C++ Programming	Student number	21300691
Homework 6	Name	Cheung, Won Sik

1. Problem Definition

   I want to make a brick-breaking game. However, I do not have enough time during the semester, so I want to write a ball-bouncing program. If you just bounce the ball is not fun, add another function.

2. Statistics

   - number of lines: 485 -> 495

   - number of classes: 1

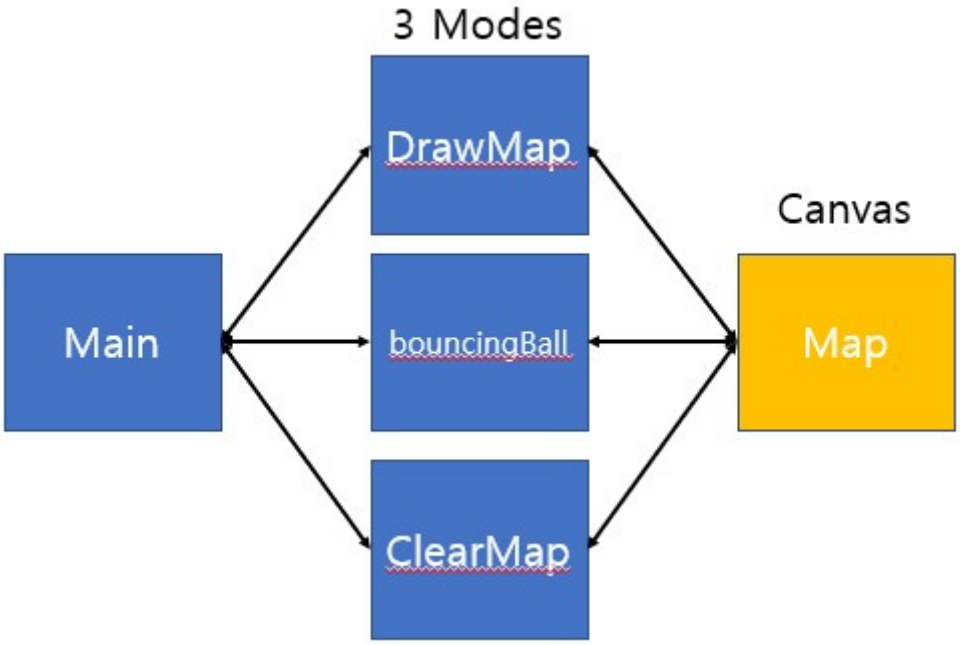
   - number of functions: 16

3. Design of my program

   - Class Diagram

```
class Canvas{
private:
    // variable which correspond with screen
    char map[24][80];
public:
    // Constructor
    Canvas(){
        initMap();
    }
    /****** function for mapCanvas ******/
    void mapCanvas();
    void markXY(int x, int y, int mode);
    void markBoundary();
    void initMap();
    void displayXY(int x, int y);
    void displayMap();

    /******* Bouncing ball function  *******/
    void bouncingBall(int sx, int sy, int dx, int dy);
};
```

- Flow Chart



4. Primary codes

- Functions

```
/********* base function *********/
void gotoxy(int x, int y);
void clrscr();
void gotoxyAdjust(int x, int y, int adjX, int adjY);
void printOption();
int absolute(int num);
int isWall(char ch);
int isBlock(char ch);
int isBlank(char ch);
int isWallBlock(char ch);
```

- Main Function-1

```
int main(){
    char ch = 0;
    Canvas screen = Canvas();

    // initialize screen array
    clrscr();
    screen.markBoundary();

    // show 3 options and activate
    do{
        switch(ch){

        // option1: draw map
        case '1':
            clrscr();
            screen.mapCanvas();
            break;
        // option2: bouncing ball
        case '2':
            clrscr();
            screen.bouncingBall(10, 10, 2, 1);
            break;
        // option3: clear screen array
        case '3':
            clrscr();
            screen.initMap();
            screen.markBoundary();
            break;
        }
```

- Main Function-2

```
        //show options
        clrscr();
        printOption();
        ch = getch();

    }while(ch!=27);

    clrscr();
    gotoxy(1,1);
    cout<<"Good Bye~"<<endl;
    system("pause");
    return 0;
}
```

