```
/*************************************************************************************************
 GeoDraw Version 1.0 (23.10) - Last revision: 01/10/2023
 - A simple C/C++ 2D drawing library that outputs drawings as JavaScript
 - Includes a simplified C function interface

 NOTICE: Copyright (C) 2023 Conor McArdle, Dublin City University - All Rights Reserved.
 Permission is granted to students registered for DCU module EE219: C/C++ for Engineers
 to use this code for their personal use only. No permission is granted to share this file
 by any means, including but not limited to, uploading it or submitting it to any private
 or public server on the Web/Internet or any online service which publishes or otherwise
 processes or stores uploaded content. This notice must stay attached to this file.
*************************************************************************************************/

#include "GeoDraw.h"

namespace GeoDraw {

const string _gd_version = "23.10";    // software version number

string _gd_html_pre;                // html to appear before canvas element
string _gd_html_post;                // html to appear after canvas element

string gd_to_string(int a) {          // utility string conversion methods
   char buffer[50];
   sprintf(buffer, "%d", a);
   return string(buffer);
}

string gd_to_string(double a, unsigned short decimalPlaces = 2) {
   char buffer[50];
   string format = "%." + gd_to_string(decimalPlaces) + "f";
   sprintf(buffer, format.c_str(), a);
   return string(buffer);
}

/*************************************************************************************************
 Canvas class - public methods - Canvas::draw() and Canvas::draw(filename)
*************************************************************************************************/

void Canvas::draw() {
   string JSDrawingString;

   // HTML output file header with drawing functions
   JSDrawingString += "<!DOCTYPE html>\n<html lang=\"en\">\n<head>\n<meta
charset=\"utf-8\">\n<title>GeoDraw</title>\n";
   JSDrawingString += "<style>\n  canvas { border: 1px solid #707070;
}\n</style>\n<script>\n\n";
   JSDrawingString += "window.onload = function() {\n";
```

```
    JSDrawingString += "var canvas = document.getElementById(\"myCanvas\");\n";
    JSDrawingString += "var ctx = canvas.getContext(\"2d\");\n";
    JSDrawingString += "ctx.fillStyle = 'rgb" + bg_color.to24BitColorString() + "';\n";
    JSDrawingString += "ctx.fillRect(0,0,canvas.width,canvas.height);\n\n";

    // Generate JavaScript drawing commands from canvas elements
    cout << "Generating " << elements.size() << " JavaScipt drawing objects ... ";
    clock_t start, end;
    start = clock();
    JSDrawingString += this->generateJSDrawingString();
    end = clock();
    cout << "done in ";
    cout << double(end - start)/double(CLOCKS_PER_SEC) << " seconds." << endl;

    // HTML output file footer
    JSDrawingString += "\n};\n</script>\n</head>\n";
    JSDrawingString += "<body style=\"background-color:gray;\">\n";
    JSDrawingString += _gd_html_pre;
    JSDrawingString += "  <canvas id=\"myCanvas\" width=\"" + gd_to_string((int)_xDim) + "\"
" + "height=\"" + gd_to_string((int)_yDim) + "\"></canvas>\n";
    JSDrawingString += "<p
style=\"font-family:Arial;font-size:12px;color:LightGray\"> Produced by GeoDraw-" +
_gd_version + " C/C++ Library, conor.mcardle@dcu.ie, 2023</p>";
    JSDrawingString += _gd_html_post;
    JSDrawingString += "</body>\n</html>\n";

    // Save the JS string to file
    std::ofstream outHTMLFile;
    outHTMLFile.open(outFileName.c_str());
    if(outHTMLFile) {
        cout << "Saving to file ... ";
        outHTMLFile << JSDrawingString;
        outHTMLFile.close();
