UniLWP.Droid Documentation

Release 0.0.2 preview.1

FinGameWorks (Haotian Zheng)

Nov 28, 2020

GENERAL

Version 0.0.2 preview.1

Date Nov 28, 2020

Contact justzht+unilwp@gmail.com

CHAPTER

ONE

INTRODUCTION

UniLWP.Droid is a live wallpaper (LWP) solution for Unity to run on Android. It is used in most live wallpaper apps from FinGameWorks, including Metropolis, Vortex, and Diorama.

UniLWP.Droid is built with customization in mind. It works with Unity's default apk build pipeline, but what makes it different from other solutions is an alternative workflow provided to deeply customize the look and bahavior of your apps through external modifications.

1.1 Requirements

- Unity 2019.3 and up (certain features require Unity 2020 or more recent releases)
- Android 7.0 and up (API 24+)
- Android programming experience needed (only if advanced build mode is in use)

1.2 Features

- C# callbacks to build data-driven live wallpapers
  - Unlock state (Locked / Ambient / Screen-on / Unlocked)
  - Dark mode
  - Wallpaper scroll offset, with page count and progress on each page
  - Window insets (to avoid overlapped UI rendering with device notch)
  - Is in wallpaper / preview / activity mode
- Unity Ads support
- Screen saver (DayDream) support
- Non-intrusive integration
  - Unity Cloud Build support
- Customization friendly design
  - Re-building project would still maintain your external modifications made using Android Studio, including java files, xml resources, and gradle dependencies

1.3 Version Comparison

UniLWP.Droid has two variants.

- One is UniLWP.Droid.Free, a free plugin in UPM format.
- Another is UniLWP.Droid.Store, an Asset-Store-listed plugin with more features.

UniLWP.Droid	Free	Store
Unity as Live Wallpaper		
Default Build Pipeline (One-Click Apk)		
Callbacks (Lock State, Scroll Offset, etc)		
Touch Events		
Modular Customization	(You need to do it yourself)	(Editor tools provided)
Advanced Build Workflow		

1.4 Demo

1.5 Legacy Documentation

For an earlier version of UniLWP.Droid documentation, please refer to Google Doc.

CHAPTER

TWO

STRUCTURE

After the import of the store (paid) version, you should have those files in the `Assets/FinGameWorks` folder:

```
.
└── UniLWP
    ├── Droid # Root folder for UniLWP.Droid.
    │   ├── CHANGELOG.md
    │   ├── Editor # Editor scripts, including settings panel, post-build␣
↪scripts, etc.
    │   │   ├── Scripts
    │   │   │   ├── Datas
    │   │   │   │   ├── AndroidManifestXml.cs
    │   │   │   │   ├── AndroidResourceXml.cs
    │   │   │   │   ├── AndroidWallpaperXml.cs
    │   │   │   │   └── BuildPathInfo.cs
    │   │   │   ├── Helpers
    │   │   │   │   ├── FileUtils.cs
    │   │   │   │   ├── ProjectUtils.cs
    │   │   │   │   └── StringUtils.cs
    │   │   │   ├── Settings
    │   │   │   │   ├── AdsProvider.cs
    │   │   │   │   ├── BehaviorProvider.cs
    │   │   │   │   ├── BuildAdvancedProvider.cs
    │   │   │   │   ├── BuildProvider.cs
    │   │   │   │   ├── BuildSimpleProvider.cs
    │   │   │   │   ├── MainProvider.cs
    │   │   │   │   ├── PreferenceProvider.cs
    │   │   │   │   ├── ResourcesProvider.cs
    │   │   │   │   ├── ResourcesScreensaverProvider.cs
    │   │   │   │   ├── ResourcesStringProvider.cs
    │   │   │   │   ├── ResourcesWallpaperProvider.cs
    │   │   │   │   ├── SimulatorProvider.cs
    │   │   │   │   └── UtilsProvider.cs
    │   │   │   └── Windows
    │   │   │       └── MenuActions.cs
    │   │   └── Settings
    │   │       └── BuildPathInfo.asset
    │   ├── JavaSource.zip # Java source files for the aar plugin
    │   ├── Plugins
    │   │   ├── UniLWP-debug.aar # The aar plugin
    │   │   └── unilwp.customize.androidlib # The customization Android␣
↪module that works on top of the aar plugin
```

```
    │   │       ├── AndroidManifest.xml
    │   │       ├── libs
    │   │       ├── project.properties
    │   │       └── res
    │   │           ├── mipmap
    │   │           │   └── unilwp_preview.jpg
    │   │           ├── values
    │   │           │   └── strings.xml
    │   │           ├── xml
    │   │           │   ├── unilwp_wallpaper.xml
    │   │           │   └── unilwp_wallpaper_template.xml
    │   │           ├── xml-v25
    │   │           │   └── unilwp_wallpaper.xml
    │   │           └── xml-v29
    │   │               └── unilwp_wallpaper.xml
    │   ├── README.md
    │   ├── Resources
    │   │   └── LiveWallpaper.Manager.Droid.asset # ScriptableObject␣
↪Singleton for global Live Wallpaper Manager access
    │   └── Scripts # Scripts that works in both editor and runtime
    │       ├── Datas
    │       │   └── Enums.cs
    │       ├── Managers
    │       │   ├── Fingleton.cs
    │       │   ├── LiveWallpaperManagerDroid.cs
    │       │   ├── LiveWallpaperMonoInjecterDroid.cs
    │       │   └── Singleton.cs
    │       └── UniLWP.cs
    ├── Droid.Demo
    │   ├── Materials
    │   │   ├── Particle.mat
    │   │   └── Trail.mat
    │   ├── Scenes
    │   │   └── Main.unity
    │   └── Scripts
    │       └── Controllers
    │           └── UIController.cs
    └── UniLWP.DroidDocumentation.pdf

27 directories, 46 files
```

CHAPTER

THREE

ANDROID PLUGIN

The scope of this page is within `Assets/FinGameWorks/UniLWP/Droid/Plugins` and the source files that contributed to the plugins in that folder.

3.1 Android Library

The Android Library here refers to the `UniLWP-(debug|release).aar` file located in `Assets/FinGameWorks/UniLWP/Droid/Plugins` folder and the source code to compile it.

The Android Library is responsible for major work of converting an ordinary Unity app into a live wallpaper, including lifecycle modification, manifest declaration, event hijacking, and C#-Java communications.

Note: The source code project as a Android Library gradle project, is included as a zip file (`JavaSource.zip`), which you can import using Android Studio to compile your own aar file with own logic.

3.1.1 Initialization

UniLWP is designed to initialize even before the Unity player itself for a total control of the player instance. To achieve this goal, it deploys an early-init technique called ContentProvider, which is mainly done in the `LiveWallpaperInitProvider.java` file.

Tip: For how Content Provider can be called even before `Application.onCreate()`, there is a detailed explaination on the Firebase Dev Blog as the Firebase SDK also deploys such techniques.

In the `onCreate()` method of `LiveWallpaperInitProvider`, UniLWP calls `LiveWallpaperManager.getInstance().Init(Context context)` with the acquired context, which would later be used to create the Unity player.

This way, UniLWP is able to extend its lifespan beyond the Unity player and consequently maintain the control to it no matter the app is launched by system in activity mode or wallpaper mode. The drawback though, is that the Unity player, by default, would be initialized at the beginning, which would let the system took a performance and memory hit early. However, this behavior can be easily tweaked to fit your own need, with the possibility to launch Unity instance when you needed (and destroy it when you don't) through the flag `unilwp.behavior.isolate` and those Java APIs:

```java
public enum LiveWallpaperManager implements IUnityPlayerLifecycleEvents {
  public static LiveWallpaperManager getInstance() {
     return INSTANCE;
```

(continues on next page)

(continued from previous page)

```
    }

    public void LoadUnity(Context context); // Load Unity instance
    public void UnloadUnity(); // Unload Unity
    public void Init(Context applicationContext); // Load UniLWP, and optionally␣
↪UniLWP if unilwp.behavior.isolate is false
    public void DeInit(Context applicationContext); // Unload UniLWP and Unity
  }
```

3.1.2 Display Target

UniLWP uses a single Unity instance for multiple render targets, including the surface view in activities, wallpaper engine in wallpaper services, and even surface view in screen saver (day dream) services. This shared instance ensures that the contents on those render targets are all the same, the only difference is the rendering resolution. However, UniLWP is required to switch render targets to only the currently active one, or the user would only see a static image.

To make this happen, UniLWP's java plugin calls this method of UnityPlayer when a switch is needed:

```
package com.unity3d.player;
public class UnityPlayer extends FrameLayout implements IUnityPlayerLifecycleEvents,␣
↪com.unity3d.player.f {
    public boolean displayChanged(int var1, Surface var2) {
        if (var1 == 0) {
            this.mMainDisplayOverride = var2 != null;
            //...
        }
        return this.updateDisplayInternal(var1, var2);
    }
}
```

The method, `displayChanged(int var1, Surface var2)`, accept two parameters, the first being the display index, while the second is the java Surface object. Index 0 refers to the main display, as you can see in the highlighted code block. If we supply a value that is bigger than 0, Unity would be updated with the new surface being a Display that you can acquire through the C# API Display[] Display.displays.