- Movement Functions

```
// go screen x,y position
void gotoxy(int x, int y){
   COORD Pos = {(x - 1), (y - 1)};

   SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), Pos);
}

// clear screen
void clrscr(void){
   COORD Cur= {0, 0};
   unsigned long dwLen;
   FillConsoleOutputCharacter(GetStdHandle(STD_OUTPUT_HANDLE) , ' ', 80*25, Cur, &dwLen);
   Cur = {10, 12};
   FillConsoleOutputCharacter(GetStdHandle(STD_OUTPUT_HANDLE) , ' ', 80*25, Cur, &dwLen);
}

// go screen x,y position which adjusted by adjX, adjY
void gotoxyAdjust(int x, int y, int adjX, int adjY){
   COORD Pos = {x + adjX - 1, y + adjY - 1};

   SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), Pos);
}
```

- mapCanvas()

```
// draw map by block before start game
void Canvas::mapCanvas(){
    int key = 0;
    int x = 0, y = 0;
    int oldx = 0, oldy = 0;
    int mode = MODE_MOVE;
    int oldMode = -1;
    string message = "1: Move, 2: Draw, 3: Erase, 4: Reverse  ||  i: up, j: left, l: right, k: down ";

    // initialize part
    displayMap();

    gotoxy(2, 2);
    cout<<"Map Canvas ";
    gotoxyAdjust(1, BOTTOM_BOUND+1, ADJUST_X, ADJUST_Y);
    cout<<message;

    gotoxyAdjust(1, BOTTOM_BOUND+2, ADJUST_X, ADJUST_Y);
    cout<<"Press ESC to go next";

    oldx = x = (LEFT_BOUND + RIGHT_BOUND) / 2;
    oldy = y = (TOP_BOUND + BOTTOM_BOUND) / 2;
```

```
// draw part
do {
    gotoxyAdjust(x, y, ADJUST_X, ADJUST_Y);
    putchar('*');

    if(oldx != x || oldy != y)
        displayXY(oldx, oldy);

     key = getch();

    oldx = x;
    oldy = y;

    oldMode = mode;

    switch(key){
    // move cursor
    case 'j':
        if(x - 1 > LEFT_BOUND)
            x--;
        break;
    case 'l':
        if(x + 1 < RIGHT_BOUND)
            x++;
        break;
     case 'i':
        if(y - 1 > TOP_BOUND)
            y--;
        break;
     case 'k':
        if(y + 1 < BOTTOM_BOUND)
            y++;
        break;
```

```
        // change drawing mode
        case '1':
            mode = MODE_MOVE;
            break;
        case '2':
            mode = MODE_DRAW;
            break;
        case '3':
            mode = MODE_ERASE;
            break;
        case '4':
            mode = MODE_REVERSE;
            break;
        }

        markXY(x, y, mode);
    // while loop break when player enter ESC
    } while (key != 27);

    clrscr();
    gotoxy(1, 1);
}
```

- bouncingBall() -1

```
// Bouncing ball appear and continue bounce and draw block
void Canvas::bouncingBall(int sx, int sy, int dx, int dy)
{
    // sx, sy : first position of x, y
    // dx, dy : x, y 's speed

    // x,y : current position
    // oldx, oldy : where x, y passess position
    // fdx, fdy : real speed of x, y
    // max : speed control variable. The smaller the speed, the faster.
    // n : clock variable when n%max == 0 cursor move
    float x = sx, y = sy;
    float oldx = 0, oldy = 0;
    float fdx = 0., fdy = 0.;
    int max = dx;

    char key = 0;
    int n = 0;

    // set fdx, fdy, max
    if(dx != 0 && absolute(dx) >= absolute(dy)){
        fdx = (dx > 0 ? 1. : -1.);
        fdy = dy / (float)absolute(dx);
    } else if(dy != 0){
        fdx = dx / (float)absolute(dy);
        fdy = (dy > 0 ? 1. : -1.);
        max = dy;
    } else {
        fdx = dx;
        fdy = dy;
    }
```

