

```
1 #include "shapes.h"
2 #include <iostream>
3 #include <sstream>
4
5 #include <GL/glew.h>
6 #include <GLFW/glfw3.h>
7 #include <glm/glm.hpp>
8
9 Shapes::Shapes() {
10
11 };
12
13 Shapes::~Shapes() {
14
15 }
16
17 void Shapes::LoadObj() {
18
19     std::vector< glm::vec3 > obj_vertices;
20     std::vector< unsigned int > vertexIndices;
21     istreamstream rawDataStream(rawData);
22     string dataLine; int linesDone = 0;
23
24     while (std::getline(rawDataStream, dataLine)) {
25         if (dataLine.find("v ") != string::npos) { // does this line have a
26             vector?
27             glm::vec3 vertex;
28
29             int foundStart = dataLine.find(" "); int foundEnd =
30                 dataLine.find(" ", foundStart + 1);
31             vertex.x = stof(dataLine.substr(foundStart, foundEnd -
32                 foundStart));
33
34             foundStart = foundEnd; foundEnd = dataLine.find(" ", foundStart
35                 + 1);
36             vertex.y = stof(dataLine.substr(foundStart, foundEnd -
37                 foundStart));
38
39             foundStart = foundEnd; foundEnd = dataLine.find(" ", foundStart
40                 + 1);
41             vertex.z = stof(dataLine.substr(foundStart, foundEnd -
42                 foundStart));
43
44             obj_vertices.push_back(vertex);
45         }
46         else if (dataLine.find("f ") != string::npos) { // does this line
47             defines a triangle face?
48             string parts[3];
49
50             int foundStart = dataLine.find(" "); int foundEnd =
51                 dataLine.find(" ", foundStart + 1);
52             parts[0] = dataLine.substr(foundStart + 1, foundEnd - foundStart
53                 - 1);
```

```

44
45         foundStart = foundEnd; foundEnd = dataLine.find(" ", foundStart
46             + 1);
47         parts[1] = dataLine.substr(foundStart + 1, foundEnd - foundStart
48             - 1);
49
50         foundStart = foundEnd; foundEnd = dataLine.find(" ", foundStart
51             + 1);
52         parts[2] = dataLine.substr(foundStart + 1, foundEnd - foundStart
53             - 1);
54
55         for (int i = 0; i < 3; i++) {           // for each part
56
57             vertexIndices.push_back(stoul(parts[i].substr(0, parts
58                 [i].find("/"))));
59
60             int firstSlash = parts[i].find("/"); int secondSlash = parts
61                 [i].find("/", firstSlash + 1);
62
63             if (firstSlash != (secondSlash + 1)) { // there is texture
64                 coordinates.
65
66                 // add code for my
67                 texture coordintes here.
68             }
69         }
70
71         linesDone++;
72     }
73
74     for (unsigned int i = 0; i < vertexIndices.size(); i += 3) {
75         vertexPositions.push_back(obj_vertices[vertexIndices[i + 0] - 1].x);
76         vertexPositions.push_back(obj_vertices[vertexIndices[i + 0] - 1].y);
77         vertexPositions.push_back(obj_vertices[vertexIndices[i + 0] - 1].z);
78
79         vertexPositions.push_back(obj_vertices[vertexIndices[i + 1] - 1].x);
