(C 프로그래밍) 팀 프로젝트 과제

보물찾기



201910162 문무현

202211295 문신혁

202200001 박준성

202200002 이윤희

제출일 : 2022-06-17

1. 과제 주요 기능
2. 난이도 별로 서로 다른 크기의 게임판을 생성합니다.
3. 생성된 게임판에 무작위로 깃발의 위치와 종류를 추가합니다.
4. 사용자에게 깃발이 추가된 게임판을 보여줍니다.
5. 사용자가 기록한 점수를 텍스트 파일에 저장하여 리더보드 메뉴에서 순위를 보여줍니다.
6. 게임 실행 중 방향키에 따라 플레이어가 움직이고, 이동한 곳에 있던 깃발의 종류에 따라 블런더, 괜찮은 수, 나쁜 수, 클리어 이벤트가 일어납니다.

10. 사용자에게 게임 첫시작 메뉴를 보여줍니다.

11. 사용자가 난이도를 선택할 수 있도록 해줍니다.

12. 사용자에게 리더보드 메뉴를 보여줍니다.

13. 사용자에게 게임의 스토리를 알려줍니다.

1. 주요기능 및 기능 구현을 위한 기술적 상세 명세
2. 난이도 별로 서로 다른 크기의 게임판을 생성합니다.

|  |
| --- |
| createMap() |
| Map1, Map2, Map3은 난이도에 따라서 서로 다른 크기의 기호 상수로 정의된 이차원 배열 변수입니다. createMap()은 게임판의 테두리는 벽으로, 안쪽 공간은 빈 공간으로 초기화 하는 함수입니다. |

1. 생성된 게임판에 무작위로 깃발의 위치와 종류를 추가합니다.

|  |
| --- |
| makeFlag() |
| Item 구조체는 생성된 깃발과 벽의 위치를 저장합니다. max 변수는 생성할 최대 깃발과 장애물의 숫자를 더한 값이며, flag는 그 중 깃발의 숫자만을 나타내는 변수입니다.  arrX와 arrY는 해당 난이도의 게임판에서 무작위로 정해진 깃발과 장애물의 위치를 선택할 때 사용하는 배열입니다. rand() 함수를 통해 위치를 정한 후, 이전과 중복된 위치 또는 사용자의 시작 위치에는 생성되지 않도록 합니다.  arrX와 arrY에 저장된 값을 item 구조체에 대입한 후, 무작위로 장애물과 깃발의 종류를 초기화합니다. 클리어 깃발이 아닌 다른 깃발들은 1부터 3까지의 값으로 기호상수를 통해 정의되어 있습니다. |

1. 사용자에게 깃발이 추가된 게임판을 보여줍니다.

|  |
| --- |
| showMap() |
| showMap()은 난이도에 따른 게임판에 저장된 요소들을 출력하는 함수입니다. 깃발, 벽, 빈 공간 등은 기호 상수로 정의되어 있습니다. createMap()과 makeFlag() 함수를 통해 초기화된 이차원 배열의 값을 통해서 서로 다른 형태로 출력합니다. |

1. 리더보드 점수 저장 배열을 점수가 높은 순으로 정렬할 때 필요한 함수입니다.

void Insert(int array[][2], int loc, int val1, int val2);

// 리더보드 배열 => 점수를 저장하기 위한 배열 설정. 차례대로 쿼터파이널, 세미파이널, 파이널 점수 저장 배열

int LB1[10][2] = { 0 };

int LB2[10][2] = { 0 };

int LB3[10][2] = { 0 };

void Insert(int array[][2], int loc, int val1, int val2)

{

// 넣을 인덱스 뒤의 값들을 하나씩 밀려 저장

for (int i = 8; i >= loc; i--)

{

array[i + 1][0] = array[i][0];

array[i + 1][1] = array[i][1];

}

// 값들을 한칸씩 밀어 남은 공간에 값들 저장.

array[loc][0] = val1;

array[loc][1] = val2;

