**2024 Spring OOP Assignment Report**

과제 번호 : 5-1

학번 : 20230499

이름 : 김재환

Povis ID : 20230499

**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**

인기있는 마피아류 게임 중 하나인 구스구스덕의 변형이다. 기존 구스구스덕은 사람이 직접 조작하여 움직일 수 있으며, 거위(시민)편은 주어진 임무를 완수한다는 목표를 가지고 있으나, 변형본에서는 텍스트를 이용해 구현해야하므로 해당 기능을 제거하고, 직업만을 채용하여 일반 마피아 게임처럼 진행된다.

일반 마피아게임의 밤과 낮 개념 대신, 한 라운드가 능력 단계(phase)와 투표 단계로 구성된다. 각 단계는 일반적인 마피아 게임의 밤, 낮 단계에 해당하며, 능력 단계에서는 정해진 순서에 따라 각 조류들이 능력을 사용하며, 투표 단계에서는 게임에 추가된 순서대로 투표를 하여, 가장 많이 투표를 받은 사람이 있다면 해당 플레이어를 사망처리한다. 능력사용 순서는 다음과 같다.  
송골매 – 암살자 오리 – 오리 – 탐정 거위 – 장의사 거위 – 거위 – 도도새

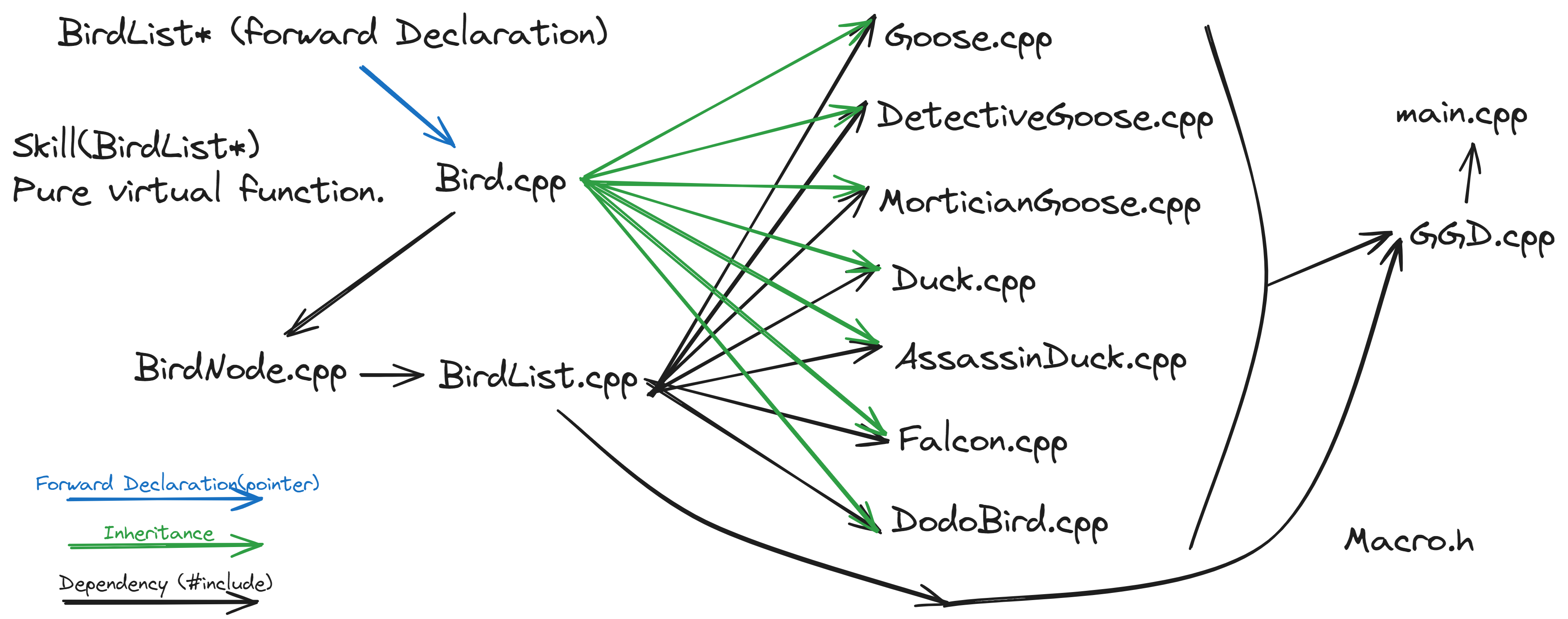
각 조류의 대략적인 능력은 다음과 같다**.**

**중립직업 : 특별한 승리 조건이 존재하며 이를 달성하면 1인 승리한다.**  
 - 송골매 : 중립직업으로, 자신을 제외한 모든 플레이어를 죽이면 승리한다. 즉, 자신과 다른 아무 조류 한 명으로 두명만 남게 되면 승리한다.  
 - 도도새 : 중립직업으로, 최다 투표를 받으면 승리한다. 능력은 없다.  
**오리진영 : 오리의 수를 과반수를 넘으면 승리한다.** - 오리 : 밤에 다른 조류 한 명을 죽일 수 있다. 한 라운드에서 오리가 사용할 수 있는 능력 횟수를 차감한다.  
 - 암살자 오리 : 두가지 능력을 사용할 수 있다. 첫 번째는 암살으로, 전체 게임에서 총 두번 사용할 수 있다. 특정 플레이어를 지목하여 역할을 맞추면 살해할 수 있고, 틀리면 스스로 자살한다. 대신 이 능력은 한 라운드에서 오리 진영이 죽일 수 있는 횟수를 차감하지 않는다. 두 번째 능력은 오리와 동일하다.  
**거위진영: 모든 오리와 송골매를 처치하면 승리한다.**  
 - 거위 : 아무 능력이 없다.  
 - 탐정 거위 : 플레이어를 지목하여, 해당 플레이어가 누군가를 죽였는지 여부를 알 수 있다.  
 - 장의사 거위 : 사망한 플레이어를 지목하여, 해당 플레이어의 직업을 알 수 있다.

