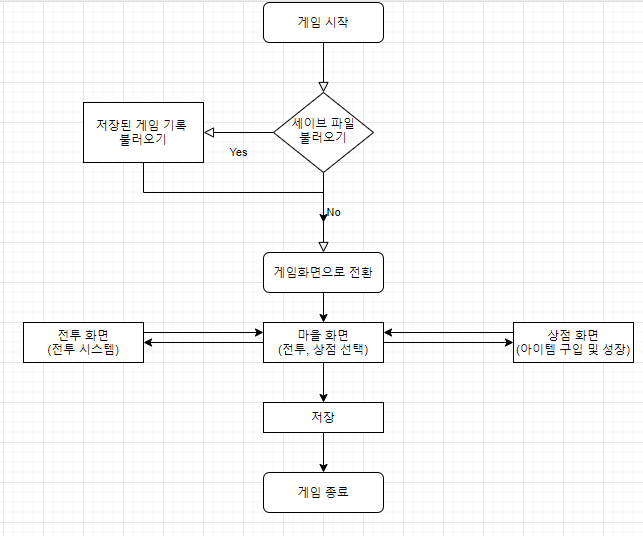
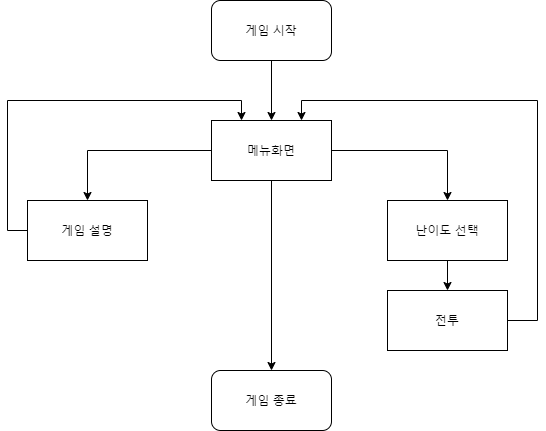
Cut the bullet 개발 문서

# **기획과 실제 개발과 차이**

기획 때의 흐름도



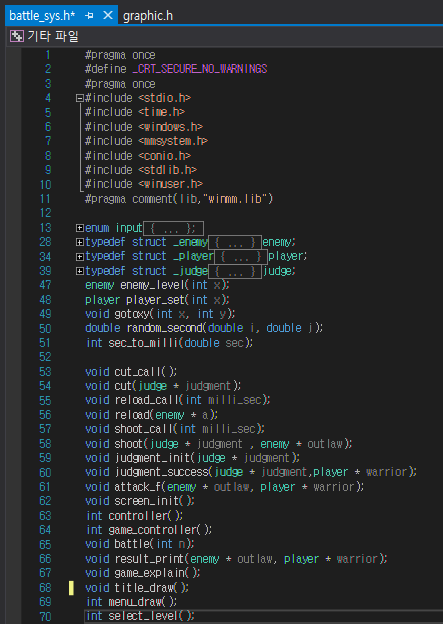
실제 개발된 게임 흐름도

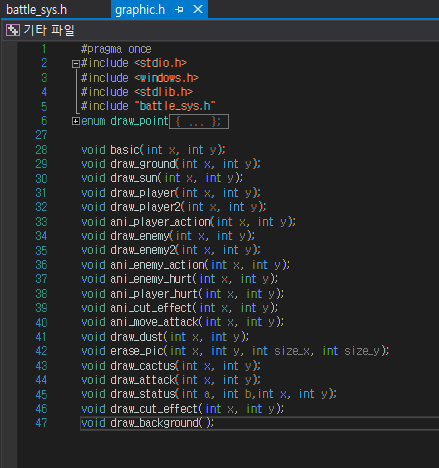


게임은 기존 텍스트 rpg에 전투시스템만 개선하고, 서부극의 무법자를 제압하는 무사의 컨셉이었다.

