Final Code Submission Explanation document

Chat gpt api를 이용한 타일 점령 게임

모바일 프로그래밍 (109038-31001) 수(6 ~ 9)

20101262 윤성현

목차

타일 점령 게임

게임 메뉴

게임 시작

설정

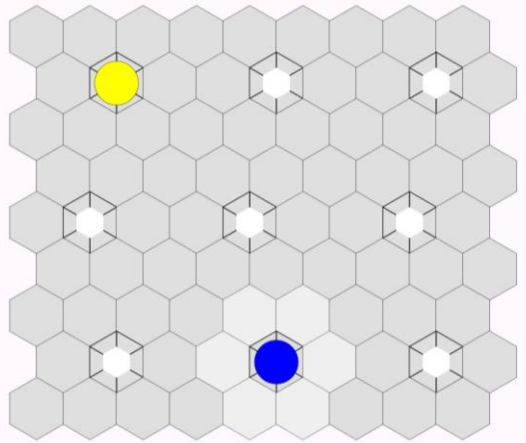
설정

name

Blue

저장

이름: name



name 의 턴 현재 턴 : 1 턴 넘기기

점령한 타일 수 : 0 상대가 점령한 타일 수 : 0

이름 : 유닛 소유자 : name

체력 : 3 공격력 : 1

❖ 프로젝트 개요

- 9x9 육각형 타일 맵에서
- 타일을 점령하거나 상대 유닛을 공격하여
- 맵의 절반 이상의 타일을 점령하거나
- 상대 유닛을 모두 없애면 승리하는 게임이다.

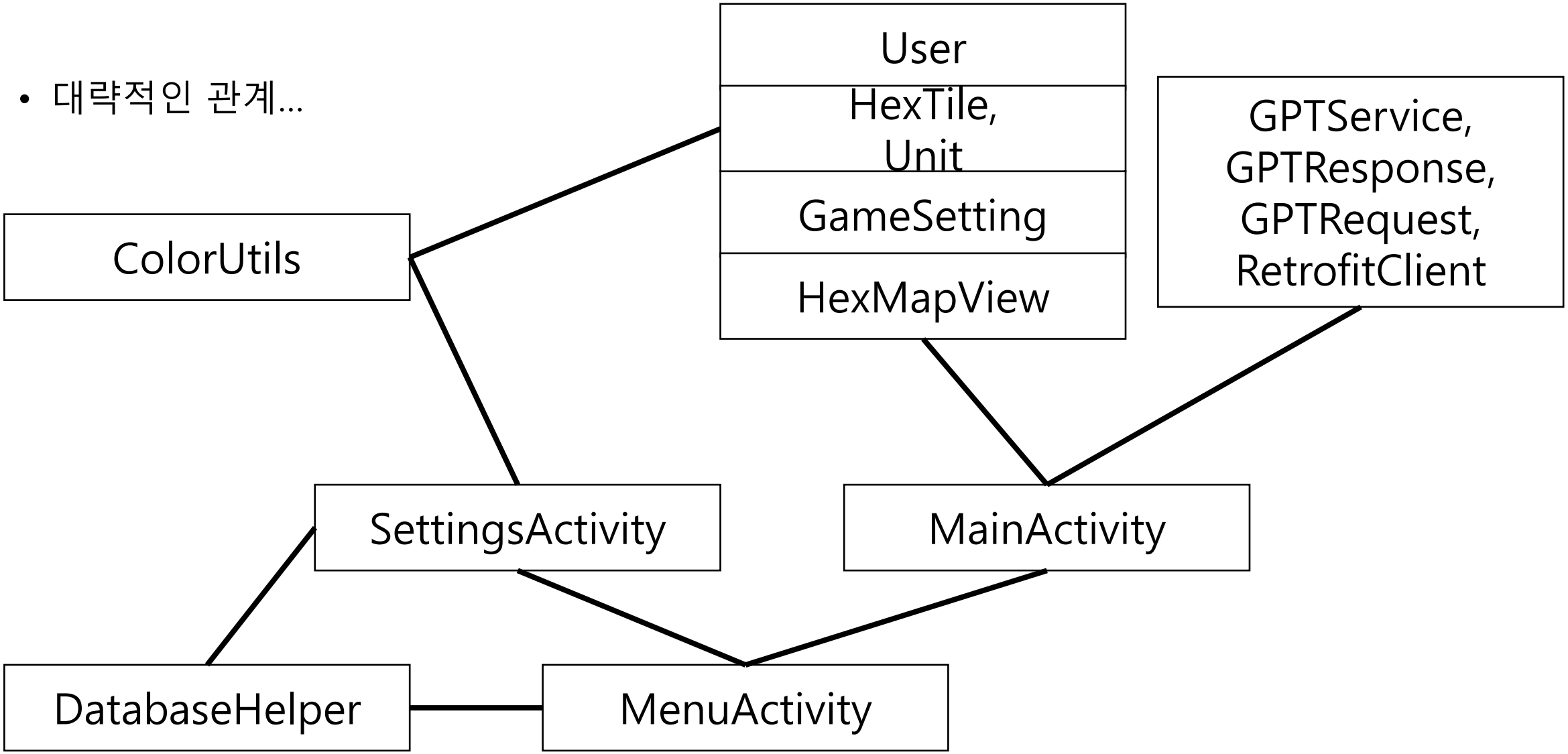
❖ 프로젝트 개요

- 클래스 목록

- 메뉴 화면 처리 : MenuActivity
- 설정 화면 처리 : SettingsActivity
- 게임 화면 처리 : MainActivity, GameSetting, HexMapView, HexTile, Unit, User
- 색상 처리 : ColorUtils
- GPT API 처리 : GPTService, GPTResponse, GPTRequest, RetrofitClient
- 데이터 베이스 : DatabaseHelper

❖ 프로젝트 개요

- 대략적인 관계...

