Issue #Apr05/2015

GENERAL PROXY-DLL PACKAGE (GPP)

INFO

GPP is intended to show 3rd party information in DirectX games. This is beta software for testing only. Be aware of the possibility of system crashes and data loss. When testing, always reboot system after any abnormal termination of your game.

GPP does work with games *using* DirectX8 and 9 (from v1.5 on, there are two different packages available. Before that, DX8 and DX9 were combined in one download). As DX is downwards compatible, this would also work if you have a higher version of DX installed. The game needs to be designed to use DirectX8 or 9 with the “correct” release version, though.

Note that DirectX8 version is discontinued since v1.6. **DX9 only now (64 and 32bit systems)**.

HISTORY

1/31/04	V1.0: Initial release
7/5/04	V1.1: added initial startup logo (d3d8.dll), added ingame text line color, weight, size and font family. moved single function calls to one combined function. Added multiline feature.
4/10/05	V1.2: added “bold text” feature to multiline functions. Added basic ingame picture functions.
4/30/05	V1.3: added “screen size information” feature. Changed picture on startup.
9/10/05	V1.4: added a basic FPS info (switchable)
10/2/05	V1.5: added more TextLines (single/multi) for ingame use. Reduced impact of dll on fps (if no actions active) somewhat. Added screenshot feature. Focused on DX9.
1/13/07	V1.5: Made a DX8 “port” of V1.5. Now, there is GPP1.5_DX8 and GPP1.5_DX9 available as different downloads. Changed readme file to MS Image Writer Format
1/7/15	V1.6 (DX9 only): Now based on DirectX SDK June2010. Added a Unity test app.
1/17/15	V1.7 (D9 only): Removed GPHook.dll from package, as not in focus of this project (not used by the demo program anymore). Statically linked MFC to gpcomms (no external mfcXX.dll anymore). Created 32 and 64 bit compiles. Moved project to VS2013. Moved all code from MB to UNICODE. Simplified the demo (not using key hook anymore, just predefined buttons). Added 32 and 64bit Unity test apps (windowed). Bugfix of text carry over in ML/SL (was preventing correct clearing of text). Changed detection of DX Surface size to work with windowd app. Changed screenshot creation to work with windowed app.
4/5/15	V1.8 (D9 only): Max number of single line objects now 10 (was 5). Max. number of multiline objects remains 5. Internally, SL_DrawTextLineIngame() now called after ML_DrawMultiLineIngame() to draw single lines on top of multilines. Added "Picture (from internal memory)” functions.

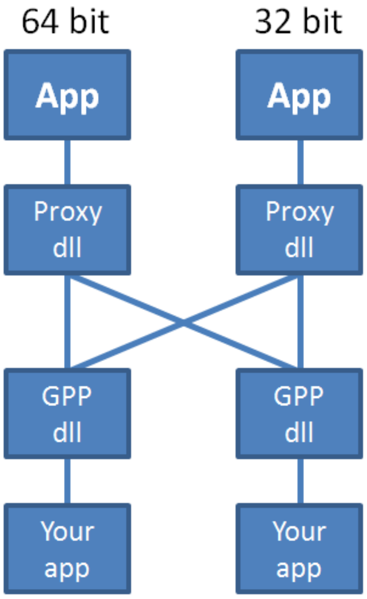
CONTACT/AUTHORS

Download page: http://www.mikoweb.eu
mail: miko@mikoweb.de

64 AND 32 BIT VERSIONS

The GPP package comes with a 32 and 64 bit compile. See the corresponding directory in the “1_Distribution” folder. Note that the correct version of d3d9.dll needs to be provided to the respective application. E.g., pure 64bit applications (games) will only load the 64bit d3d9.dll. The same applies for your application and the gpcomms.dll.

