



MONASH University
Information Technology

FIT3077 Software engineering: Architecture and design

Sprint One

Team Santorinians (005)

Team Members:

1. Uchralt Myagmarsuren - 33437823
2. Ubaid Irfan - 33886466
3. Allan Ho - 33927383
4. Aaron - 33906734

1.Team Information and technology prototypes

a. Team Name and Team Photo

Name - Santorinians

Photo -



b. Team Membership

Name	Email	Phone Number	Technical Skills	Professional Skills	Fun fact	Team lead (Sprint 1)
Ubaid Irfan	uirf0001@student.monash.edu	048024 9752	Java, Python, SQL, JS, HTML, CSS	Communication, teamwork, organisational	"I am learning to play the piano."	-
Aaron	azho0020@student.monash.edu	040261 2059	Java Python JS	Team work, communication, problem solving	"My up arrow key cap is missing"	-
Allan Ho	khoo0050@student.monash.edu	041077 3632	Java, Python, C++	Teamwork, Innovation, Communication, Problem	"I cook by myself on Tuesday"	-

				<i>Solving</i>		
Uchralt Myagma rsuren	Uchr...	041320 7875	<i>Java, Python, SQL</i>	<i>Communi- cation, teamwork, professional experience</i>	<i>"My siblings' names and mine all end with a 'T'."</i>	+

c. Team Schedule:

- i. Regular Meeting Schedule: Once every week at 4:00PM Sunday, and more when required.
- ii. Regular Work Schedule:
 1. Team meetings: Every Sunday 4:00PM and more when required
 2. Work Schedule: 1 hour every night (more if required based on team meetings and progression)
- iii. **Workload Distribution:**
 - Team members will follow agile methodology and assign themselves to the appropriate number of user stories per sprint, balancing uni work and life
 - Team progression: a Sprint Backlog will be created for each sprint, containing all the user stories needing to be implemented in a sprint. Here, team members will assign themselves to items they believe will be challenging and promote learning, but ultimately completable, so as to not burden the rest of the team. The assignee of each item will regularly (at least 1-2 times per week) update the status of the task, e.g. Not Started, In Progress, Completed, making it clear to the team how they are progressing throughout the sprint.

d. Technology Stack and Justification

- i. The team is experienced in both Java and Python, having done unit/s in it prior, e.g. FIT2099 and FIT1045. Uchralt Myagmarsuren is currently doing FIT2099 (in which he is learning Java/OOP) and thus may require extra assistance from the teaching team, however, this decision was based on a majority decision, in which we concluded that the preferred technology stack would utilise Java, as they were more familiar with Java.
- ii. Final choice - **Java (IntelliJ)**
- iii. Sketchboard - this tool enables cross communication and collaboration for design, meaning that not a single person has to complete it all, e.g. several members of the team can contribute to the low-fi prototype - allowing for overall better quality.
- iv. GIT - this will be used to:
 1. Store all the project work that the team completes. This will be stored in a secure repository, ensuring that no committed data is lost.
 2. Improve workflow - each team member will have access to the latest changes and team members won't have to utilise convoluted technology to achieve this, e.g. emailing each iteration.

2. User Stories

1. **As a player, I want to be able to pick the god cards myself so that I can use a god card whose ability matches more to my playstyle.**

Independent - selecting a god card is the first thing that happens in the game. It is not related to any other functionality and does not rely on or interfere with other logic such as movement, block placement, etc.

Negotiable – future extensions enabling (for example) a random god selector for a different game mode can be discussed.

Valuable – The gods are a heavy focus in the game. It allows interesting gameplay and strategy implementation from each player. It ensures that the game doesn't become repetitive and thus, giving the player the opportunity to pick their god from a wide array of options is a vital beginning step in the game.

Estimable – The game needs an interface for selecting god cards and logic for applying chosen abilities. Developers can estimate effort for UI and backend logic.

Small – a focused feature which simply enables the selection of god cards. It is not related to the separate god/god power logic itself, and hence is a small feature that can be easily implemented in a sprint.

Testable - Acceptance criteria.

- Each Player is able to select from a list of gods
- Both players cannot choose the same god
- The player who gets to choose the first god is randomised

- 2. As a Player, I want to be able to choose my god card before I place my workers, so that I can strategically choose where to place my workers depending on my and my opponent's god cards.**

Independent – This functionality is independent of worker placement logic. It only requires the god card selection step to occur beforehand in the game flow.

Negotiable – The exact implementation and UI layout for god card selection can be discussed and designed based on player preferences.

Valuable – Knowing each player's god card before placing workers is crucial for strategy. It allows meaningful decisions based on strengths, weaknesses, and counterplay.

Estimable – The user story is easy to estimate based on existing components.

Small – It's a discrete task that includes straightforward tasks.

