SOFTWARE ENGINEERING PROJECT

COMPUTER GRAPHICS LAB - POLYGON CLIPPING EXPERIMENT

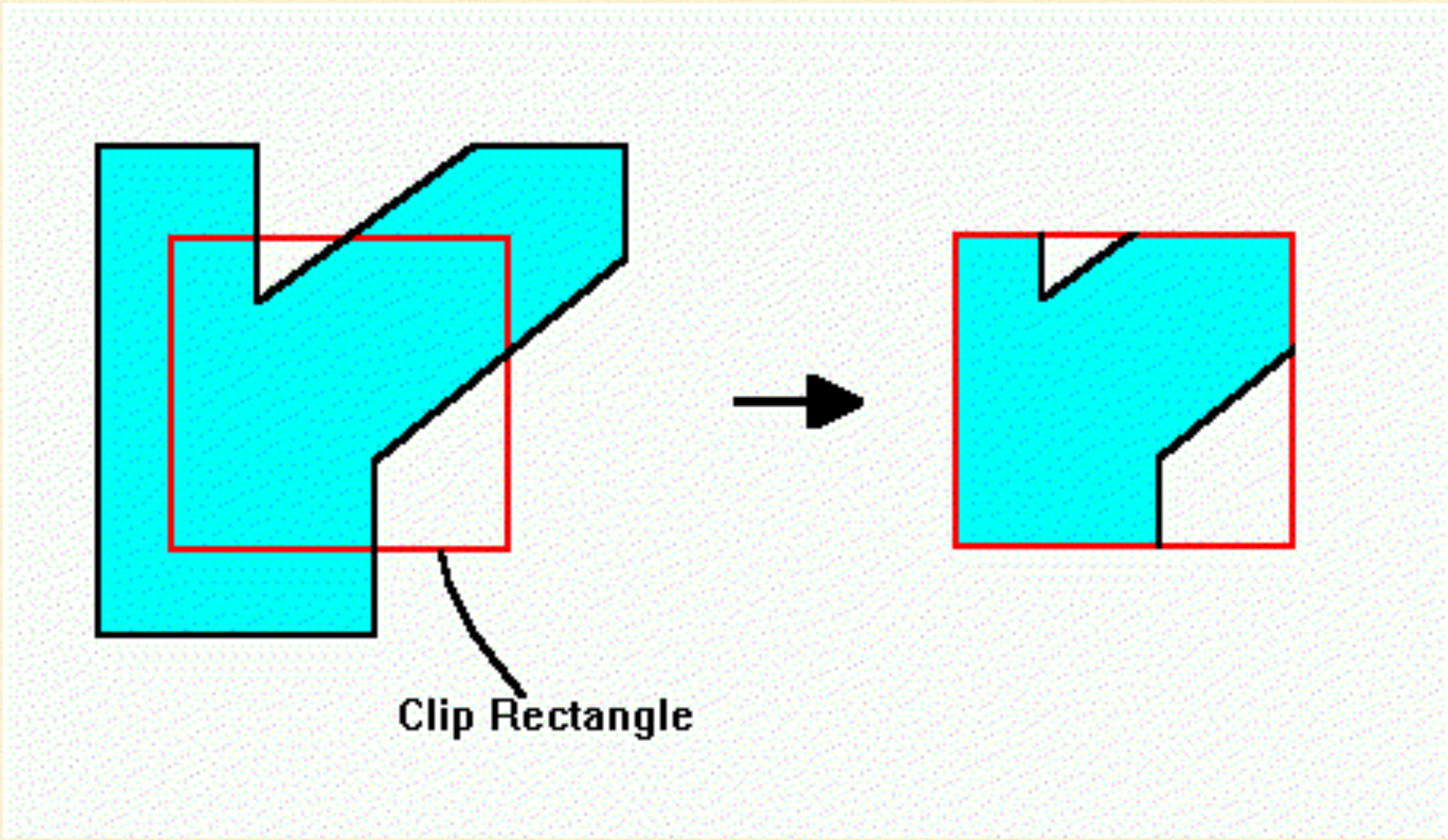
POLYGON CLIPPING

- When a polygon is displayed onto a screen, only a part of it is visible to the user .

- The unnecessary parts of the Polygon that are located outside the display area can be removed to reduce the exorbitant amount of computations involved to render them.

- This technique to remove such inessential parts of a scene lying outside the display area is called clipping.

