DM 7,50 sfr 7,50 öS 65,–

toolbox

4 ISSN 0935-8315, April 1990
DMV-Verlag, 6. Jahrgang

toolbox

4'90

Computermagazin für die Praxis

Super Basic-Programme

Inline-Generator für Turbo Basic
Quick-Basic und die Maus
Bildverarbeitung auf dem PC

Listing des Monats

Mit OOP in den Weltraum
Turbo-Pascal-Action-Spiel

Alles über VGA

Die Grafikkarten-Story
VGA-Programmierung
für Einsteiger

Die Grenzen fallen!

Wege zu erweitertem Speicher
EMS, XMS und HMA im Griff

Die Geheimnisse des Bootsektors

Multiformatter in Assembler
Kopierschutz gelüftet

Große toolbox-Diskettenaktion
Bestellen Sie Ihre SQL-Datenbank

TOP-LISTING DES MONATS

Raimond Reichert bewegt mit Turbo Pascal 5.5 Sternenschiffe durch den Weltraum

Wir wollen Ihre Idee zur Geltung bringen!

- Nehmen Sie an unserem Top-Listing-Wettbewerb teil: Schicken Sie uns das pfiffigste Werk Ihrer Programmierkunst!
- Eine bestimmte "thematische Ecke" schreiben wir nicht vor, auch keine einheitliche Programmiersprache.
- Arbeiten Sie an Ihrer Programmidee, solange es nötig ist. Den Einsendetermin bestimmen Sie selbst.
- Sourcecode bitte auf Diskette; die Länge des Programms sollte 1000 Zeilen nicht überschreiten. Begrenzen Sie bitte die Länge der Programmzeilen auf je 60 Zeichen.
- Programme, die Sie an uns schicken, müssen frei von Rechten Dritter sein, damit's keinen juristischen Ärger gibt. Also bitte nur eigene Meisterwerke schicken!
- Jeden Monat ermitteln wir einen Wettbewerbssieger. Die Siegerprogramme werden mit 1000 DM prämiert und in der TOOLBOX abgedruckt. Ihre Einsendung bleibt drei Monate lang im Rennen.
- Wir freuen uns auf Ihre Einsendung!

Der Gewinner des Monats

Raimond Reichert: Zu meinem Einstieg in die Welt der Computer kam ich vor rund zwei Jahren durch einen Freund, der mir Turbo Pascal (damals noch 3.0) in die Hand drückte. Seitdem lassen mich der Computer und das Programmieren nicht mehr los.

So kaufte ich mir dann im letzten Sommer Turbo Pascal 5.5. Nach einigen kleineren Programmen entstand "Spacewar", mein erstes Spiel. Da in diesem Spiel grafische Objekte über den Bildschirm bewegt werden, fand ich es vorteilhaft, einmal die Vorzüge der objektorientierten Programmierung für "Spacewar" zu nutzen.

Wenn ich nicht gerade am Computer sitze, gehe ich in die Schule, und zwar in die 10. Klasse des Gymnasiums Sarnen in der Nähe von Luzern. Außer dem Computer sind meine Hobbys Klavierspielen, Segelflugmodelle, Judo und Lesen.

An dieser Stelle möchte ich meinem Freund Daniel Braun besonderen Dank aussprechen. Von ihm stammt die Idee zum Spiel.

Spielerei mit Flug-Objekten

Spacewars

von Raimond Reichert

Oft erliegen auch die ernsthaftesten Programmierer der Faszination von Objekten, die über einen Spielhallen-Bildschirm flitzen. Und dann kommt auch schon einmal der Wunsch auf, selbst einmal ein schnelles "Ballerspiel" zu entwickeln. Warum nicht in Turbo Pascal mit objektorientierter Programmierung?

Bei der Überlegung, wie es zu bewerkstelligen ist, unabhängige Objekte wie eigene und "außerirdische" Raketen über den Bildschirm sausen zu lassen, ergab sich eine gute Gelegenheit, die Möglichkeiten der OOP unter Turbo Pascal 5.5 zu nutzen. Der Vorteil bei der Verwendung von Objekten gegenüber dem Aufwand mit Datenarrays für die Raketen und die Schüsse wurde schnell offenbar.

Was am Ende stehen sollte, war ein einfaches Spiel. Mehrere feindliche Raketen bewegen sich in einem zufälligem Zickzack-Kurs und feuern dabei auch ab und zu in unregelmäßigen Abständen. Gleichzeitig bewegt sich der Spieler, schießt und wird beschossen. Punkte gibt es, wenn eine "außerirdische" Rakete getroffen wird. Die eigene Rakete geht verloren, wenn ein Gegner trifft. Wie viele Raketen noch zur Verfügung stehen, wird am unteren Bildschirmrand angezeigt.

Bei der Verwirklichung der Spielidee müssen wir zunächst auf eventuelle Auflösungsprobleme mit unterschiedlichen Grafikkarten eingehen. Da Borland mit den BGI-Treibern die gängigsten Karten versorgt, ist die Erkennung und Einstellung des verwendeten Videoadapters verhältnismäßig einfach. Um das Listing kurz zu halten, haben wir auf die Berechnung der Ausgabetexte über "GetMaxX" und "GetMaxY" oder die Zentrierung des Titels verzichtet. Zur Anpassung müssen Sie lediglich die Konstanten der Ränder sowie die Prozedur "Status" ändern.

