소프트웨어공학

프로젝트 개요 보고서

- Title : Yoot Game Overview -

Class	**소프트웨어공학, Class 01**
Professor	**이찬근**
Team number	**Team 18**
Team Members	**김찬중 (20213780)** **박민용 (20213113)** **엄태형 (20202029)** **이정우 (20202021)** **태아카 (20234483)**
Author	**김찬중**

목차

1. Introduction ..... - 2 -

A. 윷놀이의 규칙 ..... - 2 –

2. Body..... - 3 -

B..... - 3 -

1.초기, 종료화면 구상..... - 3 -

2.게임화면 구상..... - 5 -

C. 게임 진행 과정..... - 7 -

D. 게임 로직 구상..... - 8 -

E. 개발 환경..... - 11--

3. Conclusion ..... - 12 -

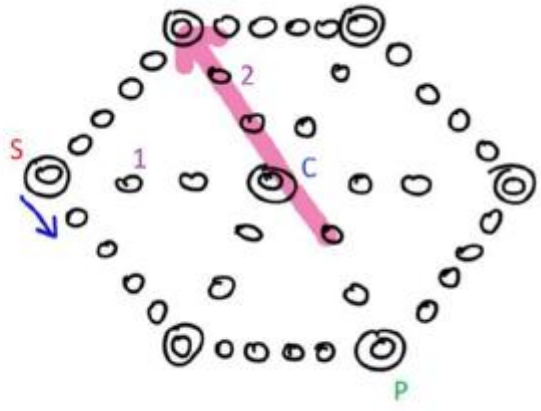
1. Introduction

A - 1) 윷놀이 규칙 (말, 참여자, 윷놀이 이동)

- 윷놀이에 참여하는 인원은 최소 2명, 최대 4명이다.
- 윷놀이에 배치되는 참여 인원 당 배정되는 말의 개수는 최소 2개, 최대 5개이다.
- 윷놀이에 사용되는 이동 규칙은 [백도, 도, 개, 걸, 윷, 모]가 있고 각각 [-1, 1, 2, 3, 4, 5] 칸을 이동해야 한다.
- 윷을 던졌을 때 '윷' 혹은 '모'가 나온 경우, 말을 이동하기 전에 다시 던져 추가로 이동할 거리를 저장해야 한다.
- 말을 이동했을 때, 그 위치에 아군의 말이 위치하고 있으면 말을 엎어야 한다.
- 말을 이동했을 때, 그 위치에 적군의 말이 위치하고 있으면 말을 잡고 해당 말의 소유자는 다시 윷을 던질 권리를 가지게 된다.
- 모든 말을 골인시키면 게임에서 승리한다.
- 승리 후 게임을 다시 시작하거나 종료할 수 있다.

A - 2) 윷놀이 규칙 (판 위에서의 규칙)

