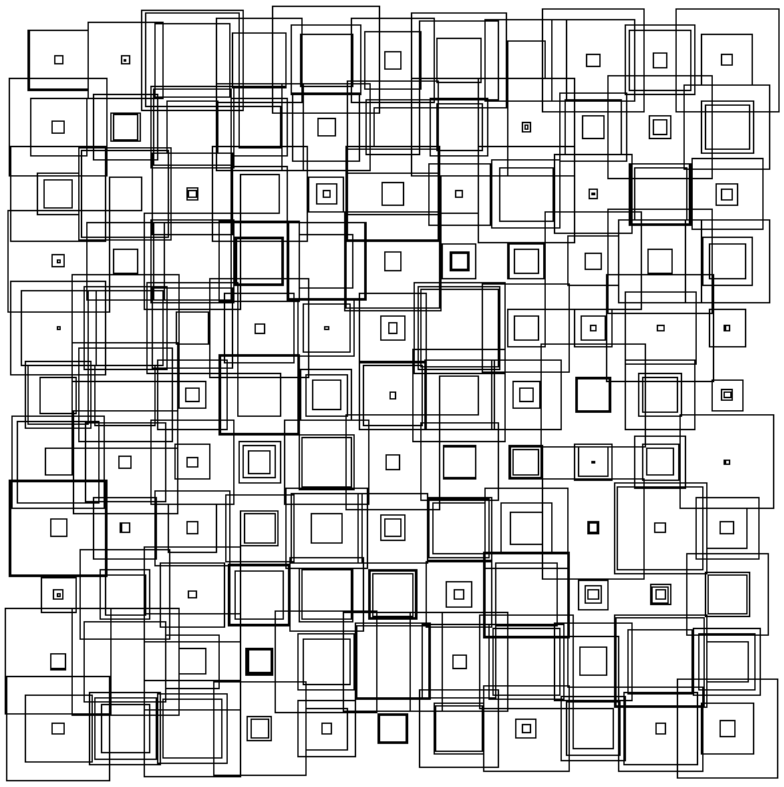
scripts und intelligente objekte programmieren

Die Programmiersprache, in der in VectorWorks Scripts (Makros) programmiert werden, heißt VectorScript. VectorScript ist eine an Pascal angelehnte Sprache, die jedoch nicht über alle Befehle eines echten Pascals (wie z.B. Think Pascal oder Turbo Pascal) verfügt. Eine ausführliche Beschreibung sämtlicher Funktionen und Prozeduren, zur Erstellung von intelligenten Objekten gibt es in der VectorScript-Online-Hilfe. Dort befindet sich auch eine kurz gehaltene Einführung in die Programmiersprache Pascal als VectorScript PDF Handbuch.



```
PROCEDURE randomQuads;

VAR
   x,y:INTEGER;
   sizex:REAL;

BEGIN
   FOR x:=0 TO 10 DO BEGIN
      FOR y:=0 TO 10 DO BEGIN
         sizex:=Random*8;
         Rect((x*10)-sizex,(y*10)-sizex,(x*10)+sizex,(y*10)+sizex);
      END;
   END;
END;

RUN(randomQuads);
```

vectorscript

script erstellen

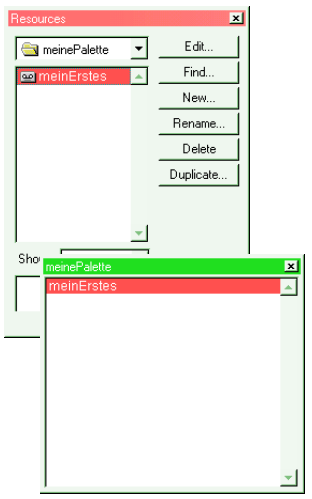
ResourcenPalette öffnen.
Mit ‘new’ eine neue ScriptPalette erzeugen
darin mit ‘new’ ein neues VectorScript erzeugen

