Code Review - Minefind B

YJH ver

1. 구성

Define 게임 구성에 필요한 정보들이 정의 되어있다.

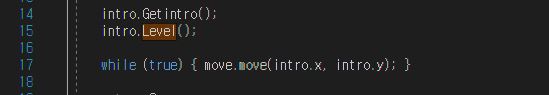
Intro 게임의 시작과 설정을 책임진다.  
Control 게임의 동작과 키 입력을 책임진다.  
Search 모든 연산이 이루어진다. GameManager 와 닮은꼴

2. 작동

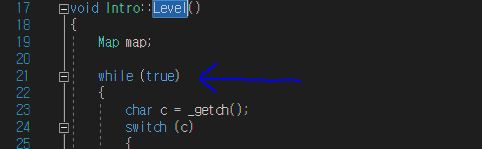
Uml 참조

3. 아쉬운점

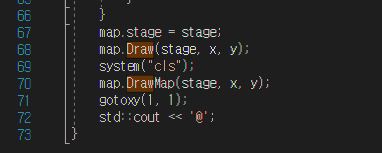
- Main()



초반 도입부의 나눠진 루프들은 ( Level() 주목 )

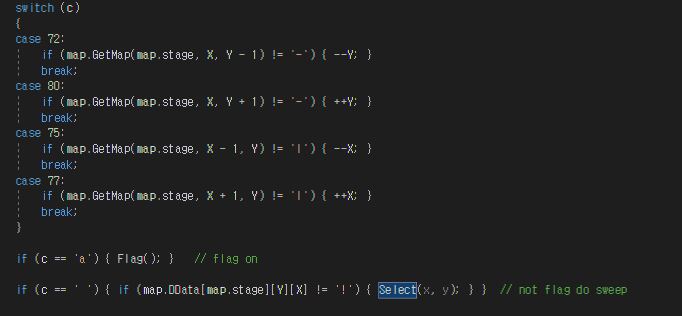
  
일관성이 없어 식별자와 구성으로는 동작이 어떻게 이루어지는지 짐작할 수가 없었다.

통일성 있게 Level() 에도 반복문을 주거나 move() 도 내부적으로 반복문을 주었으면 했다.

- Intro::Level()  
  
Draw() 는 지뢰를 그려준다. 하지만 매설한다고 하는 것이 이해친화적일 것이다.

DrawMap() 출력용함수가 여기에 있어 역할분배가 안된 것 같다.

- Control -> move( int , int )



같은 키 값을 확인하는 단계인데 switch 와 if 가 나눠있다.

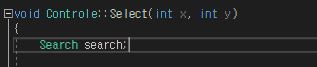
- Control -> move( int , int )



Select() 는 지금 위치에 대한 작용을 처리하는 함수지만 움직임을 담당하는 함수 안에서 불린다.

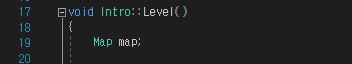
작업에대한 처리는 편할지 모르지만 move() 는 반복문에 의해 호출되므로 이와 나란히 있었으면 역할분담도되고 보기도 좋았을 것같다.

-Control -> Select ( int , int )



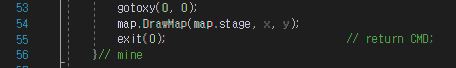
윗 단계에서 선언하여 참조하는 형태가 이해적인 부분에서 도움이 될 것 같다

-Control -> Select ( int , int )



다른 곳에서도 Map 을 필요로 했다. 싱글턴 기법이 생각난다.

-Control -> Select ( int , int )



코딩이 깊지 않고 new 도 사용되지 않아 memory lick 은 없었지만 도중에 종료선언은

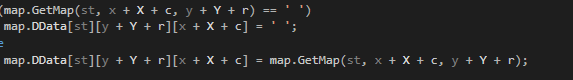
메모리 관리가 어렵지 않을까 생각했다.

- Search::Find()



이 함수가 불리기 이전에 이미 칸에 대한 확인이 모두 끝났으므로 굳이 조건문을 걸지 않아도 되었을 것이다. ( Contol::Select() 39번줄 – 지뢰인가?, Search::Find() 8번줄 – 숫자인가?)