If your application is 32bit and the game is 64bit, this would also work (as data exchange between gpcomms.dll and d3d9.dll is by shared memory), and vice versa. See here:



UNICODE

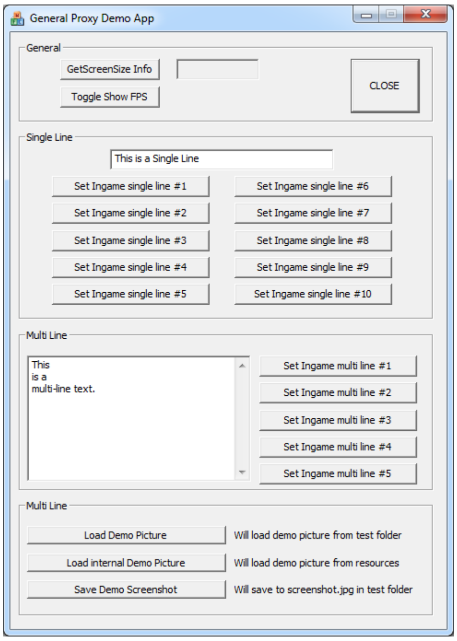
Be aware that starting with v1.7, the GPP has been switched to **UNICODE**. Before, it was Multi-Byte. If you have an app running with v1.6 and before, you will need to adapt your code!

FILES

d3d9.dll
Proxy-dll to be placed into the game's main exe directory. (DO NOT replace any d3d9.dll in your windows directories! This will damage your DirectX config!). The d3d9.dll needs d3dx9_43.dll, which is a Original Microsoft dll (DX code). Same name for 32 and 64 bit versions.

gpcomms.dll/gpcomms_64.dll
Communications wrapper for interaction of an application with the proxy-dll. Uses Microsoft MFC extensions, statically linked since v1.7.

gp_demo.exe/gp_demo_64.exe
Simple demo application for accessing the proxy dll. Can show text and pictures in game and make screenshots.


General Proxy Demo App
General
GetScreenSize Info
Toggle Show FPS
CLOSE
Single Line
Set Ingame single line #1
Set Ingame single line #2
Set Ingame single line #3
Set Ingame single line #4
Set Ingame single line #5
Set Ingame single line #6
Set Ingame single line #7
Set Ingame single line #8
Set Ingame single line #9
Set Ingame single line #10
Multi Line
Set Ingame multi line #1
Set Ingame multi line #2
Set Ingame multi line #3
Set Ingame multi line #4
Set Ingame multi line #5
Multi Line
Load Demo Picture
Load internal Demo Picture
Save Demo Screenshot

This is a Single Line

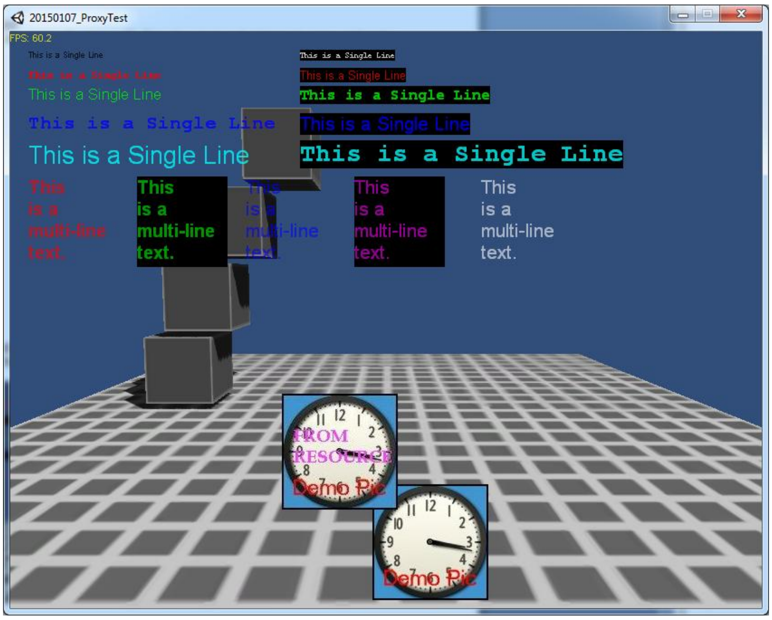
This
is a
multi-line text.

Will load demo picture from test folder

Will load demo picture from resources

Will save to screenshot.jpg in test folder

TestProgram.exe/ TestProgram_64.exe
A Unity3D app (DX9) for testing the proxy dll (windowed DX9 app).