EXAMPLE



QUALITY METRICS

Type Name	Type Rank	# BC Instructions	Afferent Coupling	Efferent Coupling
ScanPolygonMa	0.15	3611	0	30
ScanLineMain	0.15	4075	0	24
ScanConversion	0.5	1135	2	10
ReturnObject	1	15	2	1
ReturnClipPoly	0.59	3	1	1
PolygonClipping$	0.59	-	1	3
PolygonClipping	0.34	897	1	11
Polygon	2.3	150	6	3
Point	6.7	293	9	1
Messages	0.58	33	2	1
LineClipping	0.39	867	1	8
Line	1.7	239	4	3
Input	0.15	405	0	14
FrameBuffer	1.01	153	3	1
Edge	0.81	25	2	1
DrawToolkit	0.93	1049	4	8
DrawClipPolygon	0.15	2843	0	25
DrawClipLine	0.15	3551	0	20
Blink$1	0.59	10	1	3
Blink	1.85	206	1	14

CODE QUALITY METRICS

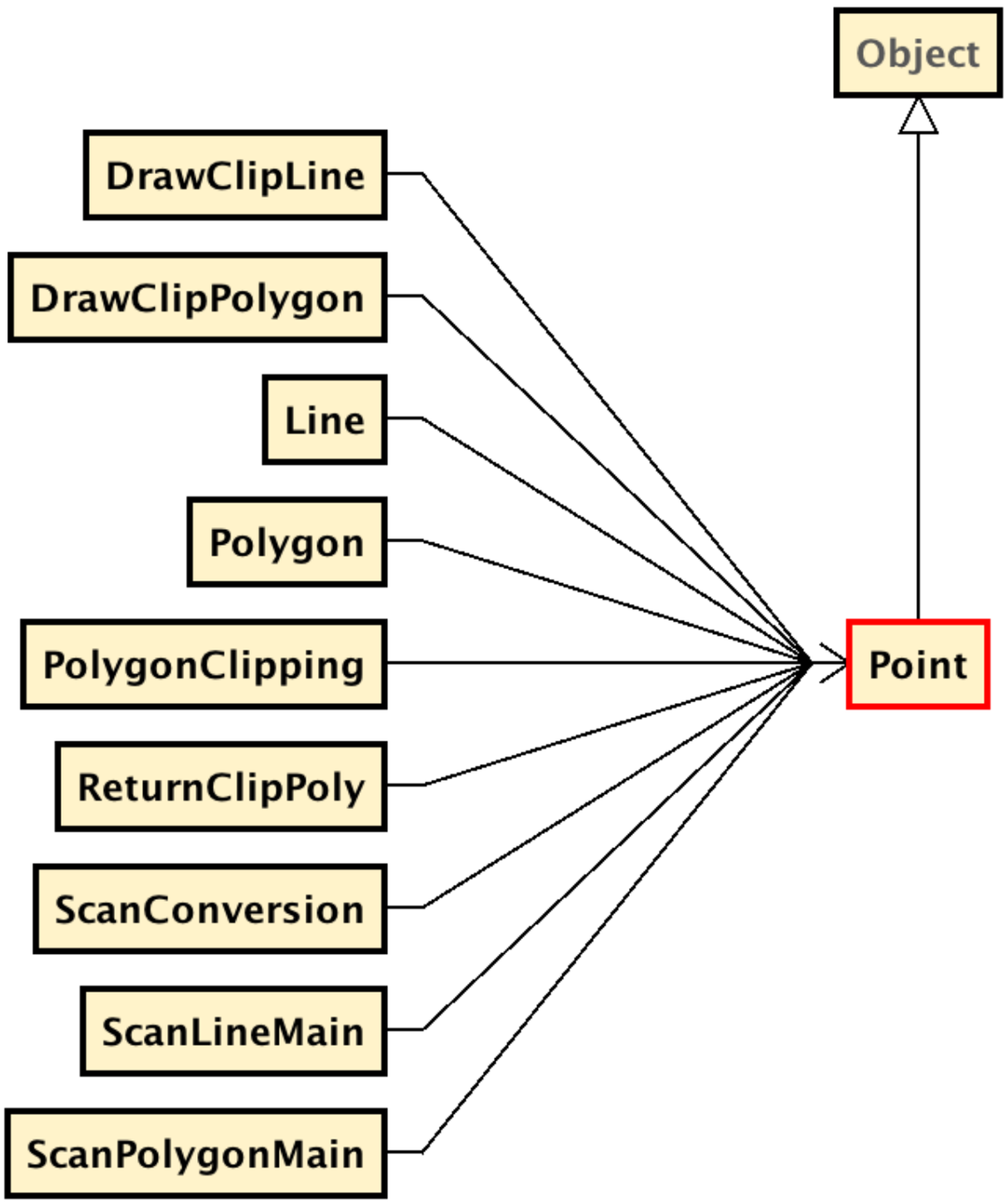
Type Name	# Instance Methods	Nb Static Methods	# Fields	Depth Of Inheritance Tree
Point	13	1	10	1
Polygon	7	0	1	1
Blink	7	0	4	5
Line	6	0	5	1
FrameBuffer	15	0	7	1
ReturnObject	4	0	3	1
DrawToolkit	22	0	0	1
Edge	2	0	4	1
PolygonClipping$	-	-	-	-
ReturnClipPoly	1	0	2	1
Blink$1	2	0	1	2
Messages	1	12	12	1
ScanConversion	17	1	11	1
LineClipping	6	1	8	1
PolygonClipping	8	2	7	1
ScanLineMain	12	0	67	5
Input	3	0	9	5
ScanPolygonMa	12	0	53	5
DrawClipPolygon	10	1	63	5
DrawClipLine	9	1	68	5

