SuperMap iMobile

AR Map

SuperMap Software Co., Ltd.

11, 2019 • Beijing

Legal Statement

The concerned software products and the updated products hereinafter in this document are developed and sold by SuperMap Software Co., Ltd.

Hereby declare.

SuperMap Software Co., Ltd.

Address: 6/F Unit E, Building 107, No. A10, Jiuxianqiao North Road,

Chaoyang District, Beijing, 100015, CHINA

Tel: +86-10-59896503

Fax: +86-10-59896666

Technical Support: globalsupport@supermap.com

Sales: request@supermap.com

Website: http://www.supermap.com

Your advice and suggestions are welcome!

Content

1. Overview

Augmented Reality (AR) is a new technology which can enhance real objects that exist in the real world, combine virtual and real worlds, and be perceived by humans.

Together with image recognition, real-time sensor, SLAM, GPS, and so on, SuperMap iMobile (hereinafter referred to as iMobile) can combine virtual and real worlds to solve the real problems in map recognition. Specific features include:

- **Multiple AR Modes**

Multiple AR display modes are provided to meet different requirements.

- Nearing: independent on maps, this mode can recognize, obtain, and display POI positions, distances, directions, attributes, and so on. It can be used in surface feature recognition, map browse, feature query, POI search, and so on.
- Following: combining real words and arrows, it can give directions of targets and position your destinations with your phone camera. It can be used in navigation.
- Infinite: this mode is used for browsing maps without dragging them. You can choose any one of them or combine them to get much better experiences.

- **AR Project**

Projection based on positions, directions, POI, and so on are supported. The targets which can be projected include images, text, maps, and videos.

- **AR Map Gesture**

When overlaying AR images and maps, you can manipulate your maps through gestures.

2. Modes Switching

You can switch them among the provided modes.

临近模式

跟随模式

- **Required codes:**

① Load library file: add jar libraries including com.supermap.ar.jar, com.supermap.data.jar, com.supermap.mapping.jar, and gson-2.2.2.jar and so libraries including libimb2d_v1000.so in the folder libs.

② Grant permissions: apart from the permissions required by iMobile, the following permissions should be granted:

```
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-feature android:name="android.hardware.camera.any"/>
<uses-feature android:name="android.hardware.camera" android:required="true"/>
<uses-feature android:name="android.hardware.camera.autofocus"
android:required="true"/>
<uses-feature android:glEsVersion="0x00010100"/>
```

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />

<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-feature android:name="android.hardware.camera.any"/>
<uses-feature android:name="android.hardware.camera" android:required="true"/>
<uses-feature android:name="android.hardware.camera.autofocus" android:required="true"/>
<uses-feature android:glEsVersion="0x00010100"/>
```

③ Add controls: add the control ARControl2. Note: the control ARControl2 only can work in a landscape orientation.

```
<com.supermap.mapping.AR.ArControl2
        android:id="@+id/arcontrol_supermap"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:visibility="visible"/>
```

- **Basic steps**

① Open a map: call functions in the classes Workspace and MapControl to open a workspace and a map.

② Associate AR: associate ARControl2 with Mapcontrol and use the function setIsArmap() to set the AR map.

```
m_Mapcontrol = m_ArControl2.mapControl;
m_Mapcontrol.getMap().setIsArmap(true); // Set a map to a AR map
```

③ Set attributes relative to AR: call functions beginAR(), setARState(), setDatasetName(), setTileName(), and setRecordset() to set AR attributes.

```
m_ArControl2.beginAR(); //starts the AR map mode
m_ArControl2.setARState(true);   //sets AR status
m_ArControl2.setDatasetName("T7_REGION_INFO"); //Sets the POI dataset of the AR map
m_ArControl2.setTileName("FT_NAME_CN"); //Sets POI titles in the AR map
m_ArControl2.setRecordset(mRecordsetAR);      //Sets the record set
```

④ Switch modes: call the function setARMode() in the class ArControl2 to switch modes.

