Androbean Studio

ALPS

Easy live wallpapers for Android

Version 6.7

Introduction

ALPS(**A**ndroid **L**ive wall**P**aper **S**ervice) is a Unity plugin that allows you to easily create live wallpapers for android without using AndroidStudio at all. That means, you can create a Unity project with target set as Android and launch it directly as a live wallpaper all from within the Unity editor, in the same way normal Unity apps are built and launched. There is no need to deal with any Java code, or AndroidStudio. ALPS takes care of all that for you.

ALPS is stable, clean and customizable and provides all functionalities that live wallpapers will generally need. ALPS is also super optimized to avoid reloading wherever possible. ALPS also provides configuration to run at lower resolution by using hardware scaler.

Starting from version 4.0, ALPS has inbuilt support for AdMob banner, interstitial and rewarded video ads. ALPS is already being used by many publishers for their live wallpapers.

From version 5.0, ALPS comes with inbuilt InAppPuchase APIs for PlayStore. Since Unity's IAP implementation won't work with a live wallpaper, applications can make use of this new API to implement their freemium models.

From version 6.0, ALPS has support for displaying as Activity. You can dynamically switch between activity and live wallpaper at runtime. This allows your app to work like a normal Android app as well as a live wallpaper at the same time. As always, you can customize the layout file. Version 6 also has API using which you can get Android Context objects for further customization. This version is a completely redesigned implementation and is not strictly backward compatible.

Version 6.5 has ALPS's UnityPlayer handling rewritten and is the most stable one to day. It has been well tested to check that no background processing runs when the live wallpaper is not visible.

Prerequisites

- Tested on Unity 2019.2.0f1
- Android device with live wallpaper support, and of API level 21 or newer.
- Be able to launch your project as a regular Android app on the device.
- AdMob app and ad units setup in AdMob console. (Optional)
- Managed IAP items setup in GooglePlay developer console. (Optional)

Key features

- Create and launch live wallpapers directly from within Unity Editor
- No Java coding or AndroidStudio needed
- Ready made Preference classes such as ColorPicker, Slider, Spinner and many more
- Custom theming for preference activity
- Ready made homescreen and freeform scrolling
- Inbuilt support for AdMob banner, interstitial and rewarded video ads
- Inbuilt IAP support for PlayStore. (Subscriptions are not supported.)
- Dynamically switch between Activity and Live wallpaper at runtime.
- Custom layout for Activity.

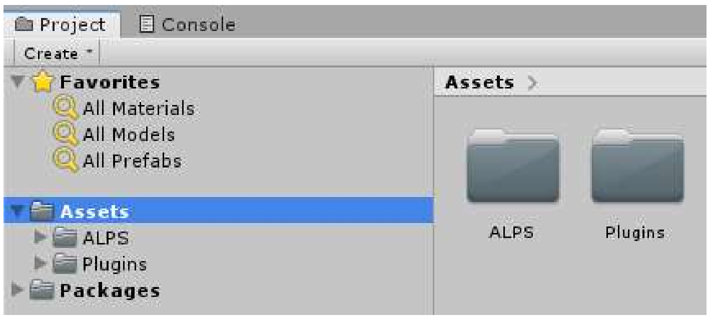
Setting up

Please follow Unity's official documentation/help pages to setup your project as a regular Android app. Once you could launch your project as a regular Android app, you are ready to import ALPS and transform your app into a live wallpaper.

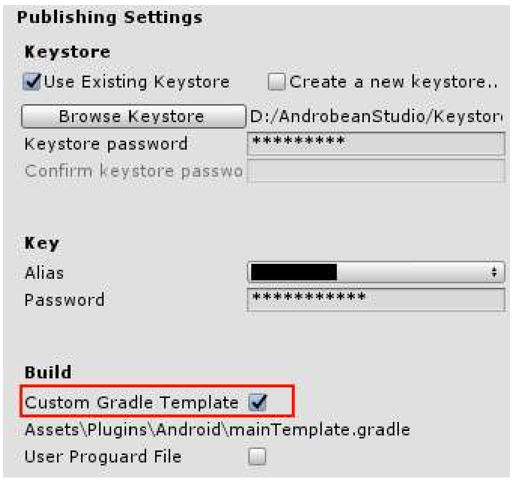
Follow the steps below carefully to setup APLS in your unity project. After setting up, you can launch the live wallpaper using Unity Editors own 'Build and Run' button.

