

System Design Practices
ANDROID BASED MARIO GAME

B.Tech. C.E. Semester –VII

Prepared By:

Krutarth Patel (CE 97)

Harsh Vyas (CE 140)

Guided By:

Prof. M.S.Bhatt



Department of Computer Engineering

Faculty of Technology

Dharmsinh Desai University, Nadiad

DHARMSINH DESAI UNIVERSITY

NADIAD-387001, GUJARAT



CERTIFICATE

This is to certify that the project carried out in the subject of Software Design Practice, entitled “**Android Based Mario Game**” and presented in this report is a bona fide report of work done by **Krutarth Patel** and **Harsh Vyas** of Department of Computer Engineering, semester VII.

Prof. M. S. Bhatt,
Project guide
Computer Engineering
Department

Date:

Prof. C. K. Bhensdadia,
Head
Computer Engineering
Department

Date:

CONTENTS

1. Acknowledgement & Abstract.....	4,5
2. Introduction.....	6
2.1 Problem Details.....	6
2.2 Purpose.....	6
2.3 Scope.....	6
2.4 Platform.....	6
2.5 Technology Used	6
2.6 Tools Used.....	6
2.7 Software Development Model.....	7
2.8 User Characteristics.....	7
3. Software Requirement Specification.....	8
3.1 Function Requirement.....	8
3.2 Nonfunctional Requirement	11
3.3 Gantt Chart & Pert Chart	12
4. System Design.....	14
4.1 Use Case Diagram.....	14
4.2 Class Diagram	15
4.3 Sequence Diagram.....	16
4.4 Activity Diagram	17
4.5 Start Diagram	18
4.6 Collaboration Diagram.....	19
4.7 Structure Chart.....	20
5. Implementation.....	21
5.1 Modules.....	21
6. Testing.....	22
6.1 Testing Plan.....	22
6.2 Testing Strategy	23
6.3 Test Cases.....	24
7. Screenshots.....	25
8. Conclusion.....	32
9. Limitation and Future Enhancement.....	33
10. Bibliography.....	34

1 ACKNOWLEDGEMENT

It gives us immense pleasure and satisfaction in presenting this report of System design practices undertaken during the 7th semester of B.Tech. From the bottom of our heart, we would like to express my sincere thanks to our Head of Department **Prof. C. K. Bhensdadia** and our project guide **Prof. M. S. Bhatt**, who gave us an opportunity to undertake such a great challenging and innovative work, without whose help and encouragement, it would be unfeasible for us to carry out the desired work. We are grateful to them for their guidance, encouragement, understanding and insightful support in the development process. Finally, we would like to thank to all faculty members of our college, our friends and our family members, who have directly or indirectly provided their support and continuous encouragement throughout the project.

With sincere regards,

Krutarth Patel & Harsh Vyas.

ABSTRACT

“Mario Game” is a 2D single-player puzzle/platform game on the Android platform in which the player’s primary form of movement like jump, walk which are relies upon the touch on different area of screen. The player will progress in given level which he/she has to clear all enemies, though the game encourages creativity and daring via branching pathways, also it creates interest, liveliness in user. In game user has to face different enemies to complete the game, user has benefits of energetic impact, and user can get points according to his/her performance. At last user can see his/her score.

2 Introduction

2.1 Problem Details:

“Mario Game” provides an easy way through an Android application to play the game for enjoyment and refreshment.

2.2 Purpose:

“Mario Game” is a game that allows users to play the game for enjoyment. “Mario Game” is a 2D single-player puzzle/platform game on the Android platform in which the player’s primary form of movement relies upon the touch on different area of screen. The player will progress in given level which he/she has to clear all enemies, though the game encourages creativity and daring via branching pathways.

2.3 Scope:

We describe what features are in the scope of the software and what are not, for the software to be developed.

In Scope:

- Mario can walk, jump in forward or backward direction.
- Mario can take coins if user wishes and get points.
- Mario can eat ice-cream and gets energy.
- Mario die if it is small and collide with enemy.

Not in Scope:

- Mario cannot fly.
- We cannot pause the game.
- Mario cannot shoot.

2.4 Platform:

Android
Windows

2.5 Technology Used:

Android

2.6 Tools used:

Eclipse IDE
ADT

2.7 Software Development Model:

Iterative Waterfall Model

2.8 User Characteristics:

- Android application can be used with basic knowledge of operating Android based device
- Users should be comfortable with the English language.

