Quick Start Guide

CONFIDENTIAL

03/11/2005
Nintendo Co., Ltd.

0. Introduction

This document explains how to use the `NITRO-SDK` development source tree snapshot (This package fixes files at a point in time. This package is used because changes occur frequently during development).

The explanation occurs in the following order:

1. Preparing Development Tools
2. Expanding the `NITRO-SDK` Package
3. Setting Environment Variables
4. Building the `NITRO-SDK` Tree
5. Trying the Samples
6. Writing a Simple Program
7. Using Build Switch

1. Preparing Development Tools

The build for the current `NITRO-SDK` has been confirmed on the following Windows environment.

- Microsoft Windows 2000 Professional

The following tools are necessary to build (compile, etc.) the Nitro SDK.

- CodeWarrior for NITRO
- Cygwin or MinGW(MSYS Tools)

In addition, one of the following tools must be used for debugging.

- NITRO Simulator `ensata`
- IS-NITRO-EMULATOR

Generally, these tools will be distributed or sold with this package. However, if this is not the case, please contact the distributors.

See `CygwinPackageList.rtf` in the `NitroSDK/docs/SDKTools` directory after unzipping the SDK to learn about installation of `Cygwin`.

See the documentation for the tools to learn about installation procedures.

2. Expanding the NITRO-SDK Package

Expand the `NITRO-SDK` package onto a disk. The package is compressed in `WinZip` format, so use the appropriate tools to decompress the files or expand the files using the `unzip` command in Cygwin.

Use the following procedure to expand files using the `tar` command.

```
% cd <directory to expand>
% unzip e:/NITRO-SDK/NitroSDK-2.0-XXXXXX.zip
```

A directory named `NitroSDK` is created ("`%`" is the prompt).

3. Setting Environment Variables

Set the absolute path of the expanded `NITRO-SDK` directory as the environment variable `NITROSDK_ROOT`. If nothing is set, the default value is `C:\NitroSDK`. This directory is referred to below as `$NitroSDK`.

4. Building the NITRO-SDK Tree

Start `bash` with Cygwin, and make the current directory `$NitroSDK`. In `$NitroSDK`, enter the following to start the build.

```
% make
```

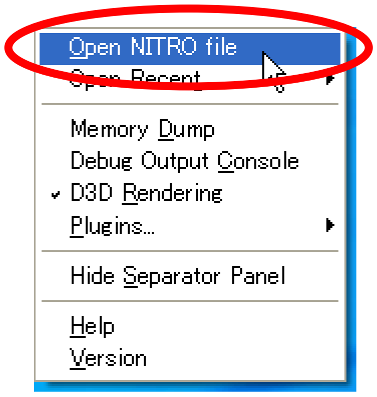
If an error occurs during the process, it is possible that a mistake was made in the settings or that a bug may exist in the package. Review the settings before contacting the distributor.

5. Trying the Samples

Run the samples to verify whether the build has completed successfully. We assume that you have `ensata` for this example.

1) Start `ensata`.
   `ensata` contains both `ensata_dx.exe` that uses DirectX for the rendering process and `ensata.exe` that does not use DirectX for the rendering process. Chose the file that is appropriate for your environment. Both of these files use DirectInput8 (`DINPUT8.DLL`) to support controller input. If your PC does not have DirectInput8 installed, you can obtain it from the Microsoft website.

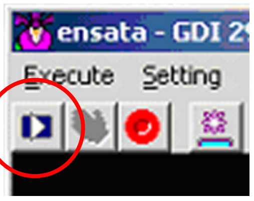
2) Right click on the window and select [Open NITRO file] (indicated with the red circle below).



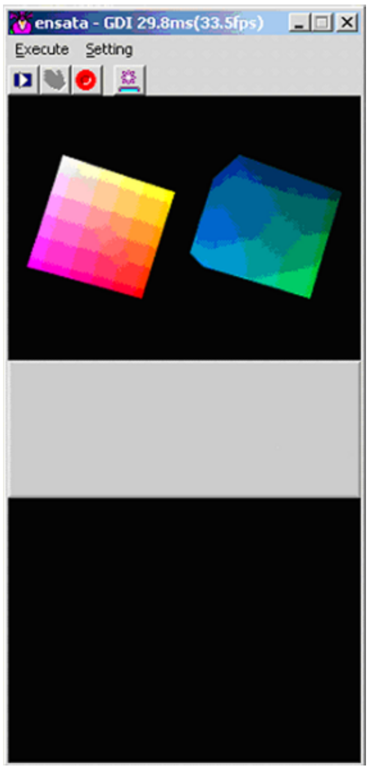
3) Specify the `.srl` file in the dialog box.
   For this sample, specify the following file that was created by the previous build.

```
$NitroSDK/build/demos/gx/UnitTours/3D_Pol_LightColor/bin/ARM9-TEG/Release/main.srl
```

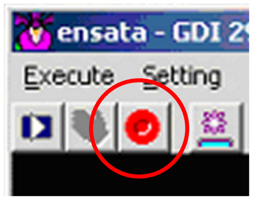
4) Press the ensata window execution button [indicated with the red circle in the figure below].