Wenn Ihnen die Raketen der "Invaders" zu groß sind, ändern Sie einfach die Prozedur "RocketImage". Falls Sie das Raumschiff des Spielers, seine Schüsse oder die der Raketen verändern wollen, können Sie das in den Konstruktoren des jeweiligen Objekts tun. Sind die Bewegungen irgendeines Objekts zu schnell, so kann auch dies über über die Variablen "DifX" beziehungsweise "DifY" des betroffenen Objekts gesteuert werden.

Objekte, Instanzen, Methoden

Das Spiel hat drei verschiedene Bilder. Das erste ist das Titelbild, das nur einmal gezeigt wird und in kürzester Form Spielerklärungen gibt, das nächste ist das Spielbild, das dritte zeigt die erreichten Punkte an und fragt, ob noch ein Spiel gewünscht wird. Ein kleiner Hinweis für glückliche Besitzer eines schnelleren Rechners: Je größer die Konstante "HOCR" am Anfang des Programms gewählt wird, desto langsamer fliegen die feindlichen Raketen, desto schneller aber auch der eigene "Space-Jet". Überhaupt kann die Geschwindigkeit des Spiels über "HOCR" und "DifX" beziehungsweise "DifY" der verschiedenen Objekte gesteuert werden.

Das Spiel verwendet vier verschiedene Objekte. Das erste ist OneShot als Elternobjekt für alle nachfolgenden. "X" und "Y" sind die Instanzvariablen für die momentane Position des Schusses. "Bottom", "Left", "Right" und "Top" sind die Bewegungsgrenzen für einen Schuß. Da sich ein Schuß aber immer nach unten bewegt, ist nur "Bottom" wichtig. Wird "Bottom" überschritten, wird die Variable "Visible" FALSE. Der Schuß ist also nicht mehr sichtbar und wird deshalb auch nicht mehr fortbewegt.

In "Im" (Image) vom Typ "ImageRec" wird das Bild eines Schusses gespeichert, das in dem Constructor "Init" auf der zweiten Bildschirmseite gezeichnet wird. "DifY" schließlich gibt an, um wie viele Punkte ein Schuß jeweils nach unten bewegt wird; hier ist die Geschwindigkeit der Raketenschüsse steuerbar. Und zuletzt noch "xl" und "yl": Sie übergeben die x- und y-Ausdehnung des Bildes an den Constructor und die Methoden von OneShot.

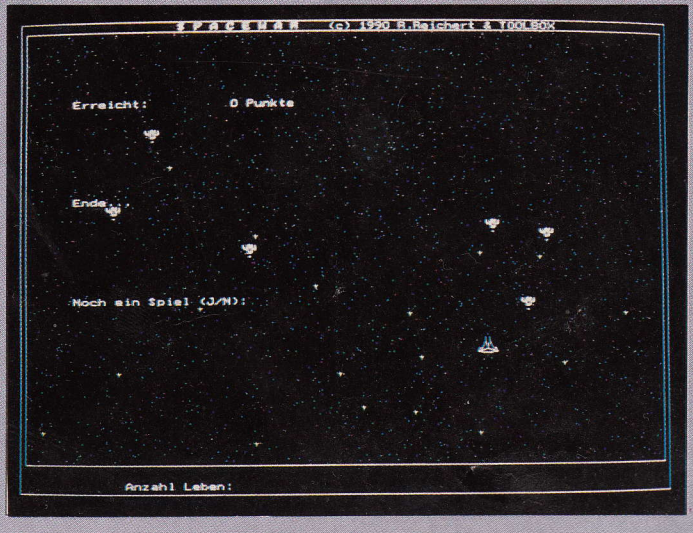
Init: "Tanken" vor dem Start

"Init" wird gebraucht, um das Objekt dynamisch zur Laufzeit auf dem Heap anzulegen. Nach der Compilation ist statisch nur der Speicher für den Zeiger auf das Objekt reserviert. Erst zur Laufzeit wird der Speicher für das Objekt belegt. Die erweiterte Syntax von "New" seit der Version 5.5 von Turbo Pascal läßt als zweiten Parameter den Namen eines Constructors zu, hier also "Init". Es wird also der Speicher belegt und "Init" aufgerufen. Natürlich könnte der Befehl in "OneRocket.Init" statt

```
New(Shots[i],Init); auch
New (Shots[i]); Shots[i]↑.Init;
```

heißen, aber das erste Verfahren ist bequemer.

Bei der Initialisierung zeichnet der Constructor "Init" das Bild eines Schusses und setzt die Instanzvariablen von OneShot. Die Methode "SetStartPos" (nx,ny : INTEGER) setzt einen Schuß in seine Startposition an "nx", "ny". Von dort aus wird sich der



Schuß gerade abwärts bewegen, entweder bis er den Spieler getroffen hat, oder bis er am Bildschirmrand verschwunden ist. Die Methode "Move" bewegt einen Schuß um "DifY" Punkte nach unten, wobei erst einmal geprüft wird, ob der Schuß nach seiner Bewegung auch noch sichtbar ist. Ist das nicht der Fall, wird "Visible" FALSE, sonst wird das Bild neugezeichnet. Anschließend wird das Bild an der alten Position auf jeden Fall gelöscht.
Übrigens läuft das Zeichnen über "PutImage" und im XOR-Modus, da es damit möglich ist, zuerst an der neuen Position zu zeichnen und hinterher an der alten zu löschen. Natürlich wäre auch die umgekehrte Reihenfolge möglich: zuerst löschen, dann zeichnen. Dadurch würde aber ein ruckartiger Bewegungsablauf entstehen, was durch den XOR-Modus einigermaßen verhindert wird. Die Prüfung, ob der Spieler getroffen wurde, geschieht über die globalen Variablen "px1"/"py1" und "px2"/"py2", die die linken oberen und den rechten unteren Ecken der momentanen Spieler bezeichnen. Ein kleiner Hinweis für Schummler: Diese vier Variablen werden in "Player-Object.Move" gesetzt.
Wenn die x- und y-Position des Schusses sich in diesem Bereich befinden, wird dem Spieler ein Leben genommen (bitte nicht allzu wörtlich nehmen!), das Schußbild gelöscht und schließlich die verminderte Anzahl Leben des Spielers noch angezeigt (der Aufruf von Status). Wer nicht sehen will, wie seine Spieler-Leben erlöschen, der sollte überall den Aufruf von Status entfernen.