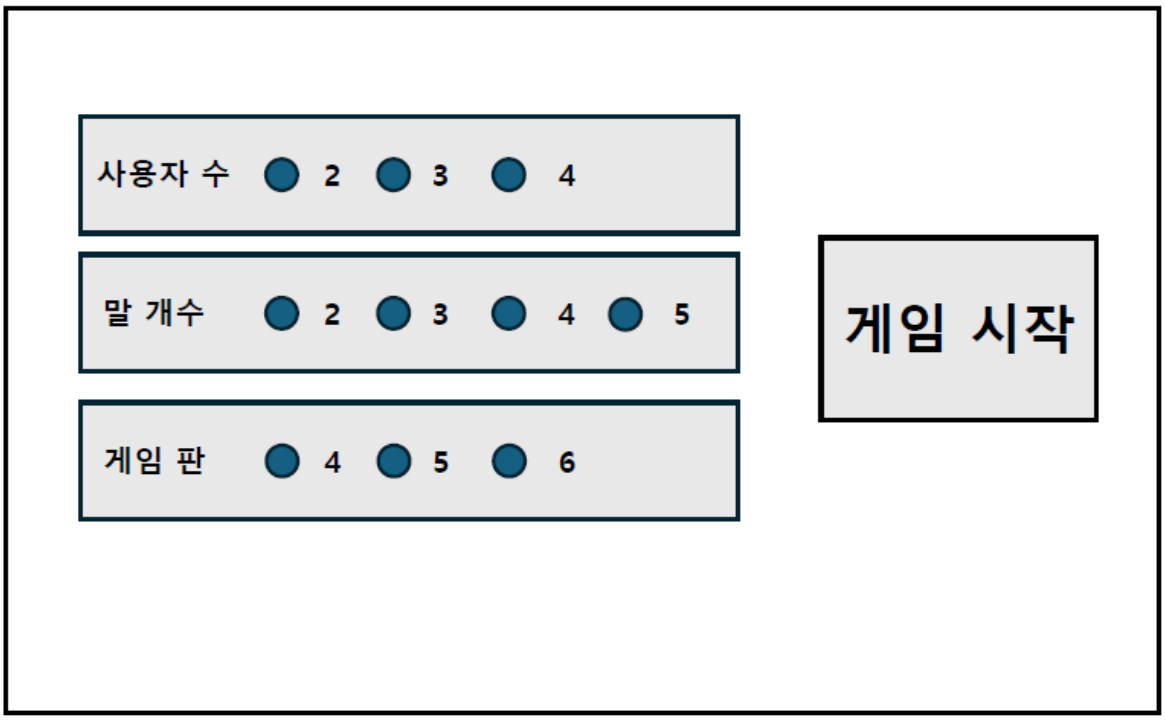
- 각 판의 꼭짓점에서 말이 움직이기 시작한다면 가운데로 이동해야 한다.
- 중앙의 이중 원 위에 위치하지 않고 지나칠 경우 말은 2번째 최단 경로에 해당하는 곳으로 이동해야 한다.



-

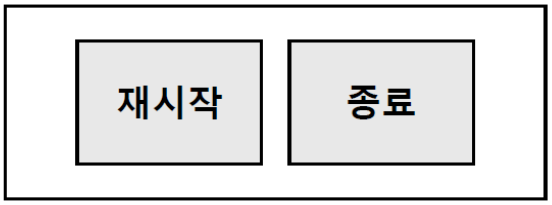
2. Body

B-1-1) 게임 초기화면의 구상



- 초기 화면에선 사용자 수, 말의 개수, 게임판의 모양을 받을 수 있게 위와 같이 UI를 구상하였다.
- 사용자 수, 말 개수, 게임 판을 중복해서 선택할 수 없게 Radio Button으로 구현하는 것이 좋을 것으로 예상된다.
- 사용자 수, 말 개수, 게임 판에서 선택되지 않은 버튼이 있을 것을 대비하여 Default 값을 따로 지정하는 것이 좋을 것 같다.
- 각 옵션을 선택 후 '게임 시작' 버튼을 누르면 게임 화면으로 넘어갈 수 있게 설계하면 좋을 것 같다.

