Progetto di Valutazione per Midnite

Sandwich game

Scena di gioco

La scena GameScene(.unity), all'interno di Assets/Scenes, è la scena del gioco, con tutti gli oggetti coinvolti nel funzionamento.
Sotto *GameManagement*, in gerarchia, è possibile trovare tutti gli oggetti coinvolti nella gestione.

Da ***LevelManager*** è possibile settare l'asset contenente i dati degli ingredienti di gioco da utilizzare (come i models/prefabs), l'offset per il posizionamento degli stessi uno sopra l'altro durante il gioco, la dimensione della griglia di gioco (anche se fissata a 4x4 da specifiche progetto) ed il generatore di livello da utilizzare.

Sotto *LevelGenerators* sono contenuti i generatori dei livelli di gioco, da settare nel ***LevelManager*** per essere utilizzati. ***RandomLevelGenerator*** è quello che interessa ai fini del progetto e settato per la build finale; può essere configurato impostando il numero minimo e massimo di ingredienti aggiuntivi da utilizzare.

Gli assets di tipo ScriptableObject utilizzati sono contenuti all'interno della cartella *Assets/Data/* (*Assets/Data/Events* per gli eventi, *Assets/Data/Sandwich* per gli ingredients'data)

Organizzazione del progetto

Per la realizzazione del progetto ho strutturato l'intero codice nelle seguenti parti:

- Gestione degli eventi
- Gestione degli input
- Gestione del livello di gioco
- Logica di gioco

La *gestione degli event*i e la *gestione degli input* sono a sé stanti rispetto alla parte specifica del gioco, e possono essere riutilizzate per giochi di tipo diverso; questo ne favorisce la modularità.
La parte di *logica di gioco* è invece strettamente collegata al gioco di Sandwich da realizzare e contiene algoritmi ad hoc.

La parte di *gestione dei livelli* di gioco è altresì relativa alle meccaniche del gioco in questione, potrebbe tuttavia essere ulteriormente modularizzata per giochi quantomeno analoghi.

Questo approccio è stato pensato anche in considerazione al poter adattare il gioco di Sandwich a quello del **2048**.

Gestione degli eventi

La gestione eventi si basa sull'uso di ScriptableObjects e sul modello Emettitore-Ascoltatore: degli assets di questo tipo possono registrare degli ascoltatori e notificare su richiesta, da parte di script per i quali sono stati impostati, mentre questi stessi ascoltatori predispongono sulla propria interfaccia la possibilità di collegare un determinato evento ad un metodo specifico da richiamare.

Le classi coinvolte sono nello specifico:

- ***GameEvent***: di tipo *ScriptableObject*, per la generazione di un evento tramite metodo Raise()
- ***GameEventWithData<T>***: di tipo *ScriptableObject* e generico (estendibile in base al tipo di data), per la generazione di un evento con data tramite metodo Raise(T data)
- ***GameEventListener***: di tipo *MonoBehaviour*, per registrarsi su un evento (senza parametri) ed associare ad esso, tramite interfaccia, un metodo da invocare
- ***GameEventWithDataListener<T>***: di tipo *MonoBehaviour* e generico (estendibile in base al tipo di data), per registrarsi su un evento con passaggio parametri ed associare ad esso, tramite interfaccia, un metodo da invocare

GameEventWithData e ***GameEventWithDataListener*** sono al momento stati estesi rispettivamente con le classi ***GameEventVector2Data*** e ***GameEventListenerVector2Data***, per la gestione eventi con passaggio di un dato di tipo Vector2 (per il passaggio di evento di tocco sullo schermo con passaggio di posizione)

Per la creazione degli assetti degli eventi, su Vista Progetto, si può eseguire dal menù contestuale: "*Create → Game Events → ...*", a seconda del tipo di evento (semplice o con data)
Per il progetto sono stati creati in tutto i seguenti assets di eventi (cartella Assets/Data/Events):

- ***LevelStartedEvent***: lanciato a livello corrente inizializzato dal gestore dei livelli
- ***LevelWonEvent***: lanciato a livello completato dal gestore dei livelli
- ***ResetLevelEvent***: lanciato tramite pulsante Reset per il reset del livello
- ***StartNewLevelEvent***: lanciato tramite pulsante New Level per lanciare nuovo livello
- ***TouchEvent***: per l'evento di tocco su schermo, con passaggio di data (Vector2) relativa alla posizione di tocco
- ***SwipeEvent***: per l'evento di swipe su schermo, con passaggio di data (Vector2) relativa alla direzione di swipe

Gestione degli input

La gestione degli input serve per rilevare gli eventi di input che devono essere poi passati alla parte di gestione di gioco, per interpretare le mosse dell'utente; riguarda sia gli eventi di "tocco" e dei "gesti" sullo schermo ma anche delle interazione con gli elementi di UI. Si avvale della gestione degli eventi, mediante gli eventi precedentemente citati.

