Wandering in the Woods

Shawn Amato, Mahnoor Malik, Rebekah Myrick, Jacob Toth, and Analyn Pureza

Lewis University, Romeoville, IL

[shawnlamato@lewisu.edu](mailto:shawnlamato@lewisu.edu), [mahnoorfmalik@lewisu.edu](mailto:mahnoorfmalik@lewisu.edu), [rebekahmyrick@lewisu.edu](mailto:rebekahmyrick@lewisu.edu), [jacobtoth@lewisu.edu](mailto:jacobtoth@lewisu.edu), and [analynpureza@lewisu.edu](mailto:analynpureza@lewisu.edu)

1. **Purpose**

The purpose of the Wandering in the Woods Simulation is to teach K-8 students about computation, computational thinking, math concepts, and computer science in a fun and engaging way. The simulation/game is divided into three stages, each with increasing levels of complexity, targeting students from grades K-2, 3-5, and 6-8. The simulation aims to provide a consistent user experience across the different stages to facilitate the learning curve and build confidence in students. The simulation/game uses the concept of people being lost in the woods and needing to find each other, represented by a grid. By moving around the grid and finding each other, students will learn about statistics, spatial reasoning, and experimental design, among other concepts. The simulation/game is designed to be played by pairs of students working together in front of the same screen, with audio directions and prompts, and assessments to check how well the students are doing with the challenges.

## II. The Game

As the game progresses the first screen will be asking, please select the grade you want to play. It displays all the three levels that are K-2, grade 3-5 and grade 6-8 from which the user picks one of the levels to start. After he chooses the level or grade it will prompt to select the grid size to play, then the user selects the grid size and next the system asks for the number of players to choose from 2 to 4.

After a selection of several players the game starts and players move from different corners to find each other once they find each other the happy music plays and the game ends. Same way if they are playing with more than 2 players if anyone finds the second person, they will move along to find others. After the end of the game, it goes back to the first page so that it becomes easy to restart the game again.

1. **Requirements**
2. *User Requirements:*
3. Basic computer literacy skills
4. Basic math and computation knowledge
5. Ability to work in pairs and communicate effectively.
6. Willingness to learn and engage with the simulation/game.
7. *Software Requirements:*
8. The Wandering in the Woods Simulation software installed on the computer or laptop.
9. A compatible web browser (Chrome, Firefox, Safari)
10. An audio playback software (Windows Media Player, QuickTime)
11. *Hardware Requirements:*
12. A computer or laptop with a modern operating system (Windows, MacOS, Linux)
13. A graphics card capable of rendering 2D graphics
14. A monitor capable of displaying a minimum resolution of 1024x768 pixels.
15. Speakers or headphones to hear the audio directions and prompts.
16. *Performance Requirements:*
17. The simulation/game should run smoothly on the hardware and software requirements listed above.
18. The audio directions and prompts should be clear and audible.
19. The assessment system should accurately measure the students' performance and provide feedback in real-time.
20. The simulation/game should be able to handle different user inputs and configurations, such as grid size, number of players, and starting positions, without crashing or slowing down.

**III. Software Requirements:**

* ● Download and install PyCharm website (https://www.jetbrains.com/pycharm/).
* ● Click the "Download" button on the top right of the page.
* ● Select the edition of PyCharm that you want to download. There are two main editions:  
  the Community Edition, which is free and open-source, and the Professional Edition,  
  which is paid and includes additional features.
* ● Select the operating system that you are using (Windows, macOS, or Linux).
* ● Click the "Download" button for your selected edition and operating system. This will  
  start the download of the PyCharm installation file.
* ● Once the download is complete, locate the downloaded file and double-click it to start the installation process. Follow the prompts to install PyCharm on your system.
* ● Once the installation is complete, you can launch PyCharm from the start menu (on  
  Windows) or the Applications folder (on macOS).

**IV. Testing Strategy**

Unit Testing:

Unit testing is a type of testing that focuses on individual units or components of the

game, such as functions or classes. For the "Wandering in the Woods" game, unit tests could be written to test the functionality of the grid, players, and GUI, as well as the game logic and movement functions.

