```cpp
#include "shapes.h"
#include <iostream>
#include <sstream>

#include <GL/glew.h>
#include <GLFW/glfw3.h>
#include <glm/glm.hpp>

Shapes::Shapes() {

};

Shapes::~Shapes() {

}

void Shapes::LoadObj() {

    std::vector< glm::vec3 > obj_vertices;
    std::vector< unsigned int > vertexIndices;
    istringstream rawDataStream(rawData);
    string dataLine;  int linesDone = 0;

    while (std::getline(rawDataStream, dataLine)) {
        if (dataLine.find("v ") != string::npos) {  // does this line have a vector?
            glm::vec3 vertex;

            int foundStart = dataLine.find(" ");  int foundEnd = dataLine.find(" ", foundStart + 1);
            vertex.x = stof(dataLine.substr(foundStart, foundEnd - foundStart));

            foundStart = foundEnd; foundEnd = dataLine.find(" ", foundStart + 1);
            vertex.y = stof(dataLine.substr(foundStart, foundEnd - foundStart));

            foundStart = foundEnd; foundEnd = dataLine.find(" ", foundStart + 1);
            vertex.z = stof(dataLine.substr(foundStart, foundEnd - foundStart));

            obj_vertices.push_back(vertex);
        }
        else if (dataLine.find("f ") != string::npos) { // does this line defines a triangle face?
            string parts[3];

            int foundStart = dataLine.find(" ");  int foundEnd = dataLine.find(" ", foundStart + 1);
            parts[0] = dataLine.substr(foundStart + 1, foundEnd - foundStart - 1);
```

```
44
45              foundStart = foundEnd; foundEnd = dataLine.find( " ", foundStart
                  + 1);
46              parts[1] = dataLine.substr(foundStart + 1, foundEnd - foundStart
                   - 1);
47
48              foundStart = foundEnd; foundEnd = dataLine.find( " ", foundStart
                  + 1);
49              parts[2] = dataLine.substr(foundStart + 1, foundEnd - foundStart
                   - 1);
50
51              for (int i = 0; i < 3; i++) {       // for each part
52
53                  vertexIndices.push_back(stoul(parts[i].substr(0, parts
                      [i].find("/"))));
54
55                  int firstSlash = parts[i].find("/"); int secondSlash = parts
                      [i].find("/", firstSlash + 1);
56
57                  if (firstSlash != (secondSlash + 1)) {  // there is texture
                      coordinates.
58                                                          // add code for my
                         texture coordintes here.
59                  }
60              }
61          }
62
63          linesDone++;
64      }
65
66      for (unsigned int i = 0; i < vertexIndices.size(); i += 3) {
67          vertexPositions.push_back(obj_vertices[vertexIndices[i + 0] - 1].x);
68          vertexPositions.push_back(obj_vertices[vertexIndices[i + 0] - 1].y);
69          vertexPositions.push_back(obj_vertices[vertexIndices[i + 0] - 1].z);
70
71          vertexPositions.push_back(obj_vertices[vertexIndices[i + 1] - 1].x);
72          vertexPositions.push_back(obj_vertices[vertexIndices[i + 1] - 1].y);
73          vertexPositions.push_back(obj_vertices[vertexIndices[i + 1] - 1].z);
74
