ANDROID NUMBER GAME

By

MADHUMITHA LOGANATHAN

A REPORT

Submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department Of Computing And Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2012

Approved by:

Major Professor
Dr.Daniel Andresen

Abstract

Mobile application development is one of the recent trends in computing Industry. Among several existing platforms for mobile, Android is one of the largest platforms in the world that run in several smart phones and tablets from various manufacturers like Google, Motorola, Samsung, HTC etc.

Android number game is a simple game application in android targeting the school children to train their mathematical skills and help them to think. The application presents a graphical user interface with several colored bubble balls moving in random directions each with a number on it. The numbers are generated randomly within a specified range. The application allows the user to burst a moving bubble by touching it. The user's goal is to burst all the bubbles in the ascending order of numbers on them.

The application contains multiple levels and the number of bubbles increase as the levels increase. Also, the complexity of the set of numbers on the bubbles increases with increasing levels. Three lives are given to clear all the levels of the game and scores are computed for every correct shot on the ball. Bonus points are also added for clearing every level. The application maintains the top 10 players with their scores.

Table of Contents

List of Figures v
List of Tables vi
Acknowledgements vii
Chapter 1 - Introduction 1
Chapter 2 - Motivation 1
Chapter 3 - Requirement Analysis 1
3.1. Requirements Gathering 1
3.2. Requirements Specification 2
3.2.1. Software Requirements 2
3.2.2. Hardware Requirements 2
Chapter 4 - Systems Architecture & Design 3
4.1. System Architecture 3
4.2. Systems Design 4
4.2.1. Use case Diagram 4
4.2.2. Class Diagram 5
4.2.3. Sequence Diagram 6
Chapter 5 - Graphical User Interface 8
Chapter 6 - Android Components 9
6.1. Android Manifest.xml 9
6.2. Activities 10
6.3. Services 10
6.4. Intent 11
6.5. Android's SQLite 11
Chapter 7 - Implementation 12
7.1. StartScreen 12
7.2. Middle activity 13
7.3. Main activity 16
7.3.1 GameView, GameLevel and GameLoopThread 18

7.3. DatabaseHelper and Database Adapter 21
Chapter 8 - Testing and Logging 21
8.1. Logging 21
8.2. Unit Testing 21
8.3. Interface Testing 23
8.4. Compatibility Testing 24
Chapter 9 - Conclusion 27
Chapter 10 - Possible Extensions 27
Chapter 11 - References: 28

List of Figures

Figure 4.1 – Use case diagram 5
Figure 4.2 – Class Diagram 6
Figure 4.3 - Sequence Diagram [Play game] 7
Figure 4.4 - Sequence Diagram [List Scores] 7
Figure 4.5 - Sequence Diagram [Instructions] 8
Figure 7.1 - StartScreen 13
Figure 7.2 - Middle Screen 14
Figure 7.3- Instructions Screen 15
Figure 7.4 - Scores 16
Figure 7.5- Main Screen 17
Figure 7.6 - Main screen with Menu 17
Figure 7.7 - Incorrect input toast message 19
Figure 7.8 - Level completion toast message 20
Figure 7.9 - Save scores 20
Figure 8.1 - Samsung tablet vertical orientation 25
Figure 8.2 - Samsung tablet horizontal orientation 25
Figure 8.3 - Google Nexus Horizontal orientation 26
Figure 8.4 - Google Nexus vertical orientation 26

List of Tables

Table 8.1- Unit test results 23
Table 8.2 - Interface testing results 24

Acknowledgements

This acknowledgement transcends the reality of formality when I would express deep gratitude and respect to all those people behind the screen who guided and inspired me for the completion of my project work.

I would like to express my deep gratitude and wholehearted thanks to Dr. Daniel Andresen for his valuable guidance, precious suggestions and encouragement in completing the project successfully.

I wish to convey my sincere thanks to my committee members Dr. Gurdip Singh and Dr. Torben Amtoft for their support throughout the project.

I am also grateful to all the administrative staff and the technical staff of Computing and Information Sciences Department who have been helpful throughout my Masters. Lastly, I would thank all my friends who tested my applications in their mobile devices and gave their suggestions and valuable comments.

Chapter 1 - Introduction

The project is to develop an android number game application targeting children as the audience. Android is one of the leading and fastest growing mobile platforms today. Android is an open source software and the Android SDK is available to developers for free. The minimal setup time and the number of tools available for android SDK to ease the development process allows the developers to concentrate in the design and implementation details of the application in the available timeframe. The main objective of this project is to experience developing an android mobile application which is one of the booming trends in computing industry. The application helps to improve the concentration of the player with little mathematical knowledge.

Chapter 2 - Motivation

The main motivation of this project is to explore the concepts of mobile application development in android. Game applications are the most liked and downloaded applications from android market. A game application that is simple and easy to be used in a mobile handset can become a hit with millions of mobile users. The motive of this application is to learn and experience developing a simple game application in android targeting a small set of users. The knowledge obtained in this process can be applied later in the career to develop any similar kind of application focusing large group of users.

Chapter 3 - Requirement Analysis

3.1. Requirements Gathering

The project needed several requirements to be gathered before proceeding to design the game application. One among them includes gathering information about the suitable mathematical order that would be easy for a normal user and easy to understand the goal of the game. To identify this I searched the existing android applications in the market and gathered information about what kind of application has been done so far and what not has been attempted.

It was also noted that more than 60% of android device users have android operating system of version 2.1 or later. The functionality requirements of the application is simple and straight forward where as the technical details involved some research on the available android technologies.

GUI is critical for any game application and Android has multiple facilities for drawing GUI for games. Few study materials in android to code with the concepts like surface view and canvas were also gathered so as to smooth the application development process. These gathered information helped to provide a clear direction of deciding the software requirements and hardware requirements.

To check the feasibility of using these technologies in android for developing the proposed game, I practiced developing few simple android applications using these concepts. Some of them include an application to draw and move a bubble, an application to do a read and write operation with android's built-in SQlite databases, an application using 'Services' concept in android and so on. The gathered materials and the exercise problems I attempted prepared me to start designing the actual 'Android Number Game' application.

3.2. Requirements Specification

3.2.1. Software Requirements

Below are the software requirements for the project:-
Operating System: Android 2.2 or higher versions
Language: Android SDK, Java
Database: SQLite
Technologies used: Java, SQLite, Canvas, Android,
Tools: Eclipse IDE, Astah Community UML diagram tool
Debugger: Android Device Bridge, Android Emulator.

3.2.2. Hardware Requirements

Device: Android Emulator / Android tablet / Android Handset
Minimum Required Space: 6MB

Chapter 4 - Systems Architecture & Design

4.1. System Architecture

Based on the requirements analysis, the application's components, modules, interfaces and interactions of the system modules with other modules have been designed. This chapter describes the System architecture diagram, Use case diagram, Class diagram and sequence diagram of this application.

The application is a single-player real time android game that follows a different architecture than the conventional frameworks followed in web applications. The Android OS runs on the phone and the application runs on top of the OS. The user input is a touch event captured by the application which simulates the game logic module. The audio and graphics module includes the creation of the 2D number bubbles and production of appropriate sounds.