- bouncingBall() -2

```
// for safety
if(max <= 0)
    max = 1;

// initialize part
displayMap();

gotoxy(2, 2);
cout<<"Bouncing Ball ";
gotoxyAdjust(1, BOTTOM_BOUND + 1, ADJUST_X, ADJUST_Y);
cout<<"Press ESC to go next";

x = oldx = sx;
y = oldy = sy;


do {
    if(n % max == 0){
        // display ball
        gotoxyAdjust((int)x, (int)y, ADJUST_X, ADJUST_Y);
        putchar('*');

        // erase previous ball
        if(x != oldx || y != oldy){
            markXY(oldx, oldy, MODE_DRAW);
            gotoxyAdjust((int)oldx, (int)oldy, ADJUST_X, ADJUST_Y);
            putchar('#');
        }
    }

    Sleep(100 / max);
```

- bouncingBall() -3

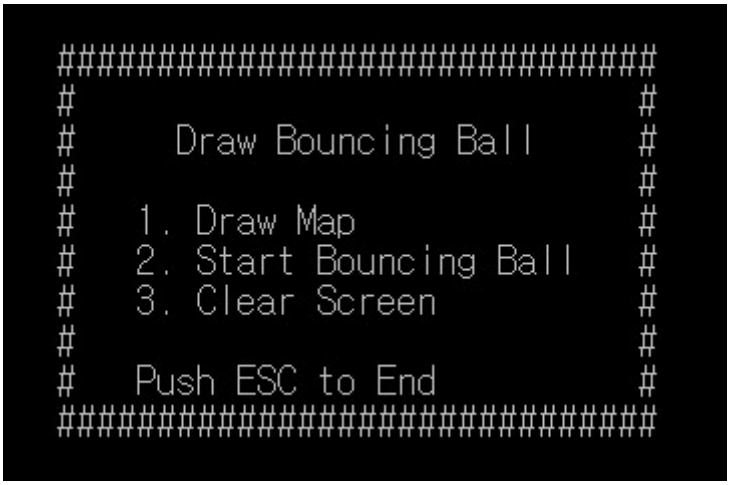
```
// save current position
if(n % max == 0){
    oldx = x;
    oldy = y;
}

x += fdx;
y += fdy;

if(isWallBlock(map[(int)y][(int)x])){
    // meet vertical wall
    if(isWallBlock(map[(int)oldy][(int)x]) && isBlank(map[(int)y][(int)oldx])){
        fdx = -fdx;
        x = oldx + fdx;
        // bounced again
        if(isWallBlock(map[(int)y][(int)x])){
            // corner
            if(isBlank(map[(int)oldy][(int)oldx])){
                x = oldx;
                y = oldy;
                fdy = -fdy;
            } else {
                fdx = -fdx;
                x = oldx;
            }
        }
    // meet horizontal wall
    } else if(isBlank(map[(int)oldy][(int)x]) && isWallBlock(map[(int)y][(int)oldx])){
        fdy = -fdy;
        y = oldy + fdy;
```

