Übungseinheit: # 1

Basisumgebung und Standardfunktionen

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

Ziel dieser Übung ist es eine Excel-Datei zu erstellen welche später als Kopiervorlage für neue Spiele verwendet werden kann.

2.) Vorbereitungen

Zuerst erstellen wir eine neue Excel-Datei und speichern sie in unserer Entwicklungsumgebung als Excel-Datei mit Makros (*.xlsm) ab.

Dann benennen wir das Tabellenblatt "**Tabelle1**" um z.B. auf "**Spielfeld**". Hat Excel automatisch weitere Tabellenblätter angelegt, dann können diese jetzt wieder gelöscht werden.

Als nächste wechseln wir in der Multifunktionsleiste von Excel zum Register: **Entwicklungstools**. (Sollte dieser Reiter rechts neben "**Ansicht**" nicht sichtbar sein, dann kann er über die Excel-Optionen eingeblendet werden: Excel 2007: Office-Schaltfläche > Excel Optionen > Häufig verwendet > rechts das 3. Kästchen von oben: "Entwicklerregistrierkarte in Multifunktionsleiste" anzeigen abhaken > OK. Spätere Excel Versionen: Datei > Optionen > Menüband anpassen > im rechten Teil das Kästchen für Entwicklertools abhaken > OK.)

Im Register **Entwicklungstools** Klick auf die Schaltfläche: **Visual Basic** (ganz links). Sollte in Visual Basic das Fenster "**Projekt Explorer**" nicht angezeigt werden, dann Klick auf Ansicht > Projekt-Explorer.

Nun legen wir uns vorab 4 leere Module an: Rechtsklick auf: Microsoft Excel Objekte > Einfügen > Modul. Danach benennen wir die Module um auf: "**Globales**", "**Funktionen**", "**Makros**" und "**Standard**". Um die Module umbenennen zu können muß das Eigenschaftenfenster eingeblendet sein. Ansicht > Eigenschaftenfenster (oder **F4**). Dort kann nun rechts neben (Name) die Bezeichnung "**Modul1**" entsprechend umbenannt werden.

Globales: Hier werden später allgemein gültige Variablen und deklarierte Funktionen hinterlegt.
Funktionen: Hier werden die speziellen Funktionen für das Spiel abgelegt.
Makros: Aufruf von Funktionen die später aus dem Tabellenblatt aufgerufen werden können.
Standard: Funktionen die auch für zukünftige Spiele oder Programme Verwendung finden können.

Zum Schluß sollte noch kontrolliert werden ob folgende Verweise auf Objektbibliotheken vorhanden sind:
Visual Basic for Applications
Microsoft Excel 12.0 Objects Library
Microsoft Office 12.0 Objects Library
OLE Automation

Dazu klicken wir auf Extras > Verweise und kontrollieren ob die Häkchen bei den benötigten Verweisen gesetzt sind. Wenn nicht, dann in der Liste den entsprechenden Verweis suchen und abhaken. Stehen für eine Objektbibliothek mehrere Versionen zur Verfügung, so sollte man die früheste Version auswählen.

Wer später seinen Programmcode ordentlich dokumentieren will (was man grundsätzlich immer machen sollte) kann den Dokumentationskopf aus Anhang 1 in die Module einkopieren und entsprechend anpassen (auch Einkopieren in die Excel-Objekte "DieseArbeitsmappe" und "Tabelle1 (Spielfeld)".

Damit hätten wir die Vorbereitungen abgeschlossen und können zum nächsten Schritt gehen.
Siehe auch Datei: Mustervorlage_01_Vorbereitungen.xlsm

3.) Musterfunktion

Als erstes basteln wir uns nun eine Musterfunktion die uns später als Kopiervorlage für weitere Funktionen dienen soll. Zunächst möchten wir aber sicherstellen, daß wir beim Programmieren nicht über irrtümliche Tippfehler stolpern die beim Testen des fertigen Programms dann kaum zu finden sind:

Dazu verwenden wir die Anweisung:

```
Option Explicit                                 'Alle Variablen muessen
                                                'explizit angegeben werden.
```

Diese wird in allen Modulen, DieseArbeitsmappe und allen weiteren Excel-Objekten als erste Programmzeile bzw. nach dem Dokumentationskopf eingefügt.

Nun wieder zur Musterfunktion. Diese sollte im Modul Standard abgelegt werden. Dazu selektieren wir das Modul Standard und klicken dann auf Einfügen > Prozedur. Bei **Name** geben wir "**Musterfunktion**", als **Typ**: **Function** und als **Gültigkeitsbereich**: **Public** ein. Damit kann die Funktion auch aus allen anderen Modulen aufgerufen werden. **Private** erlaubt nur Aufrufe aus jenem Modul, wo auch die Funktion abgelegt ist – hier also aus dem Modul **Standard**.

```
Public Function Musterfunktion()

End Function
```

Jede Funktion sollte auch einen Rückgabewert liefern. Wenn nichts Spezielles benötigt wird, dann sollte man die Funktion als Boolean definieren damit die Funktion zumindest zurückmelden kann ob sie korrekt ausgeführt wurde oder nicht. Und - genauso wie die Module sollte auch jede einzelne Funktion dokumentiert werden. Kopiervorlagen dazu siehe im Anhang 2.

```
Public Function Musterfunktion() As Boolean
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*----------------------------------------------------------------------*/
'*                                                                      */
'*      FUNKTION:       Musterfunktion                                  */
'*                                                                      */
'*      BESCHREIBUNG:   Dient als Kopiervorlage fuer weitere Funktionen.*/
'*                                                                      */
'*      RETURN          Boolean ......... True = Funktion in Ordnung    */
'*                                       False = Fehler in Funktion     */
'*      PARAMETER:      keine                                           */
'*                                                                      */
'*----------------------------------------------------------------------*/

End Function                                    'Funktionsende.
```

Jede Funktion sollte auch eine Fehlerbehandlungsroutine besitzen damit das Programm nicht willkürlich abstürzt. Grundsätzlich empfiehlt sich folgende Vorgangsweise: Der Funktion gleich zu Beginn mitteilen, wohin sie verzweigen soll, wenn ein Fehler aufgetreten ist. Dann vorab festlegen, daß die Funktion ohne Fehler ausgeführt wurde. Tritt ein Fehler auf, so wird in die Fehlerbehandlungsroutine zur Fehlermarke gesprungen und dort wird der Rückgabewert "**Funktion nicht OK**" gesetzt. Nach Ausgabe einer entsprechenden Fehlermeldung wird die Funktion dann beendet. Der Code dafür sieht so aus:

```
Public Function Musterfunktion() As Boolean
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*----------------------------------------------------------------------*/
'*                                                                      */
'*      FUNKTION:       Musterfunktion                                  */
'*                                                                      */
'*      BESCHREIBUNG:   Dient als Kopiervorlage fuer weitere Funktionen.*/
'*                                                                      */
'*      RETURN          Boolean ..... True = Funktion in Ordnung        */
'*                                    False = Fehler in Funktion        */
'*      PARAMETER:      keine                                           */
'*                                                                      */
'*----------------------------------------------------------------------*/
On Error GoTo Fehler                            'Bei Fehler zur Fehlermarke.
Musterfunktion = True                           'Returnwert = alles OK setzen.

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

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

Häufig wird bei Programmabläufen die Sanduhr eingeblendet. Daher empfehle ich vor dem Beenden der Funktion die Sanduhr wieder ausblenden zu lassen. In Ausnahmefällen kann diese Zeile dann unter Kommentar gestellt werden.

Die 2. Zeile **Resume** ist bei der Fehlersuche sehr hilfreich. Man setzt einen Haltepunkt auf die Zeile: **Resume Beenden**. Beim Auftreten eines Fehlers stoppt hier die Programmausführung. Dann kann man hier gleich die Resume-Zeile auswählen und als nächste Anweisung festlegen. Das Programm wird dann bei jener Programmzeile fortgesetzt in der auch der Fehler aufgetreten ist.

Um der Musterfunktion auch noch einen Sinn zu geben erweitern wir sie noch um einige Befehle, welche sehr oft in anderen Funktionen benötigt werden:

Häufig verwendete Variablen:

```
Dim i                   As Integer          'Allgemeine Zaehlvariable.
Dim iSpalte             As Integer          'Aktuelle Spaltennummer.
Dim lZeile              As Long             'Aktuelle Zeilennummer.
Dim stBereich           As String           'Aktueller Bereich.
Dim stZelle             As String           'Aktuelle Zelle.
Dim oTabellenblatt      As Object           'Aktuelles Tabellenblatt.
```

Sollten mehrere Excel-Dateien gleichzeitig geöffnet sein, dann sicherstellen, dass der Programmcode in der aktuellen Excel-Datei, also in jener Datei wo auch der Programmcode hinterlegt ist, ausgeführt wird:

```
ThisWorkbook.Activate                           'Aktuelle Excel-Datei festlegen.
```

Um den Anwendern mitzuteilen, dass Excel beschäftigt ist, die Sanduhr einblenden. Zur Optimierung der Programmgeschwindigkeit die ständige Aktualisierung des Bildschirmes ausschalten und keine Berechnungen in den Tabellenblättern durchführen lassen:

```
Application.Cursor = xlWait                          'Sanduhr einblenden.
Application.ScreenUpdating = False                   'Bildschirmaktualisierung
                                                     'ausschalten.
Application.Calculation = xlManual                   'Automatische Berechnung
                                                     'ausschalten.
```

Tabellenblätter definieren und selektieren, Zellen auswählen:

```
Set oTabellenblatt = Sheets("Spielfeld")             'Akt. Tabelllenblatt festlegen.
oTabellenblatt.Select                                'Tabellenblatt auswaehlen.
Range("A1").Select                                   'Zelle A1 selektieren.
```

Beispiel für eine Schleife die ganze Tabellenblätter abarbeitet:

```
lZeile = 2: iSpalte = 1                              'Startwerte festlegen.
While "" & Trim(oTabellenblatt.Cells(lZeile, iSpalte)) <> "" 'Ein Tabellenblatt
                                                     'durcharbeiten:
  oTabellenblatt.Cells(lZeile, iSpalte + 1) = "xxx" 'Im Blatt etwas veraendern.
  lZeile = lZeile + 1                                'Zeilennummer erhoehen bis
Wend                                                 'alle abgearbeitet sind.
```

Einzelne Zellen und Bereiche festlegen und Werte eintragen:

```
stZelle = "A" & Trim(Str(iSpalte))                   'Eine Zelle festlegen oder einen
stBereich = "A" & Trim(Str(iSpalte)) & ":B" & Trim(Str(iSpalte)) 'Bereich festlegen.
oTabellenblatt.Range("A1") = ""                      'Definierte Zelle befuellen.
```

Zum Schluss die automatische Berechnung wieder einschalten und die gesamte Excel-Datei neu berechnen lassen sowie die Bildschirmaktualisierung wieder einschalten und Verabschiedung:

```
Application.Calculation = xlAutomatic                'Automatische Berechnung
                                                     'einschalten.
Application.Calculate                                'Excel-Datei neu berechnen.
Application.ScreenUpdating = True                    'Bildschirmaktualisierung
                                                     'ausschalten.

MsgBox "Habe Fertig!"                                'Abschlussmeldung ausgeben.
```

Abschließend nochmals die vollständige Musterfunktion:
Siehe auch Datei: Mustervorlage_02_Musterfunktion.xlsm

```
Public Function Musterfunktion() As Boolean
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*----------------------------------------------------------------------*/
'*                                                                      */
'*      FUNKTION:       Musterfunktion                                  */
```

```
'*                                                                            */
'*      BESCHREIBUNG:   Dient als Kopiervorlage fuer weitere Funktionen.      */
'*                                                                            */
'*      RETURN          Boolean ..... True = Funktion in Ordnung              */
'*                                    False = Fehler in Funktion              */
'*      PARAMETER:      keine                                                 */
'*                                                                            */
'*----------------------------------------------------------------------------*/
Dim i                   As Integer                  'Allgemeine Zaehlvariable.
Dim iSpalte             As Integer                  'Aktuelle Spaltennummer.
Dim lZeile              As Long                     'Aktuelle Zeilennummer.
Dim stBereich           As String                   'Aktueller Bereich.
Dim stZelle             As String                   'Aktuelle Zelle.
Dim oTabellenblatt      As Object                   'Aktuelles Tabellenblatt.

On Error GoTo Fehler                                'Bei Fehler zur Fehlermarke.
Musterfunktion = True                               'Returnwert = alles OK setzen.

ThisWorkbook.Activate                               'Aktuelle Excel-Datei festlegen.
Application.Cursor = xlWait                         'Sanduhr einblenden.
Application.ScreenUpdating = False                  'Bildschirmaktualisierung
                                                    'ausschalten.
Application.Calculation = xlManual                  'Automatische Berechnung
                                                    'ausschalten.

Set oTabellenblatt = Sheets("Spielfeld")            'Akt. Tabelllenblatt festlegen.
oTabellenblatt.Select                               'Tabellenblatt auswaehlen.
Range("A1").Select

lZeile = 2: iSpalte = 1                             'Startwerte festlegen.
While "" & Trim(oTabellenblatt.Cells(lZeile, iSpalte)) <> "" 'Ein Tabellenblatt
                                                    'durcharbeiten:
  oTabellenblatt.Cells(lZeile, iSpalte + 1) = "xxx" 'Im Blatt etwas veraendern.
  lZeile = lZeile + 1                               'Zeilennummer erhoehen bis
Wend                                                'alle abgearbeitet sind.

stZelle = "A" & Trim(Str(iSpalte))                  'Eine Zelle festlegen oder einen
stBereich = "A" & Trim(Str(iSpalte)) & ":B" & Trim(Str(iSpalte)) 'Bereich festlegen.
oTabellenblatt.Range("A1") = ""                     'Definierte Zelle befuellen.

Application.Calculation = xlAutomatic               'Automatische Berechnung
                                                    'einschalten.
Application.Calculate                               'Excel-Datei neu berechnen.
Application.ScreenUpdating = True                   'Bildschirmaktualisierung
                                                    'ausschalten.

MsgBox "Habe Fertig!"                               'Abschlussmeldung ausgeben.

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

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

4.) Makros für Funktionsaufrufe

Als nächstes erstellen wir uns ein Makro welches unsere Musterfunktion aufruft. Dieser Funktionsaufruf soll im Modul **Makros** hinterlegt werden und dient zukünftig als Kopiervorlage für weitere Makros.

Für Makros gibt es eine zusätzliche, spezielle Anweisung: **Option Private Module**. Verwendet man diese Option, dann sind alle Makros in diesem Modul für die späteren Anwender der Excel-Datei unsichtbar – können also nicht über Entwicklertools -> Makros -> Ausführen gestartet werden. Plaziert wird diese Anweisung direkt nach Option Explicit:

```
Option Explicit                                 'Alle Variablen muessen
                                                'explizit angegeben werden.
Option Private Module                           'Die einzelnen Subs nicht unter
                                                'dem Menuepunkt Makros anzeigen.
```

Nun zum Erstellen des Makros. Dazu selektieren wir zuerst das Modul **Makros** und klicken dann auf Einfügen > Prozedur.
Bei **Name** geben wir "Muster_Funktion", als **Typ**: **Sub** und als **Gültigkeitsbereich**: **Private** ein.

Im Gegensatz zu Funktionen werden Prozeduren (Subs) üblicherweise als **Private** definiert, da Funktionen aus anderen Modulen unsere Start-Makros niemals aufrufen sollten. Bei der Namensvergabe für Makros verwende ich üblicherweise denselben Namen wie die aufzurufende Funktion. Da der gleiche Name jedoch nicht erlaubt ist ergänze bzw. erweitere ich den Makronamen mit "_" (Unterstrich).

```
Public Sub Muster_Funktion()

End Sub
```

Bei der Dokumentation beschränke ich mich dabei auf das nötigste. Im Gegensatz zu Funktionen haben Prozeduren keinen Rückgabewert. Auch auf eine Fehlerbehandlung kann hier getrost verzichtet werden. Das fertige Makro sieht dann so aus:

```
Public Sub Muster_Funktion()
'*************************************************************************/
'*  Makro wird ausgefuehrt bei Klick auf Schaltflaeche: .......          */
'*************************************************************************/
  If Musterfunktion = False Then                'Wenn Fehler in Funktion aufge-
                                                'treten: Fehlermeldung ausgeben.
    MsgBox "Fehler beim Ausführen der Musterfunktion!", _
           vbOKOnly + vbExclamation + vbDefaultButton1, "Musterfunktion"
  Else                                          'Funktion korrekt ausgefuehrt:
                                                'Erfolgsmeldung ausgeben.
    MsgBox "Musterfunktion erfolgreich ausgeführt.", _
           vbOKOnly + vbInformation + vbDefaultButton1, "Musterfunktion"
  End If
End Sub                                         'Prozedurende.
```

Es wird der Rückgabewert der **Musterfunktion** abgefragt – dabei wird die aufgerufene Funktion vollständig abgearbeitet – und je nachdem, ob die Funktion fehlerfrei ausgeführt wurde oder nicht, wird eine entsprechende Hinweismeldung ausgegeben.

Und das wäre auch schon alles zum Thema Makros programmieren.
Siehe auch Datei: Mustervorlage_03_Makro.xlsm

5.) Excel Oberfläche während des Spieles gestalten

Nun überlegen wir uns wie die Oberfläche (Ansicht) von Excel aussehen soll während das Spiel gespielt wird. Ich nenne das kurz Spielansicht in Gegensatz zur Entwickleransicht in welcher man das Spiel programmiert. Weiters müssen wir uns darüber Gedanken machen was passieren soll, wenn man mit mehreren Excel-Dateien gleichzeitig arbeitet. Beim Wechsel zwischen einzelnen Excel-Dateien soll die Spielansicht natürlich nur für das Excel-Spiel gelten – nicht aber für andere Dateien, deren Ansicht soll unverändert bleiben.

Um mehrere Dateien gleichzeitig geöffnet zu haben gibt es 2 Möglichkeiten: Entweder startet man Excel nur einmal und öffnet daraus mehrere Dateien, oder man startet Excel mehrere Male und öffnet mit jedem gestarteten Excel nur eine einzige Datei (oder auch mehrere). Wir werden versuchen einen Code zu entwickeln der beide Möglichkeiten berücksichtigt.

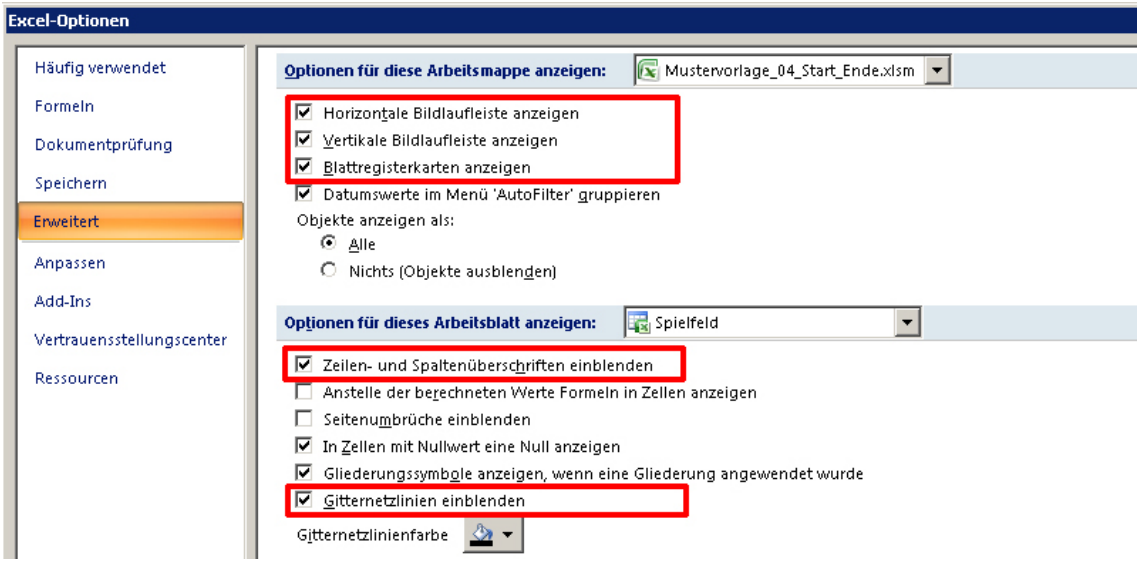
Beabsichtigte Vorgehensweise:
Beim Öffnen der Spieldatei merken wir uns die aktuellen Ansichtseinstellungen. Danach legen wir jene Einstellungen fest wie wir sie im Spiel haben möchten. Beim Beenden der Spieldatei bzw. beim Wechsel zu einer anderen Datei werden die gemerkten Einstellungen wiederhergestellt. Beim neuerlichen Aktivieren der Spieldatei wird auf die Spielansicht zurück geschaltet. Das Setzen der Einstellungen für die Spielansicht und das Wiederherstellen der Entwurfsansicht soll in einer Funktion umgesetzt werden. Damit man zu jedem Zeitpunkt auf die gemerkten Einstellungen zurückgreifen kann verwenden wir dazu global definierte Variablen. Gleichzeitig legen wir dabei auch globale Konstanten an, welche unsere Standardwerte für die Spielansicht und das Spiel selbst definieren. Zum Schluß behandeln wir noch die Problematik beim Wechsel zwischen geöffneten Excel-Dateien.

