➲ AUFGABE 3: GUTE STUBE

Lösungsidee und Programmdokumentation:

Das herausfiltern der relevanten Wörter aus einer Zeile mache ich so. Ich speichere in der Menge `stopper` alle Buchstaben ab die ein Wortende bedeuten (LEER, PUNKT, KOMMA). In einer Schleife geht das Programm jeden Buchstaben des Satzes durch und wenn `stopper` vorkommt schneidet es vom vorherigen bis zu diesem `stopper` den Text aus und speichert ihn in `wort`. Da ich davon ausgehe, dass nur maximal 10 Objekte vom gleichen Typ vorhanden sind, teste ich ob das vorletze Zeichen der Variable `wort` ein `'#'` ist, um es als Objekt zu erkennen. Im Array `befehle` befinden sich alle Wörter die eine Lagebeschreibung darstellen. Entspricht `wort` einem String aus dem Array, so speichere ich den Index dieses Wortes in der Variable `link` ab. Die 3 Varaiblen die dadurch entstehen speichere ich im dynamischen Array `auftraege` vom Typ `todos` ab. `Obj1` das erste Objekt, `Obj2` das zweite und `link` die Nummer der Verknüpfung zwischen den zwei Objekten. Dieser Vorgang befindet sich zu Beginn der Prozedur `bilderzeugung`.
Danach verarbeitet das Programm die entstandenen Aufträge vom `auftraege` Array und erzeugt die Objekte im `objekte` Array. Dabei geht es so vor:
Um ein Objekt relativ zum anderen sofort positionieren zu können muss, man bereits die Position des anderen Objekts kennen. Deshalb nimmt sich mein Programm das erste Objekt (`auftraege[0].obj1`) aus dem `auftaege` Array und schreibt dessen Namen `name` auf einen neuen Platz des `objekte` Array. `X`, `Y` und `Z` Koordinaten des Objektes werden auf den Wert des Mittelpunktes der jeweiligen Interwalle in denen sich die Koordinaten der schon vorhandenen anderen Objekte befinden gesetzt. `FX`, `FY` und `FZ` halten fest, durch welche Verschiebung sich die Koordinate des Objektes ergeben hat. Deshalb wird sie zu Beginn immer auf 0 gesetzt. Überschneidet sich das soeben beschriebene Objekt mit einem anderen so wird es so lange nach vorne verschoben, bis die Funktion `kollision(index)` den Wert –1 annimmt (=keine Kollision, sonst gibt sie den Index es Objektes an mit dem eine Kollision stattfindet). Da jetzt ein Objekt feststeht kann die Prozedur `relations(objekt.name)` gestartet werden. Sie durchsucht den `auftrage` Array nach entsprechenden Objekten und führt beim Auffinden zuerst die `setobjekt(name1, name2, link)` Funktion aus. Wenn dadurch keine Lagebeziehung hergestellt werden konnte noch einmal `setobjekt(name2, name1, -link)` und dann `relations` mit dem neuen Objekt als Parameter. So werden alle Objekte durchgegangen, die sich relativ zu schon bekannten Objekten positionieren lassen. Sind noch andere Objekte im `auftraege` Array vorhanden, so startet die ganze Prozedur wieder am Anfang und nimmt sich das erste Objekt aus dem `auftraege` Array.