- bouncingBall() -4

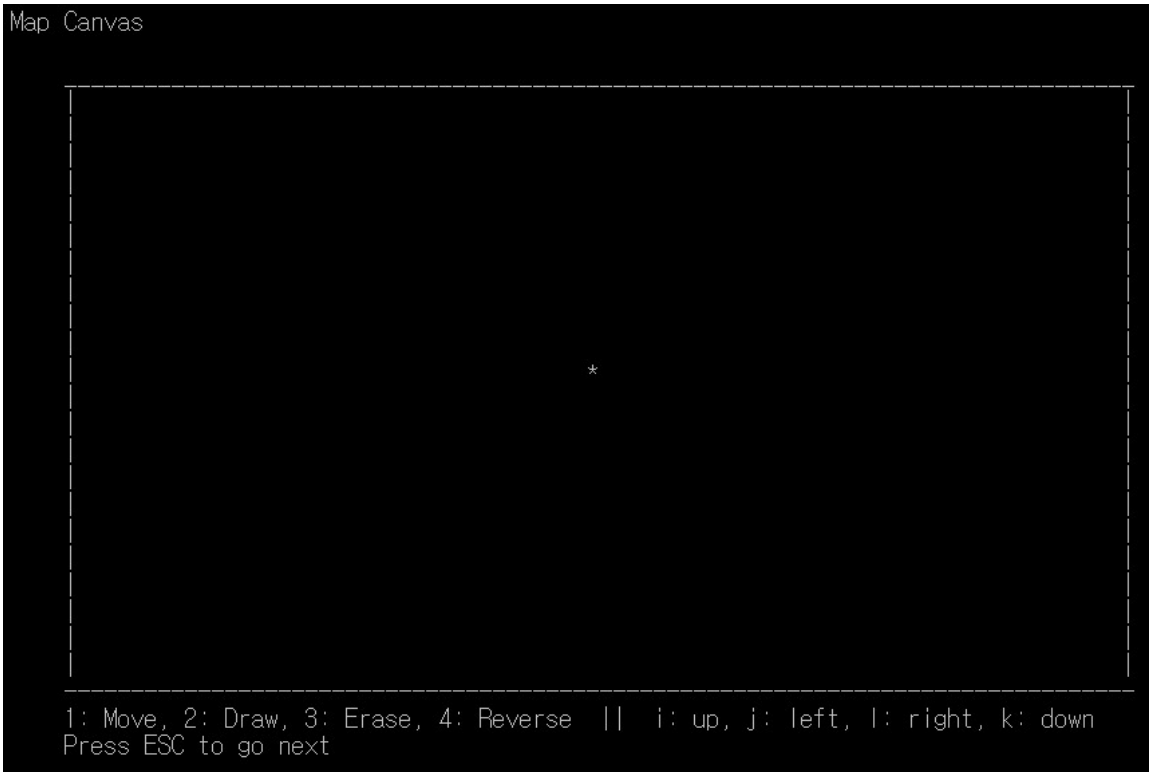
```
                // bounced again
                if(isWallBlock(map[(int)y][(int)x])){
                    // corner
                    if(isBlank(map[(int)y][(int)oldx])){
                        x = oldx;
                        y = oldy;
                        fdx = -fdx;
                    } else {
                        fdy = -fdy;
                        y = oldy;
                    }
                }
            // corner
            } else {
                fdx = -fdx;
                fdy = -fdy;
                x = oldx + fdx;
                y = oldy + fdy;
                // bounced again
                if(isWallBlock(map[(int)y][(int)x])){
                    x = oldx;
                    y = oldy;
                }
            }
        }

        if(kbhit())
            key = getch();
        n++;
    } while (key != 27);
}
```

