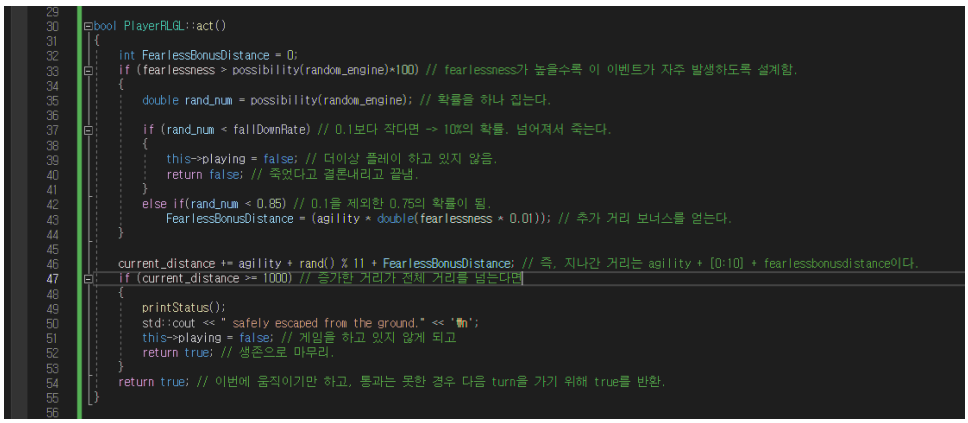
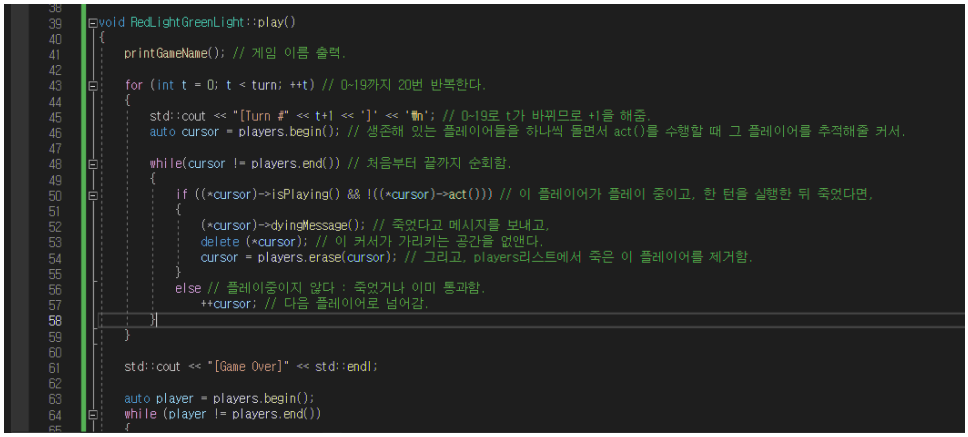
20101576 정재찬.

1. PlayerRLGL::act()함수 구현



```
bool PlayerRLGL::act()
{
    int FearlessBonusDistance = 0;
    if (fearlessness > possibility(random_engine)*100) // fearlessness가 높을수록 이 이벤트가 자주 발생하도록 설계함.
    {
        double rand_num = possibility(random_engine); // 확률을 하나 집는다.

        if (rand_num < fallDownRate) // 0.1보다 작다면 -> 10%의 확률. 넘어져서 죽는다.
        {
            this->playing = false; // 더이상 플레이 하고 있지 않음.
            return false; // 죽었다고 결론내리고 끝냄.
        }
        else if(rand_num < 0.85) // 0.1을 제외한 0.75의 확률이 됨.
            FearlessBonusDistance = (agility * double(fearlessness * 0.01)); // 추가 거리 보너스를 얻는다.
    }

    current_distance += agility + rand() % 11 + FearlessBonusDistance; // 즉, 지나간 거리는 agility + [0:10] + fearlessbonusdistance이다.
    if (current_distance >= 1000) // 증가한 거리가 전체 거리를 넘는다면
    {
        printStatus();
        std::cout << " safely escaped from the ground." << '\n';
        this->playing = false; // 게임을 하고 있지 않게 되고
        return true; // 생존으로 마무리.
    }
    return true; // 이번에 움직이기만 하고, 통과는 못한 경우 다음 turn을 가기 위해 true를 반환.
}
```

