REPORT

INHA AR Navigation

과 목	공간정보종합설계프로젝트
담당교수	김태정 교수님
이 름	박민규, 장성준, 정창현
학 번	12171218, 12171206, 12171210
제 출 일	2022. 06. 11

<목 차>

1. 설계 개요

가. 설계 목표 ; 인하대학교 캠퍼스 AR 네비게이션 제작

나. 설계 배경 :

[그림] 교내 커뮤니티 게시물

기존 교내 커뮤니티 (Everytime 앱) 등지에서 학기 초마다 교내 건물의 위치를 알지 못하여 수업에 늦거나 참석하지 못하는 경우가 발생하고 있다. 이에 처음 학교를 방문하는 사람 혹은 신입생 등을 위하여 교내 건물의 위치를 기존 2D 지도를 포함한, 기존보다 이해하기 쉬운 형태인 증강현실 AR(Augmented Reality) 네비게이션 개발을 구상하였다.

다. 설계 범위 : 인하대학교 캠퍼스 (본관 ~ 9호관) 외부

라. 개발 환경 :

이름	버전	내용
Android Studio	2021.2.1	전반적인 안드로이드 어플리케이션 개발
SQLite	3.38.5	데이터베이스 제작 및 연동
Mapbox Maps API	9.3.0	길찾기 기능을 위한 지도 API
Mapbox Navigation API	0.42.6	경로 요청 및 길찾기를 위한 API
Mapbox Unity SDK	2.1.1	AR 네비게이션 제작을 위한 유니티용 지도 SDK
Unity	19.3.43	3D 오브젝트 및 AR 구현

마. 역할 분담

박민규 : 데이터 수집, 네비게이션 기능 구현, API 연동 및 연결

장성준 : AR 네비게이션 기능 구현, UI 개발

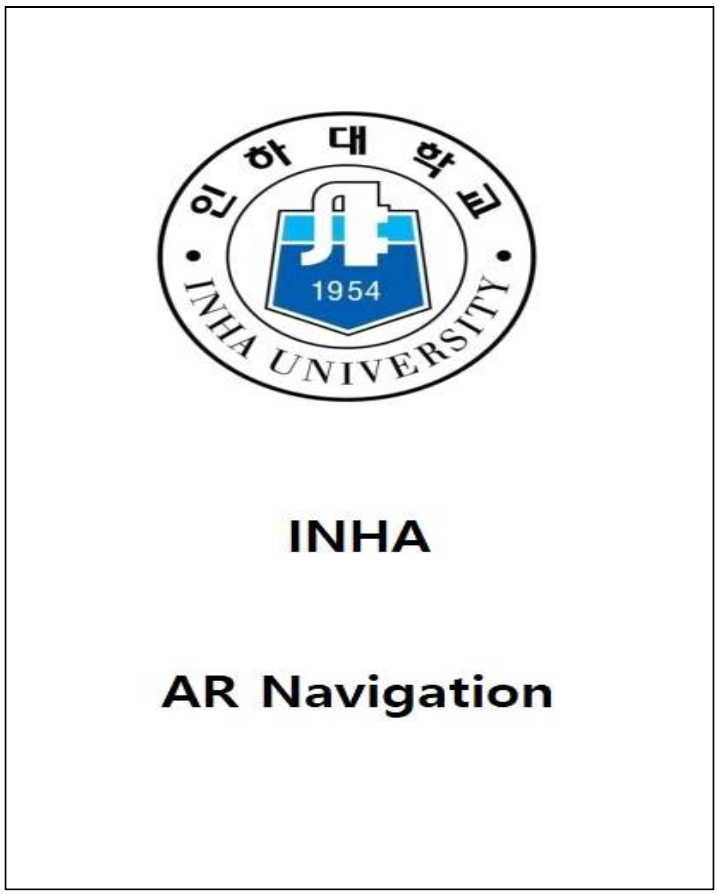
정창현 : 데이터베이스 구현, 검색기능 구현, UI 수정 및 보완

2. 개발 과정

가. UI 제작

ㄱ. Splash Activity

초기 화면에 어떤 어플리케이션인지 정체성을 나타내고 로딩화면을 설정한다. 인하대 로고와 APP의 명칭인'INHA AR Navigation'을 시각화했다.



- manifest설정에서 label을 APP 명칭으로 바꾸고 ICON을 인하대 로고를 사용해 제작한다. 초기화면을 splashactivity로 설정하고 액티비티의 테마를 설정해준다.

```
android:allowBackup="true"
android:icon="@mipmap/ic_inha"
android:label="INHA AR Navigation"
android:roundIcon="@mipmap/ic_inha_round"
```

```
</activity>
<activity
    android:name=".splashactivity"
   android:theme="@style/Theme.SplashTheme"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

- values 디렉토리에 string에서 app name을 inha ar navigation으로 바꾼다.

```
<string name="app_name">INHA AR Navigation</string>
```

- 액티비티의 테마는 values 디렉토리에 themes에서 설정한다. 새로운 테마인 splashtheme을 만들어 액션바가 없는 로딩화면(그림2)을 설정한다.

