ISSN No. 0976-5697

Volume 4, No. 6, May 2013 (Special Issue)

International Journal of Advanced Research in Computer Science

REVIEW ARTICAL

Available Online at www.ijarcs.info

Design of Game Application using Android Architecture

Miss. Shital V. Narlawar[*1], Miss. Sonal G. Chopade[2], Miss. Vaishnavi C. Vikhore[3] and Mr. Sandip A. Kahate[4]
B.E-I.T (Final Year)[1,2,3], Assistant Professor[4], Jawaharlal Darda Institute Of Engineering & Technology,
Yavatmal (MS) INDIA
snarlawar67@gmail.com[*1], sonalchopade5@gmail.com[2], vaishnavivikhore@gmail.com[3],sandip.kahate@gmail.com[4]

Abstract: Computer games and video games have become very popular for children and youths and play a prominent role in culture of young people. Games can now be played everywhere in technology-rich environments equipped with laptops, smart phones, game consoles (mobile and stationary), set-top boxes and other digital devices. So, we are focusing on architechture of android for designing a game application. Android is one such platform, where the whole operating system is open source, and application writing is supported by the programming language Java, free SDK and emulator, which are seamlessly integrated in the most used IDE, Eclipse. Because, most of the android applications are developed in Java language. The rapid development of electronic devices and network communication provides a foundation for improving the learning and teaching environments through technology. Game engine is the framework of the games. To develop a game engine, owns good performance will reduce repetitive work and technical difficulties effectively during game development process, considering the advantages and prospect of Android platform. Finally, the paper discusses the contribution from the aspects of the technology, game ideas and pedalogy.

Keywords: Android, Android architecture, Sheep Framework, XNA, Android SDK.

I. INTRODUCTION

Android is Google's Open Source Mobile Software Environment and is a software stack for mobile devices which includes an operating system, middleware and key applications. Since its official public release, android has captured the interest from companies, developers and the general audience. From that time up to now, this software platform has been constantly improved either in terms of features or supported hardware or, at the same time, extended to new types of devices different from the originally intended mobile ones. Google entered into the mobile market not as a handset manufacturer, but by launching mobile platform called as "Android" for mobile devices such as Smart phones, PDA and net books on 5th November 2007.

A. Android Background:

Android Architecture is shown in fig 1, which consists of number of layers as Applications, Application framework, Libraries, Android runtime & Linux kernel. Application layer is the uppermost layer which provides a set of core applications including an email, SMS program, calendar, maps, browser, contacts, and others. All applications are written using the Java programming language [1]. It should be mentioned that applications can be run simultaneously; it is possible to hear music and read an email at the same time. The Application Framework is a software framework that is used to implement a standard structure of an application for a specific operating system. With the help of managers, content providers and other services programmers it can reassemble functions used by other existing applications.Layer which is present below Application framework consists of two parts as Libraries which are all written in C/C++. They will be called through a Java interface. This includes the Surface Manager, 2D and 3D graphics, Media Codes like MPEG-4 and MP3, the SQL database SQLite and the web browser engine WebKit. Second part is Android Runtime which includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language. Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The lowest layer is Linux Kernel, Android basically relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

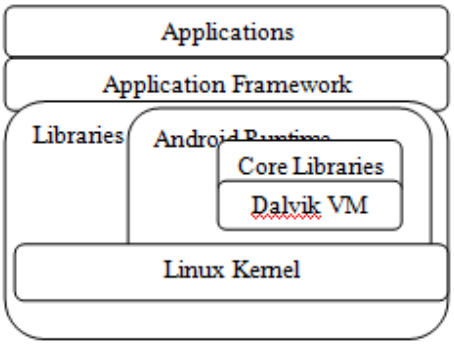

Figure 1: Android Architechture

a. Android Programming Framework:

The environment requires developing application for Android consists of the Android SDK [2], the Eclipse IDE and the Java Development Kit (JDK) which has to be preinstalled for the installation of both, Android SDK and Eclipse. The following versions of the tools mentioned above are used & presented in figure below. [3]

a. Android SDK
b. JDK: jdk1.6
c. Eclipse: eclipse 3.2

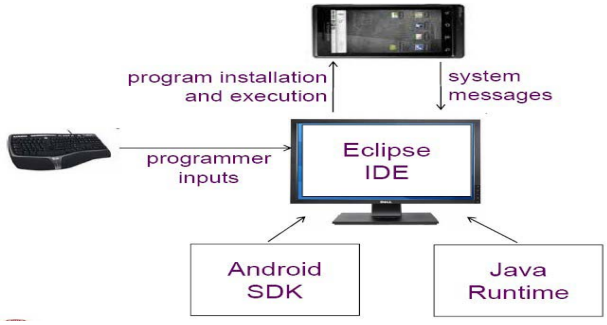

Figure 2: Android Programming Framework

II. LITERATURE REVIEW

Creating games using already made engines has become very common practice. It is very difficult to create a technologically up to date and interesting game. It requires a large budget and a lot of time to catch up with the development of constantly new technologies. The solution to this problem is using a non commercial game engine license. Creating a game based on an existing engine save us the trouble of implementing the basic functionality.

We can then focus on expanding such an engine with additional modules and capabilities to create a new and technologically innovative implementation.[1]

Android is one of the most popular platform in mobile phones today. Due to its flexibility and performance it has became the heart of every smart device around. Lots of gamers have evolved due to this and the way of providing better and real time gaming experience has always been a challenge to gaming industry.

While developing a Game in Android there are various aspects of hardware and software are explored giving an better practice of utilizing android tools such as,

a. Collision Detection
b. Game Engine
c. Touch Manipulation
d. Screen Resolution

III. ANALYSIS OF PROBLEMS ANDROID VS. SYMBIAN VS. WINDOWS MOBILE

Comparison is based on main criteria as follows:

A. Portability:

Portability is a very important assessment criterion. Symbian OS has many references in this area and having standardized architecture and the openness to software. But the fact that Symbian mostly runs on Nokia cell phones and that it is not Java based lets it fall behind Android. Unfortunately Windows Mobile also has several applications that are specific to certain hardware platforms and therefore are not portable. The Android Mobile platform is a Linux & Java based which allow us to use it on many different platforms unlike Symbian & Win Mobile. As a result Android gets one point, Symbian OS gets half a point and Windows Mobile zero points.

Total so far: Symbian OS = 0.5 Windows Mobile = 0 Android = 1

B. *Reliability:*

Reliability is very much dependent on user experience. An operating system can be tested extensively, but without having experience of several years in "the real world" it is very hard to give a good estimate. Because of many years of user experience and the amount cell phones working with each of the systems it is possible to say that both, Symbian OS and Windows Mobile, are reliable enough for all kinds of users and applications which are available at the moment. It doesn't mean that both systems run perfectly well but problems with the systems will not result in major difficulties. The Linux kernel used by Android has existed for long period which has proven that it is stable and fail-proof. Therefore it is useful for mobile applications. Because Symbian OS and Windows Mobile control the biggest part of the market and Android is Linux based so we will give every operating system one point.

Total so far: Symbian OS = 1.5 Windows Mobile = 1 Android = 2

C. Connectivity:

The mobility of a cell phone generally makes a wireless connection preferable. Symbian OS & Windows Mobile features GSM telephony, Bluetooth, Infrared and WI-FI. & their APIs enables a development that targets all of these features and categories. Android also features GSM telephony, Bluetooth, EDGE and WI-FI [7]. All developers have the same access to the framework APIs used by the core applications. All of them support the common and mainly used connectivity standards. Therefore we will give each operating system one point.

Total so far: Symbian OS = 2.5 Windows Mobile = 2 Android = 3

D. Open Platform:

An "open mobile platform" is a software stack, including an operating system, middleware and key applications, which can be used on every mobile device. It allows users to develop additional software and change or replace functionality without limitations. The most common standards for communication and connectivity are used. [5] All these functionalities have to be free of charge. The only operating system which really fits to these criteria is the Android mobile platform. Which is based on a free available operating system? Another fact is that publishing your own developed applications is free which not the case for Symbian OS and Windows Mobile. This is the reason why Android gets one point and the other operating systems half a point.

Total so far: Symbian OS = 3 Windows Mobile = 2.5 Android = 4

E. Kernel Size:

An often used assessment factor for comparing the kernel size is the "Memory footprint" which is the amount of memory used by the operating system. For a significant classification we need to find the operating system with the

lowest "Memory Footprint" which in turn maximizes the performance of the operating system. Symbian OS require 200 Kb. The Windows Mobile platform requires 3 00Kb for a typical installation. The Android OS which is using Linux kernel will need about 250 kb of memory. All the data above apply to an installation with the basic and minimal functionalities. As a result Symbian OS needs less memory than Android which needs less memory than Windows Mobile. So Symbian gets one point, Android gets half a point and Windows Mobile zero points.

