

CSC

Convertible Scene Creator

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 **Convertible Scene Creator** tool, **CSC** 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 **export Unity scenes** that may then be used inside the PokEngine.

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 Unity scene and prefab files to files readable by the PokEngine's parser via UPDC.
- 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.

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 data exported from Unity with UPDC.

- **Unity 2019.1.10f**

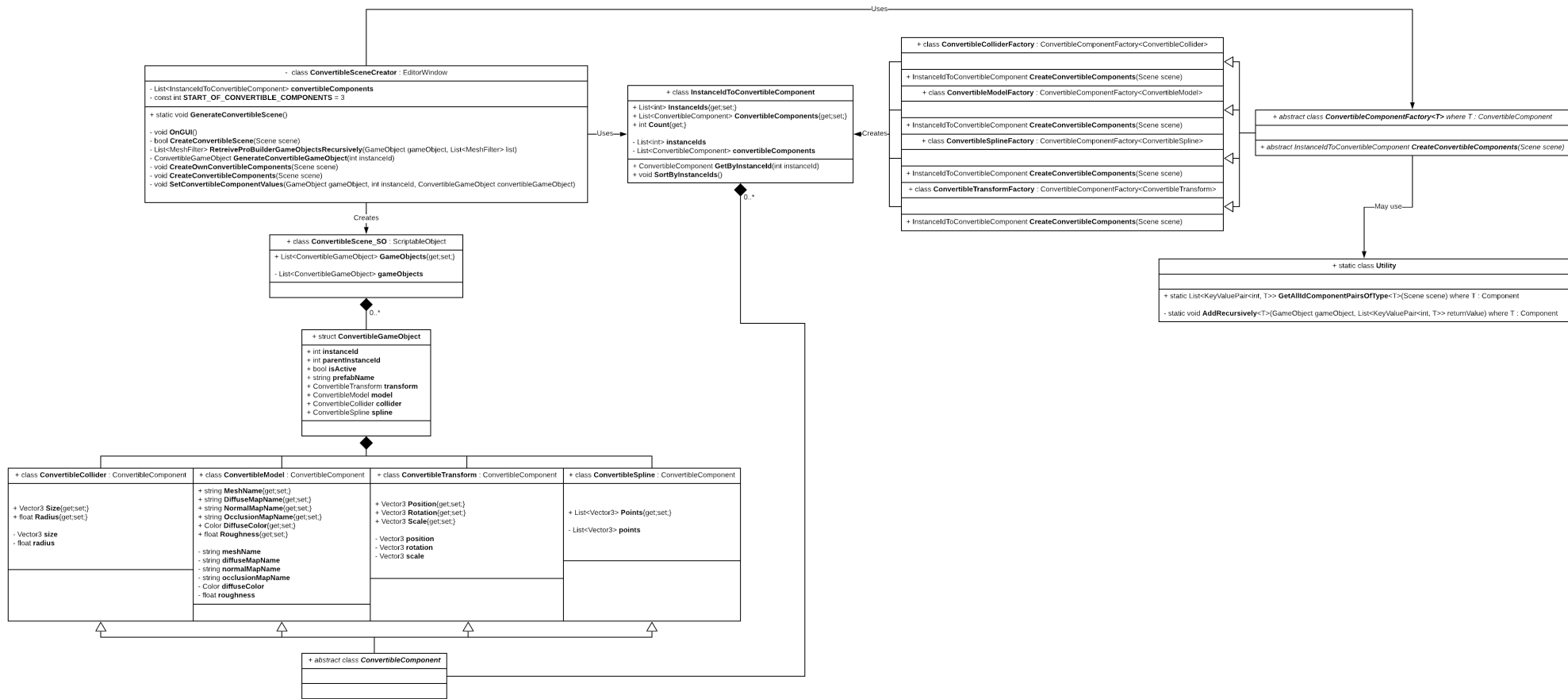
Unity 2019.1.10f is used as an **external editor**.