}

함수에 입력된 인덱스 위치에 입력된 값을 넣습니다

// 리더보드 저장 배열을 점수 순으로 정렬할 때 필요한 인서트 함수. 입력으로 배열, 값을 넣을 인덱스 위치, 값이 들어감(남은 이동횟수와 시간)

1. 리더보드 텍스트 파일을 세팅 해주는 함수입니다.

void SetLBFiles();

// 프로그램이 종료된 후에도 리더보드 정보가 남아있도록 리더보드 순위, 점수를 저장할 텍스트 파일을 설정

void SetLBFiles()

{

// 만약 리더보드 파일이 존재하지 않는다면

if ((fp1 = fopen("LB1.txt", "r")) == NULL)

{

// 쓰기모드로 텍스트 파일을 생성

fp1 = fopen("LB1.txt", "w+");

// 초기값을 전부 0으로 설정해 저장

for (int i = 0; i < 10; i++)

{

fprintf(fp1, "%d %d\n", 0, 0);

}

fclose(fp1);

}

// 리더보드 텍스트 파일을 읽기모드로 열기

fp1 = fopen("LB1.txt", "r");

// 리더보드 배열에 텍스트 파일 내용을 저장

for (int i = 0; i < 10; i++)

{

fscanf(fp1, "%d %d\n", &LB1[i][0], &LB1[i][1]);

}

// 파일 닫기

fclose(fp1);

// 위와 동일한 과정

if ((fp2 = fopen("LB2.txt", "r")) == NULL)

{

fp2 = fopen("LB2.txt", "w+");

for (int i = 0; i < 10; i++)

{

fprintf(fp2, "%d %d\n", 0, 0);

}

fclose(fp2);

}

fp2 = fopen("LB2.txt", "r");

for (int i = 0; i < 10; i++)

{

fscanf(fp2, "%d %d\n", &LB2[i][0], &LB2[i][1]);

}

fclose(fp2);

if ((fp3 = fopen("LB3.txt", "r")) == NULL)

{

fp3 = fopen("LB3.txt", "w+");

for (int i = 0; i < 10; i++)

{

fprintf(fp3, "%d %d\n", 0, 0);

}

fclose(fp3);

}

fp3 = fopen("LB3.txt", "r");

for (int i = 0; i < 10; i++)

{

fscanf(fp3, "%d %d\n", &LB3[i][0], &LB3[i][1]);

}

fclose(fp3);

}

함수가 실행되었을 때 만약 리더보드 점수를 저장한 텍스트 파일이 없다면 쓰기모드로 생성하여 값을 전부 0으로 설정해 저장. 이후 텍스트 파일을 읽기모드로 열어 내용을 리더보드 점수 저장 배열에 넣습니다.

