H1 최종 발표

Github 어렵단 [말] 이야

강승우 김수엽 김한민 조홍준

프로젝트 주제 & 개발 목표

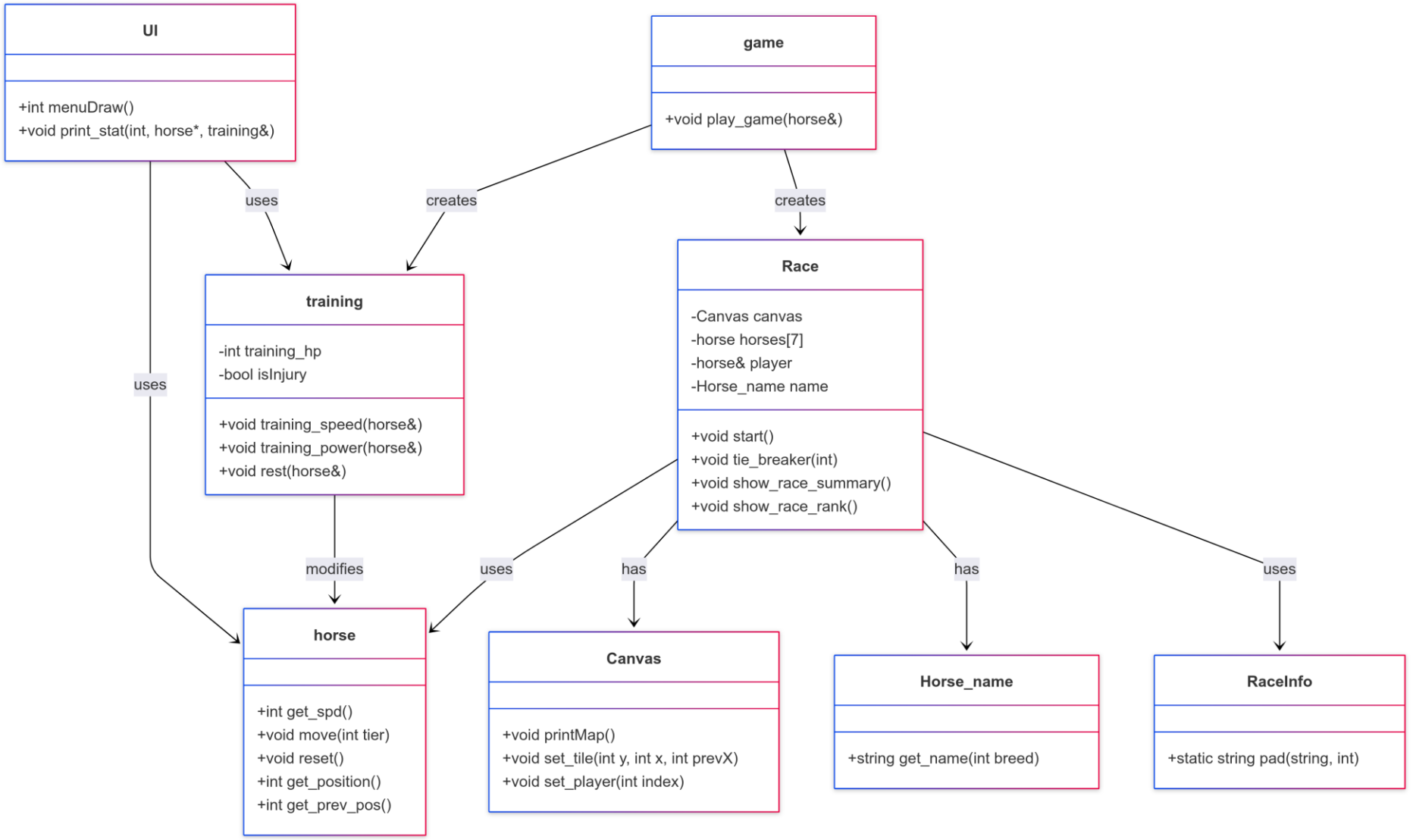
주제

→ 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)

