➲ AUFGABE 5: DOSEN PACKEN

Lösungsidee:

Als erstes möchte ich zwei Typenbezeichnungen einführen.
In `Koordinaten` Typ werden `x` und `y` Komponenten gespeichert. Ohne diese Typenbezeichnung könnte die Funktion verschiebe nicht zwei verschiedene Parameter ausgeben.
`Dosenparam` verwende ich als Typ für einen Array der alle Koordinaten und Durchmesser der sich in der Kiste befindenden Dosen beinhaltet.

```
type Koordinaten = record
     x, y: real;
     end;
type Dosenparam = record
     kor: Koordinaten;
     d : byte;
     end;
```

Nun zu meiner `verschiebe` Funktion.
Um die Endposition einer Dose nach dem Verschieben näherungsweise zu erhalten, könnte man die Dosenposition immer um ein Stück in die vorgegebene Richtung verschieben. Dies macht man dann so lange, bis der Abstand zweier Dosenmittelpunkte kleiner als die Summe der Radien ist (Pythagoras). Die Kollision mit der Wand tritt dann ein, wenn der Mittelpunkt der Dose näher als der Radius an der Wand ist. Im Quelltext würde das dann so aussehen:

```
function verschiebe(kor : Koordinaten; alpha:real): Koordinaten;
var r : real;
    i : byte;
    delta : Koordinaten;
    kollision: boolean;
BEGIN

delta.x := cos(alpha/180*PI)/10;                                    //x, y Schritte festlegen
delta.y := -sin(alpha/180*PI)/10;
r:=  dosen[dosenanz].d/2;
kollision := false;

Repeat
  kor.x := kor.x + delta.x;                                        //x, y Schritte hinzufügen
  kor.y := kor.y + delta.y;
  if (kor.x<r) or (kor.y<r) or (kor.x>100-r) or (kor.y>100-r)    //Kollision mit einer Wand
     then kollision := true else
  for i := 1 to dosenanz-1 do                                     //Einzelne Dosen Durchgehen
  if (sqr(r+dosen[i].d/2)>sqr(kor.x-dosen[i].kor.x)+sqr(kor.y-dosen[i].kor.y))
     then kollision := true;                                  //Vergleiche Radius mit Abstand
UNTIL kollision;

  kor.x := kor.x - delta.x;                                //Letzten Schritt rückgängig machen
  kor.y := kor.y - delta.y;
  verschiebe := kor;
END;
```

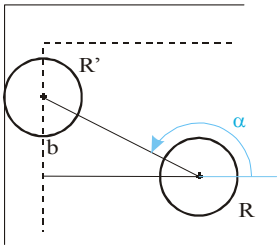
Eine weitere Möglichkeit für eine verschiebe Funktion wäre die exakte Berechnung der Endposition der Dose. Mein Programm geht dabei so vor:

Berechnung des Abstands der Dose zur Wand (Annahme minimale Länge)
Schleife für alle Dosen:
- Berechnung der Länge a (Lot durch den Dosenmittelpunkt[i] auf den Bewegungsvektor)
- Berechnung der Länge b (Dosenmittelpunkt zum Lotfußpunkt)
- Wenn die Länge a < Radius + Radius[i] und b>0 (Bewegungsdekor in Richtung Dose)
  - Berechnung der Länge zwischen momentaner Position und der neuen Berührposition
  - Wenn Länge < minimale Länge diese durch Länge überschreiben

Berechnen der neuen Koordinaten aus gegebenen Winkel und errechneten minimalen Länge

Berechnung der Länge bis zur Kollision mit der Wand:

Bei einer Kollision mit einer Wand ist, wenn es sich um eine obere oder untere Wand handelt, die y-Verschiebung bzw. bei rechter oder linken Wand die x-Verschiebung sehr leicht zu berechnen. Für die rechten und linken Wände kann unter Verwendung des Tangens bzw. für die oberen und unteren mit dem Cotangens die Länge b berechnet werden. Wenn b klein genug ist so dass die neue x bzw. y Position in den Grenzen r und 100-r bleibt, errechnet man mit dem Pythagoras die Länge der Verschiebung.

