SINGING WINE GLASSES

by

AARTI PARUPUDI

B. Tech., Jawaharlal Nehru Technological University, 2010

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

2015

Approved by:

Major Professor
Dr. Daniel A. Andresen

Copyright

Abstract

One among the many inventions of Benjamin Franklin is the Glass Armonica, a musical instrument whose sound source was a series of resonating glass vessels. However, the Irish musician Richard Pockrich is typically credited as the first to play an instrument composed of glass vessels, called the Glass Harp in 1741, by rubbing his fingers around the rims.

In this project “Singing Wine Glasses”, the principle of Franklin’s glass armonica is demonstrated with a wine glass. One hand is used to hold the glass steady at the base. The rim of glass is gently pressed with a moistened finger of the other hand and drawn in a circle around. When the pressure and amount of moisture are just right, the slight friction between the finger and the rim of glass causes vibrations in the sides of the glass. At a particular frequency, called the resonant frequency, the sides of the glass will vibrate most easily. The resonant frequency of wine glasses is typically within the range of human hearing (20-20,000 Hz), so the resulting resonant vibration is heard as a tone. The glass starts to sing when the vibration gets the molecules moving at their natural frequency. The resonant frequency changes with the amount of water filled in the glass.

This android application deals with virtual glasses that serve the purpose of wine glasses filled with different amounts of water. Swiping on the glass edges would produce music, as per Franklin’s principle. The users would be free to select the number of glasses they want to play, and the amount of water-level in each glass. This application would also come with an enhanced feature of sustaining a particular note until the finger is released from the glass.

Table of Contents

List of Figures

Acknowledgements

I would like to express my sincere gratitude to my major professor Dr. Daniel A. Andresen for his continuous support, guidance and encouragement throughout my graduate research project. I also thank him for believing in my abilities and providing me this opportunity. I would like to thank my committee members Dr. Doina Caragea and Dr. Torben Amtoft for their valuable suggestions and input to my project.

I take this opportunity to acknowledge the academic staff and colleagues for giving me wonderful memories during my time at the Department of Computing and Information Sciences.

Finally I thank my family and friends for believing in my and standing by me throughout the course of my graduate study at Kansas State University.

CHAPTER 1 – Project Description

Singing Wine Glasses is an Android application that allows users to play music on a series of virtual glasses, each with a certain amount of water filled in it. The original idea came from the Glass Harp, which is the predecessor of Benjamin Franklin's Glass Armonica. Glass Harp brings music to life in a rather unusual and an enchanting way. Swiping our fingers expertly in a circular motion along the edge of the glasses produces different sounds rendering great music!

1.1 Overview

One among the many inventions of the famous statesman, scientist, democrat, printer and inventor – Benjamin Franklin – is the Glass Harmonica, a musical instrument whose sound source was a series of resonating glass vessels.[1] However, an Irish musician Richard Pockrich, in 1741, popularized the use of wet fingers and friction to induce music out of an array of glasses. The instrument was called the Glass Harp. Although the Glass Armonica was invented by Benjamin Franklin in 1761, Richard Pockrich is typically credited for playing the first ever instrument composed of glasses, the Glass Harp (Glass Armonica's ancestor), by rubbing his fingers around the rims.

Figure 1.1: Glass Harp (the ancestor of Glass Armonica) [2]

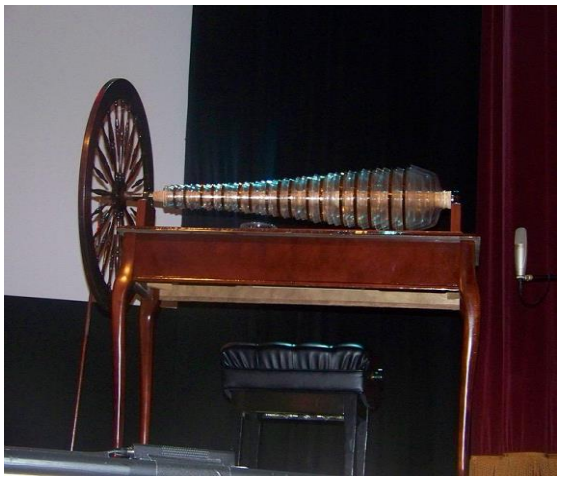

Figure 1.2: Benjamin Franklin's Glass Armonica [3]

1.2 Science

The principle of Glass Harp can be demonstrated using a wine glass. One hand is used to hold the glass steady at the base and the rim of the glass is pressed gently with moistened finger of the other hand and ran around the rim of the glass.

When the pressure and amount of moisture are just right, the slight friction between the finger and the rim of the glass causes vibrations in the sides of the glass. At a particular frequency, called the resonant frequency, the sides of the glass vibrate most easily. Since the resonant frequency of the glasses typically lie within the range of human hearing, the resulting resonant vibration is heard as a tone. The glass starts to sing when the vibration gets the molecules moving at their natural frequency. Each glass produces a different note that differs based on the size and shape of the glass or the amount of water-level in it. Sound travels much slowly through water than it does through glass which implies that an empty glass resonates at a much higher frequency than a glass with water in it. In simpler terms, the more water the glass contains, the lower the frequency - and thus the lower the note it will produce when rubbed. [1]

1.3 Application Objectives

Singing Wine Glasses is an android application that makes one "exp-ear-ience" the captivating vibrations we call music. The unique way of producing music with the common glassware found at home, lured me to develop an application that provides a virtual interface for all music lovers out there to produce enthralling tones with a simple swipe of their fingers.

1.3.1 Main features:

- The application deals with virtual glasses that serve the purpose of wine glasses filled with different amounts of water.
- Swiping on the glass edges would produce music, as per Franklin's principle.
- The users would be free to select the number of glasses they want to play, along with the amount of water-level in each glass.
- This application would also come with an enhanced feature of sustaining a particular note until the finger is released from the glass.

CHAPTER 2 – Background & Related Work

2.1 ANDROID

Android is a mobile operating system based on the Linux Kernel. It was originally developed by Android Inc. and was later acquired by Google on August 17, 2005 and released under open source licenses. [4] The open nature of Android lured a huge community of developers to use its open-source as a foundation for adding new and advanced features to the smart phones.