Total so far: Symbian OS = 4 Windows Mobile = 2.5 Android = 4.5

F. Standards:

Standards in general make the platform more open and attractive for developers. If standards exist it is easier for everyone and especially for developers, to get to know the new system. Every operating system uses the most common standards concerning networking, e-mails, messaging and communication, but only Android is based on the standardized programming language Java. This is also the programming language generally used to develop applications. The advantage of Java is that its programs can run on any platform without having to be rewrite. As a result Android gets one point, Symbian OS and Windows Mobile each half a point.

Total so far: Symbian OS = 4.5 Windows Mobile = 3 Android = 5.5

G. Special Features:

This section deals with features or applications which are designed to make the system unique. The Android mobile platform has significant advantages in this case. The new integrated browser based on the open source WebKit engine, the virtual machine Dalvik optimized for mobile devices, is a feature which enables every application runs in its own process. Windows Mobile has, due to its outstanding position in the computer market, the advantage that the synchronization between the PC and the cell phone is very easy. Symbian OS however has no special features which must be mentioned. Android gets one point, Windows Mobil half point and Symbian OS zero points.

Total so far: Symbian OS = 4.5 Windows Mobile = 3.5 Android = 6.5

From below table we can say that winner is Android.

Table I: Comparison of Android Vs. Symbian Vs. Win Mobile

Feature	Android	Symbian	Win Mobile
Portability	1	0.5	0
Reliability	1	1	1
Connectivity	1	1	1
Open system	1	0.5	0.5
Kernel size	0.5	1	0
Standards	1	0.5	0.5
Special features	1	0	0.5
Total	6.5	4.5	3.5

IV. GAME EVALUATION

The game should be innovative and contain some added value with regard to other multiplayer games playable on mobile devices. A fresh and easy to handle game play should be designed which has not been available so far. With respect to those requirements, multiple game possibilities have been analyzed. First thoughts about porting an already existing game, like Chess or Connect Four, into the Android environment were abandoned pretty fast. Implementing such a game type would add no value, since they already exist on mobile devices, even in multiplayer. The most popular exponents, in the genre of multiplayer games are Ego Shooters. From early Quake and Doom releases to recent offshoots of the Call of Duty or Battlefield series, there are millions of people competing against each other online. On a mobile device, such a game is not available yet and it would be a great challenge to implement this kind of game. But with the background of a bachelor thesis, the goal to implement an Ego Shooter would be too time consuming.

Geocoding is a very interesting topic and can be combined easily with the Android Framework. The use of Google maps is predestinated, which adds some value to the game. A Multiplayer quizzes game where multiple participants have to answer the same question in the least amount of time could be an extraordinary and fun gaming experience with lot of potential.

A. Design:

The basics for programming with the Android Framework have been evaluated. On this basis the GeoQuiz application has been designed and a prototype has been implemented.

GeoQuiz is a game, in which the geographical knowledge of the player is tested. The user answers questions and receives an amount of points for each correct answer. For example, a question could be "How many kilometers long is the Rhine river?" or "What is the second largest country in the world?". Three possible solutions should be presented to the user who has to choose one. If the correct solution has been selected, the time he needed to answer the question decides how many points he receives. The faster a player comes up with the correct solution, the more points he gets.

Some high-level requirements have been evaluated which should be implemented in the prototype. Those requirements are:

a. To provide a singleplayer mode.
b. To provide a multiplayer mode.
c. To provide an easy use.
d. To provide an innovative game play.

B. Component GeoQuiz

An overview of the application is shown in Figure 3. It shows its components and the position in the Android layer model. GeoQuiz resides in the application layer, on top of the Android Framework layer.

The Android Framework itself, neglecting the Operation system, is built on top of the Hardware layer of

CONFERENCE PAPER
"A National Level Conference on Recent Trends in Information Technology and Technical Symposium" On 09th March 2013
Organized by
Dept. of IT, Jawaharlal Darda Inst. Of Eng. & Tech., Yavatmal (MS), India

the device. GeoQuiz consists of three components: Data Access, P2P Network and Client. Data Access is responsible for accessing persistent data within the application. It uses the SQLite Helper classes provided by the Android Framework. The P2P Network component is responsible for sending and receiving messages from other peers and for storing and accessing data which is handled by the P2P network. This component uses an implementation of a DHT, tomp2p. Controlling the Data Access and P2P Network as well as managing user input and view elements is in the responsibility of the Client component. It uses the Android Activity class to present UI.

