How to adapt your AirConsole Unity game for AndroidTV

An AirConsole app for AndroidTV is available on the Google Play store (https://play.google.com/store/apps/details?id=com.airconsole.androidtv&hl=en). Unfortunately, the AndroidTV cannot run the WebGL versions of Unity games that we have in the AirConsole store. In order to run your Unity-made AirConsole games on Android TV, you need to create a native Android Build instead of the regular WebGL build. Some adjustments may have to be made to your game project, as detailed in this guide.

When you switch platforms to Android and export your game to an AndroidTV, the AirConsole Plugin creates a webview that shows AirConsole content within your AndroidTV App.

Following this guide will give you an APK file which can then be uploaded to the Google Play Store. If you are thinking about porting your AirConsole game to AndroidTV, please contact us beforehand.

Tools

- Make sure your Unity Version is 5.4.1f1 or higher
- Make sure your AirConsole Plugin version is 1.4 or higher
- Download the Android SDK and JDK from within the Unity Editor
- Go to **Unity → Preferences → External Tools** and download SDK and JDK using the download buttons available.

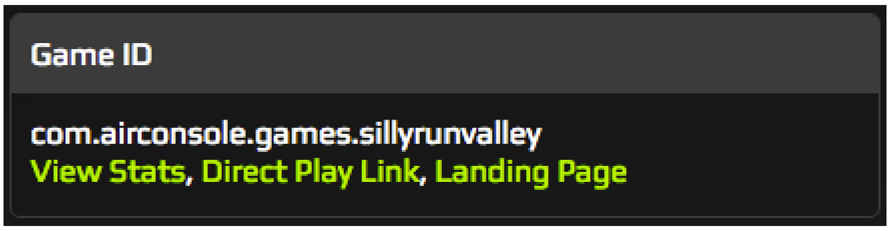
Platform

Before you start working on your AndroidTV Port, it’s important that you have uploaded a .zip of your WebGL build to airconsole.com/developers (http://airconsole.com/developers) and create the Game ID which we will need for the next steps. Read the "Publish your game" guide which explains how to upload your game.

In **File → Build Settings**, choose ‘Android’. Reimporting assets for the platform might take a moment.

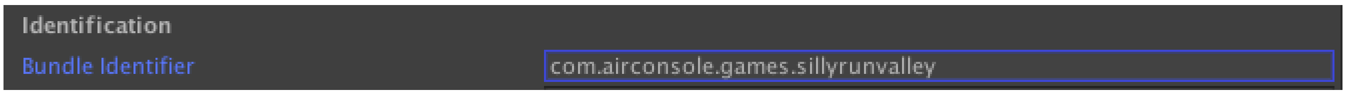
BundleID

Go to **Edit** → **Project Settings** → **Player**.
In the Other Settings tab, you'll find the parameter Bundle Identifier. For your AndroidTV build to work, this needs to be identical with the Game ID that you've given your game on airconsole.com/developers (http://airconsole.com/developers).



Game ID

com.airconsole.games.sillyrunvalley
View Stats, Direct Play Link, Landing Page

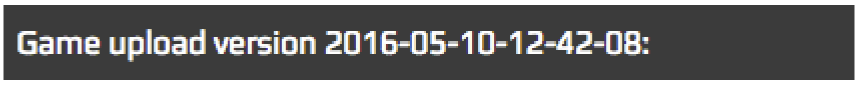


Identification
Bundle Identifier: com.airconsole.games.sillyrunvalley

Screen and Controller Data

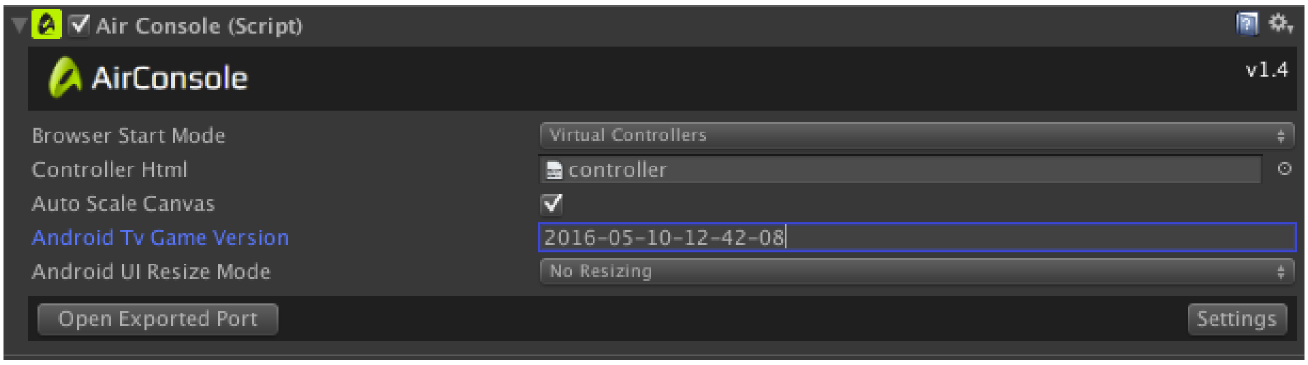
In order for your exported game to be able to access the controller and all necessary JavaScript and image files, you need to:

1. Have a WebGL build with all needed screen and controller Data uploaded to airconsole.com/developers (http://airconsole.com/developers)
2. Copy the Game upload version number of your game.



