SPRINT 1 DELIVERABLES

FIT3077 - CL_Tuesday04pm_Team063 - Coding With Goblins

Table of contents

1. Team Information and Technology Prototypes

1.1. Team Name

Coding With Goblins

1.2. Team Logo

1.3. Team Photo

1.4. Team Member Information

1.4.1. Member 1

Name: Chia Yaw Chong (Darius) – Team Leader
Email: ccho0090@student.monash.edu
Strength: Web Development, Has basic knowledge of Java
Fun Fact: I read Tech News everyday.

1.4.2. Member 2

Name: Thanh Son Truong (Son)
Email: ttru0037@student.monash.edu
Strength: Technical proficiency. Excel in Java.
Fun Fact: Java is my first programming language.

1.4.3. Member 3

Name: Nguyen Viet Phuc (Tom)
Email: vngu0096@student.monash.edu
Strength: Full stack developer but greenie, Has some experience in Java
Fun Fact: I enjoy making great UI for my website.

1.4.4. Member 4

Name: Nguyen Nhut Minh Le (Felix)
Email: nlee0075@student.monash.edu
Strength: Web Development, Proficient in Java.
Fun Fact: I code Java in my dream.

2. Team Schedule

2.1. Regular Meeting Schedule

2.1.1. Weekly Team Meetings:

- Day: Every Thursday, Friday, Saturday, Sunday
- Time: 7:00 PM - 11:00 PM
- Location: Room 205, Engineering Building / Zoom (for remote members)
- Agenda: Discuss progress, address any issues, plan tasks for the week, and review upcoming deadlines.

2.1.2. Daily Stand-Up Meetings:

- Day: Monday to Friday
- Time: 9:00 AM - 9:15 AM
- Location: Slack Channel / Zoom
- Agenda: Quick updates on what each member did yesterday, what they plan to do today, and any blockers they are facing.

2.2. Work Schedule

2.2.1. Coding Sessions:

- Day: Tuesday and Thursday
- Time: 2:00 PM - 5:00 PM
- Location: Room 210, Engineering Building / Virtual Collaboration via GitLab

2.2.2. Design and Planning Sessions:

- Day: Wednesday
- Time: 1:00 PM - 3:00 PM
- Location: Room 215, Engineering Building / Miro Board for virtual collaboration

2.2.3. Testing and Review Sessions:

- Day: Friday
- Time: 10:00 AM - 12:00 PM
- Location: Room 220, Engineering Building / Zoom for remote members

2.3. Workload Management

2.3.1. Task Assignment:

- Tasks will be assigned based on each member's strengths and expertise.
- Use ClickUp to track tasks and their statuses.
- Each task will have a clear owner responsible for its completion.

2.3.2. Progress Tracking:

- Use GitLab for version control and to track code contributions.
- Maintain a shared Google Sheet and ClickUp to log progress and update task statuses.

2.3.3. Communication:

- Use Discord for daily communication and quick updates.
- Schedule ad-hoc meetings as needed to address urgent issues or blockers.

2.3.4. Documentation:

- Maintain detailed documentation in the team's GitLab repository.
- Ensure all meeting notes, decisions, and action items are recorded and accessible to all team members.

3. Technology Stack and Justification

3.1. Decision of Project Programming language

We use **Java** as the programming language for this project.This decision is based on the following considerations:

- Team Expertise:
    - Member 1: Has basic knowledge of Java and is eager to improve their skills through this project.
    - Member 2: Familiar with Java and has completed coursework involving Java programming.
    - Member 3: Proficient in object-oriented programming with Java and has worked on several Java-based projects.
    - Member 4: Experienced in Java development, particularly in building desktop applications using Java Swing.

Given the team's collective experience with Java, we believe this language aligns well with our current expertise. However, we anticipate that Member 4 may need additional support from the teaching team to enhance their Java skills.

3.2. Justification for Choosing Java

We have selected **Java** for the following reasons:

1. Object-Oriented Design: Java is a robust object-oriented programming language, which aligns with the project's requirement to apply proper software development practices and object-oriented principles.
2. Cross-Platform Compatibility: Java applications can run on any platform with a Java Virtual Machine (JVM), ensuring that our game client can be easily executed on different operating systems.
3. Rich Library Support: Java offers extensive libraries and frameworks, such as Java Swing for building graphical user interfaces, which will be beneficial for developing the Santorini game client.
4. Team Familiarity: Most team members have prior experience with Java, which will facilitate smoother collaboration and development processes.
5. Community and Resources: Java has a large and active community, providing ample resources, tutorials, and support, which will be valuable throughout the project.

By choosing Java, we aim to leverage our team's strengths and ensure the successful implementation of the Santorini game client.

4. User Stories

- 4.1. As a player, I want to choose the god card, so that I can decide the ability that I have in that game.

INVEST Criteria:

I: Choosing a god card doesn’t depend on other mechanics.

N: The selection process (random, draft, free choice) can be discussed.

V: Players need unique abilities to strategize (if playing with God Powers).

E: The feature scope is clear (UI for selection + effect application).

S: A single action at the start of the game (if God Powers are enabled).

T: Verify that players can select a god card and that abilities apply.

- 4.2. As a player, I want to be able to move the worker, so that I can position my worker to build a building.

  INVEST Criteria:

  I: The movement action does not depend on any other action.

  N: The movement rules (e.g., range, diagonal) can be defined.

  V: The movement action is needed for gameplay progression.

  E: The movement logic can be estimated.

  S: Only covers the worker movement action.

  T: Can ensure valid/invalid movement actions are correctly handled.

- 4.3. As a player, I want to be able to build one block each turn, so that I can increase the height of the building.

  INVEST Criteria:

  I: Building action is independent of other core actions within the turn structure.

  N: The specific rules for building placement can be defined.

  V: Players need to build in order to progress and potentially win.

  E: Uses simple logic based on game rules.

  S: Only focuses on the standard build action.

  T: Can confirm if the player places the building correctly/incorrectly.

- 4.4. As a player, I want to be able to build the dome on top of the 3rd floor, so that I can prevent other players from moving to that building.

  INVEST Criteria:

  I: Dome placement logic is distinct from standard block building.

  N: The trigger condition (exactly on 3rd floor) is defined by rules.

  V: The dome placement is a key strategic blocking mechanic.

  E: Requires checking the building height before allowing dome placement.

  S: Focuses on dome placement rule only.

T: Can confirm when a player can/cannot place a dome based on height.

- 4.5. As a player, I want to be able to use my god card ability during my turn, so that I can take advantage of the game.

  INVEST Criteria:

  I: Using abilities can be triggered independently based on god card rules.

  N: The specific trigger points and effects of abilities can be adjusted per god card.

  V: Makes the game more varied and strategic (when using God Powers).

  E: Effort depends heavily on the specific god card ability being implemented.

  S: Focuses on the action of triggering/applying an ability.

  T: Make sure the specific ability being tested is used correctly according to its rules.

- 4.6. As a player, I want to see the win announcement, so that I can know that the game has already ended.

