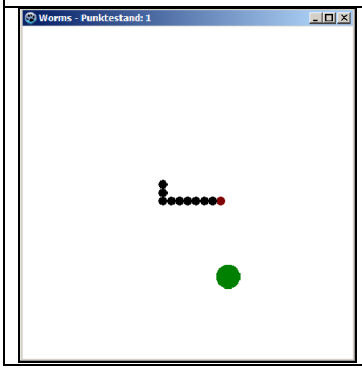
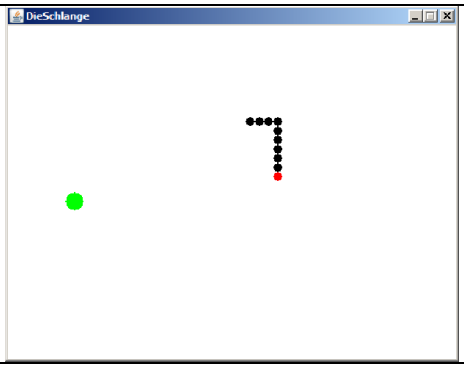
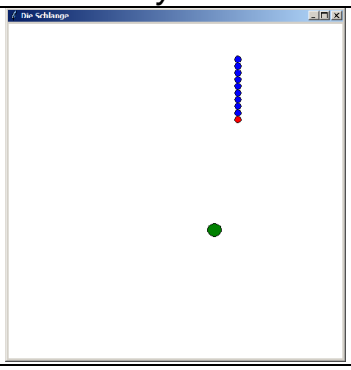
Zum Spiel
Das Spiel "Die Schlange" ist ähnlich dem Spiel „Snake“, das es in allen Varianten seit ca. 1980 gibt und wegen der simplen Grafik in seiner Umsetzung auf Handys populär wurde.

„Snake“ auf einem Handy

Lazarus	*Java*	*Python*
		

Programmierkenntnisse

- Arrays
- ereignisgesteuertes Programmieren mit Tasten

Algorithmischer Kern

- Zustandsmodellierung
- Lineare Suche im Array
- Daten im Array verschieben

Schritt 1: Schlange zeichnen

Die 10 Glieder der Schlange werden über Arrays verwaltet. Beim Start werden die Arrayeinträge mit Startwerten belegt.

Nun kann die Schlange in einer Prozedur "move" schonmal gezeichnet werden. "move" wird vom einem Timer aufgerufen.

Lazarus

```
Timer1.enabled := true;
Timer1.enabled := false;
```

Python

```
timerOn = True


def Timer():
    [. . .]
    Fenster.after(500, Timer)

#GUI
Fenster = tkinter.Tk()
   if (timerOn == True):
       Fenster.after(50, Timer)
```

Java

```
import javax.swing.Timer;

Timer t;

ActionListener TimerProzedur = new ActionListener() {
    public void actionPerformed (ActionEvent evt) {
    [. . .]
    }
}


t = new Timer(500, TimerProzedur);
t.start();
//t.stop();
```

Dabei wird hier schon das Zeichnen des Rumpfes in Schwarz vom Zeichnen des Kopfes (in einer anderen Farbe) getrennt.
Erkennbar sind dem Kopf die Arrayfelder WormX[1] und WormY[1] zugeordnet (bzw. WormX[0]), dem Rumpf die übrigen Felder.
Die einzelnen Glieder werden als Punkte dargestellt mit einem Durchmesser von 10 Pixel. Damit bewegt sich die Schlange in 10-Pixel-Schritten.
Das Formular wird weiß gemacht.

Die Nummern in der Prozedur "move" entsprechen der Reihenfolge der Programmblöcke.

