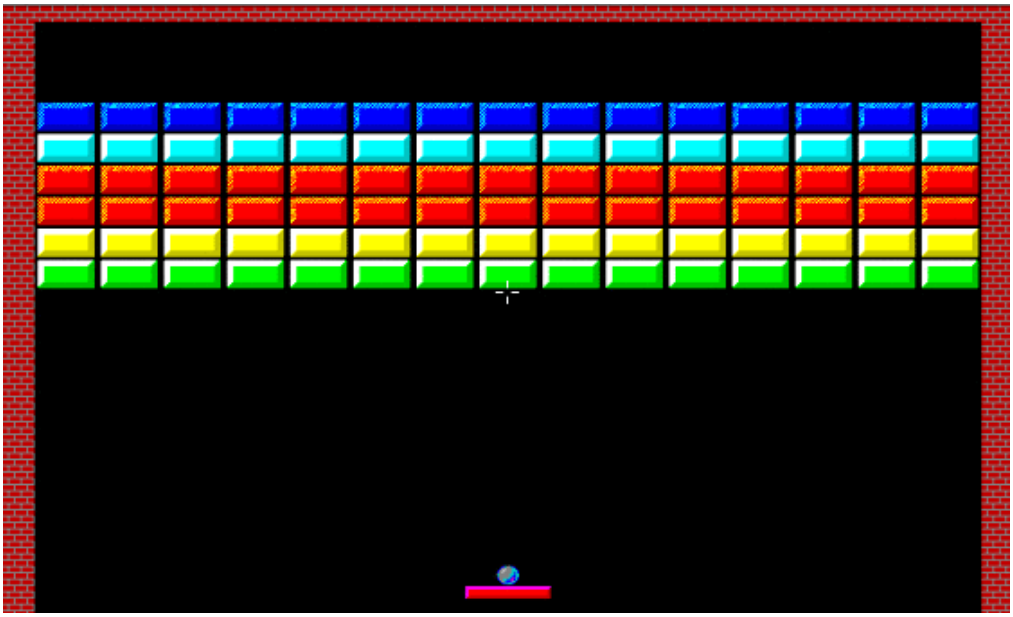
Übungseinheit: # 4

BRICKS



Entwicklungsumgebung: Excel 2007

Kursunterlagen erstell von:

Ing. Harald Mitsch
Josefgasse 6
A 3380 Pöchlarn

TECHNISCHES BÜRO MITSCH
Ihr IT-Betreuer in Pöchlarn
seit 1990 - 0676/588 09 16
http://www.tb-mitsch.at

letzte Aktualisierung: 11. Jänner 2019

INHALTSVERZEICHNIS

1.) Allgemeines

Nun wollen wir mal ein wenig Bewegung ins Spiel bringen. Dazu versuchen wir das Spiel Bricks nachzuprogrammieren.

2.) Spielregeln

Bei Bricks bewegt man am unteren Rand des Spielfeldes ein Brett nach links bzw. nach rechts und versucht damit den Ball zu treffen. Der Ball wird nach dem Prinzip Einfallswinkel = Ausfallswinkel zurück ins Spielfeld befördert. Trifft der Ball auf einen Ziegel (= Brick), dann wird er vom Spielfeld genommen und der Spieler erhält dafür Punkte. Verfehlt man mit dem Brett den Ball, dann verschwindet der Ball und man erhält einen neuen. Zu Beginn hat man eine gewisse Anzahl von Bällen zur Verfügung. Nachdem der letzte Ball verfehlt wurde ist auch das Spiel zu Ende. Ziel ist es möglichst viele Punkte zu sammeln.

3.) Überlegungen zum Spielablauf

Der Anwender soll nur auf eine Schaltfläche klicken können um das Spiel zu starten. Dabei soll nach jedem Start das Ausgangsspielfeld neu aufgebaut werden. Das Brett befindet sich in der Mitte und darauf der Ball. Mit den Pfeiltasten nach rechts und nach links soll das Brett bewegt werden. Mit der Taste nach oben wird der Ball ins Spiel gebracht. Trifft der Ball auf einen Ziegel, dann lassen wir ihn verschwinden und erhöhen die Punkte. Sind alle Ziegelsteine vom Spielfeld entfernt, dann bauen wir ein neues Spielfeld (neuer Level) auf und erhöhen die Geschwindigkeit des Balles. Nachdem alle Levels durchgespielt wurden geht das Spiel wieder bei Level 1 weiter. Zusätzlich machen wir das Brett etwas kleiner. Wird der Ball nicht getroffen, dann erhält der Spieler einen neuen Ball. Sind alle verfügbaren Bälle aufgebraucht so ist auch das Spiel zu Ende.

4.) Programmieren des Spiels

4.1.) Vorbereitungen

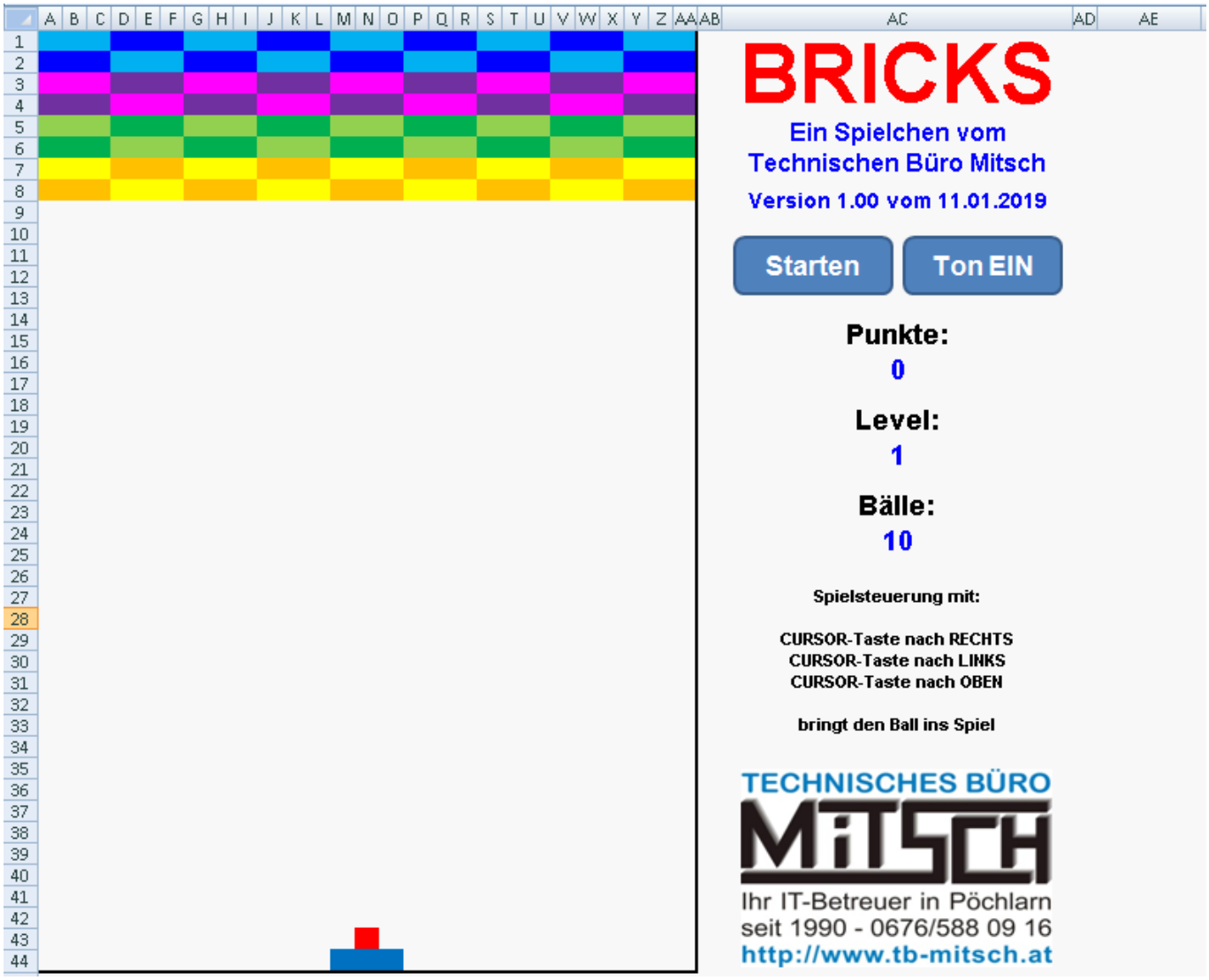
Zuerst legen wir uns einen neuen Ordner an in welchem wir alle für das Spiel benötigte Dateien ablegen. Dann kopieren wir die Datei: **Muster_Vorlage.xlsm** (die überarbeitete Excel Datei aus der Übungseinheit #3) in diesen Ordner. Von derselben Datei legen wir uns eine weitere Kopie an und benennen sie um z.B. auf: **Bricks.xlsm**. In dieser Datei werden wir das Spiel programmieren. Sollten wir dabei auf Funktionen oder andere Elemente stoßen, welche wir bei anderen Spielen ebenfalls gebrauchen könnten, dann werden wir diese zusätzlich auch in der Mustervorlage hinterlegen.

4.2.) Gestaltung des Spielfeldes

Um das Tabellenblatt möglichst optimal auszunutzen soll das Spielfeld 44 Zeilen hoch sein (bei einer Bildschirmauflösung von 1280 x 800 Pixel). Die Anzahl der Spalten hängt von der Breite der Ziegelsteine ab. Optisch sind 9 Ziegelsteine mit einer Breite von je 3 Spalten ideal. Daher definieren wir die Spaltenbreite für die ersten 27 Spalten von A bis AA mit 2. Damit die Zellen quadratisch erscheinen setzen wir die Zeilenhöhe auf 12,75. Für die Schrift (auch wenn wir sie nicht benötigen), legen wir am besten den Font Arial mit einer Schriftgröße von 10 fest. Rechts neben dem Spielfeld sollen die Punkte, der Level und die Anzahl der noch verfügbaren Bälle ausgegeben werden.

Um sich einen besseren Überblick zu verschaffen, wie das Spielfeld später genau aussieht, gestalten wir noch 8 Zeilen mit Ziegelsteinen indem wir jeweils 3 benachbarte Zellen mit einer Hintergrundfarbe hinterlegen. Damit später die Programmierung einfacher wird legen wir zusätzlich fest, daß die einzelnen Bricks keinen Rahmen und kein Muster haben sollen. Einen dicken Gesamtrahmen über das gesamte Spielfeld können wir aber anlegen.

Zum Schluß stellen wir noch am unteren Ende des Spielfeldes das Brett (3 Zellen breit) und in der Mitte des Brettes den Spielball dar. Das Spielfeld könnte dann so aussehen:



Nun benennen wir noch die Zellen für die Ausgabe der Punkte, Level und Bälle mit **PUNKTE**, **LEVEL** und **BAELLE**. Jetzt kommen wir zu einem Verhalten von Excel welches wir gleich am Anfang berücksichtigen müssen. Da wir die Steuerung über die Cursor-Tasten programmieren wollen, setzt Excel die aktuelle Cursor-Position automatisch in Abhängigkeit der gedrückten Taste jedes Mal um eine Zelle weiter. Um das zu verhindern definieren wir eine Zelle auf welche nach jedem Tastendruck der Cursor automatisch gesetzt werden soll. Hier empfiehlt sich die Zelle mit dem Spieltitel ("Bricks" in AC1). Diese Zelle benennen wir mit **CURSOR_PARKPOSITION**.

Das aktuell gestaltete Spielfeld (A1:AA42) kopieren wir nun in 5 neue Tabellenblätter welche wir mit Level1, Level 2, Level3, Level4 und Level5 bezeichnen. Darin gestalten wir später unsere Levels.

4.3.) Überlegungen zum Programmablauf

Beim Klick auf die Schaltfläche **Starten** soll ein Makro aufgerufen werden, welches zuerst eine Funktion aufruft die das Spielfeld für den ersten Spielaufruf vorbereitet. Dabei übergeben wir einen Parameter anhand dessen wir feststellen können, ob das Spiel neu gestartet wurde oder ob ein neuer Level vorbereitet werden soll. Danach soll die Hauptfunktion für den Spielablauf gestartet werden. Diese soll in einer Endlos-Schleife permanent die Tastatur abfragen und die entsprechende Aktion ausführen. Wird keine Taste gedrückt, so wird auf alle Fälle der Ball weiterbewegt.