GPCOMMS.DLL EXPORTED FUNCTIONS

Single Text Line Functions

```
bool __stdcall GPSL_SetTextLineData(     BYTE cObjectNumber,
                                         WORD wTextPosX,
                                         WORD wTextPosY,
                                         LPCWSTR  pTextLine,
                                         DWORD dwTextColor,
                                         bool bBlackBackground,
                                         BYTE cSize, bool bTextBold,
                                         BYTE cFontFamily)
```

One and only function to set up and change ingame single line text. It returns "true" on success, "false" otherwise.

cObjectNumber
Up to 10 independent Single Line Objects can be defined and positioned. This is the index.
Values: 0...9

wTextPosX, wTextPosY:
Ingame text position (upper left corner of textbox). Values are given in pixels and are based on the current screen resolution. Remember that WORD is a 16bit unsigned integer value!

PTextLine:
Text to show ingame. A pointer to a string. The string MUST be 0-terminated. Maximum length of string: 128 UNICODE characters, including the terminating '\0'.

DwTextColor:
Color of the text. Format is like the DirectX D3DCOLOR value: ARGB (Alpha, Red, Green, Blue). 0xFF000000 would be black with no transparency.

BBlackBackground:
Defines how the background of the ingame text is handled. It can be transparent (bBlackBackground = false) or a black box (bBlackBackground = true).

CSize:
The size of the font in "logical device units". BYTE value.

BTextBold:
Sets the text to be drawn bold (bTextBold = true) or normal (bTextBold = false)

CFontFamily:
Two general font family specifiers are available. The system will (hopefully) select one out of the present system fonts. BYTE value. 0 = SWISS type (proportional font like Arial), 1 = MODERN type (monospace font like Courier New)

```
bool __stdcall GPSL_ShowText(BYTE cObjectNumber , bool bShowIt)
```

cObjectNumber
Up to 5 independent Single Line Objects can be defined and positioned. This is the index.
Values: 0...4

Switches the ingame text on and off. Default is "off". Function returns "true" on success, "false" otherwise.

Multiple Text Line Functions

```
bool __stdcall GPML_SetTextMultilineData( BYTE cObjectNumber,
                                          WORD wPosX,
                                          WORD wPosY,
                                          LPCWSTR  pText,
                                          DWORD dwTextColor,
                                          bool bBlackBackground,
                                          BYTE cFontSize, bool bTextBold,
                                          WORD wSizeX,
                                          WORD wSizeY,
                                          BYTE cFontFamily)
```

One and only function to set up and change ingame multiline text. It returns "true" on success, "false" otherwise.

cObjectNumber
Up to 5 independent Mulit Line Objects can be defined and positioned. This is the index.
Values: 0...4

wPosX, wPosY:
Ingame text position (upper left corner of textbox). Values are given in pixels and are based on the current screen resolution. Remember that WORD is a 16bit unsigned integer value !

pText:
Text lines to show ingame. A pointer to a string. The string MUST be 0-terminated. Maximum length of string: 512 UNICODE characters, including the terminating '\0'. Word wrap is enabled to show the text within the screen area given by the wSizeX/Y parameters (see below). <CR><LF> will show up as carriage return/line feed. A partially visible last line (not fitting completely in the bounding rectangle) will be cancelled.

dwTextColor:
Color of the text. Format is like the DirectX D3DCOLOR value: ARGB (Alpha, Red, Green, Blue). 0xFF000000 would be black with no transparency.

bBlackBackground:
Defines how the background of the ingame text is handled. It can be transparent (bBlackBackground = false) or a black box (bBlackBackground = true).

cFontSize:
The size of the font in "logical device units". BYTE value.

BTextBold:
Sets the text to be drawn bold (bTextBold = true) or normal (bTextBold = false)

wSizeX, wSizeY:
Size of the "bounding text box". All text will be shown within this box only. Values are given in pixels and are based on the current screen resolution. Remember that WORD is a 16bit unsigned integer value !

cFontFamily:
Two general font family specifiers are available. The system will (hopefully) select one out of the present system fonts. BYTE value. 0 = SWISS type (proportional font like Arial), 1 = MODERN type (monospace font like Courier New)

bool __stdcall GPML_ShowText(BYTE cObjectNumber, bool bShowIt)

cObjectNumber
Up to 5 independent Mulit Line Objects can be defined and positioned. This is the index.
Values: 0...4

Switches the ingame multiline text on and off. Default is "off". Function returns "true" on success, "false" otherwise.

