Report

COAL

Contents

Introduction:

Candy Crush is a game of three match. When the three candies of same color are horizontally or vertically then the candies are popped. When popped the candies from top are fallen down and empty are filled with random candies. We have implemented three levels each level has a different board and different score to reach for finishing the game. Moves for each are 15. The project is compatible with MASAM 615.

Candies:

We had made 6 candies and 1 color bomb each having a unique id at the backend.

Candy Name	Backend code	Picture
Star	**0**	
Square	**1**	
Rectangle	**2**	
+ candy	**3**	
Triangle	**4**	
Rhombus	**5**	
Color bomb	**6**	

While 7 was considered to be the empty place and -1 for the blockages for higher level.

. Data block:

```
.DATA

    gameName DB "CANDY CRUSH!!! $"
    msg0     DB "Enter your name : $"
    playerN  DB 50 dup('$')
    msg1     DB "Welcome $"
    msg2     DB "RULES FOR THE GAME!!! $"
    decor    DB "**** $"
    rule1    DB "1) In this game rules will be limited      for each level. $"
    rule2    DB "2) You can combine 3,4 or 5 candies        together to pop them. $"
    rule3    DB "3) Combining 5 candies would make a        COLOUR BOMB. $"
    rule4    DB "4) Swapping the colour bomb with any       candy would pop all of that candy. $"
    rule5    DB "5) Levels passing criteria is :  $"
    append1  DB  "- Level 1 = 20  $"
    append2  DB  "- Level 2 = 30  $"
    append3  DB  "- Level 3 = 46 $"

    levelMsg DB  "Level : $"
    scoreMsg DB  "Score : $"
    moveMsg  DB  "Moves : $"

    menuMsge  DB  "Enter 1 to play Level 1 $"
    menuMsge0 DB  "Enter 2 to play Level 2 $"
    menuMsge1 DB  "Enter 3 to play Level 3 $"
    exitMsg   DB  "Enter 0 to exit $"
```

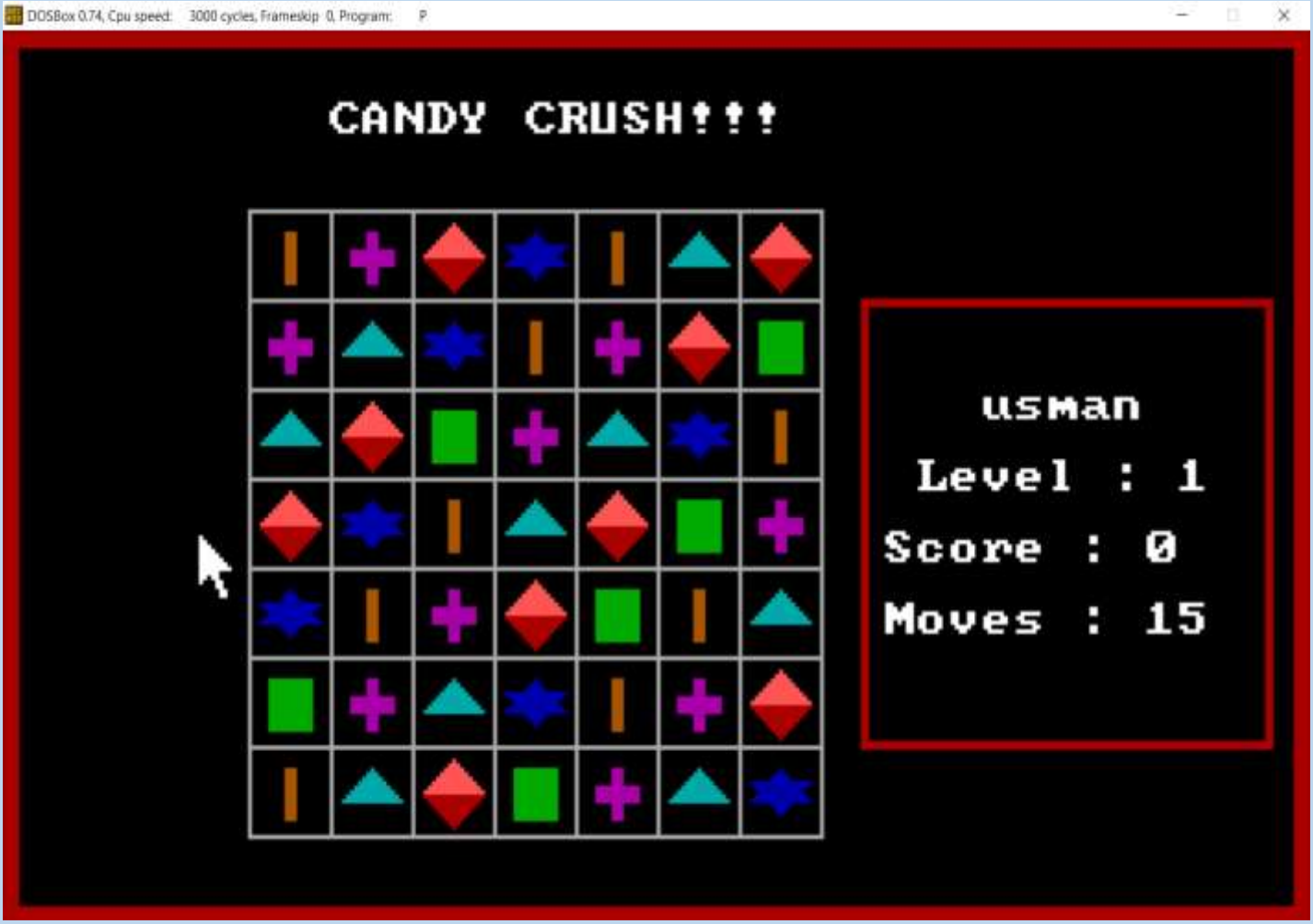
```
    victorMsg DB "         YOU WON!!!$"
    loosedMsg DB "         YOU LOST!!!$"


    ;for storing score
    score1     DW  0
    score2     DW  0
    score3     DW  0

    ;for moves
    moves     DW  15

    ;for level
    Level     DW  '1'

    ;for holding the co-ordinates
    var1      DW  0
    var2      DW  0
    var3      DW  0
    var4      DW  0
    temp      DW  0

    decision DW 0
```

```
;the board 7*7 -- the main
candyBoard DW 49 dup(0)

;the one clicked to swap
xCod    DW  0
yCod    DW  0

;new line in file
stringnewline db 13,10,'$'

;swapped with the one
xCodS   DW  0
yCodS   DW  0

;the mapped orignial co-ordinates
xStart  DW 0
xEnd    DW 0
yStart  DW 0
yEnd    DW 0


index1   DW -1 ;swap this
indeX2   DW -1 ;swap with
```

