Software Engineering

Project Design

알파제로(Alpha zero) - 오목

Subject : Software Engineering

Member : 김이삭

조동일

김진범

서정은

김범수

양병일

목 차

1. 문제의 정의

1.1. 문제의 이해

1.1.1. 문제의 범위와 원인 파악

2015년 이세돌과 알파고와의 바둑 대련 이후 인공지능이 대두되었다. 다른 게임에도 인공지능 도입이 고려되고 있고, 연구되고 있다. 이러한 배경으로부터 AI가 탑재된 오목 프로그램이 나타나길 기대하고 있다. 오목을 두기 위해선 오목판, 오목 돌을 준비해야 하고 오목을 같이 둘 상대방도 있어야 한다. 기술의 발달과 현대화로 인해 기존의 전통적인 오목을 즐길 수 있는 시간이 줄어들고 있어 새로운 오목 시스템이 필요하다.

- 같이 즐길 상대방이 필요하다. 혼자서는 즐길 수 없다.
- 오목을 두기 위해 무겁고 부피가 큰 오목판과 오목 돌이 있어야만 즐길 수 있다.

1.1.2. 문제를 둘러싼 현황 파악

사용자는 단순한 컴퓨터와 오목을 두는 것이 아닌 사람과 같은 수준의 오목을 두고자 한다. 기존의 오목은 상대방이 필요하고, 오목판, 오목 알이 필요했다. 만약 두 전제조건이 없다면 게임을 할 수 없다. 2015년 알파고 이후 다른 분야에서도 인공지능에 대해 많은 연구를 진행한다. 사회적으로 이슈되고 많은 관심을 받는 만큼 오목 게임에도 '사람'과 같이 오목을 할 수 있는 시스템이 필요하다.

1.1.3. 시스템 조사 및 정보 수립(현재의 시스템 조사, 업무 흐름 정책 등 파악)

기존의 오목 시스템에서 사용자는 이겼던 수를 똑같이 두면 무한히 컴퓨터를 이길 수 있다. 컴퓨터의 도입과 개발자들의 노력을 통해, 원시적인 오목 준비물 없이 PC 환경에서 오목을 둘 수 있는 프로그램이 많이 개발되었다. 그러나 현재의 프로그램은 개발자가 입력해두지 않은 수를 스스로 학습하는 능력이 없어 매번 사용자에게 패배한다. 현재 실제 사람과 같은 오목을 할 수 있는 시스템이 아니다.

1.1.4. 현 시스템의 이해 (면담과 서류로 심층 분석) (ex. 고객 상담, 현업의 분석, 작업의 체험)

● 같이 즐길 상대방이 필요하다.
- 오목을 즐기기 위해서는 상대방이 반드시 필요하기 때문에 혼자서 즐길 수 없다.

● 오목판, 오목알이 필요하다.
- 무겁고 부피가 큰 오목판과 오목알이 있어야만 즐길 수 있다.

● 시간과 공간적 제약이 있다.
- 오목을 즐기기 위해서는 두 사용자 간의 시간 협의와 별도의 공간이 필요하다.

1.1.5. 시스템의 정의

- 상대방 없이 혼자서도 즐길 수 있는 Python 기반의 PC 환경 인공지능 오목 시스템 개발

1.1.6. 우선순위

1) 머신러닝 도입
-> 오목게임에서 스스로 학습할 수 있는 시스템사용자를 제공해야 한다.
2) 접근성 높은 UI / UX 구성
-> 콘솔로 구성되어 있는 간단한 룰만 가진 게임에서 사용자들이 친근하게 느낄 수 있는 UI 를 제공해야 한다.

1.2. 대책 수립

1.2.1. 신규 시스템의 목표 설정

- 인공지능의 실수를 최대한 줄이고 효율성을 끌어내며, 인공지능을 상대로 사용자가 오목을 둘 수 있도록 하여 개인이 오목에 쉽게 접할 수 있도록 한다.
- 처음 사용하는 사용자가 오목에 쉽게 접할 수 있도록 가시성 높은 UI를 제공해야 한다.

1.2.2. 해결 방안 모색

- 1:1 구도이기 때문에 모든 상황을 목록화해야 한다.
- 많은 대전을 통해 인공지능을 학습시켜, 최악의 경우 중 최선의 경우를 선택할 수 있도록 한다..