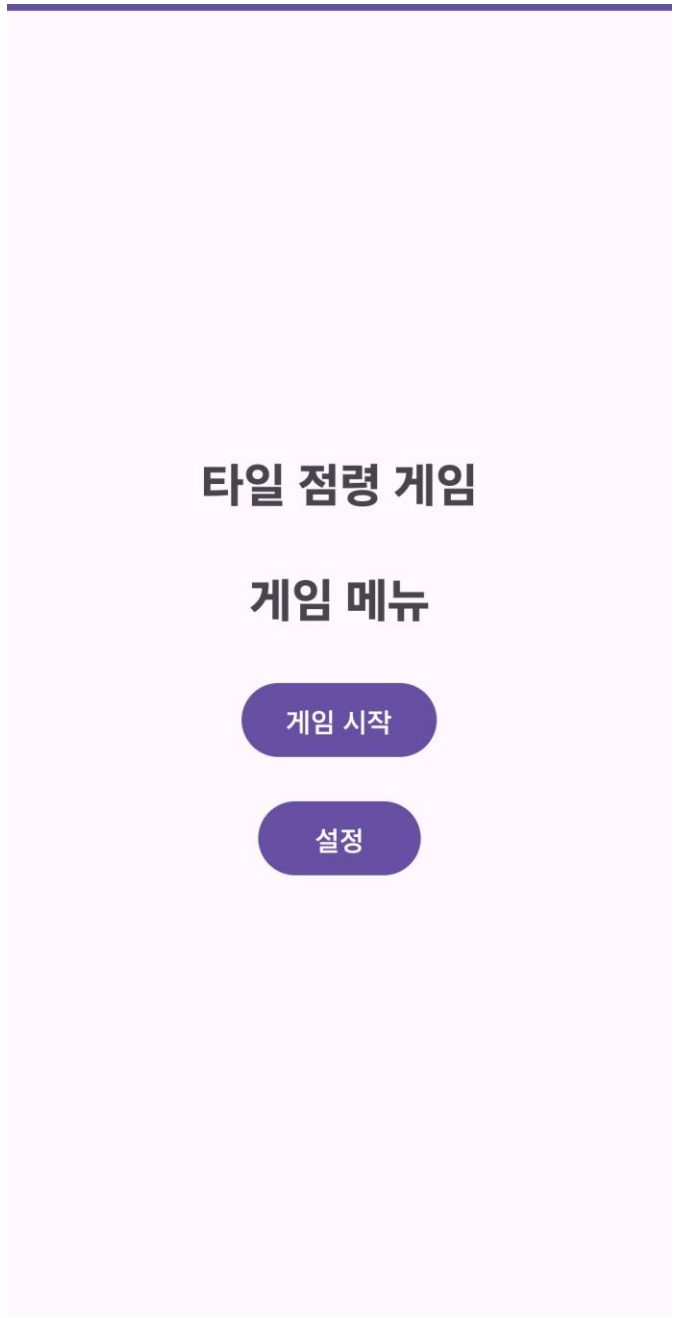


❖ 메뉴 화면

- 게임을 처음 시작하면
  표시되는 화면이다

- 게임 시작 버튼을 누르면
  게임 화면으로 이동한다

- 설정 버튼을 누르면
  설정 화면으로 이동한다

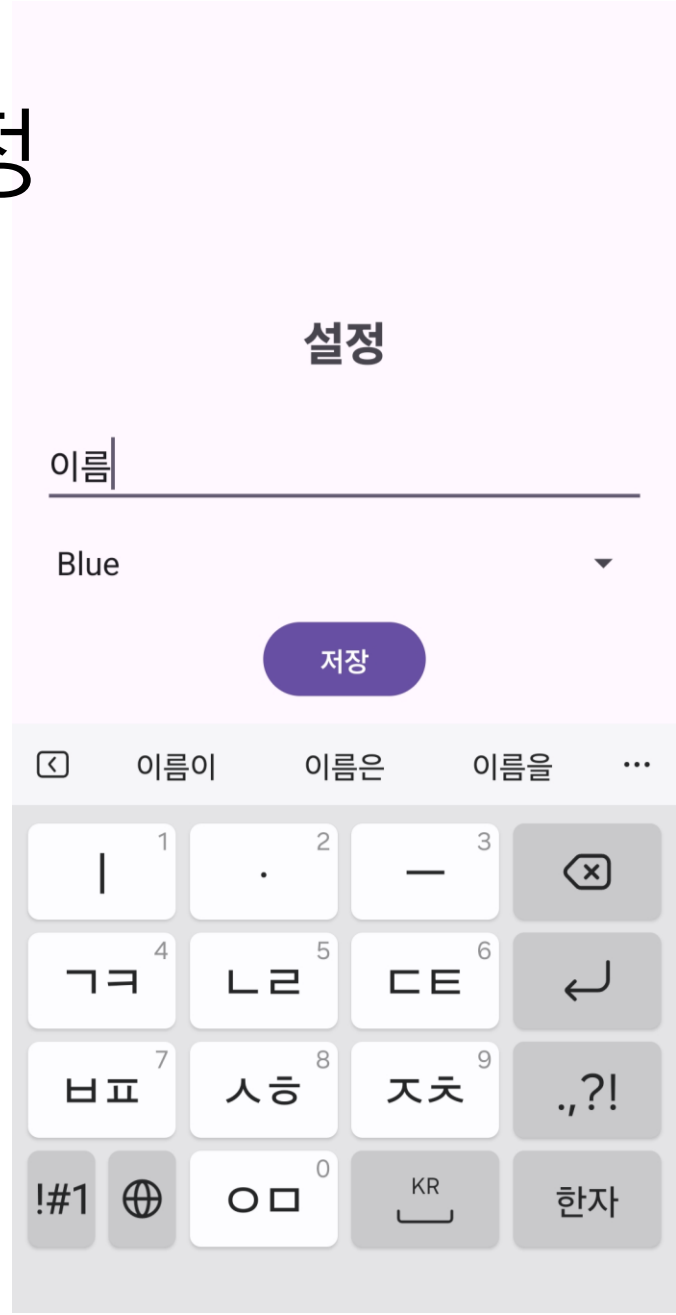


❖ 설정 화면 - 이름 설정, 색 설정

- 설정 화면은 이름과 색을 설정 할 수 있다

❖ 설정 화면 - 이름 설정, 색 설정

- 이름을 설정하려면
  이름 칸을 터치하여
  이름을 입력한다



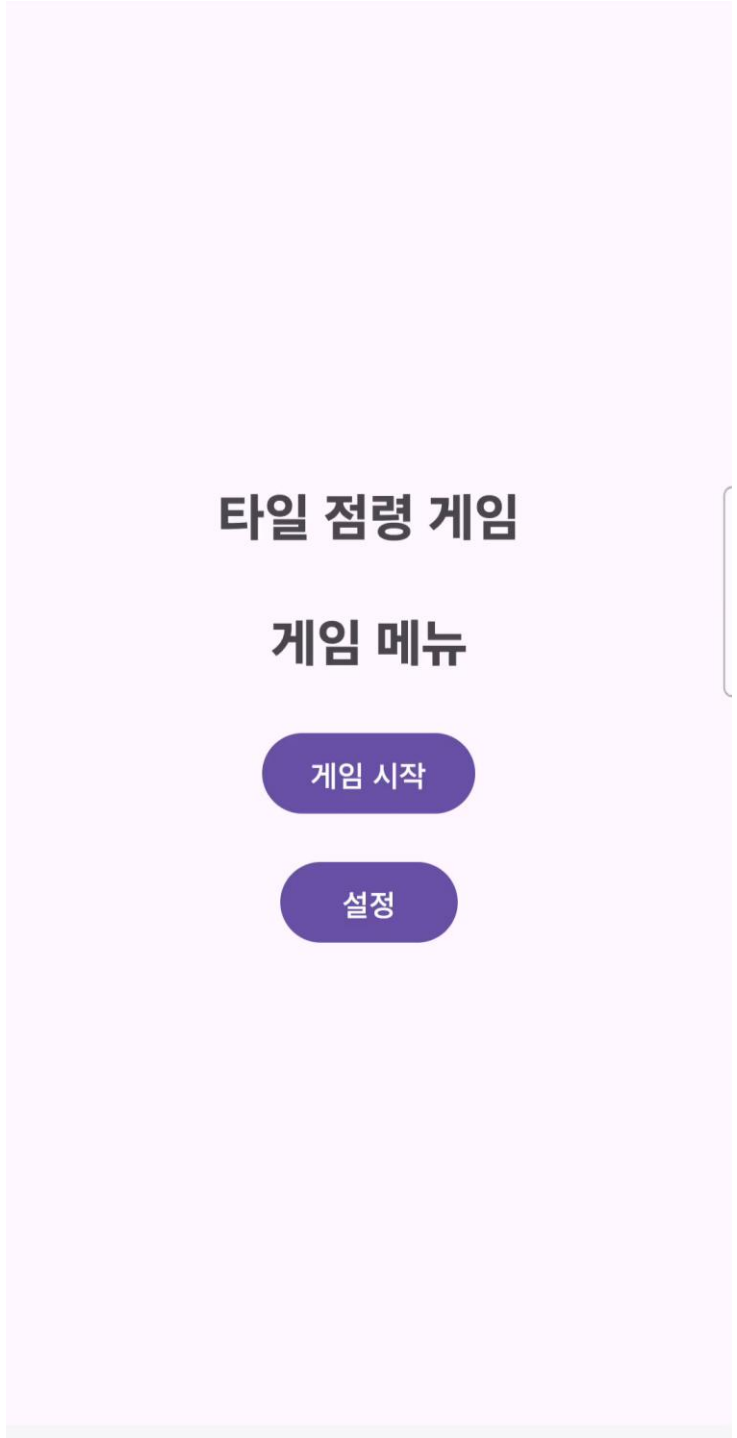
❖ 설정 화면 - 이름 설정, 색 설정

- 색을 설정하려면
  색 칸을 터치하여
  색을 고른다

❖ 설정 화면 - 이름 설정, 색 설정

- 저장 버튼을 누르면 설정한 내용이 저장된다
- 저장 버튼 아래의 색과 이름으로 변경을 확인할 수 있다

❖ 설정 화면



❖ 설정 화면

- 설정은 SettingsActivity가 DatabaseHelper를 이용하여 저장, 조회된다.

```
@Override
public void onCreate(SQLiteDatabase db) {
    // 테이블 생성 SQL
    String CREATE_TABLE = "CREATE TABLE " + TABLE_NAME + " (" +
            COLUMN_NAME + " TEXT, " +
            COLUMN_COLOR + " TEXT);";
    db.execSQL(CREATE_TABLE);

    // 기본값 삽입 SQL
    String INSERT_DEFAULTS = "INSERT INTO " + TABLE_NAME + " (" + COLUMN_NAME + ", " + COLUMN_COLOR + ") " +
            "VALUES ('Default Name', 'Red');";
    db.execSQL(INSERT_DEFAULTS);
}
```

