BuildNENR

Manual

Version 1.0.0

NTR-06-0313-001-A1

Released: November 15, 2005

Table of Contents

Revision History

Version	Revision Date	Description
1.0.0	01/31/2005	Initial version. (Transferred contents of `BuildNENR_SimpleManual.txt`.)

1 Introduction

This document explains how to use `BuildNENR.exe`. `BuildNENR.exe` is a Windows application that creates `NENR` files (G2D entity runtime binaries) based on definition text files.

2 Use

```
BuildNENR.exe [filename ] [-o/...]
```

`[filename]`	Required	Entity definition text file name. Specify the path + file name + extension to be converted.
`[-o/]`	Optional	Specify the data output directory. Include the path name after the `-o/` without a space. Be aware that there will be no output if the specified directory does not exist.

Example: `>BuildNENR.exe c:/data/text.txt -o/d:/data`

3 Definition Text File Format

`Entity` definition block

`<[type],[labelName],` Sequence Number, Sequence Number, `...` , Sequence Number `>`

`[type]`:	Character that specifies the `Entity` type. `C` indicates Cell, and `M` indicates Multicell.
`[labelName]`:	Label name (required). Avoid using characters that cannot be used in C variables. (Note: Be aware of the difference between Sequence Numbers and Cell Numbers.)

Items are stored in the `Entity Bank` in the order they are written in the definition block.

Examples:

```
< C, ent1, 0, 1, 3, 4 >

< C, ent2, 2 >

< C, ent3, 5, 6, 7, 8 >

< M, ent4, 5, 6, 7, 8 >
```

Data that is defined by the four Entities is created.

4 Example of Entity Application

An entity is defined as a concept to manage and store information close to the upper layer (user program) above data structures such as cell animation and multicell animation.

This entity should be used as information to define the basic portions of game characters. For further details, refer to the Entity Overview in the API reference.

Benefits of using the entities are listed below.

- Even if multiple game character animations are included in a single NANR (animation file), they can be selected and used separately.
- Sequences in an NANR (animation file) can be reordered and used.
- Editing (changing and revising) the definition file is easy because it is in text format.

The following is an example of applying entity information.

Assume that a NANR file has multiple game character animations stored.

Table 4-1 Sample Animation File

Sequence Number	Meaning of Animation
0	Enemy A: Wait
1	Enemy A: Move
2	Enemy A: Attack
3	Enemy B: Wait
4	Enemy B: Move
5	Enemy B: Attack
6	Enemy B: Attack (Separate pattern)

Implement a game engine where, as a rule for the game project, entity animation 0 is determined to be a wait animation, 1 is a move animation, 2 is an attack animation, etc.

Define the entity information as follows.

- Entity A references 0, 1, and 2.
- Entity B references 3, 4, and 5.

The actual definition script is as follows.

```
< C, A, 0, 1, 2 >
< C, B, 3, 4, 5 >
```

You can change the attack pattern for Entity B simply by changing the Entity B definition file to 3, 4, 6 without changing the game engine itself. The definition script file after the changes is shown below.

```
< C, A, 0, 1, 2 >
< C, B, 3, 4, 6 >
```

Since it is possible for the definition file to be automatically generated from a script, more flexibility can be anticipated than by actually replacing and modifying data on NITRO-CHARACTER.