1.3. 시스템 정의

1.3.1. 문제의 기술

- 기존의 단순한 오목 게임 프로그램에서 머신러닝을 추가한 인공지능 게임을 제공하여, 쉽게 질리지 않을 수 있는 게임을 제공(한다. 예정)
- 접근성이 떨어졌던 UI / UX를 사용자 입장에서 보다 쉽게 접근할 수 있도록 프로그램을 개선(하였다. 예정)

1.3.2. 시스템의 필요성

사용 가능한 데이터의 다양성 증가, 분석 비용의 감소, 강력해진 분석.

- 이 모든 상황을 종합해보면 아무리 규모가 큰 데이터라도 분석 모델을 자동으로 빠르게 생성함으로써 복잡한 분석에서 정확한 결과를 도출할 수 있다. 또한, 이러한 결과를 이용하여 수익성이 높은 기회를 찾아내거나 미지의 위험을 회피할 수 있다.

1.3.3. 시스템의 목표

- 사용자와 비등하게 게임 할 수 있는 인공지능 오목 프로그램 개발.

1.3.4. 제약사항

아직 인공지능의 수준이 오목 규칙만 아는 수준이므로 많은 대전을 통해 학습을 시켜 오목에 익숙해지도록 해야 함.

1.3.5. 시스템의 제공 기능

- 11X11의 바둑판을 GUI로 출력하고, 사용자가 돌을 두려는 위치를 마우스 입력을 통해 전달 받는다.
- TensorFlow 같은 머신러닝 모듈을 통해 인공지능을 학습 시킨다.

1.3.6. 사용자의 특징

앞서 설명한 두 제약 조건(원시적 준비물, 상대 사용자)없이 오목 게임을 즐기려고 하고, 단순한 상대가 아닌 높은 난이도의 AI와 오목 대결을 하길 원한다.

1.3.7. 개발, 운용, 유지보수 환경

- 현재(2019년 11월) 안양대학교 컴퓨터공학전공 학생들로 구성된 인력으로 개발을 진행한다. 기존의 오목룰을 유지하고, 버그 없이 실행될 수 있도록 설계하고, 많은 테스트를 거쳐 신뢰성이 높은 시스템을 유지하도록 한다.
- 별도의 서버 시스템 구축을 요구하지 하지 않으므로, stand-alone 환경에서 운용되는 것을 전제로 한다. 또한 개발(+운영?) 환경의 동일 플랫폼 사용으로 환경 차이에 따른 이슈를 최소화하도록 한다.
- GitHub를 통한 형상관리 및 문서화를 통해 향후 유지보수성을 높일 수 있도록 한다.

1.4. 타당성검사

1.4.1. 경제적 타당성

ㄱ.투자 효율성

- 오목 프로그램은 현재도 많은 사람들이 즐기고 있고, 단점들을 개선한다면 이용자들의 수가 지금보다 대폭 증가할 것이다.

ㄴ. 시장성

- 현재까지의 오목 프로그램은 많은 단점을 가지고 있기 때문에 이전 프로그램을 사용하던 사용자와 AI가 포함된 오목을 고대하던 사용자들을 끌어올 수 있을 것이다.

ㄷ. 비용과 수익의 비교

- 머신러닝 라이브러리는 이미 오픈소스로 사용이 가능하여 경제적 비용은 크게 들지 않지만 개발에 성공한다면 수익은 상당히 클 것이다.

1.4.2. 기술적 타당성

ㄱ. 사례 연구

- 현재 두 명의 사용자가 할 수 있는 무료공개된 오목 프로그램의 조회수와 이용자 수가 적지 않음.

ㄴ. 실패 사례 연구

- 이후로 AI 기능을 탑재하였지만 잦은 에러와 가시성 낮은 UI로 인해 인지도가 낮았음.

ㄷ. 모의실험

- 모의 실험한 결과 AI와 플레이를 함에도 에러가 발생하지 않았고, AI가 예측 불가능한 다양한 수를 둘 수 있게 되었음.

