# **PCII PROJECT REPORT**

GROUP 3 MEMBERS :

ABANSIR Rayan / AZANGUE FOMAT Gide / LI Xitong / ZHANG Boyuan

## **SUMMARY**

[PCII PROJECT REPORT 1](#_Toc3)

[SUMMARY 2](#_Toc4)

[INTRODUCTION : 3](#_Toc5)

[PROJECT SPECIFICATIONS : 4](#_Toc6)

[GLOBAL ANALYSIS : 5](#_Toc7)

[DETAILED ANALYSIS : 6](#_Toc8)

[DEVELOPMENT PLAN : 8](#_Toc9)

## **INTRODUCTION :**

Our project seeks to merge the world of gardening with real-time strategy games. Inspired by iconic titles such as Dune 2000 and Age of Empires, our game offers an immersive experience centered on farming.

In this game, players embody a farmer who must harvest corn while facing challenges such as crows threatening the crops. To help protect the crops, players can use scarecrows.

The goal is to harvest a specific amount of corn in a limited time. The game is designed to be both entertaining and educational, emphasizing key skills such as resource management, strategic decision-making, and concurrent programming.

Moreover, thanks to the use of the Java Swing library, the game offers an intuitive user interface and a smooth graphical experience.

# **PROJECT SPECIFICATIONS :**

Here are the specifications for our game:

* **Game Objective**: The game is a real-time strategy game with elements of gardening. The player takes on the role of a farmer who must harvest corn while fending off crows, primarily using scarecrows. The objective is to harvest a specific quantity of corn within a set time limit.
* **Related Actions**:
  + **Farmer**: The player controls a farmer who can move around the field, harvest corn, and place scarecrows. The farmer also manages a bag that can be filled with corn and seeds that can be planted.
  + **Crows**: Crows are autonomous entities that attempt to eat the corn. They can be repelled by the farmer or the scarecrows.
  + Scarecrows: Scarecrows are entities placed by the farmer to deter the crows. They lose their effectiveness if they remain stationary for too long.
  + **Corn**: Corn is the primary resource in the game. It can be harvested by the farmer and consumed by the crows.
  + **Seeds and Bags**: The farmer has a bag that can be filled with corn. Additionally, the farmer can have seeds that can be planted. These features are also managed by methods in the `Farmer` class.
  + **Harvesting Centers**: These are locations where the farmer can drop harvested corn. They are managed by their own classes.
* **GUI**: The game should offer an intuitive user interface and a smooth graphical experience, facilitated by the Java Swing library.
* **Educational Aspects**: The game should emphasize key skills such as resource management, strategic decision-making, and concurrent programming.

# **GLOBAL ANALYSIS :**

Here are the main features to implement for our game :

* **Farmer Control**: The player should be able to control a farmer who moves around the field, harvests corn, and places or moves scarecrows on the field. The farmer should have a limited speed of movement on the field.
* **Crow Management**: The crows, who try to eat the corn, should be programmed to appear and move autonomously on the field. They should be able to locate the corn and reach it to eat it, and should be able to flee from the farmer and the scarecrows.
* **Scarecrow Usage** : The player can place or move scarecrows on the field to repel the crows. The scarecrows should lose their effectiveness if they remain immobile for too long.
* **Corn Planting and Harvest**: The player can plant corns on the field. Each corn should grow at a certain rate on the field, and the player should be able to harvest it once it is ripe. The growth rate of the corn and the time it takes to ripen can add a resource management element to the game.
* **Harvesting Center Management**: The game have harvesting centers where the farmer can drop corn. These centers are managed by their own classes.
* **Management of Seeds and Bag**: The farmer has a bag that can be filled with corn. Also, the farmer can have seeds that can be planted. These features are also managed by methods in the Farmer class.
* **Scoring System**: A score tracking system should be implemented to count the amount of corn harvested by the player in a limited time.
* **Victory and Defeat Conditions**: The player wins if they manage to sell the required amount of corn in the allotted time. The player loses if the allotted time runs out before the required amount of corn is sold.
* **GUI**: An intuitive user interface should be developed to allow the player to easily control the farmer, plant, harvest and drop corn, place and move scarecrows, and track their score.