Testable – Acceptance Criteria:

- God card selection occurs before the game allows worker placement
- Players can view both selected god cards before choosing starting positions

- 2. As a Player, I want to be able to choose where I put my workers so that I can plan my strategy at the start of the game.**

Independent - It only deals with the placement of the player's starting workers

Negotiable – Can be discussed and refined based on game balance and implementation

Valuable – Choosing starting positions is a crucial strategic decision that impacts the entire game

Estimable – The feature is well-defined and straightforward

Small – The scope is limited to a single phase of the game

Testable – The feature is testable when a game starts

3. **As a player (worker on the board), I want to move one worker to an adjacent tile, so that I can position myself to win the game.**

Independent - only concerned with movement functionality/feature.

Negotiable – negotiable as no specification given of how the movement is to be implemented, leaving room for discussion.

Valuable – a core feature of the game, i.e. player's cannot directly progress towards winning without movement.

Estimable – well defined movement mechanics and easy to calculate

Small – a single basic movement feature which can be easily implemented in a sprint.

Testable – Acceptance criteria:

- A worker can move to an adjacent tile on the board - any tile that is above, below, left, right, and diagonally.

4. **As a player (worker on the board), I want to place a building on any tile (above or below) adjacent to the worker that I moved this turn, so that I can implement my strategy to winning the game.**

Independent - building is not directly related to any other feature in the game, e.g. movement logic (they have been split into multiple user stories).

Negotiable – the specifics in regard to which tiles a worker can/can't build upon depending on the structures already placed on the tile can be discussed/determined later.

Valuable – enables the player to come up with strategies towards winning the game.

Estimable – the scope is clear - implement building placement features. Effort for this is easy to estimate

Small – block placement is a single core feature of the game, it can be easily implemented in a sprint.

Testable – Acceptance criteria:

- A worker that has just moved can place a building on any nearby (adjacent) tile.

- Buildings can be placed on any level (above or below a worker).

5. As a player (worker on the board), I want to be able to only place a building if I have already moved, so that each turn feels purposeful and balanced, encouraging thoughtful movement before construction.

Independent - building placement only being allowed after movement is part of the main game logic, there is no direct dependency with movement - just a restriction due to enforcement of game rules.

Negotiable – can be negotiated, e.g. enable building placement without movement if a certain god's power allows it.

Valuable – Maintains the game's structure and strategic depth

Estimable – The logic is only tracking whether the player has moved before allowing a build action, which is easy.

Small – The scope is limited to a single validation check

Testable – Acceptance criteria:

- .A worker cannot place a building unless they have moved first.
- Turn order resets properly after the player's move-and-build sequence.

6. As a player, I want to be prohibited from building any storey on a level 3 tile, so that my building choices always contribute meaningfully to gameplay and I avoid making ineffective moves.

Independent – This rule governs building restrictions and does not depend on movement mechanics or special abilities.

Valuable – a core game mechanism, without this logic, a worker would be able to place buildings at any time - a great deviation from the Santorini ruleset, completely changing the game.

Estimable – easy to implement a check for this, low effort, easily estimable.

Small – a simple check condition which enables building placement once has_moved has been changed to true. It can easily be implemented in a sprint.

Testable – Acceptance criteria:

- If a player's worker has moved on their turn, then a building can be placed by that worker.
- If a worker has not moved, a building cannot be placed.

7. **As a player (worker on the board), I want to be able to build domes on top of the third floor so that I can block my opponent from winning the game.**

Independent - The user story does not depend on other actions, as building a dome is an independent mechanic which does not require additional features to function.

Negotiable – The action of building a dome can be adjusted and does not specify a certain condition.

Valuable – The strategy is valuable for players as it provides strategic advantage to the player who performs this action

Estimable – The action is predictable by the opponent player

Small – The scope is small-enough, as it only mentions the third floor

Testable – Acceptance criteria:

- Players can place a dome on third level buildings next to them

8. **As a player, I want to be restricted from building on tiles with workers, so that I can avoid causing strange and unfair interactions with the game.**

Independent - The user story does not rely on other systems, it's only about the building system

Negotiable – Method of restriction can be further discussed

Valuable – The implementation of restriction is needed to enforce core gameplay rules, ensuring the game doesn't fall apart from rule breaking plays

Estimable – The logic is clear and aligns to Santorini rules

Small – The validation logic is a simple logic

Testable – Acceptance criteria:

- Simple validation logic that does not allow the player to click on occupied cells.

9. As a Player, I want to be restricted from building on tiles with domes, so that I can avoid causing strange and unfair interactions with the game.

Independent - Building restrictions can be implemented on its own

Negotiable – The form and method of build restriction can be discussed by the team.

Valuable – The implementation of restriction is needed to ensure the game behaves predictably, prevents unfair interactions

Estimable – The logic is a straightforward logic which aligns to base Santorini rules.

Small – The validation logic is clearly scoped as it is one of the essential rules when using the build mechanic.

Testable – Acceptance criteria:

- Simple validation logic that denies player build when they are attempting to build on a domed building.

10. As a player, I want to be restricted to move to tiles with completed buildings, so that I can limit my opponent's movement choice by using buildings

Independent – This rule focuses only on movement restrictions related to completed buildings (domed tiles) and does not depend on other mechanics like building or special abilities.

