```
 1: // $Id: hello-world.cpp,v 1.24 2019-02-21 17:38:38-08 - - $
 2:
 3: #include <cmath>
 4: #include <iostream>
 5: #include <memory>
 6: #include <string>
 7: #include <vector>
 8: using namespace std;
 9:
10: #include <GL/freeglut.h>
11: #include <libgen.h>
12:
13: // Characteristics of the window.
14: struct window {
15:    string name;
16:    int width {384};
17:    int height {256};
18: } window;
19:
20: template <typename item_t>
21: struct cycle_iterator {
22:    const vector<item_t> items;
23:    size_t curr_item {0};
24:    cycle_iterator (initializer_list<item_t> list):
25:                items(vector<item_t> (list)) {}
26:    const item_t& operator*() { return items[curr_item]; }
27:    const item_t& operator[] (size_t index) {
28:       return items[(curr_item + index) % items.size()];
29:    }
30:    cycle_iterator& operator++() {
31:       curr_item = (curr_item + 1) % items.size();
32:       return *this;
33:    }
34: };
35:
36: cycle_iterator<string> greetings {
37:    "Hello, World!",
38:    "Hodie natus est radici frater.",
39:    "Goodbye, World!",
40: };
41:
42: GLubyte RED[] = {0xFF, 0x00, 0x00};
43: GLubyte GREEN[] = {0x00, 0xFF, 0x00};
44: GLubyte BLUE[] = {0x00, 0x00, 0xFF};
45: cycle_iterator<GLubyte*> colors {RED, GREEN, BLUE};
46:
47: cycle_iterator<void*> glut_fonts {
48:    GLUT_BITMAP_TIMES_ROMAN_24,
49:    GLUT_BITMAP_HELVETICA_18,
50:    GLUT_BITMAP_9_BY_15,
51: };
52:
```

```
 53:
 54: void draw_rectangle (const GLubyte* color) {
 55:    glBegin (GL_POLYGON);
 56:    glColor3ubv (color);
 57:    glVertex2f (0, 0);
 58:    glVertex2f (window.width, 0);
 59:    glVertex2f (window.width, window.height);
 60:    glVertex2f (0, window.height);
 61:    glEnd();
 62: }
 63:
 64: void draw_ellipse (const GLubyte* color) {
 65:    glBegin (GL_POLYGON);
 66:    glColor3ubv (color);
 67:    const GLfloat delta = 2 * M_PI / 64;
 68:    const GLfloat width = window.width / 2.5;
 69:    const GLfloat height = window.height / 2.5;
 70:    const GLfloat xoffset = window.width / 2.0;
 71:    const GLfloat yoffset = window.height / 2.0;
 72:    for (GLfloat theta = 0; theta < 2 * M_PI; theta += delta) {
 73:       GLfloat xpos = width * cos (theta) + xoffset;
 74:       GLfloat ypos = height * sin (theta) + yoffset;
 75:       glVertex2f (xpos, ypos);
 76:    }
 77:    glEnd();
 78: }
 79:
 80: void draw_greeting (const string& greeting, const GLubyte* color) {
 81:    const GLubyte* glgreeting =
 82:          reinterpret_cast<const GLubyte*> (greeting.c_str());
 83:
 84:    void* font = *glut_fonts;
 85:    int greeting_width = glutBitmapLength (font, glgreeting);
 86:    int greeting_height = glutBitmapHeight (font);
 87:
 88:    glColor3ubv (color);
 89:
 90:    float xpos = window.width / 2.0 - greeting_width / 2.0;
 91:    float ypos = window.height / 2.0 - greeting_height / 4.0;
 92:    glRasterPos2f (xpos, ypos);
 93:
 94:    glutBitmapString (font, glgreeting);
