WOWvx OpenGL Control

API Reference

Philips 3D Solutions
**Document Information**

<table>
<thead>
<tr>
<th>Info</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>WOWvx OpenGL Control, API Reference</td>
</tr>
<tr>
<td>Date</td>
<td>24 September 2008</td>
</tr>
<tr>
<td>Security</td>
<td>The attached material and the information contained herein are proprietary to Philips 3D Solutions. Copying, reproduction, adaptation, modification or dissemination in whole or part is not permitted without written permission from Philips 3D Solutions.</td>
</tr>
</tbody>
</table>
| Contact address | Philips 3D Solutions  
High Tech Campus 27  
5656 AE Eindhoven  
The Netherlands,  
E-mail support: 3DService@philips.com  
E-mail general: 3DSolutions@philips.com  
Website: www.philips.com/3dsolutions |
Table of Contents

1 Introduction .............................................................................................................................................................................4
2 How to use the WOWvx OpenGL Control .............................................................................................................................................................................5
  2.1 Running the application in full-screen at native resolution .................................................................................................5
  2.2 Loading the WOWvx OpenGL Control DLL .......................................................................................................................5
  2.3 Calling the WOWvx OpenGL Control DLL functions ........................................................................................................5
3 Function reference .................................................................................................................................................................6
  3.1 SetRenderingParameters .............................................................................................................................................6
  3.2 Enable3DMode ..............................................................................................................................................................7
  3.3 SignalBackgroundAvailable ..........................................................................................................................................7
  3.4 SignalZBufferAvailable ..................................................................................................................................................8
  3.5 SelectFormat ..................................................................................................................................................................8
  3.6 SetApplicationResolution ............................................................................................................................................8
  3.7 SetInputResolution .......................................................................................................................................................9
4 Optimizing depth information ............................................................................................................................................11
  4.1 inverseProjection and farNearRatio .......................................................................................................................11
  4.2 zNearInterest and zFarInterest ........................................................................................................................................12
5 Troubleshooting ....................................................................................................................................................................13
1 Introduction

The purpose of the WOWvx OpenGL Control DLL and its API is to allow an application builder to get his OpenGL application running on one of our 3D displays with minimal effort. To this end the DLL exports several functions in addition to the functions exported by the regular OpenGL DLL. These functions can be used by applications to make optimal use of the 3D display.

Figure 1 gives a schematic overview of an OpenGL application using the DLL to use a 3D display. The left shows how an OpenGL application calls functions from the regular opengl32.dll. On the right an application uses the WOWvx OpenGL Control DLL to transform the output into a format that is suitable for a 3D display.

![Figure 1: The difference between a regular OpenGL application and one using the WOWvx OpenGL Control DLL](image)

Figure 1: The difference between a regular OpenGL application and one using the WOWvx OpenGL Control DLL
2 How to use the WOWvx OpenGL Control

It is extremely easy to change an existing OpenGL application so it will work with the WOWvx OpenGL Control.

2.1 Running the application in full-screen at native resolution

The display should be set to its native resolution; otherwise it will not be able to switch to 3D mode. Additionally, the application needs to generate output that matches the native resolution of the display and covers the entire screen.

2.2 Loading the WOWvx OpenGL Control DLL

If the application does not refer to the system’s opengl32.dll using a full path, the WOWvx OpenGL Control DLL only has to be placed next to the executable in the directory of the application. This will automatically cause the WOWvx OpenGL Control DLL to be loaded instead of the system’s OpenGL DLL, regardless of whether the application uses the LoadLibrary() function or was linked to opengl32.lib at compile-time.

In case the application does use a full path to the OpenGL DLL, the path will have to be changed to point to the WOWvx OpenGL Control DLL instead.

2.3 Calling the WOWvx OpenGL Control DLL functions

It is not necessary to change the OpenGL application in order for OpenGL function calls to work properly. For the extra functions exported by our DLL, the application will have to call GetProcAddress() to retrieve function pointers. How this can be done is illustrated in the source code of the sample application that is shipped with the WOWvx OpenGL Control.

