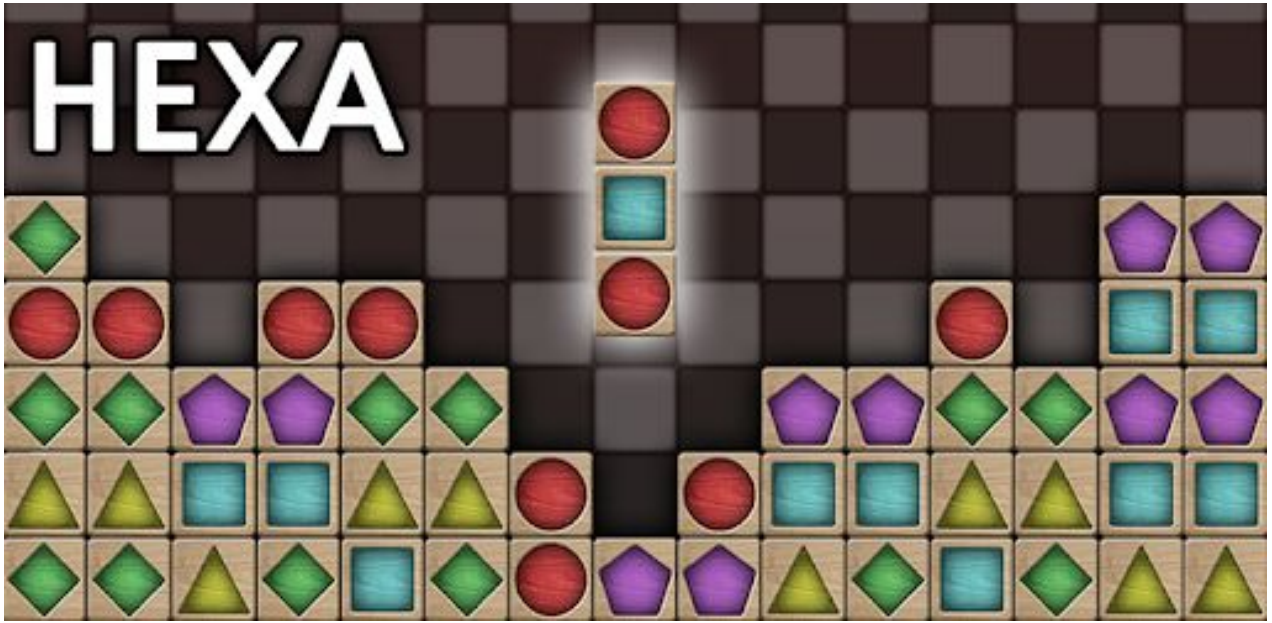
Hexa

Team.01
이대건 장세진 박종섭 김상인

목차

HEXA



Main()

```
void main()
{
    int nFrame, nStay;
    int x, y;

    setcursortype(NOCURSOR);
    randomize();
    level = 2;

    for (; ;) {
        clrscr();
        // 벽과 빈공간 생성 -> 보드 초기화
        for (x = 0; x < BW + 2; x++) {
            for (y = 0; y < BH + 2; y++) {
                board[x][y] = (y == 0 || y == BH + 1 || x == 0 || x == BW + 1) ? WALL : EMPTY;
            }
        }
        DrawScreen();
        nFrame = 20;
        score = 0;
        bricknum = 0;
        //처음 떨어질 벽돌
        MakeNewBrick();
```

Main()

```
for (; ;) {
    bricknum++;

    memcpy(brick, nbrick, sizeof(brick));
    MakeNewBrick();
    DrawNext();
    nx = BW / 2;
    ny = 3;

    //PrintBrick(TRUE);


    if (GetAround(nx, ny) != EMPTY) break;

    nStay = nFrame;
```

Main()

```
nStay = nFrame;
for (; ;) {
    if (--nStay == 0) {
        nStay = nFrame;
        if (MoveDown()) break;
    }
    if (ProcessKey()) break;
    delay(1000 / 20);
}
```

