

1. Analyse der Problemstellung

1.1. AUFGABENSTELLUNG

Entwickeln Sie ein Programm aus dem Grafik- und Spielebereich (Schwerpunkt Grafikprogrammierung).

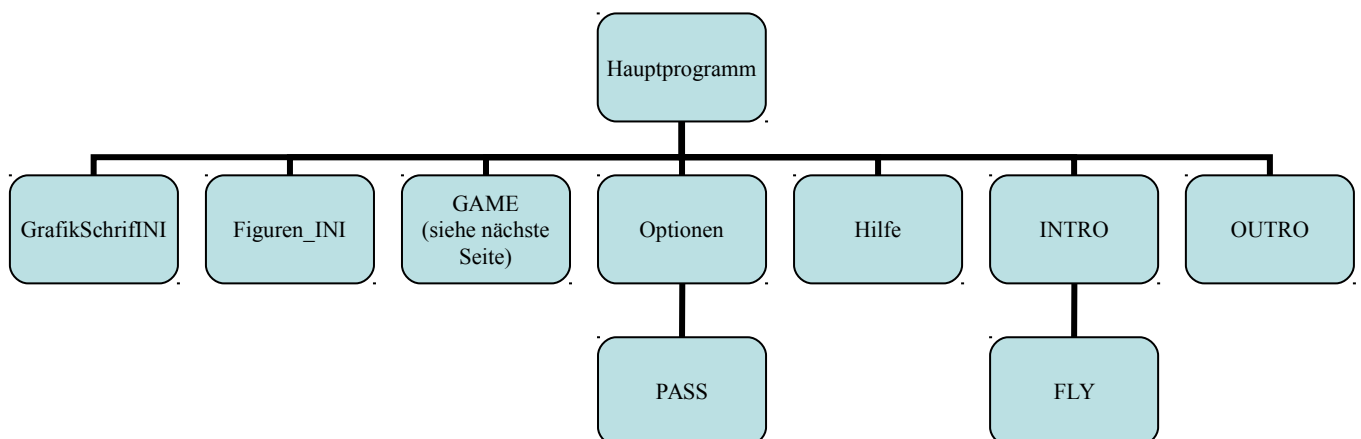
Ich entschied mich für einen Klon, des bekannten Spiels „Digger“. Das Menü sollte mit Maussteuerung sein und das Spiel komplett im Grafikmodus (mit Bildern und Animationen) laufen. Eine Highscore sollte auch dabei sein.

1.2. EINSCHRÄNKUNGEN

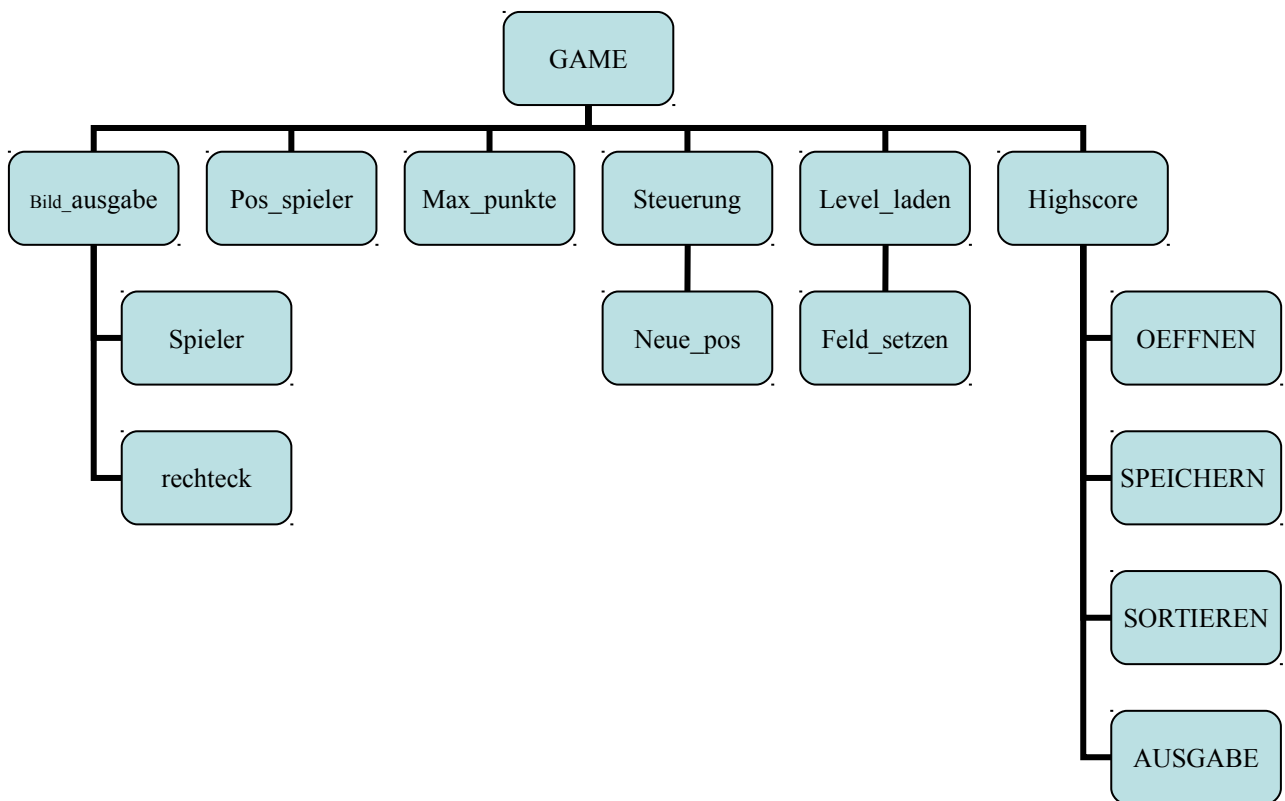
Es gibt im Spiel keinen Sound, weder über die Soundkarte noch über den Systemlautsprecher. Nur 16 Farben werden im Grafikmodus verwendet (das Borland Grafik Interface wurde benutzt). Das Spiel wurde komplett mit Turbo Pascal 7.0 kompiliert, wodurch es keine Probleme mit dem Delay Befehl gibt.

1.3. STRUKTURBAUM

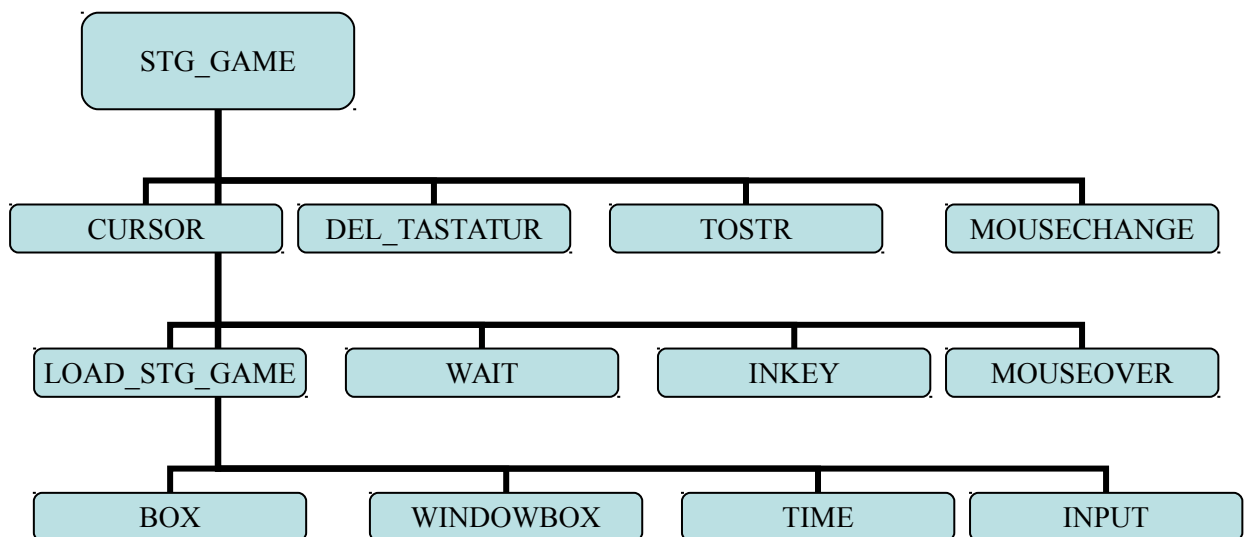
1.3.1. STRUKTURBAUM VON FUTURE.PAS



Strukturbaum der Prozedur GAME



1.3.2. STRUKTURBAUM VON STG_GAME.PAS



2. Prozeduren und Funktionen

2.1. PROZEDUREN UND FUNKTIONEN VON FUTURE.PAS

PROCEDURE GrafikSchriftINI;

für die Grafik- (Treiber) und Schriftinitialisierung

PROCEDURE FIGUREN_INI (VAR Pfiguren:figuren_);

für die Grafikinitialisierung (von den Grafikdateien), das bedeutet alle verwendeten Grafiken werden geladen (mit LOAD_STG_GRAFIK) und in „Pfiguren“ abgespeichert.

