Code Review - Minefind B

YJH ver

1. 구성

Define 게임 구성에 필요한 정보들이 정의 되어있다.

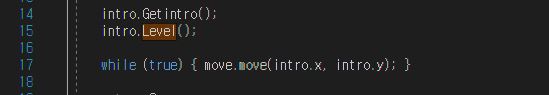
Intro 게임의 시작과 설정을 책임진다.  
Control 게임의 동작과 키 입력을 책임진다.  
Search 모든 연산이 이루어진다. GameManager 와 닮은꼴

2. 작동

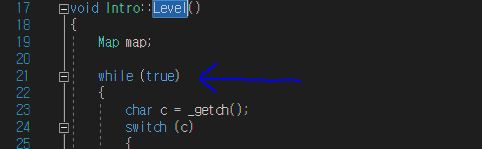
Uml 참조

3. 아쉬운점

- Main()

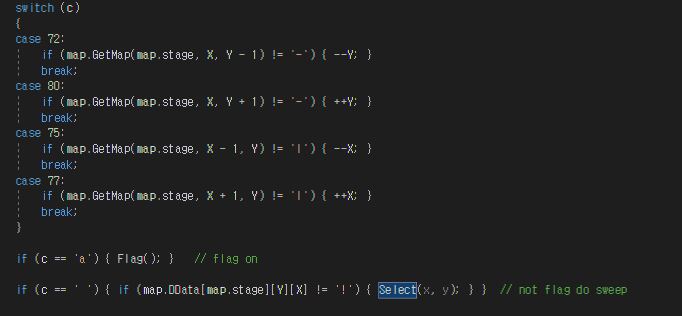


초반 도입부의 나눠진 루프들은 ( Level() 주목 )

  
일관성이 없어 식별자와 구성으로는 동작이 어떻게 이루어지는지 짐작할 수가 없었다.

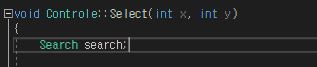
Level() 에도 반복문을 주거나 move() 도 내부적으로 반복문을 주었으면 했다.

- Control -> move( int , int )

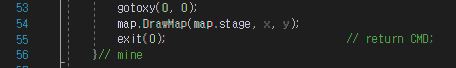


같은 키 값을 확인하는 단계인데 switch 와 if 가 나눠있다.

-Control -> Select ( int , int )



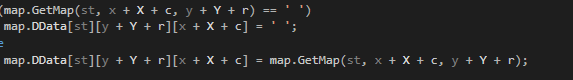
윗 단계에서 선언하여 참조하는 형태가 이해적인 부분에서 도움이 될 것 같다. (갑툭튀클라스...)



코딩이 깊지 않아서 memory lick 은 없었지만 도중에 종료선언은

메모리 관리가 어렵지 않을까 생각했다.

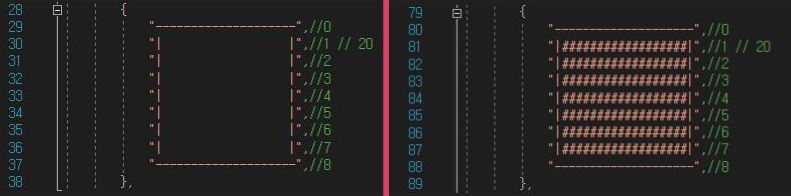
- Search::Find()



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

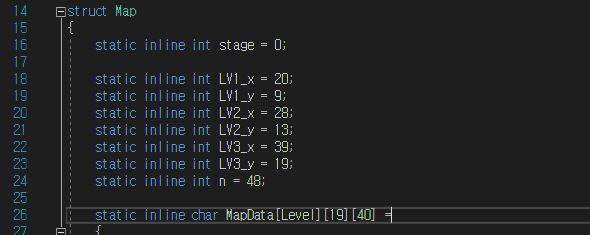
예시로 “변수 = y + Y + r “ 하여 DData[변수] 로 중복코딩과 가독성 두가지를 동시에 해결 할 수도 있다.

- Define::MapData[][][]



비어있는 맵은 이전에 언급한 Intro::Level() 외에는 사용하지 않는다. (이러한 맵은 3개다.)

- Define::MapData[][][]



위 변수들을 여러곳에서 사용하려는 의도가 보인다.

그렇다면 const 는 어디갔을까.

또는 enum 으로도 표현 가능했을 것 같다.

-식별자와 interface

짧은 코드에 따라하기도 쉬웠지만 사람의 이해방향과 살짝 어긋나있는 형식에

순서와 로직의 추론이 어려웠다.

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

4. 좋은 점