However, we are not gonna use this multiple display setup. Instead, UniLWP only uses index 0 for display switches to make existing cameras work without additional efforts. Anytime a new surface is visible, UniLWP calls `displayChanged(0, newSurface)` so Unity renders to that surface.

If you are using a customized activity or service, chances are you want your customized surface to be registered into this switch logic. Please refer to *Customize Look and Feel* for more information.

3.1.3 Callbacks

One of UniLWP's unique features is the ability to monitor native events to develop data-driven wallpapers that adapts to dark mode, lock screen, and other device environment changes.

Tip: For how to use callbacks in your C# code, please refer to *Listen To Callbacks*

Callbacks are implemented via the `LiveWallpaperListener` Java interface and `LiveWallpaperListenerManager` Java class.

```java
public enum LiveWallpaperListenerManager {
   // called in C# via AndroidJavaObject
   protected void setEventListener(LiveWallpaperListener eventListener) {
      this.eventListener = eventListener;
      // report initial status to Unity
      // ...
   }
   private LiveWallpaperListener eventListener;
}
```

When the Unity instance is initialized in Java by `LiveWallpaperManager`, the C# part of UniLWP, `LiveWallpaperManagerDroid` will be awaken along with the Unity instance, and before any scene loading, it will call `setEventListener(LiveWallpaperListener eventListener)` with a C# AndroidJavaProxy. Any Android native events would be synced through this interface and dispatched to any listener attached.

It is also suggested that you attach listeners in Monobehavior's `Start()` or `Awake()` function to only register once after the internal setup of `LiveWallpaperManagerDroid`.

3.1.4 Behaviour & Flags

UniLWP comes with a handful options to tweak its native behavior. Certain times you want UniLWP to log in detail, other times you don't. The same applies to Unity initialization, initial variable value, setting button event, etc. All those fields are organized into a single Java class `LiveWallpaperConfig`, and the values would be retrieved from `meta-data` tags in the merged final `AndroidManifest.xml`.

Tip: For how to set behaviour flags in your C# code, please refer to *Listen To Callbacks*

3.2 Android Module

The Android Module here refers to the `unilwp.customize.androidlib` folder located in `Assets/FinGameWorks/UniLWP/Droid/Plugins` folder. It is essentially an Android Library project, with the extension `androidlib` to identify itself as a plugin to Unity.

Note: The `androidlib` extension is an Unity 2020.x feature addition that, if applied, can make your plugin folder work anywhere. In other words, plugins folders with or without this extension will work if placed in the `Assets/Plugins/Android` folder, however, on 2020+ and with this extension, the plugin folder can be moved to anywhere within `Assets`.

In this sense, the Android Module will still largely work on 2019.3 if you move it into the global folder `Assets/Plugins/Android`, but that is pretty in contrast with the 'non-invasive' design philosophy of UniLWP. If you really need it to work on 2019.3, please also modify the plugin path field in setting scripts so the editor UI will point to the correct file, but it won't receive offical support.

3.2.1 Resources

As described before, this folder is in a standard Android Library project layout. Aside from `AndroidManifest.xml`, values are mostly stored in `res/xml` and `res/values` folder. In the final compile stage, these values will be merged into the app and replace any previous or default value.

3.2.2 Goal

Why adding a new folder to the plugin folder when UniLWP already has an AAR file in it? Obviously AAR file cannot be modified easily after it is compiled. While the behavior can be redirected through `meta-data` entries mentioned earlier, all those resources files (names, images) are static-referenced.

An Android Library project that exposes its content to the outside is therefore more suitable to store these values. UniLWP since 0.0.2 comes with additional editor scripts to manage values within the editor so you don't have to use a xml editor or Android Studio.

CHAPTER

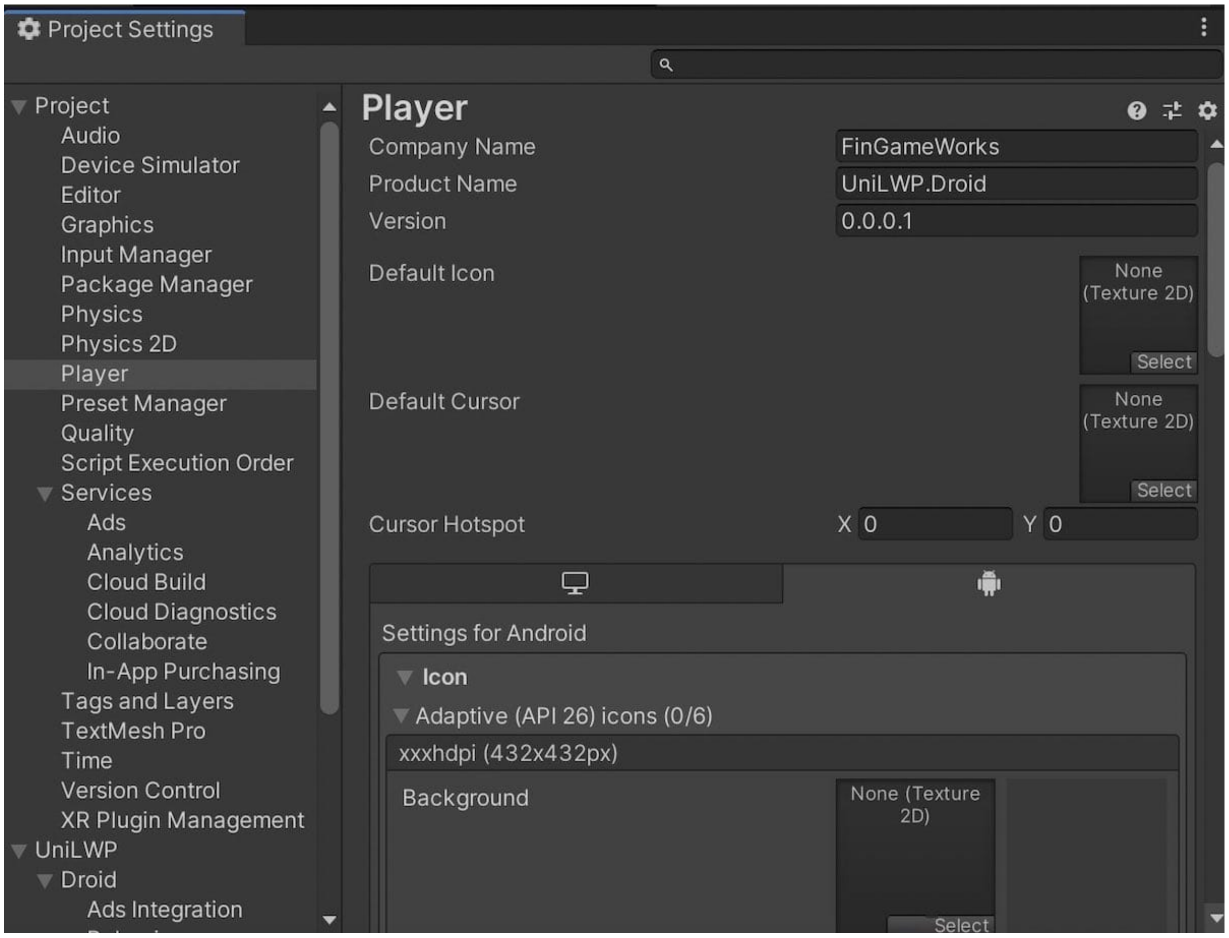
FOUR

CUSTOMIZE LOOK AND FEEL

4.1 Icon, Name, Metadata

Where to find settings

Unity Application Icon



Company Name: FinGameWorks
Product Name: UniLWP.Droid
Version: 0.0.0.1
Default Icon: None (Texture 2D)
Default Cursor: None (Texture 2D)
Cursor Hotspot: X 0 Y 0
Background: None (Texture 2D)

Available Build Type Simple | Advanced

4.1.1 Application Icon

To change the application icon, you can just use the Unity `Project Settings/Player/Icon` settings.

4.1.2 Wallpaper Preview

To change the wallpaper preview icon, replace the jpg file within the `unilwp.customize.androidlib` folder. The exact path should be `res/mipmap/unilwp_preview.jpg`. For more about the `unilwp.customize.androidlib` folder, please refer to *Android Plugin*.

Note: Optionally, since 0.0.2 preview.2, you can utilize the mipmap manage tool under in Project Settings (`UniLWP/Resources/Mipmap`) to upload images into the mipmap folder.

4.2 Advanced Behavior

Available Build Type Simple | Advanced

4.3 Native User Interface

Available Build Type Advanced Only

CHAPTER

FIVE

BUILD PIPELINE

5.1 Simple (Default)

5.2 Advanced (Optional)

CHAPTER SIX

PREPARATION

Before import UniLWP.Droid, make sure that:

- You are using Unity 2019.3 and up.
- You have switched to Android build target.
- You have an Android testing device running Android 7.0 or later.

Then, proceed to *Import & Setup*.

