CS 7A - Spring 2016 - The Rounded Box of PPP13-ex02. Due 3/31/16

The big question here is how to create the code needed to draw rounded boxes in the fltk world we have access to by the Stroustrup interface, so far.

We start by writing `Box` code for the interface library, `Graph.h`

```
//------------------------------------------------------------------------------
struct Box : Graph_lib::Shape {
    Box(Point xxyy, int ww, int hh);  //Constr prototype

    void Top_segment ();  // sets private members top_seg vector of Points
    void Bottom_segment ();  // and so on
    void Left_side_segment ();
    void Right_side_segment();
    void Get_segments(); /// Calls the above four edge builders

    void draw_lines() const;

    int height() const { return h; }
    int width () const { return w; }

private:
    Point xy; //left most corner

    int h; // height
    int w; // width

    vector<Point> top_seg;
    vector<Point> bottom_seg;
    vector<Point> left_seg;
    vector<Point> right_seg;

    double longer_side_tenth; //10% of the width that will calculate the length to r
};
```

To make these work, we write an implementation code in Graph.cpp:

```
Box::Box(Point xxyy, int ww, int hh): w(ww), h(hh), xy(xxyy) {
    if(w>=h)
        longer_side_tenth = (xy.x + w) * 0.10;
    if(h>w)
        longer_side_tenth = (xy.x + h) * 0.10;
    if (h<=0 || w<=0) error("Bad rectangle: non-positive side");
    Get_segments();
}
//------------------------------------------------------------------------------
void Box::Get_segments() {
    Top_segment ();
    Bottom_segment ();
    Left_side_segment ();
    Right_side_segment();
}
//------------------------------------------------------------------------------
```

```
void Box::draw_lines() const {
//top segment
    fl_line(
        top_seg[0].x, top_seg[0].y,
        top_seg[1].x, top_seg[1].y);
//bottom segment
    fl_line(
        bottom_seg[0].x, bottom_seg[0].y,
        bottom_seg[1].x, bottom_seg[1].y);
//left segment
    fl_line(
        left_seg[0].x, left_seg[0].y,
        left_seg[1].x, left_seg[1].y);
//right segmnet
    fl_line(
        right_seg[0].x, right_seg[0].y,
        right_seg[1].x, right_seg[1].y);
//top left arc
    fl_arc(
        xy.x, xy.y,
        longer_side_tenth, longer_side_tenth,
        90,180);
//top right arc
    fl_arc(
        (xy.x + w)-longer_side_tenth, xy.y,
        longer_side_tenth, longer_side_tenth,
        0,90);
//bottom left arc
    fl_arc(
        xy.x, (xy.y + h) - longer_side_tenth,
        longer_side_tenth, longer_side_tenth,
        180,270);
//bottom right arc
    fl_arc(
        (xy.x + w) - longer_side_tenth,
        (xy.y + h) - longer_side_tenth,
        longer_side_tenth, longer_side_tenth,
        270,360);
}
//------------------------------------------------------------------------------
void Box::Top_segment() {
    double top_seg_begin_w; //where the line segment will begin after deducting 10%
    double top_seg_end_w; //where the line segment will end after deducting 10% of w
    top_seg_begin_w = xy.x + longer_side_tenth/2;
    top_seg_end_w = (xy.x + w) - longer_side_tenth/2;
    double top_seg_y = xy.y;
    top_seg.push_back(Point(top_seg_begin_w,top_seg_y));
    top_seg.push_back(Point(top_seg_end_w,top_seg_y));
}
//------------------------------------------------------------------------------
void Box::Bottom_segment() {
    double bottom_seg_begin_w;
    double bottom_seg_end_w;
```

```
    bottom_seg_begin_w = xy.x + longer_side_tenth/2;
    bottom_seg_end_w = (xy.x + w) - longer_side_tenth/2;
    double y_bottom = xy.y + h;
    bottom_seg.push_back(Point(bottom_seg_begin_w,y_bottom));
    bottom_seg.push_back(Point(bottom_seg_end_w,y_bottom));
}
//------------------------------------------------------------------------------
void Box::Left_side_segment() {
    double left_seg_begin_h;
    double left_seg_end_h;
    left_seg_begin_h = xy.y + longer_side_tenth/2;
    left_seg_end_h = (xy.y + h) - longer_side_tenth/2;
    double x_left = xy.x;
    left_seg.push_back(Point(x_left, left_seg_begin_h));
    left_seg.push_back(Point(x_left, left_seg_end_h));
}
//------------------------------------------------------------------------------
void Box::Right_side_segment() {
    double right_seg_begin_h;
    double right_seg_end_h;
    right_seg_begin_h = xy.y + longer_side_tenth/2;
    right_seg_end_h = (xy.y + h) - longer_side_tenth/2;
    double x_right = xy.x + w;
    right_seg.push_back(Point(x_right, right_seg_begin_h));
    right_seg.push_back(Point(x_right, right_seg_end_h));
}
```

1. Make this work in your code libraries, `Graph.h` and `Graph.cpp`
2. How would you improve on this design?
3. What other decorative Box design features would you add?