Zum Üben erstellen wir dann eine zusätzliche Funktion, welche unsere Funktion zum Umschalten der Ansichten testen soll.

Soweit so gut, aber jetzt geht´s ans Eingemachte:

5.1.) Mögliche Einstellungen für Excel-Ansichten

Vorab sei erwähnt, daß sämtliche Einstellungen manuell in den Excel Optionen geändert werden können. Sollten Sie während der Programmierung ungewollte Ansichten erhalten, dann klicken Sie auf die Office-Schaltfläche > Excel Optionen > Erweitert. Die rot umrandeten Häkchen wollen wir verändern:



Dazu verwenden wir folgende Befehle:

```
ActiveWindow.DisplayHorizontalScrollBar = False 'Schiebeleiste unten ausblenden.
ActiveWindow.DisplayVerticalScrollBar = False   'Schiebeleiste rechts ausblenden.
ActiveWindow.DisplayWorkbookTabs = False        'Tabellenblaetter unten ausblenden.
ActiveWindow.DisplayHeadings = False            'Keine Zeilen- & Spaltenbeschriftung
ActiveWindow.DisplayGridlines = False           'Gitternetzlinien ausblenden.
```

Zusätzlich gibt es noch weitere Befehle (die Ansicht betreffend) die wir verwenden können:

```
Application.DisplayFullScreen = True            'Ganzer Bildschirm einschalten.
Application.WindowState = xlMaximized           'Excel Fenster maximieren.
ActiveWindow.WindowState = xlMaximized          'Tabellenblatt Fenster maximieren
ActiveWindow.Zoom = 100                         'Zoom 100% fuer Ansicht festlegen.
ActiveWindow.View = xlNormalView                'Ansicht ist nicht Seitenlayout und
                                                'nicht Umbruchvorschau, nur normal.
```

Außerdem sollten alle Tabellenblätter mit Ausnahme des **Spielfeldes** unsichtbar gemacht werden:

```
Sheets("Spielfeld").Visible = xlSheetVisible    'Tabellenblatt einblenden.
For i = 1 To Sheets.Count                       'Alle Tabellenblaetter bis auf
  If Sheets(i).Name <> "Spielfeld" Then         'Spielfeld unsichtbar machen.
    Sheets(i).Visible = xlSheetVeryHidden       '(kann nur ueber VBA wieder
                                                'sichtbar gemacht werden, nicht
                                                'jedoch ueber Excel-Befehle.)
  End If                                        'Weiter zum naechsten Tabellenblatt
Next i                                          'bis alle abgearbeitet sind.
```

Um der Spielansicht eine persönliche Note zu geben kann man auch dieses verwenden:

```
ActiveWindow.Caption = "Mein Spielprogramm"     'Ersetzt Dateinamen im Fenstertitel.
Application.Caption = "Ing. Mitsch Harald"      'Ersetzt Microsoft-Excel im Fenster-
                                                'titel mit eigenem Namen.
```

Sinnvoll ist auch noch das Deaktivieren der automatischen Speicherung der Wiederherstellungs-Informationen:

```
ActiveWorkbook.EnableAutoRecover = False        'Auto-Wiederherstellung fuer
                                                'diese Excel-Datei ausschalten.
```

Zum Schluß fällt mir noch ein, auch wenn es für Spielprogramme nicht unbedingt benötigt wird, die Formatierung von Zahlen nicht Windows zu überlassen sondern selbst zu verwalten.

```
With Application                                'Ab Excel 2007 sollte zusaetzlich
  .DecimalSeparator = ","                       'dieser Codeteil aktiviert werden,
  .ThousandsSeparator = "."                     'damit die Dezimal- und Tausender-
  .UseSystemSeparators = False                  'Trennzeichen nicht aus dem
End With                                        'Betriebssystem uebernommen werden.
```

Jetzt haben wir die einzelnen Befehle, welche für Änderungen in den Excel-Ansichten verwendet werden können, kennengelernt. Bevor wir jedoch diese Befehle anwenden definieren wir zuerst die globalen Variablen in denen die aktuellen Einstellungen gespeichert werden. Mit diesen Variablen werden später die Befehle zur Aktualisierung der Ansichten aufgerufen:

5.2.) Globale Variablen und Konstanten

Diese werden, wie Sie bereits vermutet haben, im Modul **Globales** abgelegt. Dazu gibt es nicht wirklich viel zu sagen, daher hier gleich mein Vorschlag für den Programmcode (Dokumentation beachten):

```
Option Explicit                                     'Alle Variablen muessen
                                                    'explizit angegeben werden.

Global Const PROGRAMMNAME = "Mustervorlage"         'Name des Programmes.
                                                    'Name des Programmierers:
Global Const PROGRAMMAUTOR = "© 2019 by TBM - Technisches Büro Mitsch" '(c) = Asc(184)
Global Const TABELLENBLATT = "Spielfeld"            'Tabellenblatt fuer Spielfeld.
Global oTabellenblatt                  As Object    'Variable fuer Tabellenblatt.

Global Const JA = True                              'Konstanten zur besseren Lesbarkeit
Global Const NEIN = False                           'des Programmcodes.

                                                    'Zum Merken der Anzeigeeinstellungen:
Global lTabellenblattFensterGroesse    As Long      'Application.WindowState
                                                    'Tabellenblatt Fenstergroesse.

Global bAnzeigenGanzerBildschirm       As Boolean   'Application.DisplayFullScreen
                                                    'Ganzer Bildschirm anzeigen.

Global bAnzeigenTabellenblaetter       As Boolean   'ActiveWindow.DisplayWorkbookTabs
                                                    'Tabellenblatt-Register anzeigen.

Global bAnzeigenVertikaleSchiebeleiste As Boolean   'ActiveWindow.DisplayVerticalScrollBar
                                                    'Schiebeleiste rechts anzeigen.

Global bAnzeigenHorizontaleSchiebeleiste As Boolean 'ActiveWindow.DisplayHorizontalScrollBar
                                                    'Schiebeleiste unten anzeigen.

Global bAnzeigenZeilenSpaltenBezeichnung As Boolean 'ActiveWindow.DisplayHeadings
                                                    'Zeilen und Spaltenbeschriftung.

Global bAnzeigenGitternetzlinien       As Boolean   'ActiveWindow.DisplayGridlines
                                                    'Gitternetzlinien anzeigen.

Global lExcelFensterGroesse            As Long      'ActiveWindow.WindowState
                                                    'Excel (Applikation) Fenstergroesse.

Global iTabellenblattZoom              As Integer   'ActiveWindow.Zoom
                                                    'Aktueller Zoom-Faktor (75 = 75%)


                                                    'Verhalten beim STARTEN:
Global Const SPIELANSICHT = NEIN                    'Waehrend der Spielentwicklung.
'Global Const SPIELANSICHT = JA                     'Nach Fertigstellung des Spiels.
Global Const ZOOMFAKTOR = 100                       'Gewuenschter Zoom-Faktor (100 %)

                                                    'Verhalten beim BENDEN:
Global Const AUTOSPEICHERN = JA                     'Beim Beenden autom. alles speichern.
'Global Const AUTOSPEICHERN = NEIN                  'Aenderungen automatisch Verwerfen
'Global Const FRAGESPEICHERN = NEIN                 'wenn auch FRAGESPEICHERN = NEIN oder
Global Const FRAGESPEICHERN = JA                    'zusaetzlich Nachfrage ob Speichern.
```

Die Konstanten TABELLENBLATT, SPIELANSICHT, ZOOMFAKTOR, AUTOSPEICHERN und FRAGESPEICHERN wurden zuvor noch nicht angesprochen. Sie werden jedoch im nächsten Abschnitt benötigt und dort näher erklärt.

Die Konstanten JA und NEIN sollten sich von selbst erklären und wurden lediglich zur besseren Lesbarkeit des Programmcodes hinterlegt.

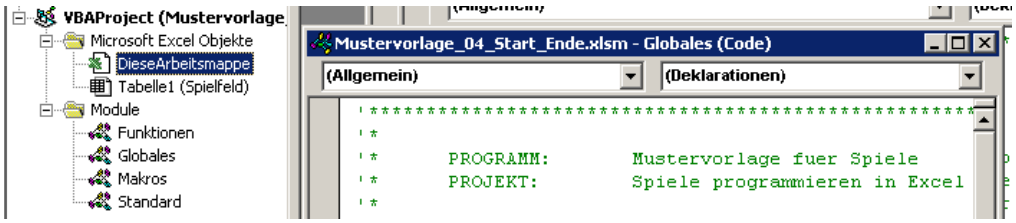
5.3.) Die ersten Funktionen beim Öffnen der Spieldatei

Erstellen Sie zunächst im Modul **Funktionen** folgende 3 Funktionen, speichern und schließen Sie die Excel-Datei und öffnen Sie sie dann erneut:

```
Public Function SpielAnsichtAktualisieren()
  MsgBox "SpielAnsichtAktualisieren"
End Function

Public Function SpielStarten()
  MsgBox "SpielStarten"
End Function

Public Function Auto_Open()
  MsgBox "Auto_Open"
End Function
```

Nachdem Sie die Datei neu geöffnet haben wird die Meldung: "Auto_Open" ausgegeben. Eine Funktion mit dem Namen Auto_Open wird beim Öffnen einer Excel-Datei automatisch ausgeführt. Das Ausführen der Funktion kann nur verhindert werden, wenn die Excel-Datei bei gedrückter Großschreibtaste (SHIFT-Taste) geöffnet wird - dasselbe gilt auch für die nächste Methode.

Für eine andere Methode doppelklicken Sie in **Visual Basic** im **Projekt-Explorer** das Microsoft Excel Objekt: **DieseArbeitsmappe**:



Wählen Sie dann rechts im Code-Fenster der Combobox (Allgemein) "Workbook" aus und daneben in der Combobox (Deklarationen) "Open" aus. Folgender Code wird automatisch generiert:

```
Private Sub Workbook_Open()

End Sub
```

Erweitern Sie die Funktion zu:

```
Private Sub Workbook_Open()
  MsgBox "Workbook_Open"
End Sub
```

Speichern und schließen Sie die Datei. Bein erneuten Öffnen erhalten Sie zuerst die Meldung "Workbook_Open" gefolgt von "Auto_Open". Daher bevorzuge ich die Methode mit Workbook_Open um automatisch Programmcode ausführen zu lassen, da sie noch vor Auto_Open ausgeführt wird. Ein wesentlicher Unterschied besteht darin, daß Auto_Open "nur" eine allgemeine Funktion ist während Workbook_Open eine Prozedur aus dem allgemein gültigen Klassenmodul: **DieseArbeitsmappe** ist.

Löschen Sie daher die Funktion Auto_Open wieder oder deaktivieren Sie sie indem Sie sie unter Kommentar setzen. Die beiden anderen Funktionen werden wir etwas später weiterentwickeln.

5.4.) Die Prozedur: Workbook_Open

Erstellen wir nun gemeinsam den Programmcode, der sofort nach dem Öffnen der Spieldatei ausgeführt wird (im Klassenmodul: DieseArbeitsmappe, Prozedur: Workbook_Open).

Eine persönliche Vorliebe von mir ist es alle Fenster in der maximalen Ansichtsgröße darzustellen. Daher auch hier als erster Befehl das Excel-Fenster maximiert darstellen und erst danach speichern:

```
Application.WindowState = xlMaximized                'Excel Fenster immer maximieren.
```

Als nächstes speichern wir alle aktuellen Anzeigeeigenschaften in den entsprechenden Variablen die wir zuvor im Modul: Globales definiert haben:

```
lExcelFensterGroesse = Application.WindowState       'Excel Fenstergroesse auslesen.
bAnzeigenGanzerBildschirm = Application.DisplayFullScreen    'Merken der aktuellen
bAnzeigenTabellenblaetter = ActiveWindow.DisplayWorkbookTabs 'Anzeigeeinstellungen
bAnzeigenVertikaleSchiebeleiste = ActiveWindow.DisplayVerticalScrollBar  'von Excel
bAnzeigenHorizontaleSchiebeleiste = ActiveWindow.DisplayHorizontalScrollBar 'um sie
bAnzeigenZeilenSpaltenBezeichnung = ActiveWindow.DisplayHeadings 'spaeter wieder-
bAnzeigenGitternetzlinien = ActiveWindow.DisplayGridlines    'herstellen zu koennen.
lTabellenblattFensterGroesse = ActiveWindow.WindowState
iTabellenblattZoom = ActiveWindow.Zoom
```

Diese Einstellungen verwalte ich gerne persönlich, vor allem wenn man häufig zwischen deutscher und englischer Excel-Version wechseln muß.

```
With Application                                     'Ab Excel 2007 sollte zusaetzlich
  .DecimalSeparator = ","                            'dieser Codeteil aktiviert
  .ThousandsSeparator = "."                          'werden damit Dezimal- & 1000er
  .UseSystemSeparators = False                       'Trennzeichen nicht aus dem
End With                                             'Betriebssystem uebernommen
                                                     'werden.
```

Während das Spiel läuft ist es sinnlos alle paar Minuten die Wiederherstellungsinformationen zu speichern – daher direkt deaktivieren. Nicht vergessen: Sollte bei Dateiwechsel oder Umschalten in die Entwickleransicht wieder aktiviert werden:

```
ActiveWorkbook.EnableAutoRecover = False             'Auto-Wiederherst. deaktivieren.
```

Uns jetzt aktualisieren wir die Ansichtseinstellungen tatsächlich. Je nachdem ob die globale Konstante SPIELANSICHT auf JA oder auf NEIN gesetzt ist, erzeugt die Funktion entweder die Spielansicht oder die Entwickleransicht.

```
Call SpielAnsichtAktualisieren(SPIELANSICHT)         'Umschalten auf Entwickleransicht
                                                     'oder auf Spielansicht.
```

Natürlich braucht jedes Spiel auch einen Zufallsgenerator – der wird gleich hier initialisiert (mit Beispiel):

```
                                                     'Indealer Zeitpunkt um auch den
Randomize (Date + Time)                              'Zufallsgenerator initialisieren.
'iZufall = Int((6 * Rnd) + 1)                        'Beispiel fuer einen Wuerfel.
```

Zum Schluß starten wir die Funktion für das Spielprogramm (muß im Modul Funktionen vorhanden sein).

```
Call SpielStarten                                    'Spiel-Startprogramm aufrufen.
```

Die Prozedur **Workbook_Open** wird normalerweise nur ein einziges Mal beim Öffnen der Datei ausgeführt, danach nie mehr. Möchten Sie die Prozedur auch aus anderen Funktionen aufrufen, dann ist die Prozedur **Workbook_Open** nicht als **Private** sondern als **Public** zu definieren.

Zum Abschluß nochmals der vollständige Programmcode:

```
Private Sub Workbook_Open()
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION - PROZEDUR              */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       Workbook_Open                                    */
'*                                                                       */
'*      BESCHREIBUNG:   Diese Funktion wird sofort aufgerufen nachdem die   */
'*                      Excel-Datei geoeffnet wurde. Hier werden all jene   */
'*                      Einstellungen festgelegt, welche im gesamten Excel  */
'*                      Gueltigkeit haben.                               */
'*                      Zum Schluss erfolgt der eigentliche Programmstart:  */
'*                                                                       */
'*                      Soll diese Prozedur nicht automatisch gestartet     */
'*                      werden, so ist beim Oeffnen der Excel-Datei die     */
'*                      SHIFT-Taste (Grossschreibtaste) gedrueckt zu        */
'*                      halten.                                          */
'*                                                                       */
'*      RETURN          nichts                                           */
'*                                                                       */
'*      PARAMETER:      keine                                            */
'*                                                                       */
'*-----------------------------------------------------------------------*/
On Error GoTo Fehler                            'Bei Fehler zur Fehlerbehandlung.

                                                'Wenn gewuenscht, dann
Application.WindowState = xlMaximized           'Excel Fenster immer maximieren.

lExcelFensterGroesse = Application.WindowState      'Excel Fenstergroesse auslesen.
bAnzeigenGanzerBildschirm = Application.DisplayFullScreen    'Merken der aktuellen
bAnzeigenTabellenblaetter = ActiveWindow.DisplayWorkbookTabs 'Anzeigeeinstellungen
bAnzeigenVertikaleSchiebeleiste = ActiveWindow.DisplayVerticalScrollBar  'von Excel
bAnzeigenHorizontaleSchiebeleiste = ActiveWindow.DisplayHorizontalScrollBar 'um sie
bAnzeigenZeilenSpaltenBezeichnung = ActiveWindow.DisplayHeadings 'spaeter wieder-
bAnzeigenGitternetzlinien = ActiveWindow.DisplayGridlines    'herstellen zu koennen.
lTabellenblattFensterGroesse = ActiveWindow.WindowState
iTabellenblattZoom = ActiveWindow.Zoom

ActiveWorkbook.EnableAutoRecover = False        'Auto-Wiederherstellung fuer
                                                'diese Excel-Datei ausschalten.

With Application                                'Ab Excel 2007 sollte zusaetzlich
  .DecimalSeparator = ","                       'dieser Codeteil aktiviert werden,
  .ThousandsSeparator = "."                     'damit die Dezimal- und Tausender-
  .UseSystemSeparators = False                  'Trennzeichen nicht aus dem
End With                                        'Betriebssystem uebernommen werden.

                                                'Idealer Zeitpunkt um auch den
Randomize (Date + Time)                         'Zufallsgenerator zu initialisieren.
'iZufall = Int((6 * Rnd) + 1)                   'Beispiel fuer einen Wuerfel.

Call SpielAnsichtAktualisieren(SPIELANSICHT)    'Umschalten auf Entwickleransicht
                                                'oder auf Spielansicht, je nach
                                                'Definition im Modul Globales.

'*-----------------------------------------------------------------------*/
Call SpielStarten                               'Startprogramm aufrufen.
'*-----------------------------------------------------------------------*/

                                                'Fehlerbehandlungsroutine:
Beenden:                                        '========================
  Application.Cursor = xlDefault                'Sanduhr ausblenden.
  Exit Sub                                      'Prozedur abbrechen.

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

5.5.) Die Funktion: SpielAnsichtAktualisieren

Nun erstellen wir den Programmcode der zwischen der Spielansicht und der Entwickleransicht hin und her schaltet. Dazu verwenden wir die zuvor angelegte Funktion **SpielAnsichtAktualisieren** im Modul **Funktionen**.

```
Public Function SpielAnsichtAktualisieren()
  MsgBox "SpielAnsichtAktualisieren"
End Function
```

Da wir die Funktion zum Umschalten der Ansichten verwenden wollen, müssen wir der Funktion noch einen Parameter übergeben, der festlegt ob die Spielansicht oder die Entwickleransicht anzuzeigen ist. Wir erweitern den Programmcode folgend:

```
Public Function SpielAnsichtAktualisieren(bSpielansicht As Boolean) As Boolean
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*----------------------------------------------------------------------*/
'*                                                                      */
'*      FUNKTION:       SpielAnsichtAktualisieren                       */
'*                                                                      */
'*      BESCHREIBUNG:   Aktiviert entweder die Spielansicht oder die    */
'*                      Entwickleransicht.                              */
'*                                                                      */
'*      RETURN          Boolean ..... True = Funktion in Ordnung        */
'*                                    False = Fehler in Funktion        */
'*                                                                      */
'*      PARAMETER:      Boolean ..... True = JA = Spielansicht          */
'*                                    False = Entwickleransicht         */
'*                                                                      */
'*----------------------------------------------------------------------*/
Dim i                   As Integer                  'Allgemeine Zaehlvariable.

On Error GoTo Fehler                                'Bei Fehler zur Fehlerbehandlung.
SpielAnsichtAktualisieren = True                    'Returnwert = alles OK setzen.
Application.Cursor = xlWait                         'Sanduhr einblenden.

If bSpielansicht = JA Then                          'Wenn Spielansicht anzeigen:

Else                                                'Spielansicht zuruecksetzen:

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

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

Die Zählvariable **i** benötigen wir erst später. Ansonst sollte sich der Funktionsablauf von selbst erklären.

Als nächstes müssen wir sicherstellen, daß die Änderungen nur in der aktuellen Spieldatei und nur im Tabellenblatt Spielfeld durchgeführt werden. Dazu fügen wir vor der If-Abfrage folgende Befehle ein:

```
ThisWorkbook.Activate                          'Akt. Arbeitsmappe aktivieren.
Sheets(TABELLENBLATT).Visible = xlSheetVisible 'Tabellenblatt einblenden.
Sheets(TABELLENBLATT).Select                   'Tabellenblatt selektieren.
Range("A1").Select                             'Zelle A1 selektieren.
```

Die globale Konstante **TABELLENBLATT** beinhaltet den Wert "Spielfeld" (wie im Modul **Globales** zuvor definiert). Unser Spielfeld muß natürlich sichtbar sein und um die nachfolgenden Befehle nur für dieses Blatt anzuwenden selektieren wir es. Den Cursor setzen wir in die linke obere Ecke.