75          vertexPositions.push_back(obj_vertices[vertexIndices[i + 2] - 1].x);
76          vertexPositions.push_back(obj_vertices[vertexIndices[i + 2] - 1].y);
77          vertexPositions.push_back(obj_vertices[vertexIndices[i + 2] - 1].z);
78      }
79  }
80
81
82  void Shapes::Load() {
83      static const char * vs_source[] = { R"(
84  #version 330 core
85
86  in vec4 position;
87  uniform mat4 mv_matrix;
88  uniform mat4 proj_matrix;
```

```
 89
 90 void main(void){
 91     gl_Position = proj_matrix * mv_matrix * position;
 92 }
 93 )" };
 94
 95     static const char * fs_source[] = { R"(
 96 #version 330 core
 97
 98 uniform vec4 inColor;
 99 out vec4 color;
100
101 void main(void){
102     color = inColor;
103 }
104 )" };
105
106     program = glCreateProgram();
107     GLuint fs = glCreateShader(GL_FRAGMENT_SHADER);
108     glShaderSource(fs, 1, fs_source, NULL);
109     glCompileShader(fs);
110     checkErrorShader(fs);
111
112     GLuint vs = glCreateShader(GL_VERTEX_SHADER);
113     glShaderSource(vs, 1, vs_source, NULL);
114     glCompileShader(vs);
115     checkErrorShader(vs);
116
117     glAttachShader(program, vs);
118     glAttachShader(program, fs);
119
120     glLinkProgram(program);
121
122     mv_location = glGetUniformLocation(program, "mv_matrix");
123     proj_location = glGetUniformLocation(program, "proj_matrix");
124     color_location = glGetUniformLocation(program, "inColor");
125
126     glGenVertexArrays(1, &vao);
127     glBindVertexArray(vao);
128
129     glGenBuffers(1, &buffer);
130     glBindBuffer(GL_ARRAY_BUFFER, buffer);
131     glBufferData(GL_ARRAY_BUFFER,
132         vertexPositions.size() * sizeof(GLfloat),
133         &vertexPositions[0],
134         GL_STATIC_DRAW);
135     glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, NULL);
136     glEnableVertexAttribArray(0);
137
138     glLinkProgram(0);   // unlink
139     glDisableVertexAttribArray(0); // Disable
140     glBindVertexArray(0);   // Unbind
141 }
```

```
142
143 void Shapes::Draw() {
144     glUseProgram(program);
145     glBindVertexArray(vao);
146     glEnableVertexAttribArray(0);
147
148     glUniformMatrix4fv(proj_location, 1, GL_FALSE, &proj_matrix[0][0]);
149     glUniformMatrix4fv(mv_location, 1, GL_FALSE, &mv_matrix[0][0]);
150
151     glUniform4f(color_location, fillColor.r, fillColor.g, fillColor.b,
        fillColor.a);
152     glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
153     glDrawArrays(GL_TRIANGLES, 0, vertexPositions.size() / 3);
154
155     glUniform4f(color_location, lineColor.r, lineColor.g, lineColor.b,
        lineColor.a);
156     glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);  glLineWidth(lineWidth);
157     glDrawArrays(GL_TRIANGLES, 0, vertexPositions.size() / 3);
158 }
159
160
161 void Shapes::checkErrorShader(GLuint shader) {
162     // Get log length
163     GLint maxLength;
164     glGetShaderiv(shader, GL_INFO_LOG_LENGTH, &maxLength);
165
166     // Init a string for it
167     std::vector<GLchar> errorLog(maxLength);
168
169     if (maxLength > 1) {
170         // Get the log file