3 Software Requirement Specification

3.1 Functional Requirements

R1. Mario Management

R1.1 Move

R1.1.1 Move Forward

Input : User touches the right lower part of the screen.

Output : Mario walks in the forward direction.

Processing : Mario walk sprite forward direction come into execution.

R1.1.2 Move Backward

Input : User touches the left lower part of the screen.

Output : Mario walks in the backward direction.

Processing : Mario walk sprite backward direction come into execution.

R1.2 Jump

R1.2.1 Jump Forward

Input : User touches the upper right part of the screen.

Output : Mario jumps in the forward direction.

Processing : Mario jump sprite forward direction come into execution.

R1.2.1 Jump Backward

Input : User touches the upper left part of the screen.

Output : Mario jumps in the backward direction.

Processing : Mario jump sprite backward direction come into execution.

R2 Moving Background

Input : User touches the screen's forward or backward direction parts lower or upper parts.

Output : Background moves accordingly in the backward direction.

R3 Hurdle impact

R3.1 Energetic impact

Input : Mario eats ice-cream.

Output : Mario gets energy and grows up.

R3.2 Death

Input : Mario collides with the villains moving towards him in his way.

Output : Mario gets destroyed.

R3.3 Hazardous impact

Input : Mario collides with the unnecessary obstacles moving towards him in his way.

Output : Mario loses his energy or gets smaller.

R4 Stimulus/Response Sequences

Step 1: The player launches the game from their portable device.

Step 2: The start screen loads and appears, prompting the player with three buttons: "Play", "Instructions", "Quit", and "About us".

Step 3: The player presses one of the buttons, triggering its respective function.

R4.1: The title screen must load and appear every time the game is launched with background music continuously playing.

R4.2: If the player quits the game during any stage of a level, they must be returned to the title screen.

R4.3: If the player presses the exit button, the game will end and return the player to the phone's regular interface.

R4.4 If the player completes the game, the game will end and return the player to the title screen.

R5 Sound

R5.1 Move Sound

Input : User touches the right lower part of the screen.

Output : Mario walks in the forward direction and beap sound generated.

R5.2 Jump sound

Input : User touches the upper part of the screen.

Output : Mario jumps on the floor and jump sound is generated.

3.2 Nonfunctional Requirement

1. Performance:-

- Our product needs to load in no more than 3 seconds and there should be no latency or lag during game play. Any latency in game play would make this product less entertaining and frustrating for users.

2. Safety Requirements:-

- Although the product should have no effect on data loss or mobile device damage, it is a concern in the early beta stages of the product. The end product should not damage physically or internally in any manner.

3. Security Requirements :-

- The end product will not consist of security or privacy issues because personal data will not be sent or retrieved.

4. Reliability:-

- Reliability needs to be implemented in our project because we don't want our product to crash or to lag.

5. Availability

- We need our product to be available to our users and they need to be able to use our product in the manner it was intended.

6. Maintainability

- Our product needs to be maintainable in case we should need to change in data.

7. Portability

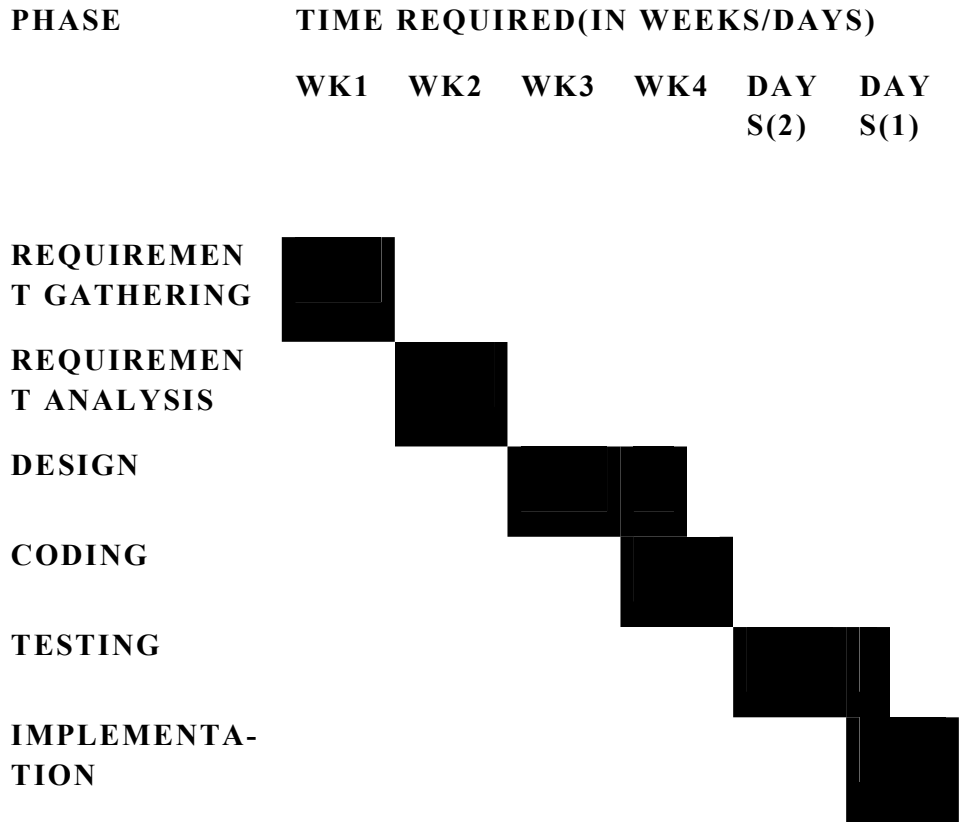
- Being a mobile device application, our product is already portable.