CHAPTER

SEVEN

IMPORT & SETUP

7.1 Free Version

Free Version Import Guide

Where is the package manager

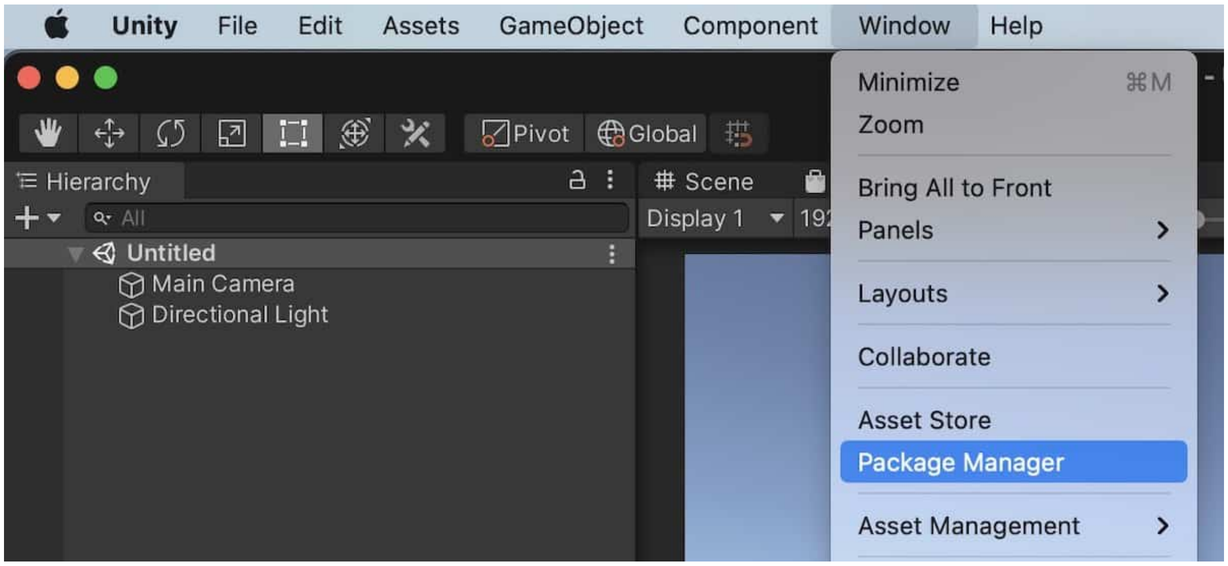


Fig. 1: Package Manager is located under menu `Window`. The menu bar would be at the top of the screen in macOS while at the top of the Unity window in Windows.

How to add UPM packages

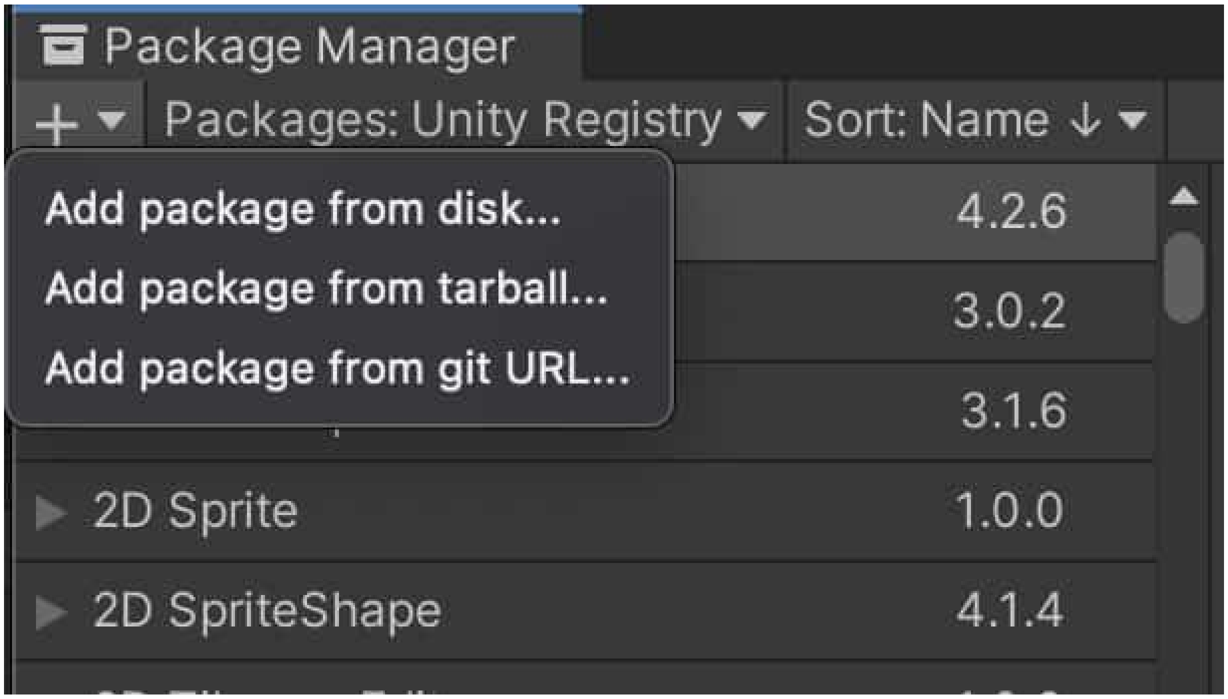
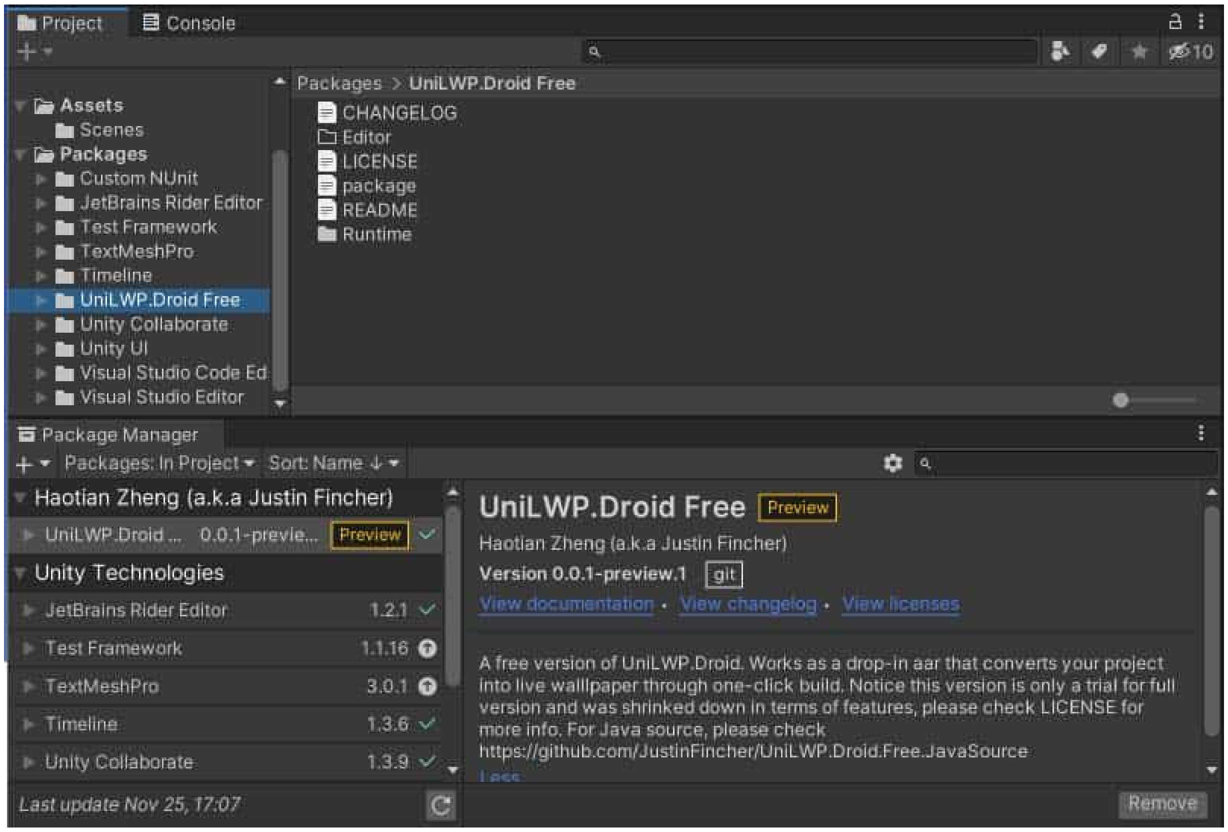


Fig. 2: If you cannot see the dropdown list, drag the title of package manager panel to un-dock the window, and then try again.