```
m_ArControl2.setARMode(ARMode.AR_NORMAL);      // Normal mode
```

```
m_ArControl2.setARMode(ARMode.AR_NEARING);      // AR nearing mode
m_ArControl2.setARMode(ARMode.AR_FOLLOWING);//AR following mode
m_ArControl2.setARMode(ARMode.AR_INFINITE);      //AR infinite mode
```

- **Reference code:**

```
//①Opens a map
Workspace mWorkspace = new Workspace();
WorkspaceConnectionInfo info = new WorkspaceConnectionInfo();
info.setServer(SDCARD + "SampleData/AR/supermapindoor.smwu"); //Sets the file name
info.setType(WorkspaceType.SMWU);    //Sets the workspace type
boolean bOpen = mWorkspace.open(info);//Opens a workspace
if ( bOpen){
//②Associates AR
    m_ArControl2 = (ArControl2)findViewById(R.id.arcontrol_supermap);
    m_Mapcontrol = m_ArControl2.mapControl;
    m_Mapcontrol.getMap().setWorkspace(mWorkspace);//Sets the workspace which is associated with the map
    String mapName = mWorkspace.getMaps().get(0);//Gets the name of the specified map
    bOpen = m_Mapcontrol.getMap().open(mapName);      //Opens a map
    if (bOpen){
        if (!m_Mapcontrol.getMap().IsArmap()) { //Whether it is an AR map
            m_Mapcontrol.getMap().setIsArmap(true); //Sets it to an AR map
        }
          //③Sets related attributes of AR
        m_ArControl2.beginAR(); //Starts the AR map
        m_ArControl2.setARState(true);   //Sets AR status
        m_ArControl2.setDatasetName("T7_REGION_INFO"); //Sets the POI dataset of AR map
        m_ArControl2.setTileName("FT_NAME_CN"); //Sets the POI titles in the AR map
        Datasource dtSource = mWorkspace.getDatasources().get(0);//Gets the specified datasource
        if (dtSource != null) {
            Dataset datasetDLTB = dtSource.getDatasets().get("T7_REGION_INFO"); //Gets the specified dataset
            if (datasetDLTB != null) {
                DatasetVector plDatasetVector = (DatasetVector) datasetDLTB;
                 Recordset mRecordsetAR = plDatasetVector.getRecordset(false, CursorType.STATIC);//Gets the record set
                 m_ArControl2.setRecordset(mRecordsetAR);      //Sets the record set
```

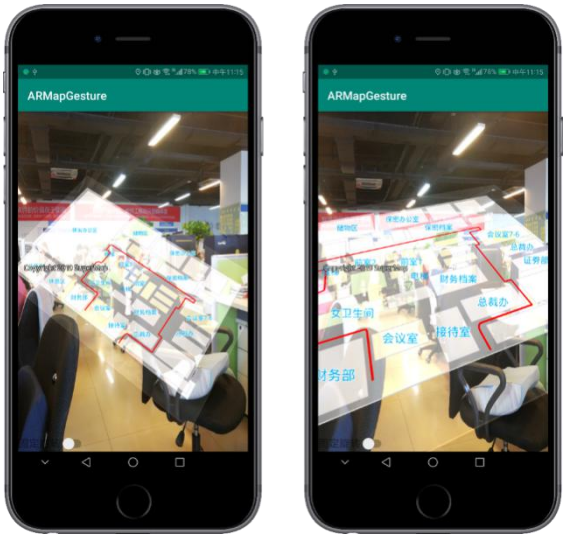
```
                }
            }
        }else{
            Toast.makeText(this,"Failed to open the map.",Toast.LENGTH_SHORT).show();
        }
}
else {
        Toast.makeText(this,"Failed to open the
workspace.",Toast.LENGTH_SHORT).show();
}

//④Switches modes
//Normal
public void buttonMap_Click(View view){
        m_ArControl2.setARMode(ARMode.AR_NORMAL);
}
//AR nearing
public void buttonNearing_Click(View view){
        m_ArControl2.setARMode(ARMode.AR_NEARING);
}
//AR following
public void buttonFollow_Click(View view){
        m_ArControl2.setARMode(ARMode.AR_FOLLOWING);
}
//AR infinite
public void buttonnInfinite_Click(View view){
        m_ArControl2.setARMode(ARMode.AR_INFINITE);
}
//Turns on the camera
public void buttonnOpenCamera_Click(View view){
        m_ArControl2.showCamera();
}
//Turns off the camera
public void buttonnCloseCamera_Click(View view){
        m_ArControl2.hideCamera();
}
//Turns on the map
public void buttonnOpenMap_Click(View view){
        m_ArControl2.showMapView(true);
}
//Turns off the map
```

