Virtual-E GL Application Programming Interface

Table of Contents

1 Introduction **1**

2 Description **2**

3 Using veGL **3**

4 Reference **4**

4.1 Types 4

4.2 Functions 7

5 About veGL **12**

1 Introduction

The Virtual-E GL (in this document referenced as veGL) is an API that works with OpenGL and the parallel port as the I/O connection to use Virtual Boy as a tridimensional display device. This document will explain the complete description to make a computer program or application that uses this display device taking advantage ot the veGL.
It is important to understand that the following information is subject to changes and this API is still in development, so before you consider veGL as a solution, first wait for the final release of this development kit and the full API. The veGL covers right now only the graphic part to interact with OpenGL, and many parts of the hardware are still unknown, therefore you can only define right now how to render your 3D scene.
Remember that veGL will help you to use Virtual Boy as a device, and any application developed with this API has obviously many restrictions, compared to a normal application. In fact, this API is recomended to write games in 3D and maybe cientific data that must be visualized on 3D spaces.

2 Description

As explained before, the veGL uses OpenGL to perform all the graphic render, but it presents some basic restrictions that you must consider before the use of this API.

- The veGL cannot use a viewport different than the default (768x224) because the Virtual Boy will receive this complete information without any stretch or special raster operation.
- The veGL cannot do any camera movement, because it performs a special 2 camera adjustment according to the natural angle of the human eye. You can move your world but the camera must be static. The natural camera position is 5 units away from the origin relative to Z-Axis and a minimal variation between the left and the right eye-style cameras.
- The veGL quality on image output depends of the complexity presented in the scene and the variety of color used for that render. Many objects, backgrounds and colors will reduce the quality of the output because Virtual Boy can only process 4 colors.

The programming basically consists on a window with the default size, the exchange of data between veGL and your application, and the functionality that will be different from application to application.

Using this programming scheme, a common veGL program will only have a floating window that shows the rendered scene and all the special stuff that you add to this program to manage and interact with the user.

3 Using veGL

The pattern to use veGL differs according the goal of your application, but it must be used in the following way:

- The application must have a default window created and attached to the veGL API using the veglOpen function.
- The render function must be equal to the standard PVEGLPROC type definition, any variation is not allowed.
- You must call veglRender with the render function as a parameter any time you need to update your scene.
- If you need to process the transfer data for something, consider the standard PVEGLTRANSFER type definitionm, any variation is not allowed.
- The veglTransfer must be called any time you need to update the screen on the Virtual Boy, and if you do not use a special transfer function to process the information, the veglTransfer function can receive a NULL value as a parameter.
- After the application has finished, you must call veglClose to be sure that all the internal elements in the API are unreferenced or closed.

All the functions in veGL are defined in the C library called VEGL.LIB provided with the veGL SDK. This library only works with Visual C++. By this moment veGL is not available in any other language, but wait for the final release, maybe it will has support for other plataforms.

4 Reference

4.1 Types

In this version of veGL, there are 2 programming types to use for make callback functions.

PVEGLPROC
PVEGLTRANSFER

PVEGLPROC

The VEGLPPROC type is used to define a function pointer for the rendering process.

```
typedef void (* PVEGLPROC)();
```

Parameters: None.

Defined in VEGL.H

PVEGLTRANSFER

The PVEGLTRANSFER type is used to define a function pointer to the transfer process.

```
typedef void (* PVEGLTRANSFER)(const BYTE* lpData, DWORD dwSize);
```

Paramenters:
`lpData` - this paramenter will receive the data that was sent to the Virtual Boy.
`dwSize`- this parameter is filled witht the size of the data buffer defined in lpData.

Defined in VEGL.H

4.2 Functions

The functions of veGL In this version are:

veglClose
veglOpen
veglRender
veglTransfer

veglOpen

The veglOpen function is the interface between the veGL API and your application because you define here the window that will work as the render device.

```
int veglOpen(HWND hWnd);
```

Parameters:
hWnd- The window handle that will act as render device.

Return Values:
If the functions succeds it returns 1, otherwise it returns 0.

Defined in VEGL.H

veglRender

The veglRender function performs the steps to render the scene defined in the procedure with the proper modifications.

```
int veglRender(PVEGLPROC procedure);
```

Parameters:
procedure- The pointer of the function that defines the scene to render.

Return Values:
If the functions succeds it returns 1, otherwise it returns 0.

Defined in VEGL.H

veglTransfer

The veglTransfer function is the core component in the API because it sends the render device to the Virtual Boy using the IO port defined.

```
int veglTransfer(PVEGLTRANSFER procedure);
```

Parameters:
procedure- The pointer of the function can process the transfered data..

Return Values:
If the functions succeds it returns 1, otherwise it returns 0.

Defined in VEGL.H

veglClose

With veglClose you will clean all the references between you application and veGL.

```
int veglClose();
```

Parameters: None

Return Values:
If the functions succeds it returns 1, otherwise it returns 0.

Defined in VEGL.H

5 About veGL

The veGL API is a freeware development tool written in C/C++. This development tool is based on information provided by David Tucker and personal research on the Virtual BoyTM Arquitecture. For support and questions, write to virtuale98@yahoo.com or visit the Virtual-E site in http://www.emuunlim.com/VirtualE/.

VIRTUAL-E Team