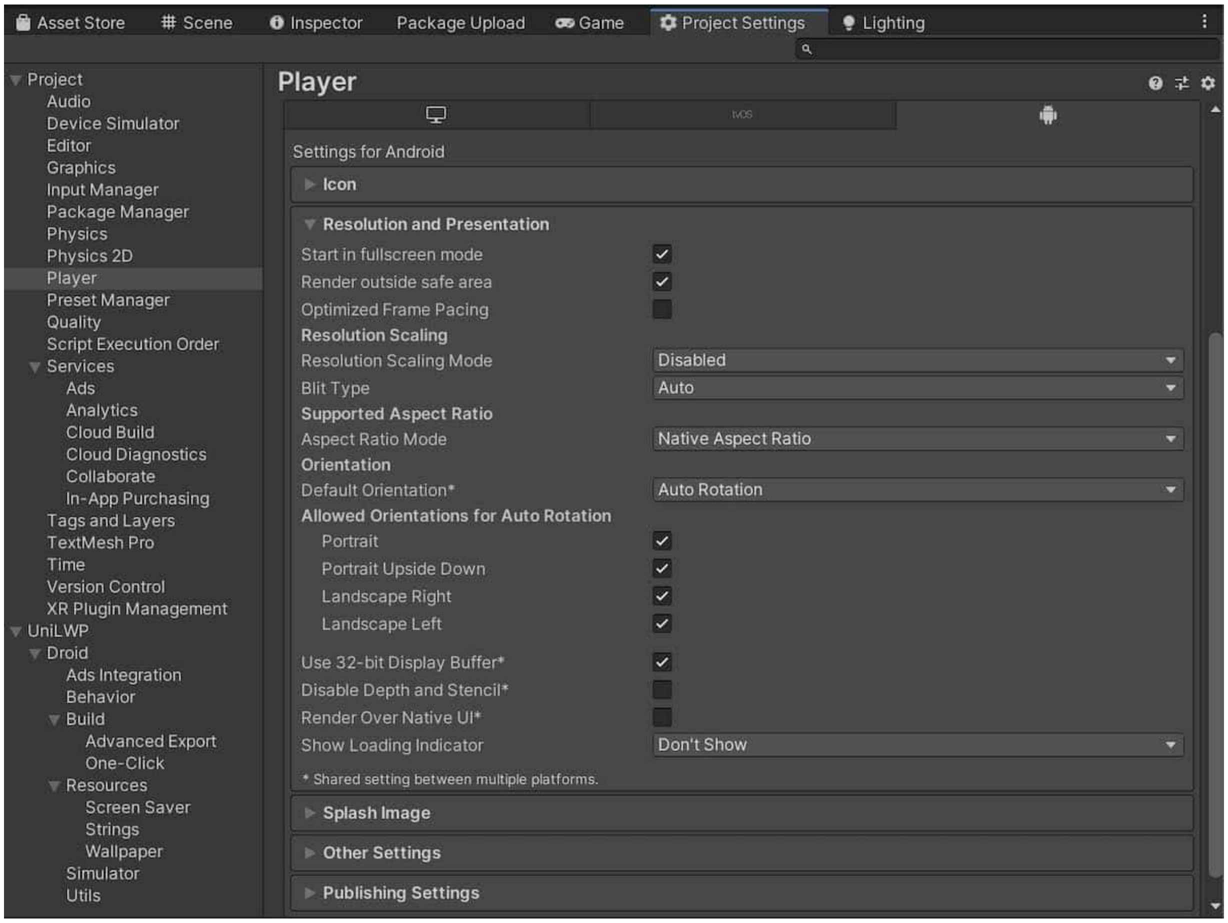
View more info about the package



A free version of UniLWP.Droid. Works as a drop-in aar that converts your project into live walllpaper through one-click build. Notice this version is only a trial for full version and was shrinked down in terms of features, please check LICENSE for more info. For Java source, please check https://github.com/JustinFincher/UniLWP.Droid.Free.JavaSource

Newly added files are groupped and shown within the `Packages` section of the project panel

Ideal project settings



Player

Settings for Android

Icon

Resolution and Presentation

Start in fullscreen mode	✓
Render outside safe area	✓
Optimized Frame Pacing	
Resolution Scaling	
Resolution Scaling Mode	Disabled
Blit Type	Auto
Supported Aspect Ratio	
Aspect Ratio Mode	Native Aspect Ratio
Orientation	
Default Orientation*	Auto Rotation
Allowed Orientations for Auto Rotation	
Portrait	✓
Portrait Upside Down	✓
Landscape Right	✓
Landscape Left	✓
Use 32-bit Display Buffer*	✓
Disable Depth and Stencil*	
Render Over Native UI*	
Show Loading Indicator	Don't Show

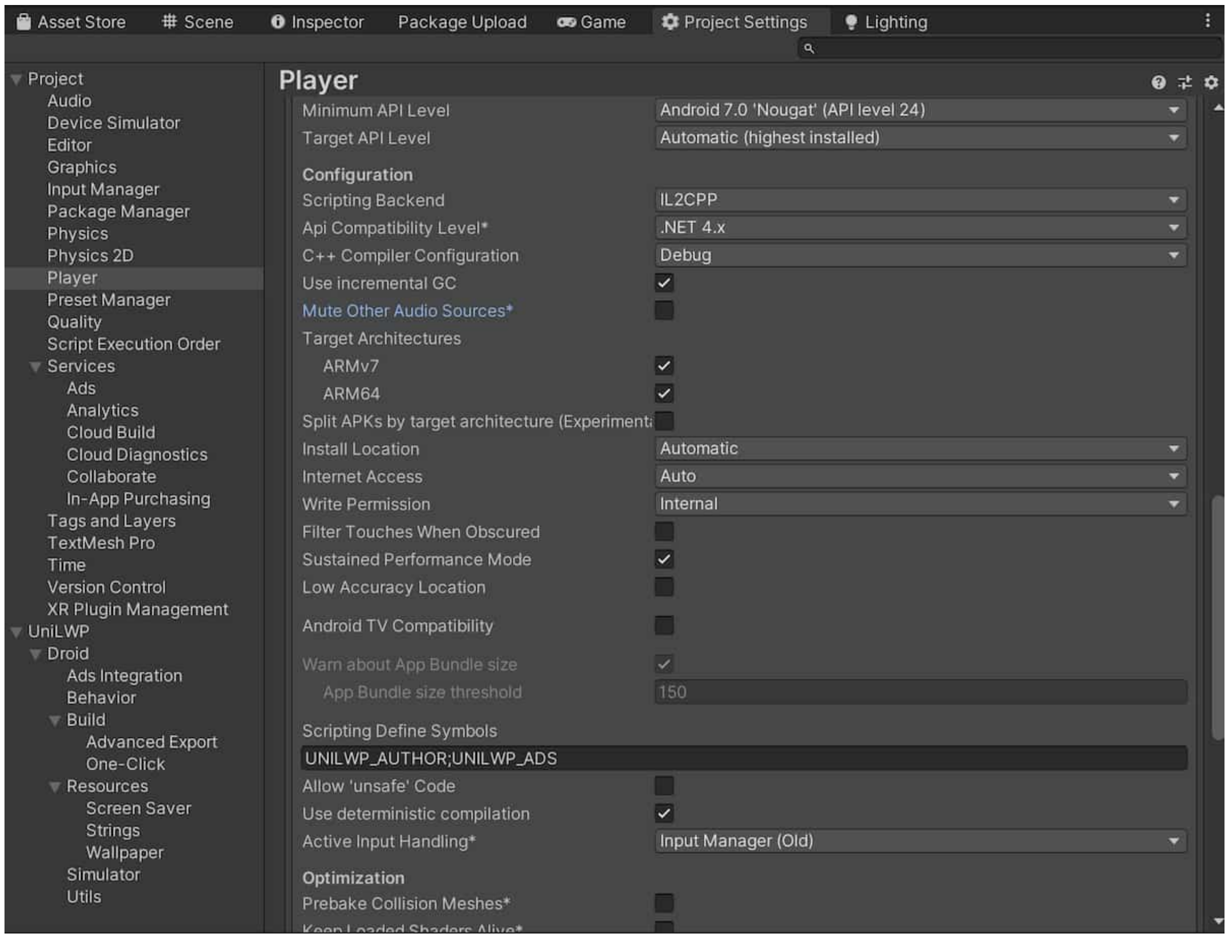
* Shared setting between multiple platforms.

Splash Image

Other Settings

Publishing Settings

`Project/Player/Resolution and Presentation`



Player

Minimum API Level	Android 7.0 'Nougat' (API level 24)
Target API Level	Automatic (highest installed)
Configuration	
Scripting Backend	IL2CPP
Api Compatibility Level*	.NET 4.x
C++ Compiler Configuration	Debug
Use incremental GC	✓
Mute Other Audio Sources*	
Target Architectures	
ARMv7	✓
ARM64	✓
Split APKs by target architecture (Experiment	
Install Location	Automatic
Internet Access	Auto
Write Permission	Internal
Filter Touches When Obscured	
Sustained Performance Mode	✓
Low Accuracy Location	
Android TV Compatibility	
Warn about App Bundle size	✓
App Bundle size threshold	150
Scripting Define Symbols	UNILWP_AUTHOR;UNILWP_ADS
Allow 'unsafe' Code	
Use deterministic compilation	✓
Active Input Handling*	Input Manager (Old)
Optimization	
Prebake Collision Meshes*	

Project/Player/Other Settings

7.1.1 Import

The free version, UniLWP.Droid.Free, is hosted on GitHub as an UPM package.

- To import, first open package manager via menu `Window/Package Manager`.
- Then, at the top-left of the newly opened package manager window, you will find a plus sign. Click on it and select `Add package from git URL...`
- Paste the following URL into the field and press enter:

```
https://github.com/JustinFincher/UniLWP.Droid.Package.Free.git
```

- Unity should download and load UniLWP.Droid as an UPM-formatted dependency.