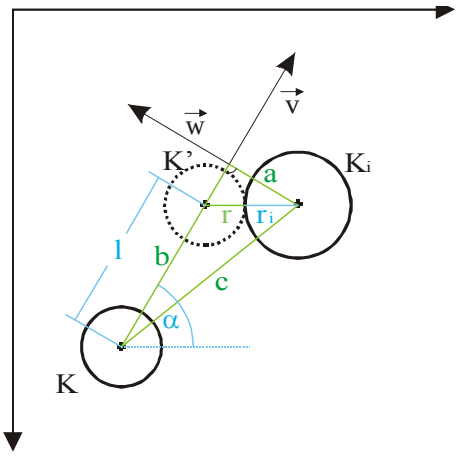


Berechnung der neuen Koordinaten bei Kollision mit einer Dose:

$$\vec{v}^0 = \begin{pmatrix} \cos\alpha \\ -\sin\alpha \end{pmatrix} \quad \vec{w}^0 = \begin{pmatrix} -\sin\alpha \\ -\cos\alpha \end{pmatrix} \quad \text{da } \vec{v} \perp \vec{w}$$

$$b * \vec{v}^0 = \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} + a * \vec{w}^0 \rightarrow \begin{matrix} b * \cos\alpha = \Delta x - a * \sin\alpha \\ b * (-\sin\alpha) = \Delta y - a * \cos\alpha \end{matrix}$$

wobei $\Delta x = x(K_i) - x(K)$ und $\Delta y = y(K) - y(K_i)$



Nach a auflösen:

$$(\Delta x - a * \sin\alpha) * (-\sin\alpha) = (\Delta y - a * \cos\alpha) * \cos\alpha$$

$$a * \sin^2\alpha + a * \cos^2\alpha = \Delta y * \cos\alpha - \Delta x \sin\alpha$$

$$a = \frac{\Delta y * \cos\alpha - \Delta x * \sin\alpha}{\sin^2\alpha + \cos^2\alpha} = \Delta y * \cos\alpha - \Delta x * \sin\alpha$$

Nach b auflösen:

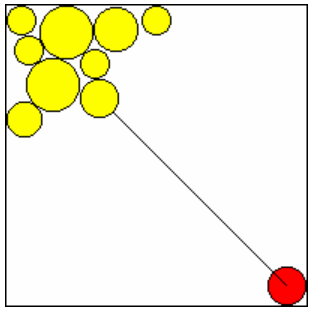
$$b = \frac{\Delta x + \Delta y * \sin\alpha * \cos\alpha - \Delta x * \sin^2\alpha}{\cos\alpha} = \frac{\Delta x * (1 - \sin^2\alpha)}{\cos\alpha} + \Delta y * \sin\alpha = \Delta x * \cos\alpha + \Delta y * \sin\alpha$$

$$l = b - \sqrt{(r + r_i) - a^2} \qquad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \vec{v}^0 * l = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \cos\alpha \\ -\sin\alpha \end{pmatrix} * l$$