- Search::Find()

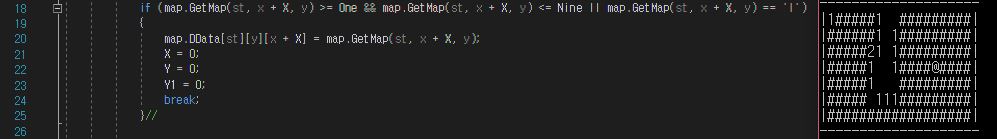


식별자의 역활이 정말 중요하다는 것을 실감했다. 주변 8칸에 대해 작업하려하는 부분이다.

식별자가 명확히 선정되던가 함수내에서 따로 y + Y + r 의 값을 받을 변수를 선언해줬으면 중복코딩과 가독성 두가지를 동시에 해결했을 것이다.

예시) 변수 = y + Y + r // DData[str][변수1][변수2] …

- Search::Find()



789

456

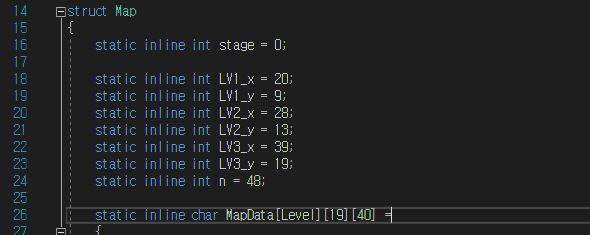
123

커서 5 기준 1,3,7,9 같은 모서리 부분에서 버그가 잦다.

함수 도중엔 X Y Y1 의 값이 변경될 수 있는데, X의 값으로 인해 좌표값이 어긋나서 그렇다.

위의 경우에는 X 가 +1 로 정작 자기위치보다 더 앞만 확인하고 break 로 나와버린다.

- Define::MapData[][][]

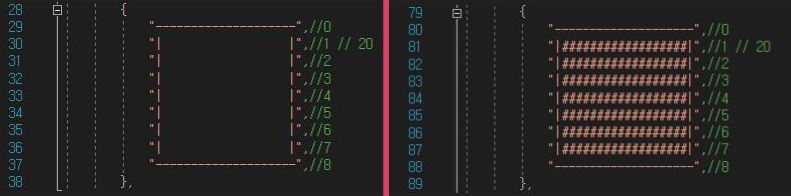


위 변수들은 지정해서 사용하려는 의도가 보인다.

그렇다면 const 는 어디갔을까.

또는 위의 enum 내 변수가 혼자 외롭게 있는데 같이 표현 가능했을 것 같다.

- Define::MapData[][][]



닮은 꼴의 두가지 형태 배열을 사용하여 GameManager처럼 처리를 담당 방법이다.

좌측의 배열은 처음에 지뢰를 설치하고 그위에 우측 배열로 덮어서 출력한다.

이후에 커서를 옳겨가며 그 좌표와 상호작용하며 우측의 ‘#’부분을 바꾸거나 지운다.

좌측 배열의 쓰임은 지뢰 설치와 지뢰와의 상호작용이다.

우측 배열의 쓰임은 출력용이다.

여기서 구조체인 struct 배열을 사용하여 위 두개로 나뉜 배열을 하나로서 사용가능 하고

지뢰 설치와 함께 모든 처리에 대한 정보를 저장할 수도 있었다고 생각한다.

- 식별자 interface

짧은 코드에 따라하기도 쉬웠지만 이해방향과 다른 식별자의 선정과 가끔 특별한 문법의 사용으로 순서와 로직에 대한 추론이 어려웠다.

class Search 에 모든 연산로직이 있는데 전반적으로 if , for 등의 난무가 아닌 상태값의 변수를 선언하기도 하고 함수로서도 좀더 분활시키는 즉, 식별자를 좀더 검토해보고 역할분배를 명확히 했으면 깔끔한 interface 를 유지할 방법이 있다고 생각함.

4. 좋은 점