Box Head
게임 개발 문서

김태우

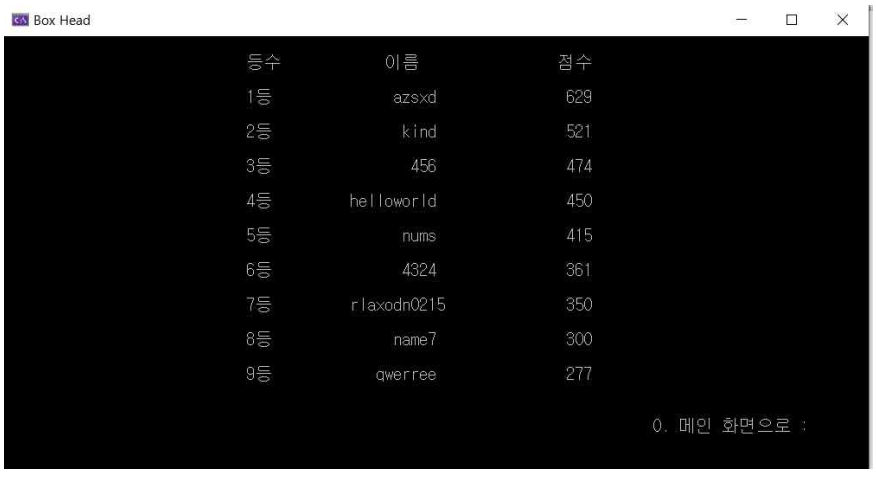
목차

1. 작품설명

“Box Head”는 한명의 플레이어가 수많은 좀비들로부터 생존하는 게임입니다. 플레이어는 권총을 가지고 좀비들을 죽일 수 있습니다. 플레이어와 좀비들은 map에서 랜덤으로 소환되고, 총알 보급 또한 랜덤으로 생성됩니다. 플레이어는 오래 생존할수록, 좀비들을 많이 죽일수록 점수가 높아집니다. 좀비들을 모두 죽이면 플레이어가 최종 승리하게 됩니다. 이 게임은 “Boxhead 2Play Rooms”를 모작으로 제작하였습니다.

<그림 1> 시작 화면

시작 화면은 <그림 1>과 같이 구성하였습니다. 번호를 입력하여 시작(Start), 랭킹 확인(Rank), 나가기(Exit)를 실행할 수 있습니다.



<그림 2> 2.Rank 화면

플레이 화면은 아래 <그림 3>과 같습니다.

<그림 3> 플레이 화면

플레이 화면에는 map과 플레이어의 상태를 나타내는 UI가 있습니다. 플레어이어(P)와 좀비(Z)는 map에서 랜덤으로 소환됩니다.

플레이어는 화살표로 움직일 수 있고, 스페이스바로 공격을 합니다. 공격을 할 시 총알이 나가고 총알(Ammo)이 하나씩 감소합니다. 총알은 일정 사거리가 지나면 사라지고, 일정시간동안 움직이지 않으면 체력이 증가합니다.

좀비는 map에 랜덤으로 소환됩니다. 일정 수가 넘으면 소환되지 않습니다. 좀비는 시야 범위가 있어 플레이어와 일정 거리 떨어져 있으면 랜덤으로 움직입니다. 하지만 플레이어가 좀비의 시야 안에 있으면 좀비는 플레이어를 추격하고 공격 범위에 들어왔을 때 공격하게 됩니다.

총알 보급(A)은 map에 랜덤으로 소환되어 플레이어에게 총알을 공급합니다.

<그림 4> 게임오버 창

플레이어의 체력이 0이 될 경우 <그림 4>와 같이 게임오버가 됩니다. 그 이후 이름을 적는 칸이 나옵니다. 이름을 적으면 기존 기록의 가장 작은 기록보다 크게 되면 Rank창에 저장됩니다. 좀비를 모두 죽인 경우 “GAME OVER” 대신 “YOU WIN”이 나옵니다.

2. 코드설명

1) Map

```
#pragma once

#define MAP_COLS (60)
#define MAP_LINES (20)

char map_0[20][60] = {
        {"111111111111111111111111111111111111111111111111111111111111"},
        {"100000000000000000000000000000000000000000000000000000000001"},
        {"100000000000000000000000000000000000000000000000000000000001"},
        {"100000111111100000000000000000000000000000011111000000000001"},
        {"100000100000000000000000000000000000000000000000000000000001"},
        {"100000100000000000000000000000000000000000000000000000000001"},
        {"100000000000000000000000111000000000000000000000000000000001"},
        {"100000000000000000000000111000000000000000000000000000000001"},
        {"100000000000000000000000111000000000000000000000000000000001"},
        {"100000000000000000000000000000000000000000000010000000000001"},
        {"100000000000000000000000000000000000000000000010000000000001"},
        {"100000000000000000000000000000000000110000000000000000000001"},
        {"100000000000000000000000000000000000110000000000000000000001"},
        {"100000000000011111100000000000000000000000000000000000000001"},
        {"100000000000011111100000000000000000000000000000000000000001"},
        {"100000000000000000000000000000000000000000000000011000000001"},
        {"100000000000000000000000000000000000000000000000011000000001"},
        {"100000000000000000000000000000000000000000000000000000000001"},
        {"100000000000000000000000000000000000000000000000000000000001"},
        {"111111111111111111111111111111111111111111111111111111111111"},

};
```

<그림 5> Map.h

<그림 5>와 같이 Map의 요소를 배열로 표현합니다. 0은 빈 공간, 1은 벽(#), Z는 좀비, P는 플레이어, A는 총알 보급으로 배열 변수로 만듭니다. 이 배열 요소를 바꾸어 Map을 수정할 수 있습니다. 또한 좀비, 플레이어, 총알 보급의 위치를 표시함으로써 함수 구현과 로직을 구성하는데 편리하게 만들었습니다.

2) GameSystem

(1) Gamesystem.h

Code Blame 18 lines (15 loc) · 401 Bytes

```
1   #pragma once
2
3   #define _CRT_SECURE_NO_WARNINGS
4   #include<stdio.h>
5   #include<stdlib.h>
6   #include<Windows.h>
7   #include<string.h>
8   #include<conio.h>
9   #include<time.h>
10  #include"Player.h"
11
12  void GotoXY(int x, int y);
13  void StartScene();
14  void DrawMap(char(*map)[60]);
15  void SpawnXY(int* posX, int*posY, char(*map)[60]);
16  void SpawnAmmo(char(*map)[60]);
17  void GameResult(Player* player, int victory);
18
```

<그림 6> GameSystem.h

GameSystem.h에서는 컴파일러 include 디렉터리와 Player.h을 가져옵니다. 그리고 필요한 함수들의 헤드들을 작성하였습니다.

(2) Gamesystem.c

```
1   #include"GameSystem.h"
2
3   void GotoXY(int x, int y)
4   {
5           COORD pos = { x,y };
6           SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), pos);
7   }
8
```

<그림 7> GotoXY() 함수

GotoXY()함수는 콘솔창의 커서를 옮기는 함수로, 커서를 옮길 좌표 값 x, y를 매개변수로 받습니다. 그리고 COORD 구조체를 선언하여 매개변수의 값을 받고 해당 좌표에 해당하는 콘솔위치로 커서를 이동시킵니다.

```
9   void StartScene()
10  {
11          system("title Box Head"); //콘솔창 제목
12          system("mode con cols=100 lines=25"); //콘솔 창 크기 가로 :100, 세로 : 25
13
14          CONSOLE_CURSOR_INFO consoleCursorInfo;
15
16          consoleCursorInfo.bVisible = 0; //콘솔 창에 커서 안보이게 하기
17          consoleCursorInfo.dwSize = 1;
18
19          SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &consoleCursorInfo);
20
```

<그림 8> StartScene() 함수-1

StartScene()함수는 시작 창을 만드는 함수입니다. 먼저 콘솔창의 제목을 “Box Head”로 바꾸고 콘솔창의 크기를 가로 100, 세로 25줄로 만듭니다. 그리고 커서를 안보이게 만듭니다.

```
20
21          int n=0;
22          while (n!=1)
23          {
24                  system("cls");
25
26                  printf("\n");
27                  printf("                  ######          ####         #    #                  \n");
28                  printf("                  #     #        #    #         #  #                   \n");
29                  printf("                  ######         #    #           ##                    \n");
30                  printf("                  #     #        #    #         #  #                   \n");
31                  printf("                  ######          ####         #    #                  \n");
32                  printf("                                                                         \n");
33                  printf("          #     #       ######           #           #####             \n");
34                  printf("          #     #       #              # #           #    #            \n");
35                  printf("          #######       ######         #####         #     #           \n");
36                  printf("          #     #       #            #     #         #    #            \n");
37                  printf("          #     #       ######       #      #        #####             \n\n");
38
39                  GotoXY(35,14);
40                  printf("1. Start");
41                  GotoXY(35, 16);
42                  printf("2. Rank");
43                  GotoXY(35, 18);
44                  printf("3. Exit");
45
46                  GotoXY(30, 20);
47                  printf("숫자를 눌러주세요 : ");
48                  scanf("%d", &n);
49                  getchar();
```