Dazu wollen wir separate Funktionen erstellen: **NachRechts** (Brett nach rechts bewegen), **NachLinks** (Brett nach links bewegen), **BallBewegen** (den Spielball um eine Position weiterbewegen) und zum Entfernen eines Ziegels, falls er getroffen wurde, die Funktion **ZiegelLoeschen**.

Gleichzeitig wollen wir beim Drücken der ESC-Taste das Spiel sofort beenden. Beim Drücken der Taste PAUSE soll das Spiel gestoppt werden (ist während der Programmierung ganz sinnvoll – kann aber am Schluß wieder deaktiviert werden).

Die Hinterlegung der Sounds und der Schaltfläche **Ton EIN** erledigen wir als Letztes und zwar genauso wie in der Übungseinheit #2 (Schiebe-Puzzle).

Zum besseren Verständnis des späteren Programmcodes gleich zu Beginn noch die Überlegungen wie wir uns die Bewegungsrichtung des Balles merken und verändern können. Grundsätzlich gibt es 9 Bewegungsrichtungen: 0=keine Bewegung, 1 bis 4 nach rechts, links oben und unten, 5 bis 8 diagonale Richtungen. Wir benötigen hier nur die vier Diagonalen. Daher verwenden wir die Buchstaben L=Links, R=Rechts, O=nach oben und U=nach unten. Die aktuelle Richtung merken wir uns dann in einer Variable **stRichtung** und wenn der Ball umgelenkt werden soll in **stRichtungNeu**. Dabei legen wir folgendes fest:

RO = Ball bewegt sich aktuell nach rechts oben
RU = Ball bewegt sich aktuell nach rechts unten
LO = Ball bewegt sich aktuell nach links oben
LU = Ball bewegt sich aktuell nach links unten

Bei diesem Spielprogramm soll eine weitere Methode zum Programmieren des Blattschutzes aufgezeigt werden. Dabei werden absolut alle Zellen der Datei als gesperrt definiert. Beim Setzen des Blattschutzes aktivieren wir nur "**Zellen formatieren**" (um die Hintergrundfarbe der Zellen verändern zu können) und "**Objekte bearbeiten**" (um die Bezeichnung der Schaltfläche **Ton EIN** auf **Ton AUS** ändern zu können).

Müssen wir im Tabellenblatt tatsächlich etwas eintragen (z.B. die Punkteanzahl), dann heben wir über den Programmcode zuerst den Schutz auf, machen dann unsere Eintragungen und setzen danach den Blattschutz erneut.

Dazu definieren wir uns im Modul **Globales** eine neue globale Konstante für ein Passwort:

```
Global Const PASSWORT = ""                          'Passwort fuer Blattschutz.
```

Zusätzlich erstellen wir im Modul **Standard** zwei neue Funktionen (welche wir auch in unsere Muster Vorlage einkopieren werden) zum Setzen und Aufheben des Blattschutzes.

Hier gleich die fertigen Funktionen mit Beispielen was man sonst noch so alles schützen kann:

```
Public Function BlattschutzSetzen()
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       BlattschutzSetzen                                */
'*                                                                       */
'*      BESCHREIBUNG:   Setzt den Blattschutz fuer das aktuell selektierte  */
'*                      Tabellenblatt.                                   */
'*                      PASSWORT im Modul Globales hinterlegen.          */
'*                                                                       */
'*      RETURN          nichts                                           */
'*                                                                       */
'*      PARAMETER:      keine                                            */
'*                                                                       */
'*-----------------------------------------------------------------------*/
On Error GoTo Fehler                                  'Bei Fehler zur Fehlermarke.
                                                      'Balttschutz setzen:
ActiveSheet.Protect Password:=PASSWORT, _
                    AllowFormattingCells:=True, _
                    DrawingObjects:=False, _
                    Contents:=True, _
                    Scenarios:=True
                                                      'Selektieren sperren fuer:
ActiveSheet.EnableSelection = xlNoRestrictions        'Alle Zellen.
'ActiveSheet.EnableSelection = xlUnlockedCells         'Gesperrte Zellen.


'                                                      'Vorlage fuer Passwortschutz
'                                                      'wenn man alles abgehakt hat:
'ActiveSheet.Protect Password:=PASSWORT, _
'                    DrawingObjects:=False, _
'                    Contents:=True, _
'                    Scenarios:=False, _
'                    AllowFormattingCells:=True, _
'                    AllowFormattingColumns:=True, _
'                    AllowFormattingRows:=True, _
'                    AllowInsertingColumns:=True, _
'                    AllowInsertingRows:=True, _
'                    AllowInsertingHyperlinks:=True, _
'                    AllowDeletingColumns:=True, _
'                    AllowDeletingRows:=True, _
'                    AllowSorting:=True, _
'                    AllowFiltering:=True, _
'                    AllowUsingPivotTables:=True
'
'                                                      'Vorlage wenn nichts abgehakt:
'ActiveSheet.Protect DrawingObjects:=True, Contents:=True, Scenarios:=True

                                                      'Fehlerbehandlungsroutine:
Beenden:                                              '=========================
  Exit Function                                       'Funktion abbrechen.

Fehler:                                               'Fehlermarke - Fehlerbehandlung:
  MsgBox Err.Description                              'Fehlermeldung ausgeben.
  Resume Beenden                                      'Bei Funktionsende weitermachen.
  Resume                                              'Nur fuer Testzwecke hinterlegt.
End Function                                          'Funktionsende.
```

```
Public Function BlattschutzAufheben()
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*----------------------------------------------------------------------*/
'*                                                                      */
'*      FUNKTION:       BlattschutzAufheben                             */
'*                                                                      */
'*      BESCHREIBUNG:   Setzt den Blattschutz fuer das aktuell selektierte  */
'*                      Tabellenblatt wieder zurueck bzw. hebt ihn auf.     */
'*                      PASSWORT im Modul Globales hinterlegen.         */
'*                                                                      */
'*      RETURN          nichts                                          */
'*                                                                      */
'*      PARAMETER:      keine                                           */
'*                                                                      */
'*----------------------------------------------------------------------*/
On Error GoTo Fehler                            'Bei Fehler zur Fehlermarke.
ActiveSheet.Unprotect Password:=PASSWORT        'Blattschutz aufheben.

                                                'Fehlerbehandlungsroutine:
Beenden:                                        '=========================
  Exit Function                                 'Funktion abbrechen.

Fehler:                                         'Fehlermarke - Fehlerbehandlung:
  MsgBox Err.Description                        'Fehlermeldung ausgeben.
  Resume Beenden                                'Bei Funktionsende weitermachen.
  Resume                                        'Nur fuer Testzwecke hinterlegt.
End Function                                    'Funktionsende.
```

4.4.) Programmierung des Spieles

4.4.1.) Globale Konstanten und Variablen

Zuerst ändern wir Modul **Globales** folgende Konstanten:

```
Global Const PASSWORT = ""                      'Passwort fuer Blattschutz.
Global Const PROGRAMMNAME = "Bricks"            'Name des Programmes.
Global Const TABELLENBLATT = "Bricks"           'Tabellenblatt fuer Spielfeld.
```

Dann hinterlegen wir noch alles was wir später beim Programmieren so benötigen werden:

```
Global Const TABELLENBLATT_LEVEL = "Level"      'Tabellenblatt fuer Levels -
                                                'ergaenzt mit 1, 2, ...
Global Const MAX_OBEN = 1                       '1. Zeile Spielfeld.
Global Const MAX_UNTEN = 44                     'letzte Zeile Spielfeld.
Global Const MAX_LINKS = 1                      '1. Spalte Spielfeld (A).
Global Const MAX_RECHTS = 27                    'letzte Spalte Spielfeld (AA).
Global Const MAX_BAELLE = 10                    'Maximale Anzahl der Baelle.
Global Const MAX_LEVEL = 5                      'Maximale Anzahl der Levels.

Global Const FARBE_HINTERGRUND = 16316664       '16316664 = weiss
Global Const FARBE_BRETT = 12611584             '12611584 = blau
Global Const FARBE_BALL = 255                   '255 = rot
```

```
Global Const STARTGESCHWINDIGKEIT = 50              'in Millisekunden (50 = Standard)
Global Const GESCHWINDIGKEITSAENDERUNG = 2          'Nach jedem Level 2 ms kuerzer.
Global iGeschwindigkeit     As Integer              'Aktuelle Geschwindigkeit.

Global stRichtung           As String               'Aktuelle Bewegungsrichtung Ball.
Global stRichtungNeu        As String               'Neue Bewegungsrichtung Ball.
Global stTastendruck        As String               'Zuletzt gedrueckte Taste.

Global iZeileBrett          As Integer              'Aktuelle Position fuer Brett:
Global iSpalteBrett         As Integer
Global iZeileBall           As Integer              'Aktuelle Position fuer Ball:
Global iSpalteBall          As Integer
Global bBallImSpiel         As Boolean              'Ball befindet sich im Spiel.

Global lPunkte              As Long                 'Aktuelle Anzahl der Punkte.
Global iBaelle              As Integer              'Anzahl noch verfuegbare Baelle.
Global iLevel               As Integer              'Akt. Level (1 bis MAX_LEVEL).
Global iLevelIst            As Integer              'Akt. Level (1 bis ...).
Global stTabelleLevel       As String               'Tabellenblattname akt. Level.
Global iAktBricks           As Integer              'Anzahl Ziegel im akt. Level.
```

4.4.2.) Funktion zum Starten des Spieles

Erstellen wir nun die Funktion **SpielStarten** im Modul **Funktionen** die folgendes erledigt: Zuerst den Blattschutz aufheben. Beim allerersten Aufruf der Startfunktion sollen die Standardwerte festgelegt und im Tabellenblatt ausgegeben werden. Bei jedem weiteren Aufruf wird unterschieden, ob ein neuer Level angezeigt werden soll oder ob nur ein neuer Ball ins Spiel gebracht wird. Nach dem Einlesen eines neuen Levels muß die Anzahl der Ziegelsteine ermittelt werden, damit wir später feststellen können ob der Level bereits zu Ende gespielt worden ist. Beim ersten Durchlauf der Levels 1 bis MAX_LEVELS soll das Brett nicht 3 sondern 5 Zellen breit sein. Zum Schluß wieder den Blattschutz setzen.

