Standalone Applications

Setup

Alrighty, time to set up the next part of our system.

Just for a quick reminder, we're creating a stand-alone project in “C:\MyProject\” for windows users (and this will be my default example) or “~/MyProject/” for Linux users.

Okay, lets get to it!

We're going to create a couple of new files first:

StandAlone.cpp and StandAlone.h

Code

Okay, Step two, our code:

Lets do some good old copy-paste for now, and afterward I'll explain what each piece does.

Firstly, We'll create the header for our “StandAlone” class:

StandAlone.h (This is a new file!)

```
#ifndef __STANDALONE_H__
#define __STANDALONE_H__
```

We can also use a simple “#pragma once” line instead, however the vagaries of my mind suggest that this is not supported “everywhere” and as I'm hoping to give you guys a nice cross-platform engine, I won't be using it.

```
#include "orx.h" // We want to interface with ORX, so including it is
helpful! :)

class StandAlone
{
public:
    static StandAlone* Instance();
```

Our “StandAlone” class is going to be a singleton, so instead of having a public constructor and deconstructor (StandAlone() and ~StandAlone()) we're instead only going to have an “Instance()” function.

```
    static orxSTATUS orxFASTCALL Init();
```

```
    static orxSTATUS orxFASTCALL Run();
    static void orxFASTCALL Exit();
```

Next are our static function definitions, Init, Run and Exit, I'll give some more details on these later.

```
protected:
    StandAlone();
    StandAlone(const StandAlone&);
    StandAlone& operator= (const StandAlone&);
```

Here we create our private constructor, and a few other lovely functions for handling our singleton class.

```
private:
    static StandAlone* m_Instance;
};

#endif // __STANDALONE_H__
```

And finally we finish up with our private instance variable, and the required #endif to close out our header.

Next on the menu, we cook up our class body. Stuff and bake some functions, and enjoy the tasty source... mmmm delicious!

StandAlone.cpp (This is a new file!)

```
#include "StandAlone.h"

StandAlone* StandAlone::m_Instance = NULL;
```

Okay to start us off, we need to include our header (but you all knew that already right? ) and nullify our singleton instance, this ensures we've got a nice clean pointer to play with later.

```
StandAlone* StandAlone::Instance ()
{

        // This is -NOT- a thread-safe function. I won't be covering thread
safety as it is outside the scope of this tutorial.
        //   (If this matters to you, you should be able to find resources
anyway, good luck! ^_^)

    if( m_Instance != NULL )
    {
        return m_Instance;
    }

    m_Instance = new StandAlone;
```

```
    return m_Instance;
}
```

Next, our instance function. This function is intended to make sure we only ever have one copy of our class, and never any more. It simply checks if we've got one already (m_instance) and if we do, it returns that, otherwise it creates a copy and puts it where we can access it. Neat hey! :)

```
StandAlone::StandAlone()
{
}
```

Our constructor, does nothing much so far... give it time  .

```
orxSTATUS orxFASTCALL StandAlone::Init()
{
    orxViewport_CreateFromConfig( "Viewport" );
    return orxSTATUS_SUCCESS;
}
```

Okay, The fun starts now! – Our Init() function is a lovely little 'setup' call, essentially this is where we can do all sorts of fun things like loading config files, manually (in code) creating scenes, or just (as in our case) load up the viewport from the default configuration file.

```
orxSTATUS orxFASTCALL StandAlone::Run()
{
    return orxSTATUS_SUCCESS;
}

void orxFASTCALL StandAlone::Exit()
{
    return;
}
```

These two functions, again don't do much yet, however 'Run' is a good place to add anything you want to run outside the default engine hierarchy... meaning anything you put in there will not conform to timing standards, won't run on the clocks, and will try to process as often as possible.

Exit is called, you guessed it, on exit! Do clean-up calls and the ilk in here. Make sure your computer memory is as pristine as it was when you arrived ;)

Righty-o... NEXT!

Main.cpp (This is a new file!)

```
#include "StandAlone.h"

int main( int argc, char** argv )
{
```

```
    orx_Execute( argc, argv, StandAlone::Init, StandAlone::Run,
StandAlone::Exit );

    return 0;
}
```

Huge file hey… orx_Execute is a nice little helper function that essentially sets up the orx internals. The “Init” function is called first, then each loop Run is called, followed by Exit when we're cleaning up afterwards. You are not required to use this function! And as it is fully implemented in the orx headers, you can take a look at exactly what it does, so you can implement your own version if you don't like it.

