The Alchemist

Technical Design

Overview

This document covers the high-level systems design and management systems used to build The Alchemist. This game will be event system driven.

Game Managers

There will be a variety of game managers to handle the various systems in The Alchemist

GridManager

The level manager will handle meta elements of gameplay (e.g. Validating player/enemy movement, attack vectors, etc.)

Handling Overlays

A variety of information will need to be conveyed to the player (e.g. enemy's next movement, terrain updates, etc.) The level manager will also dispatch changes to terrain to individual cubes.

Managing the grid

The grid is the main play field for the player. The grid consists of a block of cubes that handle actions taken on that cube. Distance is managed by the cube itself in comparison to its neighbors, rather than the player handling them.

SceneManager

The scene manager is a simple manager that will handle the following:

- Switching of UI Elements
- Loading and populating necessary information for each scene
- Keeping track of ApplicationState (e.g. pause, quit, end of scene, etc.)

EventManager

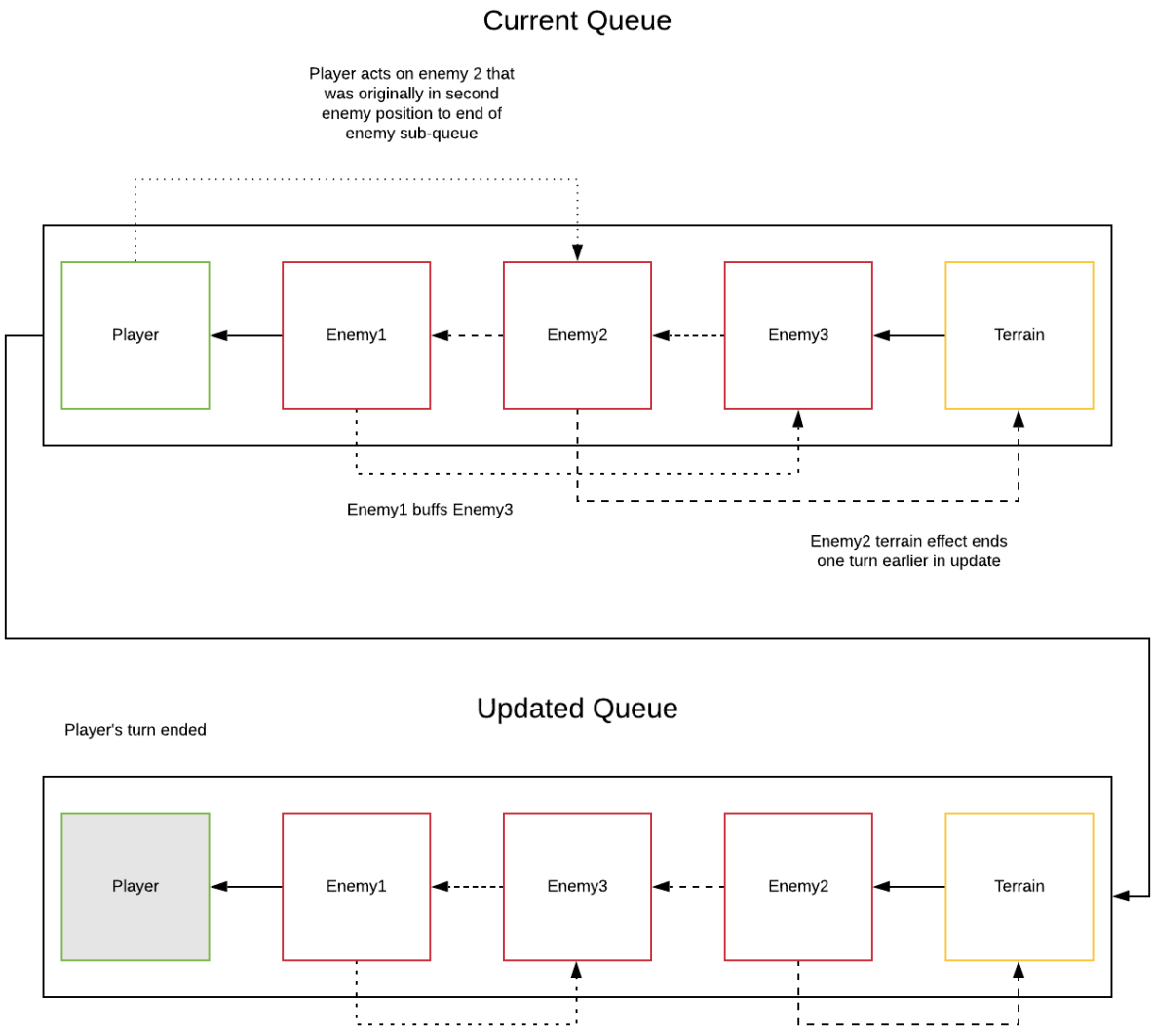
Simple manager that handles the triggering and execution of events.