<그림 9> StartScene() 함수-2

먼저 화면을 지운 다음에 게임 제목 “Box Head”를 띄웁니다. 그리고 GotoXY()함수를 이용하여 커서를 이동하고, 선택 창을 출력합니다.
(1.Start, 2.Rank, 3.Exit) 이때 변수 n을 이용하여 입력받는 번호를 저장하고 버퍼에 남아있는 개행 문자를 제거하기 위해 getchar()를 호출합니다. 플레이어가 1.Start를 시행할 때 까지 반복해야 하므로 While()안에 넣습니다.

```
50              switch (n)
51              {
52              case 1:
53                      system("cls");
54                      break;
55              case 2:
56              {
57                      system("cls");
58                      char name[25];
59                      int point;
60
61                      FILE* rank;
62                      rank = fopen("Rank.txt", "r");
63                      if (rank == EOF) return 1;
64
65                      GotoXY(28, 1);
66                      printf("등수\t    이름\t\t점수\n");
67
68                      for (int i = 0; i < 9; i++)
69                      {
70                              int res = fscanf(rank, "%s%d", name, &point);
71                              if (res == EOF) break;
72
73                              GotoXY(28, 3+2*i);
74                              printf("%d등\t\t%10s\t\t%4d\n",i+1, name, point);
75                      }
76
77                      int a=1;
78                      GotoXY(75, 22);
79                      printf("0. 메인 화면으로 : ");
80                      while (a)
81                      {
82                              scanf("%d", &a);
83                      }
84                      getchar();
85                      fclose(rank);
86
87              }
88                      break;
89              case 3:
90                      exit(1);
91                      break;
92              default:
```

<그림 10> StartScene() 함수-3

입력 받은 n의 값을 이용하여 switch문으로 선택 창을 작동시킵니다.
n==1인 경우 화면 창을 지우고 반복문을 빠져나가 게임을 실행합니다.
n==2인 경우 Rank.txt파일에서 기록들을 가져옵니다. 변수 name과 point에

기록을 저장하고 한 줄씩 출력하는 방식으로 Rank창을 나타냅니다. Rank창의 0번을 입력할 경우 다시 시작 창으로 돌아옵니다.
n==3인 경우 exit(1)를 이용하여 게임을 종료합니다.

```
99  void DrawMap(char (*map)[60])
100 {
101         for (int i = 0; i < 20; i++)
102         {
103                 for (int j = 0; j < 60; j++)
104                 {
105                         if (map[i][j] == '1')
106                         {
107                                 GotoXY(j, i);
108                                 printf("#");
109                         }
110                 }
111         }
112
113 }
```

<그림 11> DrawMap()함수

```
115 void SpawnXY(int* posX, int* posY, char(*map)[60])
116 {
117         int X;
118         int Y;
119
120         while (1)
121         {
122                 X = rand() % 60;
123                 Y = rand() % 20;
124
125                 if (map[Y][X] == '0')
126                         break;
127         }
128
129         *posX = X;
130         *posY = Y;
131 }
```

<그림 12> SpawnXY()함수

DrawMap()함수는 게임 맵을 그리는 함수로 게임 맵 배열 함수를 인수로 받습니다. 그리고 반복문을 통해 배열의 값이 1인 경우 커서를 해당 좌표로 옮기고 '#'을 출력합니다.

SpawnXY()함수는 플레이어와 좀비, 탄약 보급을 랜덤하게 소환해주는 함수로, 좌표값 x,y와 게임 맵 배열을 인수로 받습니다. x,y에 rand()를 이용하여 배열의 가로와 세로의 값을 입으로 받습니다. 이때 빈 공간에서만 소환이 되어야 함으로 배열에서 0인 구간에만 소환합니다. 만약 임의의 소한된 값이 0이 아닌 경우 while()를 통해 0인 경우에만 나오도록 반복합니다.

```
142 void GameResult(Player* player, int victory)
143 {
144         if (victory == 0)
145         {
146                 GotoXY(61, 1);
147                 printf("HP : %3d", 0);
148                 GotoXY(70, 11);
149                 printf("G A M E  O V E R");
150         }
151
152         else if(victory==1)
153         {
154                 GotoXY(70, 11);
155                 printf("Y O U  W I N ! !");
156         }
157
158         // 플레이어 이름 입력
159         GotoXY(61, 14);
160         printf("이름을 입력하세요 : ");
161         fgets(player->name, sizeof(player->name), stdin);
162
```

```
163         FILE *rank;
164
165         rank = fopen("Rank.txt", "r");
166         if (rank == NULL) return 1;
167
168         typedef struct
169         {
170                 char name[25];
171                 int points;
172                 struct Rank *nextRank;
173
174         }Rank;
175
176         Rank R[10]={"",0,NULL};
177
178         //최근 데이터는 0번째에 저장
179         strcpy(R[0].name, player->name);
180         R[0].name[strlen(R[0].name) - 1] = '\0';
181         R[0].points = player->points;
182         R[0].nextRank = &R[1];
```

```
183
184         for(int i=1;i<10;i++) //11등까지 저장
185         {
186                 char str[25] = { 0, };
187                 int point;
188
189                 int res = fscanf(rank, "%s%d", str, &point);
190                 if (res == EOF) break;
191
192                 strcpy(R[i].name, str);
193                 R[i].points = point;
194                 R[i].nextRank = &R[i+1];
195                 if (&R[i + 1] == NULL) break;
196         }
197
198         fclose(rank);
199
200
201         //내림차순 정렬
202
```

```
203         for (int i = 0; i < 9; i++)
204         {
205                 for (int j = i+1; j < 10; j++)
206                 {
207                         if (R[i].points < R[j].points)
208                         {
209                                 char temp[25];
210                                 strcpy(temp, R[i].name);
211                                 int tempNum = R[i].points;
212
213                                 strcpy(R[i].name, R[j].name);
214                                 R[i].points = R[j].points;
215
216                                 strcpy(R[j].name, temp);
217                                 R[j].points = tempNum;
218                         }
219                 }
220         }
221
222         //Rank.txt에 저장 (10위까지만)
223         rank = fopen("Rank.txt", "w");
224         if(rank==NULL) return 1;
225
226         for (int i = 0; i < 9; i++)
227         {
228                 if (!(strcmp(R[i].name, ""))) break;
229                 fprintf(rank, "%s\t\t%d\n", R[i].name, R[i].points);
230         }
231
232         fclose(rank);
233
```

<그림 13,14,15,16> GameResult()함수

GameResult() 함수는 플레이가 종료되고 출력되는 함수이다. 플레이어 구조체와 승리여부 변수를 입력받는다. 승리변수 여부에 따라 출력창이 변한다.
그리고 플레이어의 이름을 입력하는 창이 나와 플레이어의 이름을 입력받으면 플레이어 구조체 변수의 이름에 저장된다. 그리고 Rank구조체를 만들어 플레이어의 이름과 점수를 최근 것과 Rank.txt에 기록된 기록들을 저장한다. Rank는 9등까지만 필요함으로 총 10개의 구조체를 선언하고 내용들을 기록합니다. 그리고 내림차순으로 구조체들을 정렬하고 Rank.txt에 다시 기록합니다.

```
133     void SpawnAmmo(char(*map)[60])
134     {
135             int x, y;
136             SpawnXY(&x, &y, map);
137             map[y][x] = 'A';
138             GotoXY(x, y);
139             printf("A");
140     }
141
```

<그림 17> SpawnAmmo()함수

SpawnAmmo()함수는 탄약 보급을 소환시키는 함수입니다. 맵 행렬을 인수로 받고 SpawnXY()함수에서 받은 x,y값에 커서를 이동시키고 'A'를 출력 및 맵 행렬의 값 또한 변경시킵니다.

3) Player

(1) Player.h

```
1   #pragma once
2   #include"Weapon.h"
3
4   typedef struct
5   {
6           char name[25];
7           int Hp;
8           int posX;
9           int posY;
10          int points;
11          struct Weapon *weapon;
12  }Player;
13
14  void PlayerBeacon(int posX, int posY, char(*map)[60]);
15  void ShowHP(int posX, int posY, Player* player);
16  void ShowAmmo(int posX, int posY, Weapon* weapon);
17  void Score(int posX, int posY, Player* player);
18  void PlayerMove(Player* player,Weapon *weapon, char(*map)[60], int* HpTimer);
19  void PlayerAttack(Player* player, Weapon* weapon, char(*map)[60]);
20
```

<그림 18> Player.h

Player.h에는 플레이어 구조체와 플레이어에게 필요한 함수가 선언되어 있습니다. 먼저 Weapon.h를 include합니다. Player구조체에는 플레이어 이름, 체력, X,Y 좌표값, 점수, 그리고 무기가 선언되어 있습니다.

```
1   #include"Player.h"
2
3   void PlayerBeacon(int posX, int posY, char(*map)[60])
4   {
5           GotoXY(posX, posY);
6           map[posY][posX] = 'P';
7           printf("P");
8   }
```

```
201 void ShowHP(int posX, int posY, Player* player)
202 {
203         GotoXY(posX, posY);
204         printf("HP : %3d", player->Hp);
205 }
```

```
206  void ShowAmmo(int posX, int posY, Weapon* weapon)
207  {
208         if (weapon->ammo < 0)
209         {
210                 GotoXY(posX, posY);
211                 printf("Ammo :   0");
212         }
213         else
214         {
215                 GotoXY(posX, posY);
216                 printf("Ammo : %3d", weapon->ammo);
217         }
218  }
```

```
219   void Score(int posX, int posY, Player *player)
220   {
221           GotoXY(posX, posY);
222           printf("Score : %4d", player->points);
223   }
```

