10. Die `Selection`-Funktion

Die `Selection`-Funktion ist zwar keine Zeichen-Funktion im ursprünglichen Sinne, aber für eine ernsthafte CAD-Arbeit unbedingt erforderlich. Um ein bestimmtes, auf der graphischen Zeichenfläche (Grafikfenster) dargestelltes Grafik-Objekt bearbeiten zu können, muß es zunächst selektiert, bzw. markiert werden. Dazu kann man prinzipiell zwei Verfahren anwenden:

- mit der Mouse (möglichst genau) auf das betreffende Objekt clicken,
- mit der Mouse in die Nähe des `Graphic`-Objektes clicken und einen Rahmen (Rechteck) um das Grafik-Objekt, bzw. auch nur einen Teil des Grafik-Objektes ziehen.

Während das erste Verfahren vom Handling her sehr einfach erscheint, ist dessen programmtechnische Umsetzung mit erheblichen Schwierigkeiten verbunden.
Die zweite Möglichkeit ist vom Ansatz her weniger problematisch, was allerdings nicht heißt, daß die Programmierung besonders einfach sein wird.

Zur Kennzeichnung, daß ein `Graphic`-Objekt selektiert-markiert ist, wird dieses in einer anderen Farbe – bevorzugt „Rot“ – dargestellt. Als weiteres Kennzeichen kann man noch die Anfangs- und Endpunkte, bzw. auch andere `Vertices` (ein `Vertex` – Mehrzahl `Vertices` - ist ein bestimmter Punkt mit `X-, Y-` und ggf. auch `Z-`Koordinaten, z.B. bei einer Linie ein Endpunkt, oder auch der Linien-Mittelpunkt) bei komplizierteren Objekten durch besondere Merkmale (andersfarbige kleine Kreise oder Rechtecke) zusätzlich darstellen.
Bei einigen Objekten mit besonderer Ausformung, z.B. Kreisen, Elipsen o.ä., aber auch bei sonst „normalen“ Objekten - wenn sie nicht rechtwinklig im Koordinatensystem liegen - wird in vielen kommerziellen CAD-Programmen die Kennzeichnung der Selektion durch ein zusätzlich virtuelles Rechteck, welches rechtwinklig im Koordinatensystem liegt, mit eben diesen `Vertices` an den Ecken unterstützt.
Außerdem kann zusätzlich noch ein so genannter ***Bezugspunkt*** festgelegt werden.
Vorgegeben ist hierbei vielfach der Mittelpunkt des `Graphic`-Objektes, also z.B. bei der Linie eben der Mittelpunkt der Linie. Für viele Bearbeitungsmöglichkeiten wäre allerdings ein anderer Punkt, z.B. ein Endpunkt sinnvoller. Dann ist es besonders einfach, eine neue Länge, oder einen Drehwinkel zu definieren. Daraus folgt, daß die Lage des Bezugspunktes veränderbar sein muß und sich ggf. auch außerhalb des eigentlichen `Graphic`-Objektes befinden kann. Die diesbezüglichen Koordinaten des Bezugspunktes müssen deshalb mit in dem betreffenden Datensatz für das `Graphic`-Objekt im Parameter-Array abgelegt werden.

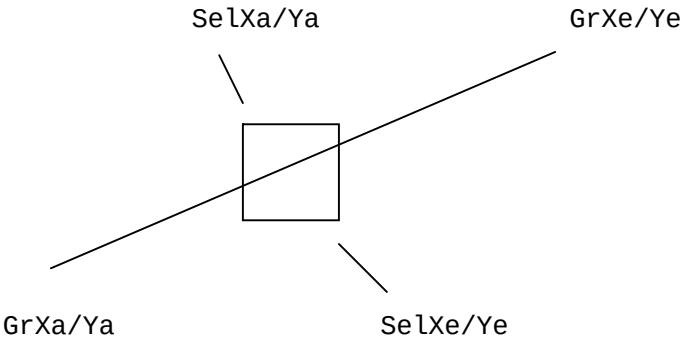
10.1. Verfahrensbeschreibung (Auszug)

Soll ein bestimmtes, im Grafik-Fenster sichtbares Grafik-Objekt selektiert (markiert) werden, so wird durch Anclicken in der Nähe des Grafik-Objektes und Ziehen mit der `Mouse` ein entsprechend großes, oder kleines Rechteck erzeugt. Dieses hat ebenso, wie bei der „Rechteck“-Funktion (siehe dazu Abschnitt 13) `X/Y`-Anfangs- und Endkoordinaten. Auch diese werden wiederum in den Parameter-Fenstern `xa, ya, xe`, und `ye` angezeigt, nur daß diesmal keine Eingabe möglich ist.