 95: }
 96:
 97: void display() {
 98:    glClearColor (0.15, 0.15, 0.15, 1.0);
 99:    glClear (GL_COLOR_BUFFER_BIT);
100:
101:    draw_rectangle (colors[0]);
102:    draw_ellipse (colors[1]);
103:    draw_greeting (*greetings, colors[2]);
104:
105:    glutSwapBuffers();
106: }
107:
```

```
108:
109: void invert_colors() {
110:    for (size_t color = 0; color < 3; ++color) {
111:       for (size_t rgb = 0; rgb < 3; ++rgb) {
112:          colors[color][rgb] ^= 0xFF;
113:       }
114:    }
115: }
116:
117: void keyboard (GLubyte key, int, int) {
118:    enum {BS = 8, LF = 10, CR = 13, ESC = 27, DEL = 127};
119:    switch (key) {
120:       case ' ': case BS: case CR: case LF: case DEL:
121:          invert_colors();
122:          break;
123:       case 'c': case 'C':
124:          ++colors;
125:          break;
126:       case 'f': case 'F':
127:          ++glut_fonts;
128:          break;
129:       case 'g': case 'G':
130:          ++greetings;
131:          break;
132:       case 'q': case 'Q': case ESC:
133:          exit (EXIT_SUCCESS);
134:    }
135:    glutPostRedisplay();
136: }
137:
138: void mouse (int button, int state, int, int) {
139:    if (state == GLUT_DOWN) {
140:       switch (button) {
141:          case GLUT_LEFT_BUTTON:
142:             ++glut_fonts;
143:             break;
144:          case GLUT_MIDDLE_BUTTON:
145:             ++greetings;
146:             break;
147:          case GLUT_RIGHT_BUTTON:
148:             ++colors;
149:             break;
150:       }
151:    }
152:    glutPostRedisplay();
153: }
154:
```

```
155:
156: void reshape (int width, int height) {
157:    window.width = width;
158:    window.height = height;
159:    glMatrixMode (GL_PROJECTION);
160:    glLoadIdentity();
161:    gluOrtho2D (0, window.width, 0, window.height);
162:    glMatrixMode (GL_MODELVIEW);
163:    glViewport (0, 0, window.width, window.height);
164:    glutPostRedisplay();
165: }
166:
167: void print_howto() {
168: cout << R"(
169: To cycle the colors: right mouse button or key 'c' or 'C'.
170: To cycle the fonts: left mouse button or key 'f' or 'F'.
171: To cycle the greetings: middle mouse button or key 'g' or 'G'.
172: To invert the colors: key Space, Backspace, Return, or Delete.
173: To quit: key 'q' or 'Q' or ESC.
174: )";
175: }
176:
177: int main (int argc, char** argv) {
178:    print_howto();
179:    window.name = basename (argv[0]);
180:    glutInit (&argc, argv);
181:    glutInitDisplayMode (GLUT_RGBA | GLUT_DOUBLE);
182:    glutInitWindowSize (window.width, window.height);
183:    glutCreateWindow (window.name.c_str());
184:    glutDisplayFunc (display);
185:    glutReshapeFunc (reshape);
186:    glutKeyboardFunc (keyboard);
187:    glutMouseFunc (mouse);
188:    glutMainLoop();
189:    return 0;
190: }
191:
```

```
 1: // $Id: bonjour-le-monde.cpp,v 1.9 2018-06-27 15:49:48-07 - - $
 2:
 3: // Draw line from (0,0) to (1,1).
