Guide to the Scripts - Implementation 2
Team SantaHatesPoorKids

Assets

Overworld->Map->Scripts

- CameraScroll -> Camera script for scrolling around in the overworld with WASD keys
- MapGenerator->Handles all functions related to procedurally generating the map at the start of the game
- MapProperties->Script that tracks the state of the overworld map such as node tracking and offers functions such as map loading and gameobject destruction when the current map ceases to be useful (i.e. on defeat and game restart)
- NodeEvent -> Not Used so far, fields are currently under NodeProperties for now
- NodeHoveScript-> Changes the color of the node on hover
- NodeProperties -> Tracks Variables for each node, also provides functionalites specific to the node such as event processing, node connections, path drawing to neighbours, and other relevant functions
- NodePartySelect -> Script that is responsible for grabbing the different components of the party node to be used for position/ property updates
- OverlayUIScript-> Scripts that deal with Canvas functionality (win/loose screen buttons, UI Status Updates)
- PartyProperties-> Tracks the variables of the party (party stats, party occupied node, party resources) as well as providing resource handling functionalities
- PartyWalk-> deals with the movement/animation aspect of the party cursor moving in the overworld, mainly lerping calculations for the “walk” animation
- TileProperties -> tiles are mainly a hidden component in the overworld. It forms the background and uses its inner radius to random generate the relative location of the nodes.

Overworld->Dialogue Scripts

- Dialogue-> serializable object used to load dialogues from JSON
- DialogueControl->Provides control for the corresponding UI elements to actually load the dialogue when needed
- DialogueSet -> each “dialogue” consists of multiple dialogue panels, which is represented by the Dialogue object. DialogueSet groups the Dialogue objects into a full “dialogue” (a conversation between characters)

Combat->Scripts

- Enemy -> serialized enemy unit class
- EnemyLayout -> serialized Layout file for enemy units
- GenerateBattlemap -> Procedurally general a combat encounter based on passed in information
- JsonHelper -> helper class for parsing json
- Maplist -> serialized map class (holding map information)
- ObstacleLayout -> serialized Obstacle Layout class
- ObstacleList -> serialized Obstacle class
- TileList -> Serialized Tile class

TBS Framework->Core->Scripts

Note that much more of the scripts in Core is being used, but the following are the only ones we modified from the Original Modified parts are commented

- CellGrid-> Class for handling the strategy gamestate
- CellGridStateUnitSelected -> actions done in CellGrid with a selected Unit
- Unit-> Class that holds Unit and all that a unit can do

UI->Scripts

Disclaimer! This section has had some issues merging with the main project, if any of these scripts are missing please contact the team.

- DisplayItemInfo-> Responsible for displaying the Item info to the Info Panel. It retrieves the Item's image, it's name and description. TODO includes updating the player stats.
- EquipControl-> This handles the equipping logic of a selected item, deciding whether or not the item can be equipped. TODO includes more condition checking for each equipment category
- ItemControl-> This script acts as the logic as to what happens to an item when it is remove, added, hidden or created in the party pool. The user can test these functions by hitting the a, s, d, f buttons.
- ItemInfo-> This script is responsible for holding all of the stat changes, item type and item description about a specific item.
- ItemInit-> This item assigns the button logic for the item, telling it what to do when it has been selected.
- ItemLayout-> This script is responsible for organizing the number of items and laying them out in the party pool based on different dimensions, play with max screen size for best performance. CURRENTLY UNUSED
- LevelInfo-> Script that randomly decides what the level of the item will be when it is first created. CURRENTLY UNUSED
- MainMenu -> Responsible for instantiating a new game when called.

- MenuOpenClose -> The small script is a hotfix for opening and closing the Item menu. Press m to open and close it. TODO includes adding a GUI button that the player can click on instead.
- MouseHover-> This script acts as a trigger to assign the item to as the displayed item in the Info panel when the item is selected. The trigger allows it to not run in updated time and eventually will trigger when the mouse is on enter of the button.
- NextImage-> mapping script that maps different types of sprites to acronyms.
- Selection-> Script that will eventually allow the user to drag the button to a desired equipment spot, however this may be scrapped as it is not implemented

UI->Scripts->SteralizableClasses

These scripts will eventually be reorganized into the project, but for now are kept here

- BattleState-> Class that creates a new instance of the HeroStates for each hero. Used at the beginning of the game initialization.
- HeroState-> Class that has a list of each Hero's stat and the respective accessors and setters to change the values.
- SaveLoad-> Script that is responsible for saving and loading the Hero's stats so that the information can be translated between scenes and save games.

UI->Scripts->InventoryFiles

These scripts are the basis of the inventory system in a Darkest Dungeon Unity Port and all the scripts in this file are renamed copies or scripts that implement them. Kept separate for clarity. The project can be found at https://github.com/Reinisch/Darkest-Dungeon-Unity and the rename is as follows
DarkestDungeonManager -> InventoryManager
DarkestDatabase -> Database
Trinket -> PassiveItem
EstateSceneManager ->InventorySceneManager
RealmInventory -> MenuInventory

- ApplyEquipment-> script that is applies the equipment to the combat scene.
- BaseSlot-> A base rect transform that can be used to describe the position of the slot.
- ConsumableItem-> The consumable items from the iteminfo, it is a list of items that has a set number of items.
- Database-> this script is responsible for instantiating the the database
- EquipedInventory-> This class is responsible for instantiating the equipped inventory for the player's campaign. right now the inventory is hard coded
- InventoryItem-> Script for creating a new inventory item
- InventoryMananger-> This script is responsible for reading the database and loading it to an instance.
- InventoryRow-> This script is responsible for creating inventory rows for the inventory UI.

- InventorySceneManager->This script is reponsible for initalizing the entire inventory scene, database and UI.
- InventorySlot-> This class states all the UI elements of a Inventory Slot and it's item information. Button items will be added
- InventorySlotData-> This class instantiates the data that needed for the inventoryslots without needing to refer to the item itself
- ItemDefinition-> This class defines information about all kinds of times, will be replaced soon by placing into InfoItems
- MenuInventory-> This class acts as a instance for the list of current items in player's inventory.
- MenuInventoryWindow-> This script creates a new inventory window that is the basis for all of the UI elements in the Inventory Window.
- PassiveItem-> This class is an extension of the ItemInfo which represents all the items that apply a passive value change to the character.
- SaveCampaignData-> This class represents the saved item data for the specific save file, not implemented 100%
- SimpleJSON-> This is a simpleJSON reader found online that comes with it’s methods and Json functionality. Found at https://simplejson.readthedocs.io/en/latest/