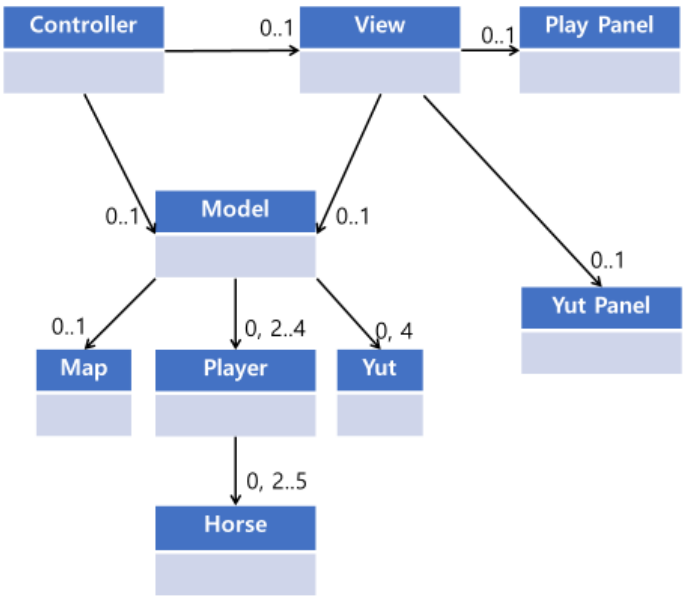
SETeam 5

Analysis & Design, Implementation, and Test Report

20153366 임동규
20154637 박상현
20155500 박성훈

I. Domain Model



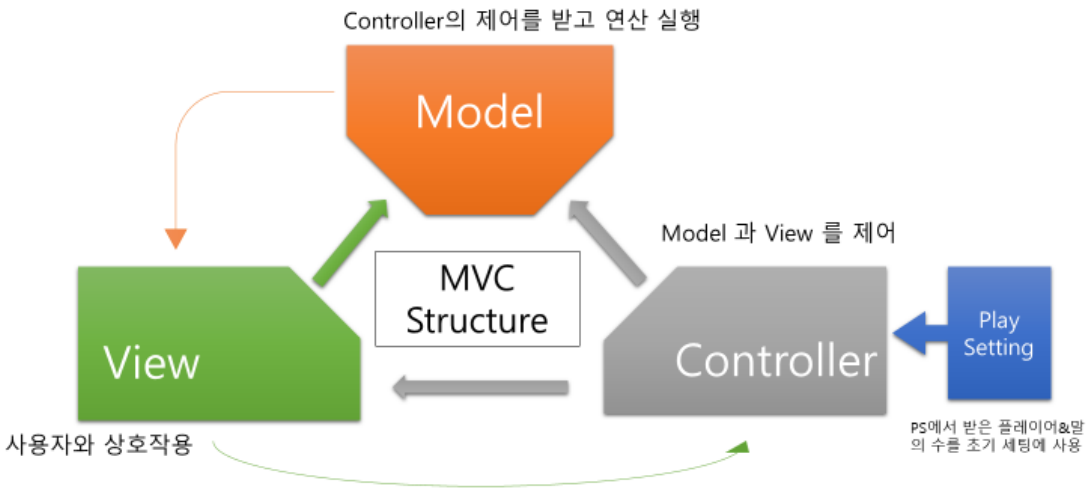
위 그림은 개발된 프로그램의 Domain Model입니다. 개발에 앞서 저희는 먼저 프로그램 비전과 개발을 진행할 언어 및 IDE를 설정했고, 이후 과제의 요구사항에 맞춰 MVC 패턴을 사용할 것을 결정했습니다.

그래서 프로그램을 각각 Model, View, Controller로 나눠 개발을 진행하게 되었고, 구현할 기능들을 단위별로 나눠 게임 내 로직 관련 class들(Map, player, Yut, Horse)은 Model에, UI 관련클래스들(GUI_PlayPan, GUI_YutPan)은 View에, Controller는 Model과 View를 컨트롤 하게 하였습니다.

각 클래스 간의 관계가 기본적으로 0..1인 이유는 게임 실행 자체를 취소하는 경우에는 객체가 생성되지 않고 프로그램이 종료되기 때문입니다. 하지만, Player와 Horse는 프로그램 시작 시 우선적으로 정해지고, 최대-최소 값이 정해져 있기에 2..4&2..5의 값을 가지게 되었습니다.

II. Software Architecture + Design Model

<< Software Architecture >>



(위 그림은, MVC Structure를 통해 개발된 프로그램의 간략한 동작방식을 나타내고 있습니다.)