Nun sorgen wir dafür, daß in der Spielansicht nur das Tabellenblatt **Spielfeld** sichtbar ist. Übrigens, in Excel muß zumindest immer ein Tabellenblatt sichtbar sein. Will man alle Tabellenblätter ausblenden erhält man einen Laufzeitfehler. Bei der Entwickleransicht hingegen sollen alle Tabellenblätter wieder angezeigt werden.

```
If bSpielansicht = JA Then                     'Wenn Spielansicht anzeigen:
  For i = 1 To Sheets.Count                    'Alle Tabellenblaetter bis auf
    If Sheets(i).Name <> TABELLENBLATT Then     'Spielfeld unsichtbar machen.
      Sheets(i).Visible = xlSheetVeryHidden     '(kann nur ueber VBA wieder
                                                'sichtbar gemacht werden, nicht
                                                'jedoch ueber Excel-Befehle.)
    End If                                      'Weiter zum naechsten Blatt
  Next i                                        'bis alle abgearbeitet sind.

Else                                            'Spielansicht zuruecksetzen:
  For i = 1 To Sheets.Count                     'Fuer alle Tabellenblaetter:
    Sheets(i).Visible = xlSheetVisible          'Blatt sichtbar machen, True, -1
  Next i                                        'weiter zum naechsten Blatt.
End If                                          'Excel-Datei wieder einschalten.
```

Die Excel-Konstante **xlSheetVeryHidden** verhindert, daß der Anwender selbst das Tabellenblatt wieder sichtbar machen kann, da dieses in der Excel-Auflistung der Tabellenblätter nicht mehr angezeigt wird. Bei der alternativen Excel-Konstante **xlSheetHidden** ist das nicht der Fall.

Jetzt werden die Ansichtseinstellungen für die Spielansicht festgelegt:

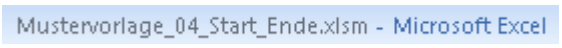
```
Application.WindowState = xlMaximized              'Excel Fenster maximieren.
Application.DisplayFullScreen = True               'Vollbildmodus einschalten.
ActiveWindow.DisplayWorkbookTabs = False           'Tabellenblattregister ausblenden.
ActiveWindow.DisplayVerticalScrollBar = False      'Schiebeleiste rechts ausblenden.
ActiveWindow.DisplayHorizontalScrollBar = False    'Schiebeleiste unten ausblenden.
ActiveWindow.DisplayHeadings = False               'Zeilen und Spaltenbeschriftung.
ActiveWindow.DisplayGridlines = False              'Gitternetzlinien ausblenden.
ActiveWindow.WindowState = xlMaximized             'Tabellenblatt Fenster maximieren.
ActiveWindow.Zoom = ZOOMFAKTOR                     'Zoom fuer Ansicht festlegen.

ActiveWorkbook.EnableAutoRecover = False           'Auto-Wiederherstellung fuer diese
                                                   'Excel-Datei ausschalten.

ActiveWindow.View = xlNormalView                   'Nicht Seitenlayout und nicht
                                                   'Umbruchvorschau verwenden.
ActiveWindow.Caption = PROGRAMMNAME                'Ersetzt Dateinamen im Excel
                                                   'Fenstertitel.
Application.Caption = PROGRAMMAUTOR                'Ersetzt Microsoft-Excel im
                                                   'Fenstertitel.
```

Die beiden letzten Befehle erzeugen zum Beispiel folgenden Fenstertitel:

Und nun wollen wir die Ansichtseinstellungen für die Entwickleransicht wiederherstellen:

```
Application.WindowState = lExcelFensterGroesse    'Excel Fenster wiederherstellen.
Application.DisplayFullScreen = bAnzeigenGanzerBildschirm    'Wiederherstellen der
ActiveWindow.DisplayWorkbookTabs = bAnzeigenTabellenblaetter 'urspruenglichen
ActiveWindow.DisplayVerticalScrollBar = bAnzeigenVertikaleSchiebeleiste
ActiveWindow.DisplayHorizontalScrollBar = bAnzeigenHorizontaleSchiebeleiste
ActiveWindow.DisplayHeadings = bAnzeigenZeilenSpaltenBezeichnung
ActiveWindow.DisplayGridlines = bAnzeigenGitternetzlinien    'Anzeigeeinstellungen.
ActiveWindow.WindowState = lTabellenblattFensterGroesse 'Tabellenblatt Fenster und
ActiveWindow.Zoom = iTabellenblattZoom            'Zoom der Ansicht wiederherstellen.

ActiveWorkbook.EnableAutoRecover = True           'Auto-Wiederherstellung fuer diese
                                                  'Excel-Datei wieder einschalten.

                                                  'Je nach Belieben kann folgendes
                                                  'unter Kommentar gestellt werden:
ActiveWindow.Caption = ThisWorkbook.Name          'Fenstertitel = Dateiname.
Application.Caption = ""                          'Schreibt wieder: - Microsoft Excel
```

Die beiden letzten Befehle erzeugen wieder den originalen Fenstertitel von Excel, zum Beispiel:



Damit wäre die Funktion **SpielAnsichtAnzeigen** fertig programmiert. Es fehlt nur mehr eine Kleinigkeit: Da diese Funktion bei allen Spielen dieselbe bleiben wird sollte sie aus dem Modul **Funktionen** in das Modul **Standard** verschoben werden!

Abschließend noch einmal der gesamte Programmcode für diese Funktion:

```
Public Function SpielAnsichtAktualisieren(bSpielansicht As Boolean) As Boolean
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       SpielAnsichtAktualisieren                        */
'*                                                                       */
'*      BESCHREIBUNG:   Aktiviert entweder die Spielansicht oder die     */
'*                      Entwickleransicht.                               */
'*                                                                       */
'*      RETURN          Boolean ..... True = Funktion in Ordnung         */
'*                                    False = Fehler in Funktion         */
'*                                                                       */
'*      PARAMETER:      Boolean ..... True = JA = Spielansicht           */
'*                                    False = Entwickleransicht          */
'*                                                                       */
'*-----------------------------------------------------------------------*/
Dim i                   As Integer                  'Allgemeine Zaehlvariable.

On Error GoTo Fehler                                'Bei Fehler zur Fehlerbehandlung.
SpielAnsichtAktualisieren = True                    'Returnwert = alles OK setzen.
Application.Cursor = xlWait                         'Sanduhr einblenden.

ThisWorkbook.Activate                               'Aktuelle Arbeitsmappe auswaehlen.
```

```
Sheets(TABELLENBLATT).Visible = xlSheetVisible      'Tabellenblatt einblenden.
Sheets(TABELLENBLATT).Select                        'Tabellenblatt selektieren.
Range("A1").Select                                  'Zelle A1 selektieren.

If bSpielansicht = JA Then                          'Wenn Spielansicht anzeigen:
  For i = 1 To Sheets.Count                         'Alle Tabellenblaetter bis auf
    If Sheets(i).Name <> TABELLENBLATT Then         'Spielfeld unsichtbar machen.
      Sheets(i).Visible = xlSheetVeryHidden         '(kann nur ueber VBA wieder
                                                    'sichtbar gemacht werden, nicht
                                                    'jedoch ueber Excel-Befehle.)
    End If                                          'Weiter zum naechsten Tabellenblatt
  Next i                                            'bis alle abgearbeitet sind.

  Application.WindowState = xlMaximized             'Excel Fenster maximieren.
  Application.DisplayFullScreen = True              'Vollbildmodus einschalten.
  ActiveWindow.DisplayWorkbookTabs = False          'Register mit Tabellen unten ausblenden.
  ActiveWindow.DisplayVerticalScrollBar = False     'Schiebeleiste rechts ausblenden.
  ActiveWindow.DisplayHorizontalScrollBar = False   'Schiebeleiste unten ausblenden.
  ActiveWindow.DisplayHeadings = False              'Zeilen und Spaltenbeschriftung.
  ActiveWindow.DisplayGridlines = False             'Gitternetzlinien ausblenden.
  ActiveWindow.WindowState = xlMaximized            'Tabellenblatt Fenster maximieren.
  ActiveWindow.Zoom = ZOOMFAKTOR                    'Zoomfaktor fuer Ansicht festlegen.

  ActiveWorkbook.EnableAutoRecover = False          'Auto-Wiederherstellung fuer diese
                                                    'Excel-Datei ausschalten.

  ActiveWindow.View = xlNormalView                  'Nicht Seitenlayout und Umbruchvorschau.
  ActiveWindow.Caption = PROGRAMMNAME               'Ersetzt Dateinamen im Fenstertitel.
  Application.Caption = PROGRAMMAUTOR               'Ersetzt Microsoft-Excel im Fenstertitel.


Else                                                'Spielansicht wieder zuruecksetzen:
  For i = 1 To Sheets.Count                         'Fuer alle Tabellenblaetter:
    Sheets(i).Visible = xlSheetVisible              'Blatt sichtbar machen, True, -1
  Next i                                            'weiter zum naechsten Tabellenblatt.

  Application.WindowState = lExcelFensterGroesse    'Excel Fenster wiederherstellen.
  Application.DisplayFullScreen = bAnzeigenGanzerBildschirm    'Wiederherstellen der
  ActiveWindow.DisplayWorkbookTabs = bAnzeigenTabellenblaetter 'urspruenglichen Anzeige-
  ActiveWindow.DisplayVerticalScrollBar = bAnzeigenVertikaleSchiebeleiste 'einstellungen.
  ActiveWindow.DisplayHorizontalScrollBar = bAnzeigenHorizontaleSchiebeleiste
  ActiveWindow.DisplayHeadings = bAnzeigenZeilenSpaltenBezeichnung
  ActiveWindow.DisplayGridlines = bAnzeigenGitternetzlinien
  ActiveWindow.WindowState = lTabellenblattFensterGroesse 'Tabellenblatt Fenster und
  ActiveWindow.Zoom = iTabellenblattZoom            'Zoom fuer Ansicht wiederherstellen.

  ActiveWorkbook.EnableAutoRecover = True           'Auto-Wiederherstellung fuer diese
                                                    'Excel-Datei wieder einschalten.

                                                    'Je nach Belieben kann folgendes
                                                    'auch unter Kommentar gestellt werden:
  ActiveWindow.Caption = ThisWorkbook.Name          'Fenstertitel = Dateiname.
  Application.Caption = ""                          'Schreibt wieder: - Microsoft Excel.
End If


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

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

5.6.) Testen der Umschaltung zwischen den Excel-Ansichten:

Getestet werden soll die Prozedur **Workbook_Open** aus dem Klassenmodul **DieseArbeitsmappe** und die Funktion **SpielAnsichtAktualisieren** im Modul **Standard**, welche zwischen den beiden Ansichten hin und her schaltet.

Zunächst müssen wir sicherstellen, daß in den globalen Variablen die aktuellen Einstellungen hinterlegt werden. Das geschieht in der Prozedur Workbook_Open. Zum Testen müssen wir daher die Datei speichern und neu öffnen. Wie wir bereits wissen wird nach dem Öffnen eine Funktion **Auto_Open** automatisch ausgeführt wenn sie vorhanden ist. Das machen wir uns zu nutze und schreiben den Code zum Testen in die **Auto_Open** Funktion.

```
Public Function Auto_Open()                              'Zum Testen...
Wiederholen:                                             'Sprungmarke zum Wiederholen

Call SpielAnsichtAktualisieren(JA)                       'Umschalten auf: Spielansicht.
If MsgBox("Auf Spielansicht umgeschaltet. Test Beenden?", vbYesNo) = vbYes Then GoTo Beenden

Call SpielAnsichtAktualisieren(NEIN)                     'Umschalten auf: Entwickleransicht.
If MsgBox("Entwickleransicht eingeschaltet. Test Beenden?", vbYesNo) = vbYes Then GoTo Beenden

GoTo Wiederholen                                         'Funktion von vorne nochmals
                                                         'durchlaufen lassen.
Beenden:                                                 'Sprungmarke zum Beenden
                                                         'der Funktion.
End Function                                             'Funktionsende.
```

Speichern sie nun die Datei, schließen Sie sie und öffnen Sie sie dann erneut.

Was sagen Sie dazu! Ist das nicht spitze!

Zuerst erhalten Sie die Meldung: **SpielStarten**. Hier würde das eigentliche Spiel gestartet werden. In die Spielansicht wurde jedoch noch nicht gewechselt, da wir im Modul **Globales** den Wert für die Variable **SPIELANSICHT** noch auf **NEIN** gesetzt haben. Dann wird die Funktion **Auto_Open** ausgeführt, welche dann zwischen den einzelnen Ansichten hin und her schaltet.

Wenn der Test erfolgreich war können Sie die **Auto_Open** Funktion wieder entfernen oder unter Kommentar stellen.

Wer sich diese Funktion für später, also bei der Programmierung der tatsächlichen Spiele, aufheben möchte, sollte die **Auto_Open** Funktion in das Modul **Standard** verschieben und umbenennen. Die vollständige Funktion könnte dann folgendermaßen aussehen:

```
Public Function Test_SpielAnsichtAktualisieren()
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       Test_SpielAnsichtAktualisieren                   */
'*                                                                       */
'*      BESCHREIBUNG:   Zum Testen der Funktion: SpielAnsichtAktualisieren */
'*                                                                       */
'*      RETURN          nichts                                           */
'*                                                                       */
'*      PARAMETER:      keine                                            */
'*                                                                       */
'*-----------------------------------------------------------------------*/
```

```
On Error GoTo Fehler                                    'Bei Fehler zur Fehlermarke.
Application.Cursor = xlWait                             'Sanduhr einblenden.

Wiederholen:                                            'Sprungmarke zum Wiederholen
                                                        'der Funktion:
                                                        'Umschalten auf: Spielansicht.
If SpielAnsichtAktualisieren(JA) = False Then           'Wenn Fehler in Funktion aufge-
                                                        'treten: Fehlermeldung ausgeben.
  MsgBox "Fehler beim Umschalten auf Spielansicht!", _
         vbOKOnly + vbExclamation + vbDefaultButton1, PROGRAMMNAME
  GoTo Beenden                                          'Funktion korrekt ausgefuehrt:
Else                                                    'Frage: Funktion beenden?
  If MsgBox("Auf Spielansicht umgeschaltet." & vbCrLf & vbCrLf & _
            "Testfunktion beenden? (nein = nochmals umschalten)", _
            vbYesNo + vbQuestion + vbDefaultButton2, PROGRAMMNAME) = vbYes Then
    GoTo Beenden                                        'Wenn ja, dann Funktion
  End If                                                'beenden lassen. Sonst:
End If
                                                        'Umschalten auf: Entwickleransicht.
If SpielAnsichtAktualisieren(NEIN) = False Then         'Wenn Fehler in Funktion aufge-
                                                        'treten: Fehlermeldung ausgeben.
  MsgBox "Fehler beim Umschalten auf Entwickleransicht!", _
         vbOKOnly + vbExclamation + vbDefaultButton1, PROGRAMMNAME
  GoTo Beenden                                          'Funktion korrekt ausgefuehrt:
Else                                                    'Frage: Funktion beenden?
  If MsgBox("Auf Entwickleransicht umgeschaltet." & vbCrLf & vbCrLf & _
            "Testfunktion beenden? (nein = nochmals umschalten)", _
            vbYesNo + vbQuestion + vbDefaultButton2, PROGRAMMNAME) = vbYes Then
    GoTo Beenden                                        'Wenn ja, dann Funktion
  End If                                                'beenden lassen. Sonst:
End If
                                                        'Funktion von vorne nochmals
GoTo Wiederholen                                        'durchlaufen lassen.

                                                        '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.
```

5.7.) Die Prozedur: Workbook_BeforeClose

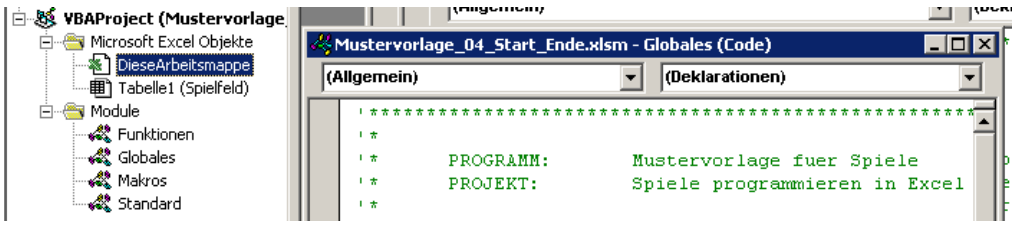
Ein nicht zu vernachlässigender Punkt ist es sich zu überlegen, was geschehen soll, wenn wir Excel beenden bzw. wenn das Spiel zu Ende gespielt worden ist und Excel automatisch beendet werden soll.

Überlegen wir uns mal folgendes: Wir spielen in der Spielansicht und verlieren, das Spiel gibt die Meldung "Spiel wird beendet!" aus und nach Bestätigung der Meldung wird die Excel-Datei geschlossen.

Danach öffnen wir eine andere Excel-Datei. Erstaunt (oder auch nicht) werden wir feststellen, daß sich Excel nach wie vor in der Spielansicht angezeigt wird. Nun haben wir aber in dieser Excel-Datei keine Funktionen zur Verfügung über welche wir die Entwickleransicht wieder herstellen könnten. Es bleibt lediglich der mühsame Umweg über die Excel-Optionen die einzelnen Ansichtseinstellungen manuell wieder herzustellen. Daher müssen wir dafür sorgen, daß vor dem Schließen der Datei die ursprünglichen Ansichtseinstellungen automatisch zurückgesetzt werden.

Ein weiterer Punkt ist es sich zu überlegen, ob beim Schließen der Datei diese automatisch gespeichert werden soll oder nicht. Soll das Spiel immer mit den gleichen Einstellungen gestartet werden, sollte nicht gespeichert werden. Hinterlegt man jedoch eine Highscore-Tabelle so ist es unbedingt notwendig die Datei zu speichern, damit der Highscore erhalten bleibt. Genauso ist es bei Spielen mit vielen Levels, hier soll sichergestellt werden, daß man nicht jedes Mal von vorne beginnen muß sondern beim letzten Level weiterspielen kann.

Zum Glück bietet uns Excel eine Methode an um obige Überlegungen relativ einfach umsetzen zu können: die Prozedur **Workbook_BeforeClose**. Um die Prozedur zu erstellen doppelklicken Sie in **Visual Basic** im **Projekt-Explorer** das Microsoft Excel Objekt: **DieseArbeitsmappe**:



Wählen Sie dann rechts im Code-Fenster der Combobox (Allgemein) "Workbook" aus und daneben in der Combobox (Deklarationen) "BeforeClose" aus. Folgender Code wird automatisch generiert:

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)

End Sub
```

Um die ursprünglichen Ansichtseinstellungen wieder herzustellen verwenden wir die zuvor entwickelte Funktion **SpielAnsichtAktualisieren** mit dem Parameter **NEIN** – Entwickleransicht anzeigen:

```
Call SpielAnsichtAktualisieren(NEIN)          'Umschalten auf Entwickleransicht.
```

Ob automatisch gespeichert werden soll legen wir im Modul **Globales** in einer globalen Konstante fest:

```
                                              'Verhalten beim BENDEN:
Global Const AUTOSPEICHERN = JA               'Beim Beenden autom. alles speichern.
'Global Const AUTOSPEICHERN = NEIN            'Aenderungen automatisch Verwerfen
'Global Const FRAGESPEICHERN = NEIN           'wenn auch FRAGESPEICHERN = NEIN oder
Global Const FRAGESPEICHERN = JA              'zusaetzlich Nachfrage ob Speichern.
```

Diese Konstanten können folgend ausgewertet werden:

```
If AUTOSPEICHERN = JA Then                    'Je nach Definition im Modul Globales
  Application.DisplayAlerts = False           'die aktuelle Arbeitsmappe automatisch
  ActiveWorkbook.Save                         'speichern oder alle Aenderungen
  Application.DisplayAlerts = True            'verwerfen. Dabei die Ausgabe von
                                              'Hinweismeldungen deaktivieren.
ElseIf FRAGESPEICHERN = NEIN Then             'Alle Aenderungen verwerfen:
  ThisWorkbook.Saved = True                   'Excel mitteilen, Datei = gespeichert
                                              '- auch wenn es nicht stimmt - daher
Else                                          'versucht Excel gar nicht zu speichern.
  'In diesem Else-Zweig stellt Excel die Frage
  'ob die Aenderungen gespeichert werden sollen.
End If
```

Noch ein Tipp am Rande: Nach Programmabstürzen oder wenn man eine Endlos-Schleife auf unsachgemäße Weise beendet hat kann es vorkommen, daß in Excel hinterlegte Ereignisse nicht mehr ausgeführt werden. Zum Beispiel bei Doppelklick auf oder Aktualisierung von Zellen. Gelegentlich habe ich erlebt, daß diese Ereignisse auch nach erneutem Öffnen deaktiviert bleiben. In der Prozedur **Workbook_Open** (wo dieser Befehl meiner Meinung nach hingehört) wurde dieses Problem scheinbar nicht gelöst. Erst nachdem ich ihn in der **Workbook_BeforeClose** hinterlegt habe trat dieser Fehler nicht mehr auf.

