H1 최종 발표

Github 어렵단 [말] 이야

강승우 김수엽 김한민 조홍준

프로젝트 주제 & 개발 목표

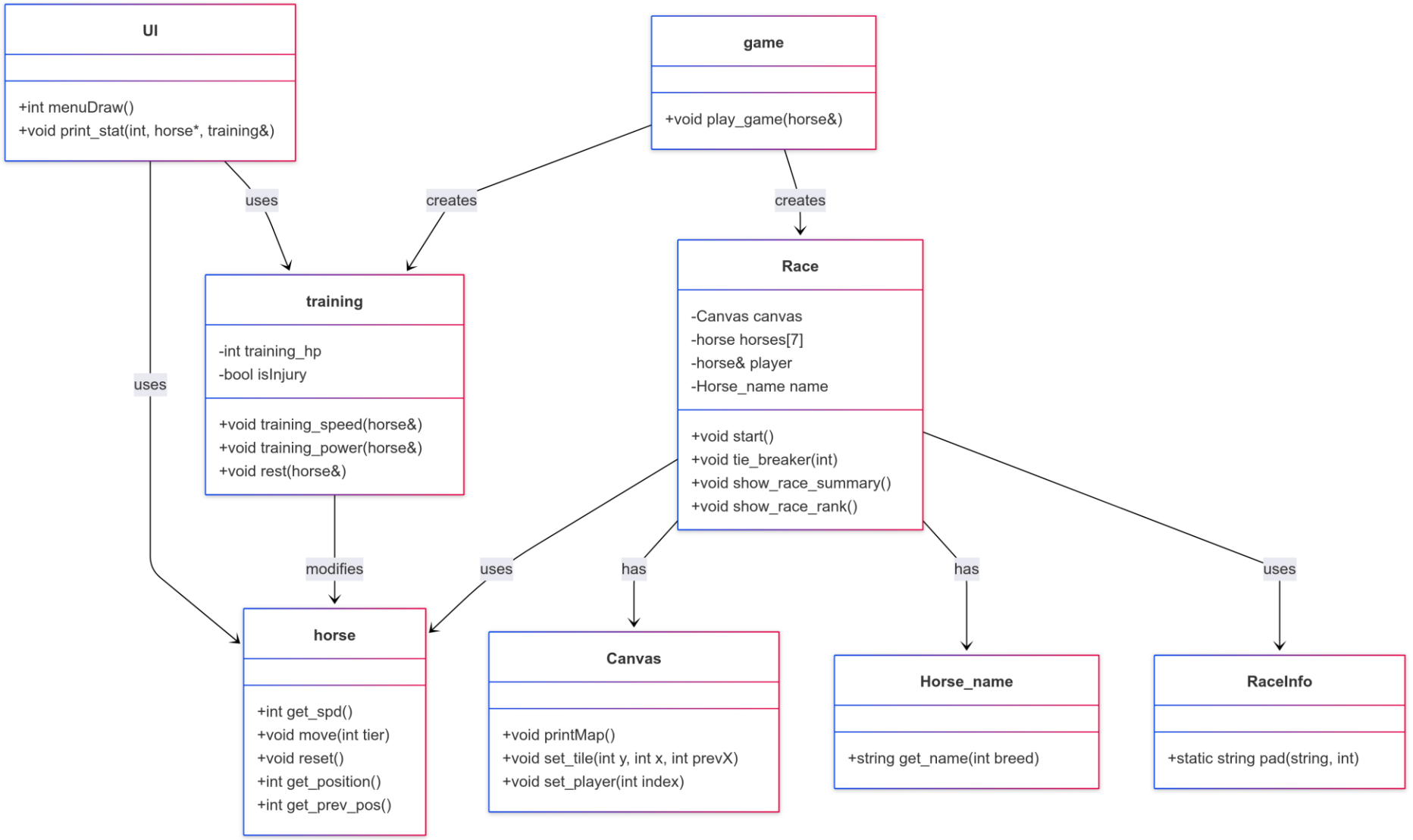
주제

→ 육성의 재미 + 경마의 짜릿함 + 아스키 아트

개발 목표

→ 객체지향 실전 적용

클래스 다이어그램


UI
+int menuDraw()
+void print_stat(int, horse*, training&)
game
+void play_game(horse&)
uses
creates
creates
uses
training
-int training_hp
-bool isInjury
+void training_speed(horse&)
+void training_power(horse&)
+void rest(horse&)
Race
-Canvas canvas
-horse horses[7]
-horse& player
-Horse_name name
+void start()
+void tie_breaker(int)
+void show_race_summary()
+void show_race_rank()
modifies
uses
has
has
uses
horse
+int get_spd()
+void move(int tier)
+void reset()
+int get_position()
+int get_prev_pos()
Canvas
+void printMap()
+void set_tile(int y, int x, int prevX)
+void set_player(int index)
Horse_name
+string get_name(int breed)
RaceInfo
+static string pad(string, int)

객체지향 설계 원칙 적용 (1/2)

1. 단일 책임 원칙 (SRP)
- 클래스별 책임 분리
 → horse: 말 능력치/이동 처리
 → training: 훈련, 체력, 부상 관리
 → race: 경기 진행, 보상, 결과 처리

2. 캡슐화
- horse 클래스 스탯은 private, getter/setter로만 접근
- training 클래스는 체력 및 부상 상태를 외부에서 직접 조작 불가

3. 추상화
- horse::move(), training_bass(), canvas::printMap()
 → 내부 동작을 감추고, 기능 호출만 가능하게 설계