Jetzt genaueres zur `setobjekt` Prozedur. Sie durchsucht zuerst den `objekte` Array nach `name1` und `name2` und ermittelt daraus index1 und index2. Wenn es ein Objekt mit `name2` noch nicht gibt, so wird ein neues mit `fx`, `fy` und `fz` = 0 erstellt. `Name1` ist immer das Objekt, welches schon eine Position hat. Da sich das Objekt als `obj1` oder `obj2` im `auftraege` Array befunden haben kann habe ich die link Variable auch negativ zugelassen. Wenn 1 für rechts steht, dann steht –1 für links. Jetzt kommt es zur Positionsbestimmung. Damit das Programm Verschiebungen von vorher nicht zerstört habe ich es so gemacht. Eine Verschiebung in x-Richtung mit `link`=1 wird nur dann gemacht, wenn `fx` 0 oder 1 ist. Wenn die `fx` Variable 0 ist, dann wurde das Objekt noch nie durch einen echten Auftrag in x Richtung positioniert, wenn `fx` 1 ist dann wurde das Objekt schon mal nach links verschoben und kann ruhig weiter nach links verschoben werden. Nach der Festlegung der x Koordinate wird `fx` dann auf 1 gesetzt. Analog wird es bei den anderen Koordinaten gehandhabt. Die nicht durch die `link` Variable betroffen Koordinaten werden dennoch nicht einfach zufällig vergeben. Die y-Koordinate wird, wenn es nicht anders verlangt wird, immer so gesetzt, dass sich die Unterkanten der Objekte immer auf gleicher Höhe befinden. Die x-Koordinate wird immer so gelegt, dass sich die Mittelpunkte auf einer Linie befinden, und die z-Koordinate immer beibehalten, wenn kein Auftrag eine spezielle Positionsbeziehung verlangt.
Durch diese Positionsgebung kann es natürlich auch vorkommen, dass sich 2 Objekte überschneiden. Deshalb wird nach jeder Koordinatenfestlegung die Funktion `kollision(index)` aufgerufen. Sie gibt –1 aus wenn keine Kollision auftritt, sonst den `index3` des Objektes mit dem eine Kollision stattfindet. Gibt die Funktion einen Index zurück, so wird die Positionsbestimmung relativ zum Objekt mit dem neuen Index durchgeführt. Diese Befehlsabfolge wird so lange durchgeführt, bis `index3` –1 wird. Dabei funktioniert die `kollision` Funktion so. In einer Schleife für alle Objekte wird getestet, ob sie die gleiche z-Koordinate besitzen. Wenn ja, dann wird getestet ob sich die Unterkante des einen Objekts oberhalb der Oberkante des anderen, oder die Oberkante des einen unterhalb der Unterkante

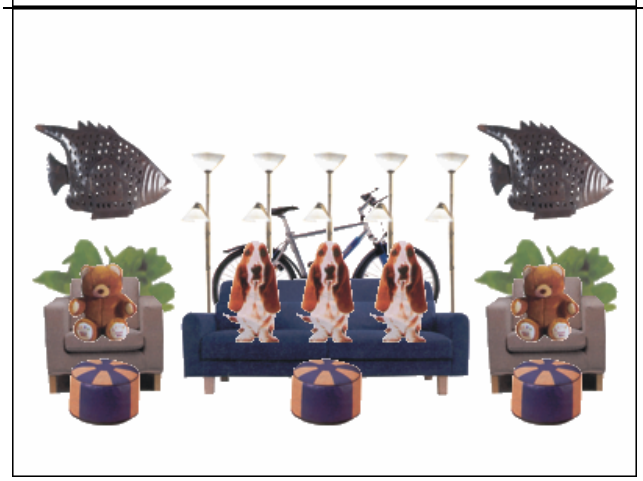
des andern oder, … . Wenn nein, findet eine Kollision statt und der Index wird ausgegeben. Die Kollisionsabfrage funktioniert also über das Gegenereignis.
Da bei den Koordinatenberechnungen die Abmessungen der einzelnen Bitmaps wichtig ist, habe ich die Funktion `getabm(name):picgeo` geschrieben. Sie gibt Breite und Höhe unter Eingebe des Namen aus, indem sie den Array `bilder` durchsucht und bei Übereinstimmung der Namen den kompletten Datensatz ausgibt.
