PHILIPS

Virtual Listen Room

Surround Room Set, inside Your Android

Durio Etgar 2190198
Eindhoven 2012

Fontys

University of Applied Sciences

GRADUATION PROJECT REPORT
FONTYS UNIVERSITY OF APPLIED SCIENCES

HBO-ICT: English Stream

Data student:

Durio Etgar

Student number: 2190198

Graduation project period: from 6 February 2012 till 29 June 2012

Data company:

Name company/institution: Philips Research

Department: Applied Sensor Technologies

Location: High Tech Campus 36, 5656 AE Eindhoven, The Netherlands

Family name + first names + position of company tutor: Yuan, Zhaorui. Software Designer.

Data university tutor:

Family name + first names company tutor: Zijlmans, Jack.

Data final report:

Title of report: Virtual Listen Room: Surround Room Set, inside Your Android

Date final report has been issued:

Approved and signed by the company tutor:

Date:

Signature Company tutor: Signature University tutor:

FOREWORD

Study becomes a responsibility for all human being. Since we saw the world at the first time, we started to learn from everything that exists around us. Everything we've got so far, sometimes need to be truly implemented, wherever it is. I had 3 years of Electrical Engineering Bachelor education on my country Indonesia, and somehow I took the chance to have a wider perspective about Information & Communication Technology here in The Netherlands. Fortunately I got a chance to convert my knowledge into something useful at Philips, one of the biggest, well-known companies in the world. This could be the most important 90 days on my career, and I realized I can't expect more than this as the first step.

The wise said, "Bind your knowledge by writing it down". I arranged this thesis to keep this priceless experience eternal, bound forever. It completely covered how I start this project, what goal should I accomplished, what problems I faced, and what method did I use to make this research more meaningful.

I try to guide myself into the next generation of technology, which is called Mobile Development, or specifically, Android Development. Since I got my first Android device, there is a big desire to do more and more, reveal all the possibilities and stretch the system into the edge. Thankfully I'm involved to their world of Open Source. It's leaving a big space for development in many aspects. My enthusiasm was followed by a golden opportunity from Philips and especially from my Company Mentor Mr. Werner Oomen and Mrs. Zhaorui Yuan. They offered me an assignment about building a 3D room visualization coupled with sensor movement, and guess what, on Android. To be honest, who would reject a wonderful company with a project that you desired most? My deepest gratitude goes to them for giving me a chance and conducting me in whole internship process.

I'm not really good on writing report, thesis, or such a kind of writing stuff. I would like to express my biggest thanks to Mr. Jack Ziljmans for the guidance during my documentation process. He led me on writing my thesis with his preferable method. It was very useful and I will safe it in my backpack to face the real world.

Last but not least, I'm beholden to all my colleagues in Philips for a warm and comfortable working environment, my family and my friends for continuous support, and for all technology founders around the world. Kudos to you all.

There's no Ivory without crack. I apologize if there's any writing error contained. Moreover, I dedicate this thesis to the advancement of technology, which I believe will bring us a better life. Kindly open the following pages and I hope you'll be impressed.

Date,

Durio Etgar

TABLE OF CONTENTS

TABLE OF FIGURES

SUMMARY

Huge technology improvements on the recent years are boosting the market demand. Complicated stuffs in life seem to be simplified due to invention of many automation and simulation systems, including sound playback set. For instance, people desires to have cinema-like surround sound without messing up with speakers and wires. Meanwhile, Android is currently hyped as one of the best IT contrivance in recent years. Android is an open platform so many vendors can easily deploy this operating system on their own devices. Both of those breakthrough developments can lead into some groundbreaking imaginations, e.g. how if we want to have a virtual 3D room inside our Android device? Is that possible? Fortunately I had a chance to inspect this possibility by doing research on Philips, one of the biggest electronic companies whose products are spread all over the world. They gave me an assignment about implementing surround sound coupled with 3D room visualization and sensor movement. With the help of embedded Android graphic machine and various libraries, 3D room can be simply realized. Android also has an accelerometer sensor, a sensor which is detecting how you tilt your phone. Combined with Philips' prebuilt MPEG surround decoder, it is possible to develop an Android application with virtual surround effect while moving the phones. Problems appeared when the heavy-load of 3D, playback, and sensor were affecting the performance. Android provides several tools for testing and it gave useful information in order to fix the issues. Compatibility problem was also spotted because of native library, which makes the application can't be installed into different Android version. This time it is still unsolved. Despite of compatibility problem, my app has met all of requirements stated. From this research I can conclude that it's likely to have a 3D graphic application on Android but then we have to also concern with the performance. Beside, I learnt a lesson that it's an obligation to decide which platform will be used if you want to work with native library later. In the whole process I still have to improve especially on how I apply research and development methodologies as well. Finally, I hope this report will be useful for anyone.

GLOSSARY

Accelerometer : a device that measures weight per unit of (test) mass

Android : Linux-based operating system for mobile devices such as smartphones and tablet computers

API : application programming interface, is a specification intended to be used as an interface by software components to communicate with each other

APK : Android application package file (APK) is the file format used to distribute and install application software

CPU : central processing unit, is the hardware within a computer system which carries out the instructions of a computer program

DDMS : Dalvik Debug Monitor Server, a debugging tool for Android

Decoder : a device which undoing the encoding so that the original information can be retrieved

Eclipse : a multi-language software development environment comprising an integrated development environment

GPU : graphics processing unit, is a specialized electronic circuit designed to accelerate the building of images in a frame buffer intended for output to a display.

GUI : graphical user interface, is a type of user interface that allows users to interact with electronic devices using images

JNI : Java Native Interface, is a programming framework that enables Java code to call, and to be called by native applications written in other languages such as C and C++

Library : a collection of resources used to develop software

MP4 : a multimedia container format standard

MPEG : a standard for the generic coding of moving pictures and associated audio information.

NDK : a companion tool to the Android SDK that lets you build performance-critical portions of your apps in native code.

OBJ : a simple data-format that represents 3D geometry

OpenGL ES : a subset of the OpenGL 3D graphics application programming interface designed for embedded systems such as mobile phones, PDAs, and video game consoles.

RAM : Random access memory, is a form of computer data storage.

SDK : software development kit, is typically a set of software development tools that allows for the creation of applications for a certain software package

Sensor : a converter that measures a physical quantity and converts it into a signal which can be read by an observer or by an (today mostly electronic) instrument

Surround Sound :Surround sound is a technique for enriching the sound reproduction quality of an audio source with additional audio channels from speakers that surround the listener

WAV : Waveform Audio File Format, is a Microsoft and IBM audio file format standard for storing an audio bitstream on PCs.

CHAPTER 1: INTRODUCTION

Everybody loves movies. Almost every week there is always at least one new movie waiting to be enjoyed. There's so many ways to watch it, whether is it sitting comfortably in the Cinema or maybe just facing computer screen with loneliness. But there are 2 kinds of experiences which could only be found at the Cinema (except you have a set of home cinema): high definition wide screen, coupled by thunderous sound which could stop your heartbeat. We will discuss much about sound, so let's put the things into perspective.

Have you ever watch action or war-based movie on the Cinema? Rhetorically should be yes. You can feel bullets sling inside your head, the explosion bang around and chopper sound above you. It can be accomplished through the SURROUND SOUND technology. The technique enhances the perception of sound spatialization by exploiting sound localization; a listener's ability to identify the location or origin of a detected sound in direction and distance. Typically this is achieved by using multiple discrete audio channels routed to an array of loudspeakers. Talking about simplification, the surround sound let you hear mixed sound from different sources and direction to make you as if you are in the middle of the scene.

The main idea of this project is generating a 3D surround sound room scenery coupled with surround sound in a handier package, namely, a "Virtual Listen Room". The skill scope is around mid-level programming to mid-level designing room arrangement. Android device had been chosen as the implementation target, since it leaves massive development spaces and mostly contains essential components needed on this project, including graphic processor unit (GPU). Graphic manipulation can be done using an embedded programming interface called OpenGL ES, which is planted in all Android devices generally. If is it too hard to swallow you can grab any 3D game as an example, which is basically rendered using OpenGL ES. Android is using Linux environment and Java as a programming language.

The problem is all kind of android devices only have maximum 2 speakers recognized. Yes, and to deal with it, Philips Research - Information and Cognition - Applied Sensor Technologies division has their own Virtual surround sound decoder called "MPEG Decoder ™ " which has been developed for months. Virtual surround is an audio system which attempts to create the perception that there are many more sources of sound than are actually present. In order to achieve the goal it is necessary to devise some means of tricking the human auditory system into thinking that a sound is coming from somewhere that it is not. Later this application will be coupled with the 3D scene, and while listening to the playback music we can tilt our scene camera view and even simulate sound changing based on head movement on the further development, thanks to Android Accelerometer Sensor.

Android graphic and sound processing are barely touched by the developers, and it might be you as well. So, this thesis could be interesting. This project covers the basic aspect of graphic processing combined with the knowledge of surround room setup, and everything was wrapped neatly into this thesis. Chapter 2 in the thesis provides information about the company, Chapter 3 describes about the project assignment in detail, Chapter 4 explain about methodology, Chapter 5 discusses the whole research process to reach the goal, and ended with Conclusion and Evaluation on Chapter 6. Is it possible to create such kind of surround room? Sit back, relax, and discover the answer on the following pages.

CHAPTER 2: THE COMPANY

2.1 Philips History

The foundations of Philips were laid in 1891 when Anton and Gerard Philips established Philips & Co. in Eindhoven, the Netherlands. The company begun manufacturing carbon-filament lamps and by the turn of the century, had become one of the largest producers in Europe. By 1910, with 2,000 employees, Philips was the largest single employer in The Netherlands.

Stimulated by the industrial revolution in Europe, Philips' first research laboratory was established in 1914 and the company started introducing its first innovations in the x-ray and radio technology. Over the years, the list of inventions has only been growing to include many breakthroughs that have continued to enrich people's everyday lives.

2.2 Philips Today

Moving into the 21st century, Philips has continued to change and grow. Long aware that for many people it is no more than a consumer electronics producer, it has dedicated itself to projecting a new and more representative image that reflects the products it offers in the areas of Healthcare, Lifestyle and Technology. By following this up in 2004 with a massive advertising campaign to unveil its new brand promise of 'sense and simplicity', the company confirmed its dedication to offering consumers around the world products that are advanced, easy to use and, above all, designed to meet their needs.

In September 2006, Philips sold 80.1% of its Semiconductors business to a consortium of private equity partners. This laid the foundation for a strong and independent new semiconductors company, named NXP, which was founded on the heritage of over 50 years of innovation at the heart of Philips. The sale marks a further milestone in our shift from cyclical activities to the building of an even stronger company, focused on Healthcare, Lifestyle and Technology, and centered around our brand promise of 'sense and simplicity'.

In September 2007, Philips communicated its Vision 2010 strategic plan to further grow the company with increased profitability targets. As part of Vision 2010, the organizational structure was simplified per January 1, 2008 by forming three sectors: Healthcare, Lighting and Consumer Lifestyle. These steps further position Philips as a market-driven, people-centric company with a strategy and a structure that fully reflect the needs of its customer base. Through Vision 2010, Philips expects to more than double EBITA per share by 2010 (compared to 2007 expected performance)

2.3 Branding

Whilst the logo of the company has been consistent since the1930s the way in which Philips has advertised and communicated to the outside world has varied. In general, until the mid-1990s all advertising and marketing campaigns were carried out at product level on a local market basis. This led to many different campaigns running simultaneously, not giving a global representation of Philips as a global company.

To establish consistent global presence, in 1995 Philips introduced the first global campaign in 1995 under the tagline “Let’s make things better”. This theme encapsulated the “One Philips” thinking and was rolled out globally in all markets and on all Philips products. This was also the first campaign that bought the whole company together, giving the employees a sense of belonging and providing a unified company look for an external audience.

In September 2004, Philips launched its “sense and simplicity” brand promise, which marked a new way forward for the company. “Sense and simplicity” reflects Philips’ commitment to be a market-driven company that provides products and services that fulfill the promise of being “designed around you, easy to experience and advanced”.

In 2008, the total estimated value of Philips brand increased by 8% to USD 8.3 billion and was ranked the 43rd most valuable brand in Interbrand’s 2008 ranking of best global brands.

2.4 Mission

Improving people’s lives through meaningful innovation

2.5 Vision

At Philips, we strive to make the world healthier and more sustainable through innovation. Our goal is to improve the lives of 3 billion people a year by 2025. We will be the best place to work for people who share our passion. Together we will deliver superior value for our customers and shareholders.

2.6 Philips Research

Philips Research as a global organization grew out of the renowned “Natuurkundig Laboratorium” or NatLab which was established in 1914 in downtown Eindhoven. The NatLab pioneered a number of breakthrough inventions, which contributed to the successful expansion of Philips into a multinational company. These included the pentode radio tube, the rotary shaver, and the compact disc. In the early 1960s, Philips Research moved to its current site in Waalre, where it expanded into IC technology, software, andsystems.

In 2000, Philips decided on a new destiny for the Research labs' site. The barriers were removed and the High Tech Campus Eindhoven was established, which has since become the largest Dutch high tech campus. It is a truly vibrant community and ecosystem for innovation. Philips Research in Eindhoven is well positioned as a cornerstone of the campus.

The former Philips Applied Technologies group was created by a merger of several Philips departments in 2005, including the Center for Manufacturing Technology and the Philips Digital Systems Lab of Philips Consumer Electronics. It moved to the High Tech Campus in 2006.

In January 2011, a large part of the Applied Technologies organization merged with Philips Research, thereby enhancing its technical competency base, and its ability to bring technology-enabled innovation closer to the market. At this time, first of a kind product development skills and advanced engineering competencies were added to Philips Research.

2.7 Philips Applied Sensor Technologies

This department has it's roots in Philips Apptech, and was formed in June 2010 through the combination of teams with background in Industrial Vision, Sensor Technology, and A/V Signal Processing.

As the department name indicates, our current focus is on the development and integration of technologies for smart sensor solutions and related applications. To this end, we unite our expertise in sensor and camera system architecture, signal and image processing and HW/SW integration to create robust solutions, prototypes and (pre)products for our business partners.

Examples are smart lighting solutions, such as Lumimotion system for Philips Lighting, and integrated sensor solutions for vital signs monitoring. Next to that, we have considerable expertise in the area of standardization and IP value creation, activities that we carry out with and for our Philips colleagues in IP&S.

2.7.1 Mission

Enabling Solutions in the area of Responsive Environments and Smart Sensor Systems, through the combination of sensing technology, signal processing, and system architecture & integration.

2.7.2 Vision

We develop technologies and solutions to sense and process sensory signals in the area of vital body signs, activity, posture & motion, vision, audio & video. Our solutions contribute to the quality of daily life, health & well-being, and safety & security, and help our partners to successfully innovate and develop profitable business.

2.7.3 Department Competence

Key Enablers (Core Competences):

- Sensor and Vision Technology & Architecture
- Signal processing: signal understanding and algorithms
- Efficient and robust implementation of SP solutions
- Application know-how & system integration

CHAPTER 3: ASSIGNMENT OVERVIEW

3.1 Initial Situation

Philips Research - Information and Cognition - Applied Sensor Technologies division is a group of experts who entirely focusing on sound and sensor processing. Alongside with Android's hype, they start to design and implement several software such as TTS (Text-to-Speech, a smart application which able to spell the words that user has written) and MPEG Surround decoder (Sound processor application which simulate surround sound virtually on user's speaker device). Further on MPEG Surround decoder, they have a certain room near the office in purpose of demonstrating their decoder application. What they want to build next in order to extend their project is a visualization based on it; a 3D room that represents similar demonstration on Android device.

Philips also interested in exploiting Android devices as they're developer friendly and also built in open source environment. There are several Philips' applications which already published to Android Market with two main intentions; to promote and to extend the functionality of their products. For instance, we can find Philips TV Buying Guide app on the Android Market which virtually render their TV product on your gadget's camera so you can estimate which TV to buy and how to place it properly inside your house. There also an application called Philips My Remote that could help you to simulate a TV remote via mobile devices.

3.2 Objectives

Philips Research - Information and Cognition - Applied Sensor Technologies division consider this opportunity to develop a new application based on their existing MPEG surround application. The main idea is to give a mini simulation on how virtual surround works without having a real demonstration, simply to make a portable one. The goals would be a 3D room application coupled with surround sound behind it.

3.2.1 Command description

The very first requirements declared the boundaries:

- An OpenGL-based desktop application, then port into an Android Application.
- I have to work only on the graphic processing (3D room) maximize the capability of Open GL ES.
- The visualization must be represented a room with 6 speaker (a pair of front, center, and rear speakers)
- Camera position has to be moved using the accelerometer sensor embedded in almost every Android device.
- Built on Android 2.2 version environment.
- Testing on LG Optimus 2X device, and also Acer Iconia A500 tablet if needed.

3.2.2 Additional Commands

After several weeks of working, I had some additional requirements to do. Here's the list:

- Checking the source code into subversion and filling Philip's Wiki page.
- Managing a presentation for the first application build.
- Importing Native Code.
- Attaching soundtrack behind the visualization.
- Providing the angle value of camera scene for further development.
- Doing performance testing and profiling.
- Optimizing the source code.

3.3 Tools

1. **JDK (Java Development Kit)**
   Providing Java Environment inside Operation System.
2. **Eclipse IDE (Integrated Development Environment)**
   One of the compilers integrated with Android Development.
3. **OpenGL and OpenGL ES**
   Graphic Library.
4. **Android Environment**
   Android SDK (Software Development Kit) and Android NDK (Native Developing Kit).
5. **Min3D**
   Framework for building 3D object.
6. **Cygwin**
   Simulate Unix commands on Windows.
7. **Adobe Photoshop**
   Manipulating picture as texture.

CHAPTER 4: METHODOLOGY

Behind every success there's always a flawless plan. In terms of football or war it's called strategy, but here we will discuss about the strategy of researching. **Methodology** is generally a guideline system for solving a problem, with specific components such as phases, tasks, methods, techniques and tools. It must be understood and be prepared well before we take any step of doing research, or otherwise we will play like a clueless footballer or fight like a brainless soldier.

In this project I have to deal with 2 methodologies: developing software and doing research. In following sections I will give brief explanations about methodologies to be used in this project.

4.1 Developing Software with Waterfall Methodology

A classically linear and sequential approach to software design and systems development, each waterfall stage is assigned to a separate team to ensure greater project and deadline control, important for on-time project delivery. A linear approach means a stage by stage approach for product building, for example:

1. The project team first **analyses**, then determining and prioritizing requirements / needs.
2. Next, in the **design phase** requirements are translated into IT solutions, and a decision taken about which underlying technology i.e. COBOL, Java or Visual Basic, etc. etc. is to be used.
3. Once processes are defined and online layouts built, code **implementation** takes place.
4. The next stage of data conversion evolves into a fully **tested** solution for implementation and testing for evaluation by the end-user.

The last and final stage involves **evaluation** and **maintenance,** with the latter ensuring everything runs smoothly.

4.2 Research Paradigms and Methods

Before deciding what method will be used we have to examine our research paradigm first; is it experimental or analytical. The experimental paradigm is an inductive approach; it uses empirical evidence to formulate generalized knowledge from the observations made. On the other hand the analytical paradigm is deductive; it takes known theories and provides new evidence for special cases.

The experimental paradigm includes the scientific method with engineering method and empirical method as subsets, while the analytical paradigm includes the mathematical method:

1. Scientific method: Observe the world, propose a model or a theory of behavior, measure and analyze, validate hypotheses of the model and theory, and if possible repeat the procedure.
    - Engineering method: Observe existing solutions, propose better solutions, build or develop, measure and analyze, and repeat the process until no more improvements appear possible.

- Empirical method: Propose a model develop statistical/qualitative methods, apply to case studies, measure and analyze, validate model and repeat the procedure.

2. Mathematical method: Propose a formal theory or set of axioms, develop a theory, derive result and if possible compare with empirical observations.

My research mainly discusses about developing new software. Hence, It's very likely to use Engineering method. It is suitable and usually works well along with waterfall methodology. For gathering data I used Unobtrusive methods by observing, analyzing documents, screens, web site etc, because with the wideness of information available on internet we can dig many advantageous sources for our research.

CHAPTER 5: RESEARCH

5.1 Virtual Surround Room demonstration.

Virtual surround is an audio system which attempts to create the perception that there are many more sources of sound than which are actually present. In order to achieve this it is necessary to devise some means of tricking the human auditory system into thinking that a sound is coming from somewhere that it is not. Most recent examples of such systems are designed to simulate the true (physical) surround sound experience using one or two loudspeakers. Such systems are popular among consumers who want to enjoy the experience of surround sound without the large number of speakers that are traditionally required to do so.

At the first day here, I got a unique demonstration about virtual surround from my colleague Jeroen. In a room wrapped by wood texture, there is a big set of approximately 12 speakers around. I had to sit in the center on the room and tilt my head to the proper direction. Jeroen started to play the sound. The first source of the sound is the speakers. I can clearly feel that the sound came from several directions: front left, front right, center, back left, and back right. Then the second simulation began. I wore a big headset which made me hardly catch any sound that come from outside. Jeroen played a sound through the headset; with particular surround decoder, and I got the same feeling compared to the first simulation; same direction, and the same assumption that the sound was coming from the speakers. What a brilliant simulation. Very simple, understandable, and gave me enough provision on my backpack to step on the next phase.

5.2 Define the first project requirement

a. An openGL 3d scene with 3 standing speakers on the right and left sides of the walls
b. Desktop version (with basic OpenGL)
c. Port it to mobile app (android)
d. Add free movement feature

5.3 First Research: OpenGL and start making the prototype

My mentor already gave me some references about OpenGL, long before the project started. That helped me enough to absorb some ideas in graphic processing. For the initial prototype, my mentor didn't want me to start nowhere from the scratch. She gave me a package of a running OpenGL application with code. I always love to start something in this kind of way because I realized that I will learn quickly from a valid example, rather than reading a bunch of basic literatures. The example application itself represents a plain room with moveable light source.

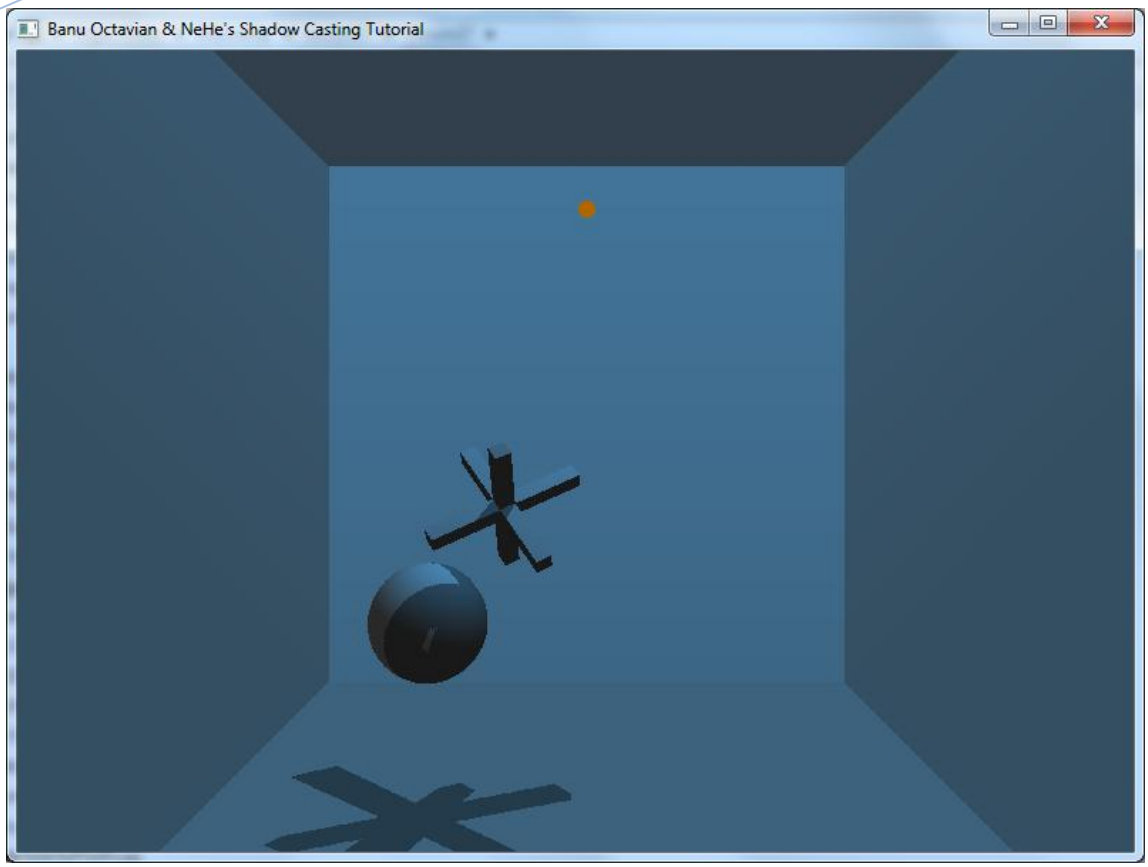


Figure 1. Very First Desktop Prototype

Somehow I had to manipulate it according to requirement. Before I went deeper into the code, I tried to do some research first and discovered several fundamental functions of OpenGL. Instead of adding standing speakers, I made 6 hanging speakerboxes using GLUT library. It's a common OpenGL library which contains several primitive objects like cube, cone, etc.