1.4.3. 법적 타당성

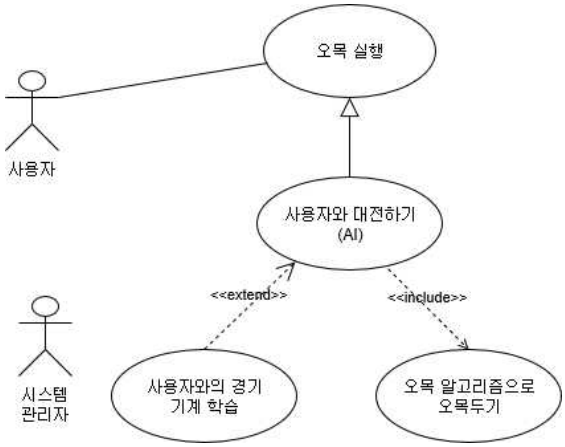
사용 도구들의 법적 권한

- 목표 프로그램 개발을 위해 사용한 툴인 PyCharm은 무료로 오픈 되어있고 기반이 될 소스코드는 MIT 라이센스가 적용되어 있는데 MIT 라이센스는 이 소프트웨어를 누구라도 무상으로 제한없이 취급해도 좋다고 나와 있기 때문에 법적 문제를 야기하지 않을 것임.
- 국가법령정보센터에서 지적 재산권 및 컴퓨터프로그램 보호법을 확인한 결과 법적으로 타당하다.

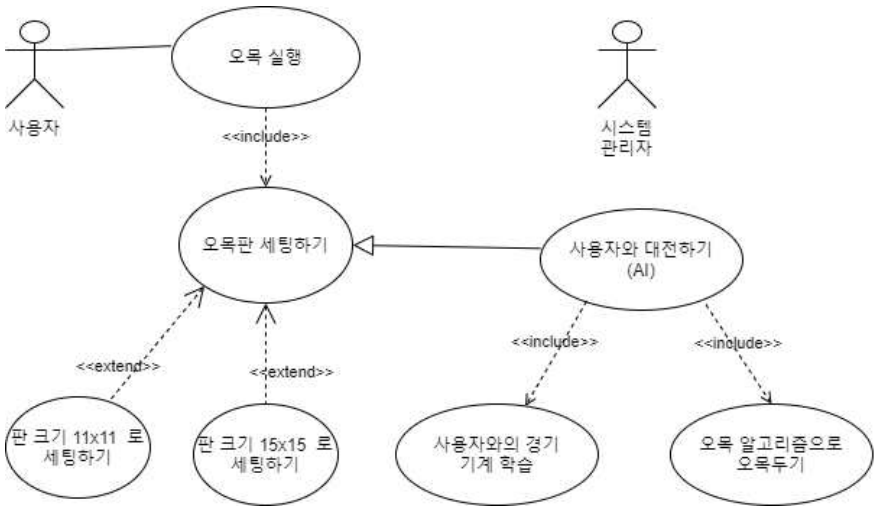
2. 사용 사례

2.1. 사용 사례 (use case)

