UPDC
Unity to PokEngine Data Converter
Technical Document

Oleg Loshkin

December 20, 2019

1 Introduction

- **Project's Context**
  As part of the game programming cursus at SAE Institute Geneva, for the **technical module GPR5100.2**, students of the second year must **assist students from the third year in completing their bachelor's project.**

  This year, the third year's **PokFamily team** develops a **video game for the Switch** and PC using a tailored **in-house engine**.
  Second year students must **assist them** by creating various **tools they will need** in order to create their game.

  This document describes the functioning of the **Unity to PokEngine Data Converter** tool, **UPDC** for short.

- **Project's Goals**
  - Create a useful tool that the PokFamily team will use to create their video game.
  - Learn to work in a non-academic environment in a team that depends on the student's performance.

- **Specific Problem**
  The PokFamily team uses the **Unity engine as an external editor**. The PokFamily team needs a tool to **convert ScriptableObjects to a JSON** format **readable by the PokEngine** parser. This tool may then be used by other Unity tools to export data.

2 Requirements

This project's requirements have two origins:

- **Academic requirements**
  - The task given by the team has been understood and done in time.
  - The tool is maintained by the student after the tool's completion.
  - The tool must be user-friendly.
  - The student understands how to manage data.
  - The student understands how a game engine interfaces with a game engine editor.
  - The student has organized himself and his work in a way to facilitate the work of others.
  - The tool's performance is reasonable.
  - The implementation is appropriately sophisticated.
  - The student understands the implications of non-academic teamwork.
- **Pragmatic requirements**
  - Convert ScriptableObject files to files readable by the PokEngine's parser.
  - The user must be able to interact with the tool via Unity.
  - The code must satisfy the quality and style expected by the team. C++ coding style is defined in the Coding Style Document. C# coding style is defined in UnityWorkOrganization document.
  - The student must communicate with the team appropriately and be dependable.
  - Allow other tools to export data via the UPDC.

3 Technologies Used

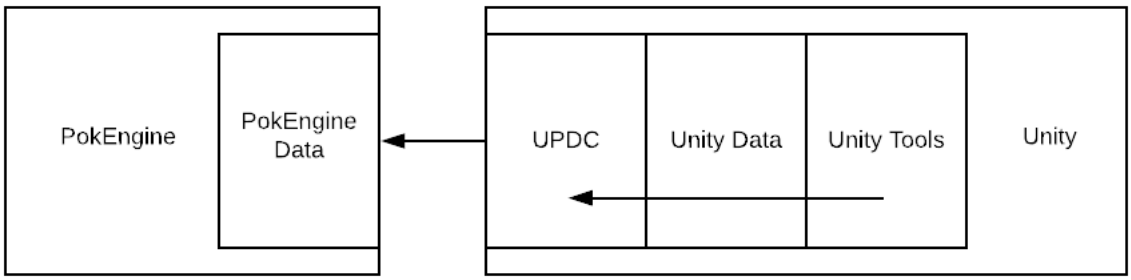
- **PokEngine**
  The **PokEngine** is the game engine developed by the PokFamily team. The engine is **written with C++ standard 2014** and **partly C++ standard 2017** for code running on the Nintendo Switch.
  The engine has a parser that is capable of reading JSON files. This parser is used to import assets exported from Unity with UPDC.

- **Unity 2019.1.10f**
  Unity 2019.1.10f is used as an **external editor**.

- **Visual Studio 2017**
  Visual Studio 2017 is used for development of the PokEngine.

