An Introduction to Scroll

What is Scroll?

Scroll is a small set of C++ files that allow you to use the object-oriented features of C++ when using Orx.

Features

- Scroll supports “object binding” to custom classes. This means when a Scroll Object is created, it can automatically be set to use a C++ of choice for its implementation. This makes it easy to implement behavior that is specific to certain object types.
- Additionally, binding object types to classes makes it easy to handle events on an object-specific basis. For example, each type of game object can have its own OnCreate function which is called whenever an object of that type is created.
- Scroll supports saving and loading of .map files for easier Scene management. Scroll includes an embedded Map Editor called ScrollEd.

Before You Begin

This tutorial assumes proficiency with Orx. In particular, you should be comfortable creating a new Orx project as a standalone application.

You should also be familiar with the concepts of inheritance and polymorphism and how to use them in C++.

Trouble?

If you encounter trouble at runtime, check the console log output or debug assertions for error messages. Make sure your Orx config .ini file is loading at runtime and is complete. At a minimum, the config file must define the Display, MainViewport, and MainCamera sections.

If you have any trouble following this tutorial, just ask in the “Help Request” section of the Orx forums. http://orx-project.org/forum

Getting the Sources

Getting started with Scroll is easy. iarwain maintains the latest version of Scroll in a Bitbucket repository.

If you use the Mercurial version control system, you can pull the latest version of Scroll with this

command:

```
hg clone ssh://hg@bitbucket.org/orx/scroll
```

To avoid some ssh problems, you can also use this command:

```
hg clone https://bitbucket.org/orx/scroll
```

Otherwise, you can download the source code directly from Bitbucket here: https://bitbucket.org/orx/scroll/src. There is a “get source” button on that page.

For GIT users:

```
git clone https://github.com/orx/scroll.git
```

Creating the Project

First, create a new project linking to the SVN Embedded Dynamic version of Orx.

Then, extract the Scroll files to their own directory and add that directory to the include path of the Orx project.

About the Scroll class

The Scroll class is the main class in a Scroll project. It is contained in Scroll.h/Scroll.inl. This class is where the game is initialized and the “main loop” runs. You don’t use the Scroll class directly, though. Instead, you create your own class deriving from it.

Deriving from the Scroll class

To derive from the Scroll class, first create the interface for the derived class. Add a new header file to the project called OrxScroll.h. Add the following code to it:

```
#ifndef __ORXSCROLL_H_
#define __ORXSCROLL_H_

//! Includes
// The following define skips compilation of ScrollEd (map editor) for now
#define __NO_SCROLLED__
```

```
#include "Scroll.h"

//! OrxScroll class
class OrxScroll : public Scroll<OrxScroll>
{
public:

private:
    //! Initialize the program
    virtual orxSTATUS Init ();
    //! Callback called every frame
    virtual orxSTATUS Run ();
    //! Exit the program
    virtual void      Exit ();
};

#endif // __ORXSCROLL_H_
```

This class interface should look familiar to anyone who has created a standalone application in Orx. In particular, note the existence of the same Init, Run, and Exit functions.

Now, let's create the implementation. Add a new file called OrxScroll.cpp to your project and add the following code:

```
//! Includes
// The following define/undef is done only once in the project. It should be
// done before including the interface of the class deriving from
// Scroll (as follows).
#define __SCROLL_IMPL__
#include "OrxScroll.h"
#undef __SCROLL_IMPL__

orxSTATUS OrxScroll::Init ()
{
    orxSTATUS result = orxSTATUS_SUCCESS;

    return result;
}

orxSTATUS OrxScroll::Run ()
{
    orxSTATUS result = orxSTATUS_SUCCESS;

    return result;
}

void OrxScroll::Exit ()
{
}
```

```
int main (int argc, char **argv)
{
  // Executes game
  OrxScroll::GetInstance ().Execute (argc, argv);

  // Done!
  return EXIT_SUCCESS;
}
```

Run It

You can now build and run your project. The Init, Run, and Exit functions will work in the normal way and you can apply all your knowledge of Orx in this class.

What's the Big Deal?

This is just the same class as the standalone tutorial, you might be thinking. So far, you're correct. What we've created in this tutorial is the bare minimum running Scroll project. The advantages of Scroll come in all the other functions available in the Scroll classes. Look into the header files to see what is available in the Scroll code; these features will be covered in future tutorials.

The next tutorial will introduce object creation (the Scroll way) and object binding.