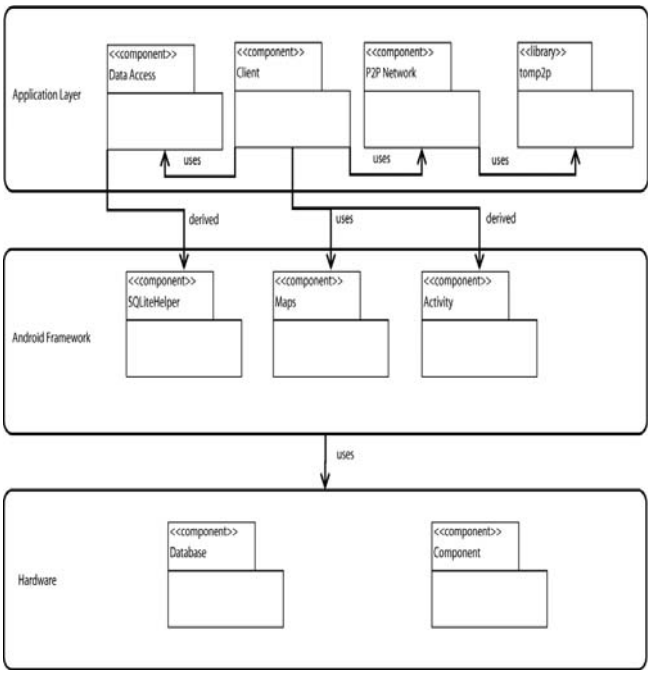

Figure 3: Component Diagram GeoQuiz

C. Data Access:

The Data Access component provides an encapsulated way to access persistent data of the application. Questions, possible answers and solutions are the biggest part of data the application relies on.

a. Table Question Type:

Question type is a helper table which defines all possible types of question in the game. Possible types are normal question, picture question and map question.

b. Table Answer:

Besides its primary key, each row in the *Answer* table holds a foreign key which points to an entry of the *Question* table.

c. Table Solution:

The *Solution* table maps the correct answer to its question. It holds references to both tables *Question* and *Answer*.

d. Table Question Location:

This table is used when a Map question is selected. In such a case (the *question type* is of type map), the user has to mark a certain spot on a map. The coordinates defining the center of the map to be displayed are stored in the *Question Location* table.

e. Table Answer Location:

This table is only used when a Map question has to be answered. It holds a reference to the *Answer* table and defines the correct coordinates for a certain map question.

f. Table Question Picture:

This table is used when a Picture question has to be answered. This table simply holds an ID of a picture which needs to be loaded along with the question. The picture itself is stored in an Android resource bundle and can be identified by this ID.

g. P2P Network:

A network component is important for enabling multiplayer game play. To establish a coordinated game, where multiple devices are connected, some sort of communication and information exchange needs to be implemented. For this task, an implementation of a DHT, namely the tom p2p library, has been chosen. The library was selected, due to the fact that it has already been ported and tested in the Android environment. [6] The Peer class of the tomp2p library provides functionality to bootstrap into a network, to store and receive data from the DHT and to send messages between the different peers. One peer needs to initialize the overlay network by listening to a specific port. Any node that wants to join this network, needs to call the bootstrap() method of the Peer class and pass the port and the IP-address as arguments. The state of neighboring peers is managed in a PeerMap. When a node bootstraps to a bootstrap node, it receives information about every peer which already participates in the overlay network and stores this information in the PeerMap. The bootstrap node also manages a PeerMap. Any change in the network is automatically reflected in the PeerMap.

V. SHEEP FRAMEWORK

A. Design goals:

From previous experiences on XNA, developers should not be involved in the programming too much time and cause less time on software architecture study, so the main goal of the Sheep framework is to allow the developers to save time in game programming. In a nutshell, the two overall goals for all major components in the Sheep framework are:

a. Simplify a common task in game development, so the developers can spend more time on structure and less time on technical issues.
b. Use known patterns to interact with client code, let them to perceive the course theory through using this

framework. According to these goals, classify the components values in the Sheep framework as:

a) ***Practical value*** means that components which simplify common tasks without requiring the use of any particular patterns. The primary goal of these components is to allow faster development, and save time.

b) ***Academic value*** means that components which require the use of certain patterns. The primary goal of these components is to illustrate the usefulness of a certain technique. Not all components achieve both goals.