Or, if you are familiar with `package.json`, you are free to do paste this line into your json file:

```
"dependencies":
{
  "com.justzht.unilwp.droid.free": "https://github.com/JustinFincher/UniLWP.Droid.Package.Free.git" // this line
}
```

7.1.2 Setup

Toggle `Project Settings` panel via menu path `Edit/Project Settings...`

- Go to `Project/Player` and adjust certain items:
  - In `Player/Resolution and Presentation`, make sure that both `Optimized Frame Pacing` and `Render Over Native UI` are unchecked.
  - In `Player/Other Settings`, make sure that both `Mute Other Audio Sources` and `Filter Touches When Obscured` are unchecked, the `Minimal API Level` is `Android 7.0 Nougat (API Level 24)`. You might also want to change `Graphics APIs` to `OpenGLES3` only, but that is optional.
- Go to `Project/Audio` and adjust certain items:
  - Check `Disable Unity Audio` if you don't want your wallpaper to play sounds.

7.1.3 Build

It is the same as the default Unity build pipeline, that you only need to trigger a build through the default `File/Build And Run` menu path. The plugin will be packed into the final apk file and be initialized as soon as possible to handle Unity lifecycles.

7.2 Store Version

Note: As Unity 2020.1 deprecated the Unity Asset Store web browser panel in favour of the package manager, you need to use package manager to import UniLWP.Droid on 2020.1+

Paid Version Import Guide

Where is the package manager

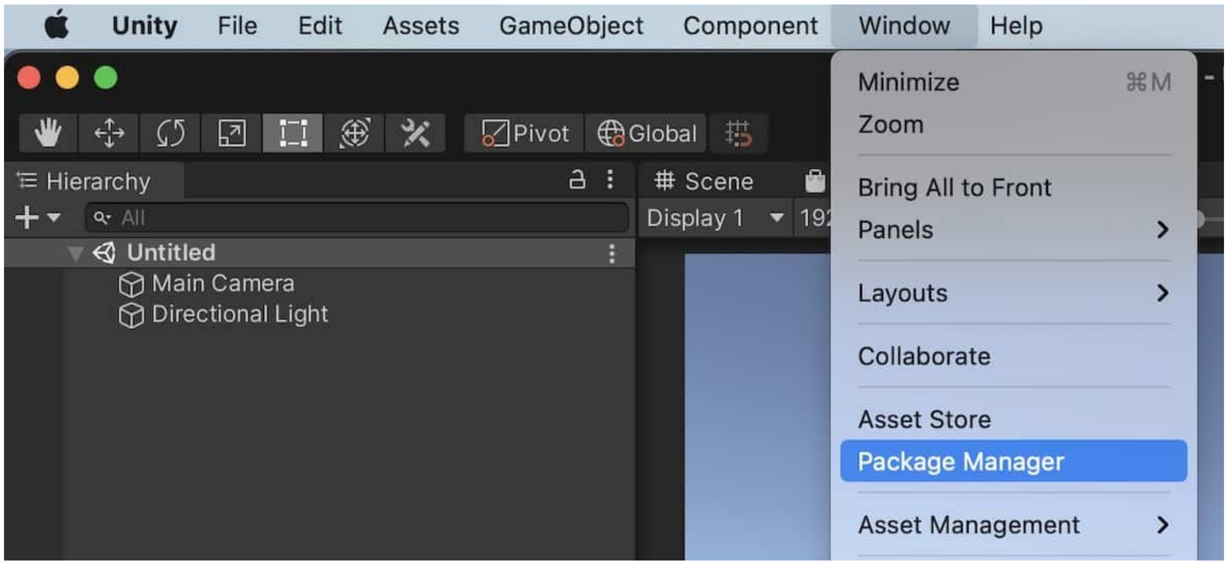


Fig. 3: Package Manager is located under menu `Window`. The menu bar would be at the top of the screen in macOS while at the top of the Unity window in Windows.

How to find asset store packages you bought before in package manager

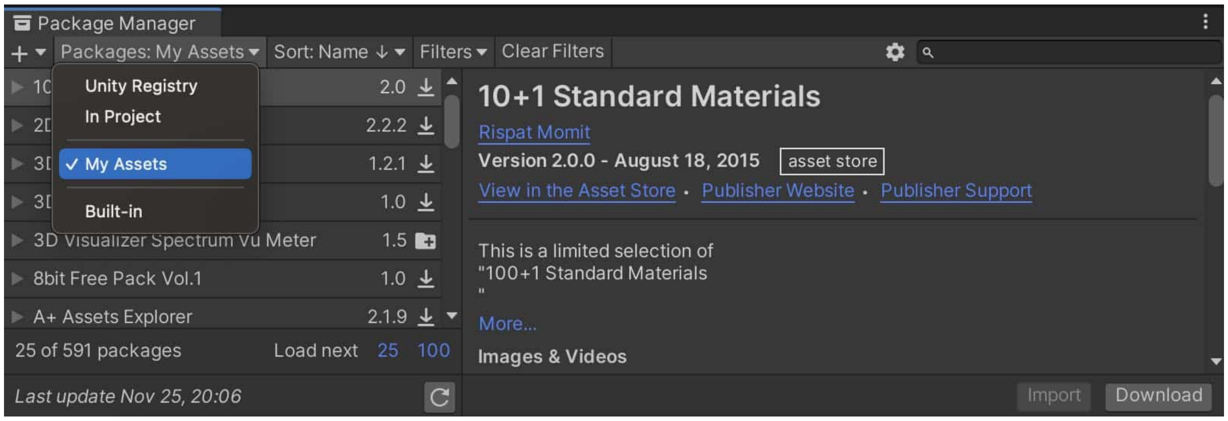
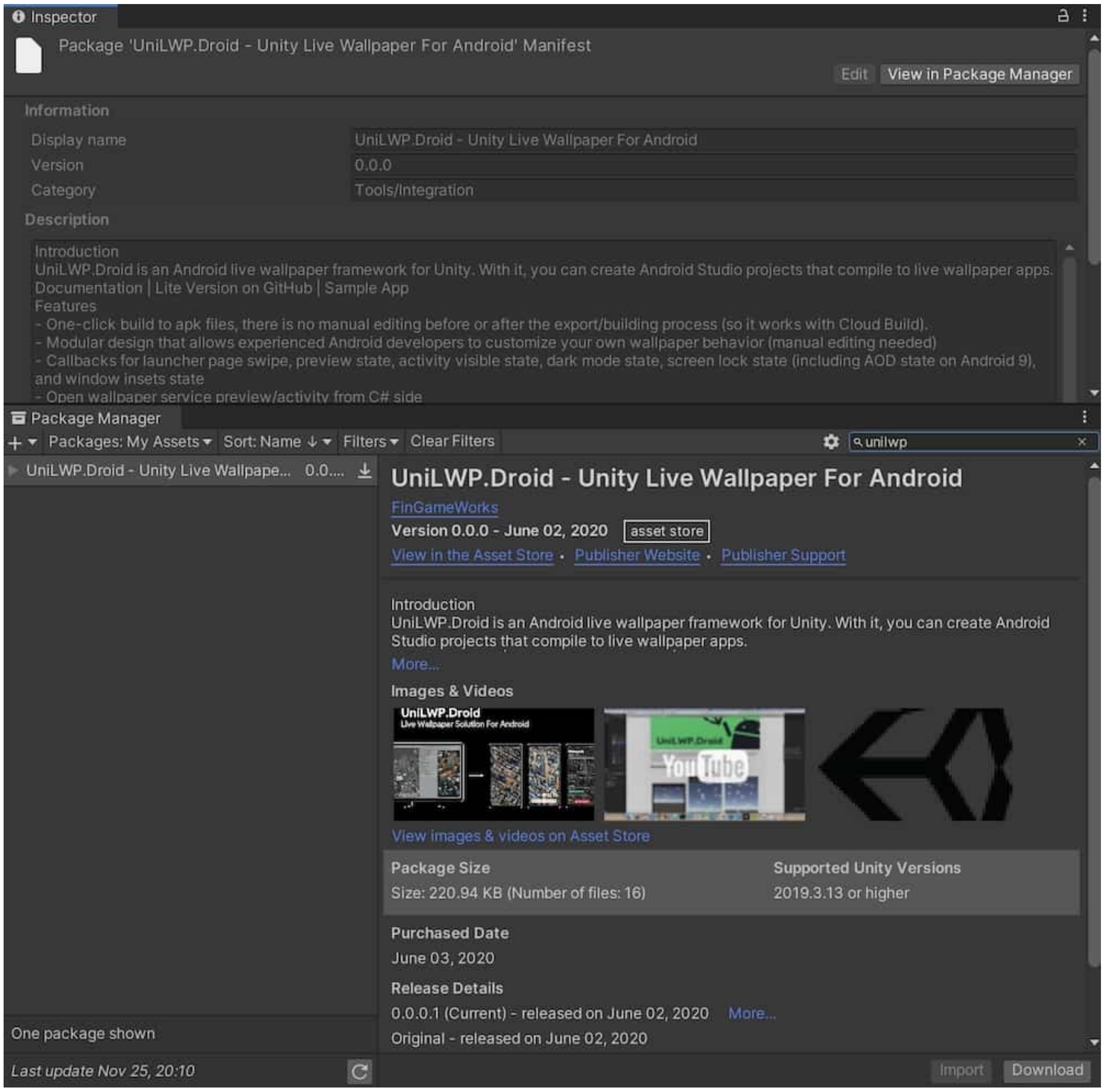


Fig. 4: Switch package scope at the top-left menu, then use the search field at the top-right to search your assets.