PROCEDURE GAME(figuren:figuren_;start_level:byte);*{Kern des Spieles}*

„Figuren“ beinhaltet alle Grafiken (Grafiken als Pointer in dem Feld FIGUREN)

„Start_level“ gibt das Startlevel an, welches im Optionsmenü geändert werden kann

{lokale Prozeduren von GAME }

PROCEDURE bild_ausgabe(eingabe:char;

VAR feld:spiel_feld;figuren:figuren_);

Ausgabe des Spielfeldes, Übergabe des geladenen Spielfeldes in „Feld“ , „Figuren“ und „eingabe“ für die Animation bei der Ausgabe

{lokale Prozeduren von bild_ausgabe }

PROCEDURE Spieler(x,y,posx,posy:integer;

feld:spiel_feld;figuren:figuren_);

Ausgabe der Spielfigur, „x“ und „y“, geben die Bildschirmkoordinaten an, „Posx“ und „posy“ die Koordinaten im Spielfeld welche für die Animation mit den Grafiken im Pointerfeld „Figuren“ benötigt werden

PROCEDURE rechteck(x1,y1,x2,y2,farbe:word);

Zeichnet ein Rechteck mit den Koordinaten „x1“/„y1“ „x2“/„y2“ wie Rectangle, aber gefüllt mit einer übergebenden „Farbe“

{lokale Prozeduren von bild_ausgabe ENDE }

PROCEDURE POS_SPIELER(feld:spiel_feld;VAR x_pos,y_pos:byte);

Ermitteln der Position des Spielers (der Zahl 2) im Spielfeld („FELD“), die Position des Spielers wird dann in „x_pos“, „y_pos“ abgespeichert.

PROCEDURE MAX_PUNKTE(FELD:SPIEL_FELD;VAR MAX_P:INTEGER);

Ermitteln der Maximalen Punkte im Level.

Suchen der 3en im Spielfeld („FELD“) und abspeichern der Punkte in „MAX_P“.

PROCEDURE Steuerung(eingabe:char;x_pos,y_pos:byte;MAX_P:integer;VAR

feld:spiel_feld; VAR punkte:word;VAR beenden:Boolean);

Auswerten der Eingabe („eingabe“), Überprüfung ob eine Bewegung möglich ist (dazu wird die Position des Spielers „x_pos“/„y_pos“ und das Spielfeld „FELD“ benötigt), wenn sich eine 3 auf der neuen Position befindet werden die „Punkte“ erhöht, und „beenden“ gibt an ob alle Punkte („MAX_P-1“) eingesammelt wurden und sich die Spielfigur über dem EXIT befindet

{lokale Prozeduren von Steuerung }

PROCEDURE neue_pos(x_alt,y_alt:byte;x,y:byte;VAR feld:spiel_feld);

Alte Position („x_alt“/„y_alt“) wird gelöscht und Neue („x“/ „y“) wird im Spielfeld („FELD“) gesetzt.

{lokale Prozeduren von Steuerung ENDE }

```

PROCEDURE Level_LADEN(level_name:string;VAR feld:spiel_feld);
Ein Level mit dem Dateinamen „LEVEL_NAME“ wird geladen und Zeilenweise
in das Spielfeld abgespeichert („FELD“)
{lokale Prozeduren von LEVEL_LADEN }
  PROCEDURE feld_setzen(inhalt:string;zeilen_nr:byte;VAR
                                feld:spiel_feld);
  Eine Zeile („INHALT“), welche Zahlen als String beinhaltet, wird in der
  Zeile („zeilen_nr“) in das Spielfeld einzeln (Spaltenweise)
  abgespeichert.
{lokale Prozeduren von LEVEL_LADEN ENDE }

PROCEDURE HIGHSCORE(zeit,punkte:longint);
„Zeit“ und „Punkte“ , welche im Spiel erreicht wurden
{lokale Prozeduren von HIGHSCORE }
  PROCEDURE OEFFNEN(VAR SCORE:HSCORE);
  High Score wird geöffnet und eingelesen in „SCORE“

  PROCEDURE SORTIEREN(VAR SCORE:HSCORE);
  High Score wird sortiert, und in „SCORE“ abgespeichert

  PROCEDURE SPEICHERN(SCORE:HSCORE);
  High Score Datei wird abgespeichert, das FELD „SCORE“ wird einzeln in
  eine die High- Scoredatei (high.dat)geschrieben.

  PROCEDURE AUSGABE(SCORE:HSCORE);
  Ausgabe der High Score („SCORE“) auf den Bildschirm
{lokale Prozeduren von HIGHSCORE ENDE }

PROCEDURE Optionen(VAR start_level:byte);
Optionsmenü, bei dem man das „Start_level“ ändern kann
{lokal Funktion von Optionen }
  FUNCTION PASS(passwort:string):boolean;
  Eine Passwortüberprüfungs- und Ausgabefunktion
{lokal Funktion von Optionen ENDE }

PROCEDURE Hilfe;
Hilfebildschirm

PROCEDURE INTRO(FIGUREN:FIGUREN_);
Figuren für das INTRO in „FIGUREN“

PROCEDURE OUTRO;
Ende des Programms mit der Ausgabe der BETATESTER

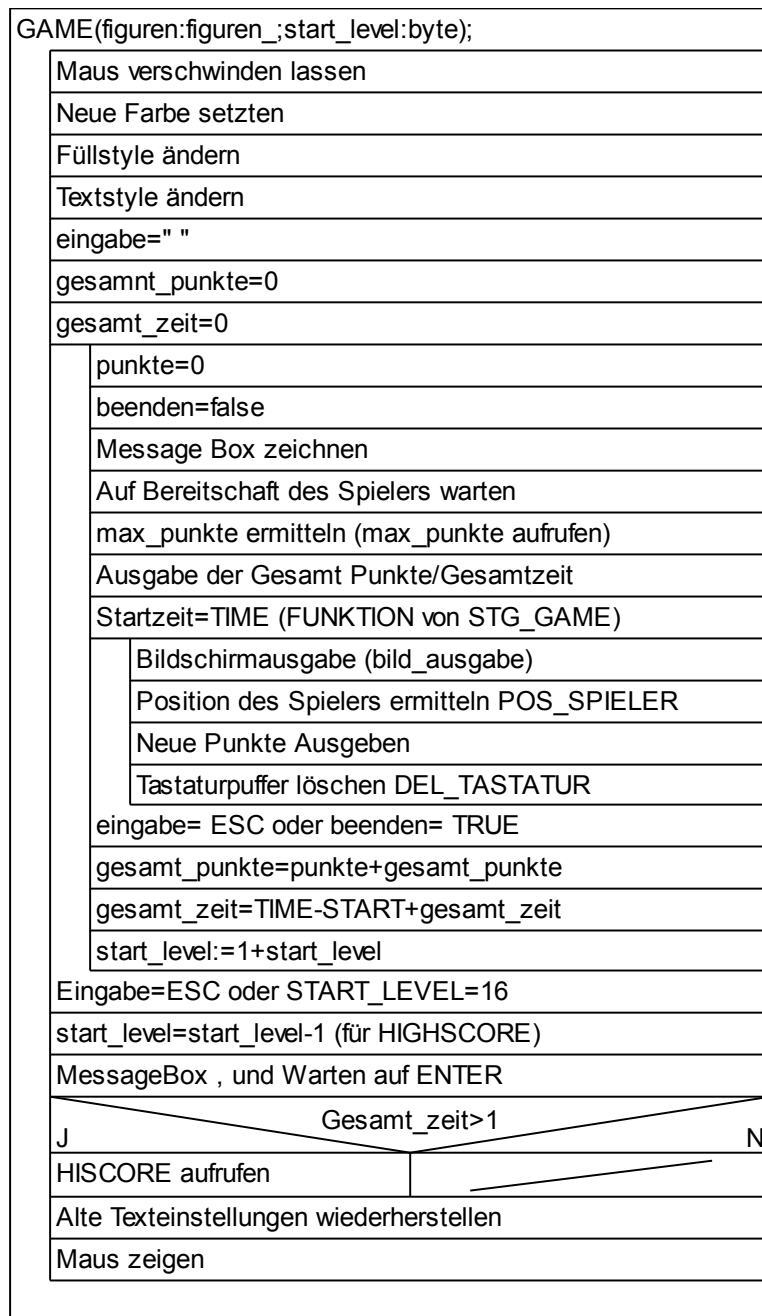
```

