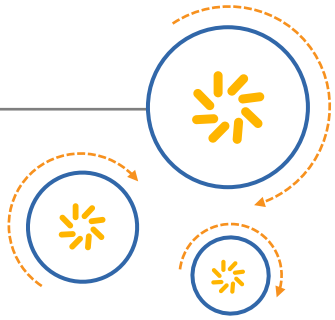
Qualcomm Technologies, Inc.



Optimal Resolution Rendering Feature

Application note

80-P1830-1 B

June 8, 2017

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision history

Revision	Date	Description
A	July 2015	Initial release
B	June 2017	Numerous changes have been made to this document to update the document for the improved version of Optimal resolution rendering feature; this document should be read in its entirety.

Contents

1 Introduction 5

1.1 Purpose 5
1.2 Conventions 5
1.3 Technical assistance 5

2 ORR feature overview 6

2.1 Limitations 8
2.2 Assumptions 8
2.3 Dependencies 8
2.4 Whitelistedapps.xml 9
2.4.1 Update whitelistedapps.xml 9
2.5 Enable ORR feature 10
2.6 Disable ORR feature 10
2.7 Obtain application attributes 11
2.8 Apply optimal resolution rendering 11
2.9 Supported games 11
2.10 Verification scope 11

3 Integration 12

3.1 Integration steps 12
3.1.1 Set density 12
3.1.2 CAF links 12

4 Feature-level debug 13

4.1 Debug setup 13
4.2 Debug steps 13

A References 14

A.1 Related documents 14
A.2 Acronyms and terms 14

Figures

Figure 2-1 Activity start call flow 7
Figure 2-2 GPU power consumption vs. resolution across chipsets 8

1 Introduction

1.1 Purpose

This document describes the Optimal resolution rendering (ORR) feature, its power-saving ability, how to enable or disable the feature, and apply it to the various games.

1.2 Conventions

Function declarations, function names, type declarations, attributes, and code samples appear in a different font, for example, `#include`.

Code variables appear in angle brackets, for example, `<number>`.

Commands to be entered appear in a different font, for example, **`copy a:*.* b:`**.

Button and key names appear in bold font, for example, click **Save** or press **Enter**.

1.3 Technical assistance

For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://createpoint.qti.qualcomm.com/.

If you do not have access to the CDMATech Support website, register for access or send email to support.cdmatech@qti.qualcomm.com.

2 ORR feature overview

ORR feature helps lower the game application resolution than the display panel resolution and then upscales the game application content as part of composition. For example, when a game is rendered at 720 × 1280, while display resolution is 1080 × 1920, then the ORR feature can be used to render the game at the display resolution of 1080 × 1920.

The ORR feature reduces the load on the GPU and helps improve power consumption and performance of the device.

The third-party games and applications render at the panel resolution. Dpi of the device is defined based on the resolution and size of the panel. The ORR feature helps optimize the power and scale the application density to one level lower than the device density, so that the application renders at optimal resolution without a noticeable quality degradation.

The whitelistedapps.xml at /system/etc/ contains the list of applications that require a resolution change. You can update this file to add any applications that require an ORR feature. To know more on how to update whitelistedapps.xml, see Section 2.1.

Figure 2-1 shows the generic call flow of any game application. The S*tartActivity()* function is triggered during the launch of any game. When the game tuple (PackName, Activity and VersionCode) matches with the tuple listed in whitelistedapp.xml file, the *OverrideDensity()* function is called which overrides the density for that application. *Activity start()* is called with new density of the application.