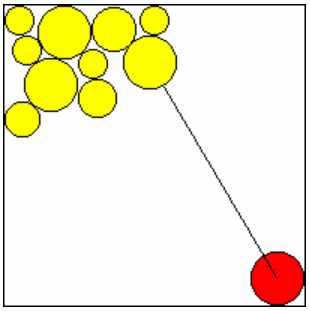
Quelltext der zweiten verschiebe Funktion:

```
function verschiebe(kor : Koordinaten; alpha:real): Koordinaten;
var a, b, l, erg, r : real;
    i : byte;
BEGIN

a := alpha;
while a>180 do a := a-360;              //Zuzückrechnen auf Winkel zwischen -180° und 180°
while a<=-180 do a := a+360;
alpha := alpha/180*PI;
r:=  dosen[dosenanz].d/2;
                                                          //Kollision mit den Wänden
if (a=0) then erg:=100-r-kor.x else if (a=180) then erg := kor.x-r else      //Sonderfälle
  if sin(alpha)>0 then BEGIN                                                  //oben
    b := kor.x+(cotan(alpha)*(kor.y-r));
    if (b>=r) and (b<=100-r) then erg := sqrt(sqr(kor.y-r)+sqr(b-kor.x))
    END
  else BEGIN                                                                  //unten
    b := kor.x-(cotan(alpha)*(100-kor.y-r));
    if (b>=r) and (b<=100-r) then erg := sqrt(sqr(100-kor.y-r)+sqr(b-kor.x));
    END;

if (a=-90) then erg:=100-kor.y-r else if (a=90) then erg := kor.y-r else     //Sonderfälle
  if cos(alpha)<0 then BEGIN                                                  //rechts
    b := kor.y+(tan(alpha)*(kor.x-r));
    if (b>=r) and (b<=100-r) then erg := sqrt(sqr(kor.x-r)+sqr(b-kor.y));
  END
  else BEGIN                                                                  //links
    b := kor.y-(tan(alpha)*(100-kor.x-r));
    if (b>=r) and (b<=100-r) then erg := sqrt(sqr(100-kor.x-r)+sqr(b-kor.y));
    END;
for i := 1 to dosenanz-1 do BEGIN                       //Kollision mit andern Dosen
  a := (kor.y-dosen[i].kor.y)*cos(alpha)-(dosen[i].kor.x-kor.x)*sin(alpha);
  b := (kor.y-dosen[i].kor.y)*sin(alpha)+(dosen[i].kor.x-kor.x)*cos(alpha);
  if ((2*abs(a)<dosen[i].d+dosen[dosenanz].d) and (b>0)) then BEGIN   //Kollision möglich?
    l := b-sqrt( sqr(dosen[i].d+dosen[dosenanz].d)/4-sqr(a));      //Entfernungsberechnung
    if l<erg then erg := l;                         //Vergleich ob nähere Kollision gefunden
  END;
END;
verschiebe.x := kor.x+cos(alpha)*erg;                   //Ausgabe der neuen Position
verschiebe.y := kor.y-sin(alpha)*erg;

END;
```

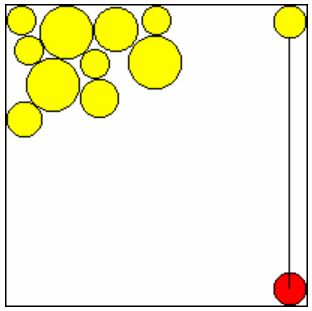
Demonstration der Funktion (zweite Methode):



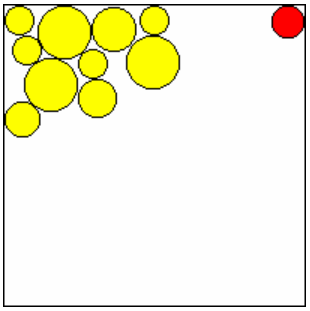
`verschiebe([94,94],135) = [31,31]`



`verschiebe([91,91],120) = [49,19]`



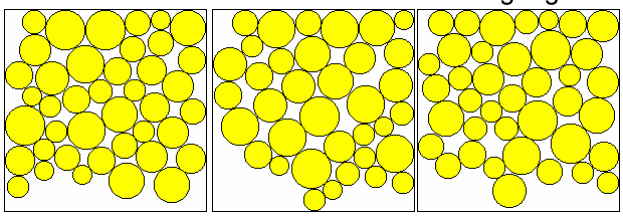
`verschiebe([94,94],90) = [94,6]`



Methoden um eine möglichst gute Packung zu erzielen:

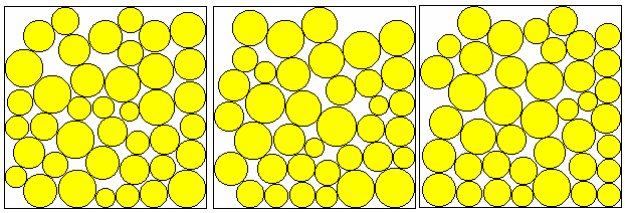
1. Zickzack Methode:

Neue Dosen starten unten in der Mitte und werden erstmals nach rechts außen geschoben. Dann werden sie andauernd von links nach rechts hin und hergeschoben wobei sie bei der Linksbewegung eine leichte Komponente nach oben erhalten und bei der Rechtsbewegung eine schwächere Komponente nach unten. Dies wird so lange durchgeführt, bis die Positionsveränderung bei einer rechts-links Bewegung unter eine gewisse Schwelle sinkt. Als verschiebe Winkel habe ich z.B. 174° und -1° verwendet und dabei nebenstehende Ergebnisse erzielt.



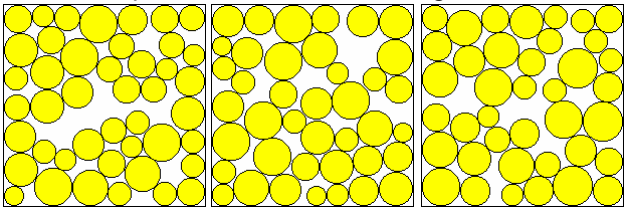
2. längster Weg Methode:

Neue Dosen werden links oben postiert. Dann wird getestet mit welchem Winkel zwischen –90° und 0° die Dose am weitesten verschoben werden kann. Dieser Winkel wird dann zum verschieben verwendet.



3. Chaos Methode:

Neue Dosen werden per Zufallsgenerator in der Kiste postiert. Wenn sich dort schon eine Dose befindet wird eine gewisse Anzahl lang eine weitere Position per Zufall bestimmt und wieder getestet ob schon eine Dose vorhanden ist. Wenn kein freier Platz mehr auffindbar ist bricht das Programm ab. Wenn ein Platz vorhanden ist dann wird per Zufall ein Winkel ausgewählt und die Dose in diese Richtung geschoben. Darauf startet eine Schleife in der die Dosen einen um maximal 45° anderen Winkel weitergeschoben werden. Die Schleife bricht ab, sobald das Spiel der Dose rechts und links kleiner als eine Schwelle wird. Dieses Spiel wird auch durch die verschiebe Funktion ermittelt.



Programmablaufprotokoll:

Die Taste „Dosen hinzufügen“ fügt eine Dose zur Kiste hinzu. Die dabei erstellte Dose nimmt den im Feld eingegebenen Radius an. In der Droppdownbox kann man sich verschiedene Startpositionen aussuchen. Mit einer Dose können dann so oft verschiebe Funktionen durchgeführt werden, bis eine neue Dose zur Kiste hinzugefügt wird. Die verschiebe Funktion kann auf zwei verschiedene Wege angesprochen werden. Die erste Variante funktioniert über die Maus. Wenn man diese über die Kiste bewegt sieht man rot die momentane Position der Dose und eine weitere gelbe Dose mit einer Linie verbunden. Bei einem Doppelklick wird die Verschiebe Funktion durchgeführt und die Dose bekommt eine neue Position. Danach stehen die Bereich Aufgabe 1 die verwendeten Start und entstandenen Endkoordinaten. Der Winkel entspricht aber nicht dem verwendeten Winkel sondern dem momentanen. Man kann die verschiebe Funktion aber auch durch Eingabe der Parameter im Aufgabe 1 Bereich durchführen. Entsprechen dabei die eingegebenen Koordinaten nicht denen auf dem Bildschirm, so wird nachgefragt, ob sie die eingegebenen oder die momentanen (die in der Grafik) Koordinaten für den Ausgangspunkt der Verschiebung verwenden wollten. Nach dem durchführen der Prozedur wird auch hier das Ergebnis in der Kiste und als Zahlen ausgegeben. Im Bereich Aufgabe2 befinden sich die 3 oben beschriebenen Vorgehensweisen um eine möglichst gute Packung zu erzeugen. Hierbei wird die Kiste nicht automatisch geleert, sondern zur bisherigen Kiste einfach hinzugefügt.