1. Make sure that you can successfully launch your project as a regular Android app on the connected Android device or emulator.
2. Set min API level as 21 in the player settings.
3. Select your 'Assets' folder in Unity editor and import ALPS

4. Select all contents inside ALPS while importing. After import, you should see an 'ALPS' and 'Plugins' folders directly under 'Assets' folder.



5. Since ALPS cannot pack AdMob/Billing libraries inside, it uses custom gradle build feature of Unity. This allows Unity to download the libraries during build time and pack them into the final apk. Towards this, enable custom gradle build in player settings, which will generate the gradle template file at 'Assets/Plugins/Android/mainTemplate.gradle'



Publishing Settings

Keystore

[✓] Use Existing Keystore [ ] Create a new keystore..

Browse Keystore D:/AndrobeanStudio/Keystor

Keystore password *********

Confirm keystore passwo

Key

Alias [redacted]

Password ***********

Build

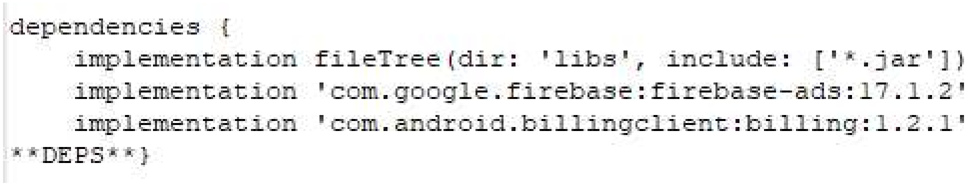
Custom Gradle Template [✓]

Assets\Plugins\Android\mainTemplate.gradle

User Proguard File [ ]

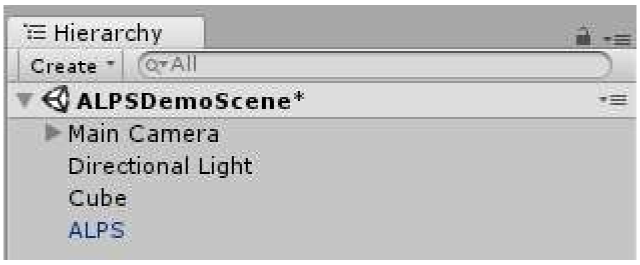
6. If your project needs AdMob, please add AdMob dependency in 'Assets/Plugins/Android/mainTemplate.gradle'. Please use exactly the following version 'com.google.firebase:firebase-ads:17.1.2' which is what I have used for my development and testing.

7. If your project needs IAP, plesae add BillingLibrary dependency in 'Assets/Plugins/Android/mainTemplate.gradle'. Please use the same versions in the picture below, which is what I have used for my development and testing.

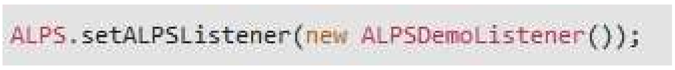


```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.google.firebase:firebase-ads:17.1.2'
    implementation 'com.android.billingclient:billing:1.2.1'
**DEPS**}
```

8. Drag the ‘ALPS’ prefab from ‘Assets/Plugins/ALPS/Core’ folder into your scene as shown below



9. From your c# script register to ALPS by providing an instance of ALPSListener. This is a mandate, and until this is done, the settings button in live wallpaper preview will not work, and on click user will see a toast telling that the live wallpaper is not ready. All the core callbacks such as onSizeChanged, onScroll and events for preference changes etc are delivered to this listener.



```
ALPS.setALPSListener(new ALPSDemoListener());
```