The user interface for Android is designed primarily for touchscreen mobile devices like smart phones and tablet computers. The OS takes inputs that correspond to real-world actions like swiping, tapping, pinching, reverse pinching, air gestures to manipulate on-screen objects. Android is a very convenient, economical and a personalized operating system for smart devices.

2.1.1 Architecture

Android Operating System is a bundle of layered software components with each layer providing different services to the layer just above it. [5]

1. **Linux Kernel:**

   At the bottom is the Linux Kernel that acts as an abstraction layer between hardware and other software layer. It provides the basic system functionality like process management, memory management, device management, etc. The Kernel handles networking and supports a vast array of device drivers that provides an interface for the users to interact with the application.

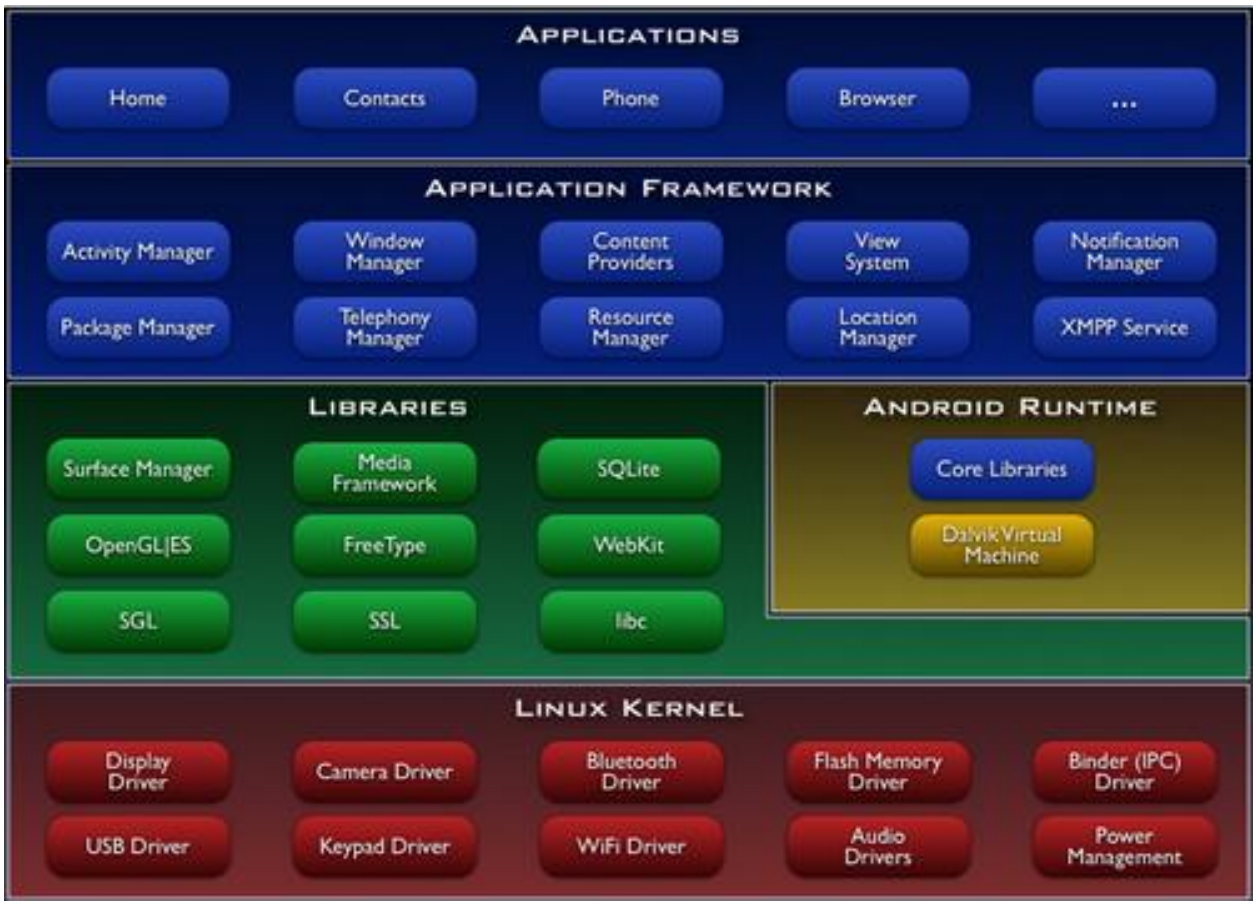

Figure 2.1: Android Architecture [5]

2. Libraries:

The layer on top of Linux Kernel has a set of native libraries that enables the device to handle different types of data.

The following are some of the important native libraries that are written in C/C++ language and are specific for a particular hardware:

i) **Surface Manager** balances window manager with off-screen buffering that is responsible for the transparency of windows.

ii) **Media Framework** embeds media in an application and allows recording and playback of different media formats.

iii) **SQLite** is the database engine used in android for data storage purposes.

iv) **WebKit** is the browser engine used to display HTML content.

v) **OpenGL** is used to render 2D or 3D graphics content to the screen.

3. **Android Runtime:**

The third section of the architecture is the Runtime that provides a key component called the *Dalvik Virtual Machine.*

The Dalvik VM is a kind of Java Virtual Machine specially designed for Android and optimized for low processing power and low memory environments. It makes use of Linux core features like memory management and Java's multithreading. It enables every Android application to run in its own process, with its own instance of the DVM. It provides higher efficiency in low resource environments and allows multiple instances of VM to be created simultaneously, thereby providing security and isolation. It implements efficient memory management and threading support.

The Android Runtime also provides a set of Core Java Libraries that enable the Android developers to write Android applications using standard Java programming language.

4. **Application Framework:**

This layer provides many higher-level services to applications in the form of Java classes that our applications directly interact with.

Important blocks of Application Framework include:

i) **Activity Manager** manages the activity life cycle of the applications.

ii) **Content Providers** manages the data sharing between applications.

iii) **Telephony Manager** manages and accesses all voice calls in an application.

iv) **Location Manager** is used for location management using GPS or cell tower.

v) **Resource Manager** manages the various types of resources used in an application.