2.2. PROZEDUREN UND FUNKTIONEN VON STG_GAME.PAS

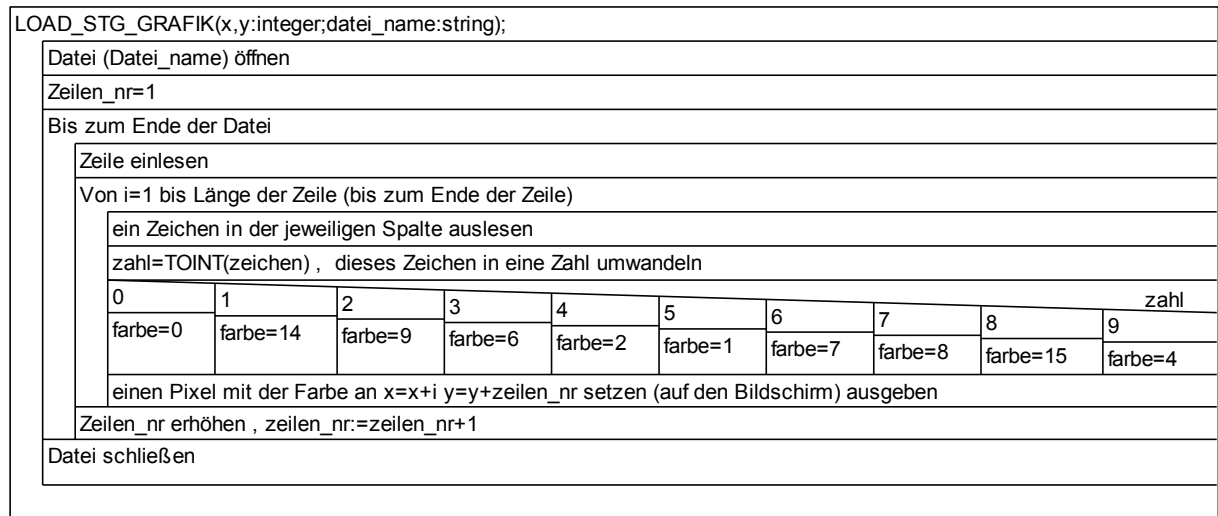
```
PROCEDURE cursor(an_aus:boolean);  
{Cursor an- oder aus- schalten im Textmodus  
Wenn an_aus TRUE ist dann ist der Cursor AN}  
  
PROCEDURE DEL_TASTATUR;  
{Lösche den Tastaturspeicher}  
  
PROCEDURE WAIT(TASTE:CHAR);  
{Wartet auf eine bestimmte Taste}  
  
PROCEDURE LOAD_STG_GRAFIK(x,y:integer;datei_name:string);  
{STG GRAFIK LADEN(mit dem NAMEN datei_name) und an die Koordinaten x/y  
ausgeben }  
  
FUNCTION ToStr(Zahl: Longint): String;  
{Zahl in String umwandeln}  
  
FUNCTION INKEY:char;  
{Zeichen ohne Warten einlesen}  
  
PROCEDURE MOUSECHANGE;  
{Wartet auf das Bewegen der Maus}  
  
FUNCTION MOUSEOVER(x1,y1,x2,y2:word):boolean;  
{überprüft ob die Maus sich in einem bestimmten Rechteck(x1,y1,x2,y2)  
befindet und gibt dann WAHR zurück }  
  
PROCEDURE BOX(zeile1,zeile2:string);  
{zeichnet eine zentrierte MessageBox  
mit dem Text „ZEILE1/ZEILE2“ }  
  
PROCEDURE WINDOWBOX(x,y,x1,y1:word;zeile_oben:string);  
{großes Fenster (x/y und x1/y1) mit Überschrift(zeile_oben) }  
  
FUNCTION TIME:longint;  
{gibt Zeit in Sekunden zurück}  
  
PROCEDURE INPUT(VAR text:string);  
{readln Ersatz für Grafikmodus (mit Outtext)  
speichert die Eingabe in TEXT}
```

3. Struktogramme

3.1. Struktogramm von GAME



3.2. Struktogramm von LOAD_STG_GRAFIK



4. Quelltexte

4.1. QUELLTEXT VON FUTURE.PAS

```
{.....}
(*
  FUTURE.pas bzw. die UNIT STG_GAME.pas
  muessen mit TP 7.0 kompiliert werden
*)
{.....}
(*
  FUTURE_DIGGER wurde von Steve Göring für eine
  Projektarbeit im Unterrichtsfach Datenverarbeitungstechnik
  in der 11.Klasse geschrieben. Copyright by Steve Göring (STG7) 2006.
  KONTAKT:      stg7@gmx.de
*)
{.....}

PROGRAM FUTURE_DIGGER;

USES CRT, MOUSE, GRAPH, STG_GAME, BGIDriv, BGIFont; {BGI** für die INTERNEN Grafik- und
Schrift-Treiber}

TYPE figuren_=ARRAY [1..11] OF POINTER;
TYPE menu_=ARRAY [1..4] OF STRING;

VAR eingabe:char;
    figuren:figuren_;
    i,auswahl:integer;
    start_level:byte;
    menu:menu_;
    taste,x,y:word;
    Exists : Boolean;
    Button : Word;
    x_,y_:word;

CONST X_ECKE=20;      {Koordinaten der Rechten oberen Ecke des Spiel-Feldes}
      Y_ECKE=50;
      LINKS='a';
      RECHTS='d';
      HOCH='w';
      RUNTER='s';
```

```

PROCEDURE GrafikSchriftINI;
VAR Gd, Gm : Integer; { ben"tigten Variablen fr den Grafikmodus}
BEGIN
  {Alle Register Befehle bewirken das laden der intern gelinkte Grafiktreiber}
  {GrafikTreiber}
  RegisterBGIDriver(@CGADriverProc);
  RegisterBGIDriver(@EGAVGADriverProc);
  RegisterBGIDriver(@HercDriverProc);
  RegisterBGIDriver(@ATTDriverProc);
  RegisterBGIDriver(@PC3270DriverProc);
  {SchriftArten}
  RegisterBGIfont(@GothicFontProc);
  RegisterBGIfont(@SansSerifFontProc);
  RegisterBGIfont(@SmallFontProc);
  RegisterBGIfont(@TriplexFontProc);
  {Initialisierung der Grafik}
  Gd := Detect;
  InitGraph(Gd, Gm, '');
  IF GraphResult <> grOk
    THEN
      BEGIN
        writeln('Fehler bei der Initialisierung der Grafik');
        readln;
        halt(1); {beenden bei falscher Initialisierung}
      END;
  CLEARDEVICE;
  SETBKCOLOR(0);
  SETCOLOR(10);
  directvideo:=false; {erm"glicht write, usw. im Grafikmodus}
  TextColor(10);
END;
{.....}
PROCEDURE FIGUREN_INI(VAR Pfiguren:figuren_);
VAR size:integer;
    i:integer;
BEGIN
  LOAD_STG_GRAFIK(99,100,'grafik\spieler.dat');
  LOAD_STG_GRAFIK(130,100,'grafik\n_r.dat');
  LOAD_STG_GRAFIK(161,100,'grafik\n_l.dat');
  LOAD_STG_GRAFIK(192,100,'grafik\r.dat');
  LOAD_STG_GRAFIK(223,100,'grafik\l.dat');
  LOAD_STG_GRAFIK(254,100,'grafik\muster.dat');
  LOAD_STG_GRAFIK(315,100,'grafik\mauer.dat');
  LOAD_STG_GRAFIK(346,100,'grafik\punkt.dat');

  SETCOLOR(8);
  SETFILLSTYLE(1,8);
  FILLELLIPSE(301,115,14,14); {der runde Stein}

  LOAD_STG_GRAFIK(99,150,'grafik\schrift\f.dat');
  LOAD_STG_GRAFIK(130,150,'grafik\schrift\u.dat');
  LOAD_STG_GRAFIK(161,150,'grafik\schrift\t.dat');
  LOAD_STG_GRAFIK(192,150,'grafik\schrift\u.dat');
  LOAD_STG_GRAFIK(223,150,'grafik\schrift\r.dat');
  LOAD_STG_GRAFIK(254,150,'grafik\schrift\e.dat');

  LOAD_STG_GRAFIK(99,200,'grafik\schrift\d.dat');
  LOAD_STG_GRAFIK(130,200,'grafik\schrift\i.dat');
  LOAD_STG_GRAFIK(161,200,'grafik\schrift\g.dat');
  LOAD_STG_GRAFIK(192,200,'grafik\schrift\g.dat');
  LOAD_STG_GRAFIK(223,200,'grafik\schrift\e.dat');
  LOAD_STG_GRAFIK(254,200,'grafik\schrift\r.dat');

  FOR i:=0 TO 6 DO {Spielefiguren und Muster}
  BEGIN
    Size := ImageSize(100+i*30, 100, 130+i*30, 130);
    GetMem(Pfiguren[i+1], Size);
    GetImage(100+i*30+i, 100, 129+i*30+i,129,Pfiguren[i+1] ^);
  END;