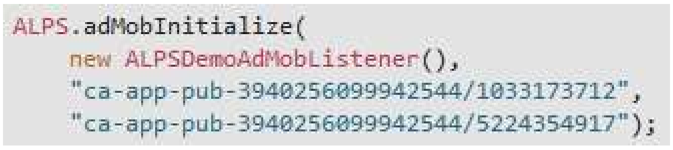
10. Apart from the callbacks, there are a number of public apis from ALPS that you may use. Please refer to the ALPS demo script ‘ALPSDemoListener.cs’ at ‘Assets/Plugins/ALPS/Demo’ for more details and sample usage.
11. At this point, if you click on the ‘Build and run’ button of Unity Editor, your project shall get launched in the Android device as an ALPS live wallpaper, and you shall get callbacks on your ALPS listener any time when scroll position, user preference or window size changes.
12. Make sure to choose/reorder your scenes in the build settings.

Advertisements (AdMob)

Starting version 4.0, ALPS has complete inbuilt support for AdMob. ALPS supports banner, interstitial as well as rewarded video ads formats. You are also provided with all the necessary callbacks and API to have full control over how, when and what type of ads get displayed. At the same time, ALPS takes care of all the fine details and makes integrating AdMob really easy for ALPS users. Please read below for the details on how to handle each type of ads.

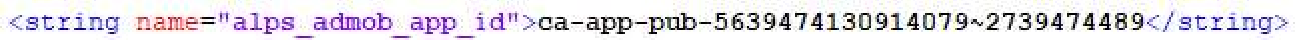
In order to use integrated AdMob in ALPS, you must initialize the same using the following call. First argument is the listener where all AdMob related callbacks are to be received.

Second and third arguments are optional(can be null), and they are the ad unit ids for interstitial and rewarded video ads respectively. While there can be only one each of interstitial and rewarded video ads types, you can have any number of banner ads simultaneously.



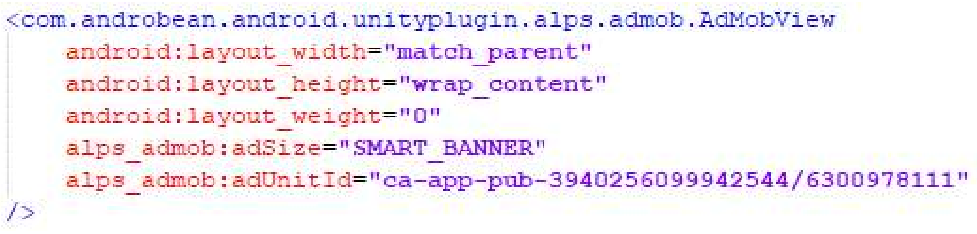
```
ALPS.adMobInitialize(
    new ALPSDemoAdMobListener(),
    "ca-app-pub-3940256099942544/1033173712",
    "ca-app-pub-3940256099942544/5224354917");
```

ALPS fetches the AdMob app id from a string resource by the name ‘alps_admob_app_id’. Make sure to put your your own AdMob app id in the ‘alps_strings.xml’ file under ‘Assets/Plugins/Android/alps_app/res/values/’ folder.



```
<string name="alps_admob_app_id">ca-app-pub-5639474130914079~2739474489</string>
```

To use AdMob banner ads, you can use AdMobView from ALPS, with your ad size and ad unit id specified in alps_preference_activity_layout.xml. This layout file may be edited to your taste as long as the ListView with id “@android:id/list” is preserved.



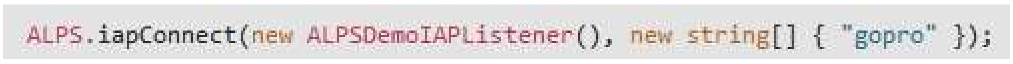
```
<com.androbean.android.unityplugin.alps.admob.AdMobView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="0"
    alps_admob:adSize="SMART_BANNER"
    alps_admob:adUnitId="ca-app-pub-3940256099942544/6300978111"
/>
```

Interstitial and rewarded video ads are loaded automatically in the background if you have supplied ad unit ids during AdMob initialization. You can choose to show these ad format at various points depending on how the application finds it appropriate. You can make use of the many callbacks that ALPS provides such as entering/exiting settings activity, long pressing etc.

