Black Box

Black box is an abstract hide and seek game. In its original form it was a board game with one person providing the setup, hints, and tallying score and the other player making guesses. Of course setting up, providing hints, and tallying score is one of those things computers are perfectly suited for.

The game is played by firing rays into a black box where in are hiding a number of atoms and by observing the ray's reactions you can determine the location of the atoms. While it would be simple to win if you fired off every possible ray doing so will cost you points. Like golf the lower the score the better. With a score of 10 or less you can consider yourself a black box expert.

For more in depth rules, especially about ray interactions, see the instructions included in the game.

Black box is written by Joseph Larson based on a board game by Godfrey Hounsfield (also inspired by a BASIC program by Jeff Keton as found in 'More Basic Computer Games' edited by David H. Ahl (c) 1979)

BLACKBOX.C	You will need: a C/C++ complier .

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <time.h>

#define SIZE 8
#define NUM 4
#define FS(a) for (a = 1; a <= SIZE; a++)
#define RS (rand () % (SIZE) + 1)

void drawbox (int b[][SIZE + 2], char hint[], int opt) {
/* opt variable - 1: input coords 2: xy coords 3: end (show atoms) */
  int x, y;

  printf ("\n   "); /* 3 spaces */
  FS(x) printf(" %c",(opt < 2) ? '0' + ((4 * SIZE - x + 1) / 10)
    : " 123"[(x / 10)]);
  printf ("\n   "); /* 3 spaces */
  FS(x) printf(" %c",'0' + ((opt < 2) ? ((4 * SIZE - x + 1) % 10) : (x % 10)));
  printf ("\n   "); /* 3 spaces */
  FS(x) printf(" %c", hint[4 * SIZE - x]);
  FS(y) {
    printf ("\n%2d%c", y, hint[y - 1]);
    FS(x) printf (" %c", (opt < 3) ? '+' : (b[x][y]) ? 'O' : '+');
    if (opt < 2) printf (" %c%-2d", hint[3 * SIZE - y], 3 * SIZE - y + 1);
    else printf (" %c", hint[3 * SIZE - y]);
  }
  printf ("\n   "); /* 3 spaces */
  FS(x) printf(" %c", hint[SIZE + x - 1]);
  if (opt < 2) {
    printf ("\n   "); /* 3 spaces */
    FS(x) printf(" %c"," 1234"[(SIZE + x) / 10]);
    printf ("\n   "); /* 3 spaces */
    FS(x) printf(" %c",'0' + ((SIZE + x) % 10));
  }
  printf("\n\n");
}

void intro (void) {
  printf ("BlackBox\n--------\n"
  "Try to find %d atoms that are hidden in a Black Box by firing rays into the "
  "box\nand observing where they emerge.\nInput the starting place of the ray "
  "you want to fire. You will be told the\nresult of your shot. When you are "
  "ready to guess input ZERO (0) for your move.\n\n"
  "Atoms can effect the ray in the following ways:\n"
```

Listing continued on page 2...

```
  " 1) Direct impact is reported as a hit.\n"
  " 2) If a ray passes to one side of the atom it will be deflected.\n"
  " 3) If a ray enters exactly next to an atom it will be reflected out.\n"
  " 4) If a ray passes exactly between two rays it will be reflected, too.\n\n"
  "\t++++++\n"
  "\t+++O++            +++++   \t+++O    O+++\n"
  "\t--\\+++Deflection  ---O+Hit\t---+ or -+++ Reflection\n"
  "\t++|+++            +++++   \t+++O    ++++\n\n"
  "Score: Each entry and exit are 1 point, so deflections are 2 point, but\n"
  "reflections and hits are only one. At the end of the game each incorrect "
  "guess\nadds 5 points to your score.\n\nPress ENTER to begin.\n", NUM);
  getchar ();
}

void setup (int b[][SIZE + 2], char hint[]) {
  int x, y, c;

  for (x = 0; x < SIZE + 2; x++) for (y = 0; y < SIZE + 2; y++) b[x][y] = 0;
  for (c = 0; c < NUM; c++) {while (b[x = RS][y = RS]); b[x][y] = 1;
  }
  printf ("%d atoms hidden.\n", NUM);
  for (c = 0; c < SIZE * 4; c++) hint[c] = ' ';
}

int shoot (int start, char hints[], char *token, int b[][SIZE + 2]) {
  int x, y, dx, dy, temp;

  switch ((start - 1) / (SIZE)) { /* determine initial direction */
    case 0 : x = 1; y = start; dx = 1; dy = 0; break;
    case 1 : x = start - SIZE; y = SIZE; dx = 0; dy = -1; break;
    case 2 : x = SIZE; y = 3 * SIZE + 1 - start; dx = -1; dy = 0; break;
    case 3 : x = 4 * SIZE + 1 - start; y = 1; dx = 0; dy = 1;
  }
  /* check for edge reflections/hits */
  if (b[x][y]) {
    puts ("The shot was absorbed (a hit)");
    hints[start - 1] = 'H';
    return 1;
  }
  if (dx) {
      if (b[x][y - 1] + b[x][y + 1]) {
      puts ("The shot was reflected.");
      hints[start - 1] = 'R';
      return 1;
    }
  } else {
    if (b[x + 1][y] + b[x - 1][y]) {
      puts ("The shot was reflected.");
      hints[start - 1] = 'R';
      return 1;
    }
  }
  while (1) {
  /* as the shot moves the 4 corners around the shot are checked for reflections
     and the shot itself is checked for hits. */
    switch ((b[x][y] * 4) | (2 * b[x + 1][y - 1] + 2 * b[x - 1][y + 1]
      + b[x + 1][y + 1] + b[x - 1][y - 1])) {
      case 4 :
      case 5 :
      case 6 :
```