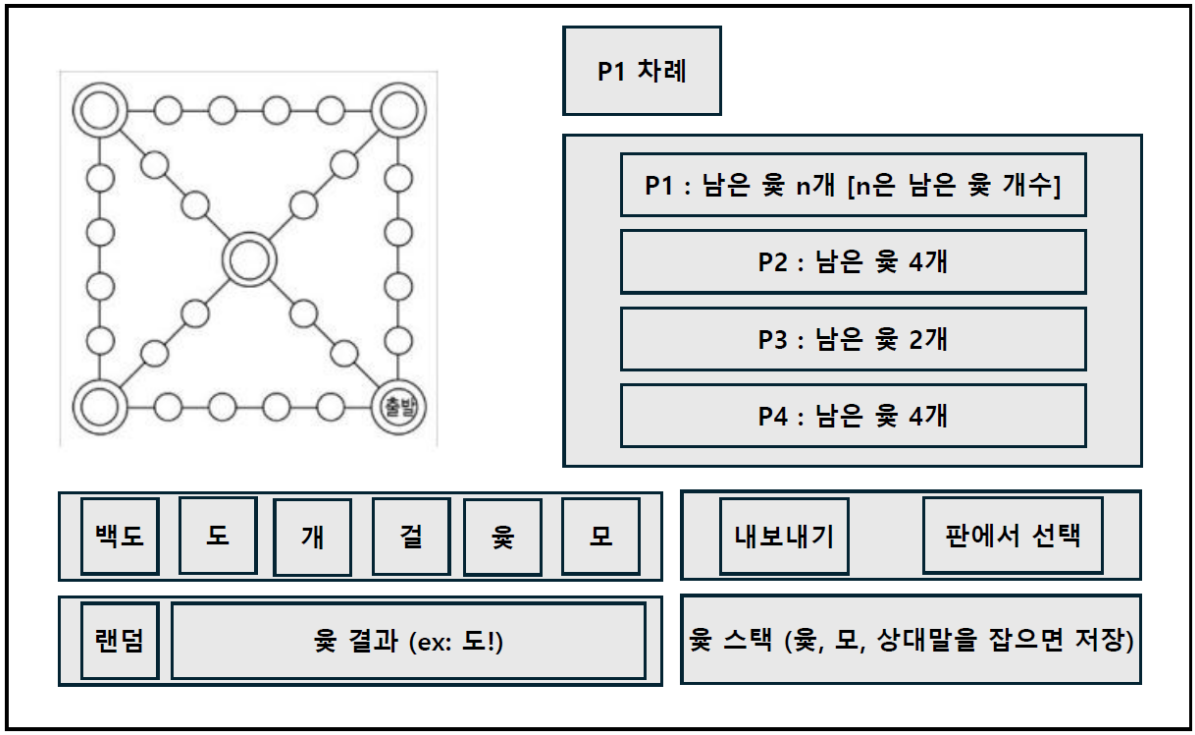
B-1-2) 게임 종료화면의 구상



- (a) 재시작 시 게임 시작 UI로 회귀
- (b) 종료 시 모든 창을 닫기

- 위의 그림과 같이 특정 게임 참여자가 승리하였을 시 재시작과 종료 둘 중하나를 고를 수 있게 버튼을 배치하면 좋을 것 같음.
- 재시작 버튼을 누르게 된다면 B-1-1)의 초기 화면으로 다시 돌아가 게임을 다시 설정할 수 있게 해야 함.
- 재시작 버튼을 누르게 된다면 현재 동적으로 할당되어 있는 모든 값을 free하고 다시 시작해야 함. Java의 Garbage Collection의 경우 전역으로 선언되어 있는 class의 값을 수거하지 않는 경우가 있으므로 이를 고려하지 않을 시 재시작을 할 때마다 메인 메모리의 공간을 조금씩 갉아먹게 된다.
- 종료 버튼을 클릭하게 된다면 프로그램 창을 완전히 종료해야 한다.
- (게임 화면에 대해 설명할 내용이 상대적으로 많기에 다음 페이지로 넘어가서 설명한다.)

B-2) 게임화면 구상



- 위의 그림은 B-1-1)에서 사용자가 게임의 환경을 설정하고 '게임 시작' 버튼 눌렀을 때 나오는 화면이다.
- 게임 화면의 좌측 상단에는 사각형, 오각형, 육각형 형태의 판이 출력 된다.
- 판 위의 원은 버튼으로 매핑하여 색깔과 숫자를 통해 참여자와 말의 엎은 개수를 표현한다.
- 우측 상단의 'P1 차례'는 해당 참여자의 순서를 나타낸다. 위의 사진에서의 예시는 참여자가 4명이므로 P4까지 출력될 것이다.
- 바로 하단에 위치한 거대한 사각형 박스에는 각 참여자의 남은 말의 개수를 표현하는 상자이다. 말이 출발에서 나와 다시 출발 지를 지나 나가게 된다면 남은 말의 개수가 하나씩 줄어들 것이다.
- [백도, 도, 개, 걸, 윷, 모]가 mapping 되어 있는 상자는 debug 용 이동 규칙 버튼이다. '도'를 누르면 사용자의 말이 한 칸 움직이게 된다.
- 그 아래에 있는 '랜덤' 버튼은 우리가 윷을 던지는 것처럼 난수를 이용해 가변적인 이동 규칙을 적용하는 버튼이다. 해당 출력 결과는 '윷 결과' 상자에 나타날 것이다.