171         glGetShaderInfoLog(shader, maxLength, &maxLength, &errorLog[0]);
172
173         cout << "--------------Shader compilation error-------------\n";
174         cout << errorLog.data();
175     }
176 }
177
178 Cube::Cube() {
179     // Exported from Blender a cube by default (OBJ File)
180     rawData = R"(
181 o Cone
182 v 0.000000 -1.000000 -1.000000
183 v 0.000000 1.000000 0.000000
184 v 0.195090 -1.000000 -0.980785
185 v 0.382683 -1.000000 -0.923880
186 v 0.555570 -1.000000 -0.831470
187 v 0.707107 -1.000000 -0.707107
188 v 0.831470 -1.000000 -0.555570
189 v 0.923880 -1.000000 -0.382683
190 v 0.980785 -1.000000 -0.195090
191 v 1.000000 -1.000000 -0.000000
192 v 0.980785 -1.000000 0.195090
```

```
193  v 0.923880 -1.000000 0.382683
194  v 0.831470 -1.000000 0.555570
195  v 0.707107 -1.000000 0.707107
196  v 0.555570 -1.000000 0.831470
197  v 0.382683 -1.000000 0.923880
198  v 0.195090 -1.000000 0.980785
199  v -0.000000 -1.000000 1.000000
200  v -0.195091 -1.000000 0.980785
201  v -0.382684 -1.000000 0.923879
202  v -0.555571 -1.000000 0.831469
203  v -0.707107 -1.000000 0.707106
204  v -0.831470 -1.000000 0.555570
205  v -0.923880 -1.000000 0.382683
206  v -0.980785 -1.000000 0.195089
207  v -1.000000 -1.000000 -0.000001
208  v -0.980785 -1.000000 -0.195091
209  v -0.923879 -1.000000 -0.382684
210  v -0.831469 -1.000000 -0.555571
211  v -0.707106 -1.000000 -0.707108
212  v -0.555569 -1.000000 -0.831470
213  v -0.382682 -1.000000 -0.923880
214  v -0.195089 -1.000000 -0.980786
215  vn 0.0878 0.4455 -0.8910
216  vn 0.2599 0.4455 -0.8567
217  vn 0.4220 0.4455 -0.7896
218  vn 0.5680 0.4455 -0.6921
219  vn 0.6921 0.4455 -0.5680
220  vn 0.7896 0.4455 -0.4220
221  vn 0.8567 0.4455 -0.2599
222  vn 0.8910 0.4455 -0.0878
223  vn 0.8910 0.4455 0.0878
224  vn 0.8567 0.4455 0.2599
225  vn 0.7896 0.4455 0.4220
226  vn 0.6921 0.4455 0.5680
227  vn 0.5680 0.4455 0.6921
228  vn 0.4220 0.4455 0.7896
229  vn 0.2599 0.4455 0.8567
230  vn 0.0878 0.4455 0.8910
231  vn -0.0878 0.4455 0.8910
232  vn -0.2599 0.4455 0.8567
233  vn -0.4220 0.4455 0.7896
234  vn -0.5680 0.4455 0.6921
235  vn -0.6921 0.4455 0.5680
236  vn -0.7896 0.4455 0.4220
237  vn -0.8567 0.4455 0.2599
238  vn -0.8910 0.4455 0.0878
239  vn -0.8910 0.4455 -0.0878
240  vn -0.8567 0.4455 -0.2599
241  vn -0.7896 0.4455 -0.4220
242  vn -0.6921 0.4455 -0.5680
243  vn -0.5680 0.4455 -0.6921
244  vn -0.4220 0.4455 -0.7896
245  vn -0.2599 0.4455 -0.8567
```

```
246 vn -0.0878 0.4455 -0.8910
247 vn 0.0000 -1.0000 0.0000
248 usemtl None
249 s off
250 f 1//1 2//1 3//1
251 f 3//2 2//2 4//2
252 f 4//3 2//3 5//3
253 f 5//4 2//4 6//4
254 f 6//5 2//5 7//5
255 f 7//6 2//6 8//6
256 f 8//7 2//7 9//7
257 f 9//8 2//8 10//8
258 f 10//9 2//9 11//9
259 f 11//10 2//10 12//10
260 f 12//11 2//11 13//11
261 f 13//12 2//12 14//12
262 f 14//13 2//13 15//13
263 f 15//14 2//14 16//14
264 f 16//15 2//15 17//15
265 f 17//16 2//16 18//16
266 f 18//17 2//17 19//17
267 f 19//18 2//18 20//18
268 f 20//19 2//19 21//19
269 f 21//20 2//20 22//20
270 f 22//21 2//21 23//21
271 f 23//22 2//22 24//22
272 f 24//23 2//23 25//23
273 f 25//24 2//24 26//24
274 f 26//25 2//25 27//25
275 f 27//26 2//26 28//26
