2024 Spring OOP Assignment Report

과제 번호 : 5
학번 : 20230642
이름 : 이채영
Povis ID : chyng

명예서약 (Honor Code)

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.
I completed this programming task without the improper help of others.

1. 프로그램 개요

□ 본 프로그램은 Goose Goose Duck 게임을 구현한 프로그램이다. Goose Goose Duck 은 온라인 멀티플레이 게임으로, 온라인 게임 '어몽어스'와 '마피아'를 모티브로 제작된 사회적 추론 게임이다. 서로 정체를 알고 있는 오리 진영과 서로 정체를 모르는 거위 진영, 중립 진영간의 싸움으로, 각 진영마다 승리 조건을 달성하기 위해 라운드마다 상대 진영 구성원을 제거해 나가는 게임이다.

□ 플레이어 역할군은 총 7가지가 있다. 오리 진영에 해당하는 오리, 암살자 오리와 거위 진영에 해당하는 거위, 탐정 거위, 장의사 거위, 그리고 중립 진영인 도도새와 송골매이다. 생존한 플레이어들은 순서대로 능력을 사용하며, 송골매, 암살자 오리, 오리, 탐정 거위, 장의사 거위, 거위, 도도새 순으로 사용한다. 역할군이 같은 플레이어의 경우, 플레이어 추가 순서를 따른다.

□ 생존한 플레이어들이 능력을 사용한 후, 추가된 순서대로 투표를 진행한다. 송골매를 제외한 플레이어들이 사망하지 않은 모든 플레이어에 대해 한 표를 행사할 수 있다. 아무에게도 투표하지 않는 무효표를 행사하는 것도 가능하다. 가장 많은 표를 받은 플레이어가 한 명일 경우 사망하고, 플레이어가 오리인지 아닌지 모두에게 공개된다.

□ 각 플레이어의 능력은 아래와 같다.

◆ 오리: 살조 - 자신을 제외한 다른 플레이어 한 명을 죽일 수 있다. 다른 오리를 대상으로도 같은 진영이지만 대의를 위해 죽일 수 있다. 단, 해당 라운드에 이미 다른 오리들이 '라운드당 오리 살조 제한 횟수'만큼 죽였다면, 해당 라운드에서는 다른 플레이어를 죽일 수 없다.

◆ 암살자 오리: 1. 암살 - 한 플레이어를 선택하여 역할을 맞추면 해당 플레이어를 죽일 수 있다. 단, 선택한 플레이어의 역할을 맞추지 못했을 경우, 프라이드에 치명적인 상처를 입게되어 선택한 플레이어를 죽이지 못한채 비관하여 자살한다.

암살 능력으로 다른 플레이어를 죽였을 경우, 라운드당 오리 살조 제한 수에 포함되지 않는다. 게임이 진행되는 모든 라운드 동안 암살 능력은 2 회만 사용할 수 있다.
2. 살조 – 오리의 살조 능력과 동일하다.

- ◆ 탐정 거위 – 조사: 생존한 플레이어 중 한 명을 선택하여, 해당 플레이어가 이번 라운드에서 다른 플레이어를 죽였는지 죽이지 않았는지 확인할 수 있다
- ◆ 장의사 거위 – 염습: 죽은 플레이어 중 한 명을 선택하여, 해당 플레이어의 역할을 알아낼 수 있다.
- ◆ 거위 – 능력이 없다.
- ◆ 도도새 – 능력이 없다.
- ◆ 송골매 – 살조: 자신을 제외한 다른 조류 한 마리를 죽일 수 있다.

또한 각 진영의 승리 조건은 아래와 같다.

- ◆ 거위 – 거위 진영에 속한 플레이어가 한 명이라도 생존해 있고, 오리 진영에 속한 모든 플레이어와 송골매가 사망한 경우에 거위가 승리한다. (도도새는 살상 능력이 없어 생존해 있어도 상관없다.
- ◆ 오리 - 오리 진영에 속한 플레이어가 한 명이라도 생존해 있고, 살아남은 오리 진영 플레이어의 수가 살아남은 다른 플레이어의 수보다 많거나 동일한 경우 오리가 승리한다. 단, 생존한 플레이어가 2 명 일 때, 생존한 조류가 오리 진영 조류가 한마리이고 송골매가 한마리인 경우에는, 오리의 승리가 아닌 송골매의 승리다.
- ◆ 중립 : 송골매와 도도새 모두 중립 진영이지만, 서로 승리 조건을 공유하지 않는다.
  - 송골매 - 자기 자신이 살아있고, 다른 플레이어가 한명 이하일 때 승리한다.
  - 도도새 - 가장 많은 투표를 받아 추방당했을 때 남은 역할의 수와 종류에 관계없이 승리한다. 마지막에 도도새 혼자만 살아남는 상황은 일어나지 않는다고 가정한다.

