

Save Game Tool

Documentation

Quick instructions

There are 3 main elements in the tool:

- Save Data Preference object
- Save Tool window
- SaveData class

In the Save Data preference object you can configure all the basic information of the system, information as save file name, save file format and if the save file will be encrypted. This object can be found in the Resources folder.

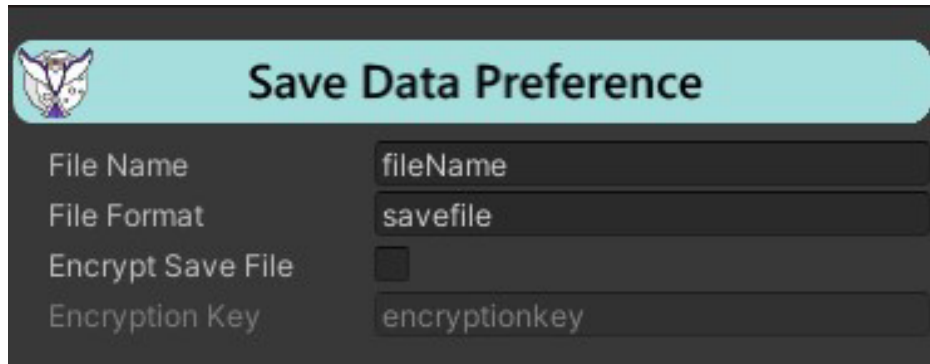
The Save Tool window allows you to decrypt and encrypt external save files for debugging, analysis, or anything you need. It also allows you to create a new Save Data Preference object if needed.

The SaveData class is how you will save your game information, multiples files and multiples variables type are supported. The class methods that can be used are *SaveValue(key, value)*, *LoadValue(key, out variable)*, *SaveDataFile(index)* and *LoadDataFile(index)*.

If you have any issue or question about the tool, want to give feedback or suggestions or want to reuse the code of my tool to create your own please contact me at arthemydevelopment@gmail.com.

Save Data Preference

The Save Data Preference is an Scriptable Object that contains the basic options of the Save System, if the object is missing or was deleted by accident it can be created with the Save Tool window, it will also be automatically created when the system is used.

A screenshot of a software window titled "Save Data Preference". The window has a dark grey background. At the top left is a small icon of a white owl with purple wings and eyes. To the right of the icon is a light blue header bar with the title "Save Data Preference" in black text. Below the header, there are four settings: "File Name" with a text field containing "fileName", "File Format" with a text field containing "savefile", "Encrypt Save File" with an unchecked checkbox, and "Encryption Key" with a text field containing "encryptionkey".

File Name	fileName
File Format	savefile
Encrypt Save File	<input type="checkbox"/>
Encryption Key	encryptionkey

These options are:

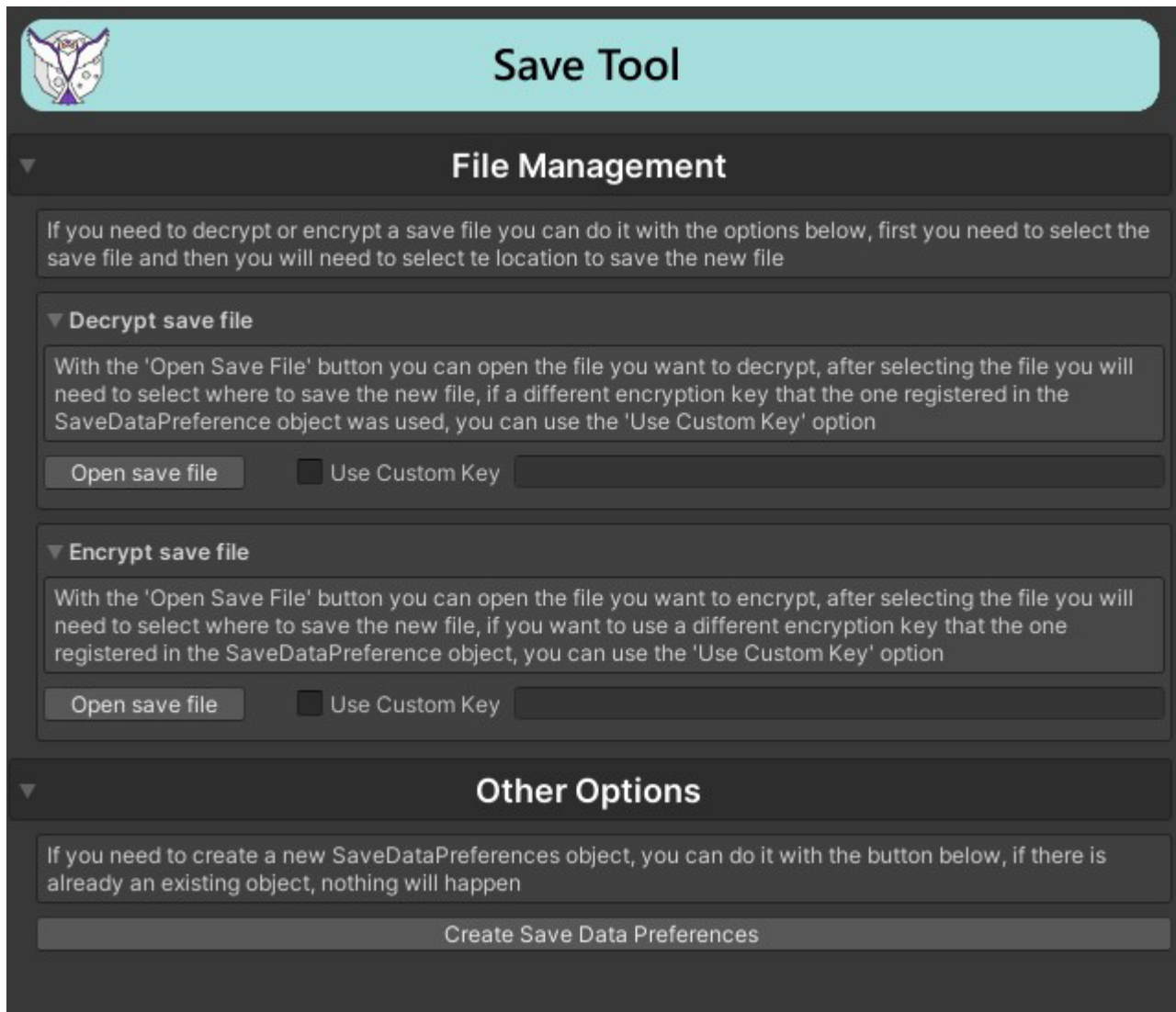
- File Name: The default name of the save files.
- File Format: The file format of the save files, this can be whatever you want, commons formats are "savefile" "save" "sav".
- Encrypt Save File: This allows you to turn on the encryption of the save files, an encrypted file will appear as byte information and won't be readable by a person.
- Encryption Key: This allows you to define the key for the encryption method, only alphanumeric characters are allowed.

The save location of the file is C:/Users/"User"/AppData/LocalLow/"Company"/"Project", the exact location base in your PC and project configuration can be found in the Unity console when a file is created.



Save Tool

The Save Tool window is divided into two sections, File Management and Other Options.



The image shows a screenshot of the 'Save Tool' application window. At the top, there is a light blue header bar with a small owl logo on the left and the title 'Save Tool' in the center. Below the header, the window is divided into two main sections. The first section, 'File Management', has a dark grey background and contains two sub-sections: 'Decrypt save file' and 'Encrypt save file'. Each sub-section includes a text box explaining the process, an 'Open save file' button, a checkbox for 'Use Custom Key', and a text input field for the key. The second section, 'Other Options', also has a dark grey background and contains a text box explaining how to create a new 'SaveDataPreferences' object, followed by a 'Create Save Data Preferences' button.