Langer Rede, kurzer Sinn – hier die fertige Funktion:

```
Public Function SpielStarten(bErsterStart As Boolean) As Boolean
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       SpielStarten                                     */
'*                                                                       */
'*      BESCHREIBUNG:   Ist der Uebergabe-Parameter bErsterStart True, dann */
'*                      wird ein neuer Level eingeblendet. Ist die aktuelle */
'*                      Punkteanzahl auch 0, dann werden die Startwerte  */
'*                      festgelegt und im Tabellenblatt angezeigt.       */
'*                      Ist der Uebergabe-Parameter False, dann wird nur */
'*                      ein neuer Ball auf dem Brett ausgegeben, die     */
'*                      aktuellen Ziegelsteine bleiben erhalten. Bei einem */
'*                      neuen Level wird auch die Anzahl der vorhandenen */
'*                      Bricks ermittelt.                                */
'*                                                                       */
'*      RETURN          Boolean ..... True = Funktion in Ordnung         */
'*                                    False = Fehler in Funktion         */
'*                                                                       */
'*      PARAMETER:      Boolean ..... True = Neuen Level aufbauen        */
'*                                    False = Nur Ball neu einblenden    */
'*                                                                       */
'*-----------------------------------------------------------------------*/
```

```
Dim iZeile              As Integer                  'Aktuelle Zeilennummer.
Dim iSpalte             As Integer                  'Aktuelle Spaltennummer.

On Error GoTo Fehler                                'Bei Fehler zur Fehlermarke.
SpielStarten = True                                 'Returnwert = alles OK setzen.
bFluchttaste = False                                'ESC wurde noch nicht gedrueckt.

Set oTabellenblatt = Sheets(TABELLENBLATT)          'Akt. Tabellenblatt festlegen.
oTabellenblatt.Select                               'Tabellenblatt auswaehlen.
Call BlattschutzAufheben                            'Blattschutz aufheben.

bBallImSpiel = False                                'Ball ist noch nicht im Spiel.
iZeileBrett = MAX_UNTEN                             'Startpositionen fuer Brett und
iSpalteBrett = Int((MAX_RECHTS - MAX_LINKS) / 2)    'Ball festlegen (Brett in Mitte,
iZeileBall = iZeileBrett - 1                        'Ball mittig und ueber Brett).
iSpalteBall = iSpalteBrett + 1
For iSpalte = MAX_LINKS To MAX_RECHTS               'Ball und Brett loeschen:
  oTabellenblatt.Cells(iZeileBall, iSpalte).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeileBall + 1, iSpalte).Interior.Color = FARBE_HINTERGRUND
Next iSpalte                                        'Ball und Brett zeichnen:
oTabellenblatt.Cells(iZeileBall, iSpalteBall).Interior.Color = FARBE_BALL
oTabellenblatt.Cells(iZeileBrett, iSpalteBrett).Interior.Color = FARBE_BRETT
oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 1).Interior.Color = FARBE_BRETT
oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 2).Interior.Color = FARBE_BRETT
If iLevelIst <= 5 Then                              'Beim ersten Level-Durchlauf
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett - 1).Interior.Color = FARBE_BRETT
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 3).Interior.Color = FARBE_BRETT
End If                                              'das Brett breiter machen.

If Int((2 * Rnd) + 1) = 1 Then                      'Zufaellige Richtung festlegen,
  stRichtung = "LO"                                 'ob der Ball beim Einwerfen
  stRichtungNeu = "LO"                              'nach links oben
Else                                                'oder
  stRichtung = "RO"                                 'nach rechts oben
  stRichtungNeu = "RO"                              'wegspringen soll. Natuerlich
End If                                              'ist das auch die neue Richtung.

If bErsterStart = True And lPunkte = 0 Then         'Beim ersten Start alle Variablen
  lPunkte = 0                                       'ruecksetzen, Standardwerte
  iBaelle = MAX_BAELLE                              'festlegen fuer maximale Anzahl
  iLevel = 1                                        'der Baelle, Level fuer Vorlage
  iLevelIst = 1                                     'und aktuell gespielter Level.
  iGeschwindigkeit = STARTGESCHWINDIGKEIT           'Startgeschwindigkeit festlegen.
  oTabellenblatt.Range("PUNKTE") = lPunkte          'Ausgabe der Startwerte im
  oTabellenblatt.Range("LEVEL") = iLevel            'Tabellenblatt.
  oTabellenblatt.Range("BAELLE") = iBaelle
End If

If bErsterStart = True Then                         'Ziegelsteine uebertragen:
  stTabelleLevel = TABELLENBLATT_LEVEL & Trim(Str(iLevel)) 'Tabellenblatt fuer die
                                                    'Vorlage des Levels festlegen.
  iAktBricks = 0                                    'Anzahl der Ziegel im Level
  For iZeile = MAX_OBEN To MAX_UNTEN - 2            'ruecksetzen. Dann das gesamte
    For iSpalte = MAX_LINKS To MAX_RECHTS           'Spielfeld abarbeiten und
      oTabellenblatt.Cells(iZeile, iSpalte).Interior.Color = _
                    Sheets(stTabelleLevel).Cells(iZeile, iSpalte).Interior.Color
      If oTabellenblatt.Cells(iZeile, iSpalte).Interior.Color <> FARBE_HINTERGRUND Then
        iAktBricks = iAktBricks + 1                 'die Anzahl der gefaerbten
      End If                                        'Zellen ermitteln. Da jeder
    Next iSpalte                                    'Ziegel aus 3 Zellen besteht
```

```
  Next iZeile                                          'errechnet sich die Anzahl
  iAktBricks = Int(iAktBricks / 3)                     'der Ziegel mit Division / 3.
End If

Range("CURSOR_PARKPOSITION").Select                    'Cursor auf Park-Position.
Call BlattschutzSetzen                                 'Blattschutz setzen.


                                                       'Fehlerbehandlungsroutine:
Beenden:                                               '=========================
  Application.Cursor = xlDefault                       'Sanduhr ausblenden.
  Exit Function                                        'Funktion abbrechen.

Fehler:                                                'Fehlermarke - Fehlerbehandlung:
  SpielStarten = False                                 'Returnwert = Fehler setzen.
  MsgBox Err.Description                               'Fehlermeldung ausgeben.
  Resume Beenden                                       'Bei Funktionsende weitermachen.
  Resume                                               'Nur fuer Testzwecke hinterlegt.
End Function                                           'Funktionsende.
```

4.4.3.) Funktion zur Steuerung nach Spielstart

Als nächstes erstellen wir nun die Funktion **Spielen** im Modul **Funktionen** welche die Ablaufsteuerung während des Spielens übernimmt. Um die neue Funktion kompilieren und testen zu können, müssen wir uns vorab noch folgende leere Funktionen anlegen:

```
Public Function NachRechts()
End Function                                           'Funktionsende.

Public Function NachLinks()
End Function                                           'Funktionsende.

Public Function BallBewegen()
End Function                                           'Funktionsende.
```

Die Spielsteuerung könnte dann so aussehen:

```
Public Function Spielen()
'************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                       */
'************************************************************************/
'*----------------------------------------------------------------------*/
'*                                                                      */
'*      FUNKTION:       Spielen                                         */
'*                                                                      */
'*      BESCHREIBUNG:   Startet eine Endlos-Schleife welche die Tastatur*/
'*                      ueberwacht und bei entsprechendem Tastendruck die*/
'*                      dazugehoerige Funktion aufruft. Mit der Taste PAUSE*/
'*                      kann die Endlos-Schleife vorzeitig beendet werden.*/
'*                      Der stTastendruck = "SPIELENDE" wird nachdem das*/
'*                      Spiel zu Ende ist in der entsprechenden Funktion*/
'*                      gesetzt: BallBewegen                            */
'*                                                                      */
'*      RETURN          nichts                                          */
'*                                                                      */
'*      PARAMETER:      keine                                           */
'*                                                                      */
'*----------------------------------------------------------------------*/
```

```
On Error GoTo Fehler                                'Bei Fehler zur Fehlerbehandlung.

stTastendruck = ""                                  'Gedrueckte Taste ruecksetzen.
While stTastendruck = ""                            'Solange bis Taste gedrueckt wird
  Range("CURSOR_PARKPOSITION").Select               'Cursor zurück auf Park-Position.

  If GetAsyncKeyState(vbKeyEscape) Then             'Wird ESC-Taste gedrueckt, dann
    stTastendruck = "ESC"                           'Abbruchsbedingung fuer Endlos-
    On Error Resume Next                            'Schleife festlegen und bei
  End If                                            'Fehler einfach weitermachen.

  If GetAsyncKeyState(vbKeyPause) Then              'WennPAUSE-Taste gedrueckt, dann
    stTastendruck = "ABBRECHEN"                     'Abbruchsbedingung fuer Endlos-
  End If                                            'Schleife festlegen.

  If GetAsyncKeyState(vbKeyRight) Then              'Wird RECHTS-Taste gedrueckt:
    Call NachRechts                                 'Brett nach rechts bewegen.
  End If

  If GetAsyncKeyState(vbKeyLeft) Then               'Wird LINKS-Taste gedrueckt, dann
    Call NachLinks                                  'Brett nach links bewegen.
  End If

  If GetAsyncKeyState(vbKeyUp) Then                 'Wird UP-Taste gedrueckt, dann
    bBallImSpiel = True                             'Ball ins Spiel einwerfen.
  End If

  If bBallImSpiel = True Then                       'Wenn Ball im Spiel ist, dann
    Call BallBewegen                                'den Ball bewegen.
  End If
  DoEvents                                          'Dem Programm zusaetzliche
                                                    'Rechenzeit zur Verfuegung
                                                    'stellen, da sonst alles
                                                    'nach ca. 4 Sekunden abstuerzt!
  Sleep iGeschwindigkeit                            'Eine Zeitlang warten (50 ms).
Wend                                                '(Steuert Spielgeschwindigkeit.)

If stTastendruck = "ESC" Then                       'Bei ESC-Taste.
  Call Fluchttaste                                  'Spiel sofort beenden und Datei
End If                                              'ohne Speichern schliessen.

If stTastendruck = "ABBRECHEN" Then                 'Bei PAUSE-.Taste:
  MsgBox "Spiel wurde abgebrochen!", _
         vbOKOnly + vbCritical + vbDefaultButton1, PROGRAMMNAME
End If

If stTastendruck = "SPIELENDE" Then                 'Bei Spielende:
  MsgBox "SPIELENDE - keine Bälle mehr!", _
         vbOKOnly + vbCritical + vbDefaultButton1, PROGRAMMNAME
End If

                                                    'Fehlerbehandlungsroutine:
Beenden:                                            '=========================
  Application.Cursor = xlDefault                    'Sanduhr ausblenden.
  Exit Function                                     'Funktion abbrechen.

Fehler:                                             'Fehlermarke - Fehlerbehandlung:
  MsgBox Err.Description                            'Fehlermeldung ausgeben.
  Resume Beenden                                    'Bei Funktionsende weitermachen.
  Resume                                            'Nur fuer Testzwecke hinterlegt.
End Function                                        'Funktionsende.
```

Das Grundgerüst für das Spiel haben wir jetzt ausprogrammiert. Jetzt brauchen wir nur mehr dafür sorgen, daß die beiden Funktionen auch wunschgemäß aufgerufen werden. Zuerst in der Prozedur **Workbook_Open** im Modul **DieseArbeitsmappe**. In dieser erweitern wir den bisherigen Funktionsaufruf folgendermaßen:

```
'*-------------------------------------------------------------------------*/
lPunkte = 0                                          'Punkteanzahl 0 setzen.
Call SpielStarten(True)                              'Startprogramm aufrufen.
'*-------------------------------------------------------------------------*/
```

Dann benötigen wir noch ein Makro (im Modul **Makros**) welches ebenfalls das Spiel startet:

```
Public Sub Spiel_Starten()
'***************************************************************************/
'*  Makro wird ausgefuehrt bei Klick auf Schaltflaeche: Starten            */
'***************************************************************************/
lPunkte = 0                                          'Punkte auf 0 setzten.
Call SpielStarten(True)                              'Spielfeld aufbauen.
Call Spielen                                         'Spiel starten.
End Sub                                              'Prozedurende.
```

Das Makro **Spiel_Starten** weisen wir abschließend der Schaltfläche **Starten** zu.

Nun können wir die Datei speichern, schließen und erneut öffnen – und jetzt testen… Beim Testen bzw. während wir das Spiel weiter ausprogrammieren empfiehlt es sich die Ansicht auf Zoom 75% zu schalten und die Multifunktionsleiste zu minimieren bzw. manuellen Zoomfaktor auf 70% zu setzen.