```



```

Size := ImageSize(316, 100, 346, 130);
GetMem(Pfiguren[10], Size);
GetImage(316, 101, 345,130,Pfiguren[10] ^);

Size := ImageSize(346, 100, 376, 130);
GetMem(pfiguren[11], Size);
GetImage(347, 101, 376,130,pfiguren[11] ^);

{Schriftzge FUTURE und DIGGER}
{FUTURE}
SIZE:=IMAGESIZE(100,150,280,180);
GETMEM(PFIGUREN[8],Size);
GETIMAGE(100,150,280,180,PFIGUREN[8]^);

{DIGGER}
SIZE:=IMAGESIZE(100,200,280,230);
GETMEM(PFIGUREN[9],Size);
GETIMAGE(100,200,280,230,PFIGUREN[9]^);

{WAIT(ENTER);}
CLEARDEVICE;
END;
{.....}
PROCEDURE GAME(figuren:figuren_;start_level:byte);{Kern des Spieles}
TYPE spiel_feld= ARRAY [1..18,1.. 9] OF INTEGER;
                {[ x , y ] }
VAR feld:spiel_feld;
    eingabe:char;
    x_pos,y_pos:byte;
    gesamt_punkte, punkte:word;
    Beenden:BOOLEAN;
    MAX_P:integer;
    START,GESAMT_ZEIT:LONGINT;{ZeitVARIABLEN}
{-----}
{Lokale Unterprozeduren von GAME mit ---- gekennzeichnet}
PROCEDURE bild_ausgabe(eingabe:char;VAR feld:spiel_feld;figuren:figuren_);
VAR x,y,z,k,i,nn:integer;
    farbe:integer;
    old:integer;
{Unterprozedure von Bild_ausgabe}
PROCEDURE Spieler(x,y,posx,posy:integer;feld:spiel_feld;figuren:figuren_);
BEGIN
    IF eingabe=RECHTS
    THEN
        IF feld[posx+1,posy]=5
        THEN PutIMAGE(x,y,figuren[4]^,0)
        ELSE PUTIMAGE(x,y,figuren[2]^,0)
        ELSE
        IF eingabe=LINKS
        THEN
            IF feld[posx-1,posy]=5
            THEN PutIMAGE(x,y,figuren[5]^,0)
            ELSE PutIMAGE(x,y,figuren[3]^,0)
            ELSE PUTIMAGE(x,y,figuren[1]^,0);
END;
{Unterprozedure von Bild_ausgabe}
PROCEDURE rechteck(x1,y1,x2,y2,farbe:word);
VAR i:integer;
BEGIN
    SETCOLOR(farbe);
    SETFILLSTYLE(1,farbe);
    BAR (x1, y1, x2, y2); {BAR zeichnet ein geflltes Rechteck}
END;
BEGIN
    FOR k:=1 TO 9 DO
        BEGIN
            y:=30*k+Y_Ecke;
            FOR z:=1 TO 18 DO
                BEGIN

```

```

x:=30*z+X_Ecke;
IF ((feld[z,k]=3) OR (feld[z,k]=5)) AND (feld[z,k+1]=0)
THEN
    BEGIN
        old:=feld[z,k];
        feld[z,k]:=0;
        feld[z,k+1]:=old;
    END;
IF ((feld[z,k]=3 ) OR (feld[z,k]=5 )) AND
((feld[z,k+1]=3 ) OR (feld[z,k+1]=5 )) AND
((feld[z+1,k+1]=0) OR (feld[z-1,k+1]=0)) AND
((feld[z+1,k]=0 ) OR (feld[z-1,k]=0 ))
THEN
    BEGIN
        old:=feld[z,k];
        IF feld[z+1,k+1]=0
        THEN feld[z+1,k+1]:=old
        ELSE
            IF feld[z-1,k+1]=0
            THEN feld[z-1,k+1]:=old;
            feld[z,k]:=0;
        END;
    CASE feld[z,k] OF
        0: rechteck(x+1,y+1,x+30,y+30,0 );      {leeres Feld}
        1: PUTIMAGE(x+1,y+1,figuren[10]^,0);    {Mauer}
        2: spieler(x+1,y+1,z,k,feld,figuren);  {Spieler}
        3: PUTIMAGE(x+1,y+1,figuren[11]^,0); {rechteck(x+1,y+1,x+30,y+30,14);}
        {Punktefeld}
        4: BEGIN
            rechteck(x+1,y+1,x+30,y+30,12);
            SETCOLOR(0);
            OUTTEXTXY(x+13,y+13,'E');
            END;
        5: PUTIMAGE(x+1,y+1,figuren[7]^,0);    {Stein}
    END;
END;
END;
END;
END;
{-----}
PROCEDURE POS_SPIELER(feld:spiel_feld;VAR x_pos,y_pos:byte);
{Ermitteln der Position des Spielers(der Zahl 2) im Spielfeld}
VAR i,z:integer;
BEGIN
    FOR i:=1 TO 9 DO
        FOR z:=1 TO 18 DO
            IF feld[z,i]=2
            THEN
                BEGIN
                    x_pos:=z;
                    y_pos:=i;
                END;
            END;
        END;
    END;
PROCEDURE MAX_PUNKTE(FELD:SPIEL_FELD;VAR MAX_P:INTEGER);
{Ermitteln der Maximalen Punkte im Level ( der 3en im Spielfeld)}
VAR i,z:integer;
BEGIN
    max_p:=0;
    FOR i:=1 TO 9 DO
        FOR z:=1 TO 18 DO
            IF feld[z,i]=3
            THEN max_p:=max_p+1;
        END;
    END;
{-----}
PROCEDURE Steuerung(eingabe:char;x_pos,y_pos:byte;MAX_P:integer;VAR
feld:spiel_feld; VAR punkte:word;VAR beenden:Boolean);
PROCEDURE neue_pos(x_alt,y_alt:byte;x,y:byte;VAR feld:spiel_feld);
BEGIN
    feld[x_alt,y_alt]:=0; {alte Position l"schen}
    feld[x,y]:=2;         {neue Position setzen}
END;

