Java Programming: Assignment 2

Seung-Hoon Na

June 8, 2020

1 Manhattan Board Game

본 과제에서는 맨해튼 보드게임을 1인용 (인공지능기반), 2-4인용 (네트워크 기반) 상에서 구동되도록 java로 프로그래밍 하는 것을 목표로 한다.

1.1 게임 규칙

자세한 게임 규칙은 아래의 링크를 참고하라.

`https://www.youtube.com/watch?v=98wxBAL0ZA4&t=80s`

Fig. 1은 맨하탄 보드 게임을 위한 구성 화편의 예를 보여준다 (화면 본인이 직접 제작한 게임 화면을 사용하여도 무방하다.) 화면의 좌상단에 '메인 보드' 패널이 배치되어 있고, 메인 보드는 총 6개의 도시 (cities)으로 구성되며, 각 도시는 9개의 스팟(spot)으로 구성된다. 플레이어는 해당 턴에 이들 중 한 도시의 스팟을 선택하여 블록을 쌓을 수 있다.

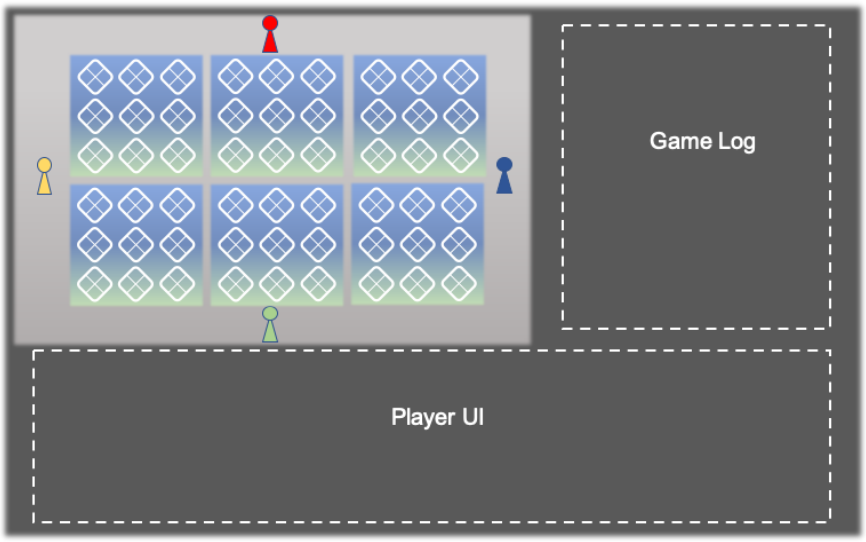


Figure 1: Manhattan Board 게임판

게임를 위해 추가로 총 45장의 빌딩 카드와 24개의 빌딩 블록이 필요하여, 이들 아이템들의 모양 및 갯수 등은 Fig. 2을 참조하라 (본인이 제작한 아이템을 사용해도 된다).

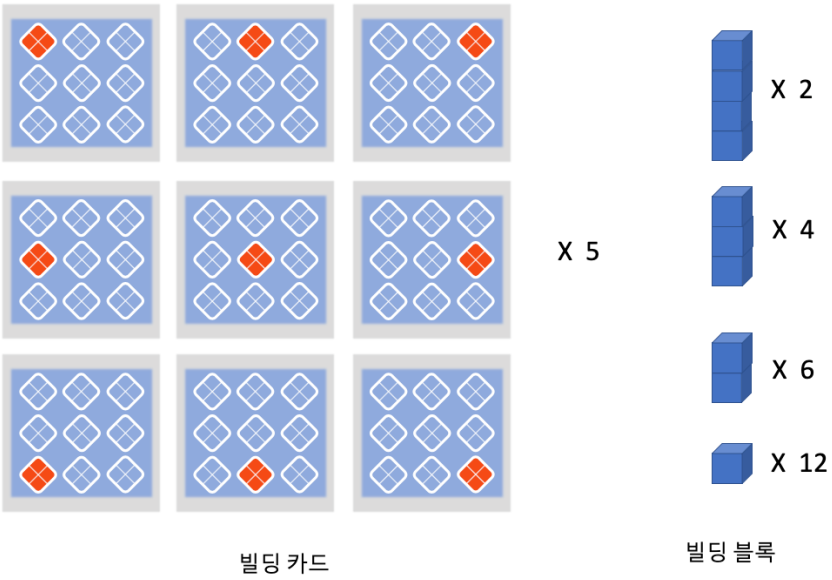
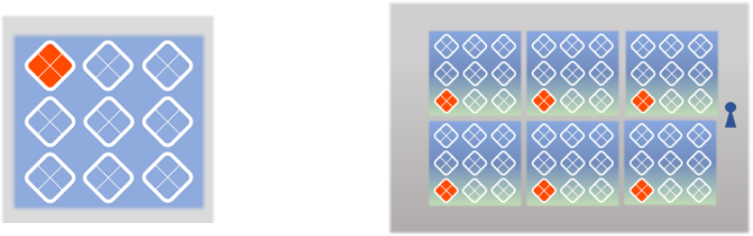