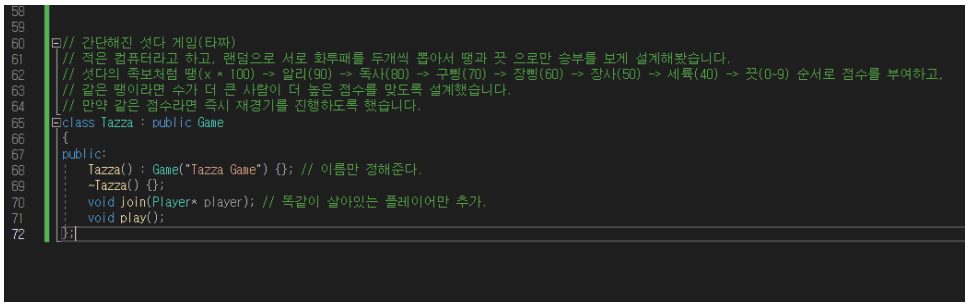
2. RedLightGreenLight::play()함수 구현



```
void RedLightGreenLight::play()
{
    printGameName(); // 게임 이름 출력.

    for (int t = 0; t < turn; ++t) // 0~19까지 20번 반복한다.
    {
        std::cout << "[Turn #" << t+1 << "]" << '\n'; // 0~19로 t가 바뀌므로 +1을 해줌.
        auto cursor = players.begin(); // 생존해 있는 플레이어들을 하나씩 돌면서 act()를 수행할 때 그 플레이어를 추적해줄 커서.

        while(cursor != players.end()) // 처음부터 끝까지 순회함.
        {
            if ((*cursor)->isPlaying() && !((*cursor)->act())) // 이 플레이어가 플레이 중이고, 한 턴을 실행한 뒤 죽었다면,
            {
                (*cursor)->dyingMessage(); // 죽었다고 메시지를 보내고,
                delete (*cursor); // 이 커서가 가리키는 공간을 없앤다.
                cursor = players.erase(cursor); // 그리고, players리스트에서 죽은 이 플레이어를 제거함.
            }
            else // 플레이중이지 않다 : 죽었거나 이미 통과함.
                ++cursor; // 다음 플레이어로 넘어감.
        }
    }

    std::cout << "[Game Over]" << std::endl;

    auto player = players.begin();
    while (player != players.end())
    {
```

3. 나만의 게임 Tazza 구현.

게임클래스를 상속받은 Tazza클래스. 선언



```
// 간단해진 섯다 게임(타짜)
// 적은 컴퓨터라고 하고, 랜덤으로 서로 화투패를 두개씩 뽑아서 땡과 끗 으로만 승부를 보게 설계해봤습니다.
// 섯다의 족보처럼 땡(x * 100) -> 알리(90) -> 독사(80) -> 구삥(70) -> 장삥(60) -> 장사(50) -> 세륙(40) -> 끗(0~9) 순서로 점수를 부여하고,
// 같은 땡이라면 수가 더 큰 사람이 더 높은 점수를 맞도록 설계했습니다.
// 만약 같은 점수라면 즉시 재경기를 진행하도록 했습니다.
class Tazza : public Game
{
public:
    Tazza() : Game("Tazza Game") {}; // 이름만 정해준다.
    ~Tazza() {};
    void join(Player* player); // 똑같이 살아있는 플레이어만 추가.
    void play();
};
```

Player클래스를 상속받은 PlayerTazza 클래스 선언

```
48
49  class PlayerTazza : public Player
50  {
51  public:
52      PlayerTazza(const Player& player) : Player(player) {};
53      bool act(); // 가상 함수 오버라이드
54      void dyingMessage(); // 가상 함수 오버라이드
55  };
```

