**Brick Breaker**

INTRODUCTION

Brick breaker is actually based upon a “Pinball” based game where you have a horizontal flat surface to bump one or two balls where balls can break tiles and each broken tile gives a specific score. The objective of this game is to achieve the highest score possible. Keeping the balls inside the screen and making a highest score. The game ends when you lose all the balls i.e. they go out of the screen.

In order to achieve such animation, we need to draw bricks by using pixels of different colors. Once you draw a block, repeat this task for rest of the other bricks by changing axes. Once all the bricks have been drawn, draw ball(s) and the flat surface for it along with the boundaries of the screen where the ball(s) can move. When the whole frame is drawn, we remove the screen before the next frame comes in. Now this is the trickiest part of this game. We need to change the parameters before drawing the next frame. After changing them, we draw them again and this process keeps up until there are no bricks or we lose the game i.e. number of balls present in the screen is 0.

* There is a welcome screen
* There is an option screen where you can start a new game, get the high score page, read about the creator of the game, help about how to play or quit the game.
* In either choice there is a textual screen showing data about the field but the new game takes the user to the actual game. After the game ends, high score screen is shown.
* Quitting the game takes the user to the final good bye page and ends the game.

### MATERIAL TO READ

BOOK CHAP 19 for Graphic

INTERNET ARTICLES:

<http://www.atariarchives.org/agagd/chapter4.php>

C++ code link: <http://www.cplusplus.com/forum/general/117629/>

Color attributes: <https://en.wikipedia.org/wiki/BIOS_color_attributes>

Macros link:

<http://courses.ee.sun.ac.za/OLD/2003/Rekenaarstelsels245/8086_Instruksies/asm_tutorial_10.html>

Local : http://stackoverflow.com/questions/14559184/how-can-i-declare-local-variables-in-8086

INSTRCUTION USED IN PROJECT

Provide list of instruction has used in your project like”

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| MOV | ADD | SUB | DIV | MUL | SHL | SHR | ROL | ROR |
| MOVSB | CMP | CMPSB | RCR | RCL | SCASB | LODSB | LEA | INT |
| PUSH | POP | JMP | CALL | NEG | RET | MACRO | INCLUDE | LOCAL |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

Macros:

Macros are just like procedures, but not really. Macros look like procedures, but they exist only until your code is compiled, after compilation all macros are replaced with real instructions. If you declared a macro and never used it in your code, compiler will simply ignore it.

Procedure is located at some specific address in memory, and if you use the same procedure 100 times, the CPU will transfer control to this part of the memory. The control will be returned back to the program by **RET** instruction. The **stack** is used to keep the return address. The **CALL** instruction takes about 3 bytes, so the size of the output executable file grows very insignificantly, no matter how many time the procedure is used.

Local:

Macros are expanded directly in code, therefore if there are labels inside the macro definition you may get "Duplicate declaration" error when macro is used for twice or more. To avoid such problem, use **LOCAL** directive followed by names of variables, labels or procedure names.

FUNCTIONS DETAILS

MOV AL,13H:

Used to invert between display and text mode. Used along with INT 10h.

DELAYER:

* Makes a 20 milliseconds delay, can be changed by changing the value of BX.

SET\_DISPLAY\_MODE:

* Used to change from text to graphics mode.

DISPLAY\_BAR:

* Displays the horizontal bar where the ball bounces. Uses a set of pixel to draw it onto the screen.

MOVE\_BAR:

* Changes the position of bar by changes the axes.

DISPLAY\_BALL:

* Displays ball on the screen. Uses a set of pixel to draw it onto the screen.

MOVE\_BALL:

* Changes the position of ball by changes the axes.

CHECK\_BALL\_BOUNDARY:

* Makes sure that ball remain inside the screen. Alters some parameters when the ball’s y axis becomes greater than standard width (768).

BLOCK:

* Draws a block on the screen by drawing pixels on the screen.

FULLBLOCK:

* Uses same mechanism as BLOCK PROC but instead it draws the whole set.

CHECKHITBLOCK:

* Checks whether the ball hits the block or not. If it does, then Remove the block instantly.

MAIN\_menu:

* Shows main menu and a cursor.

WELCOME\_SCR:

* Shows a welcome screen.

YOU\_LOSE:

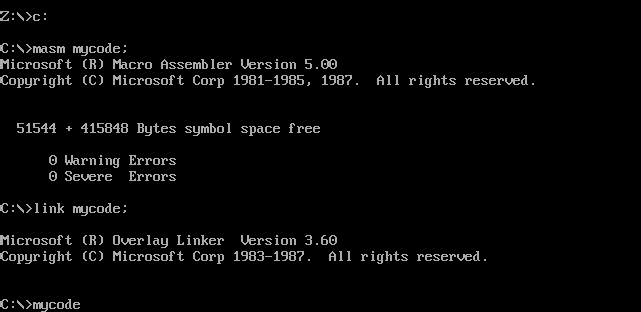
* Shows the high score and after an input, shows YOU LOSE message.

CLEAR\_SCREEN:

* Resets the console.

SCORE\_DISP:

* Shows the high score page.



PROJECT OUTPUT (SCREENSHOTS)