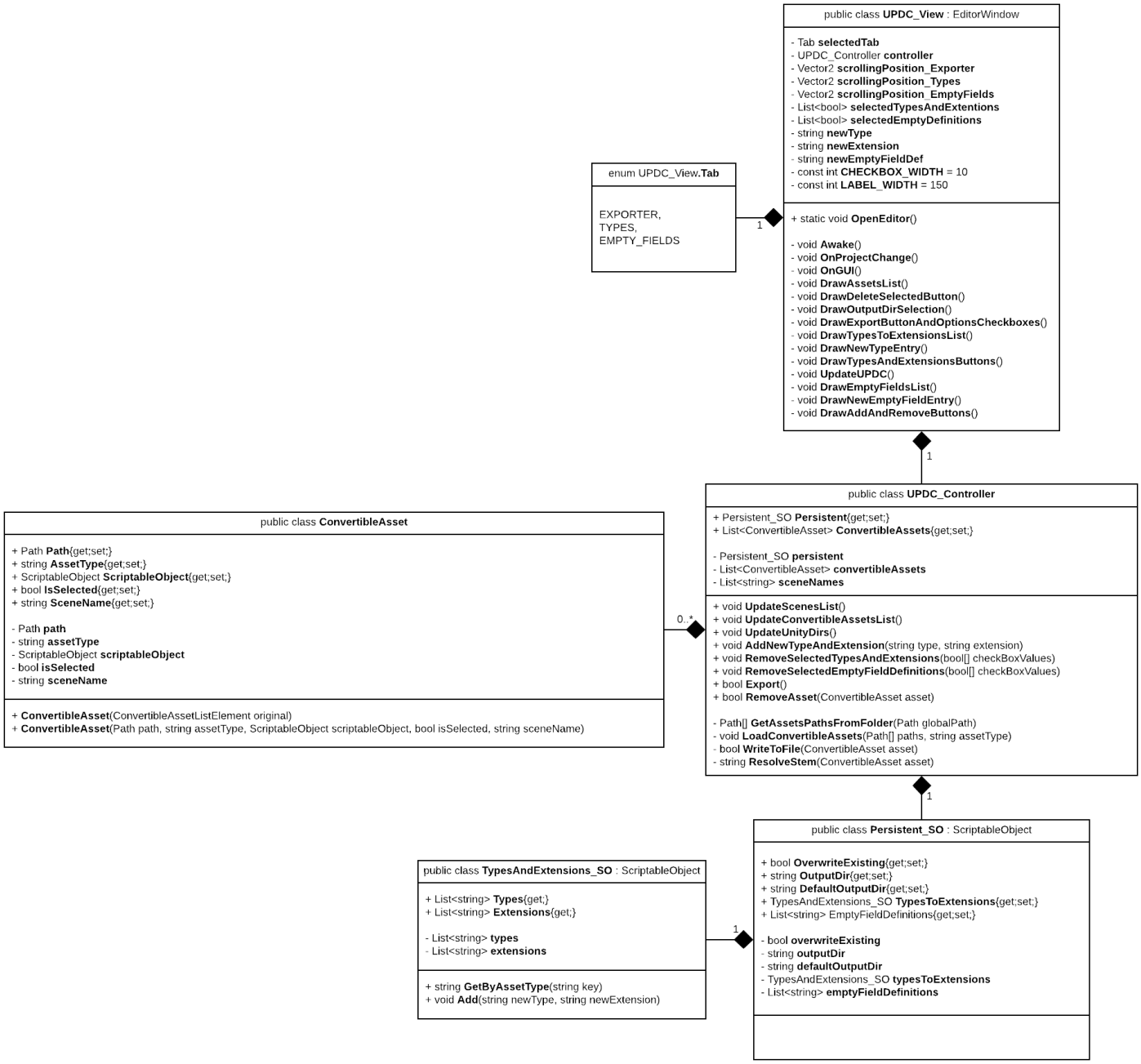
- **Git**
  github.com is used for versioning for the **PokEngine source code**. **gitlab.com** is used for versioning for the **Unity prototype source code**. Git bash is used for most interactions with the git framework. Merge conflicts are solved manually via text editor and git bash.

4 Interaction with Overall Project



UPDC allows other Unity tools to export data to be used by the PokEngine. As such, it is the last chain link that's still part of Unity before said data leaves Unity altogether to integrate the PokEngine's side of the project.

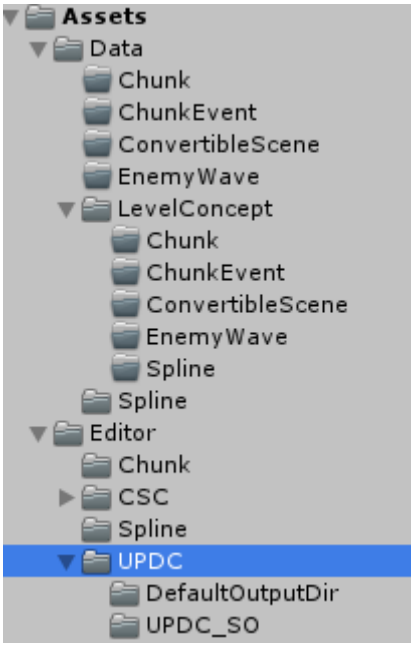
5 UML Diagram



The tool follows a **Model-View-Controller** model where the model is Unity's data, the view is `UPDC_View` and the controller is `UPDC_Controller`.
ScriptableObjects are used to store user data across sessions. Said data includes:

- Whether or not to overwrite any existing files upon export.
- The output directory for export.
- Default output directory for export.
- A dictionary like class to keep track of defined types and extensions managed by the UPDC.
- A list of substrings to remove from JSON files before writing them to disk.

