

A SHORT TUTORIAL ON WRITING DELED 0.91+ PLUGINS IN C++

by Salvatore Russo

DeleD 0.91+ Plugin System.

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

Version used will be 090 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;090;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 we must set callback reason to SETDATA (look at the code for further information).

After this small explanation let's speak of creating DeleD plugins with C++ and in particular MinGW C++ compiler with Dev-Cpp 4.9.9.2 or superior.

How you can write simple plugins for DeleD 0.91+ with DevCpp 4.9.9.2 and MinGW (Tested and working 3.2.2, 3.3.1, 3.4.2).

If you're reading this file, you must have the complete archive so you need just to copy all files in the TEMPLATES dir in Dev-Cpp templates dir and the game is done.

This way when you open a new project in the multimedia tab you'll find a template for a sample plugin exporter and importer.

The importer sets a simple cube into DeleD and shows it's main install dir, while the exporter just saves selected objects in a DMF or TXT file.

At this point you need just to compile this examples and copy the resulting DLL to DeleD\Plugins to test the result.

I suggest you to strip the DLL after compilation to reduce its size, if you're using MinGW 3.2.2 you can compress it more with UPX (Ultimate Packer for Exes), if not using MinGW 3.2.2 (if you're using 3.3.1 or 3.4.2 instead) **do not use UPX compression** cause the resulting DLL is not recognised by DeleD anymore.

Build plugins with other C++ compilers.

I've written the code in plain C++ using STL, so maybe you don't need a lot of changes, project files are contained in Import and Export directory where the main cpp code is **dllmain.cpp** and you need to add **dll.h** and **support.h** as include files.

I've not tested another C++ compiler yet, so please try and if you have some questions please send me an e-mail at info@ilbuzzo.net.

Code.

Owing to the fact that the code is fully commented I'll add it here so you can give it a look to have more specifications and explanations.

Dll.h

This is the main include files that defines callback data structure and function, as well as plugin reasons, header items and errors:

```
//Header of dllmain.cpp see it for details

#ifdef _DLL_H_
#define _DLL_H_

//this definition is very important cause each exported function
is called by DeleD
//with a stdcall and you must use extern "C" cause each function
is called by its name
//and the simplest way to export non decorated names is to use
this definition
#define DLLEXPORT extern "C" __stdcall __declspec (dllexport)

// plugin reasons have replaced plugin types
#define PR_GETINSTALLDIR 0 //used to get DeleD installation
directory
#define PR_GETMEM 1 //used to know how long is the string I want
to obtain from DeleD,
//or better how much memory I need to allocate
for getting data
#define PR_GETDATA 2 //used to retrieve data from DeleD
#define PR_SETDATA 3 //used to pass data to DeleD
```

```

    // header items are passed to know what kind of data you want
    to use
#define HI_SCENEPROPERTIES  0 //used for setting or getting scene
properties
#define HI_OBJECTS  1 //used for setting or getting all objects
#define HI_LIGHTS  2 //used for setting or getting all lights
#define HI_MATERIALS  3 //used for setting or getting materials

#define HI_SELECTEDOBJECTS  10 //used for setting or getting only
selected objects
#define HI_SELECTEDLIGHTS  11 //used for setting or getting only
selected lights
#define HI_SELECTEDMATERIALS  12 //used for setting or getting
only selected materials

    // plugin errors returned by callback function
#define PE_NOERROR  0 // no errors in callback
#define PE_NODATA  1 // no data to set or get
#define PE_INVALIDVERSION  2 // invalid version
#define PE_INVALIDMEM  3 // invalid memory size
#define PE_INVALIDGET  4 // invalid get operation
#define PE_INVALIDSET  5 // invalid set operation
#define PE_INVALIDHEADER  6 // invalid header
#define PE_INVALIDDATASTRING  7 // invalid data string
#define PE_INVALIDDATAPROPERTY  8 // invalid data property
#define PE_OBJECTDATA  11 // invalid object data
#define PE_LIGHTDATA  12 // invalid light data
#define PE_MATERIALDATA  13 // invalid material data
#define PE_UNKNOWNREASON  98 // callback reason unknown
#define PE_UNKNOWNERROR  99 // unknown error

struct TCbData{          // callback data passed back to application
(DeleD)
    unsigned Reason; // reason for callback
    char *Data; // data to get or set
    unsigned DataSize; // size of Data
};

    //this a standard definition for callback routine
typedef TCbData * PCbData ; // this is a pointer to callback data
typedef unsigned __stdcall (*TCallback)(PCbData); // callback
function prototype

                                                                    // passes a pointer to
TCbData

    //pointer to TCallback
TCallback *PCallback;

#endif /* __DLL_H__ */

