eTablaTutor – THE ELECTRONIC TABLA LEARNING GAME

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Dibakar Bhowmick

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

September 2010

Fargo, North Dakota

North Dakota State University
Graduate School

Title

eTABLATUTOR – THE ELECTRONIC TABLA LEARNING GAME

By

DIBAKAR BHOWMICK

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

ABSTRACT

Bhowmick, Dibakar, M.S., Department of Computer Science, College of Science and Mathematics, North Dakota State University, September 2010. eTablaTutor – The Electronic Tabla Learning Game. Major Professor: Dr. Kenneth Magel.

Tabla is a percussion instrument widely used in Indian classical music both as a solo instrument and as an accompaniment to vocal or instrumental music. Learning Tabla, like learning any musical instrument, is very much a discontinuous process. There are periods where one makes good progress. At other times, it feels that the learner is not making any progress at all. This alternation continues indefinitely. For one who has never learned a musical instrument, the first obstacle is the most critical. It usually occurs between 1 and 3 months. It is at this point that the initial enthusiasm has worn off and there is the aching realization that learning Tabla is not going to be quick and easy. This time is when a very large number of students drop out. If one realizes ahead of time, that it is a long and interesting journey, and that it could be more interesting and fun if he just completes a few initial steps, it often reduces the dropout rate. So, I have designed this software in order to attract people's attention for learning Tabla. This software would create people's interest in playing Tabla as a game with a beautiful interface which is easy to use and learn. For the users, there is a reference manual which describes the complete details of how to use eTablaTutor. This software mixes the old music tradition of India with the new technological learning of this musical instrument.

ACKNOWLEDGEMENTS

I would like to thank and express my sincere gratitude to Dr. Kenneth Magel, the major adviser for this paper, for his guidance and patience throughout this project. He encouraged me to bring together my passion for Tabla and computers. I would also like to thank all my committee members: - Dr. Weiyi (Max) Zhang, Dr. Jun Kong, and Dr. Sanku Mallik. I would definitely remember the support and help of my wife. I would like to thank my wife and my parents.

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENTS iv

LIST OF TABLES viii

LIST OF FIGURES ix

1. INTRODUCTION 1

1.1. Purpose of eTablaTutor 1

1.2. Evaluation of the Tabla 2

1.3. Traditional Tabla Strokes 2

1.3.1. Baya Strokes 3

1.3.2. Daya Strokes 3

2. RELATED WORK REVIEWED 5

3. BASIC IDEA, AIMS, AND OBJECTIVES 9

3.1. Basic Notes 9

3.1.1. Interface of the eTablaTutor 11

3.2. System Analysis 13

2.2.1. Identification of the Need 14

2.2.2. Preliminary Investigation 15

2.2.2.1. Conducting the Preliminary Investigation 16

2.3. Feasibility Study 17

2.3.1. Technical Feasibility 18

2.3.2. Social Feasibility 19

2.3.3. Economic Feasibility 19

2.3.4. Operational Feasibility 20

2.4. Software Engineering Paradigm Applied 21

2.5. Software and Hardware Requirement Specifications 24

4. DESIGN OF eTablaTutor 25

4.1. Module Description 25

4.1.1. Playing Mode 25

4.1.2. Tutorial Mode 25

6.2. Screen Shots of the Interface 26

5. CODING AND TESTING 31

5.1. MainInterface.cs 31

5.2. MainInterfaceMotion.cs 36

5.3. Main Form Background Code 42

5.4. Tabla.cs 59

5.5. Function Description 64

5.6. Testing 67

5.6.1. White-box Testing 68

5.6.2. Black-box Testing 69

5.6.2.1. Unit Testing 70

5.6.2.2. Integration Testing 71

5.6.2.3. System Testing 71

6. FUTURE SCOPE 73

6.1. Future Scope of the eTablaTutor 73

6.1.1. Model Tabla Structure for the Future 73

6.1.2. Make More Easy, Small Levels 74

6.1.3. Real Tabla Display in the Interface 75

REFERENCES 76

LIST OF TABLES

Table **Page**

1. Class, Function and Use 64

LIST OF FIGURES

Figure **Page**

1. Top View of Baya 10

2. Top View of Daya 11

3. Top View of Tabla 11

4. Main Screen of the eTablaTutor 27

5. "Select Level" Options in the Menu Bar 27

6. "How to Play" Options in the Menu Bar 28

7. Ball Movement Screenshot 29

8. Continuation of the Game 29

9. Burning Ball Screenshot 30

10. "UserManual" Screenshot 30

11. Model Tabla for Future eTablaTutor 73

1. INTRODUCTION

Tabla is a very popular percussion instrument. It is a pair of hand drums that is very popular in India, Bangladesh, and Pakistan. Indian music, mainly the Indian classical music, is very weak without this instrument. In recent times, it has also gained acceptance in western and fusion music. Tabla has two pieces: one is called Baya or Bayan, and the other one called Daya or Dayan. A Tabla is traditionally used to accompany North Indian vocal and instrumental music. Different types of Indian songs, e.g. classical, semi-classical, bhajan, ghajal, folk, tagore's music, and modern song, are based on Tabla. The smaller wooden drum is known as the Dayan, and the larger drum is known as the Bayan. The Dayan can be tuned by adjusting the position of the cylindrical wooden pieces on the body of the drum, and the Bayan is tuned by adjusting the tightness of the top rim. The Tabla is a very unique instrument and has a very unique sound because the drumheads of the Bayan and Dayan both have weights at the center made of a paste from iron oxide, charcoal, starch, and gum. It is a very common and effective culture of India that a Guru (teacher or master) usually teaches the Tabla to a shikshak (student).

1.1. Purpose of eTablaTutor

The purpose of this project is to help students who are eager to learn the Tabla enhance their interest for the first few initial stages with this easy learning game. People like to play computer games, and sometimes, they pass their time with meaningless, easy, old computer games. My intention is to catch their attention by making this easy computer

game and to let them discover their interest in Tabla. Other than that, it is also very difficult to find a Tabla teacher in many places. A Tabla teacher is not too difficult to find in India, Bangladesh, and Pakistan, but it is very difficult is many other countries especially in western countries. Therefore, this application will definitely help those who are waiting to find a Tabla teacher. This eTablaTutor will teach people the basic notes, sounds, boal, taals etc. In this way, they can finish a few basic and beginning levels and then go for the real face to face teacher-student learning session. Additionally, this game will also increase the popularity of Tabla.

1.2. Evaluation of the Tabla

"It was said that the famous Pakhawaj player Sidhar Khan provoked an angry dispute after losing a music contest and that his Pakhawaj (a genre of Indian drum defined by a barrel with drumheads on either side) was chopped in half by a sword. Thus, the first Tabla was created accidentally. It was created in the 18th Century by Sidhar Khan Dhar." (2)

The Dayan is made from wood. The Bayan used to be created with clay: - as the technology for producing metal alloys evolved, the Bayan started to be molded from brass and steel.

1.3. Traditional Tabla Strokes

It is important to understand the traditional playing style of the Tabla to see how this eTablaTutor models its hand movement.

1.3.1. Baya Strokes

The Bayan is also known as Duggi in some places. There are three main strokes played on the Bayan or Duggi. "The ***Ko*** stroke is executed by slapping the flat left hand down on the Bayan. The slapping hand stays on the drum until it stops the resonance and then released away." The ***Ge*** stroke is executed by striking the *Maidan* (small area, not the big area) just above the *Gab/ Syahi* with index finger of the left hand. When the finger strike, it need to release away from the drum in order the keep the resonance. "The heel of the left hand controls the pitch of the *Ge* stroke." The other stroke is ***Dha*** which is mixture of Dayan and Bayan, and it needs to use the middle finger of the left hand; other than the finger used, it is the same as the *Ge* stroke.

1.3.2. Daya Strokes

There are 6 (six) basic strokes played on the Dayan : -

Na - stroke the *Chaoni* or *Chat* with the index finger of the right hand and quickly releases it so that the sound of the drum resonates. This stroke is executed by lightly pressing the pinky finger down between the *Chaoni* and the *Maidan*, and the ring finger down between the *Gab* and the *Maidan*. The pinky and ring fingers actually mute the sound of the *Gab* and *Maidan* of the drum. This same stroke is also called *Ta*.

Te - stroke is similar to *Ta* except the middle and ring finger of the right hand strike the center of the *Gab*. This stroke does not resonate and creates a damped sound.

Tae - stroke is played by striking the *Gab* with the index finger of the right hand

Ti - stroke is executed similar to *Na*, except that the index finger strokes the *Maindan* instead of the *Chaoni.*

Tun – stroke *Gab* with the index finger lightly and no other finger will touch anything.

Dha - Dha is a combination of two strokes, one from Dayan (Na) and Bayan (Ge with index finger). This sound will resonate.

2. RELATED WORK REVIEWED

In the process of brainstorming for my eTablaTutor idea, I went through several works, some of which I tried to describe here.

Hit Song Science is a product which is for individual musicians and song writers so that they can check if their music has the potentiality for being hit music or not. Their algorithm will check the patterns to see whether the new music fits with the previous hit music trends to see if it has the same potential pattern or not. It will also check the best age group or community for those who will be more interested in it, and according to that, the composer, individual musician, and song writers will go for the TV channel specially music channel as a target. The system works in the following manner: -

- They have a large sample of music which is more than one million tracks, and they have also access to a quarter million CDs
- Their analysis application is designed to listen to any CD and be able to isolate patterns in many musical events. (e.g., melody, harmony, tempo, pitch, octave, beat, rhythm, fullness of sound, noise, etc.)
- The software will try to find a match between new music and previous music based on those patterns of the musical event
- As a result, it will provide information to an individual musician, song writer, or professional composer about the potentiality of their music
- They also have published a few real success stories to prove their potential product

Therefore, the authors have options and opportunities to change or improve their music. The music lovers will get good, quality music as they wish to have (1)

In the Electronic Tabla Controller project, the authors have used the technology to create a real-time instrument that models the Tabla. The Electronic Tabla (eTabla) has digitizing sensors, custom positioned to the traditional Tabla technique, which convert finger strikes and hand slaps to binary code which a computer can understand. These signals then trigger real-time sounds and graphics (2).

I found another paper where the author believes that the listener creates musical meaning. My understandings of that paper is that listeners, or the audience, first listen to the music, relate that music with other musical patterns they liked which are similar to this, and then consciously or unconsciously relate with that. The author describes the theory of musical meaning based on a dynamic field of musical forces, such as Inertia, Gravity, or Magnetism of the music; all experienced listeners of tonal music hear musical motion metaphorically. The ubiquity of these patterns raises interesting queries, and the author ended his presentation with some questions about searching for musical patterns like what will be the role of computers between information retrieval and musical artificial intelligence (3).

