Wandering in the Woods Game: Software Design Document and User’s Guide  
Jagadish Sagi with Meghana Priya Lakamana, Navadeep Gottigundala and Vijesh Medikonda

Table of Contents

[1.0 Introduction 3](#_Toc174631278)

[1.1 Purpose 3](#_Toc174631279)

[1.2 Scope 3](#_Toc174631280)

[1.3 Intended Audience 3](#_Toc174631281)

[2.0 Process Model 3](#_Toc174631282)

[3.0 Use Cases 4](#_Toc174631283)

[3.1 Use Case 1: Start Game 4](#_Toc174631284)

[3.2 Use Case 2: Configure Game Settings (3-5 Level) 4](#_Toc174631285)

[3.3 Use Case 3: Play Game 5](#_Toc174631286)

[3.4 Use Case 4: Display Statistics 5](#_Toc174631287)

[3.5 Use Case 5: Replay Game 5](#_Toc174631288)

[3.6 Use Case 6: Exit Game 6](#_Toc174631289)

[4.0 Functional Requirements 6](#_Toc174631290)

[5.0 UML Model 6](#_Toc174631291)

[5.1 Use Case Diagram 6](#_Toc174631292)

[5.2 Class Diagram 7](#_Toc174631293)

[5.3 State Diagram 10](#_Toc174631294)

[5.4 Activity Diagram 12](#_Toc174631295)

[6.0 User’s Guide 14](#_Toc174631296)

[6.1 Installation 14](#_Toc174631297)

[6.2 Getting Started 14](#_Toc174631298)

[6.3 Game setup 15](#_Toc174631299)

[6.4 Playing the Game 17](#_Toc174631300)

[6.5 Viewing Game Statistics 19](#_Toc174631301)

[6.6 Exiting the Game 20](#_Toc174631302)

# 1.0 Introduction

## 1.1 Purpose

The Wandering in the Woods Game is designed to teach children in K-5 school grades fundamental computer science principles, computational thinking, and elementary mathematics with an immersive and interactive GUI-based application. They are designed to help students better internalise skill learning through a fun and intuitive experience that promotes collaborative learning.

## 1.2 Scope

The scope of this is project is the development of an educational simulation on the desktop with a graphical user interface (GUI). The interface is divided into two grade levels: K-2 and 3-5. The K-2 level is to learn the basics of running and convergence on a simple grid. The 3-5 level learns scientific problem solving with variable grid lines and extra simultaneous players. On the K-2 level, the space is reinforced by visual and auditory feedback with a statistical recording tracker.

## 1.3 Intended Audience

The game is primarily targeted at K-5 students as a way to improve their understanding of computational and mathematical concepts, although educators or parents may also use it as a way to support collaborative learning in a classroom or home-based environment.

# 2.0 Process Model

We chose this process model – called the Evolutionary Process Model as shown in Figure 1 – because the process is iterative and adaptable. This meant that we could develop the 'Wandering in the Woods Game' in stages, with continuous feedback and iteration, adjusting to changes, and tailoring each iteration to the results of testing, improving the game and adding value in an effective and user-focused way.

A screenshot of a computer screen

Description automatically generated

Figure 1: Wandering in the Woods Game Evolutionary Process Model

# 3.0 Use Cases

## 3.1 Use Case 1: Start Game

|  |  |
| --- | --- |
| **Primary Actor:** | Player |
| **Preconditions:** | 1. The game application is open and running. 2. The user is on the main menu screen. |
| **Description:** | The user selects the option to start a new game from the main menu. The system then initializes the game based on the selected grade level (K-2 or 3-5) and displays the game screen, ready for the user to play. |
| **Acceptance Criteria:** | 1. The user can start a new game from the main menu. 2. Depending on the grade level that was chosen, the game initializes and shows the relevant game screen. 3. The game is ready for the user to interact with. |

## 3.2 Use Case 2: Configure Game Settings (3-5 Level)

|  |  |
| --- | --- |
| **Primary Actor:** | Player |
| **Preconditions:** | The user is in the 3-5 level game setup screen. |
| **Description:** | The user sets the grid dimensions and places characters on the grid. The inputs are checked by the system to make sure they don't overlap and are within allowable parameters. After configuration is finished, the user validates the configuration, and the game is launched with the selected parameters. |
| **Acceptance Criteria:** | 1. Character positions and grid dimensions can be entered by the user. 2. The system verifies that inputs are correct and within bounds. 3. The game initializes with the configured settings. 4. The game screen is displayed with the specified grid and characters. |

