H1 최종 보고서

팀명 : GitHub 어렵단 [말]이야
팀원: 조홍준, 강승우, 김수엽, 김한민

1. 프로젝트 개요

프로젝트 「H1」(이하 「H1」)은 텍스트 기반 경마 육성 시뮬레이션 게임으로, 레트로한 CMD 인터페이스와 아스키 아트의 독특한 매력을 담아냈다. 게임의 핵심은 플레이어가 다양한 선택지를 통해 최고의 경주마를 육성하고, 최종 레이스에서 정상에 오르는 여정을 담고 있다. 단순한 게임을 넘어 텍스트와 아스키 아트의 창의적인 융합으로 플레이어의 상상력을 자극하고, 복잡한 육성 시스템의 깊이를 재미있게 풀어낸다.

2. 개발 동기 및 목표

「H1」은이차원 배열과 for 반복문을 활용한 맵 렌더링 아이디어에서 받은 영감을 바탕으로, 텍스트와 아스키 아트가 어우러진 레트로 스타일의 독특한 게임을 개발하고자 했다. 플레이어들은 말의 특성을 전략적으로 파악하고 육성하는 과정에서 전략게임의 재미와 육성의 재미, 경쟁의 재미를 느낀다.

개발과정에서 OOP 원칙 적용의 단순한 학습용 예제를 넘어서는 실전 설계 경험을 추구했다. 객체지향 프로그래밍의 실제 적용과 게임소프트웨어 전공 역량 강화를 목표로 했으며, 팀 프로젝트를 통한 게임 개발 경험을 쌓는 것이 목적이다.

최종적으로 게임성과 OOP 학습 효과를 동시에 추구하고, 단순한 예제 연습을 넘어 실제적이고 창의적인 소프트웨어 개발 경험을 얻고자 했다.

3. 개발 환경 및 사용 도구

개발 언어: C++
IDE: Visual Studio
협업 도구: Discord, GitHub

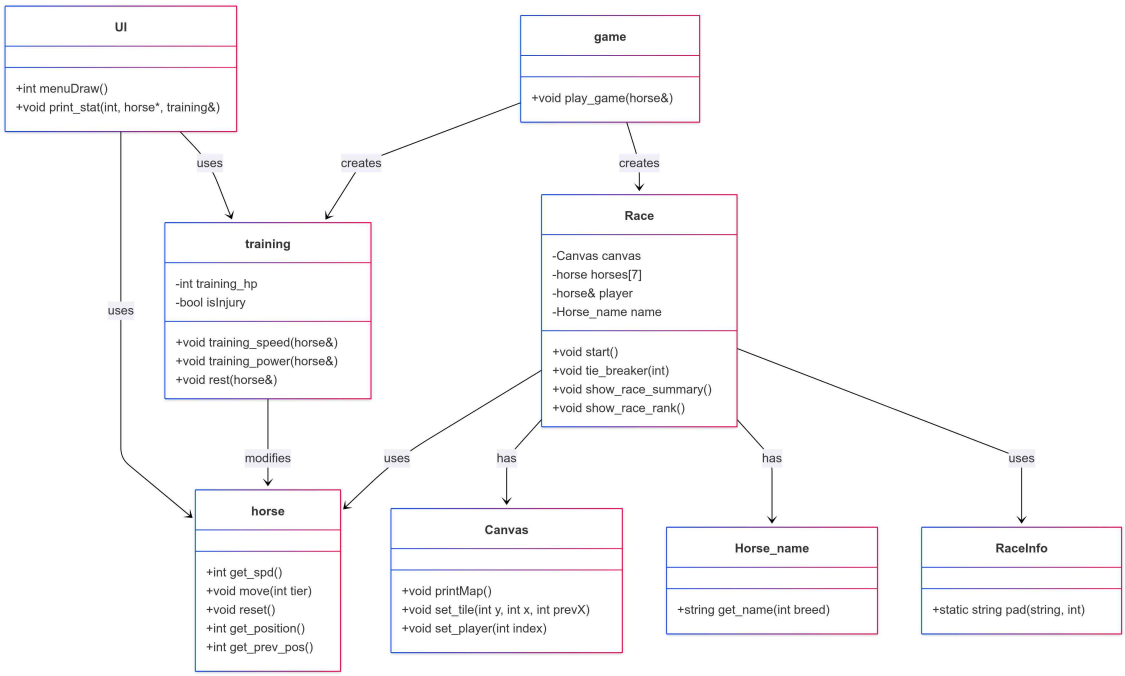
4. 전체 구조 및 클래스 설계

「H1」은 C++의 객체지향 프로그래밍 개념을 바탕으로 구성되었으며, 파일별로 명확한 역할 분담과 책임을 지도록 설계했다.