Game upload version 2016-05-10-12-42-08:

3. Paste the number into the **Android TV Game Version** field on the AirConsole GameObject in your Scene.



Air Console (Script)
AirConsole v1.4
Browser Start Mode: Virtual Controllers
Controller Html: controller
Auto Scale Canvas: ✓
Android Tv Game Version: 2016-05-10-12-42-08
Android UI Resize Mode: No Resizing

4. Save your scene and create an Android build. The **gameName.apk** file you get can be deployed to your AndroidTV device

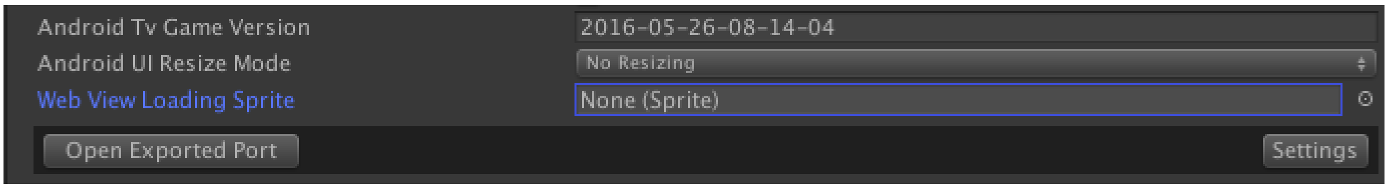
OnReady / OnGameEnd

Users can return to the AirConsole store by selecting 'Change Game' on their phone without leaving your Android App. Because of that, you need to mute all sound in your game in the OnGameEnd event, like you would in the OnAdShow event. Pause / stop as much of your game as you can to make sure no unnecessary computation is happening behind the web view after users have chosen to 'Change Game'.

Similarly, only start playing your game's sound/music in the OnReady event, not on Start / Awake / Scene Loaded.

Loading Screen Sprite

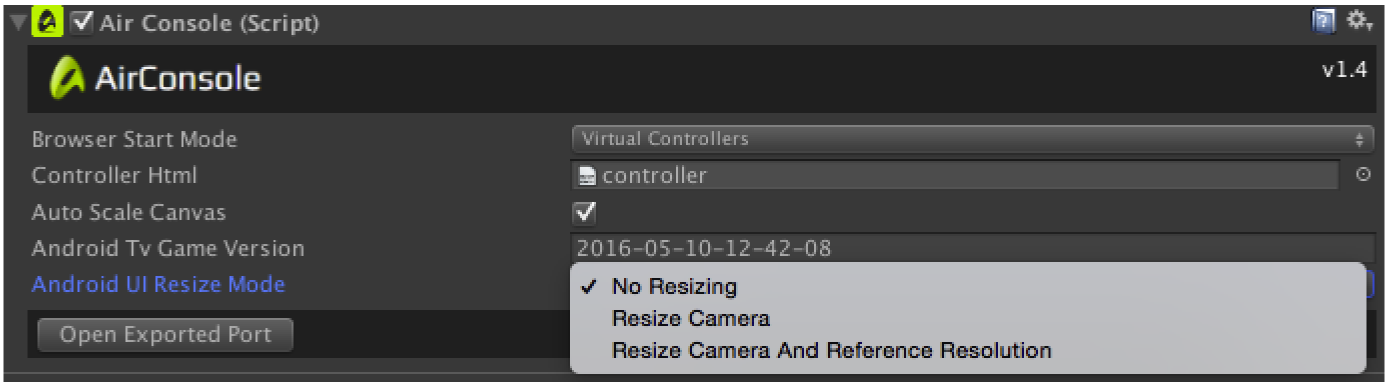
You can choose a custom loading screen logo to be displayed when your game starts on Android TV. If there is none, a default AirConsole logo will be displayed.



Android Tv Game Version: 2016-05-26-08-14-04
Android UI Resize Mode: No Resizing
Web View Loading Sprite: None (Sprite)

UI

On AndroidTV, the AirConsole Default UI bar will be laid over the game.The Default UI shows the AirConsole Logo, the session code and connected players’ portraits and connection statuses.
To make sure your UI is displayed correctly despite the cutoff at the top, the AirConsole Object offers three options on how to deal with Canvases.