```

```

BEGIN {Steuerung}
CASE eingabe OF
HOCH:
    IF NOT ((feld[x_pos,y_pos-1]=1) OR (feld[x_pos,y_pos-1]=4) OR
(feld[x_pos,y_pos-1]=5))
    THEN
        BEGIN
            IF feld[x_pos,y_pos-1]=3
            THEN punkte:=punkte+1;
            neue_pos(x_pos,y_pos,x_pos,y_pos-1,feld);
        END
        ELSE beenden:=(feld[x_pos,y_pos-1]=4) AND (Punkte>=Max_p-1);
RUNTER:
    IF NOT ((feld[x_pos,y_pos+1]=1) OR (feld[x_pos,y_pos+1]=4) OR
(feld[x_pos,y_pos+1]=5) )
    THEN
        BEGIN
            IF feld[x_pos,y_pos+1]=3
            THEN punkte:=punkte+1;
            neue_pos(x_pos,y_pos,x_pos ,y_pos+1,feld);
        END
        ELSE beenden:=(feld[x_pos,y_pos+1]=4) AND (Punkte>=Max_p-1);
LINKS:
    IF NOT((feld[x_pos-1,y_pos]=1) OR (feld[x_pos-1,y_pos]=4))
    THEN
        BEGIN
            IF feld[x_pos-1,y_pos]=3
            THEN punkte:=punkte+1;
            IF NOT(feld[x_pos-1,y_pos]=5)
            THEN neue_pos(x_pos,y_pos,x_pos-1,y_pos,feld)
            ELSE
                IF NOT((feld[x_pos-2,y_pos]=1) OR(feld[x_pos-2,y_pos]>=3))
                THEN
                    {Stein verschieben}
                    BEGIN
                        neue_pos(x_pos,y_pos,x_pos-1,y_pos,feld);
                        feld[x_pos-2,y_pos ]:=5;{Stein verschieben}
                    END;
                END
            ELSE beenden:=(feld[x_pos-1,y_pos]=4) AND (PUNKTE>=Max_p-1);
RECHTS:
    IF NOT((feld[x_pos+1,y_pos]=1) OR (feld[x_pos+1,y_pos]=4))
    THEN
        BEGIN
            IF feld[x_pos+1,y_pos]=3
            THEN punkte:=punkte+1;
            IF NOT(feld[x_pos+1,y_pos]=5)
            THEN neue_pos(x_pos,y_pos,x_pos+1,y_pos,feld)
            ELSE
                IF NOT((feld[x_pos+2,y_pos]=1) OR(feld[x_pos+2,y_pos]>=3))
                THEN
                    {Stein verschieben}
                    BEGIN
                        neue_pos(x_pos,y_pos,x_pos+1,y_pos,feld);
                        feld[x_pos+2,y_pos ]:=5;{Stein verschieben}
                    END;
                END
            ELSE beenden:=(feld[x_pos+1,y_pos]=4) AND (Punkte>=Max_P-1);
        END;
    END;
}
-----}
PROCEDURE Level_LADEN(level_name:string;VAR feld:spiel_feld);
VAR Datei:Text;
    zeile:string;
    zeilen_nr:byte;
PROCEDURE feld_setzen(inhalt:string;zeilen_nr:byte;VAR feld:spiel_feld);
VAR z:integer;
    spalte:byte;
    fehler:integer;
BEGIN
    FOR z:=1 TO length(inhalt) DO
        BEGIN

```

```

        VAL(copy(inhalt,z,1),spalte,fehler);
        feld[z,zeilen_nr]:=spalte;
    END;
END;
BEGIN
    { in einer Level-Datei bedeutet
    1= Mauer
    2= Spieler
    3= Punktefeld
    4= EXIT-Tor
    5= Stein
    }
    Assign(Datei, level_name);
    ReSET(Datei);
    zeilen_nr:=1;
    WHILE NOT (EOF(Datei)) DO
        BEGIN
            READLN (DATEI,zeile);
            feld_setzen(zeile,zeilen_nr,feld);
            zeilen_nr:=zeilen_nr+1;
        END;
    Close(Datei);
END;
{-----}
PROCEDURE HIGHSCORE(zeit,punkte:longint);
TYPE
    SCORE_ =
    RECORD
        name:string;
        punkte:longint;
        zeit:longint;
    END;
    HSCORE= ARRAY [1..10] OF SCORE_;

VAR NAME:STRING;
    frage:char;
    SCORE:HSCORE;
    old:word;
CONST SCORE_DATEI='high.dat';

PROCEDURE OEFFNEN (VAR SCORE:HSCORE);
VAR i:integer;
    DATEI: FILE OF SCORE_;
BEGIN
    ASSIGN (DATEI,SCORE_DATEI);
    RESET (DATEI);
    FOR i:=1 TO 10 DO
        read (DATEI,score[i]);
    CLOSE (DATEI);
END;

PROCEDURE SORTIEREN (VAR SCORE:HSCORE);
VAR i:integer;
    hilf:score_;
BEGIN
    FOR i:=10 DOWNTO 2 DO
        IF (score[i].punkte)>(score[i-1].punkte)
        THEN
            BEGIN
                hilf:=score[i];
                score[i]:=score[i-1];
                score[i-1]:=hilf;
            END;
    END;
END;

PROCEDURE SPEICHERN (SCORE:HSCORE);
VAR i:integer;
    DATEI: FILE OF SCORE_;
BEGIN
    Assign (DATEI, SCORE_DATEI);
    Rewrite (DATEI);
    FOR i:=1 TO 10 DO

```

```

    Write (DATEI, score[i]);
    Close (DATEI);
END;
PROCEDURE AUSGABE (SCORE:HSCORE);
VAR i:integer;
    x,y:integer;
BEGIN
    x:=GETx;
    y:=gety;
    RECTANGLE (x-3,y-3,x+26*8+3,y+8*12+3);
    OUTTEXTxy(x,y,'NR.      NAME      ZEIT Punkte');
    y:=y+1*8;
    FOR i:=1 TO 10 DO
    BEGIN
        OUTTEXTxy(x+(2-LENGTH(TOSTR(i)))*8,y+i*8,TOSTR(i)+'.');
        OUTTEXTxy(x+(12-LENGTH(score[i].name))*8,y+i*8,score[i].name);
        OUTTEXTxy(x+(19-LENGTH(TOSTR(score[i].zeit))*8,y+i*8,TOSTR(score[i].zeit));
        OUTTEXTxy(x+(26-LENGTH(TOSTR(score[i].punkte))*8,y+i*8,TOSTR(score[i].punkte));
    END;
END;
BEGIN
    old:=GETCOLOR;
    SETCOLOR(15);

    BAR(x_ECKE+30,Y_ECKE+30,X_ECKE+570,Y_ECKE+300);
    WINDOWBOX(X_ECKE+60,Y_ECKE+60,X_ECKE+540,Y_ECKE+270,'HIGHSCORE ');

    MOVETO(15*8,18*8);
    OUTTEXT('In die Highscore eintragen ? (j/n) ' );
    frage:=readkey;
    OUTTEXT(frage);
    frage:=UPCASE(FRAGE);
    OEFFNEN(SCORE);
    MOVETO(15*8,19*8);
    IF FRAGE='J'
    THEN
    BEGIN
        OUTTEXT('NAME eingeben : ');
        INPUT(name);
        score[10].name:=name;
        score[10].punkte:=punkte;
        score[10].zeit:=zeit;
        Sortieren(score);
        Speichern(score);
    END;
    MOVETO(15*8,21*8);
    Ausgabe(score);
    OUTTEXTxy(15*8,35*8,'weiter mit ENTER');
    WAIT(ENTER);
    SETCOLOR(old);
END;
{-----}
BEGIN {Prozedure GAME}
    MOUSEHIDE;
    SETCOLOR(10);
    SETFILLStyle(1,0);
    SETTEXTSTYLE(DefaultFont,HorizDir,1);
    eingabe:=' ';
    gesamt_punkte:=0;
    gesamt_zeit:=0;

    REPEAT
        punkte:=0;
        beenden:=FALSE;
        BAR(280,10,390,40);
        OUTTEXTxy(280,10,'LEVEL '+TOSTR(start_level));
        level_laden('level\level'+TOSTR(start_level)+'.dat',feld);
        BOX('Bereit fr Level '+TOSTR(start_level)+' ?','weiter mit ENTER');
        WAIT(ENTER);
        max_punkte(feld,max_p);