 4:
 5: #include <cmath>
 6: #include <iostream>
 7: #include <vector>
 8: using namespace std;
 9:
10: #include <GL/freeglut.h>
11: #include <libgen.h>
12:
13: struct {
14:    int width {384};
15:    int height {256};
16: } window;
17:
18: const GLubyte BLEU[] {0, 85, 164};
19: const GLubyte BLANC[] {255, 255, 255};
20: const GLubyte ROUGE[] {239, 65, 53};
21: const vector<const GLubyte*> colors {BLEU, BLANC, ROUGE};
22: constexpr GLfloat aspect_ratio = 2.0 / 3.0;
23:
24: void draw_french_flag() {
25:    for (size_t i = 0; i < 3; ++i) {
26:       glBegin (GL_POLYGON);
27:       glColor3ubv (colors[i]);
28:       glVertex2f (window.width * i / 3.0, 0);
29:       glVertex2f (window.width * (i + 1) / 3.0, 0);
30:       glVertex2f (window.width * (i + 1) / 3.0, window.height);
31:       glVertex2f (window.width * i / 3.0, window.height);
32:       glEnd();
33:    }
34: }
35:
36: void display() {
37:    glClearColor (0.0, 0.0, 0.0, 0.0);
38:    glClear (GL_COLOR_BUFFER_BIT);
39:    draw_french_flag();
40:    glutSwapBuffers();
41: }
42:
```

```
43:
44: void adjust_aspect (int width, int height) {
45:    if (window.width != width) {
46:       height = width * aspect_ratio;
47:    }else if (window.height != height) {
48:       width = height / aspect_ratio;
49:    }else {
50:       return;
51:    }
52:    window.width = width;
53:    window.height = height;
54:    glutReshapeWindow (window.width, window.height);
55: }
56:
57: void reshape (int width, int height) {
58:    adjust_aspect (width, height);
59:    window.width = width;
60:    window.height = height;
61:    glMatrixMode (GL_PROJECTION);
62:    glLoadIdentity();
63:    gluOrtho2D (0, window.width, 0, window.height);
64:    glMatrixMode (GL_MODELVIEW);
65:    glViewport (0, 0, window.width, window.height);
66:    glutPostRedisplay();
67: }
68:
69: int main (int argc, char** argv) {
70:    glutInit (&argc, argv);
71:    glutInitDisplayMode (GLUT_RGBA | GLUT_DOUBLE);
72:    glutInitWindowSize (window.width, window.height);
73:    glutInitWindowPosition (256, 256);
74:    glutCreateWindow (basename (argv[0]));
75:    glutDisplayFunc (display);
76:    glutReshapeFunc (reshape);
77:    glutMainLoop();
78:    return 0;
79: }
80:
```

```
 1: // $Id: ciao-mondo.cpp,v 1.5 2016-07-20 19:53:20-07 - - $
 2:
 3: // Draw line from (0,0) to (1,1).
 4:
 5: #include <cmath>
 6: #include <iostream>
 7: #include <vector>
 8: using namespace std;
 9:
10: #include <GL/freeglut.h>
11: #include <libgen.h>
12:
13: struct {
14:    int width = 384;
15:    int height = 256;
16: } window;
17:
18: const GLubyte VERDE[] {0x00, 0x92, 0x46};
19: const GLubyte BIANCO[] {0xF1, 0xF2, 0xF1};
20: const GLubyte SCARLATTO[] {0xCE, 0x2B, 0x37};
21: const vector<const GLubyte*> colors {VERDE, BIANCO, SCARLATTO};
22: constexpr GLfloat aspect_ratio = 2.0 / 3.0;
23:
24: void draw_italian_flag() {
25:    for (size_t i = 0; i < 3; ++i) {
26:       glBegin (GL_POLYGON);
27:       glColor3ubv (colors[i]);
28:       glVertex2f (window.width * i / 3.0, 0);
29:       glVertex2f (window.width * (i + 1) / 3.0, 0);
30:       glVertex2f (window.width * (i + 1) / 3.0, window.height);
31:       glVertex2f (window.width * i / 3.0, window.height);
32:       glEnd();
33:    }
34: }
35:
36: void display() {
37:    glClearColor (0.0, 0.0, 0.0, 0.0);
38:    glClear (GL_COLOR_BUFFER_BIT);
39:    draw_italian_flag();
40:    glutSwapBuffers();
41: }
42:
```

```
43:
44: void adjust_aspect (int width, int height) {
45:    if (window.width != width) {
46:       height = width * aspect_ratio;
47:    }else if (window.height != height) {
48:       width = height / aspect_ratio;
49:    }else {
50:       return;
51:    }
52:    window.width = width;
53:    window.height = height;
54:    glutReshapeWindow (window.width, window.height);
55: }
56:
57: void reshape (int width, int height) {
58:    adjust_aspect (width, height);
59:    glMatrixMode (GL_PROJECTION);
60:    glLoadIdentity();
61:    gluOrtho2D (0, window.width, 0, window.height);
62:    glMatrixMode (GL_MODELVIEW);
63:    glViewport (0, 0, window.width, window.height);
64:    glutPostRedisplay();
65: }
66:
67: int main (int argc, char** argv) {
68:    glutInit (&argc, argv);
69:    glutInitDisplayMode (GLUT_RGBA | GLUT_DOUBLE);
70:    glutInitWindowSize (window.width, window.height);
71:    glutCreateWindow (basename (argv[0]));
72:    glutDisplayFunc (display);
73:    glutReshapeFunc (reshape);
74:    glutMainLoop();
75:    return 0;
76: }
77:
```

```
 1: // $Id: hallo-welt.cpp,v 1.7 2016-07-20 19:53:20-07 - - $
 2:
 3: // Draw line from (0,0) to (1,1).