Constructor: Die Raketenwerft

Nach diesen ausführlichen Erläuterungen zu "OneShot" ist der Objekttyp "OneRocket" eigentlich schon fast erklärt, da er alle Datentypen sowie Methoden von "OneShot" erbt. Wie das geht? Ganz einfach:

```
OneRocket = OBJECT (OneShot)
```

Das bedeutet, daß "OneRocket" ein Kindobjekt von "OneShot" ist und alle Eigenschaften erbt. Warum aber müssen dann die Methoden nochmals definiert werden? Weil sie für OneRocket neu implementiert wurden. Sie haben teilweise eine leicht veränderte Aufgabe als in "OneShot".
"ChangeDir" gibt an, nach wie vielen Bewegungen die Richtung der Rakete geändert werden soll, wobei dieser Wert sich aus CDMin + Random (CDCons) errechnet. "CDMin" und "CDCons" sind globale Konstanten. Vor einer Kursänderung werden mindestens "CDMin" Bewegungen gemacht. Dazu kommt noch der Zufallswert von "CDCons". Wenn Sie strikte Regelmäßigkeit lieber haben, sollten Sie "CDCons" gleich 0 setzen. Ähnlich verhält es sich mit "NewShot". Diese Variable gibt die Anzahl von Bewegungen an, bevor die Rakete wieder schußbereit ist.
Auch über "HOCR" können Sie die Geschwindigkeit der Raketen steuern, weil "OneRocket" nur dann geruht, sich zu bewegen, wenn "HowOften" gleich 0 ist, sonst wird "HowOften" nur um eins vermindert.
Wo sind sie denn geblieben, unsere Raketenschüsse? Die Raketen sollten doch schießen, sonst ist es ja nicht lustig. Sie verstecken sich hinter der unscheinbaren Bezeichnung "Shots", ARRAY [1..MaxShots] OF↑OneShot. "MaxShots" ist eine globale Konstante und gibt die Anzahl Schüsse jeder Rakete an. Aber lassen Sie sich bitte nicht täuschen: Es bringt nämlich gar nichts, "MaxShots" möglichst hoch zu wählen. Wenn Sie mehr feindliche Schüsse sehen wollen, müssen Sie "NSMin" und "NSCons" verändern. Da dieses Array nur aus Zeigern besteht, müssen die Schüsse dynamisch angelegt werden. Das geschieht im Constructor "Init". Er initialisiert auch sämtliche Variablen, vererbte und eigene. Aufgerufen wird der Constructor von der Prozedur "InitGame", die zu Beginn des Spiels vom Haupt-

programm aus aufgerufen wird. "InitGame" belegt den Speicher für alle Raketen. Der Methode "Move" kommt hier eine wesentlich wichtigere Bedeutung zu als in "OneShot". Bei der Verschiebung um "DifX" und "DifY" Punkte prüft sie zunächst, ob die Rakete irgendeine Grenze überschritten hat: Ist sie zu weit links, taucht sie am rechten Rand auf, will sie den unteren Rand überschreiten, erscheint das Raketenbildchen oben. Die Methode achtet darauf, daß auch die Richtung zur rechten Zeit gewechselt wird und immer mal wieder ein Schuß fällt. Sie prüft auch, ob das Objekt mit dem Spieler kollidiert ist oder ob sie von einem Schuß des Spielers getroffen wurde. Dann wird die Anzahl der Leben vermindert oder die Rakete vom Bildschirm gelöscht.

Ballern mit Methode

"OnePlShot", das Objekt eines Spielerschusses, ist schnell erklärt. Es erbt alle Variablen und Methoden von "OneShot" und implementiert zwei der drei Methoden neu: "Init", weil die Spielerschüsse anders aussehen als die der feindlichen Raketen, sowie "Move", weil der Schuß in die anderen Richtung verläuft und somit eine andere Grenze berücksichtigen muß. Der Spieler beziehungsweise sein Objekt "PlayObj" ist den vorausgegangenen sehr ähnlich, was es neu definiert, ist unwesentlich. Es erbt, wie auch "OneRocket", Methoden und Instanzen von OneShot. Neu ist, daß "DifxCons" und "DifyCons" benötigt werden, da der Wert von "DifX" und "DifY" hier variieren kann. "Init" wurde geändert (Speicherbelegung der Spielerschüsse vom Typ "OnePlShot"), ebenso "Move", da es die Schüsse bewegen muß. Die Methode "ReadKeyBoard" ist für die Auswertung der Tastatureingaben verantwortlich. Gesteuert werden kann über die Cursortasten oder auch auf dem Zahlenblock, allerdings stehen hier dem Spieler noch mehr Richtungen zur Verfügung, nämlich auch diagonal, über <Home/End>, <PgUp/PgDn>. Mit <5> wird das Spielerobjekt angehalten, bis irgendeine Richtungstaste gedrückt wird. Geschossen wird mit <Space>, mit <Esc> wird beendet.

