High-Performance Computing on Android Devices

Student: **John Lampitt Adey**
Supervisor: **Dr.Muppala**

Introduction

As of May 2013, over 900 million Android devices had been activated. With these devices getting increasingly powerful, multi core CPUs and dedicated GPUs are commonly found on new devices. With this increased power, more developers are looking to harness this power for more complex applications.

Objectives

This project aimed to look at the effectiveness of Renderscript in providing maximum performance on Android devices and it's accessibility to developers. To do this we produced an Android app with common computationally intensive algorithms implemented in it.

Another aspect of this project was to look at the usability of Renderscript as a computational solution. In order to do this we looked at two of the most popular solutions, CUDA and OpenCL.

Design

Analyze and code common intensive functions in Java, Renderscript, OpenCL and CUDA:

- **Matrix Multiplication**
- **General mathematical functions**
- **Histogram**
- **Image blur**
- **Color image to grayscale**
- **N body simulation**

Implementation

We developed an Android application to showcase both the Renderscript and Java implementations.

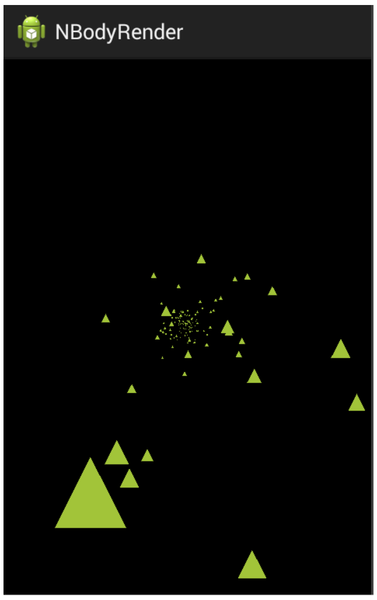


Figure 1 Showing an N Body simulation running on Android

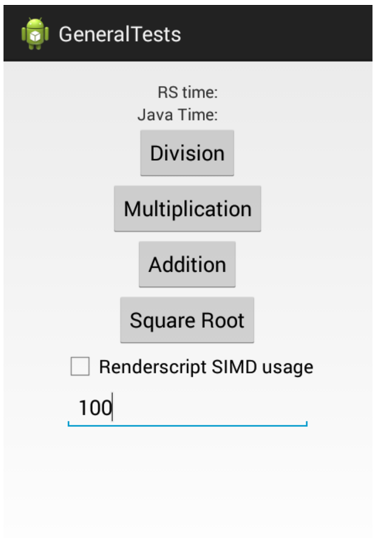


Figure 2 showing the tests cases for basic math and SIMD functions.

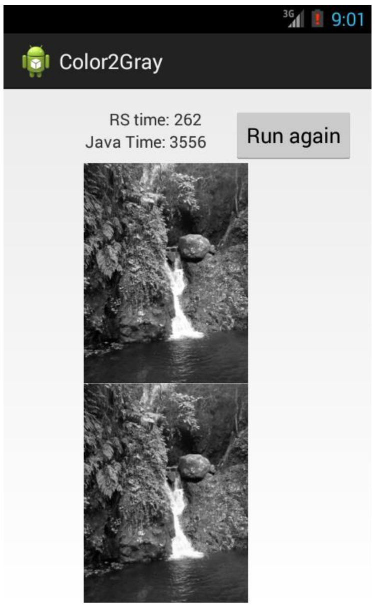
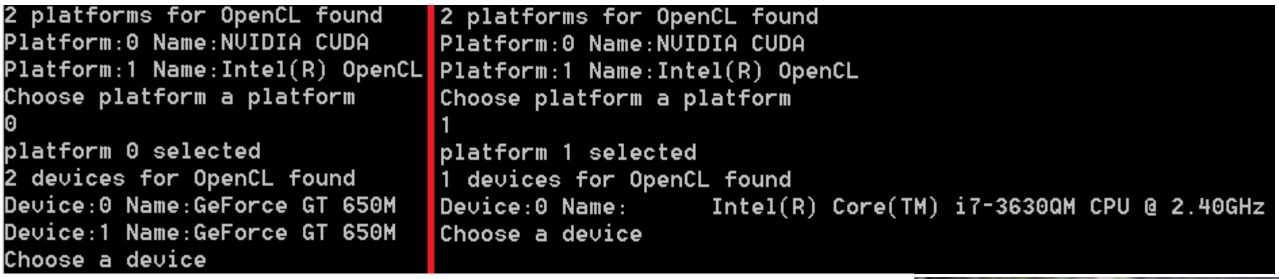


Figure 3 Showing Color Images having been transformed to Grayscale by Java and Renderscript

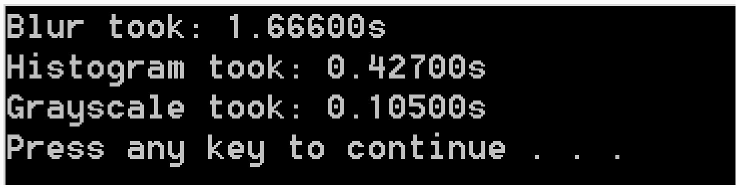
For our OpenCL and CUDA implementations, we developed a console application.



```
2 platforms for OpenCL found
Platform:0 Name:NVIDIA CUDA
Platform:1 Name:Intel(R) OpenCL
Choose platform a platform
0
platform 0 selected
2 devices for OpenCL found
Device:0 Name:GeForce GT 650M
Device:1 Name:GeForce GT 650M
Choose a device
```

```
2 platforms for OpenCL found
Platform:0 Name:NVIDIA CUDA
Platform:1 Name:Intel(R) OpenCL
Choose platform a platform
1
platform 1 selected
1 devices for OpenCL found
Device:0 Name:       Intel(R) Core(TM) i7-3630QM CPU @ 2.40GHz
Choose a device
```

