Graphic object creating - examples

Non-linear retrieving of object color

An index of logical palette of imported file is an input parameter. The function returns the index of color from logical palette which is used in D2000 System.

```
function OutputColor(x:integer) return integer is
        c:integer:=Abs(x);
        w:integer:=c mod 10;
        odtien:integer;
begin
        case w is
        when 0 => color := 4;
        when 1 => color := 2;
        when 2 => color := 5;
        when 3 => color := 3;
        when 4 => color := 7;
        when 5 => color := 8;
        when 6 => color := 9;
        when 7 => color := 11;
        when 8 => color := 14;
        when 9 => color := 15;
        when others => color := 4;
        end case;
        --
        case c is
        when 0.. 9    => return colorTable1(color);
        when 10.. 39  => return CLR_RED (color);
        when 40.. 69  => return CLR_YELLOW (color);
        when 70..119  => return CLR_GREEN (color);
        when 120..139 => return CLR_CYAN (color);
        when 140..189 => return CLR_BLUE (color);
        when 190..229 => return CLR_PINK (color);
        when 230..249 => return CLR_RED (color);<
        when 250..255 => return colorTable2(c);
        when others   => return CLR_BLUE(1);
        end case;

end OutputColor;
```

Object - Line

Creating	: *CreateObj(cLine)*
Parameters	: *position*
	: line color
	: line style
	: line width
	: line end

Example:

```
        procedure Create_Line(x1,y1,x2, y2:long_float;color,linestyle,thickness:integer) is
        begin
                CreateObj(cLine);
                --
                Set_float2 (cPosXY, x1, y1);
                Set_float2 (cPosXY, x2, y2);
                Set_integer(cLineColorIdx, color);
                Set_integer(cLineStyle, linestyle);
                Set_integer(clineWidth, thickness);
                --
                ObjAction(closeObject);
        end;
```

Special object - Line or Disjointed multiline

This example shows how to create Line that is combined with other lines (if they have the same attributes) to one object Multiline or Disjointed multiline. It is an optimizing function to reduce the number of objects and the size of picture (too big picture can not be saved). It can be used when cross-hatching the objects in Autocad.

Creating	: *CreateObj(*cLineCombined*)*
Parameters	: *position*
	: line color
	: line style
	: line width
	: line end

Example:

```
procedure Create_CombinedLine(x1,y1,x2, y2:long_float; color,linestyle,thickness:integer) is

begin
        CreateObj(cLineCombined);
        --
        Set_float2 (cPosXY, x1, y1);
        Set_float2 (cPosXY, x2, y2);
        Set_integer(cLineColorIdx, color);
        Set_integer(cLineStyle, linestyle);
        Set_integer(clineWidth, thickness);
         --
        ObjAction(closeObject);
end;
```

Object - Multiline or Polygon

Creating	: *CreateObj(*cPLine*)*
Parameters	: *position*
	: line color
	: line style
	: line width
	: line end
	: line join

Example:

```
procedure Create_Polyline( posArr : TLong_Float_Arr; color, linestyle, thickness : integer) is
   pos : integer := posArr'first;
begin
            CreateObj(cPLine);
           --
           while pos <= posArr'last-1 loop
            Set_float2 (cPosXY, posArr(pos), posArr(pos+1));
       pos:= pos+2;end loop;
            Set_integer(cLineColorIdx, color);
            Set_integer(cLineStyle, linestyle);
            Set_integer(clineWidth, 1);
            --
            objAction(closeObject);
end;
```

Special object - Multiline or Polygon

Creating	: *CreateObj(*cPLineAdd*)*
Parameters	: *position*
	: line color
	: line style
	: line width
	: line end

	: line join

Example:

```
procedure Create_PLineItem( x1, y1, x2 , y2 : long_float;
                color, linestyle, thickness : integer) is
begin
        CreateObj(cPLineAdd);
        --
        Set_float2 (cPosXY, x1, y1);
        Set_float2 (cPosXY, x2, y2);
        Set_integer(cLineColorIdx, color);
        Set_integer(cLineStyle, linestyle);
        Set_integer(clineWidth, thickness);
end;
```

In the previous example, when creating the Multiline, all position points are known. In this example, you know only data of one line and do not know whether this is the first one. A definition means to add the parameters into the Multiline if other parameters are the same except the position ones. If the Multiline is not created when adding the parameters, then create it according to its parameters.

The lines should be end as follows:

Completing:

```
procedure CLose_PLine is
begin
        objAction(closeObject);
end;
```

Object - Disjointed multiline

Creating	: *CreateObj(*cDLine*)*
Parameters	: *position*
	: line color
	: line style
	: line width
	: line end
	: line join

Example:

```
procedure Create_Polyline( posArr : TLong_Float_Arr; color, linestyle, thickness : integer) is
        pos : integer := posArr'first;
begin
        CreateObj(cPLine);
        --
        while pos <= posArr'last-1 loop
                Set_float2 (cPosXY, posArr(pos),posArr(pos+1));
                pos:= pos+2;
        end loop;
        Set_integer(cLineColorIdx, color);
        Set_integer(cLineStyle, linestyle);
        Set_integer(clineWidth, 1);
        --
        objAction(closeObject);
end;
```

Object - Arc

Creating	: *CreateObj(*cArc*)*
Parameters	: *position*
	: line color
	: line style
	: line width
	: line end