```
<style name="Theme.SplashTheme" parent="Theme.AppCompat.NoActionBar">
    <item name="android:windowBackground">@drawable/inhasplash</item>
</style>
```

- values 디렉토리에 사용할 color를 추가한다.

```
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="green">#008000</color>
    <color name="lavender">#E6E6FA</color>
    <color name="steelblue">#4682B4</color>
    <color name="powderblue">#B0E0E6</color>
    <color name="lightbule">#ADD8E6</color>


    <color name="colorPrimary">#4264fb</color>
    <color name="colorPrimaryDark">#4264fb</color>
    <color name="colorAccent">#4264fb</color>

    <color name="mapboxWhite">#ffffff</color>
    <color name="mapboxBlue">#4264fb</color>
    <color name="mapboxGrayLight">#c6d2e1</color>

</resources>
```

- handler를 사용하여 딜레이를 설정하고 화면을 보여준뒤 intent로 메인 액티비티로 넘어가는 코드를 작성한다.

```
public class splashactivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle saveInstanceState){
        super.onCreate(saveInstanceState);
        setContentView(R.layout.splashlayout);

        Handler handler=new Handler();
        handler.postDelayed(new Runnable() {
            @Override
                public void run() {
                    Intent intent=new Intent(getApplicationContext(),MainActivity.class);

                    startActivity(intent);

                    finish();
            }
        },3000);
    }
    @Override
    protected void onPause() {
        super.onPause();
        finish();
    }
}
```

ㄴ. Main Activity layout

스크롤과 깔끔한 UI를 위해 recyclerview와 cardview를 사용한다. 화면 하위에는 지도버튼을 만들어 클릭시 2D MAP으로 이동할수 있게한다.

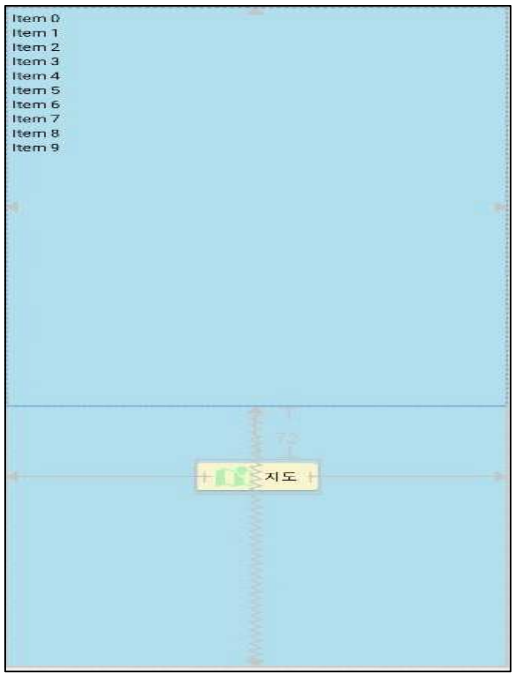
- gradle에 recyclerview와 cardview를 사용할 수 있게 다음과 같이 추가해준다.

```
implementation'androidx.recyclerview:recyclerview:1.2.1'
implementation'androidx.cardview:cardview:1.0.0'
```

- 기본적인 테마를 사용하며 actionbar가 존재하고 전체적인 색깔을 설정한다.

```
<style name="Theme.MyApplication111" parent="Theme.MaterialComponents.Light.DarkActionBar">
    <!-- Primary brand color. -->
        <item name="colorPrimary">@color/steelblue</item>
    <item name="colorPrimaryVariant">@color/powderblue</item>
    <item name="colorOnPrimary">@color/black</item>
    <!-- Secondary brand color. -->
    <item name="colorSecondary">@color/teal_200</item>
    <item name="colorSecondaryVariant">@color/teal_700</item>
    <item name="colorOnSecondary">@color/black</item>
    <!-- Status bar color. -->
    <item name="android:statusBarColor" tools:targetApi="l">?attr/colorPrimaryVariant</item>
    <!-- Customize your theme here. -->
    <item name="android:textColor">@color/black</item>
</style>
```

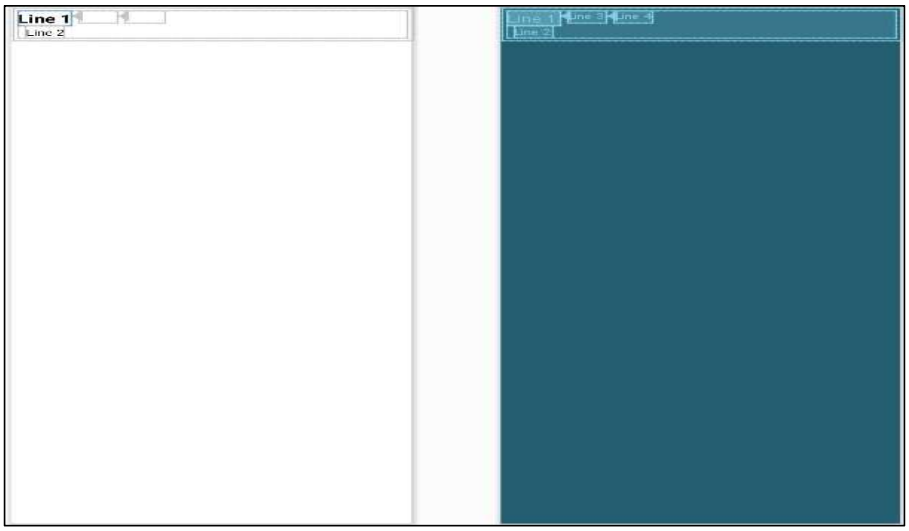
- main activity layout은 다음과 같은 화면으로 나타난다. 윗부분은 reclcyerview로 설정하고 아래 부분은 지도 버튼을 구성한다.



- cardview는 다른 layout으로 설정하고 relativelayout을 사용해 cardview 안쪽으로 textview들이 서로 나란히 배치한다. 이때 textview는 목적지, 건물, x좌표, y좌표를 나타내기 위해 4개를 설정하며 x, y좌표는 시각화 하지 않는다.