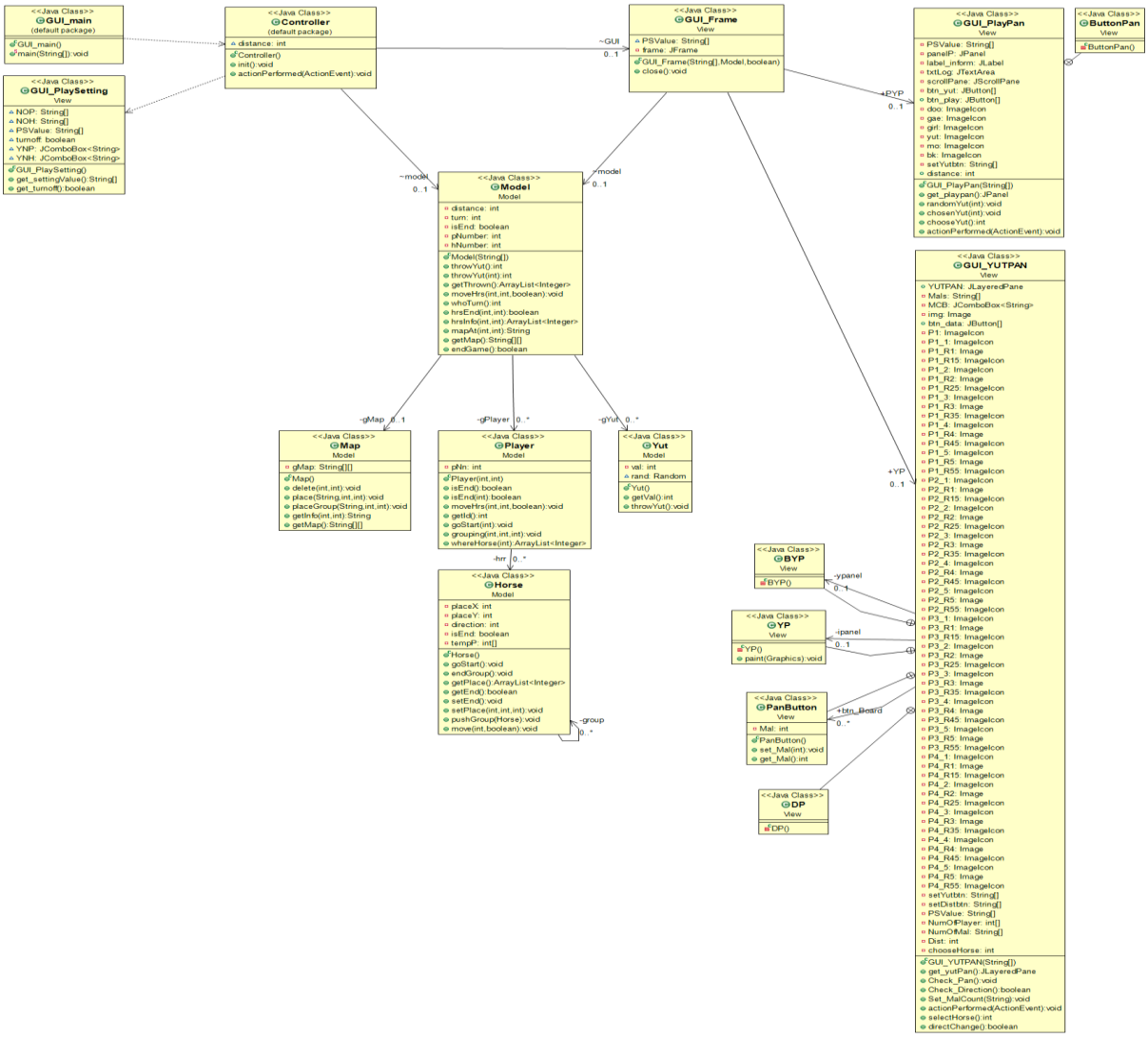
View는 Controller의 제어 하에 화면에 데이터 표시 및 사용자와 직접적인 상호작용만을 담당하며, Model은 윷놀이를 플레이함에 있어서 핵심이 되는 윷 던지기, 말 움직이기, 말의 잡기 및 업기와 관련된 기능들을 수행하도록 분리하였습니다. 단, 처음 시작 시 설정하는 플레이어 수와 말의 개수의 경우에는 게임이 시작되면 저장된 값만으로 사용이 가능하기 때문에 따로 인스턴스로 만들어 최초 1회만 사용하는 방식을 사용하도록 하였습니다.

Controller는 Model과 View에 대한 정보를 모두 가지고 있으며, View의 각 버튼 및 말 등의 표시에 필요한 값을 Model에서 받아와 설정할 수 있고, 이벤트 리스너를 이용해 View에 대한 제어를 실행합니다.

View의 랜덤/지정 윷 던지기 버튼과 말 선택 및 방향 선택에서 발생하는 사용자의 입력은 Controller가 알맞게 정제하여 Model의 메소드를 실행하는 매개 변수로 사용합니다.

Controller의 호출에 따라 Model은 윷 던지기, 말 움직이기 등의 메소드를 실행하며, 메소드 실행으로 변화한 State값을 Controller가 받아 View의 값을 갱신할 수 있도록 getter 메소드를 지원합니다. 이 값으로 Controller는 View의 값들을 세팅한 뒤, 최종적으로 View의 화면을 갱신하여 사용자가 새로운 state를 확인할 수 있도록 합니다.

<<Class Diagram>>



(클래스 다이어그램으로 표현한 프로그램의 구조는 위와 같습니다.)