Lazarus

```
var
  wormx : array [1..10] of integer;
  wormy : array [1..10] of integer;

procedure TForm1.FormCreate(Sender: TObject);
var i : integer;
begin
Form1.color := clwhite;
   wormx[1]:= 150;   wormy[1]:= 100;
   wormx[2] := 140;  wormy[2] := 100;
   wormx[3] := 130;  wormy[3] := 100;
   wormx[4]:= 120;   wormy[4]:= 100;
   wormx[5] := 110;  wormy[5] := 100;
   wormx[6] := 100;  wormy[6] := 100;
   wormx[7] := 90;  wormy[7] := 100;
   wormx[8]:= 80;   wormy[8]:= 100;
   wormx[9] := 70;  wormy[9] := 100;
   wormx[10] := 60;  wormy[10] := 100;
end;



procedure move;
var i, px, py : integer;
begin

 // 4. Kopf zeichnen
 Form1.Canvas.Pen.Color := clmaroon;
 Form1.canvas.moveto(wormx[1],wormy[1]);
 Form1.Canvas.LineTo(wormx[1],wormy[1]);

 //5.  Rumpf zeichnen
 for i := 2 to 10 do
   begin
   Form1.Canvas.Pen.Color := clblack;
   Form1.canvas.moveto(wormx[i],wormy[i]);
   Form1.Canvas.LineTo(wormx[i],wormy[i]);
   end;
end;
```

Java

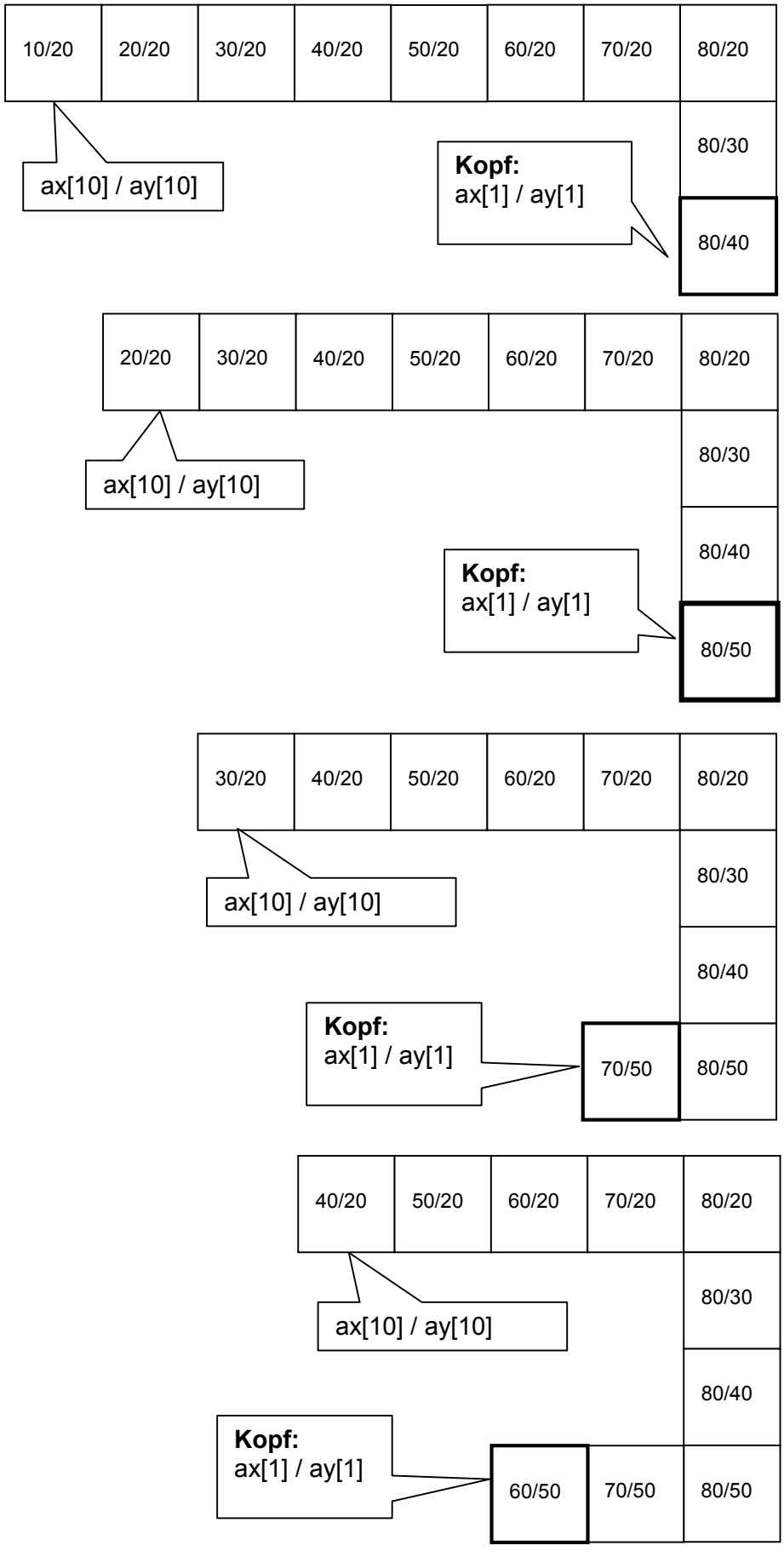
```
int [ ] wormx = new int [10];
int [ ] wormy = new int [10];

public Snake(String title) {
// Startbelegung der Schlange
    wormx[0] = 160;  wormy[0] = 100;
    wormx[1] = 150;   wormy[1] = 100;
    wormx[2] = 140;  wormy[2] = 100;
    wormx[3] = 130;  wormy[3] = 100;
    wormx[4] = 120;   wormy[4] = 100;
    wormx[5] = 110;  wormy[5]  = 100;
    wormx[6] = 100;  wormy[6] = 100;
    wormx[7] = 90;  wormy[7] = 100;
    wormx[8] = 80;   wormy[8] = 100;
    wormx[9] = 70;  wormy[9] = 100;
}

void move () {
    Graphics g = canvas1.getGraphics();
    Graphics2D c = (Graphics2D) g;

// 4. Kopf zeichnen
    c.setColor(Color.RED);
    c.fillOval(wormx[0],wormy[0],10,10);


    // 5. Rumpf zeichnen
    for (int i = 1;i<10;i++) {
      c.setColor(Color.BLACK);
      c.fillOval(wormx[i],wormy[i],10,10);
    }
} // Ende move
```