use case - version 1.1



use case - version 1.2



2.2. 사용사례 작성

액터측 액션

1. 오목 실행하기

3. 게임 시작 버튼을 클릭

7. AI와 대전하기

시스템측 반응

2. 시작화면 창을 디스플레이

4. 시작 버튼을 숨기고
   11X11, 15X15 판 크기 선택 버튼 출력

5. 오목판 디스플레이 불러오기

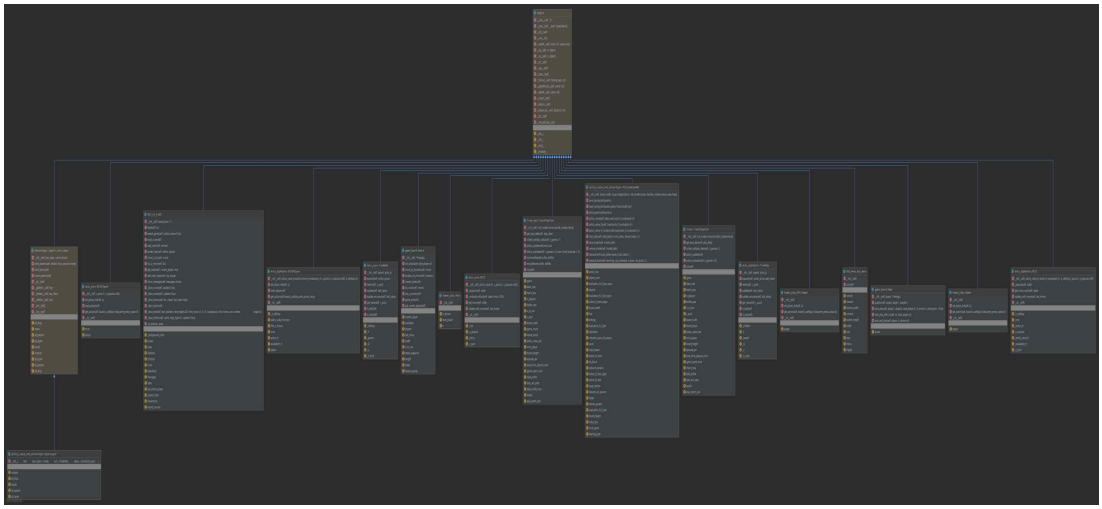
6. 사용자와 대전하기

6a. 세팅된 판 기준으로, 규칙 등 AI 기계학습