        cout << "done." << endl;
        cout << "JavaScript drawing created in " << outFileName << endl;
    } else {
        cerr << "Error opening output file " << outFileName << endl;
    }
}

void Canvas::draw(string filename) {
    outFileName = filename;
    Canvas::draw();
}

/*********************************************************************************************
 Canvas class - private helper members
*********************************************************************************************/
```

```
string Canvas::generateJSDrawingString() {
   string JSDrawingString;  // string to store JavaScript drawing code
   u_int time = 0;          // current canvas element draw time
   // Generate JavaScript for each geometry element,
   // instering drawing pause events at appropriate times
   for (u_int i=0; i<elements.size(); i++) {
      JSDrawingString += elements[i]->toJavaScript() + "\n";
      if (elements[i]->pauseAfter != 0) {
         // if there was a previous pause event, close the JS timeout function
         if (time > 0)
            JSDrawingString += "}, " + gd_to_string((int)time) + ");\n";
         // open new JS timeout function
         JSDrawingString += "\nsetTimeout(function() {\n";
         // update current display time
         time += elements[i]->pauseAfter;
      }
   }
   if (time > 0) // close last pause timeout function
      JSDrawingString += "}, " + gd_to_string((int)time) + ");\n";
   return JSDrawingString;
}


/*********************************************************************************************
 JavaScript drawing implementations for Drawable objects
*********************************************************************************************/

/*********************************************************************************************
 Point::toJavaScript() - private
 A friend of the Canvas class
 Called by Canvas::draw() to generate JavaScript to draw a Point object
*********************************************************************************************/
string Point::toJavaScript() const {
   string JS_string;
   JS_string += "ctx.beginPath();\n";
   JS_string += "ctx.fillStyle = 'rgb" + color.to24BitColorString() + "';\n";
   JS_string += "ctx.arc(" + gd_to_string(coord.x()) + "," + gd_to_string(coord.y());
   JS_string += "," + gd_to_string((int)(penWidth/2)) + ",0,2*Math.PI);\n";
   JS_string += "ctx.fill();\n";
   return JS_string;
}

/*********************************************************************************************
 LineSeg::toJavaScript() - private
 A friend of the Canvas class
 Called by Canvas::draw() to generate JavaScript to draw a LineSeg object
*********************************************************************************************/
```

```
string LineSeg::toJavaScript() const {
   string JS_string;
   JS_string += "ctx.beginPath();\n";
   JS_string += "ctx.strokeStyle = 'rgb" + color.to24BitColorString() + "';\n";
   JS_string += "ctx.lineWidth = " + gd_to_string((int)penWidth) + ";\n";
   JS_string += "ctx.moveTo(" + gd_to_string(c1.x());
   JS_string += "," + gd_to_string(c1.y()) + ");\n";
   JS_string += "ctx.lineTo(" + gd_to_string(c2.x());
   JS_string += "," + gd_to_string(c2.y()) + ");\n";
   JS_string += "ctx.stroke();\n";
   return JS_string;
}

/*****************************************************************************************
 Circle::toJavaScript() - private
 A friend of the Canvas class
 Called by Canvas::draw() to generate JavaScript to draw a Circle object
*****************************************************************************************/
string Circle::toJavaScript() const {
   string JS_string;
   JS_string += "ctx.beginPath();\n";
   if (fillState == FILLED)