 4:
 5: #include <cmath>
 6: #include <iostream>
 7: #include <vector>
 8: using namespace std;
 9:
10: #include <GL/freeglut.h>
11: #include <libgen.h>
12:
13: struct {
14:    int width = 385;
15:    int height = 231;
16: } window;
17:
18: const GLubyte SCHWARZ[] {0x00, 0x00, 0x00};
19: const GLubyte ROT[] {0xFF, 0x00, 0x00};
20: const GLubyte GOLD[] {0xFF, 0xCC, 0x00};
21: const vector<const GLubyte*> colors {GOLD, ROT, SCHWARZ};
22: constexpr GLfloat aspect_ratio = 3.0 / 5.0;
23:
24: void draw_german_flag() {
25:    for (size_t i = 0; i < 3; ++i) {
26:       glBegin (GL_POLYGON);
27:       glColor3ubv (colors[i]);
28:       glVertex2f (0, window.height * i / 3.0);
29:       glVertex2f (0, window.height * (i + 1) / 3.0);
30:       glVertex2f (window.width, window.height * (i + 1) / 3.0);
31:       glVertex2f (window.width, window.height * i / 3.0);
32:       glEnd();
33:    }
34: }
35:
36: void display() {
37:    glClearColor (0.0, 0.0, 0.0, 0.0);
38:    glClear (GL_COLOR_BUFFER_BIT);
39:    draw_german_flag();
40:    glutSwapBuffers();
41: }
42:
```

```
43:
44: void adjust_aspect (int width, int height) {
45:    if (window.width != width) {
46:       height = width * aspect_ratio;
47:    }else if (window.height != height) {
48:       width = height / aspect_ratio;
49:    }else {
50:       return;
51:    }
52:    window.width = width;
53:    window.height = height;
54:    glutReshapeWindow (window.width, window.height);
55: }
56:
57: void reshape (int width, int height) {
58:    adjust_aspect (width, height);
59:    window.height = height;
60:    window.width = width;
61:    glMatrixMode (GL_PROJECTION);
62:    glLoadIdentity();
63:    gluOrtho2D (0, window.width, 0, window.height);
64:    glMatrixMode (GL_MODELVIEW);
65:    glViewport (0, 0, window.width, window.height);
66:    glutPostRedisplay();
67: }
68:
69: int main (int argc, char** argv) {
70:    glutInit (&argc, argv);
71:    glutInitWindowSize (window.width, window.height);
72:    glutCreateWindow (basename (argv[0]));
73:    glutDisplayFunc (display);
74:    glutReshapeFunc (reshape);
75:    glutMainLoop();
76:    return 0;
77: }
78:
```

```
 1: // $Id: konnichiwa-sekai.cpp,v 1.1 2016-06-30 17:35:43-07 - - $
 2:
 3: // Draw line from (0,0) to (1,1).
