***Introduction to Graphics and Mobile Gaming***

***Arm Mobile Studio***

**Getting Started Guide**

**Issue 1.0**

**A picture containing outdoor

Description automatically generated**

Contents

[1 Introduction 1](#_Toc86237156)

[1.1 Overview 1](#_Toc86237157)

[1.2 Scope of this guide 1](#_Toc86237158)

[2 Requirements 1](#_Toc86237159)

[3 Getting started 2](#_Toc86237160)

[4 Streamline 2](#_Toc86237161)

[4.1 PC setup 2](#_Toc86237162)

[4.2 Target device setup 3](#_Toc86237163)

[4.3 Performing the capture 3](#_Toc86237164)

[5 Graphics Analyzer 5](#_Toc86237165)

[5.1 Overview 5](#_Toc86237166)

[5.2 Target device setup 5](#_Toc86237167)

[5.3 Apk setup 5](#_Toc86237168)

[5.3.1 Unity Setup 6](#_Toc86237169)

[5.4 PC setup 7](#_Toc86237170)

[6 Streamline and the Graphics Analyzer 7](#_Toc86237171)

[7 Appendix 9](#_Toc86237172)

[7.1 Not working on Windows 10 9](#_Toc86237173)

[7.2 Unauthorized connected device 9](#_Toc86237174)

[7.3 Android package name 9](#_Toc86237175)

[7.4 Path for gatord not recognized 9](#_Toc86237176)

[7.5 Sdk higher than 25 9](#_Toc86237177)

[7.6 Project doesn’t show in GA app list for apps to trace 9](#_Toc86237178)

# Introduction

## Overview

Arm Mobile Studio is a collection of free available tools for mobile games and graphics optimization. It currently includes Streamline, a performance analyzer, which offers a full system view, and the Graphics Analyzer (GA), which can be used to trace graphics performance and issues.

## Scope of this guide

This Getting Started Guide is intended to give you the necessary information to install and configure the tools mentioned above. The targeted PC platform is Windows 10, and the currently working versions of the tools are: Streamline Version: 7.0.0

Graphics analyzer Version: 5.3.0.8

If you’re working on a different platform, please refer to the appendix.

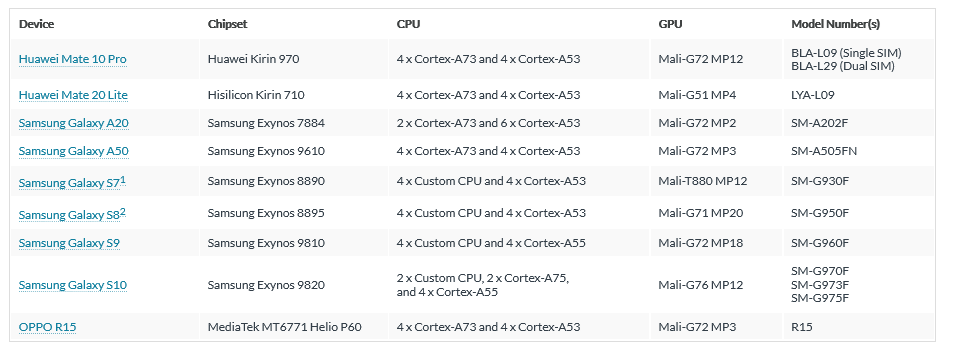
A better integration of commands is expected in later versions, but for now, we’ll do the set up from the command line. Please follow this guide step-by-step or if the version has changed, refer to the online guide.

# Requirements

In order to perform the analysis, you will require a mobile device with an Arm-based processor and an Arm Mali GPU from the list below:

* Mali Bifrost – G31
* Mali Bifrost - G51
* Mali Bifrost – G52
* Mali Bifrost – G71
* Mali Bifrost – G72
* Mali Bifrost – G76
* Mali Midgard
* Mali Utgard

List of fully supported devices:



# Getting started

First thing we need to do is install the Arm Mobile development Studio that includes Streamline Performance Analyzer and the Graphics Analyzer: <https://developer.arm.com/tools-and-software/graphics-and-gaming/arm-mobile-studio/downloads>

Try to import the samples as well, to get an idea of what we’re looking at. You can right-click on any of the samples and then **Analyze,** and it will profile an app so you can get an idea of how it works.

From here, we’re going to try to analyze the performance of the apps we’ve built in the course.

# Streamline

## PC setup

Streamline gets installed with the Arm Mobile studio. But for Streamline to be able to collect data from your mobile devices, the script ***“gatord”*** needs to run on the target device so it can send appropriate information from the mobile device to the software.

* To install this on your mobile device, you can use the ***“gatorme”*** script that will perform the installation for you. Download ***gatorme*** on your PC from here <https://developer.arm.com/tools-and-software/graphics-and-gaming/arm-mobile-studio/learn/get-started/get-started-with-streamline>

(Make sure you download/clone the whole Git repository and extract the file you need in order to avoid any download issues.)