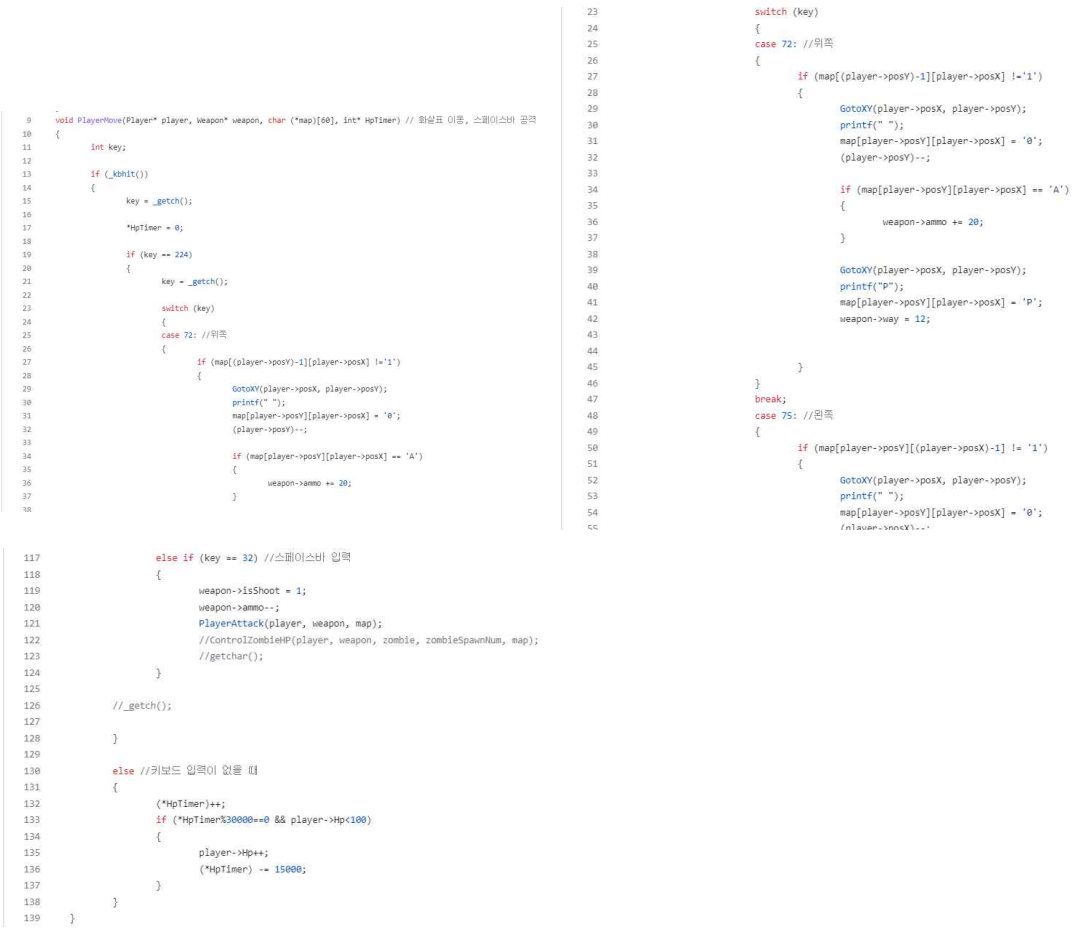
<그림 19,20,21,22> PlayerBeacon(), Score(), ShowHP(), ShowAmmo() 함수

PlayerBeacon()함수는 플레이어를 맵에 소환시키는 함수로 플레이어의 위치 값과 맵 행렬을 받습니다. 해당 좌표에 커서를 이동시키고 배열 및 맵에 “P”를 출력합니다.

Score()함수는 점수 창을 나타내는 함수로 창을 표시할 X, Y 좌표 값과 플레이어 구조체를 인수로 받습니다. 해당 좌표에 커서를 이동시키고 플레이어의 점수를 출력합니다.

ShowHP()함수는 플레이어의 체력을 나타내는 함수로 창을 표시할 X, Y 좌표 값과 플레이어 구조체를 인수로 받습니다. 해당 좌표에 커서를 이동시키고 플레이어의 체력를 출력합니다.

ShowAmmo() 함수는 플레이어가 가지고 있는 총알을 나타내는 함수로 창을 표시할 X, Y 좌표 값과 플레이어 무기 구조체를 인수로 받습니다. 해당 좌표에 커서를 이동시키고 플레이어무기의 총알을 출력합니다. 이때 총알이 0이하이면 0으로만 출력하게 만듭니다.



```
void PlayerMove(Player* player, Weapon* weapon, char (*map)[60], int* HpTimer) // 화살표 이동, 스페이스바 공격
{
	int key;

	if (_kbhit())
	{
		key = _getch();

		*HpTimer = 0;

		if (key == 224)
		{
			key = _getch();

			switch (key)
			{
			case 72: //위쪽
			{
				if (map[(player->posY)-1][player->posX] !='1')
				{
					GotoXY(player->posX, player->posY);
					printf(" ");
					map[player->posY][player->posX] = '0';
					(player->posY)--;

					if (map[player->posY][player->posX] == 'A')
					{
						weapon->ammo += 20;
					}
```

```
			switch (key)
			{
			case 72: //위쪽
			{
				if (map[(player->posY)-1][player->posX] !='1')
				{
					GotoXY(player->posX, player->posY);
					printf(" ");
					map[player->posY][player->posX] = '0';
					(player->posY)--;

					if (map[player->posY][player->posX] == 'A')
					{
						weapon->ammo += 20;
					}

					GotoXY(player->posX, player->posY);
					printf("P");
					map[player->posY][player->posX] = 'P';
					weapon->way = 12;


				}
			}
			break;
			case 75: //왼쪽
			{
				if (map[player->posY][(player->posX)-1] != '1')
				{
					GotoXY(player->posX, player->posY);
					printf(" ");
					map[player->posY][player->posX] = '0';
```

```
		else if (key == 32) //스페이스바 입력
		{
			weapon->isShoot = 1;
			weapon->ammo--;
			PlayerAttack(player, weapon, map);
			//ControlZombieHP(player, weapon, zombie, zombieSpawnNum, map);
			//getchar();
		}

		//_getch();

		}

		else //키보드 입력이 없을 때
		{
			(*HpTimer)++;
			if (*HpTimer%30000==0 && player->Hp<100)
			{
				player->Hp++;
				(*HpTimer) -= 15000;
			}
		}
	}
```

<그림 23,24,25> PlayerMove()함수

PlayerMove()함수는 플레이어의 이동, 공격, 정지 시 체력회복을 담당하고 있습니다. 이 함수의 인수는 플레이어, 무기 구조체, 맵 행렬, 체력 타이머를 가지고 있습니다. 키보드로부터 화살표의 입력을 받으면 switch문으로 그 방향에 맞게 이동합니다. 예를 들어 위로 이동할 때 이동전 플레이어의 위치를 콘솔창과 맵 행렬에 지우고 위로 한 칸인 위치에 플레이어를 출력 및 행렬에 입력합니다. 이때 그 위치에 맵 행렬을 근거로 하여 탄약 보급(A)이 있을 때 총알의 수를 20 증가시킵니다. 이와 동일한 방식으로 아래, 좌 ,우 동일한 방식으로 코딩합니다. 그리고 스페이스바를 누를 때 무기가 사용되었음을 알리고, 총알을 하나 줄입니다. 그리고 PlayerAttack()함수를 실행합니다. 키보드 입력이 없으면 타이머 수를 증가시키고 플레이어 체력이 100미만이고 일정 숫자 이상이면 체력이 1씩 증가합니다. 이때 타이머의 수를 0이 아닌 일정 상수로 하여 체력 상승속도를 조절합니다.

```
140     void PlayerAttack(Player* player, Weapon* weapon, char (*map)[60])
141     {
142             if (weapon->ammo >= 0)
143             {
144                     int posX = player->posX;
145                     int posY = player->posY;
146                     int i = 1;
147                     if (weapon->way == 12)// 위로 사격
148                     {
149                             while (i < weapon->range && map[posY - i][posX] == '0')
150                             {
151                                     GotoXY(posX, posY - i);
152                                     printf("o");
153                                     Sleep(50);
154                                     GotoXY(posX, posY - i);
155                                     printf(" ");
156                                     i++;
157                             }
158                     }
159
160                     else if (weapon->way == 3)// 오른쪽 사격
161                     {
162                             while (i < weapon->range * 2 && map[posY][posX + i] == '0')
163                             {
164                                     GotoXY(posX + i, posY);
165                                     printf("o");
166                                     Sleep(25);
167                                     GotoXY(posX + i, posY);
168                                     printf(" ");
169                                     i++;
170                             }
171                     }
```

<그림 26> PlayerAttack()함수

PlayerAttack()함수는 플레이어의 공격 애니메이션을 담당하는 함수입니다. 총알의 수가 1 이상일 때 플레이어의 현재 위치 좌표를 받습니다. 그리고 플레이어 무기의 구조체를 통해 사격 방향을 알아냅니다.(12시, 3시, 6시, 9시)
변수 I는 총알이 이동한 거리를 나타냅니다. 즉 사거리 안에 있고 앞에 빈 공간일 때 반복문을 통해 “O”를 출력 및 지움으로써 총알이 이동하는 애니메이션을 만듭니다. 이때 가로의 길이는 세로의 길이보다 짧기 때문에 사거리의 2배로 계산합니다. Sleep()을 통하여 애니메이션 속도를 조절합니다.

4) Weapon

(1) Weapon.h/Weapon.c

```h
#pragma once

typedef struct
{
        char name[20];
        int damage;
        int range;
        int ammo;
        int way; //»ç°Ý ¹æÇâ :12½Ã, 3½Ã, 6½Ã, 9½Ã
        int isShoot;

}Weapon;

void WeaponName(int posX, int posY, Weapon* weapon);
```

```c
#include"Weapon.h"

void WeaponName(int posX, int posY, Weapon* weapon)
{
        GotoXY(posX, posY);

         printf("Weapon : %s", weapon->name);
}
```

<그림 27,28> Weapon.h/WeaponName()

Weapon.h에서 무기의 구조체를 생성한다. 구조체 속성으로는 이름, 데미지, 사거리, 총알, 사격방향(시계 방향: 12, 3, 6, 9), 사격유무가 있다.

