**Champion Document**

Ian Fleming

1. **Brief Introduction**

This Champion document will feature the Grid Manager and Scoring System implementations for our game, Tile Hero. The Grid Manager will be responsible in the initialization of the game’s grid tiles which will function as the traversable and interactive map. The Scoring System will store and increment the game’s scoring counter to simulate the player’s skill (or progression). The document will present use case diagrams with scenarios, data flow diagrams, acceptance tests, and timelines.

1. **Use case diagram with scenario**

Grid Manager use case:

Diagram

Description automatically generated

Grid Storage

Scoring System use case:

**Scenarios:**

**(Grid Manager)**

**Name:** Initialize Tile Placement

**Summary:** Receives a tile placement request and initializes the tile on the grid system/map.

**Actors:** Tile Placement System

**Preconditions:** The player has started a game initializing the environment.

**Basic sequence:**

1. Player selects a tile type from the inventory.
2. Player places the tile on desired grid location.
3. Grid Manager receives request and initializes the tile.
4. Initialized tile is stored in the grid storage.
5. Tile is revealable to the player and the game proceeds.

**Exceptions:** None

**Post conditions:** Placed tile is now perceivable and playable on the map.

**Priority:** 1

**ID:** GM01

**Name:** Check if Tile Placement is Valid

**Summary:** Receives a request to determine if the tile placement is valid to place.

**Actors:** UI Manager/Tile Placement

**Preconditions:** The player has started a game initializing the environment.

**Basic sequence:**

1. Player selects a tile type from the inventory.
2. Player places the tile on desired grid location.
3. Grid Manager checks if the location of the placement is valid.
4. The tile is accepted or rejected to be placed.
5. A visual queue will determine if the placement is valid or not.

**Exceptions:**

3. Grid Manager detects that the location of the placement is not valid.

4. The placement of the tile is rejected and not initialized.

5. The player must choose another location for the tile to be placed.

**Post conditions:** The tile is either successfully placed or rejected.

**Priority:** 1

**ID:** GM02

**Name:** Return Tile Information

**Summary:** Grid Manager returns requested data to the designated system.

**Actors:** Movement

**Preconditions:** The player has started a game initializing the environment.

**Basic sequence:**

1. Player moves to a new tile.

2. Movement system sends a request to Grid Manager.

3. Grid Manager system returns the information of this new tile.

4. The game proceeds as the new tile is the current space of play.

**Exceptions:**

3. Grid Manager system returns a NULL tile.

4. Player is unable to move to this tile as it is not yet initialized.

5. Player must choose another tile to enter.

**Post conditions:** Player is able to successfully traverse about other tiles.

**Priority:** 1

**ID:** GM03

**Name:** Send Grid Config Update

**Summary:** Grid Manager sends grid configuration data to Grid Storage.

**Actors:** Grid Storage

**Preconditions:** Grid Storage has been initialized on game start up and a change has been made to tiles in the Grid Manager.

