# Main settings

Here are defined all the available config properties that will be recognized by orx as main settings. All these settings can be found in the SettingsTemplate.ini file along with their short descriptions.

Please refer to the syntax page for more info on how to write config files.

## Config module

### Summary

```
[Config]
DefaultParent = <string>
```

### Details

Only one section (`Config`) is defined for this module. Here's a list of its available properties:

- `DefaultParent`: This section will be used as implicit parent for any other config section. By default there's no default parent section.

## Console module

### Summary

```
[Console]
ToggleKey = KEY_*
ScrollSize = <uint>
Alias = <command>
MyOtherAlias = <command>
```

### Details

Only one section (`Console`) is defined for this module. Here's a list of its available properties:

- `ToggleKey`: Defines the toggle key for activating the in-game console. A list of available keys can be found here. If an invalid key is provided, the console won't be available to the user. Defaults to KEY_BACKQUOTE.
- `ScrollSize`: Number of lines to scroll at a time, defaults to 3.
- `Alias`: Aliases can be defined as shortcuts for commands. Those shortcuts can include parameters in their order of use. Any other key than ToggleKey will be added as a command alias. Commands are case insensitive. For example, `FPS = Config.SetValue Render`

ShowFPS, will create an alias named FPS. Calling it with `FPS true` will expand to the final command `Config.SetValue Render ShowFPS true` and will enable the on-screen display of the framerate. Aliases can be nested.
- `MyOtherAlias`: Any other key than ToggleKey will be added as a command alias. Commands are case insensitive.

# Clock module

## Summary

```
[Clock]
MainClockFrequency = <float>
```

## Details

Only one section (`Clock`) is defined for this module. Here's a list of its available properties:

- `MainClockFrequency`: This property allows to give a frequency for the main (aka `Core`) clock update. If no value is provided, the main clock will be updated as often as possible. Best results are achieved by not setting any value for this property and by enabling vertical synchronization.

# Display module

## Summary

```
[Display]
ScreenWidth         = <int>
ScreenHeight        = <int>
ScreenDepth         = <int>
ScreenPosition      = <vector>
RefreshRate         = <int>
FullScreen          = <bool>
Decoration          = <bool>
AllowResize         = <bool>
Title               = TitleText
Smoothing           = <bool>
VSync               = <bool>
DepthBuffer         = <bool>
ShaderVersion       = <int>
ShaderExtensionList = <+ | ->
Monitor             = <int>
Cursor              = arrow | ibeam | crosshair | hand | hresize | vresize |
default | path/to/texture # <vector>
IconList            = path/to/texture1 # ... # path/to/textureN
```

```
DebugOutput          = <bool>
```

### Details

Only one section (`Display`) is defined for this module. Here's a list of its available properties:

- `ScreenWidth`: Resolution width for the main display in pixels. Must be provided. For Android, limit the width of the display framebuffer
- `ScreenHeight`: Resolution height for the main display in pixels. Must be provided. For Android, limit the height of the display frame buffer (ignored if ScreenWidth is defined).
- `ScreenDepth`: Resolution depth for the main display in bits. Defaults to 32bit.
- `ScreenPosition`: Only used in windowed mode: position of the window in 'desktop' screen space. Defaults to OS positioning when not specified.
- `FullScreen`: Defines if the main display will be full screen or windowed. This property is set to `false` by default.
- `RefreshRate`: Fullscreen refresh rate. Defaults to 60Hz. For Android, defaults to OS refresh rate.
- `Decoration`: In windowed mode, this will define if decorations (ie. window borders, buttons, ...) will be used for the main display window. This property is enabled by default. Only used when not in fullscreen.
- `AllowResize`: Allows manual resize of the window by the user. Only works in windowed mode, defaults to false.
- `DepthBuffer`: Defaults to false, set it to true only if you plan on doing 3D rendering on your own and you need a depth buffer to be created.
- `ShaderVersion`: If defined, a matching shader version preprocessor directive will be added to the top of fragment shaders. Default is 120 which is shader version 1.20 [1)].
- `ShaderExtensionList`: If defined, shader extension directives will be added to the top of fragment shaders.
- `Monitor`: Index of monitor, defaults to 1 = primary monitor.
- `Cursor`: Defaults to 'default'. If a texture is provided, an optional vector can be added as the hotspot.
- `IconList`: Up to 16 icons can be defined, the one with the best-fitting resolution will be used automatically.
- `DebugOutput`: OpenGL platforms only. Applied upon init or when setting video mode, defaults to false.
- `Smoothing`: Sets the default rendering smoothing mode (ie. with or without antialiasing). Defaults to `false` (ie. no antialiasing).
- `Title`: Title for the main display. Will be empty if none is provided.
- `VSync`: Enables/disables the vertical synchronization. Will use system defaults if none is provided. Best results are usually achieved by enabling this property.