4.4.4.) Brett nach rechts oder links bewegen

Als nächstes erweitern wir im Modul **Funktionen** die Funktionen **NachRechts** und **NachLinks**: Die Dokumentation im Programmcode sollte die Funktionsweise klar beschreiben.

```
Public Function NachRechts()
'***************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                          */
'***************************************************************************/
'*-------------------------------------------------------------------------*/
'*                                                                         */
'*      FUNKTION:       NachRechts                                         */
'*                                                                         */
'*      BESCHREIBUNG:   Bewegt das Brett um eine Position nach rechts. Wenn */
'*                      der Ball nicht im Spiel ist, er liegt auf dem      */
'*                      Brett auf, dann den Ball auch eine Position nach   */
'*                      rechts verschieben.                                */
'*                      Gleichzeitig wird darauf geachtet, dass das Brett  */
'*                      nicht über das Spielfeld hinaus verschoben werden  */
'*                      kann.                                              */
'*                                                                         */
'*      RETURN          nichts                                             */
'*                                                                         */
'*      PARAMETER:      keine                                              */
'*                                                                         */
'*-------------------------------------------------------------------------*/
On Error GoTo Fehler                          'Bei Fehler zur Fehlerbehandlung.
```

```
If iLevelIst <= 5 Then                              'Brett ist 5 Zellen breit:
  'Aktuelle 5 Zellen mit Hintergrundfarbe loeschen:
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett - 1).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 0).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 1).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 2).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 3).Interior.Color = FARBE_HINTERGRUND

  'Spaltenbegrenzung um nicht ueber den rechten Spielfeldrand zu kommen:
  If iSpalteBrett < MAX_RECHTS - 3 Then iSpalteBrett = iSpalteBrett + 1

  'Neue 5 Zellen fuer Brett mit Brettfarbe zeichnen:
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett - 1).Interior.Color = FARBE_BRETT
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 0).Interior.Color = FARBE_BRETT
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 1).Interior.Color = FARBE_BRETT
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 2).Interior.Color = FARBE_BRETT
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 3).Interior.Color = FARBE_BRETT
Else                                                'Brett ist 3 Zellen breit:
  'Die 3 aktuellen Zellen fuer Brett mit Hintergrundfarbe loeschen:
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 0).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 1).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 2).Interior.Color = FARBE_HINTERGRUND

  'Spaltenbegrenzung um nicht ueber den rechten Spielfeldrand zu kommen:
  If iSpalteBrett < MAX_RECHTS - 2 Then iSpalteBrett = iSpalteBrett + 1

  'Neue 3 Zellen fuer Brett mit Brettfarbe zeichnen:
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 0).Interior.Color = FARBE_BRETT
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 1).Interior.Color = FARBE_BRETT
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 2).Interior.Color = FARBE_BRETT
End If
                                                    'Wenn der Ball nicht im Spiel ist,
If bBallImSpiel = False Then                        'dann auch Ball eins nach rechts.
  oTabellenblatt.Cells(iZeileBrett - 1, iSpalteBrett + 0).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeileBrett - 1, iSpalteBrett + 1).Interior.Color = FARBE_BALL
  iSpalteBall = iSpalteBrett + 1                    'Neue Zeile und Spalte fuer
  iZeileBall = iZeileBrett - 1                      'Ballposition merken.
End If
Range("CURSOR_PARKPOSITION").Select                 'Cursor auf Parkposition setzen.

                                                    'Fehlerbehandlungsroutine:
Beenden:                                            '========================
  Application.Cursor = xlDefault                    'Sanduhr ausblenden.
  Exit Function                                     'Funktion abbrechen.

Fehler:                                             'Fehlermarke - Fehlerbehandlung:
  NachRechts = False                                'Returnwert = Fehler setzen.
  MsgBox Err.Description                            'Fehlermeldung ausgeben.
  Resume Beenden                                    'Bei Funktionsende weitermachen.
  Resume                                            'Nur fuer Testzwecke hinterlegt.
End Function                                        'Funktionsende.


Public Function NachLinks()
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       NachLinks                                        */
'*                                                                       */
'*      BESCHREIBUNG:   Bewegt das Brett um eine Position nach links. Wenn  */
'*                      der Ball nicht im Spiel ist, er liegt auf dem    */
'*                      Brett auf, dann den Ball auch eine Position nach */
'*                      links verschieben.                               */
'*                      Gleichzeitig wird darauf geachtet, dass das Brett   */
```

```
'*                      nicht über das Spielfeld hinaus verschoben werden   */
'*                      kann.                                               */
'*                                                                          */
'*      RETURN          nichts                                              */
'*                                                                          */
'*      PARAMETER:      keine                                               */
'*                                                                          */
'*--------------------------------------------------------------------------*/
On Error GoTo Fehler                                'Bei Fehler zur Fehlerbehandlung.

If iLevelIst <= 5 Then                              'Brett ist 5 Zellen breit:
  'Aktuelle 5 Zellen mit Hintergrundfarbe loeschen:
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 3).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 2).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 1).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 0).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett - 1).Interior.Color = FARBE_HINTERGRUND

  'Spaltenbegrenzung um nicht ueber den linken Spielfeldrand zu kommen:
  If iSpalteBrett > MAX_LINKS + 1 Then iSpalteBrett = iSpalteBrett - 1

  'Neue 5 Zellen fuer Brett mit Brettfarbe zeichnen:
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett - 1).Interior.Color = FARBE_BRETT
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 0).Interior.Color = FARBE_BRETT
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 1).Interior.Color = FARBE_BRETT
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 2).Interior.Color = FARBE_BRETT
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 3).Interior.Color = FARBE_BRETT
Else                                                'Brett ist 3 Zellen breit:
  'Die 3 aktuellen Zellen fuer Brett mit Hintergrundfarbe loeschen:
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 2).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 1).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 0).Interior.Color = FARBE_HINTERGRUND

  'Spaltenbegrenzung um nicht ueber den linken Spielfeldrand zu kommen:
  If iSpalteBrett > MAX_LINKS Then iSpalteBrett = iSpalteBrett - 1

  'Neue 3 Zellen fuer Brett mit Brettfarbe zeichnen:
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 0).Interior.Color = FARBE_BRETT
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 1).Interior.Color = FARBE_BRETT
  oTabellenblatt.Cells(iZeileBrett, iSpalteBrett + 2).Interior.Color = FARBE_BRETT
End If
                                                    'Wenn der Ball nicht im Spiel ist,
If bBallImSpiel = False Then                        'dann auch Ball eins nach links.
  oTabellenblatt.Cells(iZeileBrett - 1, iSpalteBrett + 2).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeileBrett - 1, iSpalteBrett + 1).Interior.Color = FARBE_BALL
  iSpalteBall = iSpalteBrett + 1                    'Neue Zeile und Spalte fuer
  iZeileBall = iZeileBrett - 1                      'Ballposition merken.
End If
Range("CURSOR_PARKPOSITION").Select                 'Cursor auf Parkposition setzen.


                                                    'Fehlerbehandlungsroutine:
Beenden:                                            '========================
  Application.Cursor = xlDefault                    'Sanduhr ausblenden.
  Exit Function                                     'Funktion abbrechen.

Fehler:                                             'Fehlermarke - Fehlerbehandlung:
  NachLinks = False                                 'Returnwert = Fehler setzen.
  MsgBox Err.Description                            'Fehlermeldung ausgeben.
  Resume Beenden                                    'Bei Funktionsende weitermachen.
  Resume                                            'Nur fuer Testzwecke hinterlegt.
End Function                                        'Funktionsende.
```

Speichern, schließen und öffnen Sie die Datei erneut. Testen Sie nun ob sich das Brett mit den vorgesehenen Tasten in der beabsichtigten Art und Weise bewegen läßt.
Sieht ja schon ganz gut aus… - oder ?

4.4.5.) Den Ball im Spielfeld bewegen

Dazu erweitern wir im Modul **Funktionen** die Funktion **BallBewegen**: Legen wir uns vorerst einmal nur die Struktur der Funktion an. Zuerst löschen wir die aktuelle Ballposition indem wir diese Zelle mit der Hintergrundfarbe füllen. Dann entscheiden wir anhand der aktuellen Bewegungsrichtung, auf welche neue Position der Ball zu bewegen ist. Dann kontrollieren wir, ob sich die neue Position noch im Spielfeld befindet – besser gesagt, wir schauen uns an, ob sich die neue Position am Spielfeldrand befindet. Wenn ja, dann machen wir einen Richtungswechsel je nachdem welcher Rand getroffen wurde.

```
Public Function BallBewegen()
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       BallBewegen                                      */
'*                                                                       */
'*      BESCHREIBUNG:   Bewegt den Ball um eine Position weiter. Aendert am */
'*                      Rand die Richtung. Trifft Ball auf Ziegel dann wird */
'*                      die Funktion zum Loeschen des Ziegels aufgerufen.  */
'*                                                                       */
'*      RETURN          nichts                                           */
'*                                                                       */
'*      PARAMETER:      keine                                            */
'*                                                                       */
'*-----------------------------------------------------------------------*/
On Error GoTo Fehler                                'Bei Fehler zur Fehlerbehandlung.
                                                    'Aktuellen Ball loeschen:
oTabellenblatt.Cells(iZeileBall, iSpalteBall).Interior.Color = FARBE_HINTERGRUND

Select Case stRichtung                              'Je nach Bewegungsrichtung:
  Case "RO"                                         'Ball bewegt sich nach: rechts oben
    iZeileBall = iZeileBall - 1                     'Neue Ballposition und
    iSpalteBall = iSpalteBall + 1                   'neue Bewegungsrichtung festlegen.
    If iZeileBall <= MAX_OBEN Then iZeileBall = MAX_OBEN: stRichtungNeu = "RU"
    If iSpalteBall >= MAX_RECHTS Then iSpalteBall = MAX_RECHTS: stRichtungNeu = "LO"

  Case "LO"                                         'Ball bewegt sich nach: links oben
    iZeileBall = iZeileBall - 1                     'Neue Ballposition und
    iSpalteBall = iSpalteBall - 1                   'neue Bewegungsrichtung festlegen.
    If iZeileBall <= MAX_OBEN Then iZeileBall = MAX_OBEN: stRichtungNeu = "LU"
    If iSpalteBall <= MAX_LINKS Then iSpalteBall = MAX_LINKS: stRichtungNeu = "RO"

  Case "RU"                                         'Ball bewegt sich nach: rechts unten
    iZeileBall = iZeileBall + 1                     'Neue Ballposition und
    iSpalteBall = iSpalteBall + 1                   'neue Bewegungsrichtung festlegen.
    If iZeileBall >= MAX_UNTEN Then iZeileBall = MAX_UNTEN: stRichtungNeu = "RO"
    If iSpalteBall >= MAX_RECHTS Then iSpalteBall = MAX_RECHTS: stRichtungNeu = "LU"

  Case "LU"                                         'Ball bewegt sich nach: links unten
    iZeileBall = iZeileBall + 1                     'Neue Ballposition und
    iSpalteBall = iSpalteBall - 1                   'neue Bewegungsrichtung festlegen.
    If iZeileBall >= MAX_UNTEN Then iZeileBall = MAX_UNTEN: stRichtungNeu = "LO"
    If iSpalteBall <= MAX_LINKS Then iSpalteBall = MAX_LINKS: stRichtungNeu = "RU"
End Select
                                                    'Ball auf neue Position setzen:
oTabellenblatt.Cells(iZeileBall, iSpalteBall).Interior.Color = FARBE_BALL
stRichtung = stRichtungNeu                          'Neue Richtung festlegen.
Range("CURSOR_PARKPOSITION").Select                 'Cursor auf Parkposition setzen.


                                                    'Fehlerbehandlungsroutine:
Beenden:                                            '=========================
  Application.Cursor = xlDefault                    'Sanduhr ausblenden.
  Exit Function                                     'Funktion abbrechen.
```

```
Fehler:                                         'Fehlermarke - Fehlerbehandlung:
  MsgBox Err.Description                        'Fehlermeldung ausgeben.
  Resume Beenden                                'Bei Funktionsende weitermachen.
  Resume                                        'Nur fuer Testzwecke hinterlegt.
End Function                                    'Funktionsende.
```

Speichern Sie die Datei und testen Sie die Bewegung des Balles. Sieht jetzt ja schon ganz toll aus (auch wenn der Ball noch über die Ziegel bewegt wird). Jetzt müssen wir natürlich am unteren Spielfeldrand noch abfragen, ob sich unter dem Ball auch das Brett befindet:

Diese Kontrolle benötigen wir nur dann, wenn sich der Ball nach unten bewegt, also bei den Richtungen **RU** und **LU**. Dabei gehen wir folgendermaßen vor: Zuerst berechnen wir, wie bereits erledigt, die neue Position des Balles. Befindet sich die neue Position in der Zeile des Brettes, dann kontrollieren wir, ob die neue Position die Hintergrundfarbe des Brettes hat. Wenn ja, dann machen wir die letzte Positionsänderung wieder rückgängig. Den Richtungswechsel selbst haben wir bereits im bisherigen Programmcode festgelegt.

Befindet sich jedoch kein Brett unter dem Ball, so ist der Ball aus dem Spiel. Die Anzahl der verfügbaren Bälle wir um eins verringert. Um die aktuelle Anzahl der Bälle im Tabellenblatt zu aktualisieren müssen wir zuerst den Blattschutz aufheben. Dann aktualisieren wir das Tabellenblatt und setzen abschließend den Blattschutz erneut.

Gleichzeitig können wir hier auch überprüfen ob das Spielende erreicht wurde – nämlich dann, wenn keine Bälle mehr im Spiel sind. In diesem Fall setzen wir eine Abbruchsbedingung (in der Funktion **Spielen** muß die Variable **Tastendruck** ungleich "" sein). Anhand des Inhaltes dieser Variable können wir in der Funktion **Spielen** auch eine genauere Abschlußmeldung ausgeben (ABBRECHEN bzw. SPIELENDE).