```
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:strokeColor="@color/black"
    android:layout_margin="1dp"
    app:strokeWidth="2dp"
    android:elevation="5dp"
    app:cardCornerRadius="4dp">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="4dp">

        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:gravity="center_horizontal"
            android:text="Line 1"
            android:textColor="@color/black"
            android:textSize="20sp"
            android:textStyle="bold" />
```

• cardview에 layout은 다음과 같다.
cardview는 recyclerview에 들어가게 된다.



ㄷ. menu

메뉴를 사용한 searchview는 좀더 나은 UI를 구성하게 해준다. 따라서 menu에 viewclass를 searchview로 설정하고 icon을 바꿔주고 아이콘을 클릭시 검색을 가능하게 한다.

ㄹ. mainactivity

main activity layout에서 버튼을 클릭시 2D MAP으로 intent시켜주기 위해 onclick을 설정한다. 또한 cardview에 들어갈 데이터를 불러오기 위한 구문을 설정한다. 또한 menu 위치를 설정해준다.

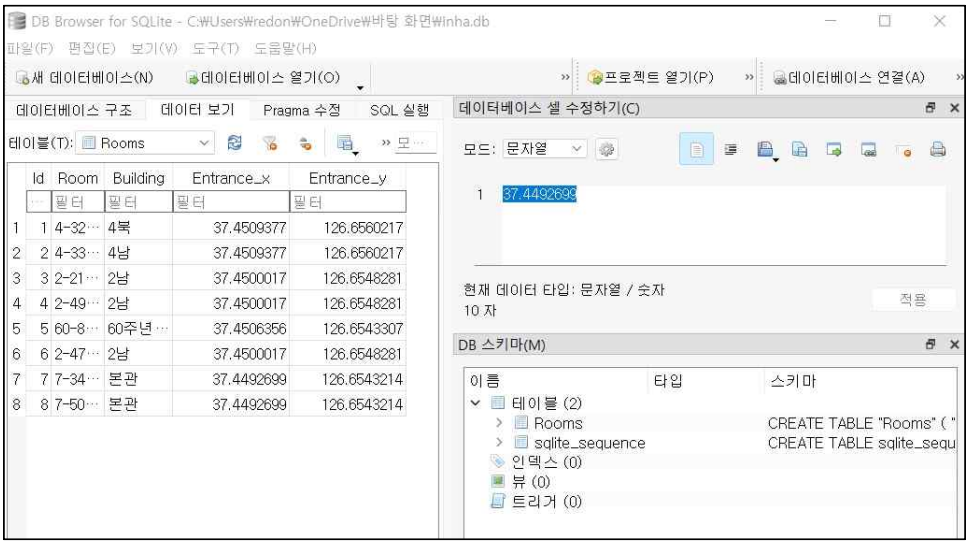
```
MenuInflater inflater = getMenuInflater();
inflater.inflate(R.menu.example_menu, menu);
```

```
SearchView searchView = (SearchView) searchItem.getActionView();

searchView.setImeOptions(EditorInfo.IME_ACTION_DONE);
searchView.setQueryHint("검색하시오");

searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
    @Override
    public boolean onQueryTextSubmit(String query) { return false; }

    @Override
    public boolean onQueryTextChange(String newText) {
        adapter.getFilter().filter(newText);
        return false;
    }
});
return true;
```

나. Database 제작

ㄱ. DB 생성

- sqlite를 다운받고 DB를 생성한다.



Id	Room	Building	Entrance_x	Entrance_y
1	4-32…	4북	37.4509377	126.6560217
2	4-33…	4남	37.4509377	126.6560217
3	2-21…	2남	37.4500017	126.6548281
4	2-49…	2남	37.4500017	126.6548281
5	60-8…	60주년…	37.4506356	126.6543307
6	6 2-47…	2남	37.4500017	126.6548281
7	7 7-34…	본관	37.4492699	126.6543214
8	8 7-50…	본관	37.4492699	126.6543214

- DB를 open하는 구문을 만든다.

database를 사용하게 해주는 openhelper class를 생성하고 database에 대한 정의를 한다. query를 설정해서 어느 속성을 사용할지 결정한다.