Da Sie mir ja künstlerische Freiheit gewährt haben (Sir), habe ich den „neben“ Befehl nicht gerade vorbildlich behandelt und ihn einfach durch eine Zufallszahl auf rechts oder links umgelegt. Wenn man es genau nehmen wollte, müsste man entscheiden welche Variante günstiger ist. Mehrdeutigkeiten mag es immer geben, da rechts, links, usw. nicht besagt wie weit. Unvollständigkeiten werden dadurch kompensiert, dass die nicht genau festgelegten Koordinaten immer so gesetzt werden, dass die Unterkante der BMPs auf gleicher Höhe ist, sich ihre Mitten auf einer vertikalen Befinden und die Tiefenkoordinaten z gleich sind. Wiedersprüche werden durch die `fx`, `fy` und `fz` Variablen aufgedeckt. Wurde ein Objekt schon einmal nach hinten Verschoben, wo ist es nicht mehr möglich es nach vorne zu schreiben. Dabei mag zwar nicht die „günstigste“ Anordnung, mit einem Maximum an befolgten Anweisungen, entstehen, das war aber nicht verlangt.

Programmablaufprotokoll:

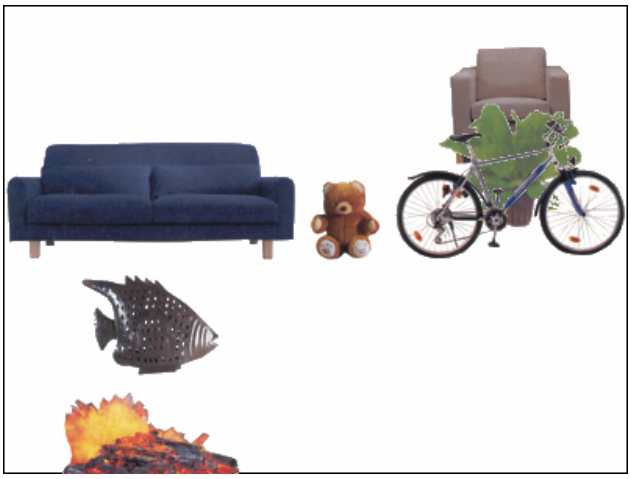
Beim Start des Programms ist immer automatisch die vorgegebene Satzfolge im Textfeld. Will man das zugehörige Bild sehen muss man auf den Knopf „Bild zeichnen“ klicken. Bei der Eingabe des Textes ist zu beachten, dass ein Befehl immer in einer Zeile stehen muss und hinter dem letzen Wort, wenn es ein Objekt ist, auch ein Punkt stehen sollte, da sonst kein Stopper vorhanden ist und das Programm das Wort nicht wahrnimmt. Mit dem Knopf „Text laden“, öffnet man den Standardladedialog, um sich eine Textdatei auszusuchen, die geladen werden soll. Nach dem Klick auf „Bild speichern“ gibt man einen Dateinamen an und das Bild wird im BMP Format gespeichert.

3. Beispiele:

- Ich hätte gerne das Fahrrad#1 links von Stehlampe#1.
- Der Sessel#1 steht links von Sessel#2.
- Couch#1 befindet sich vor Zimmerpflanze#2.
- Sessel#1 ist neben dem Lagerfeuer#1.
- Ein Sessel#2 steht hinter Lagerfeuer#1.
- Sitzkissen#1 unter Haustier#1.
- Kunstobjekt#1 schwebt über Lagerfeuer#1.
- Die Zimmerpflanze#1 hinter Teddybär#1.
- Fahrrad#1 ist irgendwo rechts vom Sessel#2.
- Stehlampe#1 sieht man links von Sessel#1.
- Doch die Zimmerpflanze#2 steht hinter dem Fahrrad#1.



- Stehlampe#1 steht rechts von Stehlampe#2.
- Stehlampe#2 rechts Stehlampe#3.
- Sitzkissen#1 vor Sessel#1.
- Couch#1 vor Stehlampe#3.
- Sessel#1 rechts Couch#1.
- Haustier#3 rechts Haustier#1.
- Zimmerpflanze#1 hinter Sessel#1.
- Kunstobjekt#2 über Zimmerpflanze#2.
- Fahrrad#1 hinter Couch#1.
- Stehlampe#4 rechts Stehlampe#5.
- Sitzkissen#2 vor Sessel#2.
- Stehlampe#3 rechts Stehlampe#4.
- Zimmerpflanze#2 hinter Sessel#2.
- Sitzkissen#3 vor Couch#1.
- Teddybär#1 über Sitzkissen#1.