**Basic sequence:**

1. Grid Manager detects an update of information has been made to a tile.

2. Sends this information to Grid Storage to be stored.

**Exceptions:** None

**Post conditions:** Game proceeds and Grid Storage has most recent grid data.

**Priority:** 2

**ID:** GM04

**Name:** Return Current Grid Config

**Summary:** Handles all the current grid configuration data and sends them to systems.

**Actors:** Grid Manager

**Preconditions:** Grid Storage has been initialized on game start up.

**Basic sequence:**

1. A request for data has been received from Grid Storage.

2. Returns this information to Grid Storage.

**Exceptions:** None

**Post conditions:** Grid Storage contains the current grid configuration.

**Priority:** 1

**ID:** GM04

**Name:** Check for Loot/Items

**Summary:** Grid Manager accesses tile information to check if loot is present.

**Actors:** Inventory Manager

**Preconditions:** Player successfully clears a tile to begin searching for loot.

**Basic sequence:**

1. Player clears dungeon of enemies and searches for loot.

2. Grid Manager checks tile if loot is present.

3. Returns this data.

**Exceptions:**

2. Grid Manager checks tile and no loot is present.

3. Returns this data that no loot is in the current tile.

**Post conditions:** Tile has been searched for loot.

**Priority:** 2

**ID:** GM05

**Name:** Access Tile Properties

**Summary:** Function to access the properties of a certain tile.

**Actors:** Inventory Manager

**Preconditions:** Tile must be initialized and is the current playable tile.

**Basic sequence:**

1. The player is playing in the current cell/tile.

2. Inventory Manager sends a request to fetch tile properties.

3. Information of the tile is returned.

**Exceptions:** None

**Post conditions:** Inventory Manager is aware of the properties of the tile.

**Priority:** 1

**ID:** GM06

**Name:** Add Loot to Inventory

**Summary:** Loot found in the current cell is added to the player’s inventory.

**Actors:** Inventory Manager

**Preconditions:** Player clears the cell of enemies and searches for loot.

**Basic sequence:**

1. Loot is found in the current cell.

2. Loot is added to the player’s inventory.

**Exceptions:**

1. Loot is not found in the current cell.

2. The player’s inventory is left unchanged.

**Post conditions:** Whatever items the player has found is added to the inventory or, if none is found, the inventory is untouched.

**Priority:** 1

**ID:** GM07

**(Scoring System)**

**Name:** Detect if Enemy Killed

**Summary:** Determines if the player’s current cell is rid of all enemies.

**Actors:** Player

**Preconditions:** The current tile has enemies in which the player can defeat.

**Basic sequence:**

1. Player enters tile with enemies.

2. Returns if an enemy has been killed.

**Exceptions:**

2. Nothing is returned if an enemy has not been killed.

**Post conditions:** Current cell is either rid or not rid of enemies.

**Priority:** 1

**ID:** SS01

**Name:** Increment Score Counter

**Summary:** Increases the score counter of the game by some value.

**Actors:** Player

**Preconditions:** Enemy must be killed.

**Basic sequence:**

1. Enemy is killed.

2. Add some value to score counter.

**Exceptions:** None

**Post conditions:** Score counter is incremented from previous value.

**Priority:** 1

**ID:** SS02

**Name:** Return the Score Counter

**Summary:** Returns the current score counter to the inventory manager.

**Actors:** Inventory Manager

**Preconditions:** Scoring system is initialized.

**Basic sequence:**

1. Inventory Manager requests score counter.

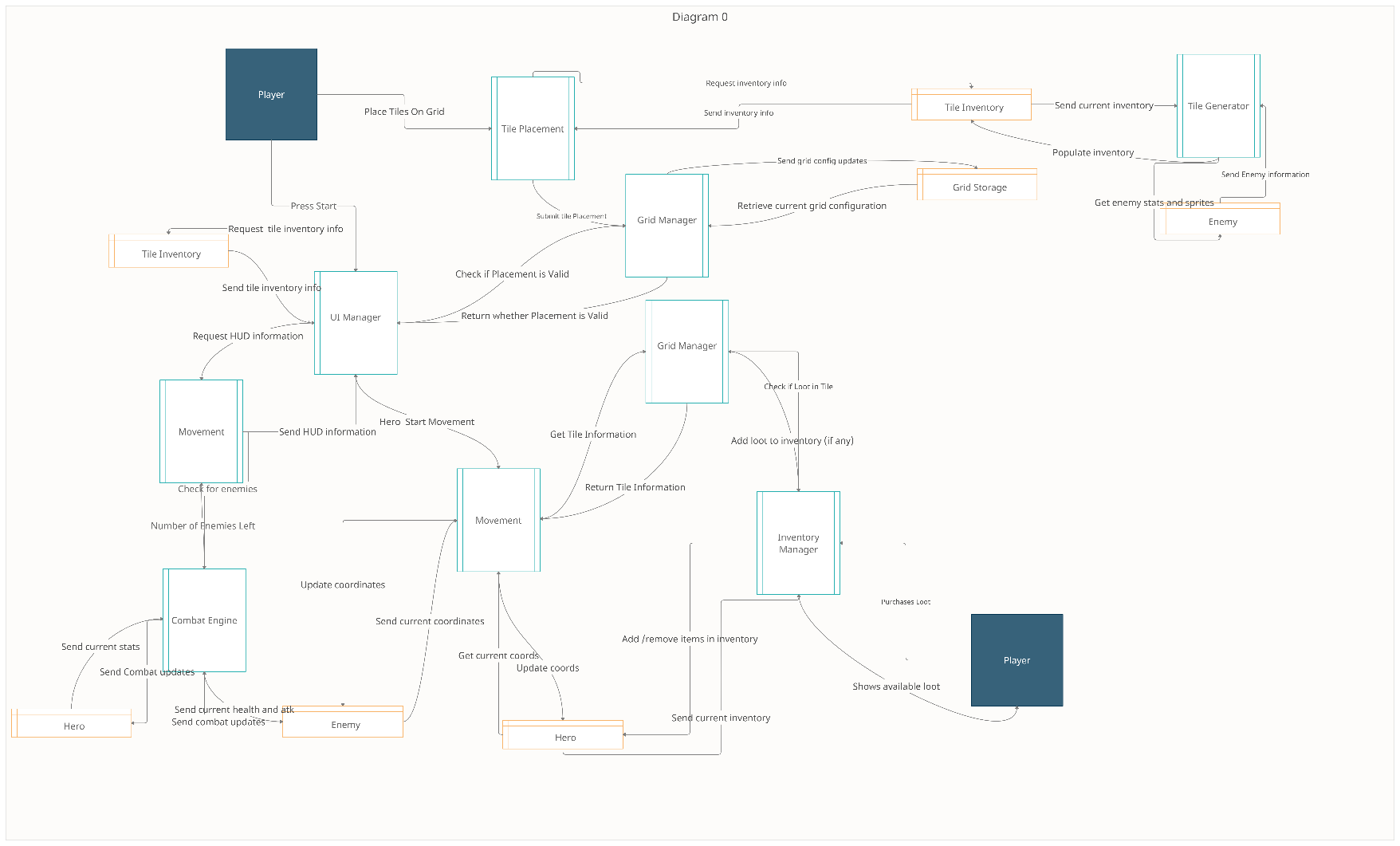
2. Score counter is returned.