```
public void OpenDatabase ()
{
    String DBPath = mContext.getDatabasePath(DBNAME).getPath();
    if (mDatabase != null && mDatabase.isOpen()) {
        return;
    }
    mDatabase = SQLiteDatabase.openDatabase(DBPath, null, SQLiteDatabase.OPEN_READWRITE);
}
```

```
public class DatabaseHelper extends SQLiteOpenHelper
    {
        public static String DBNAME;//db이름
        public static String TABLE;//db테이블이름
        public static final String DBLOCATION = "/data/data/" + G.context.getPackageName() + "/databases/";
        private Context mContext;
        private SQLiteDatabase mDatabase;
    public DatabaseHelper(Context context, String DBname)
        {
            super(context, DBNAME, null, 1);
            DBNAME = DBname + ".db";
            TABLE = "room";
            mContext = context;
        }
```

• DB를 이용하기 위해 adapter class를 생성하고 연결시킨다.
list를 adapter와 연결시켜 DB에서 open한 data를 연결시킨다.
viewholder를 사용함으로써 어떻게 data를 시각화 할지 결정한다.
itemlist(목적지)를 클릭시 어떻게 화면이 보여질지와 검색버튼을 클릭시 recylcerview가 어떻게 보여질지도 설정한다.

```
TextView textView =findViewById(R.id.txt_room);
DatabaseHelper mDBHEPLER=new DatabaseHelper(MainActivity.this,"inha");
File database = getApplicationContext().getDatabasePath(DatabaseHelper.DBNAME);
if(!database.exists())
{
    mDBHEPLER.getReadableDatabase();
    if(!copydatabase(MainActivity.this))
    {
        return;
    }
}
arrayList= mDBHEPLER.GetDesti();
adapter =new RoomAdapter(arrayList);
adapter.notifyDataSetChanged();
destination.setAdapter(adapter);
destination.setTextFilterEnabled(true);

searchView = findViewById(R.id.search_bar);
searchView.setQueryHint("검색하시오");
```

다. 검색 기능 구현

불러온 data들을 textview에 adapter를 사용해 list로 만들어 넣었다면 이를 검색 할 수 있도록 filter를 사용한다.

filter를 사용할 리스트를 만들어 목적지 검색 시 글자가 매칭되는 목적지가 나오게 끔 만든다. 이때 강의실 호수나 명칭으로 검색을 할 수 있다. 검색 시 나타나는 결과(목적지)에 onclick을 주어서 목적지 설정에 대한 확인여부를 묻는 dialog를 생성한다. 이때 intent를 사용해 좌표를 전달한다.

```
public void onBindViewHolder(ExampleViewHolder holder,int position){
    ExampleItem currentItem = exampleList.get(position);

    holder.textView1.setText(currentItem.getText1());
    holder.textView2.setText(currentItem.getText2());
    holder.itemView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String room=holder.textView1.getText().toString();
            String hool=holder.textView2.getText().toString();
            String lat=holder.textView3.getText().toString();
            String lon=holder.textView4.getText().toString();

            Intent intent = new Intent(mContext,SearchActivity.class);
            intent.putExtra("Room",room);
            intent.putExtra("Hool",hool);
            intent.putExtra("LAT",lat);
            intent.putExtra("LON",lon);
            mContext.startActivity(intent);

        }
    });
```

```
public Filter getFilter() { return exampleFilter; }
private Filter exampleFilter = new Filter() {
    @Override
    protected FilterResults performFiltering(CharSequence constraint) {
        List<ExampleItem> filteredList = new ArrayList<>();

        if (constraint == null || constraint.length() == 0) {
            filteredList.addAll(exampleListFull);
        } else {
            String filterPattern = constraint.toString().toLowerCase().trim();

            for (ExampleItem item : exampleListFull) {
                if (item.getText1().toLowerCase().contains(filterPattern)) {
                    filteredList.add(item);
                }
            }
        }