객체지향 설계 원칙 적용 (2/2)

4. 상속 대신 구성 사용
- 스타일별 주행 특성은 수치만 다르고 동작은 동일
 → horse 클래스 내 breed_mod[], stat_mod[]로 통일 관리
 → 상속을 피하고, 코드 간결성과 유지보수성 확보

5. 객체 간 협력
- race 클래스는 horse, canvas, horse_name 등 다양한 객체를 조합
 → 구성(Composition) 방식으로 협력 구조 구성
 → 객체 간 의존성을 최소화하며 전체 흐름을 조율

💡 설계 의도
- 실전 게임 구조에 객체지향 개념을 녹여냄
- 유지보수가 쉽고, 각 파트를 독립적으로 테스트 가능한 구조 설계

프로그램 주요 기능

주요 클래스	기능 요약
Horse	- 말의 능력치(스피드, 힘, 지구력, 근성) 관리 - 이동 로직: 품종별 특성과 구간별 가중치 계산 - getter/setter, 위치 정보, 랭크 저장 등
Training	- 능력치 성장 로직 (훈련 종류별 증가 수치 다름) - 체력 시스템, 부상 확률 및 회복
Race	- 플레이어 포함 말 7마리 생성 및 진행 - 레이스 상태 갱신, 등수 판정, 보상 로직 - tie_breaker(), lap_time_set() 등 핵심 알고리즘 포함
Canvas	- 맵 그리기, 말 위치 출력 (ASCII 아트) - 플레이어 말은 '★', CPU는 랜덤 기호
UI	- 품종별 이름 리스트에서 중복 없이 CPU 이름 생성 - 게임 시작 화면, 말 선택, 훈련창 및 정보 출력 - 키보드 입력 처리 및 스탯창 ASCII 출력

추가된 내용 – 프로그램 시연 시 확인

1. 게임을 main.cpp과 game.h로 **역할분리**
2. **튜토리얼 추가** - tutorial.h
3. 앤딩 아스키 아트 파일 생성 – final_reward.h
4. 깔끔한 랜더링을 위한 race_info.h 파일 추가

프로그램 주요 기능 – 개편된 이동로직

🎯 이동 로직 개선 이유

- 기존에는 능력치가 낮을 경우 말이 한 칸도 못 움직이는 문제가 발생

🎯 개선 방향

- 말의 스피드 + 스타일 + 해당 구간에서의 중요 능력치를 반영
- 말마다 스타일에 따라 중요 능력치와 구간별 보정이 다르게 적용됨
- 속도에 난수 요소와 보정값을 추가해 더 현실적이고 역동적인 이동이 가능

💡 설계 포인트

- 구간별 스탯, 말의 스타일, 경기 난이도 등 다양한 요소가 유기적으로 작용
- horse::move() 내부에 캡슐화되어 있어, 외부에서는 단순히 ‘move()’만 호출

프로그램 주요 기능 – 등수 판정 & 기록

🏁 등수 판정 (tie_breaker)

- 결승선을 넘은 말 중 아직 순위(rank)가 없는 말을 대상으로 실행
- **최종 도착 거리 = 위치 + 소수점**
- 도착 거리를 기준으로 내림차순 정렬하여 **동석차 없이 정확한 순위 부여**
- 위치 데이터와 내부 상태를 활용해 **Race 클래스 내에서 처리**됨

💡 부가 기능 – 기록 출력 (lap_time_set)

- 실제 경기처럼 도착 시간을 표시해 몰입감을 높이기 위한 **연출용 기능**
- 등수와 연동하여, 말의 도착 거리와 도착 턴을 기반으로 '기록'을 출력

역할 분담

팀원	역할 분담
강승우	- 말 클래스 설계 및 이동 구현 - 훈련 시스템(성장, 체력, 부상) 설계 및 구현 - 경주 진행 시 실시간 맵 출력 UI(Canvas) 담당
김수엽	- 게임 실행 흐름 제어 (main, game) - 게임 재시작/예외처리 로직 - 전반적인 로직 안정화 및 버그 수정
김한민	- UI 구성 및 ASCII 아트 디자인 - 말 선택 화면, 훈련 UI 등 시각 연출 - 배경 사운드 추가 및 최종 시상 연출 구현
조홍준	- 게임 기획 전반 주도 - 말 이동 로직 및 레이스 알고리즘 설계 - 동석차 판정, 말 스타일, 튜토리얼, CPU 이름 자동 생성 등 경기 요소 개발

개발 중 겪은 문제 & 해결 방법

1. 협업의 어려움

→ 방식에 대한 의문이 있을 때, 생성형 AI를 활용하여 팀 기준 정립

2. 코드 의존성 문제

→ 팀원 신뢰 및 컴파일 오류 방지 위해 미리 선언

3. 시간복잡도, 공간복잡도 고민

→ 코드 리뷰를 진행

향후 개선 방향

1. 구조 개선 (SOLID 적용 강화)
- 출력 전용 파일들에도 클래스를 도입해 책임 분리
- Race 클래스의 역할을 세분화하여 유지보수성 향상

2. 실시간 경기 몰입도 강화
- 진행 중 실시간 등수 표시
- 스킬 발동이나 전략 선택 등 플레이어 개입 요소 도입

3. 육성 요소 확장
- 랜덤 이벤트, 고유 스킬, 장비 시스템 추가
- 체력 외에도 날씨/컨디션 등 변수 도입

프로그램 시연