Im Cockpit: Das Hauptprogramm

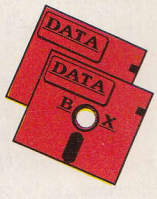
Das Hauptprogramm selbst ist nach solchen Voranstrengungen entsprechend einfach. "InitGame" belegt den Speicher für sämtliche Objekte und zeichnet das Spielbild. Dann durchläuft das Hauptprogramm eine doppelte REPEAT-UNTIL-Schleife. Die äußere ist nur für die Abfrage, ob ein weiteres Spiel gewünscht ist. Die innere ist weitaus interessanter: Sie ruft mit Hilfe einer FOR-Schleife die Methode "Move" aller Raketen auf, die wiederum bewegen alle ihre Schüsse weiter und prüfen auf Treffer. Falls eine Rakete nicht mehr sichtbar sein sollte, wird sofort eine neue ins Weltall geschossen, daher die IF-THEN-ELSE-Konstruktion. Falls ein Tastendruck vorliegt, wird "Player.ReadKeyboard" aufgerufen, das diesen auswertet und darauf mit einer Richtungsänderung, mit einem Schuß oder überhaupt nicht reagiert. Diese innere Schleife wird beendet, wenn "Quit" TRUE ist. Das ist der Fall, wenn entweder <Esc> gedrückt wurde oder der Spieler kein Leben mehr hat.

(wr)