In most cases the WOWvx OpenGL Control will generate proper output for a 3D display even if the application does not call any of the API functions. It is recommended to first try out the application without calling any of the API functions and then add calls to these functions as necessary or desired.
3 Function reference

The following sections describe the functions exported by WOWvx OpenGL Control version 1.1. A pointer to these functions can be retrieved using the GetProcAddress() function. Default values mentioned will be used if the function is not called.

3.1 SetRenderingParameters

Purpose: Set parameters that influence the 3D effect

Prototype:
```c
BOOL SetRenderingParameters( int headerFactor,
   int headerOffset,
   int horizClearEdge,
   int vertClearEdge,
   BOOL visualSmooth,
   BOOL inverseProjection,
   float zNearInterest,
   float zFarInterest,
   float farNearRatio );
```

Parameters:
- **headerFactor**: How much depth should be displayed [0..255]. The resulting quantity of depth will be (headerFactor / 64 * 100)% of the recommended depth for the display. Default value: 64
- **headerOffset**: Depth range behind the screen [0..255]. 0 = All objects are in front of the screen 255 = All objects are behind the screen Default value: 128
- **horizClearEdge**: Horizontal Clear Edge value [0..3] Default value: 0
- **vertClearEdge**: Vertical Clear Edge value [0..2] Default value: 0
- **visualSmooth**: True if smooth Visualization is enabled, false if raw visualization is enabled. Default value: FALSE
- **inverseProjection**: Enables or disables the use of the inverse of the projection matrix to calculate back from Z buffer values to Z coordinates Default value: TRUE
- **zNearInterest**: The Z buffer value of the closest pixels that are of interest [0.0 .. 1.0] Default value: 0.0
- **zFarInterest**: The Z buffer value of the farthest pixels that are of interest. [0.0 .. 1.0] Default value: 1.0
- **farNearRatio**: Value that influences the function that converts Z buffer values to Z coordinates [2.0 .. 32768.0] Default value: 512.0

Returns: TRUE Operation successful

---

1 The total quantity of depth that can be displayed is not unlimited. Depending on the value of headerOffset and on the depth factor and offset set with the display control tool, the maximum can be reached before headerFactor is set to 255.

24 September 2008
Parameters out of range

Description: The parameters set will be in effect from the next frame that will be displayed.

The Clear Edge feature will take for each pixel in the depth map the maximal depth value of all pixels in the surrounding area defined by the horizClearEdge and vertClearEdge settings. Setting horizClearEdge to 2 and vertClearEdge to 1 defines an area of 5 by 3 pixels centred on the current pixel. This will help to hide artefacts in scenes where background pixels still have a bit of foreground color in them, which could happen with antialiasing for instance.

Smooth Visualization is a mode in which the transition between viewing cones is more gradual at the expense of the depth range.

Set inverseProjection to FALSE if using the inverse projection matrix results in a depth effect that is not satisfactory. If inverseProjection is FALSE, a function is used to approximate a proper conversion from Z buffer values into Z coordinates. FarNearRatio can be used to tune this function.

Set zNearInterest and zFarInterest to indicate the range of Z buffer values that are used for objects of interest. For instance, if all objects are drawn at depths between 0.0 and 0.2 and only the background is at 1.0, it is a good idea to set zNearInterest to 0.0 and zFarInterest to 0.3. The depth of anything outside this range is clipped to [zNearInterest .. zFarInterest].

### 3.2 Enable3DMode

**Purpose:** Switch between 2D and 3D mode

**Prototype:**

```c
void Enable3DMode( BOOL enabled );
```

**Parameters:**

- `enabled` Indicates whether or not 3D mode should be enabled.
  - Default value: TRUE

**Returns:** -

**Description:** After calling this function, the next frame to be displayed will be in the requested mode.

### 3.3 SignalBackgroundAvailable

**Purpose:** Signal to the WOWvx OpenGL Control that the framebuffer and/or Z-buffer contain the right color and depth information for the background in Declipse mode.

