UPDC
Unity to PokEngine Data Converter
Technical Document

Oleg Loshkin

December 7, 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

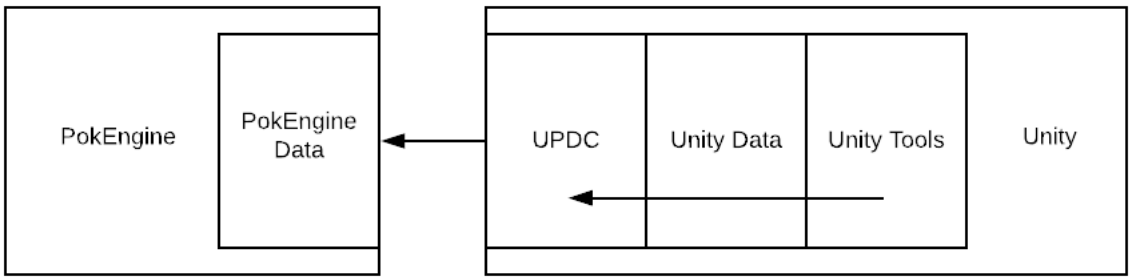
- **PokEngine**
  The **PokEngine** is the game engine developed by the PokFamily team. The engine is **written with C++ standard 2014** and **partly C++ standard 2014** for code running on the Nintendo Switch.
  The engine has a parser that is capable of reading JSON files. This parser is used to import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

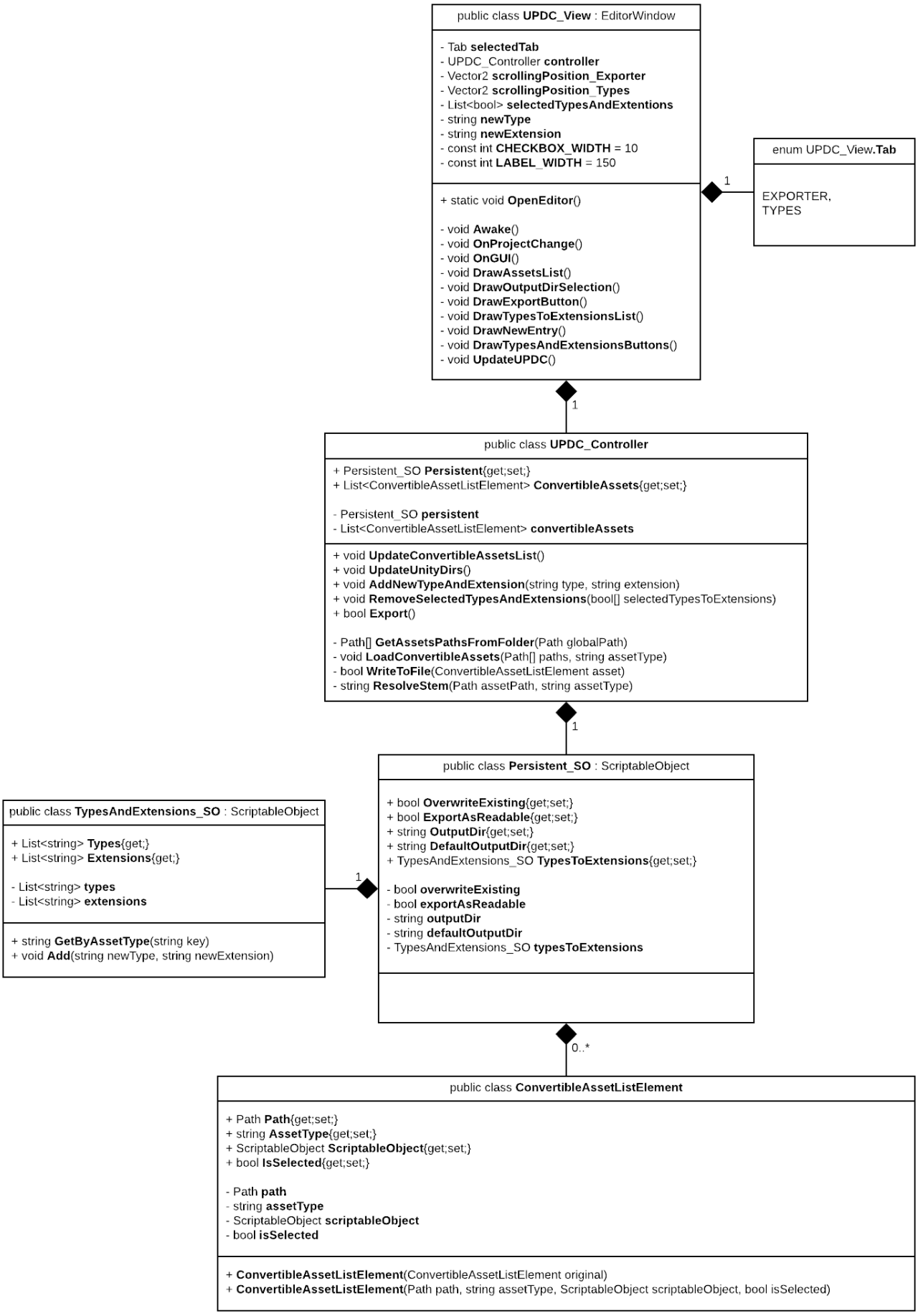
- **Git**
  github.com is used for versioning for the **PokEngine source code**. **gitlab.com** is used for versioning for the **Unity prototype source code**. Git bash is used for most interactions with the git framework. Merge conflicts are solved manually via text editor and git bash.