2. 프로그램의 구조 및 알고리즘

□ Class 설명:

- ■ GGD : Goose Goose Duck 게임의 전반적인 게임 플레이를 관리한다. 게임 상태, 플레이어 관리, 게임 라운드의 진행을 처리한다.

- Private 멤버 변수:
  - BirdList* bird_list: 게임에 있는 모든 새 플레이어 목록을 가리키는 포인터.
  - int number_to_kill: 한 라운드에서 죽일 수 있는 최대 새(오리)의 수.
  - int round: 현재 라운드 번호.
  - int num: 게임에서 살아있는 새의 총 수.
  - int duck_num: 게임에서 살아있는 오리의 수.
  - int goose_num: 게임에서 살아있는 거위의 수.
  - int dodo_num: 게임에서 살아있는 도도새의 수.
  - int falcon_num: 게임에서 살아있는 송골매의 수.
  - int win: 승리 조건을 추적; 7로 초기화되며 승리 조건이 충족되지 않았음을 나타냄.
  - int invalid_vote: 투표 단계에서 무효표의 수를 세는 변수.
  - bool dodo_win: 도도새가 승리 조건을 달성했는지 여부를 나타낸다.
  - int num_kill_bird: 현재 라운드에서 죽일 수 있는 새의 수를 추적; number_to_kill에 따라 설정된다.
- Constructor: 게임 환경을 초기화하며, 멤버 변수에 기본값을 설정하고 bird_list에 대한 메모리를 할당한다.
- Destructor: bird_list에 사용된 동적 메모리를 정리하여 메모리 누수를 방지합니다.
- void GameStart(): 게임 설정 메뉴를 표시하고 플레이어 추가, 살해 한도 설정, 게임 시작 등을 위한 사용자 입력을 처리한다.
- void RoundProgress(): 단일 게임 라운드의 진행을 관리하며, 스킬 사용, 투표 및 살아있는 새의 수 업데이트 등의 메소드를 호출한다.
- bool IsGameOver(): 현재 게임 상태를 기반으로 승리 조건이 충족되었는지 확인한다.
- void PrintGameResult(): 승리 조건에 따라 게임 결과를 출력한다.
- void AddPlayer(): 새로운 플레이어를 게임에 추가하고, 그들의 역할을 초기

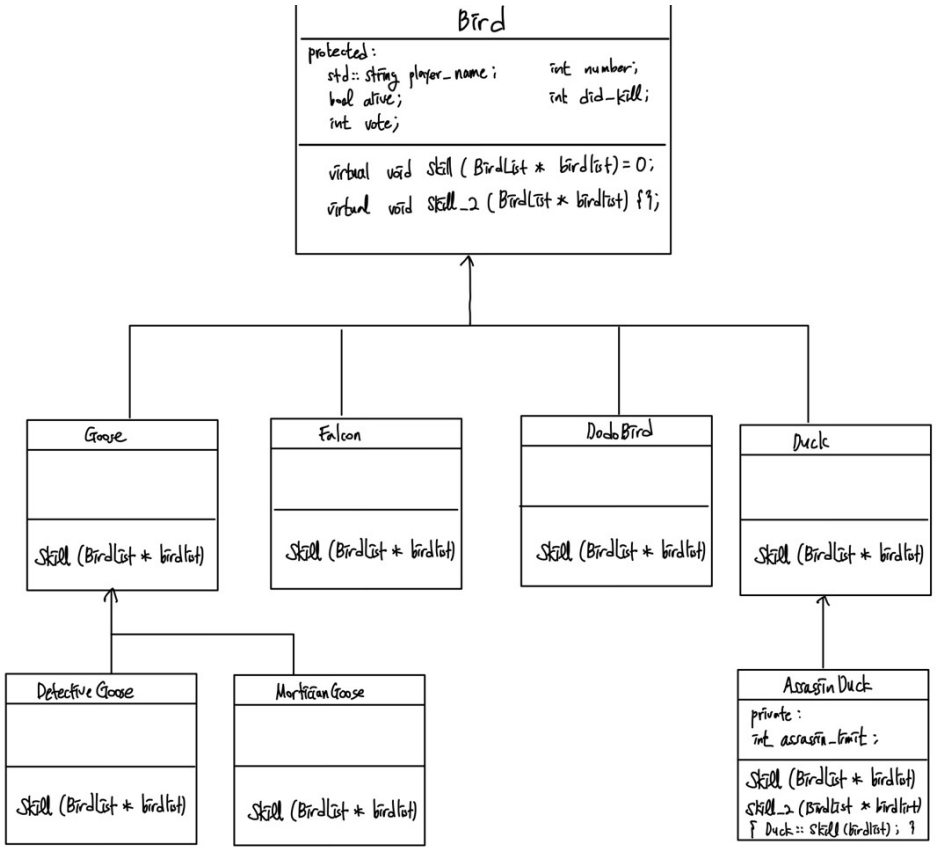
화하며 관련 카운터를 업데이트한다.