(데이터 베이스 테이블 구조)

❖ 설정 화면

- 게임 시작 시 MenuActivity가 DatabaseHelper로 데이터를 불러온 뒤, 해당 데이터를 넘겨줘서 게임 화면에 적용하도록 한다

```
Button startGameButton = findViewById(R.id.startGameButton);
startGameButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        DatabaseHelper dbHelper = new DatabaseHelper(instance);
        String[] settings = dbHelper.getSettings();

        // 게임 화면으로 이동
        GameSetting.initialize(settings);
        GameSetting.reset();
        Intent intent = new Intent( packageContext: MenuActivity.this, MainActivity.class);
        startActivity(intent);
    }
});
```

❖ 설정 화면

- 이름과 색상 데이터를 받으면 HexMapView의 public void createHexMap에서 이를 통해 상대 AI의 이름, 색상이 정해진다

```
if (GameSetting.isInitial()) {
    String user1Name = GameSetting.getNAME();
    String user1Color = GameSetting.getCOLOR();

    String user2Name = "anti-"+user1Name;
    String user2Color = "anti-"+user1Color;

    User user1 = new User(user1Name, user1Color);
    User user2 = new User(user2Name, user2Color);

    //System.out.println("처음처음처음처음처음처음처음처음처음");
```

❖ 설정 화면

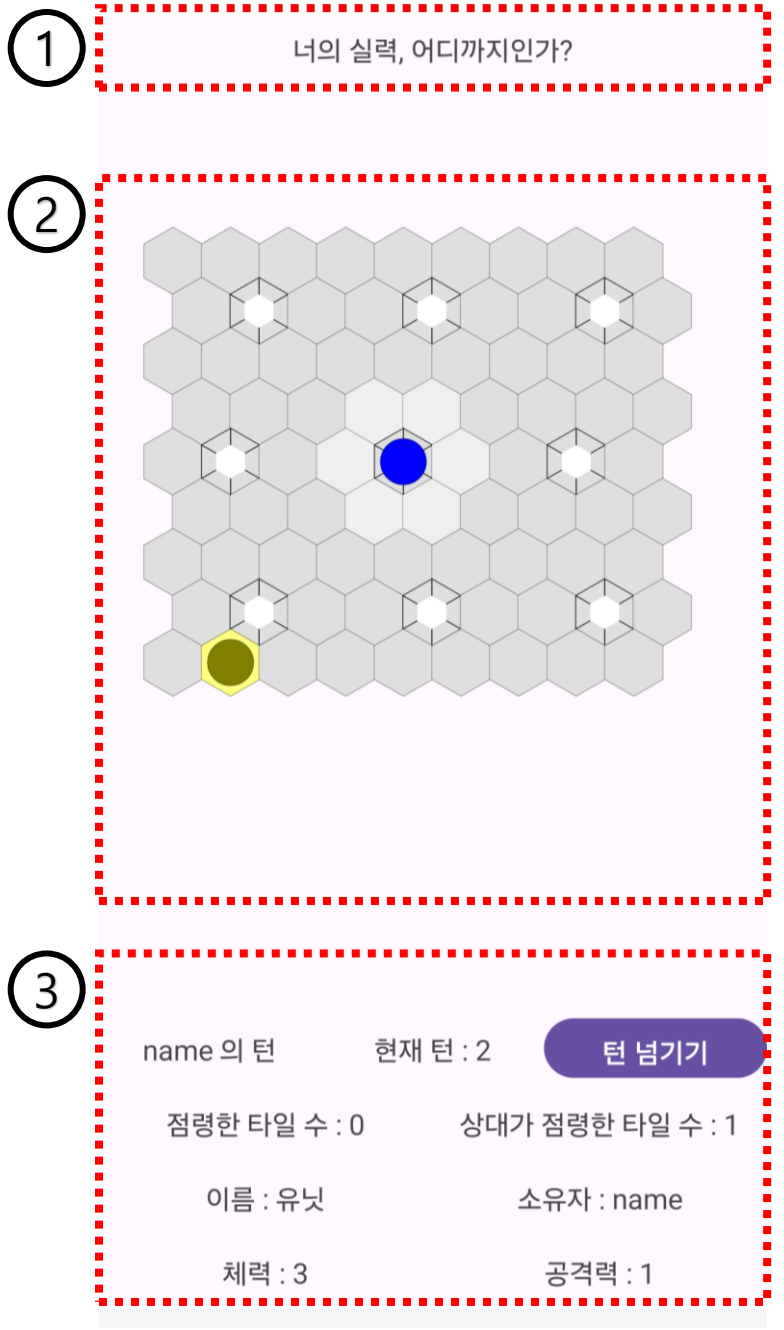
- 색상 표

```
public class ColorUtils {  6 usages
    public static int getColorFromName(String colorName) {  3 usages
            case "Yellow":
                return Color.YELLOW;
            case "Cyan":
                return Color.CYAN;
            case "Magenta":
                return Color.MAGENTA;
            case "Orange":
                return Color.rgb( red: 255,  green: 165,  blue: 0); // Orange 색상 값
            case "Blue-Green":
                return Color.rgb( red: 0,  green: 184,  blue: 184); // Blue-Green 색상 값
            case "anti-Red":
                return invertColor(Color.RED); // Red의 보색
            case "anti-Green":
```

❖ 게임 화면

- 게임 화면은 크게 3부분으로 나뉜다

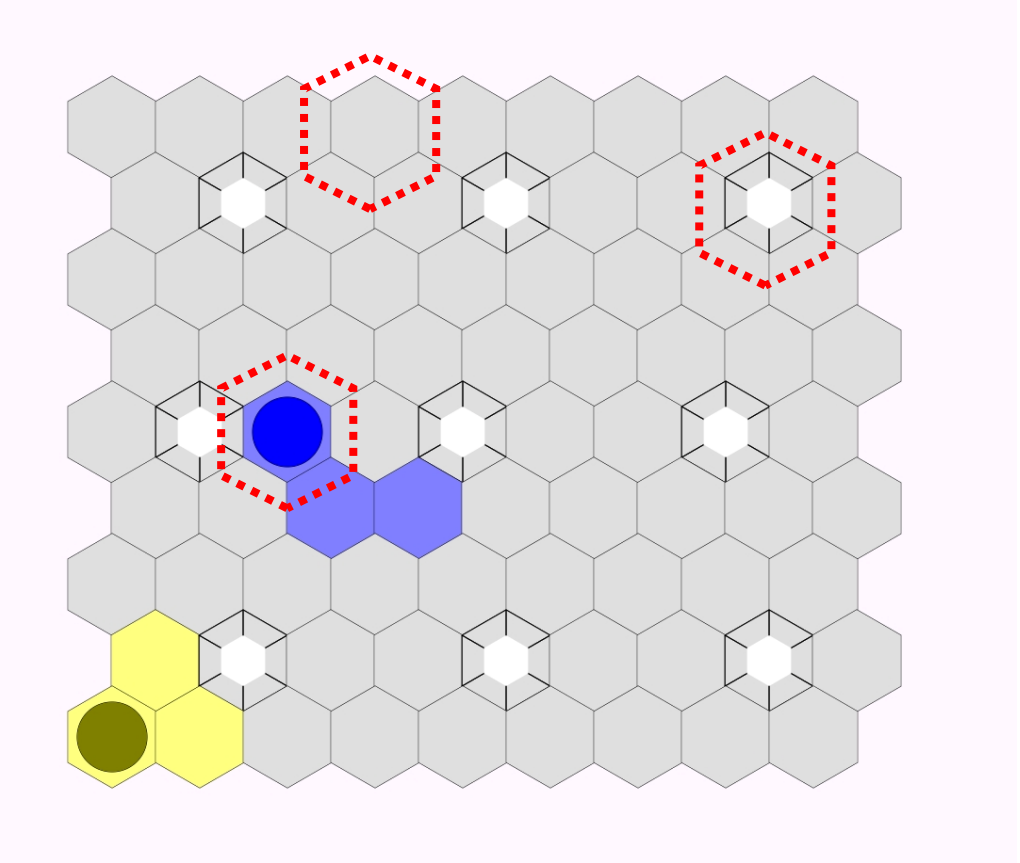
1. 상단
   - 상대 ai의 대사가 표시된다
2. 중앙
   - 게임의 맵이 표시된다
3. 하단
   - 컨트롤 패널이 표시된다