Negotiable – The exact way this restriction is enforced (e.g., whether exceptions exist for certain god powers) can be discussed and refined.

Valuable – Ensures that movement mechanics work correctly and adds strategic depth by allowing players to block tiles using completed buildings.

Estimable – The logic is simple: check if a tile has a dome, and if so, prevent movement onto it. This makes it easy to estimate development effort.

Small – The scope is limited to a single movement validation rule, making it a manageable task within a sprint.

Testable – Acceptance criteria:

- A worker cannot move onto a tile that has a dome (i.e., a fully completed building).
- If a player attempts to move onto a domed tile, the game prevents the action and provides feedback.
- Other movement rules remain unchanged, ensuring the restriction is properly enforced.

11. As a player, I want to be restricted to move to tiles with any workers, so that I can limit my opponent's strategic choice by worker movement

Independent – This movement restriction operates independently from other gameplay mechanics (e.g. building, god powers), making it modular and easy to implement.

Negotiable – The exact method of restriction and potential visual or UI feedback (e.g. tile highlight disabled, warning popup) can be discussed and refined by the team.

Valuable – Enforces fair gameplay by preventing overlapping worker positions, which is crucial for strategic plays like blocking opponents and limiting their options.

Estimable – Simple logic: check if a tile has any worker before allowing a move. Clear and easy to estimate in terms of implementation effort.

Small – The feature consists of one specific validation check during movement. It's tightly scoped and can be handled in a short development cycle.

Testable – Acceptance Criteria:

- Try moving to a tile with an opposing worker → move is blocked
- Try moving to a tile with your own worker → move is blocked
- Try moving to an unoccupied tile → move succeeds

12. As a player (worker on the board), I want to be able to jump down from any level building, so that I have more mobility, especially in the later stages of the game.

Independent - jumping down simply expands on the movement logic in the game. It is not related to any other mechanism (e.g. building placement) in the game.

Negotiable – Future considerations such as enabling fall damage can be considered for the future. There is nothing in this user story that prevents future extensions from being implemented.

Valuable – This is an essential mechanic in the game. It needs to be implemented to achieve the basic functionality in the Santorini game.

Estimable – jumping down from any level building is a clear and estimable goal, e.g. descending from one level to another has been defined in the user story.

Small – a simple rule that expands upon the movement mechanics.

Testable – Acceptance criteria:

- A worker can jump down from any level to a lower level.
- A worker cannot jump onto occupied or restricted tiles.

13. As a Player, I want to be able to move up onto a building that is a maximum of one level above me, so that advancing to the third level requires strategic planning and gradual progression rather than sudden jumps.

Independent – This rule is specific to movement and can be implemented without depending on other features like building or god powers.

Negotiable – The implementation logic and method can both be discussed and refined.

Valuable – This rule enforces a key limitation in Santorini that makes climbing to the winning level requires strategic thinking, preventing overpowered or unbalanced gameplay.

Estimable – The requirement is clear and bounded: restrict upward movement to +1 level.

Small – This logic check is focused on a single movement constraint.

Testable – Acceptance Criteria:

- Attempt to move one floor upwards per time is allowed
- Attempt to move one floor downwards per time is allowed

14. As a player with a god card, I want to be able to use the associated god power to bend the rules of the game in a strategic way, so that I can gain a competitive advantage.

Independent - The user story does not require any other actions other than moving, so that it does not require another different feature for it. Also it is locked to only multiple floor scenarios.

Negotiable – The process can be adjusted by any game rules (god powers).

Valuable – Allows players to increase their strategic potential.

Estimable – The action is estimated easily as movement is a basic feature

Small – The scope of the action only focuses on one scenario.

Testable – Acceptance Criteria:

- God abilities are able to use by each player
- The god abilities works properly as it explains on the god card

15. As a player with Atlas, I want to be able to place a dome on any level, so that I can block my opponent from moving onto specific tiles.

Independent - this item is specific to the Atlas god's special power - there is no concern with any other god power.

Negotiable – flexibility in terms of (for example) how often/when this ability can be used by the player exists, and thus can be discussed for future extensions

Valuable – this is valuable as it gives players using Atlas a unique strategy which they can implement - making the game more interesting.

Estimable – Atlas' power is well defined and comprises exactly of placing a dome on any level. The scope is clear and hence, the effort should be able to be easily determined.

Small – a simple extension of the dome placement rule.

Testable – Acceptance criteria:

- Atlas can place domes on levels 1, 2, or 3.
- Domed tiles become inaccessible for movement

16. As a player, I want the game to end immediately if I'm stuck (unable to make any legal moves), so that I clearly understand when the game is over and can move on without waiting for actions I can't take.

I - standalone logics to check if the player is out of actions

N- Style of implementation can be discussed

V- prevents soft locks, which can be game breaking

E- method of checks can be roughly estimated

S- can be scoped to a check at the start of a turn

T- simulate the situation to verify if the game ends early

17. **As a Player, I want the game to end if one of the workers advances to a 3 storey building, so that I clearly understand when the game is over and can move on without waiting for actions I can't take.**

Independent – This win condition check is separate from other actions like building.