5.4 Next Research: OpenGL ES (Android)

After creating a raw prototype of 3D room (texture-less wall and speakers), I was struggling to find the best way for porting it into android application. Hence, I collected some similar examples that represent a 3D room generally.

5.4.1 Build a 3D view

It is a lair-like 3D room, with weird textures on the wall. I turned the wall into woody one, just for adding some aesthetic values. We can navigate freely with direction pad and also tilting the view around with some tidy touches. Exactly not a bad example to start since it contains several features which matching my requirements.

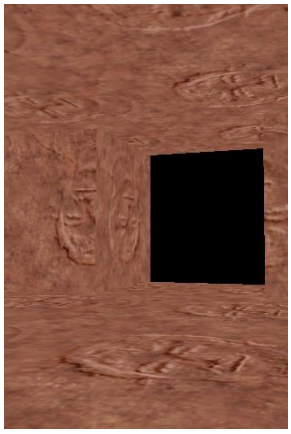


Figure 2. A Lair-like 3D room using OpenGL ES

5.4.2 Rendering a Speaker Box

I tried to gather common basic shape for my speaker. In order to get wider scenery, I swapped the screen orientation into landscape and set it as default. With a little modification and proper set up, this is my first prototype ever.

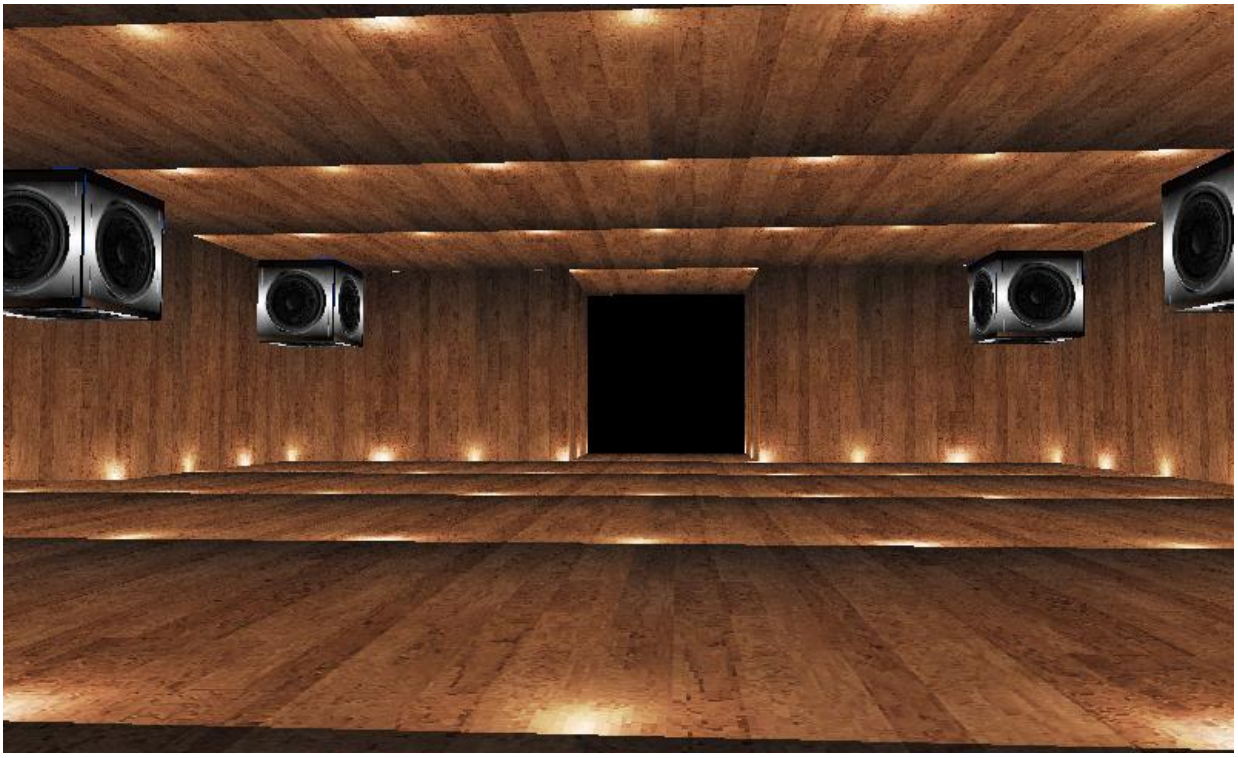


Figure 3. First Android Prototype

It's actually has 3 speakers each on both left and right side. I'm using **glTranslate()** function to determine the exact position of each object. I also attached speaker textures on each box in order to make it look like a real speaker. This application consists of 4 classes:

1. App : manage the whole visualization.
2. Cube : object representation of a cube containing vertex information, texture coordinates, vertex indices and drawing functionality, which is called by the renderer.

3. World : textual properties of the main 3D room.
4. Run : Initiation of Android Activity, setting and openGL ES Renderer class.

onTouchListener() method from Android API has been implemented so we can take a look around inside the scene via screen touch. Forward and backward movement had been set using Directional Pad by default, but nowadays almost all of Android devices don't have physical direction button. To deal with it, I ported movement function into menu button, search button, and also volume keys.

5.5 Designing Sensor Navigation

After reaching all of the first requirements, I had to assign a completely new navigation system. The view had to be changed based on the phone movement to bring more experience on the visualization. Basically Android device has some sensor to be exploited:

Constants		
int	TYPE_ACCELEROMETER	A constant describing an accelerometer sensor type.
int	TYPE_ALL	A constant describing all sensor types.
int	TYPE_AMBIENT_TEMPERATURE	A constant describing an ambient temperature sensor type
int	TYPE_GRAVITY	A constant describing a gravity sensor type.
int	TYPE_GYROSCOPE	A constant describing a gyroscope sensor type
int	TYPE_LIGHT	A constant describing an light sensor type.
int	TYPE_LINEAR_ACCELERATION	A constant describing a linear acceleration sensor type.
int	TYPE_MAGNETIC_FIELD	A constant describing a magnetic field sensor type.
int	TYPE_ORIENTATION	*This constant is deprecated. use `SensorManager.getOrientation()` instead.*
int	TYPE_PRESSURE	A constant describing a pressure sensor type
int	TYPE_PROXIMITY	A constant describing an proximity sensor type.
int	TYPE_RELATIVE_HUMIDITY	A constant describing a relative humidity sensor type.
int	TYPE_ROTATION_VECTOR	A constant describing a rotation vector sensor type.
int	TYPE_TEMPERATURE	*This constant is deprecated. use `Sensor.TYPE_AMBIENT_TEMPERATURE` instead.*

Figure 4. Various Types of Android Sensor

Considering that we had to have a head-tilt like movement, and because it is the most popular sensor around (with a lot of example on the Internet), I tried to exploit accelerometer first. Here's a short brief about it:

Sensor.TYPE_ACCELEROMETER:

All values are in SI units (m/s^2)

values[0]: Acceleration minus Gx on the x-axis

values[1]: Acceleration minus Gy on the y-axis

values[2]: Acceleration minus Gz on the z-axis

A sensor of this type measures the acceleration applied to the device (**Ad**). Conceptually, it does so by measuring forces applied to the sensor itself (**Fs**) using the relation:

Ad = - ∑Fs / mass

In particular, the force of gravity is always influencing the measured acceleration:

Ad = -g - ∑F / mass

For this reason, when the device is sitting on a table (and obviously not accelerating), the accelerometer reads a magnitude of **g** = 9.81 m/s^2

Similarly, when the device is in free-fall and therefore dangerously accelerating towards to ground at 9.81 m/s^2, its accelerometer reads a magnitude of 0 m/s^2.

It should be apparent that in order to measure the real acceleration of the device, the contribution of the force of gravity must be eliminated. This can be achieved by applying a *high-pass* filter. Conversely, a *low-pass* filter can be used to isolate the force of gravity.

Examples:

- When the device lies flat on a table and is pushed on its left side toward the right, the x acceleration value is positive.
- When the device lies flat on a table, the acceleration value is +9.81, which correspond to the acceleration of the device (0 m/s^2) minus the force of gravity (-9.81 m/s^2).
- When the device lies flat on a table and is pushed toward the sky with an acceleration of A m/s^2, the acceleration value is equal to A+9.81 which correspond to the acceleration of the device (+A m/s^2) minus the force of gravity (-9.81 m/s^2).

5.6 Discovering Graphic Engine

To get some insights on the sensor I looked for few tutorials out there. It wasn't that hard to find an example which coupled 3D scene with sensor navigation. Like catching two fishes in a swipe, I also got a quite comprehensive OpenGL ES Library called **Min3D**.

Min3D is a lightweight 3d library/framework for Android using Java with OpenGL ES targeting compatibility with Android v1.5/OpenGL ES 1.0 and higher. It tracks closely with the OpenGL ES API, which makes it ideal for gaining an understanding of the OpenGL ES API while providing the convenience of an object-oriented class library. It's very easy to render various of primitive objects like box,sphere,pyramid etc.

Min3D is able to import 3 different filetypes:

- Wavefront OBJ (text)
- 3DS (binary)
- MD2 (binary, animation)

Figure 5. Min3D Object Example

5.7 Min3D Setup and Implementation

1. Download and extract Min3D library file inside your project folder.
2. Open your project on Eclipse
3. Refresh your workspace. You will find a folder named min3d
4. Right click on the folder, choose Build path > Use as source folder.
5. Now you can call any method from min3d library

There are 3 related stuffs that could be easily implemented with this library:

1. A 3D room. In the world of 3D graphic processing we usually call it “Skybox”

First we have to declare a Skybox with (float size, int quality) parameter. Afterwards we can add textures on each side of the Skybox in to get real room experience. Method scale() determines the proportion of each axis. Then we can attach any texture on every side of the room with 1:1 width and length proportion.

Figure 6. Skybox

2. Primitive objects (Box, Hollow Cylinder, Rectangle, Sphere, Torus) also can be rendered using this library. Simply instantiate any primitive object we want and assign several parameter on it. For example, in this case I need 6 3D cubes for later being rendered as a hanging speakerbox.

We have to also attach texture into a plain 3D object to bring a real impression. Texture image must be placed inside **res/drawable** directory and later will be called using, for example in my case I named my texture as **speaker1**, `R.drawable.speaker1` ID.

Figure 7. Speaker Texture

Every images and texture which been loaded into the scene need to be recycled, just in case the environment get sort of memory. recycle() will free up some memories as soon as possible.

Figure 8. Skybox with Speakers Attached

3. Camera exploitation. Min3D also provides ease of camera management. With sensor integration we can tilt the whole scene camera along with the device's movement. First thing to do is we need to override built-in onSensorChanged() method, then pass the event value into the camera object.

Notice that I only use y and z axis, because the scene was set on landscape by default to gain wider scenery.

Min3D assembled a class called Number 3D which being instantiated as `mAccVals`. It contains arrays of numbers which useful for building a dynamic 3D view.

Figure 9. Camera Exploitation within Sensor

5.8 Rendering Particular 3D Object

There is a feature in Min3D which is too decent not to be implemented; we can bring our own 3D object into the screen. It is possible for almost all 3D models with .OBJ, .MD2 and .3DS extension.

To give life for the surround room, I decided to add 2 3D objects; an acoustic guitar and a set of home theater equipment as furnishes. You can find a lot of well-made 3D object at 3divia.com . It's better to go with .3DS object since it's easier to be rendered.

5.8.1 How to Add Premade 3D Object (.3DS)

1. Try to find any suitable 3D object with .3DS extension along with texture, or if you have any knowledge on 3D rendering you can also create it by yourself. This home theatre set is just perfect as an example.

Figure 10. 3D Object Example

Figure 11. Texture Example

2. Make sure that the .3DS file name meets the requirement. It must contain only **[a-z0-9_.] .** Let's give it a name **stereo.3ds .** You can leave the texture name as it is; **stereo.jpg**.

3. Put the file on the proper directory inside your workspace. Moreover, both of the must be placed inside **res** folder.
    - For future easier access, .3ds file should be in directory **res/raw.** Sometimes this folder already generated automatically after starting a new workspace and sometimes you have to create it manually.
    - There is a lot of **drawable** folders inside **res.** You just have to put the texture file inside any of them, because later the code will automatically find it.
4. Import the Object3dContainer and Parser libraries.
5. Declare a new object from Object3dContainer. Assume that the name is mTv.
6. Now we can import the .3ds file
7. Moreover we can also set several properties of skybox i.e. :
    - managing object's scale
    - managing object's position
    - managing object's rotation
    - x, y, z axis can be managed depends on which axis you want to use.
8. Finally, we can add the `mTv` object to the scene.

Figure 12.First Scenery Design

5.9 Defining New Requirement: Add Soundtrack

After finishing the first part of the application, I was obligated to add sounds behind the scene. The sound itself must be in .WAV or .MP4 extension. After spent a day of research, I found a way to deal with the sound:

1. Adding menu items.
2. Create a simple list that display all file inside the SD Card (file explorer).
3. Filtering .WAV and .MP4 file.
4. Play the sounds directly from the explorer.

Previous step, to make the app more "User-Friendly" and accessible, I managed to assign 3 menu items for better navigation. It would consist of:

1. Exit : completely close the application's process.
2. About : Application's properties.
3. Select Sound : later will be used to jump into the file explorer.

Later menu can be activated via device's menu button.

5.9.1 Generating Menu items

Android phones have the menu button which displays a menu with several items that provide navigation or more functionality or settings to your applications.
Android has three types of menus

1. Options menu.
2. Context menu.
3. Alternative menus.

Menu items can be grouped and each menu item can have submenu items. Options menu will be implemented since it is particularly needed at the moment.

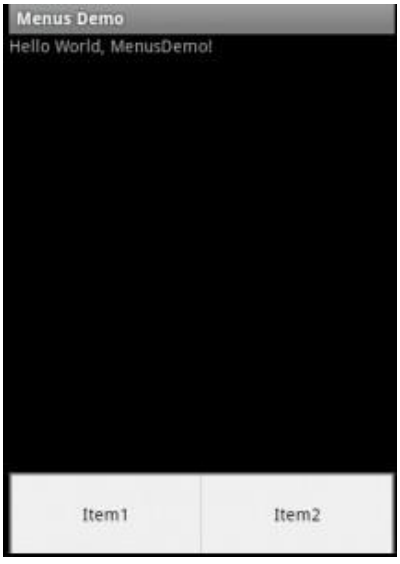


Figure 13. Android Option Menu Example

To generate option menus properly there are two parts of separated code:

1. Java code inside the main class
   Android API provides **`onCreateOptionsMenu(Menu menu)`** which is can be overrided anytime.

2. The whole method has to be added to the main class. Meanwhile, there's also some code that need to be implemented as xml file to trigger the menu items. This code located inside

the **res/layout** directory. Assume that we call it **menu.xml.** Then we should call the layout and pass it to menu **inflater** as `R.layout.menu.`

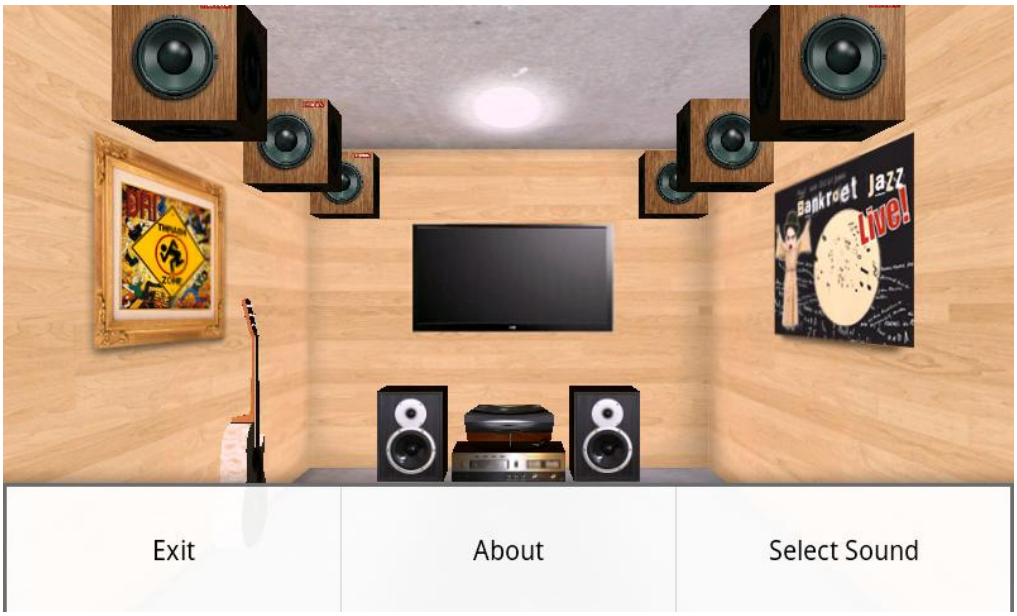


Figure 14. Menu Items

This menu item feature is declared as the most effective way of navigation. Almost all of android applications around are optimizing this feature. Later, menu item can be furnished by adding image into each item.

5.10 Philips MPEG Surround Decoder

Building a simple file explorer will be the next step afterward. I was thinking to make it from scratch, but after further discussion with my mentor, she agreed to handle the source code of her **Philips' MPEG Surround Decoder** Android application.

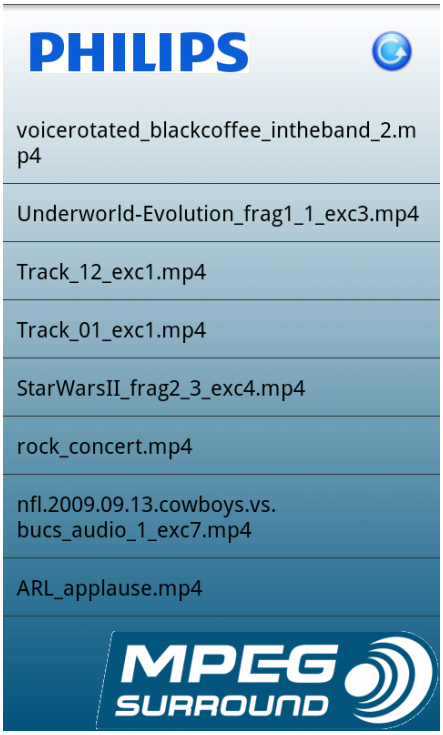


Figure 15. Philips' MPEG Surround Decoder

This is a kind of media player app which simulates surround effect on particular media file (.mp4 and .wav). It is covered almost all my plans before; simple file list explorer, filtering .wav and .mp4 files. MPEG Surround application has 2 usable button; Song repeat (on the top right corner) and MPEG switcher (on bottom side). In order to produce unique and original sound output, the application uses certain decoder formed by native sound libraries in C/C++.

5.10.1 Filtering and Listing .WAV and .MP4 File

The application will only play .WAV and .MP4 file extensions, which means that files with any other formats won't be recognized. It is possible to create an explorer that displaying all the files inside the external mobile phone storage but of course it is not effective because we have to find needed files manually through directories. One of the shortest ways is filtering the file inside certain directory. Thus, appropriate files will be gathered in a list at once.

Based on http://docs.oracle.com:, Java has a **FilenameFilter** Interface. Instances of classes that implement this interface are used to filter filenames. These instances are used to filter directory listings in the list method of class File. It must be coupled with **accept()** method.

Moreover, to collect filtered class into a list, `Mp4Filter` class implemented further in **`updateSongList()`** method.

`updateSongList()` method will put the files into an array. Furthermore, if there's a new file inserted to /sdcard the method will recheck and the updating the list. Hence, **`updateSongList()`** is essential to be repeated so we have to put it inside the **`public void onCreate(Bundle icicle)`** which means **`updateSongList()`** will be executed on every application startup.

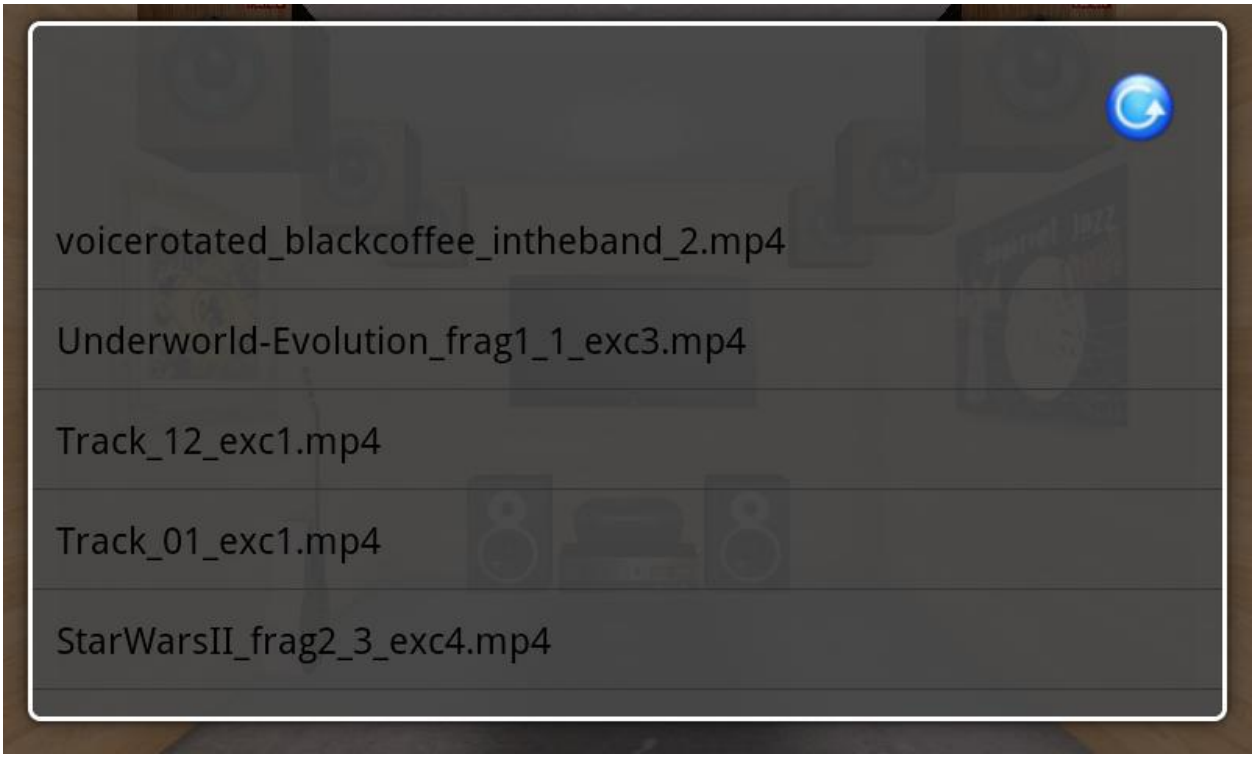


Figure 16. Simple File List Design

5.11 Go Native with the Sound Decoder

MPEG Surround Decoder still use Java Android media playback as the music player. In the context of producing virtual surround sound, media playback must be supported by specific decoder. The decoder had been written as a native code (C/C++). Thus, native library need to be imported into Android SDK so it can be ported to Android. The Android framework provides two ways to use native code:

1. One it's to write your application using the Android framework and use JNI (JavaTM Native Interface (JNI)) to access the APIs provided by the Android NDK. This technique allows you to take advantage of the convenience of the Android framework, but still allows you to write native code when necessary. You can install applications that use native code through the JNI on devices that run Android 1.5 or later.

