|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 9주차 과제 | | | | | |
| 학년 | 2 | 학번 | 2017707007 | 이름 | 강민준 |

1. 과제 설명

자동차 달리기 게임. 단순히 장애물을 피하여 결승점까지 갈 수 있도록 조작하여 승리를 얻어내면 이기는 게임이다, 단, 조작 도중 장애물에 부딪히면 패배로 끝나고, 자동차길이는 1~3, 방향키 한번 당 자동차 스피드는 1~3으로 랜덤이다.

게임을 켰다끄면 자동차길이, 장애물 길이,위치 가 바뀌고 방향키를 누를 때마다 자동차의 속도가 달라진다.

1. 과제 이론

----User.h---- 본인이 정의한 헤더파일.

1. #include<stdio.h>
2. #include<Windows.h> // printf를 원하는 좌표에서 함
3. #include<conio.h> //키보드 입력 받는거
4. #include<stdlib.h>//random
5. #include<time.h>
6. #define random() srand((unsigned int)time(NULL))
7. void print\_car2(position); //1~3길이를 가진 차 출력
8. int print\_CAR(boom, position); //위 car2함수 사용 및 장애물 충돌 처리
9. int player\_set(button); //user key설정
10. void gotoxy(int x, int y); //printf의 위치변환가능
11. void draw\_map(); //map 출력
12. void draw\_ob(boom); //장애물 출력
13. void CLEAR\_BUF(); //key입력시 버퍼비우는 전용함수
14. void win(); //게임 이겼을 시,
15. void die(); //장애물에 충돌 했을 시,
16. typedef struct user\_set //USER key setting 구조체
17. {
18. char drive;
19. char up;
20. char down;
21. }USER;
22. typedef struct obstacle //장애물 위치 구조체
23. {
24. int sero[6];
25. int garo[6];
26. }OB;
27. typedef struct car\_set // 자동차 정보 구조체 // 다이렉트로 사용하지 않고, WHERE data type에서 사용.
28. {
29. int speed;
30. int length;
31. }CAR;
32. typedef struct user\_position //자동차 위치 구조체, 자동차 정보구조체를 가지고 있음.
33. {
34. int sero;
35. int garo;
36. CAR set;
38. }WHERE;

----main.c----

1. #include "User.h"
2. /\*
3. == 구조체 datatype==
4. WHERE 사용자 위치
5. USER 플레이어 키설정
6. CAR 자동차 상태
7. OB 장애물
8. \*/
9. int main()
10. {
11. int life = 1; //게임 승리와 패배를 가름하는 flag 변수
12. random(); // user.h파일에 random() 상수값 정의.
13. USER button = {0,0,0};
14. WHERE position = { 7,3,{ 0,rand()%3+1} }; //CAR구조체를 가지고 있다. 자동차 길이를 랜덤으로 함
15. OB boom = { 0,0 };
16. for (int i = 0; i < 5; i++) { // 0~5개의 몇개의 장애물을 출력할건지 결정
17. boom.garo[i] = (rand() % 2 + 1) % 2;
18. }
19. boom.garo[0] = 0;
20. for (int i = 1; i < 5; i++) { // 장애물 세로길이 지정
21. boom.sero[i] = rand() % 6 + 1;
22. }
24. player\_set(&button); //키값 입력
25. print\_CAR(&boom, &position); // 자동차 위치 출력
27. while (1)
28. {
29. if (kbhit()) { //키보드가 눌렸는지 안눌렸는지 판단하는 함수.
30. char key = \_getch(); //키에서 입력받은 값을 key라는 변수에저장.
31. if (key == button.drive) { //전진
32. if (position.garo > 58 - position.set.length\*2) { //승리 시,
33. break;
34. }
35. (position.garo) += (position.set.speed = (int)(rand()%3+1)\*2); //차 속도를 랜덤으로 결정하기위해 사용
36. life = print\_CAR(&boom, &position);
37. if (!life) //패배 시,
38. break;
39. }
40. else if (key == button.up) { //위로 갈 시,
41. if (position.sero > 2) {
42. position.sero--;
43. life = print\_CAR(&boom, &position);
44. if (!life)
45. break;
46. }
47. }
48. else if (key == button.down) { //아래로 갈 시,
49. if (position.sero < 8) {
50. position.sero++;
51. life = print\_CAR(&boom, &position);
52. if (!life)
53. break;
54. }
55. }
56. }
57. }
58. if (life) { //승리 한다면 life가 1이라면,
59. win();
60. }
61. else {
62. die();
63. }
64. }