CODE MEMBERS AND INHERITANCE

Type Name	Lack Of Cohesion Of Methods	Lack Of Cohesion Of Methods HS	Association Between Classes
Point	0.32	0.35	1
Polygon	0	0	11
Blink	0.71	0.83	24
Line	0.27	0.32	9
FrameBuffer	0.76	0.82	1
ReturnObject	0.75	1	1
DrawToolkit	0	0	26
Edge	0.38	0.75	1
PolygonClipping$Edge	-	-	3
ReturnClipPoly	1	0	1
Blink$1	0	0	2
Messages	0	0	1
ScanConversion	0.88	0.93	34
LineClipping	0.71	0.83	15
PolygonClipping	0.7	0.79	26
ScanLineMain	0.69	0.76	69
Input	0.43	0.64	26
ScanPolygonMain	0.71	0.77	85
DrawClipPolygon	0.79	0.87	82
DrawClipLine	0.76	0.85	67

COHESION AND ASSOCIATION METRICS

CLASSES USED



Object

library::
Point

Fields

+ NO_OF_COLOURS : int
+ NO_OF_COORDINATES : int
+ bColour : int
– coordinates : int[]
+ gColour : int
+ rColour : int
– rgb : double[]
+ xCoord : int
+ yCoord : int
+ zCoord : int