- Kunstobjekt#1 über Zimmerpflanze#1.
- Teddybär#2 über Sitzkissen#2.
- Haustier#1 über Sitzkissen#3.
- Sessel#2 links Couch#1.
- Haustier#2 links Haustier#1.



- Fahrrad#1 vor Sessel#1.
- Sessel#1 hinter Zimmerpflanze#1.
- Fahrrad#1 vor Zimmerpflanze#1.
- Fahrrad#1 rechts Teddybär#1.
- Couch#1 links Fahrrad#1.
- Teddybär#1 rechts Couch#1.
- Kunstobjekt#1 über Lagerfeuer#1.
- Kunstobjekt#1 unter Couch#1.
- Lagerfeuer#1 unter Couch#1.

Quelltext (Project3.dpr):

```
unit aufgabe3;

interface

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

type
  TForm1 = class(TForm)
    Image1: TImage;
    Memo1: TMemo;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    OpenDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    procedure ZeichnenClick(Sender: TObject);
    procedure LadenClick(Sender: TObject);
    procedure OpenDialog1CanClose(Sender: TObject; var CanClose: Boolean);
    procedure SaveDialog1CanClose(Sender: TObject; var CanClose: Boolean);
    procedure SpeichernClick(Sender: TObject);

  private
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
  end;


type picgeo=record
     name : string;
     b, h : byte;
     end;
type objekt=record
     x, y, z : integer;
     fx, fy, fz : byte;
     name     : string;
     end;
type todos  = record
     obj1, obj2 : string;
     link : byte;
     end;

const stoper : set of char = [' ','.',','];
      befehle : array[1..7] of string = ('links', 'rechts', 'neben', 'vor', 'hinter',
'über', 'unter');
      bilder : array[1..10] of picgeo = ((name:'couch'; b:185; h:75),
                                         (name:'fahrrad'; b:147; h:94),
                                         (name:'haustier'; b:37; h:65),
                                         (name:'kunstobjekt'; b:89; h:65),
                                         (name:'lagerfeuer'; b:114; h:63),
                                         (name:'sessel'; b:78; h:76),
                                         (name:'sitzkissen'; b:45; h:44),
```

```
                                        (name:'stehlampe'; b:29; h:136),
                                        (name:'teddybär'; b:44; h:51),
                                        (name:'zimmerpflanze'; b:93; h:81));

var
  Form1: TForm1;
  objekte   : array of objekt;
  auftraege : array of todos;
implementation

{$R *.DFM}

procedure putpic(x, y:integer; typ:string);//Lädt ein bmp-Datei und kopiert sie ins image1
var bitmap : tbitmap;
begin
 Bitmap := TBitmap.Create;
 Bitmap.LoadFromFile(copy(typ, 1, length(typ)-2)+'.bmp');
 bitmap.Transparent := true;                                  //weiss soll Transparent sein
 bitmap.TransparentColor :=  $00FFFFFF;
 form1.image1.Canvas.Draw(x, y, Bitmap);
 bitmap.free;
END;


function getabm(name:string): picgeo;//Gibt Breite und Höhe eines Bitmaps vom Typ name aus
VAR i : byte;
BEGIN
 i := 0;
 REPEAT inc(i) UNTIL copy(name,1, length(name)-2)=bilder[i].name;
 getabm := bilder[i];
END;


function kollision(index:byte):integer;      //Testet ob sich das Objekt mit einem anderen
VAR i:byte;                        //überschneidet und gibt wenn ja dessen index aus sonst -1
    hit: boolean;
BEGIN
 i := 0;
 hit := false;
 kollision := -1;
 WHILE (i<=length(objekte)-1) and not(hit) do BEGIN
  if  (i<>index) and (objekte[index].z = objekte[i].z) and
      not(((objekte[index].x > objekte[i].x+getabm(objekte[i].name).b) or
      (objekte[index].x+getabm(objekte[index].name).b < objekte[i].x)) or