Sind noch Bälle im Spiel, dann rufen wir die Funktion **SpielStarten** mit dem Parameter **False** auf. Dabei wird ein neuer Ball ins Spiel gebracht. Der Ball wird dabei auf das Brett gelegt und das Brett samt Ball wird in der Mitte des Spielfeldes ausgegeben. Die aktuelle Anordnung der Ziegel bleibt erhalten – man kann also mit dem Spiel fortfahren ohne ganz neu von vorne beginnen zu müssen.

Nun die Umsetzung im Programmcode:

```
  Case "RU"                                     'Ball bewegt sich nach: rechts unten
    iZeileBall = iZeileBall + 1                 'Neue Ballposition und
    iSpalteBall = iSpalteBall + 1               'neue Bewegungsrichtung festlegen.
    If iZeileBall >= MAX_UNTEN Then iZeileBall = MAX_UNTEN: stRichtungNeu = "RO"
    If iSpalteBall >= MAX_RECHTS Then iSpalteBall = MAX_RECHTS: stRichtungNeu = "LU"

    'Ball trifft auf Brett oder nicht:
    If iZeileBall = MAX_UNTEN Then              'Ball in Zeile von Brett:
      If oTabellenblatt.Cells(iZeileBall, iSpalteBall).Interior.Color = FARBE_BRETT Then
        iZeileBall = iZeileBall - 1             'Brett ist unter Ball:
        iSpalteBall = iSpalteBall - 1           'Letzte Ballbewegung
      Else                                      'rueckgaengig machen.
        bBallImSpiel = False                    'Sonst ist Ball aus dem Spielfeld.
        iBaelle = iBaelle - 1                   'Anzahl der Baelle verringern.
        Call BlattschutzAufheben                'Blattschutz aufheben und
        oTabellenblatt.Range("BAELLE") = iBaelle 'im Tabellenblatt ausgeben.
                                                'Blattschutz wieder setzen.
        Call BlattschutzSetzen                  'Blattschutz setzen.
        If iBaelle > 0 Then                     'Sind noch Baelle im Spiel, dann
          Call SpielStarten(False)              'neuen Ball auf Brett legen.
        Else                                    'Wenn kein Ball mehr im Spiel:
          stTastendruck = "SPIELENDE"           'Abbruchsbedingung fuer
        End If                                  'Endlos-Schleife setzen.
      End If
    End If
```

```
  Case "LU"                                          'Ball bewegt sich nach: links unten
    iZeileBall = iZeileBall + 1                      'Neue Ballposition und
    iSpalteBall = iSpalteBall - 1                    'neue Bewegungsrichtung festlegen.
    If iZeileBall >= MAX_UNTEN Then iZeileBall = MAX_UNTEN: stRichtungNeu = "LO"
    If iSpalteBall <= MAX_LINKS Then iSpalteBall = MAX_LINKS: stRichtungNeu = "RU"

    'Ball trifft auf Brett oder nicht:
    If iZeileBall = MAX_UNTEN Then                   'Ball in Zeile von Brett:
      If oTabellenblatt.Cells(iZeileBall, iSpalteBall).Interior.Color = FARBE_BRETT Then
        iZeileBall = iZeileBall - 1                  'Brett ist unter Ball:
        iSpalteBall = iSpalteBall + 1                'Letzte Ballbewegung
      Else                                           'rueckgaengig machen.
        bBallImSpiel = False                         'Sonst ist Ball aus dem Spielfeld.
        iBaelle = iBaelle - 1                        'Anzahl der Baelle verringern.
        Call BlattschutzAufheben                     'Blattschutz aufheben und
        oTabellenblatt.Range("BAELLE") = iBaelle     'im Tabellenblatt ausgeben.
                                                     'Blattschutz wieder setzen.
        Call BlattschutzSetzen                       'Blattschutz setzen.
        If iBaelle > 0 Then                          'Sind noch Baelle im Spiel, dann
          Call SpielStarten(False)                   'neuen Ball auf Brett legen.
        Else                                         'Wenn kein Ball mehr im Spiel:
          stTastendruck = "SPIELENDE"                'Abbruchsbedingung fuer
        End If                                       'Endlos-Schleife setzen.
      End If
    End If
```

Jetzt fehlt uns nur noch ein Programmteil, nämlich das Entfernen eines Ziegels nachdem der Ball ihn getroffen hat. Dabei soll es egal sein, ob der Ball von unten, von oben oder von der Seite kommt (gemeint ist damit, die Eckpunkte von Ball und Ziegel treffen sich). Dazu werden wir anschließend noch die neue Funktion **ZiegelLoeschen** erstellen.

Grundsätzlich kann ein Ziegel aus allen Richtungen getroffen werden und nach jedem Treffer muß auch die Richtung des Balles geändert werden. Daher ist der anschließende Code bei allen Richtungen - mit entsprechender Adaptierung - zu erweitern. Hier der Programmcode nur für die Richtung **RO** (der vollständige Code für die Funktion BallBewegen siehe am Ende dieses Kapitels):

```
'Ziegel wurde getroffen:
If iZeileBall >= MAX_OBEN Then                   'Kontrolle ob die neue Position
                                                 'Teil eines Ziegels ist:
  If oTabellenblatt.Cells(iZeileBall, iSpalteBall).Interior.Color <> FARBE_HINTERGRUND Then
    Call ZiegelLoeschen(iZeileBall, iSpalteBall) 'Wenn ja, dann Ziegel entfernen
    stRichtungNeu = "RU"                         'und die Richtung aendern.
  End If
End If                                           'Zusatzkontrolle falls Ziegel
If iZeileBall > MAX_OBEN Then                    'von der Seite getroffen wird:
  If oTabellenblatt.Cells(iZeileBall - 1, iSpalteBall).Interior.Color <> FARBE_HINTERGRUND Then
    Call ZiegelLoeschen(iZeileBall - 1, iSpalteBall) 'Wenn ja, Ziegel entfernen
    stRichtungNeu = "RU"                         'und die Richtung aendern.
  End If
End If
```

Bevor wir das Programm testen können müssen wir noch die Funktion **ZiegelLoeschen** im Modul **Funktionen** erstellen. Diese Funktion ersetzt die Ziegelfarbe mit der Hintergrundfarbe anhand der neuen Position des Spielballes. Um die korrekten Zellen für einen Ziegel zu finden benutzen wir die Funktion **Mod** (Rest einer Division).

Gleichzeitig erhöhen wir in dieser Funktion die Punkteanzahl und kontrollieren, ob noch Ziegelsteine im aktuellen Level vorhanden sind. Wenn nicht, dann lassen wir den nächsten Level anzeigen und erhöhen die Geschwindigkeit des Balles indem wir die Wartezeit für das Bewegen des Balles verkleinern.

Sicherheitshalber begrenzen wir die kürzeste Wartezeit auf 5 Millisekunden. Vor der Aktualisierung der Punkte und des Levels muß natürlich der Blattschutz zuvor deaktiviert und danach wieder aktiviert werden.

Abschließend rufen wir die Funktion **SpielStarten** mit dem Parameter **True** auf; dabei wird der neue Level ins Spielfeld einkopiert. Der Ball wird dabei auf das Brett gelegt und das Brett samt Ball wird wieder in der Mitte des Spielfeldes ausgegeben.

Hier nun die fertige Funktion:

```
Public Function ZiegelLoeschen(iZeile As Integer, iSpalte As Integer)
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       ZiegelLoeschen                                   */
'*                                                                       */
'*      BESCHREIBUNG:   Loescht den Ziegel auf den der Ball getroffen ist*/
'*                      und erhoeht die Punkteanzahl.                    */
'*                                                                       */
'*      RETURN          nichts                                           */
'*                                                                       */
'*      PARAMETER:      Integer ......... Aktuelle Zeile des Balles      */
'*                      Integer ......... Aktuelle Spalte des Balles     */
'*                                                                       */
'*-----------------------------------------------------------------------*/
On Error GoTo Fehler                                'Bei Fehler zur Fehlerbehandlung.

If iSpalte Mod 3 = 0 Then                           'Ball trift letzte Zelle:
  oTabellenblatt.Cells(iZeile, iSpalte - 2).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeile, iSpalte - 1).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeile, iSpalte + 0).Interior.Color = FARBE_HINTERGRUND
ElseIf iSpalte Mod 3 = 1 Then                       'Ball trift erste Zelle:
  oTabellenblatt.Cells(iZeile, iSpalte + 0).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeile, iSpalte + 1).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeile, iSpalte + 2).Interior.Color = FARBE_HINTERGRUND
ElseIf iSpalte Mod 3 = 2 Then                       'Ball trift mittlere Zelle:
  oTabellenblatt.Cells(iZeile, iSpalte - 1).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeile, iSpalte + 0).Interior.Color = FARBE_HINTERGRUND
  oTabellenblatt.Cells(iZeile, iSpalte + 1).Interior.Color = FARBE_HINTERGRUND
End If                                              'Entsprechende Ziegelzellen
                                                    'mit Hintergrundfarbe loeschen.

Call BlattschutzAufheben                            'Blattschutz aufheben.
lPunkte = lPunkte + 10 * iLevelIst                  'Einfache Punktebewertung:

oTabellenblatt.Range("PUNKTE") = lPunkte            'Punkte ausgeben.
iAktBricks = iAktBricks - 1                         'Anzahl Ziegeln verringern.
If iAktBricks <= 0 Then                             'Keine Ziegeln mehr vorhanden:
  iLevelIst = iLevelIst + 1                         'Tatsaechlichen Level erhoehen.
  iLevel = iLevel + 1                               'Level erhoehen, falls max.
  If iLevel > MAX_LEVEL Then iLevel = 1             'Level ueberschritten, dann
                                                    'wieder mit Level 1 beginnen.
  oTabellenblatt.Range("LEVEL") = iLevelIst         'Geschwindigkeit erhoehen:
  iGeschwindigkeit = iGeschwindigkeit - GESCHWINDIGKEITSAENDERUNG 'jedoch nicht
  If iGeschwindigkeit < 5 Then iGeschwindigkeit = 5 'unter 5 Millisekunden.
  Call SpielStarten(True)                           'Neuen Level anzeigen.
                                                    'Blattschutz wird in Funktion
```

```
Else                                                    'SpielStarten wieder gesetzt.
  Call BlattschutzSetzen                                'Blattschutz wieder setzen.
End If

                                                        'Fehlerbehandlungsroutine:
Beenden:                                                '=========================
  Application.Cursor = xlDefault                        'Sanduhr ausblenden.
  Exit Function                                         'Funktion abbrechen.

Fehler:                                                 'Fehlermarke - Fehlerbehandlung:
  ZiegelLoeschen = False                                'Returnwert = Fehler setzen.
  MsgBox Err.Description                                'Fehlermeldung ausgeben.
  Resume Beenden                                        'Bei Funktionsende weitermachen.
  Resume                                                'Nur fuer Testzwecke hinterlegt.
End Function                                            'Funktionsende.
```

