

~\Downloads\GPU\_Gems\_2\_code\GPU Gems 2 code\Content\31\DEMOS\OpenGL\src\shared\pug\pug.h

```

1  /*****NVMH3****
2
3  Copyright NVIDIA Corporation 2004
4  TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED
5  *AS IS* AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS
6  OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
7  MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  IN NO EVENT SHALL
8  NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT, OR
9  CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR
10 LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS
11 INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
12 INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE
13 POSSIBILITY OF SUCH DAMAGES.
14
15 *****/
16
17 #ifndef PUG_H
18 #define PUG_H
19
20 #include <windows.h>
21 #include <GL/gl.h>
22 #include <GL/wglext.h>
23 #include <Cg/cg.h>
24
25 class PUGProgram;
26
27 enum PUGTarget
28 {
29     PUG_FRONT,
30     PUG_BACK,
31     PUG_NUM_TARGETS
32 };
33
34 struct PUGBuffer
35 {
36     PUGBuffer() { }
37     PUGBuffer(int w, int h, int nc, HDC dc, HPBUFFERARB pbuf, HGLRC rc,
38             GLuint th, GLuint ib, bool db=false);
39
40     GLuint texHandle, imageBuffer;
41
42     int width, height, nComponents;
43     bool bound[2];
44
45     // PBUFFER data
46     HPBUFFERARB hpbuffer;    // pbuffer handle
47     HDC hpbufferDC;         // pbuffer device context
48     HGLRC hpbufferRC;       // pbuffer render context

```

```

49
50 // double-buffered PBUFFER data
51 bool doublebuffered;
52 PUGTarget currentDrawBuffer;
53 };
54
55
56 struct PUGRect {
57     PUGRect() { }
58     PUGRect(int xx0, int xx1, int yy0, int yy1) {
59         x00 = x01 = xx0;
60         x10 = x11 = xx1;
61         y00 = y10 = yy0;
62         y01 = y11 = yy1;
63     }
64     PUGRect(int xx00, int xx10, int xx01, int xx11,
65             int yy00, int yy10, int yy01, int yy11) {
66         x00 = xx00; x10 = xx10; x01 = xx01; x11 = xx11;
67         y00 = yy00; y10 = yy10; y01 = yy01; y11 = yy11;
68     }
69
70     int x00, x01, x10, x11;
71     int y00, y01, y10, y11;
72 };
73
74 extern bool pugInit(const char* includePath = "./", CGcontext cgcontext = 0, bool
createOpenGLContext = true);
75 extern bool pugCleanup();
76
77 extern PUGProgram *pugLoadProgram(const char *filename,
78                                   const char *entrypoint,
79                                   const char **args = NULL);
80 extern PUGProgram *pugLoadReductionProgram(const char *filename,
81                                             const char *reduceop,
82                                             int samples = 2,
83                                             int components = 1,
84                                             const char **args = NULL);
85
86 extern bool pugBindFloat(PUGProgram *prog, const char *param,
87                           float v0, float v1 = 0, float v2 = 0, float v3 = 0);
88 extern bool pugBindStream(PUGProgram *prog, const char *param,
89                           PUGBuffer *buf, PUGTarget target = PUG_FRONT);
90 extern bool pugBindDomain(PUGProgram *prog, const char *param,
91                           const PUGRect &domain);
92 extern bool pugRunProgram(PUGProgram *prog, const PUGRect &writeRect,
93                           const PUGRect &readRect);
94 extern bool pugReadMemory(float *out, PUGBuffer *buf);
95 extern bool pugReadMemory(float *out, PUGBuffer *buf, PUGRect rect);
96 extern PUGRect pugTransposeRect(PUGRect &rect);
97

```

```

98 // over all of current output buffer
99 extern bool pugRunProgram(PUGProgram *prog, PUGBuffer *output);
100 extern bool pugRunProgram(PUGProgram *prog, PUGBuffer *output, const PUGRect domain);
101
102 extern bool pugRunProgram(PUGProgram *prog, PUGBuffer *output, const PUGRect domain,
103                          PUGTarget target);
104
105 enum PUGBufferMode {
106     PUG_READ,
107     PUG_WRITE,
108     PUG_READWRITE
109 };
110
111 extern PUGBuffer* pugAllocateBuffer(int width, int height, PUGBufferMode mode,
112                                    int components = 1,
113                                    bool doublebuffer = false);
114
115 extern void pugDeleteBuffer(PUGBuffer* buf);
116
117 extern bool pugInitBuffer(PUGBuffer *buf, const float *data,
118                          PUGTarget target = PUG_FRONT);
119
120 extern bool pugStartReadBack(PUGBuffer *);
121 extern float* pugGetBufferData(PUGBuffer *);
122 extern bool pugReleaseBufferData(PUGBuffer *);
123
124 extern bool pugReleaseBuffer(PUGBuffer *buf, PUGTarget target = PUG_FRONT);
125
126 extern bool pugBindTexture(PUGBuffer *buf, PUGTarget target = PUG_FRONT);
127 extern bool pugMakeWindowCurrent();
128
129 extern void pugWaitForThreads();
130
131 // Parallel Reductions
132
133 enum PUGDimension
134 {
135     PUG_DIMENSION_X,
136     PUG_DIMENSION_Y,
137     PUG_DIMENSION_COUNT
138 };
139
140 extern PUGBuffer* pugReduce1D(PUGProgram *prog, PUGBuffer *inbuf,
141                              PUGBuffer *dblbuf, PUGDimension dim,
142                              int nRows, int nCols);
143 extern PUGBuffer* pugReduce2D(PUGProgram *prog, PUGBuffer *inbuf,
144                              PUGBuffer *dblbuf, int nRows, int nCols);
145
146
147

```

```
148  
149 #endif // PUG_H  
150
```