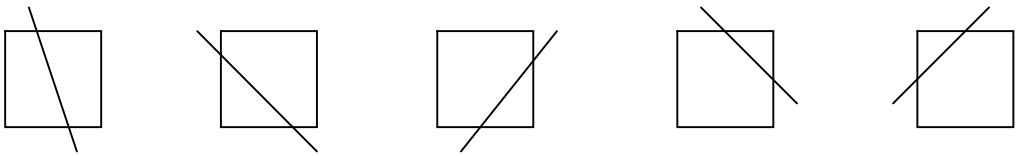
Anhand dieser `X/Y`-Parameter muß nun im Parameter-Array gesucht werden, welches der dort abgelegten Grafik-Objekte mit irgendeinem seiner `X/Y`-Parameter – d.h. nicht nur `Xa/Ya` bzw. `Xe/Ye` - in das vom `Mouse`-Rechteck beschriebene Fenster passt.

Dabei können z.B. folgende `Selection`-Fälle vorkommen:

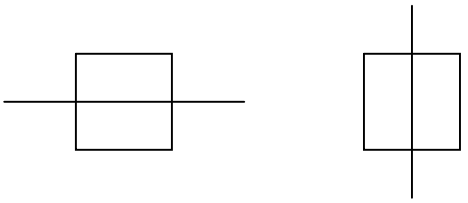
(1)



(2 - 6) Varianten dazu



(7 - 8) Spezialfälle



Alle diese Fälle haben ein gemeinsames Kriterium:
Es gibt (beim Grafik-Objekt `LINE`) immer zwei Schnittpunkte mit dem `Selection`-Rechtecks, je nach dem – entweder mit beiden Waagerechten, bzw. Senkrechten, oder mit einer Senkrechten und einer angrenzenden Waagerechten des `Selection`-Rechtecks.
Die Waagerechten werden immer von `Ya` bzw. `Ye` und die Senkrechten immer von `Xa` bzw. `Xe` des `Selection`-Rechtecks bestimmt.
Anhand der Geradengleichung (gilt nur für das Grafik-Objekt `LINE`) kann jeder Punkt auf der Geraden bestimmt werden.

```
f(y) = m * x + n
```

oder für x

```
f(x) = y – n / m
```

Im Prinzip müsste es ausreichen, wenn ein X- oder Y-Parameter des `Selection`-Rechtecks mit einem Punkt auf der Geraden übereinstimmt.
Diese Abfragen müssen nun über alle im Parameter-Array eingetragenen Grafik-Objekte in einem Iterationsverfahren durchgeführt werden. Wird eine Bedingung erfüllt, so ist anzunehmen, daß es sich um das zu selektierende Grafik-Objekt handelt. Demzufolge wird das im Parameter-Array entsprechend gekennzeichnet, d.h. `$X[$R-1][3] = True.`
Vorher wird noch geprüft, ob bereits ein anderes Grafik-Objekt selektiert war - das wird dann wieder zurückgenommen (incl. der damit zusammenhängenden Grafik-Maßnahmen).

Denkbar wäre allerdings auch eine Mehrfach-Selektion, welche z.B. mit einer speziellen Tastenkombination vorgenommen werden könnte. Dieses Feature ist hier (zunächst) nicht vorgesehen.

In der nachfolgend aufgerufenen `NewDraw()`-Routine wird das Grafik-Fenster aktualisiert. Dabei wird geprüft, ob ein Grafik-Objekt als `selected` eingetragen ist. Für diesen Fall ist dann die im Parameter-Array eingetragene Farbe nicht relevant und es wird z.B. „Rot“ an `DrawLine()` übergeben. Außerdem werden für den Anfangs- und Endpunkt der `Line` kleine Markierungskreise generiert (siehe dazu Pkt. 10.2.7).