OOP 개념 활용

캡슐화

→ 접근 지정자 사용하여 능력치 직접 접근 제한

→ getter(), setter() 사용하여 능력치 간접 접근

추상화

→ 각 클래스의 책임을 명확히 분리

→ 파일 단위로 역할을 구분

```
class horse {
private:
    HorseType type;
    // 말 능력치
    int spd = 0;   // 속도
    int pow = 0;   // 힘
    int sta = 0;   // 스태미나
    int guts = 0;  // 근성

    // 말 기본 정보
    std::string name;   // 말 이름
    int breed;          // 품종 번호 (도주 - 0 / 선행 - 1 / 선입 - 2 / 추입 - 3)

    // 위치 관련 변수
    int prev_pos = 0;         // 이전 위치
    int curr_pos = 0;         // 현재 위치
    double decimal_point = 0.0; // 소수점 누적 (정수 이동 제외 분)
    int rank = 0; // 레이스 등수 저장

protected:
    // 스텟별 가중치
    double breed_mod[3];      // 구간 가중도 0: 초반 / 1: 중반 / 2: 후반
    double stat_mod[3];       // 능력치 가중도 0: 초반 / 1: 중반 / 2: 후반
```

```
// 스탯 조회용 getter
std::string get_name() { return name; }      // 말 이름 리턴
int get_spd() { return spd; }                // 스피드 리턴
int get_pow() { return pow; }                // 파워 리턴
int get_sta() { return sta; }                // 스태미나 리턴
int get_guts() { return guts; }              // 근성 리턴
int get_breed() { return breed; }            // 품종 리턴
int get_rank() { return rank; }              // 랭크 리턴


// 스탯 추가용 setter
void set_spd(int n) { this->spd += n; }      // 스피드 증가
void set_pow(int n) { this->pow += n; }      // 파워 증가
void set_sta(int n) { this->sta += n; }      // 스태미나 증가
void set_guts(int n) { this->guts += n; }    // 근성 증가
void set_breed(int n) { this->breed = n; }   // 품종 SET
void set_rank(int n) { this->rank = n; }     // 랭크 SET
```

프로그램 주요 기능 소개 – Horse

스탯 – 스피드, 힘, 지구력, 근성

(총 속도와 초 중 후반 구간별 이동속도 추가 능력치)

이동 로직 함수 – move();

스탯 조회용 getter, setter 함수들

프로그램 주요 기능 소개 – 훈련

1. 훈련 시 **능력치 증가** → **체력 10 감소**
2. 체력에 따라 **부상** 확률 존재
3. 부상 시 **능력치 10% 감소**
4. 부상 후 특정 행동 시 **능력치 일정량 복구**
5. **휴식**을 통해 체력 회복
6. **MAX_STAT == 1400**

프로그램 주요 기능 소개 – 레이스

1. 레이스 생성자로 플레이어 말을 받아오고 **cpu말 생성**
2. 경주 참여 말들의 **사전 정보 출력**
3. 레이스 **start() 함수로 레이스 시작**
4. 7마리 전원 들어오면 **레이스 종료**
5. **보상 및 게임오버 판정**

Todo List – 중간발표 기준

완료한 목록

1. 플레이어 말 스텟 1400 이상 증가 막기
2. 경마 시작 시 CPU 말 조회
3. 일정 등수 미달 시 게임 오버
4. ★ 음향 추가 ★
5. ★ 아스키코드 아트 추가 ★
6. UI&랜더링 개선
7. Racing 부분 크기 및 랜더링 개선
8. 경마 과정에서 1234 숫자가 아닌 기호 보이게 하기
9. 훈련 증가량 조절
10. 경마 결과에 따른 스텟 보상

Todo List – 중간발표 기준

중간발표와 달라진 것

1. Save & Lode 기능 → 플레이타임이 짧아 구현하지 않기로 결정
2. 사용자의 입력에 따른 QTE시스템 → 게임과 맞지 않다고 판단하여 취소

중간 발표 이후
수정 및 추가된 내용

프로그램 주요 기능 소개 – 이동로직(구)