276 f 28//27 2//27 29//27
277 f 29//28 2//28 30//28
278 f 30//29 2//29 31//29
279 f 31//30 2//30 32//30
280 f 32//31 2//31 33//31
281 f 33//32 2//32 1//32
282 f 17//33 25//33 9//33
283 f 33//33 1//33 3//33
284 f 3//33 4//33 5//33
285 f 5//33 6//33 7//33
286 f 7//33 8//33 5//33
287 f 9//33 10//33 11//33
288 f 11//33 12//33 9//33
289 f 13//33 14//33 17//33
290 f 15//33 16//33 17//33
291 f 17//33 18//33 19//33
292 f 19//33 20//33 21//33
293 f 21//33 22//33 23//33
294 f 23//33 24//33 25//33
295 f 25//33 26//33 27//33
296 f 27//33 28//33 29//33
297 f 29//33 30//33 33//33
298 f 31//33 32//33 33//33
```

```
f 33//33 3//33 9//33
f 5//33 8//33 9//33
f 9//33 12//33 13//33
f 14//33 15//33 17//33
f 17//33 19//33 25//33
f 21//33 23//33 25//33
f 25//33 27//33 33//33
f 30//33 31//33 33//33
f 3//33 5//33 9//33
f 9//33 13//33 17//33
f 19//33 21//33 25//33
f 27//33 29//33 33//33
f 33//33 9//33 25//33
)";

    LoadObj();
}

Cube::~Cube() {

}

Sphere::Sphere() {

    rawData = R"(
o Sphere
v -0.097545 0.490393 0.000000
v -0.277785 0.415735 0.000000
v -0.415735 0.277785 0.000000
v -0.490393 0.097545 0.000000
v -0.490393 -0.097545 0.000000
v -0.415735 -0.277785 0.000000
v -0.277785 -0.415735 0.000000
v -0.097545 -0.490393 0.000000
v -0.090120 0.490393 -0.037329
v -0.256640 0.415735 -0.106304
v -0.384089 0.277785 -0.159095
v -0.453064 0.097545 -0.187665
v -0.453064 -0.097545 -0.187665
v -0.384089 -0.277785 -0.159095
v -0.256640 -0.415735 -0.106304
v -0.090120 -0.490393 -0.037329
v -0.068975 0.490393 -0.068975
v -0.196424 0.415735 -0.196424
v -0.293969 0.277785 -0.293969
v -0.346760 0.097545 -0.346760
v -0.346760 -0.097545 -0.346760
v -0.293969 -0.277785 -0.293969
v -0.196424 -0.415735 -0.196424
v -0.068975 -0.490393 -0.068975
v -0.037329 0.490393 -0.090120
v -0.106304 0.415735 -0.256640
v -0.159095 0.277785 -0.384089
```

```
352  v -0.187665 0.097545 -0.453064
353  v -0.187665 -0.097545 -0.453064
354  v -0.159095 -0.277785 -0.384089
355  v -0.106304 -0.415735 -0.256640
356  v -0.037329 -0.490393 -0.090120
357  v 0.000000 0.490393 -0.097545
358  v 0.000000 0.415735 -0.277785
359  v 0.000000 0.277785 -0.415735
360  v 0.000000 0.097545 -0.490393
361  v 0.000000 -0.097545 -0.490393
362  v 0.000000 -0.277785 -0.415735
363  v 0.000000 -0.415735 -0.277785
364  v 0.000000 -0.490393 -0.097545
365  v 0.037329 0.490393 -0.090120
366  v 0.106304 0.415735 -0.256640
367  v 0.159095 0.277785 -0.384089
368  v 0.187665 0.097545 -0.453064
369  v 0.187665 -0.097545 -0.453064
370  v 0.159095 -0.277785 -0.384089
371  v 0.106304 -0.415735 -0.256640
372  v 0.037329 -0.490393 -0.090120
373  v 0.068975 0.490393 -0.068975
374  v 0.196424 0.415735 -0.196424
375  v 0.293969 0.277785 -0.293969
376  v 0.346760 0.097545 -0.346760
377  v 0.346760 -0.097545 -0.346760
378  v 0.293969 -0.277785 -0.293969
379  v 0.196424 -0.415735 -0.196424
380  v 0.068975 -0.490393 -0.068975
381  v 0.090120 0.490393 -0.037329
382  v 0.256640 0.415735 -0.106304
383  v 0.384089 0.277785 -0.159095
384  v 0.453064 0.097545 -0.187665
385  v 0.453064 -0.097545 -0.187665
386  v 0.384089 -0.277785 -0.159095
387  v 0.256640 -0.415735 -0.106304