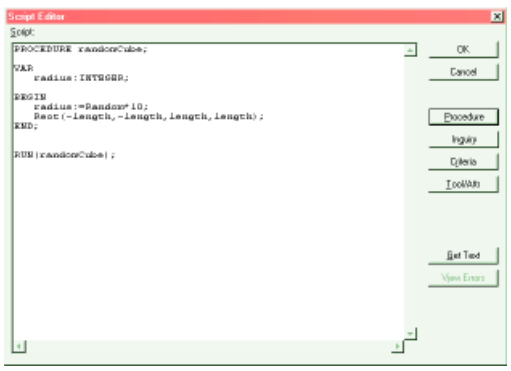
script editieren

Scriptname in der ResourcenPalette doppelklicken
der ScriptEditor öffnet sich

script starten

Scriptname in der SkriptPalette doppelklicken





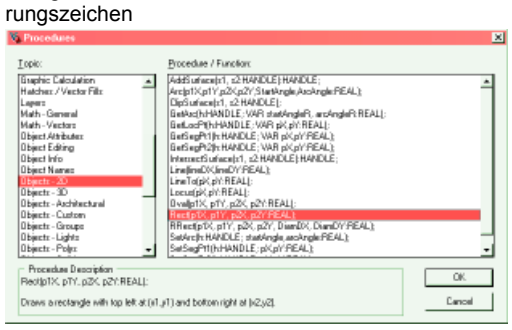
script schreiben

Standard Struktur:

1. Scriptname, eingeleitet durch PROCEDURE
2. Variablendeklaration, eingeleitet durch VAR
   VariablenTypen:
   REAL... Natürliche Zahlen
   INTEGER...Ganzzahlen
   BOOLEAN... true oder false
   STRING... Texte
3. Hauptprogramm, eingebettet in BEGIN und END;
4. Runbefehl:
   RUN(scriptname);

Alle Zeilen, ausser BEGIN, enden mit Semikolon.
Befehle in Grossbuchstaben.
Strings in einfachen Anführungszeichen

```
PROCEDURE randomCube;

VAR
   radius:REAL;

BEGIN
   radius:=Random*10;
   Rect(-radius,-radius,radius,radius);
END;

RUN(randomCube);
```



verwenden vorhandener Prozeduren:

1. im ScriptEditor auf ‘Procedure’ klicken
2. in der ProcedurePalette das Thema, z.b. Objects 2D auswählen
3. die passende Prozedur, z.b. rect auswählen.

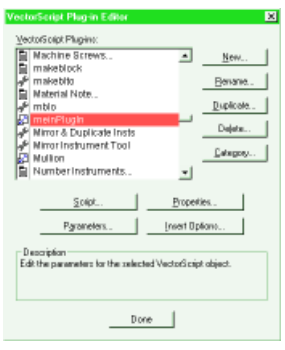
```
Rect(p1X,p1Y,p2X,p2Y:REAL);
```

4. die Werte in den Klammern zeigen, dass die Prozedur 4 Zahlen vom Typ REAL erwartet
5. im Programm müssen diese durch eigene Variablen ersetzt werden

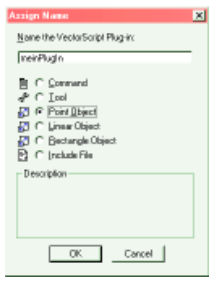
```
Rect(-radius,-radius,radius,radius);
```

plugin erstellen

Im Menu ‘Organize’ mit ‘Create Plug-In...’ das PluginFenster öffnen
mit ‘New’ ein neues Plugin erzeugen

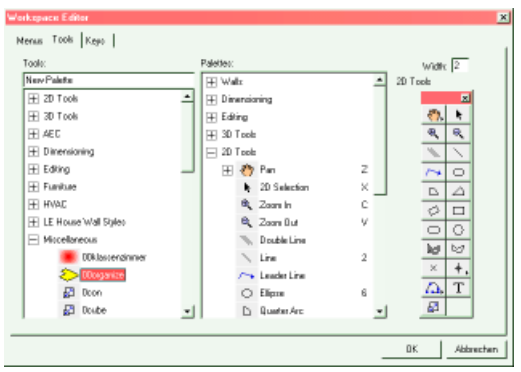


Tool:
Einfaches Kommando, das kein Objekt erzeugt. Vergleichbar mit dem normalen VectorScript.
Point Object:
Objekt mit Position
Linear Object:
Objekt mit Position und Länge.
Rectangular Object:
Objekt mit Position, Länge und Breite.



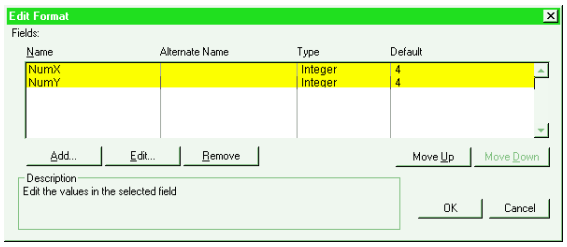
plugin einbinden

1. über das Menü ‘File’ -> ‘Workspaces’ -> ‘Workspace Editor’ einen neuen Workspace erstellen, bzw. einen vorhandenen editieren.
2. Einstellen, ob Palette (‘Tools’) oder PullDownMenu (‘Menu’)
3. Rechts ZielPalette, bzw. ZielMenü auswählen.
4. Die eigenen Plugins liegen links unter ‘Miscellaneous’
5. Per dragAndDrop das Plugin an die gewünschte Stelle ziehen.

