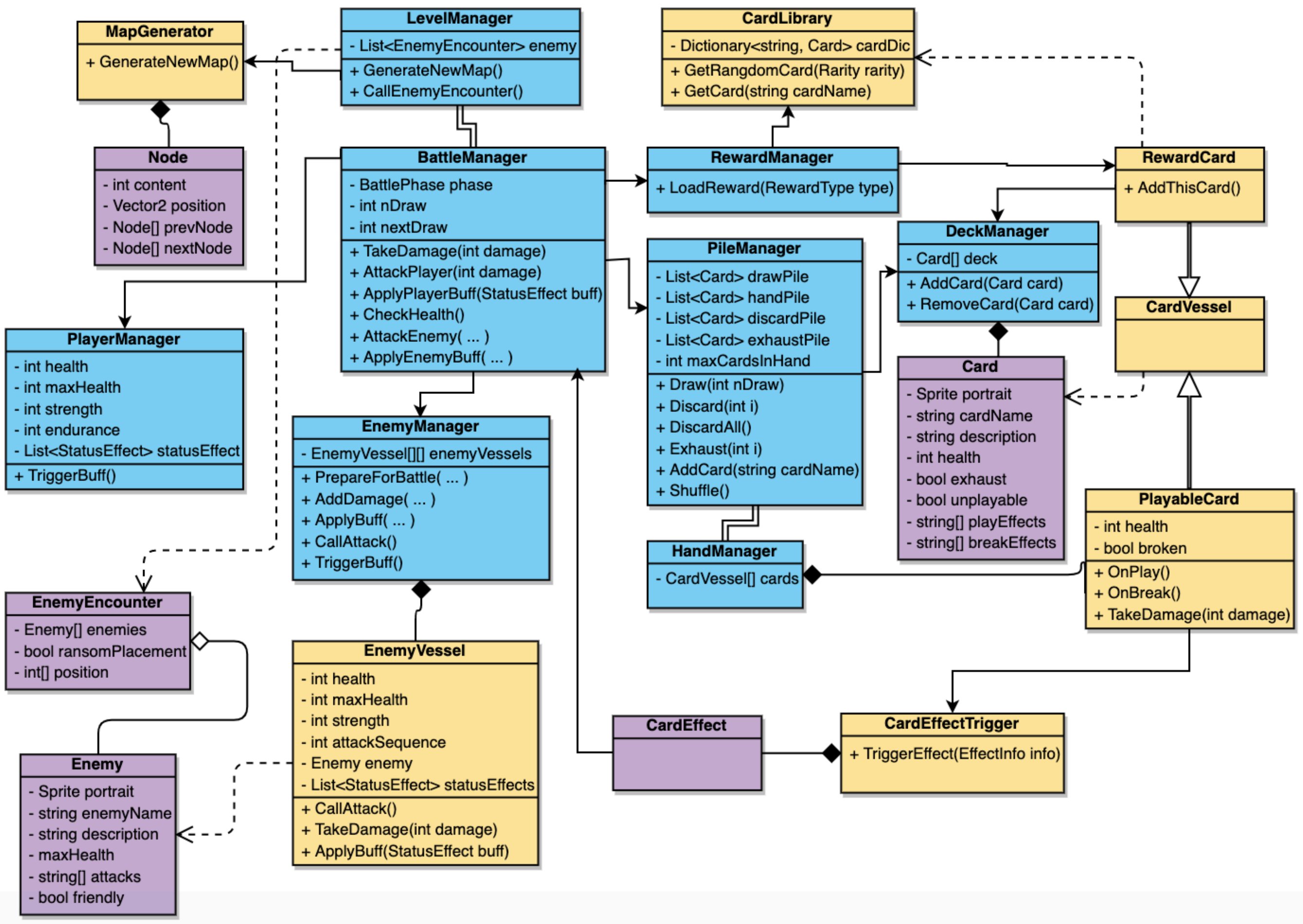
Spire-like

Catalogue

1. System Overview



2. Components

The components can be divided into several types. “Game controller”, which handles the flow and states of the game; “Card-related”, which deals with card effects and their encoding/decoding; “Enemy-related”, which deals with enemy stats and player interactions, as well as their encoding/decoding method; “Display”, which deals with minor graphical details to improve game feel; “Misc”, which does not belong to above categories and may turn into its respective category as development goes.

2.1 Game controller

2.1.1 Level Manager

Description

Level manager handles the “Worlds and Levels” level tasks. It procedurally generates the map through **MapGenerator,** decides which EnemyEncounter the player confronts in the next level, and keeps track of which level the player is at.

Development tasks

Currently, it is only suitable for one world, i.e. does not support world transitions and refresh enemy encounters according to the world. All nodes are currently enemy encounters, future nodes would randomly contain enemy, reward, shop, etc.