388  v 0.090120 -0.490393 -0.037329
389  v 0.097545 0.490393 0.000000
390  v 0.277785 0.415735 -0.000000
391  v 0.415735 0.277785 0.000000
392  v 0.490393 0.097545 0.000000
393  v 0.490393 -0.097545 0.000000
394  v 0.415735 -0.277785 0.000000
395  v 0.277785 -0.415735 0.000000
396  v 0.097545 -0.490393 -0.000000
397  v 0.090120 0.490393 0.037329
398  v 0.256640 0.415735 0.106304
399  v 0.384089 0.277785 0.159095
400  v 0.453064 0.097545 0.187665
401  v 0.453064 -0.097545 0.187665
402  v 0.384089 -0.277785 0.159095
403  v 0.256640 -0.415735 0.106304
404  v 0.090120 -0.490393 0.037329
```

```
405  v 0.068975 0.490393 0.068975
406  v 0.196424 0.415735 0.196424
407  v 0.293969 0.277785 0.293969
408  v 0.346760 0.097545 0.346760
409  v 0.346760 -0.097545 0.346760
410  v 0.293969 -0.277785 0.293969
411  v 0.196424 -0.415735 0.196424
412  v 0.068975 -0.490393 0.068975
413  v 0.000000 -0.500000 0.000000
414  v 0.037329 0.490393 0.090120
415  v 0.106304 0.415735 0.256640
416  v 0.159095 0.277785 0.384089
417  v 0.187665 0.097545 0.453064
418  v 0.187665 -0.097545 0.453064
419  v 0.159095 -0.277785 0.384089
420  v 0.106304 -0.415735 0.256640
421  v 0.037329 -0.490393 0.090120
422  v 0.000000 0.490393 0.097545
423  v 0.000000 0.415735 0.277785
424  v 0.000000 0.277785 0.415735
425  v 0.000000 0.097545 0.490392
426  v 0.000000 -0.097545 0.490392
427  v 0.000000 -0.277785 0.415735
428  v 0.000000 -0.415735 0.277785
429  v 0.000000 -0.490393 0.097545
430  v -0.037329 0.490393 0.090120
431  v -0.106304 0.415735 0.256640
432  v -0.159095 0.277785 0.384089
433  v -0.187665 0.097545 0.453063
434  v -0.187665 -0.097545 0.453063
435  v -0.159095 -0.277785 0.384089
436  v -0.106304 -0.415735 0.256640
437  v -0.037329 -0.490393 0.090120
438  v -0.068975 0.490393 0.068975
439  v -0.196424 0.415735 0.196424
440  v -0.293969 0.277785 0.293969
441  v -0.346760 0.097545 0.346760
442  v -0.346760 -0.097545 0.346760
443  v -0.293969 -0.277785 0.293969
444  v -0.196423 -0.415735 0.196424
445  v -0.068975 -0.490393 0.068975
446  v 0.000000 0.500000 0.000000
447  v -0.090120 0.490393 0.037329
448  v -0.256640 0.415735 0.106304
449  v -0.384088 0.277785 0.159095
450  v -0.453063 0.097545 0.187665
451  v -0.453063 -0.097545 0.187665
452  v -0.384088 -0.277785 0.159095
453  v -0.256640 -0.415735 0.106304
454  v -0.090120 -0.490393 0.037329
455  s off
456  f 7 14 15
457  f 3 10 11
```

```
458  f 12 3 11
459  f 8 15 16
460  f 5 12 13
461  f 2 125 124
462  f 2 9 10
463  f 6 13 14
464  f 89 8 16
465  f 122 17 9
466  f 7 128 6
467  f 20 27 28
468  f 8 129 7
469  f 22 29 30
470  f 19 26 27
471  f 29 36 37
472  f 31 22 30
473  f 89 16 24
474  f 26 33 34
475  f 24 31 32
476  f 28 35 36
477  f 122 25 17
478  f 27 34 35
479  f 37 44 45
480  f 38 29 37
481  f 89 24 32
482  f 42 33 41
483  f 32 39 40
484  f 36 43 44
485  f 31 38 39
486  f 122 33 25
487  f 43 34 42
488  f 45 52 53
489  f 46 37 45
490  f 89 32 40
491  f 43 50 51
492  f 48 39 47
493  f 52 43 51
494  f 39 46 47
495  f 50 41 49
496  f 122 41 33
497  f 53 60 61
498  f 47 54 55
499  f 46 53 54
500  f 48 55 56
501  f 60 51 59
502  f 58 49 57