1) 전체 구조 및 클래스 설계

파일명	주요 역할 및 책임
main.cpp	프로그램 진입점. 콘솔 초기화, 메뉴 및 튜토리얼 실행, 말 선택, 게임 루프 시작. 예외 처리로 재시작 기능 포함.
game.h	play_game()을 통해 전체 게임 사이클(훈련 → 레이스 → 보상) 제어. 각 시점의 흐름을 조율함.
horse.h	horse클래스 정의. 말의 능력치, 스타일, 이동 알고리즘, 이전 레이스 등수 및 상태 정보 관리.
horse_name.h	CPU 말을 위한 무작위 이름 생성 클래스. 중복 방지를 위한 풀 관리 포함.
training.h	training클래스 정의. 훈련 시 능력치 증가, 체력 관리, 부상 로직 등 훈련 시스템 전반 구현.
canvas.h	Canvas클래스 정의. 실시간 레이스 UI를 출력. 말 위치 및 트랙 시각화.
race.h	Race클래스 정의. CPU 말 생성, 레이스 진행, 등수 판정, 보상 처리까지 전체 경주 로직 구현.
race_info.h	텍스트 정렬 보조 기능 제공. 출력용 문자열 폭 계산 및 정렬 함수 포함.
UI.h	사용자 인터페이스 처리. 말 선택, 훈련 선택, 스탯 출력 등 사용자 입력 및 화면 구성.
tutorial.h	튜토리얼 출력 함수 집합. 게임 규칙과 시스템을 사용자에게 안내.
final_reward.h	최종 등수 결과에 따른 아스키 아트 보상 화면 구현. 게임 종료 연출 포함.

2) 클래스 다이어그램



3) 캡슐화와 추상화 사용 예시

개념	적용 예시	설명
캡슐화	horse클래스의 능력치 변수 (spd, pow, sta, guts)	private으로 선언하고, 외부에서는 getter/setter를 통해서만 접근할 수 있도록 함
	training 클래스의 상태 변수 (training_hp, isInjury)	외부 접근을 차단하고, 내부 로직을 통해서만 상태를 제어함
추상화	horse::move(int tier)	내부적으로 다양한 계산 로직이 존재하지만, 외부에서는 "이동"이라는 동작만 호출
	training_bass()	훈련의 공통 로직을 하나의 함수로 추상화하여 각 훈련 함수에서 재사용함
	Canvas::printMap()	내부 맵 표현 방식과 무관하게 화면에 시각적 결과만 제공하는 함수 구조

4) 상속과 다형성을 사용하지 않은 이유

이번 프로젝트에서는 클래스 간의 상속 구조를 사용하지 않았으며, 결과적으로 다형성도 적용되지 않았다.

초기 설계 단계에서는 horse 클래스를 상속하여 각 말의 주행 스타일(도주, 선행, 선입, 추입)에 따라 개별 클래스를 만드는 구조를 고려했으나, 실제 구현 과정에서 스타일 간 차이는 'breed_mod[3]'와 'stat_mod[3]' 두 배열 값 외에는 거의 없다는 점을 파악하였다.

이처럼 동작 방식에서 차이가 없고, 단순한 수치 차이에 불과한 경우에는 상속보다는 하나의 클래스 내부에서 처리하는 방식이 코드의 복잡도를 낮추고 유지보수에도 유리하다고 판단하여, 불필요한 상속을 지양하고 한 클래스 내에서 조건 분기와 데이터 설정을 통해 말의 주행 스타일 차이를 구현하는 구조를 선택하였다.