8. Re-usability

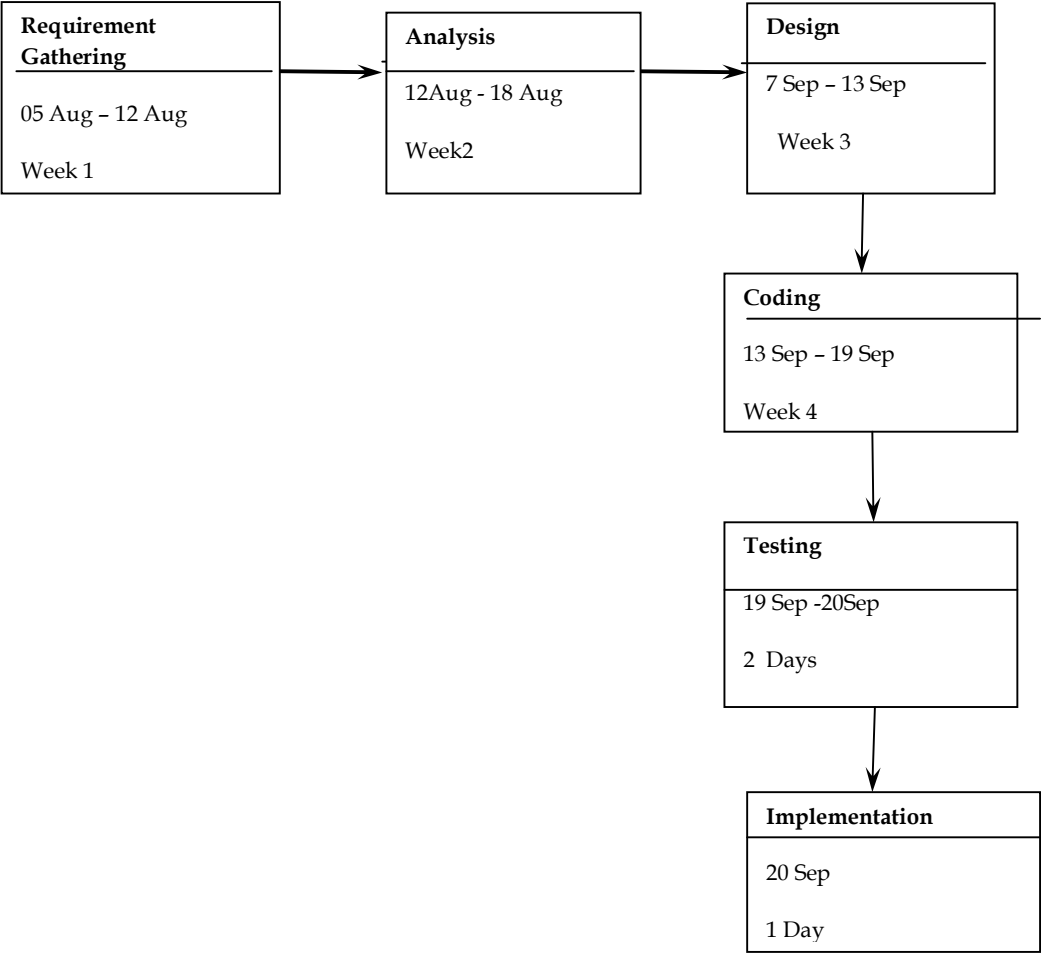
- Being a mobile device application, our product is already reusable.