  INVEST Criteria:

  I: The win announcement display is independent of the win condition logic itself.

  N: The appearance and text of the win announcement can be changed.

  V: Clearly communicates the game's conclusion to the players.

  E: Displaying a simple winning message is easily estimated.

S: Only focuses on a single event display.

T: Win announcement can be validated when win conditions are met in different scenarios.

- 4.7. As a player, I want a clear visual indicator showing whose turn it currently is, so that I know when to play and when to wait for the opponent.

  INVEST Criteria:

  I: Displaying turn status is independent of game logic execution.

  N: The style and placement of the indicator (e.g., text label, highlighted player name) are negotiable.

  V: Fundamental for game flow and clarity in a turn-based game.

  E: Simple UI update based on game state; easily estimated.

  S: A small, focused UI element.

  T: Can test by observing the indicator change correctly after each turn completes.

- 4.8. As a player, I want to undo my last action (move or build), so that I can play again if I made a wrong decision or misclick.

  INVEST Criteria:

  I: The undo action is separate from the forward gameplay mechanics.

  N: The scope (last action vs full turn) is defined here as last action for simplicity. Limits (e.g., once per turn) could be negotiated.

V: Improves user experience, allowing correction of misclicks or immediate regrets.

E: Requires tracking previous game state(s); undoing only the last action is simpler.

S: Focuses on the undo action itself, scoped to last action.

T: Check if the undo action correctly reverts the game state to the point before the last move or build.

- 4.9. As a player, I want to have a restart option, so that I can restart the game again.

  INVEST Criteria:

  I: The restart option does not affect the core gameplay logic engine.

  N: The confirmation prompt for restart can be adjusted.

  V: Helps players quickly start a new game without quitting and relaunching.

  E: Requires clearing the current game state and re-initializing.

  S: Focuses on a single game reset action.

  T: Make sure that the game board and player states reset correctly to the initial setup.

- 4.10. As a player, I want to clearly see which of my workers is currently selected, so that I know which piece will perform the move and build actions.

  INVEST Criteria:

  I: Visual selection indication is separate from movement/building execution logic.

N: The exact visual style (highlight, border, marker) can be discussed.

V: Prevents errors and confusion, crucial for controlling two workers.

E: Requires UI updates based on player input; effort is predictable.

S: Focuses on a single, distinct UI feedback element.

T: Can verify that selecting worker A highlights A, selecting B highlights B, and actions apply to the highlighted worker.

- 4.11. As a player, I want to block the opponent's movement by capping buildings with domes, so that I can prevent them from winning.

  INVEST Criteria:

  I: This story focuses on the strategic use of domes, building on the dome placement mechanic (4.4).

  N: Relies on 4.4 being implemented.

  V: Highlights an important strategic gameplay element enabled by domes.

  E: Effort is primarily covered by implementing dome placement (4.4) and movement rules correctly.

  S: Describes a strategic goal achieved through existing mechanics.

  T: Tested by attempting to move onto a domed building and verifying it's blocked.

- 4.12. As a player, I want the game to visually display the current height of each building space, so that I can quickly assess the board state and plan my moves.

INVEST Criteria:

I: Displaying height is separate from the building action itself.

N: How height is displayed (number, simple graphic change) can be decided.

V: Essential for understanding game state, win conditions, and dome placement.

E: Requires reading game state and updating the UI; straightforward.

S: Focuses on a specific visual representation of existing game data.

T: Can verify that building blocks correctly update the visual height indicator on the board.

- 4.13. As a player, I want to see descriptions of all god powers before selecting one, so that I can choose strategically.

  INVEST Criteria: I: This feature is self-contained (UI display + data fetching).

  N: UI presentation, length/detail of descriptions can be discussed.

  V: Helps players make informed decisions when choosing God Powers.

  E: Data loading and display UI can be easily scoped.

  S: Just needs a screen or overlay and fetching/displaying God Card data (if God Powers enabled).

  T: Can be tested by verifying if all available god powers are displayed with their details before selection.

- 4.14. As a player, I want to clearly see which God Power my opponent has active during the game, so that I can anticipate their special abilities and plan my strategy accordingly.

  INVEST Criteria:

  I: Displaying the opponent's chosen power is separate from the selection process or the power's execution logic.

  N: The specific location and visual style (e.g., icon in corner, text display) for showing the opponent's power can be discussed.

  V: Provides crucial strategic information necessary for informed decision-making when playing with God Powers.

  E: Requires fetching the opponent's chosen power from the game state and displaying it in the UI; the effort is relatively small and predictable.

  S: Focuses on adding a single, specific piece of information display to the game interface (if God Powers enabled).

  T: Can verify during gameplay that the opponent's correctly selected God Power is visible on the screen.

- 4.15. As a player, I want to see valid move locations highlighted when I select a worker, so that I can easily identify legal moves according to the rules.

  INVEST Criteria:

  I: Highlighting logic is separate from executing the move.

  N: The style of highlighting (simple color change, outline) can be discussed.

V: Greatly improves usability, reduces mistakes, and helps players learn rules.

E: Requires checking adjacent cells based on rules and updating the UI; effort is predictable. S: Focuses specifically on visual feedback for move options.

T: Can select a worker in various scenarios (near edge, different heights, domes, opponents) and verify only correct move locations are highlighted.

- 4.16. As a player, I want to move before building in my turn, so that I follow the standard turn sequence of Santorini.

  INVEST Criteria:

  I: Enforcing turn order is a core rule, independent of specific move/build actions.

  N: The strictness (must move then build) is defined by the rules.

  V: Essential for correct gameplay according to standard Santorini rules.

  E: Requires sequencing the player's actions within a turn; effort is low.

  S: Focuses on the basic move-then-build turn structure.

  T: Success can be measured by ensuring players must perform a move before they can perform a build action in their turn.

- 4.17. As a player, I want to view a log of the recent actions during the gameplay, so that I can review what just happened.

  INVEST Criteria:

  I: The action log feature is independent of core game execution.

N: The format (simple text), accessibility (button?), and the number of actions shown can be tailored.

V: Viewing past actions helps track the game flow.

E: Implementing a simple text-based action log is feasible and low effort.

S: This feature is specific and can be completed quickly if kept simple.

T: Effectiveness can be tested by ensuring actions are logged (e.g., "P1 moved A1->B2", "P1 built C2") and players can view the log.

- 4.18. As a player, I want to climb up my workers to adjacent buildings up to one level higher, so that I can reach higher ground.

  INVEST Criteria:

  I: The climbing rule is part of the standard movement logic (4.2).

  N: The height restriction (max 1 level up) is defined by rules.

  V: A core movement mechanic necessary for vertical progression.

  E: The task is part of implementing the standard movement validation logic.

  S: Focuses on a specific rule subset within the worker movement.

  T: Testing involves ensuring workers can move up one level but not more.

- 4.19. As a player, I want to swap the positions of my two workers during my turn, if my God Power allows it, so that I can create better positioning for my next moves.

INVEST Criteria:

I: The worker swap feature is independent and enabled only by specific God Powers.