Save Tool

File Management

If you need to decrypt or encrypt a save file you can do it with the options below, first you need to select the save file and then you will need to select the location to save the new file

▼ **Decrypt save file**

With the 'Open Save File' button you can open the file you want to decrypt, after selecting the file you will need to select where to save the new file, if a different encryption key than the one registered in the SaveDataPreference object was used, you can use the 'Use Custom Key' option

Open save file ☐ Use Custom Key

▼ **Encrypt save file**

With the 'Open Save File' button you can open the file you want to encrypt, after selecting the file you will need to select where to save the new file, if you want to use a different encryption key than the one registered in the SaveDataPreference object, you can use the 'Use Custom Key' option

Open save file ☐ Use Custom Key

Other Options

If you need to create a new SaveDataPreferences object, you can do it with the button below, if there is already an existing object, nothing will happen

Create Save Data Preferences

In File Management you can Decrypt or Encrypt external save files for debugging or analysis. The Encryption key can be the same that is registered in the Save Data Preference Object or a different one if the key was changed at some point of the development, just input the different key in the "Use Custom Key option".

In Other Options you can create a new Save Data Preference object if the previous one was deleted or is missing.



Save System

The Save System let you save multiple type of variables easilly with only one line of code.

The compatible variables are:

- String
- Int
- Bool
- Float
- Vector 2
- Vector 3
- Custom Class

For saving a custom class the interface "ISaveClass" must be added, only serializable variables can be saved from this class.

To use the system, an instance of "SaveData" must be declared in the script, this class contains the next methods:

- SaveValue(key, value)
With SaveValue you can select data to save, a indetifier key and the value is required.
- LoadValue(key, out variable)
With LoadValue you can retrive the saved data, the identifier key and a variable is requierd.
- SaveDataFile(index)
With SaveDataFile you can save the game data into a file, if you want multiple save files, a int number can be used as index, otherwise this is not required. Must be the last thing to do when saving a file.
- LoadDataFile(index)
With LoadDataFile you can load the save file, if an index was used in SaveDataFile, the same index must be used when loading the file. Must be the first thing to do when loading a file.



Save System Example

The next example can be found in the Example folder of the asset.

```
//The SaveData instance declaration  
SaveData SD = new SaveData();
```

The SaveData instance is declared, in this instance you will save all your information.

```
//Saving the variables value and link them to their key  
SD.SaveValue(ValuesKeys.PlayerName, ExampleString);  
SD.SaveValue(ValuesKeys.SuperJump, ExampleBool);  
SD.SaveValue(ValuesKeys.Hp, ExampleInt);  
SD.SaveValue(ValuesKeys.Damage, ExampleFloat);  
SD.SaveValue(ValuesKeys.MinimapPosition, ExampleVector2);  
SD.SaveValue(ValuesKeys.Position, ExampleVector3);  
SD.SaveValue(key: ValuesKeys.CustomClass, ExampleCustomClass);  
  
//Saving the file after every variable has been saved  
SD.SaveDataFile();
```

Using the SaveValue Method inside of SaveData all the variables values are being saved, an enum is being used as the keys and the variables are given as the value.

At the after that all the information is saved into a file.



Save System Example

```
//Loading the save file
SD.LoadDataFile();

//Retrieving the values to their variables using the keys
SD.LoadValue(ValuesKeys.PlayerName, out ExampleString);
SD.LoadValue(ValuesKeys.SuperJump, out ExampleBool);
SD.LoadValue(ValuesKeys.Hp, out ExampleInt);
SD.LoadValue(ValuesKeys.Damage, out ExampleFloat);
SD.LoadValue(ValuesKeys.MinimapPosition, out ExampleVector2);
SD.LoadValue(ValuesKeys.Position, out ExampleVector3);
SD.LoadValue(key: ValuesKeys.CustomClass, out ExampleCustomClass);
```

First, the save file is loaded, then every value is retrieved using the key and the variable where they belong.

```
public class Example : ISaveClass //A class compatible with the system thanks to the ISaveClass interface
{
    public int PlayerIndex;  ⚡ Serializable
    public string PlayerTag;  ⚡ Serializable
}
```

To save a custom class, the ISaveClass interface must be added to the class and all the variables must be serializables and the class must have the Serializable attribute.

