How to Use the RPG Data System

Intro

The RPG Data System is a system that aids in the making of RPG games using the Unity engine. Specifically, it's responsible for managing data that is needed in such games. Such data can include stats, experience points (XP), level progression, and more.

The RPG Data System ("RPG System" for short) has 3 main subsystems: the Data Assets, the RPG Character (which represents the data of a player in the game), and the Serializer for the RPG Character. These will be discussed in more detail below.

The RPG System currently has 2 prerequisites which are included in the project. The first is the Data Asset System located in </Assets/__Scripts/ReadOnlyDataAssetSystem > and the Save Slot System located in </Assets/__Script s/SaveSlotSystem >. The RPG System code is located in </Assets/__Scripts/RpgDataSystem >. The Data Asset System is required to make new data asset files based on data classes that inherit from ScriptableObject. The Save Slot System is actually optional, but I needed it to demonstrate saving RPG Characters to file.

There are 4 test scenes located in </Assets/__Scripts/RpgDataSystem/_Test_Scenes >. The first is `RpgDataTest_XP`, which it only testing the RPG Character's experience points (XP) and health points (HP). The second is `RpgDataTest_Stats`, which includes previous tests and adds functionality to add Stats of various kinds and the ability to add Stat Points (SP) to different stats. The third scene is `RpgDataTest_Abilities`, which includes previous tests and adds functionality to add Abilities to the RPG Character. The fourth and last scene is `RpgDataTest_Serialization`, and includes previous scenes, but includes the functionality to save RPG Characters to file using 3 save slots.

RPG Data Type Assets & RPG Data Registry

The RPG Data Type Assets (“RPG Types” for short) are data classes that are saved on file and contain read-only information that only developers can set (not players or gameplay code). The developer can create these files from the Unity Editor using the menu <Assets → Create → Spherical Cow → RPG Data System>. The data asset files are located in <`/Assets/_DataAssets/RpgSystem` >. There are 3 kinds of RPG Types:

- XP Progressors (formally known as “Progression Variables”)
- Stats
- Abilities

XP Progressors

XP Progressors are used to calculate the progression of XP goals. By “XP goals”, I mean the amount of XP needed to “level up”. Every RPG Character has XP and this concept of the “level”. Everyone starts out at Level 1, but once they get more XP, they can level up to higher levels. So XP Progressors basically contain variables to indicate how much XP does the player need in order to level up. All XP Progressors utilize the same “leveling up” formula but can have different parameters. The level up formula is:

$$N_X = M_L \times L + M_O \times O_X$$

Where $\mathbf{N_x}$ is the new amount of XP to level up, $\mathbf{O_x}$ is the old amount of XP to level up, **L** is the current level of the player, $\mathbf{M_L}$ is a multiplier for the level value, and $\mathbf{M_o}$ is a multiplier for the old XP goal amount.

The multipliers can also increment by 1 every time the player levels up. This can be set when defining the XP Progressor in the Unity Inspector.

Stats

Stats are basically “properties” that an RPG Character has. They can be abstract like “strength” or “speed” or more specific like “battle axe” or “machine gun”. All Stats have Stat Points (SP), which are values that indicate how advanced a certain stat is with the player.

There are 3 types of Stats: Base, Secondary, and Skill. Base Stats are the most abstract stats and are the basis for other stats (i.e. they contribute to the initial amount of SP or other stats). Secondary Stats are semi-abstract stats that can derive SP from Base Stats. Skill Stats are specific stats that can derive SP from any kind of stat, including other Skill Stats.

Secondary and Skill Stats have the concept of “linked stats”, which are stats from where the current stat derive SP. A percentage can be defined to indicate how much SP should be derived from the linked stat.

Abilities

Abilities can be applied onto RPG Characters in order to modify multiple applied stats. This can be used for various effects, like downgrading weapons from a curse, or upgrading strength from a pledge, for example.