How to import asset store packages in package manager



Package 'UniLWP.Droid - Unity Live Wallpaper For Android' Manifest

Information

Display name	UniLWP.Droid - Unity Live Wallpaper For Android
Version	0.0.0
Category	Tools/Integration

Description

Introduction
UniLWP.Droid is an Android live wallpaper framework for Unity. With it, you can create Android Studio projects that compile to live wallpaper apps.
Documentation | Lite Version on GitHub | Sample App
Features
- One-click build to apk files, there is no manual editing before or after the export/building process (so it works with Cloud Build).
- Modular design that allows experienced Android developers to customize your own wallpaper behavior (manual editing needed)
- Callbacks for launcher page swipe, preview state, activity visible state, dark mode state, screen lock state (including AOD state on Android 9), and window insets state
- Open wallpaper service preview/activity from C# side

unilwp

UniLWP.Droid - Unity Live Wallpape... 0.0....

One package shown

Last update Nov 25, 20:10

UniLWP.Droid - Unity Live Wallpaper For Android

FinGameWorks

Version 0.0.0 - June 02, 2020 asset store

View in the Asset Store • Publisher Website • Publisher Support

Introduction
UniLWP.Droid is an Android live wallpaper framework for Unity. With it, you can create Android Studio projects that compile to live wallpaper apps.
More...

Images & Videos

View images & videos on Asset Store

Package Size
Size: 220.94 KB (Number of files: 16)

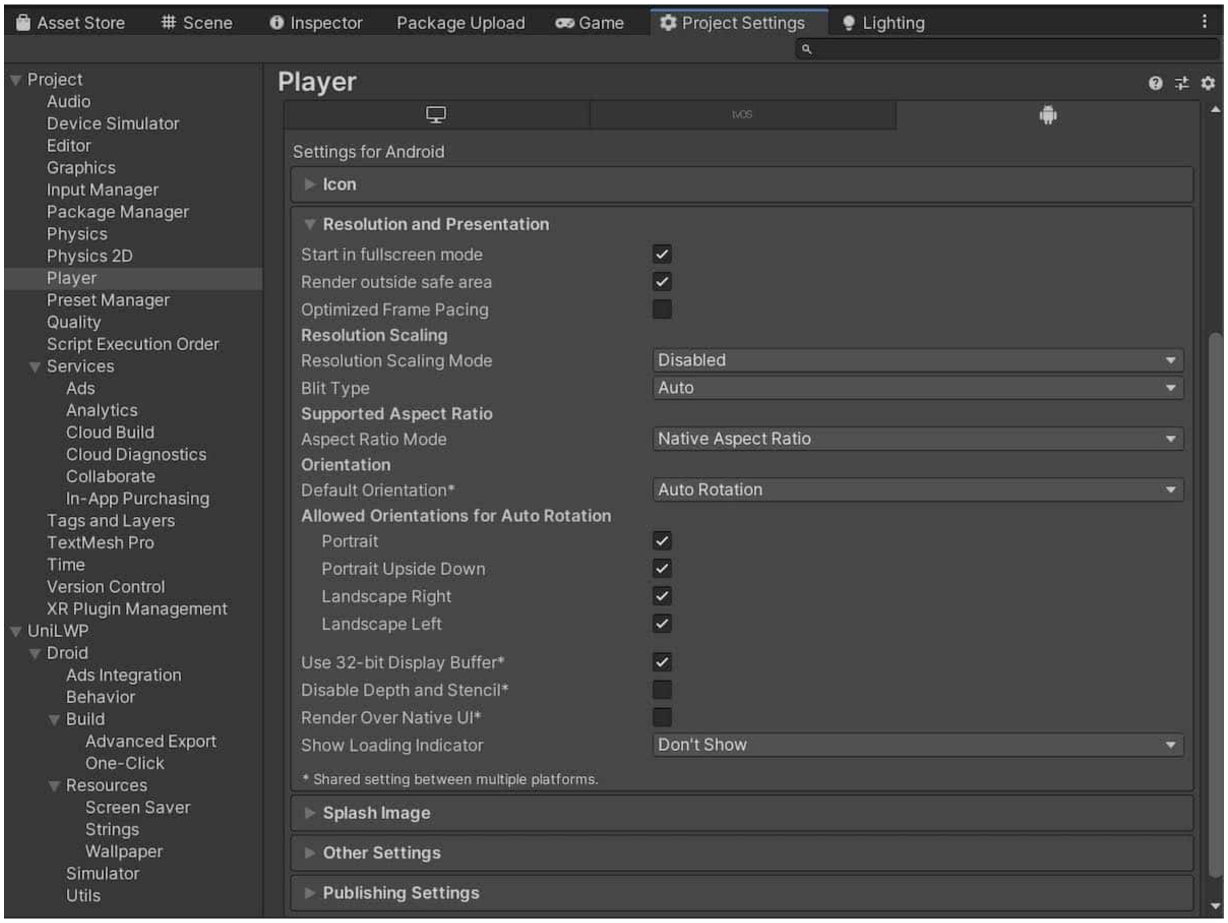
Supported Unity Versions
2019.3.13 or higher

Purchased Date
June 03, 2020

Release Details
0.0.0.1 (Current) - released on June 02, 2020 More...
Original - released on June 02, 2020

Fig. 5: Choose the asset at the left sidebar, then press `download` and then `import` buttons at the bottom-right.

Ideal project settings



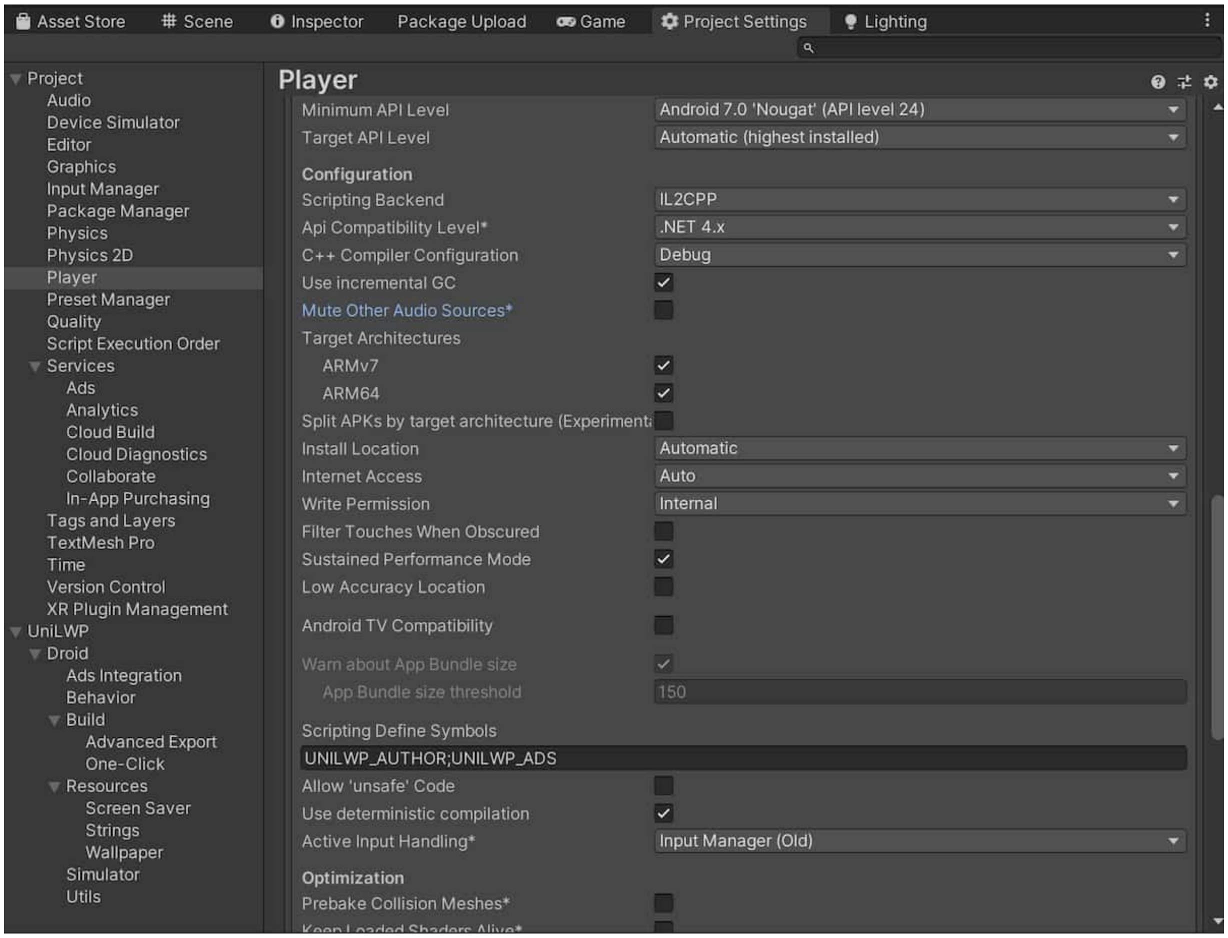
Settings for Android

Resolution and Presentation

Start in fullscreen mode	✓
Render outside safe area	✓
Optimized Frame Pacing	
Resolution Scaling	
Resolution Scaling Mode	Disabled
Blit Type	Auto
Supported Aspect Ratio	
Aspect Ratio Mode	Native Aspect Ratio
Orientation	
Default Orientation*	Auto Rotation
Allowed Orientations for Auto Rotation	
Portrait	✓
Portrait Upside Down	✓
Landscape Right	✓
Landscape Left	✓
Use 32-bit Display Buffer*	✓
Disable Depth and Stencil*	
Render Over Native UI*	
Show Loading Indicator	Don't Show

* Shared setting between multiple platforms.

