

CODE::BLOCKS TEMPLATES TO CREATE DELED PLUGINS USING VISUAL C++ TOOLKIT 2003

written by Salvatore Russo "Il Buzzo"

Well, guys after a lot of work I've finally released Code::Blocks templates to make DeleD plugins creation easier for Visual C++ Toolkit 2003 users.

REQUIREMENTS:

To use this templates you need a computer with windows, Code::Blocks (I've tested version 1.0 rc1 and rc2) and Visual C++ Toolkit 2003.

You also need Microsoft Platform SDK because of some windows api function used (MessageBox() in import plugin that can be disabled by defining NO_MESSAGES, and MessageBox() and OpenFileName() in export plugin) , they are not necessary for general DeleD development but are usefull (rapid dialogs interfaces and messages, present in every windows OS computer etc....).

INSTALLATION:

To install these templates, just copy everything contained in templates directory in this package in Code::Blocks templates directory and the

game is done.

Then start Code::Blocks, create a new project and you'll see DeleD Import and Export Plugins.

SPECIAL INSTRUCTIONS:

Just to show you the differences between a plugins that uses windows api and one that doesn't I've introduced in Import Plugin a define (NO_MESSAGES) so if you define that in compiler options by writing NO_MESSAGES=1 plugin will be compiled without using windows api.

THINGS TO KNOW:

You can call your plugins as you like but you must remember that if you change dll name (i.e. Mynewplugin.dll) you then must change that name in ExportTable.def first line too, this way:

LIBRARY Mynewplugin.dll

...

INFO:

To obtain help on this package please mail me at info@ilbuzzo.net, I will help within my possibilities.

HOW DELED PLUGIN SYSTEM WORKS (EXTRACT

FROM MY PREVIOUS TUTORIAL):

As you've certainly observed DeleD plugin system has changed a lot since its first implementation, so now it's a more flexible system but different. First of all we must notice that now everything is done with callback, not only setting data in DeleD but also getting them from it. As result the way of passing data had been unified and all data (set or get) must have the following form:

length_of_header;number_of_datablocks;version;metadata_of_block1;.....;metadata_of_blockN;datablock1;...;datablockN;

where length of header rapresents the length in characters of the red and the green part of string (including the length itself), and the black part rapresents the actual data.

Version used will be 091 from now on, so let's give a look to metadata formation, we'll see that each metadata block has the following form:

data type,number of items,offset after header,size of datablock

where data type is an unsigned value (1 for all objects, 2 for all lights, 3 for materials etc....

look at the code for more info), then we have the number of items to pass, the offset after header and finally total length of datablock.

For what concern datablocks they're made by a single string with all item informations separated by ; so, this is the string formation, in example a pointlight string is:

**18;1;091;3,1,0,68;1;1;0;0;1000;1;304;300;0;-
1;255;255;255;255;255;255;100;pointlight1;**

and supposing we're passing this data to DeleD wi must set callback reason to SETDATA (look at the code for further information).

THANKS:

Lot of thanks to Jeroen and Paul-Jan for DeleD and everyone that is supporting my work.

CHANGELOG:

-version 1.2:

First Public release.