-----func.c----- 사용되는 함수를 정의해놓은 소스파일

1. #include"User.h"
2. void gotoxy(int x, int y)
3. {
4. COORD Pos = { x - 1, y - 1 };
5. SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), Pos);
6. }
7. void draw\_map() { //맵 출력함수
8. for (int i = 1; i < 60; i += 2) {
9. gotoxy(i, 1); printf("■");
10. gotoxy(i, 9); printf("■");
11. }
12. }
13. //speed가 3이 max니깐, 장애물 최소 간격은 4가 되어야 한다. +=4 씩증가하는 for문을 구성하고,
14. //rand값으로 1/2확률을 계산해서, 장애물을 생성할건지 말건지를 판단하고 boom변수에 따른 장애물 길이를 생성.
15. //이때 장애물 넓이는 무조건 1로 가정한다.
16. void draw\_ob(OB \*boom) { //장애물 출력
17. for (int k=0, i = 1; i < 60; i +=10 ,k++) {
18. if (boom->garo[k]) {
19. boom->garo[k] = i;
20. for (int j = 0; j < boom->sero[k]; j++) {
21. gotoxy(i, j+2); printf("■");
22. }
23. }
24. }
25. }
26. int print\_CAR(OB \*boom, WHERE \*position) //자동차 위치 출력함수.
27. {
28. system("cls");
29. draw\_map();
30. draw\_ob(boom);
31. print\_car2(position);
32. for (int i = 0; i < 6; i++) {
33. for (int j = 0; j < position->set.length\*2; j += 2) {
34. if (((position->garo + j >= boom->garo[i]) && (position->garo + j <= boom->garo[i])) && position->sero - 2 < boom->sero[i]) {//-2만하기
35. return 0;
36. break;
37. }
38. }
39. }
40. }
41. void print\_car2(WHERE \*position)
42. {
43. //position.set.length == 1
44. gotoxy(position->garo, position->sero);
45. if (position->set.length == 1) {
46. printf("□");
47. }
48. else if (position->set.length == 2) {
50. printf("□□");
51. }
52. else {
53. printf("□□□");
54. }
55. }
56. int player\_set(USER \*button) //플레이어 세팅에 관한 함수.
57. {
58. printf("플레이어 세팅\n");
59. printf("플레이어 입력 Dirve key : ");
60. scanf("%c", &button->drive);
61. CLEAR\_BUF();
62. printf("플레이어 입력 Up key : ");
63. scanf("%c", &button->up);
64. CLEAR\_BUF();
65. printf("플레이어 입력 Down key : ");
66. scanf("%c", &button->down);
68. system("cls");
69. }
70. void CLEAR\_BUF() //scanf 함수 사용시, 버퍼 정리함수.
71. {
72. char buffer=0;
73. while (buffer = getchar() != '\n' && buffer != EOF);
74. }
75. void win() //승리 시
76. {
77. system("cls");
78. printf("WIN!\n");
79. }
80. void die() //패배 시
81. {
82. system("cls");
83. printf("DIE!\n");
85. }

위와 같이 3개의 파일로 구성하였으며, user.h에는 사용할 헤더파일과 함수 프로토타입이 명시되어있고, func.c에는 사용하는 함수의 define부분을 명시해 놓고 main문에서 작업하였다.

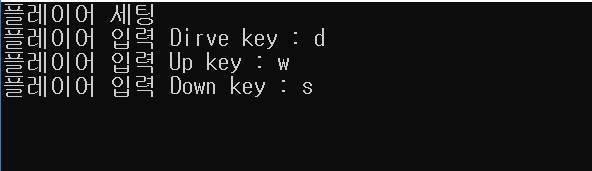
구조체를 자동차 user\_key를 입력받는 구조체, 장애물처리 구조체, 자동차 상태구조체 자동차 위치 구조체 총 4개로 나눴고, 자동차상태를 나타내는 구조체는 자동차위치를 나타내는 것의 멤버변수로 사용했고, 함수의 사용처는, 프로토타입 옆에 적어놓은 주석과 같다.

3. 주요 소스 설명 (코드 내 중요도를 매겨 판단.)