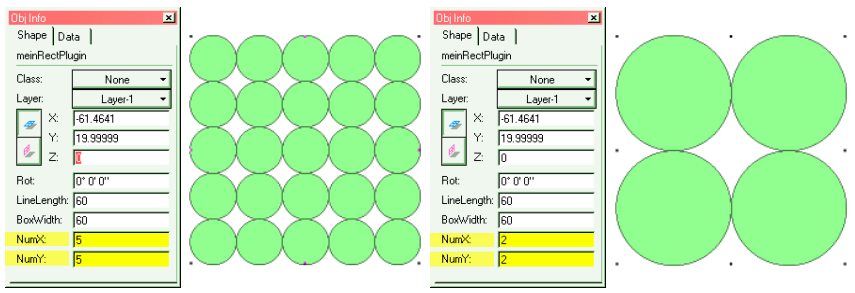


plugin parameter

Parameter sind später im ObjektInfoFenster der VectorWorks-Oberfläche vom Anwender einstellbar. Im PluginScript kann man auf diese Parameter als Variablen zugreifen. Name, Typ und Defaultwert sind anzugeben.




```
PROCEDURE meinPlug;
VAR
   x,y:INTEGER;
   centerX,centerY,dX,dY,rX,rY:REAL;
BEGIN
   FOR x:=1 TO PNUMX DO BEGIN
      FOR y:=1 TO PNUMY DO BEGIN
         dX:=(PLINELENGTH/PNUMX);
         dY:=(PBOXWIDTH/PNUMY);
         rX:=dX/2;
         rY:=dY/2;
         centerX:=(x-0.5)*dX;
         centerY:=(PBOXWIDTH/2.0)-((y-0.5)*dY);
         Oval(centerX-rX,centerY-rY,centerX+rX,centerY+rY);
      END;
   END;
END;
RUN(meinPlug);
```



schleifen und bedingungen

WHILE (Bedingung) DO BEGIN Aktion END;
solange die Bedingung wahr ist, wird die Aktion ausgeführt.

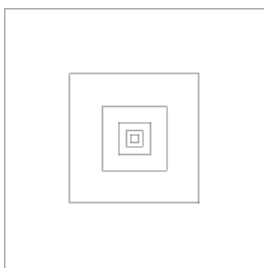
REPEAT Aktion UNTIL (Bedingung);
Aktion wird solange ausgeführt, bis die Bedingung wahr wird.

FOR Index:=Startwert TO Endwert DO BEGIN Aktion END;
die Variable Index wird von Startwert bis Endwert in Einerschritten hochgezählt. Bei jedem Schritt wird die Aktion einmal ausgeführt.

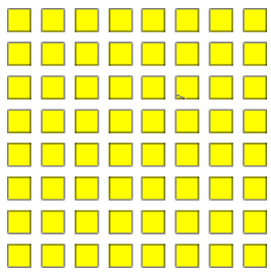
IF (Bedingung) THEN BEGIN Aktion END;
wenn die Bedingung wahr ist , wird die Aktion ausgefürt.

IF (Bedingung) THEN BEGIN Aktion1 END
ELSE BEGIN Aktion2 END;
wenn die Bedingung wahr ist , wird Aktion1 ausgefürt,
sonst wird Aktion2 ausgeführt.

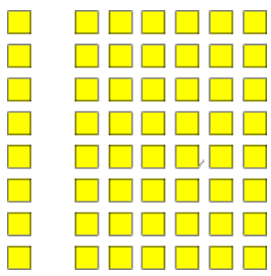
AND OR NOT
dadurch lassen sich Bedingungen verknüpfen.