1. 게임 실행 중 플레이어의 이동과 상황을 제어해주는 함수입니다.
2. // 게임 실행중 플레이어의 상황과 움직임을 제어하는 함수
3. void inGameCursor(int dif)
4. {
5. // 생성된 맵 배열에서의 위치를 저장할 변수를 선언
6. int mapX, mapY;
7. char\* blunder = "블런더! 상황이 악화됩니다."; // 블런더 깃발을 먹었을 때의 텍스트
8. char\* good = "괜찮은 수를 두었습니다. 생각할 기회가 늘었습니다."; // 괜찮은 수 깃발을 먹었을 때의 텍스트
9. char\* bad = "나쁜 수를 두었습니다. 생각할 기회가 줄었습니다."; // 나쁜 수 깃발을 먹었을 때의 텍스트
10. // 쿼터 파이널 난이도
11. if (dif == 0)
12. {
13. // 움직임 횟수제한 70회, 시간제한 20초, 시작시각 선언
14. movecount = 70;
15. timelimit = 20;
16. starttime = time(0);
17. // 플레이어 시작 위치
18. playerX = 2;
19. playerY = 1;
20. // 맵 배열에서의 시작 위치
21. mapX = mapY = 1;
22. system("cls");
23. // 맵을 출력
24. showMap(dif);
25. while (1)
26. {
27. // 남은시간과 이동횟수를 출력
28. gotoxy(60, 26);
29. printf("남은시간: %d ", (timelimit + starttime - time(0)));
30. gotoxy(60, 28);
31. printf("남은 이동횟수: %d ", movecount);
32. // 플레이어 위치에 다이아 문양 출력
33. gotoxy(playerX, playerY);
34. printf("◆");
35. // 키보드가 눌렸을 때 실행
36. if (\_kbhit()) {
37. ch = \_getch();
38. if (ch == 224)
39. {
40. ch = \_getch();
41. switch (ch)
42. {
43. case UP: // 윗 방향키가 눌렸을 때
44. mapY--; // 맵 배열에서의 y -1
45. playerY--; // 플레이어의 y -1
46. if (Map1[mapY][mapX] == WALL) { // 만약 이동한 곳이 벽이라면
47. mapY++; // 맵 배열에서의 위치와 플레이어의 위치 복구
48. playerY++;
49. }
50. break;
51. case DOWN: // 아래 방향키가 눌렸을 때
52. mapY++; // 맵 배열에서의 y +1
53. playerY++; // 플레이어의 y +1
54. if (Map1[mapY][mapX] == WALL) { // 만약 이동한 곳이 벽이라면
55. mapY--; // 위치 복구
56. playerY--;
57. }
58. break;
59. case LEFT: // 왼쪽 방향키가 눌렸을 때
60. mapX--; // 맵 배열에서의 x -1
61. playerX -= 2; // 플레이어의 x -2(-2인 이유는 플레이어를 나타내는 다이아 문양이나 벽, 깃발 문양등이 2바이트짜리 문자라 공간 2칸을 차지하기 때문)
62. if (Map1[mapY][mapX] == WALL) { // 만약 이동한 곳이 벽이라면
63. mapX++; // 위치 복구
64. playerX += 2;
65. }
66. break;
67. case RIGHT: // 오른쪽 방향키가 눌렸을 때
68. mapX++; // 맵 배열에서의 x +1
69. playerX += 2; // 플레이어의 x +2
70. if (Map1[mapY][mapX] == WALL) { // 만약 이동한 곳이 벽이라면
71. mapX--; // 위치 복구
72. playerX -= 2;
73. }
74. break;
75. }
76. // 움직이기 전 전에 위치했던 곳에 남아있는 플레이어 문양을 제거
77. printf("\b\b ");
78. // 만약 움직인 곳이 플래그 깃발이라면
79. if (Map1[mapY][mapX] == FLAG) {
80. winLevel(dif); // 게임에서 승리
81. }
82. // 만약 움직인 곳이 블런더 깃발이라면
83. if (Map1[mapY][mapX] == FAKE1) {
84. Map1[mapY][mapX] = EMPTY; // 맵 배열에서 깃발을 지우고
85. gotoxy(playerX, playerY); // 콘솔창에 출력 되어있는 깃발도 지운다
86. printf(" ");
87. playerX = 2; // 플레이어의 위치와 배열에서의 위치를 초기화시킴
88. playerY = 1;
89. mapX = mapY = 1;
90. gotoxy(0, 25);
91. alert(blunder); // 블런더 문구 출력
92. }
93. // 만약 움직인 곳이 괜찮은 수 깃발이라면
94. if (Map1[mapY][mapX] == FAKE2)
95. {
96. Map1[mapY][mapX] = EMPTY; // 맵 배열에서 깃발을 지우고
97. movecount += 10; // 이동횟수와 시간을 각각 10, 5씩 증가시킨다.
98. timelimit += 5;
99. gotoxy(0, 25);
100. alert(good); // 괜찮은 수 문구 출력
101. }
102. // 만약 움직인 곳이 나쁜 수 깃발이라면
103. if (Map1[mapY][mapX] == FAKE3)
104. {
105. Map1[mapY][mapX] = EMPTY; // 맵 배열에서 깃발을 지우고
106. movecount -= 5; // 이동횟수 5회 차감
107. gotoxy(0, 25);
108. alert(bad); // 나쁜 수 문구 출력
109. }
110. }
111. // 이동 후 만약 이동횟수가 0이 된다면