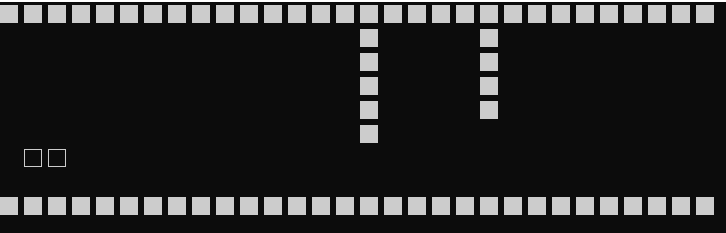
|  |
| --- |
| 1. int print\_CAR(OB \*boom, WHERE \*position) //자동차 위치 출력함수. 2. { 3. system("cls"); 4. draw\_map(); 5. draw\_ob(boom); 6. print\_car2(position); 7. for (int i = 0; i < 6; i++) { 8. for (int j = 0; j < position->set.length\*2; j += 2) { 9. if (((position->garo + j >= boom->garo[i]) && (position->garo + j <= boom->garo[i])) && position->sero - 2 < boom->sero[i]) { 10. return 0; 11. break; 12. } 13. } 14. } 15. }   라는 구문이고, 자동차위치를 출력하는 함수를 가장 중요한 함수로 꼽았다. 그 이유는, 장애물처리 연산이 들어가 있어 게임의 필수구성요소를 가리기 때문이다.  if (((position->garo + j >= boom->garo[i]) && (position->garo + j <= boom->garo[i])) && position->sero - 2 < boom->sero[i]) 와 같은 조건을 넣어 장애물 충돌 연산을 하게끔 했다.  장애물이 맵 중 어디에 위치해있는지를 넣어논 boom.garo[i]의 값과, 사용자현재위치를 받는 position->garo의 값을 비교하여 연산한다. 또한 자동차 길이에 따른 차등적 연산도 for문으로 해결했다. |
| #과제 2번  void draw\_ob(OB \*boom) { //장애물 출력  for (int k=0, i = 1; i < 60; i +=10 ,k++) {  if (boom->garo[k]) {  boom->garo[k] = i;  for (int j = 0; j < boom->sero[k]; j++) {  gotoxy(i, j+2); printf("■");  }  }  }  }  장애물을 출력하는 함수이다. random값으로 게임을 재시작할 때 마다, 장애물 길이가 바뀌게 설정해 놨으며, 장애물의 수는 제한해뒀다. |

4. 실행화면

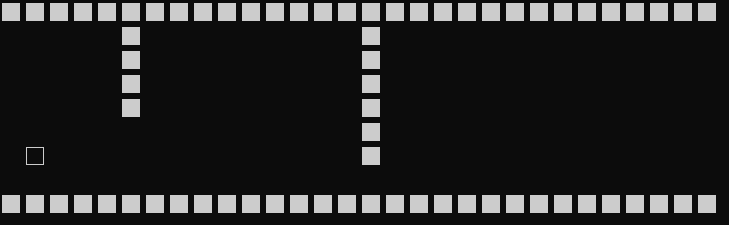
키 입력



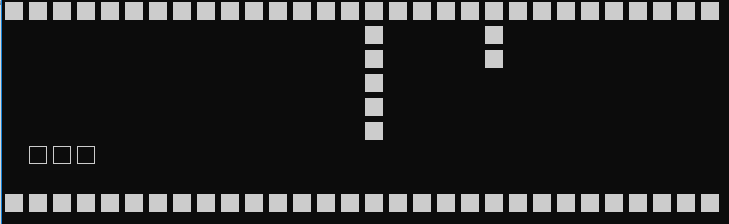
실행화면\_1 번째



실행화면\_2 번째



실행화면\_ 3 번째



자동차 길이가 랜덤이고, 장애물 길이도 랜덤인 것을 확인 할 수 있다.

1. 고찰

기존 별달리기 게임보다 재밌는 코딩작업이였음. 장애물에 부딪혀서 죽는다는 개념을 장애물위치와 사용자위치가 같으면 그렇게 하면 되지 않을까라는 생각은 했지만, 자동차 길이에 대한 개념과, 자동차 속도에 따라서 예외처리를 해줘야 하기 때문에 이부분에서 가장많은 시간소요가 있었음. 그이외에는 막힘 없이 코딩을 할 수 있었던 작업임.