TablaNet, a real-time online musical collaboration system for the Tabla, a pair of North Indian hand drums. Mihir Sarkar created *TablaNet* where musicians who aspire to play with each other, even if they are located in different countries, can now play almost as if they were in the same room. This system is based on a novel approach that combines machine listening and machine learning. *TablaNet* is obviously trained for a particular instrument, the Tabla. The system *recognizes* individual drum strokes played by the

musician and sends them as symbols over the network. A computer at the receiving end identifies the musical structure from the incoming sequence of symbols by mapping them dynamically to known musical constructs. There is a little time delay due to the inherent latency in computer networks, and to deal with transmission delays, the receiver *predicts* the next event by analyzing previous patterns before receiving the original events and synthesizes an audio output estimate with the appropriate timing. Although prediction approximations may result in a slightly different musical experience at both ends, the authors found that this system demonstrates a fair level of playability for Tabla players of various levels and functions well as an educational tool (4).

Tabla Deva is a mobile Tabla machine for the iPhone. It is also applicable for the iPod Touch and iPod. This application was developed by Acoustic World. It has different Taal playing options with different speeds: slow, medium, and fast (5).

iTablaPro is an electronic Tabla and Tanpura (drone) player for the iPhone and iPod Touch. It sounds like the real Tabla and is ideal for Indian classical musicians and students. It includes support for all common Taals used in Hindustani music, making it the perfect companion for daily riyaz (practice). This application was developed by Prasad Upasani, an Indian classical musician who has received a master's degree in computer science; he has worked 15 years in IT and is now working as an IT manager for iPhone (6).

LaDiDa is a reverse Karaoke system on the iPhone. If you sing a song, then this system will recognize the taal needed to play this song and start playing the right beat for that particular song. *LaDiDa* was developed by Parag Chordia who has worked on the Realtime Tabla recognition system. Parag Chordia completed his Ph.D at Standford

University and is now working as an assistant professor of Georgia Tech Center for Music Technology (7).

Pocket Bhagra is an iPhone application. This application has three instruments: "Dhol," "Tumbi," and "Tabla." In Tabla, there is a pair of drums that will allow you to vary the pitch of the percussive sounds. You will hear very clean, studio-quality audio samples from a professional Tabla player. The application also has a tuning mechanism to change the pitch of each Tabla drum and clear "hit zones" about where to hit the Tabla. You will also find a special "sliding hit zone" in this application to produce a waving sound on the bayan drum, and it has ability to play the Tabla along with dhol or tumbi loops (8).

Tabla Master is an iPhone application where you can pick a song and play the Tabla on your iPhone or iPod Touch. This application was developed by Sean Mikhail. On the screen, you will see the Tabla head, and if you hit the specific area, you will hear specific Tabla sounds for separately in Baya and Daya (9).

3. BASIC IDEA, AIMS, AND OBJECTIVES

The idea is to creating gaming software to learn and enjoy the Tabla. I am trying to describe a little more about the idea here -

eTablaTutor is an automatic tutor which will teach basic lessons on the Tabla. The users will learn Tabla through this gaming software. They will start learning while they are playing and enjoying the game.

3.1. Basic Notes

The Tabla has different notes, but there are 10 basic notes that can mix and produce different sounds. Those notes are –

Te	Tae	Ta	Dha	Tun
Ko	Ti	Dhi	Ge	Dhin

I also need to mention that there are some similarities in sound when a Tabla player plays some notes, so we will not consider both of them as basic notes (except for 'Na'). Those notes are as follows: -

Base Notes	Similar Notes
Ta	Na (My application has both as basic notes to make it easy to practice)
Ta	Ne
Tun	Tin

Ko	Koth (Sometimes people pronounce it K)
Ge	Gha
Te	Tie

Here is a combinations of two notes that act as one note: -

Dha + Ge = DhaGe; Na + Ge = NaGe

Na + Ko = NaKo; Ti + Tae = TiTae

Figure 1 is showing top head of Baya where user can play *Ko* and *Ge*

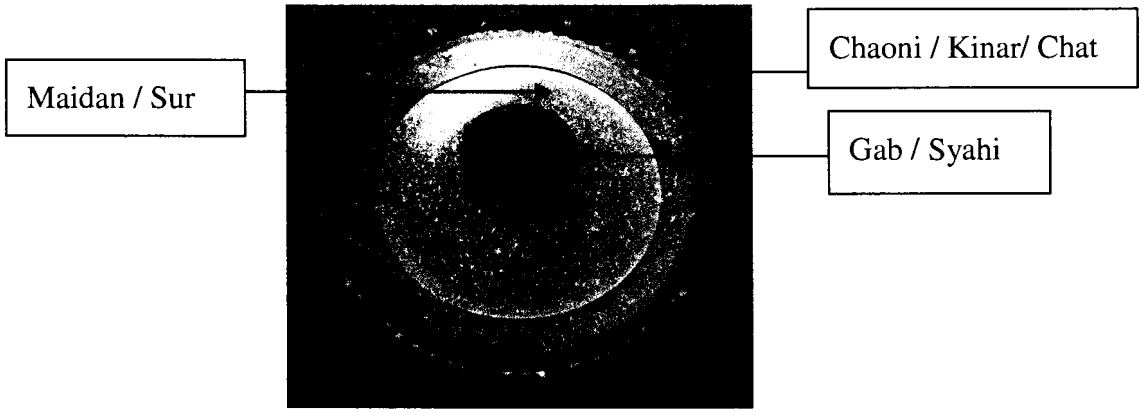

Figure 1. Top View of Baya

Figure 2 is showing top head of Daya where user can play *Ti, Te, Tae, Ta, Tin,* and *Thun*

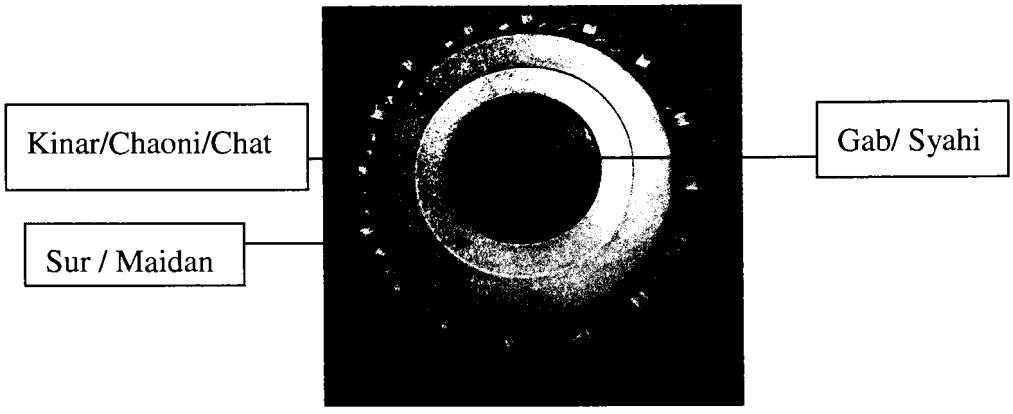

Figure 2. Top View of Daya

Figure 3 is showing top head of Tabla where user can play combine notes *Dha* and *Dhin*

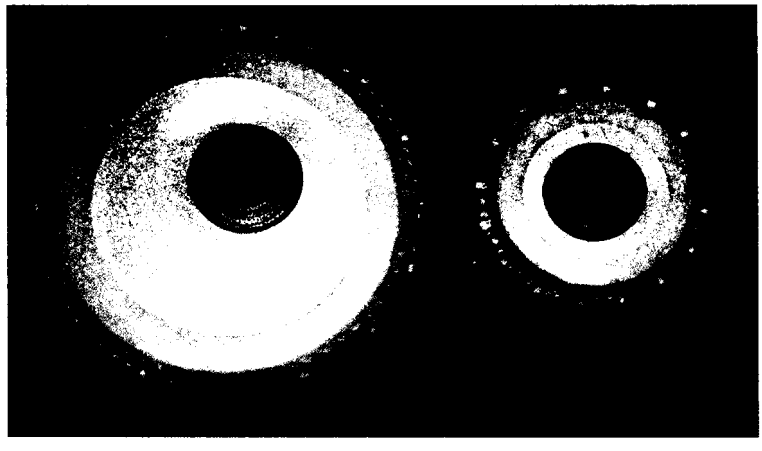

Figure 3. Top View of Tabla

3.1.1. Interface of the eTablaTutor

I have assigned some keys on the keyboard to demonstrate my work because of the cost of making a plastic or metal Tabla. On the screen, 11 basic notes with every assigned

key and the display screen are showing which basic note to play; the user will hit or click on that key. There are four levels of practice a user can try. The background picture of a Tabla player who is playing a Tabla helps the user to understand the right posture and, of course, is good to see. One dynamic ball is running here and there, and it will continue until the user misses or hits the wrong key. The learner (user) follows the ball and practices the Tabla. If the user is wrong, then the ball will burn, and it will make a sound.

The system has four levels.

Level 1 – Basic Tabla Notes Introduction

A time gap is not required. The user can play any basic note of the Tabla at any time to get familiar with those notes.

Level 2 – Bool Practice in slow speed (A time gap is required for this level.)

Dha	TiTae	KTae	Tako
Tun	TiTae	KTae	Tako
Tun	TiTae	KTae	Tako
DhaGe	TiTae	KTae	Tako

Level 3 – Dadra Taal practice in slow speed (A time gap is also required for this level.)

Dha Dhi Na Na Ti Na

OR

Dha Dhin Dha NaKo Tun Na

Level 4 – Dadra Taal practice in medium speed (A time Gap is also required for this level.)

Dha Dhi Na Na Ti Na

OR

Dha Dhin Dha NaKo Tun Na

3.2. System Analysis

It is the most important phase of the system development cycle. It is defined as "a process of examining a situation with the intent of improving it with better procedures and methods. (11)" Analysis involves deep study of the task. System analysis is the process of gathering and interpreting facts to diagnose the problems, defining the goals, designing the constraints, and using the facts to improve the system. A wrongly understood system could lead to false decisions. Systems are analyzed in terms of their objectives as well as the input, process, and output required to achieve the goals. The aim is to identify the boundaries of the system, their sub-systems, and the interface between sub-systems and systems (11). System analysis involves: -

- Studying the existing system
- Running the utilities on all systems
- Having user-friendly utilities so that everybody can use the system
- Studying the document's flow diagram
- Identifying the document involved,
- It should run on the entire hardware/operating system platform.

