25.2.4 Klassen Rect und RectF

Die Klasse *Rect (gb.clipper)* repräsentiert ein Rechteck mit *ganzzahligen* Koordinaten. In diesem Kapitel werden Eigenschaften und Methoden der Klasse *Rect* sowie einige Beispiele für die Verwendung der Klasse vorgestellt.

25.2.4.1 Eigenschaften

Die Klasse *Rect* verfügt über diese Eigenschaften:

Eigenschaft	Beschreibung
X AS Integer	Setzt die die x-Koordinate des Rechtecks oder gibt diesen Wert zurück.
Y AS Integer	Setzt die y-Koordinate des Rechtecks oder gibt diesen Wert zurück.
Height As Integer *oder* H As Integer	Setzt die Höhe des Rechtecks oder gibt diesen Wert zurück.
Width As Integer *oder* W As Integer	Setzt die die Breite des Rechtecks oder gibt diesen Wert zurück.
Left As Integer	Setzt die Position des linken Randes des Rechtecks oder gibt diesen Wert zurück. Im Gegensatz zur X-Eigenschaft ändert dieser Wert die Breite des Rechtecks, da die rechte Begrenzung des Rechtecks sich nicht verändert.
Right As Integer	Setzt die Position des rechten Randes des Rechtecks oder gibt diesen Wert zurück.
Top As Integer	Setzt die Position des oberen Randes des Rechtecks oder gibt diesen Wert zurück. Im Gegensatz zur Y-Eigenschaft ändert dieser Wert die Höhe des Rechtecks verändern, da die untere Begrenzung des Rechtecks sich nicht verändert.
Bottom As Integer	Setzt die Position des unteren Randes des Rechtecks oder gibt diesen Wert zurück.

Tabelle 25.2.4.1.1 : Eigenschaften der Klasse Rect

25.2.4.2 Ausgewählte Methoden

Für die Klasse *Rect* werden hier die Methoden beschrieben.

Methode	Beschreibung
Center( ) As Point	Gibt die Koordinaten des Mittelpunktes des Rechtecks als Schnittpunkt der Diagonalen zurück.
Clear( )	Leert das Rechteck. Alle Werte werden auf 0 gesetzt.
IsVoid( ) As Boolean	Gibt *True* zurück, wenn das Rechteck leer ist.
Copy( ) As Rect	Gibt eine Kopie des Original-Rechtecks zurück.
Translate(DX As Integer, DY As Integer)	Verschiebt das Rechteck (relativ zu den originalen Koordinaten) um DX und DY.
Move(X As Integer, Y As Integer [, Width As Integer, Height As Integer])	Verschiebt das Rechteck an die neuen Koordinaten Y, Y (BasePoint) und ändert (optional) auch die Weite und die Höhe des Rechtecks.
Resize(Width As Integer, Height As Integer)	Ändert nur die Breite und Höhe des Rechtecks.
Adjust(Left As Integer [, Top As Integer, Right As Integer, Bottom As Integer])	Verkleinert oder vergrößert das Rechteck. Ein positiver Wert bewirkt eine Verkleinerung während ein negativer für eine Vergrößerung sorgt. Wenn kein Wert für Top festgelegt wird nimmt er den Wert von Left an. Wenn kein Wert für Right festgelegt wird, nimmt er den Wert von Left an. Wenn kein Wert für Bottom festgelegt wird, nimmt er den Wert von Top an.
Contains(X As Integer, Y As Integer) As Boolean	Gibt *True* zurück, wenn der Punkt P(X|Y) im Rechteck liegt.

Methode	Beschreibung
Intersection(Rect1 As Rect) As Rect	Gibt ein Rechteck zurück, wenn sich das Rechteck Rect1 mit dem aktuellen Rechteck in einem Rechteck (Durchschnitt) überschneidet. Wenn die beiden Rechtecke sich nicht überschneiden, so wird NULL zurückgegeben.
Union(Rect1 As Rect) As Rect	Gibt das (kleinste) umschließende Rechteck zurück, das die beiden Rechtecke Rect1 und das aktuelle Rechteck bilden.

