역할

송재곤: 안드로이드 스튜디오를 이용한 폭탄 경로 시뮬레이터 제작

소스코드

MainActivity

|  |
| --- |
| package com.example.myapplication;  import android.app.AlertDialog; import android.content.DialogInterface; import android.os.Bundle; import android.view.View; import android.widget.Button; import android.widget.EditText;  import androidx.appcompat.app.AppCompatActivity;  import com.example.myapplication.databinding.ActivityMainBinding; import com.example.myapplication.ui.dashboard.DashboardFragment;  public class MainActivity extends AppCompatActivity {   private ActivityMainBinding binding;  private Button buttonGetLocation;  private EditText editTextLatitude;  private EditText editTextLongitude;   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  getSupportActionBar().setTitle("Artillery Simulator");   binding = ActivityMainBinding.*inflate*(getLayoutInflater());  setContentView(binding.getRoot());   // EditText 인스턴스 찾기  editTextLatitude = findViewById(R.id.*editTextLatitude*);  editTextLongitude = findViewById(R.id.*editTextLongitude*);   // 버튼 가져오기  buttonGetLocation = findViewById(R.id.*button\_get\_location*);   // 버튼 클릭 이벤트 리스너 설정  buttonGetLocation.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  String latitudeString = editTextLatitude.getText().toString();  String longitudeString = editTextLongitude.getText().toString();   boolean isLatitudeEmpty = latitudeString.isEmpty();  boolean isLongitudeEmpty = longitudeString.isEmpty();   if (isLatitudeEmpty || isLongitudeEmpty) {  showEmptyFieldsAlert(isLatitudeEmpty, isLongitudeEmpty);  } else {  addDashboardFragment();  }  }  });   }   // EditText가 비어 있을 때 경고를 표시하는 메서드  private void showEmptyFieldsAlert(boolean isLatitudeEmpty, boolean isLongitudeEmpty) {  AlertDialog.Builder builder = new AlertDialog.Builder(this);   if (isLatitudeEmpty && isLongitudeEmpty) {  builder.setMessage("위도와 경도를 입력해주세요.");  } else if (isLatitudeEmpty) {  builder.setMessage("위도를 입력해주세요.");  } else if (isLongitudeEmpty) {  builder.setMessage("경도를 입력해주세요.");  }   builder.setPositiveButton("확인", new DialogInterface.OnClickListener() {  @Override  public void onClick(DialogInterface dialog, int which) {  dialog.dismiss();  }  });   AlertDialog dialog = builder.create();  dialog.show();  }    // addDashboardFragment() 메서드 정의  private void addDashboardFragment() {  String latitudeString = editTextLatitude.getText().toString();  String longitudeString = editTextLongitude.getText().toString();   // 위도와 경도 값을 번들에 담아 DashboardFragment에 전달  Bundle args = new Bundle();  args.putString("latitude", latitudeString);  args.putString("longitude", longitudeString);   // DashboardFragment 인스턴스 생성 및 번들 설정  DashboardFragment fragment = new DashboardFragment();  fragment.setArguments(args);   // DashboardFragment를 화면에 추가  getSupportFragmentManager().beginTransaction()  .add(R.id.*fragment\_container*, fragment)  .addToBackStack(null) // 백 스택에 추가하여 뒤로 가기를 지원합니다.  .commit();  }  } |