Kompletter Quelltext (AUFGABE5.PAS):

```
unit aufgabe5;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls, Spin, Math;

type
  TForm1 = class(TForm)
    kiste: TImage;

    GroupBox1: TGroupBox;
    add: TButton;
    remove: TButton;
    erstellmodus: TComboBox;
    label_durchm: TLabel;
    durchm: TEdit;

    GroupBox2: TGroupBox;
    label_function1: TLabel;
    inx: TEdit;
    iny: TEdit;
    alpha: TEdit;
    label_function2: TLabel;
    ergx: TEdit;
    ergy: TEdit;
    senden: TButton;

    GroupBox3: TGroupBox;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;

    procedure FormCreate(Sender: TObject);
    procedure addClick(Sender: TObject);
    procedure removeClick(Sender: TObject);
    procedure kisteMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
    procedure bewegen(Sender: TObject);
    procedure methode1(Sender: TObject);
    procedure methode2(Sender: TObject);
    procedure methode3(Sender: TObject);
  end;


type Koordinaten = record
     x, y: real;
     end;
type Dosenparam = record
     kor: Koordinaten;
     d : byte;
```

```
    end;

const Positionen : array [1..5, 1..2] of real = ((1,0),(0,0),(1,1),(0,1), (0.5,1));
                                                  //Verschiedenen Startpostionen für Dosen
var
  Form1: TForm1;
  dosen: array[1..100] of Dosenparam;              //Alle Parameter der Dosen in der Kiste
  dosenanz: integer = 0;                                                  //Anzahl der Dosen
implementation

{$R *.DFM}


function verschiebe(kor : Koordinaten; alpha:real): Koordinaten;       //Exakte Berechnung
var a, b, l, erg, r : real;
    i : byte;
BEGIN
  a := alpha;
  while a>180 do a := a-360;            //Zuzückrechnen auf Winkel zwischen -180° und 180°
  while a<=-180 do a := a+360;
  alpha := alpha/180*PI;
  r:=  dosen[dosenanz].d/2;
                                                                  //Kollision mit den Wänden
  if (a=0) then erg:=100-r-kor.x else if (a=180) then erg := kor.x-r else    //Sonderfälle
    if sin(alpha)>0 then BEGIN                                                      //oben
      b := kor.x+(cotan(alpha)*(kor.y-r));
      if (b>=r) and (b<=100-r) then erg := sqrt(sqr(kor.y-r)+sqr(b-kor.x))
      END
    else BEGIN                                                                    //unten
      b := kor.x-(cotan(alpha)*(100-kor.y-r));
      if (b>=r) and (b<=100-r) then erg := sqrt(sqr(100-kor.y-r)+sqr(b-kor.x));
      END;

  if (a=-90) then erg:=100-kor.y-r else if (a=90) then erg := kor.y-r else   //Sonderfälle
    if cos(alpha)<0 then BEGIN                                                    //rechts
      b := kor.y+(tan(alpha)*(kor.x-r));
      if (b>=r) and (b<=100-r) then erg := sqrt(sqr(kor.x-r)+sqr(b-kor.y));
    END
    else BEGIN                                                                    //links
      b := kor.y-(tan(alpha)*(100-kor.x-r));
      if (b>=r) and (b<=100-r) then erg := sqrt(sqr(100-kor.x-r)+sqr(b-kor.y));
      END;
  for i := 1 to dosenanz-1 do BEGIN                          //Kollision mit andern Dosen
    a := (kor.y-dosen[i].kor.y)*cos(alpha)-(dosen[i].kor.x-kor.x)*sin(alpha);
    b := (kor.y-dosen[i].kor.y)*sin(alpha)+(dosen[i].kor.x-kor.x)*cos(alpha);
    if ((2*abs(a)<dosen[i].d+dosen[dosenanz].d) and (b>0)) then BEGIN //Kollision möglich?
      l := b-sqrt( sqr(dosen[i].d+dosen[dosenanz].d)/4-sqr(a));    //Entfernungsberechnung