```

Support.h

This header includes some function facilities, and it also include a String and a StringList class definition:

```

//This support library has been made by Salvatore Russo and is
released under GNU public license.
//It uses standard template libraries to create String class and
StringList class used in DeleD
//plugins made by me.
//It also includes some facilities to add or retrieve header, or
convert a long, integer or unsigned to a string,
//and get materials as a stringlist or handle callback errors.

#ifdef _SUPPORT_H_
#define _SUPPORT_H_

    //standard classes to use
#include <cstdio>
#include <string>
#include <deque>

    //so you know error types
#include "dll.h"

typedef std::string stdstring;

    //Define a class String based on STL string
class String : public stdstring {
public:
    //constructors
    String() : stdstring() {};
    String(const char* str) : stdstring(str) {};
    String(const stdstring& str) : stdstring(str) {};
    //trim function in my implementation removes spaces
but also \t and \n chars
    //from end and beginning of string only
    void Trim() {
        int k = (this->length())-1;
        while(k>=0) {
            if((*this)[k]==' ' || (*this)[k]=='\t' || (*this)[k]
=='\n') {
                this->erase(k,1); //trim
                k=k-1;
            }
            else k=-1;
        }
        k=0;
        while(k < this->length()) {
            if((*this)[k]==' ' || (*this)[k]=='\t' || (*this)[k]
=='\n') {
                this->erase(k,1); //trim
            }
            else k=this->length();
        }
    }

    //replace all instaces of "from" char with "to" char
    void Replace(char from, char to) {
        int k = (this->length())-1;
        while(k>=0) {

```

```

        if ((*this)[k]==from) (*this)[k]=to; //replace
        k=k-1;
    }
};

using std::deque;

//Define a class StringList based on STL deque and the defined
class String
class StringList : public deque <String> {
public:
    //Adds a String to StringList
    void Add(String str){push_back(str);}

    //Loads a stringlist from a file
    //note that each String added to StringList
    //is separed by a \n character and it's present
    //at the end of line.
    void LoadFromFile(const char* filename){
        FILE *nuovofile;
        nuovofile=fopen(filename,"r"); //opens new file
        int c;
        String str;
        c = fgetc(nuovofile);
        while(c!=EOF){
            if(c!='\n'){
                str.append((char*)&c);
            }
            else {
                str.append((char*)&c);
                str.Trim();
                Add(str);
                str.clear();
            };
            c = fgetc (nuovofile);
        }
    }
    //GetText returns a String containing all the text included
in
    //the StringList
    String GetText(){
        String str;
        for(int i=0;i<this->size();i++){
            str=str+(*this)[i];
        }
        return str;
    }
};

//convert long integer to a string
String toString(long x) {