WeaponName()함수는 무기의 이름을 출력함수로 좌표 변수 posX, posY를 받아 커서를 해당 위치로 이동시킵니다. 그리고 weapon구조체를 받아 무기의 이름을 출력합니다.

5) Zombie

(1) Zombie.h

```h
#pragma once
#include "Player.h"
#include"Weapon.h"

typedef struct
{
        int Hp;
        int posX;
        int posY;
        int damage;
        int points;
        int zombieDead;
        int sight;

}Zombie;

void ZombieBeacon(int posX,int posY, char(*map)[60], Zombie*zombie);
void ZombieMove(Player *player, Zombie* zombie, int zombieNum, char(*map)[60]);
void ZombieAttack(Player* player, Zombie* zombie, int zombieNum);
void ControlZombieHP(Player* player, Weapon* weapon, Zombie* zombie, int zombies, char(*map)[60], int* victory);
void ZombieDamaged(Player* player, Weapon* weapon, Zombie* zombie, int zombieNum, char(*map)[60]);
void ZombieDead(Zombie* zombie, int zombieNum, Player* player, char(*map)[60]);
```

<그림 29> Zombie.h

Zombie.h에서는 좀비 구조체와 좀비 구현에 필요한 함수들의 헤드를 정리했습니다. 좀비 구조체에서는 체력, XY좌표값, 공격력, 죽일시 획득점수, 좀비의 생사유무, 좀비의 시야가 있습니다.

```
void ZombieMove(Player *player, Zombie *zombie,int zombieNum, char(*map)[60])
{
	//좀비가 안 죽었을때
	if (zombie[zombieNum].zombieDead==0)
	{
		int existPlayer = 0;

		//시야에서 플레이어 확인
		for (int i = (zombie[zombieNum].posY - zombie->sight); i < (zombie[zombieNum].posY + zombie->sight); i++)
		{
			for (int j = (zombie[zombieNum].posX - zombie->sight*2); j < (zombie[zombieNum].posX + zombie->sight*2); j++)
			{
				if (map[i][j] == 'P')
				{
					existPlayer = 1;
				}
			}
		}

		//시야에 있으면
		if (existPlayer==1)
		{
		//플레이어 추격 및 공격
		//플레이어 공격 -> 거리 1 주변에 플레이어가 있으면
			if (map[zombie[zombieNum].posY][zombie[zombieNum].posX + 1] == 'P' || map[zombie[zombieNum].posY + 1][zombie[zombieNum].posX + 1] == 'P' ||
				map[zombie[zombieNum].posY + 1][zombie[zombieNum].posX] == 'P' || map[zombie[zombieNum].posY + 1][zombie[zombieNum].posX - 1] == 'P' ||
				map[zombie[zombieNum].posY][zombie[zombieNum].posX - 1] == 'P' || map[zombie[zombieNum].posY - 1][zombie[zombieNum].posX - 1] == 'P' ||
				map[zombie[zombieNum].posY - 1][zombie[zombieNum].posX] == 'P' || map[zombie[zombieNum].posY - 1][zombie[zombieNum].posX + 1] == 'P')
			{
				ZombieAttack(player, zombie, zombieNum);
			}

			if (map[zombie[zombieNum].posY][zombie[zombieNum].posX + 1] != '1'
				&& map[zombie[zombieNum].posY][zombie[zombieNum].posX + 1] != 'P') // 오른쪽에 벽이 없거나 플레이어가 없을때
			{
				if ((player->posX) - (zombie[zombieNum].posX) > 0) //플레이어가 오른쪽에 있을때
				{

					//오른쪽 가는 로직 만들기
					GotoXY(zombie[zombieNum].posX, zombie[zombieNum].posY);
					printf(" ");
					map[zombie[zombieNum].posY][zombie[zombieNum].posX] = '0';
					(zombie[zombieNum].posX)++;
					GotoXY(zombie[zombieNum].posX, zombie[zombieNum].posY);
					printf("Z");
					map[zombie[zombieNum].posY][zombie[zombieNum].posX] = 'Z';

				  if (map[zombie[zombieNum].posY + 1][zombie[zombieNum].posX] != '1'
					  && map[zombie[zombieNum].posY + 1][zombie[zombieNum].posX] != 'P') // 아래쪽에 벽이 없거나 플레이어가 없을 때
				  {

					  if ((player->posX) - (zombie[zombieNum].posX) > 0) //플레이어가 아래쪽에 있을때
					  {
						  //아래쪽 가는 로직 만들기
						  //오른쪽 아래 대각선
						  GotoXY(zombie[zombieNum].posX, zombie[zombieNum].posY);
						  printf(" ");
						  map[zombie[zombieNum].posY][zombie[zombieNum].posX] = '0';
						  (zombie[zombieNum].posY)++;
						  GotoXY(zombie[zombieNum].posX, zombie[zombieNum].posY);
						  printf("Z");
						  map[zombie[zombieNum].posY][zombie[zombieNum].posX] = 'Z';

					  }
				  }

				  else if (map[zombie[zombieNum].posY - 1][zombie[zombieNum].posX] != '1'
					  && map[zombie[zombieNum].posY - 1][zombie[zombieNum].posX] != 'P') // 위쪽에 벽이 없거나 플레이어가 없을때
				  {

					  if ((player->posY) - (zombie[zombieNum].posY) < 0) //플레이어가 위쪽에 있을때
					  {
						  //위로 가는 로직 만들기
						  //오른쪽 위 대각선
						  GotoXY(zombie[zombieNum].posX, zombie[zombieNum].posY);
						  printf(" ");
						  map[zombie[zombieNum].posY][zombie[zombieNum].posX] = '0';
						  (zombie[zombieNum].posY)--;
						  GotoXY(zombie[zombieNum].posX, zombie[zombieNum].posY);
						  printf("Z");
						  map[zombie[zombieNum].posY][zombie[zombieNum].posX] = 'Z';
```

```
96                  else  //오른쪽에 벽이 있으면
97                  {
98                          if (map[zombie[zombieNum].posY + 1][zombie[zombieNum].posX] != '1'
99                                  && map[zombie[zombieNum].posY + 1][zombie[zombieNum].posX] != 'P') // 아래쪽에 벽이 없거나 플레이어가 없을 때
100                         {
101
102                                 if ((player->posX) - (zombie[zombieNum].posX) > 0) //플레이어가 아래쪽에 있을때
103                                 {
104                                         //아래쪽 가는 로직 만들기
105                                         //오른쪽 아래 대각선
106                                         GotoXY(zombie[zombieNum].posX, zombie[zombieNum].posY);
107                                         printf(" ");
108                                         map[zombie[zombieNum].posY][zombie[zombieNum].posX] = '0';
109                                         (zombie[zombieNum].posY)++;
110                                         GotoXY(zombie[zombieNum].posX, zombie[zombieNum].posY);
111                                         printf("Z");
112                                         map[zombie[zombieNum].posY][zombie[zombieNum].posX] = 'Z';
113
114                                 }
115                         }
116
117                         else if (map[zombie[zombieNum].posY - 1][zombie[zombieNum].posX] != '1'
118                                 && map[zombie[zombieNum].posY - 1][zombie[zombieNum].posX] != 'P') // 위쪽에 벽이 없거나 플레이어가 때
119                         {
120
121                                 if ((player->posY) - (zombie[zombieNum].posY) < 0) //플레이어가 위쪽에 있을때
122                                 {
123                                         //위로 가는 로직 만들기
124                                         //오른쪽 위 대각선
125                                         GotoXY(zombie[zombieNum].posX, zombie[zombieNum].posY);
126                                         printf(" ");
127                                         map[zombie[zombieNum].posY][zombie[zombieNum].posX] = '0';
```

```
406             //시야에서 벗어나면
407             else
408             {
409                     //플레이어 추격하지 않기 => 랜덤 이동
410                     int randomNum = rand() % 4;
411
412                     switch (randomNum)
413                     {
414                     case 0:
415                     {
416                             if (map[zombie[zombieNum].posY][zombie[zombieNum].posX + 1] != '1') //오른쪽에 벽이 없을 때
417                             {
418                                     //오른쪽 가는 로직 만들기
419                                     GotoXY(zombie[zombieNum].posX, zombie[zombieNum].posY);
420                                     printf(" ");
421                                     map[zombie[zombieNum].posY][zombie[zombieNum].posX] = '0';
422                                     (zombie[zombieNum].posX)++;
423                                     GotoXY(zombie[zombieNum].posX, zombie[zombieNum].posY);
424                                     printf("Z");
425                                     map[zombie[zombieNum].posY][zombie[zombieNum].posX] = 'Z';
426                             }
427                     }
428                             break;
429                     case 1:
430                     {
431                             if (map[zombie[zombieNum].posY][zombie[zombieNum].posX - 1] != '1') //왼쪽에 벽이 없을 때
432                             {
433                                     //왼쪽으로 가는 로직 만들기
434                                     GotoXY(zombie[zombieNum].posX, zombie[zombieNum].posY);
435                                     printf(" ");
436                                     map[zombie[zombieNum].posY][zombie[zombieNum].posX] = '0';
437                                     (zombie[zombieNum].posX)--;
438                                     GotoXY(zombie[zombieNum].posX, zombie[zombieNum].posY);
439                                     printf("Z");
```

