Saving Game information to a Config File

As well as loading information from a config file, you can also save the config in memory (completely or partially) to a file.

This can be used for all manner of things like persisting scores, player information or any game information to a file that can be loaded back later.

Here's the function you need to call in order to do so:

```
orxConfig_Save("MySavedConfigFile.ini", bEncrypt, MySaveFilterCallback);
```

1. The first parameter is the filename you wish to save your config to. The extension can be anything you like. I usually just stick to `ini`.
2. The second parameter indicates if the file is to be written encrypted so that it won't be human readable. Handy for stopping game cheaters.
3. The last parameter is a callback function that will be called for every section and every key to let you decide if you want to save this info to the file or not.

If you pass orxNULL instead of a valid callback, every single key/value pair will be saved.

> Please note that the any comments within the original loaded config files won't be saved. They're ignored during loading and never stored in memory.

Let's work through a saving example.

Just say we had an initial config loaded by your game that was something like this:

```
[Input]
SetList = KeysForInput

[KeysForInput]
KEY_ESCAPE = Quit

[Mouse]
ShowCursor = true

[Display]
ScreenWidth = 1024
ScreenHeight = 768
```

When your Orx game loads, this config would be loaded into memory.

You can then add to the config that is in memory. To add something like a highscore value to the config in memory, we can do it with:

```
if (orxConfig_PushSection("SaveSettings")) {
```

```
    orxConfig_SetS64("HighScore", score);
    orxConfig_PopSection();
}
```

This would be added to the config in memory and the entire result would become:

```
[Input]
SetList = KeysForInput

[KeysForInput]
KEY_ESCAPE = Quit

[Mouse]
ShowCursor = true

[Display]
ScreenWidth = 1024
ScreenHeight = 768

[SaveSettings]
HighScore = 53281
```

We could then save the entire config in memory to an external file with:

```
orxConfig_Save("MySavedConfigFile.ini", orxFALSE, orxNULL);
```

Rarely would you be dumping the entire config. In our case, we only want to save the `SaveSettings` section to disk.

This is where the callback filter comes into play.

Your `MySaveFilterCallback` callback which is called many times by `orxConfig_Save` for each section and key.

This allows you to decide, in code, if you want to save some parts of this section (by returning orxTRUE) or if you'd rather skip the whole section completely (by returning orxFALSE).

This may not be very clear, so lets look at an example callback:

```
orxBOOL orxFASTCALL MySaveFilterCallback(const orxSTRING _zSectionName,
const orxSTRING _zKeyName, const orxSTRING _zFileName, orxBOOL
_bUseEncryption)
{
    // Return orxTRUE for the section "SaveSettings", orxFALSE otherwise
    // -> filters out everything but the "SaveSettings" section
    return (
        orxString_Compare(_zSectionName, "SaveSettings") == 0
        ) ? orxTRUE : orxFALSE;
}
```

This will filter out all the config sections that are in memory, except for the `SaveSettings` section.

So if you call:

```
orxConfig_Save("MySavedConfigFile.ini", orxFALSE, MySaveFilterCallback);
```

Then only the `SaveSettings` section will appear in your `SavedConfigFile.ini` file.

Of course, to load your save file back, simply do a:

```
orxConfig_Load("MySavedConfigFile.ini");
```

That should be enough to get you going, creating save files for your games.

Saving only certain filtered Keys from a Section

In order to save only a particular key or keys from a config (instead of saving all keys from a config, we need to change our callback. Let's say our `SaveSettings` section contained two keys:

```
[SaveSettings]
HighScore = 53281
LastLevel = 6
```

If we only wanted the `LastLevel` key to be saved out to the file, the callback would look like this:

```
orxBOOL orxFASTCALL MySaveFilterCallback(const orxSTRING _zSectionName,
const orxSTRING _zKeyName, const orxSTRING _zFileName, orxBOOL
_bUseEncryption)
{
    // Filters out everything but "SaveSettings" section.
    // And only the "LastLevel" key for the "SaveSettings" section.
    return (
        orxString_Compare(_zSectionName, "SaveSettings") == 0 &&
        (_zKeyName == orxNULL || orxString_Compare(_zKeyName, "LastLevel")
== 0)
        ) ? orxTRUE : orxFALSE;
}
```

In the code above, we compare the section name to find `SaveSettings`. Then the comparing of `_zKeyName` to orxNULL will get the section name itself. Then the comparing to `LastLevel` will get the specific key.

More information

There is a lot more that the config API can do. You can explore more config functions at:

http://orx-project.org/orx/doc/html/group__orx_config.html

There are some usage examples available in the Orx Curve Editor that is currently in development at: https://gitlab.com/sausagejohnson/orx-curvefx-editor

Also, don't forget there are a number of other config tutorials available.