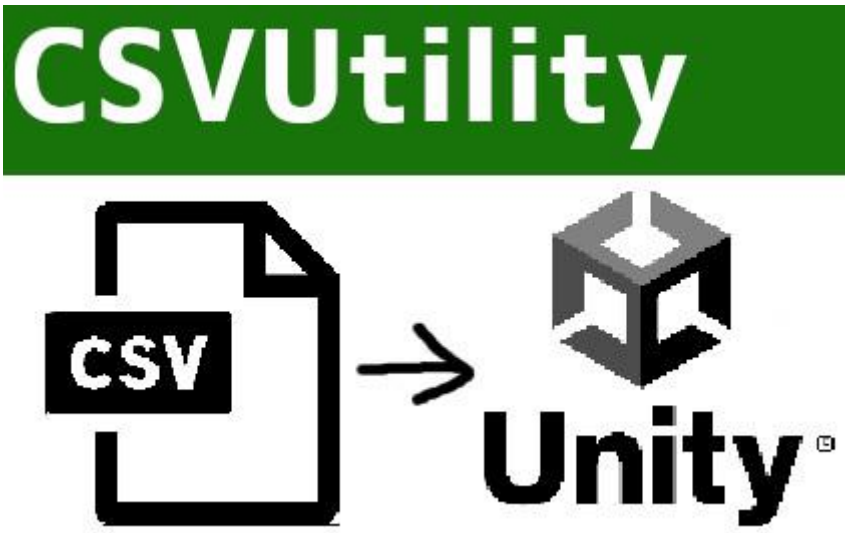
# CSVUtility



# This asset includes

This asset includes a support program to be able to read CSV files and use them as data in the game. Sample scenes are also provided to help you understand how they work.

# What you can do with the CSVUtility

The CSVUtility allows you to use csv files as in-game data. The main functions are as follows

- Importing CSV files
- Export and save CSV files

See next section for actual usage.

## Types of data that can be handled

Data types are as follows

- int
- float
- bool
- string

# Importing CSV files

The following steps are required to read CSV files.

1. Creating a parameter class that will be the type of data
2. Creation of model classes using parameter classes
3. CSV file with data to be used

## Create two classes to handle data

For example, if you are dealing with a class that has the following data

| Type | variable name |
|---|---|
| int | test_int |
| float | test_float |

The class files required for preparation are as follows

```
using anogamelib;

public class SampleModelParam : CsvModelParam
{
    public int test_int;
    public float test_float;

    public override string ToString()
    {
        return $"test_int={test_int} test_float={test_float}";
    }
}

public class SampleModel : CsvModel<SampleModelParam>
{
}
```

The ToString override is only used for debug log display, so no implementation is required.
Note the addition of using anogamelib; and the classes that each class inherits from.
In the above example, the SampleModelParam class corresponds to the type, and CSV file reading and other operations are handled via the SampleModel class.

## Prepare CSV file

The following explanation is based on the csv file corresponding to the class data in the above example.
Please prepare the following data in your spreadsheet tool or text editor.

| | A | B | C |
|---|---|---|---|
| 1 | test_int | test_float | |
| 2 | 123 | 12.3 | |
| 3 | 456 | 45.6 | |
| 4 | | | |

The first line should be the header information, the same text as each parameter name, and the second and subsequent lines should be used to add the data corresponding to each record. In this case, the data is for two records, but you can add as much as you like.
Also, change the extension to csv.

## Sample program to read data

I would like to read the csv file created in the above item in the game. Import the saved data into Unity and prepare the following script.

