Code Review - Minefind D

HJH 0905

목차

1. Application::Update()

2. enum class 와 주석

3. SceneManager::GetTargetFPS()

4. SceneManager::KeyInput()

5. class Cell

6. GameManager

7. GameManager::Update()

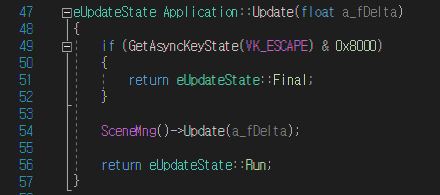
8. 로직 & 플레이

9. 긍정적인 평가

1. Application::Update()

키입력을 Update() 에서 처리하고 분배력 -1

도중에 정지하게 만들어 메모리 관리에 어려움 +1

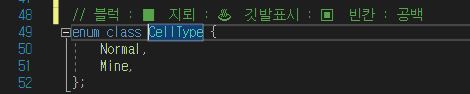


위와 상관없이 기본적으로 Lick 이 숨쉽니다.



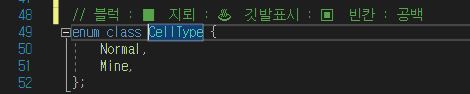
2. enum class 와 주석

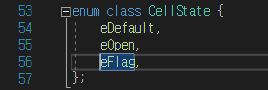
어떻게 출력되는지에 대한 주석은 매우 좋습니다만



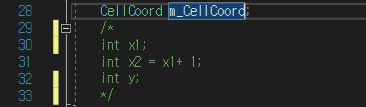
다만 어디의 누구와 호환되는지에 설명이 없습니다.

이곳에 있었다면 완벽한 주석이 될 수 있었습니다.





하나더



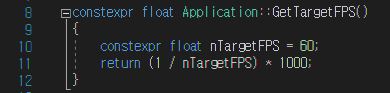
struct 임을 암시하기 위한 것 같지만

가독성에는 해가 된것 같습니다.

추가설명이 필요해 보입니다.

3. SceneManager::GetTargetFPS()

필요없는 프레임 연산 낭비



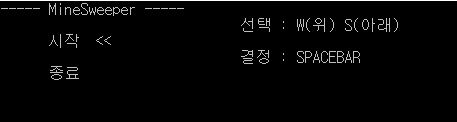
시간체크는 GameManager::Update() 에서 따로합니다.



피해자로 IntroScene::Render() 의 문자열입니다.

“ << “ 무수히 깜빡거립니다.

문자 뒤의 공백으로 지워졌다가 출력했다가 반복.

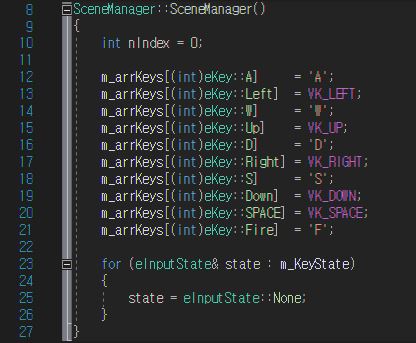


4. SceneManager::KeyInput()

복사 붙여넣기.

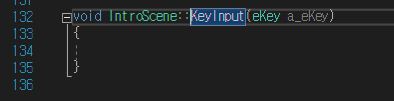
사용하는 키는 [ W, S, F, Space, Esc, 좌클, 우클 ] 7가지로,

코드자체는 확장성을 지녔지만 버려지는 정보들입니다.



당연하게도

KeyInput() 을 배신합니다..



5. class Cell

주관적이지만,

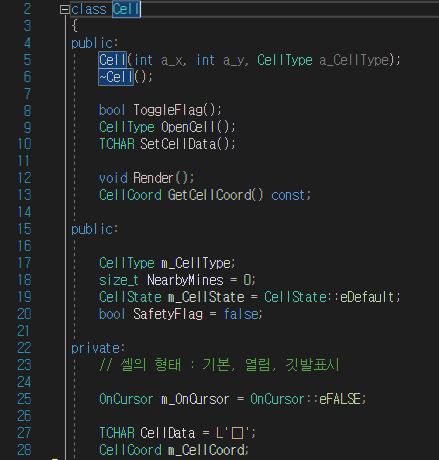
배보다 배꼽이 더 큰 느낌이드는 형태의 구성입니다.