```
public void buttonnCloseMap_Click(View view){
    m_ArControl2.showMapView(false);
}
```

3. AR Map Gesture

AR map gestures allow you to operate your map freely on an AR scene.



- **Required codes:**

① Load library file: adds jar libraries including com.supermap.ar.jar, com.supermap.data.jar, com.supermap.mapping.jar, and gson-2.2.2.jar and the so library libimb2d_v1000.so in the folder libs.

② Grant permission: apart from the permissions required by iMobile, the following permissions should be granted:

```
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-feature android:name="android.hardware.camera.any"/>
<uses-feature android:name="android.hardware.camera" android:required="true"/>
```

```
<uses-feature android:name="android.hardware.camera.autofocus"
android:required="true"/>
<uses-feature android:glEsVersion="0x00010100"/>
```

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />

<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-feature android:name="android.hardware.camera.any"/>
<uses-feature android:name="android.hardware.camera" android:required="true"/>
<uses-feature android:name="android.hardware.camera.autofocus" android:required="true"/>
<uses-feature android:glEsVersion="0x00010100"/>
```

③ Add controls: MapView and CameraView.

```
<com.supermap.mapping.MapView
    android:id="@+id/MapView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:visibility="visible">
</com.supermap.mapping.MapView>
<com.supermap.ar.CameraView
    android:id="@+id/CameraView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:visibility="visible">
</com.supermap.ar.CameraView>
```

- **Basic steps:**

① Open a map. Call functions in the class Workspace, MapControl to open a workspace and a map.

② Set map parameters. Call functions including setMapOverlay(), addOverlayMap(), enableRotateTouch(), enableSlantTouch(), and SetSlantAngle() to set map parameters.

③ Set the gesture listener. Calls the function setGestureDetector() to add the gesture listener.

④ Gestures. Add map operations including zooming, pitching, rotating, and so on.

- **Reference code:**

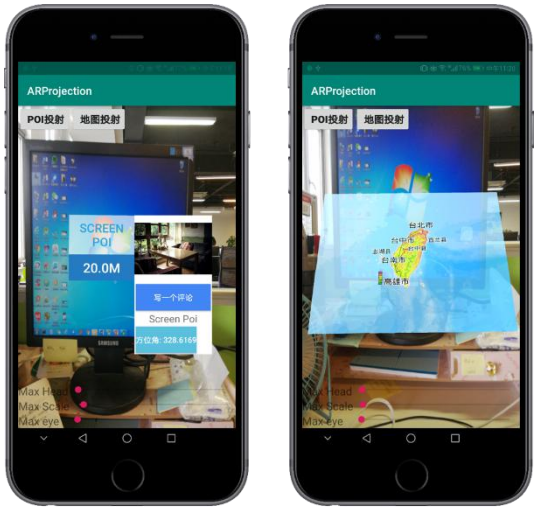
```
//①Opens a map
m_Workspace = new Workspace();
WorkspaceConnectionInfo info = new WorkspaceConnectionInfo();
info.setServer(SDCARD +"/SampleData/AR/supermapindoor.smwu");     //Sets the
  filename
info.setType(WorkspaceType.SMWU);     //Set the workspace type
boolean bOpen = m_Workspace.open(info);
if (bOpen) {
  m_MapView = (MapView) findViewById(R.id.MapView);
  m_MapControl = m_MapView.getMapControl();            //Gets the map control
  m_MapControl.getMap().setWorkspace(m_Workspace); //Gets the workspace which is
associated with the current map
  String mapName = m_Workspace.getMaps().get(0);      //Gets the map name
  bOpen = m_MapControl.getMap().open(mapName);     //Opens the specified map
  if (bOpen){
  //②Sets the map paramters
    m_MapControl.getMap().setAlphaOverlay(true);
    m_MapControl.setMapOverlay(true);        //Sets the layers available to the map
    m_MapControl.getMap().setCenter(new Point2D(116.512230,39.991812)); //Sets
      the center of the map
    m_MapView.addOverlayMap(m_MapControl);     //Adds the overlaid map
    m_MapControl.enableRotateTouch(true);            //Allows to rotate a map
    m_MapControl.enableSlantTouch(true);             //Allows to pitch a map
    m_MapControl.getMap().setIsArmap(true);          //Sets the AR map mode
    m_MapControl.getMap().setARMapAlpha(0.5f);   //AR map transparency
    m_MapControl.getMap().setARScrollEnable(true);
    m_MapControl.getMap().SetSlantAngle(30);      //Sets the initial angle
    //③Sets the gesture listener
    m_MapControl.setGestureDetector(new
GestureDetector(m_MapControl.getContext(), new
GestureDetector.OnGestureListener() {
      @Override
      public boolean onDown(MotionEvent e) {
        return false;
      }
      @Override
      public void onShowPress(MotionEvent e) {
      }
```

```
        @Override
        public boolean onSingleTapUp(MotionEvent e) {
            return false;
        }
        @Override
        public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float
distanceY) {
        //④Gesture operations
            Rectangle2D viewBounds = m_MapControl.getMap().getViewBounds();//Gets
the visible range of the current map
        m_MapControl.getMap().setLockedViewBounds(viewBounds);   //Sets the locked
visible range of the map
        m_MapControl.getMap().setViewBoundsLocked(false);          //Sets the unlocked
visible range of the map
        if (e2.getPointerCount() > 1){
            return true;
        }
        m_endDrawTime = System.currentTimeMillis();
        if(m_endDrawTime - m_startDrawTime > 20)
        {
            if (Math.abs(distanceX) > Math.abs(distanceY)) {
                m_rotateValueOfARMap += distanceX/3   ;
            } else {
                m_elevateValueOfARMap += distanceY * 5;
                m_MapControl.getMap().setARRotateCenter(m_MapControl.getMap().get
Center());
                m_MapControl.getMap().setARScrollValue((float)
m_elevateValueOfARMap);
            }
            m_MapControl.getMap().setAngle(m_MapControl.getMap().getAngle() +
            distanceX/3);
        }
        m_startDrawTime = m_endDrawTime;
        m_MapControl.getMap().refresh();
        return true;
    }
    @Override
    public void onLongPress(MotionEvent e) {
    }
```

```
    @Override
    public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {
        return false;
    }
  }));
 }else{
     Toast.makeText(this,"Failed to open the map.",Toast.LENGTH_SHORT).show();
 }
} else {
    Toast.makeText(this,"Failed to open the workspace.",Toast.LENGTH_SHORT).show();
}
```