N: Details like limitations or conditions for swapping depend on the specific God Power's rules.

V: This functionality offers strategic flexibility via specific God Powers (extension).

E: The development effort is assessable based on implementing the logic for the specific God Power that grants this.

S: This is a concise feature tied to a specific extension (God Power).

T: Success can be evaluated by ensuring the swap action works correctly when the relevant God Power is active.

- 4.20. As a player, I want to force an opponent’s worker to move if my god power allows it, so that I can disrupt their strategy and control the board.

  INVEST Criteria:

  I: The forced movement action is a distinct feature enabled by a specific God Power.

  N: Implementation specifics (activation conditions, limitations) depend on the God Power's rules.

  V: Adds interactive depth via specific powers (extension).

  E: The effort to implement this feature depends on the specific God Power's logic.

  S: This is a targeted enhancement tied to a specific extension (God Power).

T: Testing involves ensuring the forced movement functions correctly according to the God Power's rules.

- 4.21. As a player, I want to place my workers on the board at the start of the game, so that I can begin the game with my pieces in strategic positions.

  INVEST Criteria:

  I: Can be implemented without depending on other gameplay stories (it's part of setup).

  N: Details of how placement occurs (player A places 2, then B places 2, or alternating) can be discussed.

  V: Allows players to execute the initial setup phase of the game.

  E: Effort required for the placement UI and logic can be estimated.

  S: Scope is small enough to be completed within an early sprint.

  T: Can be tested by verifying workers can be placed on valid starting board positions.

- 4.22. As a player, I want to see valid build locations highlighted after I have moved a worker, so that I know where I can legally place a block or dome.

  INVEST Criteria:

  I: Highlighting logic is separate from executing the build action.

  N: The style of highlighting (simple color change, outline) can be discussed.

  V: Improves usability and prevents illegal build attempts, guiding the player.

E: Requires checking adjacent cells around the moved worker's new position based on build rules; effort is predictable.

S: Focuses specifically on visual feedback for build options.

T: Can move a worker to various locations and verify only correct, unoccupied, non-domed adjacent build spots are highlighted.

- 4.23. As a player, I want the game to automatically detect if I have no legal move OR no legal build action available on my turn and declare my opponent the winner, so that the game correctly handles losing conditions.

  INVEST Criteria:

  I: Checking for the "stuck" condition is separate from move/build execution.

  N: The exact message or visual for announcing the loss can be discussed.

  V: Implements a core rule of Santorini for game completion when a player is trapped.

  E: Requires checking all possible moves for both workers, and if any move exists, checking possible builds afterwards; effort is estimable.

  S: Focuses on a specific game-ending condition check.

  T: Can create board states where a player cannot move, or can move but cannot build afterwards, and verify the loss is triggered correctly.

- 4.24. As a player, I want to move my worker to the third level of a building, so that I can win the game.

  INVEST Criteria:

I: This win condition check is independent of other game logic, though it relies on movement.

N: The win condition (reaching the third level) is defined by standard rules.

V: Defines the primary clear win condition of the game.

E: Effort required involves checking the destination level after each valid move; low effort.

S: Scope is small enough to be completed within a sprint, tied closely to movement validation.

T: Can be tested by moving a worker to a third-level building and verifying the win condition is triggered.

- 4.25. As a player, I want simple visual feedback (e.g., a brief highlight change on the affected squares or piece) confirming that my move or build action was successfully completed, so that I know my input was registered and the turn phase progressed correctly.

  INVEST Criteria:

  I: The visual feedback mechanism is separate from the core logic of executing moves or builds.

  N: The specific type and style of simple feedback (highlight color, duration) can be determined.

  V: Improves the user experience by providing clear confirmation of actions.

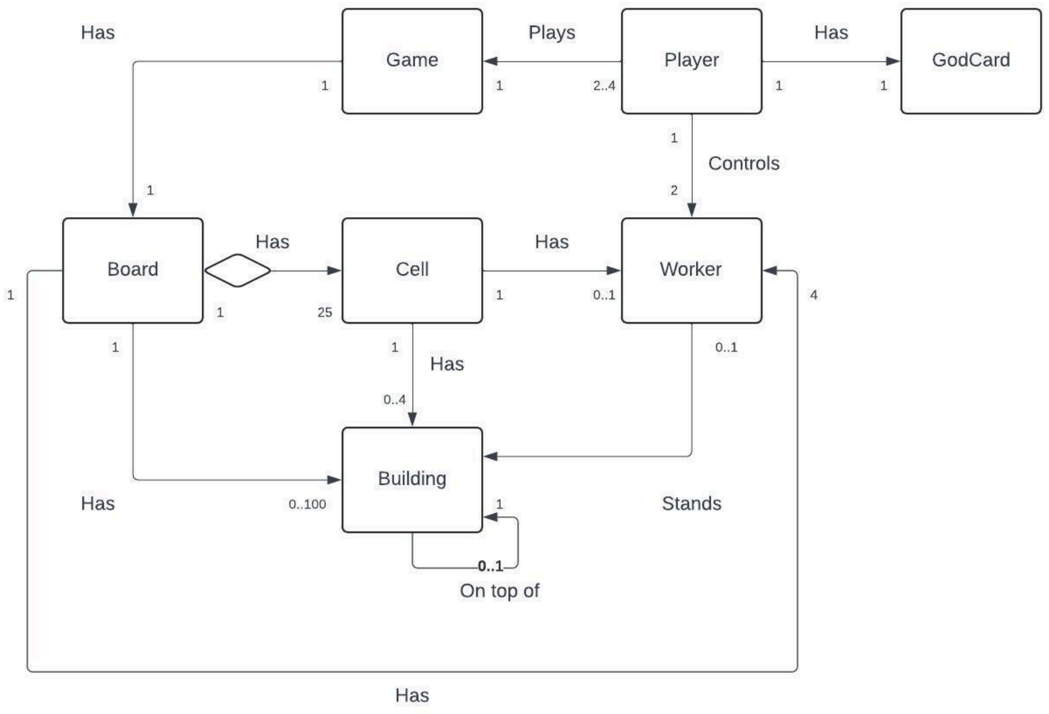
E: Implementing simple UI changes upon successful action completion is a manageable and estimable task.

S: Focuses on a specific, limited feedback element tied to core actions, avoiding complex animation.

T: Can perform valid moves and builds and verify that the defined simple visual confirmation feedback occurs consistently.

5. Domain Model

5.1. Conceptual Diagram - Domain Model



5.2. Concept Explanation

5.2.1. Entities

➔ **Game:**

- **Justification:** The Game entity represents a single instance of the Santorini game being played.
- **Relationships:**
  - **Plays (Game to Player):** A Game Plays between 2 and 4 Players. This directly reflects the rules of Santorini, which is designed for 2 to 4 players.
  - Board (Game to Board): A Game has exactly one Board. Each board belongs to exactly one Game.