Levels:

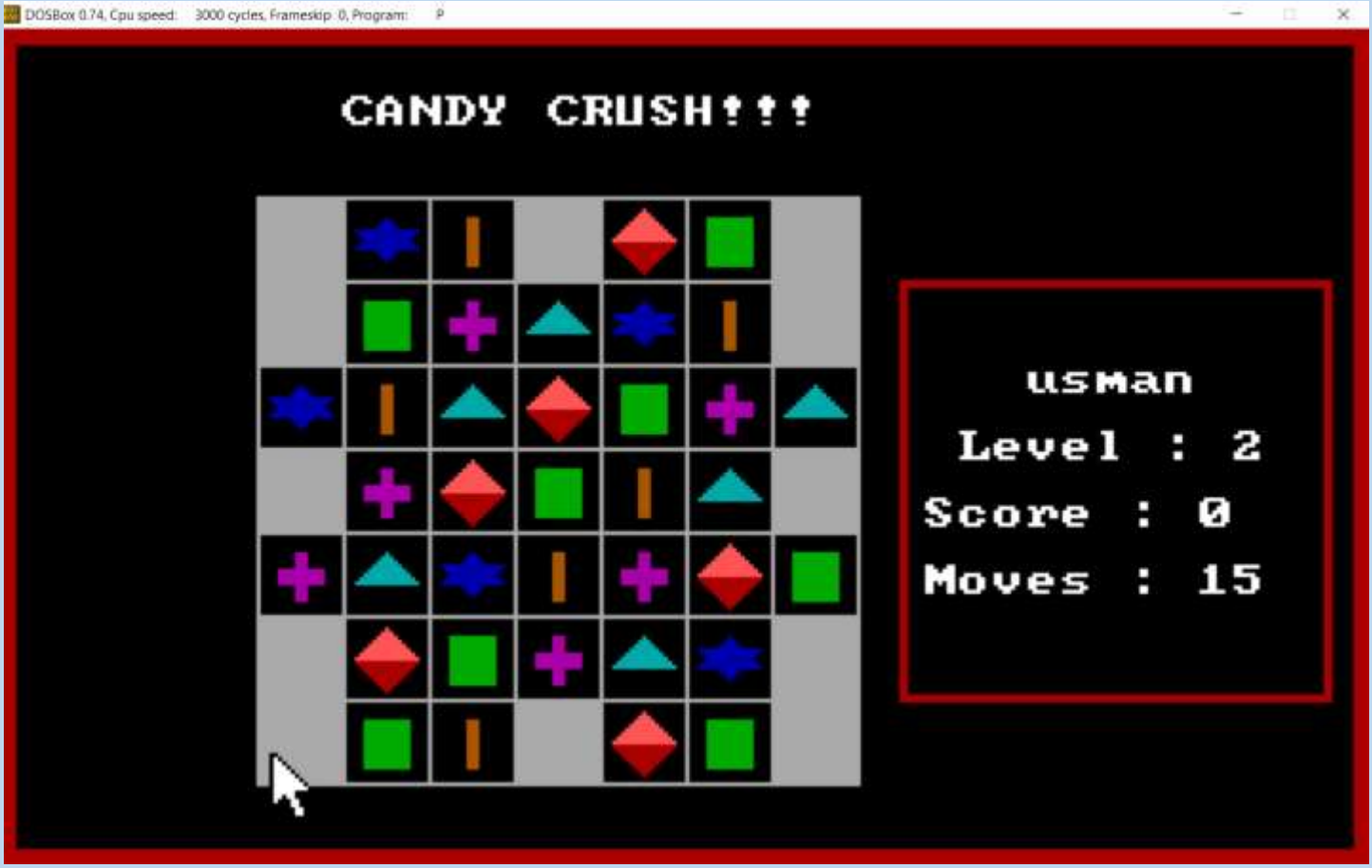
1. Level 1:

The candies were populated randomly and were displayed in the board. Also the name of the player, score, level number and moves are showing on the screen and changes accordingly.



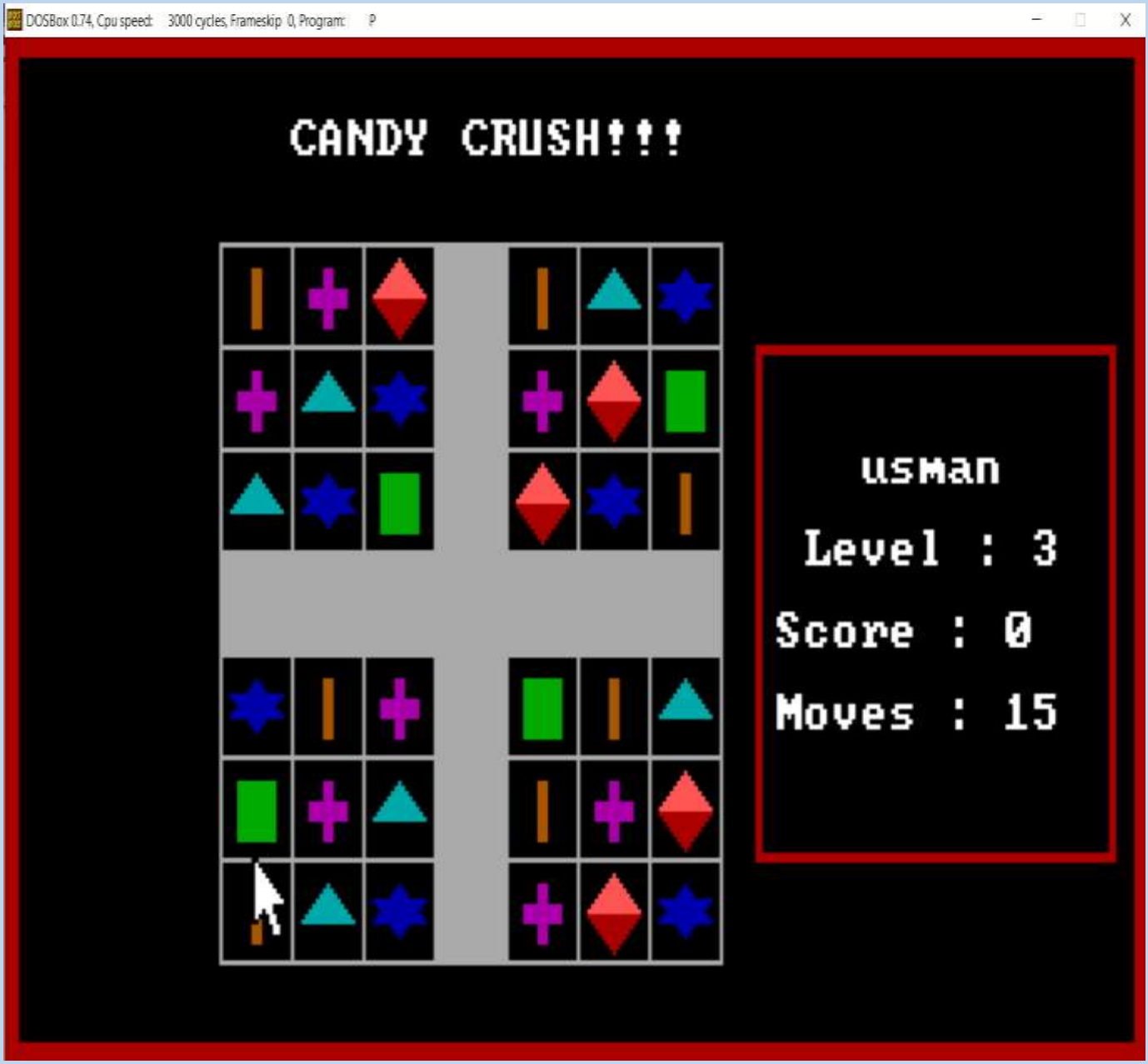
2. Level 2:

The candies were populated randomly and were displayed in the board. Also the name of the player, score, level number and moves are showing on the screen and changes accordingly. And the blockages are also displayed accordingly.



3. Level 3:

The candies were populated randomly and were displayed in the board. Also the name of the player, score, level number and moves are showing on the screen and changes accordingly. And the blockages are also displayed accordingly.

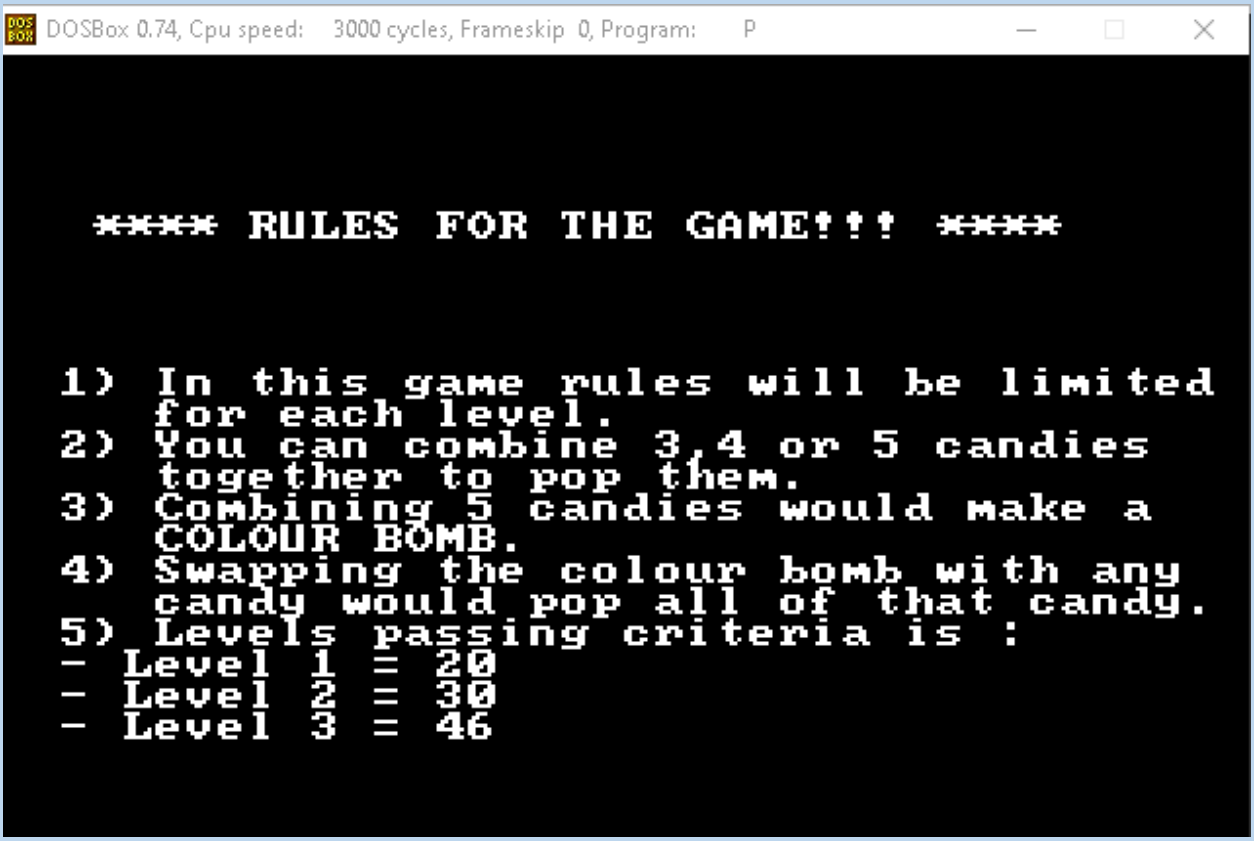


Game Output:

- **First page asking for player's name.**

- **Showing the welcome page.**

- **Page showing rules of the game.**

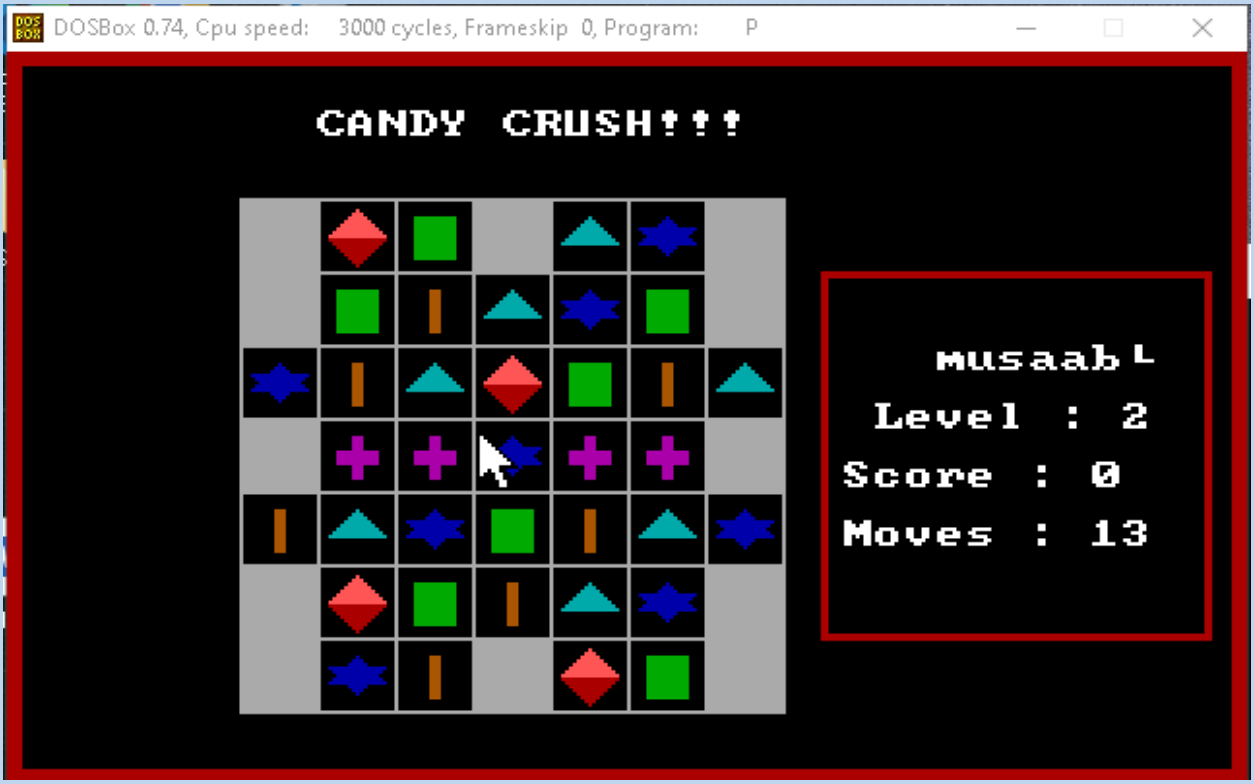


- **Page asking user for the particular level he/she wants to play.**

- **Showing the level 01 page.**

- **Showing the increase in score and making of color bomb.**

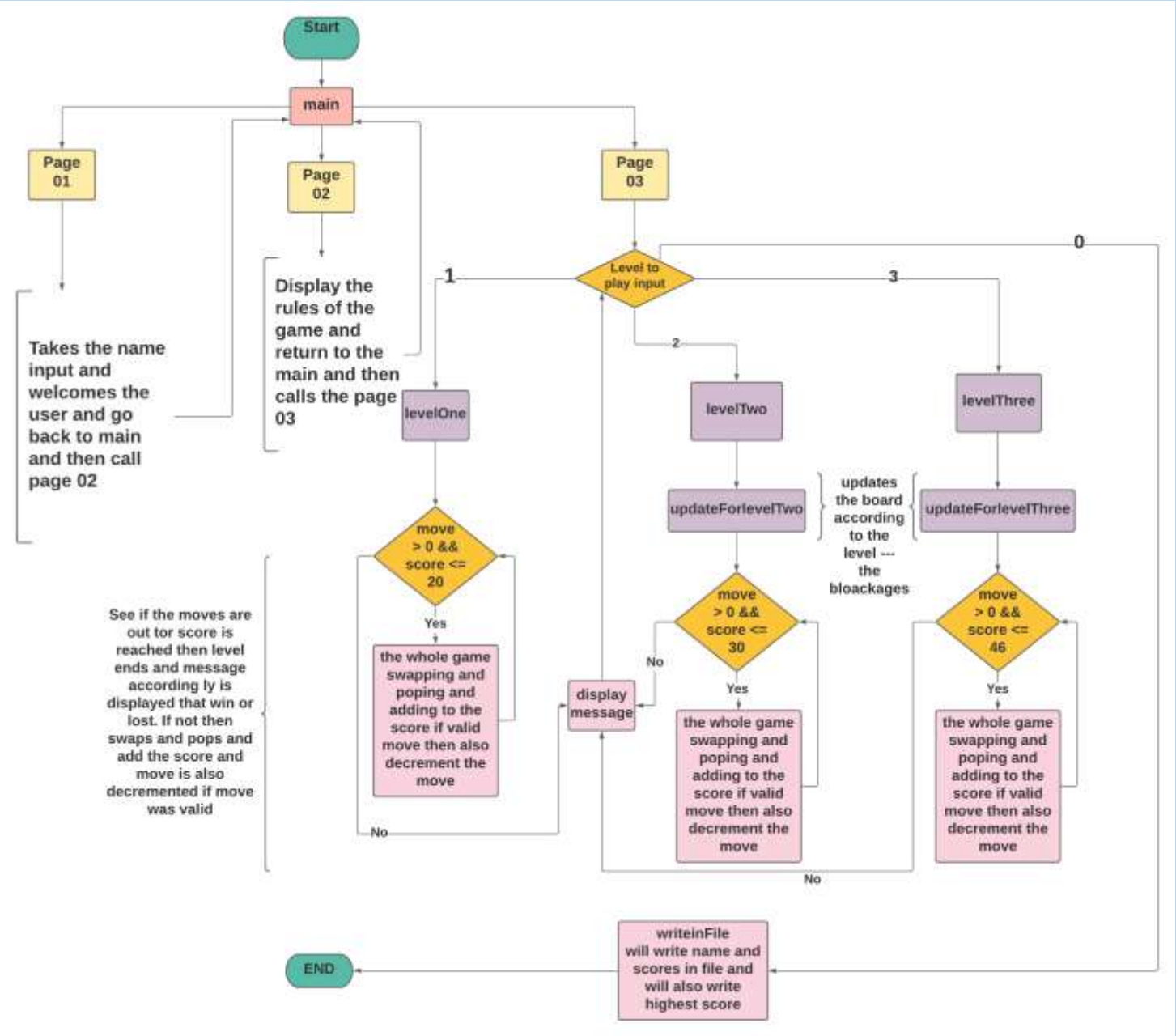
- **Level 02 starting page**



- **Increasing of score and making of color bomb.**

- **Level 03 page:**

Game Flow:



The flow chart clearly shows the game flow. Also attached the picture and pdf of above flow chart. Page 1 was for taking name of user and welcoming him. And page 2 for rules and page 3 was for the menu and when user enter 0 the game was ended. As shown above in the screen shots.

Game Logic:

Most of our work is done by using macros, we had used the procedures too.