```
BLACKBOX.EXE

    3 3 3 2 2 2 2 2
    2 1 0 9 8 7 6 5
    c
 1  + + + + + + + +  24
 2b + + + + + + + +  23
 3  + + + + + + + +  22
 4  + + + + + + + +  21
 5a + + + + + + + + d20
 6c + + + + + + + +  19
 7  + + + + + + + +  18
 8  + + + + + + + +  17
    b   H   a   d
      1 1 1 1 1 1 1
    9 0 1 2 3 4 5 6

Ray # ?
```

Listing continued on page 3...


```
      case 7 : puts ("The shot was absorbed (a hit)");
               hints[start - 1] = 'H';
               return 1;
      case 1 : temp = dx; dx = -dy; dy = -temp; break;
      case 2 : temp = dx; dx = dy; dy = temp; break;
      case 3 : puts ("The shot was reflected.");
               hints[start - 1] = 'R';
               return 1;
    }
    x += dx; y += dy;
    if (!(x % (SIZE + 1)) || !(y % (SIZE + 1))) {
      if (!x) temp = y;
      else if (!y) temp = 4 * SIZE - x + 1;
      else if (x > SIZE) temp = 3 * SIZE - y + 1;
      else temp = SIZE + x;
      printf ("Ray exited at %d.\n", temp);
      hints[start - 1] = hints[temp - 1] = *token;
      *token = *token + 1;
      return 2;
    }
  }
}

int endgame (int b[][SIZE + 2], char h[], int *s) {
  int c, x, y;
  char input;

  printf ("Do you want to guess now? (y/n) ");
  while (!isalpha (input = getchar ()));
  if (tolower(input) != 'y') {
    drawbox(b, h, 1);
    return 0;
  }
  drawbox(b, h, 2);
  for (c = 0; c < NUM; c++) {
    printf ("What is the location of atom #%d? <x,y> ", c + 1);
    scanf ("%d %*c %d", &x, &y);
    if (b[x][y] == 1) {
      puts ("\nCorrect");
      b[x][y] = 10;
    } else {
      puts ("\nIncorrect. Plus 5 points.");
      *s += 5;
    }
  }
  drawbox (b, h, 3);
  printf ("\nYour final score : %d\nHere's the Blackbox.\n  ", *s);
  return 1;
}

int playmore (void) {
  char yesno;

  printf ("\nWould you like to play again? ");
  while (!isalpha (yesno = getchar ()));
  if (tolower(yesno) != 'n') return 1;
  return 0;
}
```

Listing continued on page 4...

BLACKBOX.C	Listing Continued from page 3….

```
int main (void) {
  int board[SIZE + 2][SIZE + 2], ray, score, done;
  char hints[SIZE * 4];
  char token;

  srand (time (NULL));
  intro ();
  do {
    setup (board, hints);
    token = 'a';
    score = 0;
    done = 0;
    do {
      drawbox (board, hints, 1);
      printf ("Ray # ? ");
      scanf ("%d", &ray);
      if (ray && ray < 4 * SIZE) score += shoot (ray, hints, &token, board);
      if (!ray) done = endgame (board, hints, &score);
    } while (!done);
  } while (playmore ());
  puts ("Goodbye!");
  exit (0);
}
```

Author's Notes:

Black box is a largely abstract game. As such the 8x8 board with 4 hidden pieces is purely a matter of preference, tho it is a good setup for beginners. When you feel ready for a harder challenge try making the board bigger (or smaller) and adding more pieces.

This game bears similarities to the game of minesweeper that comes with Microsoft Windows™. Like in minesweeper one obvious enhancement to black box would be to allow the player to mark spots they think are hiding atoms before the game's end. As for myself I used a piece of grid paper, but some people don't like that. The first draft of this program didn't even keep track of the results of your guesses for you, much more like the BASIC program of the same game. In fact many graphical implementations of this game and the original board game included that feature.

The first draft of this game that I wrote was actually based on the BASIC game, but I made a number of departures from the original wherever I thought the rules didn't make sense. For one I called hits "reflections" because I thought that would increase the challenge. Also, I designed it so that the outer edge never got any atoms on it, that way I'd never have to deal with the odd edge reflection rule that didn't make any sense to me.

Then I learned that the BASIC game was actually based on a board game inspired by Godfrey Hounsfield, the man who invented the CAT scan, and suddenly I didn't feel qualified to muck about with what I now saw as his rules. So I rewrote the program to include the omissions I had built in. It now plays exactly like the original.