	: line join
	: radius
	: start angle
	: end angle

Example:

```
procedure Create_Arc(x,y,radius,stangle,endangle:long_float; color,linestyle,thickness:integer) is
begin
            CreateObj(cArc);
            --
            Set_float2 (cPosXY, x, y);
            Set_float (cCircleRadial, radius*gr_scall);
            Set_float (cCircleAngleRadStart, stangle);
            Set_float (cCircleAngleRadEnd, endangle);
            Set_integer(cLineColorIdx, color);
           Set_integer(cLineStyle, linestyle);
            Set_integer(clineWidth, thickness);
            --
           objAction(closeObject);
end;
```

Object 3-poin arc

Creating	: *CreateObj(*c3Arc*)*
Parameters	: *position*
	: line color
	: line style
	: line width
	: line end

Example:

```
procedure Create_3Arc ( x1,y1,x2,y2,x3,y3:long_float; color, linestyle, thickness : integer) is
begin
        CreateObj(cBox);
        --
        Set_float2 (cPosXY, x1, y1);
        Set_float2 (cPosXY, x2, y2);
        Set_float2 (cPosXY, x3, y3);
        Set_integer(cLineColorIdx, color);
        Set_integer(cLineStyle, linestyle);
        Set_integer(clineWidth, 1);
        --
        objAction(closeObject);
end;
```

Object - Rectangle

Creating	: *CreateObj(*cBox*)*
Parameters	: *position*
	: line color
	: line style
	: line width
	: line join
	: line color
	: pattern

Example:

```
procedure Create_Box ( x1,y1,x2,y2:long_float; color, linestyle, thickness : integer) is
begin
        CreateObj(cBox);
        --
        Set_float2 (cPosXY, x1, y1);
        Set_float2 (cPosXY, x2, y2);
        Set_integer(cLineColorIdx, color);
        Set_integer(cLineStyle, linestyle);
        Set_integer(clineWidth, 1);
        --
        objAction(closeObject);
end;
```

Object - Polygon

Creating	: *CreateObj(*cPAngle*)*
Parameters	*: position*
	: line color
	: line style
	: line width
	: line join
	: fill color
	: pattern

Example:

```
procedure Create_Polygon ( posArr : TLong_Float_Arr; color, linestyle, thickness : integer) is
        pos : integer := posArr'first;
begin
        CreateObj(cPAngle);
        --
        while pos <= posArr'last-1 loop
                Set_float2 (cPosXY, posArr(pos),posArr(pos+1));
                pos:= pos+2;
        end loop;
        Set_integer(cLineColorIdx, color);
        Set_integer(cLineStyle, linestyle);
        Set_integer(clineWidth, 1);
        --
        objAction(closeObject);
end;
```

Object - Circle

Creating	: *CreateObj(*cCircle*)*
Parameters	*: position*
	: line color
	: line style
	: line width
	: fill color
	: pattern
	: radius

Example:

```
procedure Create_Circle(x,y,radius : long_float;color,linestyle,thickness:integer) is
begin
        CreateObj(cCircle);
        --
        Set_float2 (cPosXY, x, y);
        Set_float   (cCircleRadial, radius*gr_scall);
        Set_integer(cLineColorIdx, color);
        Set_integer(cLineStyle, linestyle);
        Set_integer(clineWidth, thickness);
        --
        objAction(closeObject);
end;
```

Object - Circle sector

Creating	: *CreateObj(*cPiArc*)*
Parameters	*: position*
	: line color
	: line style
	: line width
	: line end
	: fill color
	: pattern
	: radius
	: start angle
	: end angle

Example:

```
procedure Create_PiArc(x,y,radius,stangle,endangle:long_float; color,linestyle,thickness:integer) is
begin
        CreateObj(cPiArc);
        --
        Set_float2 (cPosXY, x, y);
        Set_float (cCircleRadial, radius*gr_scall);
        Set_float (cCircleAngleRadStart, stangle);
        Set_float (cCircleAngleRadEnd, endangle);
        Set_integer(cLineColorIdx, color);
        Set_integer(cLineStyle, linestyle);
        Set_integer(clineWidth, thickness);
        --
        objAction(closeObject);
end;
```

Object - Ellipse

The ellipse is defined by a rectangle with two points:

Creating	: *CreateObj(*cElipse*)*
Parameters	*: position*
	: line color
	: line style
	: line width
	: line join
	: fill color
	: pattern

Example:

```
procedure Create_Elipse ( x1,y1,x2,y2:long_float; color, linestyle, thickness : integer) is
begin
           CreateObj(cElipse);
            --
            Set_float2 (cPosXY, x1, y1);
            Set_float2 (cPosXY, x2, y2);
            Set_integer(cLineColorIdx, color);
            Set_integer(cLineStyle, linestyle);
            Set_integer(clineWidth, 1);
            --
            objAction(closeObject);
end;
```

Object - Text

A text starts on the defined position.

Creating	: *CreateObj(*cText*)*
Parameters	: *position*
	: text color
	: text
	: text position
	: alignment in rectangle

Example:

```
procedure  Create_Text (x,y:long_float; color:integer; text_in:string) is
begin
            CreateObj(cText);
            --
            Set_float2 (cPosXY, x, y);
            Set_integer(cTextColorIdx, color);
            Set_string (cTextText, text_in);
            --
            objAction(closeObject);
end;
```

Object - Group of objects

The Group of objects merges subsequently created object to one group (till the group is closed), i.e. the object is created, the next objects are remembered and under the direction of *closeGroup* or *closeAll* is written into the list of created objects.

Creating	: *CreateObj(*cGroup*)*

```
procedure Create_Group is
begin
      CreateObj(cGroup);
end;
```

The lines will be closed as follows:

Closing:

```
procedure CLose_Group is
begin
      objAction(closeGroup);
end;
```

Related pages:

Import of vector formats into pictures of D2000 System
Graphic object creating