 4:
 5: #include <cmath>
 6: #include <iostream>
 7: #include <vector>
 8: using namespace std;
 9:
10: #include <GL/freeglut.h>
11: #include <libgen.h>
12:
13: struct {
14:    int width = 384;
15:    int height = 256;
16: } window;
17:
18: const GLubyte WHITE[] {255, 255, 255};
19: const GLubyte CRIMSON_GLORY[] {188, 0, 45};
20: constexpr GLfloat aspect_ratio = 2.0 / 3.0;
21:
22: void draw_japanese_flag() {
23:    glBegin (GL_POLYGON);
24:    glColor3ubv (WHITE);
25:    glVertex2f (0, 0);
26:    glVertex2f (window.width, 0);
27:    glVertex2f (window.width, window.height);
28:    glVertex2f (0, window.height);
29:    glEnd();
30:    glBegin (GL_POLYGON);
31:    glColor3ubv (CRIMSON_GLORY);
32:    const GLfloat delta = 2 * M_PI / 64;
33:    const GLfloat radius = window.height * 3.0 / 10.0;
34:    const GLfloat xoffset = window.width / 2.0;
35:    const GLfloat yoffset = window.height / 2.0;
36:    for (GLfloat theta = 0; theta < 2 * M_PI; theta += delta) {
37:       GLfloat xpos = radius * cos (theta) + xoffset;
38:       GLfloat ypos = radius * sin (theta) + yoffset;
39:       glVertex2f (xpos, ypos);
40:    }
41:    glEnd();
42: }
43:
44: void display() {
45:    glClearColor (0.0, 0.0, 0.0, 0.0);
46:    glClear (GL_COLOR_BUFFER_BIT);
47:    draw_japanese_flag();
48:    glutSwapBuffers();
49: }
50:
```

```
51:
52: void adjust_aspect (int width, int height) {
53:    if (window.width != width) {
54:       height = width * aspect_ratio;
55:    }else if (window.height != height) {
56:       width = height / aspect_ratio;
57:    }else {
58:       return;
59:    }
60:    window.width = width;
61:    window.height = height;
62:    glutReshapeWindow (window.width, window.height);
63: }
64:
65: void reshape (int width, int height) {
66:    adjust_aspect (width, height);
67:    window.width = width;
68:    window.height = height;
69:    glMatrixMode (GL_PROJECTION);
70:    glLoadIdentity();
71:    gluOrtho2D (0, window.width, 0, window.height);
72:    glMatrixMode (GL_MODELVIEW);
73:    glViewport (0, 0, window.width, window.height);
74:    glutPostRedisplay();
75: }
76:
77: int main (int argc, char** argv) {
78:    glutInit (&argc, argv);
79:    glutInitDisplayMode (GLUT_RGBA | GLUT_DOUBLE);
80:    glutInitWindowSize (window.width, window.height);
81:    glutCreateWindow (basename (argv[0]));
82:    glutDisplayFunc (display);
83:    glutReshapeFunc (reshape);
84:    glutMainLoop();
85:    return 0;
86: }
87:
```

```
 1: # $Id: Makefile,v 1.13 2019-02-21 17:30:15-08 - - $
 2:
 3: GPPWARN  = -Wall -Wextra -Werror -Wpedantic -Wshadow -Wold-style-cast
 4: GPPOPTS  = ${GPPWARN} -fdiagnostics-color=never
 5: GPP      = g++ -std=gnu++17 -g -O0 ${GPPOPTS}
 6: LIBS     = -lglut -lGLU -lGL -lX11 -lrt -lm
 7:
 8: SOURCES  = hello-world.cpp bonjour-le-monde.cpp ciao-mondo.cpp \
 9:            hallo-welt.cpp konnichiwa-sekai.cpp
10: BINARIES = ${SOURCES:.cpp=}
11:
12: all : ${BINARIES}
13:
14: % : %.cpp
15:         ${GPP} $< -o $@ ${LIBS}
16:
17: ci : ${SOURCES} Makefile
18:         cpplint.py.perl ${SOURCES}
19:         checksource ${SOURCES} Makefile
20:         cid + ${SOURCES} Makefile
21:
22: clean :
23:         - rm ${BINARIES} Listing.ps Listing.pdf
24:
25: test : all
26:         for exec in ${BINARIES}; do $$exec & done
27:
28: lis : ${SOURCES}
29:         mkpspdf Listing.ps ${SOURCES} Makefile
30:
31: again :
32:         ${MAKE} clean ci all lis test
33:
```