3. 객체 모델링

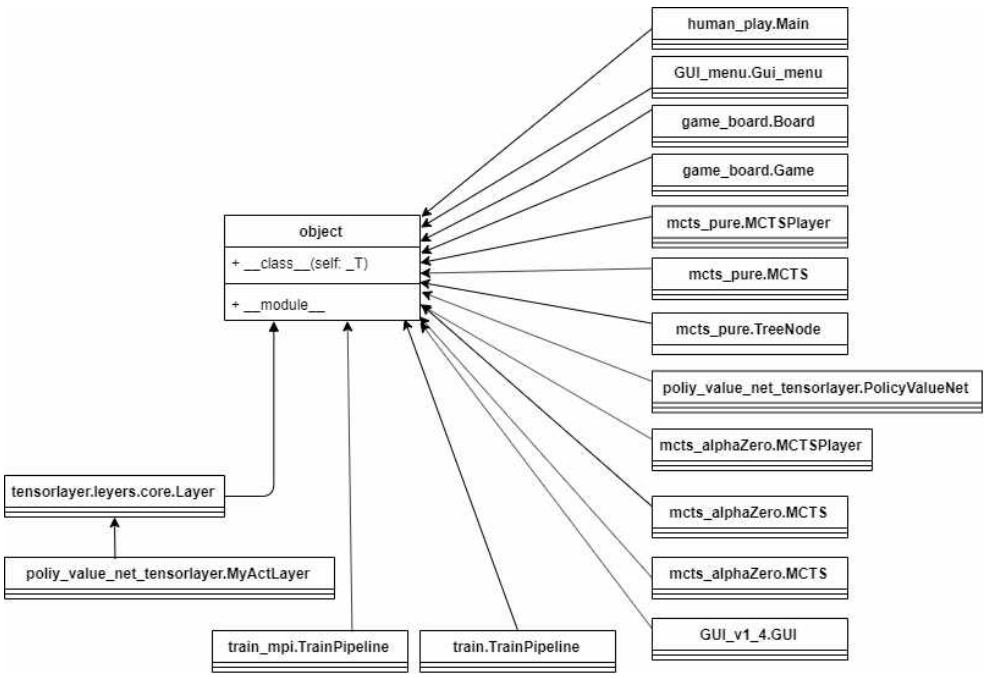
3.1. 클래스 다이어그램1

IDLE PyCharm 에서 뽑은 클래스 다이어그램



3.2. 클래스 다이어그램2

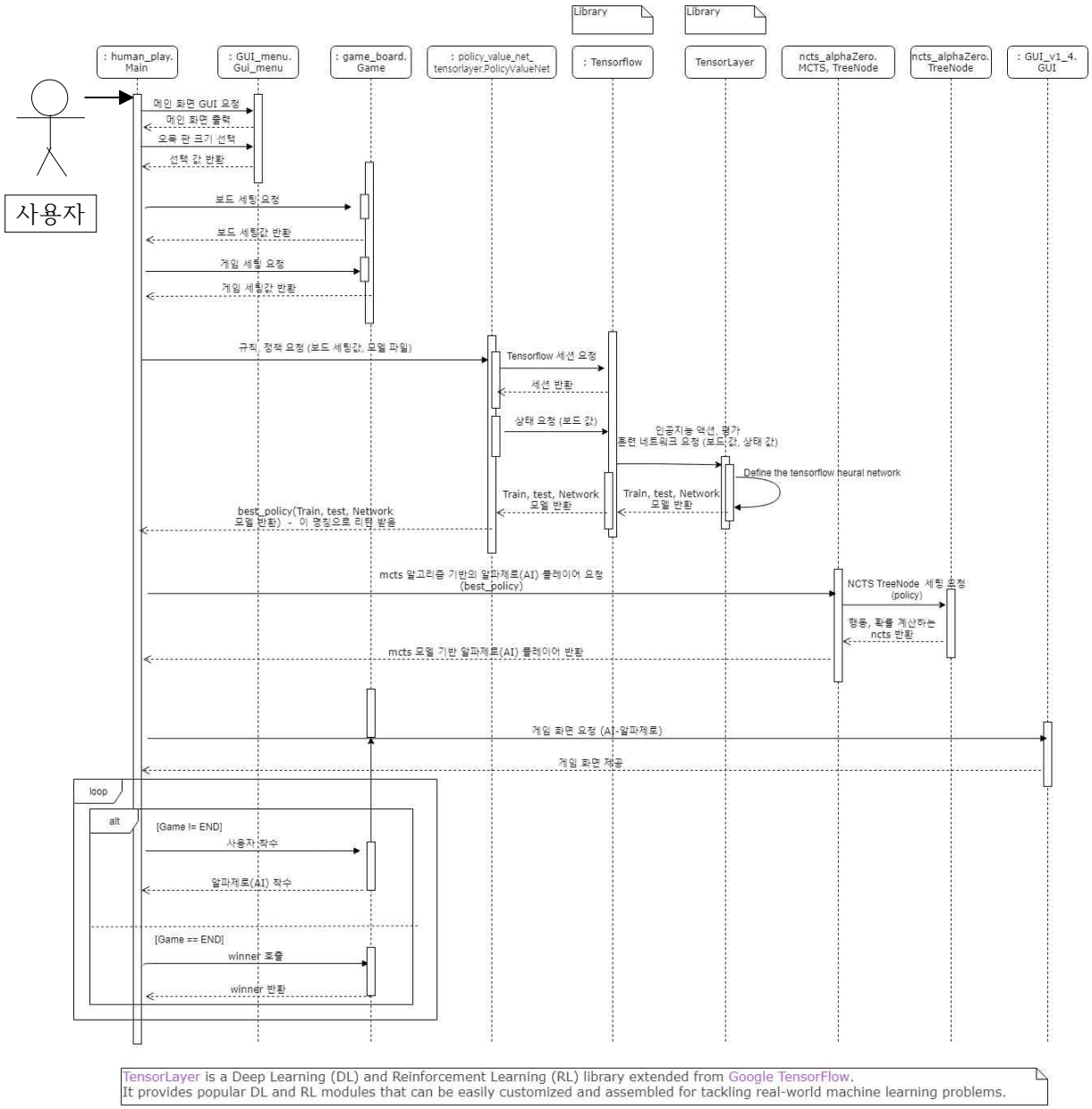
앞의 클래스 다이어그램은 IDLE에서 바로 추출한 다이어그램으로 클래스 다이어그램이 잘 보이지 않는다. 따라서 그림으로 클래스 이름만 담은 다이어그램을 표현했다.



4. 동적 모델링

4.1. 순차 다이어그램 (Sequence Diagram)