5. 게임 플로우 및 주요 기능, 조작 방식

1) 전체적인 게임 흐름

① 프로그램 실행
② 게임 제목 및 아스키 아트 출력
③ 튜토리얼 or 바로 시작 선택
④ 커스텀 경주마 생성 or 미리 만들어진 말 선택
⑤ 훈련 화면 → 1~5 훈련 중 택 1
⑥ 능력치 증가 결과 확인
⑦ 레이스 시작 → 실시간 진행 UI
⑧ 도착 순서 및 도착 시간 출력
⑨ 보상 화면 출력 → 다음 단계 진행 or 게임 종료

사용자 입력은 단순하지만, 게임 내 흐름에 명확히 반영되어 사용자 경험 측면에서도 직관적인 구조를 갖추고 있다.

2) 주요 기능 및 시연 포인트

- **훈련 시스템:** 네 가지 훈련(스피드, 파워, 지구력, 근성)에 따라 서로 다른 능력치가 증가하며, 체력과 부상 확률을 고려한 전략적 선택(휴식)이 필요
- **캔버스 UI 출력:** 실시간으로 달리는 말의 위치와 속도가 출력되어 경기 상황을 시각적으

로 확인 가능

- **레이스 로직:** 말의 능력치를 기반으로 이동 거리를 계산하며, 소수점 단위까지 고려한 정밀한 등수 판정 구현
- **결과 출력 및 보상 시스템:** 각 말의 도착 시간과 순위가 화면에 표시되며, 1~3등까지는 차등 보상을 받고 4등 이하일 경우 게임이 재시작됨

3) 조작 방식

- 사용자는 키보드 숫자 입력(1~5)과 Enter 키만으로 게임을 진행
- 훈련 화면에서 원하는 훈련 번호를 입력하면 해당 능력치가 증가하고, 결과가 즉시 출력됨
- 레이스 시작 시 Enter를 누르면 실시간 경주가 진행되며, 도착 결과에 따라 등수와 보상이 자동 출력됨
- 사용자는 이후의 선택지(다음 경기 진행 or 재시작)를 입력을 통해 결정

6. 역할 분담 및 팀 구성

팀원	역할 분담	구현 파일
강승우	훈련 클래스, 훈련 진행 흐름, 말 클래스, 켄버스 클래스 (레이싱 화면 출력) 구현	horse.h, training.h, canvas.h
김수엽	게임 시작 및 재시작 로직 구현, 버그 픽스, 아스키 아트와 메인 로직 연결성 강화	main.cpp, game.h
김한민	UI 및 아스키 아트 디자인, 사운드 출력	final_reward.h, UI.h
조홍준	게임 기획, 레이스 클래스, 이동 로직, 동석차 판정 로직 디자인 및 구현	race.h, race_info.h, tutorial.h, horse_name.h

7. 구현 일정 및 일정 관리

1) 처음 계획 vs 실제 진행 비교

주차	최초 일정	실제 진행 현황
1주차	트랙 및 경기 데이터 구조 구성, 기본 게임 루프 제작 (턴제 흐름), 말 정보, 훈련 결과 출력 기능 구현	1주차, 2주차 내용과 동일. 3주차의 "시즌(3년) 흐름 구성, 게임승리 및 게임 오버 기능 구현", 4주차의 "사용자 입력 예외 처리"까지 완성
2주차	말 특성과 능력치에 따른 이동 로직 설계, 훈련 → 경기 흐름 연결 및 전체 구조 정리	
중간 발표		
3주차	경기 결과 판정 및 순위 결정 로직 완성, 시즌(3년) 흐름 구성, 게임승리 및 게임 오버 기능 구현	사운드 및 아스키 아트 추가, 경기 결과 판정 및 순위 결정 로직 구현, 튜토리얼 추가, 이동 로직 수정
4주차	텍스트 UI 정리 및 안내 메시지 개선, 밸런스 조정 및 예외 처리, 발표자료 제작, 시연 준비 및 리허설	사용자 편의성 개선, 성장 밸런스 조정, 발표 자료 제작, 시연 영상 녹화

8. 주요 기능 설명

아래에서는 이번 프로젝트의 주요 기능인 horse::move(), Race::tie_breaker(), training::training_bass() 함수에 대해, 각 기능의 역할과 함께 소스 코드를 설명한다.