➔ **Player:**

- **Justification:** The Player entity represents an individual participating in the game. Each player controls their own set of workers and may have a God Card.
- **Relationships:**
  - **Controls (Player to Worker):** A Player Controls exactly two Workers. This is a fundamental rule of Santorini where each player has two workers to move and build with.
  - **Have (Player to GodCard):** A Player Has exactly one GodCard. This represents the use of God Cards in Santorini, where each player is assigned a unique power.

➔ **GodCard:**

- **Justification:** The GodCard entity represents a special ability or rule modification that a player possesses during the game.
- **Relationships:**
    - **Have (Player to GodCard):** As mentioned above, this relationship indicates that each player has one God Card.

➔ **Board:**

- **Justification:** The Board entity represents the physical game board on which Santorini is played. It is a 5x5 grid of cells.
- **Relationships:**
    - **Has (Board to Cell):** A Board Has exactly 25 Cells. This corresponds to the 5x5 grid structure of the Santorini board.
      Has (Board to Building): A Board Has 0 to 100 buildings. A building can stack up to 4 levels. If every cell has a 4 level building, the whole board would have 100 buildings. Each building belongs to one board.
    - **Has (Board to Worker):** A Board Has 4 workers. Each worker belongs to one board.

➔ **Cell:**

- **Justification:** The Cell entity represents a single square on the Santorini game board. Each cell can potentially contain a building and a worker.
- **Relationships:**
    - **Has (Cell to Worker):** A Cell Has zero or one Worker. A cell can either be empty or occupied by a single worker.
    - **Has (Cell to Building):** A Cell Has zero to four levels of Building. This represents the different levels of construction possible on a cell, from the ground level up to level 3, and potentially a dome (level 4).

- ➔ **Worker:**
    - ◆ **Justification:** The Worker entity represents one of the two playing pieces controlled by a player. Workers are used to move around the board and build structures.
    - ◆ **Has (Worker to Building):** A Worker can Stand on zero or one Building. A building can either be empty or occupied by a single worker.
- ➔ **Building:**
    - ◆ **Justification:** The Building entity represents the structures that are built on the cells of the board. Buildings have different levels, and a cell can have multiple levels of building on it.
    - ◆ **Relationships:**
        - **Has (Building to Building):** A building is on top of another building, and it's not always required.

5.2.2. Specific Modelling Decisions

- **Explicitly Modelling GodCard:** We chose to model GodCard as a separate entity to clearly represent the relationship between a player and their special ability. This approach allows for potential future expansion of God Card attributes and maintains a clean separation of concerns.
- **Representing the Board as a Collection of Cells:** We opted to represent the Board as having a collection of Cell entities rather than just a 2D array or a single entity with board state. This allows us to associate properties and relationships with each individual cell, such as the building level and the worker occupying it.

This makes the model more flexible and easier to understand in terms of the game's spatial structure.

- **Multiplicity of Relationships:** We carefully considered the multiplicity (the numbers on the relationship lines) to accurately reflect the game rules. For example, a Game having 2 to 4 Players enforces the player count rule. A Cell having 0 to 1 Worker reflects that a cell can be empty or have one worker, but not more. Similarly, 0 to 4 Building levels on a Cell accounts for the ground level and the three building levels plus the dome.

5.2.3. Assumptions

- **Standard Santorini Rules:** We have assumed that this domain model is intended to represent the core mechanics of the standard Santorini game. If there are specific expansions or house rules that significantly alter the entities or their relationships, the model might need to be adjusted.
- **No Distinction Between Player Workers:** The model currently does not differentiate between the two workers controlled by a player. They are treated as identical in terms of their relationship with the game board and cells. If future features require distinguishing between a player's two workers (e.g., for certain God Powers), we might need to add an attribute to the Worker entity to identify them uniquely within a player's control.
- **Building Levels as a Count:** The Building relationship with Cell (0..4) implicitly represents the level of the building. We assume that the "Building" entity itself might just be a conceptual representation of the constructed level, and the number in the relationship indicates the current level on that cell. We could have also modelled different building levels (Level1, Level2, Level3, Dome) as

separate entities, but for the basic mechanics, representing the level as a count seems sufficient and simpler.

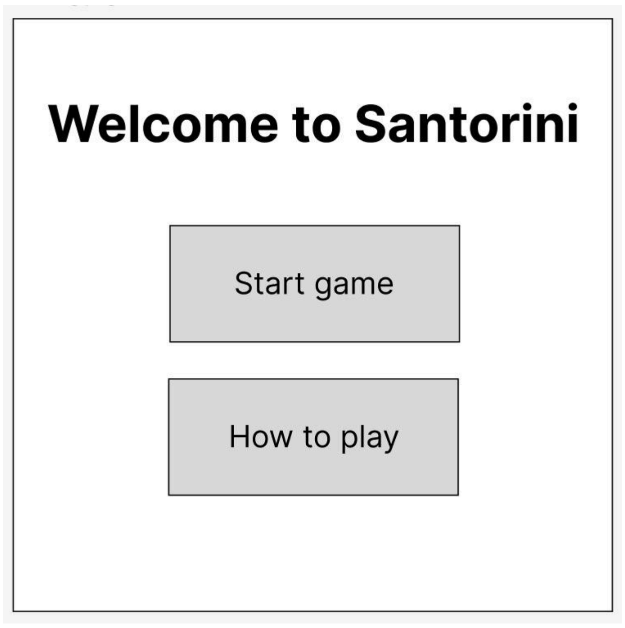
5.2.4. Other Justifications

- The domain model depicts a Player as having a 'Controls' relationship with two Worker entities, indicated by the multiplicity of 2 on the Worker side. This signifies that each player in the game of Santorini has two workers under their command throughout the duration of the game. While the model illustrates this persistent control over both workers, it's crucial to understand that the active gameplay involves each player controlling only one of their two workers during their turn. The turn structure in Santorini allows a player to choose one of their workers to move and then build. This dynamic, turn-based control of a single worker at a time is a rule of the game that will be implemented within the game logic of our digital version. The domain model focuses on establishing the fundamental ownership and availability of the two workers to each player, while the specific rules governing their activation and usage during a turn will be handled by the game's behavioral logic.

6. Basic UI Design

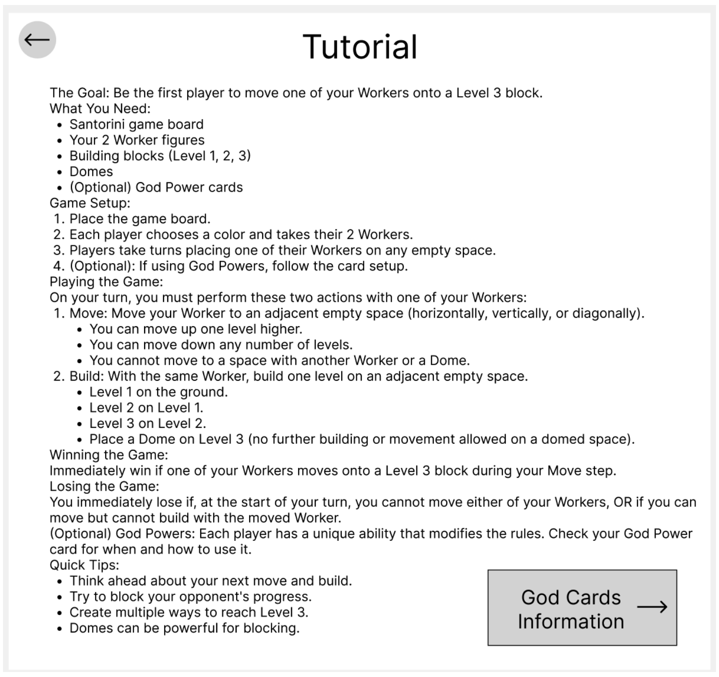
6.1. Main Menu

This is the initial screen presented to the player upon entering the game. It displays the game title "Welcome to Santorini" at the top and offers two primary options as buttons: "Start game" to begin a new game and "How to play" to access instructions or a tutorial.