11. Eigenschaften-Dialogform

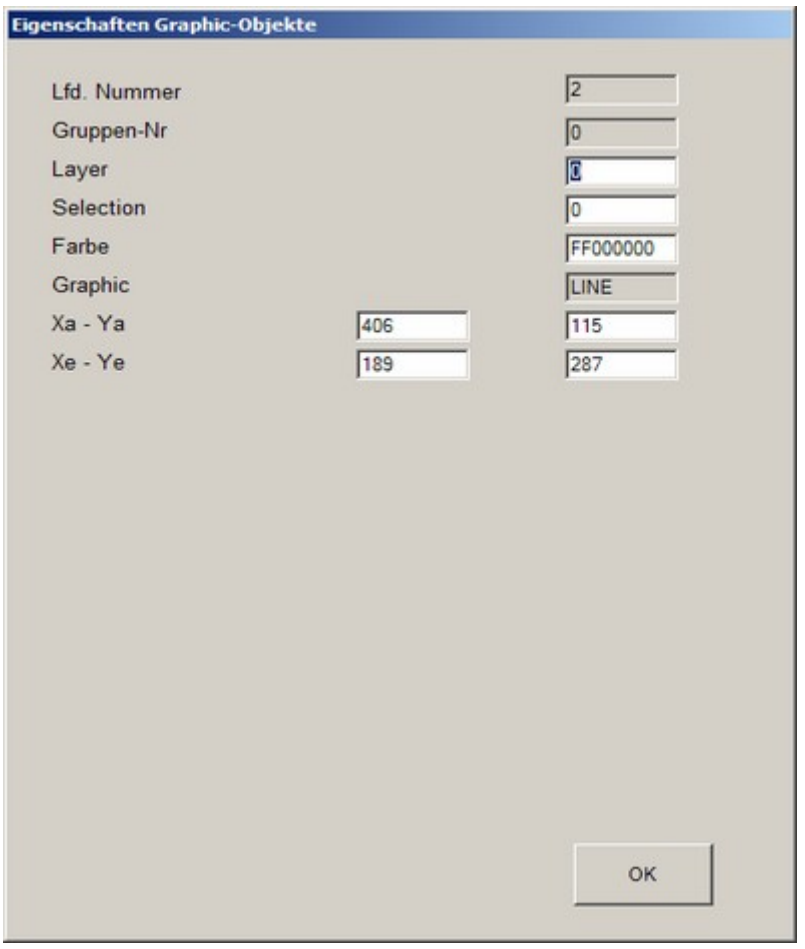
Nachdem mit der `Selection`-Funktionalität ein `Graphic`-Objekt aus der Bildschirm-Darstellung ausgewählt wurde, kann mittels einer sog. „Eigenschaften-Dialogform“ auf alle im Parameter-Array abgelegten Eigenschaften des betreffenden `Graphic`-Objekts zugegriffen werden.

Dazu wird das mit der „Rechten Mousetaste“ erzeugte `$GUI_EVENT_SECONDARYDOWN` in einen weiteren `Case`-Zweig des UP `SpecialEvents()` untergebracht.

```
. . .
Case @GUI_CtrlId = $GUI_EVENT_SECONDARYDOWN
      ; MsgBox(0, "Rechte Mousetaste gedrückt", "ID=" & @GUI_CtrlId & _
      " WinHandle=" & @GUI_WinHandle)                ; nur zum Test
      Local $i1
            For $i1 = 1 To UBound($X) - 1            ; im Parameter-Array suchen
                  If $X[$i1][3] = 1 Then             ; ein Grafik-Objekt selektiert?
                        ;
                        GraphProperties($i1)
                        ; hier Aufruf der 2. GUI mit dem Eigenschaftsdialog für Grafikobjekte
                        ;
                  EndIf
                  ;
            Next
            ;
      EndSelect
```

Der erste Versuch des UP `GraphProperties($i1)` sieht dann so:

```
Func GraphProperties($i1)
;
Local $Properties, $Label1Prop, $Input1Prop, $Button1Prop, $msg
;
$Properties = GUICreate("Eigenschaften Graphic-Objekte", 450, 600, 500, 300, _
            BitOR($WS_SIZEBOX, $WS_SYSMENU, $WS_MAXIMIZEBOX, $WS_MINIMIZEBOX))
$Label1Prop = GUICtrlCreateLabel("Lfd. Nummer", 24, 24, 95, 24)
GUICtrlSetFont(-1, 12, 400, 0, "Arial")
$Input1Prop = GUICtrlCreateInput("Input1", 320, 24, 65, 21, $ES_READONLY)
$Button1Prop = GUICtrlCreateButton("OK", 328, 520, 81, 41, 0)
;
GUISetState(@SW_SHOW)
```

```
While 1
	$msg = GUIGetMsg()
	; MsgBox (0, "ID", $msg)                    ; nur zum Test
	Select
		Case $msg = $GUI_EVENT_CLOSE          ; funktioniert so nicht!
		ExitLoop
		;
		Case $msg = $Button1Prop              ; dito!
		ExitLoop
		;
	EndSelect
	;
WEnd
;
EndFunc
```

Leider funktioniert die in der `While-WEnd`-Schleife eingebettete `Select-Case`-Abfrage nicht so, wie erwartet, wobei jedoch die `While-WEnd`-Schleife ist für eine dauerhafte Anzeige der Dialog-Form notwendig ist.