Jetzt aber, wie zuvor schon angesprochen der vollständige Programmcode der Funktion **BallBewegen**: In diesem Programmcode ist bereits die Hinterlegung von Sounds, welche wir erst im nächsten Kapitel abhandeln werden, enthalten. Diese Codezeilen sind gelb hinterlegt und werden später noch näher erklärt. Um das Spiel vorab zu testen zu können sind die gelb hinterlegten Programmzeilen vorab noch unter Kommentar zu setzen.

```
Public Function BallBewegen()
'*****************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                             */
'*****************************************************************************/
'*---------------------------------------------------------------------------*/
'*                                                                           */
'*      FUNKTION:       BallBewegen                                          */
'*                                                                           */
'*      BESCHREIBUNG:   Bewegt den Ball um eine Position weiter. Aendert am  */
'*                      Rand die Richtung. Trifft Ball auf Ziegel dann wird  */
'*                      die Funktion zum Loeschen des Ziegels aufgerufen.    */
'*                                                                           */
'*                      Trifft Ball nicht auf Brett, dann wird ein Ball      */
'*                      abgezogen. Sind keine Baelle mehr vorhanden, dann    */
'*                      ist das Spielende erreicht.                          */
'*                                                                           */
'*      RETURN          nichts                                               */
'*                                                                           */
'*      PARAMETER:      keine                                                */
'*                                                                           */
'*---------------------------------------------------------------------------*/
Dim bSoundWand          As Boolean                  'Hilfsvariable fuer Sound.
Dim stSoundAlias        As String                   'Wechen Sound abspielen.

On Error GoTo Fehler                                'Bei Fehler zur Fehlerbehandlung.
bSoundWand = False                                  'Sound Wand nicht ausgegeben.
stSoundAlias = ""                                   'Sound fuer Alias ruecksetzen.
                                                    'Aktuellen Ball loeschen:
oTabellenblatt.Cells(iZeileBall, iSpalteBall).Interior.Color = FARBE_HINTERGRUND

Select Case stRichtung                              'Je nach Bewegungsrichtung:
  Case "RO"                                         'Ball bewegt sich nach: rechts oben
    iZeileBall = iZeileBall - 1                     'Neue Ballposition und
    iSpalteBall = iSpalteBall + 1                   'neue Bewegungsrichtung festlegen.
    If iZeileBall <= MAX_OBEN Then                  'Ball geht ueber Spielfeld hinaus:
      iZeileBall = MAX_OBEN                         'Zeile fuer Ball eingrenzen.
      stRichtungNeu = "RU"                          'Richtungsaenderung machen.
      stSoundAlias = ALIAS_SOUND_WAND               'Sound fuer Wand festlegen.
    End If
    If iSpalteBall >= MAX_RECHTS Then               'Ball geht ueber Spielfeld hinaus:
      iSpalteBall = MAX_RECHTS                      'Spalte fuer Ball eingrenzen.
      stRichtungNeu = "LO"                          'Richtungsaenderung machen.
```

```
      stSoundAlias = ALIAS_SOUND_WAND                'Sound fuer Wand festlegen.
    End If

    'Kontrolle ob Ziegel getroffen wurde:           'Kontrolle ob die neue Position
    If iZeileBall >= MAX_OBEN Then                  'Teil eines Ziegels ist:
      If oTabellenblatt.Cells(iZeileBall, iSpalteBall).Interior.Color <> FARBE_HINTERGRUND
Then
        Call ZiegelLoeschen(iZeileBall, iSpalteBall) 'Wenn ja, dann Ziegel entfernen
        stRichtungNeu = "RU"                        'und die Richtung aendern.
        GoTo Fertig:                                'Restliches ueberspringen.
      End If
    End If                                          '1. Zusatzkontrolle falls Ziegel
    If iZeileBall > MAX_OBEN Then                   'von der Seite getroffen wird:
      If oTabellenblatt.Cells(iZeileBall - 1, iSpalteBall).Interior.Color <> FARBE_HINTERGRUND
Then
        Call ZiegelLoeschen(iZeileBall - 1, iSpalteBall) 'Wenn ja, Ziegel entfernen
        stRichtungNeu = "RU"                        'und die Richtung aendern.
        GoTo Fertig:                                'Restliches ueberspringen.
      End If
    End If                                          '2. Zusatzkontrolle falls Ziegel
    If iSpalteBall < MAX_RECHTS Then                'von der Seite getroffen wird:
      If oTabellenblatt.Cells(iZeileBall, iSpalteBall + 1).Interior.Color <> FARBE_HINTERGRUND
Then
        Call ZiegelLoeschen(iZeileBall, iSpalteBall + 1) 'Wenn ja, Ziegel entfernen
        stRichtungNeu = "LO"                        'und die Richtung aendern.
        GoTo Fertig:                                'Restliches ueberspringen.
      End If
    End If

  Case "LO"                                         'Ball bewegt sich nach: links oben
    iZeileBall = iZeileBall - 1                     'Neue Ballposition und
    iSpalteBall = iSpalteBall - 1                   'neue Bewegungsrichtung festlegen.
    If iZeileBall <= MAX_OBEN Then                  'Ball geht ueber Spielfeld hinaus:
      iZeileBall = MAX_OBEN                         'Zeile fuer Ball eingrenzen.
      stRichtungNeu = "LU"                          'Richtungsaenderung machen.
      stSoundAlias = ALIAS_SOUND_WAND               'Sound fuer Wand festlegen.
    End If
    If iSpalteBall <= MAX_LINKS Then                'Ball geht ueber Spielfeld hinaus:
      iSpalteBall = MAX_LINKS                       'Spalte fuer Ball eingrenzen.
      stRichtungNeu = "RO"                          'Richtungsaenderung machen.
      stSoundAlias = ALIAS_SOUND_WAND               'Sound fuer Wand festlegen.
    End If

    'Kontrolle ob Ziegel getroffen wurde:           'Kontrolle ob die neue Position
    If iZeileBall >= MAX_OBEN Then                  'Teil eines Ziegels ist:
      If oTabellenblatt.Cells(iZeileBall, iSpalteBall).Interior.Color <> FARBE_HINTERGRUND
Then
        Call ZiegelLoeschen(iZeileBall, iSpalteBall) 'Wenn ja, dann Ziegel entfernen
        stRichtungNeu = "LU"                        'und die Richtung aendern.
        GoTo Fertig:                                'Restliches ueberspringen.
      End If
    End If                                          '1. Zusatzkontrolle falls Ziegel
    If iZeileBall > MAX_OBEN Then                   'von der Seite getroffen wird:
      If oTabellenblatt.Cells(iZeileBall - 1, iSpalteBall).Interior.Color <> FARBE_HINTERGRUND
Then
        Call ZiegelLoeschen(iZeileBall - 1, iSpalteBall) 'Wenn ja, Ziegel entfernen
        stRichtungNeu = "LU"                        'und die Richtung aendern.
        GoTo Fertig:                                'Restliches ueberspringen.
      End If
    End If                                          '2. Zusatzkontrolle falls Ziegel
    If iSpalteBall > MAX_LINKS Then                 'von der Seite getroffen wird:
      If oTabellenblatt.Cells(iZeileBall, iSpalteBall - 1).Interior.Color <> FARBE_HINTERGRUND
Then
        Call ZiegelLoeschen(iZeileBall, iSpalteBall - 1) 'Wenn ja, Ziegel entfernen
        stRichtungNeu = "RO"                        'und die Richtung aendern.
        GoTo Fertig:                                'Restliches ueberspringen.
      End If
```

```
    End If


  Case "RU"                                       'Ball bewegt sich nach: rechts unten
    iZeileBall = iZeileBall + 1                   'Neue Ballposition und
    iSpalteBall = iSpalteBall + 1                 'neue Bewegungsrichtung festlegen.
    If iZeileBall >= MAX_UNTEN Then               'Ball geht ueber Spielfeld hinaus:
      iZeileBall = MAX_UNTEN                      'Zeile fuer Ball eingrenzen.
      stRichtungNeu = "RO"                        'Richtungsaenderung machen.
      bSoundWand = True                           'Sound Wand ausgeben.
    End If
    If iSpalteBall >= MAX_RECHTS Then             'Ball geht ueber Spielfeld hinaus:
      iSpalteBall = MAX_RECHTS                    'Spalte fuer Ball eingrenzen.
      stRichtungNeu = "LU"                        'Richtungsaenderung machen.
      bSoundWand = True                           'Sound Wand ausgeben.
    End If

    'Kontrolle ob der Ball auf das Brett trifft oder nicht:
    If iZeileBall = MAX_UNTEN Then                'Ball in Zeile von Brett:
      If oTabellenblatt.Cells(iZeileBall, iSpalteBall).Interior.Color = FARBE_BRETT Then
        iZeileBall = iZeileBall - 1               'Brett ist unter Ball:
        iSpalteBall = iSpalteBall - 1             'Letzte Ballbewegung
        stSoundAlias = ALIAS_SOUND_BRETT          'rueckgaengig machen und
      Else                                        'Sound abspielen.
        stSoundAlias = ALIAS_SOUND_ENDE           'Anderen Sound abspielen.
        bBallImSpiel = False                      'Sonst ist Ball aus dem Spielfeld.
        iBaelle = iBaelle - 1                     'Anzahl der Baelle verringern,
        Call BlattschutzAufheben                  'Blattschutz aufheben und
        oTabellenblatt.Range("BAELLE") = iBaelle  'im Tabellenblatt ausgeben.
        Call BlattschutzSetzen                    'Blattschutz wieder setzen.
        If iBaelle > 0 Then                       'Sind noch Baelle im Spiel, dann
          Call SpielStarten(False)                'neuen Ball auf Brett legen.
        Else                                      'Wenn kein Ball mehr im Spiel:
          stTastendruck = "SPIELENDE"             'Abbruchsbedingung fuer
        End If                                    'Endlos-Schleife setzen.
      End If
    Else                                          'Sound fuer Wand festlegen.
      If bSoundWand = True Then stSoundAlias = ALIAS_SOUND_WAND
    End If

    'Kontrolle ob Ziegel getroffen wurde:         'Kontrolle ob die neue Position
    If iZeileBall <= MAX_UNTEN - 3 Then           'Teil eines Ziegels ist:
      If oTabellenblatt.Cells(iZeileBall, iSpalteBall).Interior.Color <> FARBE_HINTERGRUND
Then
        Call ZiegelLoeschen(iZeileBall, iSpalteBall)  'Wenn ja, Ziegel entfernen
        stRichtungNeu = "RO"                      'und die Richtung aendern.
        GoTo Fertig:                              'Restliches ueberspringen.
      End If
    End If                                        '1. Zusatzkontrolle falls Ziegel
    If iZeileBall < MAX_UNTEN - 3 Then            'von der Seite getroffen wird:
      If oTabellenblatt.Cells(iZeileBall + 1, iSpalteBall).Interior.Color <> FARBE_HINTERGRUND
Then
        Call ZiegelLoeschen(iZeileBall + 1, iSpalteBall) 'Wenn ja, Ziegel entfernen
        stRichtungNeu = "RO"                      'und die Richtung aendern.
        GoTo Fertig:                              'Restliches ueberspringen.
      End If
    End If                                        '2. Zusatzkontrolle falls Ziegel
    If iSpalteBall < MAX_RECHTS Then              'von der Seite getroffen wird:
      If oTabellenblatt.Cells(iZeileBall, iSpalteBall + 1).Interior.Color <> FARBE_HINTERGRUND
Then
        Call ZiegelLoeschen(iZeileBall, iSpalteBall + 1) 'Wenn ja, Ziegel entfernen
        stRichtungNeu = "LU"                      'und die Richtung aendern.
        GoTo Fertig:                              'Restliches ueberspringen.
      End If
    End If
```

```
  Case "LU"                                       'Ball bewegt sich nach: links unten
    iZeileBall = iZeileBall + 1                   'Neue Ballposition und
    iSpalteBall = iSpalteBall - 1                 'neue Bewegungsrichtung festlegen.
    If iZeileBall >= MAX_UNTEN Then               'Ball geht ueber Spielfeld hinaus:
      iZeileBall = MAX_UNTEN                      'Zeile fuer Ball eingrenzen.
      stRichtungNeu = "LO"                        'Richtungsaenderung machen.
      bSoundWand = True                           'Sound Wand ausgeben.
    End If
    If iSpalteBall <= MAX_LINKS Then              'Ball geht ueber Spielfeld hinaus:
      iSpalteBall = MAX_LINKS                     'Spalte fuer Ball eingrenzen.
      stRichtungNeu = "RU"                        'Richtungsaenderung machen.
      bSoundWand = True                           'Sound Wand ausgeben.
    End If

    'Kontrolle ob der Ball auf das Brett trifft oder nicht:
    If iZeileBall = MAX_UNTEN Then                'Ball in Zeile von Brett:
      If oTabellenblatt.Cells(iZeileBall, iSpalteBall).Interior.Color = FARBE_BRETT Then
        iZeileBall = iZeileBall - 1               'Brett ist unter Ball:
        iSpalteBall = iSpalteBall + 1             'Letzte Ballbewegung
        stSoundAlias = ALIAS_SOUND_BRETT          'rueckgaengig machen und
      Else                                        'Sound abspielen.
        stSoundAlias = ALIAS_SOUND_ENDE           'rueckgaengig machen und
        bBallImSpiel = False                      'Sonst ist Ball aus dem Spielfeld.
        iBaelle = iBaelle - 1                     'Anzahl der Baelle verringern,
        Call BlattschutzAufheben                  'Blattschutz aufheben und
        oTabellenblatt.Range("BAELLE") = iBaelle  'im Tabellenblatt ausgeben.
        Call BlattschutzSetzen                    'Blattschutz wieder setzen.
        If iBaelle > 0 Then                       'Sind noch Baelle im Spiel, dann
          Call SpielStarten(False)                'neuen Ball auf Brett legen.
        Else                                      'Wenn kein Ball mehr im Spiel:
          stTastendruck = "SPIELENDE"             'Abbruchsbedingung fuer
        End If                                    'Endlos-Schleife setzen.
      End If
    Else                                          'Sound fuer Wand festlegen.
      If bSoundWand = True Then stSoundAlias = ALIAS_SOUND_WAND
    End If

    'Kontrolle ob Ziegel getroffen wurde:         'Kontrolle ob die neue Position
    If iZeileBall <= MAX_UNTEN - 3 Then           'Teil eines Ziegels ist:
      If oTabellenblatt.Cells(iZeileBall, iSpalteBall).Interior.Color <> FARBE_HINTERGRUND
Then
        Call ZiegelLoeschen(iZeileBall, iSpalteBall)  'Wenn ja, Ziegel entfernen
        stRichtungNeu = "LO"                      'und die Richtung aendern.
        GoTo Fertig:                              'Restliches ueberspringen.
      End If
    End If                                        '1. Zusatzkontrolle falls Ziegel
    If iZeileBall < MAX_UNTEN - 3 Then            'von der Seite getroffen wird:
      If oTabellenblatt.Cells(iZeileBall + 1, iSpalteBall).Interior.Color <> FARBE_HINTERGRUND
Then
        Call ZiegelLoeschen(iZeileBall + 1, iSpalteBall) 'Wenn ja, Ziegel entfernen
        stRichtungNeu = "LO"                      'und die Richtung aendern.
        GoTo Fertig:                              'Restliches ueberspringen.
      End If
    End If                                        '2. Zusatzkontrolle falls Ziegel
    If iSpalteBall > MAX_LINKS Then               'von der Seite getroffen wird:
      If oTabellenblatt.Cells(iZeileBall, iSpalteBall - 1).Interior.Color <> FARBE_HINTERGRUND
Then
        Call ZiegelLoeschen(iZeileBall, iSpalteBall - 1) 'Wenn ja, Ziegel entfernen
        stRichtungNeu = "RU"                      'und die Richtung aendern.
        GoTo Fertig:                              'Restliches ueberspringen.
      End If
    End If
End Select


Fertig:                                           'Zum Schluss noch den Ball auf
                                                  'die neue Position setzen und
```

```
oTabellenblatt.Cells(iZeileBall, iSpalteBall).Interior.Color = FARBE_BALL
stRichtung = stRichtungNeu                          'die neue Richtung festlegen.
Range("CURSOR_PARKPOSITION").Select                 'Cursor auf Parkposition setzen.

If stSoundAlias <> "" Then Call SoundAbspielen(stSoundAlias) 'Sound wiedergeben.

                                                    'Fehlerbehandlungsroutine:
Beenden:                                            '=========================
  Application.Cursor = xlDefault                    'Sanduhr ausblenden.
  Exit Function                                     'Funktion abbrechen.

Fehler:                                             'Fehlermarke - Fehlerbehandlung:
  MsgBox Err.Description                            'Fehlermeldung ausgeben.
  Resume Beenden                                    'Bei Funktionsende weitermachen.
  Resume                                            'Nur fuer Testzwecke hinterlegt.
End Function                                        'Funktionsende.
```