1) horse::move(int tier)

말이 한 턴 동안 이동할 거리를 계산하는 함수로, 현재 위치한 구간(초반/중반/후반)에 따라 사용하는 능력치와 보정값이 달라진다. 주행 스타일에 따라 breed_mod, stat_mod 값이 적용되며, 난수 요소를 포함한 속도 계산을 통해 매 턴 이동 거리를 유동적으로 변화시킨다.

```
int seg = position(); // 현재 구간: 0=초반, 1=중반, 2=후반

// 기본 속도 계산 (스피드 기반)
double base_speed = (static_cast<double>(spd) / BASELINE) + 2.0 + (rand() % 11 / 10.0);

// 구간별 능력치 보정
double stat_ratio = (static_cast<double>(stat_by_section[seg]) / BASELINE) * stat_mod[seg];

// 최종 이동 거리 계산
double move_distance = ((base_speed + stat_ratio) * breed_mod[seg]) * correction + decimal_point;

decimal_point = fmod(move_distance, 1.0);        // 소수점 저장
curr_pos += static_cast<int>(move_distance);     // 정수 부분만 반영
```

2) Race::tie_breaker(int rank)

같은 턴에 결승선을 통과한 말들 간의 우열을 가리기 위한 함수. 위치와 소수점 누적값을 합산하여 정확한 도착 거리를 계산하고, 이를 기준으로 내림차순 정렬하여 등수를 부여한다.

```
for (int i = 0; i < HORSE_COUNT; i++) {
    if (horses[i].get_rank() == 0 && finished[i]) {
        double total = horses[i].get_position() + horses[i].get_decimal_point();
        lap_time_set(total, i); // 시간 계산
        list.emplace_back(total, i);  // 거리, 말 번호
    }
}

sort(list.rbegin(), list.rend()); // 내림차순 정렬

for (int i = 0; i < list.size(); i++) {
    int num = list[i].second;
    horses[num].set_rank(rank + i + 1); // 동순위 없이 순위 배정
}
```

3) training.h – training_bass()

훈련의 공통 기반 함수. 체력에 따라 부상 여부를 판단하고, 성공 시에는 메인 능력치 m, 서브 능력치 s를 증가시킨다. 체력이 낮을수록 부상 확률이 높아지며, 극한 상황(체력 ≤ 10)에서는 훈련 효과가 2배가 된다.

```
if (rand() % 101 < injury_percent(h)) {
    isInjury = true;
    // 능력치 10% 감소
    m_sta[0] = static_cast<int>(h.get_spd() * 0.1);
    ...
    training_hp = 100;
    return;
}
```

```cpp
if (training_hp ≤ MIRACLE) {
    m = (rand() % 41 + 20) * 2;
    s = (rand() % 11 + 10) * 2;
    std::cout << "극한의 상황에서 훈련을 성공했습니다!\n훈련 경험치 2배!\n";
} else {
    m = rand() % 41 + 20;
    s = rand() % 11 + 10;
    std::cout << "훈련 성공!\n";
}
training_hp -= TRAINING_MHP;
```

9. 프로젝트 진행 중 발생한 문제와 해결 방법

1) 아스키 아트 제작과 사운드 선택에 많은 시간 소요

아스키 아트를 제작하고, 배경 사운드를 선택하는 과정에서 예상보다 많은 시간이 소요되었으나, 팀장과 담당자 간의 빠른 의사소통을 통해 효율적으로 조율하며 문제를 해결했다.

2) 팀 프로젝트에 익숙하지 않아, 역할 분배의 어려움

초기에는 팀 프로젝트 경험이 부족해 역할 분담에 어려움이 있었으나, 전체 구조를 기준으로 임의로 업무를 분배하고, 문제 발생 시 유동적으로 재조정했다.

유동적 재조정으로 인해 작업한 파일이 겹칠 경우, Branches를 통해 작업물을 분리하였고 파일 병합은 전체 흐름을 이해하고 있는 팀장이 주도적으로 수행하였다.