Python

```
Snake = [0 for i in range(11)]    #Zu zeichnende Schlangenglieder
Snakex = [10 for i in range(11)]  #Koordianten der Schlange
Snakey = [0 for i in range(11)]   #Koordinaten der Schlange

#GUI
Fenster = tkinter.Tk()
Fenster.title("Snake")

Fenster.after(500,timer) #Der Timer

Z = tkinter.Canvas(Fenster, height = 500, width = 500, bg="white")
Z.grid(row = 1, column = 1)

#Snake - Kopf
Snake[0] = Z.create_oval(100,50,110,60, fill = "red")
#Snake - Schwanz
Snake[1] = Z.create_oval(90,50,100,60, fill = "blue")
Snake[2] = Z.create_oval(80,50,90,60, fill = "blue")
Snake[3] = Z.create_oval(70,50,80,60, fill = "blue")
Snake[4] = Z.create_oval(60,50,70,60, fill = "blue")
Snake[5] = Z.create_oval(50,50,60,60, fill = "blue")
Snake[6] = Z.create_oval(40,50,50,60, fill = "blue")
Snake[7] = Z.create_oval(30,50,40,60, fill = "blue")
Snake[8] = Z.create_oval(20,50,30,60, fill = "blue")
Snake[9] = Z.create_oval(10,50,20,60, fill = "blue")
#GUI
Fenster.mainloop()
```

Schritt 2: Schlange bewegen

Idee der Schlangenbewegung über ein Array:



a) Bewegung des Rumpfes

Die neuen Array-Koordinaten des ***Rumpfes*** können über eine Schleife aus den "Vorgängern" berechnet werden.