      if l<erg then erg := l;                    //Vergleich ob nähere Kollision gefunden
    END;
  END;
  verschiebe.x := kor.x+cos(alpha)*erg;                      //Ausgabe der neuen Position
  verschiebe.y := kor.y-sin(alpha)*erg;
END;


procedure zeichnedose(inp : Dosenparam);        //Zeichnet einen gelben Kreis in die Kiste
begin
  form1.kiste.canvas.Ellipse(round(1+2*inp.kor.x-inp.d),round(1+2*inp.kor.y-inp.d),
                             round(1+2*inp.kor.x+inp.d),round(1+2*inp.kor.y+inp.d));
  form1.kiste.Canvas.Brush.Color := clyellow;
  form1.kiste.canvas.FloodFill(round(2*inp.kor.x), round(2*inp.kor.y), clblack, fsborder);
end;


procedure zeichnekiste();           //Löscht Kiste und zeichnet alle vorhandenen Dosen neu
var i : byte;
begin
  form1.kiste.Canvas.Brush.Color := clwhite;
  form1.kiste.Canvas.Rectangle(0,0,202,202);
  for  i := 1 to dosenanz do zeichnedose(dosen[i]);
  form1.kiste.Canvas.Brush.Color := clred;
  form1.kiste.canvas.FloodFill(round(2*dosen[dosenanz].kor.x),
                               round(2*dosen[dosenanz].kor.y), clblack, fsborder);
  form1.kiste.Refresh;
end;


function abs2(inp:koordinaten):real;       //Wendet Pythagoras auf übergebene Variablen an
```

```
BEGIN
  abs2 := sqrt(inp.x*inp.x + inp.y*inp.y);
END;


function setzedose(art:integer) :boolean;                  //Fügt neue Dose zur Kiste hinzu
var kollision : boolean;
    i : byte;
    versuche : byte;
begin
with form1 do BEGIN
  inc(dosenanz);                                                  //Eine Dose mehr vorhanden
  versuche := 0;
  dosen[dosenanz].d := StrToInt(durchm.text); //Übernimmt den Durchmesser aus dem Textfeld
  repeat
    kollision := false;
    if (art = 0) then begin                                     //x, y durch Zufall festlegen
     dosen[dosenanz].kor.x:=random*(100-dosen[dosenanz].d)+dosen[dosenanz].d/2;
     dosen[dosenanz].kor.y:=random*(100-dosen[dosenanz].d)+dosen[dosenanz].d/2;
    end else BEGIN                                               //x, y nach Wunsch festlegen
     dosen[dosenanz].kor.x:=Positionen[art,1]*(100-dosen[dosenanz].d)+dosen[dosenanz].d/2;
     dosen[dosenanz].kor.y:=Positionen[art,2]*(100-dosen[dosenanz].d)+dosen[dosenanz].d/2;
    END;
    for i := 1 to (dosenanz-1) do        //Test: Überschneidung neuer Dose mit vorhandenen
      if ((dosen[i].kor.x-dosen[dosenanz].kor.x)*(dosen[i].kor.x-dosen[dosenanz].kor.x)+
          (dosen[i].kor.y-dosen[dosenanz].kor.y)*(dosen[i].kor.y-dosen[dosenanz].kor.y)<
          ((dosen[i].d+dosen[dosenanz].d)/2)*((dosen[i].d+dosen[dosenanz].d)/2)) then
        kollision := true;
    if kollision then inc(versuche);    //wenn Zufallsbestimmung dann 200 weitere Versuche
  until (kollision = false or (versuche>200)) or (art<>0);        //sonst Schleife beenden
  if (kollision) then BEGIN                              //wenn Kollision immer noch zutrifft
    dec(dosenanz);                                                  //Dosenanzahl verringern
    setzedose:= false;                                                      //Fehlerrückgabe
    if (art=0) then                    //Fehlerausgabe bei Verwendung des Zufallsgenerators
      application.MessageBox('Es ist schwer eine freie Stelle für die Dose zu finden!',
                             'Kein Platz?', MB_OKCANCEL + MB_DEFBUTTON1)
    else                                 //Fehlermeldung bei fester Platzdefinition