프로그램이 실행되면 메뉴를 출력한다. 메뉴에서는 플레이어 추가, 한 라운드당 오리진영의 살해 최대 횟수, 게임 시작 세 가지 기능을 사용할 수 있다. 게임을 시작하기 위한 충분한 조건이 갖추어 지지 않으면 게임을 시작하지 않는다. 적절히 플레이어를 추가한 후 게임을 시작하면 능력 단계와 투표 단계를 반복하며, 특정 직업(진영)의 승리 조건이 만족된다면 결과를 출력하고 게임을 종료한다. 승리조건의 확인은 한 라운드가 종료된 후 일어난다.

1. **프로그램의 구조 및 알고리즘**

메인 소스 파일을 제외하면 각 C++파일과 헤더파일은 하나의 클래스를 구현하기 위해 작성되어 있으며, 파일들의 관계는 다음 그림과 같다.

****

표준 라이브러리를 사용하는 대신 조류, 조류 노드, 조류 리스트를 직접 구현하여 사용하였다. 각 직업의 능력을 구현할 때, 능력 멤버 함수가 모든 작업을 처리할 수 있게 작성하고자 하였다. 이를 위해서 다른 조류에 상호작용할 수 있도록 조류 리스트를 매개변수로 받아야만 하는데, 이를 Bird.cpp에 작성하려고 한다면 조류, 노드, 리스트 사이 순환 의존성이 발생하게 되었다. 이를 해결하기 위해 Bird.cpp에서 순수 추상 함수로 스킬 함수를 정의해두고, 리스트를 포인터로 선언하여 넣어 전방선언 방법으로 순환 의존성을 해결하였다. 능력 멤버 함수의 구현은 조류 클래스를 상속받은 각 직업 클래스 내에 구현하였다.

각 직업들의 승리조건을 확인하기 위해 다음 과정을 이용하였다. 먼저, 오리 진영, 거위 진영, 송골매와 도도새의 수를 센다. 이후, 송골매가 1명, 다른 조류의 수가 1 이하라면 송골매의 승리이다. 송골매의 승리가 아닐 경우, 오리의 수가 다른 조류보다 많은 경우 오리의 승리이다. 둘다 아니라면 오리와 송골매가 없을 경우 거위의 승리이다. 셋 모두 아니라면 게임을 이어서 진행한다. 단, 도도새가 투표로 죽었을 경우 모든 경우를 고려하지 않고 도도새의 승리가 된다.

Bird클래스에 몇 가지 멤버를 추가하였다.

|  |  |
| --- | --- |
| bool is\_alive | 해당 플레이어가 생존해 있는지를 나타낸다. |
| int role | 직업 코드. |
| int killMaxCount | 오리 진영일 때 사용된다. 한 라운드에서 최대 살조 횟수. |
| bool kill | 해당 라운드에서 사람을 죽였는지를 나타낸다. |
| bool win | 플레이어가 승리했는지 나타낸다. |

또한, Bird클래스에 추가된 멤버 함수는 모두 위 변수들의 Getter와 Setter 함수이다.

1. **토론 및 개선**

주어진 요구사항을 구현하는 도중에 순환 의존성 오류로 인해 해맸던 적이 있다. 전 어싸인에도 순환 의존성으로 고통받았고, 해결 방법을 많이 찾아봤던 경험이 있다. 전 어싸인에서는 여러 객체의 맴버함수가 다른 클래스의 객체에 서로 접근하는 방향으로 코드를 작성했었고, 이를 해결하기 위해 다른 클래스에 간섭하는 멤버함수의 내용을 메인 파일로 옮겼었다. 순환 의존성을 해결하기 위해서는 주로 포인터로 선언하는 전방선언을 하는 방법이 있는데, 이 방법은 포인터로 선언한 클래스의 멤버들을 사용할 수 없다. 그러므로 해당 부분의 구현을 다른 곳으로 옮겨야 한다. 이번 어싸인에서는 처음에 각 직업 클래스를 다른 파일로 나누지 않고 Bird.cpp에 모두 구현하려고 해서 생긴 문제로, 이렇게 한다면 순환 의존성이 발생한다. 이를 해결하기 위해서 각 직업 클래스를 개별 파일로 나누고, 순수 추상 함수를 오버라이딩하여 스킬 구현을 Bird.cpp에서 다른 파일로 옮김으로써 해결하였다.

직업을 구현하기 위해서 좀 더 체계적으로 클래스를 설계하면 더 좋은 코드를 작성할 수 있을 것이다. 예를 들어 직업을 표현하기 위해 순수 추상 함수만 가지는 추상 클래스(인터페이스)를 설계한다거나, 또는 모든 직업이 Bird를 상속하지 않고 각 진영별로 중간 클래스를 하나 더 작성한다면 더 확장성이 좋고 쉽게 수정할 수 있을 것이다. 그러나 이번 어싸인에서는 각 직업, 진영별로 다른 점이 스킬 하나 뿐이므로 더 복잡하게 설계하는 것이 더 방해된다고 생각하여, 확장성 대신 간편하게 코드를 작성하는 것을 선택하였다.

1. **참고 문헌**

Lecture note, 조성현.