      ((objekte[index].y > objekte[i].y+getabm(objekte[i].name).h) or
      (objekte[index].y+getabm(objekte[index].name).h < objekte[i].y))) then
      BEGIN
      kollision := i;
      hit := true;
      END;
   inc(i);
   END;
END;


function setobjekt(name1, name2:string; link:integer):boolean; //Setzt das Objekt2 relativ
Var vorhanden: boolean;                                  //zu Objekt1 mit der Regel in link
    i, index1, index2 : byte;
    index3:integer;
BEGIN
 vorhanden := false;
 setobjekt := true;
 For i := 0 to length(objekte)-1 do if name2 = objekte[i].name then BEGIN
  vorhanden := true;                       //Ermittlung des index des zu setzenden Objekts
  index2 := i;
  END;
 For i := 0 to length(objekte)-1 do if name1 = objekte[i].name then index1 := i;
                                            //Ermittlung des index des Ausgansobjektes
 if not(vorhanden) then BEGIN                //Erzeugt ein neues Objekt wenn es sich noch
  setlength(objekte, length(objekte)+1);                //nicht im Objekte Array befindet
  index2 := length(objekte)-1;
  objekte[index2].name := name2;
  objekte[index2].fz := 0;
  objekte[index2].fx := 0;
  objekte[index2].fy := 0;
  end;

 if abs(link)=3 then link:=random(2)+1;             //Neben in rechts oder links umwandeln
```

```
case link of
 2, -1: if objekte[index2].fx<>1 then BEGIN                                          //RECHTS
   if objekte[index2].fz=0 then objekte[index2].z := objekte[index1].z;
   objekte[index2].x := objekte[index1].x-10-getabm(name2).b;
   if objekte[index2].fy=0 then
    objekte[index2].y := objekte[index1].y-getabm(name2).h+getabm(name1).h;
   REPEAT
    index3 := kollision(index2);
    if index3<>-1 then objekte[index2].x := objekte[index3].x-10-getabm(name2).b;
   UNTIL index3=-1;
   objekte[index2].fx := 2;
   objekte[index1].fx := 1;
   END else setobjekt := false;
 1, -2: if objekte[index2].fx<>2 then BEGIN                                          //LINKS
   if objekte[index2].fz=0 then objekte[index2].z := objekte[index1].z;
   objekte[index2].x := objekte[index1].x+10+getabm(name1).b;
   if objekte[index2].fy=0 then
    objekte[index2].y := objekte[index1].y-getabm(name2).h+getabm(name1).h;
   REPEAT
    index3 := kollision(index2);
    if index3<>-1 then
     objekte[index2].x := objekte[index3].x+10+getabm(objekte[index3].name).b;
   UNTIL index3=-1;
   objekte[index2].fx := 2;
   objekte[index1].fx := 1;
   END else setobjekt := false;
 -6, 7: if objekte[index2].fy<>6 then BEGIN                                          //UNTER
   if objekte[index2].fz=0 then objekte[index2].z := objekte[index1].z;
   if objekte[index2].fx=0 then
    objekte[index2].x := objekte[index1].x+round((getabm(name1).b-getabm(name2).b)/2);
   objekte[index2].y := objekte[index1].y-getabm(name2).h-10;
   REPEAT
    index3 := kollision(index2);
    if index3<>-1 then  objekte[index2].y := objekte[index3].y-getabm(name2).h-10;
   UNTIL index3=-1;
   objekte[index2].fy := 7;
   objekte[index1].fy := 6;
   END else setobjekt := false;
 -7, 6: if objekte[index2].fy<>7 then BEGIN                                          //ÜBER
   if objekte[index2].fz=0 then objekte[index2].z := objekte[index1].z;
   if objekte[index2].fx=0 then
    objekte[index2].x := objekte[index1].x+round((getabm(name1).b-getabm(name2).b)/2);
   objekte[index2].y := objekte[index1].y+getabm(name1).h+10;
   REPEAT
