NEXREF Visual Intuition SDK integration guide for Android

Updated: 10th May 2017

General description

The library provides functionality of recognition of markers and projection over them given videos in Augmented Reality environment.

It provides a set of callbacks responsible for sharing of content and notifying application about successful marker recognition.

How to ensure embedded native library load correctly

The way the native libraries are searched throughout application files varies between devices. To ensure, this library will look for it's native parts in the correct place the below entry have to be added to **buildConfig**:

```
defaultConfig {
      ....
      ndk {
            abiFilters "armeabi-v7a"
      }
}
```

How to include proguard files

If in your application you're using code obfuscation provided by Proguard then you should attach correct *.pro file to your builds with enabled Proguard, for example:

```
release { minifyEnabled
     true

     proguardFiles
     getDefaultProguardFile('proguard-android.txt')

     proguardFile 'virtual_intuition.pro'

}
```

How to configure dependencies

You should use latest SDK version. Your gradle configuration should look like this:

```
compileSdkVersion 25
buildToolsVersion "22.0.1"
defaultConfig {
  minSdkVersion 14
  targetSdkVersion 25
}
```

In order to add the library to your application, include the dependencies listed below in your build.gradle configuration:

```
     compile 'com.squareup.retrofit2:retrofit:2.1.0'
     compile 'com.squareup.retrofit2:converter-gson:2.1.0'
     compile 'com.squareup.retrofit2:adapter-rxjava:2.1.0'
     compile 'com.squareup.okhttp3:logging-interceptor:3.4.1'
     compile 'com.squareup.okhttp3:okhttp:3.4.1'

     compile 'io.reactivex:rxandroid:1.1.0'

     compile 'com.github.bumptech.glide:glide:3.7.0'

     compile 'com.android.support:support-v4:25.3.1'
     compile 'com.android.support:recyclerview-v7:25.3.1'
     compile 'com.android.support:appcompat-v7:25.3.1'
```

If you want to use aar library in external project add:

```
compile(name: 'visualintuition-release, ext: 'aar')
```

where ‘visualintuition-relese’ is name of the library file. You should place it in your app/libs folder.

Initial setup - Application

Your application class should extend **VIApplication**.
Activate ViActivity callbacks. Sample code:

```
public class SampleApplication extends VIApplication {

        @Override
        public void onCreate() {
        super.onCreate();
        setViActivityCallback(new VIActivityCallback() {
        @Override
        public void onTargetDetected() {
        Log.d("onTargetDetected", "onTargetDetected: ");
        }
        });

        setViShareCallback(new VIShareCallback() {
        @Override
        public void onShareClicked() {
        Toast.makeText(SampleApplication.this, "Share clicked", Toast.LENGTH_SHORT).show();
        }
        });
        }
}
```

How to invoke

To start VI screen in order to fully use this sdk capabilities one must call:

```
final Intent intent = new Intent(SampleActivity.this,
VICameraActivity.class);
intent.putExtra(Config.PRELOAD_CAMPAIGN, "DC All Access");
startActivityForResult(intent,

        RequestCodes.TARGET_DETECTED);
```

this will effect in starting activity responsible for marker detection and movies projection.
You have to send info about campaign name.

In order to receive information whether target was detected during activity lifetime one may call this activity for result and define onActivityResult method:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
```

```
            if (requestCode == RequestCodes.TARGET_DETECTED &&
            resultCode == RequestCodes.TARGET_DETECTED) {
            Toast.makeText(this, "Target detected and reported via Activity result",
Toast.LENGTH_SHORT).show();
            }
            super.onActivityResult(requestCode, resultCode, data);

}
```

Additional info

All mentioned in this file classes are located under package com.nexref.visualintuition.sdk;
Current version of library is attached.

Communication flow - customer

1. Customer should provide the info about the release date, tasks included in a current sprint and the sprint duration.
2. Customer should support Nexref team answering questions about planned features and/or discovered bugs.
3. Customer should perform internal tests of the received application/library version (sent by Nexref).
4. Customer should report any found issue to Nexref, in time no longer than two workdays since the application/library release date.
5. Customer is releasing his app himself to Google Play Store, otherwise full account access should be provided.

Communication flow - Nexref

1. Nexref team should gather the info about the release date, tasks included in a current sprint and the sprint duration BEFORE sprint starts.
2. Nexref should assign all the tasks from the current sprint to people.
3. Nexref is responsible for setting internal release date for test version.
4. Also for performing tests after every significant change in application/library using test scenarios.
5. Nexref should release a production version and notify customer
6. Team should fix any critical issue reported by customer and resend the fixed production version.