      JS_string += "ctx.fillStyle = 'rgb" + color.to24BitColorString() + "';\n";
   else if (fillState == UNFILLED) {
      JS_string += "ctx.strokeStyle = 'rgb" + color.to24BitColorString() + "';\n";
      JS_string += "ctx.lineWidth = " + gd_to_string((int)penWidth) + ";\n";
   }
   JS_string += "ctx.arc(" + gd_to_string(cen.x()) + "," + gd_to_string(cen.y());
   JS_string += "," + gd_to_string((int) radius) + ",0,2*Math.PI);\n";
   if (fillState == FILLED)
      JS_string += "ctx.fill();\n";
   else if (fillState == UNFILLED)
      JS_string += "ctx.stroke();\n";
   return JS_string;
}

/*****************************************************************************************
 Polygon::toJavaScript() - private
 A friend of the Canvas class
 Called by Canvas::draw() to generate JavaScript to draw a Polygon object
*****************************************************************************************/
string Polygon::toJavaScript() const {
   if (vertices.size() == 0) return "";
   string JS_string;
   JS_string += "ctx.beginPath();\n";
   if (fillState == FILLED)
      JS_string += "ctx.fillStyle = 'rgb" + color.to24BitColorString() + "';\n";
   else if (fillState == UNFILLED) {
```

```
        JS_string += "ctx.strokeStyle = 'rgb" + color.to24BitColorString() + "';\n";
        JS_string += "ctx.lineWidth = " + gd_to_string((int)penWidth) + ";\n";
    }
    JS_string += "ctx.moveTo(" + vertices[0].toString() + ");\n";
    for (unsigned int i=1; i<vertices.size(); i++)
        JS_string += "ctx.lineTo(" + vertices[i].toString() + ");\n";
    JS_string += "ctx.closePath();\n";
    if (fillState == FILLED)
        JS_string += "ctx.fill();\n";
    else if (fillState == UNFILLED)
        JS_string += "ctx.stroke();\n";
    return JS_string;
}

/*********************************************************************************************
 Text::toJavaScript() - private
 A friend of the Canvas class
 Called by Canvas::draw() to generate JavaScript to draw Text on canvas
*********************************************************************************************/
string Text::toJavaScript() const {
    string font_name;
    switch (this->font) {
        case Arial: font_name = "Arial"; break;
        case Courier: font_name = "Courier"; break;
        case Times: font_name = "Times"; break;
    }
    string JS_string;
    JS_string += "ctx.font = \"" + gd_to_string((int)this->fontSize) + "px " + font_name + "\";\n";
    if (alignment == LEFT)
        JS_string += "ctx.textAlign = 'left';\n";
    else if (alignment == CENTER)
        JS_string += "ctx.textAlign = 'center';\n";
    else
        JS_string += "ctx.textAlign = 'right';\n";
    JS_string += "ctx.textBaseline = 'middle';\n";
    if (fillState == FILLED) {
        JS_string += "ctx.fillStyle = 'rgb" + color.to24BitColorString() + "';\n";
        JS_string += "ctx.fillText(\"" + this->text + "\"," + this->position.toString() + ");\n";
    }
    else if (fillState == UNFILLED) {
        JS_string += "ctx.strokeStyle = 'rgb" + color.to24BitColorString() + "';\n";
        JS_string += "ctx.strokeText(\"" + this->text + "\"," + this->position.toString() + ");\n";
    }
    return JS_string;
}

} // end GeoDraw namespace
```

```
/*********************************************************************************************
*
*   SIMPLIFIED C-STYLE INTERFACE for GeoDraw - Implementation
*
*********************************************************************************************/

namespace GeoDrawC
{
   u_int      _cgd_canvas_size_x   =   600;
   u_int      _cgd_canvas_size_y   =   600;
   u_int      _cgd_pen_width       =   2;
   Color      _cgd_pen_color       =   BLACK;
   Color      _cgd_fill_color      =   GRAY;