Performance Requirements:

- User-Friendly

The eTablaTutor is user-friendly and can easily be understood by the user without any difficulty. Then, he/she can start playing it

- Ease of Maintenance

eTablaTutor is easy for maintenance because the only thing that needs to be taken care of is the files used for playing sounds. The other code is hardcoded and is implemented through these music files only

- Less Time Consuming

The tutor is less time consuming and is ready to use. There is no need to wait for the startup of a program. It is executed in seconds

- Portability

This gaming software is portable, too, and can be deployed on any system through preparing its easy installation setup which could embed all the codes and files used

- Error Free

As it is desgined, the eTablaTutor is prone to errors during run time until all the initial requirements for this software are fulfilled.

Two of the steps involved in system analysis of eTablaTutor are identifying the need and a preliminary investigation.

2.2.1. Identification of the Need

Now these days everything is getting automated with the help of information

technology. It made everything so easy and fast that it has become very easy to play real-time games. We can easily simulate a real-time system in software that could be easier for analysis and understanding by a "newbie" who is not efficient in using those real-time systems. In order to help and serve "newbie" with handling such jobs that are not meant for them or are difficult for them, it is necessary to make a tutorial that would be helpful in understanding things easily. The same concept has been applied in the development of this software, eTablaTutor.

People fascinated with learning Indian music are also interested in playing the Tabla instrument, so they need to find a teacher to teach them the Tabla. I thought about developing a software program that could help in learning the Tabla without the intervention of a human teacher for the first few stages, thus making it easier if I used visual graphics and audio effects for learning. This requirement was the basic need that was identified in the development of this program.

2.2.2. Preliminary Investigation

Preliminary investigation is one of the important activities in the software development life cycle. It is the first step in the development life cycle and determines the feasibility of the system. The purpose of preliminary investigation is to evaluate requests. Preliminary investigation is collection of information that helps committee members to evaluate the merits of the project request and to make informed judgments about the feasibility of the proposed project.

Analysis working on the preliminary investigation should accomplish the following objectives:

- Clarify and understand requirements such as dhuns and stroke times of Tabla
- Determine the size of the eTablaTutor program
- Assess the costs and benefits of alternative approaches in designing this tutor
- Determine the technical feasibility of alternative approaches
- Report the finding to committee with recommendations outlining
- the acceptance or rejection of the proposal

2.2.2.1. Conducting the Preliminary Investigation

The data that the analysis collects during the preliminary investigation are gathered through three preliminary methods.

- Reviewing organization documents: All the documents and notes have been gathered; they were helpful in designing the logic of this tutor before the start of actual design, coding, and implementation of this eTablaTutor idea. The documents have been organized and indexed on the basis of requirements according to project-development planning
- On-site observations: The purpose of on-site observation is to "collect data by observing the activities of the system directly (11)". During the on-site observation, I have seen the real environment where the Tabla is being learned by people, and from that observation, I have created the interface design where a man is playing

with a Tabla and enjoying the essence of its music; this interface can be seen in the main page design. Its visualization enhances the user's interest in learning this instrument or playing it as a game

- Conducting interviews: Interviews are the mandatory part during the design period. It gives an idea of what the Tutor should look like and what the user actually wants to get in the same environment of learning the Tabla. Conducting interviews makes it more clear about the basic requirements of the Tabla tutor's purpose

It is often convenient to make a distinction between two kinds of questions:

- Open question
- Closed question

Open questions are general questions that establish a person's viewpoint on a particular subject. Thus, three might be a question such as

- How many boals and taals should be used for basic learning to be learn the Tabla in eTablaTutor?
- How should teaching the Tabla implemented in it (audio/video)?

Closed questions are specific and usually require a specific answer. For example,

- What are the basic rules to play a Tabla?
- What should be the flow diagram of the working Tabla tutor?

2.3. Feasibility Study

An event is said to be feasible if it is considered possible and practicable. "The feasible study is an investigation into how possible a proposed scheme might be and whether it is capable of being carried out successfully (11)". This is usually assessed on four standard criteria for feasibility, but other considerations may be relevant and necessary depending on the specific nature of the project and its environment. "It refers to the phase where the proposed system is tested for whether it is really required in the present working conditions (11)". It is a very important step in software development as its result determines whether the system has to be developed or not developed. The standard assessment criteria and typical questions they address are as follows:

- Technical feasibility: Can I do this project?
- Social feasibility: Do I want this project?
- Economic feasibility: Can I afford this project?
- Operational feasibility: Can I handle the outcome of this project?

2.3.1. Technical Feasibility

The proposed system is technically feasible because

- The proposed eTablaTutor is capable of providing adequate responses.
- It, being modular, can add more features in future, such as graphics enhancement and more levels in playing.

As far as the software and hardware are concerned, the proposed eTablaTutor has several characteristics.

It is feasible as the size of program is low and requires less disk storage.

- The technology is available to make it work.
- It is possible to achieve the proposal within the performance criteria.
- There are sufficient skilled technologists available to staff this project.
- The proposed features are new for this type of gaming software.

2.3.2. Social Feasibility

The proposed system is socially feasible because

- The proposed system is physically possible, and there are no negative repercussions and impacts on people in and outside the learning environment of Tabla.
- The proposal will influence learning practices and make people interested in playing system as a game.
- The proposed eTablaTutor will enhance people's interest in learning Indian musical instruments if they are keen to learn the Tabla, thus making a traditional instrument work in a live current-time scenario.
- This application considers the social cost, human issues, cost to the environment, society, and classical music culture.

2.3.3. Economic Feasibility

eTablaTutor is economically feasible because

- The cost of the hardware and software is quite affordable
- The playing sounds can be easily available free of cost
- The capability of the system as a tutor is high as compared to the cost of hiring a real tutor for teaching the Tabla
- Processing time and execution time are fast and reliable
- The software not only acts as a tutor, but is also a fun game and could make people interested. One will always try to hit the correct key on the keyboard and to prevent the ball from burning. This game is fun

2.3.4. Operational Feasibility

The system is operationally feasible because

- The eTablaTutor is approved by people and by the guides under which this project is being developed
- Because the new system is going to ease the learning procedure of Tabla, the proposed system will help in improving the total performance
- Users are an active part of the system and supportive to the system
- New changes can easily be implemented without any difficulty in understanding because the working manual will always be there
- The system is easy to use and handle

2.4. Software Engineering Paradigm Applied

"A software process model or software engineering paradigm is chosen based on the nature of the project or application, the methods or tools to be used, and the controls or deliverables that are required. Software engineering methods provide automated or semiautomatic support for processes and methods (12)". All these methods have been studied during the life cycle of this Tabla tutor. Points are focused under the given categories and discussed in detail.

A typical roadmap within a bespoke development project incorporates the following core development phases and their associated tasks and development stages:

1. Analysis

- Project Overview
- Feasibility Study
- Initial Requirement Analysis
- Budgeting Strategy
- Resource/Responsibility Planning
- Consultancy/Project Scoping

2. Design

- Review Project Overview
- Requirement Definition
- Data Analysis

- Software Functional Design
- Create Functional Specification Design
- Review Functional Specification
- Develop Initial Prototype Sample
- Review Project Specification
- Aesthetic Design Specification
- Specification Change Control/Acceptance Check
- Create Final Project Specification Design
- Review Final Project Specification Design
- Project Design Completion

3. Preparation

- Specification Conformance/Acceptance Check
- Code Modifications
- Integration/Data Testing
- Final Code Modifications
- Code Completion Change Control
- Help/Documentation Specification
- Help/Documentation Production

4. Development

- Development Division of Labor
- Coding

- Initial Debugging
- Report Writing
- Aesthetic Design

5. Implementation

- Beta-version User Training
- Beta-version Deployment
- Pre-implementation Modifications
- Software Completion
- System-manager Training
- Technical-environment Deployment
- User Training
- Final Data Conversion/Data Preparation
- Application Integration/Data Loading
- Parallel Installation

6. Support

- Post-implementation Review
- Future-requirement Planning/Change Control
- Technical Support
- Routine Service and Maintenance
- Future Development

2.5. Software and Hardware Requirement Specifications

Tools used:

- Front-end: Microsoft Visual C# (VC#, Enterprise edition)
- Reason: VC# is proposed as the front-end tool because it provides an integrated development environment that allows the user to create, run, and debug the program without opening any other program or application. Additionally, VC# provides a window-like work environment that supports rich Graphical User Interface (GUI).
- Platform: Windows 7 Ultimate Pack. The proposed system will be built with the said platform. The proposed system will use Windows because the users at the workspace have a good hand over Windows making them; work on some other platforms may not seem feasible.

Minimum hardware requirements are as follows:

- Pentium III Processor
- At least 128 MB RAM
- 41 GB Hard Disk
- VGA Monitor
- Floppy Disk Drive
- CD-ROM Drive (32x or higher speed recommended)
- Mouse
- Keyboard

4. DESIGN OF eTablaTutor

4.1. Module Description

The modules used are playing and tutorial mode. The modes are described in the following sections.

4.1.1. Playing Mode

There are four levels of playing this game. Each level has a unique pattern of hits to the Tabla head. Every unique hit is made of different sound patterns. The pattern in which a user is going to hit the Tabla through the keyboard's keys, will vary according to the designed music. Each hit will produce a specific sound. If a wrong hit is made, this error will be indicated through the moving ball on the screen interface.

4.1.2. Tutorial Mode

Similar to the Playing Mode, in Tutorial Mode, each level is described initially with how the sound should be prepared. You have to listen carefully to remember those music patterns as guided in the tutorial. One can clear his doubts regarding the game with the Playing Manual provided in this section. The tutorial discusses each of the playing levels in detail with the help of visible and audio instructions.

6.2. Screen Shots of the Interface

The main screen is shown in Figure 4. This main screen is complete in its user interface design and the screen is easy to understand. The ball in the center (blue color initially) is movable, and there are buttons with named dhuns and corresponding are the letters to be pressed from keyboard. The Play (bottom-right) button will start the game at a particular level. The menu at the top consists of two options:

- Select Level: The user can select a level to start playing.
- How To Play: An audio guide with a manual will describe how one can play this game.

In Figure 5 the menu bar shows that it can be expanded to give four choices of levels. The user can select any of the levels according to his skills and can play at that level.