Tabelle 25.2.4.2.1 : Methoden der Klasse Rect

25.2.4.3 Beispiele

Die Klasse *Rect* ist erstellbar und kann wie eine (statische) Funktion benutzt werden:

```
Dim hRect As Rect
hRect = New Rect(100, 100, 300, 200)
```

Im nächsten Beispiel werden mehrere Rechtecke und ein Punkt P erzeugt sowie Methoden der Klassen Point und Rect eingesetzt. Die Ergebnisse der Operationen mit den Rechtecken und dem Punkt als Operanden werden in der Konsole ausgegeben oder visualisiert:

```
[1]  Public Sub ScriptRectangles()
[2]    Dim PointP As Point
[3]    Dim RectRed, RectGreen, RectCopy, RectIntersection, RectUnion As Rect
[4]
[5]    PointP = New Point(177, 144)
[6]    RectRed = New Rect(60, 60, 400, 180)
[7]    RectGreen = New Rect(130, 20, 90, 170)
[8]    RectIntersection = New Rect
[9]    RectUnion = New Rect
[10]
[11]   Print "P in RectRed? --> "; RectRed.Contains(PointP.X, PointP.Y)
[12]   Print "P in RectRed? --> "; PointP.InRect(RectRed)
[13]   Print "Cx = "; RectRed.Center().X; " , Cy = "; RectRed.Center().Y
[14]   RectCopy = RectRed.Copy()
[15]   Print "R3x = "; RectCopy.X; " , R3y = "; RectCopy.Y
[16]   RectCopy.Clear
[17]
[18]   RectIntersection = RectRed.Intersection(RectGreen)
[19]   Print "RIx = "; RectIntersection.X; " , RIy = "; RectIntersection.Y
[20]   Print "RIW = "; RectIntersection.Width; " , RIH = "; RectIntersection.Height
[21]   RectUnion = RectRed.Union(RectGreen)
[22]   Print "RIx = "; RectUnion.X; " , RIy = "; RectUnion.Y
[23]   Print "RIW = "; RectUnion.Width; " , RIH = "; RectUnion.Height
[24]
[25]   GenerateNewPicture()
[26]   SetPictureBorder()
[27]   Paint.Begin(hPicture)
[28]     Paint.Translate(xTranslate, yTranslate)
[29]     Paint.Scale(xScale, yScale) ' +y ▲
[30]     Paint.AntiAlias = False
[31]     DrawCoordinateSystem() ' +y ▲
[32]   ' Original-Koordinatensystem --> +y-Richtung nach unten
[33]   ' Rotes Rechteck
[34]     Paint.Brush = Paint.Color(Color.Red)
[35]     Paint.Rectangle(RectRed.X, RectRed.Y, RectRed.W, RectRed.H)
[36]     Paint.Fill(True)
[37]     Paint.Brush = Paint.Color(Color.DarkGray)
[38]     Paint.LineWidth = 2
[39]     Paint.Stroke
[40]   ' Grünes Rechteck
[41]     Paint.Brush = Paint.Color(Color.Green)
[42]     Paint.Rectangle(RectGreen.X, RectGreen.Y, RectGreen.W, RectGreen.H)
[43]     Paint.Fill(True)
[44]     Paint.Brush = Paint.Color(Color.DarkGray)
[45]     Paint.LineWidth = 2
[46]     Paint.Stroke
[47]   ' Blaues Rechteck (Intersektion - Schnittmenge)
[48]     Paint.Brush = Paint.Color(Color.Blue)
[49]     Paint.Rectangle(RectIntersection.X, RectIntersection.Y, RectIntersection.W, RectIntersection.H)
[50]     Paint.Fill(True)
[51]     Paint.Brush = Paint.Color(Color.DarkGray)
[52]     Paint.LineWidth = 2
[53]     Paint.Stroke
[54]   ' Umrandetes Rechteck (Union - Vereinigungsmenge)
[55]     Paint.Brush = Paint.Color(Color.Magenta)
[56]     Paint.Rectangle(RectUnion.X, RectUnion.Y, RectUnion.W, RectUnion.H)
```

```
[57]     Paint.Dash = [2, 2]
[58]       Paint.LineWidth = 2
[59]       Paint.Stroke
[60]     Paint.Dash = Null
[61]     Paint.FillRect(260, 150, 3, 3, Color.Black) ' Mittelpunkt C einzeichnen
[62]   ' Mittelpunkt C der Diagonalen im roten Rechteck beschriften
[63]     Paint.Scale(1, -1) ' +y ▼
[64]       Paint.Font = Font["Monospace, 10"]
[65]       Paint.Brush = Paint.Color(Color.Black)
[66]       Paint.DrawText("C(260|150)", 270, -145)
[67]       Paint.Font = Font["Monospace, 8"]
[68]     Paint.Scale(1, -1) ' +y ▲
[69]   Paint.End
[70]
[71] End ' ScriptRectangles()
```