- 우측 하단에 있는 '내보내기' 버튼은 윷을 던지고 난 후 새로운 윷을 내보내기 위한 버튼이다. '윷'이 나오고 '내보내기' 버튼을 누르게 된다면 4번째 칸에 해당 참여자의 색깔과 숫자 1이 표시될 것이다. (아래의 붉은 원과 숫자 1이 그 예시이다.)

- 

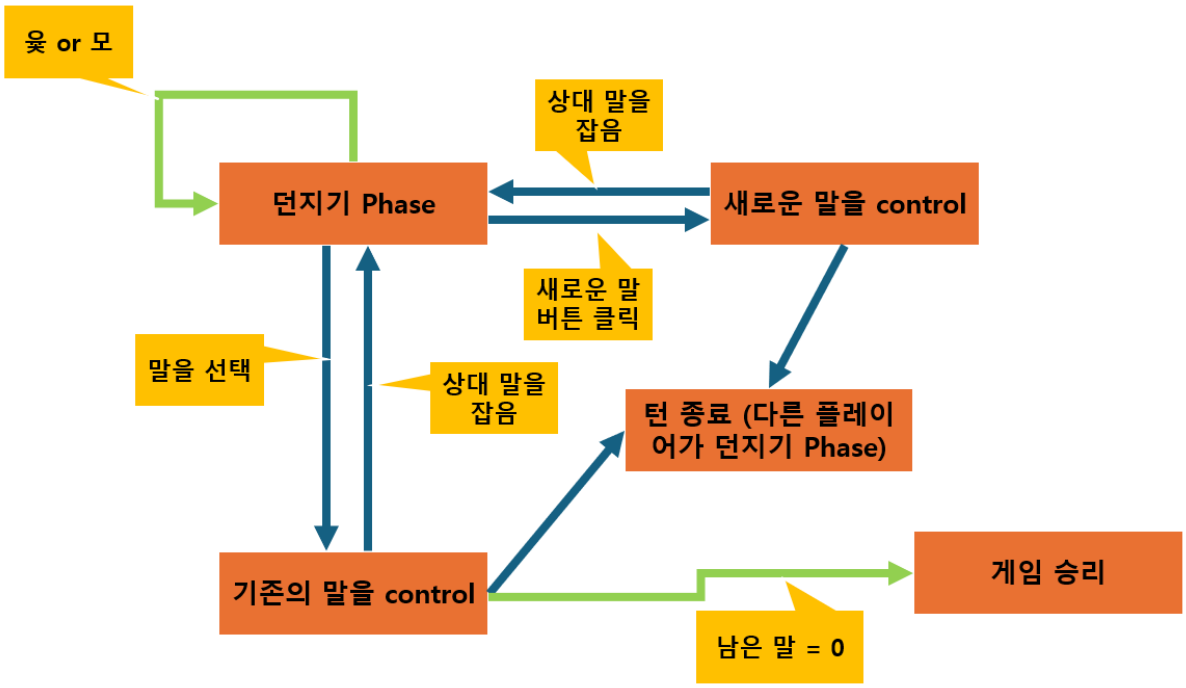
- '판에서 선택' 버튼은 판에 위치한 말을 클릭해서 해당 말에 참여자가 선택한 이동 규칙을 적용시키는 버튼이다.