**Exceptions:** None

**Post conditions:** Current score core is viewable in the inventory.

**Priority:** 2

**ID:** SS03

**3. Data Flow Diagram**

Scoring System feature is a subprocess in the Inventory Manager and thus isn’t perceivable on our Diagram 0.

**Process Descriptions**

Grid Manager:

WHILE The game is initialized, and the player started play

IF Request for tile placement THEN

IF Valid placement THEN

Initialize tile

ELSE Select other tile placements

IF Request from Inventory Manager THEN

IF Tile has loot THEN

Add to inventory

ELSE Add no items to inventory

IF Request from Movement THEN

RETURN Tile Information

IF Request from Grid Storage THEN

RETURN Current grid configuration

END WHILE

Scoring System:

WHILE The game is initialized, and the player started play, and Inventory Manager loaded

IF Enemy is killed THEN

Increment score counter (gold)

END WHILE

**4. Acceptance Tests**

**Example for Grid Manager feature:**

**\***In the following examples, the inputs are “requests” in that they are function calls to functions contained in Grid Manager. No parameters are necessarily passed.

If the Grid Manager is requested a tile placement, it will output a true or false. If true then the tile will be initialized into data and stored into the tile storage and a visual graphic will reveal. If false, then the tile will be NULL, and visuals will remain the same.

If the Grid Manager is requested from the Inventory Manager, then a Boolean value will be returned to the Inventory Manager. If true, then the current tile has loot and is added to the inventory. If false, then no loot is added to the inventory and a message will appear.

If the Grid Manager is requested from Movement, then tile information will be returned which will be an object.

If the Grid Manager is requested from Grid Storage, then the current grid configuration will be returned which will be a 2D matrix of objects.