6 Folder Hierarchy



The "Assets/Data" folder is used for storing convertible .asset files. Those stored in folders with the same name as the type of the convertible asset directly **under "Assets/Data/someType" are .asset files not linked to any particular scene. Those stored in a folder named after a scene are the ones linked to that particular scene**.
In the above example, "LevelConcept" scene may have Spline .asset files linked to it stored under "Assets/Data/LevelConcept/Spline" for instance.

The tool's own scripts are located under "Assets/Editor/UPDC". Additionally, a subfolder exists there for storing ScriptableObjects that hold data across sessions under the texttt"UPDC_SO" folder and a "DefaultOutputDir" exists to hold any converted .asset files if no specific output path has been defined before export.

7 Tackling Genericity

UPDC needs to be able to export various data types that **aren't well defined in advance**. With project advancement, new types of data may need to be exported besides the ones currently developed by the team. This means that **the implementation of the tool had to be flexible and not hard-coded**.

The **tool addresses this constraint by making use of Unity's ScriptableObject class**. ScriptableObjects are Unity's way of handling persistent data and the engine has well developed functionalities for manipulating ScriptableObjects. JSONUtility was used for generating JSON files from ScriptableObject objects for instance.

By using a string to string dictionary like class (`TypesAndExtensions_SO`), UPDC allows to **add and remove support for any convertible asset types without requiring the user to interact with UPDC's code** provided the type defined inherits from the ScriptableObject class.

8 Working with JSONUtility

Unity's JSONUtility helper class was used to convert ScriptableObjects into JSON formatted strings that were then written to disk. Using it meant that **no custom parser needed to be implemented for this tool, however there were drawbacks in using it later down the line.**

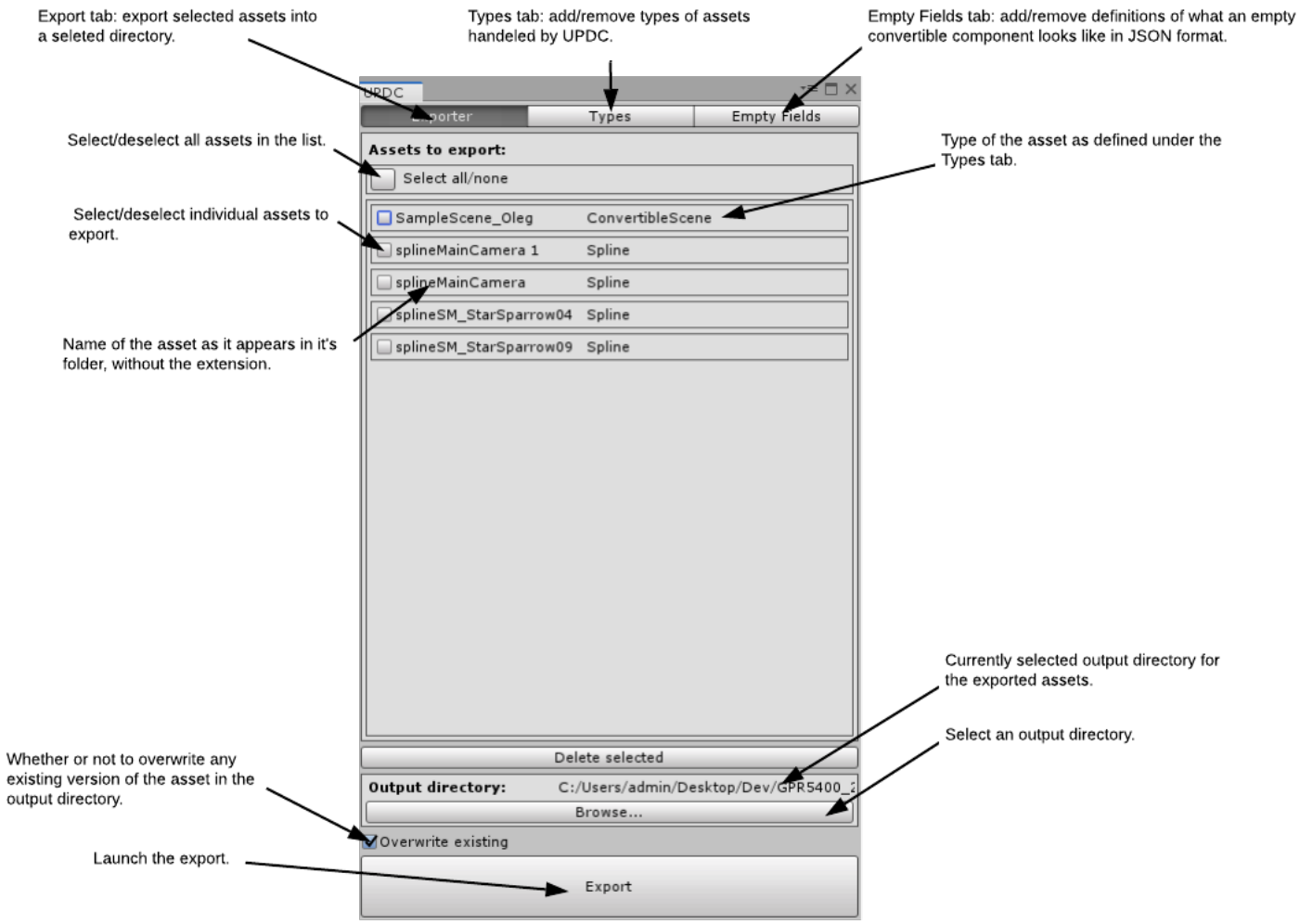
Unity's serialization system considers every object like a struct type, meaning that any serialized data structure, be it a class or a struct, is **written down in full in the output JSON string**, including default values for any uninitialized fields of the object being serialized. This leads to the **JSON string getting bloated with unnecessary data**.