TurnManager

The turn manager will handle turn queue as well as initiating of actions. Presumably the turn manager will need to start and stop enemy movement as well as keep track of the current state of enemies on the field.

Queue

The queue is an immutable List of the upcoming and previous turn lists that will be and have been acted on. Previous turn queues will be stored until the end of scene but will not be in use at the moment.

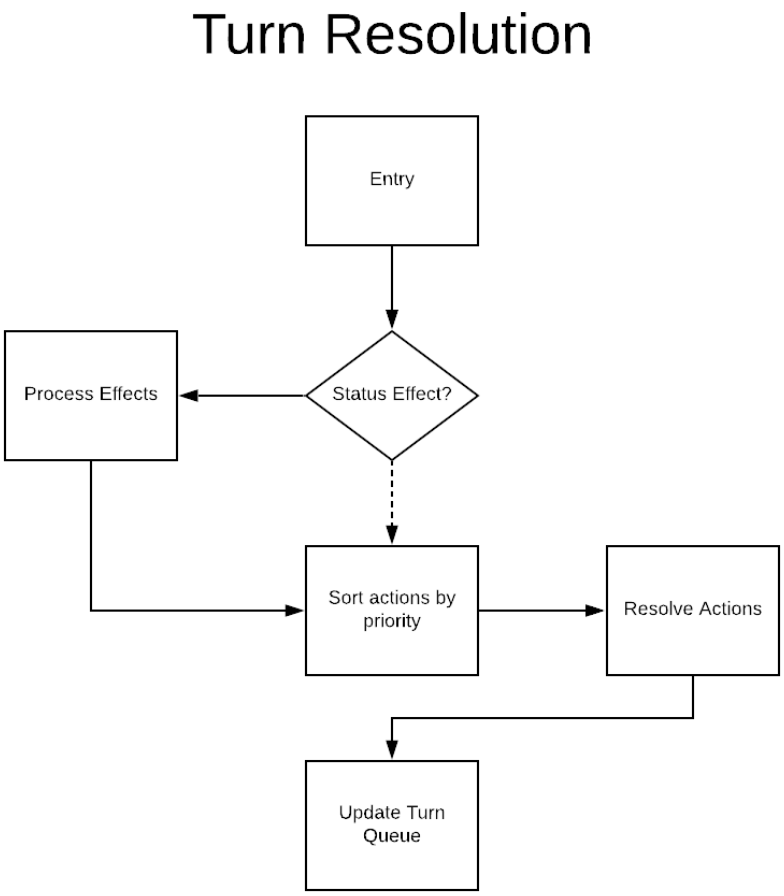


Managing Turns

Turns are handled in the following order:

1. Player
    - Actions that the player takes
    - Updating enemy turn order, if necessary
2. Enemy
    - Resolving enemy actions, in turn
    - Queuing of actions for next turn
3. Terrain
    - Resolving terrain changes, mostly reversions

The terrain actions are handled by the GridManager and dispatched to each individual cube on the grid. TurnManager is invoked to update the queue. A queue cycle is completed on the resolution of the Terrain turn.



