Drawing arcs

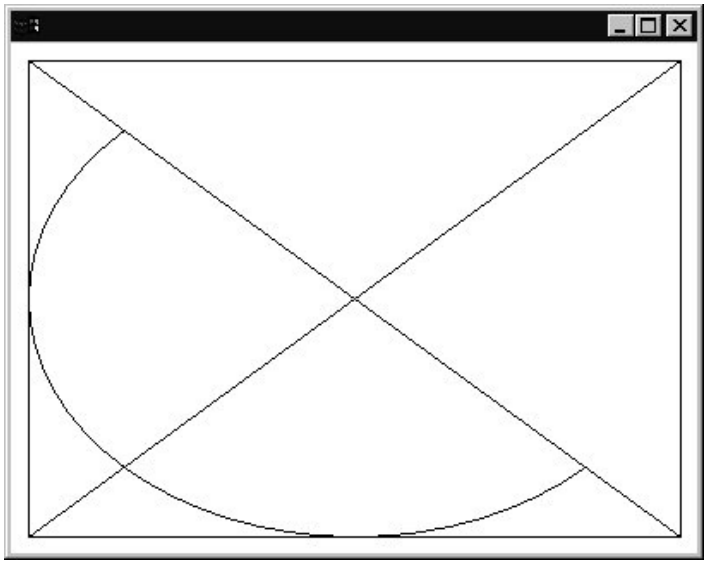
`DrawArc(x, y, width, height, start, extent)` draws an arc that is "inscribed" in the rectangle specified by the first four parameters.

`start` and `extent` specify the starting position and angular extent of the figure, just like `DrawCircle`.

Here is an example from the text, page 84:

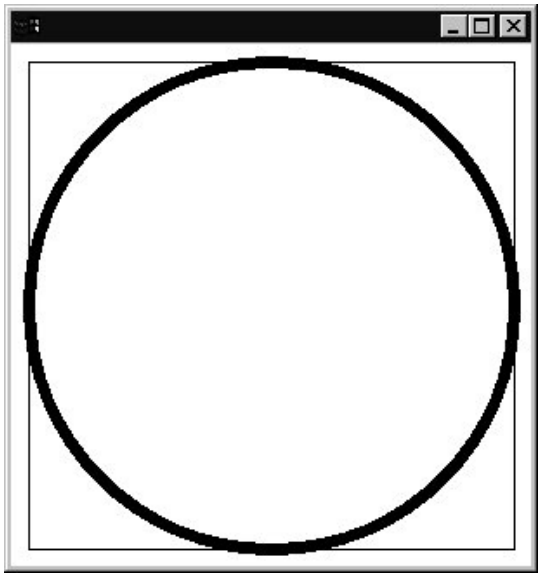
```
procedure main() # arc1, from GPiI, p.84
    WOpen("size=400,300")

    DrawRectangle(10, 10, 380, 280)

    DrawLine(10,10, 390, 290)
    DrawLine(10, 290, 390, 10)

    DrawArc(10, 10, 380, 280, &pi/4, &pi)
    WDone()
end
```

Result:



Drawing arcs, continued

Inscribing 2π arc in a square produces a circle:



Note that the thick stroke is centered on the bounding rectangle. Here's the code:

```
procedure main(args) # arc2
    WOpen("size=300,300")

    DrawRectangle(10, 10, 280, 280)

    WAttrib("linewidth=7")

    DrawArc(10, 10, 280, 280, 0, &pi*2)

    WDone()
end
```

Mnemonic aid for the order of `start` and `extent`: "The start comes first."

Sidebar: The `case` statement

Icon's `case` statement provides for execution of a block of code based on a discriminating value, much like `switch` in Java.

A simple example:

```
procedure main()

    every i := ![2,1,3,4] do {

        case i of {
            1: { write("first") }
            2: { write("second") }
            3: { write("third") }
            default: {
                write("other")
                notify_support(i)
                }
            }
        }
end
```

Output:

```
second
first
third
other
```

Note that the element following the colon is an expression. In the above example the braces are optional in the first three case clauses.

The `default` clause is optional. If omitted and no value matches, the statement fails.

The `case` statement, continued

Note that the case selectors do not need to be constants:

```
procedure main()
    writes("x? ")
    x := read()
    writes("y? ")
    y := read()

    while line := read() do {
        case line of {
            x: write("Looks like an x...")
            y: write("It's a y!")
            default: write("Hmm...")
            }
        }
end
```

Interaction:

```
x? 3
y? 7
1
Hmm...
3
Looks like an x...
6
Hmm...
7
It's a y!
10
Hmm...
```

The `case` statement, continued

A selector may be an arbitrary expression, and be generative:

```
procedure main()
    every c := !read() do {
        what := case c of {
            !&lcase:      "L"
            !&ucase:      "U"
            !&digits:     "D"
            "."|","|"?":  "P"
            whitespace(): "W"
            "\n": c
            default: "?"
            }
        writes(what)
        }
    write()
end

procedure whitespace()
    suspend !" \t"
end
```

Interaction:

```
Line?
Test #3??     (input)
ULLLW?DPP
```

Note that selection is done using exact equality (===).

```
][ case 1 of { "1": "yes" };
Failure

][ case 1 of { 1: "yes" };
   r := "yes"  (string)
```