112. if (movecount <= 0)
113. {
114. loseLevel(); // 게임에서 패배
115. break;
116. }
117. // 이동 할 때마다 이동횟수 -1
118. movecount--;
119. }
120. // while문이 돌아가다가 남은 시간이 0이하가 된다면
121. if (timelimit + starttime - time(0) <= 0)
122. {
123. loseLevel(); // 게임에서 패배
124. break;
125. }
126. }
127. }
128. else if (dif == 1) // 세미 파이널 난이도
129. {
130. // 이동횟수 110회, 시간 30초
131. movecount = 110;
132. timelimit = 30;
133. starttime = time(0);
134. // 나머지는 쿼터 파이널과 동일한 로직으로 돌아감
135. playerX = 2;
136. playerY = 1;
137. mapX = mapY = 1;
138. system("cls");
139. showMap(dif);
140. while (1)
141. {
142. gotoxy(60, 26);
143. printf("남은시간: %d ", (timelimit + starttime - time(0)));
144. gotoxy(60, 28);
145. printf("남은 이동횟수: %d ", movecount);
146. gotoxy(playerX, playerY);
147. printf("◆");
148. if (\_kbhit()) {
149. ch = \_getch();
150. if (ch == 224)
151. {
152. ch = \_getch();
153. switch (ch)
154. {
155. case UP: // 윗 방향키가 눌렸을 때
156. mapY--; // 맵 배열에서의 y -1
157. playerY--; // 플레이어의 y -1
158. if (Map2[mapY][mapX] == WALL) { // 만약 이동한 곳이 벽이라면
159. mapY++; // 맵 배열에서의 위치와 플레이어의 위치 복구
160. playerY++;
161. }
162. break;
163. case DOWN: // 아래 방향키가 눌렸을 때
164. mapY++; // 맵 배열에서의 y +1
165. playerY++; // 플레이어의 y +1
166. if (Map2[mapY][mapX] == WALL) { // 만약 이동한 곳이 벽이라면
167. mapY--; // 위치 복구
168. playerY--;
169. }
170. break;
171. case LEFT: // 왼쪽 방향키가 눌렸을 때
172. mapX--; // 맵 배열에서의 x -1
173. playerX -= 2; // 플레이어의 x -2(-2인 이유는 플레이어를 나타내는 다이아 문양이나 벽, 깃발 문양등이 2바이트짜리 문자라 공간 2칸을 차지하기 때문)
174. if (Map2[mapY][mapX] == WALL) { // 만약 이동한 곳이 벽이라면
175. mapX++; // 위치 복구
176. playerX += 2;
177. }
178. break;
179. case RIGHT: // 오른쪽 방향키가 눌렸을 때
180. mapX++; // 맵 배열에서의 x +1
181. playerX += 2; // 플레이어의 x +2
182. if (Map2[mapY][mapX] == WALL) { // 만약 이동한 곳이 벽이라면
183. mapX--; // 위치 복구
184. playerX -= 2;
185. }
186. break;
187. }
188. // 움직이기 전 전에 위치했던 곳에 남아있는 플레이어 문양을 제거
189. printf("\b\b ");
190. // 만약 움직인 곳이 플래그 깃발이라면
191. if (Map2[mapY][mapX] == FLAG) {
192. winLevel(dif); // 게임에서 승리
193. }
194. // 만약 움직인 곳이 블런더 깃발이라면
195. if (Map2[mapY][mapX] == FAKE1) {
196. Map2[mapY][mapX] = EMPTY; // 맵 배열에서 깃발을 지우고
197. gotoxy(playerX, playerY); // 콘솔창에 출력 되어있는 깃발도 지운다
198. printf(" ");
199. playerX = 2; // 플레이어의 위치와 배열에서의 위치를 초기화시킴
200. playerY = 1;
201. mapX = mapY = 1;
202. gotoxy(0, 25);
203. alert(blunder); // 블런더 문구 출력
204. }
205. // 만약 움직인 곳이 괜찮은 수 깃발이라면
206. if (Map2[mapY][mapX] == FAKE2)
207. {
208. Map2[mapY][mapX] = EMPTY; // 맵 배열에서 깃발을 지우고
209. movecount += 10; // 이동횟수와 시간을 각각 10, 5씩 증가시킨다.
210. timelimit += 5;
211. gotoxy(0, 25);
212. alert(good); // 괜찮은 수 문구 출력
213. }
214. // 만약 움직인 곳이 나쁜 수 깃발이라면
215. if (Map2[mapY][mapX] == FAKE3)
216. {
217. Map2[mapY][mapX] = EMPTY; // 맵 배열에서 깃발을 지우고
218. movecount -= 5; // 이동횟수 5회 차감
219. gotoxy(0, 25);
220. alert(bad); // 나쁜 수 문구 출력
221. }
222. }
223. // 이동 후 만약 이동횟수가 0이 된다면
224. if (movecount <= 0)
225. {
226. loseLevel(); // 게임에서 패배
227. break;
228. }
229. // 이동 할 때마다 이동횟수 -1