`Project/Player/Resolution and Presentation`



Player

Minimum API Level	Android 7.0 'Nougat' (API level 24)
Target API Level	Automatic (highest installed)
Configuration	
Scripting Backend	IL2CPP
Api Compatibility Level*	.NET 4.x
C++ Compiler Configuration	Debug
Use incremental GC	✓
Mute Other Audio Sources*	
Target Architectures	
ARMv7	✓
ARM64	✓
Split APKs by target architecture (Experiment	
Install Location	Automatic
Internet Access	Auto
Write Permission	Internal
Filter Touches When Obscured	
Sustained Performance Mode	✓
Low Accuracy Location	
Android TV Compatibility	
Warn about App Bundle size	✓
App Bundle size threshold	150
Scripting Define Symbols	UNILWP_AUTHOR;UNILWP_ADS
Allow 'unsafe' Code	
Use deterministic compilation	✓
Active Input Handling*	Input Manager (Old)
Optimization	
Prebake Collision Meshes*	

Project/Player/Other Settings

The paid version, UniLWP.Droid.Store, is distributed on Unity Asset Store as a `.unitypackage` file.

7.2.1 Import On 2020.1 +

- Open package manager via menu `Window/Package Manager`.
- Switch package scope to `Packages: My Assets` and search `UniLWP`
- Choose `UniLWP.Droid - Unity Live Wallpaper For Android`, then download and import

7.2.2 Import On 2019.3 +

- Open asset store panel via menu `Window/Asset Store`.
- Search for `UniLWP` in the newly opened web version of asset store.
- Choose `UniLWP.Droid - Unity Live Wallpaper For Android`, then download and import

7.2.3 Setup

Toggle `Project Settings` panel via menu path `Edit/Project Settings...`

- Go to `Project/Player` and adjust certain items:
  - In `Player/Resolution and Presentation`, make sure that both `Optimized Frame Pacing` and `Render Over Native UI` are unchecked.
  - In `Player/Other Settings`, make sure that both `Mute Other Audio Sources` and `Filter Touches When Obscured` are unchecked, the `Minimal API Level` is `Android 7.0 Nougat (API Level 24)`. You might also want to change `Graphics APIs` to `OpenGLES3` only, but that is optional.
- Go to `Project/Audio` and adjust certain items:
  - Check `Disable Unity Audio` if you don't want your wallpaper to play sounds.
- Go to `UniLWP` and change settings as you like. Please refer to *Customize Look and Feel* for options.

7.2.4 Build

Please refer to *Export*.

CHAPTER

EIGHT

LISTEN TO CALLBACKS

UniLWP.Droid provides callbacks for developers to register in C# so they can respond to Android native events.

8.1 Callback Types And Registration

Currently, all available callbacks are declared within `LiveWallpaperManagerDroid`, a `ScriptableObject` singleton that you can access via a global variable.

`LiveWallpaperManagerDroid` would be initialized prior to scene loading, so you can safely call `LiveWallpaperManagerDroid.Instance` in `Awake()` or `Start()` without worrying null reference exceptions.

If you want to acquire the initial state of callback values, `LiveWallpaperManagerDroid` also has a set of static variables with corrsponding names to the callback methods as initial values.

Below is an example of how to acquire initial value of the screen status and also listen for future changes.

```
using FinGameWorks.UniLWP.Droid.Scripts.Managers;

public class Demo : MonoBehaviour
{
    private void Awake()
    {
        // acquire the initial screen status
        Enums.ScreenStatus screenStatus = LiveWallpaperManagerDroid.screenStatus;
        Debug.Log("ScreenStatus " + screenStatus);

        LiveWallpaperManagerDroid.Instance.screenDisplayStatusUpdated += status =>
        {
            // triggered every time screen status has change
            Debug.Log("ScreenStatus " + status);
        };
    }
}
```

8.1.1 Insets

Insets refer to android.view.WindowInsets, a set of value describing the padding of android window where you should avoid drawing to. This is especially useful when you are trying to run Unity UI on Android devices with notches.

Called when

When the app has entered an Actvitiy or a Wallpaper Service.

Value

Padding values in pixel format. Please use in conjuntion with Screen.width to calucate the percentage if you are in a scaled UI space.

Listing 1: Declaration

```
public delegate void OnInsetsUpdatedDelegate(int left, int top, int right,␣
↪int bottom);
public OnInsetsUpdatedDelegate insetsUpdated;
public static Vector4 insets;
```

Listing 2: Example

```
LiveWallpaperManagerDroid.Instance.insetsUpdated += (left, top, right,␣
↪bottom) =>
{
};
```

8.1.2 Offsets

Offsets refer to the distance the users have scrolled on their Android home (launcher). It is derived from the WallpaperService.Engine#onOffsetsChanged method with slight modifications.

Called when

When the user is swiping across pages on Android launchers.

Value

- float xOffset: Wallpaper horizontal scoll progress in a 0-1 scale.
- float yOffset: Wallpaper vertical scoll progress in 0-1 scale. Since there isn't much support of vertical launcher pages in Android apps, this value is mostly reportede as 0.
- float xOffsetStep: The progress a horizontal full-page scroll would take in a 0-1 scale. Consequently, total horizontal page count can be calucated by 1 divided by this value `(int)(1.0/xOffsetStep)`.
- float yOffsetStep: The progress a vertical full-page scroll would take in a 0-1 scale.

- bool simulated: Since certain stock launchers and ROMs do not follow the described behavior in Android documentation (specifically, Samsung's OneUI), the value UniLWP acquired from those devices are always 0 or 0.5, resulting in no way to know the total launcher pages and progress. To work around this limitation, UniLWP is designed to deploy a gesture recoginizer to manually calucate the estimated progress, and simulated field would be true in this case.

Listing 3: Declaration

```
public delegate void OnWallpaperOffsetsUpdatedDelegate(float xOffset, float␣
↪yOffset, float xOffsetStep, float yOffsetStep, bool simulated);
public OnWallpaperOffsetsUpdatedDelegate wallpaperOffsetsUpdated;
public static Vector4 offset;
public static bool offsetSimulated;
```

Listing 4: Example

```
LiveWallpaperManagerDroid.Instance.wallpaperOffsetsUpdated += (xOffset,␣
↪yOffset, xStep, yStep, simulated) =>
{
    int xPageCount = xStep == 0 ? 0 : (int) Math.Round(1.0 / xStep);
    float xPageProgress = xPageCount * xOffset;
};
```

8.1.3 Dark Mode

Dark mode

8.1.4 Screen Display Status

Screen display status refers to the lock state of a phone. You can utlize this value to perform certian animations when the user lights up or unlocks the phone.

Note: For Android 9.0, this callback also include an always on display (AOD) value if you put `androidprv:supportAmbientMode="true"` into `wallpaper.xml`. However, since Android 10, this attribute is protected by a permission `android.permission.AMBIENT_WALLPAPER`, which is a system only permission that you normally cannot request except you are compling the ROM youself (i.e. you are also one of Android OEM companies or custom ROM makers)

Called when

When the user has turn on the screen in lock state / unlock the phone / lock the phone / leave the phone into always on display mode

Value

Enums.ScreenStatus, where:

- LockedAndOff = 0
- LockedAndAOD = 1
- LockedAndOn = 2
- Unlocked = 3

Listing 5: Declaration

```
public delegate void OnScreenDisplayStatusUpdatedDelegate(Enums.ScreenStatus␣
↪screenStatus);
public OnScreenDisplayStatusUpdatedDelegate screenDisplayStatusUpdated;
public static Enums.ScreenStatus screenStatus;
```

Listing 6: Example

```
LiveWallpaperManagerDroid.Instance.screenDisplayStatusUpdated += status =>
{
};
```

8.1.5 In Activity

This callback reflects if the Unity instance is currently displaying in an activity.

Note: Notice that, this callback only works in the scope of UniLWP's own provided activities. If you are writing a customized activity and also want the UniLWP to receive this event in the C# side, please register your activity (refer to the `Trigger Callbacks In Your Own Implementation` section)

8.1.6 In Service

This callback reflects if the Unity instance is currently displaying in wallpaper mode.