- 우측 하단의 '윷 스택'은 이동 규칙을 저장하는 리스트라고 생각하면 된다. '걸'이 나왔다고 가정하면 해당 상자에 '걸'이라는 이름의 버튼이 나오게 되고 참여자는 '걸' 버튼을 누르고 '내보내기' 혹은 '판에서 선택' 버튼을 눌러 다음 행동을 선택하게 된다. '윷', '모'가 나왔을 경우 참여자는 '윷', '모'가 아닐 때까지 윷을 던지는 것이 일반적이다. 이를 위해 연속적으로 윷을 던진 경우 각각의 이동 규칙에 해당하는 버튼을 생성하게 된다.

C) 게임 진행 과정

■ 위의 게임은 '유희왕'의 게임 진행 과정이다. 각 플레이어는 턴을 기다리며 자신의 차례를 기다리게 된다. 카드를 드로우하고 자신의 카드를 이용해 게임을 전개한다. 자신의 전개가 끝나면 자신의 턴을 종료하고 상대의 턴으로 넘기게 된다.

■ 위와 같은 카드 게임의 프로세스를 생각했을 때, 윷놀이도 비슷하다고 생각하였다.



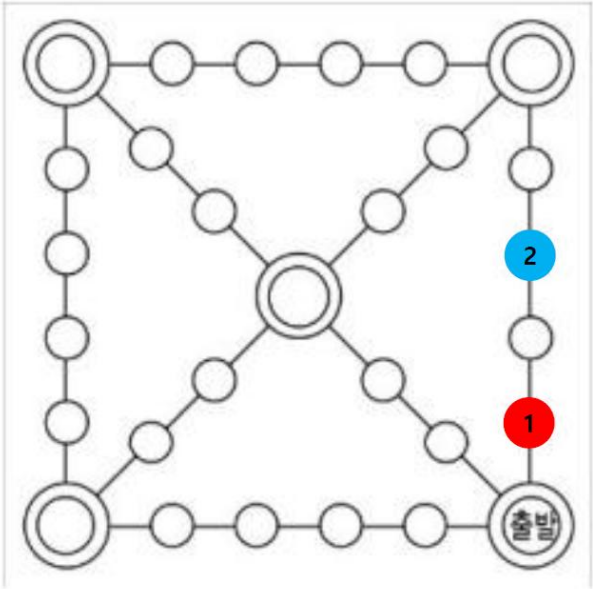
■ 윷놀이도 '던지기'의 과정을 통해 게임을 전개하게 되고 상대의 말 혹은 자신의 말과의 상호작용을 통해 새로운 이벤트를 전개할 수 있게 된다.

■ 우선 게임의 참여자는 윷을 던질 수 있는 권리를 가진다. 윷을 던지면 그 결과에 따라 자신의 말을 새로 보내거나 아니면 게임 판 위의 말을 옮길 수 있게 된다. 던져서 나온 결과가 '윷' 혹은 '모'인 경우 던질 수 있는 권리를 1회 추가로 받게 된다. 최종적으로 던질 수 있는 권리를 소모하게 되면 참여자는 자신의 말을 control할수 있는 권리를 가지게 된다.

■ 게임의 참여자가 새로운 말을 control하게 되는 경우, 상대의 말을 잡는 경우와 그렇지 않은 경우를 생각할 수 있다. 상대의 말을 잡은 경우 다시 던질 수 있는 권리를 가지게 되므로 던지기 Phase로 회귀하게 된다. 그렇지 않은 경우 자신의 턴을 종료하고 다음 참여자의 던지기 Phase로 넘어가게 된다.

■ 게임의 참여자가 게임 판 위의 기존 말을 control하게 되는 경우, 마찬가지로 상대의 말을 잡는 경우와 그렇지 않은 경우를 생각할 수 있다. 상대의 말을 잡은 경우 다시 던질 수 있는 권리를 가지게 되므로 던지기 Phase로 회귀하게 된다. 그렇지 않은 경우 자신의 턴을 종료하고 다음 참여자의 던지기 Phase로 넘어가게 된다. 그런데 여기서 하나의 경우를 더 생각해야 한다. 기존의 말을 움직이는 경우 참여자가 모든 말을 내보낼 수 있는 경우의 수가 생기게 된다. 참여자가 자신의 말을 모두 내보내고 모두 골 인 시킨 경우 게임을 종료할 수 있게 된다.

