Appendix B

C++ Agents

```
1
2  #ifndef _WGD_SHADER_
3  #define _WGD_SHADER_
4
5  #ifdef WIN32
6  #include <windows.h>
7  #endif
8  #include <GL/gl.h>
9  #include <map>
10 #include <cadence/doste/oid.h>
11 #include <cadence/dstring.h>
12 #include <cadence/agent.h>
13 #include <cadence/file.h>
14 #include <string>
15 #include <wgd/index.h>
16
17
18 namespace wgd {
19
20     class vector3d;
21     class Texture;
22
23 /**
24  * Shader Resource. This contains everything a shader needs to run,
25  * you apply the shader to an object the same way you would apply a texture
26  * with bind() and unbind(). The shader replaces normal textures so if
27  * you use a shader, you dont bind textures directly to the object.<br>
28  * You must specify both vertex shader and fragment shader source, as well as
29  * all the textures that the shader uses. <br>
30  * Shaders that need Binormals and Tangents will automatically get the tangent
31  * if applied to models and primitives, bot you must calculate the binormal
32  * in your vertex shader using: cross(gl_Normal, tangent); <br>
33  * The tangent varying variable must be named "tangent" for it to work.<br>
```

```
34    * If a shader program fails to compile, or will not run on your machine
35    * nothing will happen when you try to bind it.
36    * <br/><br/>
37    *
38    */
39       class RESIMPORT Shader : public cadence::Agent {
40
41           public:
42
43           OBJECT(Agent, Shader);
44
45           Shader();
46           Shader(const cadence::doste::OID &);
47           Shader(cadence::File &vert, cadence::File &frag);
48           Shader(const char* vertfile, const char* fragfile);
49           ~Shader();
50
51           PROPERTY_WF(cadence::File, vert, ix::vert);
52           PROPERTY_RF(cadence::File, vert, ix::vert);
53
54           PROPERTY_WF(cadence::File, frag, ix::frag);
55           PROPERTY_RF(cadence::File, frag, ix::frag);
56
57           PROPERTY_WF(bool, debug, ix::debug);
58           PROPERTY_RF(bool, debug, ix::debug);
59
60           bool make(const char *vert, const char *frag);
61           bool load();
62
63           void bind();
64           void unbind();
65
66           static Shader *current();
67
68           static void current(Shader *sh);
69
70           bool tangents(){ return m_tangents; };
71
72           void setVariable(const char *name, float v1);
73           void setVariable(const char *name, float v1, float v2);
74           void setVariable(const char *name, float v1, float v2, float v3);
75           void setVariable(const char *name, float v1, float v2, float v3, float v4);
76           void setVariable(const char *name, const wgd::vector3d &vec3);
77           void setVariable(const char *name, int v1);
78           void setVariable(const char *name, int size, int* data);
79           void setVariable(const char *name, int size, float* data);
80
81           void enableAttribArray(const char *name);
82           void attribPointer(const char *name, GLint size, GLenum type,
83                GLboolean normalised, GLsizei stride, const void *pointer);
84           void disableAttribArray(const char *name);
85
86           static void enabled(bool);
87           static bool enabled();
88
```

```
89          BEGIN_EVENTS(Agent);
90          EVENT(evt_reload, (*this)("reload"));
91          END_EVENTS;
92
93          private:
94
95          static void initialise();
96          static bool  s_available;
97          static Shader *s_current;
98
99          GLint addVariable(const char *name);
100
101         bool loadShader();
102         char *readFile(cadence::File *);
103         int logInfo(GLuint s, const char *name);
104
105         GLuint m_vertexShader;
106         GLuint m_fragmentShader;
107         GLuint m_program;
108
109         bool m_ready;
110         bool m_loaded;
111
112         bool m_tangents;
113
114         GLint getLocation(const char *name);
115
116         class ShaderVar{
117             public:
118             ShaderVar(int t, GLint loc): type(t), location(loc){};
119             int type; //1=uniform, 2=attribute
120             GLint location;
121         };
122
123         ShaderVar *getVar(const char *name);
124         cadence::doste::OID m_vars;
125
126
127     };
128 };
129
130 #endif
```

Listing B.1: shader.h