Mobile card game dungeon crawler template

Contents

Overview

Thank you for your purchase of this mobile card dungeon crawler! A simple straightforward template to build games such as Look, your loot! And Dungeon Cards. Easy to extend items, enemies and heroes.

Requirements

To run the project, you'll need DOTween as well as TextMeshPro. Both free projects, simply install those to relieve any issues with.

How it works.

Entities

Everything that can be on a card is an "Entity" and has its own class with data about itself, such as the Hero entity having what weapon is equipped, health, damage etc…

Every entity also has a "Card" object on them, which represents what card they are on.

These card objects hold a ton of logic for interacting with each card on the map, but generally when we interact with an entity, we go through their parent entity (such as hero, item, weapon etc…)

Gameplay

The core game is driven from the "MapManager.cs" class, when created, this class will invoke the MapInitializer.cs to create a map of 9 cards with entities decided by the MapInitializer.

Movement

When we move, we recursively call the MovementEngine.cs method MoveAndFill, what this will do is move us to the spot we desire, and then continuously backfill the space we left open, until we get to a point where we can't backfill anymore. When that's the case, we simply create a new card.

Afterwards we wait for input, if the input is on a card (checked with a Raycast against a GameObject with a tag of "Card"), then we figure out what was on that card and interact with it (i.e. consume item, grab weapon, fight enemy).

You can move by either clicking on a card near your hero, or using the arrow keys to navigate. The arrow key mappings can be changed in the inspector, on the "Map" object.

Tiers

Tiers is used to generate both harder enemies and rarer items, this is calculated through the TierEngine.cs and simply uses the current Coins to determine what tier we are on.

Spawning

In our MapInitializerEngine.cs we use a method called "FillCard", this will do a random roll to decide what entity it should create, either a random item, weapon or enemy.

Value deviance

A common concept you'll see is Value Deviance, this is simply a way of us altering values to give the game more randomness. For example, weapons have a value deviance which we apply when we create them to give variety to damage.

What is?

Hero

The character the player uses, built from the following data

- Health – How much health the hero starts with
- Name – The name of the hero (what is displayed)
- Sprite – What sprite we use to show the character
- Weapon – Starting Weapon
- Description – Shown in the hero select
- Precedence – What order to show the hero in (UNIQUE)
- Upgrade Costs – How much each upgrade should cost, if the hero is upgraded more times that costs you've created, it'll use the last cost as the 'perpetual upgrade' amount
- Weapon & Item Filter – Use these to only allow items and enemies within this array to be spawned. For example, add "Sword" to the weapon filter and only swords will spawn.

Enemies

- Health – Starting health of enemy
- Health Deviance – How much the health can increase by, first number is floor, second number is ceiling
- Name – Name of the enemy which is shown on the card
- Tier – What tier should this enemy appear on?
- Sprite – What sprite we use to show the enemy

Weapon

- Name – Name of the weapon which is shown on the card
- Value – The damage of the weapon
- Effect – Special effect the weapon uses
- Value Deviance – The floor/ceiling to affect the weapons damage. A random number between the floor and ceiling will be added to the damage, you can also use negative numbers to lower the damage.
- Rarity – How often the weapon should spawn, less rare weapons are spawned more often (1 = most common)
- Sprite – What sprite we use to show the enemy
- Sound Prefix – The prefix to use when playing audio, used to play random sounds from a variety of files (i.e. prefix = sword_hit, files are sword_hit_1, sword_hit_2, sword_hit_3)

Items

- Name – Name of the item which is shown on the card, can also be looked up to generate the item

- Value – Value of the item, this is used based on the effect (coin = add coins, heal = add health)
- Effect – What effect to apply when the item is picked up
- Rarity – How often the item should spawn, less rare items are spawned more often
- Sprite – What sprite we use to show the item

User Data

- File we store to hold the users progress
- Current Hero – The hero the user has selected
- Coins – Total amount of coins
- Coins Last Round -How much the user got last round
- Hero Upgrades
    - An array of name/upgrade level of each hero in the game. Defaults to upgrade level 0 if no upgrades.

Code overview

Terminology

Engines

- Used to evaluate and act on input, i.e.:
    - Combat
    - Creating a map
    - Moving

Managers

- Stateful and used to manage states of entities
    - CoinManager – Used to modify the amount of coins a user has, as well as coins last round
    - HeroSelectManager – Used in the hero select to change the current hero, the hero the user is viewing, as well as handling upgrades.
    - MapManager – Runs a lot of the core game, controls input recognition, as well as moving the character, interacting with entities and managing the game state.
    - MenuManager – Holds the logic for the menus. Also, the rewarded video ad logic.
    - AudioManager – Holds logic for playing Audio, preloads all audio files and exposes a method to play a random file given a prefix.

Repository

- Generally used to read data from JSON files
- Can also be used as an interface to store and retrieve data such as active cards

Entities

- Can exist on a map such as
    - Enemies
    - The Hero
    - Items
    - Weapons

Dependencies:

- DOTween – Easy free library for tweening and creating animations.
- TextMesh Pro – inbuilt library to display rich and beautiful text.

How to?

Add an enemy

Open enemies.json, copy an existing enemy and configure. Tier is incremented via the amount of coins a player has in the current level.

Add a weapon

Open weapons.json, copy an existing weapon and configure. Rarer items will appear more in later tiers. You can also apply an effect to the weapon when it's used to attack. If you'd like to add a new effect, open CombatEngine.cs and under "BattleEntities.cs" you can add a new effect handler.

Add a hero

Open heroes.json, copy an existing hero and configure. Precedence is unique so don't have multiple heroes of the same precedence.

Add an item

Open items.json, copy an existing item and configure. Effect is the primary driver of how this item will act. To add a new effect, open CombatEngine.cs and under the "ConsumeItem" method, add whatever logic you'd like your new item to do.

Change spawn rates

Check MapInitializerEngine.cs and down the bottom you'll see the GenerateCard method, here we generate a random int and do a IsBetween check to spawn entities. This is the easiest place to modify this behaviour.

Add a sound

Simply add a sound to Resources/Audio and use the AudioManager to play the sound where you want to in the code.

Support

For any support issues, feedback or feature requests. feel free to contact UnityBricksHelp@gmail.com and we'll get back to you as soon as possible.

Credit

Art Assets:

- https://0x72.itch.io/16x16-dungeon-tileset
- https://0x72.itch.io/dungeontileset-ii
- Heart Pixel Art - DontMind8.blogspot.com

Audio

https://opengameart.org/content/punches-hits-swords-and-squishes

https://opengameart.org/content/rpg-sound-pack

Inspiration – Dungeon Cards.
https://play.google.com/store/apps/details?id=com.The717pixels.DungeonCards&hl=en_AU