In App Purchase (PlayStore)

Starting version 5.0, ALPS provides inbuilt IAP support. This is necessary as Unity’s IAP mechanism won’t work with live wallpaper service. Please note that subscriptions are not supported. Managed items are fully supported with support for consumption as well.

In order to use ALPS IAP, you mist first connect to IAP service using the following call. The first argument is an implementation of ALPSIAPListener, which receives all the asynchronous notifications from the IAP system. The second argument is an array of strings which represents the skus of the managed items you have setup in the GooglePlay developer console.



```
ALPS.iapConnect(new ALPSDemoIAPListener(), new string[] { "gopro" });
```

Customization

1. You can configure your Preferences Activity by editing ‘alps_preferences.xml’ at ‘Assets/Plugins/Android/alps_app/res/xml’. You can use any standard Android preference classes in this file. You can use the following classes from ALPS also.
    1. ClickPreference – Not really a means to save a settings, rather you can use it to listen for a click action on a row. This may be useful to show ‘More apps’ row to the user. When user clicks on this item, ALPS will send a callback to ALPSListener telling the preference got changed.
    2. ColorPreference – This preference saves a color as an integer. It will show a color picker dialog to user and allow them to choose a color.
    3. TextPreference – This preference shows a single line text in an editable field and allows user to see and edit it. This can be used to allow users to set custom strings for some items in your live wallpaper.
    4. SliderPreference – This preference shows a slider to user and allows them to choose a value in the range of 0 to 100. It is saved as an integer.
    5. SpinnerPreference – This preference shows a drop-down list of items to user and allows them to select any one of them. It saves index of the selected item as an integer.
    6. SupportPreference – This is not a means to save a setting, rather it presents users with some buttons to connect with the developer such as email, Google+, PlayStore, Facebook and Twitter.
2. You can setup custom styles for your preference Activity by editing the style elements in ‘alps_styles.xml’ at ‘Assets/Plugins/Android/alps_app/res/values’.
3. If you want custom layout of preference screen, you can edit the layout file ‘alps_preference_activity_layout.xml’ at ‘Assets/Plugins/Android/alps_app/res/layout’ to adapt the settings Activity layout to your designs.

4. If a live wallpaper doesn’t need settings, it can do so by removing the ‘android:settingsActivity’ from the file ‘alps_wallpaper.xml’ under the location ‘Assets/Plugins/Android/alps_core/res/xml/’
5. Set ‘alps_config_launch_activity’ in the file ‘alps_config.xml’ under ‘Assets/Plugins/Android/alps_app/res/values/’ to ‘true’, if you want to start your live wallpaper as an Activity instead of the live wallpaper preview when user clicks the app icon.
6. You can customize the Activity layout by editing the layout files at ‘layout/alps_activity_layout.xml’ and ‘layout_land/alps_activity_layout.xml’ under ‘Assets/Plugins/Android/alps_app/res’. Be sure to not remove the SurfaceView or change its id. Also the wallpaper and settings button ids should not be changed. However you can remove the buttons if not needed.
7. Please see ‘ALPS.cs’ script for a complete list of all the APIs available for the applications to use.
8. Refer to the demo code at ‘Assets/ALPS/Demo’ for usage references.

Troubleshooting

- Changing Preferences doesn’t seem to have any effect.
- Resetting Preferences doesn't seem to have any effect.
- Scroll callbacks are not being fired.
    - Make sure that ALPS prefab is added into the scene.

- Unity reports error while Re-packaging apk.
- App crashes on clicking ‘Settings’ button of live wallpaper preview
    - Make sure that the alps_preferences_xml is a valid xml
    - Make sure that all Preference classes are valid
    - Make sure that all Preference attributes are valid

- No app icon in device’s app drawer
- Clicking ‘Settings’ in live wallpaper preview crashes

    - Make sure that AndroidManifest.xml at ‘Assets/Plugins/Android/ALPS’ and the one at ‘Assets/Plugins/Android’ are not tampered with.