230. movecount--;
231. }
232. if (timelimit + starttime - time(0) <= 0)
233. {
234. loseLevel();
235. break;
236. }
237. }
238. }
239. else if (dif == 2) // 파이널 난이도
240. {
241. // 이동횟수 200회, 시간 45초
242. movecount = 200;
243. timelimit = 45;
244. starttime = time(0);
245. playerX = 2;
246. playerY = 1;
247. mapX = mapY = 1;
248. //Map1[playerY][playerX] = CUR;
249. system("cls");
250. showMap(dif);
251. while (1)
252. {
253. gotoxy(60, 26);
254. printf("남은시간: %d ", (timelimit + starttime - time(0)));
255. gotoxy(60, 28);
256. printf("남은 이동횟수: %d ", movecount);
257. gotoxy(playerX, playerY);
258. printf("◆");
259. if (\_kbhit()) {
260. ch = \_getch();
261. if (ch == 224)
262. {
263. ch = \_getch();
264. switch (ch)
265. {
266. case UP: // 윗 방향키가 눌렸을 때
267. mapY--; // 맵 배열에서의 y -1
268. playerY--; // 플레이어의 y -1
269. if (Map3[mapY][mapX] == WALL) { // 만약 이동한 곳이 벽이라면
270. mapY++; // 맵 배열에서의 위치와 플레이어의 위치 복구
271. playerY++;
272. }
273. break;
274. case DOWN: // 아래 방향키가 눌렸을 때
275. mapY++; // 맵 배열에서의 y +1
276. playerY++; // 플레이어의 y +1
277. if (Map3[mapY][mapX] == WALL) { // 만약 이동한 곳이 벽이라면
278. mapY--; // 위치 복구
279. playerY--;
280. }
281. break;
282. case LEFT: // 왼쪽 방향키가 눌렸을 때
283. mapX--; // 맵 배열에서의 x -1
284. playerX -= 2; // 플레이어의 x -2(-2인 이유는 플레이어를 나타내는 다이아 문양이나 벽, 깃발 문양등이 2바이트짜리 문자라 공간 2칸을 차지하기 때문)
285. if (Map3[mapY][mapX] == WALL) { // 만약 이동한 곳이 벽이라면
286. mapX++; // 위치 복구
287. playerX += 2;
288. }
289. break;
290. case RIGHT: // 오른쪽 방향키가 눌렸을 때
291. mapX++; // 맵 배열에서의 x +1
292. playerX += 2; // 플레이어의 x +2
293. if (Map3[mapY][mapX] == WALL) { // 만약 이동한 곳이 벽이라면
294. mapX--; // 위치 복구
295. playerX -= 2;
296. }
297. break;
298. }
299. // 움직이기 전 전에 위치했던 곳에 남아있는 플레이어 문양을 제거
300. printf("\b\b ");
301. // 만약 움직인 곳이 플래그 깃발이라면
302. if (Map3[mapY][mapX] == FLAG) {
303. winLevel(dif); // 게임에서 승리
304. }
305. // 만약 움직인 곳이 블런더 깃발이라면
306. if (Map3[mapY][mapX] == FAKE1) {
307. Map3[mapY][mapX] = EMPTY; // 맵 배열에서 깃발을 지우고
308. gotoxy(playerX, playerY); // 콘솔창에 출력 되어있는 깃발도 지운다
309. printf(" ");
310. playerX = 2; // 플레이어의 위치와 배열에서의 위치를 초기화시킴
311. playerY = 1;
312. mapX = mapY = 1;
313. gotoxy(0, 25);
314. alert(blunder); // 블런더 문구 출력
315. }
316. // 만약 움직인 곳이 괜찮은 수 깃발이라면
317. if (Map3[mapY][mapX] == FAKE2)
318. {
319. Map3[mapY][mapX] = EMPTY; // 맵 배열에서 깃발을 지우고
320. movecount += 10; // 이동횟수와 시간을 각각 10, 5씩 증가시킨다.
321. timelimit += 5;
322. gotoxy(0, 25);
323. alert(good); // 괜찮은 수 문구 출력
324. }
325. // 만약 움직인 곳이 나쁜 수 깃발이라면
326. if (Map3[mapY][mapX] == FAKE3)
327. {
328. Map3[mapY][mapX] = EMPTY; // 맵 배열에서 깃발을 지우고
329. movecount -= 5; // 이동횟수 5회 차감
330. gotoxy(0, 25);
331. alert(bad); // 나쁜 수 문구 출력
332. }
333. }
334. // 이동 후 만약 이동횟수가 0이 된다면
335. if (movecount <= 0)
336. {
337. loseLevel(); // 게임에서 패배
338. break;
339. }
340. // 이동 할 때마다 이동횟수 -1
341. movecount—;
342. }
343. if (timelimit + starttime - time(0) <= 0)
344. {
345. loseLevel();
346. break;
347. }
348. }
349. }
350. }