# Input module

### Summary

```
[Input]
DefaultThreshold  = <float>
```

```
DefaultMultiplier = <float>
SetList           = InputSet1 # InputSet2

[InputSet1]
KEY_SPACE   = Jump#Validate
MOUSE_LEFT  = Select
KEY_LCTRL   = Select
JOY_1_1     = Attack
JOY_LX_2     = Move
MOUSE_X     = Move

CombineList = Select # Attack
```

### Details

One main section (`Input`) and as many subsections as defined sets are used for this module. Here's a list of the available parameters for the main `Input` section:

- `DefaultThreshold`: Sets the threshold value [2] under which input values will be ignored. Defaults to 0.15.
- `DefaultMultiplier`: Defines a value in ]0.0, +inf[ by which all input values will be multiplied. Defaults to 1.0;
- `SetList`: Provides a list of all available input sets. All these sets need to be defined in their own config section so as to be considered valid. The first valid set of this list will be selected by default as the current input set.

For every input set, here's a list of the available parameters:

- <button>/<key>/<joystick axis>: For every possible physical input [3], one or more actions are linked. Every time this physical input gets activated, the corresponding input actions will be triggered.
- `CombineList`: Provides a list of all the input actions that needs all of their physical inputs to be active before being triggered. If an action isn't listed here, every time any of its linked physical input is activated, it will be triggered.

*NB: Every input action can be linked to up to 4 different physical inputs.*

Here are lists of the available physical inputs:

- Joystick inputs
- Keyboard inputs
- Mouse inputs

## iOS module

### Summary

```
[iOS]
AccelerometerFrequency = <float>
```

### Details

#### This section is only used for iOS (iPhone/iPod Touch/iPad).

Only one section (`iOS`) is defined for this module. Here's a list of its available properties:

- `AccelerometerFrequency`: Sets the polling frequency, in Hz, of the accelerometer. If explicitely set to 0, the accelerometer will be disabled. Defaults to iOS's default value;

## Locale (localization) module

### Summary

```
[Locale]
LanguageList = Language1 # Language2

[Language1]
MyTextEntry      = <string>
MyOtherTextEntry = <string>
```

### Details

One main section (`Locale`) and as many subsections as defined languages are used for this module. Here's a list of the available parameters for the main `Locale` section:

- `LanguageList`: Provides the available languages for the localization module. A language will only be considered valid if a corresponding section with the same name exists. The first valid language of this list will be selected by default for the current language.

For every languages, there can be as many keys as needed. Each key/value pair is made of the key used in code with the `orxLocale` module [4] and its corresponding content translated in the current language.

Here's a simple example.

```
[Locale]
LanguageList = English # French

[English]
Greetings = Welcome everybody!

[French]
Greetings = Bienvenue à tous!
```

When calling

```
orxLocale_GetString("Greetings");
```

the localized content in the current language will be returned.

# Mouse module

## Summary

```
[Mouse]
ShowCursor = <bool>
```

## Details

Only one section (`Mouse`) is defined for this module. Here's a list of its available properties:

- `ShowCursor`: Tells if the mouse cursor should be displayed or not. This property is set to `true` by default.

# Param (command line parameters) module

## Summary

```
[Param]
config = path/to/config1 ... path/to/configN
plugin = path/to/Plugin1 ... path/to/pluginN
```

## Details

Only one section (`Param`) is defined for this module. Every command line parameter registered in code to the `orxParam` module [5] can be defined here. The long name of the parameter has to be used.

If a param is defined directly on the command line, it will prevail on any value entered in this config section.

Internally, orx defines a `config` command line parameter allowing to provide extra config files in addition to the main one [6].
It also defines a `plugin` command line parameter allowing to load external plugins when initialized.

*NB: Lists are not used for these properties, you need to provide the parameter value in the same way you would do it on the command line.*

To learn how to use orxParam, see: Commandline Parameters for your game or application

# Physics module

Module that handles physics interactions.

