Allegorithmic - Substance Engine - Readme Cooker

Copyright

Version

Alpha package
Document version 2.0 (10/21/08)

PC Binaries compiled with: Microsoft VC2005 SP1.

Introduction

This document contains specific informations about the Substance standalone cooker. This tool allows to compile **.sbs** packages authored by the Substance Editor into **.sbsbin** files that can be parsed by the Substance Engine. Moreover, it generates information about the *outputs* and the *inputs* of the compiled file (both in XML and C/C++ readable format).

The cooker binary is: **./bin/win32/release/sbscooker.exe**.

Software requirements

Microsoft Windows 2000/XP/Vista.

Hardware requirements

None.

Command line usage

Generic usage:

```
sbscooker [<options>..] <list of input substance files to cook>
```

Allowed options:

```
--help              Display help message.

--input <arg>       Input editor file(s) (authoring files .sbs). Implicit if last
                    position in the command line.

--output <arg>      Set the default path and basename of the cooked files. Default:
                    the first input file path and basename.

--enginebin <arg>   Name of the cooked Substance engine file (binary format .sbsbin).
                    Default: <output>.sbsbin

--iodescxml <arg>   Name of the cooked I/O description file (XML format .xml).
                    Default: <output>.xml

--iodescc <arg>     Name of the cooked I/O description file (C/C++ format .h).
                    Default: <output>.h

--bigend            Big endian cooked output (default little-endian). The '.bigend'
                    extension is appended to the cooked Substance engine default
                    filename.

--include <arg>     Add a item in the list of folders of default packages. The
                    Substance authoring tool default packages folder is automatically
                    appended to this list.
```

The Substance compiler tool can also be launched by dragging and dropping multiple texture files on the executable icon in an Explorer window.

The output cooked file names are by default `<1st input file path and basename>.<extension>`.

Generated files

The cooker tool generates three different files: the compiled textures file for Substance engine, and two inputs/outputs description files in XML and C/C++ readable format than can used by the client application (i.e. game engine).

.sbsbin result file

The resulting **.sbsbin** file contains the process to generate all the *outputs* of all the *graphs* contained in all the files given as an input to the cooker. This file can be processed by the Substance engines.

Textures outputs identifiers are not included in the **.sbsbin** file, i.e. the engine results and editor authored outputs can't be easily matched without using one of the input/output description file.

I/O description .xml (XML format) file

This file contains for each input and output (result texture) of the **.sbsbin** files the authoring informations that allows to identify it.

Output informations:

- *index*: output index (unsigned integer) in the **.sbsbin** file.
- *uid*: output unique identifier (32bits unsigned integer) in the .sbsbin file. This number identifies completely the output created in the editor (the uid does not vary for a given output).
- *pkgurl*: Substance package URL (string) of the graph that contains this output.
- *identifier*: identifier (string) of the output in the graph.
- *usertag*: user tag (string) of the output. This tag can be used to add extra information for the purpose of the client application (e.g. target image file path and format, streaming info., etc.).
- *usage*: usage (enumeration) of the output (e.g. *diffuse*, *normal*, etc.).

Input informations:

- *TODO*

I/O description .h (C/C++ readable format) file

This file is similar to the **.xml** (XML format) file: it contains the same information. However, a C/C++ readable format is used. It allows the client application to include the input/output description directly into its source code.

Output and input informations are stored in two arrays of structures. Each array is null-terminated.

Miscellaneous

- The Cooker will be available as an API in a future release of the package. This will allow to integrate the Substance cooking process in a broader cooking process.