Figure 2: 게임을 진행하기 위해 필요한 아이템들

게임은 총 4명의 플레이어로 진행되며, 모든 플레이어는 동, 서, 남, 북 에 각각 랜덤으로 배치되며, 각 플레이어는 다른 색상의 게임 말 (token)을 갖는다. 게임은 총 4라운드로 진행되며, 임의의 플레이어가 시작 플레이어가 되고 시작 플레이어는 각 라운드마다 시계 방향으로 변경된다. 각 라운드 시작 시, 모든 플레이어에게 각각 총 45장의 빌딩 카드 중 4개 씩 나누어 준다. 또한, 각 플레이어는 위의 그림과 같이 총 24개(각 플레이어 마다 24개 씩 제공)의 빌딩 블록 중 해당 라운드에 사용할 6개의 블록을 선택한다. 시작 플레이어부터 시계방향으로 번갈아 가며 게임이 진행되며, 모든 플레이어는 본인의 턴에만 행동 할 수 있다. 각 플레이어는 본인의 턴에 빌딩 카드를 한 장 소모하여 총 6개의 도시 중 원하는 도시을 선택하고, 빌딩 카드에 표시된 위치(빨간색)에 대응되는 도시내 스팟에 자신이 원하는 블록을 1개 선택하여 쌓을 수 있다. 빌딩 카드에 표시된 위치는 플레이어가 위치한 방향을 기준으로 하며 빌딩 카드는 회전할 수 없다.

예를 들어, Fig 3은 동쪽(오른쪽)에 위치한 플레이어 (East player)가 선택한 빌딩카드 및 이때 블록을 쌓을 수 있는 각 도시별 스팟을 보여준다.



(a) 선택한 빌딩 카드

(b) 블록을 놓을 수 있는 스팟

Figure 3: 동쪽 플레이어 (East player)가 선택한 빌딩카드에 대응되는 각 도시내의 스팟. a) 선택한 빌딩카드, b) a)의 빌딩카드에 대응되는 블록을 쌓을 수 있는 도시의 스팟

맨해튼 보드 게임의 세부 규칙은 다음과 같다.

(A) 게임은 총 4명의 플레이어로 진행되고, 모든 플레이어는 Fig 1의 게임 말과 같이 동, 서, 남, 북 에 각각 랜덤으로 배치된다. 게임 판은 각 9개의 스팟 (spot)을 가진 총 6개의 도시 (city)로 이루어져 있다.

(B) 게임은 총 4라운드로 진행되며, 임의의 플레이어가 시작 플레이어가 되고 시작 플레이어는 각 라운드마다 시계 방향으로 변경된다.

(C) 각 플레이어는 매 라운드 시작 마다 4개의 빌딩 카드를 랜덤으로 뽑고, 24개의 블록 중에서 이번 라운드에 사용할 6개의 빌딩 블록을 선택한다 (빌딩 카드와 블록 모양은 Fig 2참조). 빌딩 블록은 1층짜리부터 4층짜리 블록으로 구성된다. 블록은 라운드 동안 모두 사용해야 하며, 라운드 중간에 블록을 바꿀 수 없다. 각 플레이어는 본인의 턴에 반드시 빌딩 카드 한 장과 빌딩 블록 하나를 사용하여야 한다.

(D) 각 도시 내의 스팟에 빌딩 블록을 놓을 수 있다. 플레이어는 본인의 턴에 빌딩 카드를 한 장 소모하여 원하는 도시 내의 빌딩 카드가 가리키는 스팟에 본인이 소유한 블록 중 원하는 블록을 선택해 쌓을 수 있다 (빌딩 카드를 소모한 후에는 다시 랜덤으로 남은 빌딩 카드들 중에서 한 장을 보충한다. 소모한 빌딩 블록은 보충 하지 않는다.) 기존에 다른 플레이어가 스팟에 이미 블록을 쌓은 경우에는 해당 스팟에 가장 많은 층을 보유한 플레이어보다 많거나 같은 수의 층이 쌓일때만 블록을 새롭게 추가할 수 있다. 즉, **블록이 스팟에 추가될때는 동일색깔의 층수가 다른 색깔의 층수보다 많거나 같아야 한다**.
형식적으로 표현하면, 해당 스팟을 s이라고 하고, $s.blocks$는 동서남북 각 플레이어가 소요한 블록의 층 (layers) 수의 합을 가리키는 4개의 원소로 구성된 배열이라고 하자. 배열 표기를 이용하여, $s.blocks[i]$를 i번째 플레이어가 보유한 블록 층수 합이라고 하자. 이때 새롭게 추가하고자 하는 주어진 블록을 b, 그리고 $b.player \in \{1, 2, 3, 4\}$를 해당 플레이어의 인덱스라 하고, $b.layers \in \{1, 2, 3, 4\}$를 블록 b의 층수라고 하자. b가 해당 스팟 s에 추가될 조건은 다음과 같다.