총이동거리 =
(스피드능력치/1000) * 기본이동칸수
+ 품종별 보정치*(구간주요능력치/1000) * 스탯별 보정치
+ 누적소수점

→ **하위 티어에서 한 턴에 한 칸도 이동하지 못하는 문제 발생!**

프로그램 주요 기능 – 개편된 이동로직

능력치 가중도: 이 능력치가 이 말에게 얼마나 중요한지

특성 보너스: 해당 말의 특성에 따른 보정치

base_speed = 스피드/400 + random(2~3)

stat_ratio = (구간중요스텟/400) * 능력치 가중도

correction = 게임이 빨리 끝나지 않게 하기 위한 보정치

total = (base_speed + stat_ratio) * correction + 누적 소수점

```cpp
// 말 이동 로직
void move(int tier) {
    // 구간별 대응 스탯 선택: [0]:pow, [1]:sta, [2]:guts
    int stat_by_section[3] = { pow, sta, guts };
    double correction = 1.0; // 최종 이동 거리 보정치 (너무 빨리 끝나는 것을 방지)

    if (tier == 7) { correction = 0.8; } // 7티어 경기 보정치
    else if (tier < 7) { correction = 0.7; } // 1~6티어 경기 보정치

    int seg = position(); // 현재 위치한 구간

    // 속도 기반 이동 거리 + 능력치  + 누적 소수점
    double base_speed = (static_cast<double>(spd) / BASELINE) + 2.0 + (rand() % 11 / 10.0);

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

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

    // 소수점 부분은 다음 이동에 반영되도록 저장
    decimal_point = fmod(move_distance, 1.0);

    // 정수 부분만 현재 위치에 반영
    prev_pos = curr_pos;
    curr_pos += static_cast<int>(move_distance);
}
```

프로그램 주요 기능 소개 – 등수 판정

1. 결승선을 넘은 말이 있을 때 실행
2. 결승선을 넘었고, 랭크가 아직 배정되지 않은 경주마들 대상
3. 동석차 발생 시, **도착 거리 기준 내림차순 정렬을 통해 값이 가장 큰 말에게 상위 등수 부여**

```cpp
void tie_breaker(int rank) { // 타이 브레이커 및 랩타임 계산
    vector<pair<double, int>> list; // pair인 vector list(거리, 말번호)

    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);
            list.emplace_back(total, i);  // 거리, 말 번호
        }
    }

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

    for (int i = 0; i < list.size(); i++) { // 최종 등수 부여
        int num = list[i].second;
        int total_rank = rank + i + 1;
        horses[num].set_rank(total_rank);
    }
}
```

프로그램 주요 기능 소개 – 랩타임 출력

등수판정 과정을 투명하게 공개하면, 몰입도가 떨어짐
육상 스포츠에서 사용하는 **‘기록’**에 영감을 받아 특수 로직 제작

1턴 = 1.5초

10 - (최종이동거리 / 10)초 → 동석차 판정 기준을 위한 초 단위

몰입감을 위한 인위적인 초 삽입 (60~53초)

```cpp
void lap_time_set(double total, int num) {
    double time;
    int lap_correction;

    // 보정으로 인해 느려진 턴 만큼의 랩타임 감소
    switch (h_tier) { ... }

    time = (h_turn[num] * 1.5) + (10 - total / 10) + lap_correction;
    int min = time / 60;
    double sec = fmod(time, 60.0);

    // 소수점 셋째 자리에서 자르기 (반올림 없이)
    double truncated_sec = floor(sec * 1000) / 1000.0;

    // 문자열로 저장 (to_string은 기본적으로 6자리 출력함)
    string sec_str = to_string(truncated_sec);

    // 소수점 셋째 자리까지만 남기기
    size_t dot_pos = sec_str.find('.');
    if (dot_pos != string::npos && dot_pos + 4 < sec_str.size()) {
        sec_str = sec_str.substr(0, dot_pos + 4); // 소수점 이하 3자리까지
    }

    lap_time[num][0] = to_string(min); // 분 저장
    lap_time[num][1] = sec_str;        // 초 저장
}
```

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

역할 분담

강승우 – horse.h, training.h, canvas.h

김수엽 – main.cpp, game.h, horse_name.h

김한민 – final_reward.h, ui.h, 아스키아트, 사운드

조홍준 – 기획, race.h, race_info.h, tutorial.h

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

→ GitHub를 활용한 코드 리뷰를 진행

프로그램 시연