```

```

    char buffer[256];
    sprintf(buffer, "%ld", x);
    return String(buffer);
}

String toString(int x) {
    char buffer[256];
    sprintf(buffer, "%d", x);
    return String(buffer);
}

String toString(unsigned x) {
    char buffer[256];
    sprintf(buffer, "%u", x);
    return String(buffer);
}

//This function subdivides a string in a list of strings
StringList SubdivideString(String str, String divider) {
    StringList strings; //returned StringList
    strings.clear(); //clear returned stringlist

    int c=0;
    int l=str.length();

    //process entire string
    while(c<l) {
        String resultstr;
        resultstr.clear();
        //read characters until divider is encountered
        while((str[c]!=divider[0]) && c<l) {
            resultstr=resultstr + str[c];
            c++;
        }
        //Remove spaces \t and \n from string in my
implementation...
        //pay attention or change it in dll.h if you don't want to
remove
        //a particular char.
        resultstr.Trim(); //trims string resultstr
        strings.Add(resultstr); //add trimmed string
        c++;
    }
    return strings;
}

// adds the length of a string in bytes to that string,
// including the length of the 'length' string itself!
String AddLengthToHeader(String header) {
    long len=header.length()+(toString(header.length())).length()
+1;
    String tempStr = toString(len)+String(";")+header;
    if (tempStr.length() > len) len++;
    return (toString(len)+String(";")+header);
}

```

```

    // this function reads the header length and then extract
    // the whole header from data...
StringList GetHeader(String data,String delimiter){
    int position = 0;
    String myStr;
    myStr.clear();
    while (data[position] != delimiter[0]){           // headerlength
is the first value of data
        myStr = myStr + data[position];
        position++;
    }
    int headerlength = atoi(myStr.c_str());
    myStr.clear();
    myStr = data.substr(0,headerlength);             // retrieve
complete header from data
    return SubdivideString(myStr,delimiter);         // return as
stringlist
}

    //this function show an error message when giving callback
error code
void ErrorMessage(unsigned error){
    String errore;
    errore.clear();
    errore.append("Error Code ");
    errore=errore+toString(error)+String(": ");
    switch(error){
        case PE_NODATA :
            errore.append("No Data present.");
            break;
        case PE_INVALIDVERSION :
            errore.append("Invalid Version.");
            break;
        case PE_INVALIDMEM :
            errore.append("Invalid GetMem Command.");
            break;
        case PE_INVALIDGET :
            errore.append("Invalid GetData Command.");
            break;
        case PE_INVALIDSET:
            errore.append("Invalid SetData Command.");
            break;
        case PE_INVALIDHEADER:
            errore.append("Invalid Header.");
            break;
        case PE_INVALIDDATASTRING:
            errore.append("Invalid Data String.");
            break;
        case PE_INVALIDDATAPROPERTY:
            errore.append("Invalid Data Property.");
            break;
        case PE_OBJECTDATA:
            errore.append("Error in Object Data.");

```

```

        break;
    case PE_LIGHTDATA:
        errore.append("Error in Light Data.");
        break;
    case PE_MATERIALDATA:
        errore.append("Error in Material Data.");
        break;
    case PE_UNKNOWNREASON:
        errore.append("Unknown Callback Reason.");
        break;
    case PE_UNKNOWNERROR:
        errore.append("Unknown Error.");
        break;
    default:
        errore.append("Unknown Error Code.");
        break;
}
MessageBox(NULL, errore.c_str(), "Error in
Callback", MB_ICONERROR);
}

//This function returns a string made of all data
//to write materials on a DMF file.
//it includes ";" and as first element the number of materials
String GetMaterials( String materials_data, int num_materials ){
    StringList materiali; //A stringlist with all materials data
    StringList temp_list=SubdivideString(materials_data, String
(";")); //data from DeleD
    String temp=toString(num_materials)+String(";");
    materiali.Add(temp); //Add number of materials
    temp.clear();
    int offs=0;
    //Add materials
    for( int i=0; i<num_materials; i++){
        int n_layers=atoi(temp_list[offs+4].c_str());
        String curr_mat;
        for(int k=offs; k<=offs+4+n_layers; k++)
curr_mat=curr_mat+temp_list[k]+String(";");
        materiali.Add(curr_mat);
        offs+=4+n_layers+1;
    }
    String ret; //return string
    for(int i=0; i<materiali.size(); i++) ret+=materiali[i]+String
("\n");
    return ret; //return materials
}

//This function returns a string made of all data
//to write objects on a DMF file.
//it includes ";" and as first element the number of objects
String GetObjects( String objects_data, int num_objects ){
    //we'll extract only data from DeleD string and copy them
into a StringList
    StringList Objects=SubdivideString(objects_data, String
(";"));