Negotiable – The presentation of the win condition can be adjusted by team communication.

Valuable – This is the core win condition in the standard game of Santorini. It is a must implement system for players to understand the game ended.

Estimable – It is easy to estimate the effort: monitor movement actions, check the destination tile's height, and trigger game-over if it's the third level.

Small – It's a single, well-contained rule that can be implemented with a simple condition detector.

18. **As a player, I want the board to visually indicate building levels, domes and workers clearly, so that I can assess the board situation and make thoughtful and strategic decisions.**

I- UI design is not tied to any gameplay logics

N- design can be discussed and easily refined

V- essential for players to see and make decisions

E- design can be estimated quite easily

S- UI elements can be added one at a time

T- can do user tests on how visible and distinguishable UI elements

are

EXTENSIONS:

19. **As a visually impaired person, I want there to be an audio description of my opponent's move, so that I know what move they made.**
 - **Independent** - Can be implemented independently of the core UI and visual effects.
 - **Negotiable** – Audio format (text-to-speech, pre-recorded, tone + text) and level of detail can be discussed.
 - **Valuable** – Makes the game more inclusive and accessible to visually impaired players.
 - **Estimable** – Scope is well-defined, triggering audio based on the opponent's move event.
 - **Small** – One specific feature triggered by a single game event
 - **Testable** – Acceptance criteria:
 - Simulate an opponent's move** and check if the correct audio description is spoken.
 - Disable or break audio output** and confirm fallback text appears for screen readers. (This one is generated using CoPilot)

20. **As a confused player, I want to be able to view game rules and god power explanations mid game, so that I can get back on track.**

Independent – The help menu or popup can be implemented separately from the core gameplay logic, making it a standalone feature.

Negotiable – The presentation (e.g. modal popup, expandable sidebar, hover tooltips) and UI/UX for displaying rules and god powers can be discussed and adapted based on player feedback and design preferences.

Valuable – Greatly improves user experience by reducing confusion during play, especially for new players or those using complex god powers.

Estimable – Content is static or sourced from predefined text assets, and the UI components (e.g. buttons, panels) are easy to visualize and estimate.

Small – It's a focused feature that doesn't involve deep integration—just a button and an information display, making it easy to complete in a sprint.

Testable – Acceptance Criteria:

- Ensure the rules and god powers are accessible during gameplay
- Check that the displayed content is accurate and matches the selected god
- Verify the popup closes and opens correctly without interrupting gameplay

21. **As a Player, I want to be able to roll a die when a god power is specified, so that I can use that god's power to my advantage.**

I - Die-rolling can be implemented without affecting other powers.

N - Dice type (e.g., d6), UI design, and trigger method can be adjusted.

V - Enables specific god powers to function correctly and adds gameplay variety.

E - Die roll function and UI are straightforward to implement.

S - Single feature triggered conditionally by specific god powers.

T - Acceptance criteria:

- **Trigger god power** that requires a die and verify the roll button appears.
- **Click roll** and check if the result is randomly generated and shown.
- **Use result** and confirm it enables or disables the god power effect correctly.

22. As a Player (with the Tyche god), I want to be able to roll a die every turn, so that I can move twice in a single turn, if even, or place two buildings in a single turn, if odd.

I - Tied specifically to Tyche, isolated from other god powers.

N - Details like die type (e.g., d6), UI interaction, and move/build flow can be refined.

V - Enables the core mechanic of Tyche's god power, which affects gameplay significantly.

E - Involves die logic + conditional branching for move/build, which are scoped and estimable.

S - Only affects one player per turn and follows a single trigger.

