --- |
| Asset |
| * gameLock: object * assetImage: string[] * X: int * Y: int |
| + Draw(input: String): void  + Draw(x: int = 0, y: int = 0): void  +Remove(): void |

|  |
| --- |
| Scoreboard |
| * player1Score : int * player2Score : int * pointsToWin: int |
| + IncrementPlayer1Score(): void  + IncrementPlayer2Score(): void  + Player1Wins(): bool  + Player2Wins(): bool  + Win(player: int): void |

|  |
| --- |
| Home |
| + Show(): void  + ShowInfo(): void  + ReadOption(): async void |

A diagram of a computer

Description automatically generated with medium confidence