Dungeon Point

by Miquel Gironés

The main objective of this project is to create a small prototype but with a scalable architecture which lets the project grow into a full game easily.

The code architecture is independent and separated by managers. The point is that every manager has a main task or purpose independent from other tasks.

This projects has 4 managers:

- GameManager: Manages the game flow and the player logic
- MenuManager: Manages the main menu scene
- GridManager: Manages the creation of the grid and the level setup (enemies, items, doors…)
- InputManager: Manages the input logic
- AudioManager: Manages the sound system of the game.

Every object of the map is referred as an Element. There are different types of Elements at the moment. Now we divide All Elements in two groups:

- Enemy: This class inherits from Element and handles all the logic of an enemy.
- Item: This class inherits from Element and it is an abstract class which every type of item will override. Currently there are 3 types of item:
    - HealthItem: Gives the player health when picked
    - AttackItem: Gives the player attack when picked
    - ExitPointItem: This is the item needed to pass to the next dungeon (exit).

To calculate the steps needed to find the exit a simple A* Pathfinder class is used.

The camera has a small script to let it move using WASD or the arrows.

Main problems

I did not had much trouble while developing this prototype to be honest, but the part that was more difficult from my experience is the pathfinder because I had to investigate again the A* algorithm and know well what i was doing with the nodes. So this is the part which I invested the most time on.

Another part that caused me a bit of a trouble was to manage sprites an UI at the same time because I have had some problems with render overlapping between these two types.

Points of improvement

One thing I would improve is the enemy management. At the moment there is only one possible enemy, so I would create a system to add different types of enemies more efficient and visual. Maybe use more the ScriptableObject system to create enemy assets and make it simpler for a designer to tweak and create different types of enemies using only the Unity editor.

One thing I would redo if I could is to redo the visual part and only use UI and Canvases for the visual aspect instead of combining sprites and UI. The reason is that it can produce conflicts in terms of drawing order and also because the UI system is better prepared for different platforms and aspect ratios and also is more scalable and optimisable.