Config

This is a work in progress, so as yet, lacks details.

First file is the “initialisation file” – this should be named the same as your project's exe. In my case, this is “project.ini” and “project_d.ini” (for the debug version).

This file is used to set up the subsystems of the orx engine. Things like the Window size and settings, the sound options and the physics. At this point, I believe these -need- to be set up in this file, else they won't work at any later stage. [happy to be corrected!]

In our test case we do two things. First, we set up the game window to be 640×480, write the word “Project” on the window, and ask the game to load our second file, called “MainMenu.ini”. *This file should be created in the same directory as your project's INI file.*

!! IMPORTANT !! Remember you will need to name this file the same as your executable, “Project.exe” becomes “Project.ini” and our debug exe “Project_d.exe” becomes “Project_d.ini”. These files should be created in the same directory as your project's exe: “C:\MyProject\bin\Project.ini” will be loaded by “C:\MyProject\bin\Project.exe”.

Project.ini (This is a new file!)

```
;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

; This is the default initialisation configuration file.
; Display, Input, Locale, Physics and Sound go here.

;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

; [Display]
; Title              = <string>             ; Title for the main display.
Will be empty if none is provided.
; Font               = path/to/FontFile     ; Specifies a default font when
creating orxTEXT structures. If no value is specified, the arial font will
```

```
be used by default on most systems.
; Decoration          = <bool>                ; 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.

; FullScreen          = <bool>                ; Defines if the main display
will be full screen or windowed. This property is set to false by default.
; ScreenWidth         = <int>                 ; Resolution width for the main
display in pixels. Must be provided.
; ScreenHeight        = <int>                 ; Resolution height for the
main display in pixels. Must be provided.
; ScreenDepth         = <int>                 ; Resolution depth for the main
display in bits. Defaults to 32bit.
; Smoothing           = <bool>                ; Sets the default antialiasing
mode (ie. with or without antialiasing). Defaults to false (ie. no
antialiasing).
; VSync               = <bool>                ; Enables/disables the vertical
synchronization. Will use system defaults if none is provided. Best results
are usually achieved by enabling this property.

[Display] ;=====================================
Title                   = Project
Decoration              = true
FullScreen              = false
ScreenWidth             = 640
ScreenHeight            = 480
ScreenDepth             = 32
VSync                   = true
Smoothing               = false

;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

; [Input]
; DefaultThreshold = <float>               ; Sets the threshold value under
which joysticks' axis values are ignored. This property's default value is 0
(ie. any input, as small as it is, will be considered).
; SetList             = <list#list>           ; 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.

; [InputSet1]                                 ; Every input action can be
linked to up to 4 different physical inputs.
; <button>            = <list#list>           ; For every possible physical
input, one or more actions are linked. Every time this physical input gets
activated, the corresponding input actions will be triggered.
; <key>               = <list#list>
; <axis>              = <list#list>

; CombineList         = <list#list>           ; 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.

;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

; [Locale]
; LanguageList       = <list#list>           ; 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.

; [Language1]
; MyTextEntry        = <string>

;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

; [Mouse]
; ShowCursor         = <bool>                ; Tells if the mouse cursor
should be displayed or not. This property is set to true by default.

;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

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

; Please note that these lines are ignored if you run your executable using
command line parameters (-c / --config and -p / --plugin switches).

;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

; [Physics]                                 ; Box2D is ORX's default
physics plugin.
; AllowSleep         = <bool>                ; 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     = <float>               ; 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.
; Gravity            = <vector>              ; Defines the gravity vector
used in the physics simulation. Please remember that orx 2D vertical axis Y
ir oriented toward the bottom of your screen (plus = down). This value has
to be provided.
; IterationsPerStep  = <int>                 ; 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.
; WorldLowerBound    = <vector>              ; Defines the lower boundary
for the physics simulation world (in pixels). This value has to be provided
when using physics. The Z component will be ignored with 2D plugins such as
Box2D.
; WorldUpperBound    = <vector>              ; Defines the upper boundary
for the physics simulation world (in pixels). This value has to be provided
when using physics. The Z component will be ignored with 2D plugins such as
Box2D.

;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

; [Render]
; MinFrequency       = <float>               ; This defines the minimal
frequency for the rendering clock. 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.
; ShowFPS            = <bool>                ; This property tells orx to
display the current FPS in the top left corner of the main display or not.
Its default value is false.

;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

; [Screenshot]
; BaseName           = <string>              ; Base name used for your
screenshot. Its default value is “screenshot-”.
; Digits             = <int>                 ; Number of digits used at the
end of screenshot files. Its default value is 4.
; Directory          = path/to/directory     ; Directory where to store
screenshots. By default, screenshots will be stored in the current active
directory.
; Extension          = <string>              ; Extension used for screenshot
files. This also defines the type of encoding for the file. Available values
when used with orx's default display plugin based on SFML are bmp, tga, dds,
png and jpg. Its default value is tga.

;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

; [SoundSystem]
; DimensionRatio     = <float>               ; 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.

;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

; @<string>@                               ; Filename of included
```

```
configuration file.
@MainMenu.ini@

;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

MainMenu.ini (This is a new file!)

This next file is essentially a 'scene' for our game. This is where we create a bunch of objects, set up our game camera, and a whole lot of other lovely stuff!

In our case, we create a camera (the object in the scene which is our 'eyes'), a viewport (this is the place our camera will draw to) and even set the background colour of our scene.