4. AR Project

Project POI or a map into an AR scene.



POI投射　　地图投射

- **Required codes:**

① Load library file: adds jar libraries including com.supermap.ar.jar, com.supermap.data.jar, com.supermap.mapping.jar, and gson-2.2.2.jar and the so library libimb2d_v1000.so in the folder libs.

② Grant permission: apart from the permissions required by iMobile, the following permissions should be granted:

```
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-feature android:name="android.hardware.camera.any"/>
<uses-feature android:name="android.hardware.camera" android:required="true"/>
<uses-feature android:name="android.hardware.camera.autofocus"
android:required="true"/>
<uses-feature android:glEsVersion="0x00010100"/>
```

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />

<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-feature android:name="android.hardware.camera.any"/>
<uses-feature android:name="android.hardware.camera" android:required="true"/>
<uses-feature android:name="android.hardware.camera.autofocus" android:required="true"/>
<uses-feature android:glEsVersion="0x00010100"/>
```

③ Add controls: MapView and ArView. Add CameraView dynamically.

```
<com.supermap.mapping.MapView
    android:id="@+id/MapView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:visibility="visible">
</com.supermap.mapping.MapView>
<com.supermap.ar.ArView
    android:id="@+id/arView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
</com.supermap.ar.ArView>
```

- **Basic steps:**

① Parameter Settings

(1). Set the drawing mode. Call the function ARRendererInfoUtil.saveARRendererMode() and set the drawing mode to MODE_PROJECTION

```
ARRendererInfoUtil.saveARRendererMode(this,ARRendererInfoUtil.MODE_PROJECTION);
```

(2). Add the camera layout. Add CameraView to the layout.

(3). Initialize and register the sensor. Call the function registerListener() to register the magnetic field sensor and the acceleration sensor.

(4). Initialize ARView. Add the functions setOnClickArObjectListener() and setArViewAdapter().

(5). Create augmented reality world and associate ARView. Call the function setWorld() of the class ArView.

(6). Initialize the map and associate ARView. Call functions of the classes Workspace and MapControl to open the workspace and the map. Call the functions setMapView() and setMapcontrol() of the class ArView to associate ARView with the map.

(7). Initialize the progress bar. This parameter is optional. Control the height of the observation point, the size and the maximum visible range of the projected target.