   Font       _cgd_font            =   Arial;
   u_int      _cgd_font_size       =   20;
   Color      _cgd_font_color      =   BLACK;
   TextAlign  _cgd_text_alignment  =   LEFT;
   Canvas     _cgd_canvas(_cgd_canvas_size_x,_cgd_canvas_size_y);
}

u_int gd_getCanvasSizeX() {
   return GeoDrawC::_cgd_canvas.xDim();
}

u_int gd_getCanvasSizeY() {
   return GeoDrawC::_cgd_canvas.yDim();
}

void gd_resetCanvasSize(u_int xSize, u_int ySize) {
   GeoDrawC::_cgd_canvas = Canvas(xSize, ySize);
}

void gd_setCanvasColor(Color color) {
   GeoDrawC::_cgd_canvas.setBackgroundColor(color);
}

void gd_setCanvasColor(double r, double g, double b) {
   GeoDrawC::_cgd_canvas.setBackgroundColor(Color(r,g,b));
}

void gd_setPenWidth(u_int width) {
   GeoDrawC::_cgd_pen_width = width;
}

void gd_setPenColor(Color color) {
   GeoDrawC::_cgd_pen_color = color;
}
```

```
void gd_setPenColor(double r, double g, double b) {
   GeoDrawC::_cgd_pen_color = Color(r,g,b);
}

void gd_setFillColor(Color color) {
   GeoDrawC::_cgd_fill_color = color;
}

void gd_setFillColor(double r, double g, double b) {
   GeoDrawC::_cgd_fill_color = Color(r,g,b);
}

void gd_setFont(Font font) {
   GeoDrawC::_cgd_font = font;
}

void gd_setTextSize(u_int font_size) {
   GeoDrawC::_cgd_font_size = font_size;
}

void gd_setTextColor(Color color) {
   GeoDrawC::_cgd_font_color = color;
}

void gd_setTextColor(double r, double g, double b) {
   GeoDrawC::_cgd_font_color = Color(r,g,b);
}

void gd_setTextAlignment(TextAlign alignment) {
   GeoDrawC::_cgd_text_alignment = alignment;
}

void gd_point(double x, double y) {
   GeoDrawC::_cgd_canvas.add(Point(x,y), GeoDrawC::_cgd_pen_color,
GeoDrawC::_cgd_pen_width);
}

void gd_line(double x1, double y1, double x2, double y2) {
   GeoDrawC::_cgd_canvas.add(LineSeg(x1,y1,x2,y2), GeoDrawC::_cgd_pen_color,
GeoDrawC::_cgd_pen_width);
}

void gd_circle(double x, double y, double radius) {
   GeoDrawC::_cgd_canvas.add(Circle(x,y,radius), GeoDrawC::_cgd_pen_color,
GeoDrawC::_cgd_pen_width);
}
```

```
void gd_circleFilled(double x, double y, double radius) {
   GeoDrawC::_cgd_canvas.add(Circle(x,y,radius), GeoDrawC::_cgd_fill_color, FILLED);
}

void gd_triangle(double x1, double y1, double x2, double y2, double x3, double y3) {
   Polygon poly;
   poly.add(x1,y1);
   poly.add(x2,y2);
   poly.add(x3,y3);
   GeoDrawC::_cgd_canvas.add(poly, GeoDrawC::_cgd_pen_color,
GeoDrawC::_cgd_pen_width);
}

void gd_triangleFilled(double x1, double y1, double x2, double y2, double x3, double y3) {
   Polygon poly;
   poly.add(x1,y1);
   poly.add(x2,y2);
   poly.add(x3,y3);
   GeoDrawC::_cgd_canvas.add(poly, GeoDrawC::_cgd_fill_color, FILLED);
}

void gd_quad(double x1, double y1, double x2, double y2, double x3, double y3, double x4,
double y4) {
   Polygon poly;
   poly.add(x1,y1);
   poly.add(x2,y2);
   poly.add(x3,y3);
   poly.add(x4,y4);
   GeoDrawC::_cgd_canvas.add(poly, GeoDrawC::_cgd_pen_color,
GeoDrawC::_cgd_pen_width);
}

void gd_quadFilled(double x1, double y1, double x2, double y2, double x3, double y3,
double x4, double y4) {
   Polygon poly;
   poly.add(x1,y1);
   poly.add(x2,y2);
   poly.add(x3,y3);
   poly.add(x4,y4);
   GeoDrawC::_cgd_canvas.add(poly, GeoDrawC::_cgd_fill_color, FILLED);
}

void gd_text(string txt, double x, double y) {
   GeoDrawC::_cgd_canvas.add(Text(txt, x, y, GeoDrawC::_cgd_font,
GeoDrawC::_cgd_font_size, GeoDrawC::_cgd_text_alignment),
GeoDrawC::_cgd_font_color, FILLED);
}
```

```
void gd_pause(u_int pauseTimeMs) {
    GeoDrawC::_cgd_canvas.pause(pauseTimeMs);
}

void gd_clear() {
    GeoDrawC::_cgd_canvas.clear();
}

void gd_save(string filename) {
    GeoDrawC::_cgd_canvas.draw(filename);
}

//////////////////////////////////////////////////////////////
```