2. You can also write a native activity, which allows you to implement the lifecycle callbacks in native code. The Android SDK provides the *NativeActivity* class, which is a convenience class that notifies your native code of any activity lifecycle callbacks (onCreate(), onPause(), onResume(), etc). You can implement the callbacks in your native code to handle these

events when they occur. Applications that use native activities must be run on Android 2.3 (API Level 9) or later.

Because I only have a test device running Android 2.2, inevitably I had to take the first option, since it's compatible with devices that run Android 1.5 or later.

5.11.1 Java Native Interface

The **Java Native Interface** (JNI) is a programming framework that enables Java code running in a Java Virtual Machine (JVM) to call and to be called by native applications (programs specific to a hardware and operating system platform) and libraries written in other languages such as C, C++ and assembly. JNI enables one to write native methods to handle situations when an application cannot be written entirely in the Java programming language, e.g. when the standard Java class library does not support the platform-specific features or program library. It is also used to modify an existing application—written in another programming language—to be accessible to Java applications. Many of the standard library classes depend on JNI to provide functionality to the developer and the user, e.g. file I/O and sound capabilities. Including performance- and platform-sensitive API implementations in the standard library allows all Java applications to access this functionality in a safe and platform-independent manner.

The JNI framework lets a native method use Java objects in the same way that Java code uses these objects. A native method can create Java objects and then inspect and use these objects to perform its tasks. A native method can also inspect and use objects created by Java application code.

JNI is sometimes referred to as the "escape hatch" for Java developers because it enables them to add functionality to their Java application that the standard Java APIs cannot otherwise provide. It can be used to interface with code written in other languages, such as C and C++. It is also used for time-critical calculations or operations like solving complicated mathematical equations, because native code may be faster than JVM code.

5.11.2 Android NDK

The Android NDK is a toolset that lets you embed components that make use of native code in your Android applications.

Android applications run in the Dalvik virtual machine. The NDK allows you to implement parts of your applications using native-code languages such as C and C++. This can provide benefits to certain classes of applications, in the form of reuse of existing code and in some cases increased speed.

The NDK provides:

- A set of tools and build files used to generate native code libraries from C and C++ sources
- A way to embed the corresponding native libraries into an application package file **(.apk)** that can be deployed on Android devices

- A set of native system headers and libraries that will be supported in all future versions of the Android platform, starting from Android 1.5. Applications that use native activities must be run on Android 2.3 or later.
- Documentation, samples, and tutorials

5.11.3 When to Develop in Native Code

The NDK will not benefit most applications. As a developer, you need to balance its benefits against its disadvantages; notably, using native code does not result in an automatic performance increase, but always increases application complexity. In general, you should only use native code if it is essential to your application, not just because you prefer to program in C/C++.

Typical good candidates for the NDK are self-contained, CPU-intensive operations that don't allocate much memory, such as signal processing, physics simulation (both of this categories meet the virtual surround decoder), and so on. Simply re-coding a method to run in C usually does not result in a large performance increase. When examining whether or not you should develop in native code, think about your requirements and see if the Android framework APIs provide the functionality that you need. The NDK can be an effective way to reuse a large corpus of existing C/C++ code.

JNI also contains pitfalls as well as Android NDK. According to Wikipedia, these are several points to be concerned:

- Subtle errors in the use of JNI can destabilize the entire JVM in ways that are very difficult to reproduce and debug.
- Only applications and signed applets can invoke JNI.
- An application that relies on JNI loses the platform portability Java offers (a partial workaround is to write a separate implementation of JNI code for each platform and have Java detect the operating system and load the correct one at runtime).
- The JNI framework does not provide any automatic garbage collection for non-JVM memory resources allocated by code executing on the native side. Consequently, native side code (such as assembly language) must assume the responsibility for explicitly releasing any such memory resources that it itself acquires.
- Error checking is a must or it has the potential to crash the JNI side and the JVM.

5.11.4 What to Prepare Before Porting C/C++ code into Android

5.11.4.1 C/C++ Eclipse Support Installation

The Eclipse installation does not support either C or C++ right now. We don’t really need the full support, including building etc. But we would like to have syntax coloring and basic syntax checking. Thus we have to add some Eclipse features via the update mechanism, almost like when we added Android support. Steps:

1. Go to “Help > Install New Software” menu item.
2. Choose the Eclipse version you have (Galileo, Indigo etc.) as the update site (“Work with”).
3. Check “Eclipse C/C++ Development Tools” in the Programming Languages branch, when the item tree loads.
4. Then press “Next”.
5. Say “Yes” to everything, accept the licenses and let Eclipse finish the update.
6. Once it is done, you will see the prompt to restart Eclipse. Say “Yes” and wait for Eclipse to restart.
7. You have C/C++ support in your Eclipse IDE now.

5.11.4.2 Cygwin

Android is Linux based, and thus it is no surprise that when you build native code for it, you need some Unix tools. On Windows, NDK supports Cygwin 1.7.x and above. If you don’t know what Cygwin is, it’s just a set of software that emulates Unix environment on Windows which is necessary in some cases, including ours. In order to get Cygwin, go to http://www.cygwin.com/.

Cygwin Installation Steps:

1. Download or/and run Cygwin’s setup.exe.
2. Choose “Install from Internet”, then click “Next”, then choose the installation directory. Note: The whole thing is going to require up to few of space. Choose the local package directory – just use some temporary directory, you’re not likely to need it afterwards.
3. At this point Cygwin will connect to its central site and “download” the list of mirror sites. Choosing a mirror site that looks geographically close to you may save you some download time and click “Next”.
4. We need the development packages. Click (once) on the word “Default” next to the root **Devel** node and wait few seconds while the setup hangs. When it is back, you will see that “Default” changes to “Install” for the Devel node.
5. Now click “Next”.

Extra steps to confirm the correct installation:

1. Launch the Cygwin console
2. Click it once, let the Cygwin console start up and initialize.
3. To check that we have the tool that is important for Android NDK, type **make -v** in the console.
4. You should see a response that tells us that GNU Make is present in our Unix environment that is emulated by Cygwin.
5. You have now installed Cygwin.

5.11.4.3 Android NDK Installation

Installation of NDK:

1. From the table at the top of this page http://developer.android.com/sdk/ndk/index.html, select the NDK package that is appropriate for your development computer and download the package.
2. Uncompress the NDK download package using tools available on your computer. When uncompressed, the NDK files are contained in a directory called android-ndk-<version>. You can rename the NDK directory if necessary and you can move it to any location on your computer, but it is recommended to move it to where you installed your Android SDK so that the ndk will be easy to found again. Make sure that the path name does not contain any spaces, this might prevent errors in the future. This documentation refers to the NDK directory as <ndk>.
3. You are now ready to start working with the NDK.

5.11.5 Building MPEG Surround Library

Make sure all of the preparation have completed (JNI, NDK, Cygwin), then we can go to the toughest part of the research. This step-by-step explanation need to be done in chronological order:

1. We need all the files that related to the library. In my case, I got a fully loaded "MPEGDecoder" folder consists of 3 subfolders with various decoder files inside. Put our workspace folder inside the MPEGDecoder directory so it will be easier to compile.
2. Create the Java class inside the project representing the native code. I embedded the native code inside the AudioControl.java.
   This is just a Java file that lives in standard src directory in your Eclipse project. It serves as the glue to the native code that we'll write later.
3. Create the native code header file for AudioControl

   a. Using Cygwin navigate to the bin directory of your project. Then run the javah tool. The javah tool should be located under "Program Files/Java/jdk- <version>/bin/javah.exe" so run it from there if necessary. The javah tool takes two command line arguments the first will be "-jni" for us, the second is the Java native file for which you wish to generate a header. Since you are in the bin directory you only need to type the package name for the second parameter. Once run, the JNI header file should appear in your project bin directory (in my case, <EclipseWorkspace>/ dpc_android_virtual_jni/bin).
      PS: If you willing to use directories which contain space, separate it with *
      Program Files → Program*Files

   b. Next, create a "jni" directory in your project directory
   c. And then, copy the JNI header to jni folder
   d. You now have a header file inside your JNI folder. Rename the file into MPSDecoderJNI.h

4. Implement the native code by writing your C code
   a. In your <EclipseWorkspace>/NDKDemo/jni/ folder, create a .c or .cpp file with the similar name as the header. This is where we'll implement the native code. To start, copy the function signatures from the header file, and provide the implementation for those functions. We also have to include any header that related too, in this case it was **decoder_agent.h** (I got both decoder_agent.h and decoder_agent.cpp from Philips MPEG Surround Application)
   b. Notice that the name of the C function is not just random – it matches the Java class name. In addition, what the function does is it uses the **JNIEnv** object to create a Java string from a literal, and returns the string to the caller.
   c. It is also possible to call Java methods from native code, create custom objects and so on.

5. Compile everything and build you Shared Library
   a. To build the library, first we need to create an Android.mk file. This is a file containing how to compile the C and also what C files included on compilation.
   b. From your project folder, issue a command that will invoke the NDK build tool. For example here, since NDK is installed in **C:** it looks like this: **/cygdrive/c/<path to the NDK>/android-ndk-r4/ndk-build**

Caution: Be careful not to leave any space at the end of each line in the Android.mk, because when you are going to try to do ndk-build, an error is going to show up: *"No rule to make target '/ndk_demo.c' …"*

   c. A successful run of the **ndk-build** tool will create a **.so** file in a new folder called **libs** that will be created in your project root. This is what it is displayed in the cygwin command line:

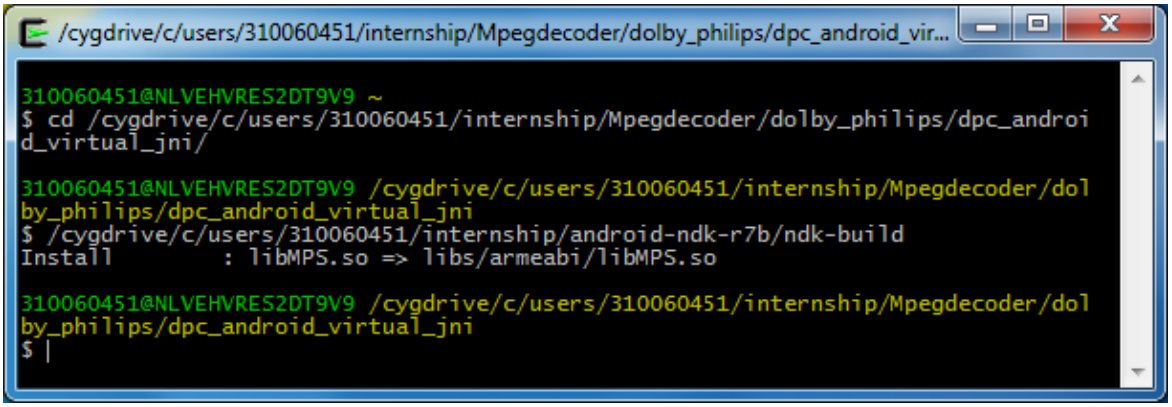


Figure 17. Successful Cygwin Command

   d. The **.so** file is the binary library that will be included into the application **.apk package** and will be available for the Java code of the app to link to.

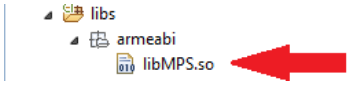


Figure 18. Where the Library File Was Generated

e. Make sure to hit **F5 (Refresh)** in Eclipse after selecting the project root to update the Eclipse project with the changes you did in the Cygwin console. If you run the project before refreshing none of the changes will take effect.

f. You have to repeat the **ndk-build** command every time you modify the C/C++ source of your NDK code. Eclipse ADT does not support NDK so you need to do it from the Cygwin console. **Don't forget to refresh your project in Eclipse every time!**

6. Use your native code inside Android activity

a. So now that we have the native C library implemented, compiled, and placed in the right place, let's see how we can call it from our Activity. It's actually rather simple - you just have to instantiate the instance of your NativeLib class and from there on, it's just a regular Java object.

b. After working with the native library, I set a menu items which will bring the user directly to MPS activity.

c. Last but not least, I had to make the sounds played behind the visualization room. To get rid of this problem, I added an "Intent" to bring the visualization back into the top of the stack immediately after choosing a song.

Figure 19. Implemented Philips MPEG Surround Decoder

5.12 Proper Room Setup

I worked with another several students, also on Android subject. We had an agreement that in every beginning of the month we have to present our project, no matter how the progress is. I encouraged myself to take the first presentation, since I felt my project was quite feasible to be presented.

I prepared a 10 minutes presentation and the rest will be an application demo on wide screen. There was an important remark on the visualization, said that **the audio setup was completely wrong.** I argued that I have no knowledge about surround room setup, but then I realized I have to concern more about it.

I tried to do some research on surround sound setup, and I found this helpful: http://www.the-home-cinema-guide.com . There are several clauses need to be fulfilled to gain a proper surround room perspective, for example, on 5.1 surround room:

1. The center speaker needs to be placed just above or below the center of the TV screen. Just try not to place it too far away from the screen or the sound may appear to be removed from the picture. This will sound unnatural and spoil the impact of the soundtrack. However, you can use any type of speaker as your center (such as a normal bookshelf speaker).

Figure 20. Recommended Front Speaker Placement

2. The front left and right speakers are the equivalent of the stereo pair you are probably used to with your hi-fi system. The front speakers should be an equal distance left and right of the TV - and both should be an equal distance from your seating position. You should imagine a triangle with the speakers and yourself on each corner.

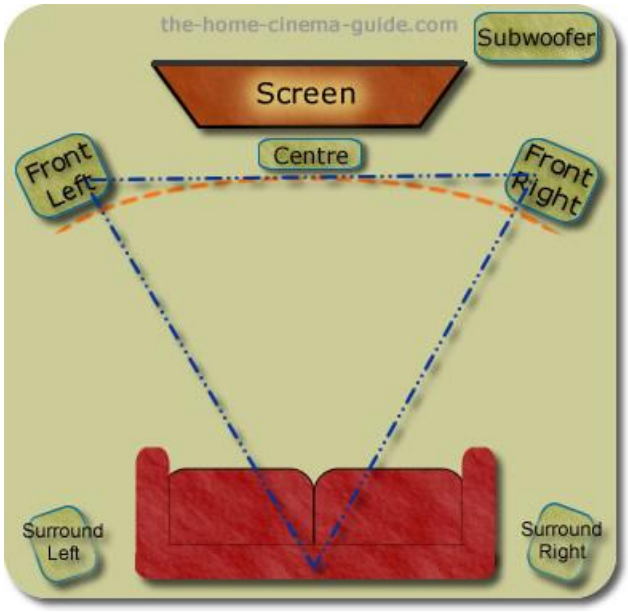


Figure 21. Front Left and Right Speaker Placement

3. Ideally, the rear surround speakers in a 5.1 surround sound configuration should be placed at the sides, and just behind, the seating position. In many rooms you may struggle to get these positioned exactly, but always try to get as close as your space will allow.

Figure 22. Rear Speakers Placement

4. Subwoofer has a very specific job, to reproduce the really low bass in a soundtrack. Therefore, subwoofer placement in a room is much less critical than with other speakers. Wherever you have a spare bit of space in your room then you can pretty much stick it anywhere.

Figure 23. Subwoofer Speaker Placement

5.13 Revising Room Setup

After finding some references, I decided to revise my room setup so at least it could resemble a proper surround room. Several points need to be considered:

1. Set of home theatre nearly represent the center, front left and also front right speakers so I can leave it as it is.
2. Six cube speakers around the room can be reduced into only one, later will be used as subwoofer.
3. Creating two standing speakers to fulfill rear speaker requirement.

5.13.1 Subwoofer Arrangement

As discussed previously, subwoofer can be stuck anywhere in the room. I precisely adopted Figure. and I shifted one of the cube speaker near the front left speaker by modifying the position.

Figure 24. New Front speakers and Subwoofer Arrangement

5.13.2 Standing Rear Speakers

Standing speaker could be created based on basic block shape, which is also can be reached by manipulating the scale of cube. Hence, I decided to turn 2 of cube speakers into standing one. The texture also need to be adapted so I got it edited using Photoshop.

Figure 25. Standing Speakers Texture

Figure 26. Rear Standing Speakers

5.14 Testing

Testing is always needed on every software development process. It is one of Software Development Cycle part that can't be left, if it's not the most important. We don't have a right to put a "Reliable" stamp on software before it passing several steps of testing. There are too many risks, or I might say it is harm to launch software without any systematic testing.

Software testing can be stated as the process of validating and verifying that a software program/application/product:

1. Meets the requirements that guided its design and development
2. Works as expected
3. Satisfied the Client

Software testing, depending on the testing method employed, can be implemented at any time in the development process. However, most of the test effort traditionally occurs after the requirements have been defined and the coding process has been completed.

5.14.1 Problem Assessment

There's some non-functional issues spotted after Implementation Phase:

1. Compatibility problem. Application will be crashed on startup if we install it on another version of Android. As mentioned before, an application that relies on JNI sometimes loses the platform portability Java offers.
2. Performance problem. Soundtrack is sometimes unstable, seems affected by the high load resource of 3D scene and Accelerometer.

The first problem seems unsolved. I've tried to force-install the application to another several Android versions, but it made the situation going worse instead of fix the problem. The application become hard to debug, sometimes the IDE finds unidentified errors without any explanations.

To get a deeper insight on the second problem, I did two ways of analysis; Profiling and Memory Analysis. Profiling covers the memory consumption of application itself, when memory analysis has a wider scope because it's also affected by the Android environment.

5.14.2 Profiling Activity

5.14.2.1 About Profiling

In software engineering, profiling ("program profiling", "software profiling") is a form of dynamic program analysis that measures, for example, the usage of memory, the usage of particular instructions, or frequency and duration of function calls. The most common use of profiling information is to aid program optimization.

5.14.2.2 DDMS Traceview Method

When you have a trace log file (generated by adding tracing code to your application or by DDMS), you can have Traceview load the log files and display their data in a window visualizes your application in two panels:

- A timeline panel -- describes when each thread and method started and stopped
- A profile panel -- provides a summary of what happened inside a method

The sections below provide addition information about the traceview output panes.

5.14.2.3 Timeline Panel

The image below shows a close up of the timeline panel. Each thread's execution is shown in its own row, with time increasing to the right. Each method is shown in another color (colors are reused in a round-robin fashion starting with the methods that have the most inclusive time). The thin lines underneath the first row show the extent (entry to exit) of all the calls to the selected method. The method in this case isLoadListener.nativeFinished() and it was selected in the profile view.

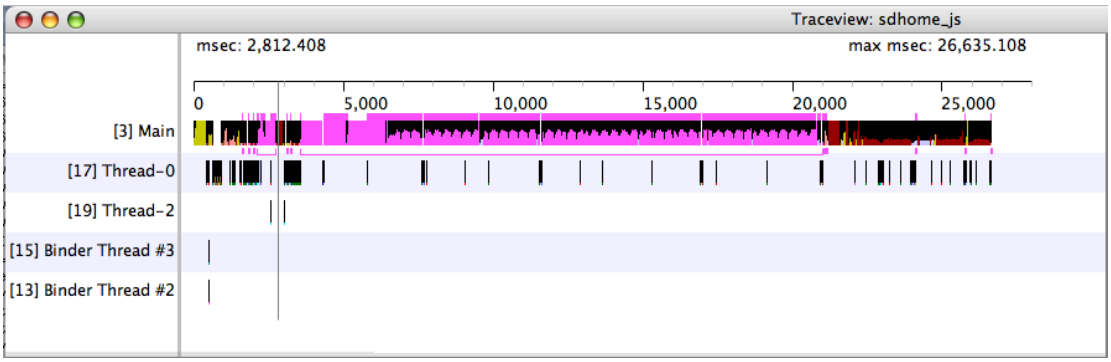
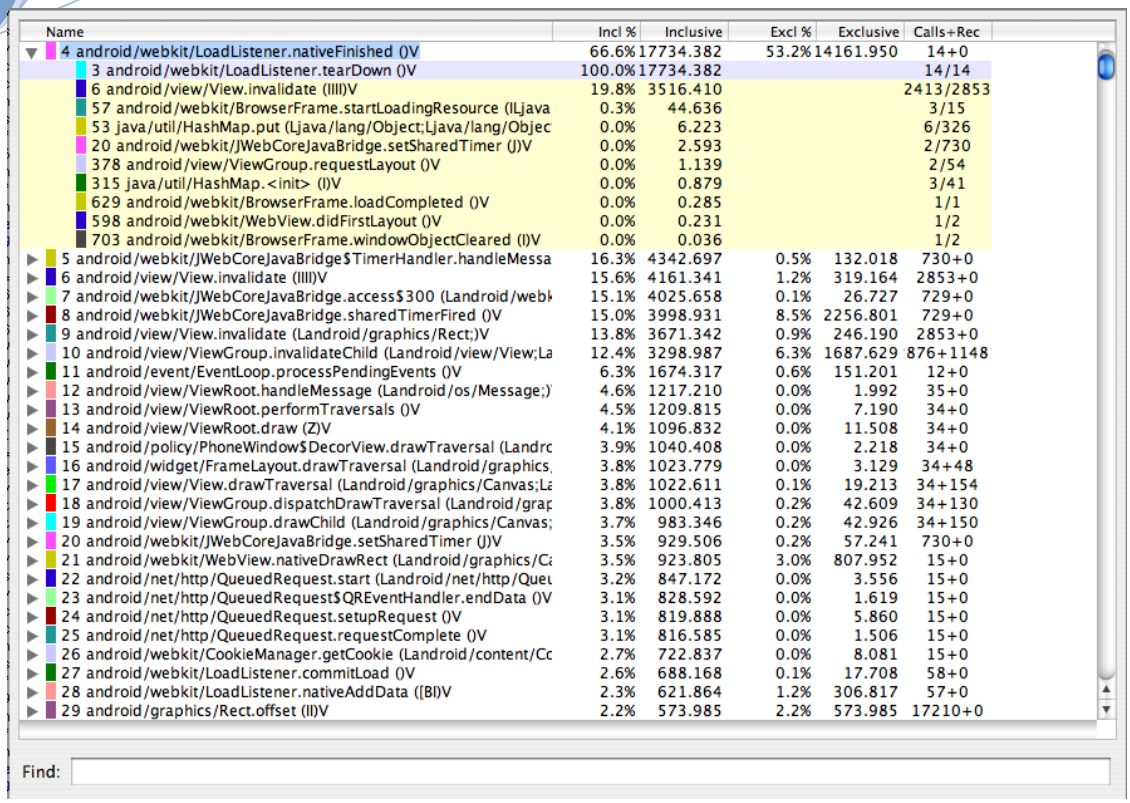


Figure 27. The Traceview Timeline Panel

5.14.2.4 Profile Panel

Figure 2 shows the profile pane, a summary of all the time spent in a method. The table shows both the inclusive and exclusive times (as well as the percentage of the total time). Exclusive time is the time spent in the method. Inclusive time is the time spent in the method plus the time spent in any called functions. We refer to calling methods as "parents" and called methods as "children." When a method is selected (by clicking on it), it expands to show the parents and children. Parents are shown with a purple background and children with a yellow background. The last column in the table shows the number of calls to this method plus the number of recursive calls. The last column shows the number of calls out of the total number of calls made to that method. In this view, we can see that there were 14 calls to LoadListener.nativeFinished(); looking at the timeline panel shows that one of those calls took an unusually long time.