      application.MessageBox('An dieser Position befindet sich bereitz eine Dose',
                             'Kein Platz?', MB_OKCANCEL + MB_DEFBUTTON1);
  END else setzedose := true;                                      //kein Fehler aufgetreten
  inx.text :=  inttostr(round(dosen[dosenanz].kor.x));    //x, y Koordinaten in Textfelder
  iny.text :=  inttostr(round(dosen[dosenanz].kor.y));
  zeichnekiste();                                                 //Kiste neu zeichnen
  END;
end;



procedure TForm1.FormCreate(Sender: TObject);                          //Initialisierung
begin
  randomize;
  kiste.Canvas.Rectangle(0,0,202,202);
  erstellmodus.ItemIndex:=0;
end;


procedure TForm1.addClick(Sender: TObject);       //beim Klick auf 'Dose hinzufügen' Knopf
begin
  setzedose(erstellmodus.ItemIndex);
end;


procedure TForm1.removeClick(Sender: TObject);     //beim Klick auf 'alle Dosen entfernen'
begin
  dosenanz := 0;
  zeichnekiste;
end;



procedure TForm1.kisteMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
var winkel: real;                                        //Maus bewegt sich über der Kiste
    dose : Dosenparam;

begin
if (dosenanz>0) then begin
```

```
  winkel := (180/PI)*arctan2(dosen[dosenanz].kor.y-y/2,x/2-dosen[dosenanz].kor.x);
  alpha.Text := inttostr(round(winkel));           //Berechnung Winkel Dose-Maus + Ausgabe

  zeichnekiste();                                                    //Kiste neu zeichen
  kiste.Canvas.MoveTo (round(dosen[dosenanz].kor.x*2), round(dosen[dosenanz].kor.y*2));
  dose.kor := verschiebe(dosen[dosenanz].kor, winkel);//Dosenkoordinaten nach Verschiebung
  dose.d := dosen[dosenanz].d;
  kiste.Canvas.LineTo(round(dose.kor.x*2), round(dose.kor.y*2));
                                                  //Linie zwischen alter und neuen Dose
  zeichnedose(dose);                                 //Dose mit neuer Position zeichnen
  end;
end;


procedure TForm1.bewegen(Sender: TObject);       //Dosenverschiebung über Button oder Bild
var winkel: real;
    pos : Koordinaten;
begin
  winkel := strtoint(form1.alpha.Text);
  pos :=  dosen[dosenanz].kor;                        //Position der neusten Dose auslesen
  if Sender.ClassName = 'TImage' then begin            //Wenn Befehl über Bild aufgerufen
    inx.text := inttostr(round(pos.x));   //Koordinaten für Verschiebung direkt übernehmen
    iny.text := inttostr(round(pos.y));
    end                                                 //bewegen über Button ausgerufen
else if (round(pos.x) <>  strtoint(inx.text)) or (round(pos.y) <>  strtoint(iny.text)) then
   if application.MessageBox('Soll die neue Position der Dose (in der Grafik sichtbar)
verwendet werden?', 'Koordinaten?', 4) = 7 then begin
     pos.x :=  strtoint(inx.text);  //Frage ob die eingegeben Koordinaten verwendet werden
     pos.y :=  strtoint(iny.text);    //sollen, oder die, welche die Dose momentan besitzt
     end;
  dosen[dosenanz].kor := verschiebe(pos, winkel);                     //Dose Verschieben
  ergx.text := inttostr(round(dosen[dosenanz].kor.x));               //Ergebnis ausgeben
  ergy.text := inttostr(round(dosen[dosenanz].kor.y));
  zeichnekiste();                                                    //Kiste neuzeichnen
end;



procedure TForm1.methode1(Sender: TObject);                          //Zick-Zack Methode
var pos, posa: Koordinaten;
begin
while setzedose(5) do begin           //Solange Dose untenmitte setzen bis kein Platz mehr