매개변수 dif로 난이도 값을 받아 그에 따른 난이도의 게임을 실행. While문을 실행하는 동안 남은 시간과 이동횟수를 게속 출력. 중간에 만약 키보드 입력이 있다면 플레이어 이동을 실행하고, 위치에 따라 이벤트가 발생합니다.

10. 사용자에게 게임 첫시작 메뉴를 보여줍니다.

|  |
| --- |
| showMenu() |
| showMenu()는 사용자에게 메뉴를 보여주는 함수입니다. 게임시작, 게임방법, 리더보드, 게임종료 라는 4가지의 선택권이 주어진 뒤 커서를 이동하여 선택하는 방식입니다. 커서의 현 y좌표에서 y좌표의 최솟값을 뺀 값을 받아 switch문을 사용하여 각각의 함수로 이동할 수 있도록 하였습니다. 만약 사용자가 y좌표의 최소보다 올라가려 하면 y값을 하나 낮춰주고 최대는 그 반대로 하여 실행되지 않도록 하였습니다. |

11. 사용자가 난이도를 선택할 수 있도록 해줍니다.

|  |
| --- |
| selectDifficulty() |
| selectDifficulty()은 사용자에게 난이도를 선택하게 해줍니다. 방향키로 커서를 위아래로 움직이기 때문에 0xE0값을 반환하고 y좌표의 최대최소 범위 내에서 움직여줍니다. 그 후 엔터키를 누르면 현 y좌표에서 최솟값을 뺀 값을 반환하여 줍니다. |

12. 사용자에게 리더보드 메뉴를 보여줍니다.

|  |
| --- |
| LBMenu() |
| LBMenu()는 리더보드 메뉴를 보여주어 난이도를 선택하도록 해줍니다. 커서를 이동시켜놓고 현위치에서 첫 위치를 빼준 값을 비교하여 각각의 값에 알맞게 함수를 호출해줍니다. 방향키를 이용해 위아래로 움직이므로 224값을 반환하고 y좌표의 최대 최소 값을 넘으면 값을 1씩 조정하여 움직이지 않도록 해줍니다. |

13. 사용자에게 게임의 스토리를 알려줍니다.

