

uLiveWallpaper

by



General Information

uLiveWallpaper is a Unity extension that allows you to create Live Wallpapers for Android using the full potential of Unity 5 in an efficient and easy way.

Bring your creativity to your phone's home screen with the power of Unity and *uLiveWallpaper*!

Highlights:

- Supports all latest Unity features.
- Supports ARMv7 and x86 devices running Android 2.3+.
- Creates projects for Android Studio — the official IDE for Android application development.
- Well-tested and extremely stable.
- Convenient editor interface. Update your Android Studio project in a single click!
- Clear and simple API.
- Allows to automatically generate a simple Preference Activity for you to expand.
- Full PlayMaker integration.
- Documented demo examples.

Prerequisites

To develop live wallpapers with *uLiveWallpaper*, you'll need the following:

- *Unity 5.3.1* or newer.
- *Android SDK v23* or newer, with API 25 support installed.
- *Android Studio 2.3.0* or newer.

Installing prerequisites is beyond the scope of this manual, but if you are doing Android development, it's likely you have Unity and Android SDK set up correctly already. Refer to this page for more information regarding Android SDK installation:

<http://docs.unity3d.com/Manual/android-sdksetup.html>

Android Studio is the official IDE for Android development. Information on how to install and use Android Studio is available at:

<http://developer.android.com/sdk/index.html>

Usage

Open *uLiveWallpaper* window by calling

Tools → *Lost Polygon* → *uLiveWallpaper*

The window consists of three tabs: “*Create Project*”, “*Update Project*”, and “*About & Support*”.

To start creating your live wallpaper, first you must *create* a Live Wallpaper project for Android Studio. After the initial project is created, in order to reflect the changes made in your Unity project, you’ll need to *update* the project.

Creating and Updating the Android Studio Project

Note: to build and run your live wallpaper project, you *must* use Android Studio. “*Build*” and “*Build And Run*” buttons in “*Build Settings*” window will build a regular Unity project that is not compatible with live wallpapers.

Project Creation (Unity 5.5 and newer)

Step 1

Start *Unity*. Open *uLiveWallpaper* window and select “*Create Project*” tab. Select the destination directory, select project generation options and click the “*Create Project*” button.

This will generate a project that can be opened in *Android Studio*.

Step 2


Start *Android Studio*. If “*Welcome to Android Studio*” splash screen appears, click “*Open an existing Android Studio project*”, otherwise, click *File* → *Open...* in the main menu. Locate and select the project you’ve just created. Click *OK*.

If a “*Gradle Sync*” window asking to use the Gradle wrapper shows up, click *OK*. Wait for project to prepare. If an “*Android Gradle Plugin Update Recommended*” window shows up, click *Don’t remind me again for this project* (Unity only supports Gradle versions up to and including v2.3.3).

Wait for the project to build.

Step 3

Test if your wallpaper project builds fine by doing “*Build* → *Rebuild Project*”. If no errors will occur, you’re all set! Otherwise, recheck previous steps.

To run the live wallpaper on your device, first connect your device to the computer, then click the “*Run*” button (), or use “*Run* → *Run*” main menu item.

See the “*Troubleshooting*” section of this manual if you encounter any issues.

Project Creation (Unity 5.4 and older)

Step 1

Start *Unity*. Open *uLiveWallpaper* window and select “*Create Project*” tab. Select the destination directory, select project generation options and click the “*Create Project*” button.

This will generate an intermediate *Eclipse* project. Now, it needs to be converted into *Android Studio* project.

Step 2

Start *Android Studio*.

In case a “Welcome to Android Studio” splash screen appears:

Click “*Import project (Eclipse ADT, Gradle, etc.)*” and select the project you’ve just created. Click *OK*. “*Import Project from ADT*” window should open. Select the destination directory for imported project. Click *Next*, leave import settings as-is, click *Finish*. Wait for import to succeed. You can delete the intermediate *Eclipse* project now, it is no longer required.

Otherwise:

Click “*File... → New → Import Project...*” and select the project you’ve just created. Click *OK*. “*Import Project from ADT*” window should open. Select the destination directory for imported project. Click *Next*, leave import settings as-is, click *Finish*. Wait for import to succeed.

Note: after importing the project, you will see some build errors. This is normal, these errors will be corrected after next steps.

Step 3

Click “*File... → New → New Module... → Import .JAR/.AAR Package*” and import library, located in “*Assets/uLiveWallpaper/Libraries/*” directory inside your Unity project. Leave all settings by default.

Step 4

Click “*File → Project Structure...*” to open the “*Project Structure*” window. In the “*Modules*” list (on the left), select the “*app*” module. Switch to “*Dependencies*” tab. Click the green “+” button at the right side and add Module dependency “*LP_uLiveWallpaper*”.

Finally, click *OK* to close the “*Project Structure*” window.

Step 5

Test if your wallpaper project builds fine by doing “*Build* → *Rebuild Project*”. If no errors will occur, you’re all set! Otherwise, recheck previous steps.