- **Drawing board:**

The video mode selected was 13h having dimensions of **(320x200).** We draw the pixels for each box length 20x20 and the whole board was 7x7. Candies function was called and the board was displayed according to the codes of candy at the backend.

```
;drawing the grid
drawGrid MACRO color, pageNo

    ;our dimensions are 320x200
    ;we have to make 49 boxes as 7x7 each of 20x20
    ;mean whole board will be of 140x140
    ;as intending from above side of 40 thus 180
    ;also giving from right of 60  making 200
    ;thus the board will be displaying on 200x180
    ; -------------------------------------------------------

    ;pushing all general purpose registers
    PUSHA

    ;count of rows
    MOV AX, 40
    MOV BX, 60

    ;count of columns
    MOV CX, 60
    MOV DX, 80
```

```
; will visit the ---- 7 rows
.WHILE(AX<180)

    MOV CX, 60
    MOV DX, 80


    ;will print the borders of columns ... visit 7 coloumns
    .WHILE(CX<200)


        drawBorder CX, DX, AX, BX, color, 2
        ADD CX, 20
        ADD DX, 20

    .ENDW

    ADD AX, 20
    ADD BX, 20
```

C: > masm615 > BIN > project > p.asm

```
drawBoard MACRO color, pageNo, dummyArray, originalArray

    PUSHA
    ;for drawind boder main
    drawBorder 0, 319, 0, 199,  04h, 02
    drawBorder 1, 318, 1, 198,  04h, 02
    drawBorder 2, 317, 2, 197,  04h, 02
    drawBorder 3, 316, 3, 196,  04h, 02


    ;drawing the grid
    drawGrid color, page

    makeCandies originalArray


    ;the score side with level
    drawBorder 210, 310, 60, 160, 04h, 02
    drawBorder 211, 309, 61, 159, 04h, 02
```

C: > masm615 > BIN > project > p.asm

```
;the score side with level
drawBorder 210, 310, 60, 160, 04h, 02
drawBorder 211, 309, 61, 159, 04h, 02

nextLine

;setting the cursor
setCursor 10 ,30,0

;printing player name
printString playerN

;setting the cursor
setCursor 12 ,28,0

printString levelMsg

MOV DX, Level
MOV AH, 02
INT 21H
```

p.asm

C: > masm615 > BIN > project > p.asm

```
nextLine

;setting the cursor
setCursor 14 ,27,0

printString scoreMsg
MOV AX, score
displayMultidigit AX ;for displaying multi digits


;setting the cursor
setCursor 16 ,27,0

printString moveMsg
MOV AX , Moves
displayMultidigit AX ;for displaying multi digits

POPA

ENDM
```

- ## Getting mouse Positions:

Then mouse positions were taken and were mapped that which index it will be according to the array and the index was saved then a delay was given and second mouse click was taken and was mapped and then checked that the moves are valid or not different for each level if valid than swapped macro was called and swapping is done. Valid moves were that neighbors are or not also checked that click was in the board or not and also was checked that the not clicked on blockages in level 2 and level 3.

```
;for showing mouse on screen
showMouse MACRO xPos, yPos ···

;for restricting the mouse
restrictMouse MACRO minX, maxX, minY, maxY ···

;for checking the co ordinates of mouse click
mouseClick MACRO x, y   ;will save cordinates of x and y in passsed variables ···

;for checking the real co-ordinates the working
mapRealCod MACRO x, y , index ;will update  the reveersed variables that will contain the co ordi

;for swaping in the back end
swapUs MACRO element1, element2 ···

;for checking whether i2 is in neighbours of i1 or not and will return decision in result paramet
checkNeighbourhood MACRO i1, i2, result ···
```

```
C > masm615 > BIN > project > p.asm
;for checking the co ordinates of mouse click
mouseClick MACRO x, y   ;will save cordinates of x and y in passsed variables

    ;pushing the all general purpose registers
    PUSHA

    ;initallay nothing
    MOV BL, 0

    .WHILE ( BL != 1 ) ;the loop checks when to break when left mouse

        ;for getting mouse value
        MOV AX, 3
        INT 33H

        ;if left click
        .IF ( BL == 1 )

            ;saving the x and y cordinate
            ;--- X and Y co ordinate
```

```
            ;--- X and Y co ordinate
            MOV x, CX
            MOV y, DX


        .ENDIF

    .ENDW

    ;poping the all general purpose registers
    POPA


ENDM
```