        FilterResults results = new FilterResults();
        results.values = filteredList;

        return results;
    }
```

라. 2D 지도 수정 및 보완

본 프로젝트는 도보 환경에서의 네비게이션을 목표로 하여 수행하였다. 이때, 대부분의 업체(구글, 네이버, 다음 등)에서 제공하는 교내 지도는 인도를 고려하지 않고 차가 통행 가능한 길을 위주로 제작되어 있었다. 따라서 최단 거리 경로로 안내를 하지만 해당 경로는 도보로 통행가능한 길을 고려하지 않아 절대적인 최단거리가 아닌 경로이다. 이러한 점을 보완하기 위해 건물의 출입구와 인도를 추가 보완하여 지도를 수정하였다.

ㄱ. 출입구 수정

아래와 같이 OpenStreetMap에서 직접 지도의 출입구 개체 및 해당 속성을 추가, 수정 하였다.

ㄴ. 인도 수정

아래와 같이 OpenStreetMap에서 직접 지도의 인도 개체 및 해당 속성을 추가, 수정 하였다.

마. 2D 네비게이션 구현

ㄱ. Mapbox Maps API 연결

2D 지도를 사용하기 위해 우리는 Mapbox에서 제공하는 지도를 사용하였다.
해당 지도는 OpenStreetMap에서 제공되는 지도로 앞서 수정 및 보완한 지도를 OpenStreetMap을 통해 우리의 앱에서 사용 가능하다.

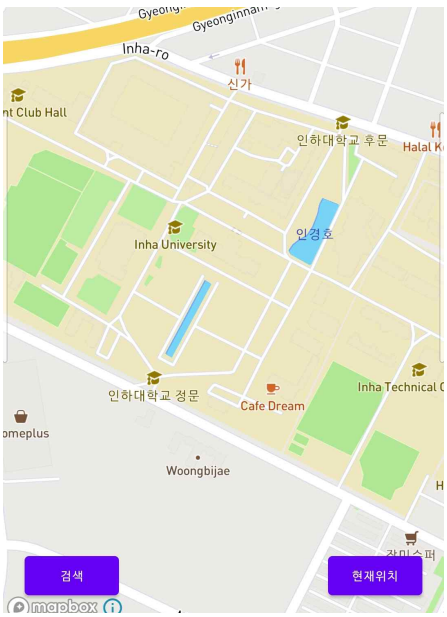
Android Studio 상에서 Gradle에 Implementation 구문을 통하여 특정 버전의 Maps API를 연동 하였다. 이때, API KEY 또한 build.Gradle에 입력하였다.

```
allprojects {
    repositories {
        maven { url 'http://maven.google.com'
            allowInsecureProtocol=true}
        maven { url 'https://jitpack.io' } // jitpack : 안드로이드, JVM 형태의 오픈소스 라이브러리 배포 플랫폼
        maven { url 'https://api.mapbox.com/downloads/v2/releases/maven'
            authentication {
                basic(BasicAuthentication)
            }
            credentials {
                // Do not change the username below.
                // This should always be `mapbox` (not your username).
                username = 'mapbox'
                // Use the secret token you stored in gradle.properties as the password
                password = project.properties['MAPBOX_DOWNLOADS_TOKEN'] ?: "sk.eyJ1IjoicGFyazUwNTBrciIsIml
            }
        }
        google()
    }
}
```

```
implementation 'com.mapbox.mapboxsdk:mapbox-android-sdk:9.3.0'
implementation 'com.mapbox.mapboxsdk:mapbox-android-core:2.0.1'
```

ㄴ. 2D 지도 구현

앞서 연동한 Maps API를 통해 지도화면을 구현하였고 해당 지도 화면은 아래와 같다.



ㄷ. Mapbox Navigation API 연결

앞서 구현한 지도상에 목적지 까지의 경로를 표시하기 위해서는 Mapbox Map과 연동되는 Mapbox Navigation API가 필요하다. Navigation API가 제공하는 기능인 Direction을 사용하여 출발지 좌표와 도착지 좌표를 입력하면 최단 경로와 잔여거리 등을 리턴 받을 수 있다.

이 또한 Maps API와 마찬 가지로 Gradle에 Implementation 구문을 통하여 특정 버전의 Navigation API를 연동하였다.

```
allprojects {
    repositories {
        maven { url 'http://maven.google.com'
            allowInsecureProtocol=true}
        maven { url 'https://jitpack.io' } // jitpack : 안드로이드, JVM 형태의 오픈소스 라이브러리 배포 플랫폼
        maven { url 'https://api.mapbox.com/downloads/v2/releases/maven'
            authentication {
                basic(BasicAuthentication)
            }
            credentials {
                // Do not change the username below.
                // This should always be `mapbox` (not your username).
                username = 'mapbox'
                // Use the secret token you stored in gradle.properties as the password
                password = project.properties['MAPBOX_DOWNLOADS_TOKEN'] ?: "sk.eyJ1IjoicGFyazUwNTBrciIsIml
            }
        }
        google()
    }
}
```