DashboardFragment

|  |
| --- |
| package com.example.myapplication.ui.dashboard;  import android.annotation.SuppressLint; import android.graphics.PointF; import android.os.Bundle; import android.os.Handler; import android.view.LayoutInflater; import android.view.View; import android.view.ViewGroup; import android.widget.Button; import android.widget.ImageView; import android.widget.TextView;  import androidx.annotation.NonNull; import androidx.fragment.app.Fragment;  import com.example.myapplication.R; import com.naver.maps.geometry.LatLng; import com.naver.maps.map.CameraUpdate; import com.naver.maps.map.MapView; import com.naver.maps.map.NaverMap; import com.naver.maps.map.NaverMapSdk; import com.naver.maps.map.OnMapReadyCallback; import com.naver.maps.map.Projection; import com.naver.maps.map.overlay.Marker; import com.naver.maps.map.overlay.PolylineOverlay;  import java.util.Arrays;  public class DashboardFragment extends Fragment implements OnMapReadyCallback {   private MapView mapView;  private TextView textView;  private ImageView bombImageView;  private Button resetButton;   private LatLng seoulCityHall;  private LatLng receivedLatLng;   private Handler handler = new Handler();  private final int UPDATE\_INTERVAL = 1000; // 1초   @SuppressLint({"MissingInflatedId", "ResourceType", "WrongViewCast"})  @Override  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  View root = inflater.inflate(R.layout.*fragment\_dashboard*, container, false);   // MapView 가져오기  mapView = root.findViewById(R.id.*mapView*);   // TextView 생성 및 설정  textView = root.findViewById(R.id.*time\_layout*);   // ImageView 생성 및 설정  bombImageView = root.findViewById(R.id.*bomb\_image\_view*);   // 리셋 버튼 생성 및 설정  resetButton = root.findViewById(R.id.*reset\_button*);  resetButton.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  // 리셋 버튼 클릭 시 위도와 경도 입력 화면으로 이동  goToInputFragment();  }  });   // 네이버 지도 설정  NaverMapSdk.*getInstance*(requireContext()).setClient(  new NaverMapSdk.NaverCloudPlatformClient("y0wyj0cbep"));   // MapView에 비동기 콜백 등록  mapView.onCreate(savedInstanceState);  mapView.getMapAsync(this);   // TextView 설정  textView.setText("폭파까지 남은 시간: ???");   return root;  }   @Override  public void onMapReady(@NonNull NaverMap naverMap) {  // 네이버 지도가 준비되면 호출됩니다.  // 이곳에서 지도 초기화 및 설정 작업을 수행할 수 있습니다.   // 서울 시청의 좌표를 설정하여 지도를 이동시킴  seoulCityHall = new LatLng(37.5662952, 126.9779451);  naverMap.moveCamera(CameraUpdate.*scrollTo*(seoulCityHall));   // 마커 추가 및 선 그리기  Bundle args = getArguments();  if (args != null) {  String latitudeString = args.getString("latitude");  String longitudeString = args.getString("longitude");   if (latitudeString != null && longitudeString != null) {  double latitude = Double.*parseDouble*(latitudeString.trim());  double longitude = Double.*parseDouble*(longitudeString.trim());   receivedLatLng = new LatLng(latitude, longitude);  Marker receivedMarker = new Marker();  receivedMarker.setPosition(receivedLatLng);  receivedMarker.setMap(naverMap);   PolylineOverlay polyline = new PolylineOverlay();  polyline.setCoords(Arrays.*asList*(seoulCityHall, receivedLatLng));  polyline.setWidth(10);  polyline.setColor(0xFF0000FF);  polyline.setMap(naverMap);   double distanceInMeters = seoulCityHall.distanceTo(receivedLatLng);  int timeInSeconds = (int) (distanceInMeters / 1000); // 소수점 제거 후 정수로 변환   // 폭탄 위치를 주기적으로 업데이트하기 위한 Runnable 객체 생성  Runnable updateBombPositionRunnable = new Runnable() {  private final long startTime = System.*currentTimeMillis*();  private final long duration = timeInSeconds \* 1000; // (단위: 밀리초)  private LatLng previousLatLng = seoulCityHall;  private Projection projection = naverMap.getProjection();   @Override  public void run() {  long elapsedTime = System.*currentTimeMillis*() - startTime;  if (elapsedTime > duration) {  // 시간이 다 되었을 때 초기화  resetBomb();  textView.setText("폭파되었습니다");  } else {  // 현재까지의 경과 시간에 따라 폭탄의 위치를 업데이트  double ratio = (double) elapsedTime / duration;  LatLng currentLatLng = new LatLng(  seoulCityHall.latitude + (receivedLatLng.latitude - seoulCityHall.latitude) \* ratio,  seoulCityHall.longitude + (receivedLatLng.longitude - seoulCityHall.longitude) \* ratio  );   // 남은 거리 계산  double remainingDistance = currentLatLng.distanceTo(receivedLatLng);  // 예상 속도 계산  double currentSpeed = remainingDistance / ((double) (duration - elapsedTime) / 1000); // (단위: m/s)  // 예상 도착 시간 계산  int estimatedTimeToArrive = (int) (remainingDistance / currentSpeed); // (단위: 초)   // 남은 시간 표시  textView.setText("폭파까지 남은 시간: 약 " + estimatedTimeToArrive + " 초");   // 폭탄 이미지의 좌표를 지도 상의 좌표로 변환  PointF newPosition = projection.toScreenLocation(currentLatLng);   // 폭탄 이미지의 크기 고려하여 위치 설정  float bombImageCenterX = newPosition.x;  float bombImageBottomY = newPosition.y;  float offsetX = bombImageView.getWidth() / 2;  float offsetY = bombImageView.getHeight();   // 새로운 위치 설정  bombImageView.setX(bombImageCenterX - offsetX);  bombImageView.setY(bombImageBottomY -offsetY);   // 이미지 회전  double angle = Math.*toDegrees*(Math.*atan2*(currentLatLng.latitude - previousLatLng.latitude, currentLatLng.longitude - previousLatLng.longitude));  bombImageView.setRotation((float) -angle + 40); // 회전 적용   // 1초 뒤에 다시 호출하여 위치 업데이트  handler.postDelayed(this, UPDATE\_INTERVAL);   // 이전 위치 업데이트  previousLatLng = currentLatLng;  }  }  };   // 핸들러를 사용하여 폭탄 위치 업데이트를 1초마다 호출  handler.postDelayed(updateBombPositionRunnable, UPDATE\_INTERVAL);  }  }  }   private void resetBomb() {  // 폭탄 초기화 작업 수행  bombImageView.setVisibility(View.*INVISIBLE*); // 폭탄 이미지를 안 보이게 함  // 시점을 도착지의 좌표에 고정  mapView.getMapAsync(new OnMapReadyCallback() {  @Override  public void onMapReady(@NonNull NaverMap naverMap) {  naverMap.moveCamera(CameraUpdate.*scrollTo*(receivedLatLng));  }  });  // 핸들러 중지  handler.removeCallbacksAndMessages(null);  }   private void goToInputFragment() {  // 현재 띄워져 있는 Fragment를 제거하는 함수  requireActivity().getSupportFragmentManager().popBackStack();  resetBomb();  } } |