80         vertexPositions.push_back(obj_vertices[vertexIndices[i + 1] - 1].y);
81         vertexPositions.push_back(obj_vertices[vertexIndices[i + 1] - 1].z);
82
83         vertexPositions.push_back(obj_vertices[vertexIndices[i + 2] - 1].x);
84         vertexPositions.push_back(obj_vertices[vertexIndices[i + 2] - 1].y);
85         vertexPositions.push_back(obj_vertices[vertexIndices[i + 2] - 1].z);
86     }
87 }
88
89 void Shapes::Load() {
90     static const char * vs_source[] = { R"(
91 #version 330 core
92
93 in vec4 position;
94 uniform mat4 mv_matrix;
95 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 Plane
182 v -1.000000 0.000000 1.000000
183 v 1.000000 0.000000 1.000000
184 v -1.000000 0.000000 -1.000000
185 v 1.000000 0.000000 -1.000000
186 vn 0.0000 1.0000 0.0000
187 usemtl None
188 s off
189 f 2 3 1
190 f 2 4 3
191 )";
192

```

```
193     LoadObj();
194 }
195
196 Cube::~Cube() {
197 }
198 }
199
200 Sphere::Sphere() {
201
202     rawData = R"(
203 o Sphere
204 v -0.097545 0.490393 0.000000
205 v -0.277785 0.415735 0.000000
206 v -0.415735 0.277785 0.000000
207 v -0.490393 0.097545 0.000000
208 v -0.490393 -0.097545 0.000000
209 v -0.415735 -0.277785 0.000000
210 v -0.277785 -0.415735 0.000000
211 v -0.097545 -0.490393 0.000000
212 v -0.090120 0.490393 -0.037329
213 v -0.256640 0.415735 -0.106304
214 v -0.384089 0.277785 -0.159095
215 v -0.453064 0.097545 -0.187665
216 v -0.453064 -0.097545 -0.187665
217 v -0.384089 -0.277785 -0.159095
218 v -0.256640 -0.415735 -0.106304
219 v -0.090120 -0.490393 -0.037329
220 v -0.068975 0.490393 -0.068975
221 v -0.196424 0.415735 -0.196424
222 v -0.293969 0.277785 -0.293969
223 v -0.346760 0.097545 -0.346760
224 v -0.346760 -0.097545 -0.346760
225 v -0.293969 -0.277785 -0.293969
226 v -0.196424 -0.415735 -0.196424
227 v -0.068975 -0.490393 -0.068975
228 v -0.037329 0.490393 -0.090120
229 v -0.106304 0.415735 -0.256640
230 v -0.159095 0.277785 -0.384089
231 v -0.187665 0.097545 -0.453064
232 v -0.187665 -0.097545 -0.453064
233 v -0.159095 -0.277785 -0.384089
234 v -0.106304 -0.415735 -0.256640
235 v -0.037329 -0.490393 -0.090120
236 v 0.000000 0.490393 -0.097545
237 v 0.000000 0.415735 -0.277785
238 v 0.000000 0.277785 -0.415735
239 v 0.000000 0.097545 -0.490393
240 v 0.000000 -0.097545 -0.490393
241 v 0.000000 -0.277785 -0.415735
242 v 0.000000 -0.415735 -0.277785
243 v 0.000000 -0.490393 -0.097545
244 v 0.037329 0.490393 -0.090120
245 v 0.106304 0.415735 -0.256640
```

```
246 v 0.159095 0.277785 -0.384089
247 v 0.187665 0.097545 -0.453064
248 v 0.187665 -0.097545 -0.453064
249 v 0.159095 -0.277785 -0.384089
250 v 0.106304 -0.415735 -0.256640
251 v 0.037329 -0.490393 -0.090120
252 v 0.068975 0.490393 -0.068975
253 v 0.196424 0.415735 -0.196424
254 v 0.293969 0.277785 -0.293969
255 v 0.346760 0.097545 -0.346760
256 v 0.346760 -0.097545 -0.346760
257 v 0.293969 -0.277785 -0.293969
258 v 0.196424 -0.415735 -0.196424
259 v 0.068975 -0.490393 -0.068975
260 v 0.090120 0.490393 -0.037329
261 v 0.256640 0.415735 -0.106304
262 v 0.384089 0.277785 -0.159095