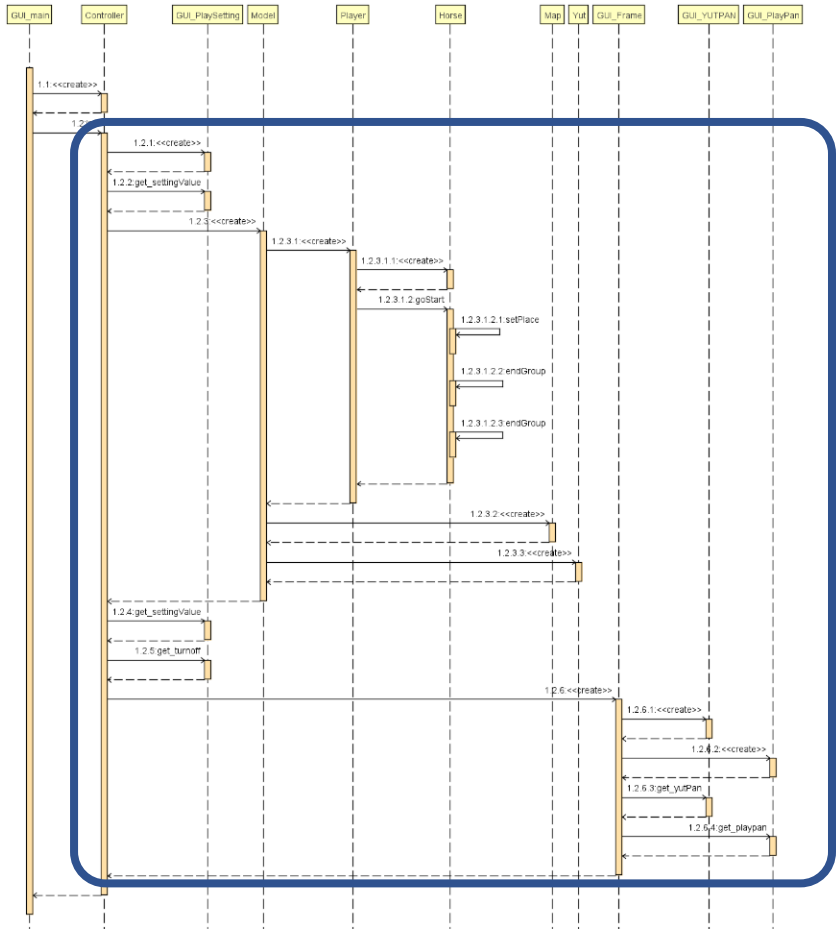
GUI_main 클래스가 윷놀이 게임 전체의 main클래스로써 동작합니다. GUI_main에서 Controller의 인스턴스를 생성하고, Controller.init()을 실행하면 바로 PlaySetting이 실행됩니다.

PlaySetting에서는 게임을 플레이할 인원 수와 말의 개수를 설정할 수 있으며, 취소를 누르면 프로그램이 종료됩니다. PlaySetting의 설정 값을통해 Controller가 Model과 View의 인스턴스를 생성합니다.

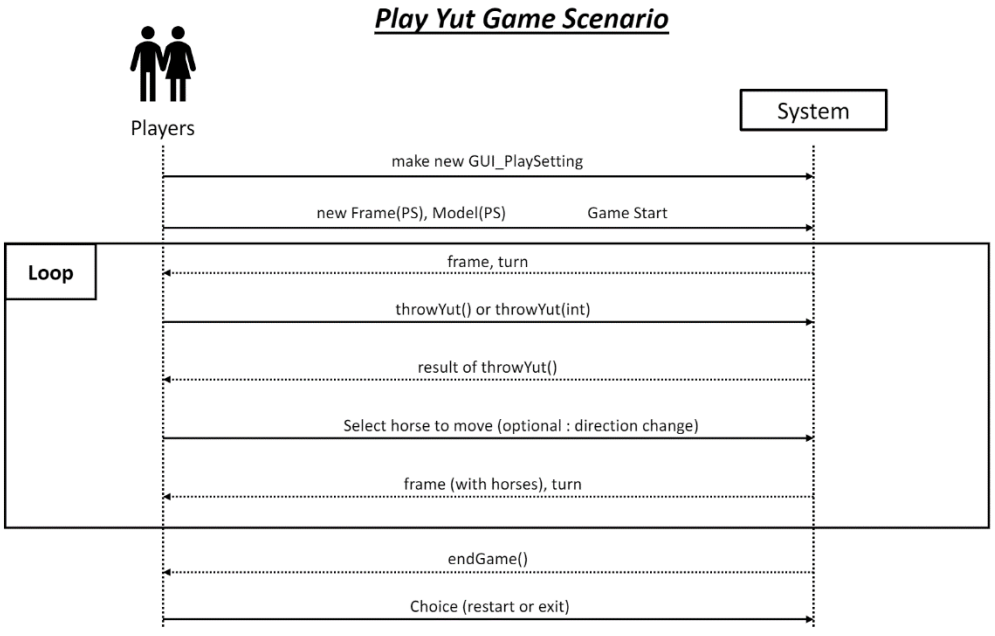
View에서 전체 컨테이너를 GUI_Frame 클래스가 담당하며, 내부에 변수로 가지고 있는 JFrame 창에 모든 UI가 표시됩니다. GUI_PlayPan은 랜덤 던지기 버튼과 지정 던지기 버튼, 던진 윷에 대한 이미지, 현재 던진 말에 대한 정보를 화면에 출력합니다. GUI_YutPan은 현재 윷 판의 상태, 플레이어 번호, 플레이어의 남은 말, 플레이어의 색깔을 표시해줍니다.