B. Structure of Sheep:

According to design goals, Sheep structure in two ways in which the Sheep framework makes the android platform more feasible for game development to learn the software architecture [2]. From aspect of time saving goal for game programming, the Sheep structure is organized as packages as follows:

a. Sheep.audio provides components for loading and playback of sound.
b. Sheep.collision contains collision detection and spatial partitioning components.
c. Sheep.graphics contains components for loading images and fonts.
d. Sheep.gui holds the graphical user interface system.
e. Sheep.input contains the input devices, and the interfaces needed to subscribe to events.
f. Sheep.math contains some math classes which aren't directly related to collision detection.
g. Sheep.util is meant to contain miscellaneous components, but for now it only contains a singleton which keeps track of time between frames.

From aspect of encouraging or requiring the use of patterns, the components in the framework are:

e. Sprites, which uses the Model-View-Controller.
f. Game states, which uses the State pattern.
g. Collision detection, which uses the Observer pattern and the Template pattern.
h. Spatial partitioning, which uses the Visitor pattern.
i. Graphical user interface system, which uses the Observer pattern and Chain of command.
j. Other components without expected patterns.

C. Packages analysis:

a. Sheep.game package:

It provides components, which help organize the game model. Game State pattern is one of the main design concepts in this package. It keeps track of the high-level states of the game. Its main controller object contains methods for loading content, updating its internal state, drawing itself, and responding to input events. The practical value is that having a complete state system in place is beneficial because it allows relevant input events to be presented more clearly and quickly to the students. And its academic value is that State pattern is a well-known pattern, which allows an object partially changes its class at run-time. [2] Specific game behavior should be implemented via subclasses of the State class. Each state represents a different view of the game, when students use it in programming, they probably would understand it clearly.

b. Sheep.collision package:

It provides functionality for detecting interactions between objects in the game world, and generates collision events, which may be subscribed by observers. The practical value is that collision detection was used in most of the student projects, and getting the details of such collision systems to work right can be incredibly time consuming. When providing the students with a full collision detection system, they could use it directly to work efficiently. The Template pattern is in the Shape class, where the overall algorithm is fixed, but some subparts are modifiable by derived classes. The Observer pattern will be used for custom collision responses. As an example, perhaps a player should lose health when it is hit by another object, A "Lose Health Listener" could then be attached to listen for collision events occurring to the player object.

c. Sheep.gui package:

It provides a graphic user interface system and can be used to create complex windowed menus or simple buttons. The practical value is that a few buttons were present to allow the user to start and quit the game, and also provide functional kit for the extensibility. The academic value is that Observer pattern is used to listen for events. The Chain of command pattern issued to control how input events are passed through the widget class hierarchy.

VI. CONCLUSION

The goal of this thesis was to develop a game on such an open platform for mobile devices. Android has been chosen as the development platform since its API is open source and available for everybody. Based on our experiment of using XNA and current experiment of using Android in software architecture, we found game motivation and surround interesting peripherals are one of most attractive factor. And we found that Google android is a suitable tool for the educational use. Further, we have developed a game development platform called Sheep based on android.

VII. REFERENCE

[1]. Bian Wu; Alf IngeWang; Anders Hartvoll Ruud, Norwegian University of Science and Technology, Norway, "Extending Google Android's Application as an Educational Tool", IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning 2010

[2]. Google."AndroidSDK",http://developer.android.com/sdk/index.html

[3]. Bimal Gadhavi & Khushbu Shah, "Analysis of the Emerging Android Market", Project Report Presented to SanJosé State University May 2010.

[4]. B. Speckmann. "The Android mobile platform". PhD thesis, Eastern Michigan University, 2008.

[5]. D. Liben-Nowell, H. Balakrishnan, and D. Karger. "Analysis of the evolution of peer to-peer systems". In Proceedings of the twenty-first annual symposium on Principles of distributed computing, pages 233–242. ACM New York, NY, USA, 2002.

[6]. Ben Morisson. "The Symbian OS Architecture Sourcebook: Design and Evolution of a Mobile Phone OS", John Wiley.