4.4.5.) Das Spiel mit Sounds hinterlegen

Die Hinterlegung von Sounds haben wir in der Übungseinheit #2 bereits im Detail abgehandelt. Daher wird hier nur mehr das Wichtigste, das Spiel Bricks betreffend, abgehandelt.

Zuerst müssen wir uns überlegen bei welchen Aktionen ein Sound wiedergegeben werden soll und vor allem welcher Sound. Bei Bricks würde ich folgendes vorschlagen:

Hintergrundmusik: Hintergrundmusik.wav.
Wenn Ball auf das Brett kommt – ein verzerrtes Piepsen – AufBrett.mp3
Wenn Ball auf einen Ziegel trifft – ein dumpfes Piepsen – AufBrick.mp3
Wenn Ball an die Wand schlägt – ein helles Piepsen – AufWand.mp3
Wenn Ball das Brett verfehlt – ein tiefer, längerer Piepston – AusSpiel.mp3

Die Piepstöne habe ich aus dem Internet, von einer Seite welche hunderte Piepstöne zur Verfügung stellt, heruntergeladen. Gefunden habe ich die Seite bei der Suche nach: Piepstöne.

Nun ergänzen wir im Modul **Globales** die Variablen und Konstanten für die Sounds:

```
'------------------------------------------------------------------------------/
'-  Globale Konstanten und Variablen - fuer Hintergrundmusik und Sounds     -/
'------------------------------------------------------------------------------/
Global bSoundsAbspielen           As Boolean  'Sounds abspielen lassen.
Global bSoundEndeOK               As Boolean  'Sound AusSpiel kann abgespielt werden.
Global bSoundBrickOK              As Boolean  'Sound AufBrick kann abgespielt werden.
Global bSoundBrettOK              As Boolean  'Sound AufBrett kann abgespielt werden.
Global bSoundWandOK               As Boolean  'Sound AufWand kann abgespielt werden.
Global stDateiHintergrundMusik    As String   'Akt. Dateiname fuer Hintergrundmusik.

Global Const DATEI_SOUND_ENDE = "\AusSpiel.mp3"
Global Const DATEI_SOUND_BRICK = "\AufBrick.mp3"
Global Const DATEI_SOUND_BRETT = "\AufBrett.mp3"
Global Const DATEI_SOUND_WAND = "\AufWand.mp3"
Global Const DATEI_SOUND_HINTERGRUND = "\Hintergrundmusik.wav"

Global Const ALIAS_SOUND_ENDE = "Alias_Sound_Ende"
Global Const ALIAS_SOUND_BRICK = "Alias_Sound_Brick"
Global Const ALIAS_SOUND_BRETT = "Alias_Sound_Brett"
Global Const ALIAS_SOUND_WAND = "Alias_Sound_Wand"
Global Const ALIAS_SOUND_HINTERGRUND = "Alias_Hintergrundmusik"
```

Dann behandeln wir die Schaltfläche **Ton EIN**. Dazu ergänzen wir die Prozedur **Workbook_Open** in **DieseArbeitsmappe** mit folgendem Programmcode:

```
'*-------------------------------------------------------------------------*/
Set oTabellenblatt = Sheets(TABELLENBLATT)          'Akt. Tabellenblatt festlegen.
On Error Resume Next                                'Bei Fehler einfach weitermachen,
                                                    'da nicht alle Shapes die
                                                    'Eigenschaft TextFrame haben.
For i = 0 To oTabellenblatt.Shapes.Count            'Alle Shapes (Frames) durchsuchen:
                                                    'Sollte angezeigt werden, den
  If oTabellenblatt.Shapes(i).TextFrame.Characters.Text = "Ton AUS" Then 'Ton NICHT
      oTabellenblatt.Shapes(i).TextFrame.Characters.Text = "Ton EIN" 'wiedergeben,
  End If                                            'dann Text umaendern, damit der
Next i                                              'Spieler Ton selbst aktivieren kann.
On Error GoTo Fehler                                'Bei Fehler wieder zur Fehlermarke.
bSoundsAbspielen = False                            'Musikwiedergabe deaktivieren.

lPunkte = 0                                         'Punkteanzahl 0 setzen.
Call SpielStarten(True)                             'Startprogramm aufrufen.
'*-------------------------------------------------------------------------*/
```

Anschließend erstellen wir im Modul **Makros** das Makro **Ton_Umschalten** welches wir der Schaltfläche zuweisen (sollte eigentlich bereits in der Mustervorlage hinterlegt sein, wenn nicht – dann jetzt ergänzen):

```
Public Sub Ton_Umschalten()
'*************************************************************************/
'*  Makro wird ausgefuehrt bei Klick auf Schaltflaeche: Ton AUS bzw. Ton EIN*/
'*************************************************************************/
Call TonUmschalten
End Sub                                                 'Prozedurende.
```

Genauso verfahren wir auch mit der Funktion **TonUmschalten** im Modul Sounds:

```
Public Function TonUmschalten()
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-------------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       TonUmschalten                                    */
'*                                                                       */
'*      BESCHREIBUNG:   Schaltet die Wiedergabe von Hintergrundmusik und */
'*                      Geraeuschen aus bzw. ein.                        */
'*                                                                       */
'*      RETURN          nichts                                           */
'*                                                                       */
'*      PARAMETER:      keine                                            */
'*                                                                       */
'*-------------------------------------------------------------------------*/
Dim i                   As Integer                  'Allgemeine Zaehlvariable.

Set oTabellenblatt = Sheets(TABELLENBLATT)          'Akt. Tabellenblatt festlegen.
oTabellenblatt.Select                               'Tabellenblatt auswaehlen.

'Sollte ein Shape (Form) nicht die Eigenschaft: TextFrame.Characters.Text
'haben, dann tritt ein Laufdzeitfehler auf. Um dieses zu verhindern wird
'der Programmcode bei einem Fehler in der naechsten Programmzeile fortgesetzt.
'Genaugenommen wird das naechste Shape abgehandelt.

On Error Resume Next                                'Bei Fehler einfach weitermachen:
For i = 0 To oTabellenblatt.Shapes.Count            'Alle Formen im Tabellenblatt
```

```
  Err = 0                                              'durcharbeiten. Fehler 0 setzen.
  If oTabellenblatt.Shapes(i).TextFrame.Characters.Text = "Ton AUS" Then
    If Err = 0 Then                                    'Wenn Ton ausschalten:
      bSoundsAbspielen = False                         'Merken: Keine Tonwiedergabe.
      Call SpielSoundsRuecksetzen                      'Alle Sounds zuruecksetzen.
      Call HintergrundMusik_AUS                        'Hintergrundmusik stoppen.
      oTabellenblatt.Shapes(i).TextFrame.Characters.Text = "Ton EIN"
      Exit For                                         'Beschriftung aendern und
    End If                                             'Funktion beenden.

  ElseIf oTabellenblatt.Shapes(i).TextFrame.Characters.Text = "Ton EIN" Then
    If Err = 0 Then                                    'Wenn Ton einschalten:
      bSoundsAbspielen = True                          'Merken: Ton wiedergeben.
      Call SpielSoundsDefinieren                       'Alle Sounds neu definieren.
      Call HintergrundMusik_EIN                        'Hintergrundmusik starten.
      oTabellenblatt.Shapes(i).TextFrame.Characters.Text = "Ton AUS"
      Exit For                                         'Beschriftung aendern und
    End If                                             'Funktion beenden.

  Else                                                 'Reserve fuer Erweiterungen:
    'nix machen
  End If
Next i
End Function                                           'Funktionsende.
```