Name	Incl %	Inclusive	Excl %	Exclusive	Calls+Rec
4 android/webkit/LoadListener.nativeFinished ()V	66.6%	17734.382	53.2%	14161.950	14+0
3 android/webkit/LoadListener.tearDown ()V	100.0%	17734.382			14/14
6 android/view/View.invalidate (IIII)V	19.8%	3516.410			2413/2853
57 android/webkit/BrowserFrame.startLoadingResource (ILjava	0.3%	44.636			3/15
53 java/util/HashMap.put (Ljava/lang/Object;Ljava/lang/Objec	0.0%	6.223			6/326
20 android/webkit/JWebCoreJavaBridge.setSharedTimer (J)V	0.0%	2.593			2/730
378 android/view/ViewGroup.requestLayout ()V	0.0%	1.139			2/54
315 java/util/HashMap.<init> (I)V	0.0%	0.879			3/41
629 android/webkit/BrowserFrame.loadCompleted ()V	0.0%	0.285			1/1
598 android/webkit/WebView.didFirstLayout ()V	0.0%	0.231			1/2
703 android/webkit/BrowserFrame.windowObjectCleared (I)V	0.0%	0.036			1/2
5 android/webkit/JWebCoreJavaBridge$TimerHandler.handleMessa	16.3%	4342.697	0.5%	132.018	730+0
6 android/view/View.invalidate (IIII)V	15.6%	4161.341	1.2%	319.164	2853+0
7 android/webkit/JWebCoreJavaBridge.access$300 (Landroid/webk	15.1%	4025.658	0.1%	26.727	729+0
8 android/webkit/JWebCoreJavaBridge.sharedTimerFired ()V	15.0%	3998.931	8.5%	2256.801	729+0
9 android/view/View.invalidate (Landroid/graphics/Rect;)V	13.8%	3671.342	0.9%	246.190	2853+0
10 android/view/ViewGroup.invalidateChild (Landroid/view/View;La	12.4%	3298.987	6.3%	1687.629	876+1148
11 android/event/EventLoop.processPendingEvents ()V	6.3%	1674.317	0.6%	151.201	12+0
12 android/view/ViewRoot.handleMessage (Landroid/os/Message;)	4.6%	1217.210	0.0%	1.992	35+0
13 android/view/ViewRoot.performTraversals ()V	4.5%	1209.815	0.0%	7.190	34+0
14 android/view/ViewRoot.draw (Z)V	4.1%	1096.832	0.0%	11.508	34+0
15 android/policy/PhoneWindow$DecorView.drawTraversal (Landro	3.9%	1040.408	0.0%	2.218	34+0
16 android/widget/FrameLayout.drawTraversal (Landroid/graphics,	3.8%	1023.779	0.0%	3.129	34+48
17 android/view/View.drawTraversal (Landroid/graphics/Canvas;La	3.8%	1022.611	0.1%	19.213	34+154
18 android/view/ViewGroup.dispatchDrawTraversal (Landroid/grap	3.8%	1000.413	0.2%	42.609	34+130
19 android/view/ViewGroup.drawChild (Landroid/graphics/Canvas;	3.7%	983.346	0.2%	42.926	34+150
20 android/webkit/JWebCoreJavaBridge.setSharedTimer (J)V	3.5%	929.506	0.2%	57.241	730+0
21 android/webkit/WebView.nativeDrawRect (Landroid/graphics/Ca	3.5%	923.805	3.0%	807.952	15+0
22 android/net/http/QueuedRequest.start (Landroid/net/http/Queu	3.2%	847.172	0.0%	3.556	15+0
23 android/net/http/QueuedRequest$QREventHandler.endData ()V	3.1%	828.592	0.0%	1.619	15+0
24 android/net/http/QueuedRequest.setupRequest ()V	3.1%	819.888	0.0%	5.860	15+0
25 android/net/http/QueuedRequest.requestComplete ()V	3.1%	816.585	0.0%	1.506	15+0
26 android/webkit/CookieManager.getCookie (Landroid/content/Cc	2.7%	722.837	0.0%	8.081	15+0
27 android/webkit/LoadListener.commitLoad ()V	2.6%	688.168	0.1%	17.708	58+0
28 android/webkit/LoadListener.nativeAddData ([BI)V	2.3%	621.864	1.2%	306.817	57+0
29 android/graphics/Rect.offset (II)V	2.2%	573.985	2.2%	573.985	17210+0

Find:

Figure 28. Profile Panel

5.14.2.5 CPU Usage per Profile and Screenshot

This screenshots were taken using DDMS Traceview on Eclipse. I managed to make 5 different profiles based on the existence of additional objects, object textures, MPEG Surround Features, and also wakelock function. All of this profile using the same Game Sensor Delay.

- Profile with addition objects, textures, full MPEG Surround Features

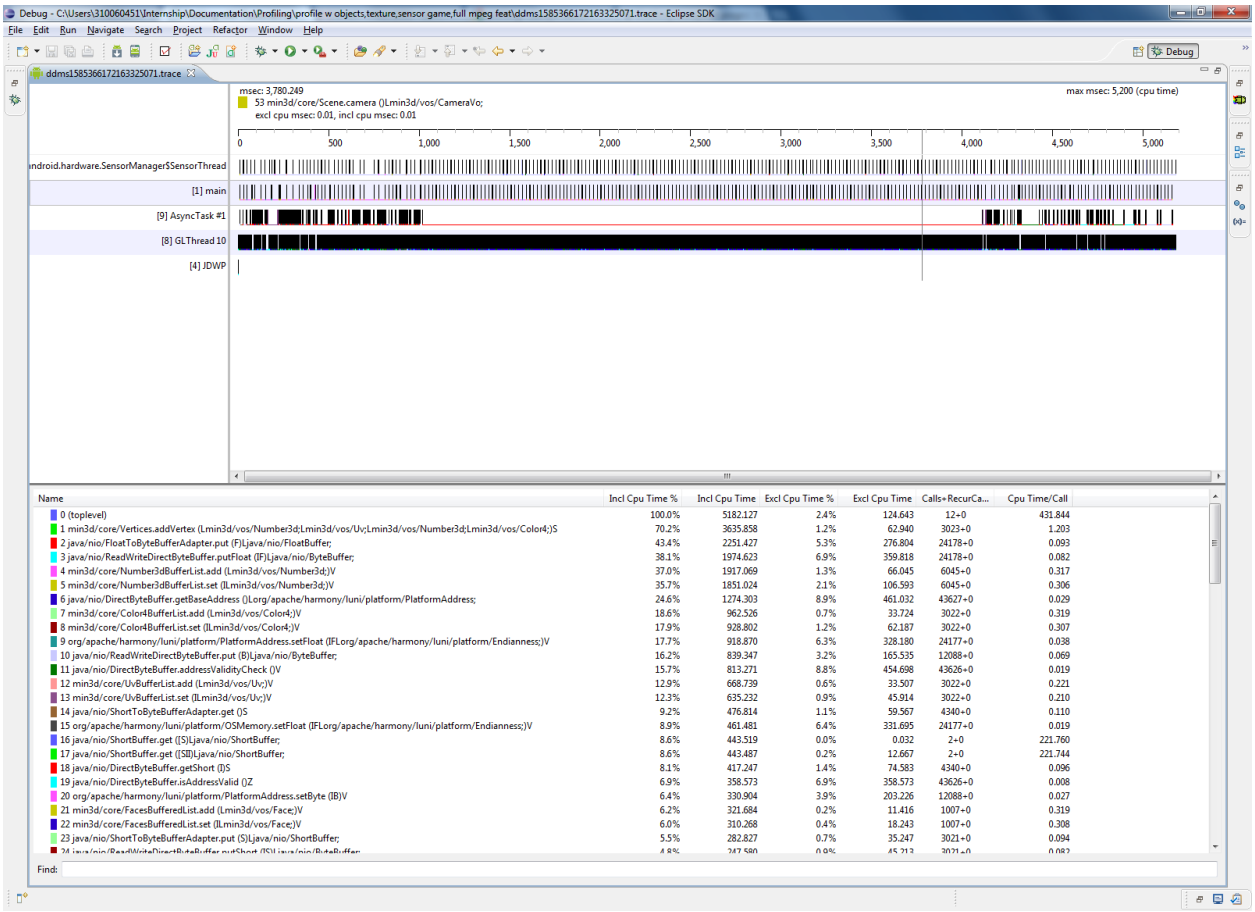


Figure 29. First Profile Panel

The highest Inclusive CPU Time % is the method addVertex() for rendering the scene at the startup, but the highest exclusive CPU Time % comes from several Buffer methods for playing the mp4 file. In the CPU Time/Call column we also can recognize that the DirectByteBuffer.getShort() and DirectByteBuffer.isAddresValid() have significant values comparing to the rest although it's only been called 2 times . Sound quality becomes worse and delayed meanwhile the visualization isn't affected at all.

- Profile with addition objects, textures, only Repeat Button on MPEG Surround Features

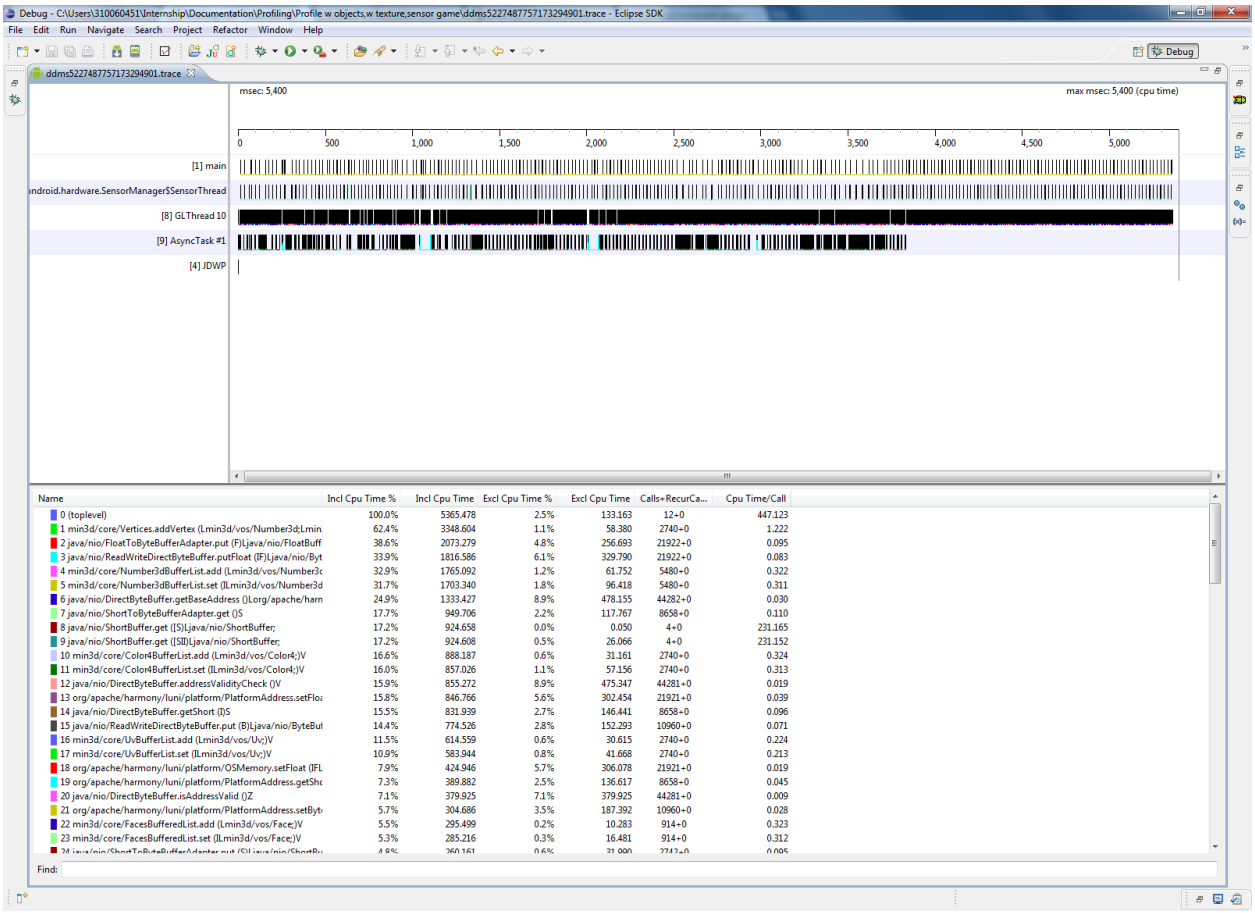


Figure 30. Second Profile Panel

The highest Inclusive CPU Time % is still the method addVertex() for rendering the scene at the startup even though it has less value than previous profile, but the highest exclusive CPU Time % come from several Buffer methods for playing the mp4 file. In the CPU Time/Call column we also can recognize that the DirectByteBuffer.getShort() and DirectByteBuffer.isAddresValid() have significant values comparing to the rest although it's only been called 4 times . The quality of the sound is getting worse and delayed meanwhile the visualization isn't affected at all.

- Profile without addition objects, without textures, only Repeat Button on MPEG Surround Features

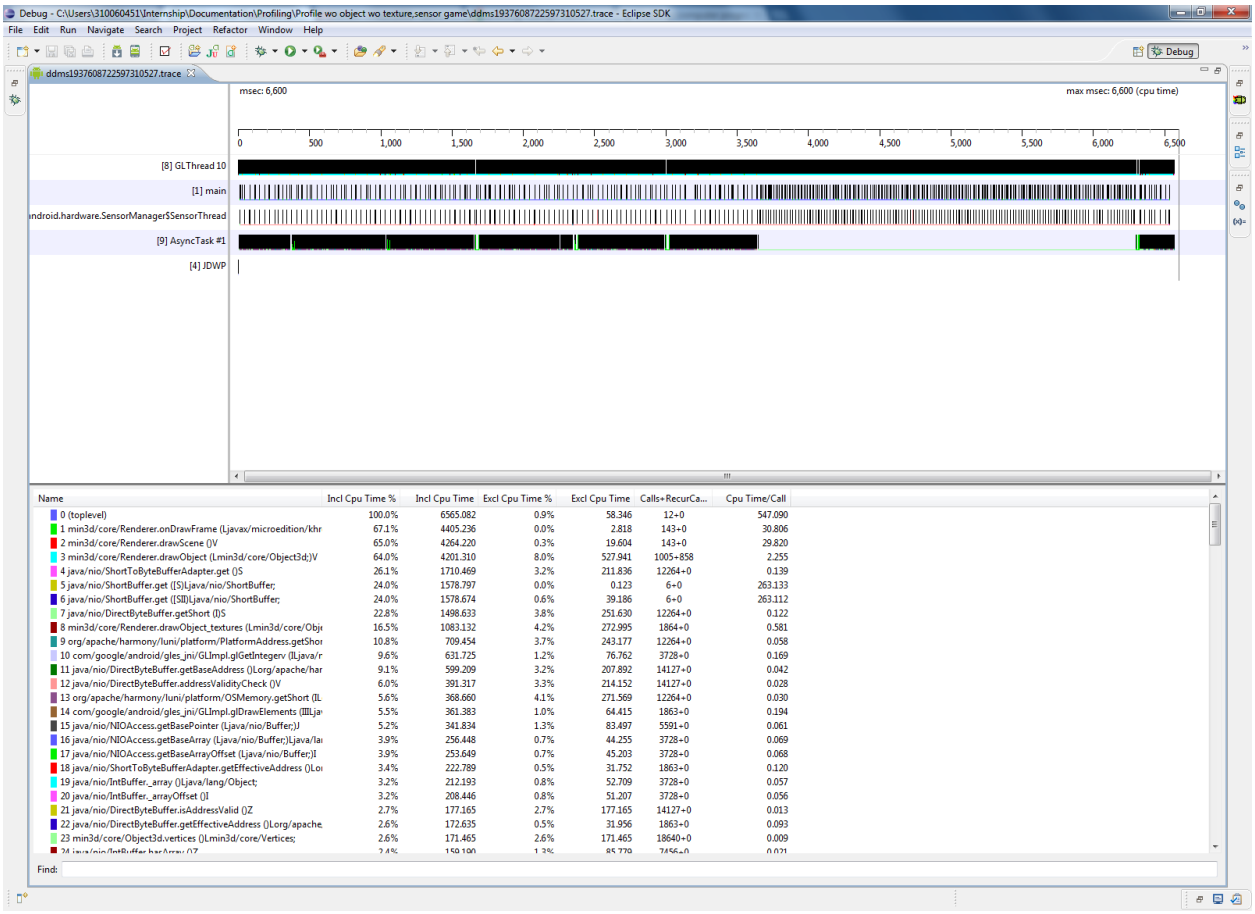


Figure 31. Third Profile Panel

The highest Inclusive CPU Time % is the method from Renderer class for drawing the scene at the startup. Now there are 3 openGL methods on top of the list. The highest exclusive CPU Time % comes from method drawObject(). It could be floated up because there are no texture at all. In the CPU Time/Call column we also can recognize that the DirectByteBuffer.getShort() and DirectByteBuffer.isAddresValid() have significant values comparing to the rest although it's only been called 6 times . The quality of the sound is better than previous profile and it's also a bit delayed meanwhile the visualization isn't affected at all.

- Profile without addition objects, with textures, only Repeat Button on MPEG Surround Features

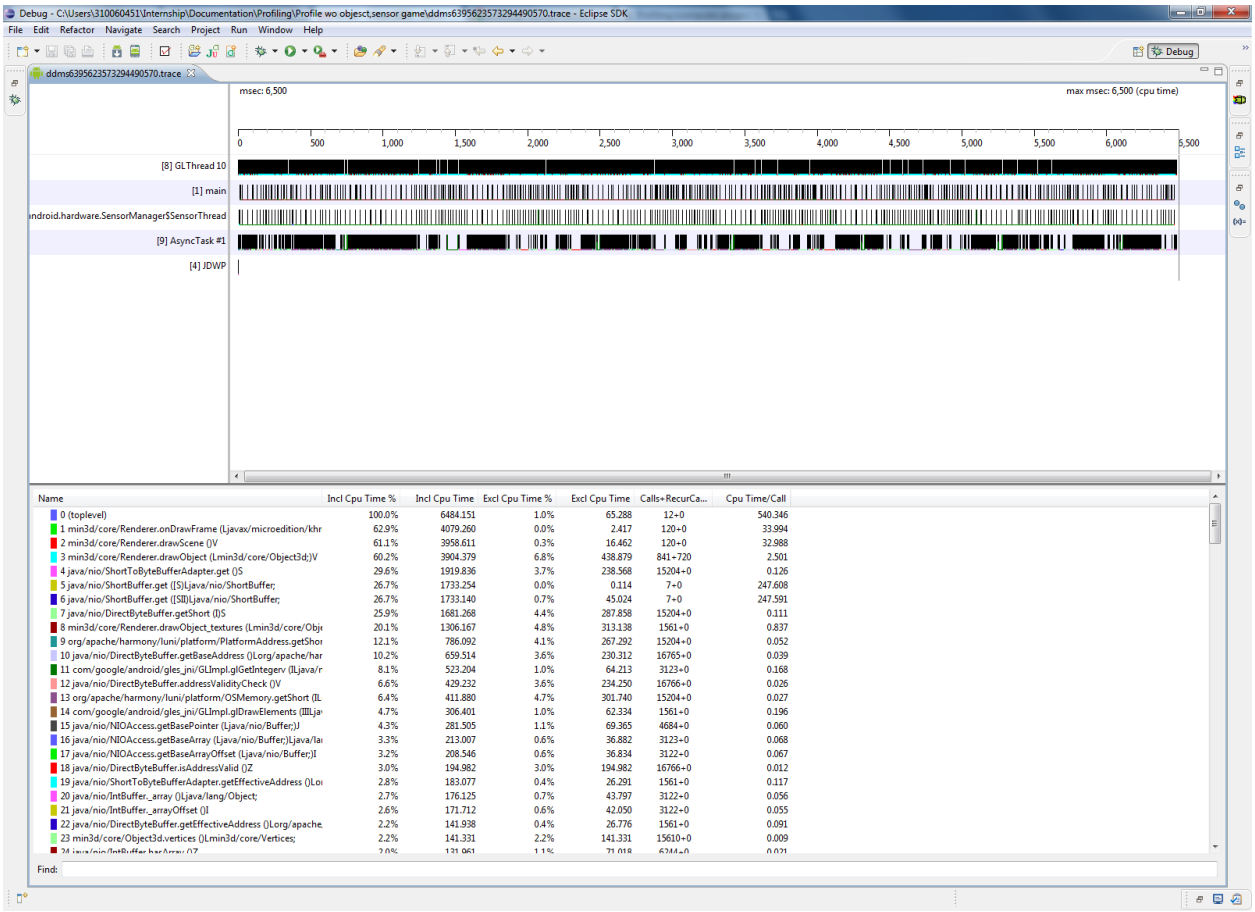


Figure 32. Fourth Profile Panel

The highest Inclusive CPU Time % is the method from Renderer class for drawing the scene at the startup. Now there are 3 openGL methods on top of the list. The highest exclusive CPU Time % comes from method drawObject(), but it has less percentage than previous profile. In the CPU Time/Call column we also can recognize that the DirectByteBuffer.getShort() and DirectByteBuffer.isAddresValid() have significant values comparing to the rest although it's only been called 6 times . The quality of the sound is surprisingly very smooth, and after I bring the additional object back to the scene the performance of the sound is still stable.

- Profile without addition objects, with textures, only Repeat Button on MPEG Surround Features, Wakelock added.

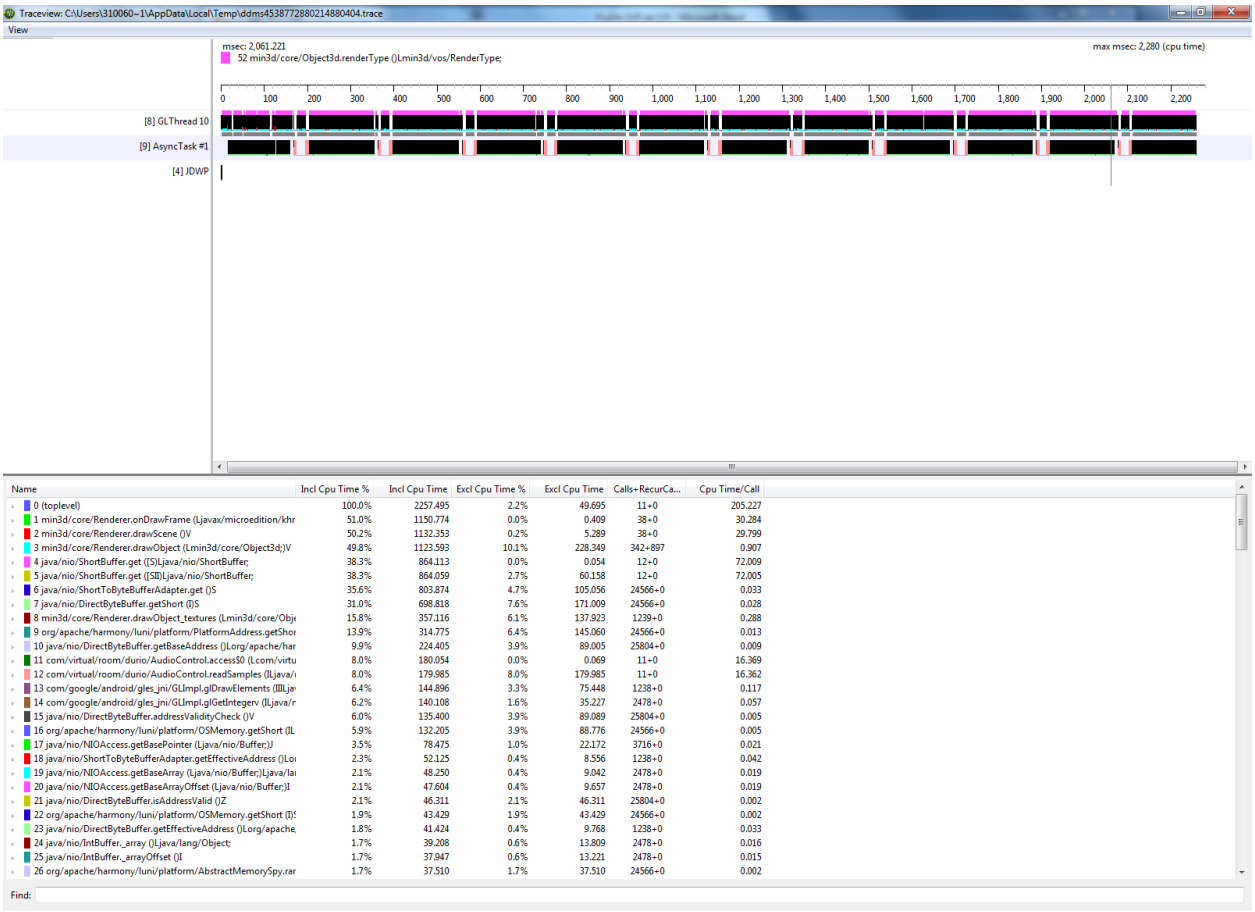


Figure 33. Fifth Profile Panel

The highest Inclusive CPU Time % is the method from Renderer class at the startup but it is significantly decreased from the last profile. Now there are 3 openGL methods on top of the list. The highest exclusive CPU Time % comes from method drawObject(), but it has less percentage than previous profile. In the CPU Time/Call column we also can recognize that the DirectByteBuffer.getShort() and DirectByteBuffer.isAddresValid() have significant values comparing to the rest, but it is also drastically decreased compared to the last profile. Quality of the sound is very smooth, even after I bring the additional object back to the scene. Conceptually, wakelock keep the processor at its peak frequency, But eventually after a few moment the sound become delayed again. This could be the effect of RAM memory which is keep growing time after time.