## 3.3 Use Case 3: Play Game

|  |  |
| --- | --- |
| **Primary Actor:** | Player |
| **Preconditions:** | The game is initialized and characters are placed on the grid. |
| **Description:** | Characters move on the grid based in a random fashion until they meet. The system updates the game state continuously. Once all characters meet, the game concludes, and the system prepares to display the game statistics. |
| **Acceptance Criteria:** | 1. Characters move and interact as expected according to the game logic. 2. The system recognizes when all characters have been met accurately. 3. The game ends properly after the meeting condition is fulfilled. |

## 3.4 Use Case 4: Display Statistics

|  |  |
| --- | --- |
| **Primary Actor:** | Player |
| **Preconditions:** | The game has ended, and the system is ready to present the results. |
| **Description:** | The system determines a number of statistics once the game is over, including the total number of moves, longest run without meeting, shortest run, and average run. An audio announcement is made to present the results, and these statistics are shown on the screen. |
| **Acceptance Criteria:** | 1. Game statistics are computed and shown by the system with accuracy. 2. The statistics are presented visually on the screen. 3. The statistics are announced audibly by the system. 4. The user can view all relevant statistics clearly. |

## 3.5 Use Case 5: Replay Game

|  |  |
| --- | --- |
| **Primary Actor:** | Player |
| **Preconditions:** | The game has ended. |
| **Description:** | The user selects the option to replay the game. If the user selects replay the same game, the system restarts the game with the same positions of the players, otherwise the system resets the game state, clears the previous data, and returns to the initial setup screen, allowing the user to start a new game session with fresh settings. |
| **Acceptance Criteria:** | 1. The system replays the game with the same grid and player coordinates when the user wants to replay the same game. 2. The system resets the game state and clears previous game data when the user selects to restart the game. 3. The user is returned to the initial setup screen for a new game session. 4. The game initializes correctly for the new session. |

## 3.6 Use Case 6: Exit Game

|  |  |
| --- | --- |
| **Primary Actor:** | Player |
| **Preconditions:** | The user is on any screen of the application or in the middle of a game session. |
| **Description:** | The user selects the option to exit the application at any point in the Game either by the Close Button on the Window or by clicking Exit Game after the game has ended. |
| **Acceptance Criteria:** | 1. The application closes properly and terminates the session. 2. The user is no longer able to interact with the application once it is closed. |

# 4.0 Functional Requirements

1. Users should be able to launch a new game, choose between the K-2 and 3-5 levels, and configure game settings including the number of participants, grid size, and starting locations for the 3-5 level.
2. The system must enable players to move within the grid, update their positions, and manage interactions such as merging when players meet.
3. The system should track and display game statistics, including the longest run without meeting, shortest run, and average run, at the end of each session.
4. The system must allow users to quit the game from any state and restart the game with the same or new settings.

# 5.0 UML Model

## 5.1 Use Case Diagram

All 6 use cases are reflected in the use case diagram as shown in Figure 2. The only actor in this diagram is a player who interacts with the “Wandering in the Woods Game” to engage in the corresponding activities. Player can start the game, set up the game settings, play the game, and observe the application through the game statistics. Player can also restart the game or quit the application whenever he/she wants. Each of these respective use cases will grant the user with the level of control and settings of the ‘Wandering in the Woods Game’, and consequently make it an intuitive activity for the group of the target age.

A diagram of a person with blue circles

Description automatically generated

Figure 2: Use case Diagram for Wandering in the Woods Game

## 5.2 Class Diagram

The Class Diagram for the Wandering in the Woods Game is shown in Figure 3 below. The classes in the diagram are described below the figure.

A diagram of a computer program

Description automatically generated

Figure 3: Class Diagram for Wandering in the Woods Game

**GameLauncher:** This is the main class that displays the Start Menu and calls the relevant classes to launch the respective Game Levels.

* **show\_rules():** Function to display the rules in a separate window when user clicks on the Text: “Click here to view game rules”
* **handle\_start\_button\_click(value, root):** Function to start the respective game selected by the User from the drop down menu after the user clicks on Start Game.
* **launch\_game():** Main Function that displays the start menu to the User and makes the relevant function calls to show\_rules() and handle\_start\_button\_click() to show the rules and start the game respectively.

**AbstractGameLauncher:** Abstract Class for handling the Launch of different levels of game like K-2, 3-5 respectively.

* **launch\_game():** Abstract Method to be implemented by Children classes to handle the Logic to launch the game.

**WanderingGameKto2Launcher:** Child Class inheriting the AbstractGameLauncher class to launch the Game for K-2 Students.

* **launch\_game():** Method to launch the game for K-2 students.

**WanderingGame3to5Launcher:** Child Class inheriting the AbstractGameLauncher class to launch the Game for 3-5 Students.

* **launch\_game():** Method to launch the game for 3-5 students.