Integration Testing:

Integration testing is a type of testing that focuses on the interactions between different.

units or components of the game. For the "Wandering in the Woods" game, integration tests could be written to test the integration of the grid, players, and GUI with the game logic and movement functions.

Usability Testing:

Usability testing is a type of testing that focuses on the user experience of the game.

For the "Wandering in the Woods" game, usability tests could be conducted to ensure that the GUI is easy to use and understand, and that the game is enjoyable to play.

Validation Testing:

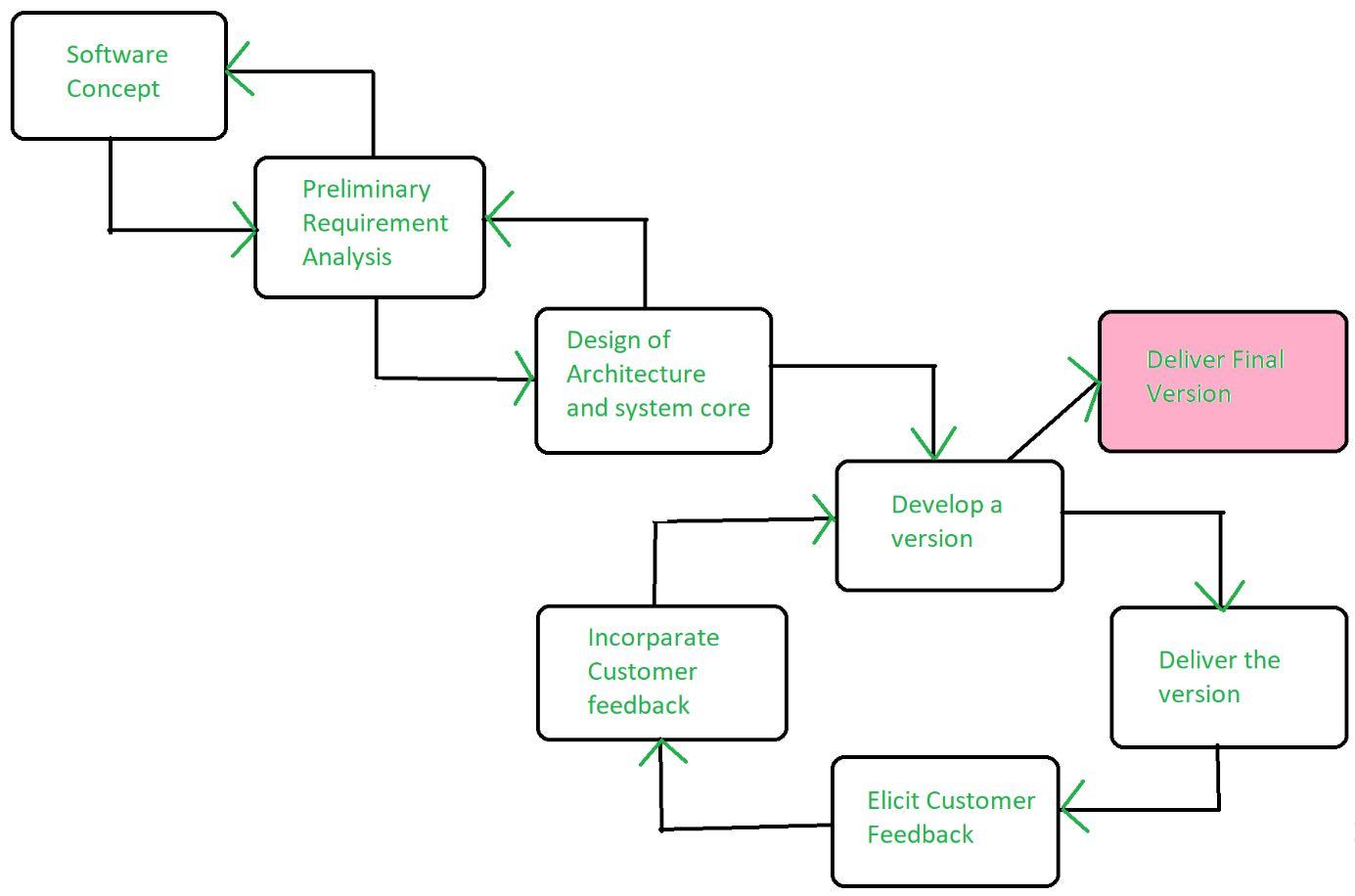
Validation testing is a type of testing that focuses on the accuracy and correctness of the game. For the "Wandering in the Woods" game, validation tests could be conducted to ensure that the game logic is correct and that the game behaves as expected under different scenarios.

