Jawbreaker

Contributed by Enno Wulff
Tuesday, 05 June 2007

The idea is to accumulate points by clearing matching coloured squares from the playing area, with fresh squares being added to play when you clear the far left column. Addictive.....

```
*---------------------------------------------------------------------*
* Report  ZZ_JAWBREAKER                                               *
*---------------------------------------------------------------------*

************************************************************************
***                                                                  ***
***                  http://www.tricktresor.de                       ***
***                                                                  ***
************************************************************************
*** Autor:   Enno Wulff                                             ***
*** Datum:   Okt/2004                                               ***
************************************************************************

*---------------------------------------------------------------------*
*
" Ziel ---------------------------------------------------------------"
* möglichst viele Punkte bekommen...
*
" Spielprinzip -------------------------------------------------------"
* Das Spielfeld enthält Spielsteine unterschiedlicher  Farbe. Jeweils
* mindestens zwei zusammenhängende Steine können "weggeklickt" werden.
* Je mehr zusammenhängende Steine auf einmal eliminiert werden, desto
* höher die Punktzahl.
*
" Todo ---------------------------------------------------------------"
* - Verwenden der "richtigen" zählweise für die Punkte. Zurzeit
*   ergibt sich die Punktezahl aus der Anzahl der Steine zum Quadrat.
*   Im Original sind dies nicht so viele...
*
*---------------------------------------------------------------------*

REPORT  zz_jawbreaker NO STANDARD PAGE HEADING.

TYPE-POOLS: icon.

TYPES:
  ty_farbe(2) TYPE n.

DATA:
  g_max_x(3)      TYPE n,
  punkte(5)       TYPE n,
  punkte1         TYPE i,
  punkte_undo     TYPE i,
  punktec(5)      TYPE c,
  l_x(2)          TYPE n,
  l_y(2)          TYPE n,
  l_line          TYPE i,
  l_feld(20)      TYPE c,
  l_tabix         TYPE i,
  l_anzahl        TYPE i,
  l_farbe         TYPE ty_farbe,
  l_spalte(2)     TYPE n,
  t_spalte        TYPE STANDARD TABLE OF ty_farbe,

  BEGIN OF spielfeld OCCURS 0,
    x(2)          TYPE n,
    y(2)          TYPE n,
    farbe(2)      TYPE n,
```

```
    mark          TYPE c,
  END OF spielfeld,
  spielfeld2      LIKE spielfeld OCCURS 0 WITH HEADER LINE,
  spielfeld_undo  LIKE spielfeld OCCURS 0 WITH HEADER LINE,
  l_spiel         LIKE spielfeld.

*** Spielfeld ***
SELECTION-SCREEN BEGIN OF BLOCK sfeld WITH FRAME TITLE text-fld.
PARAMETERS:
  p_x(2)          TYPE n DEFAULT 15,
  p_y(2)          TYPE n DEFAULT 15,
  p_farben(1)     TYPE c DEFAULT 5.
SELECTION-SCREEN END OF BLOCK sfeld.

*** Spielmodus ***
SELECTION-SCREEN BEGIN OF BLOCK modus WITH FRAME TITLE text-mod.
PARAMETERS:
  p_norm          RADIOBUTTON GROUP mod DEFAULT 'X',
  p_fort          RADIOBUTTON GROUP mod.
SELECTION-SCREEN END OF BLOCK modus.

AT SELECTION-SCREEN.
  IF p_farben > 7.
    p_farben = 7.
    MESSAGE w000(su) WITH 'Nur 7 Farben möglich!'.
  ENDIF.

START-OF-SELECTION.

  PERFORM init.

  PERFORM spielfeld_malen.

AT LINE-SELECTION.

  GET CURSOR FIELD l_feld LINE l_y.

  IF l_feld = 'ICON_SYSTEM_UNDO'.
*** Undo-Funktion
    spielfeld[] = spielfeld_undo[].
    SUBTRACT punkte_undo FROM punkte.
    CLEAR punkte_undo.
  ELSE.
    l_x = ( sy-cucol  - 1 ) / 2.


    READ TABLE spielfeld WITH KEY x = l_x
                                  y = l_y.
    IF sy-subrc = 0.
      l_tabix = sy-tabix.
      l_farbe = spielfeld-farbe.
      CASE spielfeld-mark.

        WHEN 1.