Zufall oder nicht – hier der Befehl:

```
                                        'Nach Programmabsturz kann dieser
Application.EnableEvents = True         'Befehl sehr hilfreich sein!
```

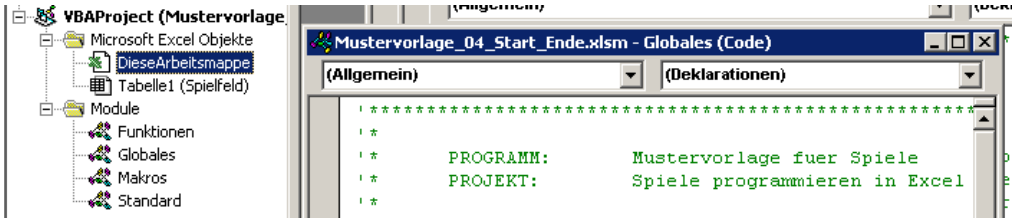
Abschließend noch einmal der vollständige Programmcode wie die **Workbook_BeforeClose** Prozedur aussehen könnte:

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION - PROZEDUR              */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       Workbook_BeforeClose                             */
'*                                                                       */
'*      BESCHREIBUNG:   Diese Prozedur wird vor den Schliessen der Excel-*/
'*                      Datei bzw. vor dem Beenden von Excel aufgerufen. */
'*                      Wird in dieser Prozedur der Parameter: Cancel    */
'*                      auf True gesetzt, so wird das Schliessen bzw. das*/
'*                      Beenden abgebrochen - Excel-Datei bleibt geoeffnet.*/
'*                                                                       */
'*      RETURN          nichts                                           */
'*                                                                       */
'*      PARAMETER:      Boolean ..... True = Funktion abbrechen          */
'*                                                                       */
'*-----------------------------------------------------------------------*/
On Error GoTo Fehler                            'Bei Fehler zur Fehlerbehandlung.

Call SpielAnsichtAktualisieren(NEIN)            'Umschalten auf Entwickleransicht.

If AUTOSPEICHERN = JA Then                      'Je nach Definition im Modul Globales
  Application.DisplayAlerts = False             'die aktuelle Arbeitsmappe automatisch
  ActiveWorkbook.Save                           'speichern oder alle Aenderungen
  Application.DisplayAlerts = True              'verwerfen. Dabei die Ausgabe von
                                                'Fehler- & Hinweismeldungen deaktivieren.
ElseIf FRAGESPEICHERN = NEIN Then               'Alle Aenderungen verwerfen:
  ThisWorkbook.Saved = True                     '(Excel mitteilen, dass Datei gespeichert
Else                                            'ist - auch wenn es nicht stimmt - daher
  'In diesem Else-Zweig stellt Excel die Frage  'versucht Excel gar nicht zu speichern.)
  'ob die Aenderungen gespeichert werden sollen.
End If
                                                'Nach Programmabsturz kann dieser
Application.EnableEvents = True                 'Befehl sehr hilfreich sein!


                                                'Fehlerbehandlungsroutine:
Beenden:                                        '========================
  Application.Cursor = xlDefault                'Sanduhr ausblenden.
  Exit Sub                                      'Prozedur abbrechen.

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

5.8.) Wechsel zwischen Spiel-Datei und anderen Excel-Dateien:

Ebenfalls sollte man berücksichtigen, daß ein Anwender zusätzlich zum Spiel auch noch weitere Excel-Dateien geöffnet haben kann. Da wir den anderen Excel-Anwendungen unsere Spielansicht nicht aufzwingen wollen, müssen wir dafür Sorge tragen dies zu verhindern.

Auch dafür bietet Excel die ideale Lösung mit vordefinierten Prozeduren. Um die Prozeduren, welche wir benötigen, zu erstellen doppelklicken Sie in **Visual Basic** im **Projekt-Explorer** das Microsoft Excel Objekt: **DieseArbeitsmappe**:



Wählen Sie dann rechts im Code-Fenster der Combobox (Allgemein) "Workbook" aus und daneben in der Combobox (Deklarationen) zuerst "Activate" und dann "Deactivate" aus. Folgender Code wird automatisch generiert:

```
Private Sub Workbook_Activate()

End Sub


Private Sub Workbook_Deactivate()

End Sub
```

In der Prozedur **Workbook_Activate**, welche gestartet wird, wenn wir von einer anderen Excel-Datei zurück auf die Spiel-Datei wechseln, legen wir fest, daß wir jene Ansicht sehen wollen, wie wir sie zuvor eingestellt haben (entweder Spielansicht oder Entwickleransicht). Zusätzlich maximieren wir das Excel-Fenster.

```
Call SpielAnsichtAktualisieren(SPIELANSICHT)        'Umschalten auf Entwickleransicht
                                                    'oder auf Spielansicht, je nach
                                                    'Definition im Modul Globales.
                                                    'Wenn gewuenscht, dann
Application.WindowState = xlMaximized               'Excel Fenster immer maximieren.
```

In der Prozedur **Workbook_Deactivate**, welche gestartet wird, sobald wir auf eine andere Excel-Datei wechseln, legen wir fest, daß der Vollbildmodus ausgeschaltet werden soll und stellen die ursprüngliche Fenstergröße wieder her.

```
Application.DisplayFullScreen = False               'Vollbildmodus ausschalten.
Application.WindowState = lExcelFensterGroesse      'Fenstergroesse wiederherstellen.
```

So! - Das wär´s dann für den Moment. Ich denke wir haben jetzt alles was die Ansichten betrifft abgearbeitet und können nun zum nächsten Schritt kommen. Vorab aber nochmals die vollständigen zwei neuen Prozeduren: (Siehe auch Datei: Mustervorlage_04_Start_Ende.xlsm)

```
Private Sub Workbook_Activate()
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION - PROZEDUR              */
'*************************************************************************/
'*----------------------------------------------------------------------*/
'*                                                                      */
'*      FUNKTION:       Workbook_Activate                               */
'*                                                                      */
'*      BESCHREIBUNG:   Diese Prozedur wird aufgerufen, wenn man aus einer  */
'*                      anderen geoeffneten Excel-Datei zu dieser zurueck-  */
'*                      wechselt. Egal ob ein Excel gestartet wurde und     */
'*                      damit mehrere Dateien geoeffnet sind oder ob Excel  */
'*                      mehrmals gestartet wurde und mit jedem Excel nur    */
'*                      eine Datei geoeffnet wurde.                         */
'*                                                                      */
'*                      Dieses Prozedur soll dafuer sorgen, dass die        */
'*                      Anzeigeeinstellungen fuer diese Datei automatisch   */
'*                      wiederhergestellt werden.                           */
'*                                                                      */
'*      RETURN          nichts                                          */
'*                                                                      */
'*      PARAMETER:      keine                                           */
'*                                                                      */
'*----------------------------------------------------------------------*/
On Error GoTo Fehler                            'Bei Fehler zur Fehlerbehandlung.

Call SpielAnsichtAktualisieren(SPIELANSICHT)    'Umschalten auf Entwickleransicht
                                                'oder auf Spielansicht, je nach
                                                'Definition im Modul Globales.
                                                'Wenn gewuenscht, dann
Application.WindowState = xlMaximized           'Excel Fenster immer maximieren.

                                                'Fehlerbehandlungsroutine:
Beenden:                                        '=========================
  Application.Cursor = xlDefault                'Sanduhr ausblenden.
  Exit Sub                                      'Prozedur abbrechen.

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

Private Sub Workbook_Deactivate()
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION - PROZEDUR              */
'*************************************************************************/
'*----------------------------------------------------------------------*/
'*                                                                      */
'*      FUNKTION:       Workbook_Deactivate                             */
'*                                                                      */
'*      BESCHREIBUNG:   Diese Prozedur wird aufgerufen, wenn man aus dieser */
'*                      Excel-Datei zu einer anderen geoeffneten Excel-Datei*/
'*                      wechselt. Egal ob ein Excel gestartet wurde und     */
'*                      damit mehrere Dateien geoeffnet sind oder ob Excel  */
'*                      mehrmals gestartet wurde und mit jedem Excel nur    */
'*                      eine Datei geoeffnet wurde.                         */
'*                                                                      */
'*                      Dieses Prozedur soll dafuer sorgen, dass die        */
'*                      Anzeigeeinstellungen fuer diese Datei NICHT in die  */
'*                      anderen Excel-Datei uebernommen wird.               */
'*                                                                      */
'*      RETURN          nichts                                          */
'*                                                                      */
'*      PARAMETER:      keine                                           */
'*                                                                      */
'*----------------------------------------------------------------------*/
```

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

Application.DisplayFullScreen = False       'Vollbildmodus ausschalten.
Application.WindowState = lExcelFensterGroesse   'Excel Fenstergroesse wiederherstellen.


                                            'Fehlerbehandlungsroutine:
Beenden:                                    '=========================
  Application.Cursor = xlDefault            'Sanduhr ausblenden.
  Exit Sub                                  'Prozedur abbrechen.

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

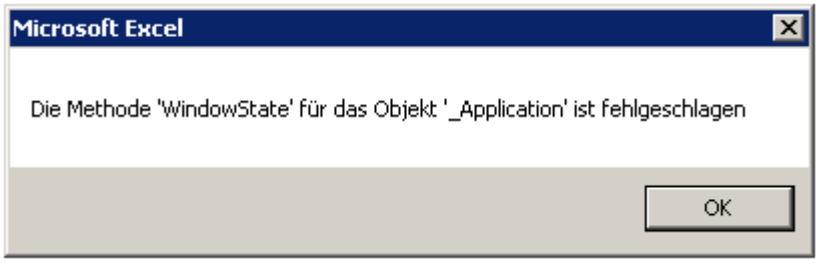
5.9.) Testen der bisherigen Prozeduren und Funktionen:

Um unsere Datei ausführlich zu testen legen Sie sich mehrere Kopien der Datei an und stellen dort unterschiedliche Zoom-Faktoren und Spielansichten (JA / NEIN) ein.

Öffnen Sie alle Dateien gleichzeitig und auch andere Excel-Dateien. Wechseln Sie willkürlich zwischen den einzelnen Dateien hin und her und beobachten Sie wie sich die Ansichten jeweils verhalten.

Sollte Ihnen auffallen, daß etwas nicht nach Ihren Wünschen ist oder Sie haben zusätzliche Ideen wie die Ansichten noch besser gestaltet werden können, dann bitte selbst machen – die Grundlagen wie man Ansichten programmiert sollten Ihnen nun geläufig sein.

Achtung:



Diese Fehlermeldung erhalten Sie (2 x hintereinander) wenn im Modul Globales eine Konstante oder eine Variable geändert, gelöscht oder hinzugefügt worden ist und danach die Excel-Datei geschlossen wird. Auslöser ist die Prozedur Workbook_BeforeClose.

Ursache: Beim Öffnen der Datei wird die Prozedur Worlbook_Open ausgeführt und dabei merkt sich Excel die globalen Definitionen. Nachdem etwas im Modul Globales geändert worden ist stehen diese Informationen nicht mehr zur Verfügung. Daher die Fehlermeldung welche mit OK quittiert werden kann. Beim nächsten Öffnen und Schließen kommt die Fehlermeldung nicht mehr.

Abhilfe:
Die Datei bei gedrückter Großschreibtaste öffnen – dann erst Änderungen im Modul Globales machen.

6.) Weitere Standard-Funktionen:

Es gibt viele Dinge in Excel die man immer wieder benötigt. Dazu schreibt man sich am besten eine Funktion und legt sie im Modul **Standard** ab. Dazu gehören vor allem die Umwandlung eines Spaltenbuchstaben in eine Spaltennummer und umgekehrt.

Weitere Standardfunktionen können sein: alle Tabellenblätter aus- oder einblenden, den Blattschutz für alle Tabellenblätter setzen bzw. aufheben (mit oder ohne Paßwort).

Hier mal ohne weitere Erklärungen meine fertigen Funktionen die ich immer benötige:
(Siehe auch Datei: Mustervorlage_05_Standardfunktionen_Farben.xlsm)

```
Function Buchstabe_aus_Spaltennummer(iSpalte As Integer) As String
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       Buchstabe_aus_Spaltennummer                      */
'*                                                                       */
'*      BESCHREIBUNG:   Liefert Spaltenbuchstabe aus Spaltennummer       */
'*                                                                       */
'*      RETURN          String ........... Buchstabe der Spalte          */
'*                                         '' Leerstring bei Fehler      */
'*                                                                       */
'*      PARAMETER:      Integer .......... Spaltennummer                 */
'*                                                                       */
'*-----------------------------------------------------------------------*/
On Error GoTo Fehler                            'Bei Fehler zur Fehlerbehandlung.
Buchstabe_aus_Spaltennummer = ""                'Returnwert = Fehler definieren.

'Maximale Spalte: XFD = 16384 max. Spalte bei Excel 2007
'Entspricht dem halben, maximal positiven Integer Wert

                                                'Zu grosse Spaltennummer:
If iSpalte > 16384 Then Buchstabe_aus_Spaltennummer = "": Exit Function

                                                'Zu keine Spaltennummer:
If iSpalte < 1 Then Buchstabe_aus_Spaltennummer = "": Exit Function

                                                'Spaltennummer auslesen:
Buchstabe_aus_Spaltennummer = Replace(Cells(1, iSpalte).Address(0, 0), "1", "")


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

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

```
Function Spaltennummer_aus_Buchstabe(stSpalte As String) As Integer
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       Spaltennummer_aus_Buchstabe                      */
'*                                                                       */
'*      BESCHREIBUNG:   Liefert Spaltennummer aus Spaltenbuchstaben      */
'*                                                                       */
'*      RETURN          Integer ..... Spaltennummer                      */
'*                                    0 bei Fehler                       */
'*                                                                       */
'*      PARAMETER:      String ..... Spaltenbuchstabe(n)                 */
'*                                                                       */
'*-----------------------------------------------------------------------*/
On Error Resume Next                                  'Bei Fehler einfach weitermachen.

Err = 0                                               'Fehlervariable ruecksetzen.
Spaltennummer_aus_Buchstabe = Columns(stSpalte).Column 'Spaltennummer auslesen.
If Err <> 0 Then                                      'Falls dabei ein Fehler auftritt,
  Spaltennummer_aus_Buchstabe = 0                     'dann Rueckgabewert 0 setzen.
End If

End Function                                          'Funktionsende.
```

Man beachte: obige Funktion hat keine Fehlerbehandlungsroutine. Begründung: Der einzige Fehler der auftreten kann wird direkt abgefangen.

7.) Farben:

Excel stellt Farben immer mit einem Farbwert dar. Dabei muß man jedoch unterscheiden ob ein Farbwert für **Color** oder **ColorIndex** übergeben wird. Ab Excel 2007 wird der **ColorIndex** nicht mehr wirklich unterstützt, kann aber aus Kompatibilitätsgründen weiter verwendet werden. Die üblichen Excel-Befehle zur Farbgestaltung sehen in etwa so aus:

```
Range("A1").Select                          'vor Excel 2007 ENTWEDER:
Selection.Interior.ColorIndex = 6           'Gelb fuer Zellenfuellung.
Selection.Font.ColorIndex = 3               'Rot  für Schriftfarbe.

Range("A2").Select                          'ODER:
Selection.Interior.Color = 65535            'Gelb fuer Zellenfuellung.
Selection.Font.Color = 255                  'Rot  für Schriftfarbe.

Range("A3").Select                          'ab Excel 2007 kein ColorIndex, ausser:
Selection.Interior.ColorIndex = xlColorIndexAutomatic 'schwarz
Selection.Font.ColorIndex = xlColorIndexNone          'weiss
                                            'UND die Indexwerte vor Excel 2007.

Range("A4").Select                          'dafür aber allgemeine vb-Farbkonstanten:
Selection.Interior.Color = vbYellow         'Gelb fuer Zellenfuellung.
Selection.Font.Color = vbRed                'Rot  für Schriftfarbe.

Range("A5").Select                          'Standard ist Zuweisung mit RGB-Funktion:
Selection.Interior.Color = RGB(255, 255, 0) 'Gelb fuer Zellenfuellung.
Selection.Font.Color = RGB(255, 0, 0)       'Rot  für Schriftfarbe.
```

Ab Excel 2007 empfehle ich daher den **ColorIndex** nicht mehr zu verwenden sondern nur mehr über die Eigenschaft **Color** zu arbeiten. Dabei bietet sich folgende Vorgangsweise an:

Zuerst globale Konstanten für die einzelnen Farbwerte definieren (im Modul **Globales**):

```
Global Const GELB = 65535        'Farbe gelb = RGB(255, 255, 0); Excel ColorIndex = 6
Global Const ROT = 255           'Farbe rot = RGB(255, 0, 0); Excel ColorIndex = 3
```

Und dann in den Funktionen diese Werte zuweisen:

```
Selection.Interior.Color = GELB 'Gelb fuer Zellenfuellung.
Selection.Font.Color = ROT      'Rot  für Schriftfarbe.
```

Natürlich kann man die ca. 16 Millionen möglichen Farben auch weiterhin über die RGB-Funktion direkt festzulegen. Im Anhang 3 siehe die in Excel hinterlegten Konstanten für Farb- Linien- und Rahmeneinstellungen.

Hier nun meine Empfehlungen für die Farbdefinitionen. Wer die Farben sehen oder mehr darüber wissen will sei auf die Dateien: Farben.xls und VBA_Code_Einfaerben_Excel.doc verwiesen. (Siehe auch Datei: Mustervorlage_05_Standardfunktionen_Farben.xlsm)

```
'-------------------------------------------------------------------------/
'-  Definitionen meiner Farbwerte fuer Word und Excel ab Version 2007      -/
'-------------------------------------------------------------------------/
Global Const WEISS = 16777215               'Farbe weiss = RGB(255, 255, 255); Excel ColorIndex = 2
Global Const ZARTGRAU = 15921906            'Farbe zartgrau = RGB(242, 242, 242);
Global Const HELLGRAU = 14211288            'Farbe hellgrau = RGB(216, 216, 216);
Global Const MITTELGRAU = 12566463          'Farbe mittelgrau = RGB(191, 191, 191); Excel ColorIndex = 15
Global Const GRAU = 10855845                'Farbe grau = RGB(165, 165, 165); Excel ColorIndex = 48
Global Const DUNKELGRAU = 8355711           'Farbe dunkelgrau = RGB(127, 127, 127); Excel ColorIndex = 16
Global Const SCHWARZ = 0                    'Farbe schwarz = RGB(0, 0, 0); Excel ColorIndex = 1

Global Const ZARTGELB = 14221311            'Farbe zartgelb = RGB(255, 255, 216); Excel ColorIndex = 19
Global Const HELLGELB = 10092543            'Farbe hellgelb = RGB(255, 255, 153); Excel ColorIndex = 36
Global Const GELB = 65535                   'Farbe gelb = RGB(255, 255, 0); Excel ColorIndex = 6
Global Const DUNKELGELB = 62207             'Farbe dunkelgelb = RGB(255, 242, 0);

Global Const GOLD = 56831                   'Farbe gold = RGB(255, 221, 0); Excel ColorIndex = 44
Global Const ORANGE = 39423                 'Farbe orange = RGB(255, 153, 0); Excel ColorIndex = 45

Global Const ZARTROT = 13421823             'Farbe zartrot = RGB(255, 204, 204);
Global Const ZARTROSA = 16764159            'Farbe zartrosa = RGB(255, 204, 255);
Global Const ROSA = 13408767                'Farbe rosa = RGB(255, 153, 204); Excel ColorIndex = 38
Global Const ROT = 255                      'Farbe rot = RGB(255, 0, 0); Excel ColorIndex = 3
Global Const BRAUN = 3368601                'Farbe braun = RGB(153, 102, 51); Excel ColorIndex = 12

Global Const ZARTGRUEN = 13434828           'Farbe zartgruen = RGB(204, 255, 204); Excel ColorIndex = 35
Global Const HELLGRUEN = 65280              'Farbe hellgruen = RGB(0, 255, 0); Excel ColorIndex = 4
Global Const GRUEN = 39168                  'Farbe gruen = RGB(0, 153, 0);
Global Const DUNKELGRUEN = 26112            'Farbe dunkelgruen = RGB(0, 102, 0); Excel ColorIndex = 10
Global Const OLIV = 52377                   'Farbe oliv = RGB(153, 204, 0); Excel ColorIndex = 43

Global Const ZARTBLAU = 16777164            'Farbe zartblau = RGB(204, 255, 255); Excel ColorIndex = 20
Global Const HELLBLAU = 16776960            'Farbe hellblau = RGB(0, 255, 255); Excel ColorIndex = 8
Global Const BLAU = 16711680                'Farbe blau = RGB(0, 0, 255); Excel ColorIndex = 5
Global Const DUNKELBLAU = 10027008          'Farbe dunkelblau = RGB(0, 0, 153); Excel ColorIndex = 11

Global Const MAGENTA = 16711935             'Farbe magenta = RGB(255, 0, 255); Excel ColorIndex = 7
Global Const LILA = 16711884                'Farbe lila = RGB(204, 0, 255);
Global Const VIOLETT = 13369446             'Farbe violett = RGB(102, 0, 204); Excel ColorIndex = 13
Global Const TUERKIS = 13421619             'Farbe tuerkis = RGB(51, 204, 204); Excel ColorIndex = 42