```
m_ArView.setHead((float)progress);    //Sets the height of the observation point
m_ArView.setDistanceFactor((float)progress/10); //Sets the size of the projected point (POI/map)
m_ArView.setMaxDistanceToRender(progress);    //Sets the maximum visible range
```

(8). Set smoothing value. How much the smoothing value is depends on the device. The smaller the value is, the slower the AR scene moves.

```
LowPassFilter.ALPHA = 0.038f;
```

(9). Add the map refreshing Timer. The parameter is used for delaying map refreshing thereby saving resources when projecting a map.

② POI Projection. Create a new xml layout for POI display. You can customize content.

(1). Create a GeoObject.

(2). Set the latitude and longitude of GeoObject.

(3). Set the name of GeoObject.

(4). Set how to display GeoObject. After that, call the function storeArObjectViewAndUri () of the class ArView to save the setting to the layout.

```
m_ArView.storeArObjectViewAndUri(poiView,tempArObject);
```

(5). Add to an AR scene. Call the method addArObject() of the class World.

```
m_World.addArObject(tempArObject);
```

③ Map Projection

(1). Add a map and set it visible.

(2). Set the latitude and longitude of GeoObject.

(3). Set the displaying style. After that, call the function storeArObjectViewAndUri() of the class ArView to save the setting to the layout.

```
m_ArView.storeArObjectViewAndUri(view,tempArObject);
```

(4). Add to an AR scene. Call the method addArObject() of the class World.

```
m_World.addArObject(tempArObject);
```

- **Reference codes:**

```
//①Sets the drawing mode which must be put before setContentView
ARRendererInfoUtil.saveARRendererMode(this,ARRendererInfoUtil.MODE_PROJECTI
ON);
setContentView(R.layout.activity_main);
//②Adds the camera layout
```

```
AddCarmeraView();
//③Initializes and registers the sensor
InitAndRegisterSensor();
//④Initializes ARView
InitARView();
//⑤Creates the augmented reality world and associates ARView
InitARWord();
//⑥Initializes the map and associates ARView
InitMap();
//⑦Initializes the progress bar (optional)
initSeekbar();
//⑧Sets the smoothing value
initLowPassFilter();
//⑨When projecting a map, delay the map refreshing to save resources.
new Handler().postDelayed(new Runnable(){
      public void run(){
            setTimer();
      }
}, 3000);

//②Add the camera to a layout
public void AddCarmeraView(){
   RelativeLayout mRelativeLayout = findViewById(R.id.relativeAR);
   FrameLayout.LayoutParams cameraViewParams = new
    FrameLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
    ViewGroup.LayoutParams.MATCH_PARENT);
   CameraView mArCameraView = new CameraView(this);
   mRelativeLayout.addView(mArCameraView, 0,cameraViewParams);
}

//③Initialize and register the sensor
public void InitAndRegisterSensor(){
   m_SensorManager = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);
   m_SensorEventListener = new SensorEventListener() {
      @Override
      public void onSensorChanged(SensorEvent event) {
      }
```

```
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }
  };
  m_SensorManager.registerListener(m_SensorEventListener, m_magneticSensor,
SensorManager.SENSOR_DELAY_NORMAL);
  m_SensorManager.registerListener(m_SensorEventListener,
m_accelerometerSensor, SensorManager.SENSOR_DELAY_NORMAL);
  m_magneticSensor =
m_SensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);   //Magnetic field
sensor
  m_accelerometerSensor =
m_SensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);//Acceleration
sensor
}

  //④Initializes ARView
public void InitARView(){
  m_listARObjects = Collections.synchronizedList(new ArrayList<ArObject>());
  m_ArView = findViewById(R.id.arView);
  m_ArView.setOnClickArObjectListener(new OnClickArObjectListener() {
    @Override
    public void onClickArObject(ArrayList<ArObject> arrayList) {//Controls whether to
display the pop-up dialog box
      if (arrayList.size() == 0) {
        return;
      }
      ArObject arObject = arrayList.get(0);
      if (m_listARObjects.contains(arObject)) {
        m_listARObjects.remove(arObject);
      } else {
        m_listARObjects.add(arObject);
      }
    }
  });
  m_ArView.setDistanceFactor(0.8f);     //Sets the default size of POI
  m_ArView.setArViewAdapter(new ArViewAdapter(this) {//Sets an adapter to make it
              available that drawing a view on the top of the AR view
    @Override
    public View getView(ArObject arObject, View view, ViewGroup viewGroup) {
```

