Save System Documentation

Setup:

* Drop the SaveSystem prefab into your scene.
  + It should already have a SavingAndLoading script on it.
  + Set the GameName to whatever you would like your save file named as.
* As you build your scene, add a SaveableEntity script to any object that you want saved.
* There is a BaseSaveData class that saves the basics that each saveable entity should have:
  + Prefab (or object) name
  + Transform position
  + Transform rotation
  + Transform scale
* You can create classes off of BaseSaveData for each of the different types of saveable entities you want to have i.e, enemies, vehicles, etc.
* Attach a BaseSaveData or inherited class to each object of the corresponding type that you want to save data on.
* Once your scene is complete and all saveable entities are setup, select your SaveSystem object in the hierarchy and press the Generate GUIDs button. This will generate guids for any of the current objects in the scene.
* If you have prefabs that are spawned during gameplay and you want their information to be saved, be sure to set them up with a Save Data and SaveableEntity script on the prefab, and then assign a reference to them on the LoadablePrefabs scriptable object.
* Now that your scene and objects are setup, you can tie in your code to call the Save() and Load() functions from SavingAndLoading. Keep in mind that this is a Singleton so you don’t need a reference to it. You will just call SavingAndLoading.Instance.Save() (or .Load()).

Notes:

* This save system uses json with AES encryption when NOT in the editor. In the editor it will not encrypt, making it easy to view save data for debugging and development purposes.