GameManager 에서 해도 될 작업들을 가지고있고

이런 형태로 자료구조에 들어간다면 세세한 부분에서

메모리관리에 허술해 질 것 같기도 하고

용량은 불필요하게 많이 사용됩니다.



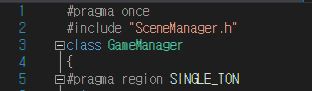
Tchar 로 가진 정보를 그래도 출력할 수 있는 점이 매우 좋았습니다.



6. GameManager

본격적으로 코딩을 시작한 모습이 상상이 됬습니다.

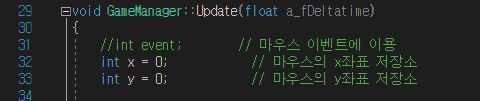
붙여넣기한 코드에는 아레의 “ SINGLE\_TON ” 문구는 없었으니까요.



이전 SceneManager 와 너무 차별되는 것 같습니다.

7. GameManager::Update()

Update() 역활 분배



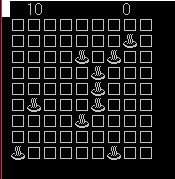
KeyInput() 에서 키입력에 따라 상태값을 바꾸고

이 곳에서 그 상태값을 기준으로 작동하게 하면

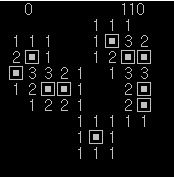
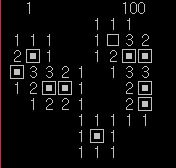
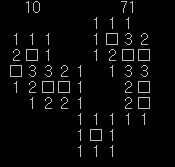
역활분배가 잘 이루어 졌을 것 같습니다.

8. 로직 & 플레이

Tchar 로서 이전 1byte 문자들과 차별화 되어 좋았습니다.



승리 조건 로직에 문제가있는데 일반적인 상황으로는 이길 수 없습니다.



이렇게 깃발을 꽂아야 됩니다.

이유는

FlagNum 과 FlagCellCount 이 같아야 하는데

깃발을 꽂아야지만 지뢰수가 감소 됩니다.



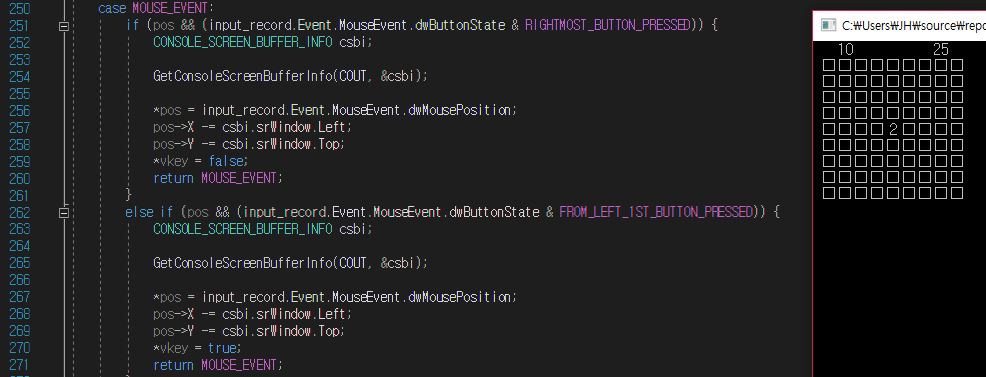


로직에서 깃발 대신에 open 시 수를 세서 비교만 하면 되므로

조금만 수정하면 될 것 같습니다.9. 긍정적인 평가.

GameManager

이 class 의 코드는 주관적으로 인상깊었습니다.



* 원래 구현하려했던 마우스 입출력.
* API 의 자연스런 사용.
* 코딩경력을 엿볼 수 있는 식별자와 인터페이스 구성.

무엇보다,

좀 더 분발해서 도달해야 할 단계가 바로 눈앞에 있었기 때문인 듯 싶습니다.

- End -