In Figure 6 the second menu bar "How to Play" shows that it can be expanded to give four choices. The user can select the first three choices that will then explain how to play the game at levels 2, 3, and 4, respectively. The fourth choice will show the complete playing manual which will describe the working of this game.

In Figure 7 shows the movement of the ball once the game starts. As the game starts, the ball in the center begins moving all over the main screen, and the next character to be pressed starts scrolling from right to left over the orange-colored panel. The user has to press the appropriate button through the keyboard to play the Tabla sound hits. The "Play" button on the bottom right changes to the "Stop" button. To stop the game, the button can be pressed again.

Figure 4. Main Screen of the eTablaTutor

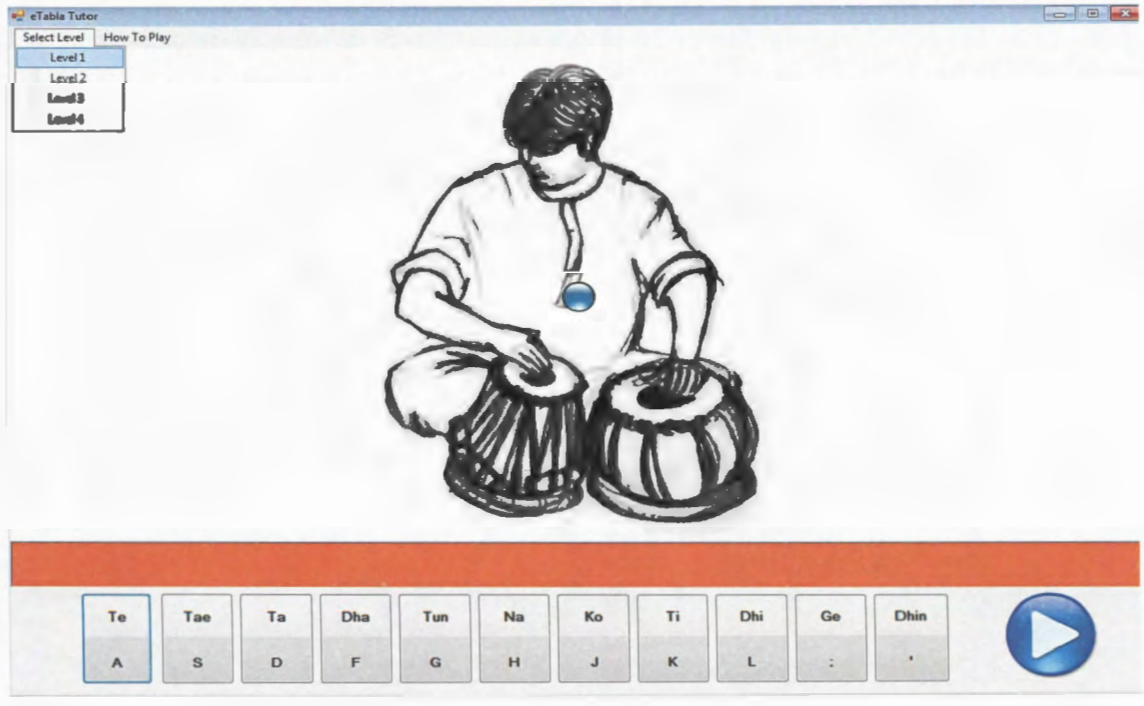

Figure 5. "Select Level" Options in the Menu Bar

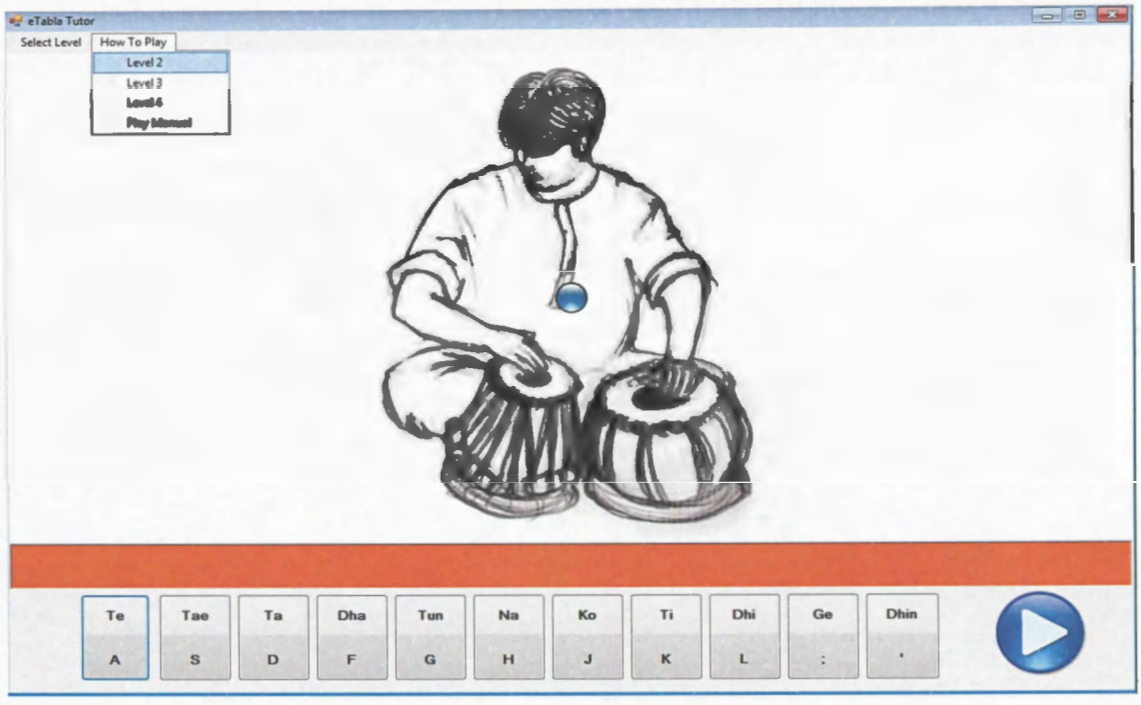

Figure 6. "How to Play" Options in the Menu Bar

In Figure 8 shows the continuation of the game with highlighted of Keys. This screenshot shows how stokes are being displayed, and the ball is still moving at different positions over the main panel of the design.

In Figure 9 shows the ball burns at the hit of wrong key. When a stroke from a keyboard is missed, then the ball changed the color.

In Figure 10 shows the screen shot of the "User Manual" of the game. The screenshot illustrates guidelines to play the game.