```
;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

; This is the Main Menu configuration file.
; Viewport, Camera and other bits and bobs go here.

;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

; [ViewportTemplate]
; BackgroundClear    = <bool>                ; Specifies if the background
should be cleared before rendering it. Its default value is true.
; BackgroundColor    = <vector>              ; Defines which color will be
used for clearing the viewport before rendering it. Its default value is
black (0, 0, 0).
; Camera             = CameraTemplate        ; Template name of the camera
that will be linked to this viewport. Each camera template will correspond
to a unique camera at runtime. This means that if you use more than one
viewport linked to the same camera, they will render the same content as
seen by this camera.
; RelativePosition   = left|right|top|bottom ; Defines where the viewport
will be placed in the main display. It should be a combination of two
attributes. Ex.: 'top left' to have your viewport in the top left corner.
Its default value is 'top left'.
; Position           = <vector>              ; Defines an absolute position
for the viewport in the main display, in pixel coordinates. This value is
only used if none is provided for RelativePosition.
; Size       = <vector>              ; Defines the viewport size relatively
to the main display's one, ie. (1, 1, 0) means that it will cover the full
display. Its default value is (1, 1, 0). The Z coordinate is ignored.
; Size               = <vector>              ; Defines the absolute viewport
size, in pixels. This value is only used if none is provided for Size.
; ShaderList         = <list#list>           ; Defines a list of shaders
that will be executed every time this viewport is rendered. Up to 4 shaders
can be specified. By default, no shader is used.
; Texture            = path/to/TextureFile   ; Defines a texture where the
```

```
viewport will be rendered. Its default value is the main display (ie.
screen). NB: orx's default display plugin based on SFML doesn't support this
property.

[Viewport] ;==================================
Camera                = Camera
BackgroundColor       = (155, 0, 55)

;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

; [CameraTemplate]
; FrustumHeight       = <float>               ; As orx's cameras are 2D ones,
their frustum are rectangle cuboids instead of real frustums.
; FrustumWidth        = <float>               ;  - If you want to achieve a
1:1 aspect ratio with your main display window, you can use the
Display.ScreenHeight and Display.ScreenWidth values.
; FrustumNear         = <float>               ; Defines the near plane for
the camera frustum. The near plane is excluded when doing render culling.
; FrustumFar          = <float>               ; Defines the far plane for the
camera frustum. The far plane is included when doing render culling.
; Position            = <vector>              ; Camera's initial position.
; Rotation            = <float>               ; Camera's initial rotation
(along its Z-axis).
; Zoom                = <float>               ; Camera's initial zoom.


[Camera] ;====================================
FrustumWidth          = @Display.ScreenWidth
FrustumHeight         = @Display.ScreenHeight
FrustumFar            = 2.0                   ; Frustum Near and Far are
based upon the position of the camera. 0.0 minimum to +infinite.
FrustumNear           = 0.0                   ; You cannot set this value
'behind' the camera.
Position              = (0.0, 0.0, -1.0)

;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

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

Permanent link:
https://www.orx-project.org/wiki/en/tutorials/standalone?rev=1613108078

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