6.2. Tutorial Screen - Rules and Gameplay

This page presents the tutorial for the Santorini game. It outlines the game's objective (being the first to move a worker onto a Level 3 block), the necessary components, setup instructions, and detailed rules for playing, including movement and building actions. It also explains how to win and lose the game, briefly mentions optional God Powers, and provides quick gameplay tips. A button labeled "God Cards Information" with an arrow suggests a link to further details on the God Power cards.



The Goal: Be the first player to move one of your Workers onto a Level 3 block.
What You Need:
- Santorini game board
- Your 2 Worker figures
- Building blocks (Level 1, 2, 3)
- Domes
- (Optional) God Power cards

Game Setup:
1. Place the game board.
2. Each player chooses a color and takes their 2 Workers.
3. Players take turns placing one of their Workers on any empty space.
4. (Optional): If using God Powers, follow the card setup.

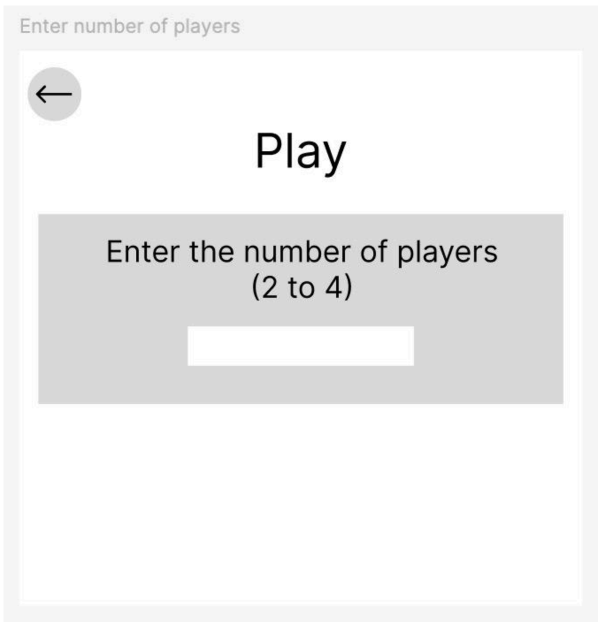
Playing the Game:
On your turn, you must perform these two actions with one of your Workers:
1. Move: Move your Worker to an adjacent empty space (horizontally, vertically, or diagonally).
   - You can move up one level higher.
   - You can move down any number of levels.
   - You cannot move to a space with another Worker or a Dome.
2. Build: With the same Worker, build one level on an adjacent empty space.
   - Level 1 on the ground.
   - Level 2 on Level 1.
   - Level 3 on Level 2.
   - Place a Dome on Level 3 (no further building or movement allowed on a domed space).

Winning the Game:
Immediately win if one of your Workers moves onto a Level 3 block during your Move step.
Losing the Game:
You immediately lose if, at the start of your turn, you cannot move either of your Workers, OR if you can move but cannot build with the moved Worker.
(Optional) God Powers: Each player has a unique ability that modifies the rules. Check your God Power card for when and how to use it.
Quick Tips:
- Think ahead about your next move and build.
- Try to block your opponent's progress.
- Create multiple ways to reach Level 3.
- Domes can be powerful for blocking.

6.3. Player Count Input Screen

This screen prompts the user to enter the number of players for the game. The text "Enter the number of players (2 to 4)" indicates that the game supports between two and four players. A blank input field is provided for the user to enter the desired number. The word

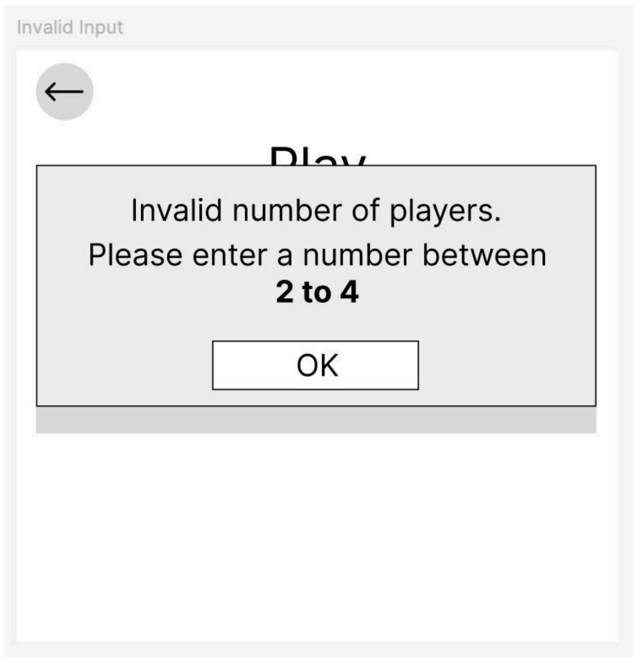
"Play" is prominently displayed at the top, suggesting this screen is part of the game initiation process. A back arrow in the top left corner allows the user to navigate to the previous screen.



Enter the number of players (2 to 4)

6.4. Invalid Player Count Error Message

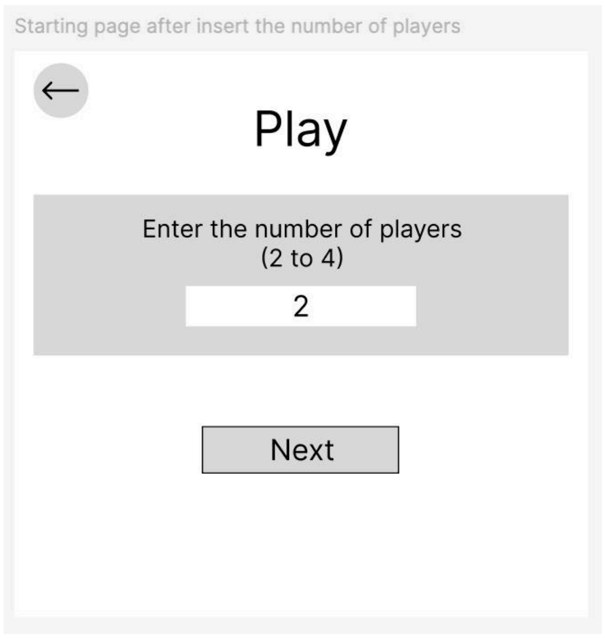
This is an error message displayed to the user. The title "Invalid Input" indicates that the previously entered data was not acceptable. The message "Invalid number of players. Please enter a number between 2 to 4" informs the user that the number of players they entered was outside the allowed range. An "OK" button provides the user with a way to dismiss the error message and return to the player count input screen to enter a valid number.