Diese Ergebnisse werden in der Konsole in der IDE ausgegeben:

```
P in RectRed? --> True
P in RectRed? --> True
Cx = 260 , Cy = 150
R3x = 60 , R3y = 60
RIx = 130 , RIy = 60
RIW = 90 , RIH = 130
RIx = 60 , RIy = 20
RIW = 400 , RIH = 220
```

In der folgenden Abbildung können Sie diese Ergebnisse auch hinreichend genau ablesen:

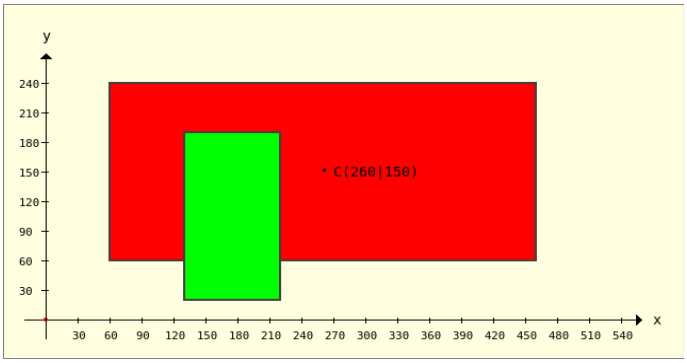

Abbildung 25.2.4.3.1: Gegenseitige Lage der Rechtecke RectRed und RectGreen

In den Zeilen 48 bis 61 werden die beiden Rechtecke zusätzlich eingezeichnet (blau und in der Farbe Magenta gestrichelt umrandet), die in den Zeilen 18 sowie 22 beim Einsatz der Methoden Rect.Intersection(..) und Rect.Union(..) aus den beiden Rechtecken RectRed und RectGreen erzeugt wurden:

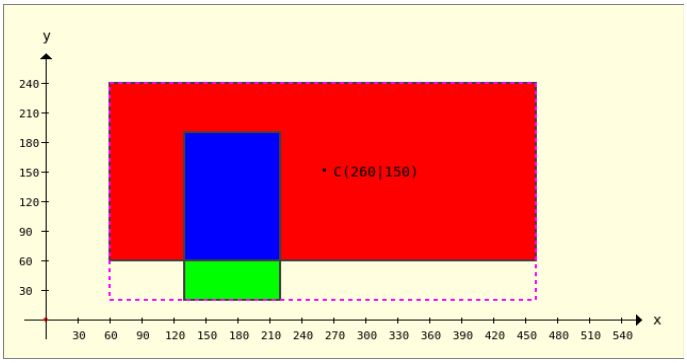

Abbildung 25.2.4.3.2: Erzeugte Rechtecke: Rect.Intersection(..) und Rect.Union(..)

25.2.4.4 Klasse RectF

Während für die Klasse *Rect* nur ganze Zahlen als Koordinaten zulässig sind, können Sie für die Klasse *RectF* reelle Zahlen für die Koordinaten X und Y sowie für die Weite und die Höhe verwenden.