```

```

        StringList temp_objs;
        temp_objs.Add(toString(num_objects)+String(";"));
        int index=0;
        for(int obj=0;obj<num_objects;obj++){
            //(Object information:
name,ID,type,visible,snapmethod,groupid,autoUV,tag)
            temp_objs.Add( Objects[index]+String(";")+
                Objects[index+1]+String(";")+
                Objects[index+2]+String(";")+
                Objects[index+3]+String(";")+
                Objects[index+4]+String(";")+
                Objects[index+5]+String(";")+
                Objects[index+6]+String(";")+
                Objects[index+7]+String(";"));
            index=index+8;
            int num_vertices=atoi(Objects[index].c_str());//(Number of
vertices in this object)
            temp_objs.Add( Objects[index]+String(";") );
            index++;
            for(int vert=0;vert<num_vertices;vert++){
                //(Vertice information: X,Y,Z values)
                temp_objs.Add( Objects[index]+String(";")+
                    Objects[index+1]+String(";")+
                    Objects[index+2]+String(";"));
                index=index+3;
            }
            int num_polygons=atoi(Objects[index].c_str());
            //(Number of polygons in this object)
            temp_objs.Add( Objects[index]+String(";") );
            index++;
            for(int poly=0;poly<num_polygons;poly++){
                int num_items = (atoi(Objects[index].c_str())*3)+2;//
(Number of data to write for each poly)
                String temp_str;
                for(int item=0;item<num_items;item++){
                    if(item<num_items-1)
temp_str+=Objects[index+item]+String(";");
                    else temp_objs.Add( temp_str+Objects
[index+item]+String(";") );
                }
                temp_str.clear();
                index=index+num_items;
            }
        }
        String ret;//return string
        for(int i=0;i<temp_objs.size();i++) ret+=temp_objs[i]+String
        ("\n");
        return ret;
    }

    //This function returns a string made of all data
    //needed to import a basic cube in DeleD.
    String CreateDeleDCube(){
        String MyChars;
        //create a cube object to pass to DeleD

```

```

    MyChars.append("Cube0;0;1;-1;1;-1;-1;User info;");//Object
information: name,ID,type,visible,snapmethod,groupid,autoUV,tag
    MyChars.append("8;");//Number of vertices in this object
    MyChars.append("128;0;128;");//Vertice information: X,Y,Z
values
    MyChars.append("-128;0;128;");
    MyChars.append("-128;0;-128;");
    MyChars.append("128;0;-128;");
    MyChars.append("128;128;128;");
    MyChars.append("-128;128;128;");
    MyChars.append("-128;128;-128;");
    MyChars.append("128;128;-128;");
    MyChars.append("6;");//Number of polygons in this object
    MyChars.append("4;0;0;1;5;4;1.0000;0.0000;-1.0000;0.0000;-
1.0000;1.0000;1.0000;1.0000;");//Polygon information: # of
vertices,textureID,vertex index values,uv values (2 per vertice)
    MyChars.append("4;0;1;2;6;5;1.0000;0.0000;-1.0000;0.0000;-
1.0000;1.0000;1.0000;1.0000;");
    MyChars.append("4;0;2;3;7;6;-
1.0000;0.0000;1.0000;0.0000;1.0000;1.0000;-1.0000;1.0000;");
    MyChars.append("4;0;3;0;4;7;-
1.0000;0.0000;1.0000;0.0000;1.0000;1.0000;-1.0000;1.0000;");
    MyChars.append("4;0;3;2;1;0;1.0000;-1.0000;-1.0000;-1.0000;-
1.0000;1.0000;1.0000;1.0000;");
    MyChars.append("4;0;4;5;6;7;1.0000;1.0000;-1.0000;1.0000;-
1.0000;-1.0000;1.0000;-1.0000;");
    return MyChars;
}

#endif /* __SUPPORT_H__ */