```
implementation 'com.mapbox.mapboxsdk:mapbox-android-navigation-ui:0.42.6'
```

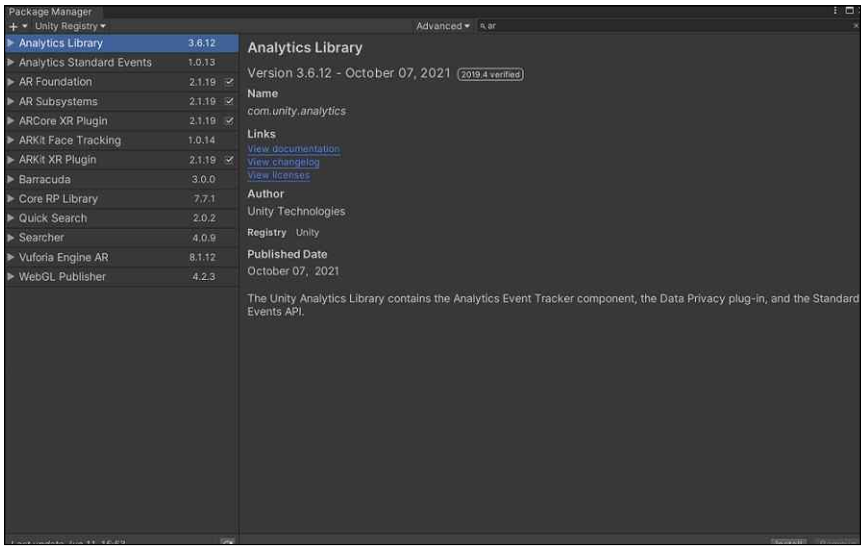
ㄹ. 2D 네비게이션 구현 및 경로 표시

앞서 구현한 지도와 연동된 Navigation API를 사용하여 경로를 표시하였다. 출발지 위치는 디바이스의 GPS 센서를 사용한 현재 좌표, 도착지 위치는 검색 Activity에서 Intent로 받아온 좌표를 사용하였다. 해당 경로는 아래와 같이 나타난다.

또한 앞서 연동된 Navigation API와 리턴 받은 경로를 사용하여 API 상에 2D 네비게이션 실행을 요청하였다. 해당 화면은 아래와 같다.

바. AR 네비게이션 구현

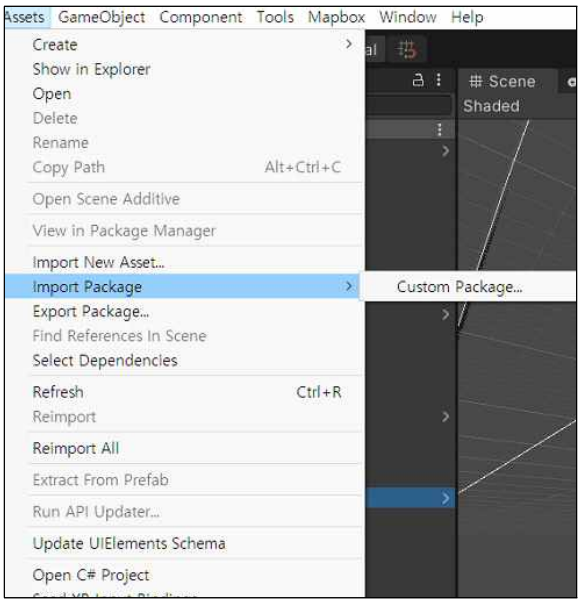
ㄱ. AR 플러그인 설치



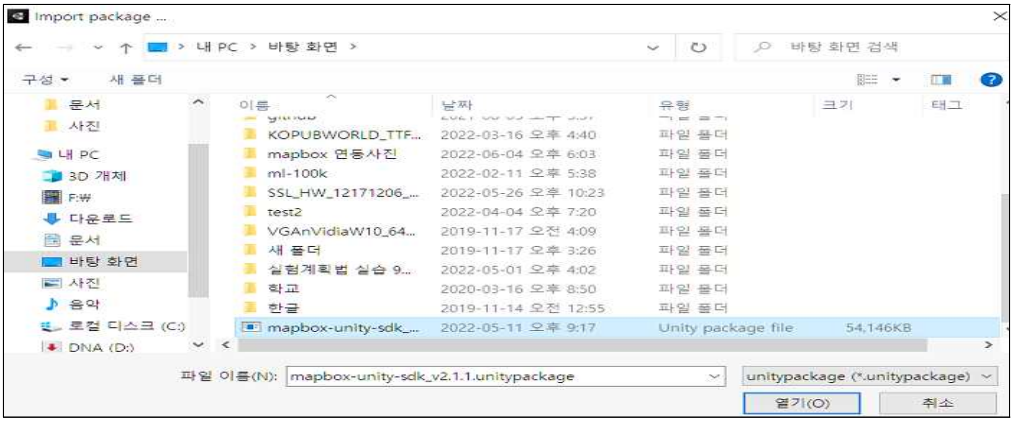
맵박스의 AR 기능을 사용하기 위해 Window-Package Manager에서 ARCore XR Plugin, AR Foundation을 설치한다.

ㄴ. Mapbox, Unity 연동

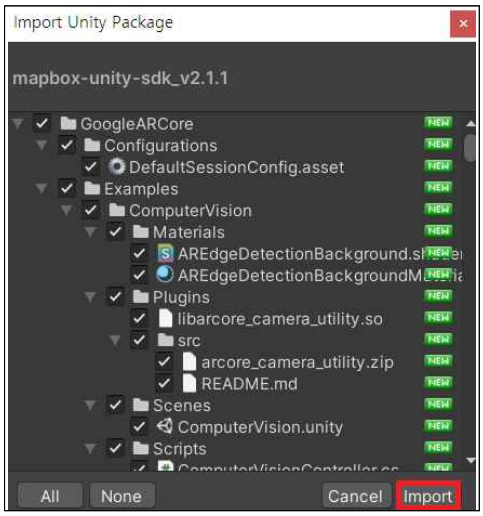
Mapbox에서 제작한 지도와 경로를 설정해주는 Direction API를 사용하기 위해 Mapbox와 Unity를 연동해야한다. 연동하는 과정은 다음과 같다.



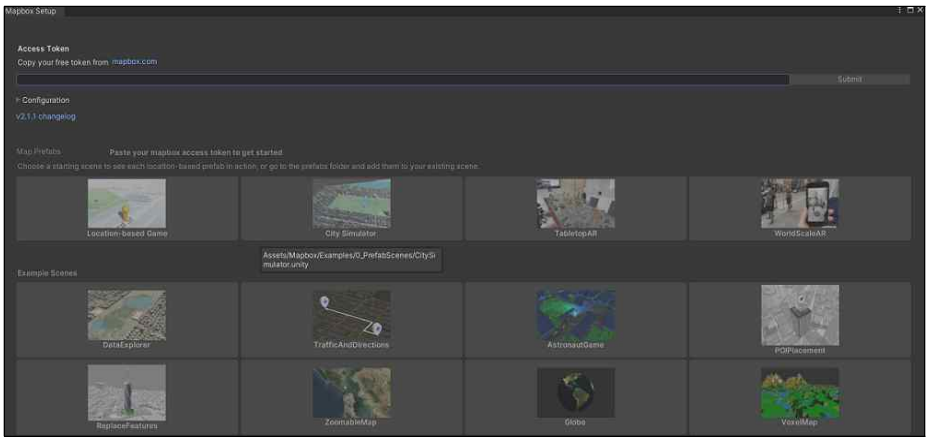
- Asset-Import Package-Custom Package 클릭한다.