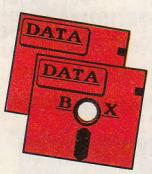
```
(* ------------------------------------------------------ *)
(*                      SPACEWAR.PAS                      *)
(*          (c) 1990  R.Reichert  &  TOOLBOX              *)
(* ------------------------------------------------------ *)
PROGRAM SpaceWar;

USES Graph, Crt;

CONST
  NrRockets = 6;              { Anzahl gegnerischer Raketen }
  HOCR      = 1;              { Zyklus für Raketen          }
  MaxShots  = 5;              { Max. Anz. Schüsse der Rak.  }
  CDCons    = 150;
                { Für Zufallswert der Richtungsänderung     }
  CDMin     = 30;
                { Mind. Anz. Beweg. vor Richtungsänderung   }
  NSCons    = 10;
                { Für Zufallswert für Zeit bis neuschießen  }
  NSMin     = 5;
                { Frühestens alle NSMin-Einheiten schießen  }
  BSpace    = 30;             { Leerraum unten              }
  LSpace    = 5;              { Leerraum links              }
  RSpace    = 5;              { Leerraum rechts             }
  TSpace    = 10;             { Leerraum oben               }
                              { Dasselbe für den Spieler...  }
  PBSpace   = 30;             { Leerraum unten              }
  PLSpace   = 5;              { Leerraum links              }
  PRSpace   = 5;              { Leerraum rechts             }
  PTSpace   = 110;            { Leerraum oben               }
  PlSMax    = 20;             { Max. Schüsse für Spieler    }

TYPE
  { "Einfangen" und Freigeben von Bildschirmausschnitten.  }
  ImageRec  = RECORD
                Size : INTEGER;
                Img  : POINTER
              END;

  OneShot   = OBJECT          { Objekttyp eines Schusses    }
                x, y,         { X- und Y-Positionen         }
                Bottom, Left,
                Right, Top : INTEGER;  { Sichtbare Grenzen  }
                Im         : ImageRec; { Das Bild           }
                DifY       : INTEGER;
                        { Verschiebung um DifY-Punkte       }
                visible    : BOOLEAN;      { Sichtbar ?     }
                xl, yl     : INTEGER;
                      { X- und Y-Ausdehnung des Bildes      }

                CONSTRUCTOR Init;         { Initialisiert   }
                PROCEDURE SetStartPos(nx, ny : INTEGER);
                    { Setzt das 1. Schußbild an nx, ny und  }
                    { visible = true, x = nx, y = ny        }
                PROCEDURE Move;
                    { Bewegt Schuß um DifY Punkte nach      }
                    { unten, wenn HowOften = 0, testet,     }
                    { ob Bottom überschritten wurde und     }
                    { setzt dementsprechend Visible         }
              END;

  OneRocket = OBJECT(OneShot)   { Obj. einer Rakete         }
                ChangeDir,      { Neue Richtung ?           }
                HowOften,       { Zur Geschw. Regulierung   }
                DifX,           { Verschiebung auf X-Achse  }
                NewShot   : INTEGER;      { Neuer Schuß ?   }
                Shots     : ARRAY [1..MaxShots] OF ^OneShot;
                                { Alle Schüsse einer Rakete }

                CONSTRUCTOR Init;
                PROCEDURE Move;
                    { Bewegt, auch alle Schüsse, prüft      }
                    { auf Untergrenze und Richtungs-        }
                    { änderung, "schießt"                   }
                DESTRUCTOR Done;
                    { Für Speicherfreigabe der Schüsse      }
              END;

  OnePlShot = OBJECT(OneShot)           { Ein Spielerschuß  }
                CONSTRUCTOR Init;
                PROCEDURE Move;
                      { bewegt nach oben, prüft auf Grenze  }
              END;

  PlayerObj = OBJECT(OneShot)           { Spielerobjekt     }
                DifX, DifxCons, DifyCons : INTEGER;
                                        { Verschiebungen    }
                ch : CHAR;          { für Tastaturabfrage   }

                CONSTRUCTOR Init;
                PROCEDURE ReadKeybord;
                              { Wertet Tastatureingaben aus }
                PROCEDURE Move;
                DESTRUCTOR Done;
                        { Für Speicherfreigabe der Schüsse  }
              END;

VAR
  AllRockets  : ARRAY [1..NrRockets] OF ^OneRocket;
  i,
  GraphMode,
  GraphDriver : INTEGER;
  Player      : PlayerObj;
  PlShots     : ARRAY [1..PlsMax] OF ^OnePlShot;
                                { Alle Schüsse des Spielers }
  px1, py1,
  px2, py2    : INTEGER;        { Position des Spielers     }
  Quit        : BOOLEAN;        { Spiel fertig              }
  Lifes       : INTEGER;        { Anz. Leben f. Spiel       }
  Points      : INTEGER;        { Punktstand                }
  Again       : BOOLEAN;
  ch          : CHAR;

  (* ------------------------------------------------------ *)
  (* "Fängt" einen rechteckigen Bildschirmausschnitt in     *)
  (* ImgVar ein.                                            *)
  PROCEDURE CatchImage(x1, y1, x2, y2 : INTEGER;
                       VAR ImgVar     : ImageRec);
  BEGIN
    ImgVar.Size := ImageSize(x1, y1, x2, y2);
    GetMem(ImgVar.Img, ImgVar.Size);
    GetImage(x1, y1, x2, y2, ImgVar.Img^);
  END;
```

```
124:  (* ------------------------------------------------- *)
125:  (* Gibt den durch ImgVar belegten Speicher frei      *)
126:  PROCEDURE FreeImage(VAR ImgVar : ImageRec);