Figure 4 showing the initial run choices in our OpenCL application



```
Blur took: 1.66600s
Histogram took: 0.42700s
Grayscale took: 0.10500s
Press any key to continue . . .
```

Figure 5 console application runs the set functions and stores results in applications folder

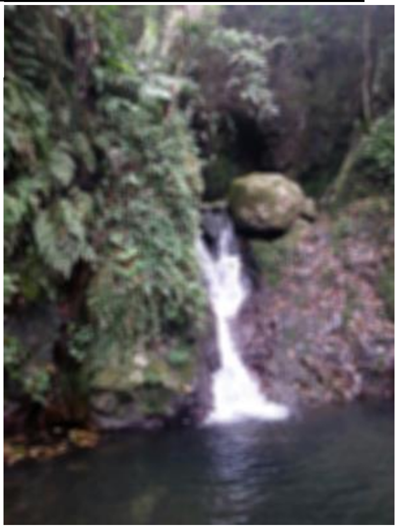


Figure 6 example image which has been blurred using our CUDA application

Results

Results showed that Renderscript could produce impactful performance gains. However it also showed us that it is not always faster, with small workloads Java with its lower overheads performed better.

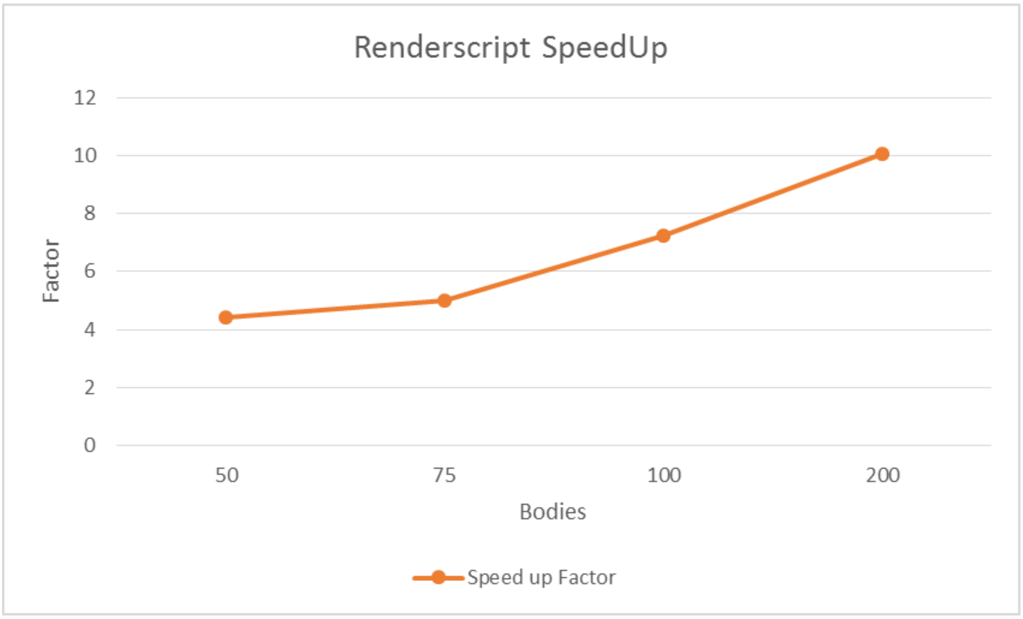
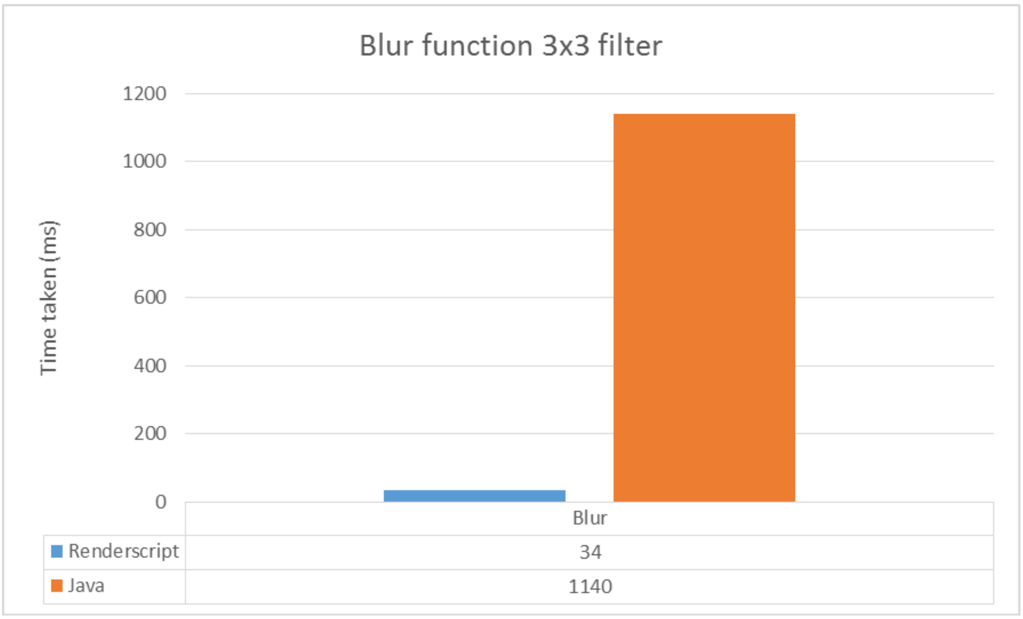


Figure 7 showing the speed up Renderscript offered over Java while performing the N Body simulation with a varying body count.



	Blur
Renderscript	34
Java	1140

Figure 8 showing the execution time of blurring an image with a 3x3 filter