Properties

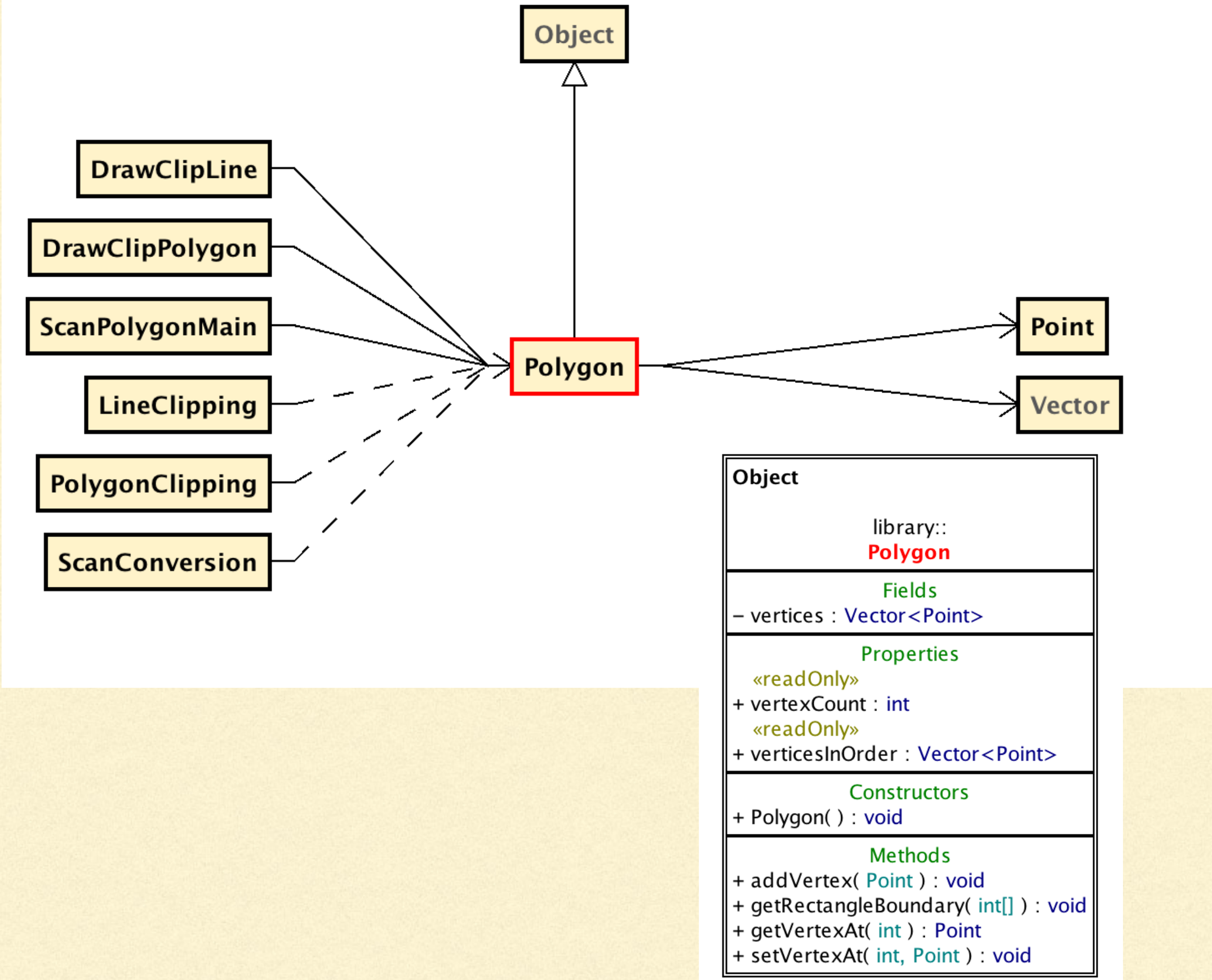
«readOnly»
+ x : int
«readOnly»
+ y : int
«readOnly»
+ z : int