263 v 0.453064 0.097545 -0.187665
264 v 0.453064 -0.097545 -0.187665
265 v 0.384089 -0.277785 -0.159095
266 v 0.256640 -0.415735 -0.106304
267 v 0.090120 -0.490393 -0.037329
268 v 0.097545 0.490393 0.000000
269 v 0.277785 0.415735 -0.000000
270 v 0.415735 0.277785 0.000000
271 v 0.490393 0.097545 0.000000
272 v 0.490393 -0.097545 0.000000
273 v 0.415735 -0.277785 0.000000
274 v 0.277785 -0.415735 0.000000
275 v 0.097545 -0.490393 -0.000000
276 v 0.090120 0.490393 0.037329
277 v 0.256640 0.415735 0.106304
278 v 0.384089 0.277785 0.159095
279 v 0.453064 0.097545 0.187665
280 v 0.453064 -0.097545 0.187665
281 v 0.384089 -0.277785 0.159095
282 v 0.256640 -0.415735 0.106304
283 v 0.090120 -0.490393 0.037329
284 v 0.068975 0.490393 0.068975
285 v 0.196424 0.415735 0.196424
286 v 0.293969 0.277785 0.293969
287 v 0.346760 0.097545 0.346760
288 v 0.346760 -0.097545 0.346760
289 v 0.293969 -0.277785 0.293969
290 v 0.196424 -0.415735 0.196424
291 v 0.068975 -0.490393 0.068975
292 v 0.000000 -0.500000 0.000000
293 v 0.037329 0.490393 0.090120
294 v 0.106304 0.415735 0.256640
295 v 0.159095 0.277785 0.384089
296 v 0.187665 0.097545 0.453064
297 v 0.187665 -0.097545 0.453064
298 v 0.159095 -0.277785 0.384089
```

```
299 v 0.106304 -0.415735 0.256640
300 v 0.037329 -0.490393 0.090120
301 v 0.000000 0.490393 0.097545
302 v 0.000000 0.415735 0.277785
303 v 0.000000 0.277785 0.415735
304 v 0.000000 0.097545 0.490392
305 v 0.000000 -0.097545 0.490392
306 v 0.000000 -0.277785 0.415735
307 v 0.000000 -0.415735 0.277785
308 v 0.000000 -0.490393 0.097545
309 v -0.037329 0.490393 0.090120
310 v -0.106304 0.415735 0.256640
311 v -0.159095 0.277785 0.384089
312 v -0.187665 0.097545 0.453063
313 v -0.187665 -0.097545 0.453063
314 v -0.159095 -0.277785 0.384089
315 v -0.106304 -0.415735 0.256640
316 v -0.037329 -0.490393 0.090120
317 v -0.068975 0.490393 0.068975
318 v -0.196424 0.415735 0.196424
319 v -0.293969 0.277785 0.293969
320 v -0.346760 0.097545 0.346760
321 v -0.346760 -0.097545 0.346760
322 v -0.293969 -0.277785 0.293969
323 v -0.196423 -0.415735 0.196424
324 v -0.068975 -0.490393 0.068975
325 v 0.000000 0.500000 0.000000
326 v -0.090120 0.490393 0.037329
327 v -0.256640 0.415735 0.106304
328 v -0.384088 0.277785 0.159095
329 v -0.453063 0.097545 0.187665
330 v -0.453063 -0.097545 0.187665
331 v -0.384088 -0.277785 0.159095
332 v -0.256640 -0.415735 0.106304
333 v -0.090120 -0.490393 0.037329
334 s off
335 f 7 14 15
336 f 3 10 11
337 f 12 3 11
338 f 8 15 16
339 f 5 12 13
340 f 2 125 124
341 f 2 9 10
342 f 6 13 14
343 f 89 8 16
344 f 122 17 9
345 f 7 128 6
346 f 20 27 28
347 f 8 129 7
348 f 22 29 30
349 f 19 26 27
350 f 29 36 37
351 f 31 22 30
```

```
352 f 89 16 24
353 f 26 33 34
354 f 24 31 32
355 f 28 35 36
356 f 122 25 17
357 f 27 34 35
358 f 37 44 45
359 f 38 29 37
360 f 89 24 32
361 f 42 33 41
362 f 32 39 40
363 f 36 43 44
364 f 31 38 39
365 f 122 33 25
366 f 43 34 42
367 f 45 52 53
368 f 46 37 45
369 f 89 32 40
370 f 43 50 51
371 f 48 39 47
372 f 52 43 51
373 f 39 46 47
374 f 50 41 49
375 f 122 41 33
376 f 53 60 61
377 f 47 54 55