Model은 맵, 윷, 플레이어 관련 객체 Map, Yut, Player를 가지며, Player는 각각 말 관련 객체 Horse를 가지고 있습니다. Controller로부터 지시를 받으면 그에 따라 메소드를 실행해 말을 옮기고, 맵 정보를 갱신하는 과정을 처리하게 됩니다.

<<Sequence Diagram & System Sequence Diagram>>

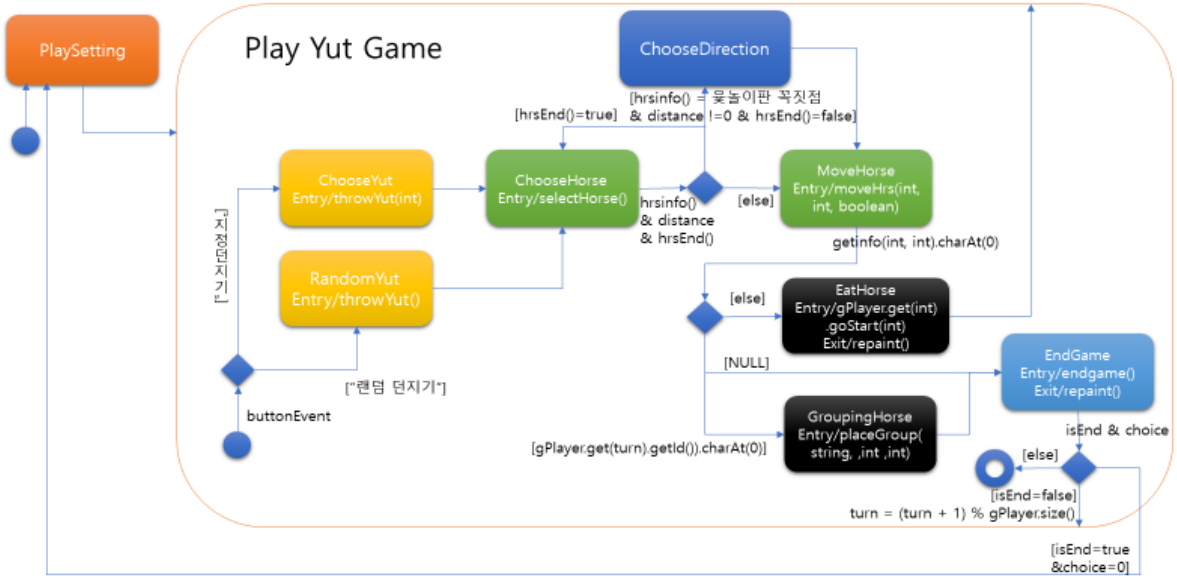


(프로그램의 시퀀스 다이어그램)



(시퀀스 다이어그램의 파란 영역이 실제 동작 상황)

<<State Chart>>



■ State Chart의 실행순서

1. 프로그램 시작 시 Play Setting을 실행하여, 플레이어와 말의 개수를 확인 & 저장 후 [Play Yut Game] State로 전이합니다.

2. [Play Yut Game] state에 진입 후 button Event 발생시 Button Event에 따라 [Choose Yut] State나 [Random Yut] State으로 이동합니다.

3. 각 State진입 시 해당하는 윷 던지기 메소드를 실행 후 [Choose Horse] state로 이동합니다.

4. [Choose Horse] state 진입 시 말을 고르는 selectHorse() 메소드를 실행 후 해당 말의 위치, 이동해야 할 거리, 말의 완주 여부에 따라 다시 말을 고르는 ChooseHorse로 가거나, 방향 이동여부를 결정하는 Choose Direction, 아니면 말을 옮기는 MoveHorse로 이동합니다.

5. MoveHorse 진입 시 말 번호, 이동거리, 방향 전환 여부를 가지고 MoveHrs를 실행합니다. 그 후 말을 두고자 하는 위치의 정보를 getinfo(int,int),charAt(0)로 읽어 값에 따라 말을 잡는 Eat Horse, 말을 업는 Grouping Horse, 둘 다 아니면 EndGame으로 이동합니다.

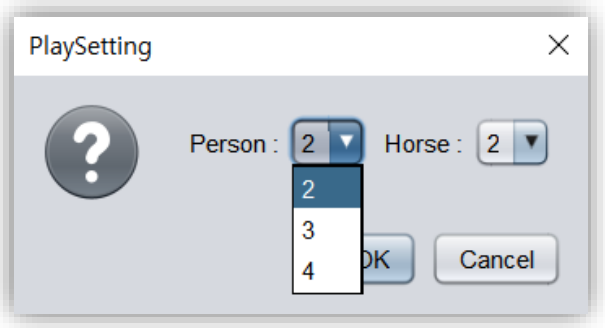
   5.1 EatHorse State의 경우 해당 플레이어가 다시 말을 던질 수 있으므로, 화면 출력을 갱신하는 repaint()를 실행하며 나가고, Play Yut State의 처음으로 돌아갑니다.

6. EndGame state에서는 endgame()으로 게임종료 여부를 확인하고, 나갈 때 repaint()로 화면을 갱신합니다. 그 후 게임 종료여부인 isEnd와 재시작 여부인 choice에 따라 프로그램을 완전히 종

료하거나 게임을 새로 시작하거나 플레이어 턴 변경 후 다시 Play Yut Game state로 돌아가 게임이 끝날 때까지 일련의 행동을 반복합니다.