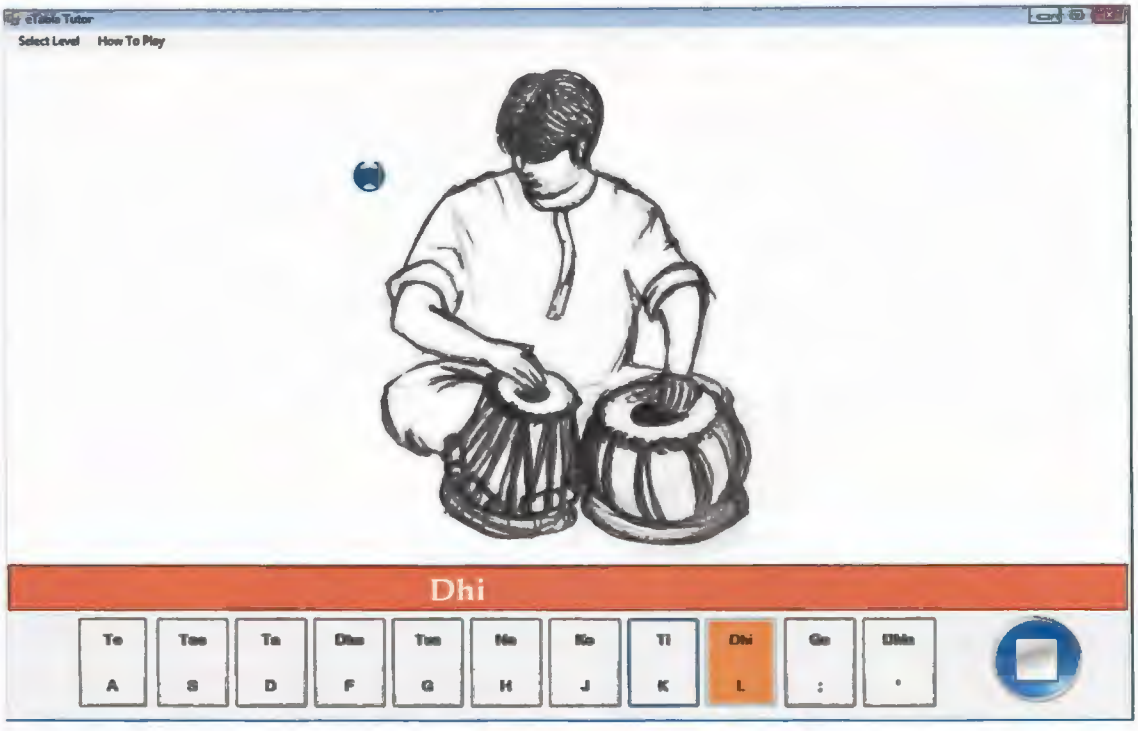

Figure 7. Ball Movement Screenshot

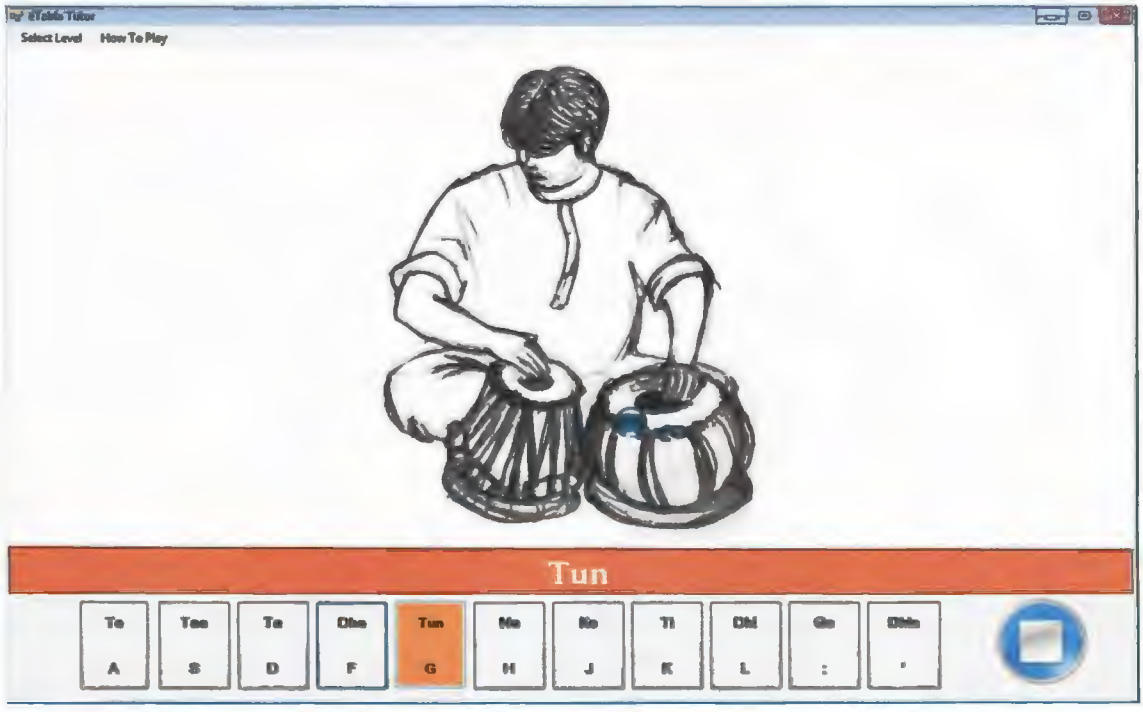

Figure 8. Continuation of the Game

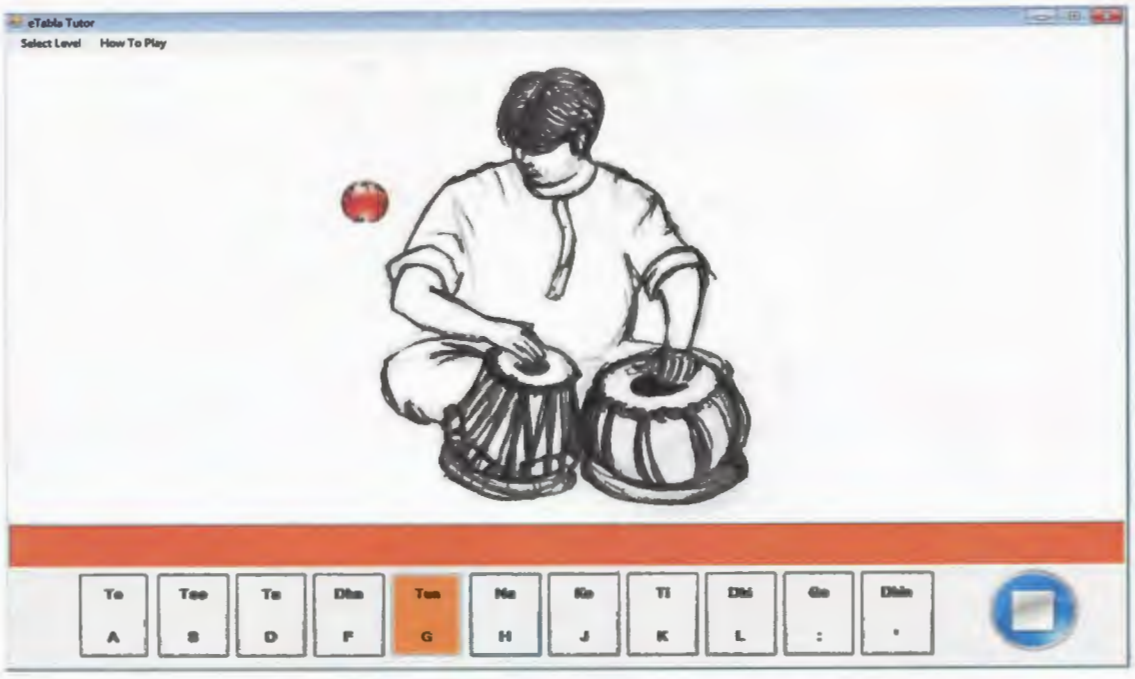

Figure 9. Burning Ball Screenshot

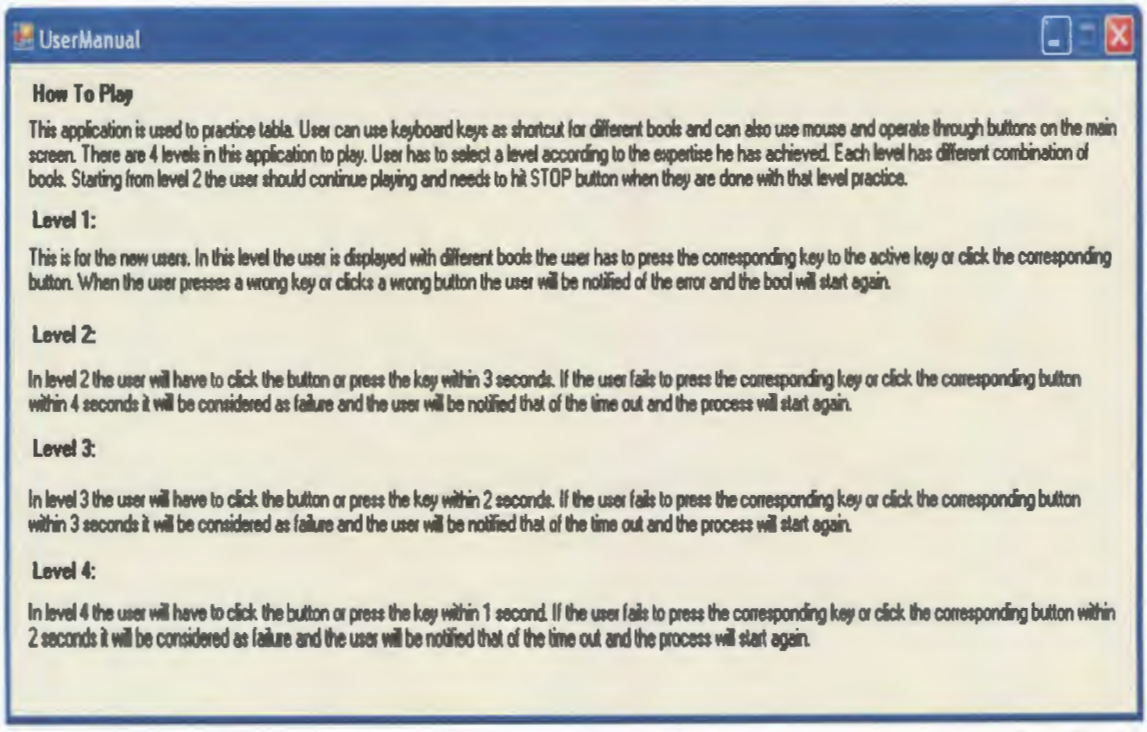

Figure 10. "UserManual" Screenshot

5. CODING AND TESTING

5.1. MainInterface.cs

This file is used for controlling the next note to be played and the event capture of the current note. Notes are also reset for a new level and animate it.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;
namespace eTabla
{
   partial class MainInterface
   {
      private string[] currentNote;
      private Point xInitNote;
      private Point currentXNote;
      private Point mainXNote;
      private int offset;
      private void InitInterface()
```

```
    {
        this.xInitNote = new
Point(this.noteLbl.Left+this.noteLbl.Width,this.noteLbl.Top);
        this.currentXNote = this.xInitNote;
        this.mainXNote = new Point(-this.noteLbl.Width, this.noteLbl.Top);
        this.offset = 3;
    }
    private void setNextNote(string[][] allNote)
    {
        for (int i = 0; i < allNote.Length; i++)
        {
            if (currentNote == allNote[i])
            {
                if (i == allNote.Length - 1)
                {
                    //randomiseNotes();
                    currentNote = allNote[0];
                    break;
                }
                else
                {
```

```
                currentNote = allNote[i + 1];
                break;
            }
        }
    }
}
private void randomiseNotes()
{
    missHit = true;
    string[] tmp = new string[] { string.Empty };
    Random rnd = new Random((int)DateTime.Now.Ticks);
    int fr, sr;
    for (int i = 0; i < allNotes.Length; i++)
    {
        fr=rnd.Next(allNotes.Length - 1);
        sr = rnd.Next(allNotes.Length - 1);
        tmp = allNotes[fr];
        allNotes[fr] = allNotes[sr];
        allNotes[sr] = tmp;
    }
    missHit = false;
```

```
}
private void resetNotePosition()
{
   this.currentXNote = this.xInitNote;
   this.noteLbl.Left = this.currentXNote.X;
}
private void animate()
{
   this.noteLbl.SuspendLayout();
   string noteLblTxt = string.Empty;
   for(int i =this.currentNoteIndex ; i<this.currentNote.Length;i++)
   {
      noteLblTxt +=this.currentNote[i];
   }
   this.noteLbl.Text = noteLblTxt;
   if (this.noteLbl.Left <= (-(this.noteLbl.Width)))
   {
      resetNotePosition();
   }
   else
   {
```