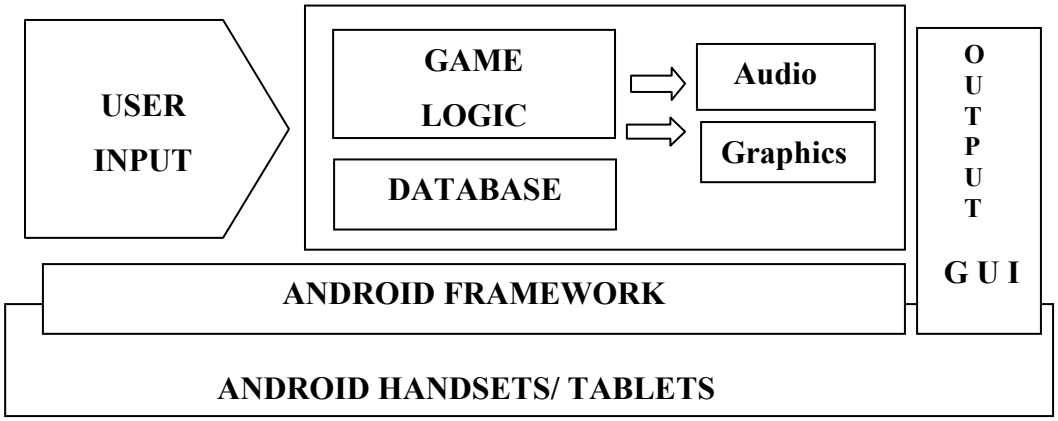

Figure 4.1: Systems Architecture Diagram

User Input: In our application, the user input is the touch event generated by touching the screen. The game engine monitors the *onTouch* event and at every touch, the corresponding coordinates are captured. If the coordinates are within the defined control areas of the device screen, the game engine would pass the control to the Game Logic module.

Game Logic: The game logic module is where the state of the game is decided based on the user inputs. The game logic includes checking any bubble collision to trigger the appropriate actions.

Audio: The audio module produces sounds based on the decisions of game logic. Different sounds are played to distinguish correct user input from an incorrect one.

Graphics: This module is responsible for rendering the game state on the screen. Android has several facilities for graphics rendering like canvas, OpenGL, etc. Our application uses 2D canvas rendering which is refreshed every 100ms and also updated based on the user's input.

Database: This module is used in saving the scores of the players in a table. SQLite database is an open source database that is embedded into Android. The game logic module interacts with this module for data persistence into the tables.

Output GUI: This output is the resulting sound and view rendering based on the game logic.

4.2. Systems Design

The high-level UML design diagrams are designed using the open source *astah community software*. Several entities were identified and the relation between these entities is described in these diagrams. The various diagrams determined for this game application include:

- Use Case Diagram
- Class Diagram
- Sequence Diagram

These diagrams are discussed in detail in the following sections.

4.2.1. Use case Diagram

A Use Case diagram is used to represent the actions by the user in a system. It has roles and actions. Each user/role has different privileges and each perform different actions.

For the Android Number Game, there is only one user 'the Player' and the player can do the below actions:

1. Enter the Game
2. View Instructions
3. View Scores
4. Mute/UnMute Background Music
5. Save Scores
6. Check Menu options
7. Touch Bubbles
8. Go to Home screen

The actions the player can do in this application are shown in the below use case diagram.

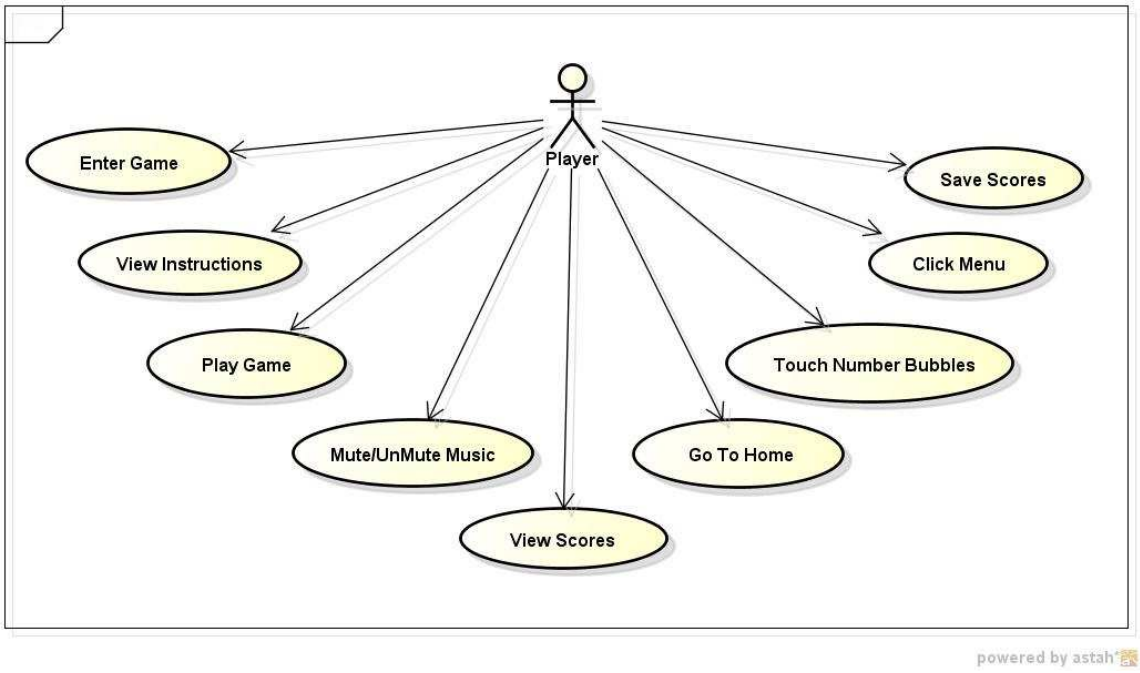

Figure 4.1 – Use case diagram

4.2.2. Class Diagram

A class diagram shows the basic types being built in the system. It forms a prototype for the application being developed and encompasses the classes, fields, methods and the relationship between these classes. The main structure of this game application can be represented by the following class diagram.