5.14.2.6 Profiling Conclusion

Profiling using DDMS Traceview is fairly good to track how many resource needed by particular method. However, the result is sometimes biased even in the same occasion. The sound works flawlessly at the moment but somehow next time I start the application I got different performance. I think amount of RAM is also affecting the stability of an application. Despite all of the biased result, it is fair to use DDMS Traceview to get a raw representation of CPU usage.

5.14.3 Memory Analysis Activity

5.14.3.1 About Memory Analysis

Memory Analysis is the science of using a memory image to determine information about running programs, the operating system, and the overall state of a computer.

5.14.3.2 Eclipse's Memory Analyzer Tool

The Eclipse Memory Analyzer is a fast and feature-rich **Java heap analyzer** that helps you find memory leaks and reduce memory consumption.

Use the Memory Analyzer to analyze productive heap dumps with hundreds of millions of objects, quickly calculate the retained sizes of objects, see who is preventing the Garbage Collector from collecting objects, run a report to automatically extract leak suspects.

5.14.3.3 Problem Scope

1. Dominator Tree : List the biggest objects and what they keep alive.
2. Top Consumers : Print the most expensive objects grouped by class and by package.
3. Duplicate Classes : Detect classes loaded by multiple class loaders.
4. Leak Suspects : includes leak suspects and a system overview
5. Top Components : list reports for components bigger than 1 percent of the total heap.

5.14.5.4 How to Install Memory Analyzer Tool on Eclipse

1. To install a new functionality, select Help → Install New Software
2. Select **All Sites** on the **Work with** dropdown.
3. After all lists appear, scroll down and find **General Purpose Tools**.
4. Tick on **Memory Analyzer** and **Memory Analyzer Charts** (Optional).
5. Do the Next-Finish formality and Installation will be started.
6. Restart the Eclipse IDE.
7. Memory Analyzer Tools is now ready to be used.

5.14.5.5 How to Use Memory Analyzer Tool on Eclipse

1. Run your application with **debug enabled**.
2. Switch to **DDMS** Perspective.
3. Select your device and click **Dump HPROF file.**
4. A setup wizard will appear. Click **Finish** to show the memory leaks overview.

5.14.5.6 Result and Screenshot

This screenshots were taken using Memory Analyzer Tools on Eclipse. I managed to make 4 different analysis based on the existence of additional objects, object textures, texture quality and MPEG Surround Features. Game Sensor Accelerometer Delay has been used for each condition.

- **Memory Analysis with addition objects, textures, full MPEG Surround Features**

1. Problem Suspect

Overview

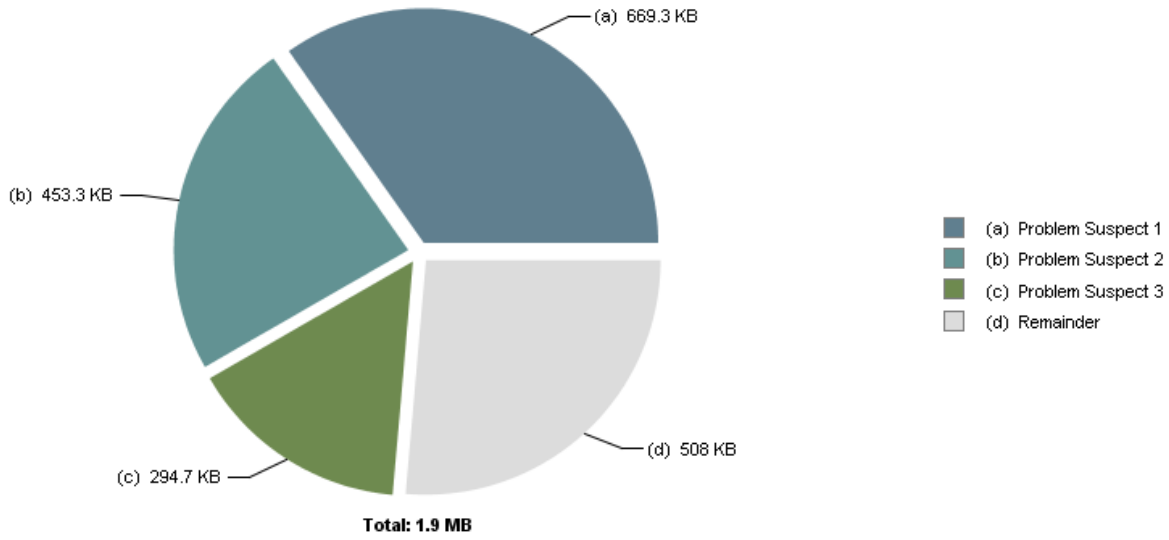


Figure 34. Problem Suspect Pie Chart 1

- **Problem Suspect 1**

2,410 instances of **"java.lang.Class"**, loaded by **"<system class loader>"** occupy **685,368 (34.76%)** bytes.

Biggest instances:

- class com.ibm.icu4jni.util.Resources$DefaultTimeZones @ 0x401fce78 - 165,432 (8.39%) bytes.
- class android.text.Html$HtmlParser @ 0x4018e040 - 126,592 (6.42%) bytes.
- class org.apache.harmony.security.fortress.Services @ 0x4008c340 - 51,456 (2.61%) bytes.
- class android.content.res.Resources @ 0x400538a8 - 38,064 (1.93%) bytes.
- class org.apache.harmony.luni.platform.PlatformAddress @ 0x4000d1d0 - 20,368 (1.03%) bytes.

Keywords
java.lang.Class

- **Problem Suspect 2**

7,385 instances of **"java.lang.String"**, loaded by **"<system class loader>"** occupy **464,192 (23.54%)** bytes.

Keywords
java.lang.String

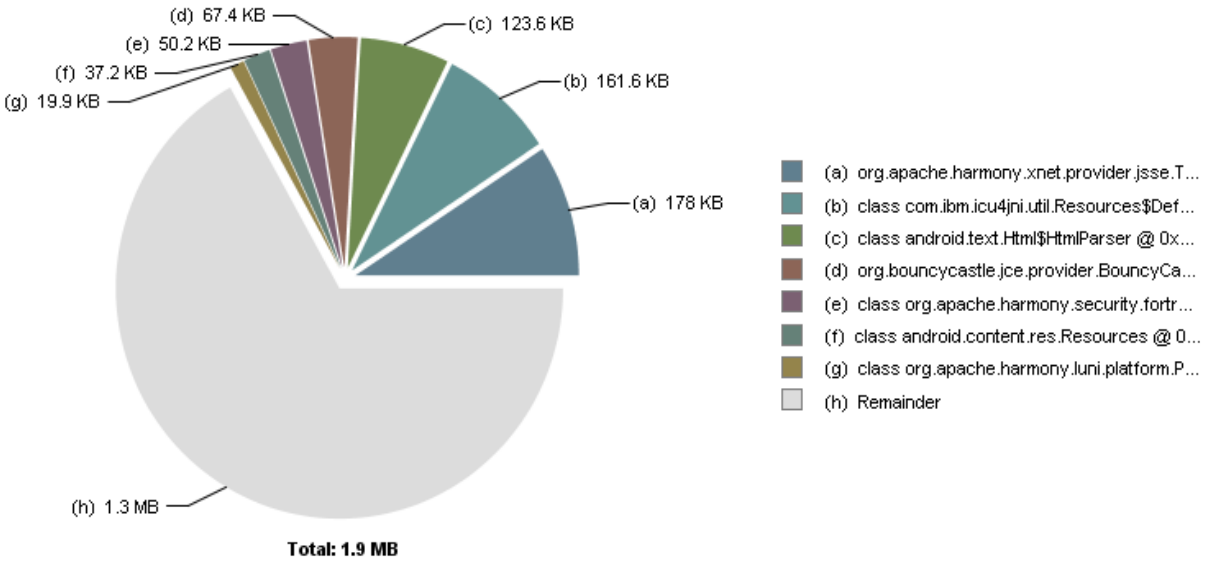
- **Problem Suspect 3**

61 instances of **"org.bouncycastle.jce.provider.X509CertificateObject"**, loaded by **"<system class loader>"** occupy **301,776 (15.31%)** bytes. These instances are referenced from one instance of **"java.util.HashMap$HashMapEntry[]"**, loaded by **"<system class loader>"**

Keywords
org.bouncycastle.jce.provider.X509CertificateObject
java.util.HashMap$HashMapEntry[]

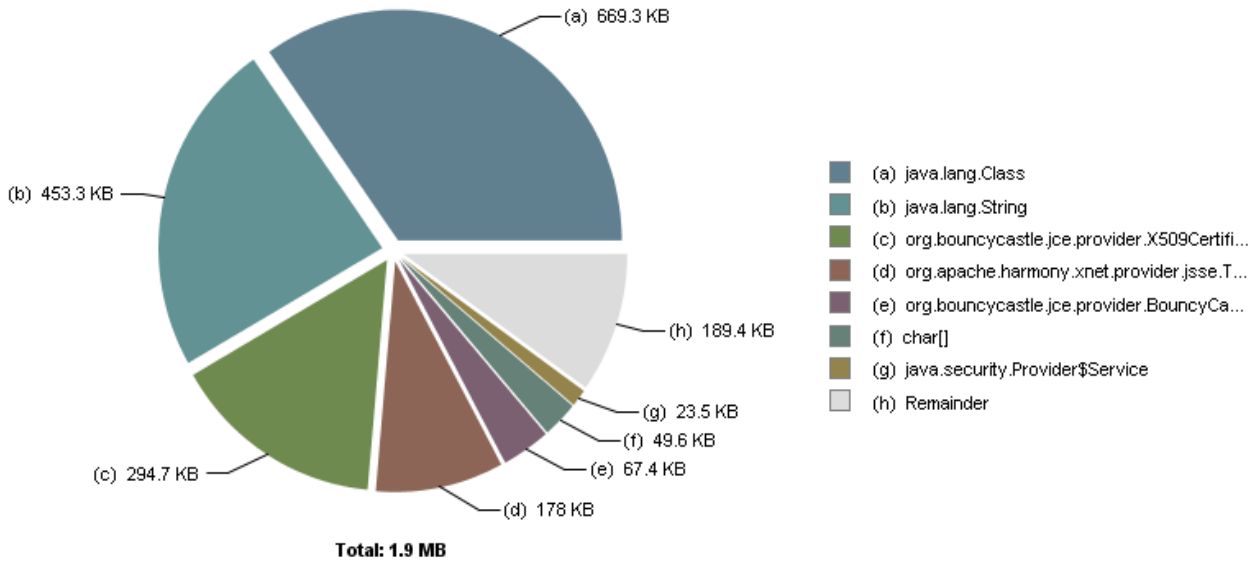
2. Top Consumer

Biggest Objects (Overview)



Biggest Objects

Biggest Top-Level Dominator Classes (Overview)



Biggest Top-Level Dominator Classes

Label	Number of Objects	Used Heap Size	Retained Heap Size	Retained Heap, %
java.lang.Class	2,410	58,032	685,368	34.76%
java.lang.String	7,385	177,240	464,192	23.54%
org.bouncycastle.jce.provider.X509CertificateObject	61	1,464	301,776	15.31%
org.apache.harmony.xnet.provider.jsse.TrustManagerImpl	1	24	182,320	9.25%
org.bouncycastle.jce.provider.BouncyCastleProvider	1	112	69,064	3.50%
char[]	392	50,840	50,840	2.58%
java.security.Provider$Service	180	7,200	24,016	1.22%
∑ Total: 7 entries	**10,430**	**294,912**	**1,777,576**	

Biggest Top-Level Dominator Class Loaders (Overview)

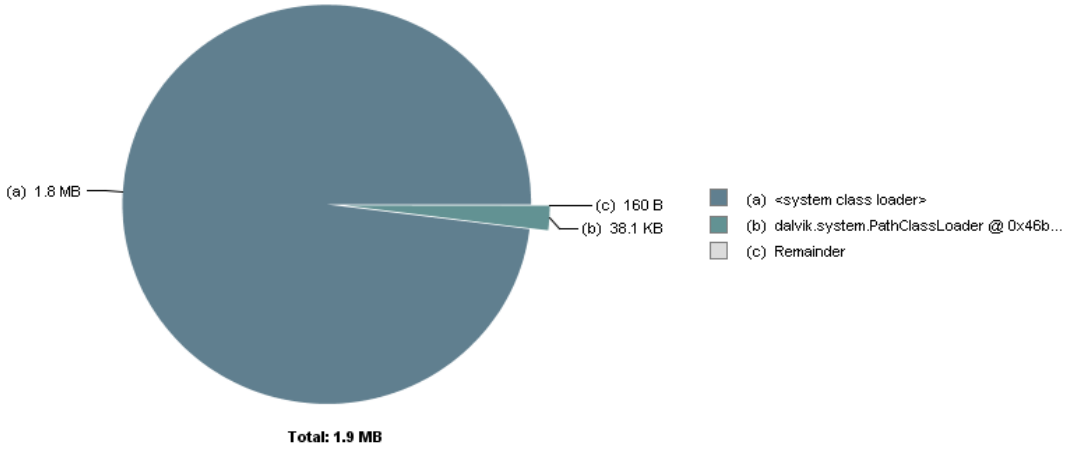


Figure 35. Biggest Objects and Dominator Classes Overview 1

3. Top Component

<system class loader> (98%)

Size: **1.8 MB** Classes: **814** Objects: **51.4k** Class Loader: **2**

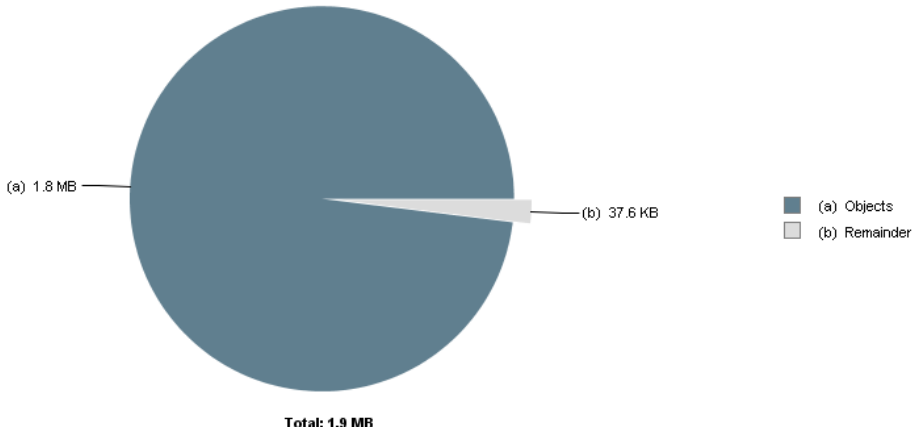


Figure 36. Top Component Overview 1

4. Possible Memory Waste

Duplicate Strings

Found 67 occurrences of char[] with at least 10 instances having identical content. Total size is 68,872 bytes.

Top elements include:

- 51 x **Atlantic Standard Time** (56 bytes)
- 84 x **GMT+00:00** (32 bytes)
- 35 x **Central European Summer Time** (72 bytes)
- 45 x **Greenwich Mean Time** (56 bytes)
- 42 x **Eastern Standard Time** (56 bytes)

Empty Collections

No excessive usage of empty collections found.

Collection Fill Ratios

Detected the following collections with fill ratios below 20%:

- 378 instances of **java.util.ArrayList** retain **>= 100,760** bytes.

Miscellaneous

Soft Reference Statistics

A total of 6 java.lang.ref.SoftReference objects have been found, which softly reference no objects.
No objects totalling 0 B are retained (kept alive) only via soft references.
No objects totalling 0 B are softly referenced and also strongly retained (kept alive) via soft references.

Weak Reference Statistics

A total of 164 java.lang.ref.WeakReference objects have been found, which weakly reference 76 objects.
No objects totalling 0 B are retained (kept alive) only via weak references.
No objects totalling 0 B are weakly referenced and also strongly retained (kept alive) via weak references.

Finalizer Statistics

Heap dump contains no java.lang.ref.Finalizer objects.
IBM VMs implement Finalizer differently and are currently not supported by this report.

Map Collision Ratios

Detected the following maps with collision ratios above 80%:

- 2 instances of **java.util.HashMap** retain **>= 752** bytes.

5. Dominator Tree

Class Name	Shallow Heap	Retained Heap	Percentage
	<Numeric>	<Numeric>	<Numeric>
org.apache.harmony.xnet.provider.jsse.TrustManagerImpl @ 0x4009cb60	24	182,320	9.25%
class com.ibm.icu4jni.util.Resources$DefaultTimeZones @ 0x401fce78 System Class	8	165,432	8.39%
class android.text.Html$HtmlParser @ 0x4018e040 System Class	8	126,592	6.42%
org.bouncycastle.jce.provider.BouncyCastleProvider @ 0x4009b8c0	112	69,064	3.50%
class org.apache.harmony.security.fortress.Services @ 0x4008c340 System Class	24	51,456	2.61%
class android.content.res.Resources @ 0x400538a8 System Class	32	38,064	1.93%
class org.apache.harmony.luni.platform.PlatformAddress @ 0x4000d1d0 System Class	24	20,368	1.03%
java.util.ArrayList @ 0x46be5610	24	17,728	0.90%
char[7938] @ 0x40076d80 Africa/AbidjanAfrica/AccraAfrica/Addis_AbabaAfrica/Algiers	15,888	15,888	0.81%
class org.apache.harmony.security.utils.AlgNameMapper @ 0x4023f018 System Class	24	15,824	0.80%
android.content.res.StringBlock @ 0x40247648	32	12,616	0.64%
class com.android.internal.util.HanziToPinyin @ 0x401e9050 System Class	32	12,272	0.62%
class org.apache.harmony.luni.internal.net.www.protocol.jar.JarURLConnectionImpl @	8	12,224	0.62%
dalvik.system.PathClassLoader @ 0x46b6bfb0	48	12,032	0.61%
android.widget.ListView @ 0x46dc9d58	776	11,824	0.60%
class android.R$styleable @ 0x40019918 System Class	4,416	10,584	0.54%
class com.android.internal.R$styleable @ 0x400447e0 System Class	4,480	10,544	0.53%
com.android.internal.policy.impl.PhoneWindow @ 0x46b75368	176	9,664	0.49%
class org.apache.harmony.luni.internal.util.ZoneInfoDB @ 0x4005ec50 System Class	56	9,600	0.49%
class org.bouncycastle.crypto.engines.AESFastEngine @ 0x402392b0 System Class	64	9,064	0.46%
class com.ibm.icu4jni.util.Resources @ 0x4001ff18 System Class	24	8,744	0.44%
class com.android.internal.telephony.GsmAlphabet @ 0x401ec900 System Class	48	8,688	0.44%
com.virtual.room.durio.AudioControl$playSoundTask @ 0x46de4aa8	48	8,272	0.42%
org.bouncycastle.jce.provider.JDKKeyStore @ 0x400d54c0	16	8,080	0.41%
class android.media.MediaFile @ 0x4006e090 System Class	256	7,832	0.40%
class java.lang.System @ 0x4000f9d8 System Class	24	7,672	0.39%
min3d.objectPrimitives.SkyBox @ 0x46de9748	104	7,272	0.37%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x40156218	24	7,216	0.37%
class java.lang.Integer @ 0x4000ec10 System Class	56	7,144	0.36%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x4014f440	24	6,928	0.35%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x40154c68	24	6,624	0.34%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x40136b48	24	6,608	0.34%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x4011a0e8	24	6,536	0.33%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400abe98	24	6,376	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400f1bd0	24	6,272	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400d3238	24	6,256	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x401024f8	24	6,232	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400cbaa8	24	6,224	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400f0da8	24	6,224	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x4013a588	24	6,224	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x401630c0	24	6,144	0.31%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x40134728	24	6,112	0.31%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400c3db0	24	6,104	0.31%

Figure 37. Dominator Tree 1

- **Memory Analysis without addition objects, with textures, only Repeat Button on MPEG Surround Features**

1. Problem Suspect

Overview

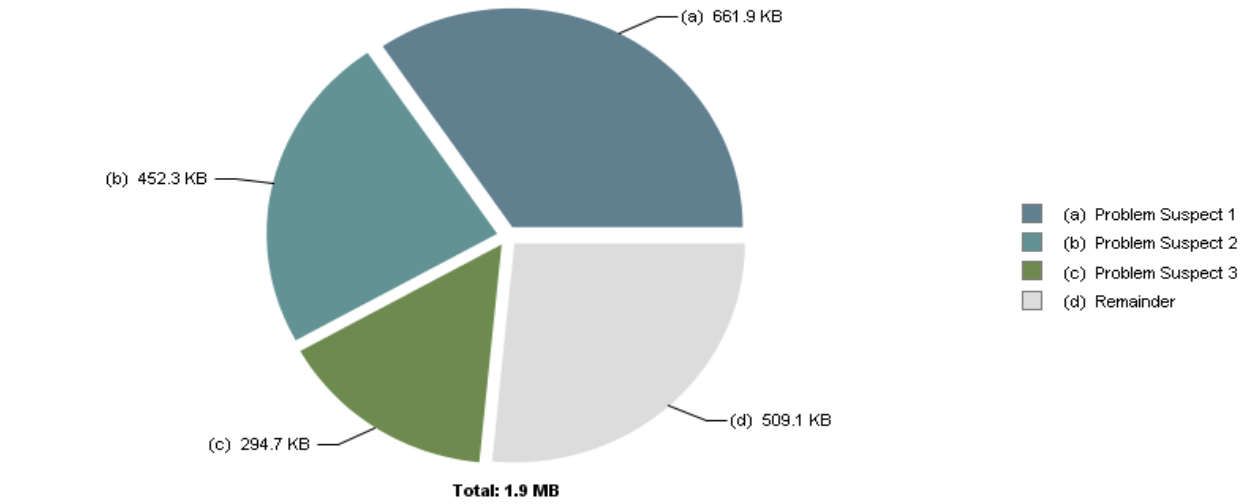


Figure 38. Problem Suspect Pie Chart 2

• Problem Suspect 1

2,396 instances of **"java.lang.Class"**, loaded by **"<system class loader>"** occupy **677,808 (34.51%)** bytes.

Biggest instances:

- class com.ibm.icu4jni.util.Resources$DefaultTimeZones @ 0x401fce78 - 165,432 (8.42%) bytes.
- class android.text.Html$HtmlParser @ 0x4018e040 - 126,592 (6.45%) bytes.
- class org.apache.harmony.security.fortress.Services @ 0x4008c340 - 51,456 (2.62%) bytes.
- class android.content.res.Resources @ 0x400538a8 - 38,064 (1.94%) bytes.

Keywords
java.lang.Class

• Problem Suspect 2

7,374 instances of **"java.lang.String"**, loaded by **"<system class loader>"** occupy **463,128 (23.58%)** bytes.