Tazza 밑의 멤버 함수들

```
124
125
126 void Tazza::join(Player* player) // 살아있는 플레이어를 받아옴.
127 {
128     players.push_back(new PlayerTazza(*player)); // 이번 게임의 참가자로 추가.
129 }
130
131 void Tazza::play()
132 {
133     printGameName(); // 게임 이름 출력
134
135     auto cursor = players.begin(); // 플레이어들 하나하나를 추적할 커서.
136     while (cursor != players.end()) // 플레이어들을 한번씩만 순회함.(한번의 act로 생사를 결정할 것임.)
137     {
138         if (!(*cursor)->act()) // 게임에 플레이어가 지면 false이므로
139         {
140             (*cursor)->dyingMessage(); // 죽었다고 띄우고
141             delete (*cursor); // 이 플레이어 객체를 동적으로 저장한 공간을 제거.
142             cursor = players.erase(cursor); // 생존자 리스트에서도 제거.
143         }
144         else
145             ++cursor; // 살았다면 다음 플레이어로
146     }
147
148     printAlivePlayers(); // 이 게임을 한 뒤에 살아남은 플레이어들을 출력.
149
150     std::cout << players.size() << " players are alive." << std::endl << std::endl;
151 }
152
```

PlayerTazza 밑의 멤버 함수들.

```cpp
bool PlayerTazza::act() // true면 플레이어가 이긴거, false면 플레이어가 진거.
{
    std::pair<int, int> enemyCard = { card_num(random_engine), card_num(random_engine) }; // 1~10사이의 카드를 두장 뽑는다.
    std::pair<int, int> playerCard = { card_num(random_engine), card_num(random_engine) }; // 1~10사이의 카드를 두장 뽑는다.
    int enemyPoint;
    int playerPoint;

    // 적(컴퓨터)의 점수
    if (enemyCard.first == enemyCard.second)
        enemyPoint = enemyCard.first * 100;
    else if (enemyCard.first + enemyCard.second == 3) // 3이 되는건 1,2 밖에 없음. 알리
        enemyPoint = 90;
    else if ((enemyCard.first == 1 && enemyCard.second == 4) || (enemyCard.first == 4 && enemyCard.second == 1)) // 독사
        enemyPoint = 80;
    else if ((enemyCard.first == 1 && enemyCard.second == 9) || (enemyCard.first == 9 && enemyCard.second == 1)) // 구삥
        enemyPoint = 70;
    else if ((enemyCard.first == 1 && enemyCard.second == 10) || (enemyCard.first == 10 && enemyCard.second == 1)) // 장삥
        enemyPoint = 60;
    else if ((enemyCard.first == 4 && enemyCard.second == 10) || (enemyCard.first == 10 && enemyCard.second == 4)) // 장사
        enemyPoint = 50;
    else if ((enemyCard.first == 6 && enemyCard.second == 4) || (enemyCard.first == 4 && enemyCard.second == 6)) // 세륙
        enemyPoint = 40;
    else
        enemyPoint = (enemyCard.first + enemyCard.second) % 10; // 0~9사이의 수를 갖는다.


    // 플레이어 점수
    if (playerCard.first == playerCard.second)
        playerPoint = playerCard.first * 100;
    else if (playerCard.first + playerCard.second == 3) // 3이 되는건 1,2 밖에 없음. 알리
        playerPoint = 90;
    else if ((playerCard.first == 1 && playerCard.second == 4) || (playerCard.first == 4 && playerCard.second == 1)) // 독사
        playerPoint = 80;
    else if ((playerCard.first == 1 && playerCard.second == 9) || (playerCard.first == 9 && playerCard.second == 1)) // 구삥
        playerPoint = 70;
    else if ((playerCard.first == 1 && playerCard.second == 10) || (playerCard.first == 10 && playerCard.second == 1)) // 장삥
        playerPoint = 60;
    else if ((playerCard.first == 4 && playerCard.second == 10) || (playerCard.first == 10 && playerCard.second == 4)) // 장사
        playerPoint = 50;
    else if ((playerCard.first == 6 && playerCard.second == 4) || (playerCard.first == 4 && playerCard.second == 6)) // 세륙
        playerPoint = 40;
    else
        playerPoint = (playerCard.first + playerCard.second) % 10; // 0~9사이의 수를 갖는다.
```

