**TextColor – не трогать**

**TextColor – ожидает создания**

**TextColor – под вопросом**

**DataManagerComponent –** управление данными. Сохранение, загрузка, удаление.

**Методы:**

**public:**

**Category [Default]:**

* **InitGameData** - инициализация определенных игровых данных, указываем какое сохранение и его тип. Инициализированные игровые данные добавляются в словарь GameDataObjects. Так же можно возвращенный результат сохранить в переменную, но перед этим желательно сделать приведение типов (cast).
* **SaveGameData** - сохранение игровых данных, в зависимости от типа
* **LoadGameData** - загрузка игровых данных, в зависимости от типа
* **DeleteGameData** – удаление игровых данных, в зависимости от типа
* **GetLoadedGameData** – получить загруженные игровые данные, в зависимости от типа, из словаря **GameDataObjects.** (НУЖНО ПРИВЕДЕНИЕ ТИПА (Cast) К НУЖНОМУ ФАЙЛУ ИГРОВЫХ ДАННЫХ)

**Переменные:**

**Category [Data]:**

**private:**

* **GameDataObjects –** словарь данных, в который сохраняются данные после создания (если нет сохраненных данных), загрузки (если есть сохраненные данные). Эти же данные используем для изменения, пересохранения, удаления.

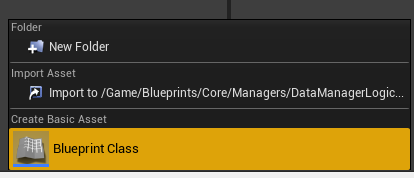
**Core/Managers/DataManagerLogic/DataHierarchy** (иерархия данных)**:**

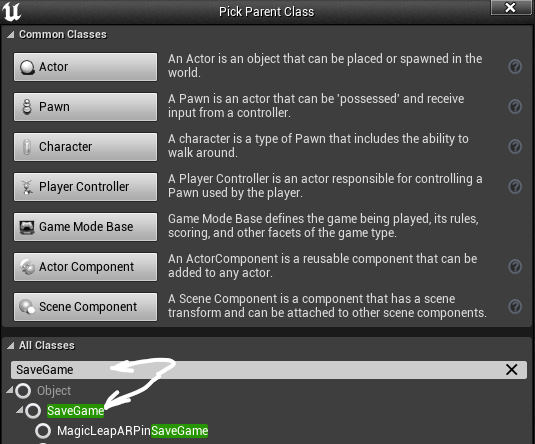
* **Gameplay –** сохранения игровых данных.
  + **Levels** – сохранения уровней
  + **Tasks** – сохранения тасок
* **GameSettings –** данные настроек игры. Например: настройки аудио, графики, управления.

**DataClassStructure –** структура, которая используется в массиве GameDataClasses, для удобного назначения данных для инициализации.

**DataType** **–** перечисление данных которые сохраняем, восстанавливаем, удаляем. При сохранение, или загрузки данных нужно имя по которому сохраняем или загружаем эти данные. Вот это имя хранится в перечислении **DataType.**

**Процесс создания файла данных**



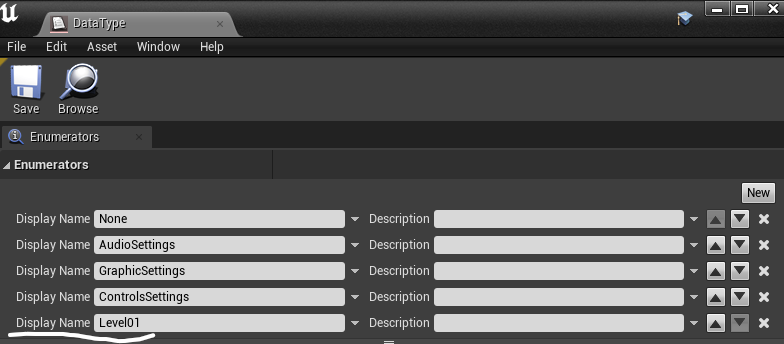


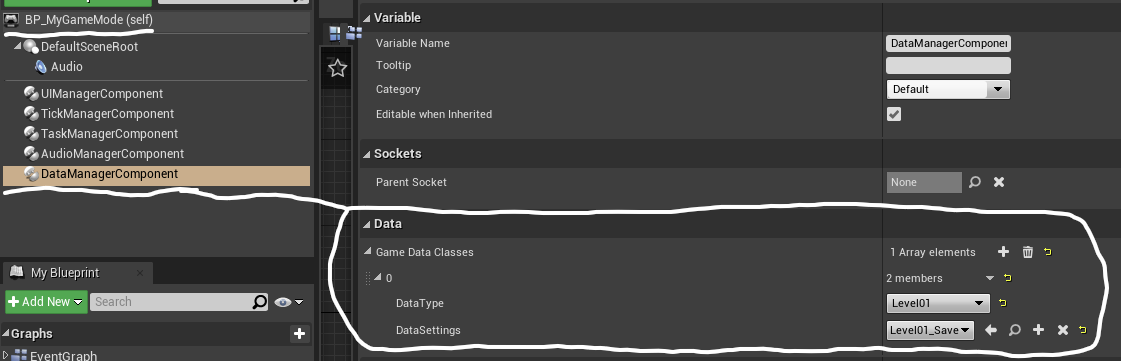
Создаем объект типа **SaveGame**. Это и будет файл для данных. Называем в соответствии с тем, какие данные там будут храниться.

Стиль именования классов данных для игрового сохранения: **НазваниеКласса\_Save.**

Например, если нам нужны данные для сохранения на первом уровне, создаем файл с названием **Level-01\_Save**.

Добавляем в перечисление **Level-01**, и добавляем этот файл в **DataManagerComponent**, в **BP\_MyGameMode**.





Процесс сохранения данных:

1. Создаем структуру, в которой будет храниться вся информация о конкретном игровом объекте, который спавнится на сцене и нужно его вносить в файл сохранения.
2. Эту структуру добавляем в файл сохранения (если одиночный объект. Например игрок (одиночная игра)) или массив этих структур (если объектов на сцене куча. Например враги или монеты)
3. Вносим изменения в определенные данные
4. Сохраняем файл данных

**Save Data Files**

Файлы, которые нужны для сохранения данных и какие данные нужно в них хранить (на примере прототипа).

GeneralTaskCounter

CurrentTaskID (-1 нету текущей таски)