3.3 Gantt Chart and Pert Chart

Gantt Chart:

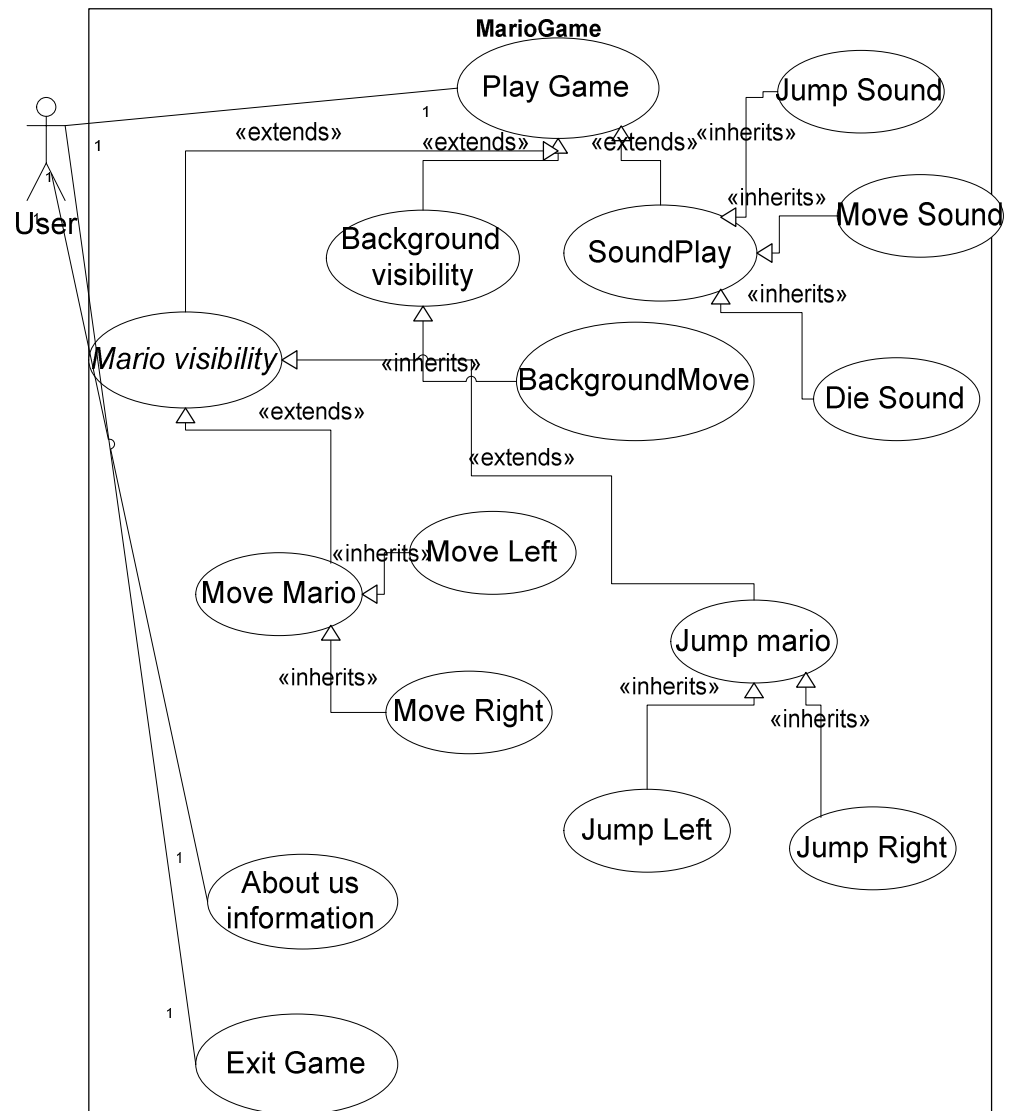


PERT Chart