    index3 := kollision(index2);
    if index3<>-1 then objekte[index2].y :=
objekte[index3].y+getabm(objekte[index3].name).h+10;
   UNTIL index3=-1;
   objekte[index2].fy := 6;
   objekte[index1].fy := 7;
   END else setobjekt := false;
 4, -5: if objekte[index2].fz<>5 then BEGIN                                          //VOR
   objekte[index2].z := objekte[index1].z+1;
   if objekte[index2].fx=0 then
    objekte[index2].x := objekte[index1].x+round((getabm(name1).b-getabm(name2).b)/2);
   if objekte[index2].fy=0 then
    objekte[index2].y := objekte[index1].y-getabm(name2).h+getabm(name1).h;
   REPEAT
    index3 := kollision(index2);
    if index3<>-1 then objekte[index2].z := objekte[index3].z+1;
   UNTIL index3=-1;
   objekte[index2].fz := 4;
   objekte[index1].fz := 5;
   END else setobjekt := false;
 5, -4: if objekte[index2].fz<>4 then BEGIN                                          //HINTER
   objekte[index2].z := objekte[index1].z-1;
   if objekte[index2].fx=0 then
    objekte[index2].x := objekte[index1].x+round((getabm(name1).b-getabm(name2).b)/2);
   if objekte[index2].fy=0 then
    objekte[index2].y := objekte[index1].y-getabm(name2).h+getabm(name1).h;
   REPEAT
    index3 := kollision(index2);
    if index3<>-1 then objekte[index2].z := objekte[index3].z-1;
   UNTIL index3=-1;
   objekte[index2].fz := 5;
   objekte[index1].fz := 4;
   END  else setobjekt := false;
 END;
```

```
END;


procedure getcenter(var midx, midy, midz, minz, maxz: integer);
var i, maxx, maxy, minx, miny: integer;
BEGIN
maxx := 0;
 maxy := 0;                                          //Ermittlung des Bereichs in dem sich
 maxz := 0;                                                    //die x,y,z Werte befinden
 minx := 8000;
 miny := 8000;
 minz := 8000;

 for i := 0 to length(objekte)-1 do BEGIN
  if objekte[i].x<minx then minx :=objekte[i].x;
  if objekte[i].y<miny then miny :=objekte[i].y;
  if objekte[i].z<minz then minz :=objekte[i].z;
  if objekte[i].x+getabm(objekte[i].name).b>maxx then
   maxx :=objekte[i].x+getabm(objekte[i].name).b;
  if objekte[i].y+getabm(objekte[i].name).h>maxy then
   maxy :=objekte[i].y+getabm(objekte[i].name).h;
  if objekte[i].z>maxz then maxz :=objekte[i].z;
 END;

 midx := (maxx+minx)div 2-200;
 midy := (maxy+miny)div 2-150;
 midz := (maxz+minz)div 2;
END;


procedure bildausgabe;                              //Gibt alle Objekte zental in image1 aus
var i, maxx, maxy, maxz, minx, miny, minz: integer;
    z, midx, midy, midz : integer;
BEGIN
 form1.image1.Canvas.Brush.Color := clwhite;                                  //Hintergrund
 form1.image1.Canvas.Rectangle(0,0,400,300);
 getcenter(midx, midy, midz, minz, maxz);
 for z := maxz downto minz do                       //Die einzelnen Tiefenebenden durchgehen
  for i := 0 to length(objekte)-1 do
  if objekte[i].z = z then                                    //Bilder in das Image übertragen
   putpic(objekte[i].x-midx, objekte[i].y-midy-(objekte[i].z-midz)*20, objekte[i].name);

 form1.Image1.Refresh;
END;



procedure relations(name:string);  //Durchsucht die auftaege nach dem schon bekannten name
var i,k : byte;
    namesaver : string;
BEGIN
 i := 0;
 if length(auftraege)<>0 then
  REPEAT
   if auftraege[i].obj1=name then BEGIN
    if not(setobjekt(name, auftraege[i].obj2, auftraege[i].link))
     then setobjekt(auftraege[i].obj2, name, -auftraege[i].link);
    namesaver := auftraege[i].obj2;