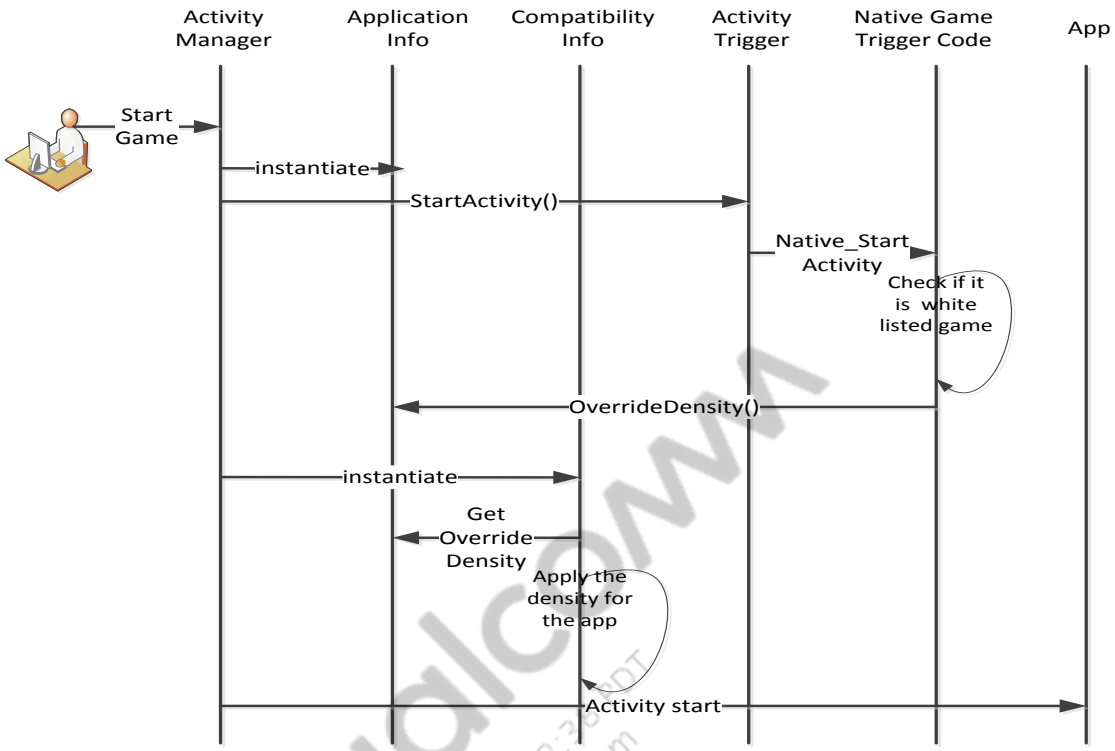


Figure 2-1 Activity start call flow

By doing some minor changes in the CompatibilityInfo, ActivityThread and ApplicationInfo Java classes, you can override the final density of the application to scale to the lower density instead of the default density.

The density changes using the ORR feature does not cause any issues such as artifacts or flickering and no noticeable quality degradation.

NOTE: The density change is triggered only for those applications listed in the selected whitelistedapps.xml file and the remaining applications are rendered at default density of the device.

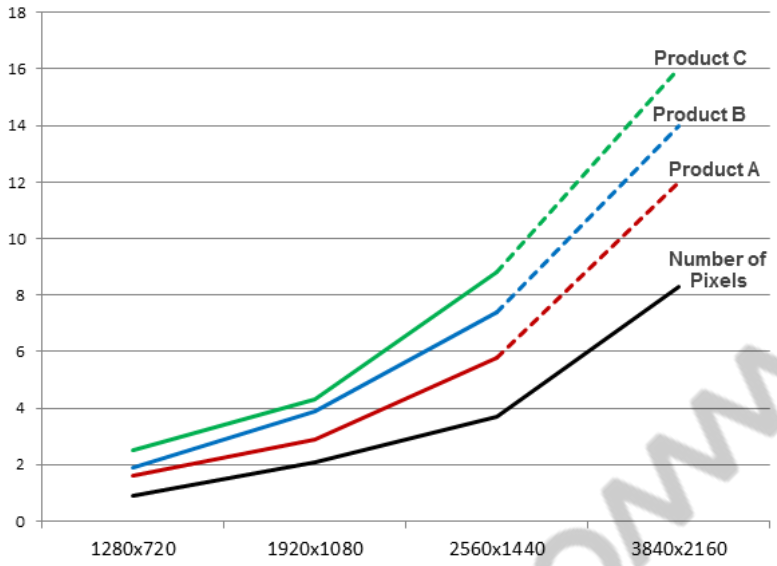


Figure 2-2 GPU power consumption vs. resolution across chipsets

Figure 2-2 shows power consumption of the GPU at different rendering resolutions, across several popular chipsets in use on Android devices.