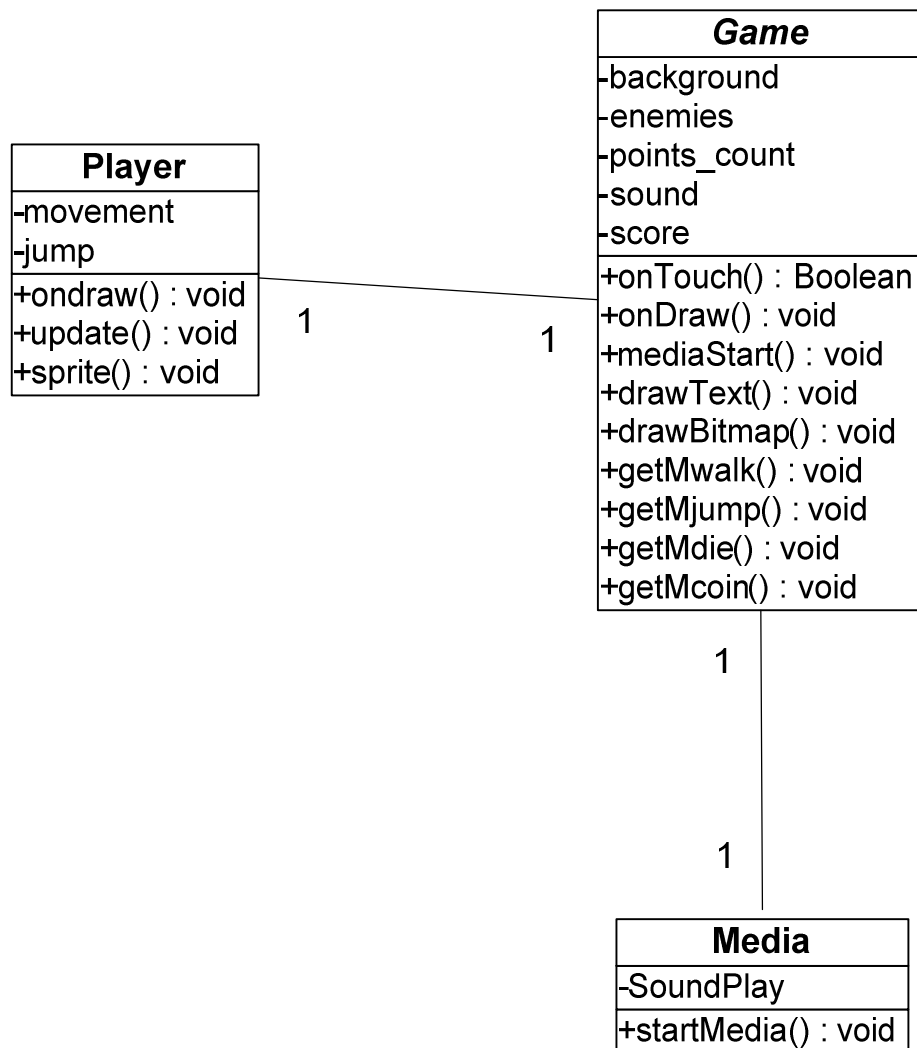


4 System Design

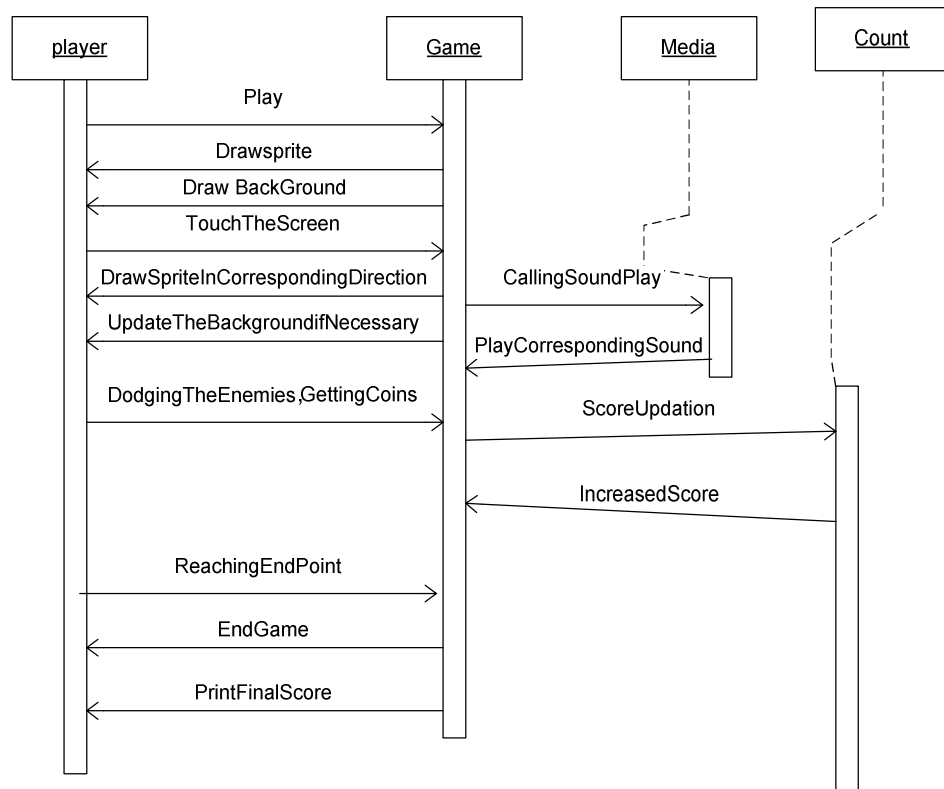