```

```

BAR(390,10,640,40);
OUTTEXTXY(400,10,'Maximale Punkte :'+TOSTR(max_p));
OUTTEXTXY(400,20,'Gesamte Punkte :'+TOSTR(gesamt_punkte));
OUTTEXTXY(400,30,'Gesamte Zeit :'+TOSTR(gesamt_zeit));

START:=TIME;
REPEAT
  bild_ausgabe(eingabe,feld,figuren);
  pos_spieler(feld,x_pos,y_pos);
  eingabe:=inkey;
  steuerung(eingabe,x_pos,y_pos,max_p,feld,punkte,beenden);
  SETCOLOR(0);
  SETFILLSTYLE(1,0);
  BAR(48,8,150,30); {alte Punkte/Zeit vom Bildschirm "l"schen" }
  SETCOLOR(10);
  OUTTEXTXY(50,10,'PUNKTE: '+TOSTR(PUNKTE));
  OUTTEXTXY(50,20,'ZEIT: '+TOSTR(TIME-START));
  DEL_TASTATUR;
UNTIL (eingabe=ESC) OR (beenden) ;
gesamt_punkte:=punkte+gesamt_punkte;
gesamt_zeit:=TIME-START+gesamt_zeit;
start_level:=1+start_level;
BAR(390,10,640,40);
OUTTEXTXY(400,20,'Gesamte Punkte :'+TOSTR(gesamt_punkte));
OUTTEXTXY(400,30,'Gesamte Zeit :'+TOSTR(gesamt_zeit));
UNTIL (eingabe=ESC) OR (start_level=16);
start_level:=start_level-1;
IF beenden
  THEN BOX('Spiel wurde durchgespielt!','weiter mit ENTER')
  ELSE BOX('ENDE','weiter mit ENTER');
WAIT(ENTER);
IF GESAMT_ZEIT>1
  THEN HIGHSCORE(gesamt_zeit,gesamt_punkte);
SETTEXTSTYLE(1,HorizDir,1);
SetUserCharSize(2, 2, 2, 2);
MOUSESHOW;
END;
{.....}
PROCEDURE Optionen(VAR start_level:byte);
VAR taste,x,y : Word;
    eingabe:char;
FUNCTION PASS(passwort:string):boolean;
VAR eingabe:string;
    taste:char;
    pass_:boolean;
BEGIN
  eingabe:=' ';
  GOTOXY(20,12);
  REPEAT
    taste:=readkey;
    write('*');
    eingabe:=eingabe+taste;
    pass_:= eingabe=' '+passwort;
  UNTIL pass_ OR (taste=ESC);
  pass:=pass_;
END;
BEGIN
  SETTEXTSTYLE(0,HorizDir,1);
  SetUserCharSize(1, 1, 1, 1);

  WINDOWBOX(X_ECKE+60,Y_ECKE+60,X_ECKE+540,Y_ECKE+270,'OPTIONEN (mit ESC zurck)');
  SETCOLOR(15);
  OUTTEXTXY(X_ECKE+100,Y_ECKE+100,'START LEVEL w,hlen: ');
  SETCOLOR(12);
  OUTTEXTXY(X_ECKE+300,Y_ECKE+100,'TASTE [L]');
  SETFILLSTYLE(1,12);
  REPEAT
    eingabe:=UPCASE(readkey);
    IF eingabe='L'
      THEN

```

```

BEGIN
  SETFILLSTYLE(1,1);
  BOX('PASSWORT EINGEBEN ','(ESC=ENDE)');
  IF PASS('FUTURESTG')
  THEN
    BEGIN
      eingabe:=' ';
      BOX('Startlevel einstellen','[W]=+,[S]=-');
      WHILE NOT(eingabe=ESC) DO
        BEGIN
          eingabe:=UPCASE(readkey);
          IF eingabe='W'
            THEN start_level:=start_level+1
            ELSE
              IF eingabe='S'
                THEN start_level:=start_level-1;
              IF start_level<1
                THEN start_level:=1
                ELSE
                  IF start_level>15
                    THEN start_level:=15;
          BAR(x_ecke+300,y_ecke+100,x_ecke+340,y_ecke+110);
          OUTTEXTxy(x_ecke+300,y_ecke+100,TOSTR(start_level));
        END;
        eingabe:=' ';
      END;
      BAR(198,198,422,252);
    END;
  UNTIL eingabe=ESC;
  SETTEXTSTYLE(1,HorizDir,1);
  SetUserCharSize(2, 2, 2, 2);
END;
{.....}
PROCEDURE Hilfe;
BEGIN
  SETTEXTSTYLE(0,HorizDir,1);
  SetUserCharSize(1, 1, 1, 1);
  WINDOWBOX(X_ECKE+60,Y_ECKE+60,X_ECKE+540,Y_ECKE+270,'HILFE (mit ESC zurck)');
  SETCOLOR(15);
  OUTTEXTxy(x_ecke+60+16,y_ecke+60+3*8, 'Gesteuert wird FUTURE DIGGER MIT');
  OUTTEXTxy(x_ecke+60+16,y_ecke+60+5*8, ' [W]=HOCH ');
  OUTTEXTxy(x_ecke+60+16,y_ecke+60+7*8, ' [A]=LINKS [S]=RUNTER [D]=RECHTS');
  OUTTEXTxy(x_ecke+60+16,y_ecke+60+9*8, 'Ziel ist es alle Gelben Steine
  einzusammeln');
  OUTTEXTxy(x_ecke+60+16,y_ecke+60+11*8,'und dann das EXIT (E) zu erreichen!');
  OUTTEXTxy(x_ecke+60+16,y_ecke+60+13*8,'Die grauen Kugeln sind dabei oft zu bewegen
  oder ');
  OUTTEXTxy(x_ecke+60+16,y_ecke+60+15*8,'ihre Spielfigur wird durch 4 Kugeln
  eingeschlossen und ');
  OUTTEXTxy(x_ecke+60+16,y_ecke+60+17*8,'das Spiel ist dann zuende!');
  OUTTEXTxy(x_ecke+60+16,y_ecke+60+20*8, ' VIEL SPAá BEIM SPIELEN .....');
  WAIT(ESC);
  SETTEXTSTYLE(1,HorizDir,1);
  SetUserCharSize(2, 2, 2, 2);
END;
{.....}
PROCEDURE INTRO(FIGURE:FIGUREN_);
VAR farbe:byte;
    i:integer;
CONST bogen=PI/180;
PROCEDURE FLY(richtung,x,y:word;FIGUR:POINTER);
VAR i:integer;
BEGIN
  {richtung=1 ist von rechts nach links }
  IF richtung=1
    THEN FOR i:=640 DOWNT0 x DO
      BEGIN
        PUTIMAGE(i,y,FIGUR^,0);
        {delay(0);}
      END
    ELSE FOR i:=1 TO x DO