5. **Applications:**

The Applications are the top layer of the Android Architecture where an application, written by the developer, is installed. Some standard applications that come pre-installed with every device include Contact manager, Dialer, Web Browser, SMS app, etc.

2.1.2 Application Components

App components are the most important building blocks of an Android application that help define the app's overall behavior. Each component is a different entry-point with a specific role through which the system can enter the app.[5] Following are the four different types of app components:

1. **Activities**

An *activity* represents visual interfaces for each task in the application. Every activity, despite being linked, is an independent entity. An activity is implemented as a subclass of *Activity.*

2. **Services**

A *service* is a component that runs in the background to perform long-running operations and work for remote processes. Services, in general, include recording the route over the duration of the journey. A service does not provide a user interface.

Another component can start the service and let it run or bind to it in order to interact with it. A service is implemented as a subclass of *Service.*

3. Content Providers

A *content provider* manages a sharing of app data with other applications, that can be stored in the file system, SQLite database, on the web, or any other accessible storage location. Content providers are also used to access data that is private to an application. A content provider is implemented as a subclass of *ContentProvider* and must implement a standard set of APIs that enable other apps to perform transactions.

4. Broadcast Receivers

A *broadcast receiver* is a component that responds to system-wide broadcast announcements. Despite of not having a user interface, broadcast receivers may create a status bar notification to alert the user when a broadcast event occurs. A broadcast receiver is implemented as a subclass of *BroadcastReceiver.*

2.2 ANDROID STUDIO

Android Studio is an Integrated Development Environment (IDE) for Android application development, based on the IntelliJ IDEA (the most INTELLIgent Java IDEA). [6]

Android Studio supports the following features:

- Live layout with WYSIWYG Editor that supports live coding and renders real-time applications
- Flexible Gradle-based build system support
- Android specific refactoring and quick fixes
- *Lint* tools to catch performance, usability, version compatibility and other problems
- ProGuard and app-signing capabilities
- Template-based wizards for the ease in building common designs and components for Android applications
- Rich layout editor that supports drag-and-drop UI components and an option to preview layouts on multiple screen configurations
- Supports developing Android Wear apps like smart-watch, Google glass, etc.
- Built-in support for Google Cloud Platform that enables integration with App Engine and Google Cloud Messaging
- Build variants and multiple *apk* file generation, and many more.

2.3 RELATED WORK:

The related work based on the idea of *rubbing wine glasses to make them sing* includes a few applications like:

2.3.1 Tiny Wine:

Tiny wine is a basic virtual version of Glass Harp for Android devices that has a very static interface including 5 glasses with a fixed note for each. It doesn't provide the flexibility to users for selecting the number of glasses they want to play music with. It neither provides the flexibility to select the number of glasses the users want to play music with, nor the option to maintain the water-level in each one. It also doesn't sustain the music as long as the user swipes around the rim of the glass. [11]

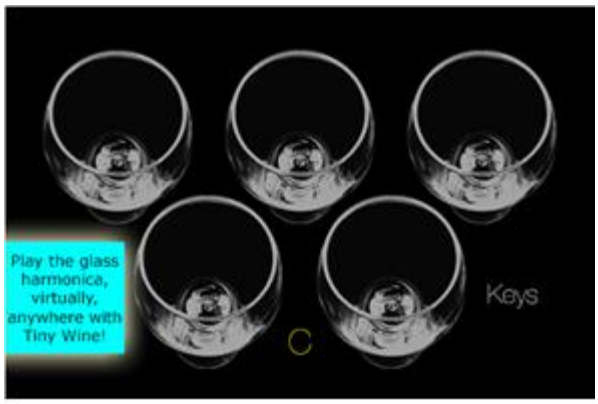

Figure 2.2: Tiny Wine App

2.3.2 Wine Glass Music:

It is an application for iPads, iPhones and devices with IOS. The major drawback of this app is that it is not supported by Android devices.[10] It also is kind of a static app that doesn't allow the user to change the number of glasses or the water-level in them.

Figure 2.3: Wine Glass Music App

2.3.3 Wine Glass Synth:

Wine Glass Synth is an IOS application that implements Glass Harp principle to a great extent.[10] It provides different flavors of music from classic glass armonica to violin and synthesizers. This application is also not dynamic, that is, it doesn't allow change in the number of glasses and the amount of water. The users are required to purchase the app (for $4) in order to use the application.

Figure 2.4: WGM Synth App

Singing Wine Glasses is an application developed for Android platform, which overcomes all the drawbacks that the above mentioned applications possess and would surely be one among the many applications that the music lovers would love to play music with! Also, it is a free application and provides the flexibility for the users to select the number of glasses and the amount of water in each glass. This, in turn would produce variations in the musical notes based on the water-level difference.

Also, since it is a mobile application, it is highly safe and portable. That is, it can be easily carried around during travelling, which is impossible with the actual crystal wine glasses that

are exceedingly fragile. The application is very user-friendly with a very easy setup. This alleviates the pain of setting up the entire apparatus.

CHAPTER 3 – System Requirements

3.1 APPLICATION REQUIREMENTS

Requirements Gathering is the most crucial part of a project. A developer needs to completely understand what a project does, to deliver it with success. Singing Wine Glasses Application started with the idea of Dr. Daniel Andresen, who also provided me with an initial overview of the app's requirements. I downloaded and experimented with various applications related to Glass Harp and other instruments, and tried to anticipate what all features an instrumental app generally has. Then, I tested the available applications in all possible ways and observed few drawbacks in those apps. I tried to beat those flaws and design my application to make it loved by all the users. And my thought process was further enhanced by Dr. Andresen.
The important requirements for the project are as follows:

- Virtual glasses serve the purpose of wine glasses filled with different amounts of water.
- Swiping on the glass edges produces music.
- Users would be free to select the number of glasses they want to play music with along with the amount of water-level in each glass.
- It would also have an enhanced feature of sustaining a particular note until the finger is released from the glass.

Application Development Life Cycle describes the process of planning, creating, testing and deploying a particular application. Requirements gathering is the initial and the most crucial part of the application development life cycle. The other phases include planning, designing, developing, integration and testing, implementing, deployment and maintenance.