D) 게임 로직 구상

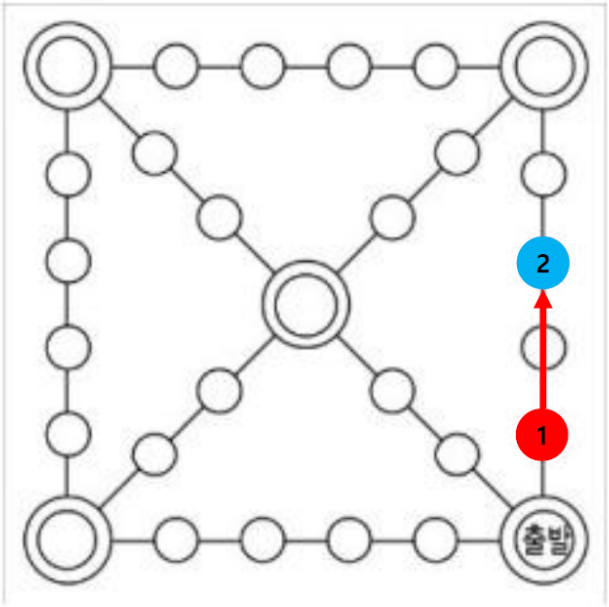


- **(a) 플레이어가 2명일 때를 가정**
- **(b) P1은 빨간색, P2는 파란색 가정**
- **(c) 그림과 같이 숫자와 색깔로 플레이어를 표현**
- **(d) Grouping을 숫자로 표현함으로써 코드의 복잡성을 줄일 수 있음**

■ 사각형 게임판을 기준으로 로직을 생각한다.

■ 참여자가 2명일 때 한 명을 빨간색으로 나머지 한 명을 파란색으로 표시할 수 있다.

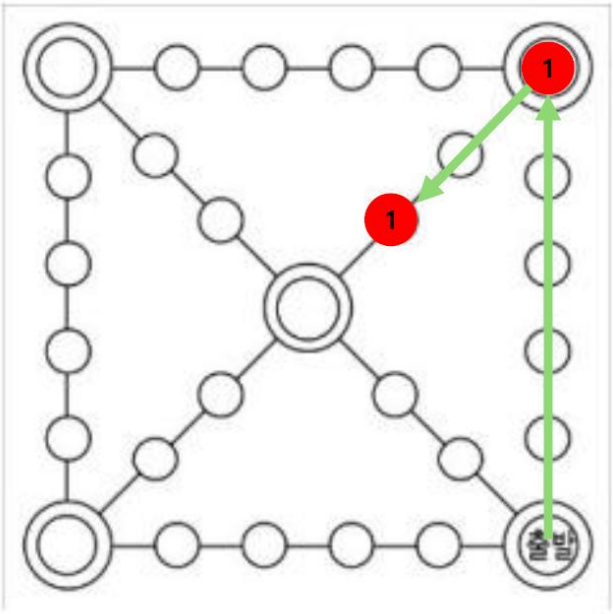
■ Grouping의 경우 말이 몇 개가 한 칸에 있는 지를 표현하는 것이므로 숫자로 표현하면 로직을 단순화할 수 있다.