- App launching as regular unity application instead of live wallpaper
    - Make sure that the folders under ‘Assets/ALPS’ are copied into ‘Assets’ folder.
    - Make sure that AndroidManifest.xml at ‘Assets/Plugins/Android/ALPS’ and the one at ‘Assets/Plugins/Android’ are not tampered with.

- Getting “Error while merging dex archives” and reason tells “ Program type already present: com.androbean.android.unityplugin.alps.AlpsActivity”
    - This happens because Unity fails to manage its cache properly when we move the Plugins folder from ALPS to ASSETS. Starting with a fresh project is often easier if this happens.

- Ads are not loading
    - Make sure that AdMob app id and add unit ids are correctly setup in AdMob dashboard.
    - Make sure that AdMob app id is added to ‘alps_strings.xml’.
    - Make sure that AdMob initialize call is being made.
    - Make sure that ad unit ids passed to AdMob initialize call are correct.
    - Make sure that ad unit id and ad size values in layout xml are correct.
    - Make sure that the device has internet connectivity.
    - Sometimes, AdMob may not have any ads to serve, in that case you can try with the test ad unit ids as published by AdMob. Retrying again after some time also works at times.
    - You can debug AdMob issues by adding debug logs in the exhaustive set of callbacks that ALPS provides through AdMobListener.

Support

Your feedback is an opportunity to improve ALPS. Please get in touch with me for support and feature requests by any of the following means.

Email	androbeanstudio@gmail.com
Website	http://www.androbeanstudio.com/alps/product.shtml
Unity forums	https://forum.unity.com/threads/announcing-alps-easy-live-wallpaper-integration-for-android.498194/

Version history

6.7

- Added support for Live Preview inside PreferenceActivity.

6.6

- Updated demo to add custom onClick handling.
- Fixed random crash by using new API to post callbacks to Unity thread.

6.5

- Added new API, launchWallpaperPreview.
- Rewritten UnityPlayer handling to fix issue with background usage.

6.4

- Fixed regression issue of crash when started from live wallpaper picker or when rebooted.

6.3

- Added version number in ALPS.CS for ease.
- Fixed the issue where Preference won't hide if it was under a PreferenceCategory.

6.2

- Minor bug fixes.
- Corrected API definition for setPreferenceName.

6.1

- Fixed AdMob hang in Nexus5
- Fixed crash at startup in Galaxy S8

6.0 (Not backward compatible.)

- Allow dynamic switching between Activity and live wallpaper mode at runtime.
- Separated ALPS into core, billing, admob and app, so that apps can remove billing or admob if they want. Also all app specific customizations are localized to alps_app.
- Replaced 'onVisibilityChanged' with 'onModeChanged'
- SetPreferenceName method got an additional parameter to specify items to hide.

- Fixed IAP issue related to using ‘:’ as delimiter.
- Renamed onSettingsOpened and onSettingsClosed for naming consistency.
- Renamed launchSettingsActivity and closeSettingsActivity for naming consistency.
- Added launchMainActivity and closeMainActivity methods.
- IsPreview method has been replaced with getState.
- Added getContext method to get Android Context object.
- Min Android API version has been set as 21.

5.0

- Added support support for PlayStore IAP
- Added API to change PreferenceActivity layout at runtime
- Added API to close PreferenceActivity programatically
- Added isPreview API to check if wallpaper is in preview mode
- Added callbacks for PreferenceActivity open action

4.0

- Added complete support for AdMob
- Avoids restarting of settings activity on resetting
- Minor bug fix

3.3

- Added new API setResetButtonEnabled(false)
- Added multiple gesture callbacks.
- Added new API launchSettingsActivity()
- Added new API setPreferenceName()
- 'Not ready' message if user tried to click the settings button while loading.

3.2

- Fix for x-axis value becoming negative.
- Restructuring

3.1

- Performance improvement.

3.0

- Support for Unity 2017.1.5 through 2018.2.7
- Fix for screen size/resolution issue.

2.0

- Bug fixes.

1.0

- Initial release.