<그림 30> ZombieMove()함수

ZombieMove()함수는 좀비의 이동AI를 담당하는 함수입니다. 먼저 좀비의 생사 유무를 확인하고 살아있을 경우 함수를 실행합니다. 그리고 플레이어가 좀비의 시야에 있는지 확인입니다. 시야에 있을 경우 좀비가 공격 사거리 안에 있을 때 ZombieAttack()함수를 통해 플레이어의 체력을 깎습니다. 만일 공격 사거리 안에 없으면 플레이어의 위치와 벽의 유무에 따라 좀비가 위, 아래, 좌, 우 뿐만 아니라 대각선도 이동합니다.(자세한 내용은 부록의 순서도를 참고 해주세요.) 만일 시야에 플레이어가 없으면 랜덤으로 이동하는데, switch문을 이용하여 해당 방향에 벽이 없는 경우 이동하게 됩니다.

```
1    #include"Zombie.h"
2
3    void ZombieBeacon(int posX, int posY, char(*map)[60],Zombie *zombie)
4    {
5            GotoXY(posX, posY);
6            map[posY][posX] = 'Z';
7            printf("Z");
8    }
```

```
480    void ZombieAttack(Player *player, Zombie *zombie, int zombieNum)
481    {
482            player->Hp -= zombie[zombieNum].damage;
483    }
```

<그림 31> ZombieBeacon(), ZombieAttack()함수

ZombieBeacon함수는 좀비를 소환시키는 함수로 소환 위치(posX, posY)와 맵 행렬을 인수로 받습니다. 하댕 좌표로 커서를 이동하고 “Z”를 맵 행렬과 콘솔창에 출력합니다.

ZombieAttack함수는 좀비가 공격하는 함수로 호출시 플레이어의 체력을 좀비의 데미지만큼 감소시킵니다.

```
484    void ControlZombieHP(Player*player, Weapon* weapon, Zombie* zombie, int zombieNum, char(*map)[60], int* victory) //플레이어 총 맞는 것도 인식
485    {
486
487            //좀비 전원 사망 확인
488            int count=0;
489            for (int i = 0; i < 50; i++)
490            {
491                    if (zombie[i].zombieDead == 0) break;
492                    count++;
493            }
494            if (count == 50) *victory = 1;
```

```
496
497         //총알이 1개 감소할 때 플레이어 사격 인식
498         if(weapon->isShoot==1)
499         {
500                 //사격 방향
501                 switch (weapon->way)
502                 {
503                 case 12:
504                 {
505                         int closetNum=-1;
506
507                         for (int i = 0; i < zombieNum; i++)
508                         {
509                                 if (zombie[i].zombieDead == 1) continue; //죽은 좀비는 계산할 필요 없다.
510
511                                 if ((zombie[i].posX == player->posX) && ((player->posY) - zombie[i].posY <= weapon->range) && ((player->posY) - zombie[i].posY > 0))
512                                 {
513                                         //사거리 안에 있는데 2마리가 같은 라인에 있는 경우 -> 비교해서 플레이어랑 가장 가까운 좀비 데미지
514                                         if (closetNum == -1) closetNum = i;
515                                         closetNum = (player->posY - zombie[closetNum].posY <= player->posY - zombie[i].posY) ? closetNum : i;
516                                 }
517                         }
518
519                         if (closetNum != -1)
520                         ZombieDamaged(player, weapon, zombie, closetNum, map);
521
522                 }
523                         break;
```

<그림 32> ControlZombieHP()함수

ControlZombieHP()함수는 소환된 좀비의 전체 체력을 관리하는 함수입니다. 먼저 좀비가 전부 사망하였는지 확인합니다. 좀비가 전부 죽었으면 승리를 나타내는 victory변수 값을 1로 만듭니다. 그 이후 플레이어가 사격을 하였는지 확인하고, 플레이어의 사격 방향에 따라 그 방향에 있는 좀비들을 확인합니다. 먼저 무기의 사거리 안에 있고 플레이어 사격위치와 맞는 좀비들을 구합니다. 이때 좀비가 2마리 이상이면 플레이어와 가장 가까운 좀비에게만 데미지를 입습니다. 따라서 플레이어와 좀비의 거리를 비교하여 가장 가까운 좀비에게만 데미지를 줍니다(ZombieDamaged()함수 호출), 사격방향은 swich()문을 통하여 구분하여 연산합니다. 연산 이후 사격의 유무는 다시 0으로 바꿉니다.

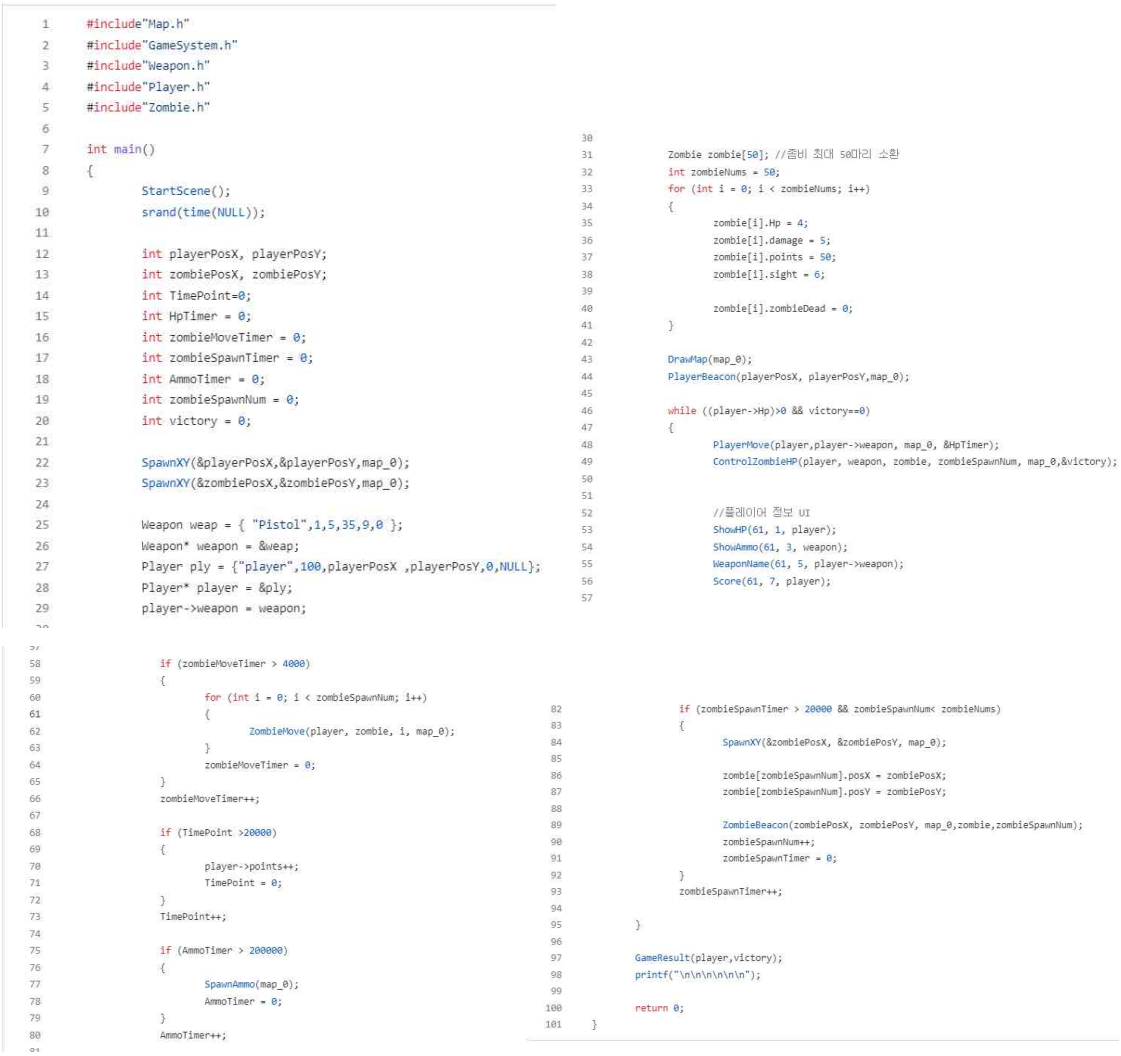
```
595     void ZombieDamaged(Player* player,Weapon* weapon, Zombie* zombie, int zombieNum, char(*map)[60] )
596     {
597             zombie[zombieNum].Hp -= weapon->damage;
598
599             if (zombie[zombieNum].Hp <= 0)
600             {
601                     ZombieDead(zombie,zombieNum,player,map);
602             }
603     }
604     void ZombieDead(Zombie* zombie, int zombieNum, Player* player, char (*map)[60])
605       {
606             GotoXY(zombie[zombieNum].posX, zombie[zombieNum].posY);
607             printf(" ");
608             map[zombie[zombieNum].posY][zombie[zombieNum].posX] = '0';
609
610             //플레이어 포인트 획득
611             if (zombie[zombieNum].zombieDead == 0)
612             {
613                     player->points += zombie[zombieNum].points;
614             }
615
616             zombie[zombieNum].zombieDead = 1;
617
618     }
```

<그림 33> ZombieDamaged(), ZombieDead() 함수

ZombieDamaged()함수는 플레이어의 데미지를 주어 플레이어의 체력을 깎는 함수입니다. Player 구조체에 접근해 체력을 좀비의 공격력만큼 깎습니다. 이때 좀비의 체력이 0이 되면 ZombieDead()함수가 실행됩니다.