❖ 게임 화면 - 맵

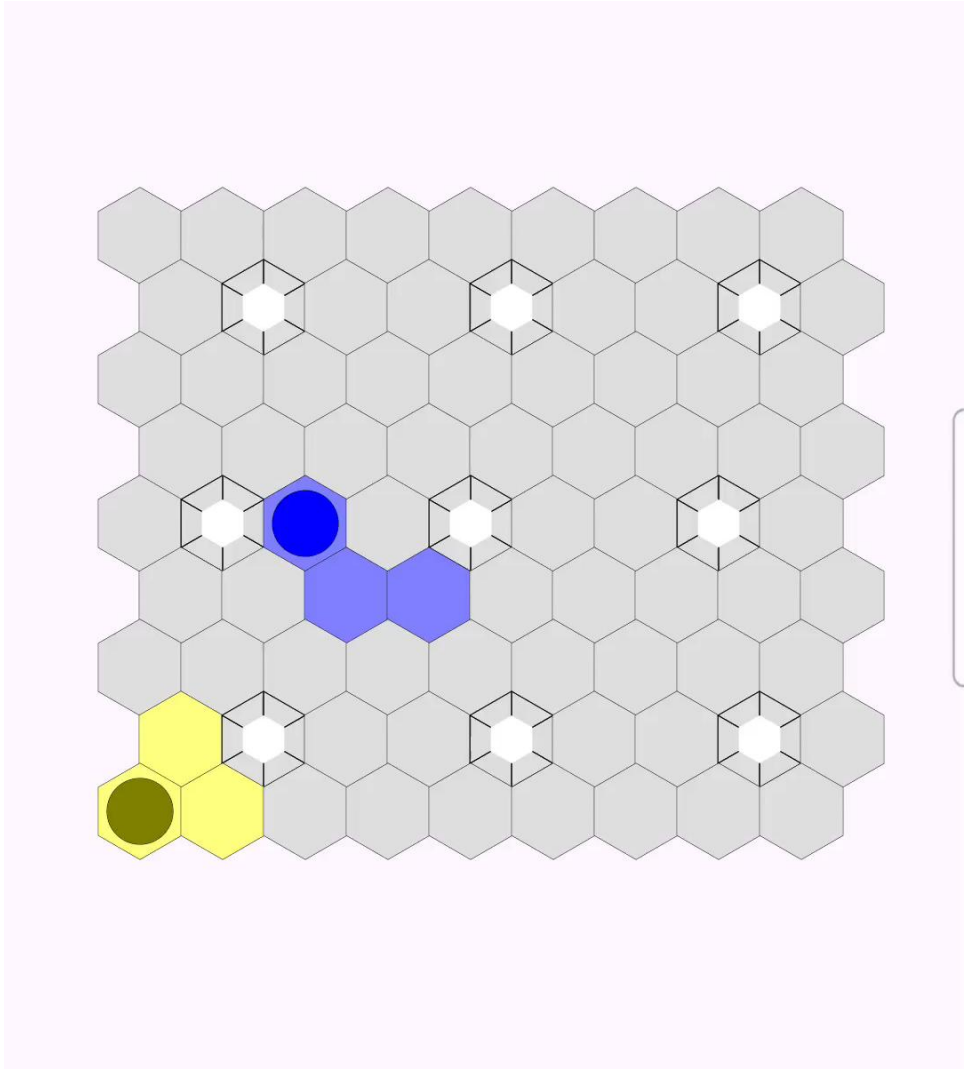
- 맵에는
  육각형의 일반 타일,
  내부에 작은 육각형이 있고
  테두리가 검은 생산 타일,
  그리고 유닛이 표시된다.

- 점령되지 않은 타일은 회색,
  점령된 타일은 해당 유닛의 색이다



❖ 게임 화면 - 맵

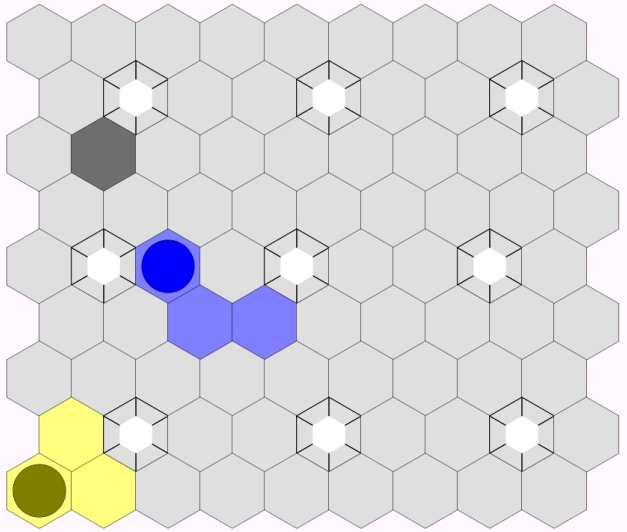
- 맵의 타일을 터치하면
  타일의 색이 어둡게 변한다
- 맵의 이동, 확대가 가능하다

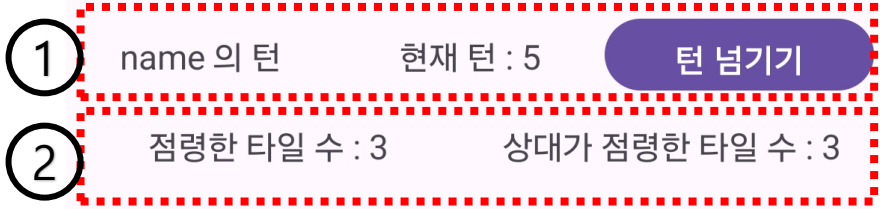


❖ 게임 화면 – 컨트롤 패널

- 컨트롤 패널은 기본적으로

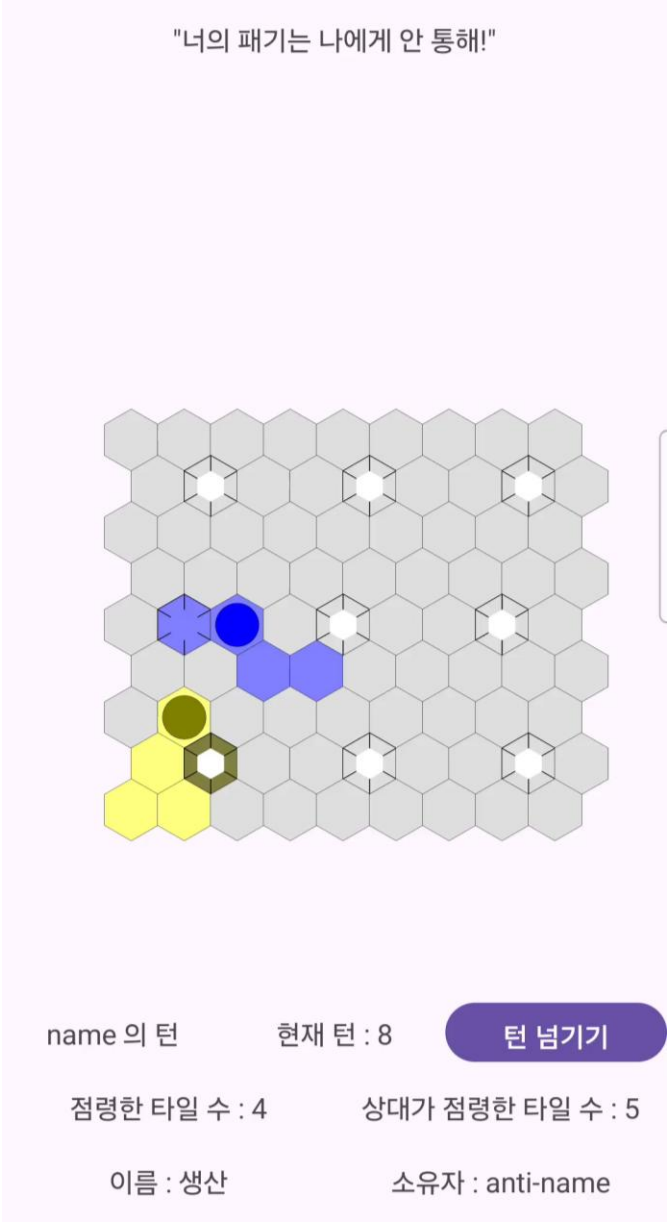
1. 어느 플레이어의 턴인지와
   현재 턴, 턴 넘기기 버튼이 표시된다
2. 점령한 타일 수,
   상대가 점령한 타일 수가 표시된다





