```
// $Id: circles.cpp,v 1.52 2019-02-22 19:16:41-08 - - $

// Draw several ellipses in window.

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

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

// Characteristics of the window.
struct {
   string name;
   int width {512};
   int height {384};
} window;

struct rgbcolor { GLubyte ubv[3]; };
const unordered_map<string,rgbcolor> colors {
   {"red",     {0xFF, 0x00, 0x00}},
   {"green",   {0x00, 0xFF, 0x00}},
   {"blue",    {0x00, 0x00, 0xFF}},
   {"cyan",    {0x00, 0xFF, 0xFF}},
   {"magenta", {0xFF, 0x00, 0xFF}},
   {"yellow",  {0xFF, 0xFF, 0x00}},
   {"white",   {0xFF, 0xFF, 0xFF}},
   {"black",   {0x00, 0x00, 0x00}},
};

void draw_xy_graph (const rgbcolor& color) {
   glLineWidth (4);
   glBegin (GL_LINES);
   glColor3ubv (color.ubv);
   glVertex2f (-window.width / 2, 0);
   glVertex2f (+window.width / 2, 0);
   glVertex2f (0, -window.height);
   glVertex2f (0, +window.height);
   glEnd();
}

```

```

void draw_circle (const rgbcolor& color, size_t multiplier,
                  GLfloat radius) {
   glLineWidth (4);
   glBegin (GL_LINE_LOOP);
   glColor3ubv (color.ubv);
   const size_t points = multiplier * 4;
   const GLfloat theta = 2.0 * M_PI / points;
   for (size_t point = 0; point < points; ++point) {
      GLfloat angle = point * theta;
      GLfloat xpos = radius * cos (angle);
      GLfloat ypos = radius * sin (angle);
      glVertex2f (xpos, ypos);
   }
   glEnd();
}

// Called by glutMainLoop to display window contents.
void display() {
   cout << __PRETTY_FUNCTION__ << ":" << endl;
   glClearColor (0.25, 0.25, 0.25, 1.0);
   glClear (GL_COLOR_BUFFER_BIT);
   draw_xy_graph (colors.at("cyan"));
   const GLfloat radius = min (window.width, window.height) / 20.0;
   for (size_t count = 1; count <= 10; ++count) {
      draw_circle (colors.at("green"), count, radius * count);
   }
   glutSwapBuffers();
}

void reshape (int width, int height) {
   cout << __PRETTY_FUNCTION__ << ": "
        << width << ", " << height << endl;
   window.width = width;
   window.height = height;
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D (-window.width / 2, +window.width / 2,
               -window.height / 2, +window.height / 2);
   glMatrixMode (GL_MODELVIEW);
   glViewport (0, 0, window.width, window.height);
   glutPostRedisplay();
}

```

```

void close() {
   cout << __PRETTY_FUNCTION__ << ":" << endl;
}

void entry (int state) {
   cout << __PRETTY_FUNCTION__ << ": ";
   switch (state) {
      case GLUT_LEFT: cout << "GLUT_LEFT"; break;
      case GLUT_ENTERED: cout << "GLUT_ENTERED"; break;
      default: cout << state; break;
   }
   cout << endl;
}

int main (int argc, char** argv) {
   cout << __PRETTY_FUNCTION__ << ": "
        << argc << ", " << argv[0] << endl;
   window.name = basename (argv[0]);
   glutInit (&argc, argv);
   glutInitDisplayMode (GLUT_RGBA | GLUT_DOUBLE);
   glutInitWindowSize (window.width, window.height);
   glutInitWindowPosition (128, 128);
   glutCreateWindow (window.name.c_str());
   glutDisplayFunc (display);
   glutReshapeFunc (reshape);
   glutEntryFunc (entry);
   glutCloseFunc (close);
   glutMainLoop();
   return 0;
}

//TEST// mkpspdf circles.ps circles.cpp*

```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting circles.cpp
checksource circles.cpp
ident circles.cpp
circles.cpp:
     $Id: circles.cpp,v 1.52 2019-02-22 19:16:41-08 - - $
cpplint.py.perl circles.cpp
Done processing circles.cpp
g++ -g -O0 -Wall -Wextra -Werror -Wpedantic -Wshadow -fdiagnostics-color=never -std=gnu++17 -Wold-style-cast circles.cpp -o circles -lm -lglut -lGLU -lGL -lX11 -ldrm -lm
rm -f circles.o
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished circles.cpp
```