- ◆ void Skill_Phase(): 각 새가 그들의 역할과 게임의 규칙에 따라 특별 능력을 사용할 수 있도록 스킬 사용 단계를 관리한다.
- ◆ void Vote_Phase(): 살아있는 각 새가 다른 새를 제거하기 위해 투표하는 투표 단계를 관리한다.
- ◆ void Update_num(BirdList* birdlist): 각 라운드 실행 후 살아있는 각 범주의 새의 수를 업데이트합니다.

■ Bird: 게임 내 모든 새 캐릭터들의 기본 클래스이다. 추상 클래스로, 다양한 종류의 새 역할에 대한 구현을 위한 기반을 제공한다.

- ◆ 멤버 변수(protected)
  - std::string player_name: 플레이어의 이름.
  - bool alive: 새가 살아있는지 여부를 나타냄 (살아있으면 true, 죽었으면 false).
  - int vote: 플레이어가 받은 투표 수.
  - int number: 플레이어의 역할 번호.
  - int did_kill: 다른 새를 살조하거나 암살했는지 여부 (했으면 1, 안 했으면 0). 탐정 거위 조사용
- ◆ Constructor: 플레이어 이름을 매개변수로 받아 객체를 초기화한다. 생존 상태는 true로, 투표 수와 살조 여부는 0으로 초기화된다.
- ◆ Destructor: 가상 소멸자를 통해 파생 클래스의 객체도 적절히 처리할 수 있도록 한다.
- ◆ std::string GetPlayerName() const: 플레이어의 이름을 반환한다.
- ◆ void SetAlive(bool status): 플레이어의 생존 상태를 설정한다.
- ◆ bool IsAlive() const: 플레이어의 생존 상태를 반환한다.
- ◆ int GetNumber() const: 플레이어의 역할 번호를 반환한다.
- ◆ void setVote(int num): 플레이어가 받은 투표 수를 설정한다. num이 0인 경우 투표 수를 리셋한다.
- ◆ int Num_vote() const: 플레이어가 받은 투표 수를 반환한다.

- ◆ void SetKill(bool status): 플레이어가 다른 새를 살조했는지 여부를 설정한다.
- ◆ bool Num_kill() const: 플레이어가 다른 새를 살조했는지 여부를 반환한다.
- ◆ virtual void Skill(BirdList* birdlist) = 0: 순수 가상 함수로, 파생 클래스에서 각 새의 특수 능력을 구현하기 위해 정의해야 한다.
- ◆ virtual void Skill_2(BirdList* birdlist): 암살자 오리의 살조 스킬 사용을 위해 정의한 함수이다. 기본 구현은 비어 있으며, AssassinDuck 클래스에서 다시 정의된다.
- ◆ Bird를 부모 클래스로 하고, 자식 클래스로 있는 클래스는 Duck, Goose, Falcon, DodoBird이다. 그리고 AssassinDuck이 Duck을 부모 클래스로 두고 있고, DetectiveGoose, MorticianGoose가 Goose를 부모 클래스로 두고 있다. 상속 관계는 아래 그림과 같다.



- ■ BirdNode : BirdList라는 연결 리스트에서 각각의 Bird 객체를 포함하는 노드를 관리한다. 이 클래스는 리스트의 각 요소를 연결하며, 새 객체에 대한 참조를 유지

한다.

- ◆ 멤버 변수
  - Bird* bird: Bird 클래스의 인스턴스를 가리키는 포인터. 이 포인터는 리스트에 있는 개별 새의 정보를 저장한다.
  - BirdNode* next: 리스트에서 다음 BirdNode 객체를 가리키는 포인터.
- ◆ Constructor
  - BirdNode(): 기본 생성자로, bird와 next를 NULL로 초기화한다.
  - BirdNode(Bird* bird): 특정 Bird 객체를 인자로 받아 bird 멤버 변수를 초기화하고, next는 NULL로 설정한다.
- ◆ void SetBird(Bird* bird): bird 멤버 변수를 설정합니다. 이 함수는 새로운 Bird 객체를 현재 노드에 연결하는 데 사용된다.
- ◆ void SetNext(BirdNode* next): next 멤버 변수를 설정합니다. 이 함수는 다음 노드를 현재 노드에 연결하는 데 사용된다.
- ◆ Bird* GetBird() const: 현재 노드에 저장된 Bird 객체의 포인터를 반환한다.
- ◆ BirdNode* GetNext() const: 현재 노드의 다음 노드를 가리키는 포인터를 반환한다.