To run the live wallpaper on your device, first connect your device to the computer, then click the “*Run*” button (▶), or use “*Run* → *Run*” main menu item.

See the “*Troubleshooting*” section of this manual if you encounter any issues.

Project Update

Start *Unity* and open your project. Open *uLiveWallpaper* window and select “*Update Project*” tab. Select the directory of the Android Studio project you want to update, and click the “*Update Project*” button. That’s it! You should now be able to build and run your updated project from *Android Studio*.

In case of any issues, make sure you’ve closed Android Studio and other applications that could be locking the updated Android Studio project directory, and try again.

Note: this will only update the Unity player data, Unity native libraries, and *uLiveWallpaper* plugin. Your Android Studio project (including *AndroidManifest.xml*, application name, icons, etc.) will remain unchanged.

Building Demo Projects

“*Virtual Box*” demo project comes with a custom Android Studio project that includes an example of a custom Preference Activity. It is located at “*Assets\uLiveWallpaper\Demos\VirtualBox\VirtualBox_Android_Studio.zip*”. Unpack the project somewhere, add “*VirtualBox*” demo scene to *Build Settings* and use “*Update Project*” function on the unpacked project.

Other demo projects do not have this requirement.

Settings Activity

Basic

If “*Settings Activity*” is set to “*Basic*” when creating the project, your live wallpaper will include a basic preferences screen with two options: quality settings (matches Quality Settings in Unity), and frame rate limit (lower frame rates result in better battery life).

At some point you might want to extend this screen with some additional options. This is quite straightforward, as standard Android *PreferencesActivity* is used. The layout of the preferences screen is described in “*res/xml/preferences.xml*” file inside your *Android Studio* project.

Official Android documentation on the subject (*"Defining Preferences in XML"* section)

<http://developer.android.com/guide/topics/ui/settings.html>

You might also need to edit *"res/values/strings.xml"* file to change values for your additional options.

Invisible Self-Closing

You might want to avoid using native Android Activity for the settings screen, and instead implement the settings screen yourself, for example, by using Unity UI. To do that, set *"Settings Activity"* to *"Invisible Self-Closing"* and create your project. After that, you must subscribe to `LiveWallpaper.PreferenceActivityTriggered` event in your code. This event will be triggered when you tap the *"Settings..."* button in live wallpaper preview screen. You can then react to this event and show/hide your settings UI.

Home Screen Shortcut

You may want to create a home screen launcher to avoid forcing user into going to wallpaper picker to change the wallpaper settings. There are three options:

1. *"None"*. No home screen shortcut will be generated.
2. *"Open Settings Activity"*. Tapping the shortcut will open the Settings Activity. This option is only available when *"Settings Activity"* is set to *"Basic"*.
3. *"Open Preview Screen"*. Tapping the shortcut will open the wallpaper preview screen on Android 4.1 and newer, or a list of live wallpapers on older Android versions.

Unity Activity

You may want to make your Unity application work as both live wallpaper and a regular application. Just check the *"Create Unity Activity"* checkbox, and `uLiveWallpaper` will automatically generate a Unity Activity that can be used together with live wallpaper service, complete with launcher shortcut.

Keep in mind — this Unity Activity will share the same Unity player instance as your live wallpaper, since it is not possible to correctly instantiate different Unity instances in one process.

It is possible to extend this Activity and modify its behavior. Please see the `UnityPlayerActivity.java` file in the Android Studio project.

There are various custom events sent to Unity in response to Activity events:

<i>Unity Activity event</i>	<i>Custom event sent to Unity</i>
UnityPlayerActivity.onStart()	UnityActivityOnStart
UnityPlayerActivity.onResume()	UnityActivityOnResume
UnityPlayerActivity.onPause()	UnityActivityOnPause
UnityPlayerActivity.onStop()	UnityActivityOnStop
Unity player resumed	UnityPlayerResumed
Unity player paused	UnityPlayerPaused

You must subscribe to `LiveWallpaper.CustomEventReceived` event to receive these events.

PlayMaker Integration

uLiveWallpaper features full PlayMaker integration that includes all *uLiveWallpaper* properties, methods, and events.

After importing *uLiveWallpaper*, you have to also import PlayMaker support package. The package is located at:

Assets\uLiveWallpaper\LiveWallpaper_PlayMakerIntegration.unitypackage

To enable PlayMaker integration, add the *uLiveWallpaper* prefab by clicking the menu item

PlayMaker/Addons/uLiveWallpaper/Components/Add PlayMaker uLiveWallpaper to Scene

After doing that, you'll have a new "*uLiveWallpaper*" category in your PlayMaker actions list.