Public parameters:

ready

Whether this script has completed initialization

Public functions:

CallEnemyEncounter(NodeObject nodeObj)

Triggers when player selects the next node to move toward, and decides which action to take according to node type specified in **Node** or **NodeObject**(not yet implemented). If the selected node is the last node, it calls the **EnemyManager** to prepare for boss encounter, specified in **EnemyEncounter boss**, or else it chooses a random enemy encounter specified in **List<EnemyEncounter> enemy** for the **EnemyManager.** To prevent repeated enemy encounter, the called encounter will be removed from the list.

ShowMap() / HideMap()

Show/Hides the map screen specified in **GameObject[] mapScreen**.

RefreshAvailablePath()

Called after completing a level. Shows the available nodes the player can progress.

2.1.2 BattleManager

Description

BattleManager handles the phase of the battle, and acts as a bridge between card effects and their targets, e.g. player, cards in hand, enemy.

Phases:

Init > Begin > Draw > Play > Friend > Enemy > Discard > Draw > Begin > …

Win/ Lose

Init : Initializes the components the first time this object is loaded.

Begin: Sets up the enemies according to the information in EnemyEncounter and initialize vars

Draw: Draw cards

Play: Player action

Friend: Player summons action

Enemy: Enemy action

Discard: Discard all cards in hand

Public parameters

handsize : Determines how much cards player draws per turn
nextDraw : Determines how much extra cards player draws next turn, affected by card effects.

Public functions:
IEnumerator Begin(EnemyEncounter enemyEncounter)
Calls **EnemyManager** to set up the enemies according to enemyEncounter. Initialize temporary stats in battle such as status effects.

void TakeDamage(int damage)
Distributes incoming damage to the cards in hand. If cards are unable to withstand the damage, the residual damage is applied to the player.

void AttackPlayer(int damage)
Deals damage directly to player

void AttackEnemy(Vector 2 center, int attack, Array2D[] range)
Calls **EnemyManager** and deals damage to enemies with **center** as center, **range** as weights, and **attack** as attack damage.

Void ApplyEnemyBuff(StatusEffect buff, Vector2 center, Array2D[] range)
Calls **EnemyManager** and deals damage to enemies with **buff** as status type, **center** as center, **range** as weights.

Void ApplyEnemyBuff(StatusEffect buff)
Calls **PlayerManager** and deals damage to enemies with **buff** as status type.

2.1.3 DeckManager
Manages the current deck the player is using

Public variables:
List<Card>deck : player deck
ready : Whether it has finished initialization

Public functions:
Void AddCard(Card card)
Adds **card** into **deck**

Void Remove(Card card)
Removes **card** from **deck**

2.1.4 PileManager
Manages the cards in piles (draw, play, discard, exhaust). Most card arrangements are done here.

Public variables:
bool ready : Whether it has completed initialization.
int maxCardsInHand : Maximum cards allow in handPile
List<Card> drawPil : discardPile, exhaustPile, handPile

Public functions:
IEnumerator PrepareForBattle()
Initializes cards in pile for battle. It removes all cards in each pile, and copy cards from deck.

void Shuffle()
Moves discardPile into drawPile.

void Draw(int drawNum)
Draws **drawNum** cards, does not allow card drawing if cards in handle reaches **maxCardsInHand**. Card drawing is done randomly from draw pile.

Void Discard(int i)
Void DiscardAll()
Void Exhaust(int i)
AddCardToHand(string cardName)
AddCardsToDraw(string draw)
AddCardToDiscard(string cardName)

2.1.5 HandManager
Manages the interface of cards in hand, serves as a bridge between card effects and **pileManager**.

Public variables:
bool ready : Whether it has completed initialization.
bool selectState : state of card selection which may happen when the player discards a card by card effects
bool selected : True when player clicks confirm during selection.

Public functions:
void Refresh()
Refreshes each display, including PlayableCard and marker position

void PlayCard(PlayableCard toPlay)
Called when a card is played, and calls pileManager to discard or exhaust the said card.

void Discard(PlayableCard toDiscard)

int TakeDamage(int damage)
Called by BattleManager, distributes damage onto the cards in hand, returns the residual damage that is not blocked by the hand.

void SetSelectState(bool state)
void SelectCard(PlayableCard selection)
PlayableCard GetSelection()
void ConfirmSelection()

void SetInteractable(bool interactable)
Toggles the availability of the cards in hand. They should not be intractable outside of Play state in BattleManager.