ZombieDead()함수는 좀비가 죽는 함수이므로 좀비를 맵 배열과 콘솔창에 지우고 플레이어에게 점수를 줍니다.

6) Main.c

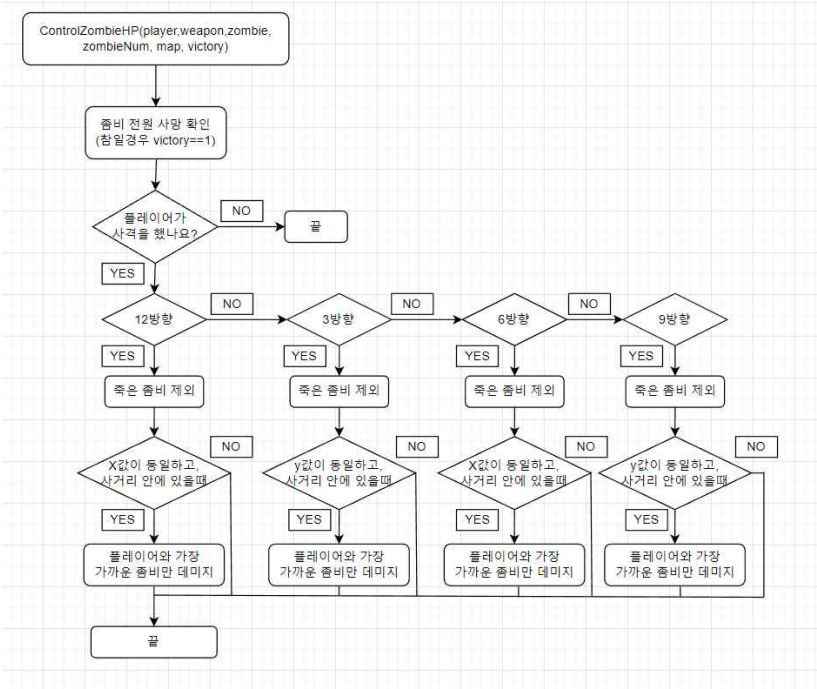


```c
#include"Map.h"
#include"GameSystem.h"
#include"Weapon.h"
#include"Player.h"
#include"Zombie.h"

int main()
{
	StartScene();
	srand(time(NULL));

	int playerPosX, playerPosY;
	int zombiePosX, zombiePosY;
	int TimePoint=0;
	int HpTimer = 0;
	int zombieMoveTimer = 0;
	int zombieSpawnTimer = 0;
	int AmmoTimer = 0;
	int zombieSpawnNum = 0;
	int victory = 0;

	SpawnXY(&playerPosX,&playerPosY,map_0);
	SpawnXY(&zombiePosX,&zombiePosY,map_0);

	Weapon weap = { "Pistol",1,5,35,9,0 };
	Weapon* weapon = &weap;
	Player ply = {"player",100,playerPosX ,playerPosY,0,NULL};
	Player* player = &ply;
	player->weapon = weapon;

	Zombie zombie[50]; //좀비 최대 50마리 소환
	int zombieNums = 50;
	for (int i = 0; i < zombieNums; i++)
	{
		zombie[i].Hp = 4;
		zombie[i].damage = 5;
		zombie[i].points = 50;
		zombie[i].sight = 6;

		zombie[i].zombieDead = 0;
	}

	DrawMap(map_0);
	PlayerBeacon(playerPosX, playerPosY,map_0);

	while ((player->Hp)>0 && victory==0)
	{
		PlayerMove(player,player->weapon, map_0, &HpTimer);
		ControlZombieHP(player, weapon, zombie, zombieSpawnNum, map_0,&victory);


		//플레이어 정보 UI
		ShowHP(61, 1, player);
		ShowAmmo(61, 3, weapon);
		WeaponName(61, 5, player->weapon);
		Score(61, 7, player);


		if (zombieMoveTimer > 4000)
		{
			for (int i = 0; i < zombieSpawnNum; i++)
			{
				ZombieMove(player, zombie, i, map_0);
			}
			zombieMoveTimer = 0;
		}
		zombieMoveTimer++;

		if (TimePoint >20000)
		{
			player->points++;
			TimePoint = 0;
		}
		TimePoint++;

		if (AmmoTimer > 200000)
		{
			SpawnAmmo(map_0);
			AmmoTimer = 0;
		}
		AmmoTimer++;

		if (zombieSpawnTimer > 20000 && zombieSpawnNum< zombieNums)
		{
			SpawnXY(&zombiePosX, &zombiePosY, map_0);

			zombie[zombieSpawnNum].posX = zombiePosX;
			zombie[zombieSpawnNum].posY = zombiePosY;

			ZombieBeacon(zombiePosX, zombiePosY, map_0,zombie,zombieSpawnNum);
			zombieSpawnNum++;
			zombieSpawnTimer = 0;
		}
		zombieSpawnTimer++;

	}

	GameResult(player,victory);
	printf("\n\n\n\n\n\n");

	return 0;
}
```

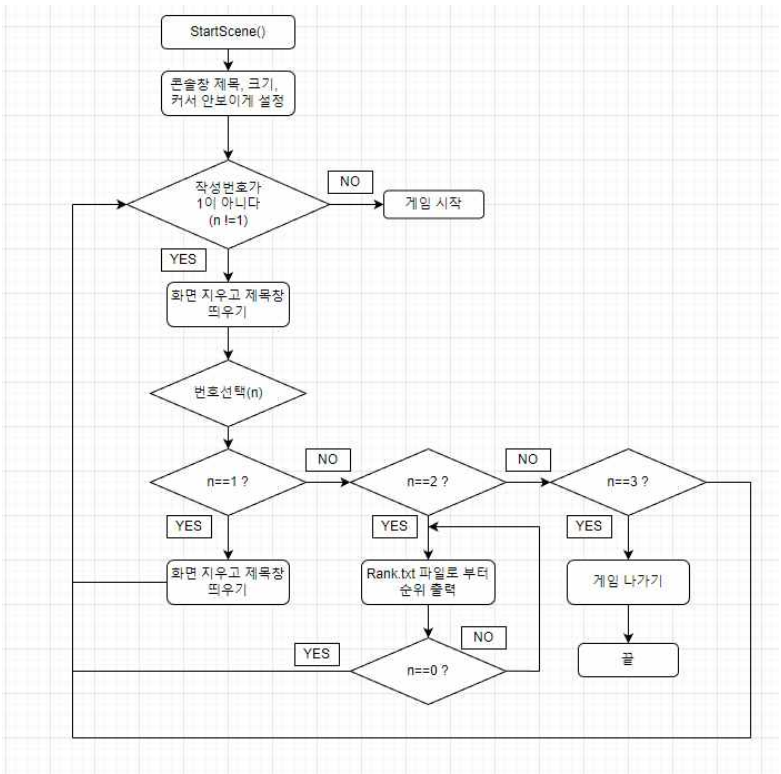
먼저 GameSystem, Weapon, Player, Zombie 헤더파일을 include합니다. 그리고 StartScene()함수를 통해 시작 창을 만듭니다. 그리고 int형 변수를 선언하는데 플레이어와 좀비 XY좌표, 시간/체력/좀비이동/좀비소환/총알보급소환/소환된 좀비 수/ 승리여부를 나타낸다. 그리고 SpawnXY()함수를 통해 플

레이어와 좀비가 소환될 위치를 받습니다. 그리고 Weapon과 Player 구조체를 초기화하고 Player구조체에 Weapon구조체를 연결합니다. Zombie 구조체를 최대좀비수에 맞게 초기화 합니다. DrawMap()함수를 통해 맵을 그리고 PlayerBeacon()함수를 통해 플레이어를 소환합니다. 게임이 플레이가 되면서 플레이어의 체력이 0이하이고 승리하지 못했을 때 반복문으로 반복 실행됩니다. PlayerMove()와 ControlZombieHP()함수를 호출하고 ShowHP(), ShowAmmo(), WeaponName(), Score()함수를 통해 플레이어 정보를 콘솔창에 나타냅니다. 그리고 Timer변수들을 통해 일정 시간이 지나면, ZombieMove(), TimePoint, SpawnAmmo(), ZombieBeacon()을 실행 시킵니다. ZombieBeacon()을 호출하기 전에 SpawnXY()함수를 통해 임의의 위치를 받고 ZombieBeacon()함수를 실행시킵니다. while()문이 끝나면 GameResult() 함수를 호출하여 게임을 종료합니다.