# III. Process Model:

## Evolutionary Process Model:

Evolutionary model is a combination of Iterative and Incremental model of software development life cycle. Delivering your system in a big bang release, delivering it in incremental process over time is the action done in this model. Some initial requirements and architecture envisioning need to be done. It is better for software products that have their feature sets redefined during development because of user feedback and other factors.

The Evolutionary development model divides the development cycle into smaller, incremental waterfall models in which users can get access to the product at the end of each cycle. Feedback is provided by the users on the product for the planning stage of the next cycle and the development team responds, often by changing the product, plan or process. Therefore, the software product evolves with time. All the models have the disadvantage that the duration of time from start of the project to the delivery time of a solution is very high. Evolutionary model solves this problem in a different approach.



*Fig 1: Evolutionary Process Model*

## Principles of Agile Methodology:

* Customer satisfaction is our first focus, and we achieve this through timely and consistent delivery of high-quality software.
* Even late in the development process, accept changing needs. Agile methodologies harness change for the benefit of the customer's competitiveness.
* Deliver working software frequently, preferably in shorter time frames of a few weeks.
* Throughout the project, businesspeople and developers must collaborate every day.
* Build initiatives around motivated people. Trust them to complete the task and provide them with the environment and assistance they require.
* A face-to-face talk is the most effective and efficient way to communicate information to and within a development team.

The Game Process Model

The process model for the "Wandering in the Woods" game describes the steps involved in playing the game. Here is a summary of the process model for the game:

* The game starts with the players in diagonally opposite corners of the grid. The grid is displayed on the canvas in the GUI, and the players' symbols are placed on the grid to indicate their starting positions.
* Player 1 enters their move by clicking a button in the GUI and selecting a direction (up, down, left, or right). The game updates the grid to reflect the player's move and redraws the grid on the canvas.
* The game checks if player 1 has found player 2. If player 1 has found player 2, the game ends and a message is displayed on the GUI indicating that player 1 has won.
* If player 1 has not found player 2, it is player 2's turn. Player 2 enters their move in the same way as player 1. The game updates the grid and redraws it on the canvas.
* The game checks if player 2 has found player 1. If player 2 has found player 1, the game ends and a message is displayed on the GUI indicating that player 2 has won.
* If neither player has found the other, the game repeats steps 2 through 5 until one player has found the other.
* This process continues until a player has found other player, at which point the game ends.

**IV. Use Cases**

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. The method creates a document that describes all the steps taken by a user to complete an activity. The following use cases were defined by the requirements for the delivery of the game. Each use case has a list of the name of the use case, primary actors, preconditions, description, and acceptance criteria.

**Description**: A user creates a new game of "Wandering in the Woods" by starting the game and setting up the grid and GUI.

**Preconditions:** The game has been installed and the user has opened it.

**Main Flow:** The user clicks the "New game" button in the GUI. The game creates a new grid and displays it on the canvas. The game places the players' symbols on the grid to indicate their starting positions. The game displays the buttons and labels needed for the players to enter their moves and for the game to display messages.

**Postconditions:** A new game of "Wandering in the Woods" has been created and is ready for the players to start playing.

## V. Creating the main screen

**Primary Actor:** Student/Player/User

**Preconditions:** The game has been installed and the user has opened it.

**Description:** A user creates a new game of "Wandering in the Woods" by starting the game and setting up the grid and GUI.

The user clicks the "New game" button in the GUI. The game creates a new grid and displays it on the canvas. The game places the players' symbols on the grid to indicate their starting positions. The game displays the buttons and labels needed for the players to enter their moves and for the game to display messages.

**Acceptance Criteria:** I can choose which grade to play and with the number players to play along with the grid size

**Post Condition:**  A new game of "Wandering in the Woods" has been created and is ready for the players to start playing.

*Use Case 1: K-2 Students*

**Primary Actor:** K-2 Student Goal, Student/Player/User: Learn basic computation and math concepts through the Wandering in the Woods Simulation.