Air Console (Script)
AirConsole v1.4
Browser Start Mode: Virtual Controllers
Controller Html: controller
Auto Scale Canvas: ✓
Android Tv Game Version: 2016-05-10-12-42-08
Android UI Resize Mode: [✓ No Resizing] Resize Camera / Resize Camera And Reference Resolution

- **No Resizing**
  Your UI is left as it is, but the Default UI is laid over. If you choose this option you can either:
  - Manually adjust your UI so that nothing important will be cut off at the top
  - Hide the default UI using AirConsole.instance.ShowDefaultUI(false) and display the session code yourself as part of your ingame UI
- **Resize Camera**
  The height of the default UI is subtracted from the MainCamera’s pixelRect. On your Canvas, set **RenderMode** to **Screen Space - Camera**. Now, if your UI elements’ RectTransforms are correctly anchored, they will not be cut off by the default UI. Resizing the Camera will change its aspect ratio.
- **Resize Camera and Reference Resolution**
  The camera is resized as described in option 2). In addition, the Reference Resolution of your Canvas Scalers is adjusted to keep the aspect ratio it had before. This will ensure that UI elements are placed correctly in relation to the 3D elements in your game scene. It means, however, that your canvas is narrower than your Camera.

Execution Order: The resizing happens in the OnLevelWasLoaded event. Do not depend on the order of OnLevelWasLoaded being before or after the OnEnable/Awake/Start Events.

Nested Canvases: We recommend not using nested canvases (i.e. a canvas as child of another canvas). Using nested canvases can lead to issues with some of the resizing options.

Performance

Even if your game is already well-optimized for Web, you may run into performance issues on AndroidTV. To test and improve performance, we recommend implementing an FPS counter in your game and testing on the targeted hardware frequently.

Testing: If you do not own an Android TV device for testing, we recommend testing on an android phone instead. Once you have a working build, send it to us and we will test it on our Android TV devices. This is especially useful for testing UI adjustments and camera resizing.

If you are having trouble, these tips have helped us improve performance in our own Android TV AirConsole Unity games:

- Tweak your quality settings and make sure an appropriately optimized setting is selected by default on Android. Disabling shadows, reducing the pixel light count and disabling Anti-Aliasing can make a big difference
- Disable post-processing and visual effects on your camera
- Reduce transparent geometry, avoid using big, transparent sprites
- Double check your shaders, replace them with optimized/mobile ones where possible
- Make development builds with the Autoconnect Profiler option enabled and see what has impact on performance

We recommend trying out one of these optimizations at a time and see which have impact, if necessary.

Release Build

Once your game is ready for release, there are a few more things to consider:

Key & Key Store

You need to sign your APK for release. To do so, you can create a new keystore in the Player Settings.

Bundle Code

The Bundle Code is an integer that is used to determine whether one version is more recent than another, with higher numbers indicating more recent versions. You can simply increase the number by one every time you release a new version - the same bundle code cannot be used twice.

Android Manifest

The Android Manifest is an .xml file that provides essential information about your app to the Android system, which the system must have before it can run any of the app's code.

In the manifest, you also determine which devices your game supports. The default AirConsole Android Manifest that we include in our plugin assumes you want your game to appear only on Android TV systems (which use the “leanback” Android UI) and not on Android Tablets.

If you want your AirConsole game to be playable on Android Tablets as well, simply remove the following lines from the Android Manifest:

```
<uses-feature
  android:name="android.software.leanback"
  android:required="true" />

<category android:name="android.intent.category.LEANBACK_LAUNCHER" />
```

Android Build System

In order for video ads to be displayed correctly, the `hardwareAccelerated` parameter in the Android Manifest needs to be set to true. For the WebView's PostBuildProcess script to be able to overwrite the Manifest, your Android Build System needs to be set to **Gradle** (not to Internal). You can set the Android Build System in the Build Settings or the Android Publishing Settings.
If your game was built using the Internal Android Build System before, you may have to delete your project's Library folder to force an asset reimport.

Check your Android Manifest: You can use Apktool (https://ibotpeaches.github.io/Apktool/) to unpack your APK after building and double check your Android Manifest. If `hardwareAccelerated` is set to false, we recommend closing your project, deleting your Library folder and making a new, clean Build using the **Gradle** Build System.

Android Banner

In your Unity Player Settings, in the Icon tab of the Android export settings, check the `'Enable Android Banner'` checkbox and assign a cover art for your game. This will be the icon that users see when they download your game onto their Android TV.

Please use the TV banner graphic preset provided here (https://drive.google.com/drive/folders/0BxEs-y1NpJ4zOWpZbnktMnAta3M?usp=sharing) to adorn your banner with the AirConsole logo.
