Class Topography for Drawing Application

Luke Testa: 6119412, Alex Freestone: 6133928, Oliver Chilton: 6129415

The base class of all shapes is ‘Shape’ which holds basic information on position and color as well as the required overloading operator functions.
‘Polygon’ and ‘Circle’ are derived classes of ‘Shape’ which provide more information of the number and positions of vertices etc. Each class overrides an input stream operator for storing shape parameters.

User defined shapes (see page 2) are stored as a List of ‘Shape’ pointers which are stepped through sequentially and drawn in order of declaration.

Shape

```
private:
        Scalar* colour;
        Point* center;
protected:
        inline Scalar getColour();
        inline Point getCenter();

        // for complex shapes
        void offsetCenter(Point pt);
public:
        Shape() : colour( NULL), center(NULL);
        Shape(Scalar &c, Point &CoM) : colour(c), center(CoM);
        Shape(const Shape& S) : colour(S.colour), center(S.center);
        virtual ~Shape() { delete colour, delete center };
        Shape operator =(const Shape& copyShape);
        virtual void draw() = 0;

        // input parameters from file and error handling
        friend istream& operator >>(istream& input, Shape& inputShape);
```

Polygon

```
private:
        Point* Vertices;

protected:
        Inline Vertices getVertices();

public:
        Polygon() : Vertices( NULL), Shape()
        Polygon(Scalar c, Point pt, Vertices pts) : Shape(c,pt),
        Vertices(pts);
        virtual void draw();

        virtual ~Polygon() {  delete Vertices };

        Polygon& operator =(Polygon& P);
```

Circle

```
private:
        unsigned int radius;

public:
        Circle() : Shape(), radius(0);
        Circle(Color c, Point CoM,  unsigned int r) : Shape(c, CoM), radius(r);
        Circle(const Circle& cir) : Shape( (Shape&) cir),  radius(cir.radius);

        virtual ~Circle() {};
        void draw();

        Circle operator =(const Circle& copyCircle);
        friend istream& operator >>(istream& input, Circle& c);

        inline int getRadius();
```

Rectangle

```
private:
        unsigned int H;
        unsigned int W;

protected:
        Vertices computeVertices();

public:
        Rectangle() : Polygon(),  W(0), H(0),
        Rectangle(Scalar c, Point pt,  unsigned int h, unsigned int w) : Polygon( c, pt, computeVertices() ), W(w), H(h);
        Rectangle(const Rectangle& R) :  Polygon( (Polygon&) R), W(R.W), H(R.H);

        virtual ~Rectangle(){ };
        void draw();
        Rectangle operator =( const Rectangle& R);
        friend istream& operator >>(istream& input, Rectangle& R);
```