```
            this.noteLbl.Left = this.noteLbl.Left - this.offset;
        }
        this.noteLbl.ResumeLayout();
        System.Windows.Forms.Application.DoEvents();
    }
    private void resetNotes()
    {
        currentNote = allNotes[0];
    }
        private void InitializeComponent()
    {
        this.SuspendLayout();
        //
        // MainInterface
        //
        this.ClientSize = new System.Drawing.Size(284, 262);
        this.Name = "MainInterface";
        this.Load += new System.EventHandler(this.MainInterface_Load);
        this.ResumeLayout(false);
}
}
```

}

5.2. MainInterfaceMotion.cs

This file contains code that is used for setting and controlling the graphics of the moving ball all over the main window while the game is being played. The rotation of the ball is set by its x and y co-ordinates of the window panel by adjusting the percentage through the margins.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;
using System.Media;
namespace eTabla
{
  partial class MainInterface
  {
    #region Motion Area
    private Point LocPoint;
    private int xDir;
```

```
    private int yDir;
    private int xOffset;
    private int yOffset;
    private bool missHit = false;
    #endregion
    public void InitMotion()
    {
        int x = this.objectContainer.Width / 2;
        int y = this.objectContainer.Height / 2;
        LocPoint = new Point(x, y);
        this.xDir = +1;
        this.yDir = +1;
        this.xOffset = 2;
        this.yOffset = 2;
        this.objectBox.SuspendLayout();
        this.objectBox.Left = LocPoint.X - (objectBox.Width / 2);
        this.objectBox.Top = LocPoint.Y - (objectBox.Height / 2);
        this.objectBox.ResumeLayout();
        this.objectBox.Invalidate();
    }
private void nextPosition()
```

```
{
    Point nextLocation = new Point(this.LocPoint.X + (xDir * xOffset),
this.LocPoint.Y + (yDir * yOffset));
    //x boundry checking
    if (nextLocation.X >= (objectContainer.Width - objectBox.Width / 2))
    {
      this.xDir = -(this.xDir);
    }
    if (nextLocation.Y >= (objectContainer.Height - objectBox.Height / 2))
    {
      this.yDir = -(this.yDir);
    }

    if (nextLocation.X - (objectBox.Width / 2) <= 0)
    {
      this.xDir = -(this.xDir);
    }
   if (nextLocation.Y - (objectBox.Height / 2) <= 0)
    {
      this.yDir = -(this.yDir);
    }
```

```
        if ((nextLocation.X <= (objectContainer.Width - objectBox.Width / 2)) &&
(nextLocation.Y <= (objectContainer.Height - objectBox.Height / 2)))
        {
            LocPoint = nextLocation;
        }
    }
    public void AnimateObject()
    {
        nextPosition();
        this.objectBox.SuspendLayout();
        this.objectBox.Left = LocPoint.X - (objectBox.Width / 2);
        this.objectBox.Top = LocPoint.Y - (objectBox.Height / 2);
        this.objectBox.ResumeLayout();
        this.objectBox.Invalidate();
    }
    public void resetAnimationObject()
    {
        int x = this.objectContainer.Width / 2;
        int y = this.objectContainer.Height / 2;
        LocPoint = new Point(x, y);
        this.xDir = +1;
```

```
        this.yDir = +1;
        this.xOffset = 10;
        this.yOffset = 10;
        this.objectBox.SuspendLayout();
        this.objectBox.Left = LocPoint.X - (objectBox.Width / 2);
        this.objectBox.Top = LocPoint.Y - (objectBox.Height / 2);
        this.objectBox.ResumeLayout();
        this.objectBox.Invalidate();
    }
public void miss()
    {
        currentNote = new string[] { "" };
        DateTime t1 = DateTime.Now;
        this.objectBox.Image = Properties.Resources.Aqua_Ball_icon2Break;
        System.Windows.Forms.Application.DoEvents();
        SoundPlayer sp = new SoundPlayer();
        string breakFile = System.Windows.Forms.Application.ExecutablePath;
        breakFile = breakFile.Substring(0, breakFile.LastIndexOf("\\"));
        breakFile = breakFile + "\\sound\\Breaking.wav";
        sp.SoundLocation = breakFile;
        sp.Play();
```

```
        TimeSpan diff;
        diff = DateTime.Now.Subtract(t1);
        while (diff.Seconds <= 2)
        {
            System.Windows.Forms.Application.DoEvents();
            diff = DateTime.Now.Subtract(t1);
        }
        resetNotePosition();
        resetNotes();
        this.objectBox.Image = Properties.Resources.Aqua_Ball_icon3;
        missHit = false;
    }
private void InitializeComponent()
    {
        this.SuspendLayout();
        //
        // MainInterface
        //
        this.ClientSize = new System.Drawing.Size(284, 262);
        this.Name = "MainInterface";
        this.Load += new System.EventHandler(this.MainInterface_Load);
```

```
        this.ResumeLayout(false);
}
}
}
```

5.3. Main Form Background Code

The code consists of the declaration of notes in Arrays and the methods that are called at different points of program execution. Keys are handled that accept user-inputted stroke hits from the keyboard. This code also contains the random control of dhuns to be input by user and the scrolling feature of the dhun on the orange-colored background strip which displays the current note to be played.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using eTabla.Service;
```

```
using System.Diagnostics;
using System.IO;
namespace eTabla
{
    public partial class MainInterface : Form
    {
        #region Variabled
        private Dictionary<string, string> cmdSoundMap;
        private Dictionary<string, string> cmdButtonMap;
        private Dictionary<string, string> keyCmdMap;
        private EventHandler clickHandler;
        private KeyPressEventHandler keyHandler;
        private Model.Tabla tabla;
        private bool playState;
        private string[][] allNotes;
        private int currentNoteIndex;
        private string[][] bool1;
        private string[][] bool2;
        private string[][] bool3;
        private string[][] bool4;
        int keyStatus = 0, activeBool = 0;
```

```
int timeoutBool2 = 3000, timeoutBool3 = 2000, timeoutBool4 = 1000;//Set the
timeout for different bools in milliseconds
#endregion
#region Constructor
public MainInterface()
{
    InitializeComponent();
    this.InitInterface();
    this.InitMotion();
    cmdSoundMap = new Dictionary<string, string>();
    keyCmdMap = new Dictionary<string, string>();
    cmdButtonMap = new Dictionary<string, string>();
    this.tabla = new Model.Tabla();
    this.bool1 = new string[][] { new string[] { "Te" }, new string[] { "Tae" }, new
string[] { "Ta" }, new string[] { "Dha" }, new string[] { "Tun" }, new string[] { "Na" },
new string[] { "Ko" }, new string[] { "Ti" }, new string[] { "Dhi" }, new string[] { "Ge" },
new string[] { "Dhin" } };
    this.bool2 = new string[][] { new string[] { "Dha" }, new string[] { "Ti", "Tae" },
new string[] { "Ko", "Tae" }, new string[] { "Ta", "Ko" }, new string[] { "Tun" }, new
string[] { "Ti", "Tae" }, new string[] { "Ko", "Tae" }, new string[] { "Ta", "Ko" }, new
string[] { "Tun" }, new string[] { "Ti", "Tae" }, new string[] { "Ko", "Tae" }, new string[] {
```

```
"Ta", "Ko" }, new string[] { "Dha", "Ge" }, new string[] { "Ti", "Tae" }, new string[] {
"Ko", "Tae" }, new string[] { "Ta", "Ko" } };

        this.bool3 = new string[][] { new string[] { "Dha" }, new string[] { "Dhin" }, new
string[] { "Dha" }, new string[] { "Na", "Ko" }, new string[] { "Tun" }, new string[] { "Na"
} };

        this.bool4 = new string[][] { new string[] { "Dha" }, new string[] { "Dhin" }, new
string[] { "Dha" }, new string[] { "Na", "Ko" }, new string[] { "Tun" }, new string[] { "Na"
} };

        this.allNotes = this.bool1;

    }

    #endregion

    #region OnLoad

    protected override void OnLoad(EventArgs e)

    {

        base.OnLoad(e);

       // get settings from file if not create settings files

        cmdSoundMap.Add("cmdK1", "Te");

        cmdSoundMap.Add("cmdK2", "Tae");

        cmdSoundMap.Add("cmdK3", "Ta");

        cmdSoundMap.Add("cmdK4", "Dha");

        cmdSoundMap.Add("cmdK5", "Tun");

        cmdSoundMap.Add("cmdK6", "Na");
```

```
cmdSoundMap.Add("cmdK7", "Ko");
cmdSoundMap.Add("cmdK8", "Ti");
cmdSoundMap.Add("cmdK9", "Dhi");
cmdSoundMap.Add("cmdK10", "Ge");
cmdSoundMap.Add("cmdK11", "Dhin");
keyCmdMap.Add("A", "cmdK1");
keyCmdMap.Add("S", "cmdK2");
keyCmdMap.Add("D", "cmdK3");
keyCmdMap.Add("F", "cmdK4");
keyCmdMap.Add("G", "cmdK5");
keyCmdMap.Add("H", "cmdK6");
keyCmdMap.Add("J", "cmdK7");
keyCmdMap.Add("K", "cmdK8");
keyCmdMap.Add("L", "cmdK9");
keyCmdMap.Add(";", "cmdK10");
keyCmdMap.Add("'", "cmdK11");
// Used to change the button color
cmdButtonMap.Add("Te", "cmdK1");
cmdButtonMap.Add("Tae", "cmdK2");
cmdButtonMap.Add("Ta", "cmdK3");
cmdButtonMap.Add("Dha", "cmdK4");
```

```
cmdButtonMap.Add("Tun", "cmdK5");
cmdButtonMap.Add("Na", "cmdK6");
cmdButtonMap.Add("Ko", "cmdK7");
cmdButtonMap.Add("Ti", "cmdK8");
cmdButtonMap.Add("Dhi", "cmdK9");
cmdButtonMap.Add("Ge", "cmdK10");
cmdButtonMap.Add("Dhin", "cmdK11");
this.currentNote = this.allNotes[0];
this.KeyPress += new KeyPressEventHandler(MainInterface_KeyPress);
clickHandler = new EventHandler(clickHand);
keyHandler = new KeyPressEventHandler(keyHand);
this.cmdK1.Click += clickHand;
this.cmdK1.KeyPress += keyHand;
this.cmdK2.Click += clickHand;
this.cmdK2.KeyPress += keyHand;
this.cmdK3.Click += clickHand;
this.cmdK3.KeyPress += keyHand;
this.cmdK4.Click += clickHand;
this.cmdK4.KeyPress += keyHand;
this.cmdK5.Click += clickHand;
this.cmdK5.KeyPress += keyHand;
```

```
        this.cmdK6.Click += clickHand;
        this.cmdK6.KeyPress += keyHand;
        this.cmdK7.Click += clickHand;
        this.cmdK7.KeyPress += keyHand;
        this.cmdK8.Click += clickHand;
        this.cmdK8.KeyPress += keyHand;
        this.cmdK9.Click += clickHand;
        this.cmdK9.KeyPress += keyHand;
        this.cmdK10.Click += clickHand;
        this.cmdK10.KeyPress += keyHand;
        this.cmdK11.Click += clickHand;
        this.cmdK11.KeyPress += keyHand;
        this.playState = false;
        this.cmdPlayStop.Click += new EventHandler(cmdPlayStop_Click);
        this.scrollTimer.Tick += new EventHandler(scrollTimer_Tick);
        this.motionTimer.Tick += new EventHandler(motionTimer_Tick);
}
    #endregion
    void motionTimer_Tick(object sender, EventArgs e)
    {
        this.AnimateObject();
```

```
}
void scrollTimer_Tick(object sender, EventArgs e)
{
   this.animate();
}
#region Play
void cmdPlayStop_Click(object sender, EventArgs e)
{
   if (this.playState)
   {
      this.playState = false;
      this.cmdPlayStop.Image = Properties.Resources.play;
      this.scrollTimer.Enabled = false;
      this.motionTimer.Enabled = false;
      if (activeBool == 2 || activeBool == 3 || activeBool == 4)
      {
         stopTimeoutTimer();//If already running stop the timeout section
         keyStatus = 0;
         timeOutTimer.Interval = 10;
         timeOutTimer.Start();
      }
```