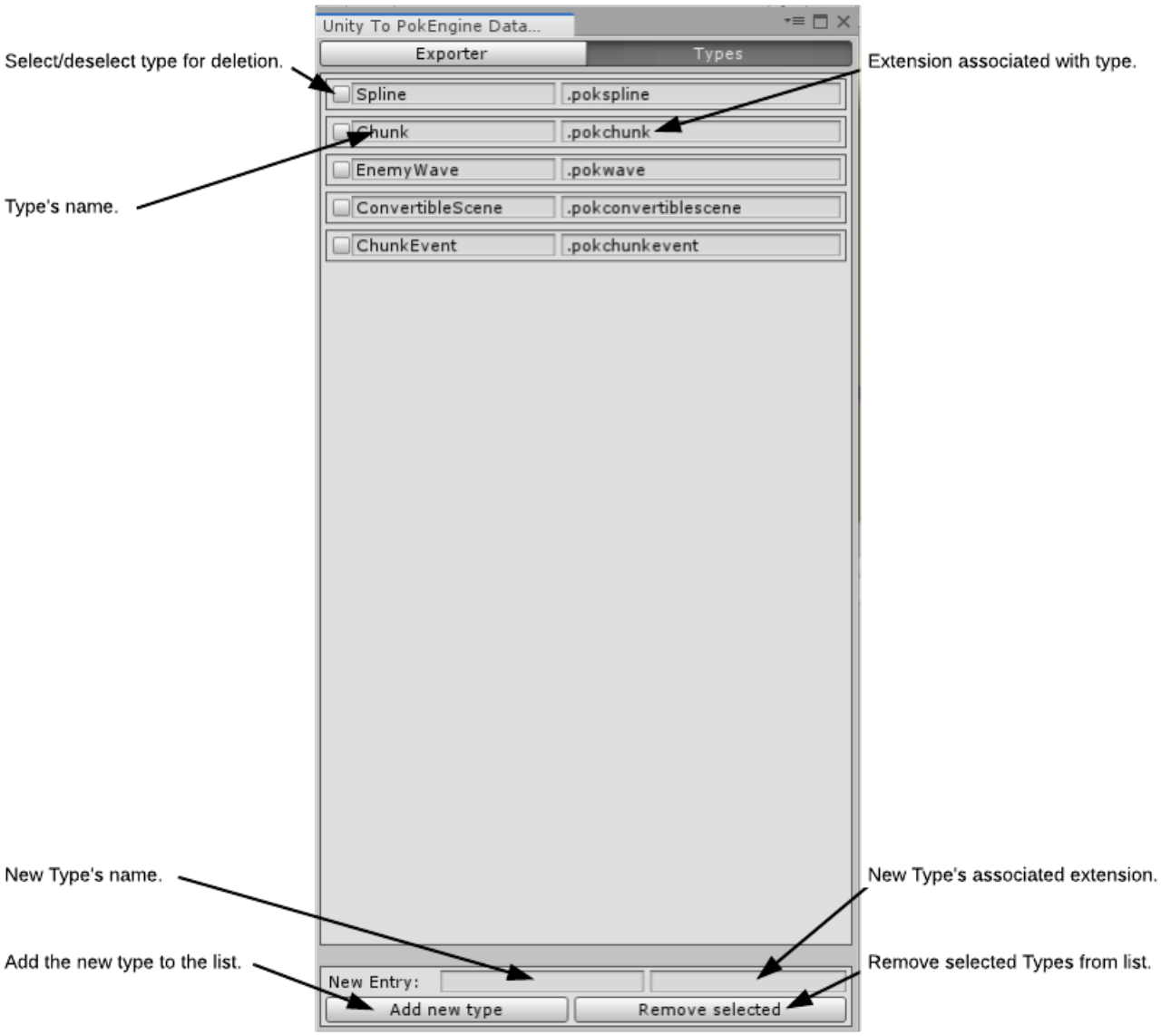
To prevent this problem, an additional parsing pass was added on UPDC's side to **remove any uninitialized fields** from the JSON string provided by JSONUtility before being written down to file.