Le classi coinvolte sono:

- ***InputManager***: rileva gli input di tocco e genera i relativi eventi
- ***UIEventTriggers***: associa gli eventi degli elementi di UI (Buttons) ai Game Events; in questo modo gli elementi di UI possono servirsi del nuovo sistema di eventi

Gestione della logica di gioco (Sandwich game)

Dati di gioco

I dati dei componenti del gioco di Sandwich sono memorizzati tramite assets di tipo *ScriptableObject*: tramite lista vengono definiti tutti gli "ingredienti" utilizzati, e per ciascuno di essi deve venire definito:
Nome dell'ingrediente, che funge anche da identificativo
Assetto del modello (model o prefab) da istanziare a run-time
Eventuale fattore di scala, per ridimensionare dopo l'istanziazione
Nota a parte per l'ingrediente relativo al pane (deve essere sempre presente ed in coppia, a differenza degli altri), che va settato in ogni caso ed a parte.

La classe che definisce questo tipo di ScriptableObject è SandwichData, mentre per la creazione dell'asset è possibile utilizzare il menù contestuale "Create → Sandwich Game → SandwichData".

Componenti di logica

La realizzazione della logica di gioco è stata effettuata considerando la realizzazione di una griglia di gioco, sulla quale i vari ingredienti vengono inizializzati e "spostati", e tenendo conto dei seguenti principi:

- la griglia è sempre in forma NxN (configurabile anche se da specifica è sempre 4x4, dimensione comunque di default)
- la griglia è inizializzata con una matrice NxN di "sezioni"
- ciascuna sezione mantiene una memoria della pila di ingredienti finora inseriti
- ciascuna sezione si occupa dell'inizializzazione iniziale (eventuale) degli ingredienti, istanziando a run-time il GameObject, impostando posizione, scala ed aggiungendo i dovuti componenti

- gli ingredienti istanziati hanno un componente per mantenere informazioni come dati di gioco di appartenenza (nel file SandwichData) e sezione attuale, aggiunto a run-time se non presente
- tale componente rende il modello dell'ingrediente “cliccabile” (tramite Raycast)
- ciascuna sezione si occupa del trasferimento degli ingredienti verso un'altra sezione (non controlla se adiacente, compito della griglia), impilandoli a quelli già esistenti ed invertendo l'ordine, e settando i relativi valori
- ciascuna sezione controlla la situazione di “completezza”: vuota o con gli ingredienti impilati in modo tale che il pane sia il primo e l'ultimo ingrediente
- la griglia si occupa di gestire le “mosse” di gioco, cioè lo spostamento degli ingredienti (se presenti) da una sezione all'altra, impartendo i comandi alle sezioni
- la griglia può verificare in qualunque momento se le condizioni di “completezza” sono verificate per tutte le sezioni, quindi se ad ogni mossa il gioco può essere terminato

Le classi coinvolte sono:
- ***SandwichGameGrid***: griglia di gioco
- ***SandwichGameGridSection***: sezione della griglia di gioco
- ***SandwichIngredientIstance***: di tipo MonoBehaviour, componente aggiunto ai GameObject degli ingredienti

Gestione del livello

Le classi coinvolte sono le seguenti:
- ***SandwichGameLevelManager***: si occupa della gestione generale del livello, funge da coordinatore con gli altri componenti, crea e mantiene un'istanza dell'oggetto per l'oggetto della griglia di gioco che utilizza per effettuare le mosse impartite dall'utente e controllare di volta in volta le condizioni di vittoria. Mantiene anche l'informazione di livello iniziale, per reimpostare in caso di Reset. Effettua il lancio degli eventi di **GameStarted** e **GameCompleted**
- ***SandwichGameInputController***: si occupa della gestione degli input e dell'elaborazione delle mosse dell'utente da passare al gestore del livello; rimane in ascolto sugli eventi di **TouchEvent** e **SwipeEvent**
- ***SandwichGameLevelGenerator***: si occupa della generazione del livello di gioco, quindi della creazione della data relativa al livello; è una classe astratta, per essere estesa in base al tipo di generazione desiderata. La classe ***SandwichRandomLevelGenerator*** è stato creato ai fini del progetto per la generazione random di un livello, con il rispetto delle condizioni di gioco (gli ingredienti di tipo “pane” vanno sempre creati adiacenti)