*** Erster Klick:
*-- Alte Auswahl löschen
          LOOP AT spielfeld INTO l_spiel WHERE mark = 0.
            l_spiel-mark = 1.
            MODIFY spielfeld FROM l_spiel.
          ENDLOOP.

*-- Undo
          spielfeld_undo[] = spielfeld[].

*-- Neue Auswahl
```

```
        CLEAR punkte1.
        spielfeld-mark = 0.
        MODIFY spielfeld INDEX l_tabix.
        PERFORM suchen USING l_x l_y l_farbe.
        l_anzahl = 0.
        LOOP AT spielfeld INTO l_spiel WHERE mark = 0.
          ADD 1 TO l_anzahl.
        ENDLOOP.
        IF l_anzahl < 2.
          READ TABLE spielfeld INDEX l_tabix.
          IF sy-subrc = 0.
            spielfeld-mark = 1.
            MODIFY spielfeld INDEX l_tabix.
          ENDIF.
        ELSE.
          punkte1 = l_anzahl * l_anzahl.
        ENDIF.

      WHEN 0.
*** Zweiter Klick: Löschen der markierten felder.
        LOOP AT spielfeld WHERE mark = 0.
          spielfeld-farbe = 0.
          spielfeld-mark  = 2.
          MODIFY spielfeld.
        ENDLOOP.
        ADD punkte1 TO punkte.
        punkte_undo = punkte1.
        CLEAR punkte1.
        PERFORM rutschen.
        PERFORM rechts_sammeln.
        PERFORM auffuellen.
        PERFORM ende_pruefen.
      WHEN OTHERS.
    ENDCASE.
  ENDIF.
ENDIF.

PERFORM spielfeld_malen.

*&---------------------------------------------------------------------*
*&      Form  suchen
*&---------------------------------------------------------------------*
*      -->F_X        text
*      -->F_Y        text
*      -->F_FARBE    text
*---------------------------------------------------------------------*
FORM suchen USING f_x f_y f_farbe.

  DATA:
    u_x(2) TYPE n,
    u_y(2) TYPE n.

  u_y = f_y.
  u_x = f_x - 1.
  PERFORM test USING u_x u_y f_farbe 'X'.

  u_x = f_x + 1.
  PERFORM test USING u_x u_y f_farbe 'X'.

  u_x = f_x.
  u_y = f_y - 1.
  PERFORM test USING u_x u_y f_farbe 'X'.

  u_y = f_y + 1.
  PERFORM test USING u_x u_y f_farbe 'X'.
```

```
ENDFORM.                                          "suchen

*&---------------------------------------------------------------------*
*&      Form  test
*&---------------------------------------------------------------------*
*      -->F_X        text
*      -->F_Y        text
*      -->F_FARBE    text
*----------------------------------------------------------------------*
FORM test USING value(f_x) value(f_y) f_farbe f_rekursiv.

  IF f_x = 0 OR f_x > p_x OR f_y = 0 OR f_y > p_y.
    EXIT.
  ELSE.
    READ TABLE spielfeld INTO l_spiel
        WITH KEY x = f_x  y = f_y.
    IF sy-subrc      = 0 AND
       l_spiel-farbe = f_farbe AND
       l_spiel-mark  = 1.
      l_spiel-mark = 0.
      MODIFY spielfeld FROM l_spiel INDEX sy-tabix.
      IF f_rekursiv = 'X'.
        PERFORM suchen USING f_x f_y f_farbe.
      ENDIF.
    ENDIF.
  ENDIF.

ENDFORM.                                          "test

*&---------------------------------------------------------------------*
*&      Form  init
*&---------------------------------------------------------------------*
FORM init.

  g_max_x = p_x * 3 + 1.

  DO p_x TIMES.
    l_x = sy-index.

    DO p_y TIMES.
      l_y = sy-index.

      spielfeld-x = l_x.
      spielfeld-y = l_y.
      PERFORM zufallsfarbe CHANGING spielfeld-farbe.
      spielfeld-mark = 1.
      APPEND spielfeld.

    ENDDO.

  ENDDO.

  spielfeld_undo[] = spielfeld[].

ENDFORM.                                          "init

*&---------------------------------------------------------------------*
*&      Form  spielfeld_malen
*&---------------------------------------------------------------------*
FORM spielfeld_malen.

  DATA l_pos TYPE i.
