<자료구조 및 실습 기말프로젝트 최종 보고서>

2071185 김민기

2171255 김종윤

1) 주제

-> Gallag 게임을 모티브로 하여 자료구조를 사용하여 게임을 만들었습니다.

2) 프로그램 목적

->히어로를 움직이면서 총알을 발사해 적을 없애고 보스를 없애면서 점수를 올리는 게임을 만들었습니다. 여러 구조체와 자료구조때 배운 내용을 가지고 게임을 구현하면서 코딩 능력 및 협업 능력을 향상하려고 하였습니다.

3) 프로그램 기능

-> 적이 생성되어 아래로 내려오고 보스가 생성되어 총알을 발사하고 히어로는 이것을 피해 총알을 발사하여 적을 없애며 점수를 올려 게임을 진행하도록 하였습니다. 점수를 저장하고 이 점수를 링크드 리스트를 활용하여 점수를 정렬하고 마지막에 점수 부분을 보이도록 하였습니다.

4)프로그램 동작 결과텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

이렇게 보스가 생성되고 옆에는 보스의 체력과 히어로의 점수 그리고 히어로의 체력등을 보여주었습니다.

5)발생문제 및 해결방법

김민기

1)적이 움직이는 것을 만들기

->아래로만 떨어지는 움직임 말고 더 많은 움직임을 만드는 것이 어려웠습니다. 경계값을 설정하여 해당 움직임이 자유롭게 설정하였습니다.

2) 적과 히어로의 충돌

-> 충돌 함수를 만들다보니 대부분의 구현이 비슷하게 돌아가는 것을 파악하였습니다. 논리적인 부분을 먼저 생각하고 구현하여 히어로의 크기를 생각하여 적와 만나면 적을 지우고 히어로의 체력이 깎이도록 하였습니다.

3) 오류를 잡는 과정

-> 디버깅을 해보면서 적게는 1개부터 많게는 연계되어 발생하여 20개정도의 오류가 발생하였었습니다. 이 부분은 지피티를 이용하여 도움을 얻거나 구글등에서 만들어진 갤러그 게임등을 보면서 해당 코드를 만들고 오류를 수정하였습니다.

4)링크드 리스트를 활용해 점수 나오게 하는 부분

-> 해당 자료구조를 사용하여 우리의 프로그램에 적용하여야하는데 이 부분을 어떻게 적용해야하는지에 대한 고민을 많이 하였습니다. 수업 중에 배운 코드를 수정하여 만들고 포인터를 다시한번 공부해보면서 점수를 정렬하거나 하는 것을 만들면서 큰 도움이 된 것 같습니다.

김종윤

1)히어로 총알과 적과의 충돌

-> 문제: 히어로의 총알과 적이 충돌했을 때, 충돌을 정확히 감지하지 못하거나 충돌 처리가 부정확하게 이루어짐.

해결 방법: 히트박스(Hitbox) 개념을 적용하여 충돌을 감지.

히트박스는 총알과 적의 위치 및 크기를 기준으로 사각형 영역을 설정한 뒤, 충돌 여부를 계산.

2) 타이머 구현

문제: Snake 게임에서 사용했던 타이머와 유사하게, 부드러운 움직임을 구현하기 위한 타이머가 필요했으나 초기 설정에서 매끄럽지 못한 움직임이 발생.

해결 방법: GetTickCount를 사용해 시간 간격을 계산하여 부드러운 움직임 구현.

이전 프레임과 현재 시간의 차이를 계산하여 게임 요소의 위치를 업데이트.

if (currentTime - previousTime >= interval) 조건을 통해 움직임 간격을 조절.

3)보스 구현 과정에서 논리적 오류

문제 1: 보스가 죽었을 때 화면에서 지워지지 않고 계속 움직이는 현상 발생.

원인: 보스의 존재 상태(boss.exist)를 False로 설정했으나, 보스를 지우는 함수가 호출되지 않거나 이후에도 boss.update() 함수가 계속 실행됨.

해결 방법:

보스가 죽었을 때 erase\_boss() 함수를 호출하여 이미지를 제거.

boss.exist가 False일 경우 boss.update() 함수가 실행되지 않도록 조건문 추가.

문제 2: 보스가 시작하자마자 생성되는 문제 발생.

원인: 보스 생성 타이머의 초기값(for\_bossUpdate)이 0으로 설정되어 즉시 보스가 생성됨.

해결 방법:

초기값을 현재 시간(currentTime)으로 설정.

if (currentTime - for\_bossUpdate >= BOSS\_UPDATE\_INTERVAL) 조건으로 일정 시간 이후에 보스가 생성되도록 수정.

4. 아이템 구현 과정에서 총알 및 아이템 갯수 제한

문제: 총알 및 아이템의 무제한 사용으로 게임 밸런스가 깨지는 문제 발생.

해결 방법: active 변수를 도입하여 상태를 제어.

총알이나 아이템이 활성화된 경우(active=True)에만 동작하도록 구현.

사용 후에는 active=False로 설정하여 다시 활성화되지 않도록 제어.

이를 통해 아이템의 갯수와 사용 가능 여부를 정밀하게 조정.

위 해결 방법들을 적용하면서 부드럽고 논리적인 게임 진행이 가능하도록 최적화되었습니다.

6) 조원별 역할분담 내역 및 상호평가

|  |  |  |
| --- | --- | --- |
| 평가자 | 김민기 | 김종윤 |
| 김민기 | 50 | 50 |
| 김종윤 | 50 | 50 |

느낀점)

김민기(mhn06159@naver.com): 프로젝트를 하면서 코드를 이렇게 많이 적어본 적이 많이 없었는데 수행하면서 모르는 부분을 찾아보고 구글링하고 지피티를 이용하여 도움을 얻고 이것을 나의 것으로 만들어가는 과정이 너무 좋았습니다. 9월의 나를 생각하면 차이가 많이 날 정도로 코딩 실력이 증가한 것 같습니다. 1000줄은 안되지만 동료가 한 코드를 보면서 아이디어를 얻을 때도 있었고 도움을 청하거나 도움을 주는 과정을 통해서 협업에 대한 이해가 많이 늘어난 것 같습니다.

김종윤(kenny0428@naver.com):

예상치 못한 버그를 해결하며 체계적인 접근과 기존 경험을 활용하는 능력의 중요성을 깨달았고, 모듈화를 통해 코드의 가독성과 유지보수성을 개선하며 효율적인 설계의 중요성을 느꼈습니다.

참고사이트: <https://dongkyung.tistory.com/entry/C%EC%96%B8%EC%96%B4%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%B4-%EB%A7%8C%EB%93%A0-%EC%8A%88%ED%8C%85%EA%B2%8C%EC%9E%84-1>

<https://lsj29188.tistory.com/13>

<https://m.blog.naver.com/stizms/220187088276>

<https://chatgpt.com>