```
            else
            {
                stopTimeoutTimer();
            }
        }
        else
        {
            this.playState = true;
            this.cmdPlayStop.Image = Properties.Resources.stop;
            this.scrollTimer.Enabled = true;
            this.motionTimer.Enabled = true;
            if (activeBool == 2 || activeBool == 3 || activeBool == 4)//stop the timeout timer
if bool is 2 or 3
                timeOutTimer.Stop();
        }
        Application.DoEvents();
        highlightCurrentBool();
    }
    #endregion
    #region Key press and Click events
    void keyHand(object sender, KeyPressEventArgs e)
```

```
{
    MainInterface_KeyPress(sender, e);
}
void clickHand(object sender, EventArgs e)
{
    try
    {
        if (!missHit)
        {
            Button btn = sender as Button;
            if (btn != null)
            {
                string name = btn.Name;
                string note = cmdSoundMap[name];
                if (note == this.currentNote[this.currentNoteIndex])
                {
                    Service.ITablaManager tManager = new Service.TablaManager();
                    tManager.playNote(note, this.tabla);
                    if (this.currentNoteIndex == this.currentNote.Length - 1)
                    {
                        this.currentNoteIndex = 0;
```

```
                setNextNote(this.allNotes);
            }
            else
            {
                this.currentNoteIndex += 1;
            }
            if (activeBool == 2 || activeBool == 3 || activeBool == 4)
            {
                stopTimeoutTimer();
                //start the timeout timer to reset the time to 3 seconds
                keyStatus = 1;//keyStatus is changed to note the right key is pressed
within time
                timeOutTimer.Interval = 10;
                timeOutTimer.Start();
            }
            highlightCurrentBool();
        }
        else
        {
            this.currentNote = this.allNotes[0];
            this.currentNoteIndex = 0;
```

```
            if (activeBool == 2 || activeBool == 3 || activeBool == 4)
               stopTimeoutTimer();
            // missHit = true;
            miss();
          }
        }
      }
    }
}
void MainInterface_KeyPress(object sender, KeyPressEventArgs e)
{
   try
   {
     string keyPress = e.KeyChar.ToString().ToUpper();
      Button btn;
      if (keyCmdMap.Keys.Contains(keyPress))
      {
         btn = new Button();
         btn.Name = keyCmdMap[keyPress];
         this.clickHand(btn, null);
      }
```

```
        }
    }
    #endregion
    #region Button Back Color
    private void highlightCurrentBool()
    {
      try
      {
        string bollText = currentNote[currentNoteIndex];
        string buttonName = this.cmdButtonMap[bollText];//.Controls.Find(btnName,
true);
        resetButtonBackColor();
        Control[] buttons = this.Controls.Find(buttonName, true);
        Button currentbtn = (Button)buttons[0];
        currentbtn.BackColor = Color.Orange;
      }
    }
    private void resetButtonBackColor()
    {
      try
      {
```

```
        cmdK1.BackColor = button1.BackColor;
        cmdK2.BackColor = button1.BackColor;
        cmdK3.BackColor = button1.BackColor;
        cmdK4.BackColor = button1.BackColor;
        cmdK5.BackColor = button1.BackColor;
        cmdK6.BackColor = button1.BackColor;
        cmdK7.BackColor = button1.BackColor;
        cmdK8.BackColor = button1.BackColor;
        cmdK9.BackColor = button1.BackColor;
        cmdK10.BackColor = button1.BackColor;
        cmdK11.BackColor = button1.BackColor;
    }
}
#endregion
#region Timeout between 2 clicks
private void stopTimeoutTimer()
{
    try
    {
        resetButtonBackColor();
        timeOutTimer.Stop();
```

```
    }
}
private void timeOutTimer_Tick(object sender, EventArgs e)
{
    try
    {
        if (activeBool == 2)
            timeOutTimer.Interval = timeoutBool2;
        else if (activeBool == 3)
            timeOutTimer.Interval = timeoutBool3;
        else if (activeBool == 4)
            timeOutTimer.Interval = timeoutBool4;
        if (keyStatus == 0)
        {
            resetButtonBackColor();
            this.currentNote = this.allNotes[0];
            this.currentNoteIndex = 0;
            stopTimeoutTimer();

            miss();
        }
```

```
        else
            keyStatus = 0;
    }
}
#endregion
private void level1ToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.allNotes = this.bool1;
    this.currentNote = this.allNotes[0];
    activeBool = 1;
}
private void level2ToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.allNotes = this.bool2;
    this.currentNote = this.allNotes[0];
    activeBool = 2;
}
private void level3ToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.allNotes = this.bool3;
    this.currentNote = this.allNotes[0];
```

```
    activeBool = 3;
}
private void level4ToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.allNotes = this.bool4;
    this.currentNote = this.allNotes[0];
    activeBool = 4;
}
private void level2ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    TablaManager tab = new TablaManager();
    tab.playDemo(2);
}
private void level3ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    TablaManager tab = new TablaManager();
    tab.playDemo(3);
}
private void level4ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    TablaManager tab = new TablaManager();
```

```
        tab.playDemo(4);
    }
    private void playManualToolStripMenuItem_Click(object sender, EventArgs e)
    {
        UserManual manual = new UserManual();
        manual.Show();
    }
    }
}
```

5.4. Tabla.cs

This file contains the setter and getter methods to be called during execution of the program at a particular dhun encountered and returns the value of the file to be played or stopped.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace eTabla.Model
{
```

```
public class Tabla
{
  private string basePath = System.Windows.Forms.Application.ExecutablePath;
   private string te;
   public string Te
   {
      get { return te; }
      set { te = value; }
   }
   private string tae;
   public string Tae
   {
      get { return tae; }
      set { tae = value; }
   }
   private string level2;
   public string Level2
   {
      get { return level2; }
      set { level2 = value; }
   }
```

```
private string ta;
public string Ta
{
   get { return ta; }
   set { ta = value; }
}
private string dha;
public string Dha
{
   get { return dha; }
   set { dha = value; }
}
private string tun;
public string Tun
{
   get { return tun; }
   set { tun = value; }
}
private string na;
public string Na
{
```

```
    get { return na; }
    set { na = value; }
}
private string ko;
public string Ko
{
    get { return ko; }
    set { ko = value; }
}
private string ti;
public string Ti
{
    get { return ti; }
    set { ti = value; }
}
private string dhi;
public string Dhi
{
    get { return dhi; }
    set { dhi = value; }
}
```

```
private string dhin;


public string Dhin
{
   get { return dhin; }
   set { dhin = value; }
}
private string ge;
public string Ge
{
   get { return ge; }
   set { ge = value; }
}
public Tabla()
{
   this.basePath = this.basePath.Substring(0, (this.basePath.LastIndexOf("\\")));
   this.basePath = this.basePath + "\\sound\\";
   this.te = this.basePath + "Te.wav";
   this.tae = this.basePath + "Tae.wav";
   this.ta = this.basePath + "Ta.wav";
   this.dha = this.basePath + "Dha.wav";
```

```
        this.tun = this.basePath + "Tun.wav";
        this.na = this.basePath + "Na.wav";
        this.ko = this.basePath + "Ko.wav";
        this.ti = this.basePath + "Ti.wav";
        this.dhi = this.basePath + "Dhi.wav";
        this.ge = this.basePath + "Ge.wav";
        this.dhin = this.basePath + "Dhin.wav";
    }
  }
}
```

5.5. Function Description

Table 1 describes class, function and their uses in eTablaTutor. Here I tried to explain little more details about the coding structure.

Table 1. Class, Function and Use

Number	Class Name	Function	Use
1	Tabla Class		This class is used to get/set different sounds of the tabla used in the application.
2	TablaManager Class		This class is used to play the table sounds.

Table 1. (Continued)

Number	Class Name	Function	Use
		public void playNote(string note,Model.Tabla tabla)	This function plays the bool for the corresponding key press or mouse click.
3	MainInterfaceMotion Class		This class is used for different animations in the application.
		public void InitMotion()	This initializes the animation object and area.
		private void nextPosition()	This function calculates the next position of the animation.
		public void AnimateObject()	This function is used to animate the animation object.
		public void resetAnimationObject()	This function resets the animation object to the start position once it is completed or when there is an error and error animation is played.
		public void miss()	This function is used to play an error animation to notify the user that he made a mistake in playing the bool.
4	MainInterface Class		This class is used to implement the eTabla logic.
			Controls the active bool.
			Checks the key press and mouse click.

Table 1. (Continued)

Number	Class Name	Function	Use
			Resets the bools on an error.
5	eTabla Logic Class	void cmdPlayStop_Click(object sender, EventArgs e)	This function starts by playing the selected bool and stops the bool being played.
		void MainInterface_KeyPress(object sender, KeyPressEventArgs e)	This function handles the keypress event and checks whether the user pressed the right key. If the user pressed the right key, the next bool is displayed for play; on the other hand, in case of error, an error animation is palyed, and the bool is started again.
		private void highlightCurrentBool()	This function changes the color of the button mapped for the currently active bool so that the user should have some visual hint about what he is going to play next.
		private void resetButtonBackColor()	This function resets the color of all the buttons when the user makes a mistake in playing the bool.

Table 1. (Continued)

Number	Class Name	Function	Use
		private void timeOutTimer_Tick(object sender, EventArgs e)	This timer event is used to check timeout for Levels 2, 3, and 4. If the user has not pressed the expected key within the timeframe, an error animation is played.

5.6. Testing

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design, and coding. Testing requires that the developer discard preconceived notions about the "correctness" of the software just developed and overcome a conflict of interest that occurs when errors are uncovered.

"Testing is a process of executing a program with the intent of finding an error. Testing is also one element of a broader topic often referred as Validation & Verification (12)".

