3D Room Visualization on Android Based Mobile Device (with Philips™' Surround Sound Music Player)

Durio Etgar[1], Wahyul Amien Syafei,[2], Jack Zijlmans[3], Zhaorui Yuan[4]

[1]Electrical EngineeringDiponegoro University
Jl. Prof Sudharto, Tembalang, Semarang, Indonesia.
[2]Fontys University of Applied Sciences
Rachelsmolen 1, 5612 MA Eindhoven, The Netherlands.
[3]Philips Research
High Tech Campus 36, 5656 AE Eindhoven, The Netherlands.

[1]darno90@gmail.com;
[2]wasyafei@yahoo.com;
[3]j.zijlmans@fontys.nl;
[4]zhaorui.yuan@philips.com

ABSTRACT *- This project's specifically purposed as a demo application, so anyone can get the experience of a surround audio room without having to physically involved to it, with a main idea of generating a 3D surround sound room scenery coupled with surround sound in a handier package, namely, a "Virtual Listen Room". Virtual Listen Room set a foundation of an innovative visualization that later will be developed and released as one of way of portable advertisement. This application was built inside of Android environment. 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. Further, Android has a Accelerometer Sensor that is needed to be coupled with scene to produce a dynamic movement of the camera. Surround sound effect can be reached with a decoder from Phillips called MPEG Surround Sound Decoder. To sum the whole project, we got an application with sensor-dynamic 3D room visualization coupled with Philips' Surround Sound Music Player. We can manipulate several room's properties; Subwoofer location, Room light, and how many speakers inside it, the application itself works well despite facing several performance problems before, later to be solved.*

[Keywords : Android,Visualization,Open GL; ES; 3D; Surround Sensor]

I. INTRODUCTION

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[1] 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. This kind of sound setup has been used widely, if I can't say it's almost everywhere. That's why surround sound was chosen as the music player's decoder.

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". It's specifically purposed as a demo application, so anyone can get the experience of a surround audio room without having to physically involved to it. Virtual Listen Room set a foundation of an innovative visualization that later will be developed and released as one of way of portable advertisement. This application, even the idea, had never been invented before.

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[1] Later this application is coupled with the 3D scene, and while listening to the playback music we can tilt our scene camera view and

Player. We can manipulate several room's properties; Subwoofer location, Room light, and how many speakers inside it.

II. BASIC THEORY

2.1Android [1]

Android is a Linux-based operating system for mobile devices such as smartphones and tablet computers. It is developed by the Open Handset Alliance, led by Google.Android has a large community of developers writing applications that extend the functionality of the devices. Developers write primarily in a customized version of Java. In June 2012, there were more than 600,000 apps available for Android, and the estimated number of applications downloaded from Google Play was 20 billion.

Android became the world's leading smartphone platform at the end of 2010. Analysts point to the advantage to Android of being a multi-channel, multi-carrier OS. Beside that, Android Operating System can be easily planted into various kind of device. We can prove it in Figure 1 below

.

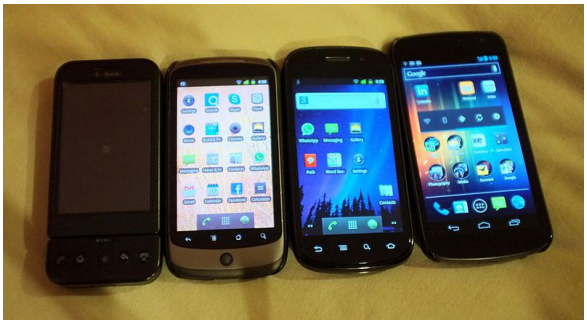

Fig 1. Android Phones

From left to right: Verizon, HTC, Samsung, and Google's Galaxy Nexus. We can pull a conclusion that Android can be widely implemented, with many consider it as an immense advantage.

2.2 Surround Sound [1]

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. 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 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, so this app could be interesting for anyone. Also with the fact that this project was a pure invention. There's no one has done this before on a mobile device. To sum the whole project result, we got an application with sensor-dynamic 3D room visualization coupled with Philips' Surround Sound Music 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. This simulation based on 5.1 Surround room arrangement. This picture below (Figure 2) is a raw representation of 5.1 surround sound setup with 6 sound resources coming from different angles.

Fig 2. Surround Sound Simulation

2.3Open GL ES [1]

OpenGL for Embedded Systems (OpenGL ES) is a subset of the OpenGL graphics application programming interface (API) designed for embedded systems such as mobile phones, PDAs, and video game consoles.