Main()

```
clrscr();
gotoxy(30, 12); puts("G A M E  O V E R");
gotoxy(25, 14); puts("다시 시작하려면 Y를 누르세요");
if (tolower(getch()) != 'y') break;
```

PrintInfo()

```
void PrintInfo()
{
    gotoxy(50, 9); printf("난이도 : %d  ", level);
    gotoxy(50, 10); printf("점수 : %d     ", score);
    gotoxy(50, 11); printf("벽돌 : %d 개  ", bricknum);
}
```

MakeNewBrick()

```
void MakeNewBrick()
{
    int i;

    do {
        for (i = 0; i < 3; i++) {
            nbrick[i] = random(level) + 1;
        }
    } while (nbrick[0] == nbrick[1] && nbrick[1] == nbrick[2] && nbrick[0] == nbrick[2]);
}
```

#define / 상수 / 전역변수

```
// keyboard 입력
#define LEFT 75
#define RIGHT 77
#define UP 72
#define DOWN 80

// 난이도 조절 및 종료키
#define PGUP 73
#define PGDN 81
#define ESC 27

// 생성위치
#define BX 5
#define BY 1

// 게임 스크린
#define BW 10
#define BH 20
```

```
enum {
    EMPTY,              // 값이 아무것도 지정안되있으면 처음 숫자는 0
    B1, B2,             // 다음 숫자부터는 +1씩 증가해서 값을 저장해둔다.
    B3, B4,
    B5, B6,
    B7, B8,
    B9, B10,
    WALL };

char* arTile[] = { ". ","■",
"●","★","♣","☎","◆","♠",
"♥","♨","♬","□" };

int board[BW + 2][BH + 2];
int nx, ny;         // 놓을 블럭의 x,y 좌표
int brick[3];       // 블럭
int nbrick[3];      // 임의로 다음 나올꺼
int score;          // 점수
int bricknum;       // 생성된 블럭개수
int level;          // 난이도
```

DrawScreen()

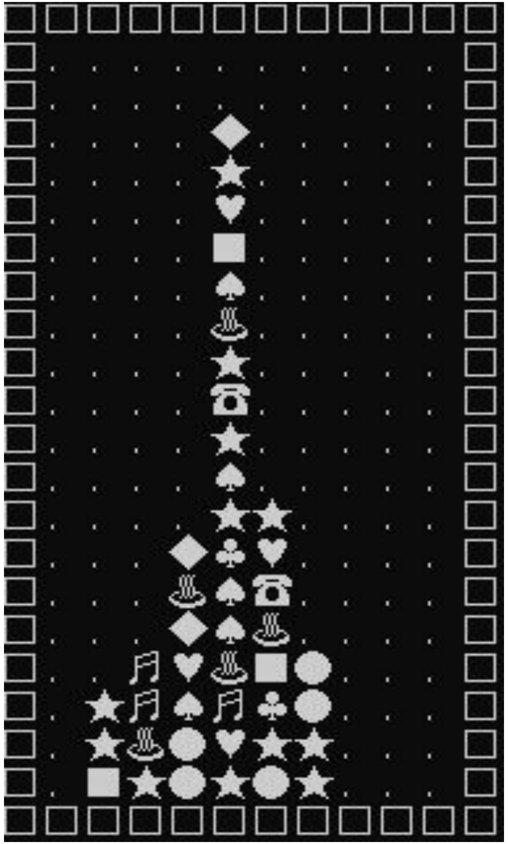
```
void DrawScreen()
{
    int x, y;

    for (x = 0; x < BW + 2; x++) {
        for (y = 0; y < BH + 2; y++) {
            gotoxy(BX + x * 2, BY + y);
            puts(arTile[board[x][y]]);
        }
    }

    gotoxy(50, 3); puts("Hexa Ver 1.0");
    gotoxy(50, 5); puts("좌우:이동, 위:회전, 아래:내림");
    gotoxy(50, 6); puts("공백:전부 내림, ESC:종료");
    gotoxy(50, 7); puts("P:정지,PgUp,PgDn:난이도 조절");
    DrawNext();
    PrintInfo();
}
```

```
char* arTile[] = { ". ","■",
"●","★","✚","☎","◆","♠",
"♥","♨","♬","□" };
```