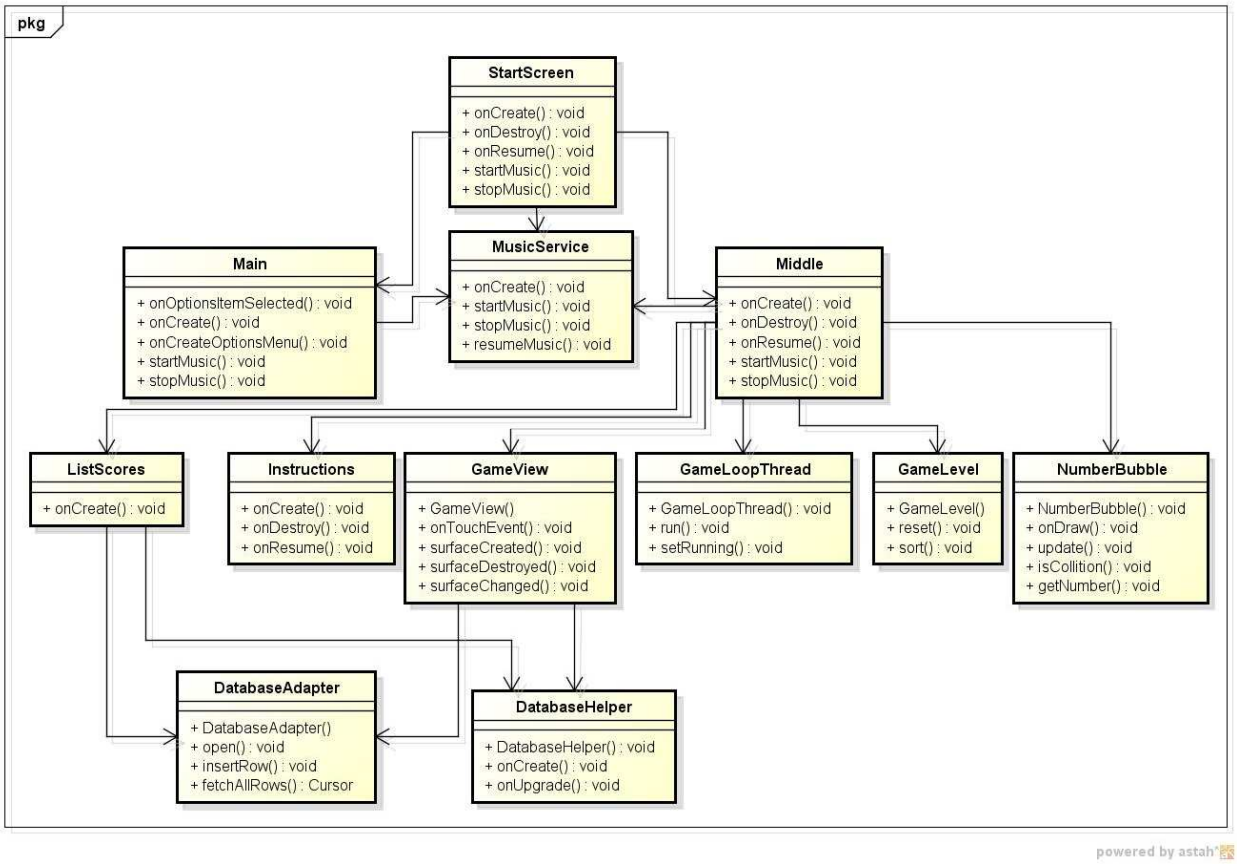

Figure 4.2 – Class Diagram

4.2.3. Sequence Diagram

A sequence diagram shows the interaction between the various classes and processes and the interaction order necessary to perform the functionality of the scenario. It showcases the classes involved in an interaction and the function calls and sequence of messages exchanged in that interaction. It forms a prototype to represent the behavior of various modules of an application. The interaction between the various classes for three major features of this game application is represented by the following sequence diagrams. The first sequence diagram explains the ‘game play’ sequence which is followed by the sequence diagrams for ‘list scores’ and ‘view instructions’ functionality.

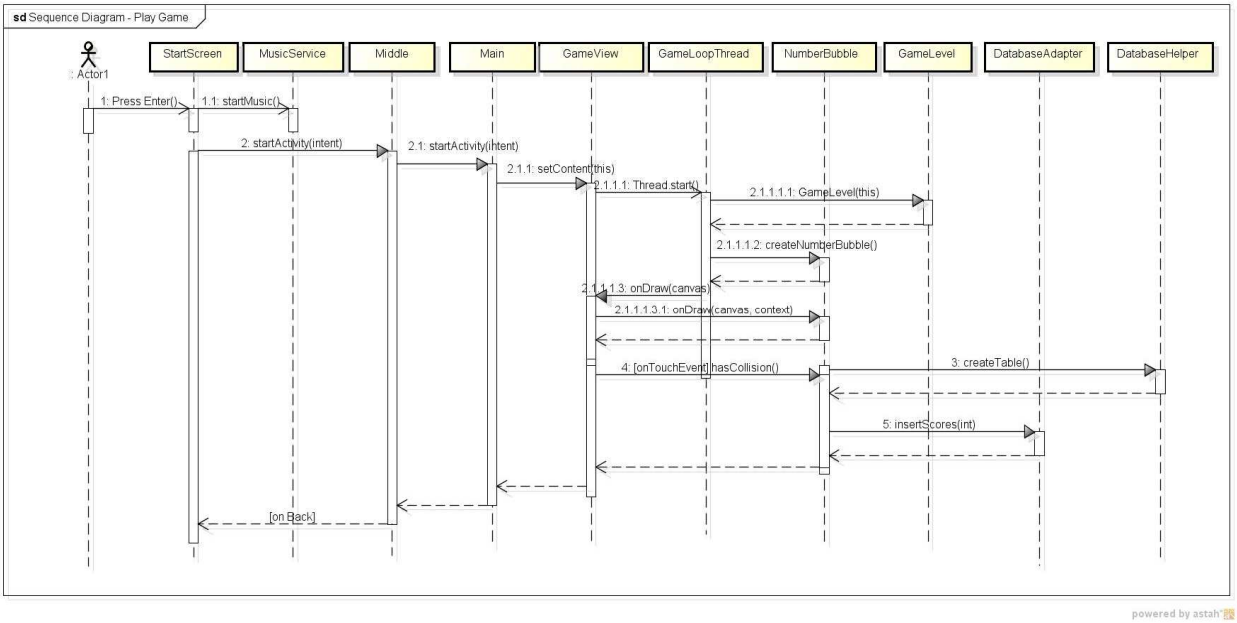

Figure 4.3 - Sequence Diagram [Play game]