127:  BEGIN
128:    IF ImgVar.Img <> NIL THEN BEGIN
129:      FreeMem(ImgVar.Img, ImgVar.Size);
130:      ImgVar.Size := 0;
131:      ImgVar.Img  := NIL;
132:    END;
133:  END;
134:
135:  (* ------------------------------------------------- *)
136:  (* Zeigt die Anzahl Leben des Spielers an, bzw.      *)
137:  (* löscht ein "Leben" vom Bildschirm.                *)
138:  PROCEDURE Status;
139:  BEGIN
140:    WITH Player DO
141:      PutImage(200 + (Lifes + 1) * (xl + 3),
142:               GetMaxY - yl - 5, Im.Img^, XorPut);
143:  END;
144:
145:  (* ------------------------------------------------- *)
146:  (* Zeichnet im unsichtbaren Bildschirm das Bild einer  *)
147:  (* Rakete (ACHTUNG: Wer eine niedrigere Auflösung als  *)
148:  (* Hercules hat, der muß x und y anpassen. Nicht mit   *)
149:  (* den anderen 'Zeichnungen' durcheinanderbringen !)   *)
150:  PROCEDURE RocketImage(VAR Im : ImageRec);
151:  CONST
152:    l1 = 5; l2 = 3; l3 = 1;
153:    b1 = 2; b2 = 6; b3 = 3; b4 = 1;
154:    x = 400; y = 100;
155:  BEGIN
156:    SetLineStyle(1, 0, 3);      SetActivePage(1);
157:    MoveTo(x, y+2);             LineTo(x, y);
158:    LineTo(x-b1, y-l1);         LineTo(x-b1-b2, y-l1-l2);
159:    LineTo(x+b1+b2, y-l1-l2);   LineTo(x+b1+1, y-l1);
160:    LineTo(x, y);               MoveTo(x+b4, y-l1-l2);
161:    LineTo(x+b3, y-l1-l2-l3);   LineTo(x-b3, y-l1-l2-l3);
162:    LineTo(x+b4, y-l1-l2);
163:    Line(x-b1, y-l1, x-b1-b2+4, y-l1-l2);
164:    Line(x+b1, y-l1, x+b1+b2-4, y-l1-l2);
165:    CatchImage(x-b1-b2-b3, y+3,
166:               x+b1+b2+b3, y-l1-l2-l3-2, Im);
167:
168:    SetLineStyle(0, 0, 1);    SetActivePage(0)
169:  END;
170:
171:  (* ------------------------------------------------- *)
172:  (* Setzt den Grafikmodus, zeichnet den Titelbildschirm, *)
173:  (* initialisiert das SpielerObjekt und die Raketen und  *)
174:  (* setzt sie irgendwo auf den Bildschirm in eine        *)
175:  (* Startposition.                                       *)
176:  PROCEDURE InitGame;
177:  BEGIN
178:    Quit := FALSE;  Lifes := 10;  Points := 0;
179:    GraphDriver := Detect;
180:    InitGraph(GraphDriver, GraphMode, '');
181:    Rectangle(0, 0, GetMaxX, GetMaxY);
182:    Rectangle(LSpace, TSpace,
183:              GetMaxX-RSpace, getmaxY-BSpace);
184:    OutTextXY(151, 1, 'S P A C E W A R');
185:    OutTextXY(150, 2, 'S P A C E W A R');
186:    OutTextXY(300, 2, '(c) 1990 R.Reichert & TOOLBOX');
187:    FOR i := 1 TO 2000 DO
188:      PutPixel(LSpace + Random(GetMaxX-LSpace-RSpace),
189:               TSpace + Random(GetMaxY-TSpace-BSpace),
190:               Random(MaxColors));
191:    OutTextXY(100, GetMaxY-12, 'Anzahl Leben: ');
192:    SetWriteMode(XorPut);
193:    Player.Init;
194:    FOR i := 1 TO Lifes DO
195:      WITH Player DO
196:        PutImage(200 + i * (xl + 3),
197:                 GetMaxY-yl-5, Im.Img^, XorPut);
198:    FOR i := 1 TO NrRockets DO BEGIN
199:      New(AllRockets[i], Init);
200:      AllRockets[i]^.SetStartPos
201:        ((GetMaxX-RSpace-LSpace) DIV
202:          NrRockets*(i-1)+LSpace,
203:          Random(GetMaxY-TSpace)+TSpace)
204:    END;
205:  END;
206:
207:  (* ------------------------------------------------- *)
208:  (* Beendet das Spiel, gibt die gemachten Punkte aus  *)
209:  PROCEDURE QuitGame;
210:  VAR
211:    ps : STRING;
212:  BEGIN
213:    SetTextStyle(GothicFont, HorizDir, 5);
214:    SetWriteMode(CopyPut);
215:    Str(points:10, ps);
216:    OutTextXY(50, 80,  'Erreicht: ' + ps + ' Punkte');
217:    OutTextXY(50, 180, 'Ende... ');
218:    OutTextXY(50, 280, 'Noch ein Spiel (J/N): ');
219:    REPEAT
220:      ch := ReadKey;  ch := UpCase(ch);
221:    UNTIL (ch = 'N') OR (ch = 'J');
222:    IF UpCase(ch) = 'N' THEN
223:      Again := FALSE
224:    ELSE
225:      Again := TRUE;
226:  END;
227:
228:  (* ------------------------------------------------- *)
229:  (* Titelbild mit Informationen                       *)
230:  PROCEDURE FirstInit;
231:  BEGIN
232:    GraphDriver := Detect;
233:    InitGraph(GraphDriver, GraphMode, '');
234:    FOR i := 1 TO 5000 DO
```

```
235:      PutPixel(Random(GetMaxX),
236:               Random(GetMaxY), Random(MaxColors));
237:    SetTextStyle(GothicFont, HorizDir, 5);
238:    OutTextXY(50, 40, 'S P A C E W A R');
239:    OutTextXY(30, 90, '--------------------');
240:    SetTextStyle(GothicFont, Horizdir, 1);
241:    OutTextXY(30, 200, 'Gesteuert wird mit Cursortasten'+
242:                       ', geschossen mit Space.');
243:    OutTextXY(30, 220, 'Steuerung ist auch über den '+
244:                       'Zahlenblock möglich, wobei mit '+
245:                       'Home/End');
246:    OutTextXY(30, 240, 'und PgUp/PgDn diagonal'+
247:                       ' gelenkt werden kann.');
248:    OutTextXY(30, 260, 'Dafür muß aber NumLock aktiv sein.'+
249:                       ' Anhalten: "5"');
250:    OutTextXY(30, 280, 'Mit Esc wird das Spiel abgebro'+
251:                       'chen und die Punktzahl angezeigt.');
252:    OutTextXY(30, 320, 'Beginnen mit Enter');
253:    ReadLn;
254:  END;
255:
256:
257:  CONSTRUCTOR OneShot.Init;
258:  BEGIN
259:    DifY := 4;  xl := 3;  yl := 4;  visible := FALSE;
260:    Bottom := GetMaxY - BSpace - yl;
261:    Left   := 0 + LSpace + xl;
262:    Right  := GetMaxX - RSpace - xl;
263:    Top    := 0 + TSpace + yl;
264:    SetActivePage(1);