```

```
  SET BLANK LINES ON.
  sy-lsind = 0.
  SKIP TO LINE 1.

  IF punkte =  0.
    punktec = 'null'.
  ELSE.
    WRITE punkte TO punktec NO-ZERO LEFT-JUSTIFIED.
  ENDIF.
*** Titel: "Jawbreaker: & Punkte"
  SET TITLEBAR 'JAW' WITH punktec.

  DO p_y TIMES.
    l_y = sy-index.
    l_pos = 1.
    DO p_x TIMES.
      l_x = sy-index.
      READ TABLE spielfeld WITH KEY x = l_x
                                    y = l_y.
      IF sy-subrc = 0.
        IF spielfeld-farbe = 2.
          WRITE: '..' NO-GAP HOTSPOT OFF.
        ELSEIF spielfeld-mark = 1.
          WRITE: '  ' COLOR = spielfeld-farbe NO-GAP
                   HOTSPOT ON INTENSIFIED = spielfeld-mark.
        ELSE.
          WRITE: '**' COLOR = spielfeld-farbe NO-GAP
                   HOTSPOT ON INTENSIFIED ON .
        ENDIF.
      ENDIF.
      ADD 2 TO l_pos.
    ENDDO.
    NEW-LINE.
  ENDDO.

  SKIP 1.
  WRITE: / 'Punkte:', punkte1 NO-ZERO.
  WRITE AT 30 icon_system_undo AS ICON HOTSPOT ON.

ENDFORM.                    "spielfeld_aufbauen

*&---------------------------------------------------------------------*
*&      Form  zufallsfarbe
*&---------------------------------------------------------------------*
FORM zufallsfarbe CHANGING f_farbe.

  DATA:
    l_min TYPE i,
    l_max TYPE i,
    l_rnd TYPE i.

  l_min = 3.
  l_max = p_farben + 2.

  CALL FUNCTION 'QF05_RANDOM_INTEGER'
      EXPORTING
          ran_int_max   = l_max
          ran_int_min   = l_min
      IMPORTING
          ran_int       = l_rnd
      EXCEPTIONS
          invalid_input = 1
          OTHERS        = 2.

  f_farbe = l_rnd.
```

```
ENDFORM.                    "zufallsfarbe

*&---------------------------------------------------------------------*
*&      Form  rutschen
*&---------------------------------------------------------------------*
FORM rutschen.

  DATA:
    l_y2(2)  TYPE n.

  DO p_x TIMES.
    l_x = sy-index.
    l_y = p_y.
    CLEAR t_spalte.
    DO p_y TIMES.
      READ TABLE spielfeld WITH KEY x = l_x y = l_y.

      IF spielfeld-farbe <> 0.
        APPEND spielfeld-farbe TO t_spalte.
      ENDIF.
      SUBTRACT 1 FROM l_y.

    ENDDO.
    l_y2 = p_y.
    DO p_y TIMES.
      READ TABLE spielfeld WITH KEY x = l_x y = l_y2.
      l_tabix = sy-tabix.
      READ TABLE t_spalte INTO spielfeld-farbe INDEX sy-index.
      IF sy-subrc = 0.
        spielfeld-mark = 1.
      ELSE.
        spielfeld-farbe = 2.
        spielfeld-mark  = 1.
      ENDIF.
      MODIFY spielfeld INDEX l_tabix.
      SUBTRACT 1 FROM l_y2.
    ENDDO.

  ENDDO.

ENDFORM.                                    "rutschen

*&---------------------------------------------------------------------*
*&      Form  rechts_sammeln
*&---------------------------------------------------------------------*
FORM rechts_sammeln.

  DATA:
    l_y2(2)  TYPE n.

*** Rechts beginnen
  l_x = p_x.
  DO p_x TIMES.
    DO p_x TIMES.
*** Prüfen, ob die Spalte leer ist
      LOOP AT spielfeld WHERE x = l_x
                        AND farbe <> 2.
        EXIT.
      ENDLOOP.
      IF sy-subrc = 0.
*** spalte nicht leer; schleife verlassen
        EXIT.                               "from do
      ELSE.
** spalte ist leer
```

```
      l_spalte = l_x - 1.
      DO l_x TIMES.
        PERFORM verschiebe_spalte USING l_spalte.
        SUBTRACT 1 FROM l_spalte.
      ENDDO.
    ENDIF.