- **Visual Studio 2017**

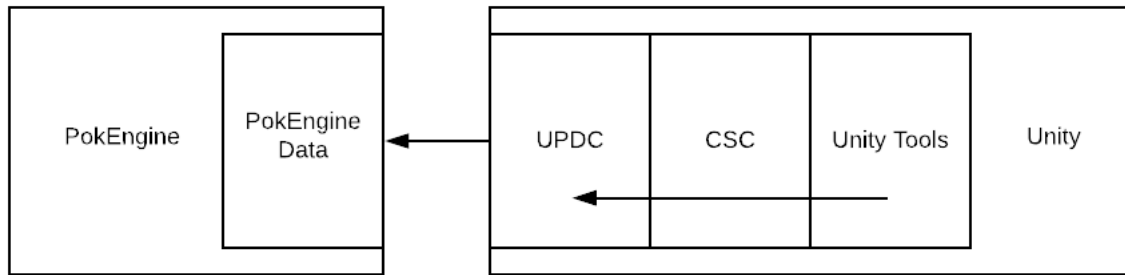
Visual Studio 2017 is used for development of the PokEngine.

- **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.

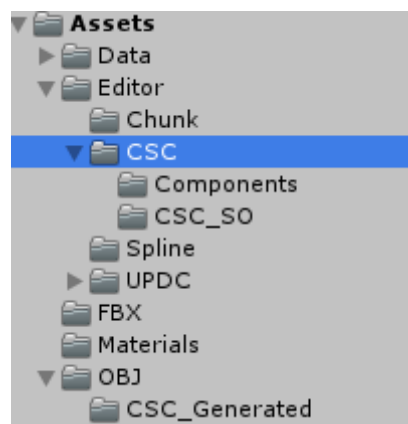


4 Interaction with Overall Project



CSC integrates with UPDC to export scenes and prefabs. As such, it interacts with Unity's database from which it creates .asset files for UPDC to export.

5 Folder Hierarchy



The tool's scripts are stored under "Assets/Editor/CSC". A "Components" folder is located inside for files containing `ConvertibleComponentFactory<T>` and `ConvertibleComponent` inheriting classes.

"CSC_SO" contains the definition of the `ConvertibleScene` class that inherits from `ScriptableObject`.

The "Assets/OBJ/CSC_Generated" contains the .obj files generated from any found ProBuilder game objects in the scene. This folder is emptied every time a scene conversion is launched.

6 Tackling Genericity

In an earlier version of the tool, it was capable of successfully exporting scenes with their game objects, colliders, models and prefabs. However, the implementation exportation of any additional components located on game objects would have required the user to dive deep into the code of the CSC.

This is why the tool was redesigned to allow easier implementation of exportation of new convertible components. This was done by splitting the `ConvertibleSceneCreator` between two .cs files (`ConvertibleSceneCreator.cs` and `Components.cs`) using the "partial" keyword of the C# language.

In doing so, **only the relevant part of CSC code is exposed to the user and the exportation implementation** for new convertible components **can be done by adding a few lines of code only** inside the `ConvertibleSceneCreator` class **at locations very explicitly commented**. The user **no longer needs to understand the functioning of CSC in depth to expand the tool's functionalities**.

The decoupling of the tool in this manner did not require the need for runtime reflection, which can be tricky to implement. All that was needed were a few abstract classes and the use of some C# generics.

7 Tackling Prefabs

Prefabs being too specific to the game engine, only the name of the prefab is exported. This will allow PokEngine to link the appropriate PokEngine prefab to the game object instance in the converted scene.

8 Integrating with ProBuilder

The tool exports ProBuilder generated meshes as .obj files into "Assets/OBJ/CSC_Generated" upon scene conversion using KeliHlodversson's `ObjExporter` class.

`ConvertibleGameObjects` references those files by name to allow PokEngine to load the appropriate .obj files upon rebuilding the scene on it's own side.

9 Potential Improvements

- Reflection might have been used to reduce the amount of code required from the user to write in order to implement the exportation of a new component.
- ProBuilder generating new meshes for each duplicate of a game object, the tool generated a lot of identical meshes that differ in name only. An additional pass might have been added to reduce the amount of .obj files generated.

10 Summary

The CSC allows the user to create a .asset convertible file for UPDC. It allows exportation of a whole scene with components attached to game objects. It also provides the user with an easy way to add new components to include in the convertible scene's .asset file.

The tool also integrates with ProBuilder to export ProBuilder generated meshes into .obj files for use by the PokEngine.