```

dllmain.cpp of export plugin

This is the main code of the export plugin:

```

/*****
*****
* This is the main code of a DeleD export plugin template for
C++.
*
* Author: Salvatore Russo "Il Buzzo"
*
* Version: 1.2.0
*
* DeleD version: 1.1.0
*
* License: GNU General Public License.
*
* Made with: Dev-Cpp 4.9.9.2 and MinGW (Tested with GCC 3.3.1,
GCC 3.2.2
*
* and GCC 3.4.2)
*

```

```

* Link Libraries: libcomdlg32.a (it includes calling to
OpenDialog etc..) *
* Linker parameters: --no-export-all-symbols --add-stdcall-alias
*
* Additional Compiler Parameters: -DBUILDING_DLL=1
*
* End Notes: This is an adaptation to new plugin system, so there
are some *
*             changes from the old code, but as result I'm still
adding to *
*             to code try{}catch() blocks for a better error
handling. *
*             It's not so difficult so if you don't want to wait
for another *
*             version you can implement it without difficulty.
*
*****
*****/

//Main header
#include "dll.h"
//Necessary for DLL to work
#include <windows.h>

//support.h contains some classes used, defined by me
//like String and StringList and some defined for DeleD
//in Jeroen sample code.
//Look at that file for major details.
#include "support.h"

//TCallback variable used to receive a pointer by DeleD
TCallback callback;

/* EXPORTED FUNCTIONS NECESSARY TO DELED

PCHAR PluginName() : returns name of Plugin
PCHAR PluginDescription() : returns a small description of
Plugin
PCHAR PluginDeleDVersion() : returns version of DeleD
PCHAR PluginVersion() : returns version of Plugin
PCHAR PluginAuthor() : returns Author's name
PCHAR PluginEmail() : returns Author's e-mail
void PluginSetCallback(void*cp) : pass callback pointer from
DeleD to callback
void PluginExecute() : it executes plugin code

notes:
PCHAR is the same as char* so it's a null terminated c
string.
Each function is defined as DLLEXPORT for major details see
the DLLEXPORT definition in dll.h file.
As you can notice the function PluginType() is no longer
used.
*/

```

```

DLLEXPORT PCHAR PluginName() {
    return "DeleD Export Plugin Sample";
}

DLLEXPORT PCHAR PluginDescription() {
    return "This is just a sample to test main plugin system
functionality.";
}

DLLEXPORT PCHAR PluginDeleDVersion() {
    return "1.1";
}

DLLEXPORT PCHAR PluginVersion() {
    return "1.0";
}

DLLEXPORT PCHAR PluginAuthor() {
    return "Your Name";
}

DLLEXPORT PCHAR PluginEmail() {
    return "youremail@yourdomain";
}

DLLEXPORT void PluginSetCallback(void *cp) {
    callback=TCallBack(cp);
}

DLLEXPORT void PluginExecute() {
    //variables used
    TCbData cbData;//callback data

    //create a structure for Save Dialog (salva is the italian
word for save :-))
    OPENFILENAME salva;

    //create a buffer for file name and path I hope it's big
enough :-))
    char szFile[512];

    //initialize structures to zeros
    ZeroMemory(&salva, sizeof(OPENFILENAME));
    ZeroMemory(szFile, 512*sizeof(char));

    //initialization of structure salva
    salva.lStructSize = sizeof(OPENFILENAME);
    salva.hwndOwner = NULL;//Parent window
    salva.lpstrFile = szFile;//file name buffer
    salva.nMaxFile = 512;//buffer size
    salva.lpstrFilter = "DMF Files (*.dmf)\0*.dmf\0Text files
(*.txt)\0*.txt\0All files (*.*)\0*.*\0";//filter data
    salva.nFilterIndex = 1;//default index
    salva.lpstrFileTitle = NULL;
    salva.lpstrTitle = "Select a file to export data string from
DeleD";//Save dialog title