**Precondition**: The student has access to a computer with the simulation installed and some basic knowledge of counting where one selects the plotted K-2 grade block.

**Description:** As a player I want to be able to choose/ select a K2 grade block. I should

be able to click on the block to select the K2 grade. I should be able to play

with 2 players initially and after the game finishes a happy music should be

played and a deadly music should be played if the players are moving in the

opposite direction.

**Acceptance Criteria:** I can select and choose the players while choosing the grade blocks

**Steps**

1. A game has been designed where the one selects the plotted K-2 grade block.
2. The student starts the simulation and selects the appropriate stage for their grade level.
3. The student sets up the game by selecting the size of the square grid and the starting positions of the two players.
4. The student plays the game, watching the players move randomly around the grid.
5. When the two players bump into each other, the student sees the happy graphics display and hears the statistics announced audibly.
6. *Use Case 2: 3-5 Students*

**Primary Actor:** 3-5 Student Goa, Student/Player/User: Learn more advanced math and computation concepts through the Wandering in the Woods Simulation.

**Precondition:** The student has access to a computer with the simulation installed and has some understanding of basic math concepts. The game has been designed to hold the different grade blocks data, a K2 grade block has been created

**Acceptance Criteria:** I can be able to choose the number of players while choosing the grade blocks.

**Steps:**

1. The student starts the simulation and selects the appropriate stage for their grade level.
2. The student sets up the game by selecting the size of the rectangular grid and the starting positions of two, three, or four players.
3. The student plays the game, attempting to find all the players and improve the game statistics.
4. The student uses the statistics to analyze the data, looking for patterns and relationships.
5. The student runs experiments with different grid sizes and player configurations, recording and analyzing the data from each experiment.
6. *Use Case 3: 6-8 Students*

**Primary Actor:** 3-5 Student/User/Player

**Goal:** Learn more advanced math and computation concepts through the Wandering in the Woods Simulation

**Precondition:** The student has access to a computer with the simulation installed and has some understanding of basic math concepts. The game has been designed to hold the different grade blocks data. A K2 grade block and a K3-5 grade block has been created.

**Steps:**

1. The student starts the simulation and selects the appropriate stage for their grade level.
2. The student sets up the game by selecting the size and shape of the grid, the number of players, and their starting positions.
3. The student plays the game, attempting to find all the players and improve the game statistics.
4. The student runs experiments with different wandering protocols, recording and analyzing the data from each exploration.

Use case: Edit Game

**Primary Actor:** Player/ User/ Student

**Preconditions:** The game has been designed to hold the different grade blocks

data. A K2 grade block, K3-5 grade block and a K6-8 grade blocks ar created.

**Description:** As a user I want to be able to rechoose the blocks, the number of players, the grid size and the music that must be changed

**Acceptance Criteria:** I can make changes to the number of players and the grid size.

# VI. UML Model

Different levels of detail are represented by systems in Unified Modeling Language (UML) models. While other models provide more detail, others represent a system at a higher, more abstract level. UML models include one or more diagrams that depict a particular view of a system together with model features like actors, use cases, classes, and packages. A model may also include additional, more comprehensive models.

## Use Case Diagram

Use case diagrams are used to compile a system's requirements, considering both internal and external factors. Most of these needs are for the design. Therefore, use cases are created and actors are identified when a system is evaluated to gather its functionality.

Below figure.1 displays the use cases included in the use case diagram. The user is the only player who interfaces with the Game Designer System. The user can choose the number of players, and the size of the grid. Among the use cases the user can change the type of music, the players and the size of the grids based on the created grade blocks. Each of the created use cases can be extended by the Edit Game use case. This Edit game use case will help the user to modify any changes to the different use cases.

Diagram

Description automatically generated

*Fig 2: Use Case Diagram*

## Deployment Diagram

## 

A deployment diagram identifies the actual hardware that the software system will run on. Additionally, it controls the software's implementation on the underlying hardware. It associates system software components with the hardware that will run them.