❖ 게임 화면 - 컨트롤 패널

- 턴 넘기기 버튼 클릭 시 컨트롤 패널
- 어느 플레이어의 턴인지와
  현재 턴이 갱신된다



❖ 게임 화면 – 컨트롤 패널

- 생산 타일 클릭 시 컨트롤 패널

해당 타일이 점령되지 않은 경우

상대가 해당 타일을 점령한 경우

자신이 해당 타일을 점령한 경우

❖ 게임 화면 – 컨트롤 패널

- 유닛 클릭 시 컨트롤 패널

상대의 유닛을 클릭한 경우

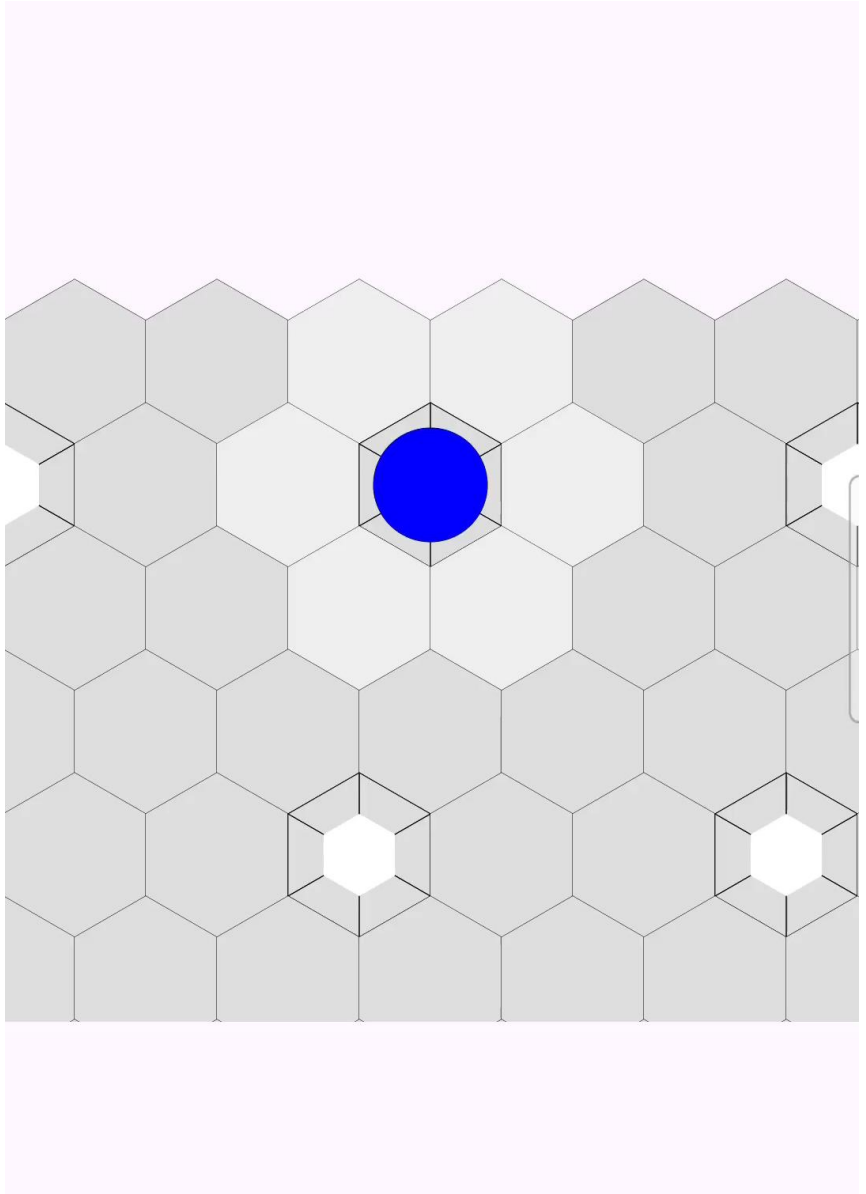
자신의 유닛을 클릭한 경우

❖ 게임 화면 - 유닛

- 유닛은 체력과 공격력이 있다
- 상대 유닛 공격 시 공격력 만큼
  상대 유닛의 체력을 감소시킨다
- 유닛의 체력이 0 이하가 되면
  유닛은 삭제된다
- 유닛은 한 턴에 한번, 주위의 1타일
  안에서 이동하거나 공격 할 수 있다