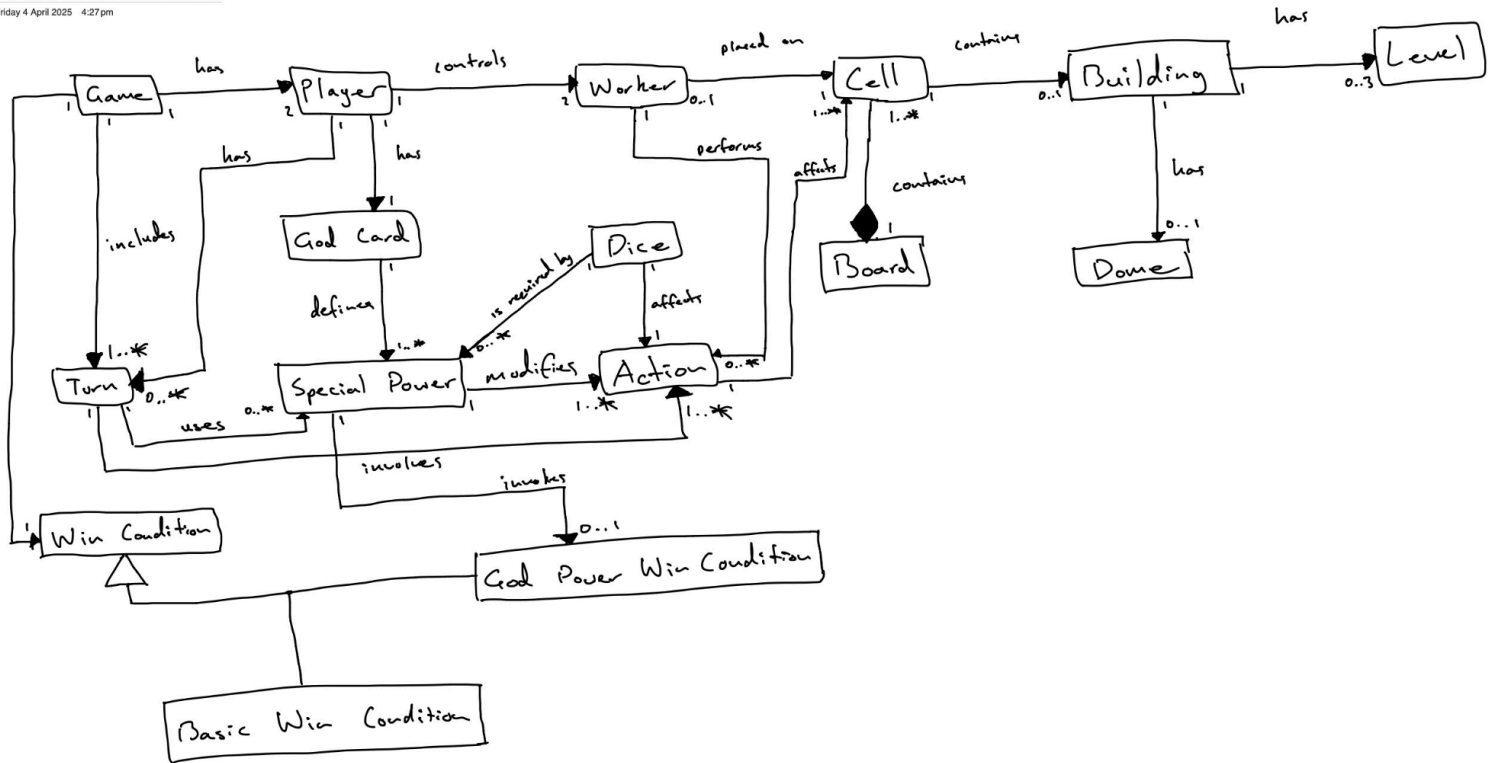
T - Acceptance criteria:

- **Roll the die at the start of a turn** and verify it shows a valid result (1–6).
- **If result is even**, confirm that the player can move twice in a single turn.
- **If result is odd**, confirm that the player can place two buildings after movement.

3. Domain Model

Domain Model 3rd Iteration

Friday 4 April 2025 4:27 pm



3.1 Domain Model Justification (design rationale)

3.1.1 Rationale and Justifications

1. Entities and Relationships

- **Game:** Central entity that encapsulates the state and flow of the gameplay.
- **Player:** Represents one of two participants; each manages two Workers and assigned a GodCard. But the game entity cannot be there without a player. Also, each player is assigned 1 god card and 1 god card can be assigned by 1 player.
- **Worker:** Each player has two workers, and workers perform actions and move on the board. So, a player must have 2 workers and each worker is assigned to 1 player. Each worker is on exactly one cell at any time but a cell can be empty or contain one worker.
- **Board / Cell:** Board is a grid; each grid square is a Cell. Workers and buildings occupy cells. Many cells can be a part of a board but there is only 1 and only board.
- **Building / Dome / Level:** Buildings are constructed level-by-level (1–3), with the fourth level being a Dome. A cell can have 0-3 level building but each building needs to be placed on a cell.
- **Turn / Action:** A turn involves player actions (Move, Build, etc.). Capturing actions explicitly allows extension (e.g., skipping moves or special builds).
- **GodCard / SpecialPower:** Each card can grant unique movement, build, or win condition capabilities. This is critical in modeling the extension.
- **WinCondition:** Separating this allows different ways of winning to be defined or overridden (especially with GodCard win conditions).