### Summary

```
[Physics]
AllowSleep          = <bool>
DimensionRatio      = <float>
Gravity             = <vector>
IterationsPerStep   = <int>
ShowDebug           = <bool>
```

### Details

Only one section (`Physics`) is defined for this module. Here's a list of its available properties:

- `AllowSleep`: Defines if objects are allowed to go into sleep mode in the physics simulation when not stimulated by anything. This improves performances in the physics simulation and should be set to true unless you have a good reason. Its default value is `true`.
- `DimensionRatio`: Defines the ratio between orx's world (sizes are expressed in pixels) and the physics simulation world (sizes are expressed in meters). Its default value is 0.01 which means 1 pixel is represented by 0.01 meters. [7)]
- `Gravity`: Defines the gravity vector used in the physics simulation. Please remember that orx 2D vertical axis Y is oriented toward the bottom of your screen [8)]. If not provide, the gravity will be null, ie. body won't “fall” by default.
- `IterationsPerStep`: Defines the number of iterations made by the physics simulation for every step (frame). The higher this number is, the more precise and time consuming the simulation will be. Its default value is 10, don't change it unless you feel you could use a better precision or, on the contrary, a faster physics simulation.
- `ShowDebug`: If set to true, debug shapes will be drawn for all physics bodies in Debug and Profile modes only. Defaults to false.

Related config options used on objects with bodies can be found at: orxBODY structure

# Plugin module

This module provides functions for loading dynamic code into the core engine and finding symbols within the dynamic modules.

### Summary

```
[Plugin]
DebugSuffix = DebugSuffixString
```

### Details

Only one section (`Plugin`) is defined for this module. Here's a list of its available properties:

- `DebugSuffix` : The suffix to add to your game. If none is given, orx will use “d” as default. For example, the debug version of mygame.exe would be mygamed.exe unless specified otherwise.

## Render module

Render plugin module. Renders visible objects on screen, using active cameras/viewports.

### Summary

```
[Render]
ShowFPS                = <bool>
ShowProfiler           = <bool>
MinFrequency           = <float>
ProfilerOrientation    = portrait|landscape
ConsoleBackgroundColor = <vector>
ConsoleBackgroundAlpha = <float>
ConsoleSeparatorColor  = <vector>
ConsoleSeparatorAlpha  = <float>
ConsoleLogColor        = <vector>
ConsoleLogAlpha        = <float>
ConsoleInputColor      = <vector>
ConsoleInputAlpha      = <float>
ConsoleCompletionColor = <vector>
ConsoleCompletionAlpha = <float>
```

### Details

Only one section (`Render`) is defined for this module. Here's a list of its available properties:

- `ShowFPS`: Displays current FPS in the top left corner of the screen. Its default value is `false`.
- `ShowProfiler`: Tells orx to display the CPU profiling marker values in real time. Its default value is `false`.
- `MinFrequency`: This defines the minimal frequency for the rendering clock [9]. This means that if your game framerate drops below this frequency, your clocks will be provided a DT maxed by this frequency, resulting in a smooth slowdown of your game rather than in a jerky/laggy

rendering. Uses your target framerate as value here (often 30 or 60 Hz). Its default value is 60Hz, meaning that a game that won't be able to render at least 60 fps will appear to run slower than it should. Defaults to 10Hz. If you don't want a maxed DT, just set explicitly a negative value.
- `ProfilerOrientation`: portrait|landscape; NB: Defaults to landscape;
- `ConsoleBackgroundColor`: If specified, will override console's background color.
- `ConsoleBackgroundAlpha`: If specified, will override console's background alpha.
- `ConsoleSeparatorColor`: If specified, will override console's separator color.
- `ConsoleSeparatorAlpha`: If specified, will override console's separator alpha.
- `ConsoleLogColor`: If specified, will override console's log color.
- `ConsoleLogAlpha`: If specified, will override console's log alpha.
- `ConsoleInputColor`: If specified, will override console's input color.
- `ConsoleInputAlpha`: If specified, will override console's input alpha.
- `ConsoleCompletionColor`: If specified, will override console's completion color.
- `ConsoleCompletionAlpha`: If specified, will override console's completion alpha.