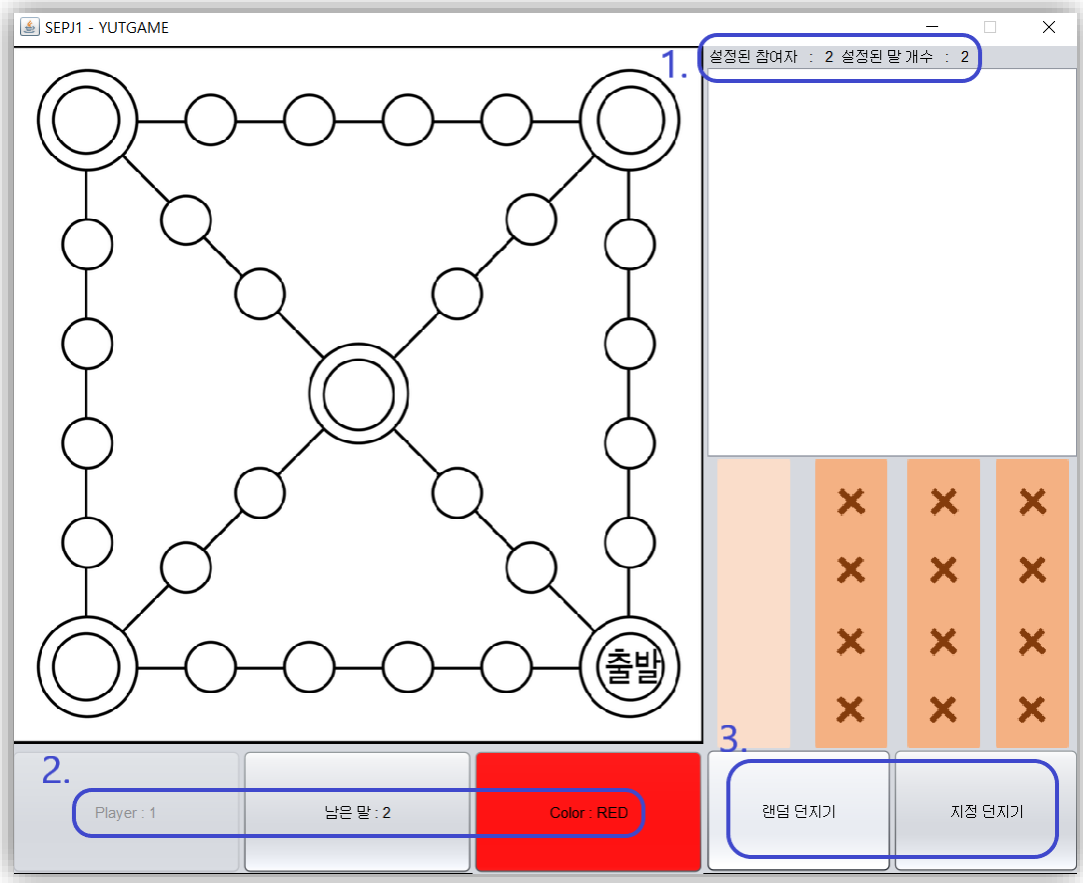
III.usage of program and the screen shots

1. 프로그램을 첫 실행하면 PlaySetting 창이 뜨며 Player와 Horse의 개수를 정할 수 있습니다. 그 후 OK를 누르면 게임이 시작됩니다.

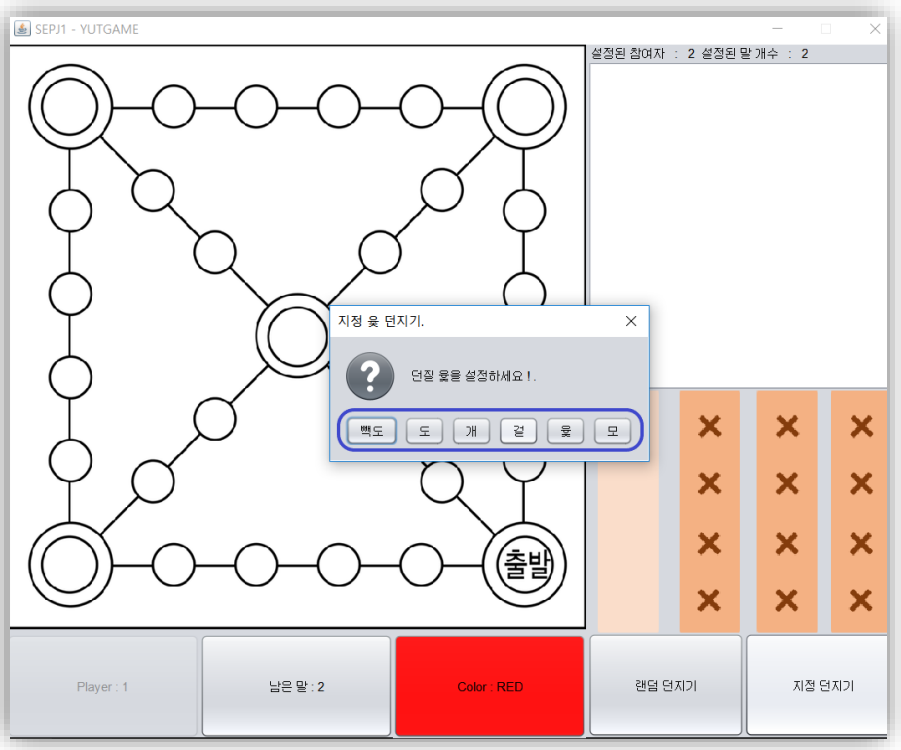


2. 전체적인 윷놀이 게임 UI이며 한눈에 현재 정보를 알 수 있습니다.