86 % 문제가 검색되지 않음

Squid Game PlayerTazza

```cpp
        enemyPoint = 50;
    else if ((enemyCard.first == 6 && enemyCard.second == 4) || (enemyCard.first == 4 && enemyCard.second == 6)) // 세륙
        enemyPoint = 40;
    else
        enemyPoint = (enemyCard.first + enemyCard.second) % 10; // 0~9사이의 수를 갖는다.


    // 플레이어 점수
    if (playerCard.first == playerCard.second)
        playerPoint = playerCard.first * 100;
    else if (playerCard.first + playerCard.second == 3) // 3이 되는건 1,2 밖에 없음. 알리
        playerPoint = 90;
    else if ((playerCard.first == 1 && playerCard.second == 4) || (playerCard.first == 4 && playerCard.second == 1)) // 독사
        playerPoint = 80;
    else if ((playerCard.first == 1 && playerCard.second == 9) || (playerCard.first == 9 && playerCard.second == 1)) // 구삥
        playerPoint = 70;
    else if ((playerCard.first == 1 && playerCard.second == 10) || (playerCard.first == 10 && playerCard.second == 1)) // 장삥
        playerPoint = 60;
    else if ((playerCard.first == 4 && playerCard.second == 10) || (playerCard.first == 10 && playerCard.second == 4)) // 장사
        playerPoint = 50;
    else if ((playerCard.first == 6 && playerCard.second == 4) || (playerCard.first == 4 && playerCard.second == 6)) // 세륙
        playerPoint = 40;
    else
        playerPoint = (playerCard.first + playerCard.second) % 10; // 0~9사이의 수를 갖는다.


    if (playerPoint > enemyPoint)
        return true;
    else if (playerPoint < enemyPoint)
        return false;
    else
    {
        printStatus();
        std::cout << "는 카드게임에서 비겼기에 재경기를 진행합니다." << '₩n';
        act();
    }
}


void PlayerTazza::dyingMessage()
{
    printStatus();
    std::cout << "가 카드 게임에서 졌습니다..." << std::endl;
}
```

86 % 문제가 검색되지 않음

4. 타짜게임 설명.

- 이 게임은 영화 타짜에 나오는 '섯다'를 간단하게 구현을 해 보았습니다.
- 그 카드의 패는 1월 ~ 10월 까지의 상징이 그려져 있는 카드들이고, 그 월의 수를 이용해서 높고 낮음이 결정이 됩니다.

- 저는 여기서 플레이어와 상대(컴퓨터)가 2개의 카드를 뽑고(광이나 이런 것은 무시한 뒤) 구현할 수 있는 범위 내의 카드쌍들의 점수를 더 높은 단계의 순서대로 배치했습니다.
- 그리고 뽑은 카드에 맞는 점수를 플레이어와 상대(컴퓨터)에게 준 뒤 플레이어가 더 낮은 점수라면 패배(죽음), 더 높은 점수라면 승리(생존), 점수가 같다면 재경기를 하도록 설계했습니다.
- 재경기는 다른 플레이어가 모두 끝난 뒤 하는 게 아닌 게임이 끝나자마자 바로 다시 실행하게 했습니다.
- 이 타짜게임을 수행한 뒤 생존 확률은 50%라고 기대하고 있습니다.

5. 타짜게임의 출력 결과(일부)

- 출력 전체는 sample_output.txt파일을 참고해주시면 될 것 같습니다.