IMG Vertex Program Compiler

Reference Manual

Filename	:	IMG Vertex Program Compiler.Reference Manual.doc
Version	:	1.0.3a External Issue
Issue Date	:	26 Oct 2005
Author	:	PowerVR

Contents

1. The VGP compiler 4
1.1. Introduction 4
1.2. Using the compiler 4
1.2.1. Command line arguments 4
1.2.2. Using the compiled program 4

List of Code

Code 1 - Loading the compiled code in header form 5
Code 2 - Loading the compiled code in binary form 5

1. The VGP compiler

1.1. Introduction

The compiler provided in Utilities\VGPCompiler with the SDK enables users to write custom VGP. Using the compiler allows code to be written, specific to a given task, which will operate on the vertices of incoming geometry. Please see the GL_IMG_vertex_program.txt file for more information on the IMG vertex program extension.

1.2. Using the compiler

Write your program in a text file, named <myfile.vp>. The program must start with the following string

```
!!IMGvp1.0
```

This lets the compiler know that the following code is IMG vertex program, version 1.0.

Next, you must run the compiler, which will produce two alternate forms of the compiled program which you can load and use in your Open GL ES code.

Running the compiler with no arguments will show the usage-help:

1.2.1. Command line arguments

```
C:\DevRel\Utilities>VGPCompiler.exe

Error: Syntax is compiler -i<source file> -o<output name> <options>

       Options:
       -do                    -  Disable optimisations
       -t                     -  Generate timing information
       -vgplite               -  Target HW is VGP Lite
       -target=<targetname> -  Name of target platform (default is 'generic')
       -debug                 -  Generates a debug binary
       -version               -  Displays compiler version number

i.e. compiler -imytest.vp -omytestbin -do -t -vgplite -target=generic -debug
```

Normal usage should leave the optimisations enabled (do not specify `–do`), and should supply the target name (eg `-target=generic` or `–target=armvp` for the ARM VP platform). Suppose you have a program called myVertexProgram.vp, this should be compiled as follows:

```
C:\DevRel\Utilities>VGPCompiler.exe –imyVertexProgram.vp –omyVertexProgram –target=generic
```

The output name does not necessarily have to match the input name (this is useful when generating different versions of the compiled code for different targets.

1.2.2. Using the compiled program

Two files will be produced - myVertexProgram.h and myVertexProgram.bin. The .h file is a C header file which can be included and used directly in code. It has a structure named vgp_<outputfilename>, which is used as below (assuming you use the provided class to get function entrypoints):

```
#include "OGLESTools.h"

//…

GLuint myVertexProgramID;
CPVRTglesExt   g_glesExt;

//…

g_glesExt.Init();

//…

/* Generate IDs for 2 Vertex Programs */
g_glesExt.glGenProgramsARB(1, &myVertexProgramID);

/* Bind and Load first program */
g_glesExt.glBindProgramARB(GL_VERTEX_PROGRAM_ARB, myVertexProgramID);
g_glesExt.glProgramStringARB(GL_VERTEX_PROGRAM_ARB, GL_PROGRAM_FORMAT_BINARY_IMG,
        sizeof(vgp_myVertexProgram), vgp_myVertexProgram);

/* Check for problems */
if (glGetError()!=GL_NO_ERROR)
{
        return false;
```

Code 1 - Loading the compiled code in header form

The binary fine myVertexProgram.bin must be loaded at runtime. The code below will do this for you.

```
#include "OGLESTools.h"

//…

GLuint myVertexProgramID;
CPVRTglesExt   g_glesExt;

unsigned int codeHeader[42];
unsigned int codeSize;
unsigned int *code = 0;

FILE* codeFile = fopen(“myVertexProgram.bin”,”rb”);
fread(codeHeader,42*sizeof(unsigned int),1,codeFile);
codeSize = codeHeader[2];
fseek(codeFile,0, SEEK_SET);
code = (unsigned long*) malloc(sizeof(unsigned long) * (42 + codeSize));
fread(code,sizeof(unsigned long) * (42 + codeSize),1,codeFile);
fclose(codeFile);

g_glesExt.Init();

//…

/* Generate IDs for 2 Vertex Programs */
g_glesExt.glGenProgramsARB(1, &myVertexProgramID);

/* Bind and Load first program */
g_glesExt.glBindProgramARB(GL_VERTEX_PROGRAM_ARB, myVertexProgramID);
g_glesExt.glProgramStringARB(GL_VERTEX_PROGRAM_ARB, GL_PROGRAM_FORMAT_BINARY_IMG,
        sizeof(codeSize), code);

/* Check for problems */
if (glGetError()!=GL_NO_ERROR)
{
        return false;
}
```

Code 2 - Loading the compiled code in binary form