Global Const FARBE_HINTERGRUND = 16777215   'entspricht: keine Fuellung = WEISS
'Global Const FARBE_HINTERGRUND = 16316664  '16316664 = ein etwas anderes WEISS
Global Const HALBTRANSPARENT = 8421504      'Farbe irgendwo zwischen dunkelgrau und schwarz,
                                            'ziemlich gut sichtbar, aber deutlich heller.
```

8.) Zellen mit Namen versehen (Namens-Manager):

Folgendes Problem: Führt man in einem Tabellenblatt Berechnungen mit Formeln aus und fügt später Zeilen oder Spalten hinzu bzw. man löscht welche, so korrigiert Excel automatisch die Zellenbeziehung in den verwendeten Formeln. Wird jedoch eine Berechnung im VBA-Code ausgeführt, zum Beispiel **Sum("A1:A5"),** so wird der Programmcode natürlich nicht aktualisiert. Das heißt, nach jedem Einfügen oder Entfernen einer Zeile oder Spalte muß der gesamte Programmcode auf notwendige Anpassungen manuell überprüft und überarbeitet werden.

Eine einfache Möglichkeit sich diese Arbeit zu ersparen besteht im Festlegen von globalen Konstanten:

```
Global Const SUMME_START = "A1"
Global Const SUMME_ENDE = "A5"
Global Const SUMME_AUSGABE = "A6"

Function Test()
Dim stBereich As String

stBereich = SUMME_START & ":" & SUMME_ENDE
Range(SUMME_AUSGABE).Formula = "=Sum(" & stBereich & ")"

'oder einfacheres Beispiel mit direkter Wertzuweisung:
Range(SUMME_AUSGABE) = 0

End Function
```

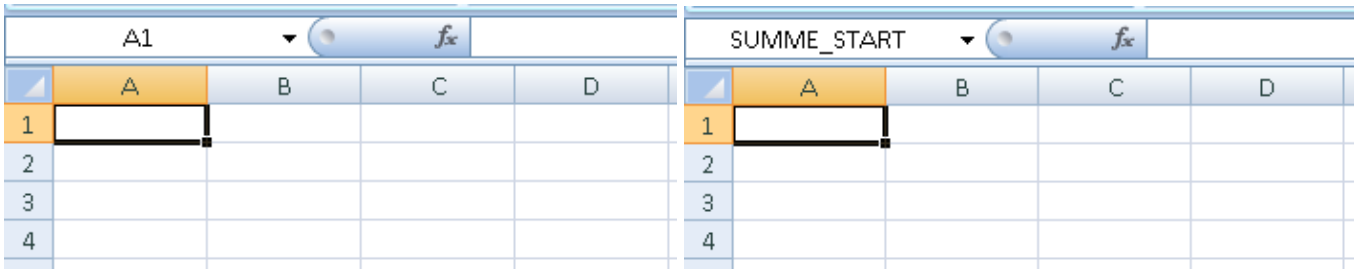
Bei dieser Methode müssen nur mehr die globalen Konstanten angepaßt werden.

Die beste Methode ist jedoch jene Zellen, welche über den Programmcode angesprochen werden, mit einem eigenen, für sich sprechenden Namen zu versehen. Obiger Code würde dann so aussehen und muß nicht mehr geändert oder angepaßt werden:

```
Function Test()
Dim stFormel As String

Set oTabellenblatt = Sheets("Tabelle1")

stFormel = "=Sum("
stFormel = stFormel & Buchstabe_aus_Spaltennummer(oTabellenblatt.Range("SUMME_START").Column)
stFormel = stFormel & oTabellenblatt.Range("SUMME_START").Row
stFormel = stFormel & ":"
stFormel = stFormel & Buchstabe_aus_Spaltennummer(oTabellenblatt.Range("SUMME_ENDE").Column)
stFormel = stFormel & oTabellenblatt.Range("SUMME_ENDE").Row
stFormel = stFormel & ")"

oTabellenblatt.Range("SUMME_AUSGABE").Formula = stFormel

'oder einfacheres Beispiel mit direkter Wertzuweisung:
oTabellenblatt.Range("SUMME_AUSGABE") = 0

End Function
```

In diesen Fall benötigt man keine globalen Konstanten und der Code muß auch nicht angepaßt werden. Es ist lediglich darauf zu achten, daß die definierten Namen im Tabellenblatt in den richtigen Zellen hinterlegt sind. Dazu kann der **Namens-Manager** von Excel verwendet werden.

Wie vergibt man nun Namen für Zellen? Wir nehmen uns ein leeres Tabellenblatt und selektieren die Zelle A1. Im Namens-Bereich wird A1 angezeigt. Diesen Eintrag überschreiben wir nun mit **SUMME_START** und schließen die Eingabe mit der Bestätigungstaste (Enter, Return) ab. Nun wird jedesmal der Name der Zelle angezeigt sobald man sie selektiert.



Wiederholen Sie die Namensvergabe für Zelle A3 = **SUMME_ENDE** und A4 = **SUMME_AUSGABE** und klicken dann auf das kleine Dreieck rechts im Namens-Bereich. Sie sehen nun eine Auflistung aller in der Excel-Datei vergebenen Namen.

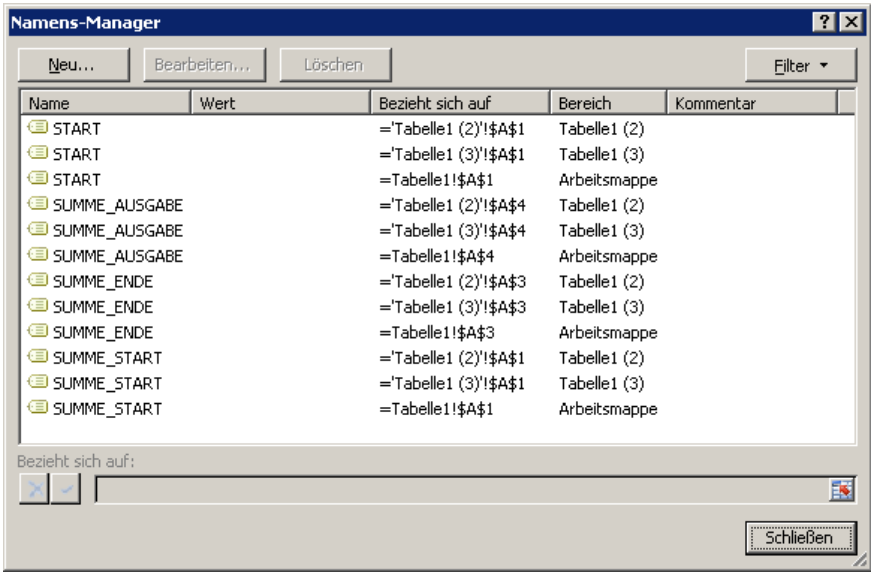
Namen kann man aber auch für Bereiche definieren. Dazu selektiert man zuerst alle gewünschten Zellen (zusammenhängend oder auch nicht) und vergibt dann einen Namen.

Bei der Vergabe von Namen ist folgendes zu berücksichtigen: Eine Standard-Zellenbezeichung wie zum Beispiel A5 kann nicht auf C3 geändert werden. Außerdem darf der Name nicht bereits in einem Modul, einer Funktion oder Prozedur oder als globale Variable oder Konstante verwendet worden sein. Auch andere für Excel reservierte Wörter dürfen Sie nicht verwenden.

Hinweis: Da der Namens-Bereich nicht verbreitert werden kann, sollten Sie darauf achten, nicht all zu lange Namen zu verwenden; sie sollten immer vollständig angezeigt werden können.

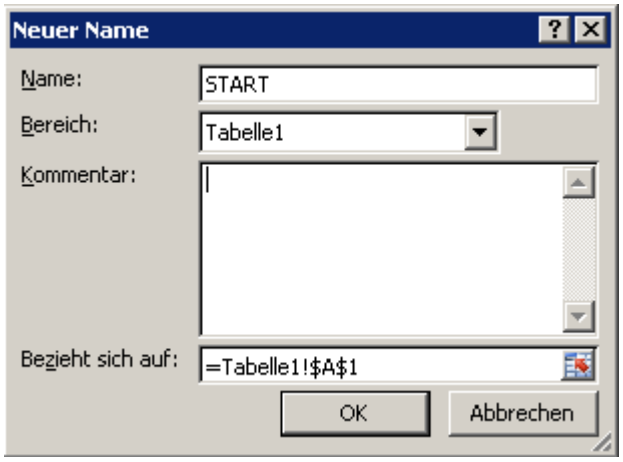
Vergeben sie nun nochmals einen Namen für die Zelle A1 indem Sie **SUMME_START** auf **START** ändern. Wie Sie sehen, kann eine Zelle auch mehrere Namen haben. Auf diese Art der Namensvergabe können jedoch Zellen auf unterschiedlichen Tabellenblättern nicht mit demselben Namen hinterlegt werden. Da dies aber manchmal sinnvoll ist wenden wir vorab einen Trick an um zu sehen wie die Namen von Excel verwaltet werden: Dazu erstellen wir zwei Kopien des Tabellenblattes in welchem die Namen bereits vergeben wurden.

Öffnen wir nun den Namens-Manager mit Klick im Excel-Register: **Formeln** und auf **Namens-Manager**:



Die ursprünglich angelegten Namen wurden für den Bereich **Arbeitsmappe** (also für die gesamte Excel-Datei) hinterlegt, die Kopien hingegen nur für das entsprechende Tabellenblatt. Um gleiche Namen zu verwenden müssen wir daher den Bereich **Arbeitsmappe** auf den Tabellenblattnamen ändern.

Über die Schaltfläche **Bearbeiten** kann das leider nicht bewerkstelligt werden. Daher müssen wir den Namen für die **Arbeitsmappe** zuerst löschen und dann wieder neu anlegen – jetzt können wir auch den Bereich festlegen:



Sollte im **Namens-Manager** bei **Wert** oder **Bezieht sich auf** der Text **"#BEZUG"** aufscheinen, so bedeutet das, daß die Zelle mit dem Namen nicht mehr existiert (die Zelle wurde gelöscht). Diese Namen sollten gelöscht werden.

Name	Wert	Bezieht sich auf	Bereich	Kommentar
START		=Tabelle1!$A$1	Tabelle1	
START	#BEZUG!	='Tabelle1 (2)'!#BEZUG!	Tabelle1 (2)	
START		='Tabelle1 (3)'!$A$1	Tabelle1 (3)	

Auch doppelte Namensvergaben für eine Zelle können nur über den Namens-Manager gelöscht werden.

Während des Programmierens sollte man den Namens-Manager gelegentlich immer wieder öffnen um zu kontrollieren, ob die vergebenen Namen und Definitionen auch noch mit den beabsichtigten Werten übereinstimmen.

9.) Deklarierte Funktionen (API-Funktionen):

Für ein Spiel benötigt man hauptsächlich zwei API-Funktionen: **Sleep** (den Programmcode für eine bestimmte Zeit lang anhalten) zur Steuerung der Spielgeschwindigkeit und **GetAsyncKeyState** (stellt fest ob während des Programmablaufes eine bestimmte Taste gedrückt worden ist) zum Steuern des Spiels.

API-Prozeduren sind in separaten Dateien, meistens vom Typ *.dll oder *.exe, enthalten und befinden sich meist im Windows\System32 Verzeichnis. Um zum Beispiel die Funktion **Sleep** aus der Datei kernel.dll zu nutzen, muß im Deklarationsbereich eines Moduls mit der **Declare-Anweisung** auf sie verwiesen werden.

Wichtig! Bei der Deklaration von API-Funktionen muß man unterscheiden, ob die Anwendung (hier Excel) als 32 Bit oder 64 Bit Version ausgeführt wird. Um unser Spiel in beiden Versionen spielen zu können deklarieren wir die beiden Funktionen im Modul Globales folgendermaßen:

```
Option Explicit                                         'Alle Variablen muessen
                                                        'explizit angegeben werden.
'-------------------------------------------------------------------------------/
'-  Deklarierte API-Funktionen fuer 64 Bit Office Versionen                     -/
'-------------------------------------------------------------------------------/
#If VBA7 Then                                           'Fuer 64 Bit Excel
  Public Declare PtrSafe Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As LongPtr)
  Public Declare PtrSafe Function GetAsyncKeyState Lib "user32" (ByVal vKey As Long) As Integer

'-------------------------------------------------------------------------------/
'-  Deklarierte API-Funktionen fuer 32 Bit Office Versionen                     -/
'-------------------------------------------------------------------------------/
#Else                                                   'Fuer 32 Bit Excel
  'Fuer Sleep-Funktion.
  Public Declare Sub Sleep Lib "kernel32.dll" (ByVal dwMilliseconds As Long)
  'Fuer Tastendruckabfrage.
  Public Declare Function GetAsyncKeyState Lib "user32.dll" (ByVal vKey As Long) As Integer
#End If
```

Um die deklarierten Funktionen zu testen schreiben wir gleich eine Testfunktion, welche wir im Modul **Standard** hinterlegen werden. Damit diese Testfunktion auch einen Nutzen hat, gestalten wir sie so, daß sie später als Kopiervorlage für die Funktion **SpielStarten** herangezogen werden kann.

Die Funktion **Sleep** soll lediglich 1 Sekunde bzw. 1000 Millisekunden lang warten. Bei Spielen werden üblicherweise Werte von 50-400 ms verwendet und können als globale Konstante **GESCHWINDIGKEIT** im Modul **Globales** hinterlegt werden.

Für die Funktion **GetAsyncKeyState** erstellen wir eine Endlosschleife welche die 4 Richtungstasten abfragt und eine entsprechende Meldung ausgibt. In einem Spiel wird dann anstelle der Meldung eine Funktion aufgerufen, welche eine Spielfigur in eine bestimmte Richtung bewegt.

Wie beenden wir aber die Endlosschleife? Am besten wenn man die ESC-Taste drückt! Nur – die ESC-Taste bewirkt, daß der Programmcode angehalten und eine Fehlermeldung ausgegeben wird. Das können wir verändern, indem wir Excel mitteilen, daß beim Drücken der ESC-Taste die aktuelle Fehlerbehandlungsroutine aufgerufen werden soll:

```
Application.EnableCancelKey = xlErrorHandler            'Damit die ESC-Taste nicht den
                                                        'Programmcode unterbricht sondern
                                                        'das Spiel sofort beendet.
```

In der Fehlerbehandlung fragen wir den Fehlercode 18 ("Unterbrechung durch Benutzer") ab und teilen dem Programm mit was es zu machen hat. Eine Möglichkeit wäre gar nichts zu tun oder eine eigene Hinweismeldung ausgeben. Das Programm wird einfach beendet.

Eine andere Möglichkeit könnte sein das Spielprogramm zu schließen, was wir hier auch anwenden werden. (Sehr von Vorteil wenn man während der Arbeit spielt und der Chef kommt rein.)

Dazu verwenden wir den Befehl:

```
ActiveWorkbook.Close savechanges:=True                  'Excel-Datei sofort schliessen
                                                        '(ohne Nachfrage ob speichern).
```

Dabei werden zuvor noch die Prozeduren **Workbook_BeforeClose** und dann **Workbook_Deaktivate** ausgeführt.

Hier die vollständige Test-Funktion:

```
Public Function Teste_Tastendruck() As Boolean
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       Teste_Tastendruck                                */
'*                                                                       */
'*      BESCHREIBUNG:   Testet die deklarierten Funktionen: Tastendruck  */
'*                      (Alias: GetAsyncKeyState) und Sleep.             */
'*                                                                       */
'*      RETURN          Boolean ..... True = Funktion in Ordnung         */
'*                                    False = Fehler in Funktion         */
'*      PARAMETER:      keine                                            */
'*                                                                       */
'*-----------------------------------------------------------------------*/
On Error GoTo Fehler                            'Bei Fehler zur Fehlerbehandlung.
Teste_Tastendruck = True                        'Returnwert = alles OK vordefinieren.

Application.EnableCancelKey = xlErrorHandler    'Damit die ESC-Taste nicht den
                                                'Programmcode unterbricht sondern
                                                'das Spiel sofort beendet.

Wiederholen:                                    'Sprungmarke zum Wiederholen
                                                'der Funktion:

Sleep (1000)                                    'Eine Sekunde lang warten (1000 ms).

If GetAsyncKeyState(vbKeyRight) Then            'Wird RECHTS-Taste gedrueckt, dann
  MsgBox "nach rechts"                          'Meldung ausgeben
End If

If GetAsyncKeyState(vbKeyLeft) Then             'Wird LINKS-Taste gedrueckt, dann
  MsgBox "nach links"                           'Meldung ausgeben
End If

If GetAsyncKeyState(vbKeyDown) Then             'Wird DOWN-Taste gedrueckt, dann
  MsgBox "nach unten"                           'Meldung ausgeben
End If

If GetAsyncKeyState(vbKeyUp) Then               'Wird UP-Taste gedrueckt, dann
  MsgBox "nach oben"                            'Meldung ausgeben
End If

DoEvents                                        'Dem Programm zusaetzliche Rechenzeit
                                                'zur Verfuegung stellen, da sonst alles
                                                'abstuerzen koennte!
                                                'Funktion von vorne nochmals
GoTo Wiederholen                                'durchlaufen lassen.

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

Fehler:                                         'Fehlermarke - Fehlerbehandlung:
  If Err = 18 Then                              'Wenn ESC-Taste gedrueckt wurde:
    ActiveWorkbook.Close savechanges:=True      'Excel-Datei sofort schliessen.
  Else
    Teste_Tastendruck = False                   'Returnwert = Fehler setzen.
    MsgBox Err.Description                      'Fehlermeldung ausgeben.
  End If
  Resume Beenden                                'Bei Funktionsende weitermachen.
  Resume                                        'Nur fuer Testzwecke hinterlegt.
End Function                                    'Funktionsende.
```

10.) Sound und Musik abspielen:

Ein Spiel benötigt natürlich auch einen Sound. Dazu legen wir uns am besten ein neues Modul mit Namen **Sounds** an.

Die hier gezeigten Funktionen beziehen sich auf einfache Beep-Töne und Systemklänge. Aber auch das Abspielen beliebiger Sound-Dateien, wie z.B. MP3 oder WAV soll behandelt werden. Grundsätzlich ist die Wiedergabe ganzer Musikdateien mit der deklarierten Funktion **ShellExecute** möglich. Dabei wird ein bestimmtes Player-Programm gestartet, man übergibt die anzuspielende Sound-Datei und diverse Parameter. Diese Vorgangsweise hat den Nachteil, daß ein zusätzlicher Task in der Taskleiste eingeblendet wird und der Aufruf relativ kompliziert ist. Das Beenden gestaltet ist noch komplexer und daher verzichte ich auf **ShellExecute**. Es gibt andere, viel einfachere Methoden um ein Spiel mit Musik und Geräuschen zu hinterlegen.

10.1.) Beep und MessageBeep Funktion

VBA stellt von Haus aus die VBA-Funktion **Beep** zur Verfügung. Das Problem: Sie bewirkt nichts.

Das hat mit der langen, wechselhaften Entwicklung zu tun, wie Sounds auf Computern abgespielt werden können. **Beep** stammt nämlich noch aus der Frühzeit des Computers, als es noch nicht üblich war, in einen Computer eine Soundkarte einzubauen. Mit dem Aufkommen von Soundkarten sollte das interne Piepsen des PCs abgeschafft werden und zwischenzeitlich wurde **Beep** auch auf API-Ebene einfach deaktiviert. Wurde aber schon bald wieder eingeführt - nur eben nicht in VBA.

Als API-Funktion ist **Beep** jedenfalls längst wieder möglich. Die folgende Beispielfunktion zeigt aber auch einen Nachteil von **Beep** auf: Solange ein Ton abgespielt wird, wartet die Anwendung. Daher empfiehlt es sich die Beep-Funktion nur für kurze Tonausgaben zu verwenden.

Zuerst deklarieren wir die Funktion:

```
'Gibt einen Ton in bestimmter Laenge aus.
'Der Frequentbereich dwFreq liegt zwischen 37 und 32767
Public Declare Function Beep Lib "kernel32.dll" _
  (ByVal dwFreq As Long, ByVal dwDuration As Long) As Boolean
```

Dann legen wir uns noch einige Konstanten an um Töne einfacher ausgeben zu können:

```
'-------------------------------------------------------------------------/
'-  Globale Konstanten - Moegliche Parameter fuer: Beep (Notenlaenge)      -/
'-------------------------------------------------------------------------/
Global Const TONLAENGE = 500                'Dauer fuer viertel-Note: (Basiswert)
'Global Const DAUER_GANZ = 2000             'Dauer fuer ganze Note: 2 Sekunden
'Global Const DAUER_HALB = 1000             'Dauer fuer halbe Note: 1000 Millisek.
'Global Const DAUER_VIERTEL = 500           'Dauer fuer viertel-Note: (Basiswert)
'Global Const DAUER_ACHTEL = 250            'Dauer fuer achtel-Note:

'-------------------------------------------------------------------------/
'-  Globale Konstanten - Moegliche Parameter fuer: Beep (Tonhoehe)         -/
'-------------------------------------------------------------------------/
'Global Const TON_A4 = 440                  'Definition einzelner Noten:
'Um sich die Definition aller 120 Noten zu ersparen wird spaeter die Funktion:
'BerechneFrequenz verwendet. Diese berechnet anhand der GRUNDFREQUENZ, der Oktave
'(wird direkt im Programmcode definiert) und der Ton-Nummer die Frequenz zur Note:
```

```
Global Const GRUNDFREQUENZ = 30.7           'Bei Kammerton A = 440 Hz.
Global Const C = 1                          'Definitionen der Ton-Nummern:
Global Const Cis = 2
Global Const D = 3
Global Const Dis = 4
Global Const E = 5
Global Const F = 6
Global Const Fis = 7
Global Const G = 8
Global Const Gis = 9
Global Const A = 10
Global Const B = 11
Global Const H = 12
```

Und gleich vorab die Funktion für die Frequenzberechnung (siehe auch Tabellenblatt Töne):

```
Public Function FrequenzBerechnen(iOktave As Integer, iTon As Integer) As Long
FrequenzBerechnen = _
  Round(GRUNDFREQUENZ * (2 ^ (iOktave - 1)) * (2 ^ (1 / 12)) ^ (iTon - 1), 0)
End Function                                        'Funktionsende.
```

Und nun geben wir den ersten Ton aus:

```
Public Function DemoKammerton_A()
Dim iOktave As Integer                              'Aktuelle Oktave.

iOktave = 4                                         '1. Standardoktave = 4.
                                                    'Kammerton 5 Sek. lang ausgeben.
Beep FrequenzBerechnen(iOktave, A), TONLAENGE * 10  'Tonlaenge ¼ Note = 500 ms,
                                                    '500 ms * 10 = 5 Sekunden.
MsgBox "Fertig"                                     'Meldung ausgeben.