265:    PutPixel(10,  8, 14);  PutPixel(10, 10, 14);
266:    PutPixel(10, 11, 14);  PutPixel(11, 10, 14);
267:    PutPixel( 9, 10, 14);  PutPixel(10,  9, 14);
268:    PutPixel(10, 12, 14);  PutPixel(12,  9, 14);
269:    PutPixel( 8,  9, 14);
270:    CatchImage(8, 8, 12, 12, Im);
271:    SetActivePage(0);
272:  END;
273:
274:  PROCEDURE OneShot.SetStartPos(nx, ny : INTEGER);
275:  BEGIN
276:    x := nx;   y := ny;  visible := TRUE;
277:    PutImage(x, y, Im.Img^, XorPut);
278:  END;
279:
280:  PROCEDURE OneShot.Move;
281:  BEGIN
282:    Inc(y, DifY);
283:    IF y > Bottom THEN
284:      visible := FALSE
285:    ELSE
286:      PutImage(x, y, Im.Img^, XorPut);
287:    PutImage(x, y-DifY, Im.Img^, XorPut);
288:
289:                          { Prüfen, ob Spieler getroffen: }
290:    IF (x > px1) AND (x < px2) AND
291:                     (y > py1) AND (y < py2) THEN BEGIN
292:      visible := FALSE;
293:      Dec(Lifes);
294:      IF Lifes = 0 THEN Quit := TRUE;
295:      Status;
296:      PutImage(x, y, Im.Img^, XorPut);
297:    END;
298:  END;
299:
300:  CONSTRUCTOR OnePlShot.Init;
301:  BEGIN
302:    DifY := 7;  xl := 3;  yl := 4;  Top := 0 + TSpace+yl;
303:    Visible := FALSE;
304:    SetActivePage(1);
305:    PutPixel(7,10,13);
306:    PutPixel(108,110,13);  PutPixel(110,109,13);
307:    PutPixel(110,110,13);  PutPixel(109,111,13);
308:    PutPixel(111,111,13);  PutPixel(110,113,13);
309:    PutPixel(109,110,13);  PutPixel(111,110,13);
310:    PutPixel(108,112,13);  PutPixel(112,112,13);
311:    CatchImage(108, 107, 112, 113, Im);
312:    SetActivePage(0);
313:  END;
314:
315:  PROCEDURE OnePlShot.Move;
316:  BEGIN
317:    Dec(y, DifY);
318:    IF y < Top THEN
319:      Visible := FALSE
320:    ELSE
321:      PutImage(x, y, Im.Img^, XorPut);
322:    PutImage(x, y+DifY, Im.Img^, XorPut);
323:  END;
324:
325:  CONSTRUCTOR OneRocket.Init;
326:  VAR
327:    i : INTEGER;
328:  BEGIN
329:    HowOften := HOCR;  xl := 22;  yl := 12;
330:    ChangeDir := CDMin + Random(CDCons);
331:    Bottom := GetMaxY - BSpace - yl;
332:    Left   := 0 + LSpace;
333:    Right  := GetMaxX - RSpace - xl;
334:    Top    := 0 + TSpace + yl;
335:    DifY := 3; DifX := 6;
336:    NewShot := NSMin + Random(NSCons);
337:    FOR i := 1 TO MaxShots DO
338:      New(Shots[i], Init);
339:    RocketImage(im);
340:  END;
341:
342:  PROCEDURE OneRocket.Move;
343:  VAR
344:    j, i, OldX, OldY : INTEGER;
345:    Killed           : BOOLEAN;
346:  BEGIN
```



```
347:      Killed := FALSE;
348:      IF HowOften = 0 THEN BEGIN
349:        OldY := y;  Inc(y, DifY);
350:        OldX := x;  Inc(x, DifX);
351:                                  { Irgendeine Grenze verletzt ? }
352:        IF y > Bottom THEN y := Top;
353:        IF x > Right  THEN x := Left;
354:        IF x < Left   THEN x := Right;
355:
356:        IF ChangeDir = 0 THEN BEGIN
357:          ChangeDir := CDMin + Random(CDCons);
358:          DifX := -DifX
359:        END ELSE
360:          Dec(ChangeDir);
361:        PutImage(x, y, Im.Img^, XorPut);
362:        PutImage(Oldx, OldY, Im.Img^, XorPut);
363:                                    { In den Spieler geflogen ? }
364:        IF (x > px1) AND (x < px2) AND (y > py1) AND
365:           (y < py2) OR  (x + xl > px1) AND (x + xl < px2) AND
366:           (y + yl > py1) AND (y + yl < py2) THEN BEGIN
367:          Killed := TRUE;
368:          Dec(lifes);
369:          Status;
370:        END;
371:                                  { Vom Spieler abgeschossen ? }
372:        FOR i := 1 TO PlsMax DO
373:          IF PlShots[i]^.Visible THEN
374:            IF (PlShots[i]^.x > x)    AND
375:               (PlShots[i]^.x < x+xl) AND
376:               (PlShots[i]^.y > y)    AND
377:               (PlShots[i]^.y < y+yl) THEN BEGIN
378:              Killed := TRUE;
379:              Inc(Points, 500);
380:            END;
381:        HowOften := HOCR;
382:
383:        IF (NewShot = 0) THEN BEGIN
384:          i := 0;
385:                  { Welcher Schuß ist noch nicht unterwegs? }
386:          REPEAT
387:            Inc(i);
388:          UNTIL (i = MaxShots) OR (NOT Shots[i]^.visible);
389:          IF i <> MaxShots THEN
390:            Shots[i]^.SetStartPos(x+8, y+15);
391:          NewShot := NSCons + Random(NSMin);
392:        END;
393:        Dec(NewShot)
394:      END ELSE
395:        Dec(HowOften);
396:                                            { Schüsse bewegen: }
397:      FOR i := 1 TO MaxShots DO
398:        IF Shots[i]^.visible THEN Shots[i]^.Move;
399:                                                { Getroffen ? }
400:      IF Killed THEN BEGIN
401:        Visible := FALSE;  PutImage(x, y, Im.Img^, XorPut);
402:        IF Lifes = 0 THEN Quit := TRUE;
403:        FOR i := 1 TO MaxShots DO
404:          IF Shots[i]^.Visible THEN
405:            WITH Shots[i]^ DO BEGIN
406:              PutImage(x, y, Im.Img^, Xorput);
407:              visible := FALSE;
408:            END;
409:      END;
410:    END;
411:
412:    DESTRUCTOR OneRocket.Done;
413:      VAR
414:        i : INTEGER;
415:    BEGIN
416:      FOR i := 1 TO MaxShots DO Dispose(Shots[i]);
417:    END;
418:
419:    CONSTRUCTOR PlayerObj.Init;
420:    CONST
421:      b1 = 10; b2 = 8; b3 =  6; b4 = 2;
422:      l1 =  4; l2 = 2; l3 = 10; l4 = 1;
423:      px = 300; py = 100;
424:    VAR
425:      i : INTEGER;
426:    BEGIN
427:      FOR i := 1 TO PlSMax DO