5. Screenshot of the result

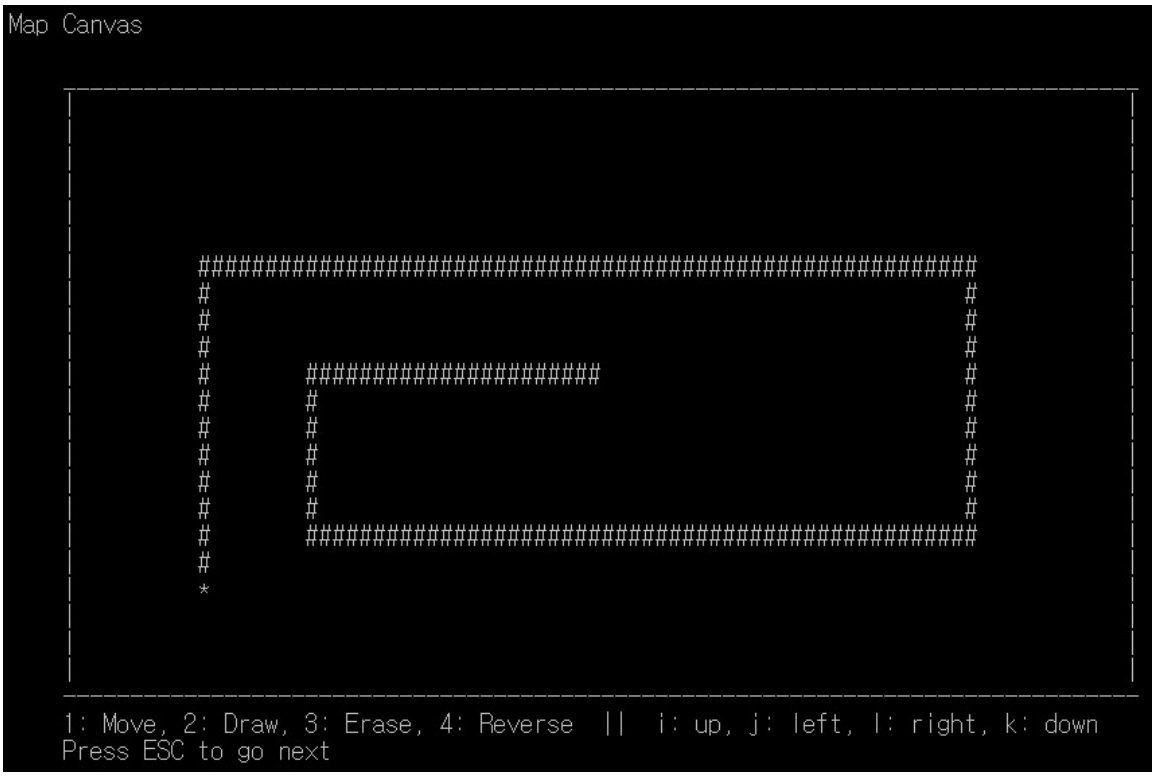
- Main Page



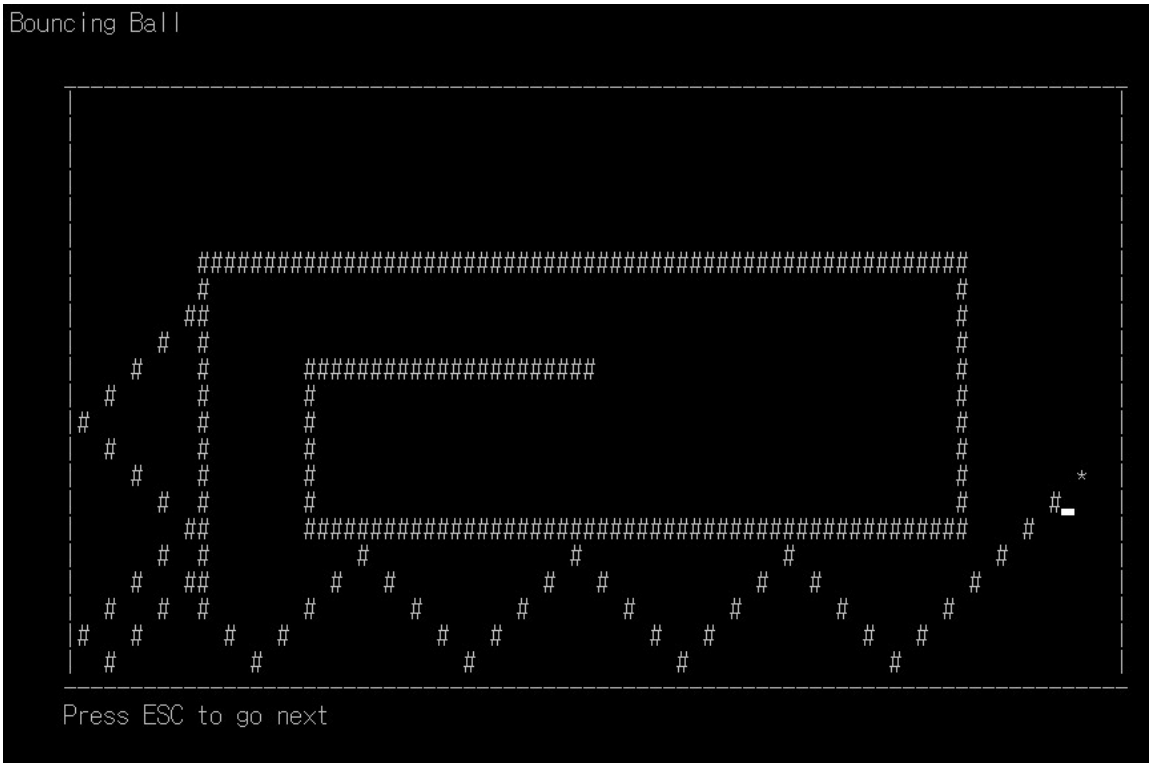
- Draw Map



- Draw Map



- Bouncing Ball



- After Clear Screen

6. Discussion

It was a meaningful time to think about the class. However, the program I created is related to the output of the monitor. So only one object was available and efficiency did not increase. But if you inherit it and add new features (with two balls), I think you can take full advantage of object oriented program.