```

```

        BEGIN
            PUTIMAGE(i,y,FIGUR^,0);
            {delay(0);}
        END;
    END;
BEGIN
    SETCOLOR(10);

    SetUserCharSize(2, 2, 2, 2);

    OUTTEXTxy(10,10,'STG7 (Steve G"ring) pr,,sentiert:');
    delay(5);
    FLY(1,X_ECKE+30,Y_ECKE,FIGUREN[8]);
    FLY(2,x_ECKE+390,Y_ECKE+300,FIGUREN[9]);
    SETUSERCHARSIZE(1,1,1,1);
    FOR i:=-180 TO 270 DO
    BEGIN
        CASE abs(i) MOD 5 OF
            0: farbe:=4;
            1: farbe:=12;
            2: farbe:=6;
            3: farbe:=12;
            4: farbe:=4;
        END;
        SETFILLSTYLE(1,farbe);
        x:= 300+TRUNC(200*cos(bogen*i));
        y:= 200+TRUNC(100*sin(bogen*i));
        BAR(x,y,x+30,y+30);
        delay(5);
    END;
    OUTTEXTxy(300,204,'S');
    delay(500);
    OUTTEXTxy(308,204,'T');
    delay(500);
    OUTTEXTxy(316,204,'G');
    delay(500);
    OUTTEXTxy(324,204,'7');

    OUTTEXTxy(450,400,'weiter mit ENTER');
    WAIT(ENTER);
    CLEARDEVICE;
    OUTTEXTxy(x_ecke,400,'MIT (ESC) BEENDEN');
    OUTTEXTxy(450,400,'COPYRIGHT BY STG7-2006');
END;
{.....}
PROCEDURE OUTRO;
VAR BETA: ARRAY[1..7] OF STRING; {beinhaltet alle BETATESTER}
    i:integer;
    SIZE:word;
    TEXT:POINTER;
BEGIN
    BETA[1]:=' Steve G"ring';
    BETA[2]:=' Peter G"ring';
    BETA[3]:=' Alexander Flock';
    BETA[4]:=' Sebastian Funke';
    BETA[5]:=' Benjamin Goldau';
    BETA[6]:=' Christian Zengerling';
    BETA[7]:=' Sascha Thiel und viele weitere...';

    CLEARDEVICE;
    SETCOLOR(10);
    SETTEXTSTYLE(0,0,1);
    OUTTEXTxy(100,100,'STG7 pr,,sentierte FUTURE DIGGER');
    OUTTEXTxy(100,116,'COPYRIGHT 2006 by Steve G"ring stg7@gmx.de');
    OUTTEXTxy(100,132,'BETATESTER: ');
    FOR i:=1 TO 7 DO
        OUTTEXTxy(100,138+i*16,BETA[i]);
    OUTTEXTxy(100,148+8*16,'weiter mit ENTER');
    Size := ImageSize(100,100,600,148+9*16);
    GetMem(TEXT, Size);

```



```

GetImage(100,100,600,148+9*16,TEXT^);
CLEARDEVICE;
i:=480;
WHILE NOT ((i<=0)OR(INKEY=ENTER)) DO
  BEGIN
    PUTIMAGE(100,i,TEXT^,0);
    i:=i-1;
  END;
END;
{.....}

BEGIN {Hauptprogramm}

menu[1]:=' Spielen  ';
menu[2]:=' Optionen  ';
menu[3]:=' Hilfe     ';
menu[4]:=' ENDE      ';

GrafikSchriftINI;
FIGUREN_INI(figuren);
INTRO(FIGUREN);

MOUSEINIT(Exists,Button);
MOUSESHOW;

start_level:=1;

FOR i:=0 TO 19 DO {Muster zeichnen}
  PUTIMAGE(X_ECKE+30*i,Y_ECKE,figuren[6]^,0);
FOR i:=0 TO 19 DO
  PUTIMAGE(X_Ecke+30*i,Y_Ecke+300,figuren[6]^,0);

PUTIMAGE(X_ECKE+30,Y_ECKE,figuren[8]^,0);
PUTIMAGE(X_ECKE+390,Y_ECKE+300,figuren[9]^,0);
PUTIMAGE(x_ECKE+30,Y_ECKE+60,figuren[1]^,0);
PUTIMAGE(560,y_ECKE+240,FIGUREN[2]^,0);

SETTEXTSTYLE(1,HorizDir,1);
SetUserCharSize(2, 2, 2, 2);
auswahl:=0;
REPEAT
  MOUSEEVENT(taste,x,y);
  SETFILLSTYLE(1,0);
  FOR i:=1 TO 4 DO
    BEGIN
      x_:=230;
      y_:=90+40*i;
      IF MOUSEOVER(x_,y_,x_+180,y_+40)
        THEN BEGIN
          SETCOLOR(4);
          IF TASTE=1
            THEN auswahl:=i;
          END
        ELSE SETCOLOR(10);
      OUTTEXTxy(x_,y_,menu[i]);
    END;
  MOUSECHANGE;
CASE auswahl OF
  1:BEGIN
    GAME(figuren,start_level);
    auswahl:=0;
    SETFILLSTYLE(1,0);
    BAR(0,0,640,Y_ECKE);
    BAR(X_ECKE+30,Y_ECKE+30,X_ECKE+570,Y_ECKE+300);
  END;
  2:BEGIN
    Optionen(start_level);
    auswahl:=0;
    SETFILLSTYLE(1,0);

```

```

        BAR(X_ECKE+60,Y_ECKE+30,X_ECKE+540,Y_ECKE+300);
    END;
3:BEGIN
    Hilfe;
    auswahl:=0;
    SETFILLSTYLE(1,0);
    BAR(X_ECKE+60,Y_ECKE+30,X_ECKE+540,Y_ECKE+300);
    END;
    END;
    eingabe:=inkey;
UNTIL (eingabe=ESC) OR (AUSWAHL=4);

MOUSEHIDE;

BOX('Spiel wird beendet!','weiter mit ENTER');

WHILE NOT (EINGABE=ENTER) DO {Animation}
BEGIN
    IF KEYPRESSED
    THEN eingabe:=readkey;
    FOR i:=1 TO 5 DO
    BEGIN
        PUTIMAGE(x_ECKE+30,Y_ECKE+60,figuren[i]^,NORMALPUT);
        DELAY(100);
    END;
    END;
    OUTRO;
    CloseGraph; {Beenden des Grafikmodus}
END.{Hauptprogramm}
{.....}

{Hinweis :
  Ein Problem kann bei mehrmaligen RUN^RUN auftreten,
  DER COMPILER Zeigt dann
  HEAP OVERFLOW Error an
  Um das Programm dann noch einmal zu starten muss TurboPascal beendet und
  neugestartet werden.
}
```

4.2. QUELLTEXT VON STG_GAME.PAS

```
UNIT STG_GAME;
INTERFACE
  USES CRT, GRAPH, DOS, MOUSE;
  CONST HOCH=#72;
        RUNTER=#80;
        LINKS=#75;
        RECHTS=#77;
        ENTER=#13;
        ESC=#27;
        AN=TRUE;
        AUS=FALSE;
        CenterX=320;
        CenterY=240;

  PROCEDURE cursor(an_aus:boolean);
    {Cursor an aus schalten im Textmodus}
  PROCEDURE DEL_TASTATUR;
    {Lösche den Tastaturspeicher}
  PROCEDURE WAIT(TASTE:CHAR);
    {Wartet auf eine bestimmte Taste}
  PROCEDURE LOAD_STG_GRAFIK(x,y:integer;datei_name:string);
    {STG GRAFIK LADEN}
  FUNCTION ToStr(Zahl: Longint): String;
    {Zahl in String umwandeln}
  FUNCTION INKEY:char;
    {Zeichen ohne Warten einlesen}
  PROCEDURE MOUSECHANGE;
    {Wartet auf das Bewegen der Maus}
  FUNCTION MOUSEOVER(x1,y1,x2,y2:word):boolean;
    {überprüft ob die Maus sich in einem bestimmten Rechteck(x1,y1,x2,y2)
      befindet}
  PROCEDURE BOX(zeile1,zeile2:string);
    {zeichnet eine MessageBox}
  PROCEDURE WINDOWBOX(x,y,x1,y1:word;zeile_oben:string);
    {großes fenster mit Überschrift }
  FUNCTION TIME:longint;
    {Zeit in Sekunden}
  PROCEDURE INPUT(VAR text:string);
    {readln Ersatz für GrafikModus (mit Outtext)}
```