428:        New(PlShots[i], Init);
429:      DifY     := 2;     Difx     := 4;
430:      DifxCons := DifX;  DifyCons := Dify;
431:      xl       := 20;    yl       := 19;
432:      visible  := FALSE;
433:      Bottom   := GetMaxY - PBSpace;
434:      Left     := 0 + PLSpace;
435:      Right    := GetMaxX - PRSpace;
436:      Top      := 0 + PTSpace;
437:      x := (Right - Left) DIV 2 + Left;
438:      y := Bottom - yl * 2;
439:      px1 := x;       py1 := y;
440:      px2 := x + xl; py2 := y + yl;
441:      SetActivePage(1);
442:      SetWriteMode(CopyPut);
443:      MoveTo(px - b1, py);        LineTo(px - b2, py - l1);
444:      LineTo(px - b3, py - 12);   LineTo(px - b4, py - 13);
445:      LineRel(0, -4);             LineRel(0, 4);
446:      LineTo(px, py-l1);          LineTo(px + b4, py - 13);
447:      LineRel(0, -4);             LineRel(0, 4);
448:      LineTo(px + b3, py - 12);   LineTo(px + b2, py - l1);
449:      LineTo(px + b1, py);        LineTo(px, py + 14);
450:      LineTo(px - b1, py);        LineTo(px, py - l1 + 1);
451:      LineTo(px + b1, py);
452:      CatchImage(px - b1, py - l3 - 8, px + b1, py + 14, Im);
453:      SetActivePage(0);
454:      PutImage(x, y, Im.Img^, XorPut);
455:    END;
456:
457:    PROCEDURE PlayerObj.ReadKeybord;
458:      VAR
```



```
459:       i : INTEGER;
460:   BEGIN
461:     ch := ReadKey;
462:     IF ch = #0 THEN BEGIN
463:       ch := ReadKey;
464:                                              { Cursortasten: }
465:       CASE ch OF
466:         #75 : BEGIN DifX := -DifxCons; DifY := 0  END;
467:         #77 : BEGIN DifX :=  DifxCons; DifY := 0  END;
468:         #72 : BEGIN DifY := -DifyCons; DifX := 0  END;
469:         #80 : BEGIN DifY :=  DifyCons; DifX := 0  END;
470:       END
471:     END ELSE
472:       CASE ch OF
473:                                              { Space: schießen }
474:         #32 : BEGIN
475:                 i := 0;
476:                 REPEAT
477:                   Inc(i);
478:                 UNTIL (NOT PlShots[i]^.Visible) OR
479:                       (i = PlSMax);
480:                 IF i <> PlsMax THEN BEGIN
481:                   Plshots[i]^.SetStartPos(x+8, y);
482:                                        { Geschw. erniedrigen }
483:                   IF i MOD 4 = 0 THEN BEGIN
484:                     Inc(DifYCons);  Inc(DifXCons);
485:                     IF NOT DifX = 0 THEN DifX := DifXCons;
486:                     IF NOT DifY = 0 THEN DifY := DifYCons;
487:                   END;
488:                 END;
489:               END;   { Schießen }
490:         #27 : Quit := TRUE;
491:                                        { Zahlenblock (NUM aktiv) }
492:         #53 : BEGIN DifX := 0;          DifY := 0  END;
493:         #52 : BEGIN DifX := -DifXCons; DifY := 0  END;
494:         #54 : BEGIN DifX :=  DifXCons; DifY := 0  END;
495:         #56 : BEGIN DifY := -DifYCons; DifX := 0  END;
496:         #50 : BEGIN DifY :=  DifYCons; DifX := 0  END;
497:         #55 : BEGIN
498:                 DifX := -DifXCons;  DifY := -DifYCons;
499:               END;
500:         #49 : BEGIN
501:                 DifX := -DifXCons;  DifY := DifYCons;
502:               END;
503:         #57 : BEGIN
504:                 DifX := DifXCons;  DifY := -DifYCons;
505:               END;
506:         #51 : BEGIN
507:                 DifX := DifXCons;  DifY := DifYCons;
508:               END;
509:       END;
510:     Move;
511:   END;
512:
513:   PROCEDURE PlayerObj.Move;
514:   VAR
515:     i : INTEGER;
516:   BEGIN
517:     FOR i := 1 TO PlsMax DO                { Schüsse bewegen }
518:       IF PlShots[i]^.Visible THEN BEGIN
519:         PlShots[i]^.Move;
520:                 { Wenn abgeschossen, dann Geschw. erhöhen }
521:         IF (NOT PlShots[i]^.Visible) AND
522:            (i MOD 4 = 0) THEN BEGIN
523:           Dec(DifYCons);  Dec(DifXCons);
524:           IF NOT Difx = 0 THEN DifX := DifXCons;
525:           IF NOT DifY = 0 THEN DifY := DifYCons;
526:         END;
527:       END;
528:     IF x > Right  - xl THEN DifX := -DifXCons;
529:     IF x < Left        THEN DifX :=  DifXCons;
530:     IF y > Bottom - yl THEN DifY := -DifYCons;
531:     IF y < Top         THEN DifY :=  DifYCons;
532:     PutImage(x + DifX, y + DifY, Im.Img^, XorPut);
533:     PutImage(x, y, Im.Img^, XorPut);
534:     Inc(x, DifX);
535:     Inc(y, DifY);
536:     px1 := x;  py1 := y;  px2 := x + xl;  py2 := y + yl;
537:   END;
538:
539:   DESTRUCTOR PlayerObj.Done;
540:   BEGIN
541:     FOR i := 1 TO PlSMax DO DisPose(PlShots[i]);
542:   END;
543:
544: BEGIN
545:   FirstInit;
546:   REPEAT
547:     Again := TRUE;
548:     InitGame;
549:     REPEAT
550:       FOR i := 1 TO NrRockets DO
551:         IF AllRockets[i]^.visible THEN
552:           AllRockets[i]^.Move
553:         ELSE
554:           AllRockets[i]^.SetStartPos
555:             ((GetMaxX-RSpace-LSpace) DIV
556:                 NrRockets*(i-1)+Lspace,
557:              Random(PTSpace)+TSpace);
558:       IF KeyPressed THEN Player.ReadKeybord
559:                     ELSE Player.Move;
560:     UNTIL Quit;
561:     QuitGame;
562:   UNTIL NOT Again;
563:   FOR i := 1 TO NrRockets DO Dispose(AllRockets[i], Done);
564:   CloseGraph;
565:   ClrScr;
566: END.
567: (* ------------------------------------------------------ *)
568: (*                Ende von SPACEWAR.PAS                   *)
```