# Resource module

## Summary

```
[Resource]
Config              = path/to/storage1 # ... # path/to/storageN
Sound               = path/to/storage1 # ... # path/to/storageN
Texture             = path/to/storage1 # ... # path/to/storageN
WatchList           = Texture # Config # Sound # ... ; NB: If defined, orx
will monitor these groups of resources and will reload them as soon as
modified. Dev feature, only active on computers;

UserResourceGroupX = path/to/storage1 # ... # path/to/storageN
```

## Details

- `Config`: Config group: orx will look for resources following the order defined by this list, from first to last.
- `Sound`: Sound group: orx will look for resources following the order defined by this list, from first to last.
- `Texture`: Texture group: orx will look for resources following the order defined by this list, from first to last.
- `WatchList`: If defined, orx will monitor these groups of resources and will reload them as soon as modified. Dev feature, only active on computers.

Only one section (`Resource`) is defined for this module. This section allows the user to define all the `storages`[10] for all the resource groups.

In addition to orx's own internal groups (respectively `Config`, `Sound` and `Texture`), the user can add his own groups for his own custom resources.

When a resource of a given group is requested, orx will go through all the storages that have been

defined for that group[11] till a resource with a matching name is found.

By default, orx only has a single resource type[12]: the file system.

However new custom-defined resource types, such as network URIs, compressed archives, etc..., can easily be added. Please see the Resource tutorial for more info on this[13]. In that tutorial, the Resource config section is defined in the main config file, and the code for adding a custom resource type can be found in resource.cpp.

As a development feature, orx supports hotloading of modified resources at runtime. This is enabled by defining the "WatchList" list. Resource files defined here will be reloaded when modified on the filesystem, immediately updating them in the game. Additionally, if "Config" watching is enabled, any shaders, spawners, fonts, texts, and sound configuration sections will be updated immediately.

# Screenshot module

## Summary

```
[Screenshot]
BaseName  = <string>
Digits    = <int>
Directory = path/to/directory
Extension = <string>
```

## Details

Only one section (`Screenshot`) is defined for this module. Here's a list of its available properties:

- `BaseName`: Base name used for your screenshot. Its default value is "screenshot-".
- `Digits`: Number of digits used at the end of screenshot files. Its default value is 4.
- `Directory`: Directory where to store screenshots. By default, screenshots will be stored in the current active directory.
- `Extension`: Extension used for screenshot files. This also defines the type of encoding for the file. Available values [14] on Win/Linux/OS X are tga (default), png, bmp & dds. On iOS only png (default) & jpg are supported.

*NB: If using all default settings, screenshots will be stored in the current active directory [15] and named `screenshot-0001.tga`, `screenshot-0002.tga` and so on.*

# SoundSystem module

## Summary

```
[SoundSystem]
```

```
DimensionRatio = <float>
StreamBufferNumber = <int>
StreamBufferSize = <int>
```

## Details

Only one section (`SoundSystem`) is defined for this module. Here's a list of its available properties:

- `DimensionRatio`: Defines the ratio between orx's world (sizes are expressed in pixels) and the sound simulation world (sizes are expressed in meters). Its default value is 0.01 which means 1 pixel is represented by 0.01 meters. [16)]
- `StreamBufferNumber`: Number of buffers to use for sound streaming. Needs to be at least 2, defaults to 4.
- `StreamBufferSize`: Size of buffers to use for sound streaming. Needs to be a multiple of 4, defaults to 4096.

## Latest config settings for the Development Version

We endeavor to keep the config properties on this page up to date as often as possible. For up to the minute config information for the latest version of Orx, check the most recent published at:

CreationTemplate.ini and

SettingsTemplate.ini

Additionally these files can be found under your orx source tree in the `orx/code/bin` folder.

1)
OpenGL and GLSL shader versions:
https://www.opengl.org/wiki/Core_Language_%28GLSL%29#Version
2)
between 0.0 and 1.0
3)
keyboard key, mouse button, joystick button or joystick axis
4)
or referenced by an `orx`TEXT in config file
5)
whether it's a parameter internally registered by orx or one you registered in your own code
6)
the one named after the executable
7) , 16)
in other words, 100 pixels = 1 meters
8)
you need a positive Y component for your gravity vector for your objects to fall down
9)
which is also the main/core clock
10)
The location containers. For the file system resource type, those are folders.
11)
following a left to right priority order
12)

Except on Android where .apk archives are a separate resource type.
13)

Where support for ZIP files has been taken as an example.
14)

when used with orx's default display plugin based on GLFW
15)

usually the executable's one

From:
https://www.orx-project.org/wiki/ - **Orx Learning**

Permanent link:
**https://www.orx-project.org/wiki/en/orx/config/settings_main/main?rev=1692771867**

Last update: **2025/09/30 17:26 (14 months ago)**