End Function                                        'Funktionsende.
```

Mit der Beep-Funktion kann man aber auch einfach ganze Lieder abspielen indem man eine Note nach der anderen abspielen läßt:

```
Public Function DemoGoetterfunken()
Dim iOktave             As Integer                  'Aktuelle Oktave.
iOktave = 4                                         '1. Standard-Oktave = 4.

Beep FrequenzBerechnen(iOktave, E), TONLAENGE       'Freu
Beep FrequenzBerechnen(iOktave, E), TONLAENGE       'de
Beep FrequenzBerechnen(iOktave, F), TONLAENGE       'schoe
Beep FrequenzBerechnen(iOktave, G), TONLAENGE       'ner
Beep FrequenzBerechnen(iOktave, G), TONLAENGE       'Goet
Beep FrequenzBerechnen(iOktave, F), TONLAENGE       'ter
Beep FrequenzBerechnen(iOktave, E), TONLAENGE       'fun
Beep FrequenzBerechnen(iOktave, D), TONLAENGE       'ken
Beep FrequenzBerechnen(iOktave, C), TONLAENGE       'Toch
Beep FrequenzBerechnen(iOktave, C), TONLAENGE       'ter
Beep FrequenzBerechnen(iOktave, D), TONLAENGE       'aus
Beep FrequenzBerechnen(iOktave, E), TONLAENGE       'E
Beep FrequenzBerechnen(iOktave, E), TONLAENGE * 1.5 'ly
Beep FrequenzBerechnen(iOktave, D), TONLAENGE / 2   'si
Beep FrequenzBerechnen(iOktave, D), TONLAENGE * 2   'um
End Function                                                    'Funktionsende.
```

Wie sie hören klingt das wirklich sehr primitiv – aber besser als ein stummes Spiel. Natürlich kann man auch Tonfolgen über eine Funktion berechnen lassen. Hier noch ein kleines Beispiel (ist noch sehr verbesserungswürdig):

```
Public Function DemoSirene()
Dim i As Integer                                    'Allgemeine Zaehlvariable.
Dim j As Integer                                    'Allgemeine Zaehlvariable.

For i = 1 To 3                                      'Sirene 3 mal abspielen:
  For j = 1000 To 4000 Step 200                     'Tonhoehe erhoehen und
    Beep j, 100                                     'abspielen.
  Next j
  For j = 4000 To 1000 Step -200                    'Tonhoehe verringern un
    Beep j, 100                                     'abspielen
  Next j
Next i                                              'Naechster Durchlauf
End Function                                        'Funktionsende.
```

Abschließend sei hier noch die Funktion **MessageBeep** erwähnt. Diese Funktion spielt einen Ton ab, in Abhängigkeit vom Soundschema, das der Anwender in den Windows Systemeinstellungen ausgewählt hat. Die Anwendung läuft weiter, ohne das Abspielen des Tons abzuwarten.

Zuerst deklarieren wir zuerst die Funktion:

```
'Spielt einen Ton ab, Abhängigkeit vom Soundschema in den Systemeinstellungen:
Public Declare Function MessageBeep Lib "user32.dll" _
  (ByVal uType As eType) As Boolean
```

Dann legen wir uns noch einige Konstanten an um die Systemtöne einfacher ansprechen zu können:

```
'-------------------------------------------------------------------------/
'-  Globale Konstanten - Moegliche Parameter fuer: MessageBeep           -/
'-------------------------------------------------------------------------/
Public Enum eType
  MB_OK = 0                   'Standard Sound
  MB_ICONERROR = &H10         'Kritischer Stop
  MB_ICONQUESTION = &H20      'Frage (gibt bei mir nichts aus)
  MB_ICONWARNING = &H30       'Warnung
  MB_ICONINFORMATION = &H40   'Information
End Enum
```

Und nun geben wir die 5 Systemklänge aus:

```
Public Function DemoMessageBeep()
MessageBeep (MB_OK)
MsgBox "OK", vbOKOnly

MessageBeep (MB_ICONERROR)
MsgBox "Fehler", vbOKOnly + vbCritical

MessageBeep (MB_ICONQUESTION)
MsgBox "Frage", vbYesNo + vbQuestion

MessageBeep (MB_ICONWARNING)
MsgBox "Warnung", vbOKOnly + vbExclamation

MessageBeep (MB_ICONINFORMATION)
MsgBox "Information", vbOKOnly + vbInformation
End Function                                        'Funktionsende.
```

Wichtig! Zuerst den Ton abspielen lassen und erst dann die Meldung ausgeben.

10.2.) Wave-Dateien abspielen

Zuerst deklarieren wir die Funktion zum Abspielen einer Wave-Datei: Achten Sie unbedingt darauf, daß als Alias "**sndPlaySoundA**" angegeben wird und nicht "**PlaySoundA**" – das ist eine ältere Version der Funktion und benötigt andere Parameter und Konstanten als hier beschrieben.

```
'Zum Abspielen einer WAV Datei:
Public Declare Function PlayWaveSound Lib "winmm.dll" _
  Alias "sndPlaySoundA" (ByVal lpszName As String, _
  ByVal dwFlags As Long) As Long
```

Dann legen wir uns noch die möglichen Konstanten für den Funktionsaufruf an:

```
'-------------------------------------------------------------------------/
'-  Globale Konstanten - Moegliche Parameter fuer: PlayWaveSound (Wave)     -/
'-------------------------------------------------------------------------/
Global Const SND_SYNC = &H0          '(Standard) Spielt gesamtes Lied synchron ab.
                                     'Code wird bis Ende des Liedes angehalten.
Global Const SND_ASYNC = &H1         'Spielt gesamtes Lied asynchron ab.
                                     'Code wird beim Starten NICHT angehalten.
Global Const SND_LOOP = &H8          'Lied in einer Endlos-Schleife wiedergeben bis
                                     'nächstes Lied aufgerufen wird.
Global Const SND_PURGE = &H40        'Lied in aktueller Endlos-Schleife stoppen.
```

Synchron bedeutet hier, daß das Lied gestartet wird und der Programmcode erst dann weiter ausgeführt wird, wenn das Lied zu Ende gespielt worden ist oder das Abspielen an anderer Stelle beendet wird. **Asynchron** startet das Lied und der Programmcode wird sofort weiter abgearbeitet. Daß ist jene Konstante, die wir beim Spielprogramm verwenden werden – zusammen mit **SND_LOOP**. Eine kleine Wave-Datei kann somit als Hintergrundmusik während des ganzen Spiels abgespielt werden. Bei Spielende stoppen wir dann die Wiedergabe mit dem Parameter **SND_PURGE**.

Zur Veranschaulichung hier eine Funktion um alle Möglichkeiten zu Testen:

```
Public Function DemoWaveAbspielen()
Dim stDateiname1        As String                    'Sound-Datei 1.
Dim stDateiname2        As String                    'Sound-Datei 2.
stDateiname1 = ActiveWorkbook.Path & "\Mist1.wav"    'Dateinamen fuer Musik 1.
stDateiname2 = ActiveWorkbook.Path & "\Musik.wav"    'Dateinamen fuer Musik 2.


Call PlayWaveSound(stDateiname, SND_SYNC)            'Spielt Musik bis zum Ende,
                                                     'Programmcode wird angehalten.
Call PlayWaveSound(stDateiname, SND_ASYNC)           'Spielt Musik bis zum Ende,
                                                     'Programmcode laeuft weiter.
Call PlayWaveSound(stDateiname2, SND_ASYNC Or SND_LOOP)'Musik in Endlos-Schleife,
                                                     'Programmcode laeuft weiter.
Sleep 10000                                          '10 Sekunden spielen lassen,
                                                     'dann:
Call PlayWaveSound(0, SND_PURGE)                     'Aktuelle Wiedergabe beenden.

End Function                                         'Funktionsende.
```

Der Nachteil dieser Funktion liegt darin, daß wir nicht mehrere Wave-Dateien gleichzeitig abspielen können. Daher verwenden wir sie nur zur Wiedergabe einer Hintergrundmusik. Mit der nächsten Funktion können zwar mehrere Dateien gleichzeitig abgespielt werden, hat aber keine LOOP-Funktion.

10.3.) MP3 und andere Musikformate abspielen

Zuerst deklarieren wir eine Funktion zum Abspielen von mp3-Dateien und anderer Musikformate:

```
'Zum Abspielen einer MP3 Datei:
Private Declare Function PlayMP3Sound Lib "winmm.dll" _
  Alias "mciSendStringA" (ByVal lpszCommand As String, _
  ByVal lpszReturnString As String, _
  ByVal cchReturnLength As Long, _
  ByVal hwndCallback As Long) As Long
```

Weiters benötigen wir eine Funktion, welche den Dateinamen in die alte DOS 8.3 Schreibweise umformt. Dies ist wichtig, da die Funktion zum Abspielen von Musikdateien bei Dateinamen mit Leerzeichen oder Sonderzeichen Probleme hat. Sollte man auch für die vorherige Funktion **PlayWaveSound** verwenden.

```
'Ermittelt den DOS 8.3 Dateinamen der abzuspielenden Datei. Wird benoetigt, da
'Dateinamen mit Leerzeichen bei den Abspielfunktionen Probleme bereiten koennen.
Private Declare Function GetShortPathName Lib "kernel32.dll" _
  Alias "GetShortPathNameA" (ByVal lpszLongPath As String, _
  ByVal lpszShortPath As String, _
  ByVal cchBuffer As Long) As Long
```

Und nun spielen wir einfach mal eine Musikdatei ab:

```
Public Function DemoMP3Abspielen() As Boolean
Dim stDateiname         As String                   'Sound-Datei.
Dim stAlias             As String                   'Aliasname fuer Sound-Datei.
Dim stBuffer            As String                   'Buffer fuer DOS 8.3 Dateinamen.

stDateiname = ActiveWorkbook.Path & "\Musik.mp3"    'Dateinamen für Musik festlegen.
stAlias = "Musik"                                   '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
  Call PlayMP3Sound("play " & stAlias & " from 0", 0, 0, 0) 'Zuerst MCI oeffnen und
End If                                              'wenn OK, dann MP3 abspielen.

Sleep 10000                                         '10 Sekunden spielen lassen.

PlayMP3Sound "stop " & stAlias, 0, 0, 0             'Wiedergabe stoppen und
PlayMP3Sound "close " & stAlias, 0, 0, 0            'MCI wieder schliessen.

End Function                                        'Funktionsende.
```

Wir gehen davon aus, daß sich die Musik-Datei im selben Verzeichnis befindet wie die Spiel-Datei. Damit können wir die Pfadangaben direkt aus Excel übernehmen. Die Dateinamen hinterlegen wir als globale Konstanten (mit "\" vor dem Namen). Zusätzlich benötigen wir für die Musik-Datei einen Alias-Namen. Das kann ein beliebiger Text sein, muß aber für jede Datei, die gleichzeitig abgespielt werden soll, unterschiedlich sein.

Dann wandeln wir den Dateinamen in die DOS 8.3 Schreibweise um.

Nun rufen wir das Multimedia-Interface-Control (MCI) auf und öffnen die abzuspielende Musik-Datei. Dabei wird dem Musikstück der Aliasname zugeordnet. Alle weiteren Befehle an das MCI benötigen dann nur mehr den Aliasnamen.

Mit dem Parameter "**play**" wird das Abspielen der Musik-Datei gestartet. Im Parameter "**from 0**" kann festgelegt werden ab welcher Position die Musik wiedergegeben werden soll. 0 bedeutet von Anfang an, 2000 würde die ersten 2 Sekunden überspringen. Danach wird die Musik bis zum Ende ausgegeben. Der Programmcode läuft weiter.

Mit dem Parameter "**stop**" wird das Abspielen der Musik-Datei beendet und kann dann mit "**play**" wieder neu gestartet werden. Leider kann man damit keine Endlos-Schleife für die Wiedergabe generieren.

Mit dem Parameter "**close**" wird die Musik-Datei endgültig aus dem MCI entfernt und der Aliasname wird für andere Verwendungen wieder freigegeben. Das Abspielen wird dabei natürlich auch beendet. Ein erneutes Abspielen der Musik-Datei ist erst dann wieder möglich, indem man den Aliasnamen mit dem Parameter "**open**" wieder an das MCI übergibt.

Sollte das MCI nicht korrekt beendet worden sein, so kommt es vor, daß "**play**" keine Musik abspielt, weil die Musik bereits vollständig abgespielt worden ist und das MCI noch nicht beendet wurde. Das läßt sich vermeiden indem man vor dem **Öffnen** ein zusätzliches **Schließen** ausführt.

```
PlayMP3Sound "close " & stAlias, 0, 0, 0            'MCI schliessen.
Call PlayMP3Sound("open " & stDateiname & " type MPEGVideo alias " & stAlias, 0,0,0)
Call PlayMP3Sound("play " & stAlias & " from 0", 0, 0, 0) 'MCI oeffnen und abspielen.
```

Abschließend noch eine Funktion, wie die Soundausgabe in Spielen gestaltet werden könnte:

```
Public Function DemoSounds() As Boolean
Dim stDateiname1        As String                   'Sound-Datei 1.
Dim stDateiname2        As String                   'Sound-Datei 2.
Dim stDateiname3        As String                   'Sound-Datei 3.

Dim stAlias1            As String                   'Aliasname fuer Sound-Datei 1.
Dim stAlias2            As String                   'Aliasname fuer Sound-Datei 2.
Dim stAlias3            As String                   'Aliasname fuer Sound-Datei 3.
Dim stBuffer            As String                   'Buffer fuer DOS 8.3 Dateinamen.

stDateiname1 = ActiveWorkbook.Path & "\Musik.mp3"   'Datei fuer Sound 1 festlegen.
stDateiname2 = ActiveWorkbook.Path & "\Mist3.mp3"   'Dateinamen fuer Musik 2 festlegen.
stDateiname3 = ActiveWorkbook.Path & "\Mist1.wav"   'Dateinamen fuer Musik 3 festlegen.

stAlias1 = "HintergrundMusik"                       'Aliasnamen fuer Hintergrund.
stAlias2 = "DreiMalMist"                            'Aliasnamen fuer Sound 2.
stAlias3 = "EinmalMist"                             'Aliasnamen fuer Sound 3.

stBuffer = Space$(255)                              'DOS 8.3 Dateinamen ermitteln:
If GetShortPathName(stDateiname1, stBuffer, Len(stBuffer)) <> 0 Then    'Wenn OK,
  stDateiname1 = Left$(stBuffer, InStr(stBuffer, vbNullChar) - 1)   'aus Buffer die
End If                                              'Leerzeichen am Ende entfernen.

stBuffer = Space$(255)                              'DOS 8.3 Dateinamen ermitteln:
If GetShortPathName(stDateiname2, stBuffer, Len(stBuffer)) <> 0 Then    'Wenn OK,
  stDateiname2 = Left$(stBuffer, InStr(stBuffer, vbNullChar) - 1)   'aus Buffer die
End If                                              'Leerzeichen am Ende entfernen.

stBuffer = Space$(255)                              'DOS 8.3 Dateinamen ermitteln:
If GetShortPathName(stDateiname3, stBuffer, Len(stBuffer)) <> 0 Then    'Wenn OK,
  stDateiname3 = Left$(stBuffer, InStr(stBuffer, vbNullChar) - 1)   'aus Buffer die
End If                                              'Leerzeichen am Ende entfernen.
```

```
                                                'Wave-Datei in Endlos-Schleife
Call PlayWaveSound(stDateiname1, SND_ASYNC Or SND_LOOP) 'als Hintergrundmusik starten:

Wiederholen:                                    'Zusaetzliche Toene ausgeben bis Beenden:

MessageBeep (MB_ICONINFORMATION)                'Ein Piepsen ausgeben, dann Fragen:
If MsgBox("Mist ausgeben?", vbYesNo + vbQuestion + vbDefaultButton2, PROGRAMMNAME) = vbYes
Then
  If Int((2 * Rnd) + 1) = 1 Then                'Einmal Mist ausgeben.
    PlayMP3Sound "stop " & stAlias2, 0, 0, 0    'Wiedergabe stoppen und MCI
    PlayMP3Sound "close " & stAlias2, 0, 0, 0   'schliessen.
    If PlayMP3Sound("open " & stDateiname2 & " type MPEGVideo alias " & stAlias2,0,0,0)=0 Then
      Call PlayMP3Sound("play " & stAlias2 & " from 0", 0, 0, 0) 'Zuerst MCI oeffnen und
    End If                                      'wenn OK, dann MP3 abspielen.
  Else                                          '3 mal Mist ausgeben.
    PlayMP3Sound "stop " & stAlias3, 0, 0, 0    'Wiedergabe stoppen und MCI
    PlayMP3Sound "close " & stAlias3, 0, 0, 0   'schliessen.
    If PlayMP3Sound("open " & stDateiname3 & " type MPEGVideo alias " & stAlias3,0,0,0)=0 Then
      Call PlayMP3Sound("play " & stAlias3 & " from 0", 0, 0, 0) 'Zuerst MCI oeffnen und
    End If                                      'wenn OK, dann MP3 abspielen.
  End If
  GoTo Wiederholen                              'Naechsten Ton ausgeben.
Else
  GoTo Fertig                                   'Funktion beenden lassen.
End If
Fertig:

Call PlayWaveSound(0, SND_PURGE)                'Hintergrundmusik sofort beenden.

PlayMP3Sound "stop " & stAlias2, 0, 0, 0        'Wiedergabe stoppen und MCI
PlayMP3Sound "close " & stAlias2, 0, 0, 0       'schliessen.
PlayMP3Sound "stop " & stAlias3, 0, 0, 0        'Wiedergabe stoppen und MCI
PlayMP3Sound "close " & stAlias3, 0, 0, 0       'schliessen.

End Function                                    'Funktionsende.
```

Als Ergänzung noch ein Beispiel Multimedia-Dateien über die **Shell-Execute** Funktion zu starten:

```
'Zum Abspielen von Multimedia-Dateien ueber bestimmten Player:
Public Declare Function ShellExecute Lib "shell32.dll" _
  Alias "ShellExecuteA" (ByVal hwnd As Long, _
  ByVal lpOperation As String, _
  ByVal lpFile As String, _
  ByVal lpParameters As String, _
  ByVal lpDirectory As String, _
  ByVal lpnShowCmd As Long) As Long


Public Function DemoShellExecute()
Dim stDateiname        As String                'Sound-Datei.

stDateiname = ActiveWorkbook.Path & "\Musik.mp3"    'Dateinamen für Musik festlegen.