Keywords
java.lang.String

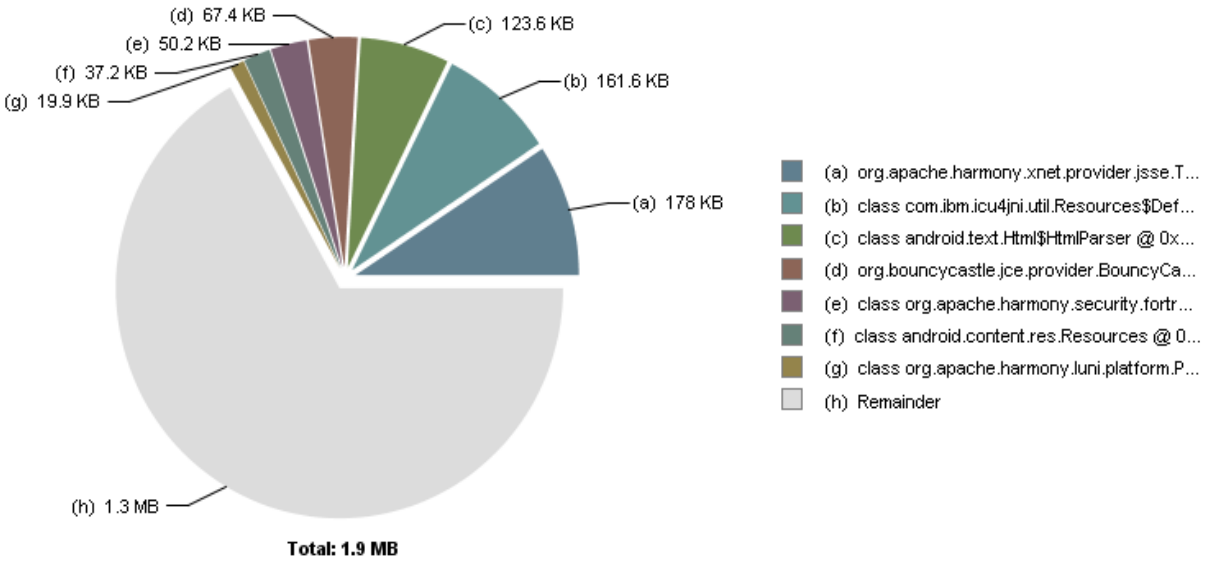
- **Problem Suspect 3**

61 instances of **"org.bouncycastle.jce.provider.X509CertificateObject"**, loaded by **"<system class loader>"** occupy **301,776 (15.37%)** bytes. These instances are referenced from one instance of **"java.util.HashMap$HashMapEntry[]"**, loaded by **"<system class loader>"**

Keywords
org.bouncycastle.jce.provider.X509CertificateObject
java.util.HashMap$HashMapEntry[]

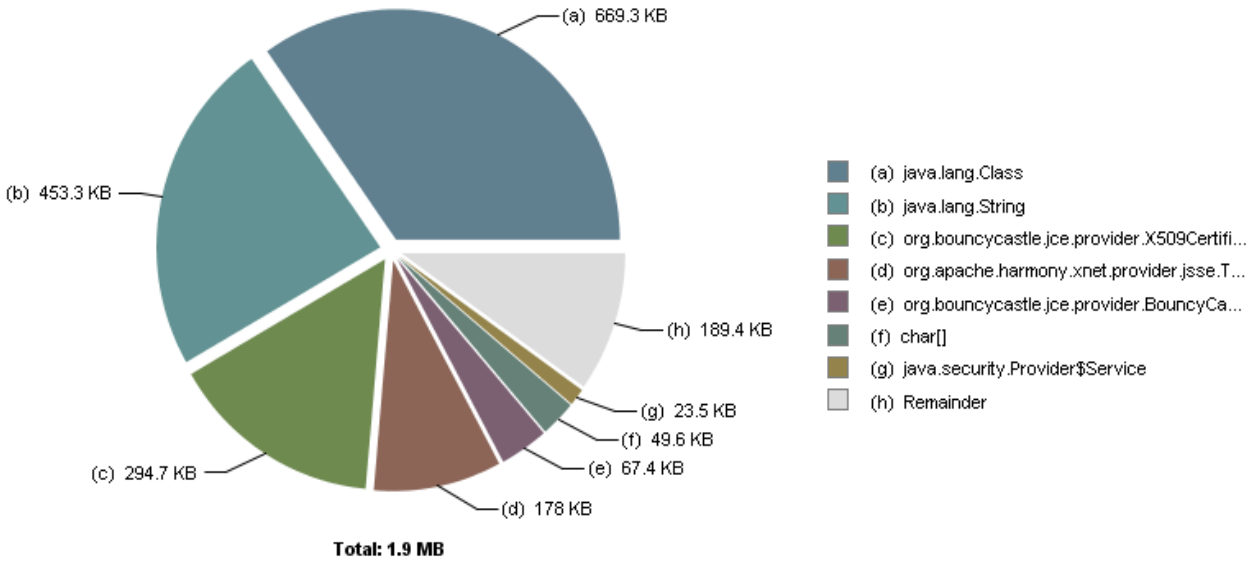
2. Top Consumer

Biggest Objects (Overview)



Biggest Objects

Biggest Top-Level Dominator Classes (Overview)



Biggest Top-Level Dominator Classes

Label	Number of Objects	Used Heap Size	Retained Heap Size	Retained Heap, %
java.lang.Class	2,396	58,008	677,808	34.51%
java.lang.String	7,374	176,976	463,128	23.58%
org.bouncycastle.jce.provider.X509CertificateObject	61	1,464	301,776	15.37%
org.apache.harmony.xnet.provider.jsse.TrustManagerImpl	1	24	182,320	9.28%
org.bouncycastle.jce.provider.BouncyCastleProvider	1	112	69,064	3.52%
char[]	394	50,904	50,904	2.59%
java.security.Provider$Service	180	7,200	24,016	1.22%
∑ Total: 7 entries	10,407	294,688	1,769,016	

Biggest Top-Level Dominator Class Loaders (Overview)

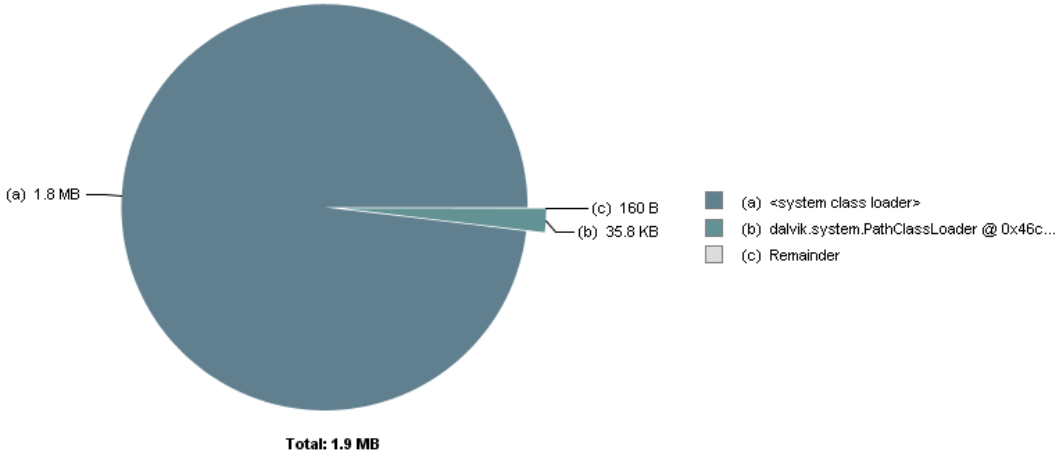


Figure 39. Biggest Objects and Dominator Classes Overview 2

3. Top Component

There's no components bigger than 1 percent of the total heap.

4. Possible Memory Waste

There's no components bigger than 1 percent of the total heap.

5. Dominator Tree

Class Name	Shallow Heap	Retained Heap	Percentage
<Regex>	<Numeric>	<Numeric>	<Numeric>
org.apache.harmony.xnet.provider.jsse.TrustManagerImpl @ 0x4009cb60	24	182,320	9.28%
class com.ibm.icu4jni.util.Resources$DefaultTimeZones @ 0x401fce78 System Class	8	165,432	8.42%
class android.text.Html$HtmlParser @ 0x4018e040 System Class	8	126,592	6.45%
org.bouncycastle.jce.provider.BouncyCastleProvider @ 0x4009b8c0	112	69,064	3.52%
class org.apache.harmony.security.fortress.Services @ 0x4008c340 System Class	24	51,456	2.62%
class android.content.res.Resources @ 0x400538a8 System Class	32	38,064	1.94%
char[7938] @ 0x40076d80 Africa/AbidjanAfrica/AccraAfrica/Addis_AbabaAfrica/AlgiersAfrica/	15,888	15,888	0.81%
class org.apache.harmony.security.utils.AlgNameMapper @ 0x4023f018 System Class	24	15,824	0.81%
class org.apache.harmony.luni.platform.PlatformAddress @ 0x4000d1d0 System Class	24	14,144	0.72%
android.content.res.StringBlock @ 0x40247648	32	12,616	0.64%
class com.android.internal.util.HanziToPinyin @ 0x401e9050 System Class	32	12,272	0.62%
class org.apache.harmony.luni.internal.net.www.protocol.jar.JarURLConnectionImpl @ 0x4003	8	12,224	0.62%
dalvik.system.PathClassLoader @ 0x46c63140	48	12,032	0.61%
android.widget.ListView @ 0x46c95958	776	11,824	0.60%
class android.R$styleable @ 0x40019918 System Class	4,416	10,584	0.54%
class com.android.internal.R$styleable @ 0x400447e0 System Class	4,480	10,544	0.54%
class org.apache.harmony.luni.internal.util.ZoneInfoDB @ 0x4005ec50 System Class	56	9,600	0.49%
class org.bouncycastle.crypto.engines.AESFastEngine @ 0x402392b0 System Class	64	9,064	0.46%
class com.ibm.icu4jni.util.Resources @ 0x4001ff18 System Class	24	8,744	0.45%
class com.android.internal.telephony.GsmAlphabet @ 0x401ec900 System Class	48	8,688	0.44%
org.apache.harmony.xml.ExpatParser @ 0x46c9a228 Unknown	48	8,360	0.43%
com.virtual.room.durio.AudioControl$playSoundTask @ 0x46cb0fe0	48	8,272	0.42%
org.bouncycastle.jce.provider.JDKKeyStore @ 0x400d54c0	16	8,080	0.41%
class android.media.MediaFile @ 0x4006e090 System Class	256	7,832	0.40%
class java.lang.System @ 0x4000f9d8 System Class	24	7,672	0.39%
min3d.objectPrimitives.SkyBox @ 0x46cb5ea8	104	7,272	0.37%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x40156218	24	7,216	0.37%
class java.lang.Integer @ 0x4000ec10 System Class	56	7,144	0.36%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x4014f440	24	6,928	0.35%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x40154c68	24	6,624	0.34%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x40136b48	24	6,608	0.34%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x4011a0e8	24	6,536	0.33%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400abe98	24	6,376	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400f1bd0	24	6,272	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400d3238	24	6,256	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x401024f8	24	6,232	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400cbaa8	24	6,224	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400f0da8	24	6,224	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x4013a588	24	6,224	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x401630c0	24	6,144	0.31%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x40134728	24	6,112	0.31%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400c3db0	24	6,104	0.31%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400f2130	24	6,064	0.31%

Figure 40. Dominator Tree 2

- **Memory Analysis without addition objects, without textures, only Repeat Button on MPEG Surround Features**

1. Problem Suspect

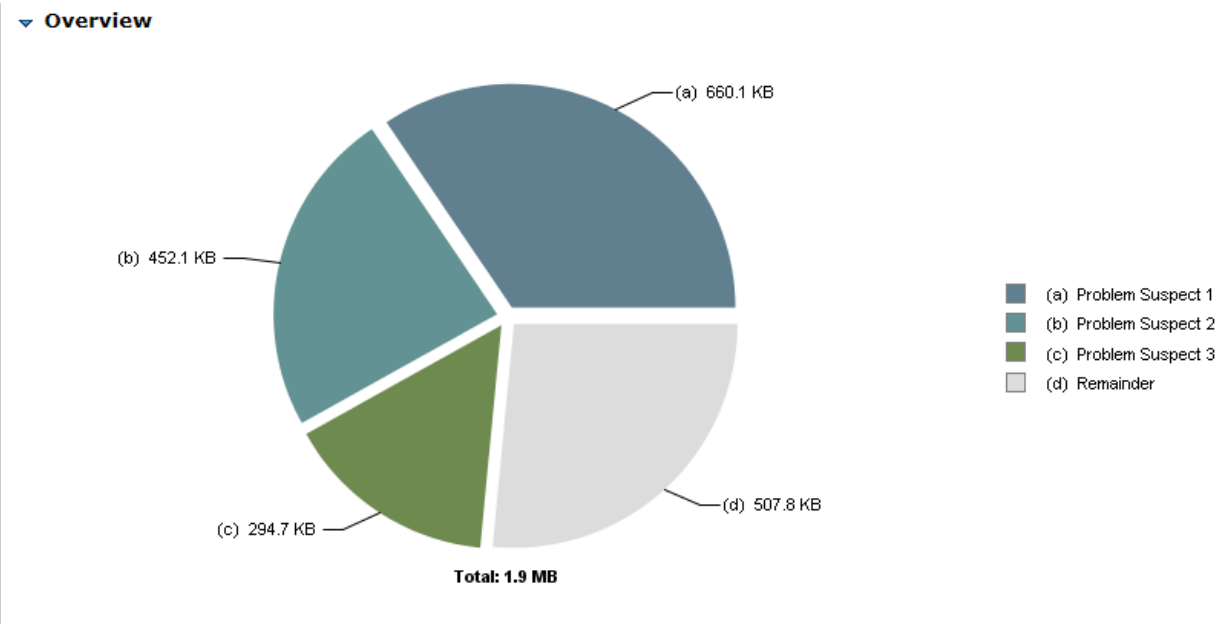


Figure 41. Problem Suspect Pie Chart 3

• Problem Suspect 1

2,396 instances of **"java.lang.Class"**, loaded by **"<system class loader>"** occupy **675,904 (34.47%)** bytes.

Biggest instances:

- class com.ibm.icu4jni.util.Resources$DefaultTimeZones @ 0x401fce78 - 165,432 (8.44%) bytes.
- class android.text.Html$HtmlParser @ 0x4018e040 - 126,592 (6.46%) bytes.
- class org.apache.harmony.security.fortress.Services @ 0x4008c340 - 51,456 (2.62%) bytes.
- class android.content.res.Resources @ 0x400538a8 - 38,064 (1.94%) bytes.

Keywords
java.lang.Class

• Problem Suspect 2

7,371 instances of **"java.lang.String"**, loaded by **"<system class loader>"** occupy **462,992 (23.61%)** bytes.

Keywords
java.lang.String

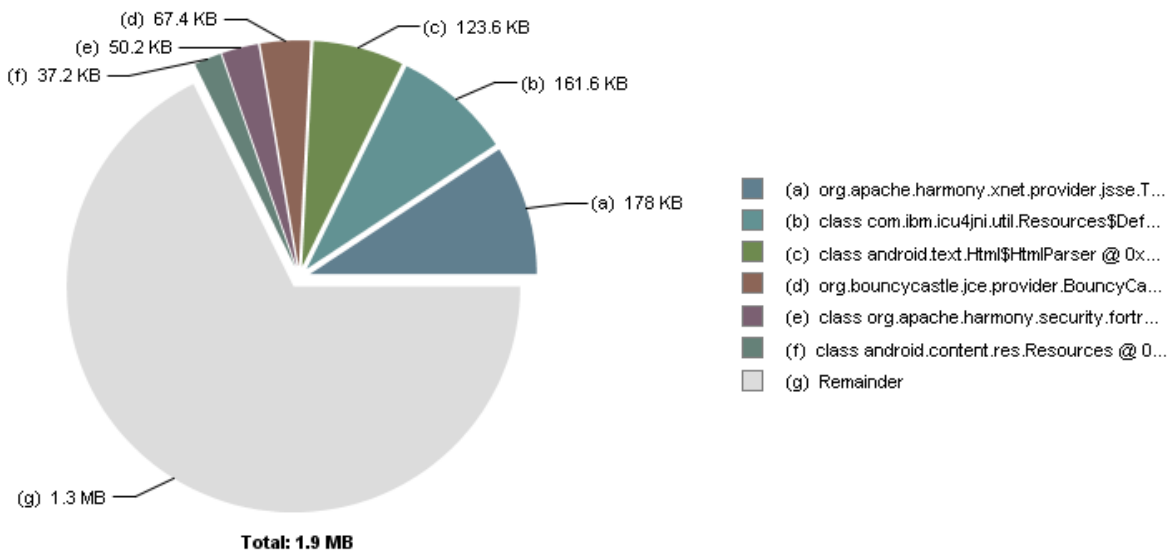
- **Problem Suspect 3**

61 instances of **"org.bouncycastle.jce.provider.X509CertificateObject"**, loaded by **"<system class loader>"** occupy **301,776 (15.39%)** bytes. These instances are referenced from one instance of **"java.util.HashMap$HashMapEntry[]"**, loaded by **"<system class loader>"**

Keywords
org.bouncycastle.jce.provider.X509CertificateObject
java.util.HashMap$HashMapEntry[]

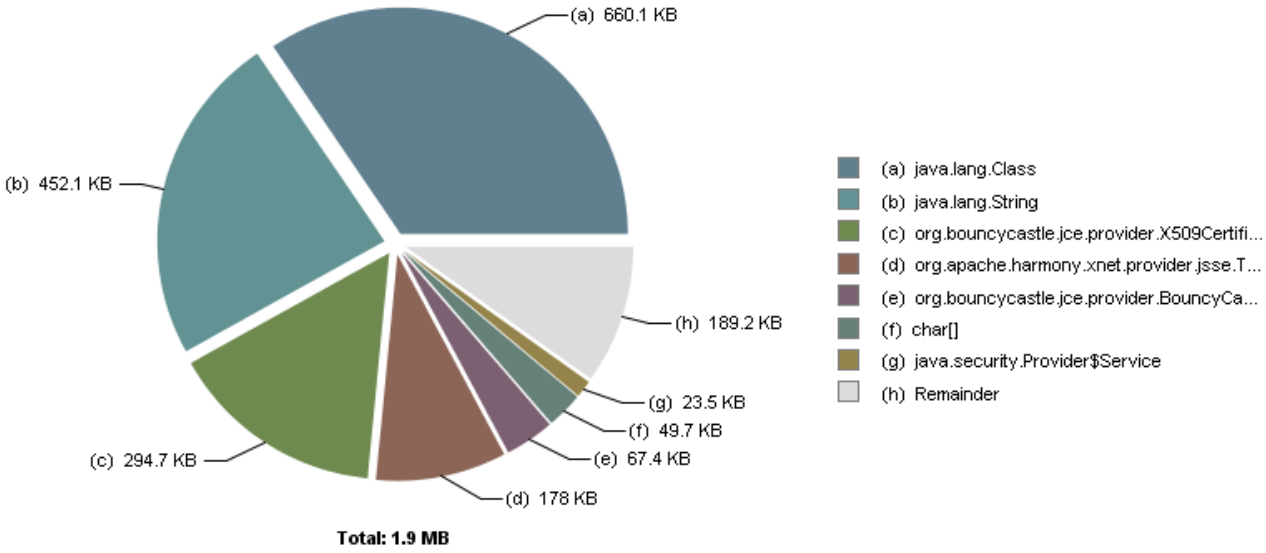
2. Top Consumer

Biggest Objects (Overview)



Biggest Objects

Biggest Top-Level Dominator Classes (Overview)



Biggest Top-Level Dominator Classes

Label	Number of Objects	Used Heap Size	Retained Heap Size	Retained Heap, %
java.lang.Class	2,396	58,008	675,904	34.47%
java.lang.String	7,371	176,904	462,992	23.61%
org.bouncycastle.jce.provider.X509CertificateObject	61	1,464	301,776	15.39%
org.apache.harmony.xnet.provider.jsse.TrustManagerImpl	1	24	182,320	9.30%
org.bouncycastle.jce.provider.BouncyCastleProvider	1	112	69,064	3.52%
char[]	393	50,856	50,856	2.59%
java.security.Provider$Service	180	7,200	24,016	1.22%
∑ Total: 7 entries	**10,403**	**294,568**	**1,766,928**	

Biggest Top-Level Dominator Class Loaders (Overview)

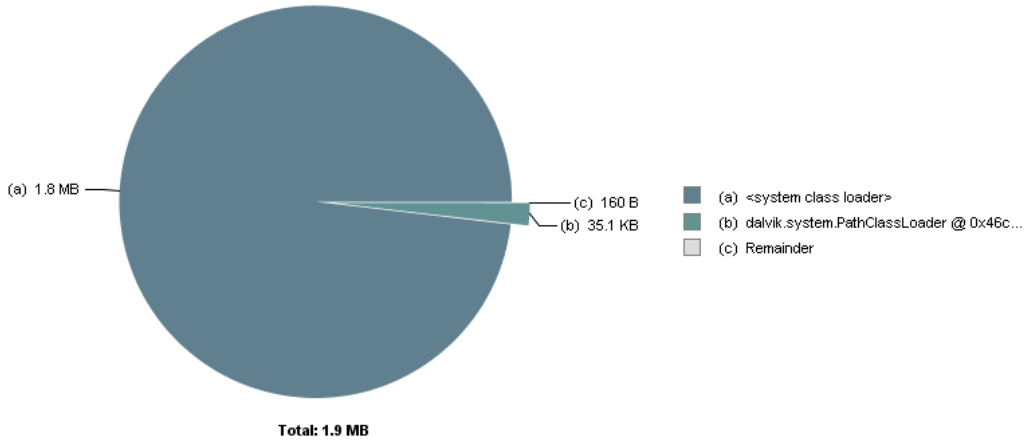


Figure 42. Biggest Objects and Dominator Classes Overview 3

3. Top Component

There's no components bigger than 1 percent of the total heap.

4. Possible Memory Waste

There's no components bigger than 1 percent of the total heap.