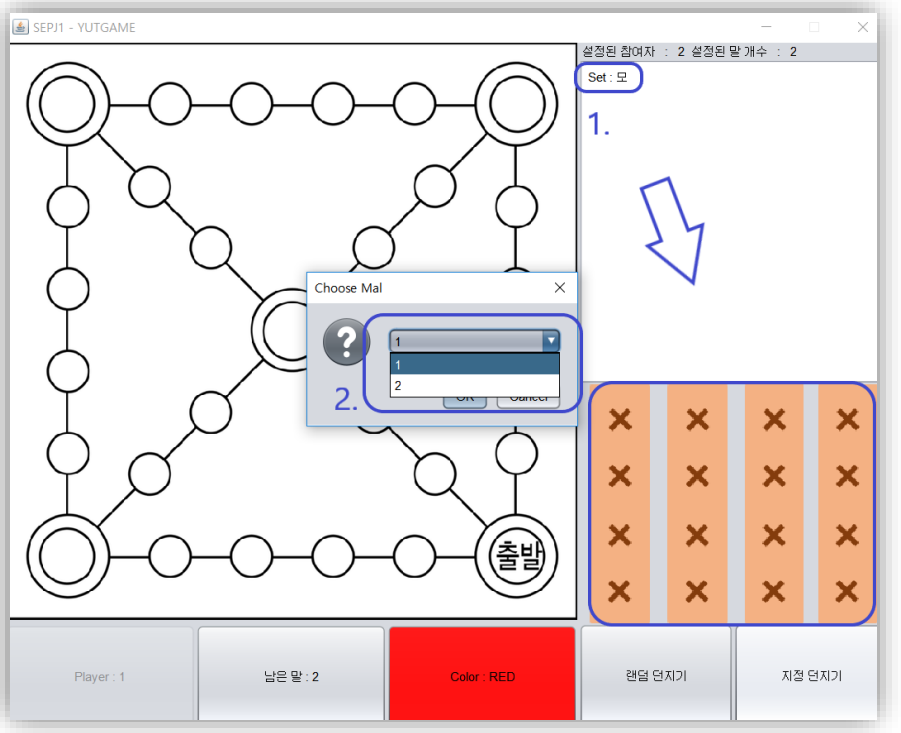
1) 현재 설정된 플레이어의 수와 말 개수를 라벨을 이용해 표시.
2) 현재 플레이어 번호와 해당 플레이어의 남은 말 개수, 플레이어 색깔을 버튼에 표시.
3) 윷을 던지는 버튼으로 랜덤 던지기는 임의의 윷이, 지정 던지기는 말을 고르는 창이 표시.



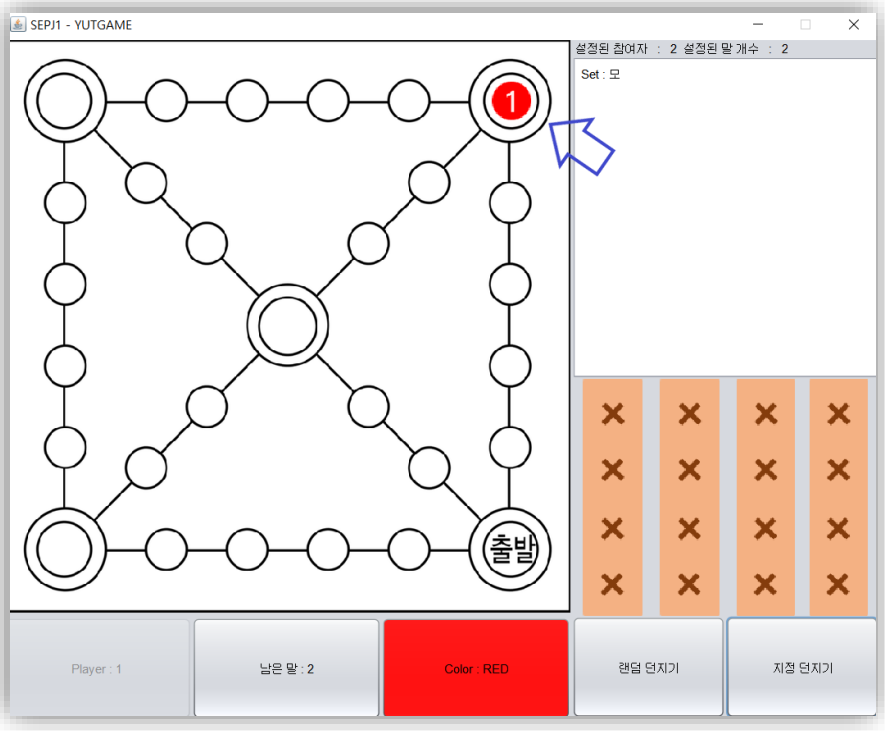
3. 지정 던지기를 눌렀을 때의 팝업 창이며, 6개의 윷 중 원하는 윷을 선택해 던질 수 있습니다.