정보의 부족으로 인한 개발 방향성에 의구심이 들 경우, chatGPT 등의 생성형 AI를 통해 아이디어를 얻으며 방향을 잡았다.

3) 분업 중에 발생하는 코드 의존성 문제

작업을 분리하는 과정에서 불가피하게 코드 간 의존성이 발생했으며, 이를 해결하기 위해 컴파일 에러가 발생하지 않도록 빈 헤더파일, 클래스, 멤버 함수 등을 미리 정의했다.

또한 코드 작성 시 각주를 적극적으로 활용하고, 함수명과 자신의 작업내용을 팀원들과 상시 공유하는 문화를 정착시켰다.

4) 객체지향 프로그래밍 구조 설계의 어려움

효율적인 구조 설계를 위해 팀 단체 채팅방에서 의문이 생기는 코드나 알고리즘에 대해 함께 토의하고 코드 리뷰를 통해 더 나은 방식으로 개선했다.

5) GitHub 사용 미숙

초기에는 GitHub 사용에 어려움이 있었지만, 반복적으로 사용하면서 익숙해졌고 문제 발생 시 검색과 문서를 통해 원인을 분석하고 해결하였다. 대표적으로 문서를 Push했는데, 파일이 깨져, UTF-8로 다시 인코딩하여 문제를 해결했다.

10. 향후 개선 방향

시간이 더 주어졌다면, 다음과 같은 요소들을 추가하여 게임의 완성도와 재미를 더욱 향상할 수 있었을 것이라 생각한다.

1) 코드 구조 개선

현재 UI.h, tutorial.h, final_reward.h 같은 파일들은 클래스를 사용하지 않았다. 개발 초기에는 단순한 기능만 담당하고, 주로 텍스트 출력만 수행하기에 굳이 클래스를 도입하지 않아도 된다고 판단했다. 하지만 오히려 그렇게 단순한 역할이기 때문에, 명확한 책임을 부여하는 의미에서 클래스를 사용하는 것이 더 적절하지 않았을까 하는 생각이 들었다.

Race.h 파일의 경우, 레이스와 관련된 거의 모든 기능을 담당하고 있다. 경주마 정보 출력, 레이스 진행, 보상 처리 등 레이스와 관련된 다양한 기능이 한 클래스에 통합되어 있으며, 이를 좀 더 세분화하고 분리했다면 'SOLID' 원칙에 더욱 부합했을 것이라는 시사점을 얻게 되었다.

이러한 구조적 고민들은 개발 당시에는 미처 고려하지 못했지만, 프로젝트가 마무리되고 피드백을 통해 돌아보면서 개선의 여지가 있다고 느끼게 되었다.

2) 경주 몰입도 향상

현재는 도착 시점에 등수가 판정되지만, 경주 도중 실시간 등수를 출력하면 몰입감과 긴장감을 높일 수 있다. 또한, 경기 중 스킬 사용이나 전략 선택 등으로 경주에 직접 관여하게 하면 플레이어의 재미가 향상될 것이다.

3) 스토리 및 진행 방식의 다양화

말과 트레이너 간의 서사나 배경 스토리를 추가해 게임의 감정적 연결을 강화하면서, 단조로운 선형 구조를 벗어나, 자유로운 레이스 선택 및 도전과제 기반 진행 방식을 도입하면 전략적 선택의 폭을 넓히고 리플레이 가치를 높일 수 있다.

4) 육성 요소 확장

육성과정에서 랜덤 이벤트를 등장시키고, 경주마의 고유 스킬, 장비 시스템 등을 통해 육성에 전략성과 변수를 추가할 수 있다. 더불어, 날씨, 경주에 큰 영향을 주는 컨디션 시스템을 도입하면 경기를 단순한 능력치 싸움이 아닌 복합 전략 요소로 승화시킬 수 있다.

5) 트랙 및 환경 다양화

기존의 직선형 트랙 대신 원형 트랙이나 다양한 지형(예: 잔디, 더트)을 구현하면, 말의 스타일과 능력에 따른 차별화된 플레이 경험이 가능할 것이다.