DrawScreen()



11	11	11	11	11	11	11	11	11	11	11	11
11	0	0	0	0	0	0	0	0	0	0	11
11	0	0	0	0	0	0	0	0	0	0	11
11	0	0	0	0	0	0	0	0	0	0	11
11	0	0	0	0	0	0	0	0	0	0	11
11	0	0	0	0	0	0	0	0	0	0	11
11	0	0	0	0	0	0	0	0	0	0	11
11	0	0	0	0	0	0	0	0	0	0	11
11	0	0	0	0	0	0	0	0	0	0	11
11	0	0	0	0	3	0	0	0	0	0	11
11	0	0	0	0	5	0	0	0	0	0	11
11	0	0	0	0	3	0	0	0	0	0	11
11	0	0	0	0	7	0	0	0	0	0	11
11	0	0	0	0	3	3	0	0	0	0	11
11	0	0	0	6	4	8	0	0	0	0	11
11	0	0	0	9	7	5	0	0	0	0	11
11	0	0	0	6	7	9	0	0	0	0	11
11	0	0	10	8	9	1	2	0	0	0	11
11	0	3	10	7	10	4	2	0	0	0	11
11	0	3	9	2	8	3	3	0	0	0	11
11	0	1	3	2	3	2	3	0	0	0	11
11	11	11	11	11	11	11	11	11	11	11	11

DrawBoard();

```
void DrawBoard()
{
    int x, y;

    for (x = 1; x < BW + 1; x++) {
        for (y = 1; y < BH + 1; y++) {
            gotoxy(BX + x * 2, BY + y);
            puts(arTile[board[x][y]]);
        }
    }
}
```

BOOL ProcessKey()

```
#define LEFT 75
#define RIGHT 77
#define UP 72
#define DOWN 80
#define PGUP 73
#define PGDN 81
```

```
BOOL ProcessKey()
{
    int ch;
    int t;

    if (_kbhit()) {
        ch = _getch();
        if (ch == 0xE0 || ch == 0) {
            ch = _getch();
            switch (ch) {
            case LEFT:
                if (GetAround(nx - 1, ny) == EMPTY) { // 현블럭 x -1 값이 empty 면
                    PrintBrick(FALSE);
                    nx--;
                    PrintBrick(TRUE);
                }
                break;
            case RIGHT:
                if (GetAround(nx + 1, ny) == EMPTY) { // 현블럭 x +1 값이 empty 면
                    PrintBrick(FALSE);
                    nx++;
                    PrintBrick(TRUE);
                }
```

BOOL ProcessKey()

```
case UP:
    PrintBrick(FALSE);
    t = brick[0];
    brick[0] = brick[1];  // 블럭의 순서 swap
    brick[1] = brick[2];
    brick[2] = t;
    PrintBrick(TRUE);
    break;
case DOWN:
    if (MoveDown()) {
        return TRUE;
    }
    break;
case PGDN:
    if (level > 2) {
        level--;
        PrintInfo();
    }
    break;
case PGUP:
    if (level < 10) {
        level++;
        PrintInfo();
    }
    break;
```

BOOL ProcessKey()

```
        else {
            switch (tolower(ch)) {
            case ' ':
                while (MoveDown() == FALSE) { ; }
                return TRUE;
            case ESC:
                exit(0);
            case 'p':
                clrscr();
                gotoxy(15, 10);
                puts("Tetris 잠시 중지. 다시 시작하려면 아무 키나 누르세요.");
                _getch();
                clrscr();
                DrawScreen();
                PrintBrick(TRUE);
                break;
            }
        }
    }
    return FALSE;
}
```

대문자->소문자

MoveDown() 이 FASLE값을 반활할 때까지 계속 실행하여라

프로그램 재가동

PrintBrick()

```
void PrintBrick(BOOL Show)
{
    int i;

    for (i = 0; i < 3; i++) {
        gotoxy(BX + nx * 2, BY + ny + i);
        puts(arTile[Show ? brick[i] : EMPTY]);
    }
}
```