  ENDDO.
  SUBTRACT 1 FROM l_x.
ENDDO.

ENDFORM.                    "rechts_sammeln


*&---------------------------------------------------------------------*
*&      Form  verschiebe_spalte
*&---------------------------------------------------------------------*
FORM verschiebe_spalte USING f_spalte.

  DATA: l_spalte2(2) TYPE n.
  CLEAR t_spalte.

  LOOP AT spielfeld WHERE x = f_spalte.

    APPEND spielfeld-farbe TO t_spalte.
    spielfeld-farbe = 2.
    spielfeld-mark  = 1.
    MODIFY spielfeld.
  ENDLOOP.

  l_spalte2 = f_spalte + 1.
  l_tabix = 0.
  LOOP AT spielfeld WHERE x = l_spalte2.
    ADD 1 TO l_tabix.
    READ TABLE t_spalte INTO spielfeld-farbe INDEX l_tabix.
    MODIFY spielfeld.

  ENDLOOP.

ENDFORM.                    "verschiebe_spalte

*---------------------------------------------------------------------*
*       FORM ende_pruefen                                             *
*---------------------------------------------------------------------*
FORM ende_pruefen.

  DATA:
      u_x(2)  TYPE n,
      u_y(2)  TYPE n,

      v_x(2)  TYPE n,
      v_y(2)  TYPE n,

      v_farbe TYPE ty_farbe.


  spielfeld2[] = spielfeld[].


  DO p_x TIMES.
    v_x = sy-index.
    DO p_y TIMES.
      v_y = sy-index.
      READ TABLE spielfeld2 WITH KEY x = v_x y = v_y.
      CHECK spielfeld2-farbe > 2.
      v_farbe = spielfeld2-farbe.
```

```
      u_x = v_x - 1.
      u_y = v_y.
      PERFORM test USING u_x u_y v_farbe ' '.

      u_x = v_x + 1.
      PERFORM test USING u_x u_y v_farbe ' '.

      u_x = v_x.
      u_y = v_y - 1.
      PERFORM test USING u_x u_y v_farbe ' '.

      u_y = v_y + 1.
      PERFORM test USING u_x u_y v_farbe ' '.
    ENDDO.
  ENDDO.

  LOOP AT spielfeld WHERE mark = 0.

  ENDLOOP.
  IF sy-subrc > 0.
    SKIP p_y.
    SKIP 4.
    WRITE: / 'Spiel ist zuende! Gewinnpunkte:', punkte NO-ZERO.
  ELSE.
    spielfeld[] = spielfeld2[].
  ENDIF.

ENDFORM.

*---------------------------------------------------------------------*
*       FORM spalte_auffuellen                                        *
*---------------------------------------------------------------------*
FORM spalte_auffuellen USING value(fx).

  DATA:
    l_frb TYPE ty_farbe,
    l_anz TYPE i,
    l_min TYPE i,
    l_max TYPE i,
    l_rnd TYPE i,
    l_row TYPE i.

  l_max = p_y.
  l_min = 1.

  CALL FUNCTION 'QF05_RANDOM_INTEGER'
      EXPORTING
          ran_int_max   = l_max
          ran_int_min   = l_min
      IMPORTING
          ran_int       = l_rnd
      EXCEPTIONS
          invalid_input = 1
          OTHERS        = 2.

  l_row = p_y.
  DO l_rnd TIMES.
    READ TABLE spielfeld WITH KEY x = fx y = l_row.
    l_tabix = sy-tabix.
    PERFORM zufallsfarbe USING spielfeld-farbe.
    MODIFY spielfeld INDEX l_tabix.
    SUBTRACT 1 FROM l_row.
  ENDDO.

ENDFORM.
```

```
*---------------------------------------------------------------------*
*       FORM auffuellen                                               *
*---------------------------------------------------------------------*
FORM auffuellen.

  DATA l_spalte(2) TYPE n.

  IF p_fort = 'X'.

    l_spalte = p_x.
    DO p_x TIMES.
      LOOP AT spielfeld WHERE x = l_spalte AND farbe <> 2.
      ENDLOOP.
      IF sy-subrc > 0.
        PERFORM spalte_auffuellen USING l_spalte.
      ENDIF.
      SUBTRACT 1 FROM l_spalte.
    ENDDO.
  ENDIF.

ENDFORM.
```