

GENERAL PROXY-DLL PACKAGE (GPP)**INFO**

GPP is intended to show 3rd party information in DirectX fullscreen mode 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.

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. |

FILES**d3d9.dll v1.6**

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).

gpcomms.dll v1.6

Communications wrapper for interaction of an application with the proxy-dll. Uses Microsoft MFC extensions (MFC71.dll needs to be installed on your system).

gp_demo.exe v1.6

Demo application for accessing the proxy dll. Run before starting your game or do afterwards (in case you are able to alt-tab out). Set text to show ingame (single line or multiline), set position of text ingame (screen *ingame* dimension values). Select a bitmap file on disk to be shown ingame. Select background type of text (black/transparent). Show ingame.

GPHook.dll v1.4

Keyboard hook dll, to be used ingame. This is provided for convenience.

TestProgram.exe

A Unity3D app (DX9) for testing.

CONTACT/AUTHORS

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

Keyboard hooking: Mike "Mogar" Welch
GPP core: Michael "Miquant" Koch

GPCOMMS.DLL EXPORTED FUNCTIONS

Single Text Line Functions

| | |
|---|---|
| bool __stdcall GPSL_SetTextLineData(| BYTE cObjectNumber, WORD wTextPosX, WORD wTextPosY, LPCTSTR 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 5 independent Single Line Objects can be defined and positioned. This is the index.
Values: 0...4

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 bytes (chars), 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,
                                           LPCTSTR 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 Mult 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: 1024 bytes (chars), 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 Multiline 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

| |
|---|
| bool __stdcall GPPIC_LoadNewPicture(LPCTSTR 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 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 screen in pixels. Note that this function does not work when the DirectX application is minimized. 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, LPCTSTR sPathToScreenshot) |
|--|

Function to request a screenshot. The frontbuffer 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")*

GPHOOK.DLL EXPORTED FUNCTIONS AND USAGE

Functions

| |
|---|
| bool __stdcall HookOn (HWND hAppWindow, bool bDebug) |
|---|

Sets the global keyboard hook. Call this on application startup. Sends a message to the window specified by hAppWindow (see below) whenever a key is pressed (or released). Provides additional debug output when bDebug = true.

| |
|--------------------------------------|
| bool __stdcall HookOff (void) |
|--------------------------------------|

Disables the global keyboard hook. Call this on application termination.

Usage

An application (window) must provide a message handler to receive information from the global keyboard hook. This message handler must response to to a custom command number as seen here:

```
#define CM_COMMAND_KEY (WM_USER + 0x1000)
```

The message handler routine is standard format (e.g., for MFC/Visual C++):

```
// -----  
BEGIN_MESSAGE_MAP(CMyMainDialog, CDialog)  
    ON_MESSAGE(CM_COMMAND_KEY, OnMyHookKeyHandler)  
END_MESSAGE_MAP()  
  
LRESULT CMyMainDialog::OnMyHookKeyHandler(WPARAM wParam, LPARAM lParam)  
...  
// -----
```

Filtering keys within this routine could look like this:

```
// -----  
if (wParam == VK_INSERT) // the insert key  
{  
    if (GetKeyState(VK_INSERT)&(0x0001<<7)) // it is being PRESSED  
    {  
        // do weird things here  
    }  
}  
// -----
```