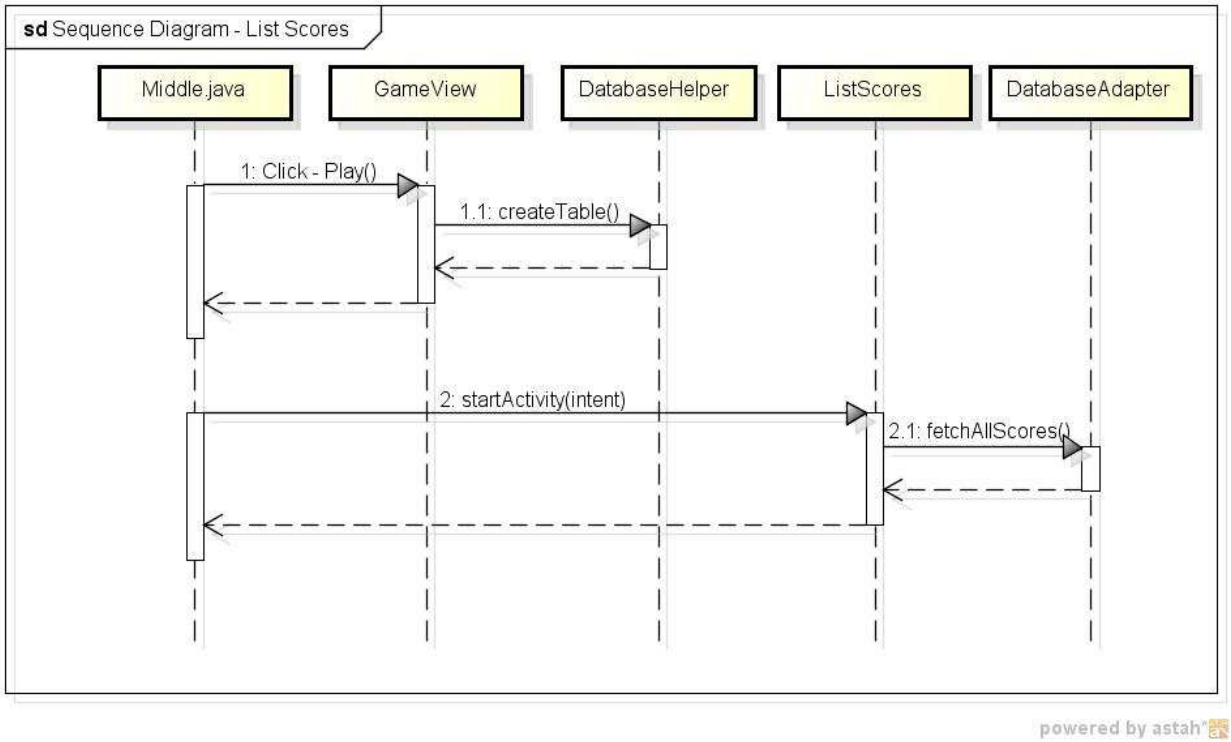

Figure 4.4 - Sequence Diagram [List Scores]

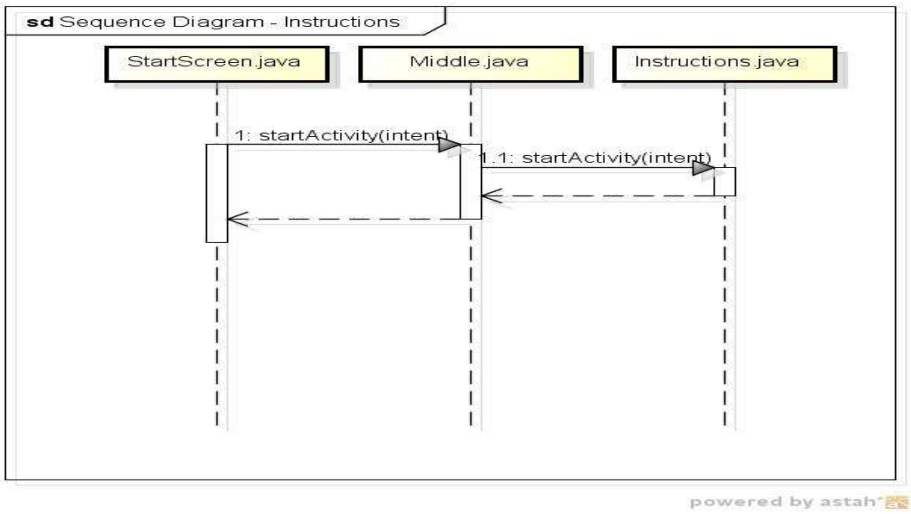

Figure 4.5 - Sequence Diagram [Instructions]

Chapter 5 - Graphical User Interface

The user interface is made simple and as intuitive as possible. “Keep it Simple” was the motive behind out UI design. The number bubbles are made big enough to aid easy navigation on smart phones and hence various sized drawable images are included to support different device configurations. ‘Krug’s law of Usability’ says, “Any application must be simple and easily understandable”. Consistent layout with easy navigation and simple instructions ensures understandability.

Attractive User Interface is very essential for any game application.

1. Decorative fonts are used in the start screen and all over the application.
2. Bright and attractive colors and contrasting color combinations are used in background and also for the bubbles.
3. Cartoon background images as the application is designed targeting school children.
4. Instead of having 2D circle objects embedded into Canvas renderer, translucent gradient images that gives a 3D illusion is used for number bubbles. Most of the facebook bubble games use gradient bubbles that give a 3D appearance rather than a plain 2D circle.
5. Toast messages are used to display when there is a change of levels or when there is loss of life. These messages are asynchronous messages and are displayed only for the specified time. They serve as a better option than having dialogs when it is enough to display the information and do not require any user input.

Chapter 6 - Android Components

Though programming in android SDK is using java language, android has its own set of concepts that make easier to design and code for application development. This chapter briefly discusses various concepts in android that this game application has used. (Developers Guide, 2011)

6.1. Android Manifest.xml

An application in android must have an AndroidManifest.xml file in its development root directory. This xml file presents essential information about the application to the Android system and the information the android system should have before it can start running the application's code.

AndroidManifest.xml file contains all the other android components that our application has used within the application. The components include activities, services, broadcast receivers and Content providers. The AndroidManifest.xml file also shows which application components communicate with each other and shows which' Intent' and 'Intent filter' are tied to which application component within our application.

More importantly android manifest file specifies the list of permissions that our application would need to be installed in an android device. The permission model in android makes sure that an application that has not been granted to access few resources or services within the device indeed is prevented from accessing them.

The Androidmanifest.xml file for this game application looks like below:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.edu" android:versionCode="1" android:versionName="1.0">
 <uses-sdk android:minSdkVersion="8" />
 <application android:icon="@drawable/icon" android:label="@string/app_name">
      <activity  android:name=".StartScreen"  android:label="@string/app_name"
android:configChanges="orientation"android:launchMode="singleTop"
android:theme="@android:style/Theme.Light.NoTitleBar.Fullscreen">
       <intent-filter>
            <action android:name="android.intent.action.MAIN" />
             <category android:name="android.intent.category.LAUNCHER" />
```