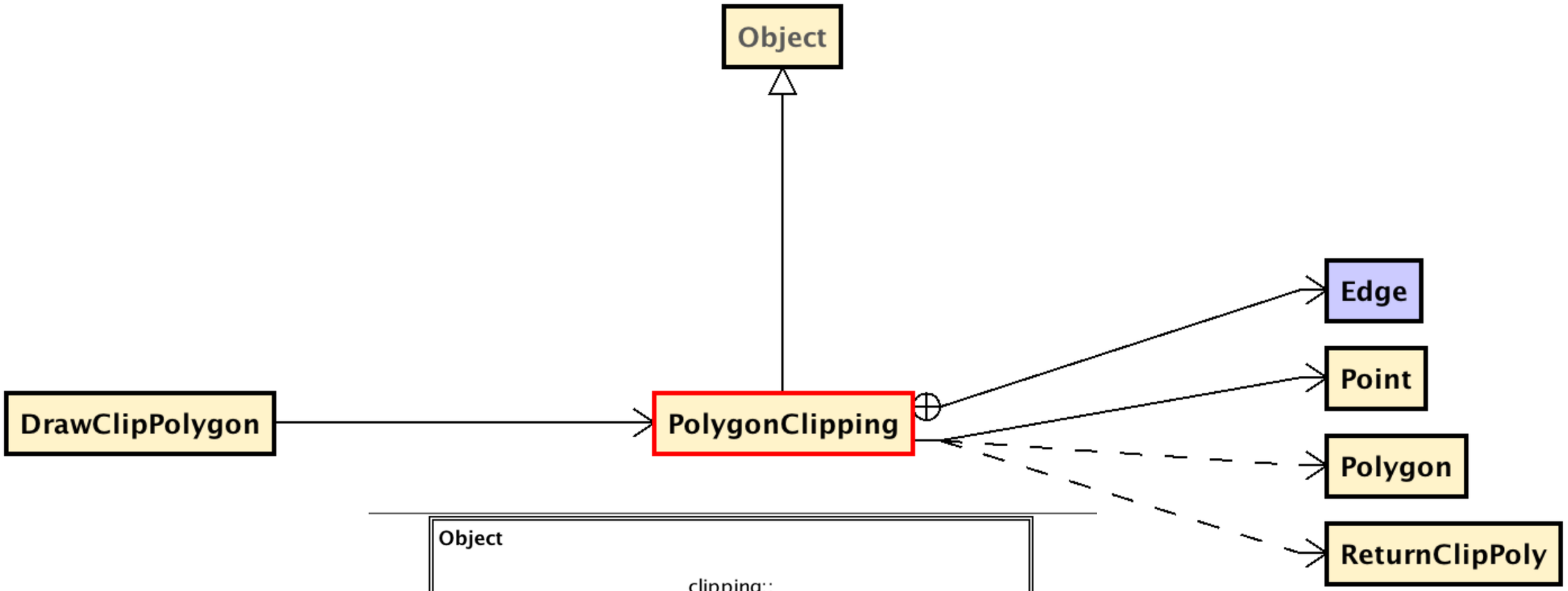
Constructors

+ Point( ) : void
+ Point(int, int) : void
+ Point(int, int, double, double, double) : void
+ Point(int, int, int) : void
+ Point(int, int, int, double, double, double) : void
+ Point(Point) : void

Methods

+ getColour(double[]) : void
+ getCoordinates(int[]) : void
+ setPoint(int, int) : void
+ setPoint(int, int, int) : void
+ setPoint(Point) : void


Object
DrawClipLine
DrawClipPolygon
ScanPolygonMain
LineClipping
PolygonClipping
ScanConversion
Polygon
Point
Vector
Object
library::
Polygon
Fields
– vertices : Vector<Point>
Properties
«readOnly»
+ vertexCount : int
«readOnly»
+ verticesInOrder : Vector<Point>
Constructors
+ Polygon( ) : void
Methods
+ addVertex(Point) : void
+ getRectangleBoundary(int[]) : void
+ getVertexAt(int) : Point
+ setVertexAt(int, Point) : void



Object

clipping::
PolygonClipping

Fields

«synthetic»
− $SWITCH_TABLE$clipping$PolygonClipping$Edge : int[]
− ObjectLimit_x : int
− ObjectLimit_y : int
~ RenderPolygon : int[][]
~ cnt : int
− counter : int
~ pOut : Point[]

Constructors

~ PolygonClipping( ) : void

Methods

«synthetic»
~ $SWITCH_TABLE$clipping$PolygonClipping$Edge( ) : int[]
+ Clip(Polygon, Polygon) : int[][]
+ ClipPoint(Point, Edge, int, int, int, int, Point) : ReturnClipPoly
− cross(Point, Point, Edge, int, int, int, int) : int
− inside(Point, Edge, int, int, int, int) : int
− intersect(Point, Point, Edge, int, int, int, int) : Point
− lookUpEdge(int) : Edge
~ saveData(int, Point, int) : void
~ savePoint(int, int, int, int, int, int, int) : void