3.2 APPLICATION SPECIFICATIONS

The following are required to successfully develop, implement and deploy an Android application:

3.2.1 Hardware Specifications

- An Android smart device with 2.2 or a higher version
- A processor with Pentium IV or higher
- 512 MB RAM
- 250 MB or higher Disk Space

3.2.2 Software Specifications

- Operating System: Windows XP, Vista or higher, Mac OS 10.6 or later, Ubuntu 10.4 (Linux)
- Platform: Android SDK with an API of 10 or higher
- Tools: Android Studio 1.1.0, Git distributed version control system
- Technologies: Java, Android

CHAPTER 4 – Project Implementation

4.1 SYSTEM DESIGN & ARCHITECTURE

System design is the process of defining the interfaces, components, modules, architecture and data for a system in order to satisfy the specified requirements of the users. It applies system theory to develop a product. It deals with the architectural, logical and physical designs of system development. And the visualization of the system's design is provided by UML (Unified Modeling Language) Diagrams and it goes without saying that a picture is worth a thousand words.

These designs help the developer to best determine the functionalities of the system and model the application that fulfills the requirements of the users. In Singing Wine Glasses Application, the GUI is designed in a very user-friendly style. These designs help users get an overall view of all the functionalities that the application provides and smoothly interact with the system.

4.2 SYSTEM DESIGN

Following are some of the UML diagrams that define my system design that helped me develop my Android application. [7]

4.2.1 Class Diagram

A class diagram is a structural diagram that represents the static view of an application and visualizes different aspects of a system along with constructing an executable code of the software application. It describes the attributes and operations of a class and the constraints imposed on the system. A class diagram is a collection of classes, interfaces, associations, collaborations and constraints.

The class diagram that I designed for my application – Singing Wine Glasses – is shown and explained below.

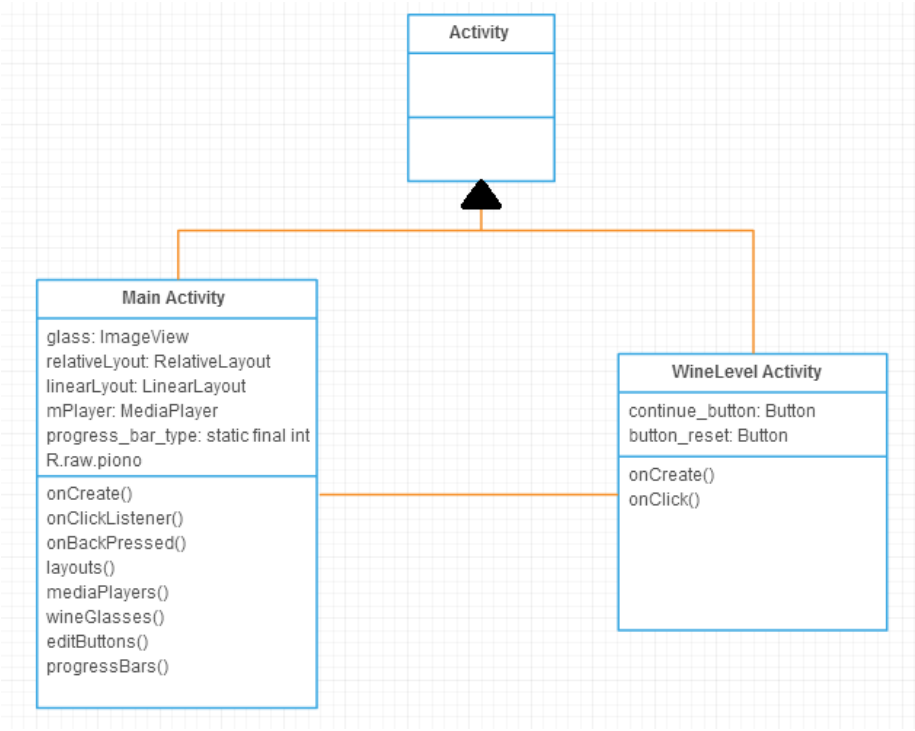

Figure 4.1: Class Diagram

Singing wine glasses has two main classes – MainActivity and WineLevelActivity that extend the *Activity* class, that takes care of creating a window to place UI for the application. Both the classes have a set of variables and methods embedded in them. MainActivity has ImageView, RelativeLayout, LinearLayout, MediaPlayer objects and methods like onCreate(), onClickListener(), onBackPressed(), etc. It has listeners for each event that decide the activity to be performed for each action the user does. This class deals with the main musical instrument screen where the music is played and the navigation bars to go to the glasses and water-level selector screen.

WineLevelActivity has Button objects and onCreate() and onClick() methods. This class has methods that deal with the glass and the water-level selector screens. It has listeners to the actions performed by users on trying to change the water-level in the glass.

4.2.2 Use Case Diagram

Use case diagrams capture the dynamic behavior of a system. A use case diagram uses actors, use cases and their relationships to model the system or a subsystem of an application. A single use case diagram captures a particular functionality of a system. It also identifies the internal and external factors influencing the system.

An *actor* in a use case diagram specifies a particular role played by a user or an entity that interacts with the system. A use case is a list of steps that defines the interactions between an actor and the system, to complete a function. The relationship represents the interaction between actor and the system.

The use case diagram for my application is shown below.