```
        </intent-filter>
      </activity>
      <activity      android:name=".Main"      android:label="@string/app_name"
android:configChanges="orientation">
      </activity>
      <activity     android:name=".Middle"     android:label="@string/app_name"
android:configChanges="orientation">
      </activity>
      <activity  android:name=".Instructions"  android:label="@string/app_name"
android:configChanges="orientation">
      </activity>
      <activity   android:name=".ListScores"   android:label="@string/app_name"
android:configChanges="orientation">
      </activity>
      <activity    android:name=".NewHighScore"    android:label="Game    Over!"
android:configChanges="orientation"android:theme="@android:style/Theme.Dialog
">
      </activity>
      <activity  android:name=".LostLife"  android:label="You  lost  a  life!"
android:configChanges="orientation"android:theme="@android:style/Theme.Dialog
">    </activity>
      <service android:name=".MusicService" android:enabled="true"></service>
 </application>
</manifest>
```

6.2. Activities

An Activity is an application component in android providing a screen with which users can interact in order to do something. The activity with some of the resource files represents the user interface in android. (Samy, 2010) The android number game application contains several activities to show different screens within our application as follows:

- StartScreen, Middle, Main, ListScores, Instructions, MaxLife, LostLife

6.3. Services

A Service is an application component that can perform long-running operations in the background and does not provide a user interface. Any component within our application can start a service and it will continue to run in the background even if the user switches to another application. It does not affect the performance of the fore ground activities in the user interface as android handles the services as a separate thread by default and isolates it from the user interface. (Vogel, 2009)

This component is useful in our application to play the music in background while the user is playing game. The music is started as a service when the first screen is launched in our number game application. We will discuss later about other features of number game application to mute and unmute the music from within our application while playing game.

The only service in the number game application is:

- Music Service – for playing background music

6.4. Intent

Inter application communication and Intra application communication in android happen with Intent. Intent is an abstract description of an operation to be performed that can be used to launch an activity or service from the current application component. An Intent has two primary attributes namely *action* and *data*. Action represents the general action to be performed such as ACTION_VIEW, ACTION_EDIT etc. Data represents the information to act on when intent is raised.

There are two types of Intents namely Explicit Intents and Implicit intents. Explicit intents provide the exact component the intent has to act on and it provides the exact class to be run when the intent is raised. Implicit intents do not explicitly mention the component that can handle these intents. Any application component which has an intent filter matching the action, data category of the intent can handle the intent raised.

The Android Number Game application uses several explicit intents for different scenarios to launch activities and services.

6.5. Android's SQLite

Android has a built in database engine SQLite which helps applications to do database operations. Any database created by an android application is accessible by name to any class within the application but not outside the application. Extending the class SQLiteOpenHelper in android and overriding the method onCreate is the simplest way to create a new SQLite database in android and execute the queries. (Hipp, 2011)

Android number game creates the following database and tables to keep track of top 10 players and their scores:

Database: APPLICATION_DATA; Table: SCORE

Chapter 7 - Implementation

The implementation in android starts with creating an new android project in Eclipse which creates separate folders for source code, resource files like images, xml files representing user interface layouts, gen folder that transforms each individual element names in resource file to unique identifiers that would be referred by the android system and other folders.

This chapter discusses the details of classes implemented for this project, implementation details of user interface xml files and other intricacies like handling bubble collisions, changing the moving directions of bubbles when hitting the screen's boundaries and handling the orientation change of android devices.

7.1. StartScreen

This is the first screen shown to the player when the application is launched from the application tray. The first screen to show is identified by the android system by checking the android manifest file of the application. The Start screen is a user interface implemented as an 'Activity' in android number game application. An activity having an intent filter tag with action field *android.intent.action.MAIN* and category *android.intent.category.LAUNCHER* is the one that will be shown first when an application is launched from the application tray. In our application's androidmanifest.xml file, Startscreen activity is mentioned to be started first when this application is launched. Below is the piece of code from AndroidManifest.xml file of our application.