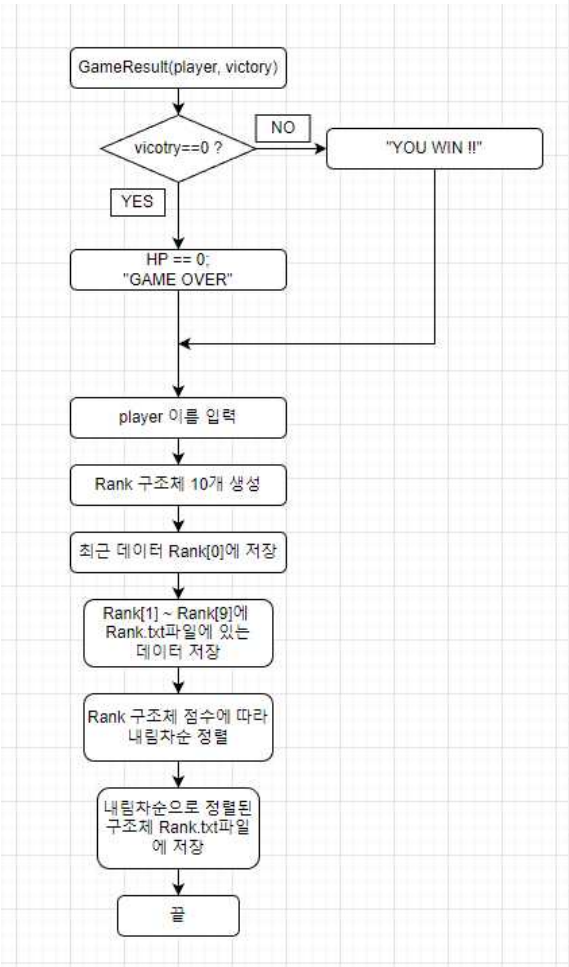
3. 부록(순서도)



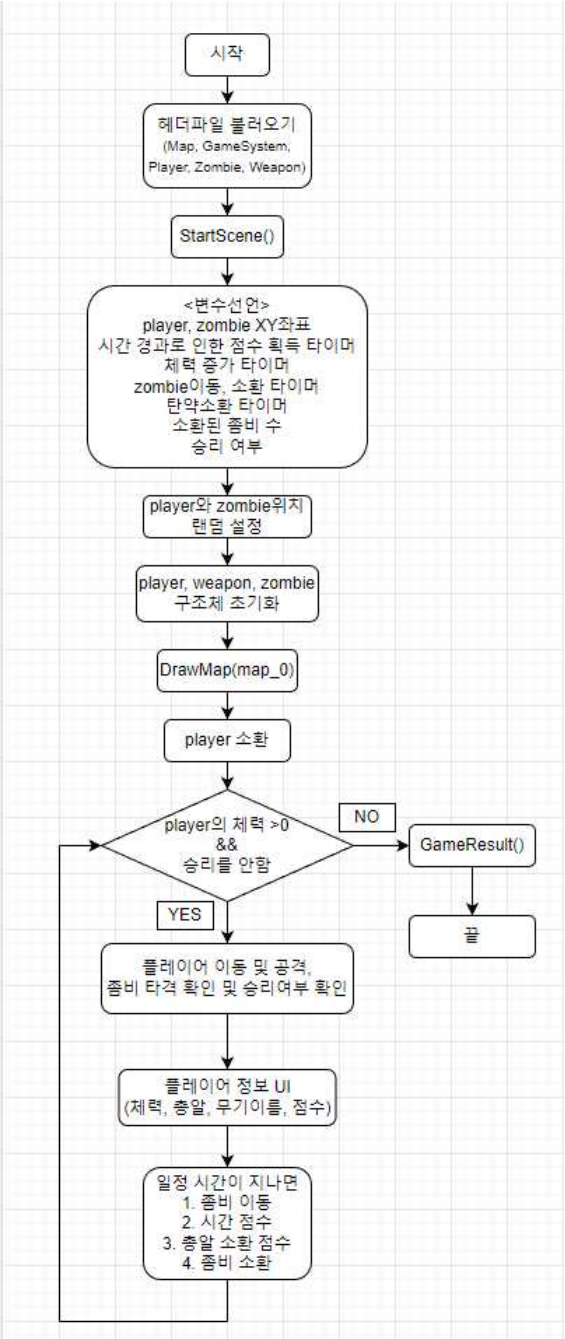
ControlZombieHP()



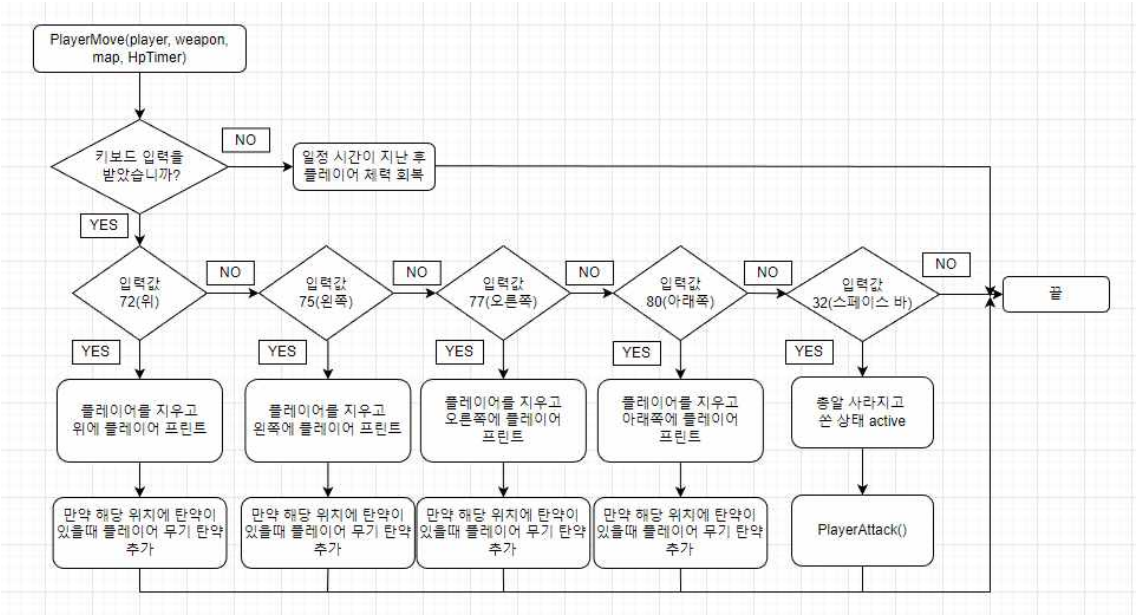
StartScene()



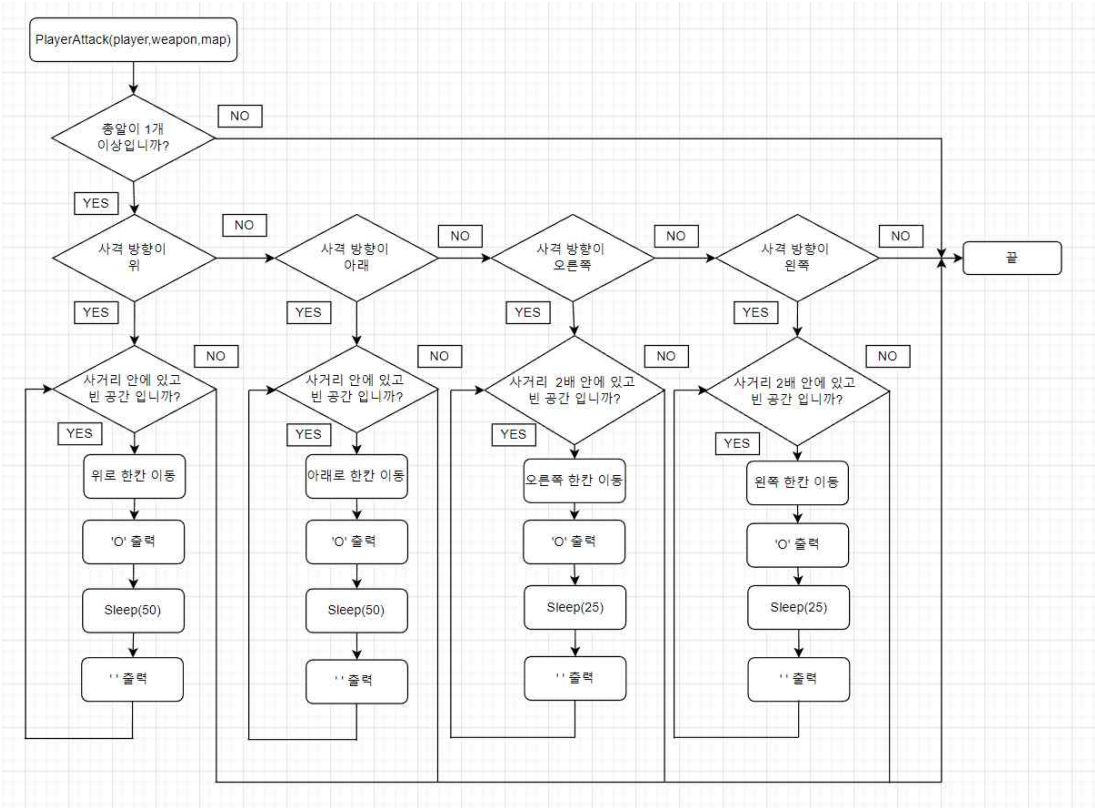
GameResult()