p.asm

C: > masm615 > BIN > project > p.asm

```
;for checking the real co-ordinates the working
mapRealCod MACRO x, y , index ;will update  the reveersed variables that will contain the co ordi

    ;pushing the all general purpose registers
    PUSHA
    ;local variables
    PUSH temp ;used for mapping correct vlue
    PUSH var1 ;the counting

    MOV var1 , 0

    MOV AX, x
    SHR AX, 1 ;dividing with two
    MOV temp , AX  ;saving the value

    ;count of rows
    MOV AX, 40
    MOV BX, 60

    ;count of columns
    MOV CX, 60
```

```
; will visit the ---- 7 rows
.WHILE(CX<200 && index == -1)

    MOV AX, 40
    MOV BX, 60


    ;will print the borders of columns ... visit 7 coloumns
    .WHILE(AX<180 && index == -1)


        ;now checking which co-ordinate
        .IF (temp > CX && temp < DX && y > AX && y < BX)



            ;the co ordinate values aslo saving
            mov xStart , CX
            mov xEnd   , DX
            mov yStart , AX
```

```
            mov yEnd   , BX


            ;saving values
            PUSHA

                ;also saving the index
                MOV AX, var1
                MOV index, AX

                ;reset mouse
                MOV AX, 0
                INT 33H

            POPA ;getting back

            ;drawing boder against selected
            drawBorder CX, DX, AX, BX, 04h, 02

        .ENDIF
```

```
            ADD AX, 20
            ADD BX, 20

            ;count var1
            INC var1

        .ENDW

        ADD CX, 20
        ADD DX, 20



    .ENDW

    ;restore old value
    POP var1
    POP temp

    ;poping the all general purpose registers
    POPA
```

- **Swapping:**

If swapping is done, then move is also decremented

```
;for swaping in the back end
swapUs MACRO element1, element2

    ;pushing the all general purpose registers
    PUSHA
    PUSH SI
    PUSH DI


    MOV SI, OFFSET candyBoard ;address of first element
    MOV DI, OFFSET candyBoard ;address of second element

    ;as word size adding two times
    ADD SI, element1
    ADD SI, element1

    ;as word size adding two times
    ADD DI, element2
    ADD DI, element2
```

```
    ;swapping
    MOV AX, [SI]
    MOV DX, [DI]

    MOV [SI] ,DX
    MOV [DI] , AX

    POP DI
    POP SI
    ;poping the all general purpose registers
    POPA

ENDM
```

- **Popping:**

Score is updated accordingly.

```
;for checking combinations and poping
checkCombination MACRO i1, i2, result···

;for checking combinations
checkComboLeft MACRO element, result···

;for checking combinations
checkComboRight MACRO element, result···

;for checking combinations
checkComboTop MACRO element, result···

;for checking combinations
checkComboBottom MACRO element, result···

;the 5 combinations in row
makeBombHorizontal MACRO element, result···

;the 5 commbination in column
makeBombVertical MACRO element, result···
```

```
;the 5 commbination in column
makeBombVertical MACRO element, result···

;in between combination horizontallay
checkcomboinbetweenHorizontal MACRO element, result···

;in between combination vertically
checkcomboinbetweenVertically MACRO element, result···

;the bomb when swaped
bomInitiate MACRO element, element2, result···

;for updating and bringing candies down
updateBoard  MACRO return···
updateforLevelTwo MACRO ···

updateforLevelThree MACRO ···
```

- **Writing in File**

```
writeInfile MACRO

    PUSHA

    mov AH,3ch
    mov CL,2
    mov DX,offset fname
    int 21h

    mov AH,3dh
    int 21h

    mov CX,lengthof array
    mov BX,AX
    mov DX,offset array
    mov AH,40h
    int 21h

    mov DX, name
    mov AH,40h
    int 21h
```

```
mov dx,offset stringnewline
mov ah,09h
int 21h

mov DX, score1
mov AH,40h
int 21h

mov dx,offset stringnewline
mov ah,09h
int 21h

mov DX, score2
mov AH,40h
int 21h

mov dx,offset stringnewline
mov ah,09h
int 21h
```

```
    mov DX, score3
    mov AH,40h
    int 21h

    mov dx,offset stringnewline
    mov ah,09h
    int 21h

    mov DX, maxscore
    mov AH,40h
    int 21h


    POPA

ENDM
```

data.txt - Notepad

File Edit Format View Help

```
usman
22
31
36
36
```

THE END