Lazarus

```
// 2. Rumpf bewegen
for i := 10 downto 2 do
   begin
   wormx[i] := wormx[i-1];
   wormy[i] := wormy[i-1];
   end;
```

Java

```
// 2. Rumpf bewegen
    for (int i = 9; i>0; i--) {
      wormx[i] = wormx[i-1];
      wormy[i] = wormy[i-1];
    }
```

Python

```
def timer():
  Bewegung()
  if (timerOn == True):
    Fenster.after(200, timer)

def Bewegung():
  for i in range(10,0,-1): #Rückwärtszählschleife
    Z.move(Snake[i],Snakex[i],Snakey[i])
    Snakex[i] = Snakex[i-1]
    Snakey[i] = Snakey[i-1]
```

b) Richtungssteuerung → Bewegung des Kopfes

Die Schlange soll über die ***Richtungstasten (Pfeiltasten)*** bewegt werden. Es gibt vier Bewegungszustände **z**:

1 (oben)
4 (links) 2 (rechts)
3 (unten)

Lazarus

Die Tasteneingaben werden über sog. "Virtuelle Tastencodes" in der Formularmethode (Formularereignis) "FormKeyDown" als Mengeneintrag abgefangen. Dazu wird bei *Units*: `units LCLType;` ergänzt. "FormKeyDown" soll hier vorgegeben werden:

```
var
  Richtung : integer;

procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  case Key of
     VK_UP : Richtung := 1;
     VK_RIGHT : Richtung := 2;
     VK_DOWN : Richtung := 3;
     VK_LEFT : Richtung := 4;
     VK_SPACE : Restart;
  end;
```

```
end;


procedure move;
var i, px, py : integer;
begin
[. . .]
// 3. Kopfbewegen
case Richtung of
   1 : wormy[1] := wormy[1]-10;
   2 : wormx[1] := wormx[1]+10;
   3 : wormy[1] := wormy[1]+10;
   4 : wormx[1] := wormx[1]-10;
   end;

end;
```

Java

Um die Tastatureingaben auf der Canvas abzufangen, sollte zunächst der Fokus auf das Canvas gesetzt werden.

```
canvas1.requestFocus();
```

In der Klasse "KeyEvent" sind die sog. virtuellen Tastencodes enthalten, die in einem *KeyListener* bei Tastendruck über "getKeyCode()" abgefangen und abgefragt werden können.

```
int Richtung = 2;

canvas1.addKeyListener(new KeyAdapter() {
      public void keyPressed(KeyEvent evt) {
        canvas1_KeyPressed(evt);

        //     1
        //  4     2
        //     3

        if (evt.getKeyCode() == KeyEvent.VK_RIGHT ) {
          Richtung = 2;
        } else if (evt.getKeyCode() == KeyEvent.VK_LEFT ) {
          Richtung = 4;
        } else if (evt.getKeyCode() == KeyEvent.VK_UP ) {
          Richtung = 1;
        } else if (evt.getKeyCode() == KeyEvent.VK_DOWN ) {
          Richtung = 3;
        }

        repaint();
      }
    });

void move () {
[. . .]
// 3. Kopfbewegen
    switch (Richtung) {
      case 1:
      wormy[0] = wormy[0]-10;
      break;
      case 2:
      wormx[0] = wormx[0]+10;
      break;
      case 3 :
      wormy[0] = wormy[0]+10;
      break;
      case 4 :
      wormx[0] = wormx[0]-10;
      break;
      }
} // Ende move
```

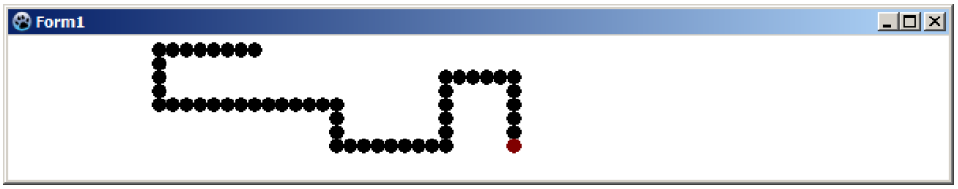
Python

```
Richtung = 2

def rechts (event):
  global Richtung
  Richtung = 2


def links (event):
    global Richtung
    Richtung = 4

def hoch(event):
    global Richtung
    Richtung = 1

def runter(event):
    global Richtung
    Richtung = 3

def Bewegung():
  global Bewegungx
  global Bewegungy
  if (Richtung == 1):
      Bewegungx = 0
      Bewegungy = -10
  if (Richtung == 2):
      Bewegungx = 10
      Bewegungy = 0
  if (Richtung == 3):
      Bewegungx = 0
      Bewegungy = 10
  if (Richtung == 4):
      Bewegungx = -10
      Bewegungy = 0
[. . .]

Fenster.bind("<KeyPress-Left>", links)
Fenster.bind("<KeyPress-Right>", rechts)
Fenster.bind("<KeyPress-Down>", runter)
Fenster.bind("<KeyPress-Up>", hoch)
```

c) Schlange bei Bewegung zeichnen

Der Timer bzw. die Prozedur "move" sollte nun aus den bislang erarbeiteten Codeabschnitten bestehen.