**Prototype:**

```c
void SignalBackgroundAvailable( BOOL colorBuffer,
                                BOOL zBuffer );
```

**Parameters:**

- `colorBuffer` A boolean indicating that the color buffer is ready
- `zBuffer` A boolean indicating that the Z buffer is ready

**Returns:** -

**Description:** If the Declipse format is selected for the input of the WOWvx OpenGL Control, this function should be called when the frame buffer and/or Z buffer contain the information for the background. The boolean parameters indicate what information (color and/or depth) is available at the time of the call, therefore multiple calls can and should be made if not all of the information is available at a single point in time. If this function has not been called before SwapBuffers, then the WOWvx OpenGL Control will fall back to non-Declipse mode for that frame.

Note: the first frame of output will always be 2D plus depth even if SignalBackgroundAvailable is called. This is because part of the WOWvx OpenGL Control’s initialization happens during the first call to SwapBuffers.

For more information about the Declipse format, please refer to section 4 of the 3D Content Creation Notes available in the download section of our website ([http://www.philips.com/3dsolutions](http://www.philips.com/3dsolutions)).
3.4 **SignalZBufferAvailable**

Purpose: Signal to the WOWvx OpenGL Control that the Z-buffer contains the right depth information.

Prototype: `void SignalZBufferAvailable();`

Parameters: -

Returns: -

Description: This function should be called when the Z buffer contains the depth information that is to be used for the 3D effect. If this function has not been called, then the Z buffer will be read at the time SwapBuffers is called. This function is useful if the Z-buffer does not contain the proper depth information when the buffers are swapped (e.g. when multi-pass rendering).

3.5 **SelectFormat**

Purpose: Select the format of the 3D output that will be provided to the display.

Prototype: `BOOL SelectFormat ( int format );`

Parameters:
- format: 3D format to use (0 = 2D plus depth, 1 = Declipse)
  - Default value: 0

Returns:
- TRUE: Operation successful
- FALSE: Function called after rendering started

Description: Call this function to select the format for 3D output to the display. For Declipse output the application will have to call SignalBackgroundAvailable as well or the WOWvx OpenGL Control will still output regular 2D plus depth.

Note: This function can only be called before any rendering is done.

3.6 **SetApplicationResolution**

Purpose: Disable auto-detection of the application resolution and use the specified values instead.

Prototype: `BOOL SetApplicationResolution ( int resAppW, int resAppH );`

Parameters:
- resAppW: Width of the application window.
  - Default value: desktop resolution width
- resAppH: Height of the application window.
  - Default value: desktop resolution height

Returns:
- TRUE: Operation successful
- FALSE: Parameters out of range or function called after rendering started

Description: In general it should not be required to call this function and it is recommended not to call it if it isn’t necessary. The WOWvx OpenGL Control attempts to auto-detect the application resolution by retrieving the desktop resolution. Figure 2 shows what the output can look like when an incorrect application resolution is used.

Note: This function can only be called before any rendering is done.
Figure 2: The result of an incorrect application resolution

### 3.7 SetInputResolution

**Purpose:** Set resolution of input that will be generated for the WOWvx OpenGL Control.

**Prototype:**
```c
BOOL SetInputResolution( int resInputW,
                          int resInputH );
```

**Parameters:**
- `resInputW` Width of the input picture to be generated for the WOWvx OpenGL Control.
  - Default value: application resolution width / 2
- `resInputH` Height of the input picture to be generated for the WOWvx OpenGL Control.
  - Default value: application resolution height / 2

**Returns:**
- TRUE Operation successful
- FALSE Parameters out of range or function called after rendering started

**Description:** The application can be forced to render at a different resolution by changing the OpenGL viewport. This results in a lower resolution 2D input to the WOWvx OpenGL Control and influences the quality of the output on the 3D display. Output
quality in 2D mode is optimal if the input resolution is equal to the application resolution. Optimal quality in 3D mode as well as highest performance is achieved when the input resolution is set to a quarter of the application resolution (both width and height divided by 2). Which values to choose depends on how the application is going to be used. If this function is not called, the WOWvx OpenGL Control will use a quarter of the application resolution.