Actions

Actions are the representation for every movement and state change. Player actions are resolved immediately, whereas Enemy and Terrain actions are queued at the end of each turn.

Events

Event Name	Payload (DataType)	Priority
REMOVE_EFFECT_UNIT	Unit Name (string) Enum (EFFECT_TYPE)	3
REMOVE_EFFECT_TERRAIN	Grid Position (string) Enum (EFFECT_TYPE)	3
ADD_EFFECT_UNIT	Unit Name (string) Enum (EFFECT_TYPE)	4
ADD_EFFECT_TERRAIN	Grid Position (string) Enum (EFFECT_TYPE)	4
MOVE_UNIT	currentPosition (string) newPosition (string)	1
USE_ITEM	Grid Position (string) Enum (ITEM_TYPE)	2
DAMAGE_UNIT	Name (string) Damage (integer)	1

Systems

Potions

The potion system is the core gameplay system of The Alchemist.

Potions will be represented in Unity ScriptableObjects which are capable of storing data that is easily passed around between systems (e.g. Inventory, GridManager, etc.)

Properties

	DataType	Use	Values
Name	String	Potion Title	
Icon	Sprite	UI representation of potion	
Description	String	Information for UI and Recipe List	
Effect Range	Integer	Grid unit area of effect	
Weight	Integer	Penalty against throwing range	
Duration	Float	Number of turns that the effect will be enabled	
Type	Enum (POTION_TYPE)	Representation of category of effect	Healing, Debuff, Terrain, Movement, etc.
Effect	Enum (EFFECT_TYPE)	Representation action to be taken by effect	Ignite, Destroy, Heal, etc.
Potency	Float	Stat multiplier	
Wait Time	Integer	Turns before potion can take effect	

Combining Potions

This system will be handled by calculating a value based on the combination of the Effect and Type potion properties.

Example:

Name	Type	Effect	
Small Healing Potion	BUFF	HEALING	
Fire Potion	TERRAIN	AOE	+
Healing Fire Potion	TERRAIN	HEALING	

In this example, no matter the two potions added in, if one is of BUFF/HEALING and another of TERRAIN/AOE, it will always result in the Healing Fire Potion with TERRAIN/HEALING.

Materials

Properties

	DataType	**Use**	**Values**
Type	Enum (MATERIAL_TYPE)	Representation of	Blood, Water, Rock, Metal
Quantity	Integer	Count of	
Icon	Sprite	UI Representation of Material	
Name	String	Flavor-text name	
Bundle	Integer	Stackable bundle count	

Crafting

The crafting system encompasses both potion making and combining of potions.

Inventory

Properties

	DataType	Use	Values
Slots	Integer	Max Inventory Spots	
Items	Array of ITEM	List of items (materials, potions, armor)	
Type	Enum (INVENTORY_TYPE)	Designates type of Inventory	Personal, Chest

Grid

The grid is the total collection of grid units: cubes representing movement and attack vectors. Each grid unit has a state and list of properties and manages such.

Properties

	DataType	Use	Values
Position	String	Position name from GridManager	
Icon	Sprite	UI Representation of Affected Land	
Effect	Array of Enum (EFFECT_TYPE)	Representation of potion effect	Ignite, Destroy, Heal, etc.
Duration	Integer	Turns to state reversion	
Traversable	Boolean	Can units move through	
Occupied	Boolean	Unit occupying space	

Grids can be acted upon on every turn of battle, but don't end effects until the Terrain turn.

States

The player, enemy, and grid tiles will all keep track of their own state.

Unit

Properties

	DataType	Use	Values
Name	String	Name of unit	
Health	Integer	Health of unit	
Action Count	Integer	Allowable actions	
Action Queue	Array of Action	List of actions to take	
Effect Queue	Array of Enum (EFFECT_TYPE)	List of Effect to apply	
Inventory	Array of Inventory	Items available for use by Unit	
Position	String	Position name from GridManager	

Effects

Properties

Burning
Icy
Moutainous