2.1 Limitations

- Typically, games use SurfaceView and render using OpenGL. For the games, which have a different layout, the ORR feature may not work properly. In such cases, exclude those applications from the whitelistedapps.xml.
- Density cannot be set lower than 120 dpi. New density should be lower than default density for power benefit without impacting the visual quality.
- Currently, this feature is tested on Android N build and is working fine.
  For Android O build, this feature is still under evaluation.

2.2 Assumptions

This feature depends completely on upscaling content during composition. The assumption is that scaling is performed by a display hardware accelerator that consumes less power.

2.3 Dependencies

- Software – Whitelistedapps.xml
- Touch – None
- External – None

2.4 Whitelistedapps.xml

The whitelistedapps.xml at /system/etc/ contains the list of applications that require the ORR feature. The .xml structure for each application listed in the Whitelistedapps.xml is as following:

```
<AppAttributes
   PackageName=”com.abc.xyx”
   ActivityName=”com.abc……”
   VersionCode=”123.
/>
```

NOTE: In AppAttributes, PackageName must be defined, whereas the remaining attributes are optional.

As a default setting, one or two applications are listed in the Whitelistedapps.xml, you can update Whitelistedapps.xml to add multiple applications that require the ORR feature.

2.4.1 Update whitelistedapps.xml

Use the following methods to add applications to the whitelistedapps.xml:

- Update PackageName only

  In AppAttributes, update PackageName. ActivityName and VersionCode are optional attributes and are used to further restrict the whitelisting.

  ```
  <AppAttributes PackageName="com.imangi.templerun" />
  ```

- Update PackgeName and ActivityName

  To override the resolution of the game, update PackageName and ActivityName using the following command:

  ```
  <AppAttributes PackageName="com.imangi.templerun"
  ActivityName="com.unity3d.player.UnityPlayerProxyActivity" />
  ```

  The function OverrideDensity() of game is triggerred only when the game launches with the specific ActivityName and the game specific PackageName as mentioned in the xml structure.

- Update PackageName and VersionCode

  Use the attribute values obtained from logcat logs to update the whitelistedapps.xml.

  The function setOverrideDensity() of game is triggered when the PackageName and VerisionCode name matches the package name and version code listed in the whitelistedapps.xml.

  ```
  <AppAttributes PackageName="com.imangi.templerun"
  VersionCode="11" />
  ```

NOTE: Only the exact application version that matches with the version code in whitelistedapps.xml are allowed for ORR.

- Update all attributes

To update all the attributes, run the following command:

```
<AppAttributes PackageName="com.imangi.templerun"
ActivityName="com.unity3d.player.UnityPlayerProxyActivity"
VersionCode="11" />
```

NOTE: If artifacts and flickers are observed while playing game, then remove the game from the whitelistedapps.xml.

2.5 Enable ORR feature

To enable the ORR feature for 1080p display resolution, set the application density to the required density and run the following command to enable this feature:

```
adb shell setprop persist.debug.appdensity <value. Ex: 240>
```

Set the value to the required density (should be less than screen default density and cannot be lesser than 120 dpi). The following steps show how to calculate the required density:

1. Calculate the scaling factor as:

$$Scaling\ factor = \frac{Display\ panel\ width}{Game\ application\ \text{required}\ width}$$

2. Calculate application density as:

$$Required\ Density = \frac{Device\ density}{Scaling\ factor}$$

For example, if the display panel is at 1080p and to render the game at 720p, the required density is calculated as:

Scaling factor = 1080/720 = 1.5

Required app density = (360)/1.5 = 240

Here if 360 Dpi is your 1080p display panel density or else replace with your device display density here.