기획에서 세이브 기능, 마을 화면 ,상점 화면 ,아이템 등 다양한 요소를 집어넣고 싶었지만, 막상 구현한 것은 전투와 게임 종료 뿐이다…

# 개발된 소스 파일





[자세한 건 소스파일 참고]

# 개발 상세 기술

* 입력

함수를 보면 컨트롤러가 두 개가 있다. 아무리 봐도 부자연스럽다.

Controller()와 game\_controller()

처음 개발한 controller는 kbhit()와 \_getch()를 이용했다.

Kbhit()는 사용자가 입력을 감지하는 함수

\_getch()는 사용자의 입력을 받지만 엔터를 누르지 않고 바로 입력이 된다. 또한 화면에 입력을 표시하지 않는다.

하지만 중요한 문제가 있다.

게임에서 사격음을 듣고 반응하고 입력하여 적의 공격을 막아내는 연출을 의도하고 싶었다. 이때 컨트롤러가 말썽이었다.

선입력의 문제.

* 입력이 버퍼에 저장된다. \_getch() 가 바로 입력되는 것 같아 버퍼를 안 거치나? 라고 생각하지만 아니다. 버퍼에 저장되었다가 다음 호출 때 써 먹는다. 사격음을 듣고 반응해야 하는데 미리 키를 눌러 놓는다면 무조건 사격을 막아낸다.

그럼 버퍼를 비우면 되지 않는가?

c언어 함수로 입력 버퍼를 직접적으로 비우는 것은 함수는 찾아볼 수가 없었다. fflush(stdin)은 컴파일러마다 다르게 지원하며, getchar()함수를 이용한 비우기를 사용하면, 만약 선입력을 안했을 때. 그때 엔터를 눌러줘야 한다. 즉, 입력을 하려면, 입력 시작시 엔터를 누르고 나서 입력을 해야한다. 이 또한 의도한 바는 아니다.

그외에 다른 비우기 방법이 있지만, 엔터를 한번씩 눌러줘야 하거나 등등 뭔가 한가지씩 부족한 점이 있다.

그러다 winapi에 getasynckeystate()를 발견했다.

이것을 이용하면 호출 시점 이전에 입력이 되었어도 그것을 무시하고 호출된 이후의 입력만 받을 수 있다.

그것으로 개발된 게 game\_controller()함수.

근데 이미 controller로 구현해 왔다 보니, 전투 시스템 내에서만 적용하고

나머지는 나중에 수정해야지 하고 방치해 두었다.

그래서 입출력을 처리하는 controller함수가 두개나 있다.

* 그래픽 출력

그래픽 처리는 단순하다.

Gotoxy()는 출력 커서를 화면의 x,y 좌표에 바로 옮길 수 있다.

그 위에 printf 를 이용하여 화면을 덧그리는 방식을 사용했다.

문자를 출력하고 공백을 출력하면 사람은 문자가 지워졌다고 생각하니 이를 이용하면 애니메이션을 할 수 있다.

Draw\_xxxx, ani\_xxx등 함수는 xy좌표를 받아 해당 위치에 그린다.

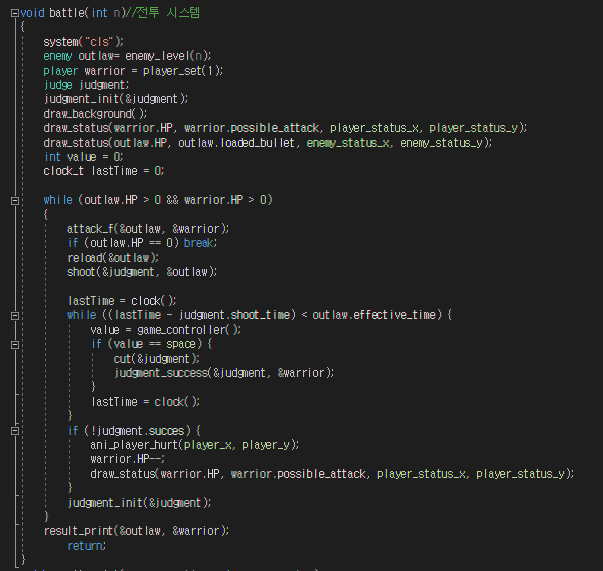
열거형Drawpoint를 이용하여 그리는 위치를 바꾸기 편하게 했다.