```
<activity      android:name=".StartScreen"      android:label="@string/app_name"
android:configChanges="orientation"android:launchMode="singleTop"
android:theme="@android:style/Theme.Light.NoTitleBar.Fullscreen">
      <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
</activity>
```

StartScreen activity performs the following three tasks when shown to the player:

- Starts playing music as a 'Service' in the background
- Background image is set

- Displays an ‘ENTER’ Button for the game

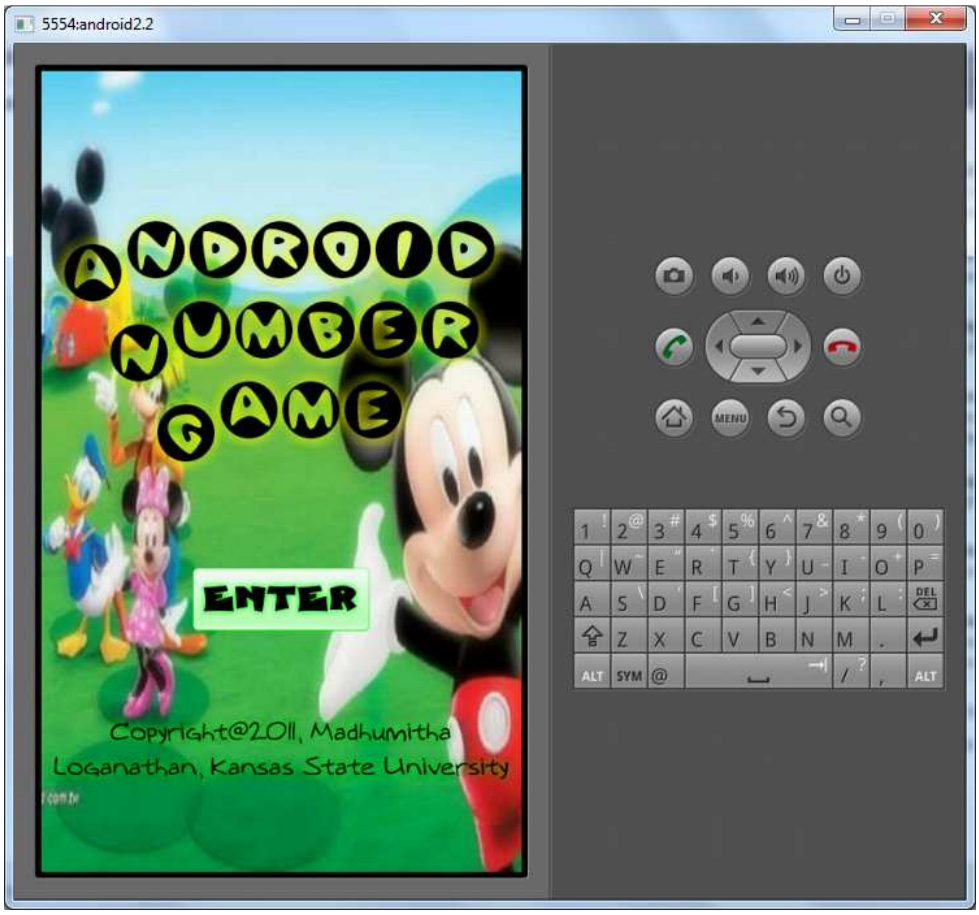

Figure 7.1 - StartScreen

StartScreen activity binds the ENTER button to the onclick listener. When the player clicks the ENTER button , an intent is raised to launch the next activity ‘Middle’ which we will discuss below.

7.2. Middle activity

The Middle Activity is the class that is launched by touching the ENTER button in the StartScreen activity by the player. This class sets the user interface to the user with the following buttons.

Figure 7.2 - Middle Screen

- Play, Instructions, Mute, Scores, Back

On touching the *Play* button the player is taken to MainScreen UI for playing the game. Before taking the player to the main screen, MiddleActivity is pushed to the back stack of the task automatically. This concept of back stack is useful when this activity is called again from any other activity in the application. The behavior of the activity in the back stack when launched back to the same view is decided by the task mode affinity parameter of the activity set in AndroidManifest.xml file.

On touching the *Instructions* button, the rules and goal of the game is shown to the user in a new activity. This is useful for the user who plays this game for the first time. On touching the *mute* button, the player can stop the music that is playing in the background. Some players

find it difficult to play the game when music is run in the background. Some others prefer to listen to music while playing. When the music is running in the background this button shows *Mute* and when the music is not running it shows *UnMute* which on touching would start the music again.

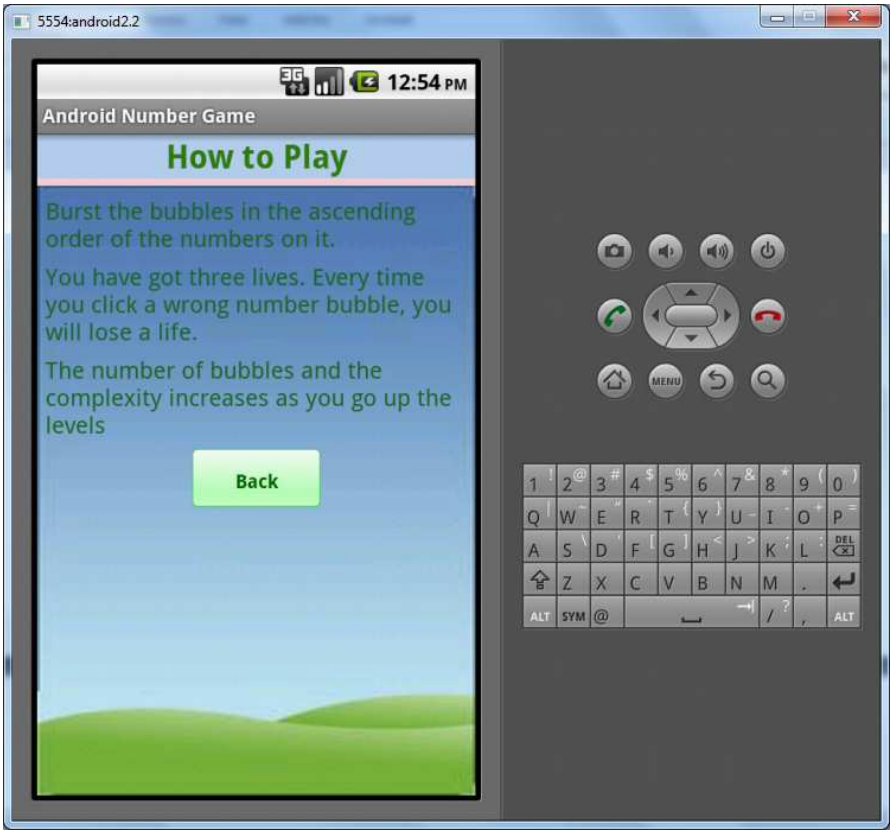

Figure 7.3- Instructions Screen

On touching the *Scores* button, the player can check the top 10 scores and the player names. When the user clicks this button, this class and the Database helper classes checks if *applicationdata.db* is present or not. If it is not present the database is created by invoking the methods in DatabaseAdapter class. Once the database is created a table named 'score' is created in the database. The details of the database table, their fields are discussed in the subsections later on. If the database and the table were already present with fewer or more entries, then the rows are all fetched and ordered by descending scores and top 10 scores and players are shown to the user by populating the screen in the form of a table. If the table has no scores, an appropriate message to indicate that there are no available scores is displayed.

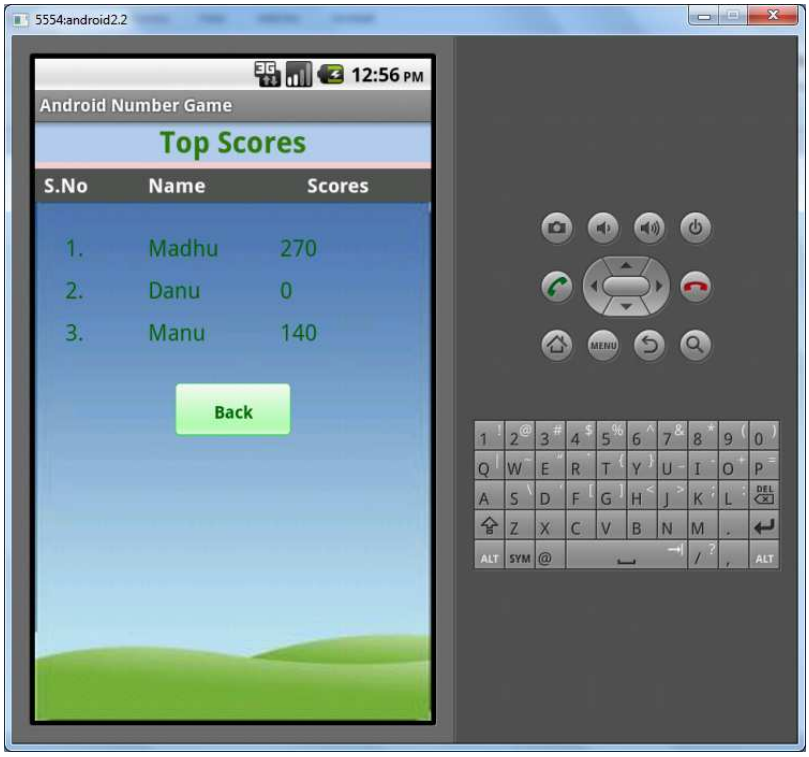

Figure 7.4 - Scores

On touching the *Back* button the player is taken back to the MiddleScreen Activity.

7.3. Main activity

The Main Activity is launched by the user by touching the *Play* button in MiddleActivity. Main activity sets its content by instantiating the *GameView* class. GameView class extends the Android's SurfaceView class and encompasses the main screen for the game play. GameView also includes for the implementation of the game's logic and it acts like the engine of the application. MainActivity works with GameView, GameLevel and GameLoopThread and takes corresponding action for user's input.

Figure 7.5- Main Screen

This activity also set two menu items ‘Home’ and ‘Mute/UnMute’. On touching the ‘Home’ menu current activity is destroyed and user is taken to MiddleActivity. On touching the ‘Mute’ menu the music in the background is stopped and the text is changed to ‘UnMute’. When ‘UnMute’ is touched music will be started again as a service in the background.

Figure 7.6 - Main screen with Menu

The application flow is based on two threads: main thread created with every Android application and a rendering thread. Rendering thread draws a frame as frequent as possible and Android Canvas renderer is used to draw the two dimensional number bubbles.

7.3.1 GameView, GameLevel and GameLoopThread

GameView is a surfaceview object and it closely works with *GameLoopThread* class and the *GameLevel* class in this Main activity. (A basic game architecture, 2010)

GameLevel: The *GameLevel* class decides the number of bubbles to be drawn at each level and the numbers that have to be embedded on top of each bubble. The complexity of the set of numbers to be embedded on bubbles increase as the levels increase.

GameView: The main responsibility of the GameView class is to create the bubbles and embed numbers on each bubble and prepare them to move on the screen. The resource folder contains several bubble images in different colors. GameView class picks the colored bubbles randomly and the numbers returned by GameLevel class are drawn on each bubble. Once these bubbles are generated it starts *GameLoopThread* class.

GameLoopThread: The *GameloopThread* invokes the onDraw method once every 100 milliseconds. This onDraw method is responsible for calculating the position of each bubble once every 100ms. This implementation of drawing the bubbles once in every 100ms at new position (either x co-ordinate or y co-ordinate or both co-ordinates are incremented by 1) creates an illusion to the user that the bubbles are moving. The calculation of x and y positions of each bubble in this two dimensional screen in the onDraw method takes care of changing the direction of each bubble when it hits the boundaries of the screen.

The GameView class handles the onTouchEvent of the moving bubbles in MainActivity. When the bubble with the least number on it in the current set of bubbles is touched by the player, the bubble bursts and score increases for every such correct touch. When a wrong bubble is touched, a life is reduced. If all three are used, the game ends and Player will be prompted an Activity that displays the user's current score and the highest score and prompts to collect the