```

```

        salva.lpstrInitialDir = NULL;
        salva.Flags = OFN_OVERWRITEPROMPT; //ask to overwrite an
        existent file

        //First of all we'll try to get DeleD directory to know where
        textures
        //are located and then we'll create a string containing
        material data

        String DeleDir; //DeleD directory
        DeleDir.clear();

        //just a simple sample code to show DeleD install dir
        try{
            unsigned error; // errors returned by callback
            cbData.Reason = PR_GETINSTALLDIR; //reason of callback
            cbData.Data = new char[512]; //allocate new space for string
            error=callback(&cbData); //pass Data to callback retrieving
            eventual errors
            if(!error){ //if no errors detected then go on
                String myStr; //get directory
                DeleDir=String(cbData.Data);
                MessageBox(NULL,DeleDir.c_str(),"DeleD install
                dir:",MB_ICONINFORMATION); //show info
            }
            else throw error; //if errors generate an exception
            delete cbData.Data; //destroy previously allocated memory
        }
        catch( unsigned exception) {
            ErrorMessage(exception); //show error message (you can use
            this block to handle errors)
        }

        //Now we do the real job of retrieving data by asking DeleD
        how many
        //space I need to allocate to get selected objects in example
        if(GetSaveFileName(&salva)) {
            try{
                unsigned errors; // errors returned by callback
                String myStr;
                myStr.clear();
                if(MessageBox(NULL,"Do you want to export all objects in
                this scene?\nIf so click yes or no if you want just to export
                selection.", "Export All Objects?", MB_YESNO|MB_ICONINFORMATION)
                ==IDYES){
                    myStr = String("2;091;3,0,0,0;1,0,0,0;"); //number of
                    headers ;version;items wanted (materials), set to 0 number of
                    items,offset and length;items wanted (all objects), set to 0
                    number of items,offset and length;
                }
                else myStr = String("2;091;3,0,0,0;10,0,0,0;"); //number of
                headers;version;items wanted (materials), set to 0 number of
                items,offset and length;items wanted (selected objects), set to 0
                number of items,offset and length
            }
        }
    }
}

```

```

        myStr = AddLengthToHeader(myStr); //add header
        cbData.Reason = PR_GETMEM; //reason of callback is getting
data wanted size
        cbData.DataSize = 0; //we don't know actual data length
        cbData.Data = new char[(myStr.length()+1)]; //allocate new
memory
        sprintf(cbData.Data, "%s", myStr.c_str()); //copy String to
Data
        errors=callback(&cbData); //executes callback
        delete cbData.Data; //destroy previously allocated memory
        if(!errors) {
            if (cbData.DataSize > 0) { //there's something to get
                try{
                    unsigned errors1; // errors returned by callback
                    cbData.Reason = PR_GETDATA; //reason of callback
                    cbData.Data = new char[cbData.DataSize+1]; //
allocate necessary memory
                    sprintf(cbData.Data, "%s", myStr.c_str()); //copy
header to let DeleD know what we want
                    errors1=callback(&cbData); //executes callback
                    if(!errors1){
                        FILE *F1;
                        F1=fopen (salva.lpstrFile, "w"); //creates a
new file
                        myStr.clear();
                        myStr = String(cbData.Data); //string returned
by DeleD

                        //we start to write some informations
                        fprintf(F1, "DeleD Map File;\n"); // (File
identifier. You can use it to check if this is a DeleD Map File)
                        fprintf(F1, "Version 1.1;\n"); // (Version
information)
                        fprintf(F1, "no-name-yet;00FFFFFF;75;\n"); //
(Scene information: name, ambient RGB values, shadow intensity)

                        //we get the header as a stringlist
                        StringList head=GetHeader(myStr, String(";"));

                        //we divide header as to obtain a stringlist
with material numbers and offset
                        StringList head1=SubdivideString(head[3],
String(", "));

                        //offset, total length of data and number of
objects
                        int offs=atoi(head[0].c_str())+atoi(head1[2].
c_str())-1;
                        int tot_len=atoi(head1[3].c_str());
                        int n_materials=atoi(head1[1].c_str());

                        //Get materials informations from retrieved
data
                        String Materials=GetMaterials(myStr.substr
(offs, offs+tot_len), n_materials);