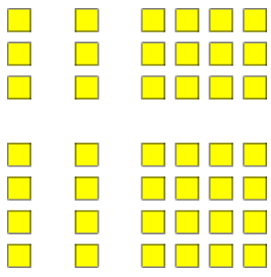
```
PROCEDURE whileSchleife;

VAR
   radius:REAL;

BEGIN
   radius:=100;
   WHILE (radius>2) DO BEGIN
       Rect(-radius,-radius,radius,radius);
      radius:=radius/2;
   END;
END;

RUN(whileSchleife);
```



```
PROCEDURE forSchleife;

VAR
   a,d:REAL;
   x,y:INTEGER;

BEGIN
   d:=10;
   a:=5;
   FOR x:=1 TO 8 DO BEGIN
      FOR y:=1 TO 8 DO BEGIN
         Rect(x*(d+a),y*(d+a),x*(d+a)+d,y*(d+a)+d);
      END;
   END;
END;

RUN(forSchleife);
```



```
BEGIN
   d:=10;
   a:=5;
   FOR x:=1 TO 8 DO BEGIN
      FOR y:=1 TO 8 DO BEGIN
         IF (NOT(x=2)) THEN BEGIN
            Rect(x*(d+a),y*(d+a),x*(d+a)+d,y*(d+a)+d);
         END;
      END;
   END;
END;
```



```
BEGIN
   d:=10;
   a:=5;
   FOR x:=1 TO 8 DO BEGIN
      FOR y:=1 TO 8 DO BEGIN
         IF (NOT((x=2)OR(y=5)OR(x=4))) THEN BEGIN
            Rect(x*(d+a),y*(d+a),x*(d+a)+d,y*(d+a)+d);
         END;
      END;
   END;
END;
```

handle

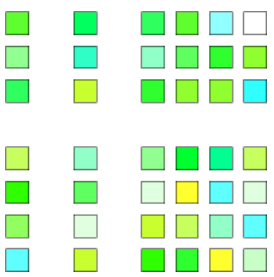
Alle Elemente einer VektorWorks-Zeichnung (Polygone, Kreise, Plugins) werden im VektorSkript über sogenannte HANDLEs angesteuert.

FActLayer
gibt das erste Element des aktiven Layers zurück.
LActLayer
gibt das letzte Element des aktiven Layers zurück.
NextObj(Objekt)
gibt das nächste Element nach 'Objekt' zurück.

```
PROCEDURE assignColor;

VAR
   currentObject:HANDLE;
   rot,gruen,blau:LONGINT;


BEGIN
   Layer('meinLayer');
   currentObject:=FActLayer;

   REPEAT
      rot:=Random*65535;
      gruen:=1.0*65535;
      blau:=Random*65535;
      SetFillBack(currentObject,rot,gruen,blau);
      currentObject:=NextObj(currentObject);
   UNTIL (currentObject=LActLayer);
END;

RUN(assignColor);
```

farben

Jede Farbe hat einen LONGINT Rot-, Grün- und Blau-Wert, zwischen 0 und 65535.
SetFillBack(Element,r,g,b) setzt die ElementFarbe.
FillBack(r,g,b) setzt die aktuelle Zeichenfarbe.

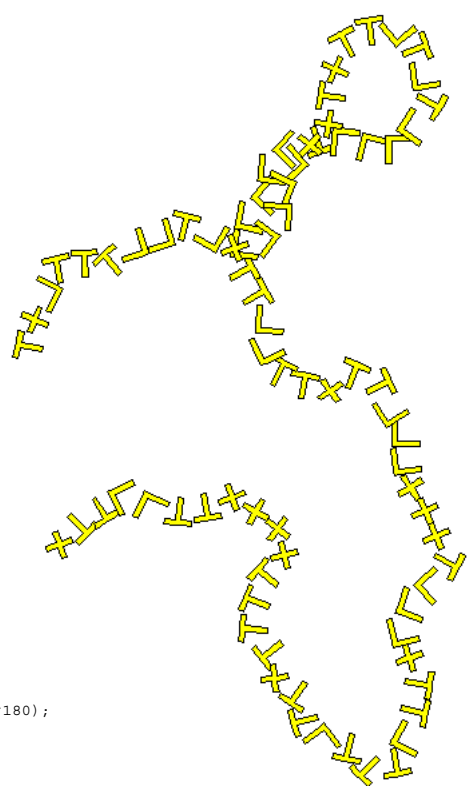


random

Die Prozedur Random erzeugt eine natürliche Zufallszahl zwischen 0.0 und 1.0

symbole

erzeugen:
Symbol('SymbolName',x,y,winkel)



```
PROCEDURE randomSymbols;

VAR
   posX,posY:REAL;
   angle,length:REAL;
   i,test:INTEGER;
   symb:STRING;

BEGIN
   posX:=0;
   posY:=0;
   angle:=0;
   length:=20;
   FOR i:=1 TO 80 DO BEGIN
      angle:=angle+2*(Random-0.5);
      posX:=posX+(length*Cos(angle));
      posY:=posY+(length*Sin(angle));
      test:=(Random*3)+1;
      IF (test=1) THEN BEGIN
         symb:='crossSymb';
      END;
      IF (test=2) THEN BEGIN
         symb:='lSymb';
      END;
      IF (test=3) THEN BEGIN
         symb:='tSymb';
      END;
       Symbol(symb,posX,posY,angle/PI*180);
   END;
END;
```