Invalid number of players.
Please enter a number between **2 to 4**

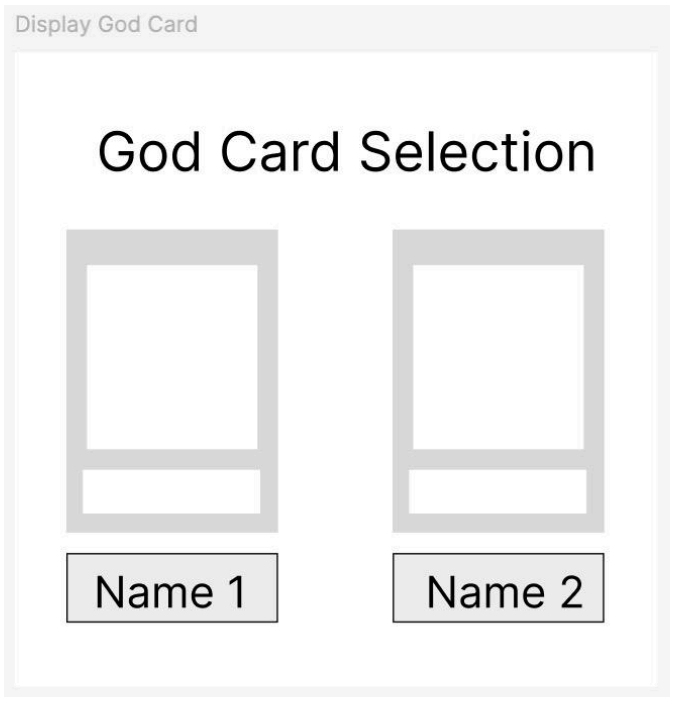
6.5. Player Count Confirmation Screen

This screen appears after the player has entered the number of players. The input field now displays the entered value, which is "2" in this case. A "Next" button is now present, allowing the player to proceed with the game setup using the confirmed player count.


Starting page after insert the number of players
←
Play
Enter the number of players
(2 to 4)
2
Next

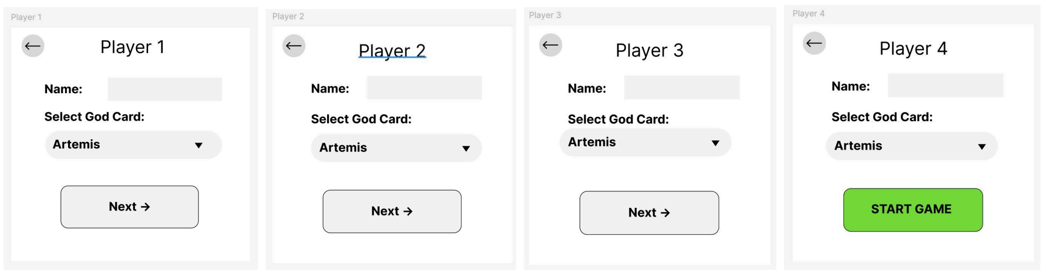
6.6. God Card Selection Screen

This screen is dedicated to the selection of God Cards for the game. Two placeholder card outlines are shown, each with a space below likely intended to display the name of the God Card once selected. Buttons labeled "Name 1" and "Name 2" are positioned below each card placeholder, suggesting that players will choose their God Cards, and their chosen names will be associated with them here.



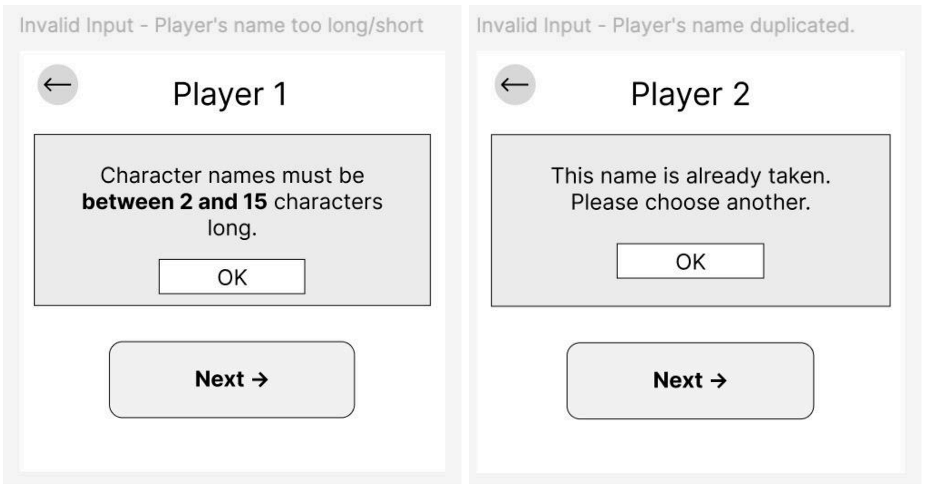
6.7. Player (1, 2, 3, 4) Setup Screen

This screen allows the first player to configure their in-game settings. It is clearly labeled "Player 1" at the top. The screen includes a field labeled "Name:" where the player can enter their desired name. Below that is a "Select God Card:" dropdown menu, currently showing "Artemis" as the selected card. A "Next" button with an arrow indicates that the player will proceed to the next step after confirming their name and God Card selection. The last player's setup screen will display a “Start Game” button.



6.8. Player Name Error Message

This screen displays an error message informing the player about an issue with their chosen name. The message can appear in scenarios where the entered character name does not meet the required length (e.g., it's too short or too long) or when the chosen name is already taken by another player. An "OK" button allows the player to dismiss the message and return to the name input field to correct their entry.


Invalid Input - Player's name too long/short

Player 1

Character names must be **between 2 and 15** characters long.

OK

Next →

Invalid Input - Player's name duplicated.

Player 2

This name is already taken. Please choose another.