# **DETAILED ANALYSIS :**

Here is a quick overview of the difficulty and priority of the features :

* **Class Creation**: Implementing the basic classes for the game entities: farmer, corn, crows, and scarecrows. Each class defines the basic attributes and methods for these entities.
  + - Difficulty: Medium
  + - Priority: High
* **Game Engine Development**: Creating the GameEngine class, the heart of our game. It manages the game's state and interactions between different game entities.
  + - Difficulty: Medium
  + - Priority: High
* **Game Frame Construction**: Creating the main window frame for our game using the JFrame class provided by Java Swing. The frame serves as the container for all other graphical components in our game.
  + - Difficulty: High
  + - Priority: High
* **Game Panel Construction**: Setting up the main game panel which will hold all the game components and handle the main game rendering.
  + - Difficulty: Medium
  + - Priority: High
* **Support Panel Display Updating:** Implementing a feature to update the support panel display based on the game state. This includes displaying the current score, the number of remaining lives, and other relevant game information.
  + - Difficulty: Medium
  + - Priority: High
* **Button Clickability:** Implementing methods in the Farmer class to check if various actions are clickable, such as harvesting corn, dropping corn, planting seeds, eating corn, placing corn in a bag, taking a scarecrow, and placing a scarecrow.
  + - Difficulty: Low
  + - Priority: Meidum
* **Farmer Movement Management**: When the farmer is selected, a right-click on another location on the canvas sets that location as the destination. The farmer then moves directly towards this destination.
  + - Difficulty: Medium
  + - Priority: High
* **Farmer Health Management:** Implementing a feature to manage the farmer's health. This includes reducing the farmer's health when a crow attacks, and increasing the farmer's health when the farmer eats corn.
  + - Difficulty: Medium
  + - Priority: Medium
* **Crow Generation**: Creating a CrowThread class that generates crows at fixed intervals. It uses a Timer to schedule the generation of crows.
  + - Difficulty: Medium
  + - Priority: High
* **Crow Movement**: Creating a Crow class that represents a crow in the game. It extends the MovingUnits class, inheriting its movement capabilities.
  + - Difficulty: High
  + - Priority: High
* **Harvesting Centers Management**: Implementing a feature to manage the harvesting centers. This includes checking if the farmer is close to a harvesting center, and if so, allowing the farmer to harvest corn.
  + - Difficulty: Medium
  + - Priority: High
* **Corn Actions Management**: Implementing features to manage various corn actions. This includes harvesting corn, planting corn seeds, dropping corn, selling corn, converting corn into seeds, placing corn in the bag, the farmer eating corn, and the crow eating corn.
  + - Difficulty: High
  + - Priority: High
* **Corn Lifecycle Management**: Implementing a feature to manage the lifecycle of corn. This includes the stages of seed, growing, mature, and withered. The system checks the age of the corn and updates its lifecycle stage accordingly.
  + - Difficulty: High
  + - Priority: High
* **Graphic Resource Management and Rendering**: This involves loading and rend ering multiple image resources, including farmers, crows, corn, and scarecrows.
  + - Difficulty: Medium, Managing and loading images efficiently, with error handling, is necessary.
  + - Priority: High, Graphics are a central part of the game's user interface and crucial for player experience.
* **Scene Management**: Currently using a static background, adding dynamic backgrounds to increase the richness and appeal of the game.
  + Difficulty: Low. Requires managing resources for different scenes and optimizing memory and load times.
  + Priority: Medium. Although not essential for functionality, it significantly enhances the game's feel and depth.
* **User Interface Management**: Implemented a complex interface that includes multiple screens (start screen, game interface, game over screen) and a control panel.
* Difficulty: Medium. Involves complex layout management and component control.
* Priority: High. This is the main interface for player interaction, crucial for player experience.
* **Multithreading and Game Logic Management**: Uses multithreading to manage dynamic elements in the game, such as the generation of crows and corn, and the decrement of the farmer's health.
* Difficulty: High. Requires proper management of thread lifecycles and synchronization issues.
* Priority: High. These threads directly affect the game's dynamic features and real-time performance, crucial for smooth operation and user experience.
* **Sound Management**: Loads and controls background music and other sound effects in the game.
* Difficulty: Medium. Involves handling the loading, playing, and volume control of audio files.
* Priority: Medium. While sound effects enhance the immersive experience of the game, they do not directly impact game mechanics.
* **Event Handling**: Implements event listening and response for buttons and timers, handling updates to game state and interface feedback.
* Difficulty: Medium. Involves the correct use of the Swing event dispatch thread and clear logic for event handling.
* Priority: High. Event handling is the core of game interaction, ensuring the game responds to user actions; without it, gameplay is not possible.
* **Game Start and Reset Logic**: Controls the start, end, and reset of the game, including managing the game loop and reinitialization of resources.
* Difficulty: Medium. Involves managing game states and correctly resetting and starting the game at different stages.
* Priority: High. Proper game flow management is key to ensuring players can play continuously and the game state is correctly initialized.