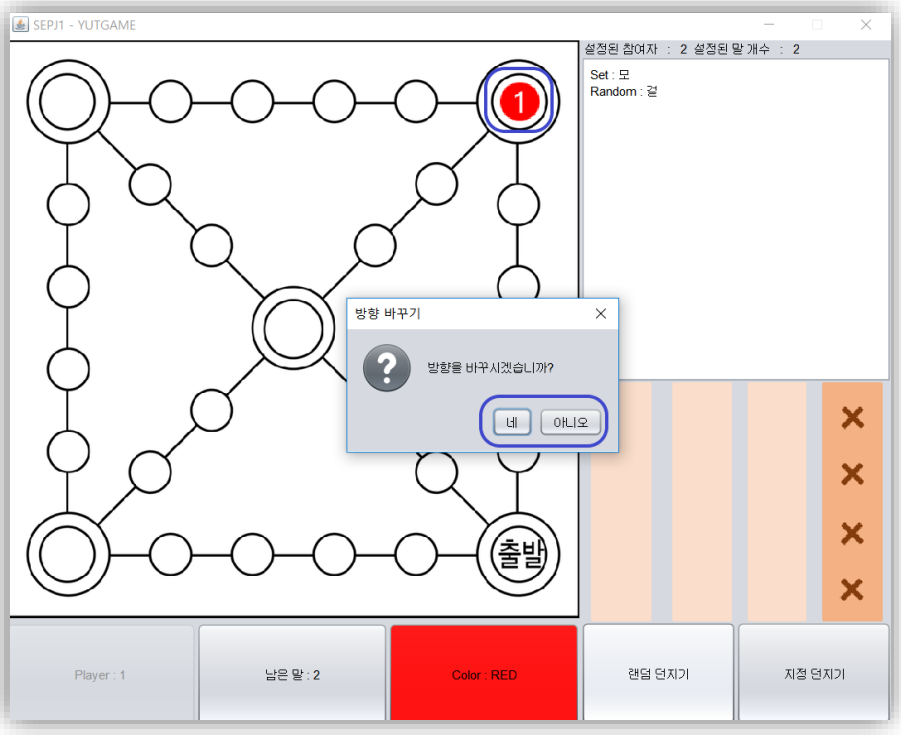
4. 윷 던지기를 실행 후의 화면입니다.
    1) 방금 던진 윷의 정보와 해당 윷 이미지가 표시됩니다.
    2) 어떤 말을 옮길지 선택할 수 있습니다.



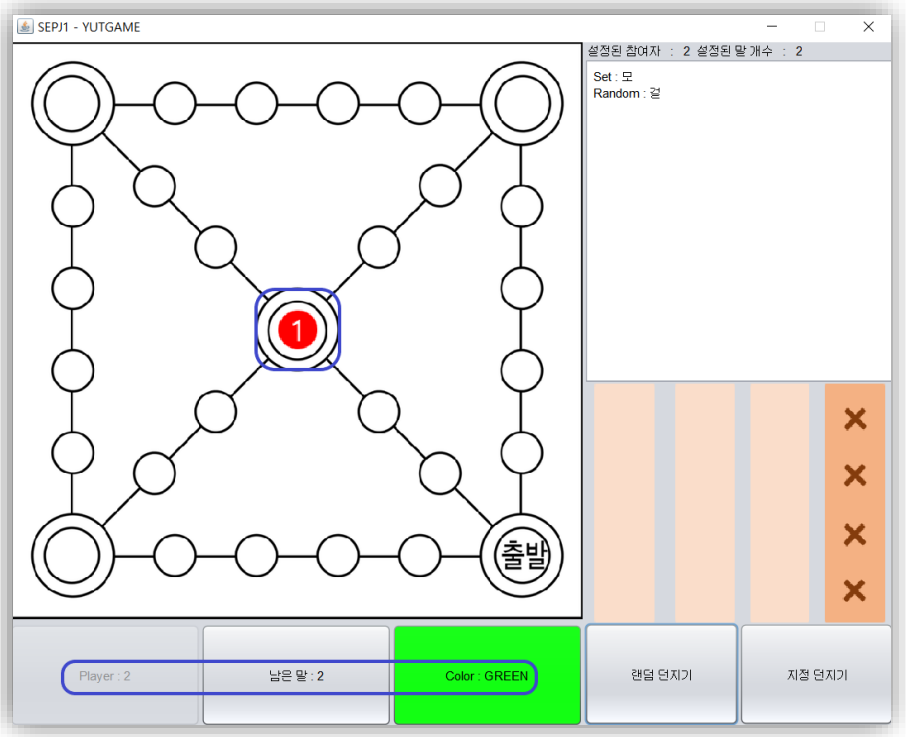
5. 플레이어가 선택한 대로 말이 옮겨진 모습입니다. (숫자1 은 말의 개수를 표시)