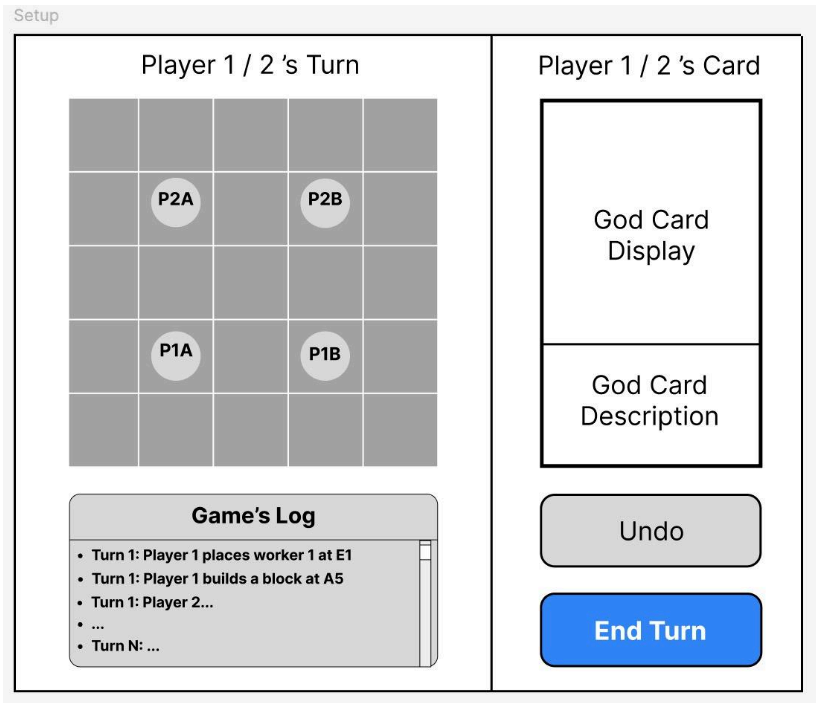
OK

Next →

6.9. Gameplay Screen

This screen represents the main gameplay interface. It is divided into several sections:

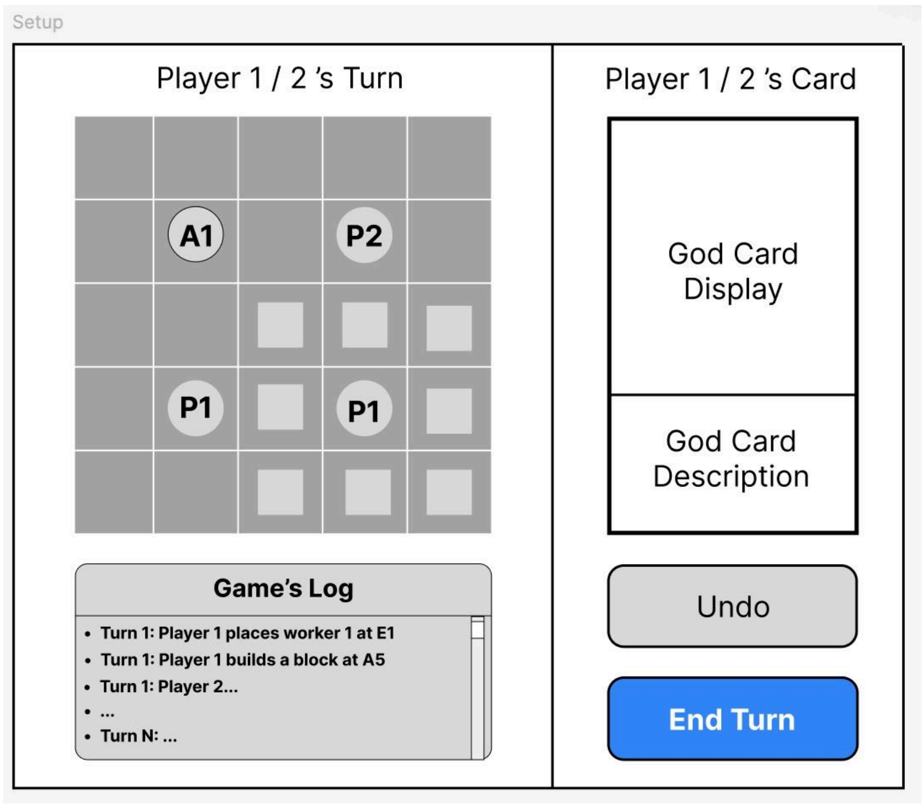
- **Top Left: Game Board:** A grid showing the Santorini game board with the positions of player workers. "P1A" and "P1B" indicate the two workers of Player 1, while "P2A" and "P2B" indicate the two workers of Player 2. The text above suggests it is currently "Player 1 / 2's Turn," indicating whose turn it is.
- **Top Right: Player Card Area:** A space dedicated to displaying the current player's God Card. It is divided into a "God Card Display" area (presumably for an image of the card) and a "God Card Description" area (for the card's special ability).
- **Bottom Left: Game's Log:** A scrollable text area that records the actions taken during the game, turn by turn. It shows examples like "Turn 1: Player 1 places worker 1 at E1" and "Turn 1: Player 1 builds a block at A5."
- **Bottom Right: Action Buttons:** Buttons for "Undo" (allowing the player to take back their last action) and "End Turn" (to conclude the current player's turn and pass control to the opponent).



- Turn 1: Player 1 places worker 1 at E1
- Turn 1: Player 1 builds a block at A5
- Turn 1: Player 2...
- ...
- Turn N: ...

6.10. Gameplay Screen - Turn In Progress with Move/Build Indicator

The key difference highlighted here is the visual indicator on the game board.



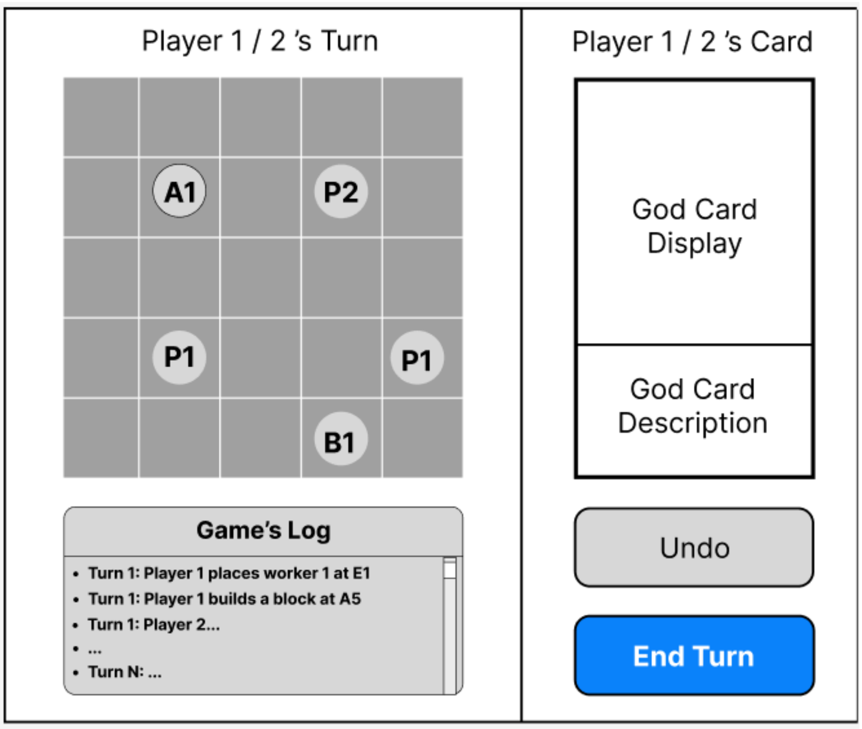
- Turn 1: Player 1 places worker 1 at E1
- Turn 1: Player 1 builds a block at A5
- Turn 1: Player 2...
- ...
- Turn N: ...