378 f 46 53 54
379 f 48 55 56
380 f 60 51 59
381 f 58 49 57
382 f 122 49 41
383 f 89 40 48
384 f 61 68 69
385 f 55 62 63
386 f 54 61 62
387 f 51 58 59
388 f 58 65 66
389 f 68 59 67
390 f 122 57 49
391 f 56 63 64
392 f 89 48 56
393 f 63 70 71
394 f 62 69 70
395 f 59 66 67
396 f 69 76 77
397 f 66 73 74
398 f 122 65 57
399 f 64 71 72
400 f 76 67 75
401 f 89 56 64
402 f 79 70 78
403 f 70 77 78
404 f 67 74 75
```



```
405 f 77 84 85
406 f 72 79 80
407 f 122 73 65
408 f 76 83 84
409 f 89 64 72
410 f 74 81 82
411 f 87 78 86
412 f 86 77 85
413 f 75 82 83
414 f 85 93 94
415 f 80 87 88
416 f 84 92 93
417 f 122 81 73
418 f 89 72 80
419 f 91 81 90
420 f 87 95 96
421 f 86 94 95
422 f 83 91 92
423 f 94 101 102
424 f 93 100 101
425 f 89 80 88
426 f 122 90 81
427 f 91 98 99
428 f 88 96 97
429 f 95 102 103
430 f 92 99 100
431 f 102 109 110
432 f 96 103 104
433 f 122 98 90
434 f 89 88 97
435 f 99 106 107
436 f 105 96 104
437 f 109 100 108
438 f 108 99 107
439 f 110 117 118
440 f 104 111 112
441 f 122 106 98
442 f 89 97 105
443 f 107 114 115
444 f 103 110 111
445 f 117 108 116
446 f 113 104 112
447 f 108 115 116
448 f 118 126 127
449 f 120 111 119
450 f 122 114 106
451 f 115 123 124
452 f 111 118 119
453 f 89 105 113
454 f 113 120 121
455 f 126 116 125
456 f 119 127 128
457 f 116 124 125
```

```
458 f 120 128 129
459 f 89 113 121
460 f 121 129 130
461 f 122 123 114
462 f 89 121 130
463 f 122 1 123
464 f 89 130 8
465 f 3 126 125
466 f 5 126 4
467 f 15 22 23
468 f 10 17 18
469 f 24 15 23
470 f 13 20 21
471 f 18 25 26
472 f 14 21 22
473 f 21 28 29
474 f 12 19 20
475 f 11 18 19
476 f 1 124 123
477 f 122 9 1
478 f 6 127 5
479 f 7 6 14
480 f 3 2 10
481 f 12 4 3
482 f 8 7 15
483 f 5 4 12
484 f 2 3 125
485 f 2 1 9
486 f 6 5 13
487 f 7 129 128
488 f 20 19 27
489 f 8 130 129
490 f 22 21 29
491 f 19 18 26
492 f 29 28 36
493 f 31 23 22
494 f 26 25 33
495 f 24 23 31
496 f 28 27 35
497 f 27 26 34
498 f 37 36 44
499 f 38 30 29
500 f 42 34 33
501 f 32 31 39
502 f 36 35 43
503 f 31 30 38
504 f 43 35 34
505 f 45 44 52
506 f 46 38 37
507 f 43 42 50
508 f 48 40 39
509 f 52 44 43
510 f 39 38 46
```

```
511 f 50 42 41
512 f 53 52 60
513 f 47 46 54
514 f 46 45 53
515 f 48 47 55
516 f 60 52 51
517 f 58 50 49
518 f 61 60 68
519 f 55 54 62
520 f 54 53 61
521 f 51 50 58
522 f 58 57 65
523 f 68 60 59
524 f 56 55 63
525 f 63 62 70
526 f 62 61 69
527 f 59 58 66
528 f 69 68 76
529 f 66 65 73
530 f 64 63 71
531 f 76 68 67
532 f 79 71 70
533 f 70 69 77
534 f 67 66 74
535 f 77 76 84
536 f 72 71 79
537 f 76 75 83
538 f 74 73 81
539 f 87 79 78
540 f 86 78 77
541 f 75 74 82
542 f 85 84 93
543 f 80 79 87
544 f 84 83 92
545 f 91 82 81
546 f 87 86 95
547 f 86 85 94
548 f 83 82 91
549 f 94 93 101
550 f 93 92 100
551 f 91 90 98
552 f 88 87 96
553 f 95 94 102
554 f 92 91 99
555 f 102 101 109
556 f 96 95 103
557 f 99 98 106
558 f 105 97 96
559 f 109 101 100
560 f 108 100 99
561 f 110 109 117
562 f 104 103 111
563 f 107 106 114
```

```
564 f 103 102 110
565 f 117 109 108
566 f 113 105 104
567 f 108 107 115
568 f 118 117 126
569 f 120 112 111
570 f 115 114 123