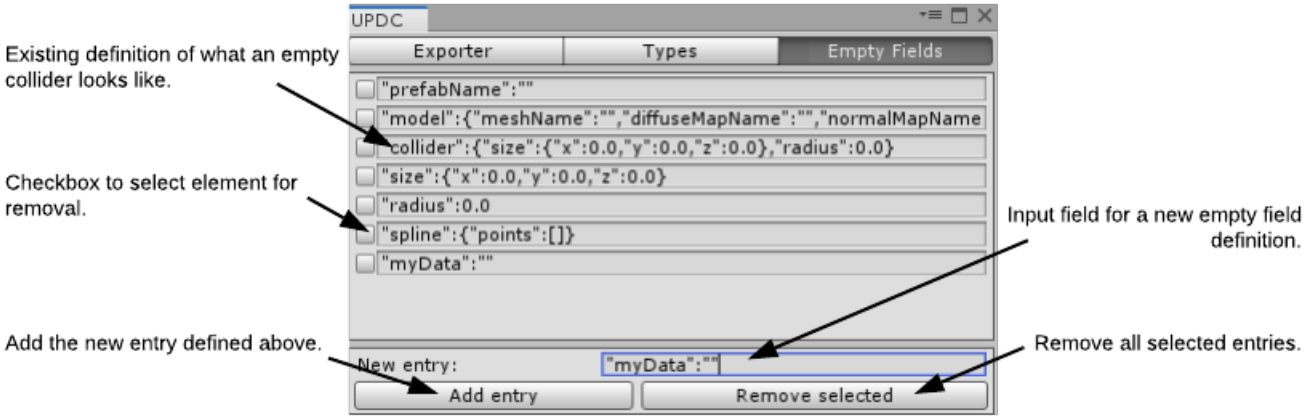
```
{
    "myData":
    [
        {"value": 25.458742},
        {"value": 0},
        {"value": 0},
        {"value": 0}
    ]
}
```

```
{
    "myData":
    [
        {"value": 25.458742}
    ]
}
```

9 Tool's UI




Unity To PokEngine Data...
Exporter
Types
Spline
.pokspline
Chunk
.pokchunk
EnemyWave
.pokwave
ConvertibleScene
.pokconvertiblescene
ChunkEvent
.pokchunkevent
New Entry:
Add new type
Remove selected
Select/deselect type for deletion.
Extension associated with type.
Type's name.
New Type's name.
New Type's associated extension.
Add the new type to the list.
Remove selected Types from list.


UPDC
Exporter
Types
Empty Fields
"prefabName":""
"model":{"meshName":"","diffuseMapName":"","normalMapName
"collider":{"size":{"x":0.0,"y":0.0,"z":0.0},"radius":0.0}
"size":{"x":0.0,"y":0.0,"z":0.0}
"radius":0.0
"spline":{"points":[]}
"myData":""
New entry:
"myData":""
Add entry
Remove selected
Existing definition of what an empty collider looks like.
Checkbox to select element for removal.
Input field for a new empty field definition.
Add the new entry defined above.
Remove all selected entries.

10 Potential Improvements

- Implementation of an new convertible type still requires the user to create their data structure and store the ScriptableObject instances in the correct folder on their own. It would have been better to provide an automated way to do so.
- The user still needs to define the aspect of empty fields for them to be removed from the final JSON file, an automated way to do so would have been welcome.

11 Summary

The UPDC tool allows other Unity tools to export Unity data into JSON files readable by the PokEngine parser and **it's implementation is flexible enough to quickly integrate new data types**.