```

```

        //Let's print to file
        fprintf(F1,"%s",Materials.c_str());

        head1.clear();

        //we divide header as to obtain a stringlist
with objects numbers and offset
        head1=SubdivideString(head[4],String(", "));

        //offset,total length of data and number of
objects
        offs=atoi(head[0].c_str())+atoi(head1[2].c_str
())-1;

        tot_len=atoi(head1[3].c_str());
        int n_objects=atoi(head1[1].c_str());

        //Get Objects informations from retrieved Data
String Objects=GetObjects(myStr.substr
(offs,offs+tot_len),n_objects);
        //Let's print to file
        fprintf(F1,"%s",Objects.c_str());

        fprintf(F1,"0;\n");//no lights exported
        fclose(F1);
        MessageBox(NULL,salva.lpstrFile,"DeleD
selected objects have been saved to:",0);
    }
    else throw errors1;
}
    catch(unsigned exception){
        ErrorMessage(exception);//show error message (you
can use this block to handle errors)
    }
}
    else throw errors;
}
    catch(unsigned exception){
        ErrorMessage(exception);//show error message (you can use
this block to handle errors)
    }
}

    MessageBox(NULL,"DeleD C++ templates version 1.1 written by:\n
Salvatore Russo.\nLot of thanks to Jeroen and Paul Jan\nfor this
new improved version of DeleD!!!","Information",0);
}

```

dllmain.cpp of the import plugin.

This is the main code of import plugin:

```

/*****
*****
* This is the main code of a DeleD import plugin template for
C++.
*
* Author: Salvatore Russo "Il Buzzo"
*
* Version: 1.2.0
*
* DeleD version: 1.1.0
*
* License: GNU General Public License.
*
* Made with: Dev-Cpp 4.9.9.2 and MinGW (Tested with GCC 3.3.1,
GCC 3.2.2
*
* and GCC 3.4.2)
*
* Link Libraries: libcomdlg32.a (it includes calling to
OpenDialog etc..)
*
* Linker parameters: --no-export-all-symbols --add-stdcall-alias
*
* Additional Compiler Parameters: -DBUILDING_DLL=1
*
* End Notes: This is an adaptation to new plugin system, so there
are some
*
* changes from the old code, but as result I'm still
adding to
*
* to code try{}catch() blocks for a better error
handling.
*
* It's not so difficult so if you don't want to wait
for another
*
* version you can implement it without difficulty.
*
*****
*****/

//Main header
#include "dll.h"
//Necessary for DLL to work
#include <windows.h>

//support.h contains some classes used, defined by me
//like String and StringList and some defined for DeleD
//in Jeroen sample code.
//Look at that file for major details.
#include "support.h"

//TCallback variable used to receive a pointer by DeleD
TCallback callback;

/* EXPORTED FUNCTIONS NECESSARY TO DELED

PCHAR PluginName() : returns name of Plugin
PCHAR PluginDescription() : returns a small description of
Plugin