5) The following two cubes should be displayed in the screen window. The color of the cube on the right changes constantly.



6) Press the STOP button [indicated with a red circle in the figure below] to stop the `.bin` file emulation.



6. Writing a Simple Program

The following describes how to write a simple program.

1) Make an appropriate work directory and copy the following files into it.

```
$NitroSDK/build/demos/_template/src/main.c
$NitroSDK/build/demos/_template/makefile
```

2) Make the following changes to `main.c`.

Lines numbered 16 and later

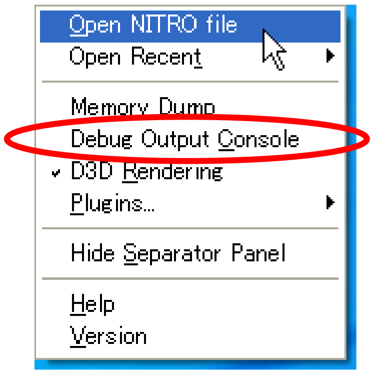
```
#include <nitro.h>

void NitroMain(void)
{
    OS_Init( );
    OS_Printf( "Hello World of NITROid. \n" );          ← Add this line
    while (1){}
}
```

3) Execute the `make` command. The following will be output (slight differences in display may occur due to revisions in the SDK). If an error or warning is displayed, please review step 2 to check for errors.

```
% make
==== /c/NitroSDK/xxx

C:/Program\ Files/Metrowerks/CodeWarrior\ for\ NITRO\ V0.3/ARM_Tools/Command_Line
_Tools/mwccarm.exe -lang c -proc arm946e -nothumb -nopic -nopid -interworking -O
4 -opt speed -inline on,noauto -g -msgstyle std -w all -enc SJIS -char unsigned
-stdinc -enum int -stdkeywords off -avoid_strb all,err  -DSDK_CWBUG_INLINE -DSDK
_TEG -DSDK_4M -DSDK_ARM9 -DSDK_CW -DSDK_RELEASE -DSDK_CODE_ARM -gccinc -I./inclu
de -I./src -Id:/dev/NitroSDK/include -cwd proj -c main.c -o obj/ARM9-TEG/Release/
main.o

C:/Program\ Files/Metrowerks/CodeWarrior\ for\ NITRO\ V0.3/ARM_Tools/Command_Line
_Tools/mwldarm.exe -proc arm946e -nothumb -nopic -nopid -interworking -g -msgsty
le std -w on -stdlib -map closure -main _start -L./lib/ARM9-TEG/Release -Ld:/dev
/NitroSDK/lib/ARM9-TEG/Release ./obj/ARM9-TEG/Release/main.o -llibfx.a -llibgx.a
-llibos.a -llibirissyscall.a -lcrt0.o d:/dev/NitroSDK/include/Nitro/ARM9-TEG.lcf -o
bin/ARM9-TEG/Release/main.nef

C:/Program\ Files/Metrowerks/CodeWarrior\ for\ NITRO\ V0.3/ARM_Tools/Command_Line
_Tools/elftobin.exe bin/ARM9-TEG/Release/main.nef d:/dev/NitroSDK/tools/elftobin/
spIdle.elf d:/dev/NitroSDK/tools/elftobin/romHeader.bin  -o bin/ARM9-TEG/Release/
main.bin%
```

4) Start Ensata. Select [Open NITRO file] from the right click menu. Specify `bin/ARM9-TEG/Release/main.srl` in the work directory.

5) Select [Debug Output Console] (indicated with the red circle in the figure below) from the right click menu.



6) When the execution button is pressed, the debug output window displays the following.

```
Hello World of NITROid.
```

7) Press the STOP button to stop the `.bin` file emulation.

7. Using Build Switch

SDK has several build settings. The IS-NITRO-EMULATOR (or TS board) Release-version library is linked by default, but the Debug version build can occur by changing the settings of the macro at build. This procedure is called a Build Switch.

See `$NitroSDK/docs/SDKRules/Rules-Defines.html` for details about a Build Switch.