11. 결론

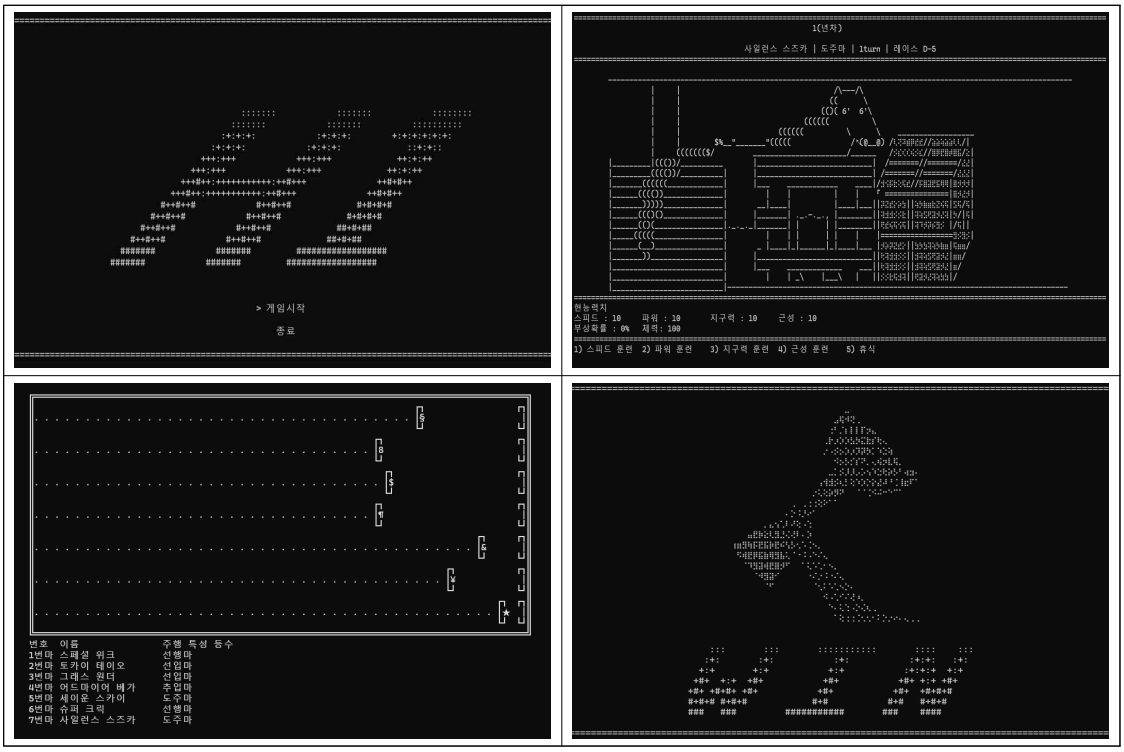
이번 프로젝트는 객체지향 프로그래밍 기반으로 말 육성과 경주 시뮬레이션 시스템을 구현하고, 아스키 아트를 통해 시각적인 재미까지 더한 콘솔 게임을 개발하였다. 프로젝트 진행 과정에서 협업 방법, 역할 분담, 코드 통합 등의 여러 도전을 겪었고, 이를 해결하며 팀워크와 개발 실력을 함께 향상할 수 있었다. 비록 제한된 시간과 인원으로 인해 구현하지 못한 기능들도 존재하지만, 기본적인 육성 시스템, 레이스 로직, 보상 구조를 통해 게임의 완성도 있는 흐름을 만들어냈다고 판단한다.

이번 프로젝트를 통해 단순한 기능 구현을 넘어 사용자 경험, 협업 효율, 구조 설계 등 다양한 측면에서 배움을 얻었다. 앞으로의 프로젝트에서는 이번 경험을 바탕으로 이번 경험을 바탕으로 팀 간 의사소통과 업무 지시, 보고를 더욱 원활히 하여, 더욱 뛰어난 게임을 개발할 수 있는 초석을 다졌다고 자신한다.

끝.

- 부록 -

「H1」 주요 인게임 화면



GitHub, YouTube Qr-cord