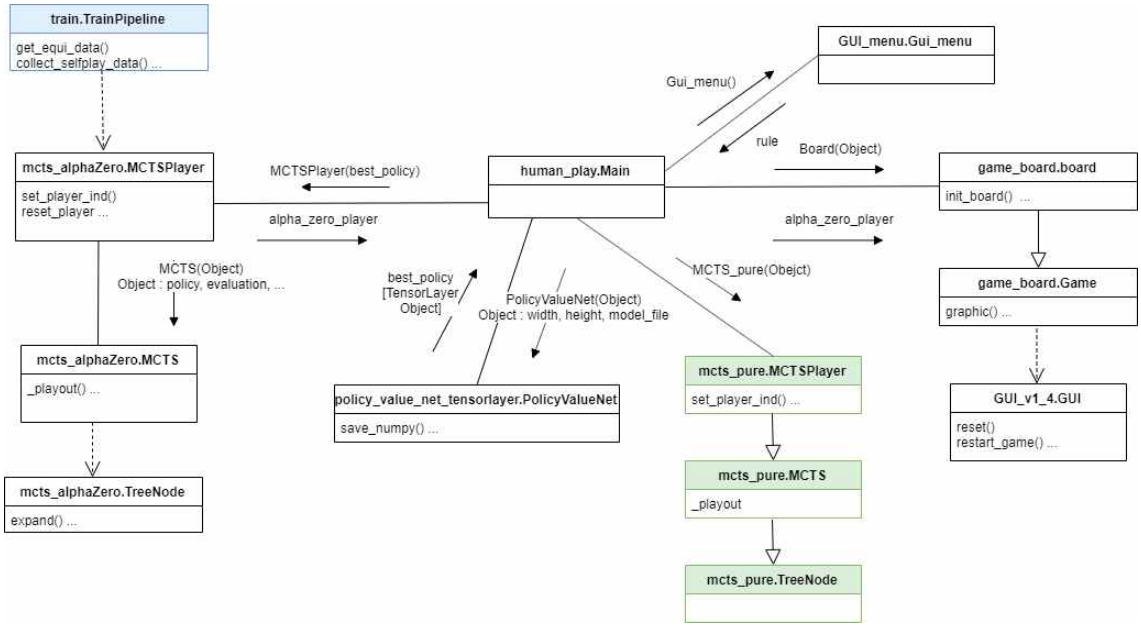
클래스의 호출, request, response를 시간의 흐름에 따라 나타낸 상호작용 다이어그램이다. 객체 간의 동적 상호작용을 시간적 개념을 중심으로 모델링 하는 것을 말한다. 다이어그램의 수직 방향이 시간의 흐름을 나타낸다.



TensorLayer is a Deep Learning (DL) and Reinforcement Learning (RL) library extended from Google TensorFlow.
It provides popular DL and RL modules that can be easily customized and assembled for tackling real-world machine learning problems.

4.2. 협동 다이어그램 (Collaboration Diagram)

협동 다이어그램은(Collaboration Diagram)은 인스턴스들이 어떻게 협동하는지를 묘사한다. 하나의 협동 인스턴스 집합에 포함된 인스턴스들의 협동 모델을 직접적으로 표현한다.



5. 시스템 설계

5.1. 시스템 설계 <별첨>

[별첨] 시스템 설계서

6. 구현

6.1. 추가된 기능 구현 목록

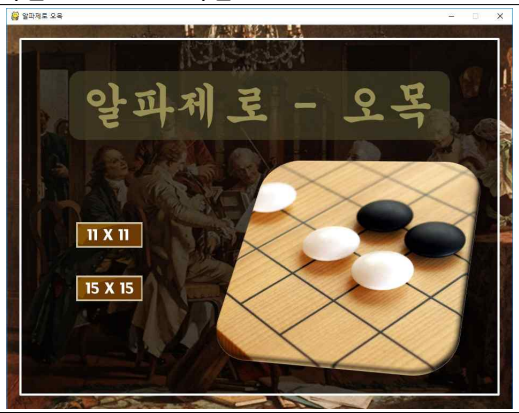
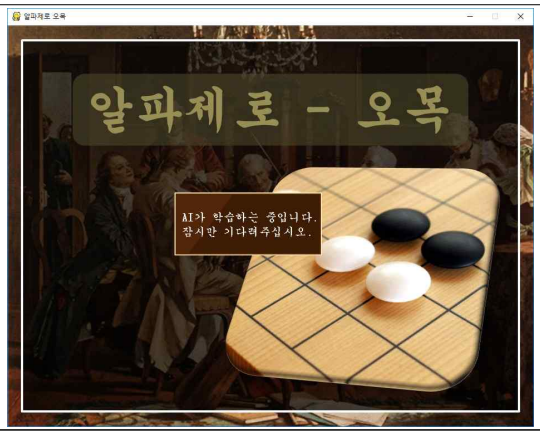
기존의 소프트웨어에서 자체적 새로운 요구 사항에 의해 추가된 기능을 구현.