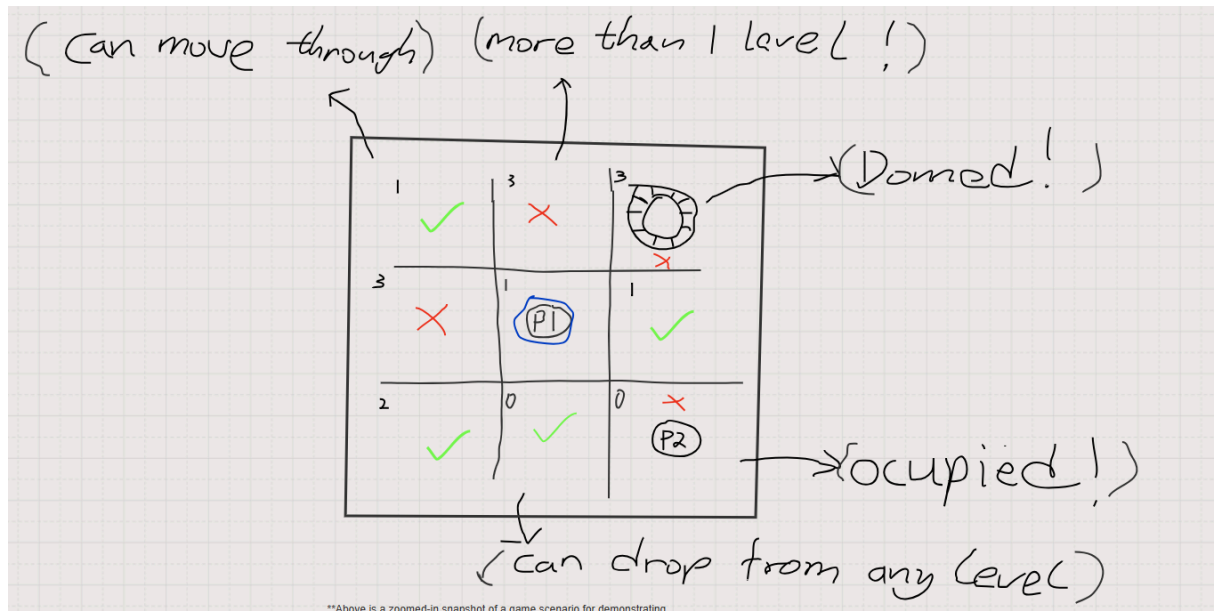
3.1.2 Modeling Decisions & Assumptions

1. **Modeling Cell over Coordinate:** Instead of raw coordinates, Cell allows us to attach behavior like “contains Worker” or “contains Building” so that it is easier to choose which cell is in action.
2. **Modeling Action as an entity:** Instead of bundling movement and building into Turn, breaking them into Action allows for more flexible game flow (e.g., Minotaur’s push).
3. **Distinct Building and Dome:** We modeled domes as distinct so that cards like Atlas (can build domes at any level and recognize full building etc) are easier to model conceptually.
4. **Separate WinCondition:** Allows implementation of alternate win conditions from god powers without rewriting base rules or checking unnecessary win condition check when the god card isn’t in the game instance.
5. **Extension Tyche GodCard:** This god card utilizes a new mechanic using a dice. So, we had to create a new entity that is related to the god card (so that it is recognized as god card and available from the deck) and action (so that the dice determines the action instead of the special power.)
6. **Excluded more than 2 player mode:** We made an assumption that based on the discussions on Edstem, that the game would not have different gamemode (e.g. 2v2, 3-way).
7. **GodCard unique:** Based on the rulebook and the online version, 1 god card can be only picked by 1 player. No duplicate in the same game.

4. Basic UI Design

Scenarios:

1. Moving a worker to a higher level, Moving a worker to tile that is on the same level (Aaron)
2. Moving a worker to a lower level (Aaron)



(Above is a zoomed-in snapshot of a game scenario for demonstrating

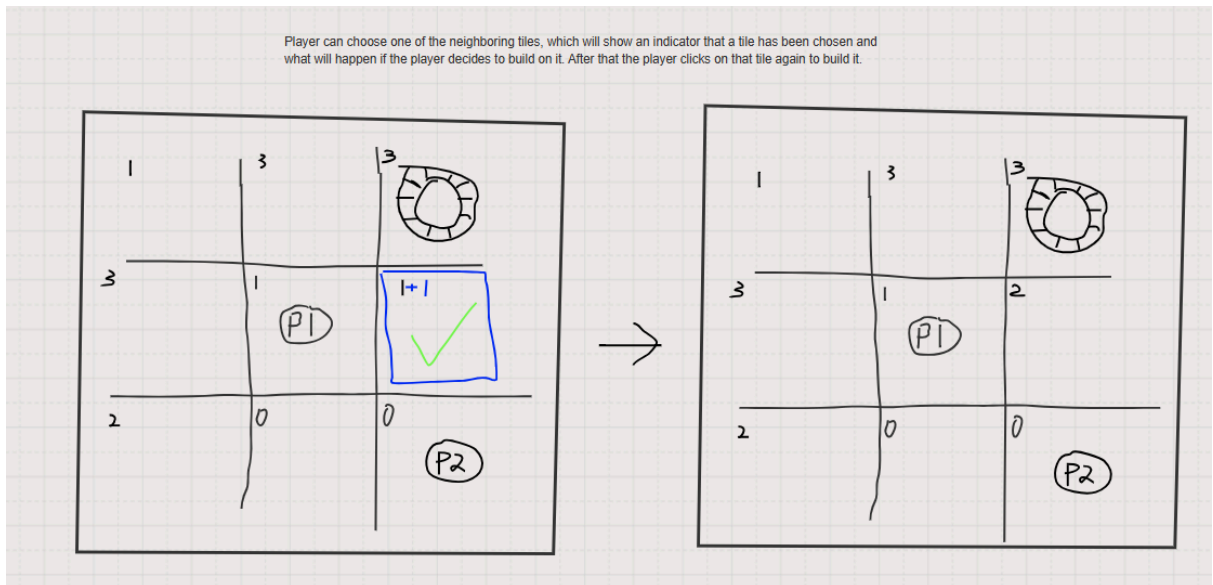
When the worker is selected, it will be highlighted in blue