**AbstractWanderingGame:** Abstract Class to Run the Game for the respective classes of students.

* **create\_grid():** Method to create the Grid on which the players move.
* **run\_game():** Method containing the logic to run the game.
* **show\_statistics():** Method to display the statistics to the Users once the game is completed.

**WanderingGameKto2:** Child Class of AbstractWanderingGame Class to run the game for K-2 Students.

* **create\_grid():** Method to create the Square Grid on which the players move.
* **run\_game():** Method containing the logic to run the game for K-2 students.
* **show\_statistics():** Method to display the statistics to the Users once the game is completed.
* **check\_if\_together():** Method to check if players have met on the grid.
* **move\_players():** Method to handle the logic of moving players randomly on the grid.
* **announce\_success():** Method to announcing the Success Message that players have met along with the statistics of the game to it’s Users.

**WanderingGame3to5:** Child Class of AbstractWanderingGame class to run the game for 3-5 Students.

* **create\_grid():** Method to create the rectangular grid on which the Players move till they meet.
* **run\_game():** Method containing the logic to run the game.
* **show\_statistics():** Method to display the statistics to the Users once the game is completed.
* **check\_if\_together():** Method to check if two players have met on the grid.
* **move\_players():** Method containing the logic to move the players randomly on the grid.
* **update\_statistics():** Method to update the statistics once 2 players have met on the grid.
* **reset\_game():** Method to reset the game so that it can be replayed or a new game is started.
* **close\_game():** Method containing the logic to properly close the game at any point of time.
* **replay\_game():** Method containing the logic to replay the game with the same coordinates.
* **start\_new\_game():** Method containing the logic to start a new game.

**Player:** Class holding the information and methods related to the Player.

* **create\_avatar():** Method to create the avatar for the player on the grid.
* **get\_valid\_moves():** Method to get the set of valid moves for the player depending on their position on the grid.
* **move():** Method to make a move for the player on the grid.
* **get\_position():** Method to get the current position of the player on the grid which is useful in determining whether two players have met on the grid or not.

## 5.3 State Diagram

The state diagram shown below in Figure 4 illustrates the high-level behaviour for the 'Wandering in the Woods' game. It represents the four main individual states of the system in a typical operation.

This diagram shows how the system operates, starting from the Start Screen state, where the game starts. On the Start Screen state the user can quit the app or start a new game. If starting a new game, the system will go to the Play Game state. On the Play Game state, the game is being played. While the game is being played, the players are taking turns moving around the grid until the game ends. In the Show Statistics state, the system shows the result of the game on the screen, like the number of moves took to finish the game, and other statistic.

Based on the show statistics state, your user has the option to either restart the game, which moves the system back to the Play Game state, or to quit the application, which lands our final state of the application quitting. Note that also the user has the option to exit directly from the start screen or play game states, going directly to the quit state.

A screen shot of a game

Description automatically generated

Figure 4: State Diagram for Wandering in the Woods Game

## 5.4 Activity Diagram

Figure 5 represents the activity diagram that describes the high-level behaviour of the Wandering in the Woods game.

When the user launches the game, they have to make a choice between the K-2 or the 3-5 level. If the K-2 level is chosen the system responds by initialising a simple square grid with the size randomly selected. Otherwise, if the user decides to play at the level 3-5, player has to configure the game settings, such as the grid size and the coordinates of the players.

When the game parameters are set, it splits into two branches the grid is initialized and the players are set at the respective locations and they start moving randomly till they all meet. If there are more than two players, when two players, they merge together and start moving as if they were a Single Player.

When all players have met, the system shows the statistics to the Users such as the Longest Run, Average Run, etc. and the system offers the player to replay the game or quit. If the user decides to replay the game, and the Level is K-2, the system replays the game by randomly selecting the grid size, else the system asks whether the player wants to replay the game with same coordinates or start a new game and if the player wants to play with the same coordinates, the systems restarts the game with the same coordinates else it starts again asking the User for the dimensions of the grid and the number of players.

A screenshot of a black screen

Description automatically generated

Figure 5: Wandering in the Woods Game Activity Diagram

# 6.0 User’s Guide

## 6.1 Installation

For Installing the Wandering in the Woods game, from the Github repository provided: <https://github.com/Jags8421/Wandering-In-The-Woods-Game> , download the executables zip file, which contains the following files: “game\_launcher.exe” file and “happy-and-joyful-children.wav”. Both files are required and be present in the same folder for successfully launching the game.

Once the zip file is downloaded and extracted in the Local Computer, open the “game\_launcher.exe” file to launch the game. In case if you get any pop up from Windows defender or any other Anti-virus issuing a Warning, ignore it the file has no harmful content to affect your device.