■ BirdList: BirdNode를 사용하여 Bird 객체들의 연결 릿트를 관리한다. 게임 내의 모든 캐릭터를 추적하는 데 사용된다.

- ◆ 멤버 변수
  - BirdNode* head: 리스트의 시작 부분을 가리키는 포인터
  - BirdNode* tail: 리스트의 끝 부분을 가리키는 포인터
- ◆ Constructor: 생성자에서는 head와 tail을 nullptr로 초기화하여 빈 리스트를 생성한다.
- ◆ Destructor: 소멸자에서는 리스트를 순회하면서 각 BirdNode 객체를 삭제하여 메모리 누수를 방지한다. 모든 노드가 제거되면 head와 tail도 nullptr로 설정된다.
- ◆ BirdNode* GetHead() const: 리스트의 첫 번째 노드인 head를 반환한다.
- ◆ BirdNode* GetTail() const: 리스트의 마지막 노드인 tail을 반환한다.

- ◆ void AddBirdNode(BirdNode* node): 새로운 BirdNode를 리스트의 끝에 추가한다. 만약 리스트가 비어 있다면, head와 tail 모두 이 노드를 가리키게 설정한다. 그렇지 않으면 tail의 next를 새 노드로 설정하고 tail을 업데이트 한다.

□ 프로그램 실행 흐름

1. 게임 시작 및 초기 설정 (GameStart)

게임은 GGD 클래스의 GameStart 함수 호출로 시작된다. 사용자는 게임 설정 메뉴를 통해 플레이어를 추가하고, 라운드당 오리 살조 제한 횟수를 설정할 수 있다. 모든 설정이 완료되고 게임 시작 조건이 충족되면, 게임이 시작된다.

2. 플레이어 추가 (AddPlayer)

사용자는 플레이어의 이름과 역할을 입력하여 게임에 참가시킬 수 있다. 각 플레이어는 Bird 클래스의 파생 클래스 인스턴스(예: Duck, Goose, Falcon 등)로 생성되며, BirdList에 추가된다.

3. 라운드 진행 (RoundProgress)

각 라운드는 RoundProgress 함수를 통해 관리된다. 이 함수는 다음과 같은 세부 단계를 포함한다.

- Skill_Phase: 각 플레이어는 자신의 역할에 따른 특수 능력(살조, 암살, 조사, 염습)을 사용한다.
- Vote_Phase: 생존한 플레이어들은 다른 플레이어를 게임에서 제거하기 위해 투표한다. 가장 많은 표를 받은 플레이어가 게임에서 제거된다.

4. 승리 조건 확인 (IsGameOver)

각 라운드 이후, IsGameOver 함수를 통해 승리 조건이 충족되었는지 검사한다. 특정 진영에서 승리 조건을 만족하면 게임이 종료된다.

5. 게임 결과 출력 (PrintGameResult)

게임이 종료되면, PrintGameResult 함수를 통해 최종 승리 진영을 출력한다.

6. 메모리 정리

게임 종료 후, 생성된 모든 객체와 동적 할당된 메모리가 각 클래스의 소멸자를 통해 해제된다.

3. 토론 및 개선

- □ 본 프로그래밍 과제에서는 클래스 상속, 동적 메모리 할당, Function Overloading 등을 활용해 다양한 객체들을 관리하며 게임을 작동시키는 프로그램을 설계하였다. 비슷한 특성을 가진 캐릭터 역할군의 부모 클래스를 정의하고, 순수 추상 함수를 이용해 다형성을 증진함으로서 객체 지향 프로그래밍의 특성을 이해할 수 있었다.
- □ 본 프로그램에 추가할 수 있는 기능으로는 더욱 다양한 역할과 캐릭터를 추가하는 것이 있다. 역할군 추가를 통해 게임의 다양성과 전략적 깊이를 증가시킬 수 있다. 또한 다른 규칙이나 게임 플레이스타일을 제공하는 여러 게임 모드를 추가할 수 있다.
- □ 본 프로그램에서 개선할 수 있는 부분은, 중복되는 부분을 함수로 분리하거나 추가적인 클래스 메소드를 정의하여 코드를 더 간결하고 효율적으로 만들 수 있다. 또한 사용자 입력 처리 부분에서 잘못된 입력에 대한 예외 처리를 강화하여 프로그램 오류를 줄일 수 있다.

4. 참고 문헌

- https://en.cppreference.com/w/cpp/language/overload_resolution
- Function overloading에 관한 내용을 참고하였다.