* Install the ***adb*** on your PC. You should already have this installed from previous labs, but if you don’t, the adb is available with the Android SDK platform tools that can be found here to download: <https://developer.android.com/studio/releases/platform-tools.html>.
* A screenshot of a cell phone

  Description automatically generated We need the **adb** to be accessible to the **gatorme** script so you must set the Android SDK platform tools directory in the PATH environmental variable. In Windows 10, click on **Start** and look for “*edit the system environmental variables*” and then select **Environment Variables;** in the **System variables** window, find “*path*.” Click on **Edit** and add the full path of the directory where you installed the platform tools.

## Target device setup

Now let’s move to your device:

* Ensure **Developer Mode** is enabled, as well as **USB Debugging** (you should have this enabled from previous labs)
* Connect your mobile device to your machine. Open a command prompt and run the command: adb devices.

This should display the Id of the devices connected. If the Id device appears with “unauthorized” besides it, please follow the instructions in the appendix

* Install the application to be analyzed. You can do this by navigation to the directory where you have the apk and typing the following in the command prompt:

adb install App.apk

If your app is already installed, you don’t need to do anything else here.

## Performing the capture

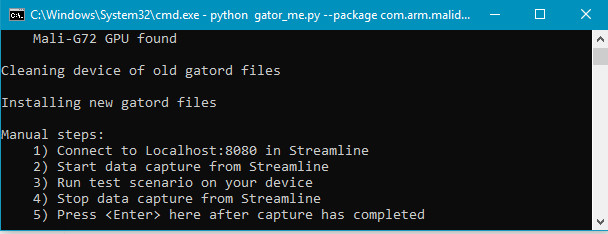
To perform a capture of the analyzed system, you need to run the **gatorme** script on your host machine.

Arguments:

1. The **Android package name** of the app you want to analyze (refer to appendix if you don’t know)
2. The short name of the **Mali GPU** on the mobile device. You can check this on the manufacturer page
3. The **path** on the host to the gatord binary installation. This is usually in *<streamline\_directory>/bin/arm/* for 32-bit architecture CPU or *<streamline\_directory>/bin/arm64* for 64-bit architectures. These specifications can also be found on the manufacturer page (refer to appendix).

For example, for a Samsung S9 with a Mali-G72 GPU from the directory where ***gatorme*** is located:

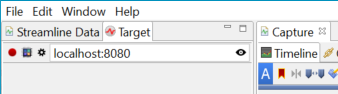
python gator\_me.py –-package com.arm.<app\_name> --daemon <gatord\_path>

The cmd should now display something like the picture on the right, remember to run cmd **as an Administrator**.

At this point, we need to perform the manual capture:

1. Launch Streamline
2. Navigate to the **Target**tab and set it to localhost:8080. This is the number of the port that gatorme script uses.

**Counter config**



1. Next, configure the counters to collect. Press the **Counter Configuration** button and then choose **Add counters**. Select a counter template appropriate for the target GPU from the drop-down list:

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated



1. Click on Start Capture button.

Grab your mobile device and launch the app that you want to profile. Launching any other app will not get anything recorded because you have already specified the name of the package you wish to be analyzed.

A screenshot of a computer

Description automatically generatedA live timeline statistic should now appear in Streamline. You can stop the capture at any point using the stop button.

# Graphics Analyzer

## Overview

The Arm Mobile Studio package also includes the Graphics Analyzer (GA) tool. This is targeted at OpenGL ES, EGL, OpenCL, and Vulkan application that allows in-depth analysis of individual function calls and returns on the target. It also provides trace information, a live timeline, GPU states, and can detect misuses of the API.

Integrated with Streamline, it paints a much nicer picture, bringing new charts so we can see the frame rate, draw calls per frame, number of vertices, and number of vertices per frame. From here, we can observe when frame rate drops and analyze why, or if certain frames seem to have a much higher count of vertices than other it might be worth looking into what causes that. A constant increase in vertices is not a good sign as it means that objects get added but not destroyed over time that can clog up the flow of the application.

## Target device setup

You should already have the Graphics Analyzer installed, along with the ADB. Make sure the Android SDK platform tools directory has been added to the *path* environmental variable.

* On the target device, enable **USB debugging** just as we have done it before. We assume that your device is unrooted. For rooted device guidelines, please refer to the user guide (open Graphics Analyzer-> Help -> Help contents)
* Connect the Android device to the host PC. Run adb devices in the command prompt to make sure the device is recognized.
* Install the GA application on the Android target device:
  1. Open a terminal and navigate to <GraphicsAnalyzerDirectory>/target/android/arm
  2. Install the GA by running the following command: adb install -r AGA.apk

## Apk setup

We will need to modify some files in the target application you wish to analyze, before we install it on the device.