**Example for Scoring System feature:**

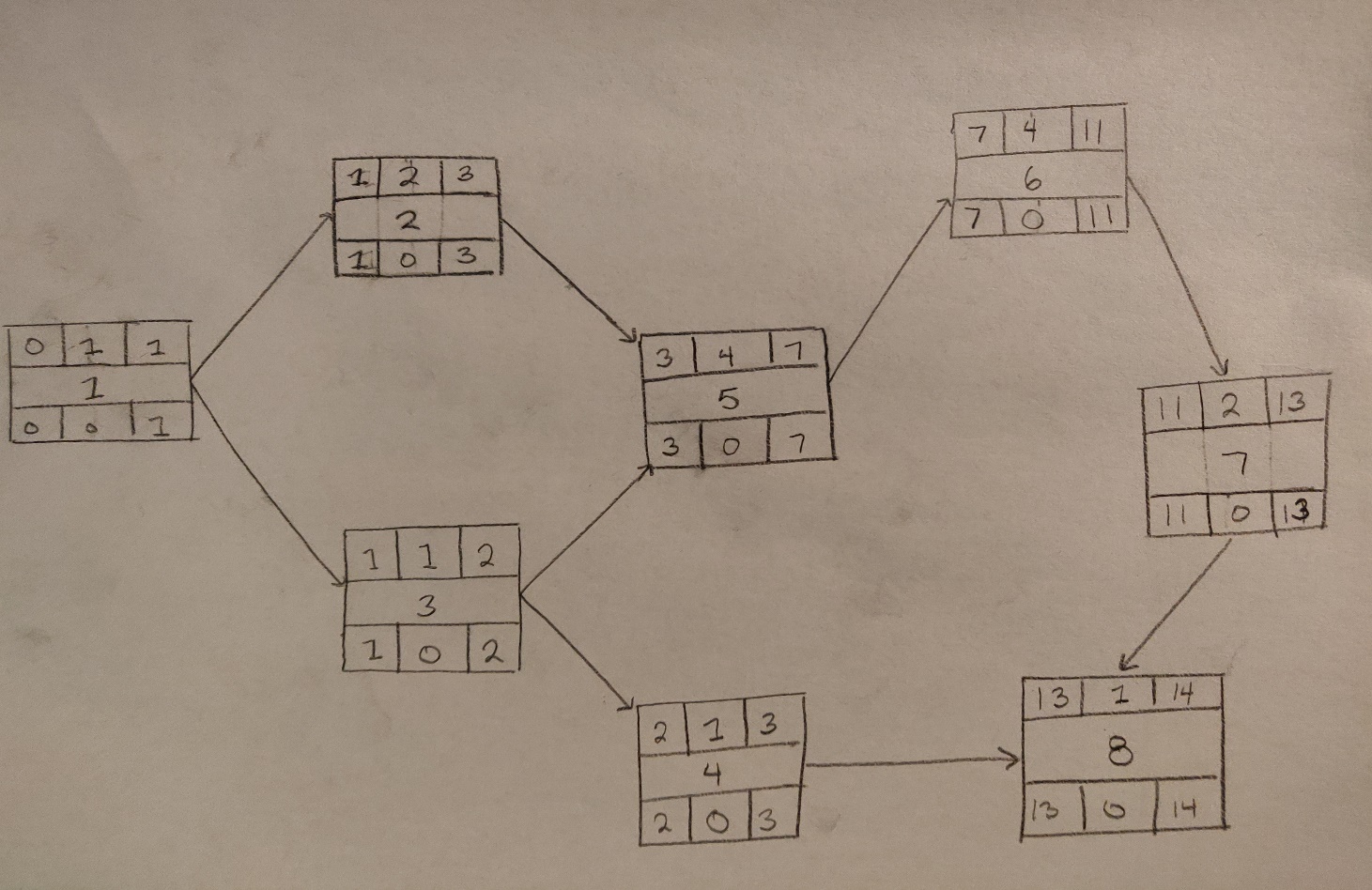
If 5 enemies are in the player’s current tile and the player manages to successfully defeat the enemies then the player’s score will increment to 5 gold.

5. Timeline

**Work Items**

|  |  |  |
| --- | --- | --- |
| **Task** | **Duration (PWks)** | **Predecessor Task(s)** |
| 1. Preparation/Planning | 1 | - |
| 2. Pseudo Code | 2 | 1 |
| 3. Report Design | 1 | 1 |
| 4. User Documentation | 1 | 3 |
| 5. Grid Storage Construction | 4 | 2, 3 |
| 6. Programming | 4 | 5 |
| 7. Testing | 2 | 6 |
| 8. Installation | 1 | 4, 7 |

**Pert Chart**



]

**Chart

Description automatically generatedGantt Chart**