Die Schlange müsste sich nun bewegen und steuern lassen. Allerdings wird die ursprünglich gezeichnete Schlange bei der Bewegung noch nicht gelöscht und "zieht" einen langen "Schwanz" hinter sich her.



Damit die Schlange sich tatsächlich bewegt, muss vor dem Bewegen die Schlange noch gelöscht werden, hier erreicht man das durch weißes Überzeichnen der Schlange.

Lazarus

```
// 1. Schlange löschen

Form1.Canvas.Pen.Width := 11;
for i := 1 to 10 do
begin
  Form1.Canvas.Pen.Color := clwhite;
  Form1.canvas.moveto(wormx[i],wormy[i]);
  Form1.Canvas.LineTo(wormx[i],wormy[i]);
end;

  Form1.Canvas.Pen.Color := clwhite;
  Form1.canvas.moveto(wormx[10],wormy[10]);
  Form1.Canvas.LineTo(wormx[10],wormy[10]);
```

Java

```
// 1.Schlange löschen
    for   (int i = 0; i<10; i++) {
      c.setColor(Color.WHITE);
      c.fillOval(wormx[i],wormy[i],10,10);
      }
```

Python

Trifft nicht zu.

Schritt 3: „Randcrash“ und “Schlangenselbstcrash” programmieren

Randcrash und Schlangencrash sind recht einfach zu implementieren. Es werden lediglich die Koordinaten des Schlangenkopfes mit den Dimensionen des Fensters, bzw. den Koordinaten des Schlangenrumpfes verglichen.

Lazarus

```
// 6. Abfrage Crash Rand
if  (Wormx[1] < 0) or (Wormx[1] > Form1.width-10)
         or (Wormy[1] < 0) or (Wormy[1] > Form1.Height-10) then
            begin
            Form1.timer1.enabled := false;
            Form1.Canvas.TextOut(100, 100,' C R A S H -
                        Nochmal: Spacetaste');  exit;
            end;

 // 7. Abfrage Crash eigener Schwanz
 for i := 2 to 10 do
   begin
   if (wormx[1] = wormx[i]) and (wormy[1] = wormy[i]) then
        begin
        Form1.timer1.enabled := false;
        Form1.Canvas.TextOut(100, 100,' C R A S H -
                  Nochmal: Spacetaste');  exit;
        end;
   end;
```

Java

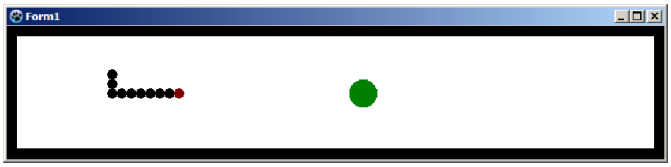
```
// 6. Crash mit dem Spielfeldrand

    if ( (wormx[0] < 0) || (wormx[0] > canvas1.getWidth()) ||
            (wormy[0] < 0) || (wormy[0] > canvas1.getHeight()) ) {
                  t.stop();
    }


    // 7. Crash mit dem eigenen Schwanz
    for (int i=1 ; i<10 ;i++) {
      if ((wormx[0] == wormx[i]) && (wormy[0] == wormy[i]))  {
```

	`t.stop();`
	`}`
	`}`

Python	`def testeKollisionRand():`
	`  global timerOn`
	`  Tupel = Z.bbox(Snake[0])`
	`  if (Tupel[0]<0) or (Tupel[1]<0) or (Tupel[2] > Z.winfo_width()) or (Tupel[3] > Z.winfo_height()):`
	`  timerOn = False`
	
	`def testeKollisionSchwanz():`
	`  global timerOn`
	`  for i in range (1, 10):`
	`  TupelKopf = Z.bbox(Snake[0])`
	`  TupelGlied = Z.bbox(Snake[i])`
	`  if (TupelKopf[0] ==`
	`  TupelGlied[0]) and (TupelKopf[1] == TupelGlied[1]):`
	`  timerOn = False`

Schritt 4: Apfel fangen

a) Der grüne Apfel kann einfach als grüner Punkt zufällig irgendwohin gezeichnet werden. Zuvor wird der alte Apfel weiß übermalt.