```
        if (!m_listARObjects.contains(arObject)) {
            return null;
        }
        LayoutInflater inflater = (LayoutInflater)
MainActivity.this.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        if (view == null) {
            view = inflater.inflate(R.layout.ar_object_view, null);
        }
        arObject.setIsShow(true);
        TextView textView = (TextView) view.findViewById(R.id.titleTextView);
        textView.setText(arObject.getName());
        Button button = (Button) view.findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
            }
        });
        button.setText("Azimuth: " + arObject.getAngle().z);
        button.setTag(arObject.getName());
        setPosition(arObject.getScreenPositionTopRight());
        return view;
    }
  });
}

  //⑤Creates the augmented reality world and associates ARView
public void InitARWord(){
    m_World = new World(this);
    m_World.setGeoPosition(23.626947,120.811991);     //Sets the current position
    m_ArView.setWorld(m_World);              //Associates the AR scene
}

    //⑥Initializes the map and associates ARView
public void InitMap(){
  Workspace workspace = new Workspace();
  WorkspaceConnectionInfo info = new WorkspaceConnectionInfo();
  info.setServer(sdcard+"/SampleData/Taiwan/Taiwan.smwu");
  info.setType(WorkspaceType.SMWU);
```

```
  if (workspace.open(info)) {
    m_MapView = findViewById(R.id.mapView);
    m_MapControl = m_MapView.getMapControl();
    m_MapControl.getMap().setWorkspace(workspace);
    String mapName = workspace.getMaps().get(2);
    if (m_MapControl.getMap().open(mapName)) {
        m_MapControl.getMap().setAlphaOverlay(true);
        m_MapControl.setMapOverlay(true);
        m_ArView.setMapView(m_MapView);
        m_ArView.setMapcontrol(m_MapControl);
        m_MapControl.getMap().setIsArmap(true);//Sets to an AR map
        m_MapControl.getMap().setARMapType(5); //Sets the type of ARMap to POI view
mode
        m_ArView.getMapChangedMatrix(m_transformMatirx,m_projectionMatrix);
        m_MapControl.getMap().setTransformMatrix(m_transformMatirx);
        m_MapControl.getMap().setProjectMatrix(m_projectionMatrix);
        m_MapControl.getMap().refresh();
      }
      else{
        Toast.makeText(this,"Failed to open the
map.",Toast.LENGTH_SHORT).show();
        return;
      }
    }
    else{
      Toast.makeText(this,"Failed to open the
workspace.",Toast.LENGTH_SHORT).show();
      return;
    }
}

    //⑦Initializes the progress bar
public void initSeekbar() {
  m_ArView.setMaxDistanceToRender(8);
    //Sets the height of the observation point
  m_SeekHeadFactor = (SeekBar) findViewById(R.id.seekHead);
  m_SeekHeadFactor.setMax(100);
  m_SeekHeadFactor.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
```

```
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {
        if(seekBar == m_SeekHeadFactor) {
            m_ArView.setHead((float)progress);   //Sets the height of the observation point
        }
    }
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {
    }
    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
    }
  });
    //Sets the size of the projected target
  m_SeekDistanceFactor = (SeekBar) findViewById(R.id.seekFactorRender);
  m_SeekDistanceFactor.setMax(100);
  m_SeekDistanceFactor.setProgress(20);
  m_SeekDistanceFactor.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {
        if(seekBar == m_SeekDistanceFactor) {
            m_ArView.setDistanceFactor((float)progress/10); //Sets the visible size of the
projected target
        }
    }
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {
    }
    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
    }
  });
    //Sets the maximum visible range
  m_SeekMaxRender = (SeekBar) findViewById(R.id.seekMaxRender);
  m_SeekMaxRender.setMax(100);
```

```
  m_SeekMaxRender.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
      @Override
      public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {
         if(seekBar == m_SeekMaxRender) {
            m_ArView.setMaxDistanceToRender(progress);    //Sets the maximum visible
range
         }
      }
      @Override
      public void onStartTrackingTouch(SeekBar seekBar) {
      }
      @Override
      public void onStopTrackingTouch(SeekBar seekBar) {
      }
  });
}

      //⑧Sets the smoothing value
public void initLowPassFilter(){
      LowPassFilter.ALPHA = 0.038f;//How much the smoothing value is depends on the device.
}