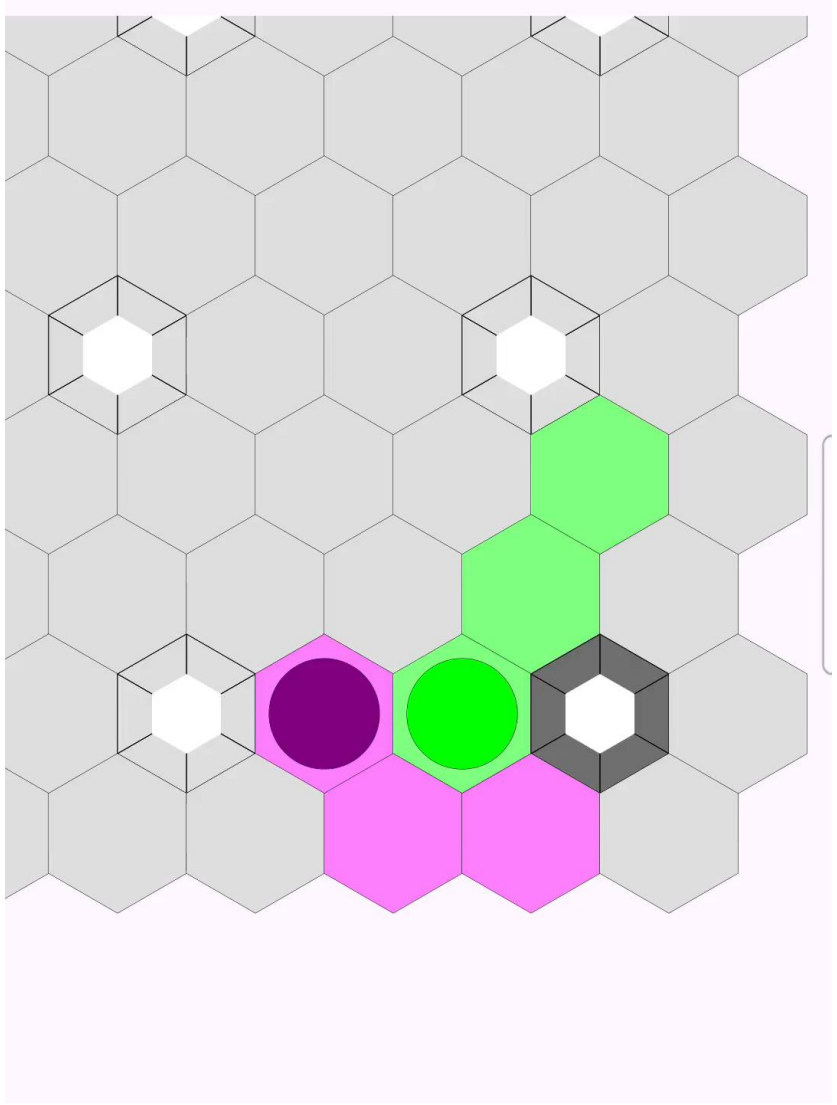
❖ 게임 화면 - 유닛

- 유닛 이동 시 해당 타일을 점령한다
- 만약 이미 점령된 타일이라면
  체력을 1 감소 한 대가로
  해당 타일을 점령 할 수 있다



❖ 게임 화면 - 유닛

- 유닛 이동 시 1타일 내에
  상대 유닛이 존재한다면
  해당 타일이 노란색으로 표시되며
  상태 유닛을 터치하면 공격한다

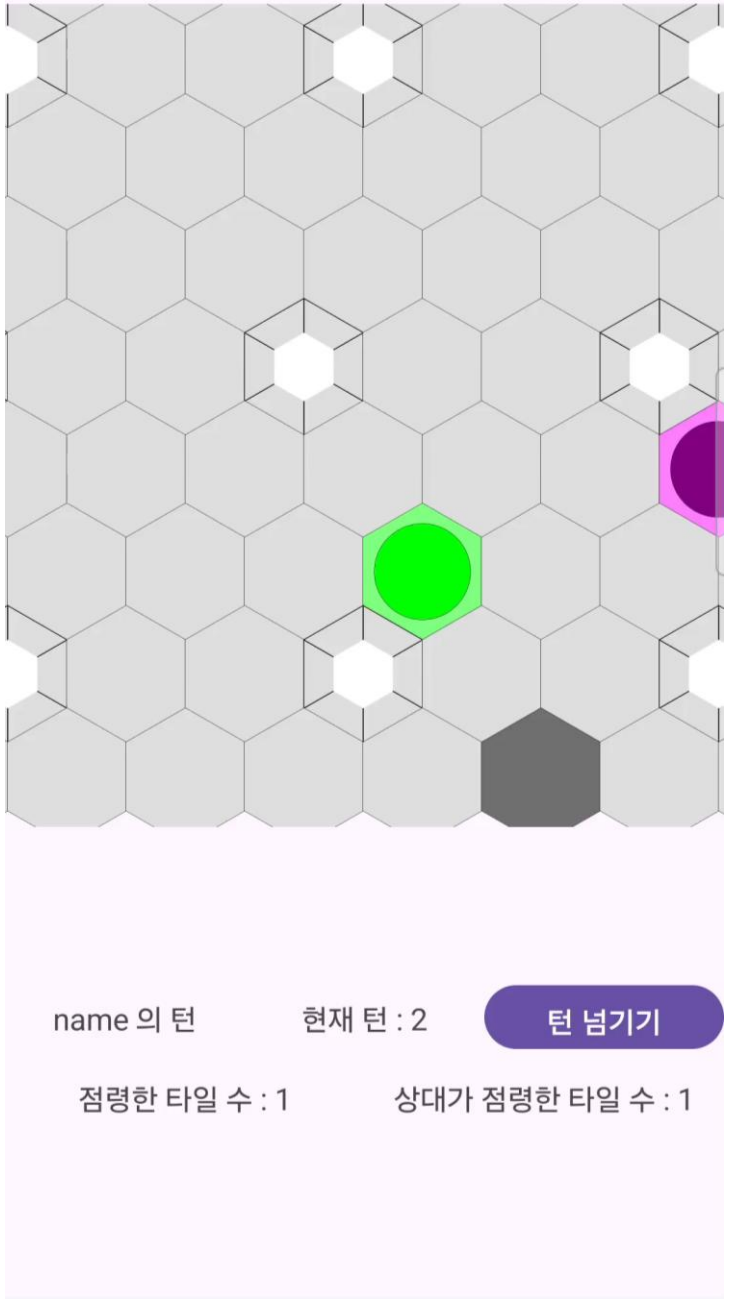


❖ 게임 화면 - 생산

- 생산 타일을 점령하고
  해당 타일을 터치하면
  컨트롤 패널에 생산하기
  버튼이 표시된다

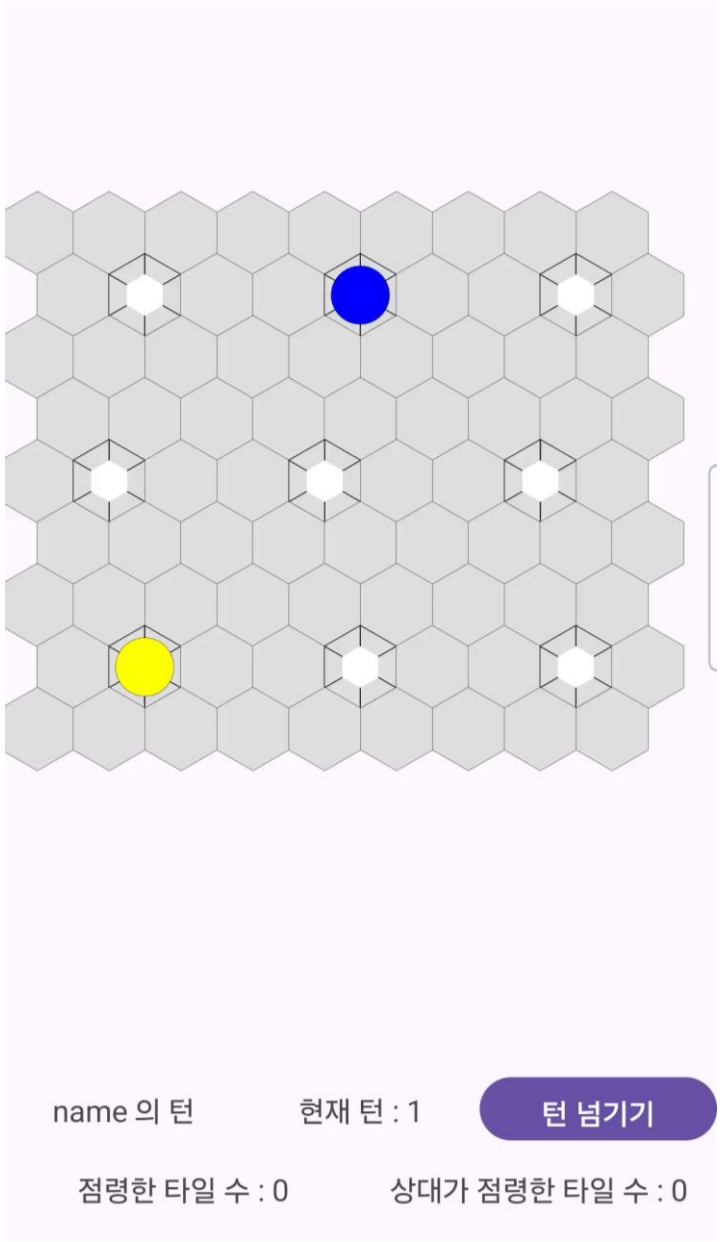
- 점령한 타일 수가 비용 이상이면
  유닛을 생산할 수 있다

- 점령한 타일 수가 부족하면
  toast로 경고 메시지가 표시된다



❖ 게임 화면 - 상대 ai

- 턴 넘기기 버튼을 누르면
  상대 ai가 게임을 진행한다
- 선택 가능한 유닛을
  가중치의 랜덤 값으로
  이동시키거나 공격시킨다
- 생산 가능한 타일이 있다면
  유닛을 생산한다