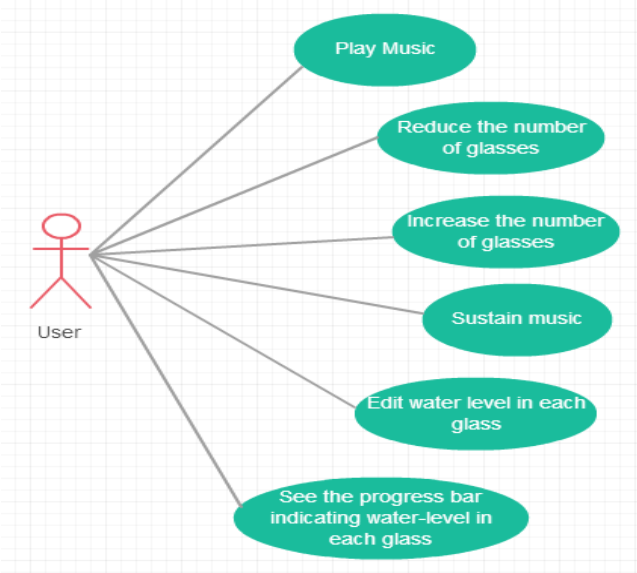

Figure 4.2: Use Case Diagram

The use case diagram of the Singing Wine Glasses Application has user as an actor and the use cases include:

- Play music
- Reduce the number of glasses
- Increase the number of glasses
- Sustain music
- Edit water level in each glass
- See the progress bar indicating water-level in each glass

4.2.3 Sequence diagram

Sequence diagram is a type of interaction diagram, which in turn is a part of dynamic behavior of the system. The interactive behavior is represented by *Sequence diagram* and *Collaboration diagram.* Sequence diagram focuses on time sequence of messages and

collaboration diagram deals with the structural organization of the objects that send and receive messages.

A sequence diagram shows the sequence of messages exchanged between different classes and objects as part of interactions needed to carry out a particular behavior. The parallel vertical lines, called *lifelines,* consist of different objects that live simultaneously, and messages are exchanged on the horizontal arrows, in the order in which they occur. This completely specifies simple runtime behaviors in a graphical manner.

The sequence diagram for Singing Wine Glasses app is shown and explained below.

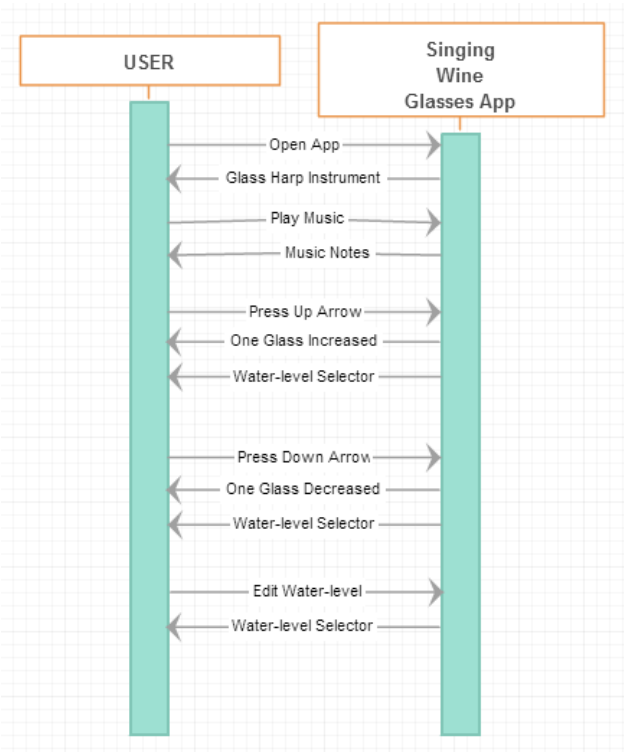

Figure 4.3: Sequence Diagram

The sequence diagram for Singing Wine Glasses is shown above. It includes a sequence of actions carried out from starting the app, carrying out different functionalities until closing the app.

4.3 IMPLEMENTATION

Singing Wine Glasses is an Android application that is a virtual representation of the musical Glass Harp, invented by Richard Pockrich in 1741. It is a great application for all the music lovers, who love experimenting with different instruments, without the pain of setting up any apparatus or buying a new instrument altogether. It is a pretty much mess-free application.

Music is actually produced in the application with a media player that has notes assigned to each level of water in the glass ranging from 10 - 100%. The lower the level, the more the frequency, and hence, higher the note assigned. Real piano notes were recorded and synthesized for the musical notes of this application. The notes almost produce the same tones as the real *Glass Harp.*

The different *modules* included in the Singing Wine Glasses that serve different functionalities are as follows:

- Instrument
- Glass Selector – Reducing the number of glasses
- Glass Selector – Increasing the number of glasses
- Water-level Selector
- Progress Bar
- Music sustainability

Instrument

This is the main module of the application that houses the musical instrument – Glass harp! It has virtual glasses that play the role of the actual wine glasses. The initial screen, with a fixed 3 glasses each with a water-level of 40%, 60% and 80% respectively, pops up as soon as

the user opens the application. The water-levels range from 10-100% with a demarcation of 10%. Each water-level is assigned a particular note, to represent the variation in the frequency with the amount of water in the wine glass. The lesser the amount of water in a glass, the higher the frequency and hence, higher the note it produces.

Glass Selector

Glass Selector is an important module that allows the user to select the number of glasses to be present on the screen to play music with. The user chooses the number of glasses based on either the size of his finger or the notes he want or the speed with which he can move his fingers or the expertise with which he can play the music. There should be at least one glass for playing and it can go up to 5 glasses. The app would throw an error if the user tries to reduce the number of glasses below 1 or increase it beyond 5.

From the initial welcome screen, there are two options using which, the user can either reduce or increase the number of glasses. If the user selects the *down* arrow, the last glass would disappear and the user would be prompted to the screen to select the amount of water in the remaining glasses. He can either change the water-levels or can continue to the instrument to play music. In this screen, again the user has an option of selecting the number of glasses.

When the user selects the *up* arrow, another glass fills the screen and again the user would be prompted to the screen to select the water-level. Here again, the user can change the water-level of any glass or can manipulate the number of glasses and continue to the instrument screen.

Water-level Selector

The application also provides the user with another option of modifying the water-level without actually having to change the number of glasses. When the users select the edit option, they would be prompted to the screen with water levels where they can select the amount of water for each glass individually.

Progress Bar

The application provides each glass with a progress bar in the main screen housing the instrument. The progress bar represents and gives a basic idea to the user of the water-level in each glass. This also helps the user to estimate the note for each glass because notes vary based on the frequency, which in turn, differs with the water-level of the glass.

Music sustainability

This allows the users to sustain the music as long as they continue swiping the edges of the glass. The music stops once the user lifts the finger off the screen.

There are listeners for every action performed by the user. The application captures the motion of the finger and starts the music. The media player is set on a loop to be played as long as the finger is in contact with the glass. Once the user lifts the finger off the screen, *action_up* is sensed and the media player stops playing the music.