## 6.2 Getting Started

Once the game is launched, you will see the Start Menu as shown in the Figure 6 below, with the options to select the Game Level. The Dropdown helps the players select the game level. K-2 represents the First Level and is for the students from Kindergarten to Grade 2. 3-5 represents the Second Level and is for the students from Grade 3 to 5. The User can also get to know about the rules of the game when they click on the ‘Click here to view game rules’ as shown in Figure 7 below.

A screenshot of a computer

Description automatically generated

Figure 6: Start Menu for the game when it is launched.

A screenshot of a game

Description automatically generated

Figure 7: Screenshot of the Window showing the Game Rules

## 6.3 Game setup

Once the user selects the Game Level and clicks on Start Game, Depending on the selected Game Level relevant action is taken. If the Selected Game Level is ‘K-2’ then the size of the Square Grid is randomly selected, and the two players positions are set to the ends of the Principal Diagonal.

When the user selects ’3-5’ Level to start the game, the user is provided with the following window as seen in the Figure 8 below to get the user inputs for the dimensions of the Rectangular Grid and the number of players, the number of rows should be at least 2 and at most 15 and the same constraints are applied for the columns. And 2-4 players are allowed per game. Failing to provide inputs in this range will result in a ValueError and user will be asked to provide the inputs again as shown in Figure 9 below. Once the user confirms the information, the user will be asked for the co-ordinates of the player position as shown in Figure 10 below. The co-ordinates are 1-indexed, and no 2 players are allowed to start from the same position, and if the user provides incorrect coordinate information like 2 players starting from the same position or players coordinate is out of bounds, then the User is asked to provide the coordinate information again as shown in Figure 11 below.

A screenshot of a computer

Description automatically generated

Figure 8: Screenshot showing the Window displayed to the User for 3-5 Grade Students

A screenshot of a computer

Description automatically generated

Figure 9: Screenshot showing Error window when invalid input is entered.

A screenshot of a computer

Description automatically generated

Figure 10: Screenshot of the Window where the User can enter the coordinates of players

A screenshot of a game

Description automatically generated

Figure 11: Screenshot showing the response of the Game, when the user enters invalid coordinates

## 6.4 Playing the Game

Once all the relevant required information is provided by the user, the game starts. In K-2 Level, the players start from the opposite ends of the principal diagonal and music starts playing as players move randomly in one of the four possible valid directions (up, down, left, right). They move randomly until both arrive in the same cell of the grid at the same time as shown in Figure 12 below. The players are represented as a Circle in the Cell with each player having a distinct colour of their own. When both the players merge, the colour of the Merged Player will be made ‘purple’ indicating that both the players have merged. When both the players are merged the game ends.

A screenshot of a game

Description automatically generated

Figure 12: Screenshot showing the Game played for K-2 Grade Students

For 3-5 Level, the rectangular grid is loaded onto screen with each player set at their respective positions as provided by the user and each player is represented as a Circle in the Cell with each player having a distinct colour. Like K-2 Level, the players move randomly in one of the 4 possible valid directions as shown in Figure 13, till they meet. When two players merge, they move together as a Merged player and the colour of the merged player will be unique to be able to distinguish them. Once all the players are merged, the game ends.

A screenshot of a game

Description automatically generated

Figure 13: Screenshot showing the Game midway for Grades: 3-5 Students

## 6.5 Viewing Game Statistics

Once all the players are merged, the game ends and a Statistics Window is displayed to the User, For K-2 Level, the statistics displayed are the Total moves made by the Players to meet. In the K-2 Level, these statistics are also announced to the User. The Statistics Window will also have the Option for the User to replay the game as shown in Figure 14 and when the user selects that option, the game is replayed by randomly selecting the size of Square Grid.

A screenshot of a video game

Description automatically generated

Figure 14: Screenshot showing the Statistics Window displayed for K-2 Grade Students

For 3-5 Level, the following statistics are displayed: “Longest Run Without Meeting”, “Shortest Run”, “Average Run” to the User as shown in Figure 15, along with this, the user is given the option to replay the game with the same coordinates or start a new game. When the user selects the button to replay the game with same coordinates, we will restart the game, with the exact number of players, grid dimensions and player coordinates that the user has already provided. However, when the User selects “Start a New Game”, we start from the beginning of 3-5 Level, asking the User for Number of players and Grid Dimensions.

A screenshot of a computer

Description automatically generated

Figure 15: Screenshot showing the statistics window for the game for 3-5 Grade Students

## 6.6 Exiting the Game

A User can exit the game at any point of time, by clicking on the Close Window Button or Close Window Button keyboard shortcut. Once the game is over and the statistics window is displayed, the user can click on Exit Game to close the Game.