ShellExecute 0, "open", stDateiname, "", _
             "C:\Programme\Windows Media Player\wmplayer.exe", 1
End Function                                    'Funktionsende.
```

Obwohl hier der Windows Media Player aufgerufen wird, startet bei mir jene Anwendung, welche ich als Standardprogramm zum Öffnen von mp3 (Winamp) oder mpg Dateien (VLC-Player) festgelegt habe. Der Parameter 1 am Ende sollte den zu öffnenden Fenstertyp festlegen. Bleibt bei mir wirkungslos. Ideal wäre der Parameter: vbMinimizedNoFocus (als Symbol in der Taskleiste ohne zu fokusieren).

(Siehe auch Datei: Mustervorlage_06_Sounds.xlsm bzw. hier im Anhang 4)

Anhang 1:

```
'*********************************************************************************/
'*                                                                               */
'*                                \\\\|////                                      */
'*                                \\ _ _ //                                      */
'*                                 ( o o )                                       */
'*=============================oOOo-(_)-oOOo=================================*/
'* WICHTIG: Dieser Code benoetigt zusaetzliche Verweise auf die ActiveX-dll      */
'* Klick: Extras > Verweise > Visual Basic for Applications                      */
'* Klick: Extras > Verweise > Microsoft Excel 12.0 Objects Library               */
'* Klick: Extras > Verweise > Microsoft Office 12.0 Objects Library              */
'* Klick: Extras > Verweise > OLE Automation                                     */
'*=======================================Oooo====================================*/
'*                                oooO  (  )                                     */
'*                                (  )  ) /                                      */
'*                                 \ ( (_/                                       */
'*                                  \_)                                          */
'*                                                                               */
'*********************************************************************************/
'*********************************************************************************/
'*                                                                               */
'*      PROGRAMM:       Mustervorlage fuer Spiele                                */
'*      PROJEKT:        Spiele programmieren in Excel                            */
'*                                                                               */
'*      MODULNAME:      DieseArbeitsmappe                                        */
'*      MODULNAME:      Funktionen, Globales, Makros, Standard, ...              */
'*      ARCHIVIERT:     -                                                        */
'*                                                                               */
'*      VERSION:        V  01.00                                                 */
'*      VERSIONSDATUM:  11 Jan 2019 12:00:00                                     */
'*                                                                               */
'*      BEARBEITER:     Ing. Harald Mitsch, TBM                                  */
'*                                                                               */
'*===============================================================================*/
'*                                                                               */
'*      (C) 2019 by TBM-Technisches Buero Mitsch   Elektrotechn. Planungen       */
'*      Lizenz: FREEWARE                           und ET - Installationen       */
'*      Josefgasse 6, 3380 Poechlarn               EDV - Selfmade Software       */
'*      Tel.:02757 / 46 56 bzw. 0676 /588 09 16    allgemeines Zeichenbüro       */
'*                                                                               */
'*===============================================================================*/
'*                                                                               */
'*      BESCHREIBUNG:   Behandelt Funktionen zu ...                              */
'*                                                                               */
'*      NEBENEFFEKTE:   -                                                        */
'*                                                                               */
'*      GRENZEN:        -                                                        */
'*                                                                               */
'*===============================================================================*/
'*                                                                               */
'*      DEKLARIERTE FUNKTIONEN:   keine                                          */
'*                                                                               */
'*      EXPORTIERTE FUNKTIONEN:   keine                                          */
'*                                                                               */
'*      IMPORTIERTE FUNKTIONEN:   keine                                          */
'*                                                                               */
'*===============================================================================*/
'*                                                                               */
'*      EXPORTIERTE VARIABLEN:    keine                                          */
'*                                                                               */
'*      IMPORTIERTE VARIABLEN:    alles aus Modul Globales                       */
'*                                                                               */
'*===============================================================================*/
'*                                                                               */
'*      INTERNE FUNKTIONEN:       Starten                                        */
'*                                SpielAnsichtAktualisieren                      */
'*                                                                               */
'*===============================================================================*/
'*                                                                               */
'*      AENDERUNGEN: - Modification History:                                     */
'*                                                                               */
'*      V 01.00     11.01.2019    Mitsch Harald, TBM, Initial Version            */
'*                                                                               */
'*********************************************************************************/
```

Anhang 2:

```
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       Funktionsname                                    */
'*                                                                       */
'*      BESCHREIBUNG:   Beschreibung was Funktion macht.                 */
'*                                                                       */
'*      RETURN          nichts                                           */
'*                                                                       */
'*      PARAMETER:      keine                                            */
'*                                                                       */
'*-----------------------------------------------------------------------*/


'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       Musterfunktion                                   */
'*                                                                       */
'*      BESCHREIBUNG:   Dient als Kopiervorlage fuer weitere Funktionen. */
'*                                                                       */
'*      RETURN          Boolean ..... True = Funktion in Ordnung         */
'*                                    False = Fehler in Funktion         */
'*      PARAMETER:      keine                                            */
'*                                                                       */
'*-----------------------------------------------------------------------*/


'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       Spaltennummer_aus_Buchstabe                      */
'*                                                                       */
'*      BESCHREIBUNG:   Liefert Spaltennummer aus Spaltenbuchstaben      */
'*                                                                       */
'*      RETURN          Integer ..... Spaltennummer                      */
'*                                    0 bei Fehler                       */
'*                                                                       */
'*      PARAMETER:      String ..... Spaltenbuchstabe(n)                 */
'*                                                                       */
'*-----------------------------------------------------------------------*/


'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       Buchstabe_aus_Spaltennummer                      */
'*                                                                       */
'*      BESCHREIBUNG:   Liefert Spaltenbuchstabe aus Spaltennummer       */
'*                                                                       */
'*      RETURN          String ..... Buchstabe der Spalte                */
'*                                   '' Leerstring bei Fehler            */
'*                                                                       */
'*      PARAMETER:      Integer ..... Spaltennummer                      */
'*                                                                       */
'*-----------------------------------------------------------------------*/
```

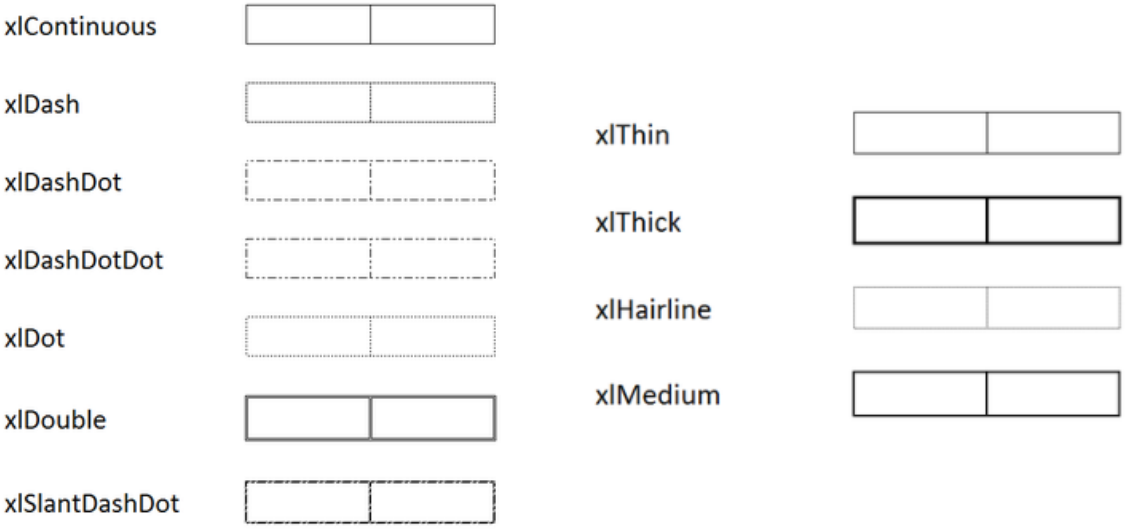
Anhang 3:

Alle Farbwerte für den Befehl: ColorIndex

1		29	
2		30	
3		31	
4		32	
5		33	
6		34	
7		35	
8		36	
9		37	
10		38	
11		39	
12		40	
13		41	
14		42	
15		43	
16		44	
17		45	
18		46	
19		47	
20		48	
21		49	
22		50	
23		51	
24		52	
25		53	
26		54	
27		55	
28		56	

Alle Farbkonstanten für den Befehl: Color

Konstanten für Rahmenlinien (Typ) und Linienstärke



Anhang 4: - Vollständiger Programmcode aus Modul: Sounds

```
'*************************************************************************/
'*                                                                       */
'*      PROGRAMM:       Mustervorlage fuer Spiele                        */
'*      PROJEKT:        Spiele programmieren in Excel                    */
'*                                                                       */
'*      MODULNAME:      Sounds                                           */
'*      ARCHIVIERT:     -                                                */
'*                                                                       */
'*      VERSION:        V  01.00                                         */
'*      VERSIONSDATUM:  11 Jan 2019 12:00:00                             */
'*                                                                       */
'*      BEARBEITER:     Ing. Harald Mitsch, TBM                          */
'*                                                                       */
'*=======================================================================*/
'*                                                                       */
'*      (C) 2019 by TBM-Technisches Buero Mitsch   Elektrotechn. Planungen  */
'*      Lizenz: FREEWARE                           und ET - Installationen  */
'*      Josefgasse 6, 3380 Poechlarn               EDV - Selfmade Software  */
'*      Tel.:02757 / 46 56 bzw. 0676 /588 09 16    allgemeines Zeichenbüro  */
'*                                                                       */
'*=======================================================================*/
'*                                                                       */
'*      BESCHREIBUNG:   Behandelt alles was mit Sounds zu tun hat.       */
'*                                                                       */
'*      NEBENEFFEKTE:   -                                                */
'*                                                                       */
'*      GRENZEN:        -                                                */
'*                                                                       */
'*=======================================================================*/
'*                                                                       */
'*      DEKLARIERTE FUNKTIONEN:   Beep                                   */
'*                                MessageBeep                            */
'*                                GetShortPathName                       */
'*                                PlayWaveSound                          */
'*                                PlayMP3Sound                           */
'*                                ShellExecute                           */
'*                                                                       */
'*      EXPORTIERTE FUNKTIONEN:   keine                                  */
'*                                                                       */
'*      IMPORTIERTE FUNKTIONEN:   keine                                  */
'*                                                                       */
'*=======================================================================*/
'*                                                                       */
'*      EXPORTIERTE VARIABLEN:    alles as Deklarationsbereich           */
'*                                                                       */
'*      IMPORTIERTE VARIABLEN:    keine                                  */
'*                                                                       */
'*=======================================================================*/
'*                                                                       */
'*      INTERNE FUNKTIONEN:       FrequenzBerechnen                      */
'*                                DemoKammerton_A                        */
'*                                DemoGoetterfunken                      */
'*                                DemoSirene                             */
'*                                DemoMessageBeep                        */
'*                                DemoWaveAbspielen                      */
'*                                DemoWaveStoppen                        */
'*                                DemoMP3Abspielen                       */
'*                                DemoSounds                             */
'*                                DemoShellExecute                       */
'*                                                                       */
'*                                EndloseMusic                           */
'*                                PlayMusic                              */
'*                                PauseMusic                             */
'*                                StopMusic                              */
'*                                                                       */
'*=======================================================================*/
'*                                                                       */
'*      AENDERUNGEN: - Modification History:                             */
'*                                                                       */
'*      V 01.00     11.01.2019    Mitsch Harald, TBM, Initial Version    */
'*                                                                       */
'*************************************************************************/
```

```
Option Explicit                                         'Alle Variablen muessen
                                                        'explizit angegeben werden.
'-------------------------------------------------------------------------------/
'-  Deklarierte API-Funktionen fuer 64 Bit Office Versionen                     -/
'-------------------------------------------------------------------------------/
#If VBA7 Then                                           'Fuer 64 Bit Excel
  'Hier eventuell Parameter vom Typ Long noch aendern auf LongPtr. (Noch nicht getestet!)

  'Gibt einen Ton in bestimmter Laenge aus. Der Frequentbereich dwFreq zwischen 37 und 32767
  Public Declare PtrSafe Function Beep  Lib "kernel32" (ByVal dwFreq As Long, ByVal dwDuration As Long) As
Boolean

  'Spielt einen Ton ab, Abhängigkeit vom Soundschema in den Systemeinstellungen:
  Public Declare PtrSafe Function MessageBeep Lib "user32" (ByVal uType As eType) As Boolean

  'Ermittelt den DOS 8.3 Dateinamen der abzuspielenden Datei. Wird benoetigt, da
  'Dateinamen mit Leerzeichen bei den Abspielfunktionen Probleme bereiten koennen.
  Private Declare PtrSafe Function GetShortPathName Lib "kernel32.dll" _
    Alias "GetShortPathNameA" (ByVal lpszLongPath As String, _
    ByVal lpszShortPath As String, _
    ByVal cchBuffer As Long) As Long

  'Zum Abspielen einer WAV Datei:
  Public Declare PtrSafe Function PlayWaveSound Lib "winmm.dll" _
    Alias "sndPlaySoundA" (ByVal lpszName As String, _
    ByVal dwFlags As Long) As Long

  'Zum Abspielen einer MP3 Datei:
  Private Declare PtrSafe Function PlayMP3Sound Lib "winmm.dll" _
    Alias "mciSendStringA" (ByVal lpszCommand As String, _
    ByVal lpszReturnString As String, _
    ByVal cchReturnLength As Long, _
    ByVal hwndCallback As Long) As Long

  'Zum Abspielen von Multimedia-Dateien ueber bestimmten Player:
  Public Declare PtrSafe Function ShellExecute Lib "shell32.dll" _
    Alias "ShellExecuteA" (ByVal hwnd As Long, _
    ByVal lpOperation As String, _
    ByVal lpFile As String, _
    ByVal lpParameters As String, _
    ByVal lpDirectory As String, _
    ByVal lpnShowCmd As Long) As Long

'-------------------------------------------------------------------------------/
'-  Deklarierte API-Funktionen fuer 32 Bit Office Versionen                     -/
'-------------------------------------------------------------------------------/
#Else                                                   'Fuer 32 Bit Excel
  'Gibt einen Ton in bestimmter Laenge aus. Der Frequentbereich dwFreq zwischen 37 und 32767
  Public Declare Function Beep Lib "kernel32.dll" _
    (ByVal dwFreq As Long, ByVal dwDuration As Long) As Boolean

  'Spielt einen Ton ab, Abhängigkeit vom Soundschema in den Systemeinstellungen:
  Public Declare Function MessageBeep Lib "user32.dll" _
    (ByVal uType As eType) As Boolean

  'Ermittelt den DOS 8.3 Dateinamen der abzuspielenden Datei. Wird benoetigt, da
  'Dateinamen mit Leerzeichen bei den Abspielfunktionen Probleme bereiten koennen.
  Private Declare Function GetShortPathName Lib "kernel32.dll" _
    Alias "GetShortPathNameA" (ByVal lpszLongPath As String, _
    ByVal lpszShortPath As String, _
    ByVal cchBuffer As Long) As Long

  'Zum Abspielen einer WAV Datei:
  Public Declare Function PlayWaveSound Lib "winmm.dll" _
    Alias "sndPlaySoundA" (ByVal lpszName As String, _
    ByVal dwFlags As Long) As Long

  'Zum Abspielen einer MP3 Datei:
  Private Declare Function PlayMP3Sound Lib "winmm.dll" _
    Alias "mciSendStringA" (ByVal lpszCommand As String, _
    ByVal lpszReturnString As String, _
    ByVal cchReturnLength As Long, _
    ByVal hwndCallback As Long) As Long
```

```
  'Zum Abspielen von Multimedia-Dateien ueber bestimmten Player:
  Public Declare Function ShellExecute Lib "shell32.dll" _
    Alias "ShellExecuteA" (ByVal hwnd As Long, _
    ByVal lpOperation As String, _
    ByVal lpFile As String, _
    ByVal lpParameters As String, _
    ByVal lpDirectory As String, _
    ByVal lpnShowCmd As Long) As Long
#End If


'-------------------------------------------------------------------------/
'-  Globale Konstanten - Moegliche Parameter fuer: Beep (Notenlaenge)      -/
'-------------------------------------------------------------------------/
Global Const TONLAENGE = 500                'Dauer fuer viertel-Note: (Basiswert)
'Global Const DAUER_GANZ = 2000             'Dauer fuer ganze Note: 2 Sekunden
'Global Const DAUER_HALB = 1000             'Dauer fuer halbe Note: 1000 Millisek.
'Global Const DAUER_VIERTEL = 500           'Dauer fuer viertel-Note: (Basiswert)
'Global Const DAUER_ACHTEL = 250            'Dauer fuer achtel-Note:


'-------------------------------------------------------------------------/
'-  Globale Konstanten - Moegliche Parameter fuer: Beep (Tonhoehe)         -/
'-------------------------------------------------------------------------/
'Global Const TON_A4 = 440                  'Definition einzelner Noten:
'Um sich die Definition aller 120 Noten zu ersparen wird die Funktion: BerechneFrequenz
'verwendet. Diese berechnet sie anhand der GRUNDFREQUENZ, der Oktave und Ton-Nummer:
Global Const GRUNDFREQUENZ = 30.7           'Bei Kammerton A = 440 Hz.

Global Const C = 1                          'Definitionen der Ton-Nummer:
Global Const Cis = 2
Global Const D = 3
Global Const Dis = 4
Global Const E = 5
Global Const F = 6
Global Const Fis = 7
Global Const G = 8
Global Const Gis = 9
Global Const A = 10
Global Const B = 11
Global Const H = 12

'-------------------------------------------------------------------------/
'-  Globale Konstanten - Moegliche Parameter fuer: MessageBeep            -/
'-------------------------------------------------------------------------/
Public Enum eType
  MB_OK = 0                     'Standard Sound
  MB_ICONERROR = &H10           'Kritischer Stop
  MB_ICONQUESTION = &H20        'Frage
  MB_ICONWARNING = &H30         'Warnung
  MB_ICONINFORMATION = &H40     'Information
End Enum

'-------------------------------------------------------------------------/
'-  Globale Konstanten - Moegliche Parameter fuer: PlayWaveSound (Wave)   -/
'-------------------------------------------------------------------------/
Global Const SND_SYNC = &H0         '(Standard) Spielt gesamtes Lied synchron ab. Code wird bis Liedende
angehalten.
Global Const SND_ASYNC = &H1        'Spielt gesamtes Lied asynchron ab. Code wird beim Starten NICHT
angehalten.
Global Const SND_LOOP = &H8         'Lied in einer Endlos-Schleife wiedergeben bis nächstes Lied
aufgerufen wird.
Global Const SND_PURGE = &H40       'Lied in aktueller Endlos-Schleife stoppen.

                                     'Zusaetzliche Konstanten fuer alte Version: PlaySoundA
'Global Const SND_NODEFAULT = &H2    'Falls Datei nicht gefunden wird, keinen Standardsound abspielen
sondern still bleiben.
'Global Const SND_MEMORY = &H4       'lpszSoundName is a memory file of the sound. Not used in VBA/VB6.
'Global Const SND_NOSTOP = &H10      'Aktuelles Lied nicht anhalten bevor das nächste Lied gestartet wird.
'Global Const SND_FILENAME = &H20000 'Immer anzugeben wenn es sich um eine Wave-Datei handelt?
```

```
'-------------------------------------------------------------------------/
'-  Globale Konstanten fuer abzuspielende Dateien                        -/
'-------------------------------------------------------------------------/
Global Const HINTERGRUNDMUSIK_DEMO_WAV = "\Musik.wav"
Global Const HINTERGRUNDMUSIK_DEMO_MP3 = "\Musik.mp3"
Global Const MUSIK_DEMO_WAV = "\Mist1.wav"
Global Const MUSIK_DEMO_MP3 = "\Mist3.mp3"


Public Function FrequenzBerechnen(iOktave As Integer, iTon As Integer) As Long
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       FrequenzBerechnen                                */
'*                                                                       */
'*      BESCHREIBUNG:   Berechnet die Frequenz als Ganzzahl fuer einen Ton  */
'*                      anhand der Grundfrequenz (30.7 Hz), der Oktave (1-8)*/
'*                      und der Tonnummer (1-8) innerhalb der Oktave.    */
'*                                                                       */
'*      RETURN          Long ............. Frequenz des Tons           */
'*                                                                       */
'*      PARAMETER:      Integer .......... Oktaven-Nummer                */
'*                      Integer .......... Ton-Nummer                    */
'*                                                                       */
'*-----------------------------------------------------------------------*/
On Error GoTo Fehler                            'Bei Fehler zur Fehlerbehandlung.
FrequenzBerechnen = True                        'Returnwert = alles OK setzen.

FrequenzBerechnen = Round(GRUNDFREQUENZ * (2 ^ (iOktave - 1)) * (2 ^ (1 / 12)) ^ (iTon - 1), 0)

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

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


Public Function DemoKammerton_A() As Boolean
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       DemoKammerton_A                                  */
'*                                                                       */
'*      BESCHREIBUNG:   Gibt den Kammerton mit 440 Hz 5 Sekunden lang aus.  */
'*                                                                       */
'*      RETURN          Boolean .......... True = Funktion in Ordnung    */
'*                                         False = Fehler in Funktion    */
'*      PARAMETER:      keine                                            */
'*                                                                       */
'*-----------------------------------------------------------------------*/
Dim iOktave             As Integer              'Aktuelle Oktave.

On Error GoTo Fehler                            'Bei Fehler zur Fehlerbehandlung.
DemoKammerton_A = True                          'Returnwert = alles OK setzen.

iOktave = 4                                     '1. Standard-Oktave = 4.
                                                'Kammerton 5 Sek. lang ausgeben.
Beep FrequenzBerechnen(iOktave, A), TONLAENGE * 10  'Tonlaenge 1/4 Note = 500 ms,
                                                '500 ms * 10 = 5 Sekunden.
MsgBox "Fertig"                                 'Meldung ausgeben.


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

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

Public Function DemoGoetterfunken() As Boolean
'************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                       */
'************************************************************************/
'*----------------------------------------------------------------------*/
'*                                                                      */
'*      FUNKTION:       DemoGoetterfunken                               */
'*                                                                      */
'*      BESCHREIBUNG:   Spielt den Anfang von Freude schoener Goetterfunken.*/
'*                                                                      */
'*      RETURN          Boolean ..... True = Funktion in Ordnung        */
'*                                    False = Fehler in Funktion        */
'*      PARAMETER:      keine                                           */
'*                                                                      */
'*----------------------------------------------------------------------*/
Dim iOktave             As Integer              'Aktuelle Oktave.