Ingame Bitmap Functions

Basically, you can set the GPP to show one picture loaded from disk and one picture sent by internal stream (byte data of “file in memory”). That makes a total of two pictures max to be shown. The “load from disk” and “internal data” picture features are independent from each other.

bool __stdcall GPPIC_LoadNewPicture(LPCWSTR sPathToFile)

Set a picture from disk to be loaded (and shown ingame later). Allowed formats: .bmp, .dds, .dib, .hdr, .jpg, .pfm, .png, .ppm, and .tga. There is no transparency.
Having a picture size (X*Y) with the powers of 2 is recommended for compatibility with some graphics cards (read: better use 256x256 pixels than e.g. 200x200).

bool __stdcall GPPIC_ShowPicturePos(bool bShowIt, WORD wPosX, WORD wPosY)

bShowIt:
Switches the ingame picture (loaded from disk) on and off. Default is "off". Function returns "true" on success, "false" otherwise.

wPosX, wPosY:
Ingame picture position (upper left corner of picture). Values are given in pixels and are based on the current screen resolution. Remember that WORD is a 16bit unsigned integer value!

bool __stdcall GPPICI_LoadNewInternalPicture(BYTE* pData, UINT uiDataSize)

Send the raw data of a picture “as file in memory” to be shown in game. Allowed formats: .bmp, .dds, .dib, .hdr, .jpg, .pfm, .png, .ppm, and .tga. There is no transparency. In the demo app (where the picture resides in the app’s resources), it is done like so:

```
HRSRC picResource = ::FindResource(NULL, MAKEINTRESOURCE(IDB_BITMAP1), RT_RCDATA);
HGLOBAL picResourceData = ::LoadResource(NULL, picResource);
BYTE* pPicBinaryData = (BYTE*) ::LockResource(picResourceData);

GPPICI_LoadNewInternalPicture(pPicBinaryData, SizeofResource(NULL, picResource));
GPPICI_ShowInternalPicturePos(true, 300, 400);
```

pData:
Pointer to a byte blob of the raw picture data “as on disk” (so, not an e.g. DDB bitmap’s data area).

uiDataSize:
Size of the data in bytes.

bool __stdcall GPPICI_ShowInternalPicturePos(bool bShowIt, WORD wPosX, WORD wPosY)

bShowIt:
Switches the ingame picture (loaded by byte stream) on and off. Default is "off". Function returns "true" on success, "false" otherwise.

wPosX, wPosY:
Ingame picture position (upper left corner of picture). Values are given in pixels and are based on the current screen resolution. Remember that WORD is a 16bit unsigned integer value!

System Information Functions

bool __stdcall GPSI_GetScreenSize(int& piScreenX, int& piScreenY)

piScreenX, **piScreenY**
Retrieve the size of the DirectX screen in pixels. Should also work for windowed apps. If the GPP can not determine the screen’s size, both values are set to 0.

bool __stdcall GPSI_ShowFPS(bool bShowIt)

bShowIt:
Switches the ingame FPS info on and off. Default is "off". FPS info is always shown in the upper left screen corner. It is not changeable in size and/or color. Function returns "true" on success, "false" otherwise.

Graphics Functions

bool __stdcall GPSS_MakeScreenshot(bool bFormat, LPCWSTR sPathToScreenshot)

Function to request a screenshot. The backbuffer will be saved, including any data added by the GPP. If a previous screenshot request is still pending/being processed, a subsequent call to GPSS_makeScreenshot will fail. Function returns "true" on success, "false" otherwise.
Note that “true” indicates that the request has successfully been passed to the gpp package. It does not indicate that the screenshot has already been taken.

bFormat
Two formats are supported: *.bmp (= false) and *.jpg (=true).

sPathToScreenshot
The path and filename where the data will be saved. The calling program must take care of right path and suffix of the created file. E.g., a valid call would be *GPSS_MakeScreenshot(true, “C:\\test.jpg”)*