571 f 111 110 118
572 f 113 112 120
573 f 126 117 116
574 f 119 118 127
575 f 116 115 124
576 f 120 119 128
577 f 121 120 129
578 f 3 4 126
579 f 5 127 126
580 f 15 14 22
581 f 10 9 17
582 f 24 16 15
583 f 13 12 20
584 f 18 17 25
585 f 14 13 21
586 f 21 20 28
587 f 12 11 19
588 f 11 10 18
589 f 1 2 124
590 f 6 128 127
591
592 );
593
594     LoadObj();
595 }
596
597 Sphere::~Sphere() {
598 }
599 }
600
601 Arrow::Arrow() {
602
603     rawData = R"(
604 o Cone
605 v 0.000000 0.800000 -0.100000
606 v 0.070711 0.800000 -0.070711
607 v 0.100000 0.800000 -0.000000
608 v 0.000000 1.000000 0.000000
609 v 0.070711 0.800000 0.070711
610 v -0.000000 0.800000 0.100000
611 v -0.070711 0.800000 0.070711
612 v -0.100000 0.800000 -0.000000
613 v -0.070711 0.800000 -0.070711
614 s off
615 f 4 7 6
616 f 5 7 2
```

```
617 f 4 8 7
618 f 3 4 5
619 f 5 4 6
620 f 4 9 8
621 f 4 1 9
622 f 2 1 4
623 f 2 4 3
624 f 9 1 2
625 f 2 3 5
626 f 5 6 7
627 f 7 8 9
628 f 9 2 7
629 o Cylinder
630 v 0.000000 0.000000 -0.050000
631 v 0.009755 0.900000 -0.049039
632 v 0.019134 0.000000 -0.046194
633 v 0.027779 0.900000 -0.041573
634 v 0.035355 0.000000 -0.035355
635 v 0.041573 0.900000 -0.027779
636 v 0.046194 0.000000 -0.019134
637 v 0.049039 0.900000 -0.009755
638 v 0.050000 0.000000 -0.000000
639 v 0.049039 0.900000 0.009755
640 v 0.046194 0.000000 0.019134
641 v 0.041573 0.900000 0.027779
642 v 0.035355 0.000000 0.035355
643 v 0.027779 0.900000 0.041573
644 v 0.019134 0.000000 0.046194
645 v 0.009755 0.900000 0.049039
646 v -0.000000 0.000000 0.050000
647 v -0.009755 0.900000 0.049039
648 v -0.019134 0.000000 0.046194
649 v -0.027779 0.900000 0.041573
650 v -0.035355 0.000000 0.035355
651 v -0.041574 0.900000 0.027778
652 v -0.046194 0.000000 0.019134
653 v -0.049039 0.900000 0.009754
654 v -0.050000 0.000000 -0.000000
655 v -0.049039 0.900000 -0.009755
656 v -0.046194 0.000000 -0.019134
657 v -0.041573 0.900000 -0.027779
658 v -0.035355 0.000000 -0.035355
659 v -0.027778 0.900000 -0.041574
660 v -0.019134 0.000000 -0.046194
661 v -0.009754 0.900000 -0.049039
662 s off
663 f 13 15 14
664 f 16 14 15
665 f 17 19 18
666 f 18 16 17
667 f 19 21 20
668 f 20 18 19
669 f 21 23 22
```

```
670 f 22 20 21
671 f 23 25 24
672 f 24 22 23
673 f 25 27 26
674 f 26 24 25
675 f 27 29 28
676 f 28 26 27
677 f 29 31 30
678 f 30 28 29
679 f 31 33 32
680 f 32 30 31
681 f 33 35 34
682 f 34 32 33
683 f 35 37 36
684 f 36 34 35
685 f 37 39 38
686 f 38 36 37
687 f 41 40 39
688 f 40 38 39
689 f 41 10 40
690 f 29 21 37
691 f 11 12 10
692 f 24 32 16
693 f 15 17 16
694 f 11 13 12
695 f 14 12 13
696 f 10 41 11
697 f 13 11 41
698 f 41 39 37
699 f 37 35 33
700 f 33 31 29
701 f 29 27 25
702 f 25 23 29
703 f 21 19 17
704 f 17 15 13
705 f 13 41 37
706 f 37 33 29
707 f 29 23 21
708 f 21 17 13
709 f 13 37 21
710 f 40 10 12
711 f 12 14 16
712 f 16 18 20
713 f 20 22 24
714 f 24 26 28
715 f 28 30 32
716 f 32 34 36
717 f 36 38 40
718 f 40 12 16
719 f 16 20 24
720 f 24 28 32
721 f 32 36 40
722 f 40 16 32
```

```
723 );  
724  
725     LoadObj();  
726 }  
727  
728 Arrow::~Arrow() {  
729  
730 }
```