Als nächstes überarbeiten wir die Funktionen **SpielSoundsDefinieren**, **HintergrundMusik_EIN**, **HintergrundMusik_AUS**, **SoundAbspielen** und **SpielSoundsRuecksetzen** im Modul **Sounds**:

```
Public Function SpielSoundsDefinieren() As Boolean
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       SpielSoundsDefinieren                            */
'*                                                                       */
'*      BESCHREIBUNG:   Definiert die Aliasnamen von MP3-Sounddateien.   */
'*                                                                       */
'*      RETURN          Boolean ..... True = Funktion in Ordnung         */
'*                                    False = Fehler in Funktion         */
'*      PARAMETER:      keine                                            */
'*                                                                       */
'*-----------------------------------------------------------------------*/
Dim stDateiname         As String                  'Sound-Datei.
Dim stAlias             As String                  'Aliasname fuer Sound-Datei.
Dim stBuffer            As String                  'Buffer fuer DOS 8.3 Dateinamen.

On Error GoTo Fehler                               'Bei Fehler zur Fehlerbehandlung.
SpielSoundsDefinieren = True                       'Returnwert = alles OK setzen.
Call SpielSoundsRuecksetzen                        'Zuerst alles ruecksetzen.

'*-----------------------------------------------------------------------*/
'*- Sound nachdem der Ball aus dem Spielfeld ist                        -*/
'*-----------------------------------------------------------------------*/
stDateiname = ActiveWorkbook.Path & DATEI_SOUND_ENDE 'Dateiname für Sound und
stAlias = ALIAS_SOUND_ENDE                         'Namen fuer Alias festlegen.
stBuffer = Space$(255)                             'DOS 8.3 Dateinamen ermitteln:
If GetShortPathName(stDateiname, stBuffer, Len(stBuffer)) <> 0 Then
  stDateiname = Left$(stBuffer, InStr(stBuffer, vbNullChar) - 1)
End If                                             'Leerzeichen am Ende entfernen.
If PlayMP3Sound("open " & stDateiname & " type MPEGVideo alias " & stAlias, 0, 0, 0) = 0 Then
  bSoundEndeOK = True                              'Sound kann abgespielt werden.
Else
  bSoundEndeOK = False                             'Sound nicht abspielbar.
End If
```

```
'*-----------------------------------------------------------------------*/
'*- Sound nachdem der Ball auf einen Ziegel stoesst                     -*/
'*-----------------------------------------------------------------------*/
stDateiname = ActiveWorkbook.Path & DATEI_SOUND_BRICK 'Dateiname für Sound und
stAlias = ALIAS_SOUND_BRICK                           'Namen fuer Alias festlegen.
stBuffer = Space$(255)                                'DOS 8.3 Dateinamen ermitteln:
If GetShortPathName(stDateiname, stBuffer, Len(stBuffer)) <> 0 Then
  stDateiname = Left$(stBuffer, InStr(stBuffer, vbNullChar) - 1)
End If                                                'Leerzeichen am Ende entfernen.
If PlayMP3Sound("open " & stDateiname & " type MPEGVideo alias " & stAlias, 0, 0, 0) = 0 Then
  bSoundBrickOK = True                                'Sound kann abgespielt werden.
Else
  bSoundBrickOK = False                               'Sound nicht abspielbar.
End If

'*-----------------------------------------------------------------------*/
'*- Sound nachdem der Ball auf das Brett kommt                          -*/
'*-----------------------------------------------------------------------*/
stDateiname = ActiveWorkbook.Path & DATEI_SOUND_BRETT 'Dateiname für Sound und
stAlias = ALIAS_SOUND_BRETT                           'Namen fuer Alias festlegen.
stBuffer = Space$(255)                                'DOS 8.3 Dateinamen ermitteln:
If GetShortPathName(stDateiname, stBuffer, Len(stBuffer)) <> 0 Then
  stDateiname = Left$(stBuffer, InStr(stBuffer, vbNullChar) - 1)
End If                                                'Leerzeichen am Ende entfernen.
If PlayMP3Sound("open " & stDateiname & " type MPEGVideo alias " & stAlias, 0, 0, 0) = 0 Then
  bSoundBrettOK = True                                'Sound kann abgespielt werden.
Else
  bSoundBrettOK = False                               'Sound nicht abspielbar.
End If

'*-----------------------------------------------------------------------*/
'*- Sound nachdem der Ball an die Wand trift                            -*/
'*-----------------------------------------------------------------------*/
stDateiname = ActiveWorkbook.Path & DATEI_SOUND_WAND 'Dateiname für Sound und
stAlias = ALIAS_SOUND_WAND                            'Namen fuer Alias festlegen.
stBuffer = Space$(255)                                'DOS 8.3 Dateinamen ermitteln:
If GetShortPathName(stDateiname, stBuffer, Len(stBuffer)) <> 0 Then
  stDateiname = Left$(stBuffer, InStr(stBuffer, vbNullChar) - 1)
End If                                                'Leerzeichen am Ende entfernen.
If PlayMP3Sound("open " & stDateiname & " type MPEGVideo alias " & stAlias, 0, 0, 0) = 0 Then
  bSoundWandOK = True                                 'Sound kann abgespielt werden.
Else
  bSoundWandOK = False                                'Sound nicht abspielbar.
End If

'*-----------------------------------------------------------------------*/
'*- Hintergrundmusik                                                    -*/
'*-----------------------------------------------------------------------*/
stDateiHintergrundMusik = ActiveWorkbook.Path & DATEI_SOUND_HINTERGRUND 'Dateinamen für
                                                      'Hintergrundmusik 2 festlegen.
stBuffer = Space$(255)                                'DOS 8.3 Dateinamen ermitteln:
If GetShortPathName(stDateiHintergrundMusik, stBuffer, Len(stBuffer)) <> 0 Then
  stDateiHintergrundMusik = Left$(stBuffer, InStr(stBuffer, vbNullChar) - 1)
Else                                                  'Die Leerzeichen am Ende entfernen.
  stDateiHintergrundMusik = ""                        'Hintergrundmusik nicht abspielbar.
End If
                                                      'Fehlerbehandlungsroutine:
Beenden:                                              '=========================
  Exit Function                                       'Funktion abbrechen.

Fehler:                                               'Fehlermarke - Fehlerbehandlung:
  SpielSoundsDefinieren = False                       'Returnwert = Fehler setzen.
  MsgBox Err.Description                              'Fehlermeldung ausgeben.
  Resume Beenden                                      'Bei Funktionsende weitermachen.
  Resume                                              'Nur fuer Testzwecke hinterlegt.
End Function                                          'Funktionsende.
```

```
Public Function HintergrundMusik_EIN()
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       HintergrundMusik_EIN                             */
'*                                                                       */
'*      BESCHREIBUNG:   Startet Wiedergabe der Hintergrundmusik in einer */
'*                      Endlos-Schleife.                                 */
'*                                                                       */
'*      RETURN          nichts                                           */
'*                                                                       */
'*      PARAMETER:      keine                                            */
'*                                                                       */
'*-----------------------------------------------------------------------*/
If bSoundsAbspielen = True And stDateiHintergrundMusik <> "" Then 'Wenn abgespielt
  Call PlayWaveSound(stDateiHintergrundMusik, SND_ASYNC Or SND_LOOP) 'werden soll und
End If                                                  'Datei fuer Wiedergabe vorhanden.
End Function                                            'Funktionsende.


Public Function HintergrundMusik_AUS()
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       HintergrundMusik_AUS                             */
'*                                                                       */
'*      BESCHREIBUNG:   Stoppt Wiedergabe der Hintergrundmusik.          */
'*                                                                       */
'*      RETURN          nichts                                           */
'*                                                                       */
'*      PARAMETER:      keine                                            */
'*                                                                       */
'*-----------------------------------------------------------------------*/
Call PlayWaveSound(0, SND_PURGE)                        'Wiedergabe sofort beenden.
End Function                                            'Funktionsende.


Public Function SoundAbspielen(stAliasname As String)
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       SoundAbspielen                                   */
'*                                                                       */
'*      BESCHREIBUNG:   Startet Wiedergabe der entsprechenden Datei zum  */
'*                      uebergebenen Alias-Namen.                        */
'*                                                                       */
'*      RETURN          nichts                                           */
'*                                                                       */
'*      PARAMETER:      String ..... Aliasname fuer abzuspielender       */
'*                                   Geraeusch.                          */
'*                                                                       */
'*-----------------------------------------------------------------------*/
If bSoundsAbspielen = True Then                         'Wenn abgespielt werden soll:
  Select Case stAliasname                               'Je nachdem welches Geraeusch:
    Case ALIAS_SOUND_ENDE                               'Wenn dazugehoerige Datei
      If bSoundEndeOK = True Then                       'abgespielt werden kann, dann
        Call PlayMP3Sound("play " & stAliasname & " from 0", 0, 0, 0)
      End If                                            'diese abspielen lassen.
```

```
    Case ALIAS_SOUND_BRICK                          'Wenn dazugehoerige Datei
      If bSoundBrickOK = True Then                  'abgespielt werden kann, dann
        Call PlayMP3Sound("play " & stAliasname & " from 0", 0, 0, 0)
      End If                                        'diese abspielen lassen.

    Case ALIAS_SOUND_BRETT                          'Wenn dazugehoerige Datei
      If bSoundBrettOK = True Then                  'abgespielt werden kann, dann
        Call PlayMP3Sound("play " & stAliasname & " from 0", 0, 0, 0)
      End If                                        'diese abspielen lassen.

    Case ALIAS_SOUND_WAND                           'Wenn dazugehoerige Datei
      If bSoundWandOK = True Then                   'abgespielt werden kann, dann
        Call PlayMP3Sound("play " & stAliasname & " from 0", 0, 0, 0)
      End If                                        'diese abspielen lassen.

    Case Else

  End Select
End If
End Function                                        'Funktionsende.


Public Function SpielSoundsRuecksetzen()
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       SpielSoundsRuecksetzen                           */
'*                                                                       */
'*      BESCHREIBUNG:   Stoppt alle Wiedergaben und schließt MCI´s.      */
'*                                                                       */
'*      RETURN          nichts                                           */
'*                                                                       */
'*      PARAMETER:      keine                                            */
'*                                                                       */
'*----------------------------------------------------------------------*/
PlayMP3Sound "stop " & ALIAS_SOUND_ENDE, 0, 0, 0    'Wiedergabe stoppen.
PlayMP3Sound "close " & ALIAS_SOUND_ENDE, 0, 0, 0   'MCI schliessen.

PlayMP3Sound "stop " & ALIAS_SOUND_BRICK, 0, 0, 0   'Wiedergabe stoppen.
PlayMP3Sound "close " & ALIAS_SOUND_BRICK, 0, 0, 0  'MCI schliessen.

PlayMP3Sound "stop " & ALIAS_SOUND_BRETT, 0, 0, 0   'Wiedergabe stoppen.
PlayMP3Sound "close " & ALIAS_SOUND_BRETT, 0, 0, 0  'MCI schliessen.

PlayMP3Sound "stop " & ALIAS_SOUND_WAND, 0, 0, 0    'Wiedergabe stoppen.
PlayMP3Sound "close " & ALIAS_SOUND_WAND, 0, 0, 0   'MCI schliessen.

End Function                                        'Funktionsende.
```

Zum Schluß binden wir die Piepstöne noch ins Spiel ein. Dazu fügen wir als erste Programmzeile in der Funktion **ZiegelLoeschen** folgende Codezeile ein:

```
Call SoundAbspielen(ALIAS_SOUND_BRICK)              'Sound abspielen.
```

Die weiteren Piepstöne werden in der Funktion BallBewegen abgespielt. Da 3 unterschiedliche Piepser abgespielt werden können, stellen wir zuerst fest, ob und welcher Piepston ausgegeben werden soll. Dazu speichern wir den Aliasnamen des Sounds in einer Variablen. Beinhaltet am Ende der Funktion die Variable einen Wert, dann wird genau dieser Sound abgespielt.

Die benötigten Ergänzungen wurden bereits im vorherigen Kapitel abgebildet und die betreffenden Programmzeilen gelb hervorgehoben. (Bei Bedarf dort nachschauen.)

Kontrollieren bzw. ergänzen Sie in der Prozedur **Workbook_BeforeClose** und **Workbook_Deactivate** ob die Hintergrundmusik auch wieder ausgeschaltet wird. Wenn nicht, dann ergänzen Sie folgenden Aufruf:

```
Call HintergrundMusik_AUS                    'Hintergrundmusik beenden.
```

Kontrollieren bzw. ergänzen Sie in der Prozedur **Workbook_Activate** ob die Hintergrundmusik wenn nötig auch wieder eingeschaltet wird:

```
If bSoundsAbspielen = True Then              'Wenn Soundwiedergabe aktiviert:
  Call HintergrundMusik_EIN                  'Hintergrundmusik wiedergeben.
End If
```

5.) Schlußbemerkungen

Für ein Spiel sollten Sie auch ein **Icon** erstellen, welches man einer Verknüpfung zuweisen kann. Zum Erstellen eigener Icons verwende ich den kostenlosen **Greenfish Icon Editor Pro**. Damit wurde auch das Icon für das Bricks-Spiel erstellt:

Bei komplexeren Spielen sollte auch eine detaillierte **Spielbeschreibung** erstellt werden.

Und das wär´s dann wieder Mal für heute – viel Spaß beim Nachprogrammieren oder Überarbeiten des Spieles.