4.4 FRAMEWORK COMPONENTS

AndroidManifest.xml

The AndroidManifest.xml is the most important file for an Android application that contains information of the package, including application components like activities, services, broadcast receivers, content providers, etc. It is present inside the root directory of every application. [8]

The manifest file:

- Names the Java package for the application.
- Describes the application components that the application is composed of.
- Determines the processes that will host the application components.
- Declares which permissions the application must have in order to access protected parts of the API and interact with other applications
- Declares the permissions that others are required to have in order to interact with the application's components.
- Lists the Instrumentation classes as the application is running.
- Declares the minimum level of the Android API required by an application
- Lists the libraries that the application must be linked against.

The AndroidManifest.xml file used in my application is given below:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.musicwine"
android:versionCode="1"
android:versionName="1.0" >
<uses-sdk
android:minSdkVersion="8"
android:targetSdkVersion="19" />
<application
android:allowBackup="true"
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme" >
<activity
android:name=".WineLevelActivity"
android:label="@string/app_name"
android:screenOrientation="portrait" >
</activity>
<activity
android:name=".MainActivity"
android:screenOrientation="landscape"
android:label="@string/title_activity_music" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```

Figure 4.4: Android Manifest File

The AndroidManifest.xml file of my application contains information about the Android package and version. Other details include application settings like the icon, labels for each screen, theme, screen orientation and name of the application.

4.5 Graphical User Interface (GUI)

Graphical user interface, in this context, allows users to interact with the application through icons and other visual notations. Singing Wine Glasses provide users with a very friendly interface that can be easily understood and smoothly operated by anyone, to produce enthralling music.

The GUI for Singing Wine Glasses is shown below:

Initial Screen

This is the welcome screen that user sees on opening the application. The user will be directed to the musical instrument with default water-levels of 60%, 80% and 40% respectively.

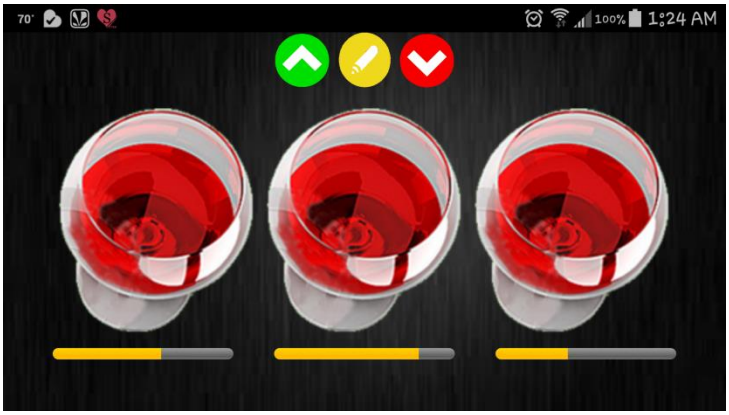

Figure 4.5: Initial Screen

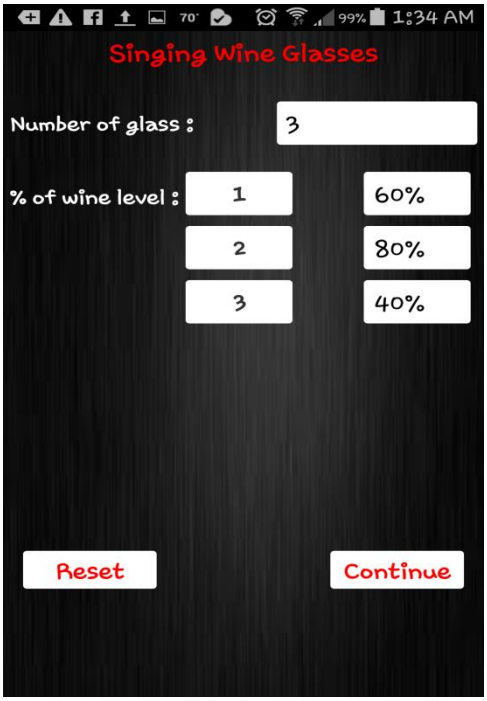

Figure 4.6: Initial Water Levels

After Pressing the *Down Arrow* icon

When the user clicks the icon with down arrow, the last glass would vanish and he would be directed to the screen for selecting the amount of water in each glass (or leave it as default). Here, he can also change the number of glasses again.

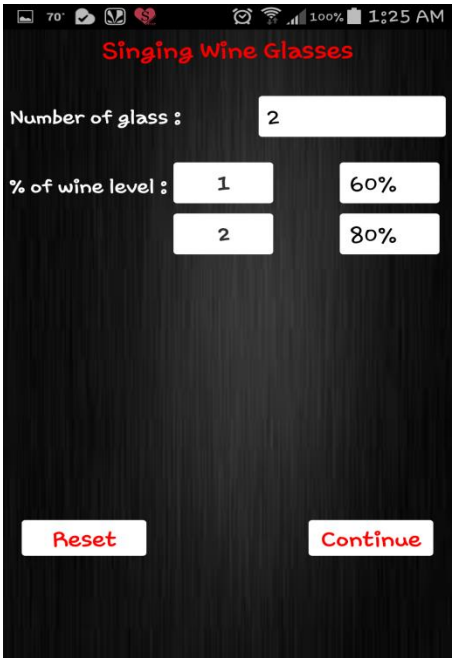

Figure 4.7: After pressing the *Down Arrow* icon

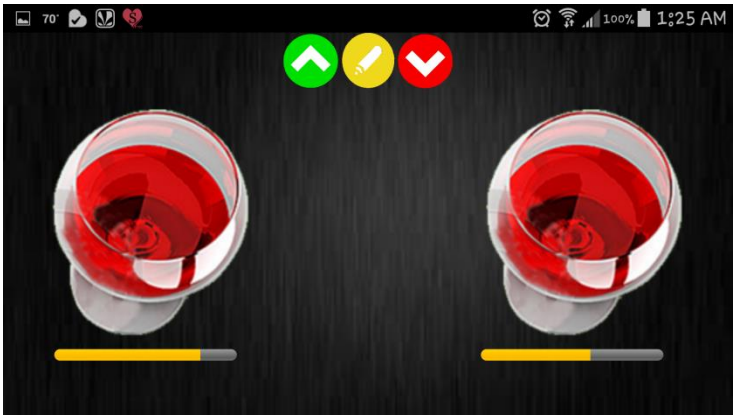

Figure 4.8: Instrument with 2 Glasses

After Pressing the *Up Arrow* icon

When the user clicks the icon with up arrow, another glass gets adjusted on the screen and the user would be directed to the screen for selecting the amount of water in each glass (or leave it as default). Here again the number of glasses can be changed.