Exhaustive testing of almost any non-trivial is impractical due to the fact that the domain of input data values for most practical software systems is either extremely large or infinite. Therefore, I must design an *optimal test suite* is of reasonable size so that I can uncover as many errors in the system as possible. Actually, if test cases are selected randomly, many of the selected test cases do not contribute to the *significance* of the test

suite; i.e., they do not detect any additional defects not already being found by other test cases in the suite. This implies that the test suite should be carefully designed rather than picked randomly. Therefore, a systematic approach should be followed to design an optimal test suite. There are essentially two main approaches to systematic testing:

- White-box testing
- Black-box testing

5.6.1. White-box Testing

Sometimes, this type of testing is also called glass-box testing. "This is a test case design method that uses the control structure of procedural design to derive the test (11)". Using white-box testing guarantees that

- All independent paths within a module have been exercised at least once
- All logical decision were exercised to their true and false side
- All tests were executed to their boundaries within their operational bounds
- Internal data structure was exercised to ensure their validity

The following approaches to designing white-box test cases are used:

- Statement Coverage
- Branch Coverage
- Conditional Coverage
- Path Coverage

White-box tests used with eTablaTutor were as follows:

- Control flow check between class files.
- Control flow checks between modules of sound playing, ball movement, etc.
- Event-handling priority checks between switching buttons with different strokes.
- Checks for capturing strokes from the keyboard.

5.6.2. Black-box Testing

In the black-box approach, test cases are designed using only the functional specification of software, i.e., without any knowledge about the internal structure of the software. For this reason, black-box testing is also known as ***functional testing***. Test cases require a thorough knowledge of internal structure, so the testing method is also called ***structural testing***. "In black-box testing, test cases are designed from an examination of the input/output values only, and no knowledge of design or code is required". It finds errors for the following problems:

- Incorrect or missing function
- Interface errors
- Errors in data structure or external database access
- Performance errors
- Initialization and termination errors

Black-box tests involved during the creation of eTablaTutor were as follows:

- Unhandled exception checks in class parameter passing.

- Checks applied for the interaction of the interface with the music files.
- Test cases generated for Null pointer Exception checks.
- Method overloading checks for similar class files with the same names.
- Interface loading error handling.
- Control flow of pointer during the continuation of a game from one event to another when clicking on different buttons.

A software product goes through three levels of testing. The testing levels are outlined in the following sections.

5.6.2.1. Unit Testing

Unit testing (or module testing) tests different units (or modules) of a system in isolation. Unit testing is undertaken when a module has been coded and successfully reviewed. "It is always a good idea to first test the module in isolation before integration because it makes debugging easier. Unit testing makes heavy use of white-box testing techniques (12)". The unit testing for eTablaTutor was done with the following modules:

- Motion timer and scroll timer-tick checks.
- Dhun play event capture checks.
- Load test of main interface design.
- Suspension of playing a note after a certain time period.
- Resetting of notes.
- Random calling of notes.

5.6.2.2. Integration Testing

"The primary objective of integration testing is to test the module interfaces in order to ensure that there are no errors in the parameter passing when one module invokes another module". During integration testing, different modules of a system are integrated in a planned manner using an integration plan. The integration plan specifies the steps and the order in which modules are combined to realize the full system. After each integration step, the partially integrated system is tested. The integration testing was done with these modules:

- Coordinate the working of the motion-timer and scroll-timer ticks
- Simultaneous working for the motion of the ball and the scrolling of exact dhuns, highlighting buttons to be clicked at that event
- Error-control test when a mismatch is found with the change of ball display to a burning ball

5.6.2.3. System Testing

System tests are designed to validate a fully developed system to assure that it meets its requirement. There are essentially three main kinds of system testing:

- Alpha Testing: Alpha testing refers to the system testing carried out by the test team within the developing organization

- Beta Testing: Beta testing refers to the system testing performed by a select group of friendly customers
- Acceptance Testing: Acceptance testing is the system testing performed by the customer to determine whether to accept or reject the delivery of the system

6. FUTURE SCOPE

6.1. Future Scope of the eTablaTutor

This eTablaTutor can be modified in many different ways. There are three things I would like to implement in my eTablaTutor in future.

6.1.1. Model Tabla Structure for the Future

Figure 11 is showing that in the future, this game will have a model Tabla made with plastic or something. I do not need any button for Dhin or Dhi because **Dhin** = Dha + Tun; **Dhi** = Dha + Ti

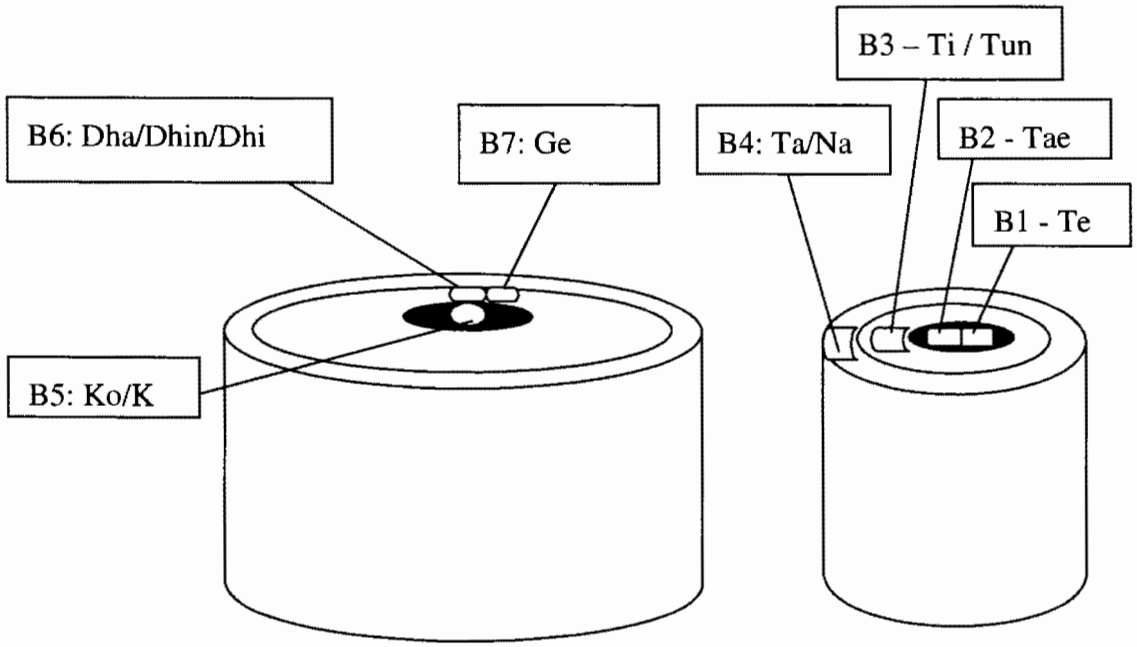

Figure 11. Model Tabla for Future eTablaTutor

The Dayan (right-hand drum) will have two buttons in the Gab (center): one button for "Te" and one button for "Tae." It will be exactly in the same position according to the real Tabla. The Dayan will also have one button on the Kinar (corner) to play "Ta" and "Na" (same sound) and one button on the Sur (between Gab and Kinar) to play "Ti" (with press stroke) and "Tin" (with light stoke). The Bayan (left-hand drum) will have one button which is a little wide on Gab to play "Ko," one button on the Maidan to play "Ge," and one button to play "Dha" and "Dhin.".

Level 5:

It will have few more options for a little advance level. For example,

Moderate Level (playing Taal with simple songs), Listen Bols –by professionals,

Listen Taal –by professionals, Test, Test results, Suggestion

Level 6:

It will have an advanced level of Tabla practice. For example,

Practice Taal (with a variety of songs)

At least a hundred songs for each Taal will be stored in a database. The user will search by Taal, and the results will be shown by songs (maybe instrumental or vocal). In the background, a query will be run; it will retrieve the songs that match a particular Taal. Then, the list of songs will appear on the screen, and the user can click the song (or any song) and will be able to listen it.

6.1.2. Make More Easy, Small Levels

There is always an option here to create many small and easy levels for students so that they can keep their interest in the Tabla, continuing their learning and having more fun. In this way, the game will be more popular and helpful for students. This system will reduce the face-to-face teacher-student learning time.

6.1.3. Real Tabla Display in the Interface

The eTablaTutor interface will display a Baya top-head on the left side of the interface and a Daya top-head on the right side of the interface. In the future when students hit the notations in that proposed model Tabla, those Baya and Daya top-heads will highlight the real Tabla area along with the correct finger display.

REFERENCES

[1] Hit Songs Science. U Playa. No Dates. Retrieved on 17 Dec 2007 <http://uplaya.com/>

[2] A. Kapur, G. Essl, P Davidson and P. R. Cook. "Deo's maa digital art project site." ETabla Controller. 12 March 2008. Retrieved on 29 July 2010 <http://deodesign.wordpress.com/2008/03/12/etabla-conrtoller/> and <http://www.cs.princeton.edu/sound/research/controllers/etabla/Petthesis.pdf>.

[3] Steve Larson. "Searching for meaning: Melodic patterns, Combinations, and Embellishments". No dates. Retrieved on 20 August 2009 http://ciir.cs.umass.edu/music2000/papers/invites/larson_invite.pdf

[4] Mihir Sarkar. "TablaNet: a Real-Time Online Musical Collaboration System for Indian Percussion". Master's Thesis, MIT Media Lab, Aug. 2007.

[5] Acoustic World. "Tabla Deba". iTunes Previews. February 2010. Retrieved on 29 August 2010. <http://itunes.apple.com/us/app/tabla-deva/id350516434?mt=8>

[6] Prasad Upasani. iTablaPro. No dates. Retrieved on 30 August 2010. <http://www.upasani.org/home/iTablaPro.html>

[7] Parag Chordia. Ladada Highlights. 11 September 2010. Retrieved on 20 September 2010. <http://paragchordia.com/>

[8] Choppin Broccoli Inc. "Pocket Bhangra for iPhone". No Dates. Retrieved on 30 August 2010. <http://itunes.apple.com/us/app/pocket-bhangra-dhol-tumbi/id383057121?mt=8#>

[9] Sean Mikhail. "Tabla Master". No Dates. Retrieved on 04 September 2010. <http://itunes.apple.com/us/app/tabla-master/id385413671?mt=8>

[10] Mihir Sarkar and Barry Vercoe. "Recognition and Prediction in a Network Music Performance System for Indian Percussion"; International Conference on New Interfaces for Musical Expression (NIME 2007), New York, NY, USA, June 6-9, 2007.

[11] Rajib Mall. Fundamental of Software Engineering. Second Edition. New Delhi: Prentice Hall publications, 2004

[12] D.S.Yadav. Foundations of Information Technology. Third Edition. New Delhi. New Age International (p) limited publications. 2006

[13] Pencil sketch of Soumyashree Mohapatra. TAIS'ZINE. March 2001. Volume 12, issue 1. Retrieved on 20 June 2010. <http://www.awn.com/tais/TAIS_Zinegal_mar01.html>

[14] MSDN Library. No Dates. Retrieved on 19 August 2009. <http://msdn.microsoft.com/en-us/library >