# **DEVELOPMENT PLAN :**

Here's a rough estimation of the time required for analysis, design, development and testing for each feature, considering the level of the team members:

* **Class Creation**:
  + Analysis: 2 hours
  + Design: 1 hour
  + Development: 2 hours
  + Testing: 2 hours
* **Game Engine Development**:
  + Analysis: 1 hours
  + Design: 2 hours
  + Development: 2 hours
  + Testing: 3 hours
* **Game Frame Construction**:
  + Analysis: 1 hour
  + Design: 2 hour
  + Development: 2 hours
  + Testing: 2 hour
* **Game Panel Construction**:
  + Analysis: 1 hour
  + Design: 1 hour
  + Development: 2 hours
  + Testing: 1 hour
* **Button Clickability**:
  + Analysis: 1 hour
  + Design: 1 hour
  + Development: 1 hours
  + Testing: 1 hour
* **Support Panel Display Updating**:
  + Analysis: -1 hour
  + Design: -1 hour
  + Development: 1 hour
  + Testing: 1 hour
* **Farmer Movement Management**:
  + Analysis: 2 hours
  + Design: 2 hour
  + Development: 3 hours
  + Testing: 2 hours
* **Farmer Health Management**:
  + Analysis: -1 hour
  + Design: -1 hour
  + Development: 1 hour
  + Testing: 1 hour
* **Crow Generation**:
  + Analysis: 2 hours
  + Design: 2 hours
  + Development: 1 hour
  + Testing: 2 hour
* **Crow Movement**:
  + Analysis: 2 hours
  + Design: 2 hours
  + Development: 3 hours
  + Testing: 3 hours
* **Harvesting Centers Management**:
  + Analysis: -1 hour
  + Design: 1 hour
  + Development: 1 hour
  + Testing: 1 hour
* **Corn Actions Management**:
  + Analysis: 3 hours
  + Design: 2 hours
  + Development: 4 hours
  + Testing: 3 hours
* **Corn Lifecycle Management**:
  + Analysis: 3 hours
  + Design: 2 hours
  + Development: 2 hours
  + Testing: 2 hours
* **Scene Management**
  + Analysis: 2 hours
  + Design: 3 hours
  + Development: 4 hours
  + Testing: 2 hours
* **Event Handling**
  + Analysis: 1 hours
  + Design: 1 hours
  + Development: 1 hours
  + Testing: 1 hours
* **User Interface Management**
  + Analysis: 1 hours
  + Design: 1 hours
  + Development: 4 hours
  + Testing: 1 hours
* **Multithreading and Game Logic Management**
  + Analysis: 1 hours
  + Design: 2 hours
  + Development: 5 hours
  + Testing: 2 hours