5. Dominator Tree

Class Name	Shallow Heap	Retained Heap	Percentage
<Regex>	<Numeric>	<Numeric>	<Numeric>
org.apache.harmony.xnet.provider.jsse.TrustManagerImpl @ 0x4009cb60	24	182,320	9.30%
class com.ibm.icu4jni.util.Resources$DefaultTimeZones @ 0x401fce78 System Class	8	165,432	8.44%
class android.text.Html$HtmlParser @ 0x4018e040 System Class	8	126,592	6.46%
org.bouncycastle.jce.provider.BouncyCastleProvider @ 0x4009b8c0	112	69,064	3.52%
class org.apache.harmony.security.fortress.Services @ 0x4008c340 System Class	24	51,456	2.62%
class android.content.res.Resources @ 0x400538a8 System Class	32	38,064	1.94%
char[7938] @ 0x40076d80 Africa/AbidjanAfrica/AccraAfrica/Addis_AbabaAfrica/Algiers	15,888	15,888	0.81%
class org.apache.harmony.security.utils.AlgNameMapper @ 0x4023f018 System Class	24	15,824	0.81%
class org.apache.harmony.luni.platform.PlatformAddress @ 0x4000d1d0 System Class	24	13,856	0.71%
android.content.res.StringBlock @ 0x40247648	32	12,616	0.64%
class com.android.internal.util.HanziToPinyin @ 0x401e9050 System Class	32	12,272	0.63%
class org.apache.harmony.luni.internal.net.www.protocol.jar.JarURLConnectionImpl @	8	12,224	0.62%
dalvik.system.PathClassLoader @ 0x46c637f0	48	12,032	0.61%
android.widget.ListView @ 0x46c8e300	776	11,824	0.60%
class android.R$styleable @ 0x40019918 System Class	4,416	10,584	0.54%
class com.android.internal.R$styleable @ 0x400447e0 System Class	4,480	10,544	0.54%
class org.apache.harmony.luni.internal.util.ZoneInfoDB @ 0x4005ec50 System Class	56	9,600	0.49%
class org.bouncycastle.crypto.engines.AESFastEngine @ 0x402392b0 System Class	64	9,064	0.46%
class com.ibm.icu4jni.util.Resources @ 0x4001ff18 System Class	24	8,744	0.45%
class com.android.internal.telephony.GsmAlphabet @ 0x401ec900 System Class	48	8,688	0.44%
org.apache.harmony.xml.ExpatParser @ 0x46c92c10 Unknown	48	8,360	0.43%
com.virtual.room.durio.AudioControl$playSoundTask @ 0x46ca93f8	48	8,272	0.42%
org.bouncycastle.jce.provider.JDKKeyStore @ 0x400d54c0	16	8,080	0.41%
class android.media.MediaFile @ 0x4006e090 System Class	256	7,832	0.40%
class java.lang.System @ 0x4000f9d8 System Class	24	7,672	0.39%
min3d.objectPrimitives.SkyBox @ 0x46cae4f0	104	7,272	0.37%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x40156218	24	7,216	0.37%
class java.lang.Integer @ 0x4000ec10 System Class	56	7,144	0.36%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x4014f440	24	6,928	0.35%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x40154c68	24	6,624	0.34%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x40136b48	24	6,608	0.34%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x4011a0e8	24	6,536	0.33%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400abe98	24	6,376	0.33%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400f1bd0	24	6,272	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400d3238	24	6,256	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x401024f8	24	6,232	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400cbaa8	24	6,224	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400f0da8	24	6,224	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x4013a588	24	6,224	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x401630c0	24	6,144	0.31%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x40134728	24	6,112	0.31%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400c3db0	24	6,104	0.31%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400f2130	24	6,064	0.31%

Figure 43. Dominator Tree 3

- **Memory Analysis with addition objects, reduced textures quality, full MPEG Surround Features**

1. Problem Suspect

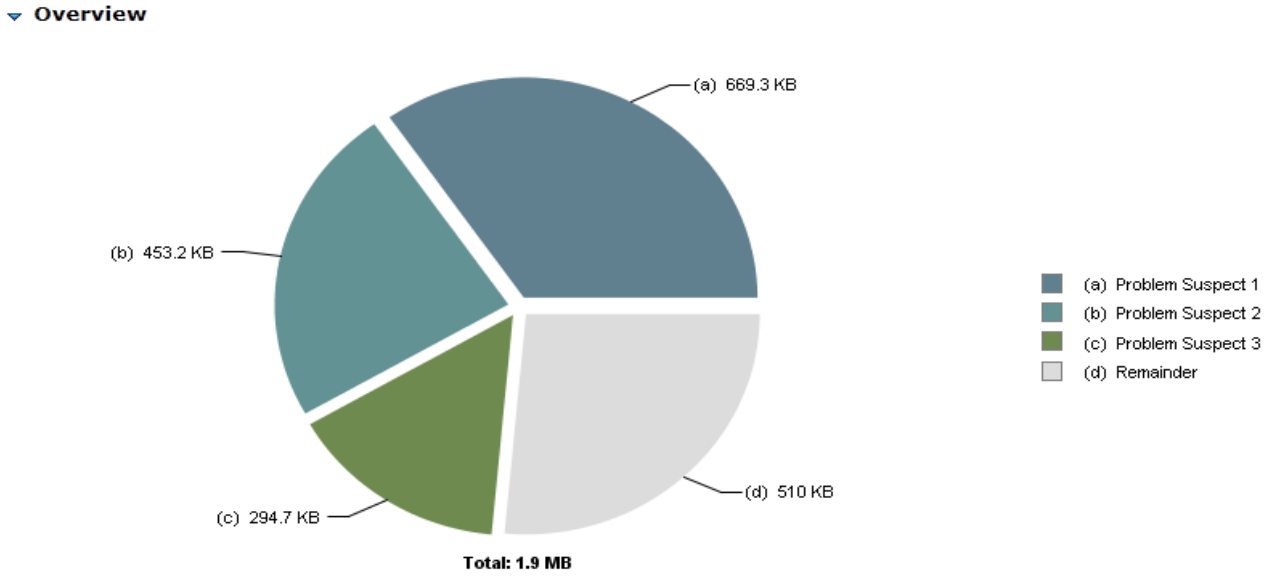


Figure 44. Problem Suspect Pie Chart 4

- **Problem Suspect 1**

2,411 instances of **"java.lang.Class"**, loaded by **"<system class loader>"** occupy **685,352 (34.73%)** bytes.

Biggest instances:

- class com.ibm.icu4jni.util.Resources$DefaultTimeZones @ 0x401fce78 - 165,432 (8.38%) bytes.
- class android.text.Html$HtmlParser @ 0x4018e040 - 126,592 (6.42%) bytes.
- class org.apache.harmony.security.fortress.Services @ 0x4008c340 - 51,456 (2.61%) bytes.
- class android.content.res.Resources @ 0x400538a8 - 37,792 (1.92%) bytes.
- class org.apache.harmony.luni.platform.PlatformAddress @ 0x4000d1d0 - 20,488 (1.04%) bytes.

Keywords
java.lang.ClasS

- **Problem Suspect 2**

7,385 instances of **"java.lang.String"**, loaded by **"<system class loader>"** occupy **464,032 (23.51%)** bytes.

Keywords
java.lang.String

Details »

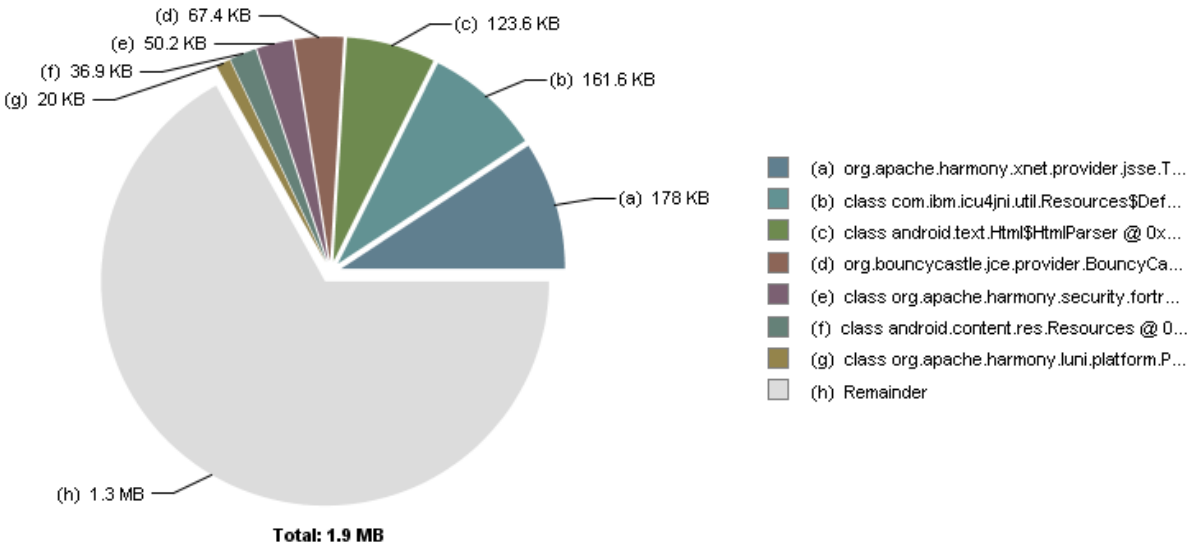
- **Problem Suspect 3**

61 instances of **"org.bouncycastle.jce.provider.X509CertificateObject"**, loaded by **"<system class loader>"** occupy **301,776 (15.29%)** bytes. These instances are referenced from one instance of **"java.util.HashMap$HashMapEntry[]"**, loaded by **"<system class loader>"**

Keywords
org.bouncycastle.jce.provider.X509CertificateObject
java.util.HashMap$HashMapEntry[]

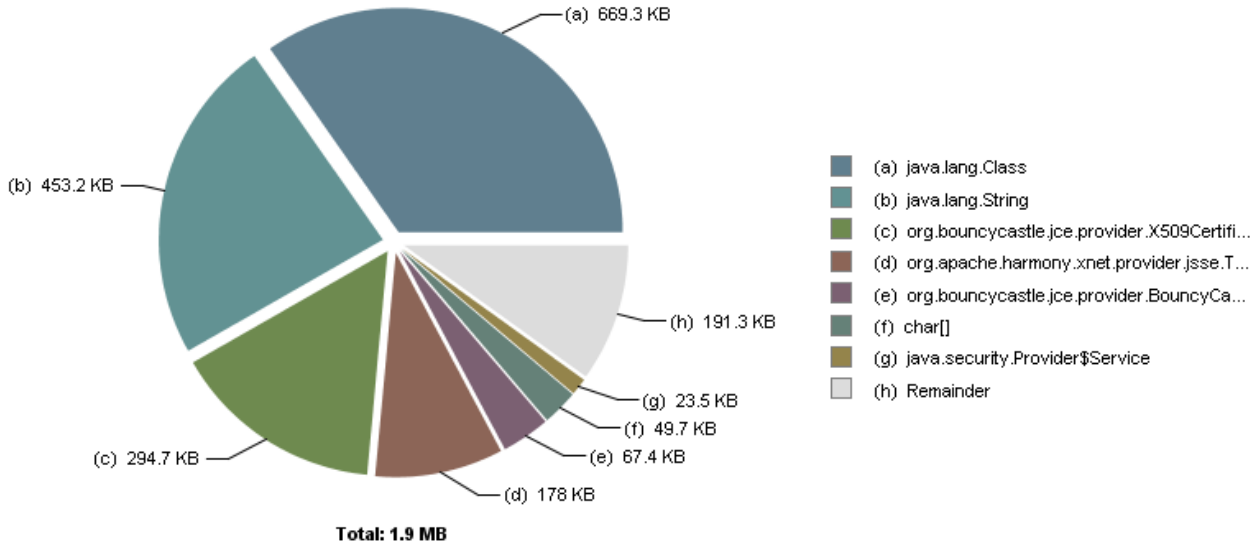
2. Top Consumer

Biggest Objects (Overview)



Biggest Objects

Biggest Top-Level Dominator Classes (Overview)



Biggest Top-Level Dominator Classes

Label	Number of Objects	Used Heap Size	Retained Heap Size	Retained Heap, %
java.lang.Class	2,411	58,032	685,352	34.73%
java.lang.String	7,385	177,240	464,032	23.51%
org.bouncycastle.jce.provider.X509CertificateObject	61	1,464	301,776	15.29%
org.apache.harmony.xnet.provider.jsse.TrustManagerImpl	1	24	182,320	9.24%
org.bouncycastle.jce.provider.BouncyCastleProvider	1	112	69,064	3.50%
char[]	393	50,888	50,888	2.58%
java.security.Provider$Service	180	7,200	24,016	1.22%
∑ Total: 7 entries	**10,432**	**294,960**	**1,777,448**	

Biggest Top-Level Dominator Class Loaders (Overview)

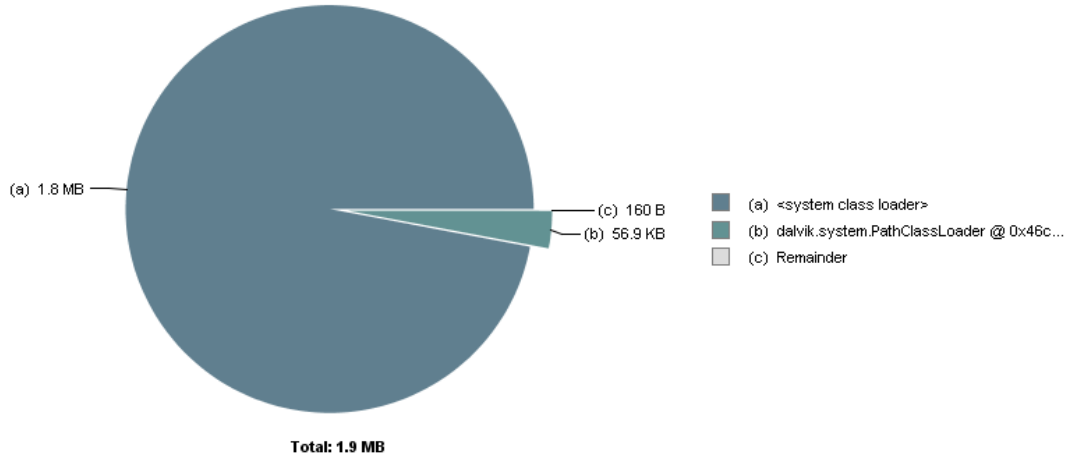


Figure 45. Biggest Objects and Dominator Classes Overview 4

3. Top Component

<system class loader> (97%)

Size: **1.8 MB** Classes: **804** Objects: **50.8k** Class Loader: **2**

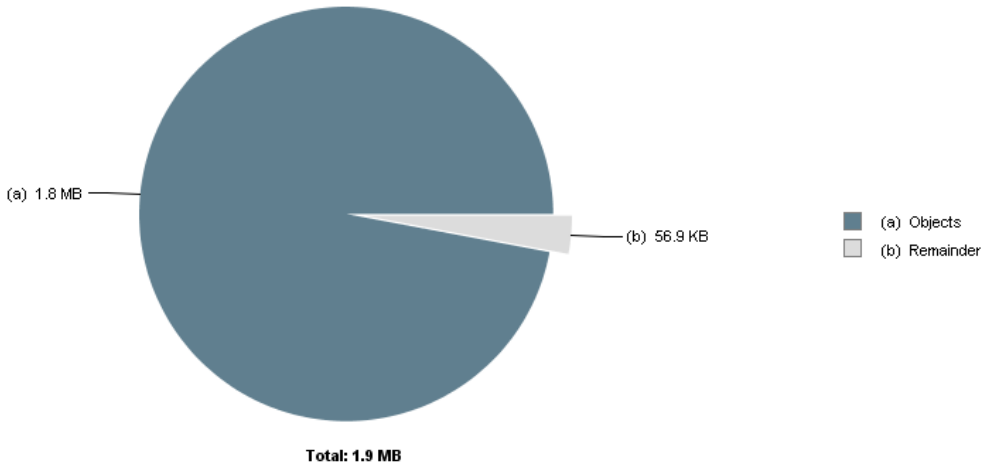


Figure 46. Top Component Overview 4

4. Possible Memory Waste

Duplicate Strings

Found 67 occurrences of char[] with at least 10 instances having identical content. Total size is 68,480 bytes.

Top elements include:

- 51 x **Atlantic Standard Time** (56 bytes)
- 84 x **GMT+00:00** (32 bytes)
- 35 x **Central European Summer Time** (72 bytes)
- 45 x **Greenwich Mean Time** (56 bytes)
- 42 x **Eastern Standard Time** (56 bytes)

Empty Collections

No excessive usage of empty collections found.

Collection Fill Ratios

No serious amount of collections with low fill ratios found.

Miscellaneous

Soft Reference Statistics

A total of 6 java.lang.ref.SoftReference objects have been found, which softly reference no objects.
No objects totalling 0 B are retained (kept alive) only via soft references.
No objects totalling 0 B are softly referenced and also strongly retained (kept alive) via soft references.

Weak Reference Statistics

A total of 163 java.lang.ref.WeakReference objects have been found, which weakly reference 75 objects.
54 objects totalling 2.1 KB are retained (kept alive) only via weak references.
No objects totalling 0 B are weakly referenced and also strongly retained (kept alive) via weak references.

Finalizer Statistics

Heap dump contains no java.lang.ref.Finalizer objects.
IBM VMs implement Finalizer differently and are currently not supported by this report.

Map Collision Ratios

Detected the following maps with collision ratios above 80%:

- 2 instances of **java.util.HashMap** retain **>= 752** bytes.

5. Dominator Tree

Class Name	Shallow Heap	Retained Heap	Percentage
<Regex>	<Numeric>	<Numeric>	<Numeric>
org.apache.harmony.xnet.provider.jsse.TrustManagerImpl @ 0x4009cb60	24	182,320	9.24%
class com.ibm.icu4jni.util.Resources$DefaultTimeZones @ 0x401fce78 System Class	8	165,432	8.38%
class android.text.Html$HtmlParser @ 0x4018e040 System Class	8	126,592	6.42%
org.bouncycastle.jce.provider.BouncyCastleProvider @ 0x4009b8c0	112	69,064	3.50%
class org.apache.harmony.security.fortress.Services @ 0x4008c340 System Class	24	51,456	2.61%
class android.content.res.Resources @ 0x400538a8 System Class	32	37,792	1.92%
class org.apache.harmony.luni.platform.PlatformAddress @ 0x4000d1d0 System Class	24	20,488	1.04%
min3d.core.Object3dContainer @ 0x46e4dea0	88	19,272	0.98%
char[7938] @ 0x40076d80 Africa/AbidjanAfrica/AccraAfrica/Addis_AbabaAfrica/Algiers	15,888	15,888	0.81%
class org.apache.harmony.security.utils.AlgNameMapper @ 0x4023f018 System Class	24	15,824	0.80%
android.content.res.StringBlock @ 0x40247648	32	12,616	0.64%
class com.android.internal.util.HanziToPinyin @ 0x401e9050 System Class	32	12,272	0.62%
class org.apache.harmony.luni.internal.net.www.protocol.jar.JarURLConnectionImpl @	8	12,224	0.62%
dalvik.system.PathClassLoader @ 0x46c63980	48	12,032	0.61%
android.widget.ListView @ 0x46ec6908	776	11,704	0.59%
class android.R$styleable @ 0x40019918 System Class	4,416	10,584	0.54%
class com.android.internal.R$styleable @ 0x400447e0 System Class	4,480	10,544	0.53%
com.android.internal.policy.impl.PhoneWindow @ 0x46c6c4f8	176	9,664	0.49%
class org.apache.harmony.luni.internal.util.ZoneInfoDB @ 0x4005ec50 System Class	56	9,600	0.49%
class org.bouncycastle.crypto.engines.AESFastEngine @ 0x402392b0 System Class	64	9,064	0.46%
class com.ibm.icu4jni.util.Resources @ 0x4001ff18 System Class	24	8,744	0.44%
class com.android.internal.telephony.GsmAlphabet @ 0x401ec900 System Class	48	8,688	0.44%
com.virtual.room.durio.AudioControl$playSoundTask @ 0x46ee2488	48	8,272	0.42%
org.bouncycastle.jce.provider.JDKKeyStore @ 0x400d54c0	16	8,080	0.41%
class android.media.MediaFile @ 0x4006e090 System Class	256	7,832	0.40%
class java.lang.System @ 0x4000f9d8 System Class	24	7,672	0.39%
min3d.objectPrimitives.SkyBox @ 0x46ee7300	104	7,272	0.37%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x40156218	24	7,216	0.37%
class java.lang.Integer @ 0x4000ec10 System Class	56	7,104	0.36%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x4014f440	24	6,928	0.35%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x40154c68	24	6,624	0.34%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x40136b48	24	6,608	0.33%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x4011a0e8	24	6,536	0.33%
android.os.Message @ 0x46c669d8	56	6,440	0.33%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400abe98	24	6,376	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400f1bd0	24	6,272	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400d3238	24	6,256	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x401024f8	24	6,232	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400cbaa8	24	6,224	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x400f0da8	24	6,224	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x4013a588	24	6,224	0.32%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x401630c0	24	6,144	0.31%
org.bouncycastle.jce.provider.X509CertificateObject @ 0x40134728	24	6,112	0.31%

Figure 47. Dominator Tree 4

5.14.5.7 Memory Analysis Conclusion

1. Result for every parameter nearly the same, unless there are two which have components bigger than 1 percent of the total heap.
2. Two parameters which have problems with memory waste contains parsed .obj 3D object (Home Theater set). Problems that exist mostly about referencing.
3. After reducing the resolution of textures, **android.content.res.Resources** memory allocation also slightly decreased.
4. System Classes from Android API are dominating the percentage of heap.
5. If parsed 3D object is added into the scene, recognized User Class with the higher consumption is min3D Object3Dcontainer, followed by StringBlock, android.widget.listView, AudioControl PlaySoundTask, and then SkyBox.
6. Without 3D object added into the scene, recognized User Class with the higher consumption is StringBlock, android.widget.listView, AudioControl PlaySoundTask, and then SkyBox.
7. Unfortunately, sound's performance is still unstable despite of manipulations on the visualization, but the delay seems slightly disappeared with reduced textures' resolutions and without additional object.
8. Since it is a memory analyzing, another activity outside that affect the application will also be displayed. With so many System Classes consume huge amount of memories, it echoes Froyo's bad memory management issues.

5.15 Problem Solving

Existing problem mostly caused by memory leak, badly affected the application performance. From memory analysis we also get the amount of memory which allocated into the application. Since it is only 1.9 MB, it shouldn't be a problem because inside the test device there is 512 MB amount of RAM. Indeed it is shared with the graphic processor unit for about 128 MB but still with 384 MB remains the application should work flawlessly.

Based on both previous analysis activities I prepared 2 possible solutions for each result:

1. On the profiling analysis I discovered that OpenGL classes always be the biggest CPU consumer. There are two options to minimize CPU consumption: reduce the amount of 3D object loaded and reduce the quality of the texture. First option definitely out of consideration because the objects itself are a part of requirements. Second option would make sense, because reducing texture quality wouldn't be much notable.
2. Android has many "memory management" applications out there, which would be useful to free some RAM so it will give more space to this heavy-loaded application.

5.15.1 Profiling Follow Up: Reducing Texture Quality

It would be nice to have high definition textures on this 3D visualization, but then I have to think twice if it's sacrificing the whole performance. Some object texture's quality can be reduced a bit to raise the sound quality. Home theatre set texture is attached into the object, so I can't manipulate it. Rests are possible. I use Paint application to resize the texture into 50% of actual size, whilst still maintaining the aspect ratio.

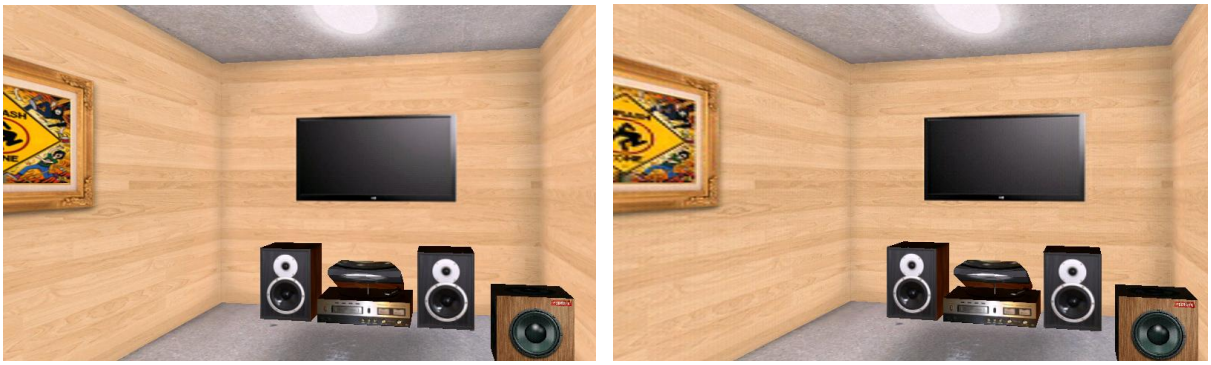


Figure 48. Texture Quality Comparison

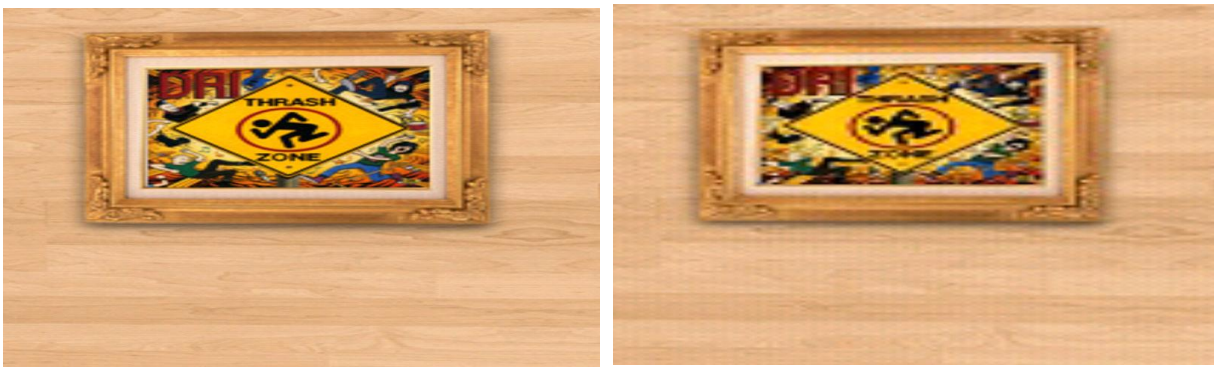


Figure 49. Detailed Texture Quality Comparison

The 3D scene after quality reduction definitely becomes worse than before. We can examine figure. where the frame hanging on the wall becomes pixelated. Also the wall color becomes paler. It's not a big deal because at least the sound quality improved significantly. Playback delay was getting shorter and the crack noises were slightly gone. I also got faster application start-up time, from about 5 seconds to less than 3 seconds.