  pos := dosen[dosenanz].kor;                                        //Position einlesen
  pos := verschiebe(pos, 0);                                      //Nach rechts verschieben
  repeat
    posa := pos;                                          //Position für vergleich speichern
    pos := verschiebe(pos, 175);
    pos := verschiebe(pos, -1);
  until (abs(posa.x-pos.x) <0.01) and (abs(posa.y-pos.y) <0.01);   //Repeat bis delta<0.01
  dosen[dosenanz].kor := pos;                             //neue Position der Dose speichern
  zeichnekiste();                                                  //Kiste neu zeichnen
  durchm.text := inttostr(round((random*10))+10);      //per Random neuen Radius festlegen
  end;
end;


procedure TForm1.methode2(Sender: TObject);                        //Längster Weg Methode
var  alpha, l, lmax: real;
pos : koordinaten;
w : integer;
begin
while setzedose(2) do BEGIN          //Solange Dosen linksobern setzen bis kein Platz mehr
  pos := dosen[dosenanz].kor;                                        //Position einlesen
  lmax := 0;                                              //Maximale Verschiebelänge = 0
  for w := -360 to 0 do begin                            //Alle Winkel von -90° bis 0° (*4)
    l := abs2(verschiebe(pos, w/4));                     //Länge der Verschiebung berechnen
    if l > lmax then begin             //Wenn längere länge erzeihlt wurde diese speichern
      lmax := l;
      alpha := w/4;
      end;
    end;
  dosen[dosenanz].kor := verschiebe(pos, alpha);          //Verschiebung mit besten Winkel
  zeichnekiste;                                                    //Kiste neu zeichen
  durchm.text := inttostr(round((random*10)+10));               //Random für neuen Radius
  end;
end;
```

```
procedure TForm1.methode3(Sender: TObject);                              //Chaos Methode
var  wink, spiel : real;
pos, spos1, spos2, delta : koordinaten;
begin
  while setzedose(0) do BEGIN                                  //Dose mit Zufallsgenertator setzen
    pos := dosen[dosenanz].kor;                                              //Position einlesen
    wink := random(360);                                    //Zufällig Verschiebungswinkel wählen
    repeat
      wink := wink+random(90)-45;                      //Dose weiterscheiben (aber nicht zurück)
      pos := verschiebe(pos, wink);
      spos1 := pos;                                 //Position für Berechung des Spiels speichern
      pos := verschiebe(pos, wink+90);                                 //Dose nach links schieben
      spos2 := pos;                                              //Postion für Spiel speichern
      pos := verschiebe(pos, wink-90);                                 //Dose nach rechts schieben
      delta.x := spos1.x-spos2.x;                                          //Berechnung des Spiels
      delta.y := spos1.y-spos2.y;
      spiel := abs2(delta);
      delta.x := spos2.x-pos.x;
      delta.y := spos2.y-pos.y;
      spiel := spiel+abs2(delta);
      until spiel<0.1;                      //Wiederhole bis die Dose weniger als 0.1mm Spiel hat
    dosen[dosenanz].kor := pos;                                      //neue Dosenposition übernehmen
    zeichnekiste;                                                            //Kiste neuzeichnen
    durchm.text := inttostr(round((random*10)+10));                     //Neuen Winkel festlengen
  end;
end;

end.
```