Player can then click on any tile to try to move that worker to the selected tile

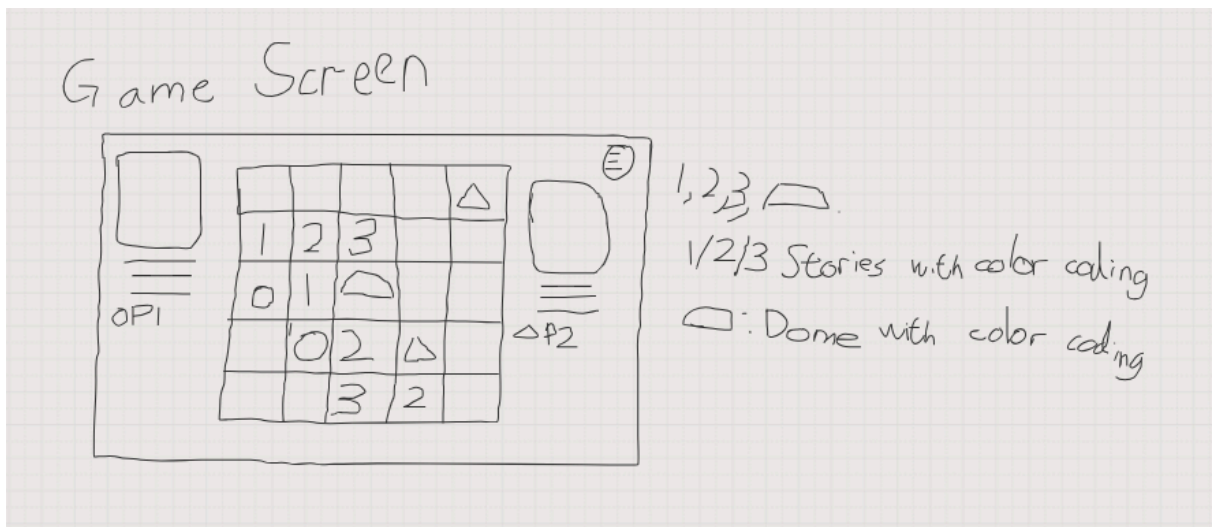
if the move is legal(indicated by green arrows for Low-Fi purpose only), then the worker will be transferred to that tile

if not(indicated by red crosses for Low-Fi purpose only), then the worker will not move)

3. Building a story (1, 2, 3 and a dome) (Uchralt)



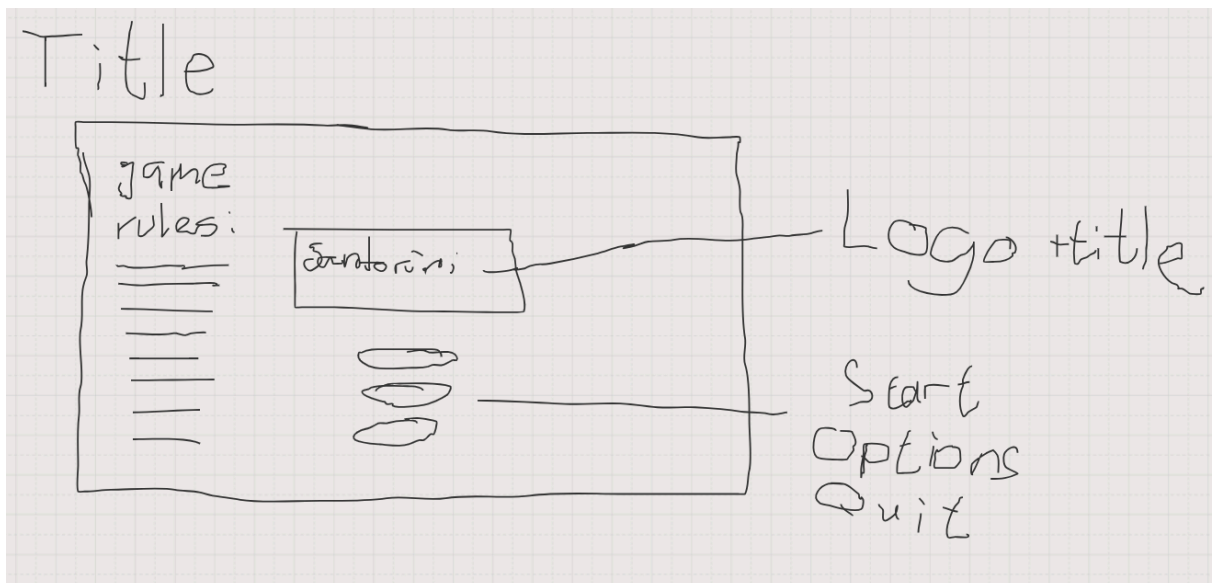
4. Basic Game UI (Allan)