To set the app density to 240, run the command:

```
adb shell setprop persist.debug.appdensity <240>
```

2.6 Disable ORR feature

To disable the ORR feature, set the density to zero or an undefined value. To set the density to zero, run the following command:

```
adb shell setprop persist.debug.appdensity 0 –
```

2.7 Obtain application attributes

To obtain the application attributes such as package name, activity name, and version code, follow the steps listed:

1. Enable the game trigger logs using the command:
   ```
   adb shell setprop persist.debug.gtlogs_enable 1
   ```
2. Run the game and run the `grep` command for *perf* in the logcat logs.

2.8 Apply optimal resolution rendering

To apply ORR feature to game applications, add the games that run seamlessly when rendered at optimal resolution to *name/activity/version* in the whitelistedapps.xml file at /system/etc/.

2.9 Supported games

- ThunderCross
- Pacific Air War
- Candy Crush
- Subway Surfer

2.10 Verification scope

Testing must ensure that artifacts and flickers do not occur and cover the following areas:

- Rotation
- Power suspend-resume
- App switching
- Various games

Extensive testing is recommended for use cases with SurfaceView like camera, video, and browser.

3 Integration

3.1 Integration steps

1. Add the list of applications that require ORR, in whitelistedapps.xml at /system/etc/.
2. Add the supported application's PackageName in the .xml structure of each application in the whitelistedapps.xml.
3. Verify if libqti-gt.so lib is added, if it is not present then build or add this library.

3.1.1 Set density

Use the following setproperty to set the density:

```
persist.debug.appdensity
```

3.1.2 CAF links

Download the ORR feature software from the following links:

- ORR feature implementation:

  https://source.codeaurora.org/external/gigabyte/platform/frameworks/base/commit/?h=caf/android-framework.lnx.2.0.r16-rel&id=a28956ae9031fd938a9f422768613cd8f5cbef0d

- ORR feature handling in split screen:

  https://source.codeaurora.org/external/gigabyte/platform/frameworks/base/commit/?h=caf/android-framework.lnx.2.0.r16-rel&id=107acc226d92dd25d48867698b441464406f4338

4 Feature-level debug

4.1 Debug setup

An ADB shell environment is required.

Use Activity trigger module to check if the application is whitelisted or not. Search for the highlighted message in the logcat logs.

```
Log.e(TAG, "activityStartTrigger: whiteListed " + activity +" appInfo.flags
- " + Integer.toHexString(appInfo.flags));
Or
Log.e(TAG, "activityStartTrigger: not whiteListed" + activity);
```

4.2 Debug steps

1. Check which override resolution is set using the following command:

   ```
   adb shell getprop persist.debug. appdensity
   ```

2. Check the rendering resolution for the game from SurfaceFlinger dumps using the following command:

   ```
   adb shell dumpsys SurfaceFlinger
   ```

Output – Sample output; here source crop is at rendering resolution and frame is at composition resolution. Only the game window scales to optimal resolution and other layer windows do not change.

```
type   |  handle  | hint | flag | tr | blnd |   format    |     source crop
(l,t,r,b)      |          frame         | name----------+----------+------
+------+----+------+------------+-------------------------------+--------
---------------+------
 HWC | 7fa3a33610 | 0002 | 0000 | 00 | 0100 | ?
00000115  |    0.0,    0.0,  960.0, 1611.0 |    0,    0, 1440, 2417 |
SurfaceView -
com.king.candycrushjellysaga/com.king.candycrushjellysaga.StritzActivity
      HWC | 7fa342b540 | 0002 | 0000 | 00 | 0105 |
RGBA_8888   |    0.0,    0.0, 1440.0,  144.0 |    0, 2416, 1440, 2560 |
NavigationBar
FB TARGET | 7fa3a31fc0 | 0000 | 0000 | 00 | 0105 |
RGBA_8888   |    0.0,    0.0, 1440.0, 2560.0 |    0,    0, 1440, 2560 |
HWC_FRAMEBUFFER_TARGET
```

A References

A.1 Related documents

Title	Number
Standards	
Android Developers Blog	http://android-developers.blogspot.in/2013/09/using-hardware-scaler-for-performance.html

A.2 Acronyms and terms

Acronym or term	Definition
ADB	Android Debug Bridge
AOSP	Android Open Source Project
ORR	Optimal Resolution Rendering