```
// $Id: mousetrace.cpp,v 1.66 2019-02-22 17:38:43-08 - - $

#include <cmath>
#include <iostream>
#include <string>
using namespace std;

#include <GL/freeglut.h>
#include <libgen.h>

struct window {
   string name;
   int width {256};
   int height {192};
} window;

template <typename number>
struct coordinate {
   number x_coord {};
   number y_coord {};
   string to_string() {
      return "(" + std::to_string (x_coord) + ","
                 + std::to_string (y_coord) + ")";
   }
};

const GLubyte RED[]    {0xFF, 0x00, 0x00};
const GLubyte YELLOW[] {0xFF, 0xFF, 0x00};
const GLubyte GREEN[]  {0x00, 0xFF, 0x00};
const GLubyte WHITE[]  {0xFF, 0xFF, 0xFF};

```

```

struct mouse {
   int entered {GLUT_LEFT};
   coordinate<int> coord;
   struct { int left; int middle; int right; } state
        = { GLUT_UP,  GLUT_UP,    GLUT_UP    };
   bool mouse_down() {
      if (entered == GLUT_LEFT) return false;
      return state.left == GLUT_DOWN
          or state.middle == GLUT_DOWN
          or state.right == GLUT_DOWN;
   }
   string to_string() {
      return coord.to_string()
           + (state.left == GLUT_DOWN ? "L" : "")
           + (state.middle == GLUT_DOWN ? "M" : "")
           + (state.right == GLUT_DOWN ? "R" : "");
   }
   void draw() {
      if (entered == GLUT_LEFT) return;
      void* font = GLUT_BITMAP_9_BY_15;
      glColor3ubv (WHITE);
      glRasterPos2i (5, 5);
      glutBitmapString (font,
                        reinterpret_cast<const GLubyte*>
                        (to_string().c_str()));
   }
} mouse;

```

```

struct ellipse {
   coordinate<GLfloat> coord;
   const GLubyte* color;
   ellipse() { coord.x_coord = window.width / 2;
               coord.y_coord = window.height / 2;
             }
   inline GLfloat width() const { return window.width / 10; }
   inline GLfloat height() const { return window.height / 10; }
   bool has_mouse() {
      if (mouse.entered == GLUT_LEFT) return false;
      GLfloat delta_x = mouse.coord.x_coord - coord.x_coord;
      GLfloat delta_y = mouse.coord.y_coord - coord.y_coord;
      return pow (delta_x, 2) / pow (width(), 2)
           + pow (delta_y, 2) / pow (height(), 2) <= 1;
   }
   void set_color_coord() {
      if (not has_mouse()) {
         color = GREEN;
      }else if (not mouse.mouse_down()) {
         color = YELLOW;
      }else {
         color = RED;
         coord.x_coord = mouse.coord.x_coord;
         coord.y_coord = mouse.coord.y_coord;
      }
   }
   void draw() {
      set_color_coord();
      glBegin (GL_POLYGON);
      glColor3ubv (color);
      GLfloat delta = 2 * M_PI / 64;
      for (GLfloat theta = 0; theta < 2 * M_PI; theta += delta) {
         GLfloat x = width() * cos (theta) + coord.x_coord;
         GLfloat y = height() * sin (theta) + coord.y_coord;
         glVertex2f (x, y);
      }
      glEnd();
   }
} ellipse;

```

```

void display_func() {
   glClear (GL_COLOR_BUFFER_BIT);
   ellipse.draw();
   mouse.draw();
   glutSwapBuffers();
}

void reshape_func (int width, int height) {
   window.width = width;
   window.height = height;
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D (0, window.width, 0, window.height);
   glMatrixMode (GL_MODELVIEW);
   glViewport (0, 0, window.width, window.height);
   glClearColor (0.25, 0.25, 0.25, 1.0);
   glutPostRedisplay();
}

void mouse_func (int button, int state, int mouse_x, int mouse_y) {
   mouse.coord = {mouse_x, window.height - mouse_y};
   switch (button) {
      case GLUT_LEFT_BUTTON: mouse.state.left = state; break;
      case GLUT_MIDDLE_BUTTON: mouse.state.middle = state; break;
      case GLUT_RIGHT_BUTTON: mouse.state.right = state; break;
   }
   glutPostRedisplay();
}

void entry_func (int entered) {
   mouse.entered = entered;
   glutPostRedisplay();
}

void motion_func (int mouse_x, int mouse_y) {
   mouse.coord = {mouse_x, window.height - mouse_y};
   glutPostRedisplay();
}

void passivemotion_func (int mouse_x, int mouse_y) {
   mouse.coord = {mouse_x, window.height - mouse_y};
   glutPostRedisplay();
}

```

```

int main (int argc, char** argv) {
   window.name = basename (argv[0]);
   glutInit (&argc, argv);
   glutInitDisplayMode (GLUT_RGBA | GLUT_DOUBLE);
   glutInitWindowSize (window.width, window.height);
   glutCreateWindow (window.name.c_str());
   glutDisplayFunc (display_func);
   glutReshapeFunc (reshape_func);
   glutMouseFunc (mouse_func);
   glutMotionFunc (motion_func);
   glutEntryFunc (entry_func);
   glutPassiveMotionFunc (passivemotion_func);
   glutMainLoop();
   return 0;
}

//TEST// mkpspdf mousetrace.ps mousetrace.cpp*

```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting mousetrace.cpp
checksource mousetrace.cpp
ident mousetrace.cpp
mousetrace.cpp:
     $Id: mousetrace.cpp,v 1.66 2019-02-22 17:38:43-08 - - $
cpplint.py.perl mousetrace.cpp
Done processing mousetrace.cpp
g++ -g -O0 -Wall -Wextra -Werror -Wpedantic -Wshadow -fdiagnostics-color=never -std=gnu++17 -Wold-style-cast mousetrace.cpp -o mousetrace -lm -lglut -lGLU -lGL -lX11 -ldrm -lm
rm -f mousetrace.o
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished mousetrace.cpp
```