8.2 Trigger Callbacks In Your Own Implementation

CHAPTER

NINE

BEHAVIOR

CHAPTER

TEN

EXPORT

10.1 One-Click

10.2 External Modification

CHAPTER

ELEVEN

ADS

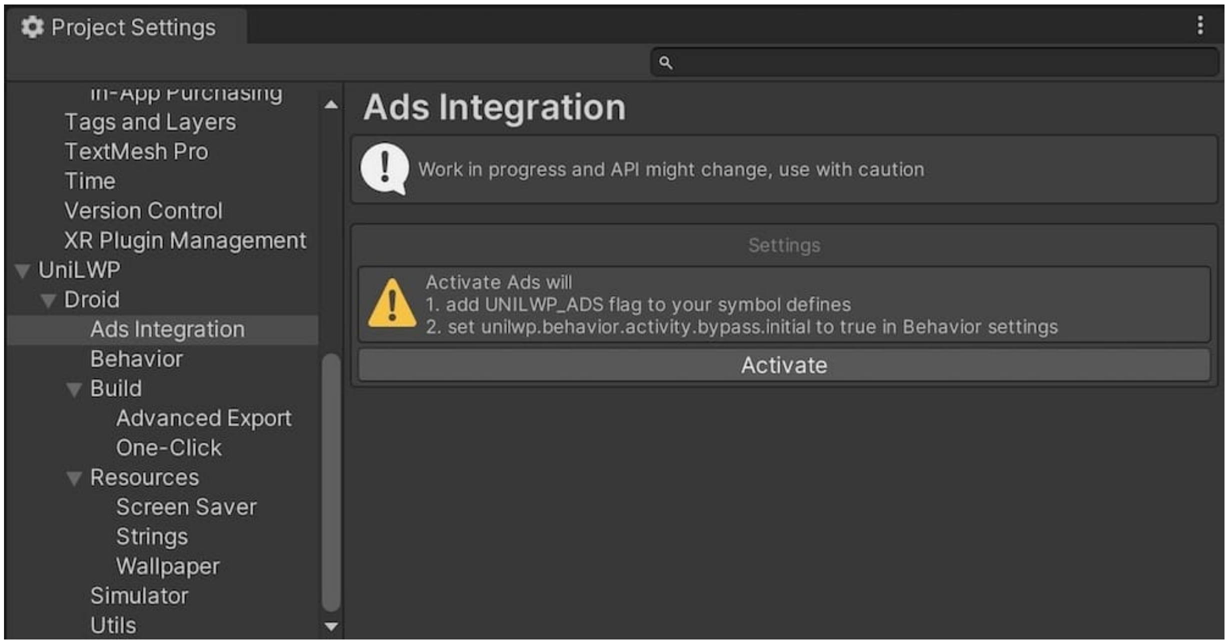
Warning: Unity Ads support is experimental but it should work on 0.0.2 and up.

Ads support has two limitations for now:

- The `Initial Activity Check Bypass` flag should be enabled (The editor script will do this automatically)
- You can only show an ad in activity, not wallpaper.

Ads Settings

Before enabling Unity Ads support

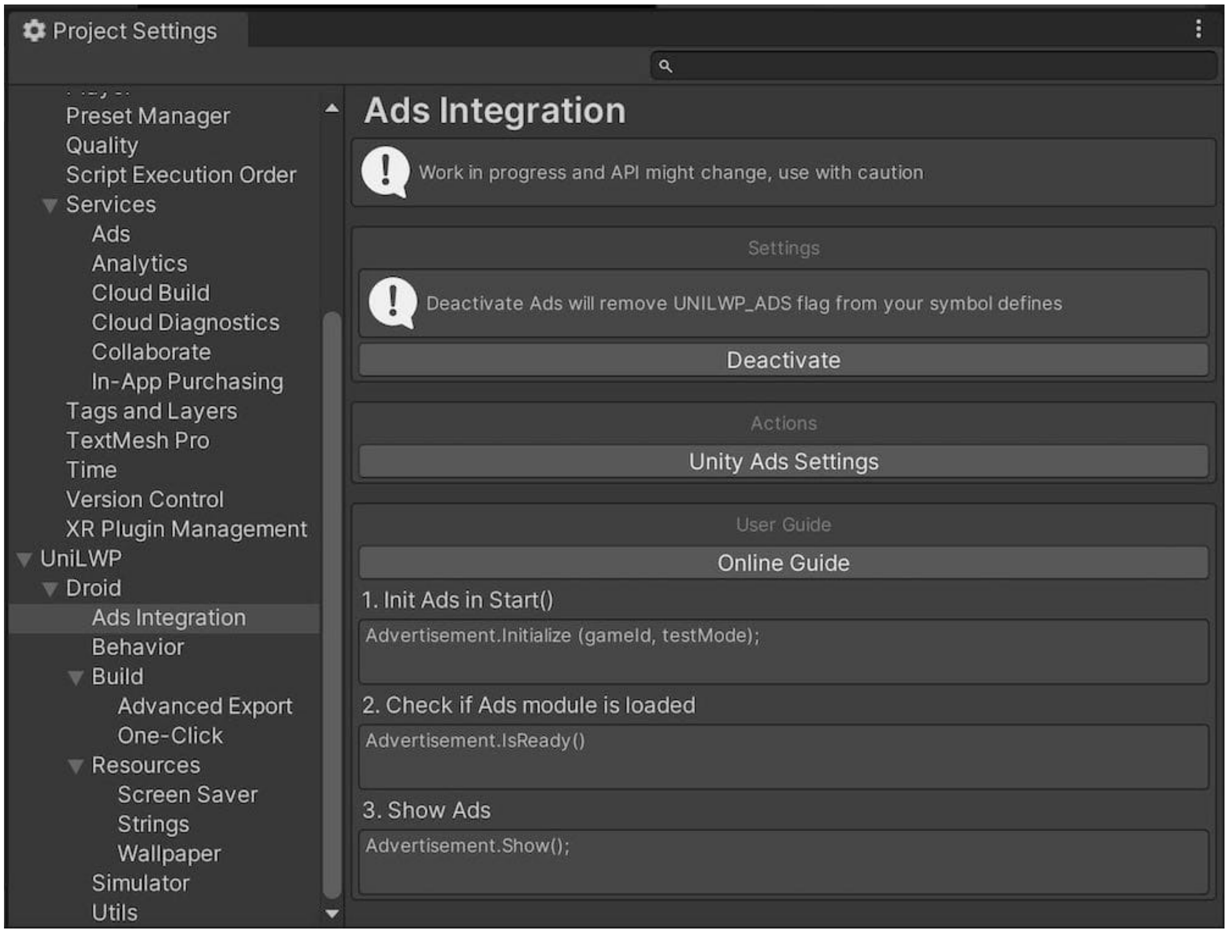


Ads Integration

Work in progress and API might change, use with caution

Activate Ads will
1. add UNILWP_ADS flag to your symbol defines
2. set unilwp.behavior.activity.bypass.initial to true in Behavior settings

After enabling Unity Ads support



Ads Integration

Work in progress and API might change, use with caution

Settings

Deactivate Ads will remove UNILWP_ADS flag from your symbol defines

1. Init Ads in Start()

```
Advertisement.Initialize (gameId, testMode);
```

2. Check if Ads module is loaded

```
Advertisement.IsReady()
```

3. Show Ads

```
Advertisement.Show();
```

To enable it, you need to:

- Go to the UniLWP Ad setting panel at `Project Settings/UniLWP/Droid/Ads Integration`
- Click 'Activate' button. This step will:
  - Add `UNILWP_ADS` to your Android platform define symbols to enable certain blocks of code bound by `#ifdef` conditional compile flags
  - Set `unilwp.behavior.activity.bypass.initial` to true so that Unity Ads native library could acquire a proper context via `UnityPlayer.currentActivity` at startup.
- Click 'Unity Ads Settings' button to setup your ads and optionally add ads dependency.

CHAPTER

TWELVE

PREFERENCE

CHAPTER THIRTEEN

SIMULATOR

CHAPTER

FOURTEEN

CHANGELOG

14.1 0.0.2 (preview.2)

Date Nov 29, 2020

Comment This is a preview release of 0.0.2, with many break changes both in C# and Java modules. Please delete UniLWP folder before upgrading to 0.0.2.

Added

- Added mipmap settings panel in the `Project Settings` editor window.

Fixes

N/A

Removed

N/A

14.2 0.0.2 (preview.1)

Date Nov 24, 2020

Comment This is a preview release of 0.0.2, with many break changes both in C# and Java modules. Please delete UniLWP folder before upgrading to 0.0.2.

Added

- Added UniLWP.Droid settings panels in the `Project Settings` editor window.
- Unity Ads integration (experimental).
  - To use Unity Ads (currently only within activities), you need to follow guidelines in `Project Settings/UniLWP/Droid/Ads Integration`.

Fixes

- Menu items within `FinGameWorks/UniLWP/Droid` are re-arranged into groups.
- Fixed a crash when Unity UI input field is activated.

Removed

N/A

14.3 0.0.1

Date Jun 2, 2020

Comment Initial Release