

~\Downloads\GPU_Gems_2_code\GPU Gems 2

code\Content\31\DEMOS\OpenGL\src\gpgpu_reactiondiffusion\gpgpu_reactiondiffusion.cpp

```

1  #include <gl/glut.h>
2  #include <gl/glext.h>
3  #include <Cg/cg.h>
4  #include <Cg/cggl.h>
5  #include <stdio.h>
6  #include <algorithm>
7  #include <string>
8
9  #include <shared/pug/pug.h>
10
11 #define GET_GLERROR(ret) \
12     { \
13         GLenum err = glGetError(); \
14         if (err != GL_NO_ERROR) { \
15             fprintf(stderr, "[%s line %d] GL Error: %s\n", \
16                 __FILE__, __LINE__, gluErrorString(err)); \
17             fflush(stderr); \
18             exit(1); \
19             return (ret); \
20         } \
21     }
22
23 std::string path = "../shared/pug/";
24
25 CGcontext g_cgContext;
26 CGprofile g_profile;
27
28 PUGBuffer *g_rdBuffer = NULL;
29 PUGProgram *g_rdProgram = NULL;
30
31 PUGTarget g_currentTarget = PUG_BACK;
32 PUGTarget g_currentSource = PUG_FRONT;
33
34 CGprogram g_displayProgram = NULL;
35 CGparameter g_displayTexParam = NULL;
36
37 int g_width = 256;
38 int g_height = 256;
39
40 float g_rK = 0.052f;
41 float g_rF = 0.012f;
42 float g_rDiffusionU = 0.0004f;
43 float g_rDiffusionV = 0.0002f;
44
45 void cgErrorCallback()
46 {
47     CGerror lastError = cgGetError();

```

```

48
49     if(lastError)
50     {
51         printf("%s\n\n", cgGetErrorString(lastError));
52         printf("%s\n", cgGetLastListing(g_cgContext));
53         printf("Cg error, exiting...\n");
54         exit(1);
55     }
56 }
57
58 void reset_rd()
59 {
60     float *pFloatData = new float[4 * g_width * g_height];
61     int k = 0;
62     int i, j;
63
64     for (i = 0; i < g_width * g_height; ++i)
65     {
66         pFloatData[k++] = 1.0;
67         pFloatData[k++] = 0.0;
68         pFloatData[k++] = 0.0;
69         pFloatData[k++] = 0.0;
70     }
71
72     for (i = (0.48f)*g_height; i < (0.52f)*g_height; ++i)
73     {
74         for (j = (0.48f)*g_width; j < (0.52f)*g_width; ++j)
75         {
76             pFloatData[4 * (i * g_width + j)    ] = .5;
77             pFloatData[4 * (i * g_width + j) + 1] = .25;
78             pFloatData[4 * (i * g_width + j) + 2] = 0;
79             pFloatData[4 * (i * g_width + j) + 3] = 0;
80         }
81     }
82
83     // Now perturb the entire grid. Bound the values by [0,1]
84     for (k = 0; k < g_width * g_height * 4; ++k)
85     {
86         if ( pFloatData[k] < 1.0 )
87         {
88             float rRand = .02f*(float)rand() / RAND_MAX - .01f;
89             pFloatData[k] += rRand * pFloatData[k];
90         }
91     }
92
93     pugInitBuffer(g_rdBuffer, pFloatData, g_currentSource);
94     delete [] pFloatData;
95 }
96
97 int update_rd()