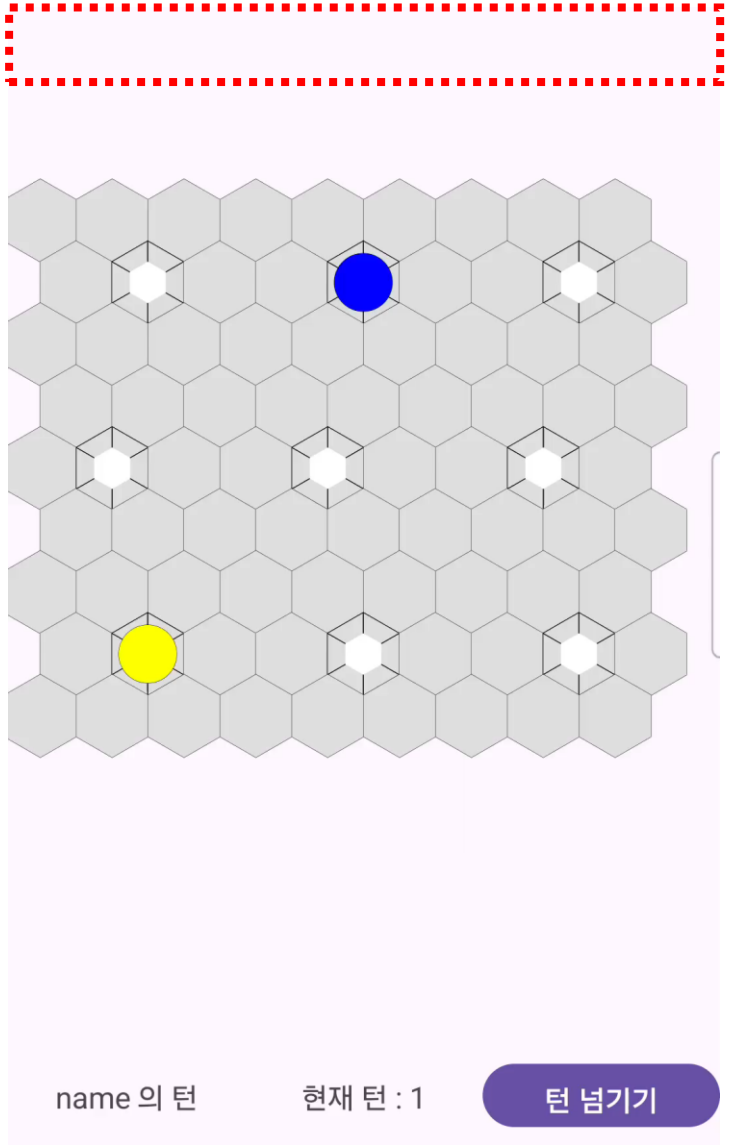
❖ 게임 화면 - 상대 ai

- HexMapView의 public void computerAi를 이용해 구현 하였다
- 타일의 소유권에 따라 각 타일로 이동할 가중치가 정해진다

```
HexTile targetTile = null;
// 확률적으로 선택: 소유자 없는 타일을 더 선호
double randomValue = Math.random();
if (!neutralTiles.isEmpty() && randomValue < 0.6) { // 60% 확률로 소유자 없는 타일 선택
    targetTile = neutralTiles.get((int) (Math.random() * neutralTiles.size()));
} else if (!ownTiles.isEmpty() && randomValue < 0.9) { // 30% 확률로 자기 소유 타일 선택
    targetTile = ownTiles.get((int) (Math.random() * ownTiles.size()));
} else if (!enemyTiles.isEmpty()) { // 10% 확률로 상대 타일 선택
    targetTile = enemyTiles.get((int) (Math.random() * enemyTiles.size()));
} else {
    int randomIndex = (int) (Math.random() * movableAndAttackableTiles.size());
    targetTile = movableAndAttackableTiles.get(randomIndex);
    Default = 1;
}
```

❖ 게임 화면 - Chat gpt로 상대 ai의 대사 생성

- 상대 ai의 턴이 거의 종료되면
  Chat gpt로 생성한, 플레이어를
  도발하는 대사를 상단에 표시한다



❖ 게임 화면 - Chat gpt로 상대 ai의 대사 생성

- MainActivity의 private void callGPTAPI가 쿼리를 받으면 GPTService, GPTResponse, GPTRequest, RetrofitClient를 이용하여 대사를 생성한다.

```
// GPT API 호출
callGPTAPI( prompt: "턴제 게임에서 상대방을 도발하는 대사를 20글자 이내로 창의적으로 작성해줘.", () -> {
    isAPICalled.set(true);
    checkAndShowTurnButton(isAICompleted, isAPICalled);
});
```

❖ 게임 화면 - Chat gpt로 상대 ai의 대사 생성

```
// GPT API 호출 메서드
private void callGPTAPI(String prompt, Runnable onComplete) {  1 usage
    String apiKey = BuildConfig.GPT_API_KEY;
            ;//"asdf"; // 환경 변수나 안전한 저장소에서 가져오는 것을 권장
    Retrofit retrofit = RetrofitClient.getClient(apiKey);
    GPTService gptService = retrofit.create(GPTService.class);

    GPTRequest request = new GPTRequest(prompt);

    Log.d( tag: "GPTRequest",  msg: "Request Data: " + new Gson().toJson(request));

    gptService.getResponse(request).enqueue(new retrofit2.Callback<GPTResponse>() {
        @Override
        public void onResponse(Call<GPTResponse> call, retrofit2.Response<GPTResponse> response) {
            System.out.println(response.message());
            if (response.isSuccessful() && response.body() != null) {
                // 응답 본문이 null이 아닌지 확인
```

❖ 게임 화면 - Chat gpt로 상대 ai의 대사 생성

```java
package com.example.term_project;

import ...

public interface GPTService {  2 usages
    @Headers({  1 usage
            "Content-Type: application/json"
    })
    @POST("v1/chat/completions")
    Call<GPTResponse> getResponse(@Body GPTRequest request);
}
```

❖ 게임 화면 - Chat gpt로 상대 ai의 대사 생성

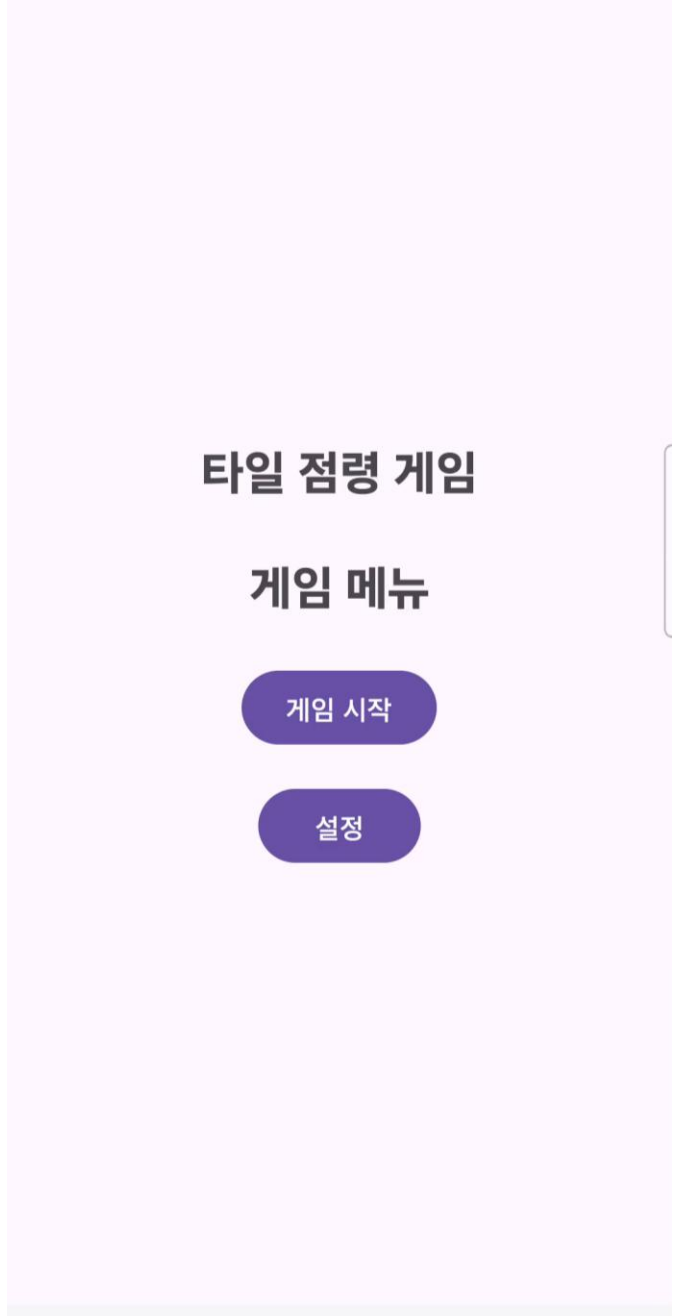
```java
// GPTRequest.java
public class GPTRequest {  3 usages
    private String model;  1 usage
    private List<Message> messages;  3 usages

    private String prompt;  no usages

    public GPTRequest(String prompt) {  1 usage
        this.model = "gpt-4o-mini"; // GPT 모델 선택
        this.messages = List.of(new Message( role: "user", prompt));
    }
```