$$s.blocks[b.player] + b.layers \geq \max_{i \neq b.player} s.blocks[i]$$

위의 조건에 따라 블록을 추가하면서, 해당 스팟에 가장 마지막으로 블록을 쌓은 플레이어가 빌딩을 소유하게 된다.

(E) 모든 플레이어가 6개의 빌딩 블록을 다 사용한 경우 라운드가 종료되며, 다음 라운드의 시작 플레이어는 이번 라운드의 시작 플레이어의 왼쪽 플레이어가 된다.

(F) 점수 계산 (각 라운드 마다)

1 전체 중 가장 높은 빌딩을 소유한 플레이어 +3 (동점인 경우 아무도 받지 못함)

2 각 도시별로 가장 많은 빌딩을 소유한 플레이어 +2 (동점인 경우 아무도 받지 못함)

3 각 플레이어가 소유한 빌딩 1채 당 +1

(G) 4 라운드를 마치고 가장 높은 점수를 얻은 플레이어가 승리.
동점이면 가장 높은 빌딩을 소유한 플레이어가 승리.
그래도 동점이면 가장 많은 빌딩을 소유한 플레이어가 승리

1.2 Problem: 맨해튼 보드 게임 구현

앞 절의 그림들을 참고하여 아래의 가이드라인을 따라 간단한 맨해튼 보드게임을 Java로 작성하시오 (가이드라인에 언급되지 않은 부분은 자유롭게 구현 한다.)

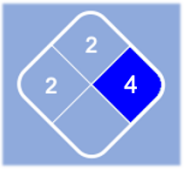
시작하기 전 화면은 다음과 같은 메뉴로 구성되어야 한다.

1. 1인용 게임
2. 2-4인용 네트워크 게임 (게임 생성)
3. 2-4인용 네트워크 게임 (게임 참여)
4. 환경설정
5. 종료

메뉴의 각 아이템은 버튼(이미지 버튼)으로 구성하며 화면의 수직/수평 중앙정렬로 배치 한다. 각 플레이어의 위치(동, 서, 남, 북), 초기 라운드 시작 플레이어는 랜덤으로 부여 받는다. 모든 플레이어의 행동은 게임 화면 오른쪽의 Game Log 를 통해 나타난다. 게임 화면 아래의 Player UI 에는 본인이 소유하고 있는 빌딩 카드, 빌딩 블록, 이번 라운드의 시작 플레이어, 현재 라운드 수 , 이번 턴의 행동 플레이어 등을 표시한다. 게임 화면 왼쪽 위의 게임 판에는 다음과 같이 모든 플레이어의 빌딩 건설 상황을 표시 할 수 있다. (아래의 방법 중 하나를 선택하여 사용)

1. 각 스팟의 동, 서, 남, 북 칸에 각 플레이어의 빌딩 건설 상황을 표시한다. 즉, 블록이 1개이상 있는 플레이어에 대해 Fig 5의 예처럼 스팟그림의 해당 칸에 플레이어가 보유한 블록의 층수 (number of layers)를 기술한다. 1개 이상의 블록이 있는 스팟은 다른 색상으로 0개 블록 스팟과 구분한다. (해당 스팟의 건물을 소유한 플레이어를 다른 배경색으로 표시하는 것을 권장한다 (optional))

(a) 블록이 없을때의 스팟그림. 층수 0인 경우는 표시할 필요가 없다



(b) 각 플레이별로 블록이 쌓인 상태의 스팟 그림. 각 칸의 숫자는 해당 플레이어가 소유한 블록의 층수 (layers)를 의미한다

Figure 4: 스팟그림의 각 칸에 해당 플레이어가 놓은 블록의 층수를 표시한 그림:(a) 블록이 없을때의 스팟 그림 상황. (b) 플레이어별 블록의 층수가 명시된 상황

2. 각 스팟 를 버튼으로 구성하여 버튼 클릭 시에 팝업 창으로 각 플레이어의 빌딩 건설 상황을 표시한다.
3. 기타 방법

게임 화면 내의 원하는 위치에 각 턴의 제한시간을 출력하고, 제한 시간 내에 플레이어가 행동하지 않으면 게임 규칙에 어긋나지 않도록 랜덤으로 행동을 수행한다.