1) 게임 배경음악 삽입
 - 메인 화면 배경음악 삽입
 - 게임 화면 배경음악 삽입
2) 게임 메인화면 및 오목판 선택 화면 구현
 - 메인 화면
 - 게임 시작 버튼
 - 11x11 버튼
 - 15x15 버튼
3) 오목 대전시 타이머 추가
 - 오목 대전시 오목을 두지 않으면 타이머 적용하여 턴이 상대방에게 넘어감
4) 오목 대전 후, 결과 메시지 알림 창 나타내기
 - 오목 대전 이후 나타나지 않았던 결과를 메시지 알림 창을 띄어 표현함

*모든 것을 GUI로 구현함

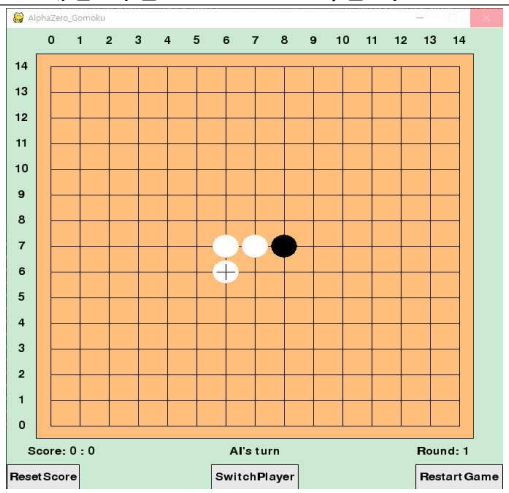
6.2. 구현 화면

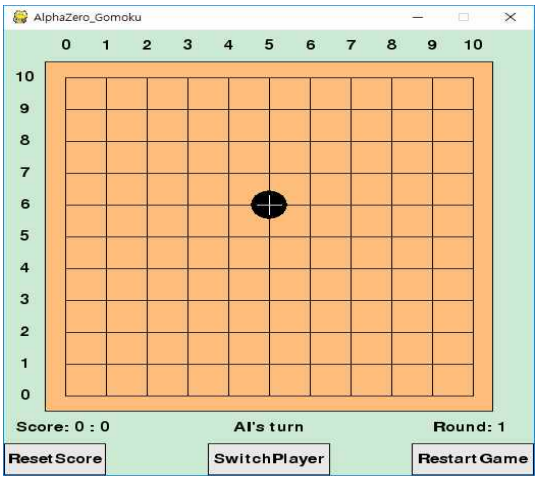
GUI 메인 화면 - 게임 시작 버튼

메인 화면 - 11 x 11 버튼, 15 x 15 버튼

메인화면 - 로딩


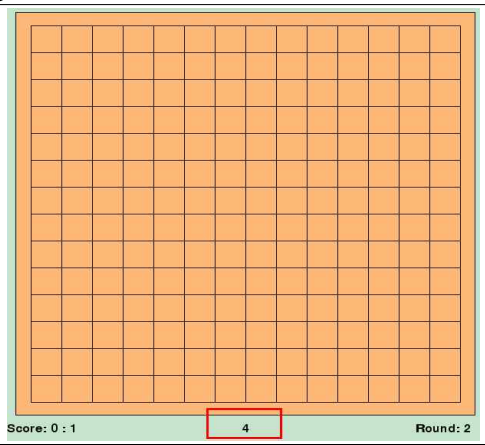
6.2. 구현 화면

GUI 게임 화면 - 15x15 오목판 / 11x11 오목판

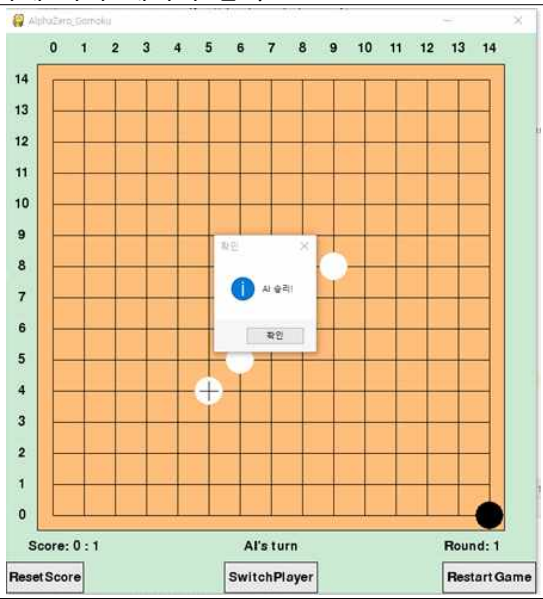




게임 화면 : 타이머 기능



메시지 창 : 대전 결과에 따라 메시지 출력



7. 테스트 및 검증

7.1. 단위테스트 결과서 〈별첨〉

[별첨] 단위 테스트 결과서