4 Interaction with Overall Project



UPDC provides 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 side of the project.

5 UML Diagram



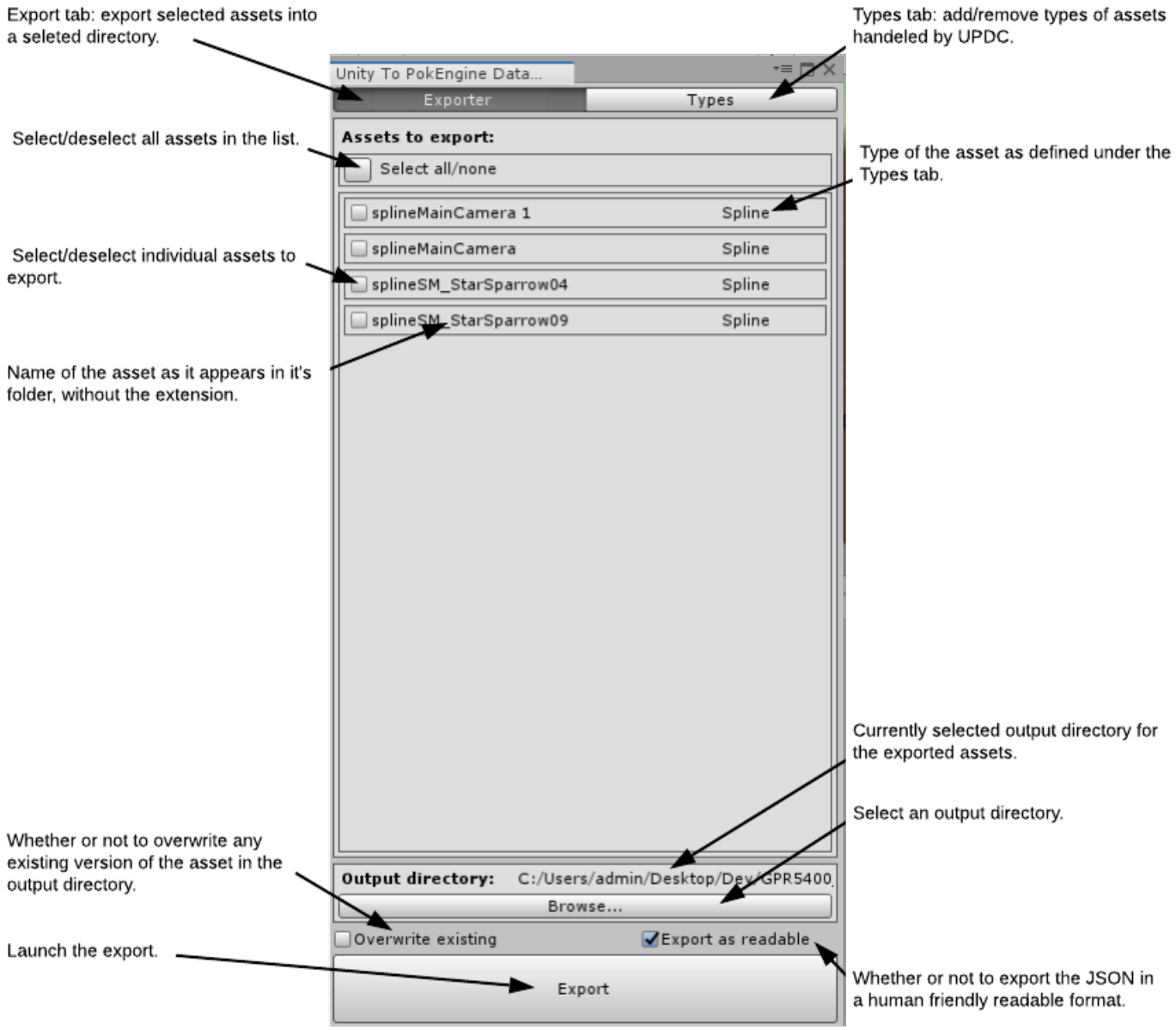
The tool follows a **Model-View-Controller** model where the model is Unity's data, the view is `UPDC_View` and the controller is `UPDC_Controller`.
ScriptableObjects are used to store user data across sessions. Said data includes:

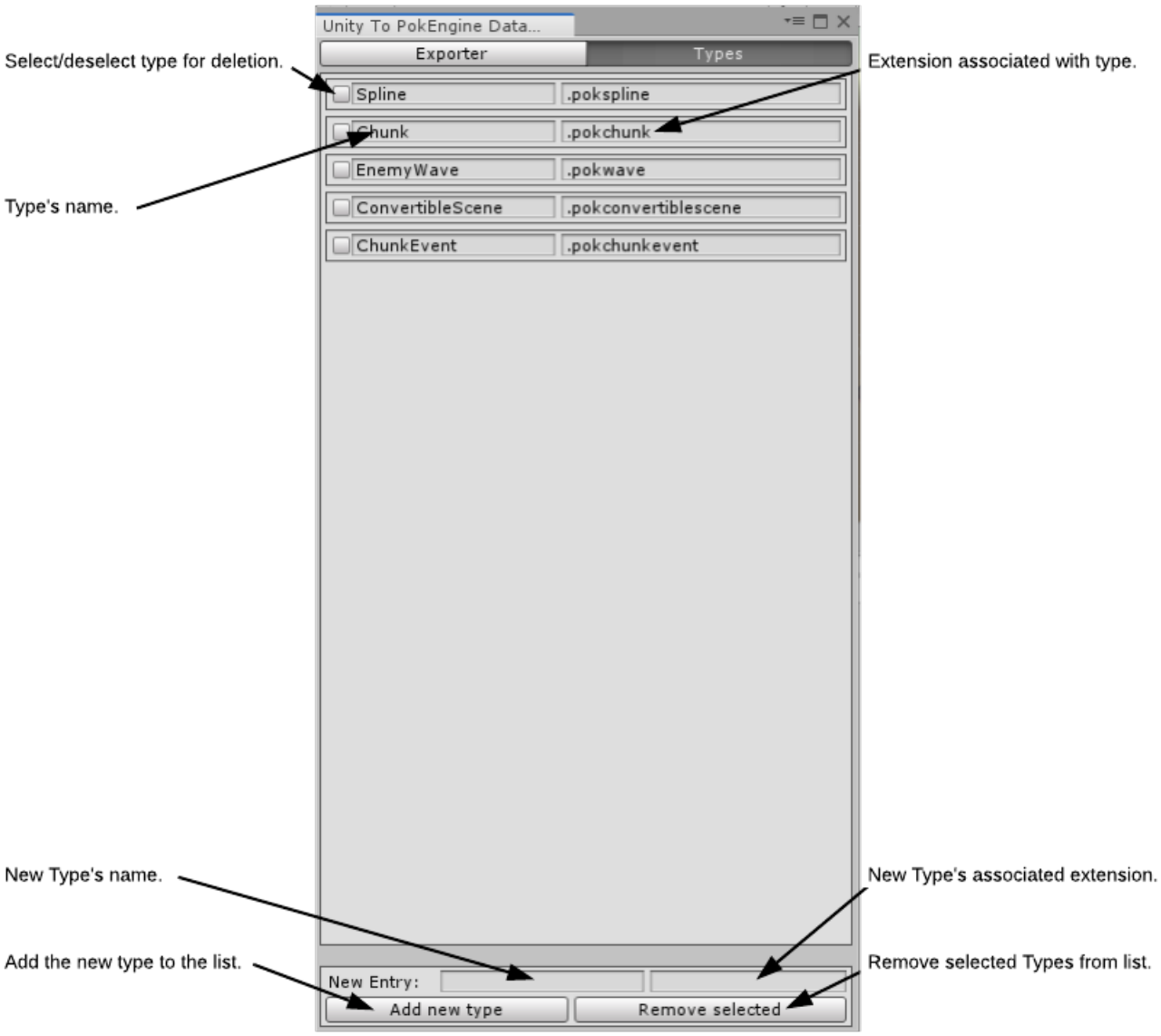
- Whether or not to overwrite any existing files upon export.
- Whether or not to export the JSON data in a format easily readable by humans, or to prefer optimisation.
- Last selected output directory for export.
- Default output directory for export.
- A dictionary like class to keep track of defined types and extensions managed by the UPDC.