**Note:** This function can only be called before any rendering is done.
4 Optimizing depth information

To optimize the output of the WOWvx OpenGL Control it is usually easiest to start with a normal 2D display. On such a display the image will be shown on the left and the depth information on the right as in Figure 3.

![Figure 3: The output of the WOWvx OpenGL Control shown on a 2D display.](image)

Generation of depth information is influenced by the last four parameters of the SetRenderingParameters function, namely inverseProjection, zNearInterest, zFarInterest and, depending on the value of inverseProjection, nearFarRatio. We'll look into each of these parameters more closely to see how they can be used to optimize the depth information.

4.1 inverseProjection and farNearRatio

It is usually best to use the inverse function of what is used by OpenGL to calculate Z-buffer values from Z coordinates. This is done by inverting the projection matrix. If your application is using a projection matrix that cannot be inverted, an approximation function can be used by setting inverseProjection to FALSE. This is not recommended however, as it makes tuning the output to look right in 3D more difficult. For scenes containing prominent straight lines toward the horizon (such as the intersection between the floor and the wall in Figure 3), it may require you to alternate between adjusting zNearInterest/zFarInterest to get a sufficient use of the range from white to black and adjusting nearFarRatio to improve the perspective of these lines when viewed from a different angle. If the farNearRatio is chosen poorly, lines that should be straight could appear to be curved when seen from certain angles.
4.2 zNearInterest and zFarInterest

In general the objects of interest in an OpenGL scene stay within a certain range of distance to the viewer. In order not to waste the range of depth a 3D display can produce on depth values that are hardly ever in the user’s view, a range of interest can be defined with zNearInterest and zFarInterest. When selecting a range of interest, it is important to keep in mind that anything outside that range will appear to be flat. This is particularly important in the foreground where flat objects may attract attention to themselves because of their incorrect perspectives.

When defining the range of interest for the application in Figure 3, a trade-off had to be made. It was decided to set zNearInterest to the depth at which the walls usually go out of view at the edges of the screen. zFarInterest was set to a value such that other player objects would get around a 50% gray value in the depth information, which corresponds to the screen surface and gives the sharpest representation on a 3D display. That the parts of the scene further away than zFarInterest are really flat is hardly noticeable in this case. The result is a very good depth impression in the part of the scene that matters most at the expense of a lack of depth within the player’s weapon.

A note about choosing the right values: you should start out with the default values for zNearInterest and zFarInterest, which are 0.0 and 1.0 respectively. If the depth information is mostly black, zNearInterest should be increased. If it is mostly white, zFarInterest should be decreased.
5 Troubleshooting

Problem: My application is using the NV_multisample_filter_hint extension on a NVIDIA GeForce graphics card and I see the image on the left half of the screen and a gray scale depth map on the right half of the 3D display.

Solution: Do not call glHint() to set MULTISAMPLE_FILTER_HINT_NV to GL_NICEST. The multisampling done will interfere with the operation of the WOWvx OpenGL Control. Setting the filter hint to GL_FASTEST should eliminate the problem.

Problem: When I start my application I get the error “Can not find import, comdlg32.dll, Error 998, dll may be corrupted or wrong version”.

Solution: The solution to this problem is provided in the knowledge base article: http://support.microsoft.com/kb/137273. None of the files mentioned in the knowledge base article are replaced or changed by the OpenGL Control installation.

- 0 – 0 – 0 – 0 – 0 -