name of the user. On collecting the user name the score and player name is saved in the score table of the database.

If the user has cleared a level i.e. all the bubbles in a level have been burst, the GameLoopThread is invoked again with a new level number. The GameLoopThread works with GameLevel and GameView classes to repeat the same process again.

The application is designed to have a maximum of ten levels with complexity increasing with each level. If all the levels are cleared, the *MaxLife* activity is invoked, the user details are received and the user score is saved in the scores database.

Toast messages are displayed if the user losses a life and after clearing every level. These are asynchronous messages and are displays for the specified time interval. (2 sec)

Figure 7.7 - Incorrect input toast message

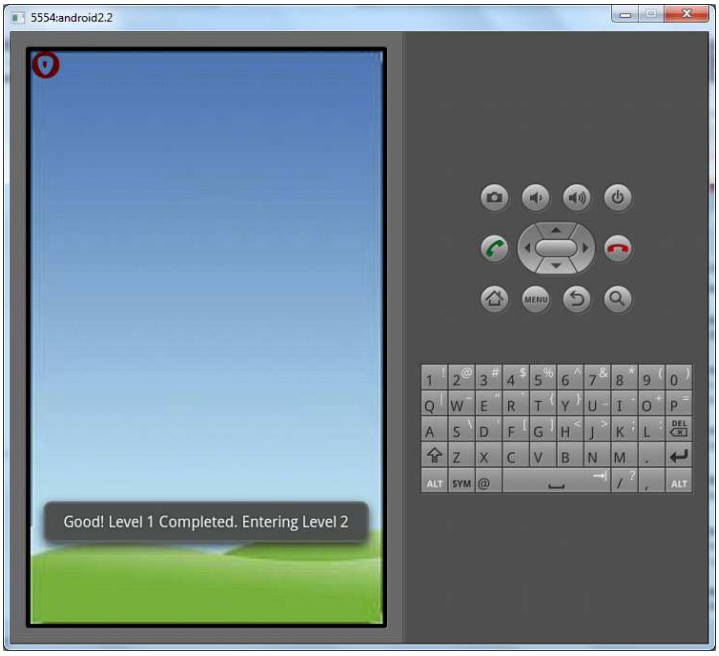

Figure 7.8 - Level completion toast message

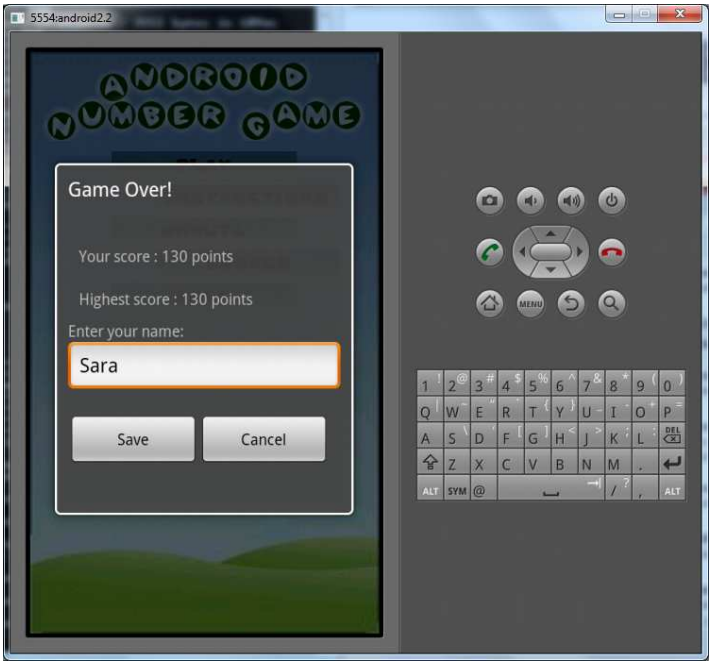

Figure 7.9 - Save scores

7.3. DatabaseHelper and Database Adapter

The DatabaseHelper is a class that creates the database 'APPLICATION_DATA when the player clicks the Scores button for the first time or when the user wishes to save the data after he finishes playing the game. The table called 'SCORE with two columns username and score is created. This table holds the list of players and their scores.

DatabaseAdapter is the class that provides API to execute queries from other classes to insert entries to the table and fetch entries from the table. (Hipp, 2011)

Chapter 8 - Testing and Logging

The Android SDK consists of a virtual mobile device emulator that helps to test the application without having a physical device as it provides all the functionalities of a typical physical mobile device.

8.1. Logging

Logging and debugging in android is done with the tool Android Device Bridge (ADB). ADB is one of the useful tools that come with the Android SDK. The system level logs and user defined logs are shown when 'adb logcat' command is executed while the application is running in emulator or in the device. Also, the Dalvik Debug Monitor Server [DDMS] which comes integrated with Android plugin for Eclipse helps to read the log messages for a specific emulator instance running our application.

ADB tool also provides the shell access to the android system of the device connected to system via USB cable. This tool provides access to SQLite database and the tables created and can be used to display the entries in the table. (Sugrue, 2010)

8.2. Unit Testing

In Unit testing each independent unit is tested separately, by isolating it from the remainder of the code to ensure parts of the code are working properly. Unit is the smallest testable part of the code, as in here the classes are treated as the base unit. Since the game application involves threading concept, it is not possible to leverage the jUnit tool for testing the individual components.

Unit Testing - Android Number Game				
S.No	**Test Modules**	**Test Case**	**Expected Result**	**Result**
1	GameLevel	Check the numbers on the bubbles	Random numbers are generated in the bubbles and no duplicates are allowed	Pass
2	GameLevel	Check the count of bubbles after each level	The number of bubbles should increase proportionately with increasing levels	Pass
3	GameLevel	Check the number range on the bubbles after each level	The number range on the bubbles should increase proportionately with increasing levels	Pass
4	GameView	Check the behavior of bubbles when they touches the screen edges	The direction of the bubble should reverse.	Pass
5	GameView	Bubble behavior on press of the next high numbered bubble	The bubble should burst and right sound is played.	Pass
6	GameView	Bubble behavior on press of an incorrect bubble	Sound for wrong input is played and the bubble is not burst	Pass
7	GameView	Scores updation on press of the correct bubble	The score has to increase at every correct input	Pass
8	GameView	Score updation and lives on press of an incorrect bubble	The score should remain same as before the input and number of lives is reduced	Pass
9	GameView	Bubble behavior on pressing overlapping bubbles	The lower valued bubble is taken as the clicked bubble	Pass
10	MusicService	Mute/UnMute behavior in Menu	If the music is being played currently, the display text is UnMute else the display text is Mute in the Menu. Pressing UnMute should start the music and pressing mute should stop the music	Pass
11	GameView	Toast message behavior for every level	After every level, toast messages should be displayed	Pass

12	Generic	Application behavior when there is an exception.	The music service should be terminated	Pass

Table 8.1- Unit test results

8.3. Interface Testing

Interface testing is done to check whether the individual modules are communicating properly one among each other as per the specifications. This testing is critical for any mobile application as they involve many interfaces and navigation between the various interfaces. The main idea is to check the consistency of the application, navigation and applications behavior for any or every possible set of user inputs.

Interface Testing - Android Number Game				
S.No	**Test Modules**	**Test Case**	**Expected Result**	**Result**
1	StartScreen	Launch the android number game application	The application should be launched and the music service should be started as a background process	Pass
2	StartScreen	Quit the application	The application must be terminated and also the music service is expected to stop	Pass
3	Main	Click the Enter button in the StartScreen	The application should take to the next middle screen that displays five buttons for various functionalities	Pass
4	Middle, MusicService	Press the Mute button in the Middle screen	The music service should be stopped and the label of the button should change to 'UnMute'	Pass
5	Middle, MusicService	Press the UnMute button in the Middle screen	The music service should be started and the label of the button should change to 'Mute'	Pass