Inputfragment

|  |
| --- |
| package com.example.myapplication.ui;  import android.os.Bundle; import android.view.LayoutInflater; import android.view.View; import android.view.ViewGroup;  import androidx.annotation.NonNull; import androidx.annotation.Nullable; import androidx.fragment.app.Fragment;  import com.example.myapplication.R;  public class InputFragment extends Fragment {   @Nullable  @Override  public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  // 여기에 레이아웃을 inflate하여 반환합니다.  return inflater.inflate(R.layout.*activity\_main*, container, false);  } } |

나머지 xml 파일및 gradle파일은 분량 관계로 깃허브 주소 참고 부탁드립니다.

<https://github.com/worhs02/ArtillerySimulator/tree/main/My%20Application>

진행과정: 처음엔 unity, godot등 게임 엔진을 통해 시뮬레이터를 구현하고자 하였습니다.

다만 실제로 구현을 하며 여러 회의를 거치던 중, 저희가 구현하려는 기능은 지도 api를 통해 구현하는 편이 효율적이고 더 가시성 있게 보여줄 수 있을 것 같다는 결론에 도달했습니다. 해서 저희는 개발환경을 안드로이드 스튜디오로 변경한 후 주제를 조금씩 다듬어가며 현재 시뮬레이터에 도달하였습니다.

시뮬레이터 기능: 범준님이 만든 좌표표시기를 통해 좌표를 구하고 구한 좌표를 입력하면 입력된 좌표로 폭탄이 날아가는 일직선 상의 거리를 지도에 표시하고 날아가는 모습을 시각화 하였습니다. 위도나 경도가 입력되지 않으면 입력하라는 메시지로 알림을 띄우도록 설정했습니다. 폭탄의 초기 위치는 백골 신병교육대에 임의로 설정하였습니다. 폭탄의 머리부분이 항상 목적지를 향하도록 조정했습니다. 폭탄의 속도는 980m/s로 설정하였으며, 폭탄의 위치를 1초마다 갱신 시켰습니다. 폭탄이 목적지에 도착하면 화면에서 보여주던 남은 도착 예상 시간을 “폭파 되었습니다” 메시지로 변경하였습니다. 또한 리셋 버튼을 통해 초기 위도와 경도를 입력 받는 화면으로 돌아가 재입력 할 수 있게 했습니다. 전체적인 UI는 군대의 초록색으로 통일감을 높였으며, 최근 굳건이에서 변경된 힘찬이를 통해 꾸며보았습니다.

개선사항 및 느낀 점: 소수 인원을 통한 프로젝트였고 각자 다른 어플, 프로그램을 만들었기 때문에 혼자 프론트와 백을 모두 경험해 볼 수 있는 좋은 기회였던 것 같습니다. 다만 앞에서 말씀드린 대로 스스로 여러 부분을 진행해야 했기에 초기와 달라진 점도 많고, UI에서 조금 아쉬운 부분을 느꼈습니다. 감사합니다