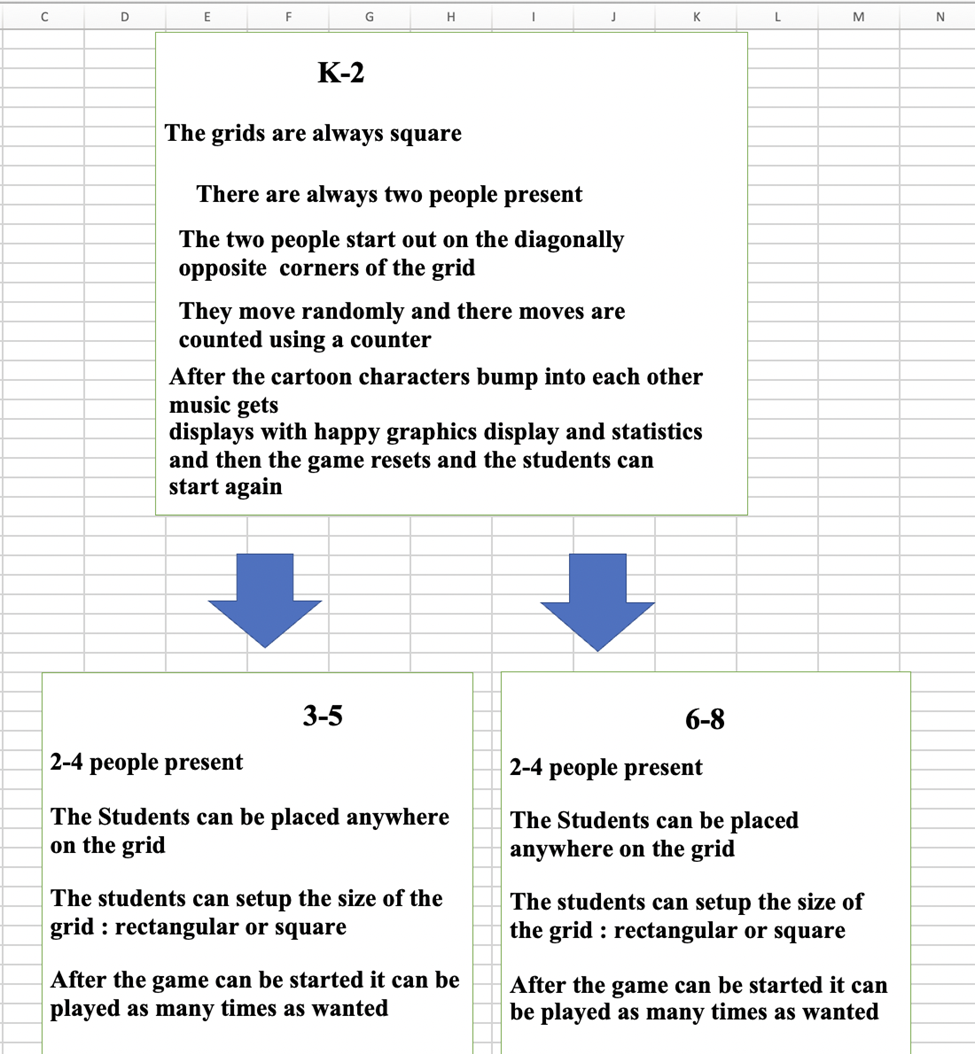
A picture containing diagram

Description automatically generated

*Fig 3: Deployment Diagram*

## Class Diagram

One of the most helpful forms of diagrams in UML are class diagrams, which accurately depict a system's structure by modeling its classes, properties, operations, and relationships among objects. Making these diagrams is easier than it would seem with the help of our UML diagramming tools. You will learn how to comprehend, organize, and produce your own class diagrams using this manual.