- **(1) P1이 '개'가 나와서 P2를 잡는 상황을 가정**
- **(2) 우선 움직일 말에 다른 말이 있는지에 대한 여부를 탐색**
- **(3-1) 있으면 그 말이 아군인지 적인지를 판별**
- **(4-1-1) 적이라면 해당말의 색깔을 자신의 말로 바꾸고 숫자를 움직이는 말의 숫자로 바꿈. 그리고 잡힌 말의 숫자만큼 남은 말 표시를 바꾸어 줌**
- **(4-1-2) 아군이라면 해당말의 숫자를 가산해주어서 Grouping을 표현**
- **(3-2) 없으면 현재 자리의 숫자와 색깔을 흰색으로 전환**
- **(4-2) 해당 자리의 색깔을 지금 말의 색깔로 바꾸고 숫자도 똑같이 복사해서 집어넣음**

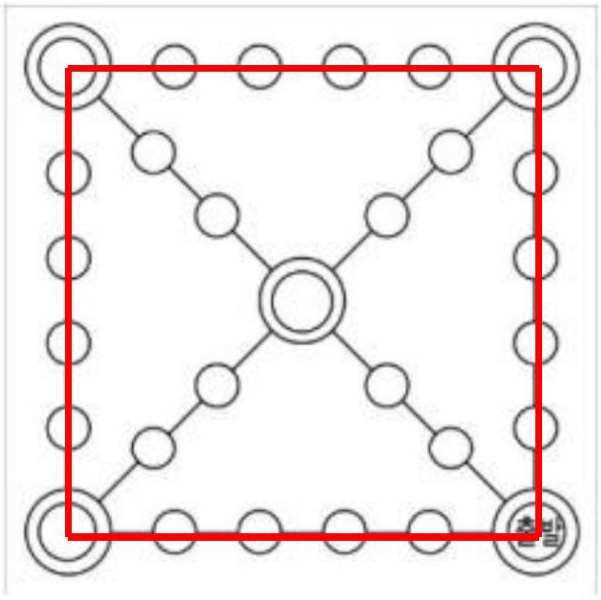
■ 빨간색의 말이 위의 상황에서 '개'가 나와 2칸을 움직이게 된다면 현재 자신의 좌표의 버튼 색을 흰색으로 바꾸고 숫자를 0으로 초기화한 후 움직일 좌표의 색을 빨간색, 숫자를 1로 바꾸어 주면 된다. 이 경우 파란색 참여자는 자신이 내보낼 수 있는 말의 개수를 2개 더해주어야 한다.

■ 위의 사진에서 파란색 말의 위치에 빨간색 말이 위치한 경우, 현재 자신의 좌표의 버튼 색을 흰색으로 바꾸고 숫자를 0으로 초기화한 후 움직일 좌표의 색은 동일하게 유지하되 숫자는 [해당 위치의 숫자 + 움직일 말의 숫자]로 갱신해주면 된다.



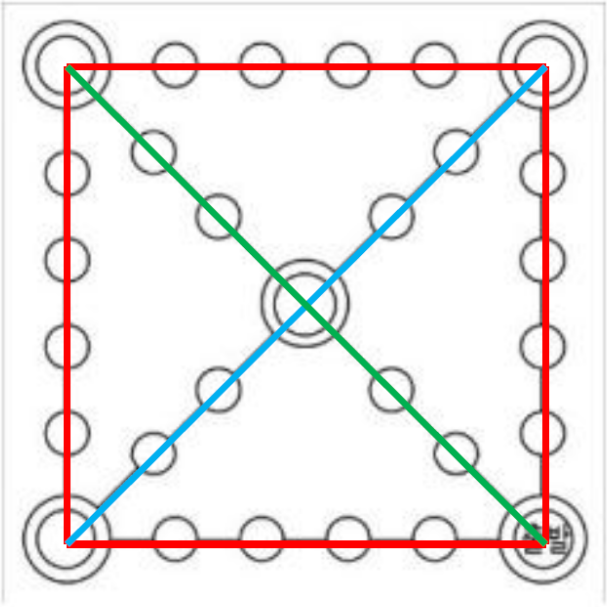
- (a) 윷을 던졌을 때 결과가 모가 나온 다면 다시 던져서 결과를 대기함
- (b) 다시 던져서 개가 나온 경우 사용자가 '모'를 먼저 사용할 지 '개 ' 를 먼저 둘 지를 선택 해야함.
- (c) 그림의 경우는 모 – 개 순서로 사용자가 선택한 경우 임.