PlayMaker integration includes a simple "Rotating objects" demo that shows some basics on how to use the actions, and to react to preferences changes. It comes with a custom Android Studio project that includes an example of adding custom preferences. It is located at

"*Assets\LiveWallpaper\PlayMakerIntegration\Demos\RotatingObjects\RotatingObject_Android_Studio.zip*". Unpack the project somewhere, add "*RotatingObjects*" demo scene to *Build Settings* and use "*Update Project*" function on the unpacked project.

API Reference

API reference is available online at:

<http://static.lostpolygon.com/unity-assets/ulivewallpaper/api-reference>

You can also check the included demo scenes for examples on how to practically use the *uLiveWallpaper* API.

Disabling Unity Sound System

By default, *Unity* activates its sound system on startup. This is an issue for live wallpapers, since when sound system is enabled, device volume buttons control media volume instead of ringtone volume, which is a major annoyance for a lot of users. To disable the sound system, go to

Edit → *Project Settings* → *Audio*

and check “Disable Unity Audio”. This will disable the sound system in runtime, which means you won’t be able to play sounds in your live wallpaper, but device volume buttons will control the ringtone volume as intended.

Wallpaper Offset Data Emulation

Some launchers on some devices do not report the scrolling information to live wallpaper, which makes it impossible to implement scrolling wallpapers by using standard methods. This issue is most prominent in the default launcher on Samsung devices — the TouchWiz launcher. Some HTC devices also suffer from this.

All Android live wallpapers have to deal with this somehow, and there is no “correct” solution to this problem. However, *uLiveWallpaper* provides a workaround by emulating scrolling data using touch input, in case a faulty launcher is detected. That way, your live wallpaper will work consistently on all devices, even though on some devices the scrolling experience is expected to be somewhat degraded.

The wallpaper offset data emulation is enabled automatically; you don’t have to do anything manually. However, if you want to tune the emulation settings (like the number of home screens), then you are able to do that. Just drag the following prefab into your scene and edit the properties in the inspector:

Assets\uLiveWallpaper\Resources\DefaultWallpaperOffsetEmulator

Note that emulation is also useful even when launcher reports scrolling information, since it also allows using scrolling during wallpaper preview, which is a nice feature. You don’t have to do anything in order to support that, this is done by default.

Reducing APK Size

Projects generated by *uLiveWallpaper* are regular Unity projects (except the live wallpaper service), so all advices about reducing the APK size for Unity project apply here as well.

Official Unity documentation to get you started:

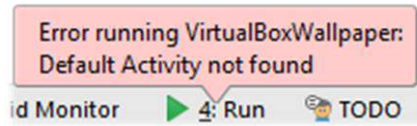
<http://docs.unity3d.com/Manual/ReducingFilesize.html>

Other things that you can try in order to reduce the APK size:

- Use [Sprite Packer](#) to reduce the amount and size of textures.
- Go to “Edit → Project Settings → Graphics” and disable all graphical features you are not using in the project.
- Make separate builds for ARM and x86 CPU architectures. By default, Unity builds a “fat” binary that contains Unity engine for both architectures. This is an easy way to reduce the APK size by around 9 MB.
To change the build CPU architecture, change “Project Settings → Android → Other Settings → Device Filter”.

Troubleshooting

1. After clicking the “Run” button (▶), or using “Run → Run” main menu item, Android Studio shows ‘Error launching <project-name>: Default Activity not found’.



It is typical for live wallpaper projects to not have a default Activity, but Android Studio expects the project to have one. To fix this, select menu “Run → Edit Configurations”. There, in the “General” tab for “Android App”, look for “Launch Options” and select *Nothing* instead of *Default Activity*.

2. After successfully installing the live wallpaper on the device, Android Studio logs ‘Launched on (<your-device-model>)’ error message.

This message is not an actual error and can be safely ignored. It means that your live wallpaper was only installed, but not launched. This happens because “Home Screen Shortcut” is set to “None”, so Android Studio has no idea on how to start the app. You have to manually select your wallpaper in the wallpaper picker dialog, or select something in “Home Screen Shortcut” while creating the project.

3. Opening settings from the preview screen is delayed when running from Android Studio.

This sometimes happens on some devices, but it’s nothing to worry about — settings will open quickly when running a Release build.

4. I’ve updated the Android Studio project, but can’t see the changes.

Use “Build → Clean Project” menu item in Android Studio and try again.

5. My live wallpaper not starting or acting not as expected on my device. What can I do?

First, enable verbose logs. This is done by adding the following tag to end of the <application> tag in AndroidManifest.xml in your Android Studio project:

```
<meta-data android:name="uLiveWallpaper.VerboseLog" android:value="true" />
```

Contact the developer about your problem and attach the full log captured from the device.

Contact

For any questions about this plugin, feel free to contact me at:

Unity forums thread: <https://forum.unity.com/threads/ulivewallpaper-develop-android-live-wallpapers-with-unity.375255/>

E-mail: contact@lostpolygon.com

Skype: [serhii.yolkin](https://www.skype.com/people/serhii.yolkin)

Web-site: <http://lostpolygon.com>