    for k := i+1 to length(auftraege)-1 do auftraege[k-1]:=auftraege[k];
    setlength(auftraege, length(auftraege)-1);                      //Entfernt den Auftrag
    relations(namesaver);                                            //Führt Rekursion aus
    END
   ELSE if auftraege[i].obj2=name then BEGIN
    if not(setobjekt(name, auftraege[i].obj1, -auftraege[i].link))
     then setobjekt(name, auftraege[i].obj1, -auftraege[i].link);
    namesaver:=auftraege[i].obj1;
    for k := i+1 to length(auftraege)-1 do auftraege[k-1]:=auftraege[k];
    setlength(auftraege, length(auftraege)-1);                      //Entfernt den Auftrag
    relations(namesaver);                                            //Führt Rekursion aus
   END
   ELSE inc(i);
  UNTIL i >= length(auftraege);
END;



procedure bilderzeugung;                  //Filtert aus dem Memo1 die auftraege heraus und
```

```
var i, s, e, b : byte;                                          //startet die Rekursion
    wort : string;
    auftrag : todos;
    midx, midy, midz, dummy: integer;
begin
 setlength(auftraege, form1.memo1.Lines.Count);
 setlength(objekte, 0);
 for i := 0 to form1.memo1.Lines.Count-1 do BEGIN         //die einzelnen Zeilen von memo1
  if form1.memo1.Lines.Strings[i]<>'' then BEGIN
   e := 0;
   s := 1;
   auftrag.link := 0;
   auftrag.obj1 := '';
   auftrag.obj2 := '';
   REPEAT
    REPEAT
     inc(e);                                 //Testet alle Zeichen bis es ein Stopzeichen ist
    UNTIL form1.memo1.Lines.Strings[i][e] in stoper;
    wort :=  lowercase(copy(form1.memo1.Lines.Strings[i],s, e-s));//Schneidet das Wort aus
    if wort[length(wort)-1] = '#' then if  auftrag.obj1 = '' then auftrag.obj1 := wort
             else auftrag.obj2 := wort;   //Wenn das vorletze Zeichen ein # ist --> Objekt
    for b := 1 to 7 do if wort = befehle[b] then auftrag.link := b;
    s := e+1;                                  //testet ob es ein Lagebeschreibungs-Wort ist
    auftraege[i] := auftrag;                     //Fügt den Auftrag zu den Auftraegen hinzu
   UNTIL s >= length(form1.memo1.Lines.Strings[i]);
  END;
 END;

 REPEAT
  setlength(objekte, length(objekte)+1);                 //Setzt das erste Objekt an 0,0,0
  objekte[length(objekte)-1].name := auftraege[0].obj1;
  getcenter(midx, midy, midz, dummy, dummy);
  objekte[length(objekte)-1].x := midx+200;
  objekte[length(objekte)-1].y := midy+150;
  objekte[length(objekte)-1].z := midz;
  objekte[length(objekte)-1].fx := 0;
  objekte[length(objekte)-1].fy := 0;
  objekte[length(objekte)-1].fz := 0;
  if length(objekte)<>1 then
   REPEAT                                  //Schiebt Objekt bei kollision um eins nach forne
    dec(objekte[length(objekte)-1].z);
   UNTIL kollision(length(objekte)-1)=-1;
  relations(auftraege[0].obj1);    //Startet Suche nach Objekten die sich festlegen lassen
 UNTIL length(auftraege) = 0;
end;


procedure TForm1.ZeichnenClick(Sender: TObject);
begin
bilderzeugung;
bildausgabe;
end;


procedure TForm1.LadenClick(Sender: TObject);
begin
opendialog1.Execute;
end;


procedure TForm1.OpenDialog1CanClose(Sender: TObject; var CanClose: Boolean);
begin
memo1.Lines.LoadFromFile(opendialog1.FileName);
end;


procedure TForm1.SaveDialog1CanClose(Sender: TObject; var CanClose: Boolean);
begin
image1.Picture.SaveToFile(savedialog1.filename);
end;


procedure TForm1.SpeichernClick(Sender: TObject);
begin
savedialog1.Execute;
end;

end.
```