■ 사각형 형태의 판에서 가운데로의 진입은 각 꼭짓점에 말이 위치했을 경우만 가능하다.



- (1) 해당 루트를 point[0][0~19]로 표현
- (2) 좌표가 point[0][5 or 10]일 때 안쪽으로 꺾는 것을 표현하기

■ 이러한 움직임을 표현하기 위해 겉의 경로를 이차원 배열의 [0]번째 인덱스로 매핑하면 된다. 가운데로 진입하게 되는 점의 좌표는 point[0][5], point[0][10]일 때이다. (이는 실제 좌표가 아닌 논리적 좌표이다.)



- (a) 파란 선을 point[1][0~6]으로 표현
- (b) 초록 선을 point[2][0~6]으로 표현
- (c) 빨간 선의 좌표가 point[0][5]가 된다면 point[1][0]으로 치환하기
- (d) 빨간 선의 좌표가 point[0][10]이면 point[2][0]으로 치환하기
- (e) 파란 선의 좌표가 point[1][3]이 된다면 초록 선상의 좌표로 변환 (point[2][3])

- <종점 판단>
- point[2][현재 좌표 + 움직일 좌표] > point[2][6] -> 남은 말 개수 하나 줄이기
- Point[0][현재 좌표 + 움직일 좌표] > point[0][20] -> 남은 말 개수 하나 줄이기
- 종료 후에 남은 창과 모든 말, player을 해제 해야함.

- 가운데로 향하는 좌표의 경우 각각 [1], [2]에 인덱싱하면 된다. 추가적으로 파란색 선의 경우 좌표가 point[1][3]일 때 point[2][3]으로 바꾸어 주는 절차를 진행해주어야 한다.
- 종점 판단은 크게 두 가지로 나눌 수 있다. 빨간색 선을 통해 나가는 경우와 초록색 선을 통해 나가는 경우. 전자의 경우 논리적 좌표가 point[0][20]을 넘길 때, 후자의 경우 논리적 좌표가 point[2][6]을 넘길 때 종점을 지났다고 판단할 수 있다.
- 오각형과 육각형도 이러한 형태의 논리적 좌표 매칭을 이용하면 된다.

E) 개발 환경

- **Intelij IDE**를 이용해 **JDK 21**을 사용해 프로그래밍을 진행한다.
- 코드는 YootProject package에 작성한다.
- 이후 진행될 Junit test는 src directory가 아닌 project 하위에 test directory를 새로 만들어서 진행한다.
- Img 파일을 import할 때에는 상대적 주소를 이용해 가져온다. 절대 주소를 이용하면 다른 환경에서의 실행이 문제가 된다.

3. Conclusion (프로젝트 진행 시 기대하는 점)

- Java에 대한 개발 경험이 다들 많지 않으나 이번 프로젝트를 통해 Java의 package, Unit test에 대해 더 자세히 알아갈 수 있을 것 같다.
- Github 사용 경험이 적은 조원이 있으나 branch 관리, git 사용 방법을 서로 알려주며 교학상장할 수 있는 기회를 얻은 것 같아 기대가 된다.
- Java swing을 이용하면서 MVC 패턴을 직접 구현할 수 있는 기회를 얻어서 좋은 것 같다.
- 점진적 개발 방법론을 이용해 많은 토론을 하며 코드를 디버깅하는 프로세스를 경험할 기회가 적었는데, 이번 프로젝트를 통해 많은 것을 얻어 갈 수 있을 것 같다.
- 조원의 역할은 아래와 같다.

이름	역할
김찬중 (20213780)	**GitHub 관리, 프로젝트 기획 및 디버깅, 문서화 및 발표**
박민용 (20213113)	**Logic 구현 (말의 움직임 및 좌표 매핑)**
태아카 (20234483)	**Logic 구현 및 Unit test**
엄태형 (20202029)	**UI 구현 (메인 화면)**
이정우 (20202021)	**UI 구현 (게임 판)**