|  |
| --- |
| story() |
| story()는 사용자가 게임의 스토리를 알 수 있도록 해줍니다. 사용자가 키보드 위의 키를 누를 때까지 스크립트를 반복문을 사용하여 한글자씩 출력하고 실행이 잠시 멈추게 됩니다. |

1. 토의사항
2. 팀프로젝트 진행

프로젝트를 진행하면서 소통의 중요성을 크게 느꼈습니다. 팀원들에게 자신의 현재 진행 상황을 다른 팀원들에게 알리지 않으면, 작업의 분배가 매우 비효율적으로 이루어지거나 서로 다른 팀원이 같은 기간 동안 동일한 작업을 하는 경우가 생깁니다.

그렇기 때문에 git을 사용하는 팀원들이 많았음에도 프로젝트에 활용하지 않은 것이 아쉽게 느껴집니다. 이번 학기 동안 프로젝트를 진행하면서, 다른 사람과 함께 프로젝트를 할 때 그러한 프로그램을 사용하는 것의 중요성을 느꼈습니다. 앞으로도 협업을 하는 일이 많이 있을 테니 학교에서 경험을 쌓았으면 좋았을 것 같다는 아쉬움이 남습니다.

1. 기능 구현

첫번째로 팀 프로젝트를 진행하는 것이니 만큼 역할을 잘 나누고 기능에 따라 파일을 분할하여 작성하고 싶었는데, 의도대로 진행되지 않았습니다. 다른 파일을 만들지 않게 되어서 지역 변수를 사용할 수 있음에도 전역 변수를 사용하는 부분들이 생겼습니다.

두번째로 대부분의 함수들이 난이도에 따라서 Map1, Map2, Map3 변수에 동일한 코드를 반복하고 있는 것이 아쉽습니다. 이러한 중복 코드는 반드시 피할 수 있는 방법이 있을 텐데 설계 단계에서 더 깊게 생각하지 않았습니다. 떠오른 방법 중 하나는 Map 변수를 하나로 만들고 동적 할당하는 것입니다. 하지만 게임판들의 크기가 이미 정해져 있는 만큼 동적 할당보다 더 좋은 해결 방법이 있을 것이라고 가정하고, 어떻게 해결하는 것이 가장 좋을지 생각하고 있습니다.

세번째로 showMenu() 함수가 반환되지 않은 상태로 여러 곳에서 호출하고 있다는 것이 마음에 걸립니다. 콜 스택에 대해 제대로 된 개념이 잡혀 있지는 않아 명확히 설명할 수 없지만, 이러한 구조는 좋지 않아 보입니다. 함수마다 반환 값을 설정하고 showMemu() 함수로 돌아가는 구조로 만드는 것을 고민하고 있었지만, 기한 안에 해결하지 못했습니다. 하지만 다음에도 비슷한 상황을 겪을 수 있으니 더 고민해보고 이 문제를 해결할 것입니다.

프로젝트 경험 부족 (이윤희)

개인적으로 코딩을 해보는 것이 처음이라 프로젝트를 구현할 때 사용되는 다양한 함수들이 낮설었고 이해하기 위해 노력해야하는 점이 어려웠습니다. 앞으로 또다른 프로젝트에 참여하고 스스로 개발능력을 키우기 위해 많은 시간을 투자해야겠다는 것을 깨닫게 되었습니다.

1. 기여도

문무현 25%

* 스토리 구상
* 메인 메뉴 및 난이도 선택 메뉴
* 맵 초기화 및 출력
* 깃발 및 장애물 생성
* 난이도 설정
* 패배 조건 설정

문신혁 25%

* 리더보드 관련 기능 구현
* Ingamecursor 함수 내 플레이어 이동 구현
* 플레이어가 이동할 때마다 콘솔 창 출력이 깜빡거리는 현상 수정
* 메뉴에서 커서 깜빡임 제거, \*모양 표시 구현
* 콘솔 창 크기 고정

박준성 25%

이윤희 25%

* ppt 만들기
* 사용자 주위 장애물 없애기