– https://www.mapbox.com/install/unity/ 에서 다운받은 SDK를 연동한다.



- 연동이 완료되면 Import Unity Package에서 Import를 클릭해 설치를 진행한다.

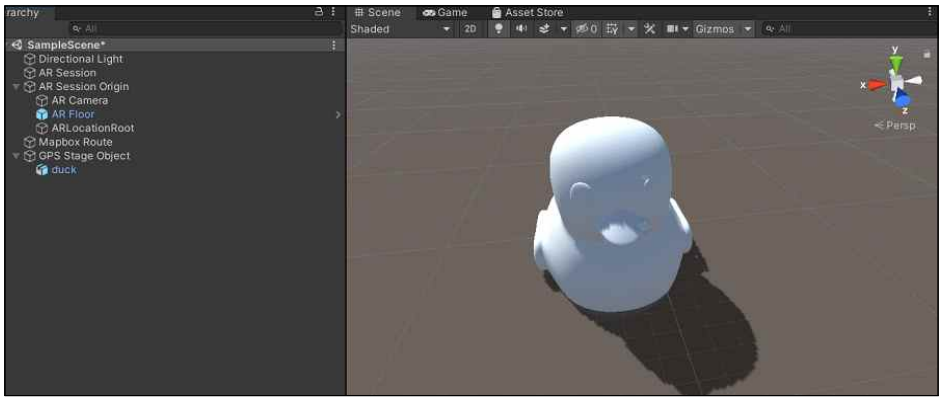


– Import가 완료되면 맵박스 토큰을 입력하는 창이 출력된다. 미리 제작한 지도의 토큰 입력 후 'Submit'을 클릭한다.

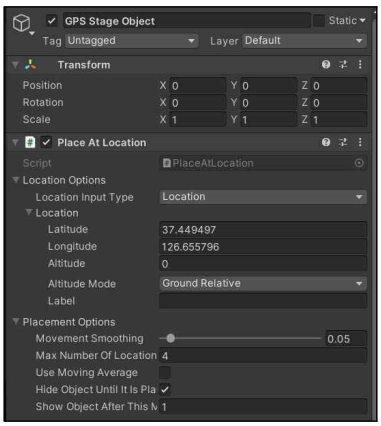
ㄷ. 오브젝트 구현

화장실, 흡연장과 같은 공공장소나 지역을 대표하는 랜드마크의 위치를 3D 오브젝트로 구현했다.

– 오브젝트 파일을 Meshes 폴더에 추가한다.



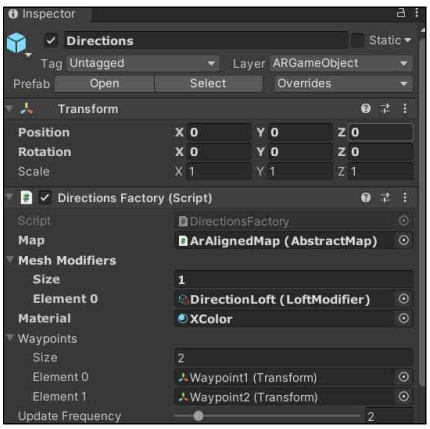
– 폴더에 추가한 오브젝트 파일에서 확장자명이 .fbx인 파일을 GPS Stage Object에 추가한다.



– GPS Stage Object 오브젝트의 위치를 설정하는 Place At Location에서 경도와 위도를 입력한다.

ㄹ. 길찾기 루트 구현

길찾기 구현은 Direction API를 사용했다. Direction API의 Directions.prefab에서 Waypoint를 지정해주면 지정한 시간마다 Direction API에 요청을 보내주고 응답을 한다. 응답을 통해 Waypoint 사이의 경로를 갱신해준다.

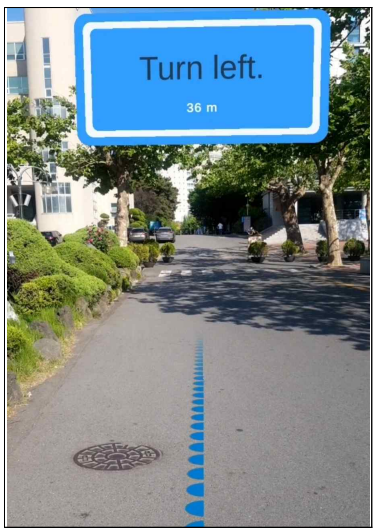


- directions에서 waypoint를 지정해준다. waypoint1은 현재 위치, waypoint2는 목적지로 설정했다.