6.11. Building Display

A circle with letter B will be used to display the building, the number next to it will display the number of floor that building has:
Example:

- B1: 1 floor building
- B2: 2 floors building
- B3: 3 floors building

- D: Dome building (Block building)



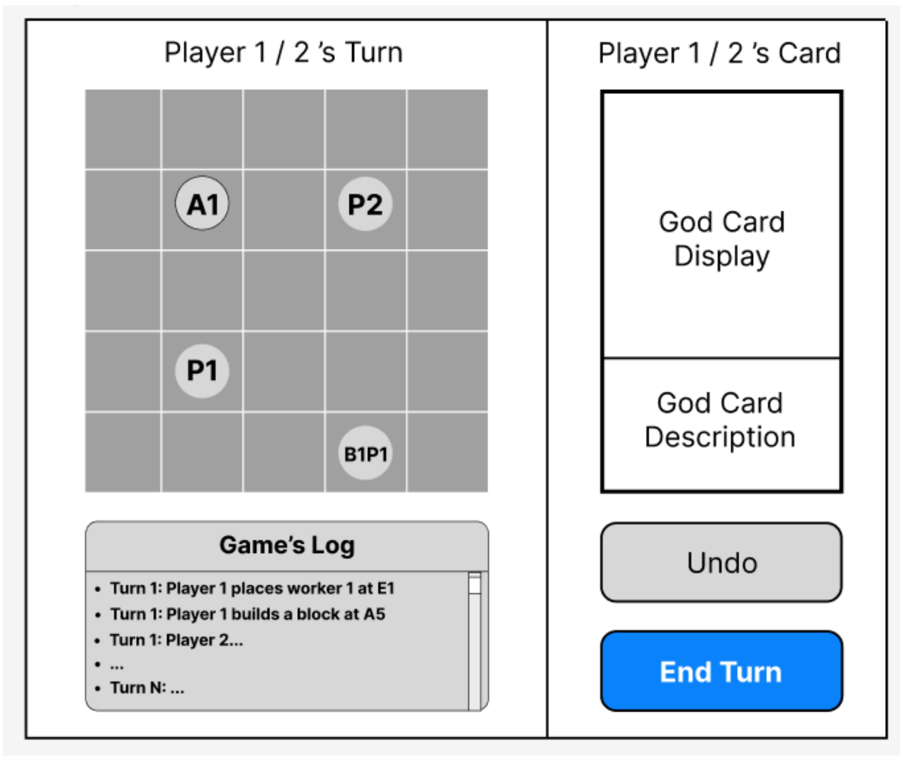
- Turn 1: Player 1 places worker 1 at E1
- Turn 1: Player 1 builds a block at A5
- Turn 1: Player 2...
- ...
- Turn N: ...

6.12. Worker on building

The circle that displays in concept: B..P.. will be used to display the building that had the worker on it.

Example:

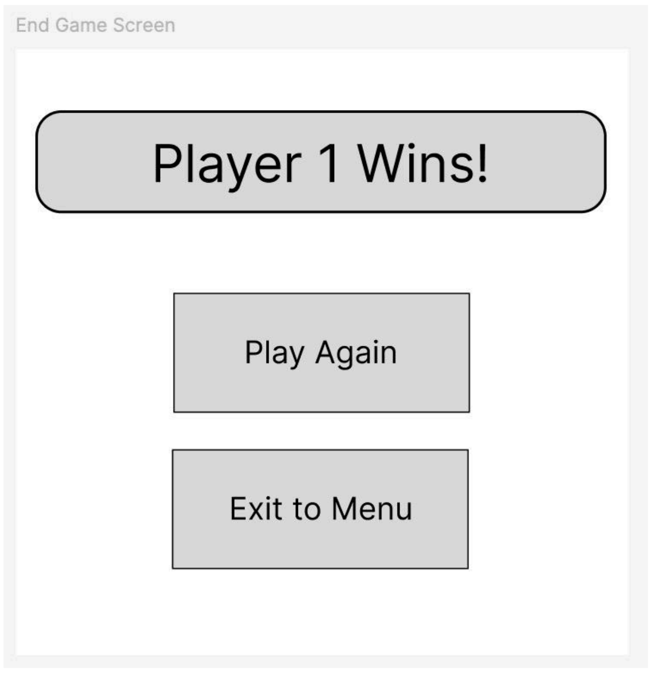
- B1P1: Worker of player 1 is standing on this 1 floor building.



- Turn 1: Player 1 places worker 1 at E1
- Turn 1: Player 1 builds a block at A5
- Turn 1: Player 2...
- ...
- Turn N: ...

6.13. End Game Screen

This screen appears at the conclusion of the game, indicating that "Player 1 Wins!" This is prominently displayed at the top. Below the winning announcement are two button options for the player: "Play Again" to start a new game, and "Exit to Menu" to return to the main menu of the game.



7. Important Links

7.1. Shared Google Drive

Below is the shared Google Drive Folder for all the files to be stored and tracked.

Shared Google Drive:
https://drive.google.com/drive/folders/1GEkzkiHXZX_tPKsu9NfN7ClcH6KK0Nw2?usp=sharing

7.2. Availability Timetable

All team meetings are decided through every member availability using the application below.
Availability Timetable: https://www.when2meet.com/?29700065-hKshW

7.3. GitLab Repository

The team GitLab Repository is stated as below to store all the work and implementation of coding for the whole team project.
GitLab Repository:
https://git.infotech.monash.edu/FIT3077/fit3077-s1-2025/assignment-groups/CL_Tuesday04pm_Team063/project/

7.4. Contribution Log

Every member's contribution is being tracked using Google SpreadSheets.
Contribution Log:
https://docs.google.com/spreadsheets/d/1i_DzwSv9734uOWyQVHOtAnEF2fW-8_6J2zfxrkSH98g/edit?usp=sharing

7.5. ClickUp

ClickUp application is used for team project management which includes all the backlog and sprint tasks, meetings and links to all files.

Additionally, please send an email to vngu0096@student.monash.edu for the invitation to ClickUp if needed.

7.6. Basic UI Design

The basic UI design was made via Figma with all frames self-designed.
Basic UI Design:
https://www.figma.com/design/1bRwDIbvbqpgcTI2uYSjSm/3077---UI-DESIGN?node-id=22-2&t=ewu8OKopghBMivXQ-1

7.7. Domain Model

LucidChart is the application that the team chose to use for the project domain model which includes all the game entities and relationships.
Domain Model:
https://lucid.app/lucidchart/8a3aa098-7051-4e73-aefe-02bf6a92088e/edit?invitationId=inv_de6480d0-9d79-49d5-8ca5-080068840d1b

8. References

1. Workshop slides Weeks 1–5. FIT3077: Software Engineering. Faculty of Information Technology, Monash University. https://learning.monash.edu/course/view.php?id=28766&section=3
2. Microsoft. (2025). *Copilot* (GPT-4) [Large language model]. https://chat.openai.com/