❖ 게임 화면 - Chat gpt로 상대 ai의 대사 생성

```java
package com.example.term_project;

import ...

public class RetrofitClient {  1 usage
    private static Retrofit retrofit;  3 usages

    public static Retrofit getClient(String apiKey) {  1 usage
        OkHttpClient client = new OkHttpClient.Builder()
                .addInterceptor(new Interceptor() {
                    @Override
                    public Response intercept(Chain chain) throws IOException {
                        Request original = chain.request();
                        Request.Builder requestBuilder = original.newBuilder()
                                .header( name: "Authorization",  value: "Bearer " + apiKey);
                        return chain.proceed(requestBuilder.build());
                    }
                }).build();

        if (retrofit == null) {
            retrofit = new Retrofit.Builder()
                    .baseUrl("https://api.openai.com/") // GPT API URL
                    .addConverterFactory(GsonConverterFactory.create())
                    .client(client)
                    .build();
        }
        return retrofit;
    }
}
```

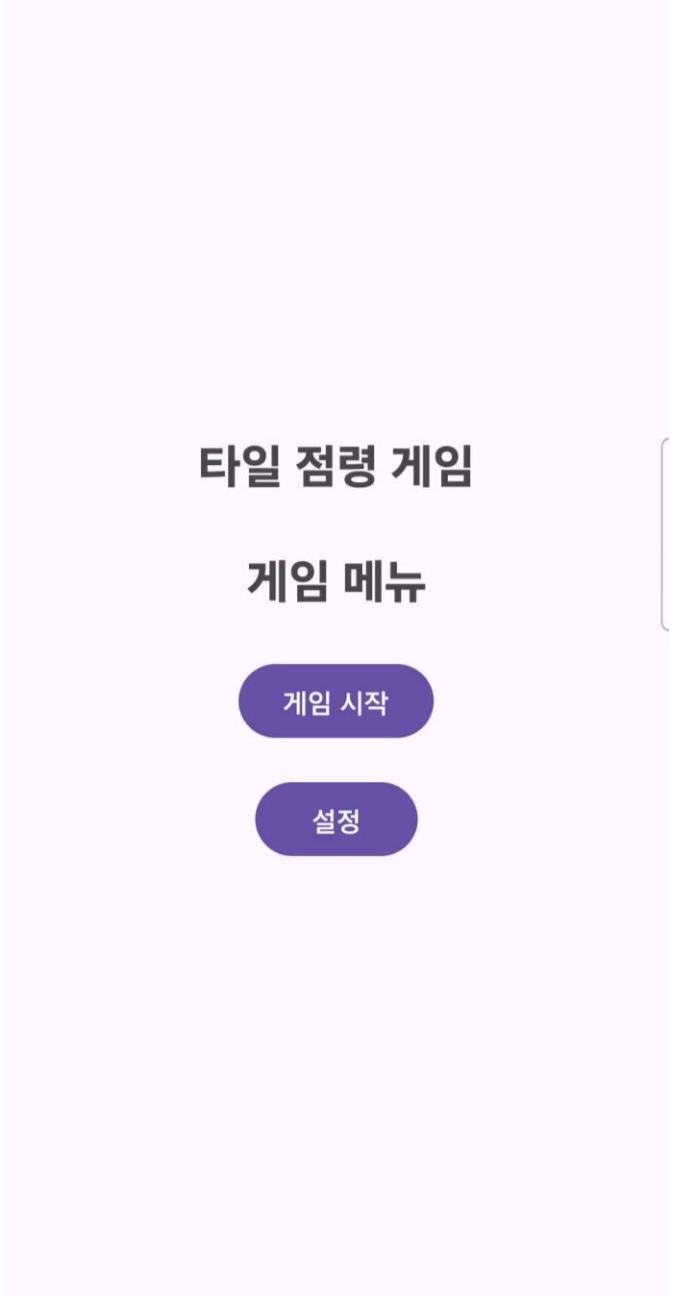
❖ 시연

- 맵의 절반 이상의 타일을 점령하여 승리하는 경우



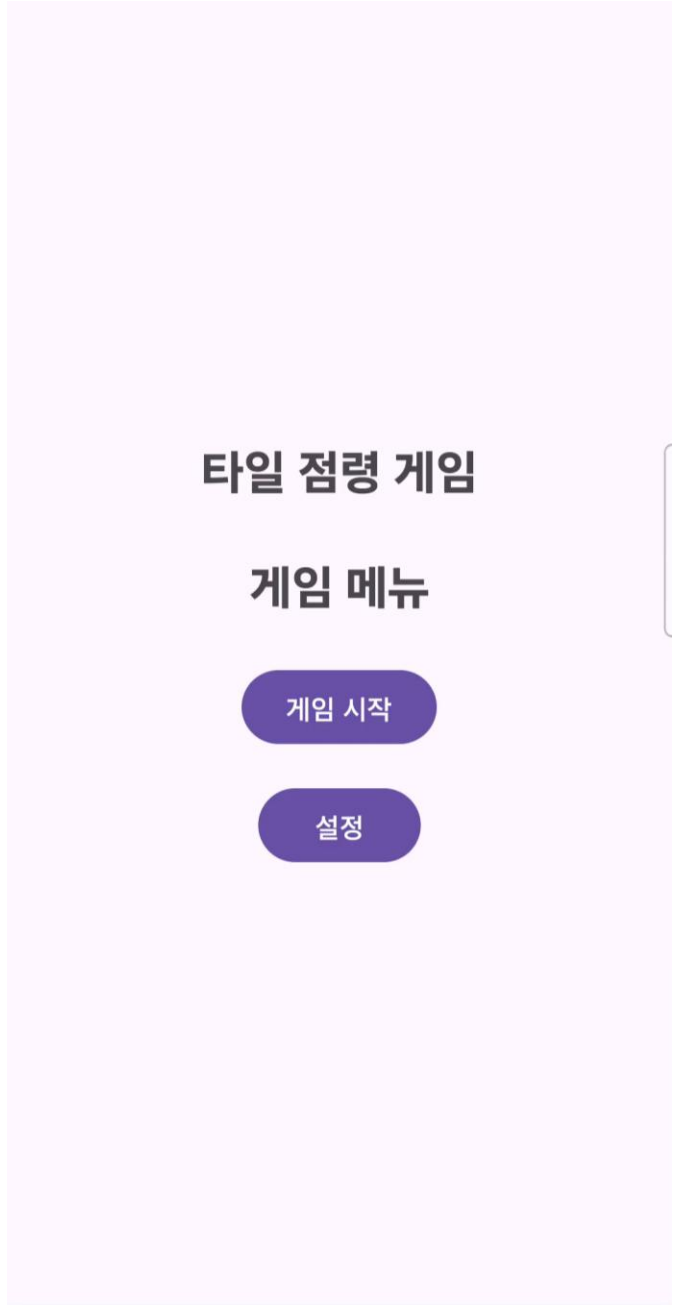
❖ 시연

- 상대 유닛을 모두 제거하여 승리하는 경우



❖ 시연

- 패배하는 경우



❖ 개선 사항 및 향후 발전 계획

- 기존의 상대 AI에 강화 학습을 도입하여 상대 AI가 지능적으로 게임을 할 수 있도록 한다
- 소켓 통신을 통한 멀티플레이를 구현하여 AI 대신 사람들이 플레이 할 수 있도록 한다
- Chat gpt의 대사 생성이나 유닛의 이동, 피격, 생산 등 여러 이벤트에 애니메이션을 추가한다.
- UI, 맵, 유닛의 디자인을 개선한다.
- 현재 코드가 복잡하므로 확장성과 관리를 위해 리팩토링을 진행한다.