- 1인용 게임의 경우 난수 생성에 기반한 간단한 인공지능을 통해 구현한다.(* 추가 점수)
- 2-4인용 게임의 경우 동일 컴퓨터상에서 localhost를 이용하여 개별 프로그램으로 테스트가 되어야 한다.

(교재에 기술되지 않은 내용들은 각자 다른 참고자료를 통해 정보를 얻을 것.)
아래는 구현과 관련한 상세 comments 이다.

- **1인용 게임**: 간단한 인공지능을 통해 1인용 게임을 진행한다. 인공지능 플레이어는 게임 규칙에 어긋나지 않는 행동을 랜덤으로 수행한다.
- **2-4인용 네트워크 게임**: 단일 컴퓨터상에서 4인용 게임을 플레이 할 수 있도록 한다. 최대 4명의 플레이어가 게임에 참여 할 수 있고, 참여 플레이어가 4명 미만인 경우 나머지 플레이어를 인공지능 플레이어로 대체 한다.
- **2-4인용 네트워크 게임 (게임생성)**: 현재 플레이어의 컴퓨터가 서버가 되며, 게임 대기상태 윈도우가 새로 나타나, 상대방의 게임 참여를 기다린다. 상대방이 게임을 참여하고, 서버 컴퓨터의 플레이어가 게임시작 버튼을 누르게 되면 게임이 시작된다.
- **2-4인용 네트워크 게임 (게임참여)**: 현재 플레이어의 컴퓨터가 클라이언트가 되고, 메뉴를 선택하면 게임대기상태 윈도우가 실행되며, 이 윈도우는 해당 서버의 IP를 입력하는 단일라인 텍스트 박스와 접속 버튼, 게임 접속 상태의 로그를 보여주는 다중라인 텍스트 박스로 구성된다. 사용자의 IP입력 후 접속버튼을 누르면 접속버튼은 임시로 비활성상태가 되면서 게임의 참여를 시도하고, 다중라인 텍스트 박스에서 네트워크 접속 진행상태가 나타난다. 만약 해당 서버의 IP가 없거나 네트워크 에러 발생 시 에러 메시지를 출력하고, 접속버튼은 다시 활성화되며 사용자의 IP변경 등 수정입력을 기다린다. 성공적으로 접속되면 서버 컴퓨터의 플레이어의 게임시작을 기다리게 된다.
- 2-4인용 네트워크 클라이언트-서버 프로토콜 (Hint): 네트워크는 TCP/IP 상에서 구동되도록 하고, 이를 위해 적절한 프로토콜이 정의가 되어야한다. 클라이언트-서버 프로토콜은 최소한으로 하되, 유한 상태 오토마타 수준으로 미리 설계가 되어야 한다. 서버 및 클라이언트 각 상태를 정의하고, 단순화를 위해 메시지는 텍스트 문자열로 정의하라. 추가로, 서버와 클라이언트 각 상태마다 메시지 종류에 따라 다음 상태가 어떻게 바뀌는지도 미리 정의해야 한다. 메시지는 종류 (type) 및 인자 (arguments)로 구성시키는 것이 일반적이다. 각 메시지의 구성요소는 해당 프로토콜 정의에 맞게 설계되어야 한다. 또한, 사용자와 입력과 관계없이 연결 및 동기 (synchronization)과 관련된 내부 메시지가 별도로 필요할 수 있다.
- **Sound (효과음)**: 각 플레이어의 턴이 끝나거나 라운드가 끝나면 1초 이내의 사운드 파일을 효과음으로 재생한다. 웹에서 적절한 파일을 찾아서 연결하면 된다.
- **Configuration (환경설정)**: 사용자 이름 변경, 제한시간(t, 서버 컴퓨터의 플레이어에 따름), 효과음 크기 등과 관련된 게임 상세 옵션 변경을 위한 메뉴이다.

1.3 제출 내용 및 평가 방식

본 맨해튼 보드 게임 과제 결과물로 필수적으로 제출해야 내용들은 다음과 같다.

- **코드 전체**
- **결과보고서**: 코드 설계 (클래스 계층도 포함), 구현 방법 및 실행 결과를 요약한 보고서

추가로, 본 맨해튼 보드 게임 과제 평가항목 및 배점은 다음과 같다.

- **게임의 기본기능 및 세부기능의 범위 및 완결성 (70점)**
- **구성 코드의 정확성, 간결성, Readability, 및 객체지향적 접근성 (20점)**
- **결과 보고서의 구체성 및 완결성 (10점)**
- ***보너스 항목: 1인용 인공지능 심화 (추가 30점) - 인공지능의 정교함 및 완성도를 평가**