503  f 122 49 41
504  f 89 40 48
505  f 61 68 69
506  f 55 62 63
507  f 54 61 62
508  f 51 58 59
509  f 58 65 66
510  f 68 59 67
```

```
511  f 122 57 49
512  f 56 63 64
513  f 89 48 56
514  f 63 70 71
515  f 62 69 70
516  f 59 66 67
517  f 69 76 77
518  f 66 73 74
519  f 122 65 57
520  f 64 71 72
521  f 76 67 75
522  f 89 56 64
523  f 79 70 78
524  f 70 77 78
525  f 67 74 75
526  f 77 84 85
527  f 72 79 80
528  f 122 73 65
529  f 76 83 84
530  f 89 64 72
531  f 74 81 82
532  f 87 78 86
533  f 86 77 85
534  f 75 82 83
535  f 85 93 94
536  f 80 87 88
537  f 84 92 93
538  f 122 81 73
539  f 89 72 80
540  f 91 81 90
541  f 87 95 96
542  f 86 94 95
543  f 83 91 92
544  f 94 101 102
545  f 93 100 101
546  f 89 80 88
547  f 122 90 81
548  f 91 98 99
549  f 88 96 97
550  f 95 102 103
551  f 92 99 100
552  f 102 109 110
553  f 96 103 104
554  f 122 98 90
555  f 89 88 97
556  f 99 106 107
557  f 105 96 104
558  f 109 100 108
559  f 108 99 107
560  f 110 117 118
561  f 104 111 112
562  f 122 106 98
563  f 89 97 105
```

```
564  f 107 114 115
565  f 103 110 111
566  f 117 108 116
567  f 113 104 112
568  f 108 115 116
569  f 118 126 127
570  f 120 111 119
571  f 122 114 106
572  f 115 123 124
573  f 111 118 119
574  f 89 105 113
575  f 113 120 121
576  f 126 116 125
577  f 119 127 128
578  f 116 124 125
579  f 120 128 129
580  f 89 113 121
581  f 121 129 130
582  f 122 123 114
583  f 89 121 130
584  f 122 1 123
585  f 89 130 8
586  f 3 126 125
587  f 5 126 4
588  f 15 22 23
589  f 10 17 18
590  f 24 15 23
591  f 13 20 21
592  f 18 25 26
593  f 14 21 22
594  f 21 28 29
595  f 12 19 20
596  f 11 18 19
597  f 1 124 123
598  f 122 9 1
599  f 6 127 5
600  f 7 6 14
601  f 3 2 10
602  f 12 4 3
603  f 8 7 15
604  f 5 4 12
605  f 2 3 125
606  f 2 1 9
607  f 6 5 13
608  f 7 129 128
609  f 20 19 27
610  f 8 130 129
611  f 22 21 29
612  f 19 18 26
613  f 29 28 36
614  f 31 23 22
615  f 26 25 33
616  f 24 23 31
```

```
617  f 28 27 35
618  f 27 26 34
619  f 37 36 44
620  f 38 30 29
621  f 42 34 33
622  f 32 31 39
623  f 36 35 43
624  f 31 30 38
625  f 43 35 34
626  f 45 44 52
627  f 46 38 37
628  f 43 42 50
629  f 48 40 39
630  f 52 44 43
631  f 39 38 46
632  f 50 42 41
633  f 53 52 60
634  f 47 46 54
635  f 46 45 53
636  f 48 47 55
637  f 60 52 51
638  f 58 50 49
639  f 61 60 68
640  f 55 54 62
641  f 54 53 61
642  f 51 50 58
643  f 58 57 65
644  f 68 60 59
645  f 56 55 63
646  f 63 62 70
647  f 62 61 69
648  f 59 58 66
649  f 69 68 76
650  f 66 65 73
651  f 64 63 71
652  f 76 68 67
653  f 79 71 70
654  f 70 69 77
655  f 67 66 74
656  f 77 76 84
657  f 72 71 79
658  f 76 75 83
659  f 74 73 81
660  f 87 79 78
661  f 86 78 77
662  f 75 74 82
663  f 85 84 93
664  f 80 79 87
665  f 84 83 92
666  f 91 82 81
667  f 87 86 95
668  f 86 85 94
669  f 83 82 91
```

```
670  f 94 93 101
671  f 93 92 100
672  f 91 90 98
673  f 88 87 96
674  f 95 94 102
675  f 92 91 99
676  f 102 101 109
677  f 96 95 103
678  f 99 98 106
679  f 105 97 96
680  f 109 101 100
681  f 108 100 99
682  f 110 109 117
683  f 104 103 111
684  f 107 106 114
685  f 103 102 110
686  f 117 109 108
687  f 113 105 104