array

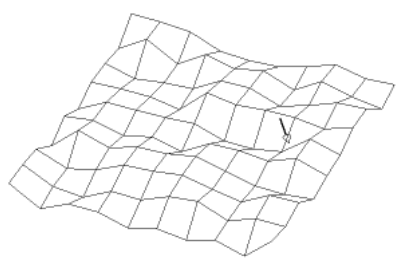
mehrere Variablen gleichen Typs kann man zu Feldern, sogenannten ARRAYS zusammenfassen.
Einzelne Elemente eines Arrays werden über Index angesprochen.

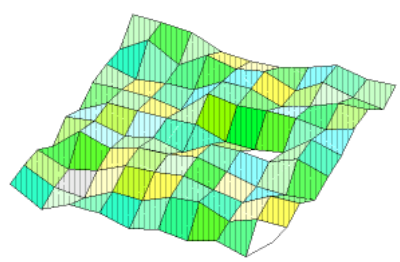
ARRAY [1..5] OF REAL;
erzeugt ein 1dimensionales Feld aus 5 Real-Variablen.

ARRAY [1..10,1..10] OF INTEGER;
erzeugt ein 2dimensionales Feld aus 100 Integer-Variablen.

FeldName [1,3]:=10;
weist dem Element in der 1.Reihe und 3. Spalte des Arrays FeldName den Wert 10 zu.

```
PROCEDURE tuch;

VAR
   x,y:INTEGER;
   hoehe:ARRAY[1..10,1..10] OF REAL;
   px,py,dx,dy:REAL;

BEGIN
   FOR x:=1 TO 10 DO BEGIN
      FOR y:=1 TO 10 DO BEGIN
         hoehe[x,y]:=Random*10;
      END;
   END;

   FOR x:=1 TO 9 DO BEGIN
      FOR y:=1 TO 9 DO BEGIN
         dx:=10;
         dy:=10;
         px:=x*dx;
         py:=y*dy;
         Poly3D(px,py,hoehe[x,y],
                px+dx,py,hoehe[x+1,y],
                px+dx,py+dy,hoehe[x+1,y+1],
                px,py,hoehe[x,y]);
      END;
   END;
END;

RUN(tuch);
```



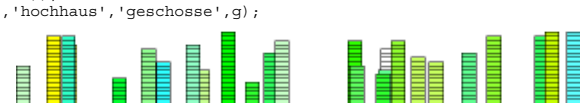


plugins

erzeugen:
CreateCustomObject('PluginName',x,y,winkel)
Parameter setzen:
SetRField(Objekt,'PluginName','ParameterName','Wert')

```
PROCEDURE hochhaus;
VAR
   i:INTEGER;
   h,b:REAL;
BEGIN
   h:=3;
   b:=10;
   FOR i:=0 to (PGESCHOSSE-1) DO BEGIN
      Rect(0,i*h,b,(i+1)*h);
   END;
END;
RUN(hochhaus);
```

```
PROCEDURE skyline;
VAR
   i:INTEGER;
   g:STRING;
   meinHochHaus:HANDLE;
BEGIN
   FOR i:=1 TO 27 DO BEGIN
      meinHochHaus:=CreateCustomObject('hochhaus',Random*400,0,0);
      g:=Num2Str(0,5+(Random*12));
      SetRField(meinHochHaus,'hochhaus','geschosse',g);
   END;
END;
RUN(skyline);
```



Parameter abfragen:
GetRField(Objekt,'PluginName','ParameterName')

```
PROCEDURE closerToSky;
VAR
   gi:INTEGER;
   gs:STRING;
   currentHochHaus:HANDLE;
BEGIN
   Layer('meinLayer');
   currentHochHaus:=FActLayer;
   REPEAT
      gs:=GetRField(currentHochHaus,'hochhaus','geschosse');
      gi:=Str2Num(gs);
      gi:=gi+1;
      gs:=Num2Str(0,gi);
      SetRField(currentHochHaus,'hochhaus','geschosse',gs);
      currentHochHaus:=NextObj(currentHochHaus);
   UNTIL(currentHochHaus=LActLayer);
   SelectAll;
   MoveObjs(0,0,FALSE,FALSE);
END;
RUN(closerToSky);
```