```
74          void Query()
75          {
76              var count = _waypoints.Length;
77              var wp = new Vector2d[count];
78              for (int i = 0; i < count; i++)
79              {
80                  wp[i] = _waypoints[i].GetGeoPosition(_map.CenterMercator, _map.WorldRelativeScale);
81              }
82              var _directionResource = new DirectionResource(wp, RoutingProfile.Walking);
83              _directionResource.Steps = true;
84              _directions.Query(_directionResource, HandleDirectionsResponse);
85          }
```

- 도보로 이동하므로 DirectionsFactory 스크립트에서 Driving을 Walking으로 변경했다.

ㅁ. 구현된 오브젝트와 루트



- AR 네비게이션 경로

- AR 오브젝트

3. 결과 및 보완점

가. 최종 결과

본 프로젝트의 결과는 “INHA AR NAVIGATION” 안드로이드 어플리케이션으로 목적지 검색 기능, 2D 길안내, 2D 경로 표시, AR 길안내, AR 경로 표시, AR 오브젝트 구현 등 기능을 모두 포함하여 제작 완료하였음.

나. 보완점

먼저, AR 경로 위치 정확도가 개선되어야 한다. 개발한 앱에서는 스마트폰에 장착된 GPS 칩을 사용하여 위치를 받아오게 되는데, 스마트폰에 장착된 GPS는 저가의 GPS로 절대적으로 낮은 위치정확도를 갖고 있다. 테스트 중 실행 시마다 AR 내비게이션상에서 보여야 할 오브젝트나 도착지의 위치가 수m에서 수십m 정도 어긋나는 것을 확인할 수 있었다. 2D 네비게이션의 경우 칼만필터 등을 사용하여 위치 오차를 매핑하여 제공하지만, 현재 우리의 AR 네비게이션의 경우 해당 오차를 개선하여야 한다. 따라서 영상측위 등을 적용하는 것이 해결방법이 될 수 있다. 두 번째로 2D 지도를 수정할 필요가 있다. 본 프로젝트는 도보 환경에서의 네비게이션을 목표로 하여 수행하였다. 이때, 대부분의 업체(구글, 네이버, 다음)에서 제공하는 교내 지도는 인도를 고려하지 않고 차가 통행할 수 있는 길을 위주로 제작되어 있었다. 따라서 최단 거리 경로로 안내를 하지만 해당 경로는 도보로 통행할 수 있는 길을 고려하지 않아 절대적인 최단거리를 제공할 수 없다. 이에 이번 프로젝트에서 해당 지도를 수정하는 작업은 OpenStreetMap 수정을 통해 이루어졌으나 해당 지도가 OpenStreetMap 상에서 승인되고 적용되는 기간이 비교적 길어 본 프로젝트에 수정된 지도를 적용하지 못하였다. 더욱 정확한 도보 경로 및 건물 입구 위치등을 반영하여 수정된 지도를 OpenStreetMap의 승인을 받아 적용하는 과정을 통해 보완하여야 할 점이다. 다음으로 버전 호환성이다. 본 프로젝트에 적용되어 있는 다수의 API와 SDK를 연결하는 과정에서 버전 호환성을 고려하지 못하였다. 그 결과로 현재의 앱은 적어도 우리가 보유하고 있는 안드로이드 폰 기종 중에서는 Galaxy Note 8에서만 정상 작동한다. 따라서 API와 SDK 간의 호환성을 확인하고 버전 업데이트 등을 통해 버전 호환성을 보완해야 한다.