      //POI projection
public void btnPOIProjection_onClick(View view){
  Point3D intersectionPoint =
m_ArView.getIntersectionPoint(getResources().getDisplayMetrics().widthPixels/2,get
Resources().getDisplayMetrics().heightPixels/2);
  if(intersectionPoint != null){
      //①Creates a GeoObject(POI) and give it an id
      GeoObject tempArObject = new GeoObject(System.currentTimeMillis());
      //②Sets the latitude and longitude of the GeoObject(POI)
      tempArObject.setGeoPosition(m_World.getLatitude()+intersectionPoint.y/10781
7.51838439942D,m_World.getLongitude()+intersectionPoint.x/107817.51838439942
D,m_World.getAltitude()+intersectionPoint.z/107817.51838439942D);
      //③Sets the name of the GeoObject(POI)
      tempArObject.setName("Screen Poi");
      DecimalFormat df = new DecimalFormat("0.00");
```

```
    tempArObject.setDistanceFromUser(Double.parseDouble(df.format(20)));//Adds
the distance information
    //④Sets the display style of the GeoObject(POI)
    View poiView = getLayoutInflater().inflate(R.layout.static_ar_object_view, null);
    Button btnPOIName = (Button)poiView.findViewById(R.id.btn_poi_name);
    btnPOIName.setText(tempArObject.getName());
    Button btnPOIDistance = (Button)poiView.findViewById(R.id.btn_poi_distance);
    btnPOIDistance.setText(""+tempArObject.getDistanceFromUser()+"m");
    m_ArView.storeArObjectViewAndUri(poiView,tempArObject); //Saves UI according
to the layout
    //⑤Adds to an AR scene
    m_World.addArObject(tempArObject);
  }
}

    //Map projection
public void btnMapProjection_onClick(View view){
  List<ArObjectList> arObjectList = m_World.getArObjectLists();
  ArObjectList localARObjectList = arObjectList.get(0);
  for(int i = 0;i<localARObjectList.size();i++){
    if(localARObjectList.get(i).getId() == ArView.PROJECTION_MAP_ID){
        m_World.remove(localARObjectList.get(i));
    }
  }
    //①Adds a layer
  int layersNum = m_MapControl.getMap().getLayers().getCount()
  for(int i = 0;i<layersNum;i++){
     m_MapControl.getMap().getLayers().get(i).setVisible(true);
  }
  Point3D point =
m_ArView.getIntersectionPoint(getResources().getDisplayMetrics().widthPixels/2,get
Resources().getDisplayMetrics().heightPixels/2);
    if(point != null){
    //②Sets the latitude and longitude of the GeoObject(POI)
    GeoObject tempArObject = new GeoObject(ArView.PROJECTION_MAP_ID);
    tempArObject.setGeoPosition(m_World.getLatitude()+point.y,m_World.getLongi
tude()+point.x,m_World.getAltitude()+point.z);
    //③Sets the display style
    tempArObject.setName("");
```

```
        DecimalFormat df = new DecimalFormat("0.00");
        tempArObject.setDistanceFromUser(Double.parseDouble(df.format(20)));//
        Adds the distance information
        m_ArView.storeArObjectViewAndUri(view,tempArObject);//Adds to the AR scene
        m_World.addArObject(tempArObject);
    }
}

private void setTimer(){
        Message message = m_handler.obtainMessage(TIMER);
        m_handler.sendMessageDelayed(message, 1000);
}
private Handler m_handler = new Handler(){
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        switch (msg.what){
            case TIMER:
                m_ArView.getMapChangedMatrix(m_transformMatirx,m_projectionMatrix);
                m_MapControl.getMap().setTransformMatrix(m_transformMatirx);
                m_MapControl.getMap().setProjectMatrix(m_projectionMatrix);
                m_MapControl.getMap().refresh();
                 Message message = m_handler.obtainMessage(TIMER);
                 m_handler.sendMessageDelayed(message, 200);
              break;
              default:
              break;
            }
        }
};

@Override
protected void onResume() {
        super.onResume();
        m_SensorManager.registerListener(m_SensorEventListener, m_magneticSensor,
SensorManager.SENSOR_DELAY_NORMAL);
        m_SensorManager.registerListener(m_SensorEventListener,
m_accelerometerSensor, SensorManager.SENSOR_DELAY_NORMAL);
}
```

```
@Override
protected void onPause() {
    super.onPause();
    m_SensorManager.unregisterListener(m_SensorEventListener);
}
```