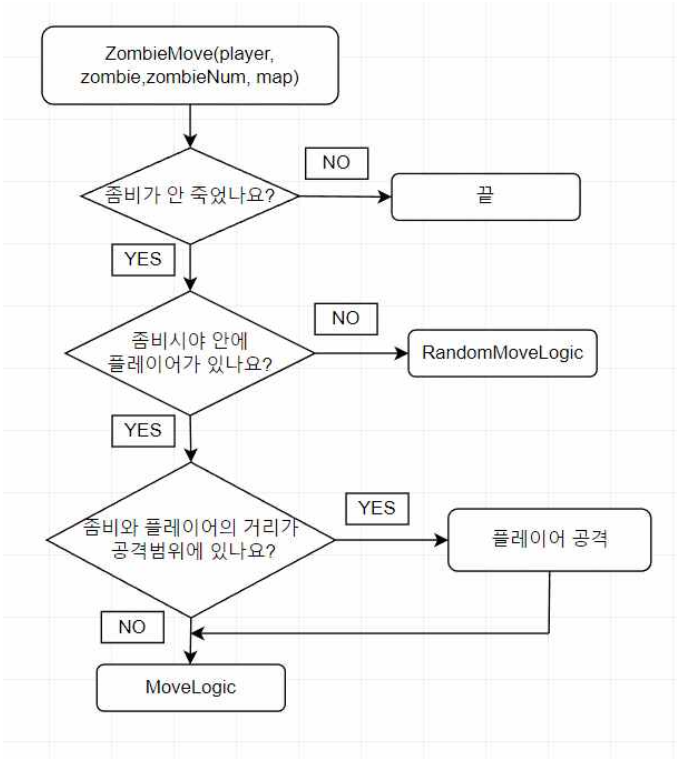
Main.c

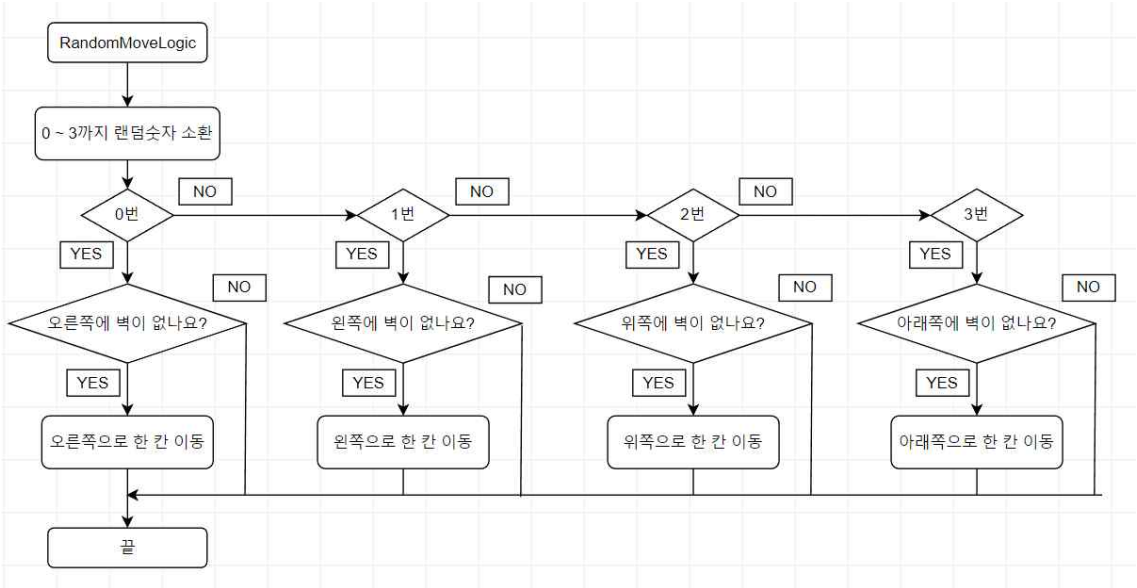


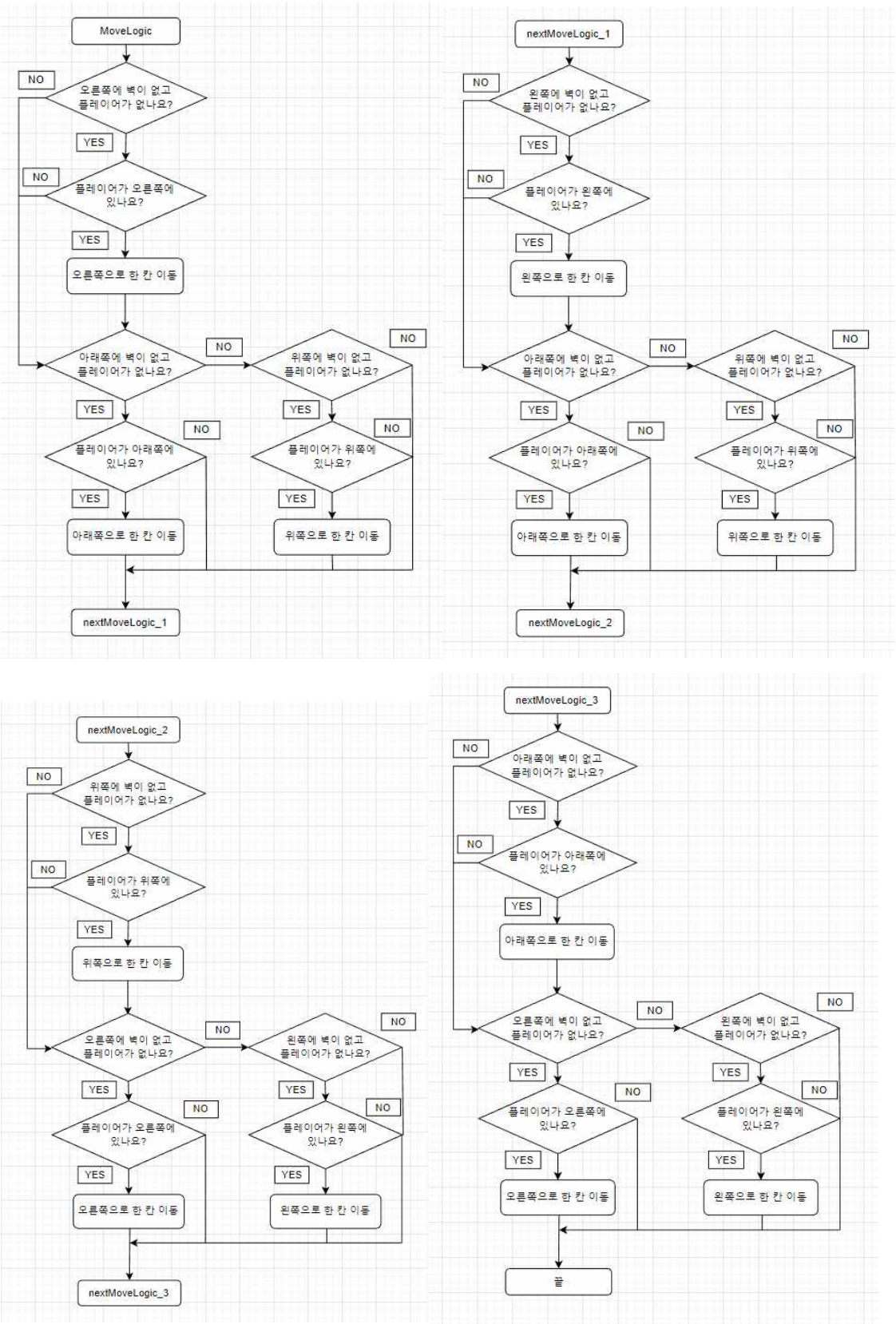
PlayerMove()



PlayerAttack()


ZombieMove(player,
zombie,zombieNum, map)
좀비가 안 죽었나요?
NO
끝
YES
좀비시야 안에
플레이어가 있나요?
NO
RandomMoveLogic
YES
좀비와 플레이어의 거리가
공격범위에 있나요?
YES
플레이어 공격
NO
MoveLogic


RandomMoveLogic
0 ~ 3까지 랜덤숫자 소환
0번
NO
1번
NO
2번
NO
3번
YES
YES
YES
YES
NO
NO
NO
NO
오른쪽에 벽이 없나요?
왼쪽에 벽이 없나요?
위쪽에 벽이 없나요?
아래쪽에 벽이 없나요?
YES
YES
YES
YES
오른쪽으로 한 칸 이동
왼쪽으로 한 칸 이동
위쪽으로 한 칸 이동
아래쪽으로 한 칸 이동
끝


MoveLogic
오른쪽에 벽이 없고 플레이어가 없나요?
NO
YES
플레이어가 오른쪽에 있나요?
NO
YES
오른쪽으로 한 칸 이동
아래쪽에 벽이 없고 플레이어가 없나요?
NO
YES
위쪽에 벽이 없고 플레이어가 없나요?
NO
YES
플레이어가 아래쪽에 있나요?
NO
YES
플레이어가 위쪽에 있나요?
NO
YES
아래쪽으로 한 칸 이동
위쪽으로 한 칸 이동
nextMoveLogic_1
nextMoveLogic_1
왼쪽에 벽이 없고 플레이어가 없나요?
NO
YES
플레이어가 왼쪽에 있나요?
NO
YES
왼쪽으로 한 칸 이동
아래쪽에 벽이 없고 플레이어가 없나요?
NO
YES
위쪽에 벽이 없고 플레이어가 없나요?
NO
YES
플레이어가 아래쪽에 있나요?
NO
YES
플레이어가 위쪽에 있나요?
NO
YES
아래쪽으로 한 칸 이동
위쪽으로 한 칸 이동
nextMoveLogic_2
nextMoveLogic_2
위쪽에 벽이 없고 플레이어가 없나요?
NO
YES
플레이어가 위쪽에 있나요?
NO
YES
위쪽으로 한 칸 이동
오른쪽에 벽이 없고 플레이어가 없나요?
NO
YES
왼쪽에 벽이 없고 플레이어가 없나요?
NO
YES
플레이어가 오른쪽에 있나요?
NO
YES
플레이어가 왼쪽에 있나요?
NO
YES
오른쪽으로 한 칸 이동
왼쪽으로 한 칸 이동
nextMoveLogic_3
nextMoveLogic_3
아래쪽에 벽이 없고 플레이어가 없나요?
NO
YES
플레이어가 아래쪽에 있나요?
NO
YES
아래쪽으로 한 칸 이동
오른쪽에 벽이 없고 플레이어가 없나요?
NO
YES
왼쪽에 벽이 없고 플레이어가 없나요?
NO
YES
플레이어가 오른쪽에 있나요?
NO
YES
플레이어가 왼쪽에 있나요?
NO
YES
오른쪽으로 한 칸 이동
왼쪽으로 한 칸 이동
끝

ZombieMove()