예상대로 그리는 화면이 빈번하지 않다 보니 이중 버퍼까지 활용할 필요는 없었다.

* 사운드

playsound함수를 사용했다.

* 전투 시스템 소스코드



Battle()(전투 시스템) 매개 변수 n은 난이도 이다.

난이도를 조절하면 적의 체력, 최대 장전 수, 총알 판정이 변한다.

원래 판정을 담당하는 구조체 judge는 플레이어의 반응 시간과 적의 공격시간을 받아 자신을 판정 함수에 넣어 결과를 반환하는 용도로 쓸려고 했으나 애초에 전투시스템 설계를 잘못했었다. 입력을 항상 받게 끔 설계 했어야 했는데, 그렇지 않았다. 그렇게 구조를 바꾸려다 보니 너무 크게 바꿔야하는 상황이어서, 구조를 조금 바꿔 적이 총을 쏘고 제한된 시간안에만 입력을 받을 수 있게 끔 바꾸었다. 그러다 보니 judge 구조체 내부의 선언된 값은 하나를 빼고는 다 필요가 없어졌다. 이렇게 되면 구조체를 선언할 필요가 없었는데 말이다. 물론 마감 시간을 지켜야 하기에 코드 수정은 안하고 방치해 두었다.

# 결과물 및 피드백





결과물이 매우 안 좋다는 것은 아니지만 만족스럽지는 않다.

왜 이런 결과가 나왔을까?

1. 제한된 시간

누군가에게 프로그램을 만들 때 무한한 시간을 준다면, 고품질 프로그램을 만들어 낼 수 있을 것이다.

하지만 무한한 시간을 줄 수도 없으며, 기업 입장에서는 그것은 곧 돈이기에 어느 정도 개발 기간을 제한 할 수 밖에 없다.

다가오는 마감 기간에 급해지기 시작하면, 기존의 규칙을 무시하게 된다.

그러다 원하는 기능만 작동하는 함수를 짜다가, 이젠 원하는 기능조차 못 구현하고 버그만 터진다. 머리를 쥐어짜서 버그를 수정하고 고쳐서 작동시킨 뒤 다시 코드를 보면 스파게티가 되어 있다.

1. 엉성한 기획

이번 프로젝트의 경우 기획서를 만들긴 했지만 엉성하다.

그나마 나은 점은 개인 프로젝트라 기획서 의존이 크게 있지는 않다.

그도 그럴 것이 게임의 컨셉, 연출 등이 자신 머릿속에 있으니 말이다.

물론 이 뜻이 기획을 대충해도 상관없다는 뜻은 아니다.

내가 원하는 대로 개발하는 것이 효율적인 개발을 의미 하는 것은 아니기 때문이다.

완벽하게 기획할 필요는 없지만 기획단계에서 필요한 함수 라든지 라이브러리 등등 좀 더 구체적으로 알아 보았으면, 얼마나 시간을 절약할 수 있었을까.

구체적인 기획은 시간을 아낄 수 있게 하고, 이는 프로그램 품질에 영향을 줄 수 밖에 없다.

1. 기술적 지식의 부재 및 설계 오류

이번 경우에만 해도 \_getch를 쓰다가 getasyckeystate를 쓰기까지 상당한 시간을 소비했다. 이미 알고 있는 사람들이라면 우스울 정도지만, 모르는 사람 입장에서는 난감할 뿐이다. 단순히 함수 하나 모른다고 시간을 잡아먹은 게 매우 크다.

또한 전투 시스템 개발 과정에서 사용자의 입력을 항상 받을 수 있게 끔 설계 했어야 했는데, 잘못 설계했다. 이러한 과정이 게임의 논리가 흔들리고 이를 수정해야할 수 밖에 없었다. 이 부분에서도 개발 일정의 지연을 가져왔다고 생각한다.

* 다른 원인도 있지만 결국은 시간이다. 제한된 시간 내에 얼마나 효율적이고, 완성도 높게 개발할 지. 시간관리가 프로그램 품질을 좌우하지 않나 싶다.