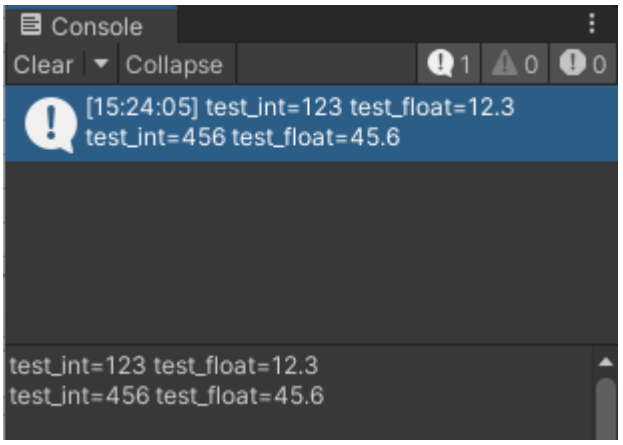
```
using UnityEngine;

public class DemoScript : MonoBehaviour
{
    [SerializeField] private TextAsset localCsvFile;

    private void Start()
    {
        SampleModel sampleModel = new SampleModel();
        sampleModel.Load(localCsvFile);
        sampleModel.CheckDebugLog();
    }
}
```

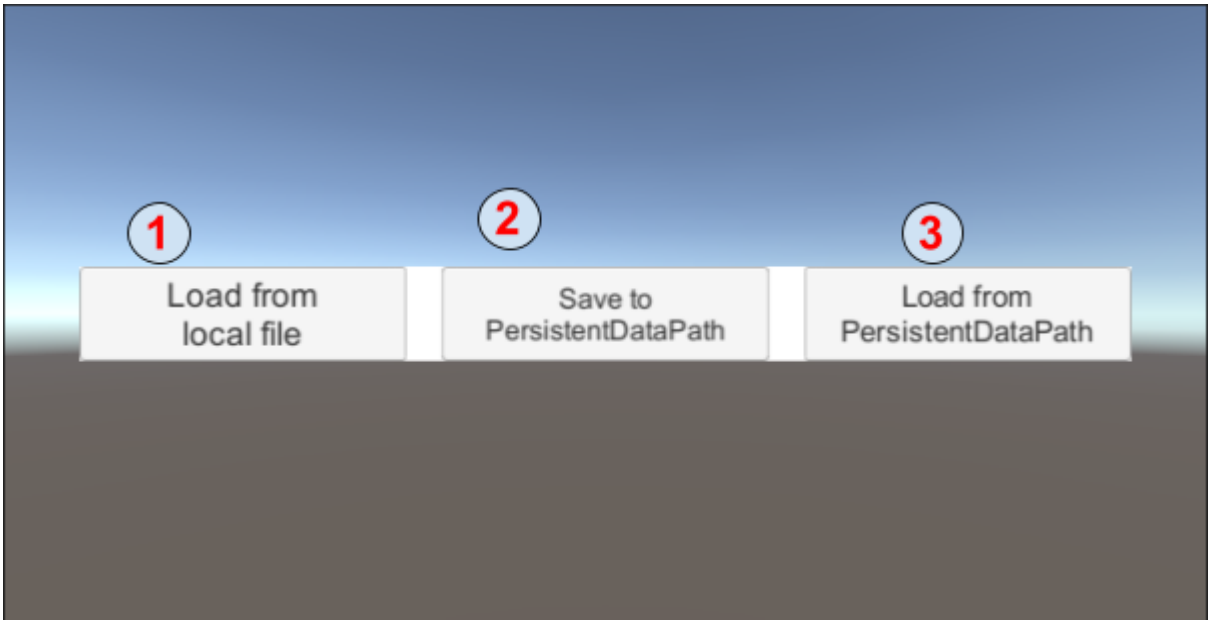
When the script is ready, attach an empty game object to the scene and set the csv file you are saving to the inspector's LocalCsvFile.

If it is done well, you will see the following debug log.




# About the demo scene

In the demo scene, you will see the following operations




1. Demonstration of loading a local csv file.
2. Save the loaded model to PersistentDataPath.
3. Load a csv file saved in PersistentDataPath.

By using 2 and 3 well, it is possible to save and load game data.
Please check the debug log for the save location and other information.