* For unrooted devices, we need to package the **Interceptor** in the source code of the application. We’ll do this by adding the GA unrooted library folder to the list of locations the Android build looks for native libraries.
* Open the source apk package in Android Studio preferably.

a. Navigate to the **app -> build.gradle** file in your application package

b. Under **android**, add the following code:

android {

sourceSets {

main {

jniLibs.srcDirs += ['<GA path>\\target\\android\\arm\\unrooted']

}

}

…

}

* Next step is to enable the Interceptor. Our example applications developed in previous labs include both Java and C++ code, so we’ll need to add the following code at the beginning of the main activity in the app:

static

{

try

{

System.loadLibrary("AGA");

}

catch (UnsatisfiedLinkError e)

{

// Feel free to remove this log message.

Log.e("[ GA ]", "GA not loaded: " + e.getMessage());

Log.d("[ GA ]", Log.getStackTraceString(e));

}

}

(For applications with C/C++ only, please refer to the instructions in the user guide (open Graphics Analyzer-> Help -> Help contents))

* The Interceptor for unrooted devices is only available for armeabi-v7a and arm64-v8a ABIs, so we’ll need to specify them in the same build.gradle file as before:

Under android -> defaultConfig, specify your target:

android {

…

defaultConfig {

…

ndk {

abiFilters 'armeabi-v7a', 'arm64-v8a'

}

}

}

* Ensure the Android application build.gradle specifies an **android:targetSdkVersion** of **25** or lower. If the sdk in your app is higher than 25, please refer to the appendix.

Recompile the application and install it on the target device, either by Build and Run or manual install using the adb.

### Unity Setup

If you have a Unity project, you need to do the following steps:

1. Go to ***Build Settings*** and tick **Development Build**
2. Navigate to <graphics\_analyzer>\target\android\arm\unrooted\[armeabi-v7a or arm64-v8a] and copy the libMGD.so file. Place this in a new directory **Assets/Plugins/Android**.

## A screenshot of a social media post Description automatically generatedPC setup

Open the Graphics Analyzer on your host PC.

* From the **Debug** menu, select **Open the Device Manager**. Make sure the path to the platform tools is recognized. Once the device appears in the list, right-click and select: **(Re-)install the Graphics Analyzer Android App on the selected device(s)**.
* Connect the host to the target device. For this, in the Graphics Analyzer host application, select **Debug -> Open the Device Manager**. It should automatically detect your device. Right-click and select: **Connect to selected device**.
* A screenshot of a cell phone

  Description automatically generatedOptionally set the process to control which types of API assets will be collected by the analyzer. If you enable many asset types, your app might slow down, and it can cause the GA to require more memory and generate larger files. For now, just select **Everything**.
* The next step involves launching the GA on the target device. You should already see a Graphics Analyzer app on your device. Launch and ensure to Enable Graphics Analyzer Daemon. If all the code inserted in your application is correct, you should see the app you wish to analyze in the list of applications to trace.
* Click on the app, and you should now see a live timeline on the PC while the app runs on the device

Now that we’ve seen both analyzers separately, we can integrate the GA with Streamline.

# Streamline and the Graphics Analyzer

Integration will result in Streamline outputting more fields, relevant to graphics analysis.

* First, ensure that you follow all the steps in the APK setup from the Graphics Analyzer instructions. The apk you wish to analyze should have the Interceptor packaged. Thanks to the Interceptor, Streamline will be able to capture more information.
* Next step is to launch the app as if you want to analyze it with Streamline. Please follow all steps from Performing capture.

Note: The aga-daemon daemon application should not be running when using the Streamline tracing functionality.

# Appendix

## Not working on Windows 10

Please refer to the guide [here](https://developer.arm.com/tools-and-software/graphics-and-gaming/arm-mobile-studio/downloads?_ga=2.45812391.204797223.1565099297-1867192701.1559896154) .

## Unauthorized connected device

* Go to **Settings -> Developer options -> Revoke USB debugging authorizations**. Disconnect device
* Restart the adb server by typing the following commands in your command prompt:

adb kill-server

adb start-server

* Reconnect your device. Make sure you allow the connection when prompted by the device
* Now check the adb devices list with the adb devices command

## Android package name

The Android package name was set at the beginning of the project and should be of the form com.arm.malideveloper.<app>. If you set the name of your package differently, please refer to that.

Alternatively, if you don’t know the name: With the device connected, open a command prompt and run adb shell pm list packages. This command will display all packages installed on your mobile device, and you can find your app there.

## Path for gatord not recognized

Sometimes, the Windows cmd would not recognize spaces in a path passed to it as an argument. What you can try is to put the path in quotes.

If that doesn’t work, access the ***gatord*** directory manually, copy the ***gatord,*** and place it in the same directory as your ***gatorme*** script. Open a cmd in the directory with both scripts, and you can type:

python gator\_me.py –-package com.arm.<app\_name> --daemon gatord

## Sdk higher than 25

In this Streamline version, capturing for a higher sdk version might not be possible, depending on the device. You can either try to manually change the target sdk in the **app -> build.gradle** file, which is not ideal, or please refer to this page for further instructions <https://community.arm.com/developer/tools-software/graphics/b/blog/posts/graphics-analyzer-android-sdk-26>

## Project doesn’t show in GA app list for apps to trace

Restart your device. Sometimes installing a new app or reinstalling an app makes GA not pick it up anymore.