```

```

PCHAR PluginDeleDVersion() : returns version of DeleD
PCHAR PluginVersion() : returns version of Plugin
PCHAR PluginAuthor() : returns Author's name
PCHAR PluginEmail() : returns Author's e-mail
void PluginSetCallback(void*cp) : pass callback pointer from
DeleD to callback
void PluginExecute() : it executes plugin code

notes:
    PCHAR is the same as char* so it's a null terminated c
string.
    Each function is defined as DLLEXPORT for major details see
the DLLEXPORT definition in dll.h file.
    As you can notice the function PluginType() is no longer
used.
    */

DLLEXPORT PCHAR PluginName(){
    return "DeleD Import Plugin Sample";
}

DLLEXPORT PCHAR PluginDescription(){
    return "This is just a sample to test main plugin system
functionality.";
}

DLLEXPORT PCHAR PluginDeleDVersion(){
    return "1.1";
}

DLLEXPORT PCHAR PluginVersion(){
    return "1.0";
}

DLLEXPORT PCHAR PluginAuthor(){
    return "Your Name";
}

DLLEXPORT PCHAR PluginEmail(){
    return "youremail@yourdomain";
}

DLLEXPORT void PluginSetCallback(void *cp){
    callback=TCallBack(cp);
}

DLLEXPORT void PluginExecute(void){
    //variables used
    TCbData cbData;//callback data
    String MyChars;//a string containing scene info

    //clear the string to store cube data
    MyChars.clear();

    //First of all we'll try to get DeleD directory to show new
plugin

```

```

//system enhancements
try{
    unsigned error;// errors returned by callback
    cbData.Reason = PR_GETINSTALLDIR;//reason of callback
    cbData.Data = new char[512];//allocate new space for string
    error=callback(&cbData);//pass Data to callback retrieving
eventual errors
    if(!error){//if no errors detected then go on
        String myStr = String(cbData.Data);//get directory
        MessageBox ( NULL, myStr.c_str(),"This is DeleD main
directory:",MB_ICONINFORMATION);//show it
        myStr.clear();
    }
    else throw error;//if errors generate an exception
    delete cbData.Data;//destroy previously allocated memory
}
catch( unsigned exception) {
    ErrorMessage(exception);//show error message (you can use
this block to handle errors)
}

//Now we do the real job of passing a cube to DeleD
try{
    unsigned errors;// errors returned by callback
    //we're now creating string to pass to DeleD
    MyChars=CreateDeleDCube();//create a basic cube
    String MyData=String("1;091;1,1,0,");//Number of
Headers,Version,Header Item (in this case object), Number of
items,Offset from end of header,
    MyData = MyData + toString(MyChars.length()) + String
(";");//we must now add total length of object data as the last
header field;
    MyData = AddLengthToHeader(MyData);//We add in front of
header total lenght of it
    MyData = MyData + MyChars;//Final string made by
header+data

    cbData.Reason = PR_SETDATA;//reason of callback
    cbData.DataSize = (MyData.length()+1);//size of data to be
sent, a null terminated string
    cbData.Data = new char[ (MyData.length()+1) ];//allocates
memory for string
    sprintf(cbData.Data,"%s",MyData.c_str());//copy memory to
Data
    MyData.clear();//clear Data strings
    MyChars.clear();
    errors=callback(&cbData);//pass Data to callback retrieving
eventual errors
    delete cbData.Data;//destroy previously allocated memory
    if(!errors) MessageBox(NULL,"Cube sent to DeleD
correctly","Information",0);
    else throw errors;
}
catch(unsigned exception){

```

```
        ErrorMessage(exception);//show error message (you can use  
this block to handle errors)  
    }  
    //We have finished our works so a message of information.  
    MessageBox(NULL,"DeleD C++ templates version 1.1 written by:\n  
Salvatore Russo.\n\nLot of thanks to Jeroen and Paul Jan\nfor this  
new improved version of DeleD!!!","Information",0);  
}
```