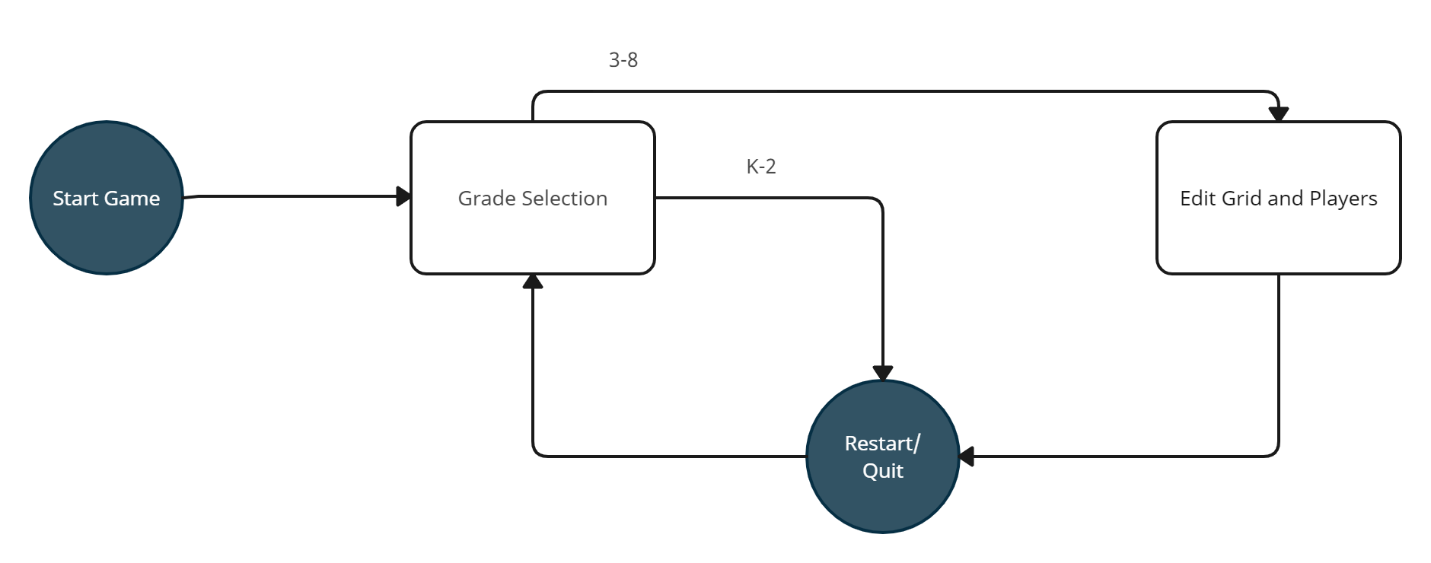


*Fig 4: Class Diagram*

## State Diagram

This state diagram shows the different states that the "Wandering in the Woods" game can be in, as well as the events that trigger transitions between those states.

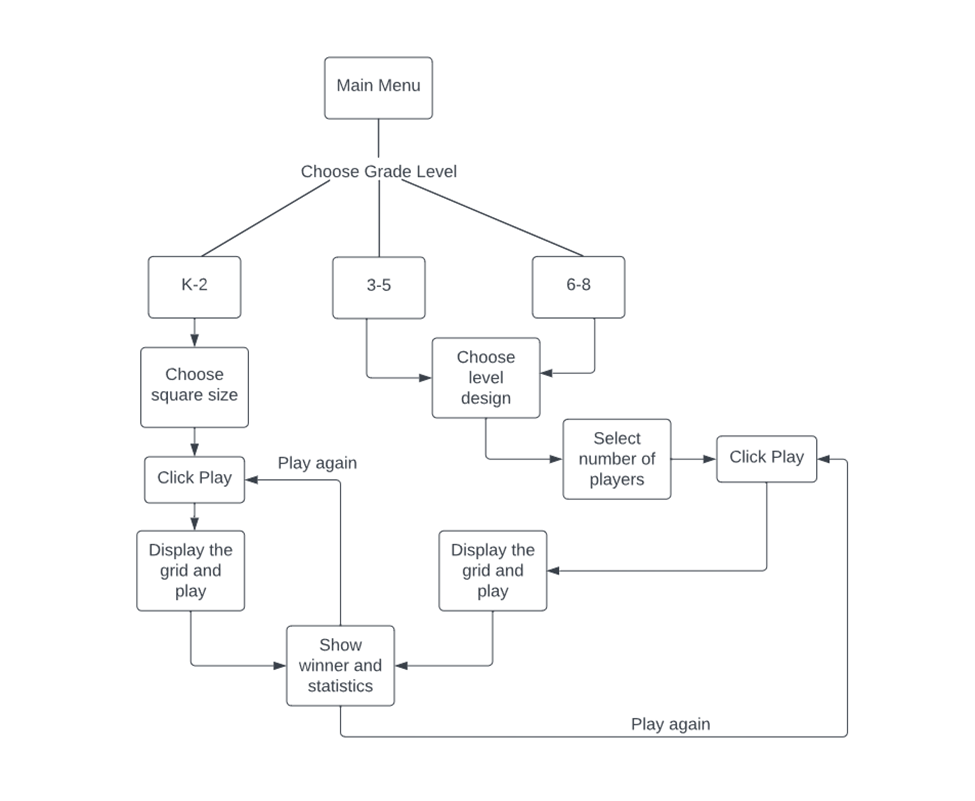
The game starts in the "Start" state and then transitions to the "Player 1 turn" state when the "Create game" event is triggered. From the "Player 1 turn" state, the game can either transition to the "Player 2 turn" state if the "Player 1 move" event is triggered, or to the "Game over" state if the "Player 1 win" event is triggered. Similarly, from the "Player 2 turn" state, the game can either transition to the "Player 1 turn" state if the "Player 2 move" event is triggered, or to the "Game over" state if the "Player 2 win" event is triggered.