Abilities are actually a collection of modification units called “Ability Modifiers” (or “Modifiers” for short). Abilities can have any number of Modifiers that the developers desire. A Modifier is composed of a stat to modify, a modification type, and a modification value. There are 4 types of modifications:

- Increase By - increases a stat’s SP by the given value
- Decrease By - decrease a stat’s SP by the given value
- Increase To - jump a stat’s SP to be at least the given value
- Decrease To - drop a stat’s SP to be at most the given value

The RPG Data Registry

The RPG Data Registry (or “Registry” for short) is a singleton class that contains all the XP Progressors, Stats, and Abilities accessible from code. If developers make new RPG Type files, it will be automatically added to the Registry, and if an RPG Type file is deleted, it will be automatically removed from the Registry.

The RPG Character Class

The RPG Character Data class (or “Character” for short) is the data representation of an individual character in an RPG game. This will be usually be used for the player character, but it may be possible to use this with non-playable characters as well. The Character class is not a script (MonoBehaviour), so scripts can be made around Characters if desired.

The Character contains data about XP, HP, XP Progression, applied Stats, and applied Abilities. When making a new Character, you will need to provide: a GUID (for unique identification), an XP Progressor (search it from the Registry), HP and maximum HP, and an SP Assignment flag, and optionally, a name (highly recommended).

SP Assignment flags indicate to the RPG System how new SP will be assigned to the Character when it levels up. There are currently 2 methods: Point Assignment (PA) and Use Assignment (UA).

With PA, when a Character levels up, they are rewarded SP that is put into an unallocated pool, and the player must chose in which applied Stats to allocate SP. PA is not recommended for non-playable characters, unless you want to automate SP assignment onto stats yourself.

With UA, when a Character levels up, it is rewarded SP and that SP is automatically allocated into all applied Stats. The amount of rewarded SP is automatically calculated based on how often an applied Stat is “used”. In the gameplay code, the developer must mark a stat as used [see `MarkedStatAsUsed()` in code] whenever a Stat is used (call as often as used). This accumulates a “Use Factor” in the applied stat, which is used to calculate the rewarded SP. The higher the Use Factor, the more SP is rewarded.

When making a new Character, it will start out with no applied Stats or applied Abilities. It will have 0 XP and start at Level 1. The gameplay code is responsible for adding Stats and Abilities into the Character. Also, when Stats are used, the gameplay code should mark the stat as used. When enemies are defeated, the gameplay code should add XP into the Character. If using Point Assignment, the gameplay code has to allow players to allocate SP into whatever stats that the players desire.

Serializing RPG Character into an XML File

When making an RPG game, it becomes necessary to save the progress of the player's Character in order for the player to continue the game later. This requires writing data about the Character to a file on disk. The RPG Character Serializer (or "Serializer" for short) is a utility class that does exactly that.

Each RPG Character is saved into its own XML file inside Unity's Persistent Data Path. The filename is based off the name of the Character and its GUID. This is done for unique identification (in case there are Characters with the same name).

Saving a Character onto disk only requires that instance of the Character to be passed into the Serializer for saving. To load a Character from disk, the Serializer needs to be passed the name of the Character to load and its GUID (as a string). In order for this to work, the developer should somehow save a Character's name and GUID in a separate system, such as Unity's PlayerPrefs, or use the Save Slot System that I provided in the project.

Known Limitations

- Hybrid SP Assignment was not implemented. This means that the SP Assignment method (Point or Use) cannot be chosen per Stat. In the current system, it can only be chosen per Character.
- Skill Stats don't add SP to their linked stats (something that appeared to be specified in the design doc). In the current system, Skill Stats get SP contributions from their linked stats.
- The RPG Character currently doesn't have the capability to switch XP Progressors (although this could be implemented without too much trouble).
- There is no automated way to add SP to newly added stats ***after*** the Character leveled up many times. This is up to the gameplay developer.
- The RPG Character currently doesn't have the capability to switch SP Assignment methods after creation.
- The RPG Character currently doesn't have the capability to change its name after creation (this would complicate serialization code because it depends on the Character's name).
- The RPG System is implemented on the assumption of being Set-Point-Progression based. Failure-Based-Progression was not implemented and not considered when designing the system.