- OpenGL ES for Android: version 1.0, 1.1 and 2.0
- Possible to create 2D and 3D object
- Reduced function from OpenGL because of limited hardware

OpenGL ES' 3D capability is often used for games and visualizations. For instance a racing game in Figure 3 maximizes OpenGL ES 3D engine.

Fig 3. OpenGL ES' 3D Example

On the other side we also able to create 2D graphic with OpenGL ES, commonly for building games or user interfaces. Figure 4 represent a simple 2D triangle object.

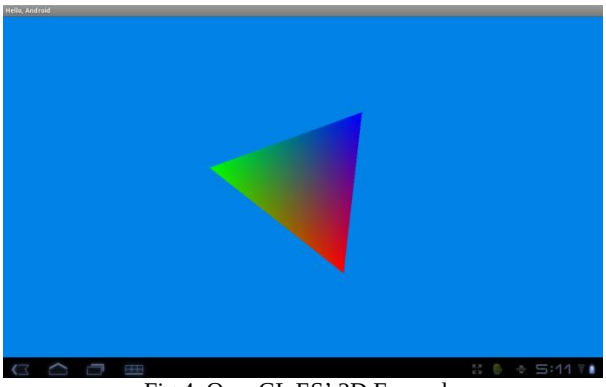

Fig 4. OpenGL ES' 2D Example

2.3.1 Min3D [2]

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
- 3DS
- MD2

2.4 Accelerometer [3]

Accelerometer is a device that measures weight per unit of (test) mass. Android applies this accelerometer as a sensor which is detecting how you tilt your phone. This sensor is used in almost all of Android devices nowadays.

Programmatically we can take the advantages of *Sensor.TYPE_ACCELEROMETER* Class on Android to make use of the Sensor's values. Here are the physic explanations on how can we get accelerometer value*:*

All values are in SI units ($\frac{m}{}$)

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. Conceptually, it does so by measuring forces applied to the sensor itself using the relation:

$$Ad \quad - \quad \frac{}{ma}$$

Ad$_v$is the value of acceleration applied to the device inside vacuum room, while **Fs** is theforces applied to the sensor itself. **mass** is thedevice's mass.

In particular, the force of gravity is always influencing the measured acceleration, as expressed in the following formula:

$$Ad \quad - \quad -Ad$$

Ad$_e$is the value of acceleration applied to the device with influence from earth gravity, while **g** is the constant force of gravity (9.81 $\frac{m}{}$)

For this reason, when the device is sitting on a table (and obviously not accelerating), the accelerometer reads a magnitude of **g** = 9.81 $\frac{m}{}$

Similarly, when the device is in free-fall and therefore dangerously accelerating towards to ground at 9.81 $\frac{m}{}$, its accelerometer reads a magnitude of 0 $\frac{m}{}$.

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 $\frac{m}{}$) minus the force of gravity (-9.81 $\frac{m}{}$).
- When the device lies flat on a table and is pushed toward the sky with an acceleration of A $\frac{m}{}$, the acceleration value is equal to A+9.81 which correspond to the acceleration of the device (+A $\frac{m}{}$) minus the force of gravity (-9.81 $\frac{m}{}$).

2.5 Proper Audio Room Setup [4]

Recommended room setting for 5.1 surround room system:

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).

Fig 5. Recommended Front Speaker Placement

2. 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.

Fig 6. Subwoofer Speaker Placement

III. BUILDING APPLICATION

This application was built using Eclipse IDE, a Java compiler that directly recommended by Google itself for every Android Developer. To stretch OpenGL ES capacity into the edge, 3D engine for Android called Min3D is used.

3.1 Min3D Setup[5]

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.
   Now you can call any method from min3d library

3.2 Build a Skybox[6]

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 7 below is a complete skybox with wall and floor textures so we could get a "room" impression.

Fig7. Skybox

3.3 Build the Subwoofer

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 6 3D cubes are needed for later being rendered as a Subwoofer. We have to also attach texture into a plain 3D object to bring a real impression.

Fig8. Speaker Texture

Figure 8 above is a wood speaker texture which is later will be used as subwoofer. The scale of width and length must be 1:1 to make it completely matchs the 3D box object.
\

3.4Add Premade Object[7]

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.

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. For instance, We found this premade audio set (Figure 9) with 3 speakers, later will be given roles as center, front left, and front right speakers.