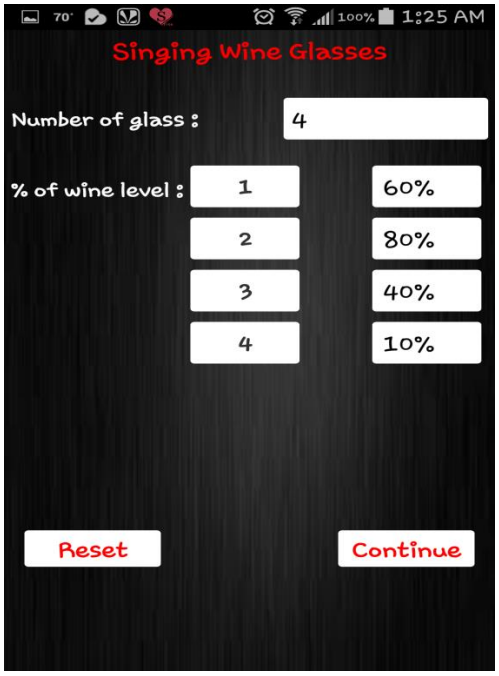

Figure 4.9: After Pressing the *Up Arrow* icon

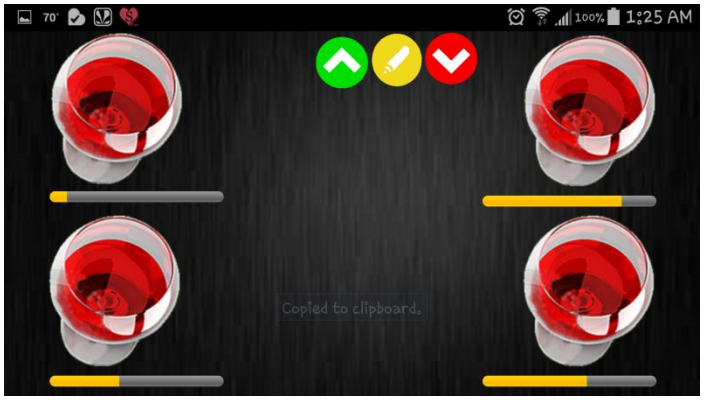

Figure 4.10: Instrument with 4 glasses

Filling the screen with the number of glasses

The screen gets automatically filled with the number of glasses a user selects, along with the progress bars underneath, that indicate the amount of water in each glass.

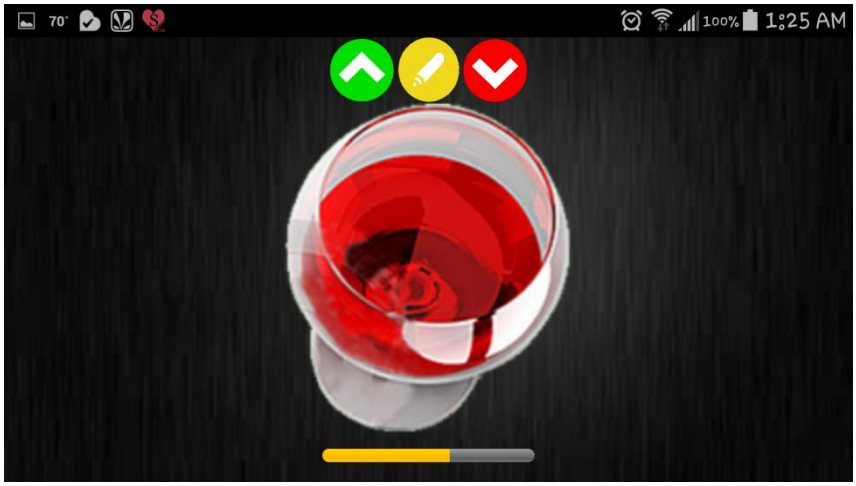

Figure 4.11: Instrument with 1 glass

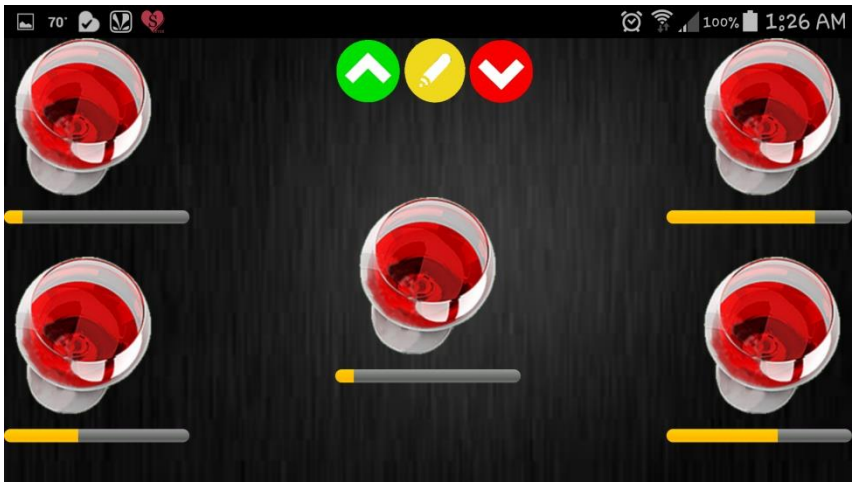

Figure 4.12: Instrument with 5 glasses

Error messages

The number of glasses has to range from 1 to 5. If the user tries to reduce the number of glass than 1 or increase the number beyond 5, then an error message pops up.