6	Middle, Instructions	Press the Instructions button in the Middle screen	Instructions activity should be launched displaying the game instructions	Pass
7	Middle, ListScores	Press the Scores button in the Middle screen	ListScores activity should be launched displaying the game top 10 scores from the database	Pass

8	Middle, StartScreen	Press the Back button in the Middle screen	The home page must be launched	Pass
9	Middle, Main	Press the Play button in the Middle screen	Main activity should be launched and movement of balls observed	Pass
10	Main	Press the Menu button in the Middle screen	Menu options are displayed	Pass
11	Main, Middle	Press the Back button from Menu page	StartScreen activity is launched	Pass
12	Main, Database Adapter	Press 'Save' in the dialog shown when all lives are lost.	User name is received and table is updated with new scores	Pass
13	Main	Press 'Cancel' in the dialog shown when all lives are lost.	The dialog is quit and control goes back to the Main activity	Pass
14	ListScores, Middle	Press the 'Back' button from ListScores page	Middle activity is launched	Pass
15	Instructions, Middle	Press the 'Back' button from Instructions page	Middle activity is launched	Pass
16	Main, MaxLife	Check the application behavior if all the levels are cleared	MaxLife activity is launched	Pass
17	GUI Orientation Changes - All modules	Change the orientation from horizontal to vertical orientation.	The application should not be restarted and should continue from the same state it was before the changed orientation.	Pass
18	GUI Orientation Changes - All modules	Change the orientation from vertical to horizontal orientation.	The application should not be restarted and should continue from the same state it was before the changed orientation.	Pass

Table 8.2 - Interface testing results

8.4. Compatibility Testing

Variations in software versions, configurations, display resolutions, servers and Internet connect speeds can heavily impact the application behavior. Different specifications of devices can also make the applications to behave differently. People use different android devices and hence a good application must be 100% reliable and give best visualization effects irrespective of the device specifications. The application does not require internet hence the speed of internet is not relevant or necessary scenario in this case. To check the device compatibility, the application

is tested in the both Android tablet and smart phone. The application requires Android SDK 2.2 or higher versions.

1. Android Samsung Galaxy Tablet 10.1

   Android OS v 3.1, 10.1" wide screen HD WXGA (1280*800) TFT Display, Dual Core Processor (1GHz * 2), 16GB

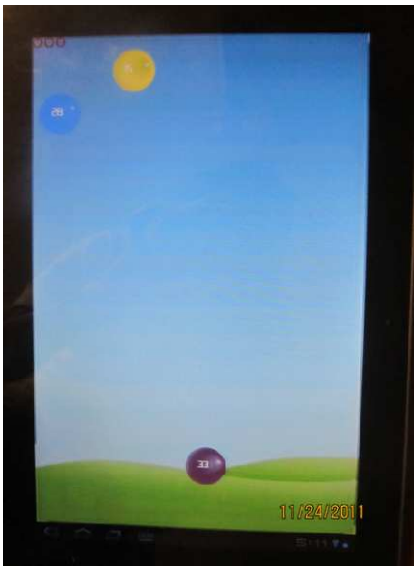

Figure 8.1 - Samsung tablet vertical orientation

Figure 8.2 - Samsung tablet horizontal orientation

2. Google Nexus

   Android 2.2, 1 GHz processor, (123.9 x 63 x 10.9 mm), 16 GB storage

Figure 8.3 - Google Nexus Horizontal orientation

Figure 8.4 - Google Nexus vertical orientation

3. Android Emulator Testing

   Android 2.2 “WVGA800” (800×480), LCD Density: “240”

I also requested my friends to test the application in their mobiles and got their feedback as part of user testing. Some of the feedback they gave was:

1. Good color combination and many of them liked the gradient bubbles.
2. Few persons did not like the music and few others liked them. The persons who did not like the music were glad that they had a Mute button.
3. Application was easy to understand.

Chapter 9 - Conclusion

The application has been designed, implemented and tested with real devices with users successfully. The project helped in understanding the challenges involved in developing a game application for android handsets and tablets, the ways to overcome them and in better understanding the intricacies of mobile application development. The project also helped in understanding the value of designing the components of overall application before implementing them. The project has also taught me game programming skills and refining the design and implementation logic of the software at every phase of the development life cycle to improve the overall performance of the application.

Chapter 10 - Possible Extensions

This android number game can be extended further in many possible ways and some of them are listed below:

- The game can be extended to display mathematical query on top of the screen and design it as a quiz application where the bubbles contain answers to the query and allowing the player to touch a bubble with the right answer.

- The application can be modified slightly to suit a different set of users. The application can be modified to have bubbles of different sizes with the same game logic and allowing the players to touch the bubbles in the order of decreasing or increasing sizes.

- The game can be improved to have different shapes for different levels and to have time settings for each level.

- With fewer changes the game can be extended to save the scores in a centralized server and update the high scores comparing with all the players of this game across the world.

Chapter 11 - References:

Developers Guide. (2011). Retrieved September 15, 2011, from Android Developers: http://developer.android.com/guide/index.html

Hipp, R. (2011). *sqlite*. Retrieved July 17, 2011, from documentation: http://www.sqlite.org/docs.html

Samy, M. (2010). *Android*. Retrieved August 5, 2011, from CodeProject: http://www.codeproject.com/KB/android/AndroidSQLite.aspx

Sugrue, J. (2010, October 20). *Debugging Android: Using DDMS To Look Under The Hood*. Retrieved August 4, 2011, from Javalobby: http://java.dzone.com/articles/debugging-android-using-ddms

Vogel, L. (2009). *Android Development Tutorial*. Retrieved August 2, 2011, from vogella: http://www.vogella.de/articles/Android/article.html

widgetguide. (2010). Retrieved October 27, 2011, from Droiddraw: http://www.droiddraw.org/widgetguide.html

A basic game architecture. (2010, July 26). Retrieved Nov 20, 2011, from Against the grain: http://obviam.net/index.php/2-1-a-little-about-game-architecture/