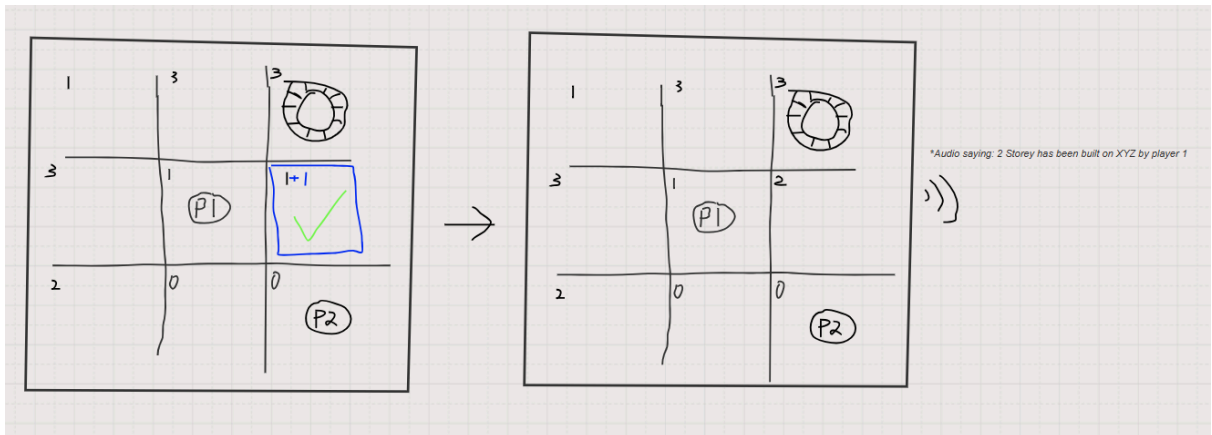
5. Winning (by the default rule and by the god card) (Ubaid)



6. Welcome page (Options, Quit, Start Game) (Allan)



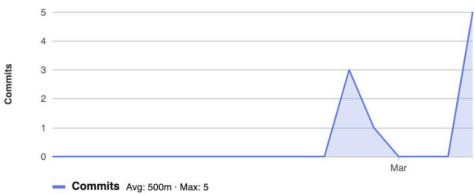
7. Extension: Audio support for hearing loss (Uchralt)



5. Contributor Analytics:

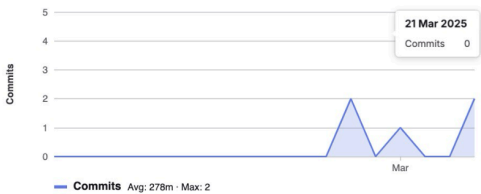
Ubaid Irfan

9 commits (ubaidirfan01@gmail.com)



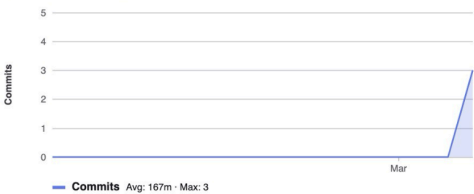
khoo0050

5 commits (khoo0050@student.monash.edu)



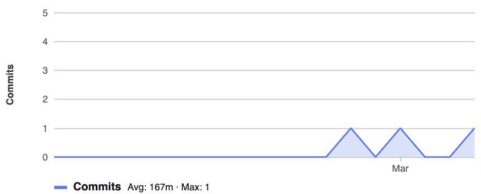
umya0001

3 commits (umya0001@student.monash.edu)



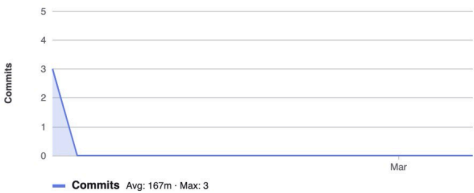
azho0020

3 commits (azho0020@student.monash.edu)



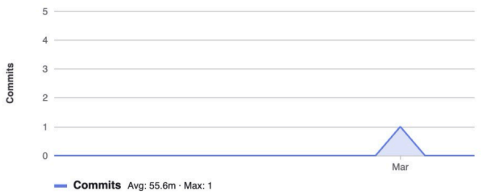
Jordan.Nathanael

3 commits (jordan.nathanael@monash.edu)



khoo0050

1 commit (khoo0050@studnet.monash.edu)




6.Team Document Links:

Team Google Docs link:

 FIT3077 Team Santorinians (005) A1 Deliverables - Team Information ...

Sprint 1 Contribution Log (Live):

 Contribution Log

The contribution log is also located at this location in the Git repository (as required):

https://git.infotech.monash.edu/FIT3077/fit3077-s1-2025/assignment-groups/CL_Tuesday06pm_Team005/project/-/wikis/Sprint-1-Contribution-Log