On Error GoTo Fehler                            'Bei Fehler zur Fehlerbehandlung.
DemoGoetterfunken = True                        'Returnwert = alles OK setzen.

iOktave = 4                                     '1. Standard-Oktave = 4.

Beep FrequenzBerechnen(iOktave, E), TONLAENGE       'Freu
Beep FrequenzBerechnen(iOktave, E), TONLAENGE       'de
Beep FrequenzBerechnen(iOktave, F), TONLAENGE       'schoe
Beep FrequenzBerechnen(iOktave, G), TONLAENGE       'ner

Beep FrequenzBerechnen(iOktave, G), TONLAENGE       'Goet
Beep FrequenzBerechnen(iOktave, F), TONLAENGE       'ter
Beep FrequenzBerechnen(iOktave, E), TONLAENGE       'fun
Beep FrequenzBerechnen(iOktave, D), TONLAENGE       'ken

Beep FrequenzBerechnen(iOktave, C), TONLAENGE       'Toch
Beep FrequenzBerechnen(iOktave, C), TONLAENGE       'ter
Beep FrequenzBerechnen(iOktave, D), TONLAENGE       'aus
Beep FrequenzBerechnen(iOktave, E), TONLAENGE       'E

Beep FrequenzBerechnen(iOktave, E), TONLAENGE * 1.5 'ly
Beep FrequenzBerechnen(iOktave, D), TONLAENGE / 2   'si
Beep FrequenzBerechnen(iOktave, D), TONLAENGE * 2   'um

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

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


Public Function DemoSirene() As Boolean
'************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                       */
'************************************************************************/
'*----------------------------------------------------------------------*/
'*                                                                      */
'*      FUNKTION:       DemoSirene                                      */
'*                                                                      */
'*      BESCHREIBUNG:   Versucht eine Sirene zu simulieren mit Beep.    */
'*                                                                      */
'*      RETURN          Boolean ..... True = Funktion in Ordnung        */
'*                                    False = Fehler in Funktion        */
'*      PARAMETER:      keine                                           */
'*                                                                      */
'*----------------------------------------------------------------------*/
```

```
Dim i                   As Integer              'Allgemeine Zaehlvariable.
Dim j                   As Integer              'Allgemeine Zaehlvariable.

On Error GoTo Fehler                            'Bei Fehler zur Fehlerbehandlung.
DemoSirene = True                               'Returnwert = alles OK setzen.

For i = 1 To 3                                  'Sirene 3 mal abspielen:
  For j = 1000 To 4000 Step 200                 'Tonhoehe erhoehen und
    Beep j, 100                                 'abspielen.
  Next j
  For j = 4000 To 1000 Step -200                'Tonhoehe verringern un
    Beep j, 100                                 'abspielen
  Next j
Next i                                          'Naechster Durchlauf

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

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


Public Function DemoMessageBeep() As Boolean
'************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                       */
'************************************************************************/
'*----------------------------------------------------------------------*/
'*                                                                      */
'*      FUNKTION:       DemoMessageBeep                                 */
'*                                                                      */
'*      BESCHREIBUNG:   Gibt die aktuellen Systemklaenge wieder je nachdem  */
'*                      welches Soundschema in der Windows Systemsteuerung  */
'*                      ausgewaehlt ist.                                */
'*                                                                      */
'*      RETURN          Boolean ..... True = Funktion in Ordnung        */
'*                                    False = Fehler in Funktion        */
'*      PARAMETER:      keine                                           */
'*                                                                      */
'*----------------------------------------------------------------------*/
On Error GoTo Fehler                            'Bei Fehler zur Fehlerbehandlung.
DemoMessageBeep = True                          'Returnwert = alles OK setzen.

MessageBeep (MB_OK)                             'Systemsound fuer OK abspielen
MsgBox "OK", vbOKOnly                           'und Meldung dazu anzeigen.

MessageBeep (MB_ICONERROR)                      'Systemsound fuer OK abspielen
MsgBox "Fehler", vbOKOnly + vbCritical          'und Meldung dazu anzeigen.

MessageBeep (MB_ICONQUESTION)                   'Systemsound fuer OK abspielen
MsgBox "Frage", vbYesNo + vbQuestion            'und Meldung dazu anzeigen.

MessageBeep (MB_ICONWARNING)                    'Systemsound fuer OK abspielen
MsgBox "Warnung", vbOKOnly + vbExclamation      'und Meldung dazu anzeigen.

MessageBeep (MB_ICONINFORMATION)                'Systemsound fuer OK abspielen
MsgBox "Information", vbOKOnly + vbInformation  'und Meldung dazu anzeigen.


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

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

```
Public Function DemoWaveAbspielen() As Boolean
'************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                       */
'************************************************************************/
'*----------------------------------------------------------------------*/
'*                                                                      */
'*      FUNKTION:       DemoWaveAbspielen                               */
'*                                                                      */
'*      BESCHREIBUNG:   Musteraufrufe zum Abspielen von Wave-Dateien.   */
'*                                                                      */
'*      RETURN          Boolean ..... True = Funktion in Ordnung        */
'*                                    False = Fehler in Funktion        */
'*      PARAMETER:      keine                                           */
'*                                                                      */
'*----------------------------------------------------------------------*/
Dim stDateiname1        As String                   'Sound-Datei 1.
Dim stDateiname2        As String                   'Sound-Datei 2.
Dim stBuffer            As String                   'Buffer fuer DOS 8.3 Dateinamen.

On Error GoTo Fehler                                'Bei Fehler zur Fehlerbehandlung.
DemoWaveAbspielen = True                            'Returnwert = alles OK setzen.

stDateiname1 = ActiveWorkbook.Path & MUSIK_DEMO_WAV 'Dateinamen für Musik 1 festlegen.
stDateiname2 = ActiveWorkbook.Path & HINTERGRUNDMUSIK_DEMO_WAV 'Dateinamen für Musik 2 festlegen.

stBuffer = Space$(255)                              'DOS 8.3 Dateinamen ermitteln:
If GetShortPathName(stDateiname1, stBuffer, Len(stBuffer)) <> 0 Then    'Wenn OK,
  stDateiname1 = Left$(stBuffer, InStr(stBuffer, vbNullChar) - 1)   'aus Buffer die
End If                                              'Leerzeichen am Ende entfernen.

stBuffer = Space$(255)                              'DOS 8.3 Dateinamen ermitteln:
If GetShortPathName(stDateiname2, stBuffer, Len(stBuffer)) <> 0 Then    'Wenn OK,
  stDateiname2 = Left$(stBuffer, InStr(stBuffer, vbNullChar) - 1)   'aus Buffer die
End If                                              'Leerzeichen am Ende entfernen.

Call PlayWaveSound(stDateiname1, SND_SYNC)          'Spielt Musik bis zum Ende, Programmcode wird
angehalten.
Call PlayWaveSound(stDateiname2, SND_ASYNC)         'Spielt Musik bis zum Ende, Programmcode laeuft
weiter.
Call PlayWaveSound(stDateiname2, SND_ASYNC Or SND_LOOP) 'Musik in Endlos-Schleife, Programmcode laeuft
weiter.
Sleep 10000                                         '10 Sekunden spielen lassen, dann:
Call PlayWaveSound(0, SND_PURGE)                    'Aktuelle Wiedergabe sofort beenden.

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

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


Public Function DemoWaveStoppen() As Boolean
'************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                       */
'************************************************************************/
'*----------------------------------------------------------------------*/
'*                                                                      */
'*      FUNKTION:       DemoWaveStoppen                                 */
'*                                                                      */
'*      BESCHREIBUNG:   Beendet das Abspielen einer Wave-Datei sofort.  */
'*                                                                      */
'*      RETURN          Boolean ..... True = Funktion in Ordnung        */
'*                                    False = Fehler in Funktion        */
'*      PARAMETER:      keine                                           */
'*                                                                      */
'*----------------------------------------------------------------------*/
On Error GoTo Fehler                                'Bei Fehler zur Fehlerbehandlung.
DemoWaveStoppen = True                              'Returnwert = alles OK setzen.
```

```
Call PlayWaveSound(0, SND_PURGE)                    'Aktuelle Wiedergabe sofort beenden.

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

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


Public Function DemoMP3Abspielen() As Boolean
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       DemoMP3Abspielen                                 */
'*                                                                       */
'*      BESCHREIBUNG:   Musteraufrufe zum Abspielen von MP3-Dateien.     */
'*                      Auch fuer wav und andere Musikformate verwendbar.*/
'*                                                                       */
'*      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.
DemoMP3Abspielen = True                             'Returnwert = alles OK setzen.

stDateiname = ActiveWorkbook.Path & HINTERGRUNDMUSIK_DEMO_MP3 'Dateinamen für Musik 1 festlegen.
stAlias = "Musik"                                   '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
  Call PlayMP3Sound("play " & stAlias & " from 0", 0, 0, 0) 'Zuerst MCI oeffnen und
End If                                              'wenn OK, dann MP3 abspielen.

Sleep 10000                                         '10 Sekunden spielen lassen.

PlayMP3Sound "stop " & stAlias, 0, 0, 0             'Wiedergabe stoppen und
PlayMP3Sound "close " & stAlias, 0, 0, 0            'MCI wieder schliessen.


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

Fehler:                                             'Fehlermarke - Fehlerbehandlung:
  DemoMP3Abspielen = False                          'Returnwert = Fehler setzen.
  MsgBox Err.Description                            'Fehlermeldung ausgeben.
  Resume Beenden                                    'Bei Funktionsende weitermachen.
  Resume                                            'Nur fuer Testzwecke hinterlegt.
End Function                                        'Funktionsende.
```

```
Public Function DemoSounds() As Boolean
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*-----------------------------------------------------------------------*/
'*                                                                       */
'*      FUNKTION:       DemoSounds                                       */
'*                                                                       */
'*      BESCHREIBUNG:   Muster wie die Sounds in einem Programm gehandhabt  */
'*                      werden koennen.                                  */
'*                                                                       */
'*      RETURN          Boolean ..... True = Funktion in Ordnung         */
'*                                    False = Fehler in Funktion         */
'*                                                                       */
'*      PARAMETER:      keine                                            */
'*                                                                       */
'*-----------------------------------------------------------------------*/
Dim stDateiname1        As String                   'Sound-Datei 1.
Dim stDateiname2        As String                   'Sound-Datei 2.
Dim stDateiname3        As String                   'Sound-Datei 3.
Dim stAlias1            As String                   'Aliasname fuer Sound-Datei 1.
Dim stAlias2            As String                   'Aliasname fuer Sound-Datei 2.
Dim stAlias3            As String                   'Aliasname fuer Sound-Datei 3.
Dim stBuffer            As String                   'Buffer fuer DOS 8.3 Dateinamen.

On Error GoTo Fehler                                'Bei Fehler zur Fehlerbehandlung.
DemoSounds = True                                   'Returnwert = alles OK setzen.

stDateiname1 = ActiveWorkbook.Path & HINTERGRUNDMUSIK_DEMO_MP3   'Datei fuer Sound 1 festlegen.
stDateiname2 = ActiveWorkbook.Path & MUSIK_DEMO_MP3 'Dateinamen fuer Musik 2 festlegen.
stDateiname3 = ActiveWorkbook.Path & MUSIK_DEMO_WAV 'Dateinamen fuer Musik 3 festlegen.

stAlias1 = "HintergrundMusik"                       'Aliasnamen fuer Hintergrund.
stAlias2 = "DreiMalMist"                            'Aliasnamen fuer Sound 2.
stAlias3 = "EinmalMist"                             'Aliasnamen fuer Sound 3.

stBuffer = Space$(255)                              'DOS 8.3 Dateinamen ermitteln:
If GetShortPathName(stDateiname1, stBuffer, Len(stBuffer)) <> 0 Then    'Wenn OK,
  stDateiname1 = Left$(stBuffer, InStr(stBuffer, vbNullChar) - 1)   'aus Buffer die
End If                                              'Leerzeichen am Ende entfernen.

stBuffer = Space$(255)                              'DOS 8.3 Dateinamen ermitteln:
If GetShortPathName(stDateiname2, stBuffer, Len(stBuffer)) <> 0 Then    'Wenn OK,
  stDateiname2 = Left$(stBuffer, InStr(stBuffer, vbNullChar) - 1)   'aus Buffer die
End If                                              'Leerzeichen am Ende entfernen.

stBuffer = Space$(255)                              'DOS 8.3 Dateinamen ermitteln:
If GetShortPathName(stDateiname3, stBuffer, Len(stBuffer)) <> 0 Then    'Wenn OK,
  stDateiname3 = Left$(stBuffer, InStr(stBuffer, vbNullChar) - 1)   'aus Buffer die
End If                                              'Leerzeichen am Ende entfernen.

                                                    'Wave-Datei in Endlos-Schleife
Call PlayWaveSound(stDateiname1, SND_ASYNC Or SND_LOOP) 'als Hintergrundmusik starten:

Wiederholen:                                        'Zusaetzliche Toene ausgeben bis zum Beenden:
MessageBeep (MB_ICONINFORMATION)                    'Ein Piepsen ausgeben, dann Fragen:
If MsgBox("Mist ausgeben?", vbYesNo + vbQuestion + vbDefaultButton2, PROGRAMMNAME) = vbYes Then
  If Int((2 * Rnd) + 1) = 1 Then                    'Einmal Mist ausgeben.
    PlayMP3Sound "stop " & stAlias2, 0, 0, 0        'Wiedergabe stoppen und MCI
    PlayMP3Sound "close " & stAlias2, 0, 0, 0       'schliessen.
    If PlayMP3Sound("open " & stDateiname2 & " type MPEGVideo alias " & stAlias2, 0, 0, 0) = 0 Then
      Call PlayMP3Sound("play " & stAlias2 & " from 0", 0, 0, 0) 'Zuerst MCI oeffnen und
    End If                                          'wenn OK, dann MP3 abspielen.
  Else                                              '3 mal Mist ausgeben.
    PlayMP3Sound "stop " & stAlias3, 0, 0, 0        'Wiedergabe stoppen und MCI
    PlayMP3Sound "close " & stAlias3, 0, 0, 0       'schliessen.
    If PlayMP3Sound("open " & stDateiname3 & " type MPEGVideo alias " & stAlias3, 0, 0, 0) = 0 Then
      Call PlayMP3Sound("play " & stAlias3 & " from 0", 0, 0, 0) 'Zuerst MCI oeffnen und
    End If                                          'wenn OK, dann MP3 abspielen.
  End If
  GoTo Wiederholen                                  'Naechsten Ton ausgeben.
Else
  GoTo Fertig                                       'Funktion beenden lassen.
End If
Fertig:
```

```
Call PlayWaveSound(0, SND_PURGE)                    'Aktuelle Wiedergabe sofort beenden.
PlayMP3Sound "stop " & stAlias2, 0, 0, 0            'Wiedergabe stoppen und MCI
PlayMP3Sound "close " & stAlias2, 0, 0, 0           'schliessen.
PlayMP3Sound "stop " & stAlias3, 0, 0, 0            'Wiedergabe stoppen und MCI
PlayMP3Sound "close " & stAlias3, 0, 0, 0           'schliessen.

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

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


Public Function DemoShellExecute() As Boolean
'*************************************************************************/
'*      INTERNES UNTERPROGRAMM - INTERNE FUNKTION                        */
'*************************************************************************/
'*----------------------------------------------------------------------*/
'*                                                                      */
'*      FUNKTION:       DemoShellExecute                                */
'*                                                                      */
'*      BESCHREIBUNG:   Muster fuer Abspielen von Multimedia-Dateien ueber  */
'*                      speziellen Player (fuer Musik und Videos).      */
'*                                                                      */
'*                      Obwohl hier der Windows Media Player aufgerufen     */
'*                      wird, startet bei mir jene Anwendung, welche ich    */
'*                      als Standardprogramm zum Öffnen von mp3 (Winamp)    */
'*                      oder mpg Dateien (VLC-Player) festgelegt habe.      */
'*                                                                      */
'*                      Der Parameter 1 am Ende sollte den zu öffnenden     */
'*                      Fenstertyp festlegen. Bleibt bei mir wirkungslos.   */
'*                      Ideal wäre der Parameter: vbMinimizedNoFocus        */
'*                      (als Symbol in der Taskleiste ohne zu fokusieren).  */
'*                                                                      */
'*      RETURN          Boolean ..... True = Funktion in Ordnung        */
'*                                    False = Fehler in Funktion        */
'*                                                                      */
'*      PARAMETER:      keine                                           */
'*                                                                      */
'*----------------------------------------------------------------------*/
Dim stDateiname        As String                    'Sound-Datei.

On Error GoTo Fehler                                'Bei Fehler zur Fehlerbehandlung.
DemoShellExecute = True                             'Returnwert = alles OK setzen.

stDateiname = ActiveWorkbook.Path & HINTERGRUNDMUSIK_DEMO_MP3    'Dateinamen für Musik festlegen.
'stDateiname = "C:\Bibliotheken\Musik\Musik_Videos\Van_der_Graaf_Generator.mpeg"

ShellExecute 0, "open", stDateiname, "", "C:\Programme\Windows Media Player\wmplayer.exe", 1
'ShellExecute 0, "open", stDateiname, "", "C:\Programme\Windows Media Player\wmplayer.exe",
vbMinimizedNoFocus

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

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


'-----------------------------------------------------------------------/
'-  Abspielen von Multimedia-Dateien mit Hilfe von quartz.dll           -/
'-----------------------------------------------------------------------/
'* Klick: Extras > Verweise > ActiveMovie control type library          */

'Verweis auf: ActiveMovie control type library (c:\windows\system32\quartz.dll)
```

```
'Excel-VBA -> Extras -> Verweise behauptet die quartz.dll befindet sich in:
'C:\Users\Admin\Documents\quartz.dll - Was aber nicht stimmt, daher kommt beim
'Setzen des Verweises eine Fehlermeldung. Kurzerhand einfach die Datei aus dem
'System32 Verzeichnis in die Dokumente einkopieren - dann funktioniert alles.
'Keine Ahnung warum, diesbezuegliche Eintraege in der Registry haben alle System32!

'Normalerweise im Deklarationsbereich eines Modules zu hinterlegen:
'Public oFM As FilgraphManager                      'Objekt mit New anlegen.
'Public oMP As IMediaPosition                       'Objekt mit Set oMP = oFM anlegen.
'Public oME As IMediaEvent                          'Objekt mit Set oME = oFM anlegen.
'Global bPauseAktiv      As Boolean                 'Pause ist aktiviert oder nicht.

'Moegliche Aufrufe - wichtig
'    oFM.RenderFile ""
'    oFM.Run
'    oFM.Stop
'    oFM.Pause
'    oMP.CurrentPosition
'    oMP.Duration

'Weitere Moegliche Aufrufe - unwichtig
'    oFM.GetState
'    oFM.AddSourceFilter
'    oFM.StopWhenReady
'    oFM.FilterCollection
'    oFM.RegFilterCollection
'    oFM.StopWhenReady
'
'    oMP.PrerollTime
'    oMP.Rate
'    oMP.StopTime
'    oMP.CanSeekBackward
'    oMP.CanSeekForward
'
'    oME.CancelDefaultHandling
'    oME.FreeEventParams
'    oME.GetEvent
'    oME.GetEventHandle
'    oME.RestoreDefaultHandling
'    oME.WaitForCompletion

'Public Function EndloseMusic()
'Dim i                   As Integer                 'Allgemeine Zaehlvariable.
'For i = 1 To 3                                     'Musik dreimal abspielen.
'  Call PlayMusic                                   'Abspielen starten.
'  While oMP.CurrentPosition < oMP.Duration         'Warten bis Musik fertig
'  Wend                                             'abgespielt wurde.
'  Call StopMusic                                   'Abspielen beenden.
'Next i                                             'Danach Musik nochmals starten.
'End Function                                       'Funktionsende.

'Public Function PlayMusic()
'Set oFM = New FilgraphManager                      'Neues Objekt erstellen.
'Set oMP = oFM                                      'IMediaPosition definieren.
'Set oME = oFM                                      'IMediaEvent definieren.
'oFM.RenderFile ActiveWorkbook.Path & HINTERGRUNDMUSIK_DEMO_MP3 'Datei fuer Wiedergabe festlegen.
'oFM.Run                                            'Wiedergabe starten.
'bPauseAktiv = False                                'Pause ist nicht aktiv.
'End Function                                       'Funktionsende.

'Public Function PauseMusic()
'If bPauseAktiv Then                                'Wenn Pause aktiv:
'  oFM.Run                                          'Wiedergabe weiterlaufen lassen.
'Else                                               'Pause ist nicht aktiv:
'  oFM.Pause                                        'Wiedergabe anhalten.
'  bPauseAktiv = True                               'Merken das jetzt die Pause
'End If                                             'aktiv ist.
'End Function                                       'Funktionsende.

'Public Function StopMusic()
'oFM.Stop                                           'Wiedergabe beenden und
'Set oFM = Nothing                                  'Objektvariable ruecksetzen.
'End Function                                       'Funktionsende.
'-------------------------------------------------------------------------------/
```