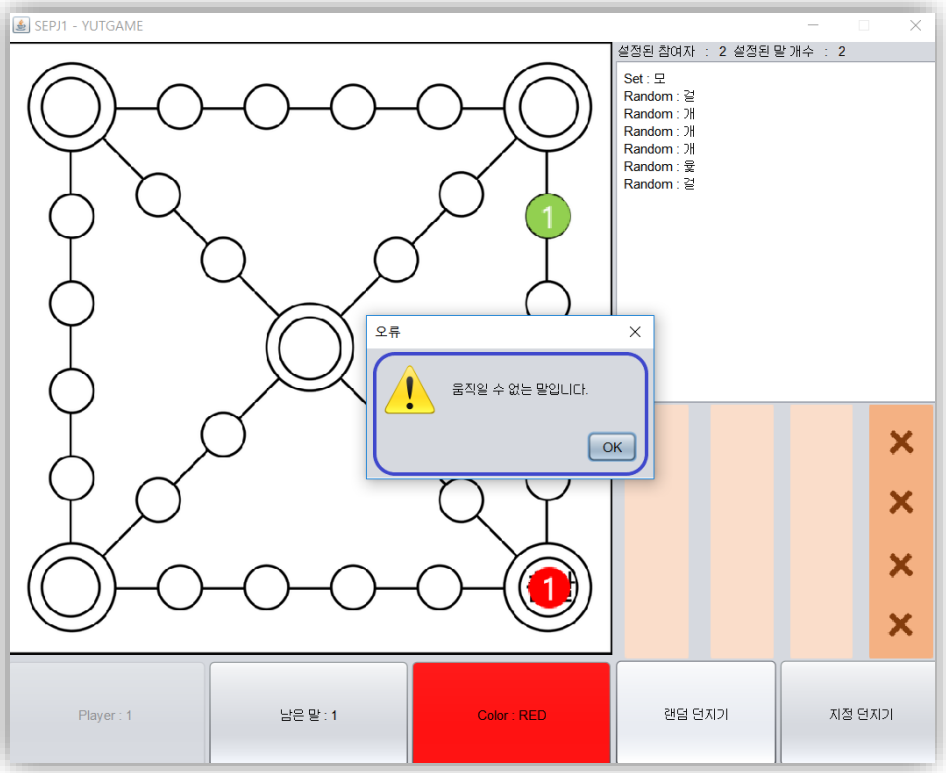
6. 모를 던져 턴이 바뀌지 않고 다시 턴을 진행, (상대방 말을 잡거나 윷이 나와도 동일) 방향을 결정할 수 있는 위치에 있기에 방향결정 여부를 묻는 창이 뜬 상황입니다.
('예' 를 누를 경우 대각선으로 '아니오' 를 누르면 왼쪽으로 이동함.)



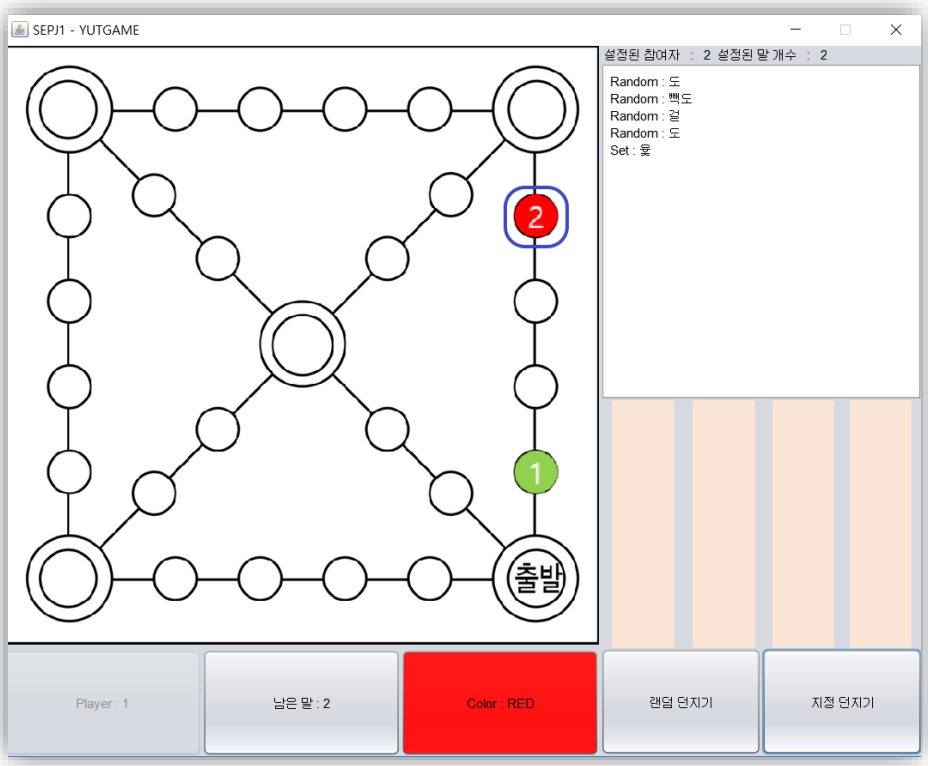
7. '6'에서 예를 눌렀을 때의 화면, 현재 플레이어 턴이 다음 플레이어로 넘어간 모습입니다.



8. 만약 플레이어의 1번 말이 골인한 후 1번 말을 옮기려 할 경우,
위와 같은 경고창이 뜨며 다시 말을 고르게 합니다.



9. 플레이어가 자신의 말을 업은 경우 위와 같이 업힌 말의 수가 변경됩니다.



10. 특정 플레이어가 모든 말을 골인 시킨 경우 위와 같은 메시지가 출력되며 '네'를 누를 경우 1번으로 돌아가며, '아니오'를 누를 경우 프로그램이 종료됩니다.