Figure 4.13: Error message when numbers goes below 1

Figure 4.14: Error message when number goes above 5

CHAPTER 5 – Testing & Evaluation

5.1 TESTING

Testing is the most important step of the Application Development Life Cycle and is generally carried out before delivering the project. During testing, a system along with its components are evaluated for its correctness and whether or not it satisfies all the specified requirements. It is done as a final step to identify any errors or missing requirements. [9]

5.1.1 Methods

The different methods used for testing a software/application are:

1. **Black-Box Testing**

   Black-box testing is a technique of testing that doesn't require any knowledge of the interior functionality of an application. The tester interacts with the UI, provides inputs and examines outputs being oblivious of how and where the inputs are worked upon.

2. **White-Box Testing**

   It requires detailed knowledge of the application's internal logic and structure of the code. Each chunk of code has to be checked for errors or inappropriate behavior.

3. **Grey-Box Testing**

   The user needs to have a limited knowledge of the internal workings of an application in order to carry out this testing technique.

Singing Wine Glasses application underwent several types of testing in order to prove its efficiency and awesome functionalities:

5.1.2 Unit Testing

Unit testing is performed on individual chunks of source code. Its goal is to isolate each unit of the application and show the correctness in terms of requirements and functionality.

The application passed the following required test cases while performing unit testing:

- Navigating to the instrument page as soon as the app is opened.
- Musical notes being played when the user swipes the rim of the glass with his fingers.
- Different notes for glasses with different water-levels.
- Music being sustained as long as the user swipes the rim of the glass.
- Music stops as soon as the user lifts the fingers off the screen.
- Up arrow to increase the number of glasses.
- Down arrow to decrease the number of glasses.
- Edit icon to change the water-level of the available glasses on screen.
- Being directed to the water-level selector screen on pressing up arrow icon.
- Being directed to the water-level selector screen on pressing down arrow icon.
- Being able to change the number of glasses from the water-level selector screen.
- Progress bar being displayed under each glass on the main instrument screen.
- New progress bars after changing the number of glasses.
- Filling the screen with the number of selected glasses.

5.1.3 System Testing

System testing tests the system as a whole after all the components are integrated, to check if the system meets the specified requirements. Singing Wine Glasses application also went through system testing after all the functionalities were summed up. The application did survive the test.

5.1.4 Compatibility Testing

An application needs to be checked for its compatibility with other platforms, operating systems and devices. Singing Wine Glasses application was successfully deployed on various Android devices like Samsung Note, S4, S3 and HTC.

5.2 Personal Evaluation

My experience of working on this application has been really great. I got to expand my knowledge on various topics. With the huge research about the glass harp instrument and the principle and science behind it, I was able to learn about frequency, how that changes with the type of glass and the amount of water in it. I also learnt a lot about the musical notes.

Developing the application in Android Studio was really fantastic. It provides a friendly platform for the developers to create amazing applications. Using native libraries to develop such applications makes it even more preferable. I learnt about the differences between using Android Studio and using Eclipse with ADT plug-ins. Creating different screens, expanding and contracting the images based on the number of glasses and producing different notes for different water-levels, were some of the functionalities that were really interesting to work with and deploy in the application.

CHAPTER 6 – Conclusion & Future Work

6.1 Conclusion

Singing Wine Glasses is an excellent application for all music lovers out there to experiment with a new and a different kind of musical instrument. Who could have imagined that the common glassware found on the table could produce such breathtaking music!!

This is a very user-friendly application. The application provides an interface that makes it easy for the users to set up the instrument. It is highly portable and reduces the risk of moving the entire apparatus from one place to the other without breaking it. This application also has an edge over several other musical apps related to Glass Harp, since it is more flexible and dynamic.

Singing Wine Glasses is sure to leave the users with a wonderful musical “exp-ear-ience”!!

6.2 Future Work

Singing Wine Glasses still hold a scope of improvement. It can be further enhanced with the following features:

- The app can be made to record the music played by the user.
- The recorded music files can be shared via social media.
- The app can be further enhanced to include an option for selecting the type of the glass, since music varies with the type of the glass too.
- The user interface can be further improved with 3-d image drawing.
- The app can be made to support other platforms, like iOS.
- Further improvement can be made with the sustainability of the music.
- Different flavors ranging from classical to lush violin and sweeping synthesizers can be included.

References

1. Science Buddies. (October 6, 2014). *Singing Wine Glasses*. Retrieved May 2, 2015, from http://www.sciencebuddies.org/science-fair-projects/project_ideas/Music_p008.shtml
2. Wikimedia Commons (n.d.). *Glass Harp Image*. Retrived Jan 12, 2013, from http://commons.wikimedia.org/wiki/File:Robert_tiso_glass_harp_beethoven.png
   The file is made availalble under Creative Commons CC0 1.0 Universal Public Domain Dedication
3. Wikimedia Commons (n.d.). *GlassArmonica Image.* Retrieved April 17, 2012, from http://commons.wikimedia.org/wiki/File:Glassarmonica.jpg
   The file is released into public domain worldwide
4. Tutorials Point. (n.d.). *Android.* Retrieved Feb 20, 2015, from http://www.tutorialspoint.com/android/android_overview.htm
5. Tutorials Point. (n.d.). *Android Architecture.* Retrieved March 13, 2015, from http://www.tutorialspoint.com/android/android_architecture.htm
6. Google, Inc. (n.d.). *Android Studio.* Retrieved n.d., from http://developer.android.com/tools/studio/index.html
7. Tutorials Point. (n.d.). *UML – Standard Diagrams.* Retrieved March 13, 2015, from http://www.tutorialspoint.com/uml/uml_standard_diagrams.htm
8. Google, Inc. (n.d.). *App Manifest.* Retrieved Feb 17, 2015, from http://developer.android.com/guide/topics/manifest/manifest-intro.html
9. Tutorials Point. (n.d.). *Software Testing.* Retrieved March 20, 2015, from http://www.tutorialspoint.com/software_testing/software_testing_quick_guide.htm
10. AppCrawler. (n.d.). *Wine Glass Music. Retrieved n.d.,* from http://appcrawlr.com/ios-apps/best-apps-wine-glass
11. Google Play. (n.d.). *Tiny Wine.* Retrieved n.d., from

https://play.google.com/store/apps/details?id=com.bluecoastapps.tinywine&hl=en