5.15.2 Memory Analyzing Follow Up : Memory Booster

Android has a bunch of task killer applications on their market. Mainly, task killer is used to free some memory spaces by killing background applications, so it will create a space for another applications coming. Simple application like clock or messaging of course won't be bothered by amount of RAM remaining, but 3D scene + Native Decoder + Accelerometer Sensor of course could causing a havoc.

Randomly I found (and tried) Memory Booster Lite application just to figure out whether it is helpful or not. This application has 2 main functions; Monitoring how the Memory goes and also killing certain selected apps in order to release some free memory.

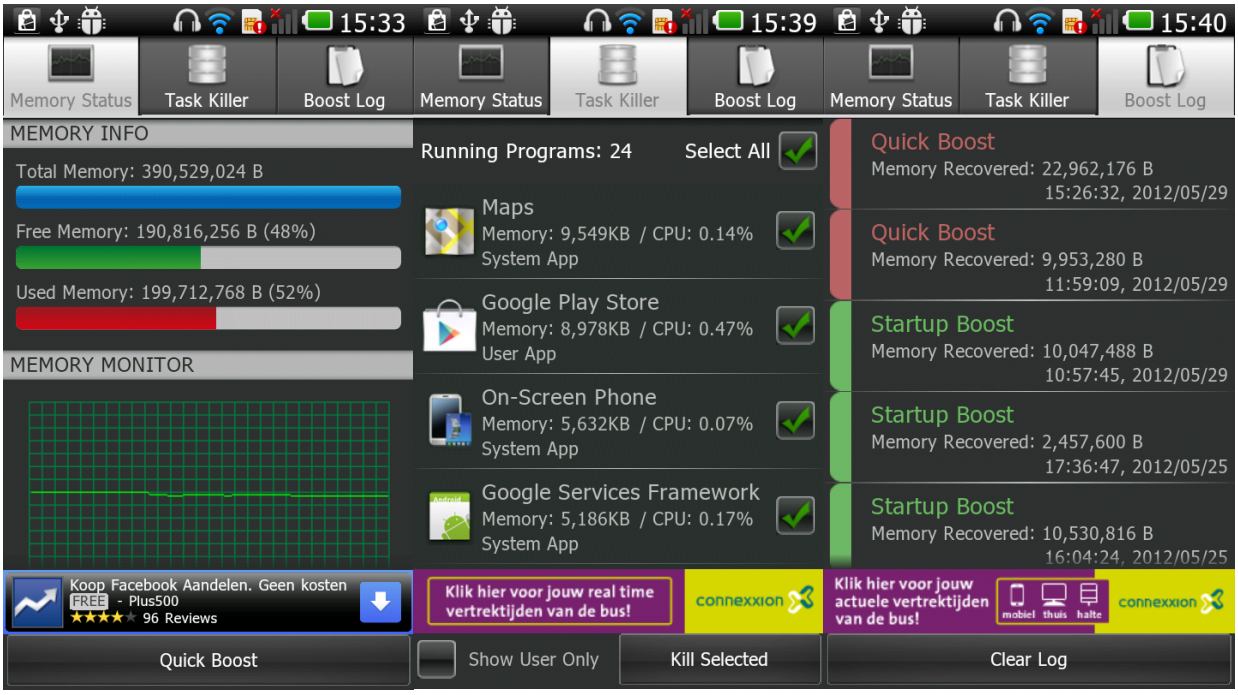


Figure 50. Memory Booster Lite

Memory monitor gives you detailed calculation of free memory alongside with used memory. Task Killer is on the second tab, gives you a list of running apps. This list is sorted based on amount of memory allocated for relevant application. As we can see Google Maps absorb almost 10 MB of memory, and as long we don't use it in our application we can kill it. It gives notable improvement on the sound quality, sweeps almost all delays and cracks on the playback. Boost Log tab displays what activities that we've been done since the first time we installed this application.

CHAPTER 6: CONCLUSION & RECOMMENDATION

People are getting smarter time after time. One that indicated previous statement is how they virtually simulate a system which is usually done physically. Surround sound simulation decreases general difficulty of setting traditional multichannel audio, with less of hardware and less of wire. Which means it's possible for anyone who wants to feel a surround sound experience despite of lack of space. Meanwhile, massive expansion also occurs on Mobile Phone development. Android OS has been begun to rule the world of smart phones with its open source community. And once a groundbreaking idea to have virtual surround room on Android appeared, could it be truly implemented?

After approximately 6 months of research I found it possible to create a mini 3D surround room inside Android environment, so the recent answer is "yes". Android has a graphic machine called OpenGL ES, and the observation had brought me to a simple yet useful library which could help a lot in building 3D scene without draw it line-per-line. Sensor movement also can be realized thanks to Android accelerometer. Scene camera can be moved around depend on how you tilt the phone. Within the accelerometer gravitation concept it's also possible to zoom in and out the room. I also imitated a 5.1 surround sound room setup from internet to fix my wrong room setup before, which is very likely to happen if you have almost zero of audio knowledge like me.

Surround Sound "MPEG Decoder ™ " is a reliable Android surround sound simulation application provided by my company. Although it's using low-level native programming language decoder, I can import it and it is well coupled with mine. But here's the problems rise. Native library diminished compatibility feature that Java has, so the application will be defected or even corrupted when I try to run it on another operating system. The compatibility problem is followed by performance issue where some noises and cracks are distracting sound playback. Heavy loaded 3D graphic + native decoder + accelerometer combination seem enough to produce major performance issue.

Unfortunately, compatibility problem seems unlikely to be solved, as it is sourced from the environment. From this case we can learn something that if we want to develop application with native library we have to decide what operation system exactly will be used at the very first time. Otherwise all performance problems completely spotted after some testing and analysis. 3D scene indeed consumes most of the phone capabilities and resources. Hence, possible solution could be decreasing the quality of the scene to boost performance and thankfully it works like a charm. Finally playback sound and 3D scene are coped well without any faults, so then I can conclude that my application has met all the requirements apart from compatibility problem. To finish this application, further development phase will try to implement different sound effects for different head positions. Overall it was fun to develop an Android application. Moreover, now I can say agree when someone said there're so many spaces on Android development. With various features available to be explored and big community ready to give you a hand anytime, you can bring your wildest imagination into a mobile app, no matter how hard it is, because as a programmer we have to find any possible and impossible ways to solve every single problems and errors. Don't only imagine you have a virtual surround room inside your phone, but give your best effort to make it true.

EVALUATION

Actually this is my first experience of developing a real Android application. What I created before were just mini applications with very limited function. Overall I enjoyed this research very much because I have big passion on this mobile phone field. Hence, I can learn new things easily because I've already known about Android device ins and outs. The development went smoothly. After only several weeks I've generated a working application, but then I met performance problem which is never occurred before on my previous application. This took a long time to be resolved and causing desperation. But then I spotted the problem after optimizing analysis tools on Eclipse IDE and fixed it to finish my research. After about 3 months of intensive documentation work finally I've done this report. It was a bit out of methodology because on the first 3 months I only focused on the software without writing any documentation. I only made a little footage about my work and it's very general so maybe I missed some detailed parts of this project. In the end It's became a huge problem when I need to review some parts of experiment because there's no records about it. Thanks to Mr. Zijlmans for reminding me this essence of writing a report. For me this could be a valuable lesson. On the next research I won't forget to write every single step and every single change I've made. I found it very important to track our research by writing it because it's too naïve if I just rely on my own memory. Last but not least, I got priceless experiences here, especially on how to deal with real software development environment. I couldn't ask for more, this was the biggest first step I possibly take and now I can look forward for my career further.

REFERENCE LIST

[1] http://en.wikipedia.org/

[2] http://developer.android.com/index.html

[3] http://www.mat-d.com/site/tutorial-load-a-3d-obj-model-with-min3d-for-android/

[4] http://code.google.com/p/min3d/

[5] http://www.rozengain.com/blog/

[6] http://insanitydesign.com/wp/projects/nehe-android-ports/

[7] http://stackoverflow.com/

[8] http://www.srombauts.fr/android-ndk-r5b/docs/ANDROID-MK.html

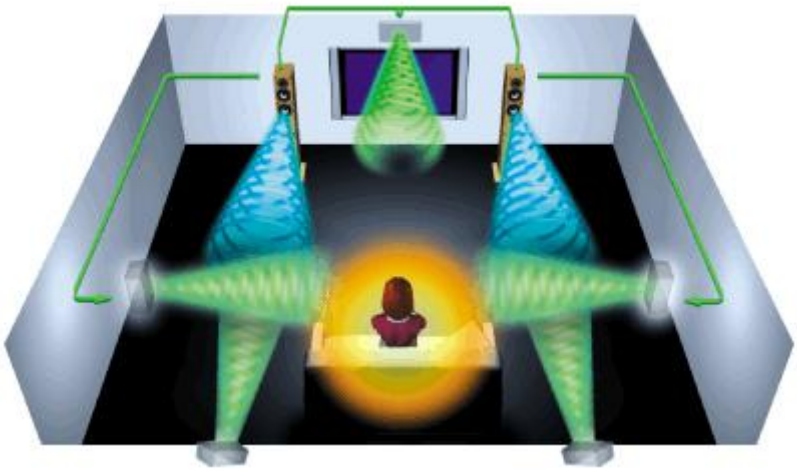
[9] **https://plato.natlab.research.philips.com/svnroot/android/trunk/docs/Android%20guide-17November-Philip-Final.pdf**

[10] http://www.eclipse.org/mat/

[11] http://www.the-home-cinema-guide.com

Project Plan

Virtual Surround Room on Mobile Phone



Mentor : Yuan Zhaorui

Developer : Durio Etgar

Contents

1. Project Introduction

Android phones and tablets are being widely used nowadays. In order to simulate a virtual surround room on an handy stuff, Philips Applied Sensor Technologies wants to develop android user interface that can show us how the virtual surround room works. In this application we can navigate around a room with several speakers in it and also hear the sound changing when we make a move to another direction. The angle of our head movement will determine how loud the sound that we hear. It will also be very useful when we want to do a demonstration, because nowadays Android phones have a HDMI port so we can connect it into a big screen. We will maximize the OpenGL graphic engine which already embedded in every Android phones. Based on the contract, the project starts in February 6th, 2012 and terminates in June 29th, 2012. Duration of the project will be approximately 5 months.

2. Project Statement

This sub defines the problem to be solved so that work can begin on the project. It is useful for providing an initial understanding on the scope of the project. After getting some insights through the pre-cases we hope there will be better preparation and less risk during the project.

2.1 Formal Client

Philips is a well-known worldwide company who works with variety technologies which are advanced, easy to use, and designed around the needs of all our users.

Another fact about Philips:

- Founded in 1891
- Headquartered in Amsterdam, The Netherlands
- One of the largest global diversified industrial company with sales in 2011 of EUR 22.6 billion
- A multinational workforce of 121,888 employees (January 2012)
- Globally present with manufacturing sites in 100 countries and sales outlets in 100 countries
- An R&D force with expenditures of EUR 1, 6 billion (2012)

The company itself has many divisions spread all around the world. Exactly, I work at:

Philips Research - Information and Cognition - Applied Sensor Technologies
High Tech Campus 36, 5656 AE Eindhoven, The Netherlands
Building HTC36-p.058, High Tech Campus 36, 5656 AE Eindhoven, The Netherlands

2.2 Project Organizational

Project Leader : Werner Oomen. Technology specialist

Tel: +31 40 2740235

Fax: +31 40 2742630
+31649704005

E-mail: werner.oomen@philips.com
Web: www.apptech.philips.com
<http://www.apptech.philips.com/>

Company Mentor : Zhaorui Yuan. Software Designer

Tel: +31 40 2741756

Fax: +31 40 2744108

E-mail: zhaorui.yuan@philips.com

Web: www.research.philips.com

2.3 Current Situation

Philips Research - Information and Cognition - Applied Sensor Technologies are currently focusing on virtual surround sound development. They have an exclusive room for demonstration and also application both for desktop and mobile platform. What they want to build next in order to extend their project is a visualization based on it; a 3D room that represents raw demonstration on Android device.

With the surround sound application already provided by the company, the visualization app can be coupled through imported native interface. Moreover, head movement can also be simulated thanks to the help of Android device accelerometer sensor. The boundary has been approved and it said that I will work only on the graphic section without any involvement on the surround sound decoder. Final product would be a 3D room visualization with surround sound behind it, and the sound will be changed based on the device movement.

2.4 Project Justification

This project was rising along with the hype of smartphone devices. Philips itself also interested in exploring further possibility, especially on Android devices as they're developer friendly and also built in open source environment. There are several Philips applications which already published to Android Market with two main intentions; to promote and to extend the functionality of their products. For instance, we can find Philips TV Buying Guide app on the Android Market which virtually render their TV product on your gadget's camera so you can estimate which TV to buy and how to place it properly inside your house. There also an application called Philips My Remote that could help you to simulate a TV remote via mobile devices.

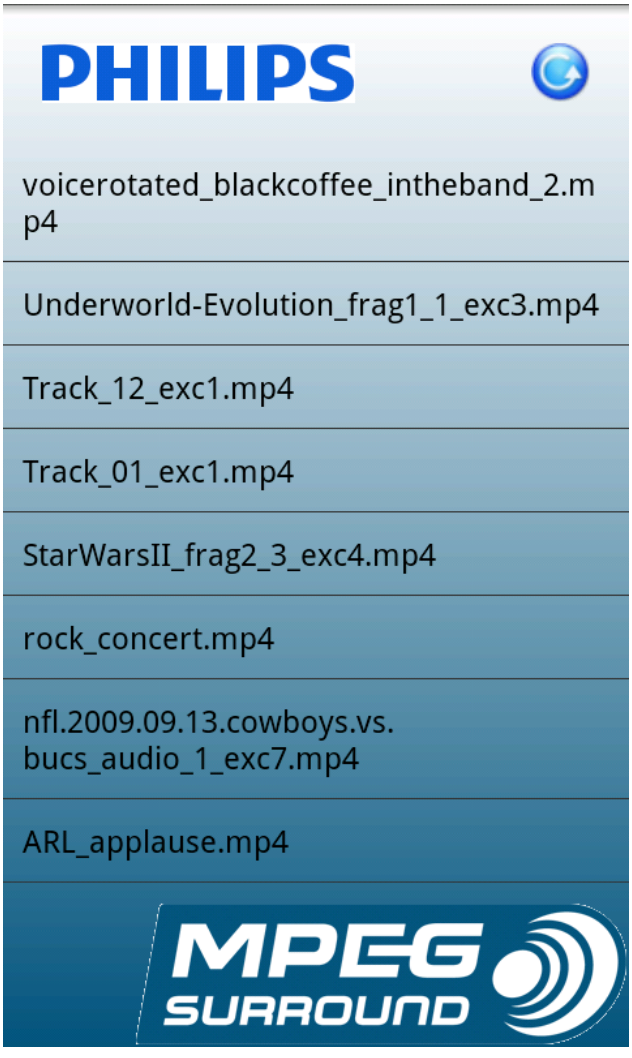


Figure1. Philips MPEG Surround Application

Philips Research - Information and Cognition - Applied Sensor Technologies division consider this opportunity to develop a new application based on their existing MPEG

surround application. The main idea is to give a little simulation on how virtual surround works without having a real demonstration, simply to make a portable one. The goals would be a 3D room application coupled with surround sound behind it.

2.5 Project Product

- User Requirement Document
- Project Plan
- Design Document
- Virtual Listen Room Application on Android Devices
- Final report

2.6 Project deliverables and non-deliverables

- User Requirement Document

Describe all of user requirements that have been defined early. Consist of implemented functions and features.

- Project Plan

Generally describe how the project being planned with the details required.

- Design Document

Describe the initial design phase represented by various graphs and diagrams.

- Virtual Listen Room Application on Android Devices

A running application that simulates surround sound effect perfectly inside mobile devices.

- Final Report

A document that covers everything related to the project.

2.7 Project Constraints

- Built inside of Android environment
- Using OpenGL ES for graphic rendering
- Using Java and Native Interface (C/C++)
- Implemented on Mobile Phone
- Coupled with MPEG Surround Player
- Accelerometer sensor integration

2.8 Project Risks

Project Risks			
Risks	Impact Level	Occurrence	Mitigations
Insufficient Hours of work.	High	Unlikely	Arrange time allocation plan, try to work based on it.
Lack of OpenGL knowledge	High	Likely	Looking for a book or reference related to the terms. Ask the expert.
Lack of Knowledge about Software Development Life Cycle	Medium	Likely	Consult with Company mentor and school tutor and do more research.
Lack of Communication with both Mentors	Medium	Unlikely	Create a communication plan to arrange it, either frequently or maybe occasionally.

3. Project Phasing

The project consists of the following phases:

Phase	Activities	Sub-deliverables
Initiative phase	- Starting up the project - Organizing the workspace - Interview with the mentor - Make a project plan - Search information about OpenGL	- Project Plan (Fixed) - Documentation about the knowledge of the OpenGL - Weekly report updated
Definition phase	- Do an investigation for OpenGL - Discuss the result with the mentor - Make a report	- Project Plan (Fixed) - Weekly report updated
Design phase	- Make an OpenGL desktop version design - Discuss it with the Mentor - Make a report including Design Document	- Weekly report updated - C++ OpenGl Prototype
Implementation phase	- Port the desktop version into Android Platform - Add the navigation function - Add the manipulated sound - Make a report	- Weekly report updated - Android Application Prototype
Transfer phase	- Presenting the final product - Presentation	- Final report - Final presentation

2.8 Project Plan and Schedule

ID	Activity	Duration	Start at	Finish at
Initiative Phase		**13 days**	**06-02-12**	**16-02-12**
1	Instalation the workspace	1 day	06-02-12	07-02-12
2	Meeting with mentor to discuss the Project	1 day	07-02-12	08-02-12
3	Self-work - Writing a draft of a Project Plan	2 days	07-02-12	09-02-12
4	Formal meeting - Discussing the Project Plan	1 day	09-02-12	10-02-12
5	Formal meeting - Discussing of the Project Plan	1 day	11-02-12	12-02-12
6	Self study – Search OpenGL code that can be used in Android	4 days	07-02-12	11-02-12
7	Formal meeting – First version Project Plan	1 day	13-02-12	14-02-12
8	Delivery of the final version of the Project Plan	1 day	14-02-12	15-02-12
9	Meeting with client - Approval of the Project Plan	1 day	15-02-12	16-02-12
Definition Phase		**20 days**	**16-02-12**	**07-03-12**
10	Self –work - summarize the brainstorming	2 days	02-03-11	02-07-11
11	formal meeting - requirement for interview	1 day	08-03-11	09-03-11
12	Self- work for the requirements generated	5 days	10-03-11	17-03-11
13	formal meeting – discuss the functions	1 day	18-03-11	21-03-11
14	Self-work – report updated	2 days	22-03-11	24-03-11
15	Self – work – benchmarking	3 days	25-03-11	07-03-12
Design Phase		**20 days**	**07-03-12**	**27-03-12**
16	Self- work –select and determine the implementable functions	1 day	30-03-11	31-03-11
17	Self-work - Specifying data and functions in report	5 days	01-04-11	07-04-11
18	Self-work – draft the user interface	5 days	08-04-11	15-04-11
19	formal meeting - Specifications reviewing	1 day	18-04-11	19-04-11
20	Self- work- determine the user interface	2 day	20-04-11	22-04-11
21	formal meeting - Documents reviewing	1 day	25-04-11	26-04-11
22	Delivery of the Process Document and report updated	1 day	27-04-11	27-03-12
Implementation Phase		**40 days**	**27-03-12**	**06-05-12**
23	Specifying the class diagrams and sequence diagrams	10 days	29-04-11	12-05-11
24	Self-work – code designing	25 days	13-05-11	17-06-11

25	Formal meeting – discuss the program problem	1 day	20-06-11	21-06-11
26	Self-work –fixing possible problems and errors	2 days	22-06-11	06-05-12
Transfer Phase		**2 days**	**27-06-12**	**30-06-12**
27	Formal meeting –discuss the final product	1 day	27-06-12	28-06-12
28	Self – work –write the final report and prepare the presentation	1 day	29-06-12	30-06-12

4. Project Management Plan

To achieve the project deliverables we have limited resources. To guard these resources and make sure that we get a high quality, we develop a management plan. In this plan six elements are discussed:

- **MO**ney
- **S**kills
- **Qu**ality
- **I**nformation
- **T**ime
- **O**rganisation

known as **MOSQUITO**.

Since this is a non-profit project, there's no need to explain the Money element.

4.1 Skills

Some skills required to have in this project are:

Personal known skill:

- Programming skill in Java on preferable windows environment
- Programming skill in Android Platform (Android Software Developing Kit)
- Programming skill in OpenGL
- Programming skill in native language (C/C++)
- Some experience with requirements engineering and software design
- Communication and Organizational skill

Learning skill:

- Working platform: Java in Eclipse 3.7 (Indigo)
- General concepts in Android OpenGL
- Basic knowledge about “UI programming“
- Knowledge of virtual surround concept
- Knowledge of sensor motion event
- Experience with sound and coordinate manipulation
- Get native functions with Java Native Interface
- Android Native Developing Kit
- Knowledge of Cygwin Unix terminal emulator
- Preferably some experience with “UI programming”

4.2 Quality

There are several parameters to indicate a successful graduation project:

- Fully-implemented and working application
- Well-structured documentation
- Enough knowledge of working environment
- Improvement on socialization
- Improvement on presentation skill
- Improvement on Programming skill
- Discover many innovations based on the research result

4.3 Information

This part will describe how the whole document being managed. It also describes the flow of information and attaches role for each party whom contributed.

	Project Plan	User Requirements	Design Document	Day Declaration	Final Reports
Trainee **Durio Etgar**	Dr, D	Dr, D	Dr, D	Dr	Dr, D, Ar
Project Leader **Mr. Werner Oomen**	D, A	Dr, D, A	Dr, D, A	A	D, A
Company Mentor **Ms. Zhaorui Yuan**	D, A	D, A	Dr, D, A	A	D, A
School Tutor **Mr. Jack Zijlmans**	D, A, R	-	-	-	D, A, R
Fontys Secretariat	R, Ar	-	-	R, Ar	R, Ar
Internship Office **(Stagebureau)**	R, Ar	-	-	R, Ar	R, Ar

Terms:

- Dr = Draw up
- D = Discuss
- A = Approve
- R = Receive
- Ar = Archive

4.4 Time

- Startdate: 6th February 2012
- End date: 29th June 2012
- Total working hours: approximately 90 days (720 hours of works)

4.5 Organization

- Project Leader : Mr. Werner Oomen

  Job description :

  - Set up the initial idea of the Project
  - Define general requirements
- Company Mentor: Mrs. Zhaorui Yuan

  Job description :

  - Guiding and supervising the whole project
  - Define specific requirement
  - Giving advices on every technical aspects
- School Tutor : Mr. Jack Zijlmans

  Job description :

  - Guiding and supervising the whole project
  - Giving remark on the progress and documentation
  - Visiting the company at least twice
- Trainee : Durio Etgar

  Job description :

  - Work on the project based on the requirements.
  - Writing all the documents needed
  - Doing research to solve problems
  - Managing a communication plan between all parties

4.6 Communication Plan

From	To	Subject	Via	Week/Frequency
Student Trainee	School Tutor	Progress	Email	Once Every 2 weeks
Student Trainee	School Tutor	Project Plan	Email	Week 1-3
Student Trainee	Company Mentor	Project Discussion	Meeting	Week 1
Student Trainee	Company Mentor	Progress and Technical Discussion	Meeting	Week 2
Student Trainee	Company Mentor	Project Information and Material	Meeting, Email	If Necessary
School Tutor	Student Trainee	Project Plan Feedback	Email	Week 4
School Tutor	Student Trainee	Company Visit	Meeting On the Site (Company)	At least twice
School Tutor	Company Mentor	Project Iteration	Meeting	• Start at Week 10 • Every 2 weeks
Student Trainee	School Tutor	Report and Documentation Progress	Email	• Start at Week 8 • Every 2 weeks
School Tutor	Student Trainee	Report and Documentation Feedback	Email	Occasionally
Student Trainee	School Tutor	Problems	Email/Phone	If problem(s) occurs
Student Trainee	School Tutor	Presentation Rehearsal	At school	Week 17/18