Lazarus	`var`
	`  apfelx, apfely : integer;`
	`  apfelgroesse : integer = 30;`
	
	`procedure neuerApfel;`
	`  begin`
	`  // Apfel weiß übermalen`
	`  Form1.Canvas.Pen.Width := apfelgroesse;`
	`  Form1.Canvas.Pen.Color := clwhite;`
	`  Form1.canvas.moveto(apfelx,apfely);`
	`  Form1.Canvas.LineTo(apfelx,apfely);`
	
	`  // neue Koordinaten für den Apfel`
	`  apfelx := random(Form1.width-100) + 50;`
	`  apfely := random(Form1.Height-100) + 50;`
	
	`  // Apfel neu zeichnen`
	`  Form1.Canvas.Pen.Width := apfelgroesse;`
	`  Form1.Canvas.Pen.Color := clgreen;`
	`  Form1.canvas.moveto(apfelx,apfely);`
	`  Form1.Canvas.LineTo(apfelx,apfely);`
	`end;`
	
	`procedure move;`
	`var i, px, py : integer;`
	`begin`
	`[. . .]`
	`// 8. Apfel zeichnen`
	`Form1.Canvas.Pen.Width := apfelgroesse;`
	`Form1.Canvas.Pen.Color := clgreen;`

	`Form1.canvas.moveto(apfelx,apfely);`
	`Form1.Canvas.LineTo(apfelx,apfely);`
	`end;`

Java	`int Apfelx = -30;`
	`int Apfely = -30;`
	
	`void neuerApfel (){`
	`  Graphics g = canvas1.getGraphics();`
	`  Graphics2D c = (Graphics2D) g;`
	
	`  c.setColor(Color.WHITE);`
	`  c.fillOval(Apfelx,Apfely,20,20);`
	
	`  Apfelx = (int) (Math.random()* canvas1.getWidth() -100) +50  ;`
	`  Apfely = (int) (Math.random()* canvas1.getHeight() -100) +50  ;`
	
	`  c.setColor(Color.GREEN);`
	`  c.fillOval(Apfelx,Apfely,20,20);`
	`}`

Python	Siehe unten.

b) Dann wird geprüft, ob die Koordinaten des Schlangenkopfes in einem Toleranzbereich um die Apfelmitte (Durchmesser des Apfels) liegen.

Lazarus	`// 9. Abfrage Apfel gefunden`
	`if (wormx[1] > apfelx-15)  and (wormx[1] < apfelx+15)`
	`  and (wormy[1] > apfely-15) and (wormy[1] < apfely+15)  then`
	`  neuerApfel;`

Java	`void move () {`
	`[. . . ]`
	`// 8. Apfel`
	`  if ((wormx[0] < Apfelx+10) && (wormx[0] > Apfelx-10) && (wormy[0] < Apfely+10) && (wormy[0] > Apfely-10) ) {`
	`  neuerApfel();`
	`  } else {`
	`  c.setColor(Color.GREEN);`
	`  c.fillOval(Apfelx,Apfely,20,20);`
	`  } // end of if-else`
	`}`

Python	`def testeKollisionApfel():`
	`  global Apfelx`
	`  global Apfely`
	`  Liste = Z.find_overlapping(Apfelx,Apfely, Apfelx+20, Apfely+20)`
	`  if len(Liste) > 1:`
	`  Apfelx = random.randint(20,480)`
	`  Apfely = random.randint(20,480)`
	`  Z.coords(Apfel, Apfelx, Apfely, Apfelx+20, Apfely+20)`