2.1.6 EnemyManager
Manages the enemies on the board, a bridge between **BattleManager** and all **EnemyVessel**s.

Public variables:
int slotDim : How many rows/columns there are in the board.
bool attacking: States whether the enemies are still performing their attacks.
bool ready: Whether initialization is completed.

Public functions:
void PrepareForBattle(EnemyEncounter enemyEncounter)
Sets up the board for a new battle with the information contained in **enemyEncounter**.

Vector2 ShowRange(Vector2 position, Array2D[] range)
Calculates the location on board from **position** and changes the **EnemyVessel** display according to **range**. It returns the location calculated from **position**, which is the typically the center of the card effect. If the **position** does not overlap any **EnemyVessel**, it returns (-1, -1), which means that it is out of range.

void AddDamage(Vector2 center, int damage, Array2D[] range)
Adds the damage onto a matrix **damageCounter** with values calculated from the inputs. **Center** determines where the center(2,2) in **range** locates on the board, **range** as the weights that will be multiplied by **damage**.

void ApplyBuff(StatusEffect buff, Vector2 center, Array2D[] range)
Similar to **AddDamage**, but applies **StatusEffect** instead.

IEnumerator CallFriendAttack()
Calls each player summons to take action.

IEnumerator CallEnemyAttack()
Calls each enemy to take action.

2.1.7 RewardManager
Determines the drop rate of each card type, retrieves the reward cards from **CardLibrary**.
The drop chance is hard-scripted in **GetChanceArray**()

Public variables
enum RewardType : the types of reward.
bool ready : Whether it has completed initialization.

Public functions
void LoadReward(Reward Type)
Refreshes the reward screen with reward cards chosen according to the type of reward.

void ShowRewardScreen()
void HideRewardScreen()

2.1.8 PlayerManager
Stores player stats and manages display of player status.

public variables:
int health
int maxHealth
int strength : Adds x extra damage to each attack
int endurance : Adds x extra health to each card

Public functions
IEnumerator PrepareForBattle()
Removes all temporary effects.

void Heal(int power)
void TakeDamage(int damage)
bool IsDead()
void ApplyBuff(StatusEffect buff)
void TriggerBuff()

2.2 Card structure
Each **Card** contains the static information of the card.
It is interfaced with player through **CardVessel**, which contains dynamic information generated by copying the information in **Card**.

2.2.1 Card
Variables:
Sprite portrait
string description, cardName

int health
bool exhaust, unplayable
string[] playEffects, breakEffects

public functions:
void OnPlay(Vector2 center)
Triggered when the card is played, decodes all effects in playEffects and calls the CardEffectTrigger to apply it.

void OnBreak()
Similar to OnPlay, but is triggered when the card breaks after taking damage.

Card effects are encoded as **string**s in the following format:
"EffectName,power,range,other"
EffectName corresponds to the **CardEffect** scripts, e.g., "**pAttack**" triggers **CardEffect_pAttack**, and "**Draw**" triggers **CardEffect_Draw** script.

2.2.2 EffectInfo
Variables
string effectName
int power
Vector2 center
Array2D[] range
string other

The public function :
EffectInfo GetEffectInfo(string effect)
Decodes the string into **EffectInfo** struct.

2.2.3 Array2D
A 2D array that was meant to be serializable in unity inspector. It is often used as card effect range, which is a 5x5 array. It contains several presets that can be obtained by **Array2D[] GetRange(Presets rangeType)**

2.2.4 CardVessel
The object that contains a card's dynamic information. It manages the display of card information. It has two subclasses: **PlayableCard** and **RewardCard**.

PlayableCard
CardVessel for displaying cards in hand during battle. Manages display, card selection, and positioning.

RewardCard
For reward selection. Returns the card to add to deck in reward screen.

2.2.5 CardLibrary

Public variables
Dictionary<string, Card> cardDic : Dictionary of all **Card**s that should be in play, with card name as key.
enum Rarity {basic, common, rare, legend, special}

Public functions
Card GetRandomCard(Rarity rarity)
Returns a random **Card** of certain rarity type.

Card GetCard(string cardName)
Returns certain **Card** with name **cardName**.

2.2.6 CardEffectTrigger
A central unit of card effects. The effects are called through this script. Requires those **CardEffect** components in the same **GameObject**, which will be loaded onto the **GameObject** on start. The **CardEffect** scripts should be contained in **Resource**/**CardEffects**.

Public functions
OnPlay(EffectInfo info)
OnBreak(EffectInfo info)

2.2.7 CardEffect
Card effects manifest as **MonoBehaviours** attached to **CardEffectTrigger's GameObject**. This script is an abstract class for applying card effects. It often serves non card-specific effects, i.e. general effects such as dealing damage and applying status effect.

Public function:
void OnPlay(EffectInfo info)
void OnBreak(EffectInfo info)