PROJECT IN JAVASCRIPT

```
<!DOCTYPE HTML>
<html>
<head>
  <script type="text/javascript" src="canvasjs.min.js"></script>
  <script type="text/javascript">

      //THIS
      var clipwindow = function(){
          this.chart =null;
          this.xmin = 0;
          this.xmax = 0;
          this.ymin = 0;
          this.ymax = 0;
          this.rowmindps = [];
          this.rowmaxdps = [];
          this.colmindps = [];
          this.colmaxdps = [];
          this.polydps =[];
          this.clippolydps = [];
          // this.clippolydps = [];
          this.currentLine = [];
          this.initializechart = function (){
              this.chart = new CanvasJS.Chart("chartContainer", {
              axisX:{
                      minimum: 0,
                      maximum: 400

                    },
              axisY:{
                      minimum: 0,
                      maximum: 400,
                      gridThickness : 0
                    },
              data:[{
                      type: "line",
                      color: "DarkSlateBlue",
                      indexLabel: "ymin : {y}",
                      indexLabelOrientation: "vertical",
                      indexLabelFontSize: 12,
                      dataPoints: this.rowmindps
                    },
```

```
s.drawcanvas= function(){
  if((this.rowmaxdps.length & this.rowmindps.length & this.colmaxdps.length & this.colmindps.length) !=0)  {
      console.log('free dps data 1');
      this.rowmindps.length =0;
      this.rowmaxdps.length =0;
      this.colmindps.length =0;
      this.colmaxdps.length =0;
  }
  // bounding box
  this.colmindps.push({x:this.xmin,y:0},{x:this.xmin,y:400});
  this.colmaxdps.push({x:this.xmax,y:0},{x:this.xmax,y:400});
  this.rowmindps.push({x:0,y:this.ymin},{x:400,y:this.ymin});
  this.rowmaxdps.push({x:0,y:this.ymax},{x:400,y:this.ymax});

  //Polygon
  if(this.polydps.length==0)
    this.polydps.push({x:50, y:200}, {x:150, y:350}, {x:200, y:200}, {x:250, y:350}, {x:350, y:200}, {x:200, y:50},

      // this.polydps.length = 0;
      // this.clipdatadps.length = 0;
      this.clippolydps.length = 0;
      // this.finalclipdps.length = 0;

 this.chart.render();

.defaultwindow = function(){
document.getElementById("xmin").value = 100;
document.getElementById("ymin").value = 100;
document.getElementById("xmax").value = 300;
document.getElementById("ymax").value = 300;
this.xmin = parseInt(document.getElementById("xmin").value);
this.xmax = parseInt(document.getElementById("xmax").value);
this.ymin = parseInt(document.getElementById("ymin").value);
this.ymax = parseInt(document.getElementById("ymax").value);
this.initializechart();
this.drawcanvas();
}
```

```
//THIS
document.getElementById("next").addEventListener("click",fu
  // alert(boxiterator.getIndex());
   if(line<5) {
         // alert(polyiterator.getIndex());
         var pr = "clippoly: ";
         for(var i=0; i<objwindow.clippolydps.length; i++) {
           pr += objwindow.clippolydps[i].x + "," + objwindow.clippolydps[i
         }
         console.log(pr);
         if(polyiterator.hasNext()) {
             if(polyiterator.hasPrevious()==false) {
               p0 = polyiterator.first();
               p1 = polyiterator.next();
             }
             else {
               p0 = p1;
               p1 = polyiterator.next();
             }
             // alert(p0.x);
             //Sutherland Algorithm
             var c = cross(line, p0.x, p0.y, p1.x, p1.y, start[0], start[1],
             // objwindow.currentLine = [{x:p0.x,y:p0.y},{x:p1.x,y:p1.y}];
             // console.log(c);
```

```
                // if(this.clippingRequired())
                //     {
                //          this.slope = (end[1]-start[1])/(end[0]-start[0]);

                //   }
            }
        };
        //THIS
        var Iterator = function(items) {
           this.index = 0;
           this.items = items;
           this.first = function() {
              this.reset();
              return this.next();
            },
           this.next = function() {
```

EXAMPLE TEST CASES FOR JS CODE

Lab Name	Feature	Requirements	Test Step Id	Test Case Type	Test Description	Test Steps	Expected Result	Status	Test Case Owner	Pre/Post Conditions	Data/Environment Required
Computer Graphics Lab - Polygon Clipping Experiment	Position of a point wrt. a side of rectangle	Function should output whether point is on the inside of line(towards the bounding box) or outside	1	Positive	For each side of the rectangle(left,right,bottom,top) check for interior and exterior points	1. Construct a rectangle 2. Test for points on interior(towards the bounding box) and exterior of each side of rectangle	1.Interiority/exteriority of the points are predicted correctly	Reviewed	Pritam	NONE	OS: Windows 7, Linux Browsers: Firefox,Chrome Bandwidth : 100Mbps Hardware Configuration:8 GBRAM , Processor:i5
Computer Graphics Lab - Polygon Clipping Experiment	Position of pair of points wrt. line	Function should output whether the points are both exterior/both interior/one interior and other exterior(interior means towards the bounding box wrt.line)	1	Positive	For each side of the rectangle test for all the 4 possible combinations of pair of points	1. Construct a rectangle 2. Choose pair of points for all the 4 possible cases on each side of the rectangle	1.Output correctly which of the four cases a point belongs to.	Reviewed	Pritam	NONE	OS: Windows 7, Linux Browsers: Firefox,Chrome Bandwidth : 100Mbps Hardware Configuration:8 GBRAM , Processor:i5
Computer Graphics Lab - Polygon Clipping Experiment	Intersection Point of line joining two points with a side of rectancle	Funciton should return the intersection point of line joining two points with a side of	1	Positive	For each side of the rectangle choose points that lie on either side of the rectangle's side	1. Construct a rectangle 2. Choose pair of points for each side of the rectangle that lie on opposite	Output correctly the co-ordinates of the point of intersection	Reviewed	Pritam	Refer to the test step id 1	OS: Windows 7, Linux Browsers: Firefox,Chrome Bandwidth : 100Mbps Hardware