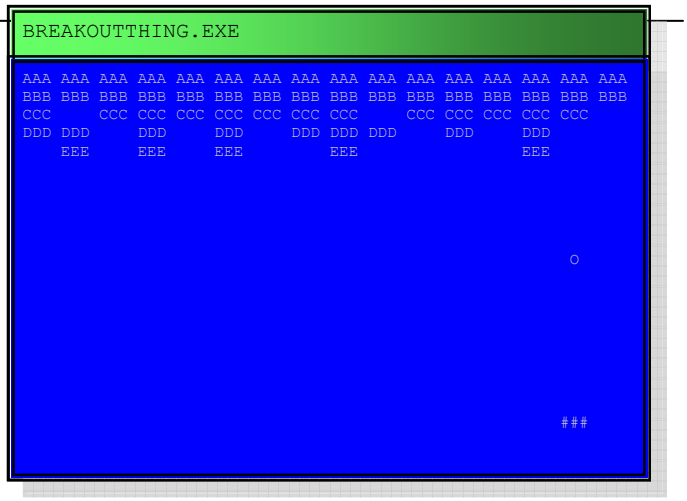
Breakout Thing



The game of breakout is perhaps one of the oldest known game types. Direct a paddle to keep a moving ball in play until all the bricks are eliminated to win. It's been made and remade the world over. And while the game itself is exciting the tedium of it can wear at times.

Breakout Thing does away with the tedium of moving the paddle by automating the process. All you have to do is sit back and relax as the computer wins the game for you. It's actually quite enjoyable to just sit and watch the game play, a bit like when I was younger and would go to the arcade with a dollar worth of quarters, but end up spending hours just watching the players that were so much better than I beat my favorite games on one credit.

Breakout thing is written by Andrew Paterson (Dragon) from Johannesburg, South Africa.

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

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

#define WALL_WIDTH  19
#define WALL_HEIGHT 5
#define SCREEN_WIDTH 76
#define SCREEN_HEIGHT 24

char wall[WALL_WIDTH][WALL_HEIGHT];
char brick[] = {'A', 'B', 'C', 'D', 'E'};
char display[(SCREEN_WIDTH+1)*SCREEN_HEIGHT+1];

int batx = SCREEN_WIDTH/2-1;
int baty = SCREEN_HEIGHT-1;
int batWidth = 3;
int ballx = SCREEN_WIDTH/2;
int bally = SCREEN_HEIGHT-2;
int speedx = 1;
int speedy = -1;

/***************************/
void PrintWall(void) {
    int x, y, i;
    int brickWidth;
    int index;

    brickWidth = (SCREEN_WIDTH / WALL_WIDTH) -1;
    for (y = 0; y < WALL_HEIGHT; y++) {
        for (x = 0; x < WALL_WIDTH; x++) {
            for (i = 0; i < brickWidth; i++) {
                index = ((x*(brickWidth+1))+i) + (y+1)*(SCREEN_WIDTH+1);
                display[index] = wall[x][y];
            }
        }
    }
}

/***************************/
void PrintSpace(void) {
```

Listing continued on page 2...

```
    int x, y;

    for (y = 0; y < SCREEN_HEIGHT; y++) {
        for (x = 0; x < SCREEN_WIDTH+1; x++) {
            display[x +(SCREEN_WIDTH+1)*y] = ' ';
        }
        display[SCREEN_WIDTH+(SCREEN_WIDTH+1)*y] = '\n';
    }
    display[(SCREEN_WIDTH+1)*SCREEN_HEIGHT] = '\0';
}

/***************************/
void PrintBat(void) {
    int i;

    for (i = 0; i < batWidth; i++) {
        display[(batx + i) + (SCREEN_WIDTH+1)*baty] = '#';
    }
}
/***************************/
void PrintBall(void) {
    display[ballx + (SCREEN_WIDTH+1)*bally] = 'O';
}

/***************************/
void SetupWall(void) {
    int x, y;

    for (y = 0; y < WALL_HEIGHT; y++) {
        for (x = 0; x < WALL_WIDTH; x++) {
            wall[x][y] = brick[y];
        }
    }
}

/***************************/
void MoveBall(void) {
    int brickWidth;
    int wallx;
    int wally;

    bally += speedy;
    ballx += speedx;

    if ((bally >= 1) && (bally < WALL_HEIGHT+1)) {
        brickWidth = (SCREEN_WIDTH / WALL_WIDTH);
        if (ballx % brickWidth != brickWidth-1) {
            wallx = ballx/brickWidth;
            wally = bally-1;

            if (wall[wallx][wally] != ' ') {
                wall[wallx][wally] = ' ';
                speedy = 1;
                bally += 2;
            }
        }
    }
    if (bally < 0) {
        bally = 1;
        speedy = 1;
```

Listing continued on page 3...

```
    }
    if (bally == baty) {
        bally = baty-1;
        speedy = -1;
    }

    if (ballx < 0) {
        ballx = 1;
        speedx = 1;
    }
    if (ballx > SCREEN_WIDTH-1) {
        ballx = SCREEN_WIDTH-2;
        speedx = -1;
    }
}

/***************************/
void MoveBat(void) {
    batx = ballx-1;
    if (batx < 0) {
        batx = 0;
    }
    if (batx > SCREEN_WIDTH-batWidth) {
        batx = SCREEN_WIDTH-batWidth;
    }
}

/***************************/
void SlowDown(void) {
    int start, current;

    start = clock();
    do {
        current = clock();
    } while (current - start < 50);
}

/***************************/
int Ended(void) {
    int count;
    int x, y;

    if (bally > baty) {
        printf("Whoopsy, you missed the ball.\n\n");
        return true;
    }
    count = 0;
    for (y = 0; y < WALL_HEIGHT; y++) {
        for (x = 0; x < WALL_WIDTH; x++) {
            if (wall[x][y] != ' ') {
                count++;
            }
        }
    }
    if (count == 0) {
        printf("Well Done!\n\n");
        return true;
    }
    return false;
}
```

Listing continued on page 4...

TITLE.C	Listing Continued from page 3….

```
/***************************/
void Draw(void) {
    printf(display);
}
/***************************/
int main(int argc, char** argv) {
    SetupWall();

    while (!Ended()) {
        PrintSpace();
        PrintWall();
        PrintBat();
        PrintBall();

        MoveBall();
        MoveBat();

        Draw();
        SlowDown();
    }
    return 0;
}
```

Editor's Notes:

The limitations of the standard C++ library so not support action based games very well due to the lack of being able to check input without pause and to write output anywhere on the screen.

This game uses a method similar to 3DMaze or Robot Escape to draw the screen, that is it simply draws what would be a whole screen of text at once and relies on the screen to scroll fast enough that you don't see it. However unlike 3DMaze or Robot Escape this game isn't inhibited by player input, it must force it's own pause between screen redraws. Overall this is the closest the standard library can get to an action based game.