TRUE 면 brick[i] 블럭 그리기

FALSE 면 EMPTY ' . ' 그리기

GetAround(int x, int y)

```
int GetAround(int x, int y)
{  //ex ) y = 7
      //  7,8,9 리턴
    int i, k = EMPTY;  // EMPTY = 0

    for (i = 0; i < 3; i++) {
        k = max(k, board[x][y + i]);  //두가지 값
                                      //비교해서 큰값
    }
    return k;
}
```

입력받는 매개변수 x, y

Move Down()

```
BOOL MoveDown()
{
    if (GetAround(nx, ny + 1) != EMPTY) {
        TestFull();
        return TRUE;
    }
    PrintBrick(FALSE);
    ny++;
    PrintBrick(TRUE);
    return FALSE;
}
```

TestFull()

```
int i, x, y;
int t, ty;
BOOL Remove;
static int arScoreInc[] = { 0,1,3,7,15,30,100,500 };
int count = 0;
BOOL Mark[BW + 2][BH + 2];//.

// 배열에 기록
for (i = 0; i < 3; i++) {
    board[nx][ny + i] = brick[i];
}
```

```
DrawBoard();
score += arScoreInc[min(count / 3, 7)];
PrintInfo();
```

```
for (;;) {
    // 연속 무늬 점검
    memset(Mark, 0, sizeof(Mark)); //
    Remove = FALSE;
    for (y = 1; y < BH + 1; y++) {
        for (x = 1; x < BW + 1; x++) {
            t = board[x][y];
            if (t == EMPTY) continue;

            // 수평
            if (board[x - 1][y] == t && board[x + 1][y] == t) {
                for (i = -1; i <= 1; i++) Mark[x + i][y] = TRUE;
                Remove = TRUE;
            }
            // 수직
            if (board[x][y - 1] == t && board[x][y + 1] == t) {
                for (i = -1; i <= 1; i++) Mark[x][y + i] = TRUE;
                Remove = TRUE;
            }
            // 우하향
            if (board[x - 1][y - 1] == t && board[x + 1][y + 1] == t) {
                for (i = -1; i <= 1; i++) Mark[x + i][y + i] = TRUE;
                Remove = TRUE;
            }
            // 좌하향
            if (board[x + 1][y - 1] == t && board[x - 1][y + 1] == t) {
                for (i = -1; i <= 1; i++) Mark[x - i][y + i] = TRUE;
                Remove = TRUE;
            }
        }
    }
```

TestFull()

```
    // 제거 애니메이션
    for (i = 0; i < 6; i++) {
        for (y = 1; y < BH + 1; y++) {
            for (x = 1; x < BW + 1; x++) {
                if (board[x][y] != EMPTY && Mark[x][y] == TRUE) {
                    gotoxy(BX + x * 2, BY + y);
                    puts(arTile[i % 2 ? EMPTY : board[x][y]]);
                }
            }
        }
        delay(150);
    }

    // 연속된 무늬 삭제
    for (y = 1; y < BH + 1; y++) {
        for (x = 1; x < BW + 1; x++) {
            if (board[x][y] != EMPTY && Mark[x][y] == TRUE) {
                for (ty = y; ty > 1; ty--) {
                    board[x][ty] = board[x][ty - 1];
                }
                count++;
            }
        }
    }

    DrawBoard();
    score += arScoreInc[min(count / 3, 7)];
    PrintInfo();
}
```

DrawNext()

```
void DrawNext()
{
    int x, y, i;

    for (x = 50; x <= 70; x += 2) {
        for (y = 12; y <= 18; y++) {
            gotoxy(x, y);
            puts(arTile[(x == 50 || x == 70 || y == 12 || y == 18) ? WALL : EMPTY]);
        }
    }

    for (i = 0; i < 3; i++) {
        gotoxy(60, 14 + i);
        puts(arTile[nbrick[i]]);
    }
}
```

Q&A

END

Team.01
이대건 장세진 박종섭 김상인