*Fig 5: State Diagram*

## Activity Diagram

This activity diagram shows the various steps involved in playing a game of "Wandering in the Woods”. It starts with the "Create game" activity, which sets up the grid and GUI for the game. The "Player 1 move" and "Player 2 move" activities represent the players taking turns entering their moves. The "Check for win" activity represents the game checking if either player has found the other player. If a player has won, the "Game over" activity is triggered and the game ends. If no player has won, the process repeats with the next player's turn.



*Fig 6: Activity Diagram*

Design of the Game:

The purpose of the game is to teach students K-8 about computation, computational thinking, math concepts, and computer science.

The functional requirements of the game include the ability to select different modes of play (K-2, 3-5, 6-8), customize the size and shape of the grid, place players on the grid, and track the movement and meeting of players. The K-2 mode features a square grid and two players that start out in diagonally opposite corners. The 3-5 and 6-8 modes allow for rectangular grids and the option to have 2, 3, or 4 players, who can be placed anywhere on the grid. When two players meet in the 3-5 and 6-8 modes, they will move together until they find the third player, and if the third player is found (assuming there are 4 players), the three players will move together until the fourth player is found. In modes 3-5 and 6-8 there also displays statistics of each run after the game.

In addition to the functional requirements, the game also has non-functional requirements including good performance, security, and usability. The game should run smoothly and efficiently, and it should be secure to protect users' personal information. It should also be easy for users to understand and navigate.

To help visualize how the game will work, I have created a flowchart that illustrates the steps a user will take to interact with the game. The user will start by selecting the mode of play and customizing the grid and player positions, if applicable. They will then start the game and watch as the players move around the grid randomly. When two players meet, there will be a happy graphics display and statistics from the wandering will be displayed and announced audibly. The game can then be reset and played again. In the 3-5 and 6-8 modes, statistics such as the longest run without meeting, shortest run, and average run will be displayed.

I have also designed the user interface using wireframes or mockups. The interface should be visually appealing and easy to use, with clear buttons and instructions for navigating the game.My development plan includes tasks such as coding the game mechanics, implementing the user interface, and testing the game. I will use python or higher to develop the game, and it will require a compatible hardware and software setup to run.

The game mechanics include rules for movement and finding other players. In the K-2 mode, the players will move randomly around the grid, and when they meet, the game will end and display the statistics. In the 3-5 and 6-8 modes, if two players meet, they will move together until they find the third player, and if the third player is found (assuming there are 4 players), the three players will move together until the fourth player is found. I have identified some potential risks or challenges that may arise during development or deployment of the game, such as difficulty implementing certain features or compatibility issues. I will address these risks as they arise and make any necessary revisions to the design as needed.

Overall, the Wandering in the Woods game in python is an interactive and engaging way for students to learn about probability and statistical analysis. It allows for customization of the grid size and player positions, and it provides a fun and interactive way for students to explore different protocols for wandering and determine which is the most effective at minimizing the time it takes for players to meet up. The game also provides an opportunity for students to develop their critical thinking and problem-solving skills by conducting experiments and analyzing data.