Fig9. 3D Object Example

3.5 Standing Rear Speakers

Standing speaker can be created based on basic block shape, which is also can be reached by manipulating the scale of cube. Hence, We decided to turn 2 cubes into blocks. The texture also need to be adapted so It's got edited using Photoshop.

3.6 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 only y and z axis are used, because the scene was set on landscape by default to gain wider scenery.

3.7 Room Setup

This room setup was made according to Proper Audio Room Setup**.** On the room's front side there are an LCD screen, 2 pictures a furnishes, and an audio set as center, front left, and front right speakers with subwoofer as we can see in Figure 10.

Fig10. Front speakers and Subwoofer Arrangement

To finish the whole visualization, as being represented in Figure 11, 2 standing speakers were added and act as Rear left and Rear right sound resources.

Fig11. Rear speakers Arrangement

3.8. Philips' Music Player with MPEG Surround Sound Decoder

This is a kind of media player app which simulates surround effect on particular media file (.mp4 and .wav). It has several functions as listed: 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++. Below (Figure 12) is the Philip's Music Player interface.

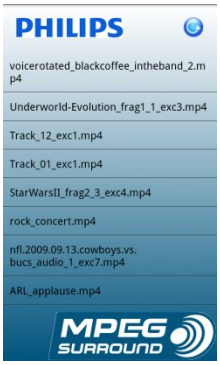

Fig12. Philips' MPEG Surround Decoder

IV. TESTING, ANALYSIS, AND PROBLEM SOLVING

4.1 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(Java Native Interface) 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. We'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, two ways of analysis were done; 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.

4.2 Profiling Activity

This analysis were taken using DDMS Traceview on Eclipse. We 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.

4.2.1 Profiling Summary

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 the application start we got different performance. We 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.

4.3 Memory Analysis

This analysis were taken using Memory Analyzer Tools on Eclipse. We 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.

4.3.1 Memory AnalysisSummary

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, Audio Control Play SoundTask, and then Sky Box.

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

4.4 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 we prepared 2 possible solutions for each result:

1. On the profiling analysis, it's 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.

4.4.1 Profiling Follow Up: Reducing Texture Quality

It would be nice to have high definition textures on this 3D visualization, but then developer 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 It can't be manipulated. Rests are possible. Paintapplication was used to resize the texture into 50% of actual size, whilst still maintaining the aspect ratio.

Fig13. Texture Quality Comparison

The 3D scene after quality reduction definitely becomes worse than before. We can examine Figure 13 and then Figure 14 where the frame hanging on the wall becomes pixelated. Also the wall color becomes paler.

Fig14. Detailed Texture Quality Comparison

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. Application start-up time also got faster, from about 5 seconds to less than 3 seconds.

4.4.2 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 developer 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.

Fig 15. Memory Booster Lite

Figure 15 shows us every single tab in Memory Booster Lite application. 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

V. CLOSING AND CONCLUSION

It's possible to fullfil the main idea of this project; create a mini 3D surround room inside Android environment. Android has a graphic machine called OpenGL ES, and the observation had brought 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. A 5.1 surround sound room setup was also imitated from internet to get a real impression of Audio room, even with a cinematic sound playback.

Surround Sound "MPEG Decoder ™ " is a reliable Android surround sound simulation application provided by Philips. Although it's using low-level native programming language decoder, It can be imported and it is well coupled with the visualization. But here's the problems rise. Native library diminished compatibility feature that Java has, so the application will be defected or even corrupted when it tried to be run 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 can be concluded that this 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.

Generally, it can be concluded that this application works as purposed. User can get the "portable" idea, especially after it's been ported to Android mobile device. Some room manipulation settings were added as an extra features like Subwoofer location, Room light, and how many speakers inside the scene. We got it was an entertaining experience after working with Android, and we kindly recommend this currently-hyped mobile development to all of software engineering out there.

References

[1] (2012) Wikipedia. [Online]Available:http://wikipedia.org/
[2] (2012) min3D Library. [Online] Available: http://code.google.com/p/min3d/
[3] (2012) Android Developer. [Online] Available: http://developer.android.com/index.html
[4] (2012) The Home Cinema Guide. [Online] Available: http://www.the-home-cinema-guide.com
[5] (2012) min3D Setup. [Online] Available: .http://code.google.com/p/min3d/setup/
[6] (2012) Rozengain Creative Technology Blog. [Online] Available: http://www.rozengain.com/blog/
[7] MatD (2012) Load a 3D OBJ Model with min3D for Android. [Online] Available: http://www.mat-d.com/site/tutorial-load-a-3d-obj-model-with-min3d-for-android/