```

```
98 {
99     pugBindStream(g_rdProgram, "concentration", g_rdBuffer, g_currentSource);
100
101     PUGRect range(0, g_width, 0, g_height);
102     pugRunProgram(g_rdProgram, g_rdBuffer, range, g_currentTarget);
103
104     std::swap(g_currentTarget, g_currentSource);
105
106     GET_GLERROR(0);
107
108     return 0;
109 }
110
111 void init_opengl()
112 {
113     glDisable(GL_DEPTH_TEST);
114
115     // Create cgContext.
116     g_cgContext = cgCreateContext();
117     cgSetErrorCallback(cgErrorCallback);
118
119     // Setup display program
120     g_profile = cgGLGetLatestProfile(CG_GL_FRAGMENT);
121     cgGLSetOptimalOptions(g_profile);
122
123     const char *args[2];
124     args[0] = new char[strlen("-I../shared/pug")+1];
125     strcpy(const_cast<char*>(args[0]), "-I../shared/pug");
126     args[1] = 0;
127
128     g_displayProgram = cgCreateProgramFromFile(g_cgContext, CG_SOURCE, "rd.cg",
129                                              g_profile, "passthrough", args);
130     cgCompileProgram(g_displayProgram);
131     cgGLLoadProgram(g_displayProgram);
132     g_displayTexParam = cgGetNamedParameter(g_displayProgram, "tex");
133
134     // initialize gpu framework, create a buffer and the r-d program.
135     pugInit(path.c_str(), g_cgContext, false);
136     g_rdBuffer = pugAllocateBuffer(g_width, g_height, PUG_READWRITE, 4, true);
137
138     g_rdProgram = pugLoadProgram("rd.cg", "rd");
139
140     // initialize the parameters of the simulation
141     pugBindFloat(g_rdProgram, "windowDims", g_width, g_height,
142                 -g_rK - g_rF + (1 - 655.36f * g_rDiffusionV), 0);
143     pugBindFloat(g_rdProgram, "rdParams", 655.36f * g_rDiffusionU,
144                 655.36f * g_rDiffusionV, g_rK, g_rF);
145
146     reset_rd(); // initialize the r-d input
147     pugMakeWindowCurrent(); // switch back to the graphics window
```

```
148 }
149
150 void display()
151 {
152     update_rd();           // update the r-d simulation
153     pugMakeWindowCurrent(); // switch back to the graphics window
154
155     glClear(GL_COLOR_BUFFER_BIT);
156
157     pugBindTexture(g_rdBuffer, g_currentSource);
158     cgGLBindProgram(g_displayProgram);
159     cgGLSetTextureParameter(g_displayTexParam, g_rdBuffer->texHandle);
160     cgGLEnableTextureParameter(g_displayTexParam);
161     cgGLEnableProfile(g_profile);
162
163     glBegin(GL_QUADS);
164         glTexCoord2f(0, 0);           glVertex2f(-1, -1);
165         glTexCoord2f(g_width, 0);     glVertex2f( 1, -1);
166         glTexCoord2f(g_width, g_height); glVertex2f( 1,  1);
167         glTexCoord2f(0, g_height);    glVertex2f(-1,  1);
168     glEnd();
169
170     cgGLDisableTextureParameter(g_displayTexParam);
171     cgGLDisableProfile(g_profile);
172
173     glutSwapBuffers();
174 }
175
176 void idle()
177 {
178     glutPostRedisplay();
179 }
180
181 void key(unsigned char k, int x, int y)
182 {
183     switch(k)
184     {
185     case 27:
186     case 'q':
187         exit(0);
188     case 'r':
189         reset_rd();
190         break;
191     }
192
193     glutPostRedisplay();
194 }
195
196 void resize(int w, int h)
197 {
```

```
198     if (h == 0) h = 1;
199     glViewport(0, 0, w, h);
200
201     glMatrixMode(GL_PROJECTION);
202     glLoadIdentity();
203
204     gluOrtho2D(-1, 1, -1, 1);
205
206     glMatrixMode(GL_MODELVIEW);
207     glLoadIdentity();
208 }
209
210 int main(int argc, char **argv)
211 {
212     glutInit(&argc, argv);
213     glutInitWindowSize(512, 512);
214     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
215     glutCreateWindow("gpgpu_reactiondiffusion");
216
217     init_opengl();
218
219     glutDisplayFunc(display);
220     glutIdleFunc(idle);
221     glutKeyboardFunc(key);
222     glutReshapeFunc(resize);
223
224     glutMainLoop();
225     return 0;
226 }
```