6 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. This did however come with the drawback of not being able to ignore empty fields, that are currently written to the JSON file when it would have been preferable to remove these altogether.
By using a string to string dictionary like class (`TypesAndExtensions_SO`), UPDC allows to **add and remove support any new convertible asset types without requiring the user to interact with UPDC's code** provided the type defined inherits from the ScriptableObject class.

7 Tool's UI


Export tab: export selected assets into a seleted directory.
Types tab: add/remove types of assets handeled by UPDC.
Unity To PokEngine Data...
Exporter
Types
Select/deselect all assets in the list.
Assets to export:
Select all/none
Type of the asset as defined under the Types tab.
splineMainCamera 1
Spline
splineMainCamera
Spline
Select/deselect individual assets to export.
splineSM_StarSparrow04
Spline
splineSM_StarSparrow09
Spline
Name of the asset as it appears in it's folder, without the extension.
Currently selected output directory for the exported assets.
Select an output directory.
Output directory:
C:/Users/admin/Desktop/Dev/GPR5400
Browse...
Whether or not to overwrite any existing version of the asset in the output directory.
Overwrite existing
Export as readable
Launch the export.
Export
Whether or not to export the JSON in a human friendly readable format.



8 Potential Improvements

- Currently the user is required to manually set the default output directory of the tool before use.
- A button to remove a given convertible asset from the project directly from the tool's UI would have been a nice addition.
- Reliance on Unity's JSON parser implies an inability to easily remove empty objects from the resulting JSON file, which currently contains empty properties for uninitialized objects.
- 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.

9 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**.
This project's **challenges have been firstly to establish the requirements** of the tool that were left rather nebulous, **learn to use the Unity's GUI API and AssetDatabase API**. The rest hasn't provided much trouble, time being the only constraint for the remaining tasks.
In addressing the first challenge, this project has provided a great opportunity to get first hand experience in working on a complex future commercial title where clear **communication is necessary**. The second and third challenges of the project have allowed me to practice my **ability to tackle unknowns**, and manage them to deliver the required product of the required quality and on time.