IMPLEMENTATION

```
PROCEDURE cursor(an_aus:boolean);{Quelle www.webplain.de}
VAR regs:registers;
BEGIN
  IF an_aus
  THEN
    BEGIN
      regs.ax := $0100;
      regs.cx := $0607;
      intr($10, regs);
    END
  ELSE
    BEGIN
      regs.ax := $0100;
      regs.cx := $2607;
      intr($10, regs);
    END;
  END;
END;

PROCEDURE DEL_TASTATUR;{Quelle www.webplain.de}
BEGIN
  inline($FA);
  memw[$40 : $1A] := memw[$40 : $1C];
  inline($FB);
END;

PROCEDURE WAIT (TASTE:CHAR);
VAR EINGABE:CHAR;
BEGIN
  REPEAT
    EINGABE:=READKEY;
  UNTIL EINGABE=TASTE;
END;

PROCEDURE LOAD_STG_GRAFIK(x,y:integer;datei_name:string);
VAR Datei:Text;
    zeile:string;
    zeilen_nr:byte;
    i:integer;
    farbe,zeichen,fehler:integer;
BEGIN
  Assign(Datei, datei_name);
  ReSET(Datei);
  zeilen_nr:=1;
  WHILE NOT (EOF(Datei)) DO
    BEGIN
      READLN (DATEI, zeile);
      FOR i:=1 TO LENGTH(zeile) DO
        BEGIN
          VAL(copy(zeile,i,1),zeichen,fehler);
          CASE zeichen OF
            0: farbe:=0;      {... Schwarz ...}
            1: farbe:=14;     {... Gelb ...}
            2: farbe:=9;      {... hellblau ...}
            3: farbe:=6 ;     {... braun ...}
            4: farbe:=2 ;     {... grn ...}
            5: farbe:=1 ;     {... blau ...}
            6: farbe:=7;      {... hellgrau ...}
            7: farbe:=8;      {...dunkelgrau...}
            8: farbe:=15;     {... weiß ...}
            9: farbe:=4       {... rot ...}
```

```

        END;
        PUTPIXEL(x+i,y+zeilen_nr,farbe);
    END;
    zeilen_nr:=zeilen_nr+1;
END;
Close(Datei);
END;

FUNCTION ToStr(Zahl: Longint): String;
{Zahl in String umwandeln }
VAR
    s: string[11];
BEGIN
    Str(Zahl, S);
    ToStr := S;
END;

FUNCTION INKEY;
BEGIN
    IF KEYPRESSED
    THEN inkey:=readkey
    ELSE inkey:=' ';
END;

PROCEDURE MOUSECHANGE;
VAR taste,x,y:word;
    tastel,xl,y1:word;

    CHANGE:boolean;
BEGIN
    REPEAT
        MouseEvent(taste,x,y);
        delay(5);
        MouseEvent(tastel,xl,y1);
        CHANGE:=(X<>X1) AND
            (y<>y1) OR
            (TASTE<>TASTE1);
    UNTIL (CHANGE) OR (INKEY=ESC);
END;

FUNCTION MOUSEOVER(x1,y1,x2,y2:word):boolean;
VAR x,y:word;
BEGIN
    MOUSEGETPOS(x,y);
    MOUSEOVER:= (x>x1) AND (x<x2) AND (y>y1) AND (y<y2)
END;

PROCEDURE BOX;
VAR Text_Settings: TextSettingsType;
    Fill_settings: FillSettingsType;
    alte_farbe:word;
BEGIN
    GETFILLSETTINGS(FILL_SETTINGS);
    alte_farbe:=GETCOLOR;
    GETTEXTSETTINGS(Text_Settings);

    SETFILLSTYLE(1,1);
    SETTEXTSTYLE(DefaultFont,HORIZDIR,1);

    SETFILLSTYLE(1,15);
    BAR(198,198,422,252);

    SETFILLSTYLE(1,1);
    BAR(200,200,420,250);

```

```

SETCOLOR(15);
OUTTEXTXY(210,210,zeile1);
OUTTEXTXY(210,230,zeile2);

{Alten Zustand wiederherstellen}
WITH TEXT_SETTINGS DO
BEGIN
    SetTextJustify(Horiz, Vert);
    SetTextStyle(Font, Direction, CharSize);
END;
WITH Fill_SETTINGS DO
    SetFillStyle(Pattern, Color);

    SETCOLOR(alte_farbe);
END;
PROCEDURE WINDOWBOX;
VAR Text_Settings: TextSettingsType;
    Fill_settings: FillSettingsType;
    alte_farbe:word;
BEGIN
    GETFILLSETTINGS(FILL_SETTINGS);
    alte_farbe:=GETCOLOR;
    GETTEXTSETTINGS(Text_Settings);

    SETFILLSTYLE(1,1);
    SETTEXTSTYLE(DefaultFont,HORIZDIR,1);

    SETFILLSTYLE(1,15);
    BAR(x,y,x1,y1);

    SETFILLSTYLE(1,1);
    BAR(x+2,y+2,x1-2,y1-1);

    SETCOLOR(15);
    OUTTEXTXY(x+5,y+5,zeile_oben);

    {Alten Zustand wiederherstellen}
WITH TEXT_SETTINGS DO
BEGIN
    SetTextJustify(Horiz, Vert);
    SetTextStyle(Font, Direction, CharSize);
END;
WITH Fill_SETTINGS DO
    SetFillStyle(Pattern, Color);
    SETCOLOR(alte_farbe);
END;
FUNCTION TIME:longint;
VAR std,min,sec,hsec:word;
BEGIN
    GETTIME(std,min,sec,hsec);
    TIME:=std*60*60+min*60+sec;
END;
PROCEDURE INPUT(VAR text:string);
VAR taste:char;
BEGIN
    text:='';
    WHILE NOT (TASTE=ENTER) DO
        BEGIN
            taste:=readkey;
            IF NOT (TASTE=ENTER)

```

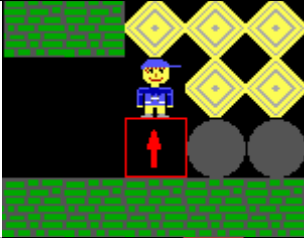
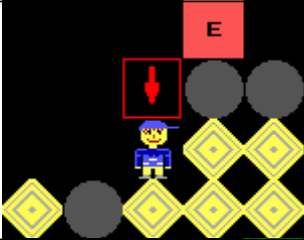
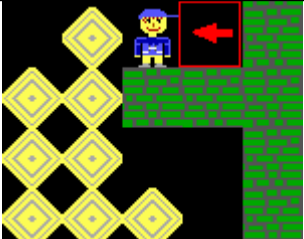


```

    THEN
    BEGIN
        text:=text+taste;
        OUTTEXT(taste);
    END;
END;
END;

BEGIN
END.

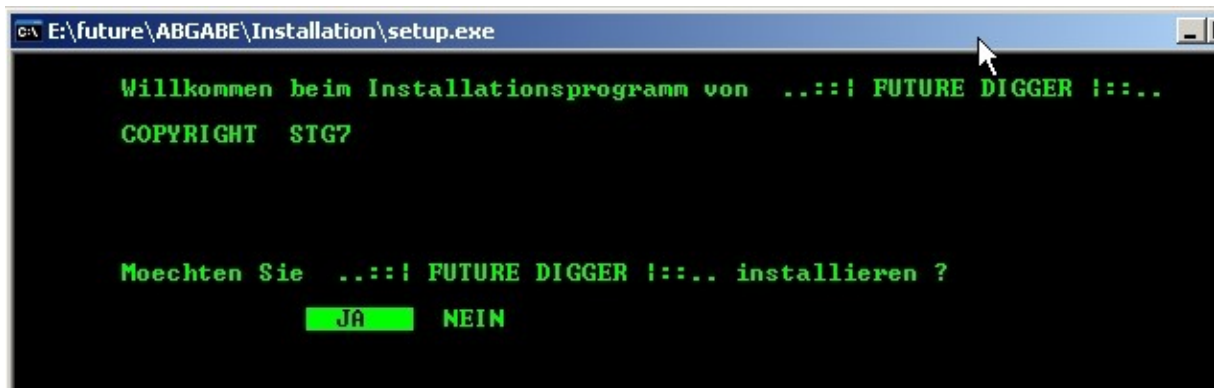
```

5. Test

<u>EINGABE</u>	<u>AUSGABE</u>	
TASTE W		Bewegung der Spielfigur nach oben
TASTE S		Bewegung der Spielfigur nach unten
TASTE A		Bewegung der Spielfigur nach rechts
TASTE D		Bewegung der Spielfigur nach rechts
Andere TASTE (außer ESC)		keine Veränderung feststellbar
ESC	Beendet das Spiel	

6. Installation

Das Spiel wird einfach mit dem mitgelieferten Installationsprogramm installiert, indem man die Anweisungen im Programm befolgt.



Das Programmpaket umfasst alle Quelltexte, alle benötigten Units, einen Grafik- und einen Leveleditor (mit FREEBASIC Quellcode) und die Spieldaten (Levels, Grafiken, Dokumentation)

7. Benutzerhandbuch

Bitte lesen Sie die Datei „Liesmich.doc“ im Verzeichnis „Dokumentation\“ im Installationsordner von FUTURE DIGGER.