688  f 108 107 115
689  f 118 117 126
690  f 120 112 111
691  f 115 114 123
692  f 111 110 118
693  f 113 112 120
694  f 126 117 116
695  f 119 118 127
696  f 116 115 124
697  f 120 119 128
698  f 121 120 129
699  f 3 4 126
700  f 5 127 126
701  f 15 14 22
702  f 10 9 17
703  f 24 16 15
704  f 13 12 20
705  f 18 17 25
706  f 14 13 21
707  f 21 20 28
708  f 12 11 19
709  f 11 10 18
710  f 1 2 124
711  f 6 128 127
712
713  )";
714
715      LoadObj();
716  }
717
718  Sphere::~Sphere() {
719
720  }
721
722  Arrow::Arrow() {
```

```
723
724     rawData = R"(
725  o Cone
726  v 0.000000 0.800000 -0.100000
727  v 0.070711 0.800000 -0.070711
728  v 0.100000 0.800000 -0.000000
729  v 0.000000 1.000000 0.000000
730  v 0.070711 0.800000 0.070711
731  v -0.000000 0.800000 0.100000
732  v -0.070711 0.800000 0.070711
733  v -0.100000 0.800000 -0.000000
734  v -0.070711 0.800000 -0.070711
735  s off
736  f 4 7 6
737  f 5 7 2
738  f 4 8 7
739  f 3 4 5
740  f 5 4 6
741  f 4 9 8
742  f 4 1 9
743  f 2 1 4
744  f 2 4 3
745  f 9 1 2
746  f 2 3 5
747  f 5 6 7
748  f 7 8 9
749  f 9 2 7
750  o Cylinder
751  v 0.000000 0.000000 -0.050000
752  v 0.009755 0.900000 -0.049039
753  v 0.019134 0.000000 -0.046194
754  v 0.027779 0.900000 -0.041573
755  v 0.035355 0.000000 -0.035355
756  v 0.041573 0.900000 -0.027779
757  v 0.046194 0.000000 -0.019134
758  v 0.049039 0.900000 -0.009755
759  v 0.050000 0.000000 -0.000000
760  v 0.049039 0.900000 0.009755
761  v 0.046194 0.000000 0.019134
762  v 0.041573 0.900000 0.027779
763  v 0.035355 0.000000 0.035355
764  v 0.027779 0.900000 0.041573
765  v 0.019134 0.000000 0.046194
766  v 0.009755 0.900000 0.049039
767  v -0.000000 0.000000 0.050000
768  v -0.009755 0.900000 0.049039
769  v -0.019134 0.000000 0.046194
770  v -0.027779 0.900000 0.041573
771  v -0.035355 0.000000 0.035355
772  v -0.041574 0.900000 0.027778
773  v -0.046194 0.000000 0.019134
774  v -0.049039 0.900000 0.009754
775  v -0.050000 0.000000 -0.000000
```

```
776 v -0.049039 0.900000 -0.009755
777 v -0.046194 0.000000 -0.019134
778 v -0.041573 0.900000 -0.027779
779 v -0.035355 0.000000 -0.035355
780 v -0.027778 0.900000 -0.041574
781 v -0.019134 0.000000 -0.046194
782 v -0.009754 0.900000 -0.049039
783 s off
784 f 13 15 14
785 f 16 14 15
786 f 17 19 18
787 f 18 16 17
788 f 19 21 20
789 f 20 18 19
790 f 21 23 22
791 f 22 20 21
792 f 23 25 24
793 f 24 22 23
794 f 25 27 26
795 f 26 24 25
796 f 27 29 28
797 f 28 26 27
798 f 29 31 30
799 f 30 28 29
800 f 31 33 32
801 f 32 30 31
802 f 33 35 34
803 f 34 32 33
804 f 35 37 36
805 f 36 34 35
806 f 37 39 38
807 f 38 36 37
808 f 41 40 39
809 f 40 38 39
810 f 41 10 40
811 f 29 21 37
812 f 11 12 10
813 f 24 32 16
814 f 15 17 16
815 f 11 13 12
816 f 14 12 13
817 f 10 41 11
818 f 13 11 41
819 f 41 39 37
820 f 37 35 33
821 f 33 31 29
822 f 29 27 25
823 f 25 23 29
824 f 21 19 17
825 f 17 15 13
826 f 13 41 37
827 f 37 33 29
828 f 29 23 21
```

```
f 21 17 13
f 13 37 21
f 40 10 12
f 12 14 16
f 16 18 20
f 20 22 24
f 24 26 28
f 28 30 32
f 32 34 36
f 36 38 40
f 40 12 16
f 16 20 24
f 24 28 32
f 32 36 40
f 40 16 32
)";

    LoadObj();
}

Arrow::~Arrow() {

}
```