4.1 Use case Diagram:-



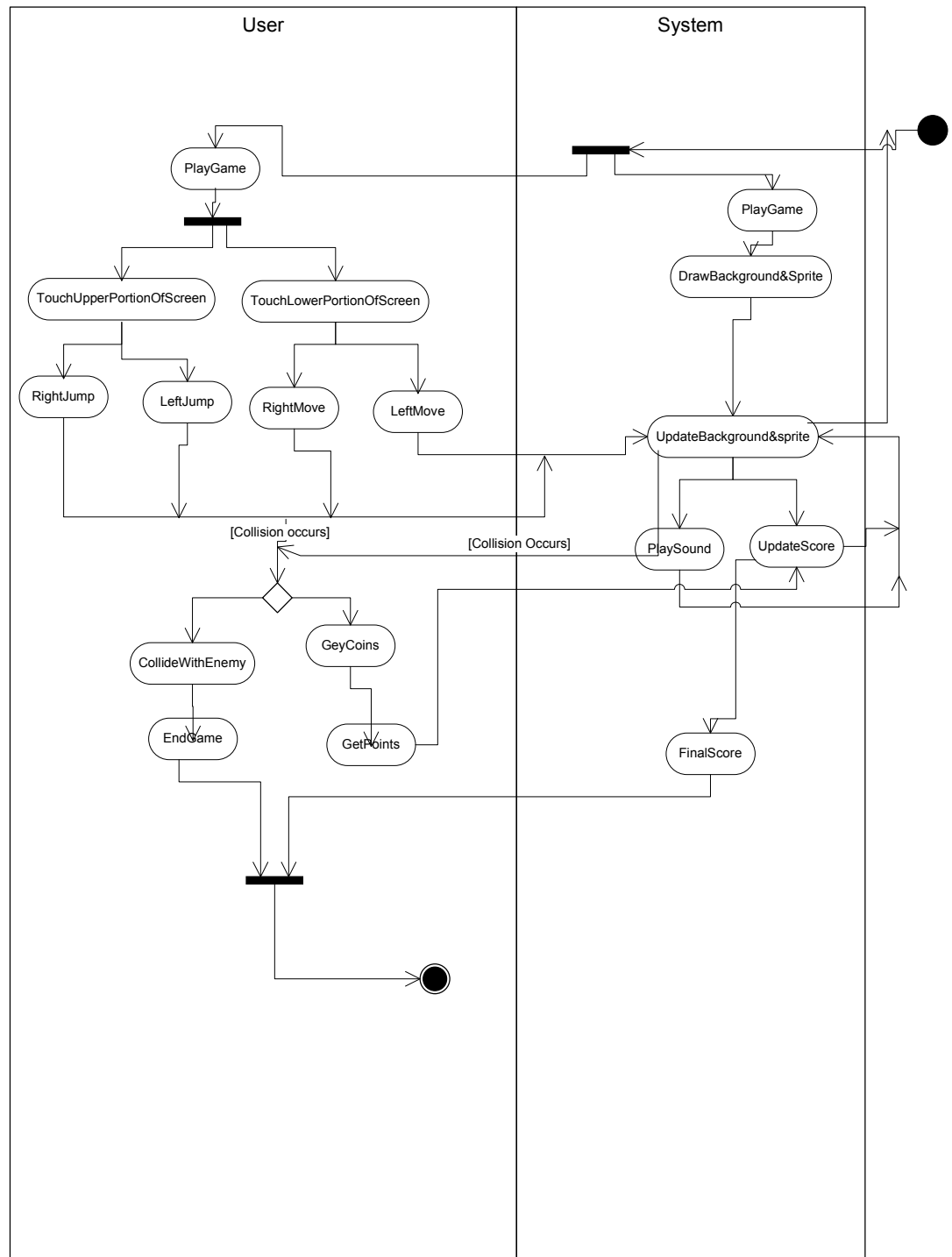
4.2 Class Diagram:-



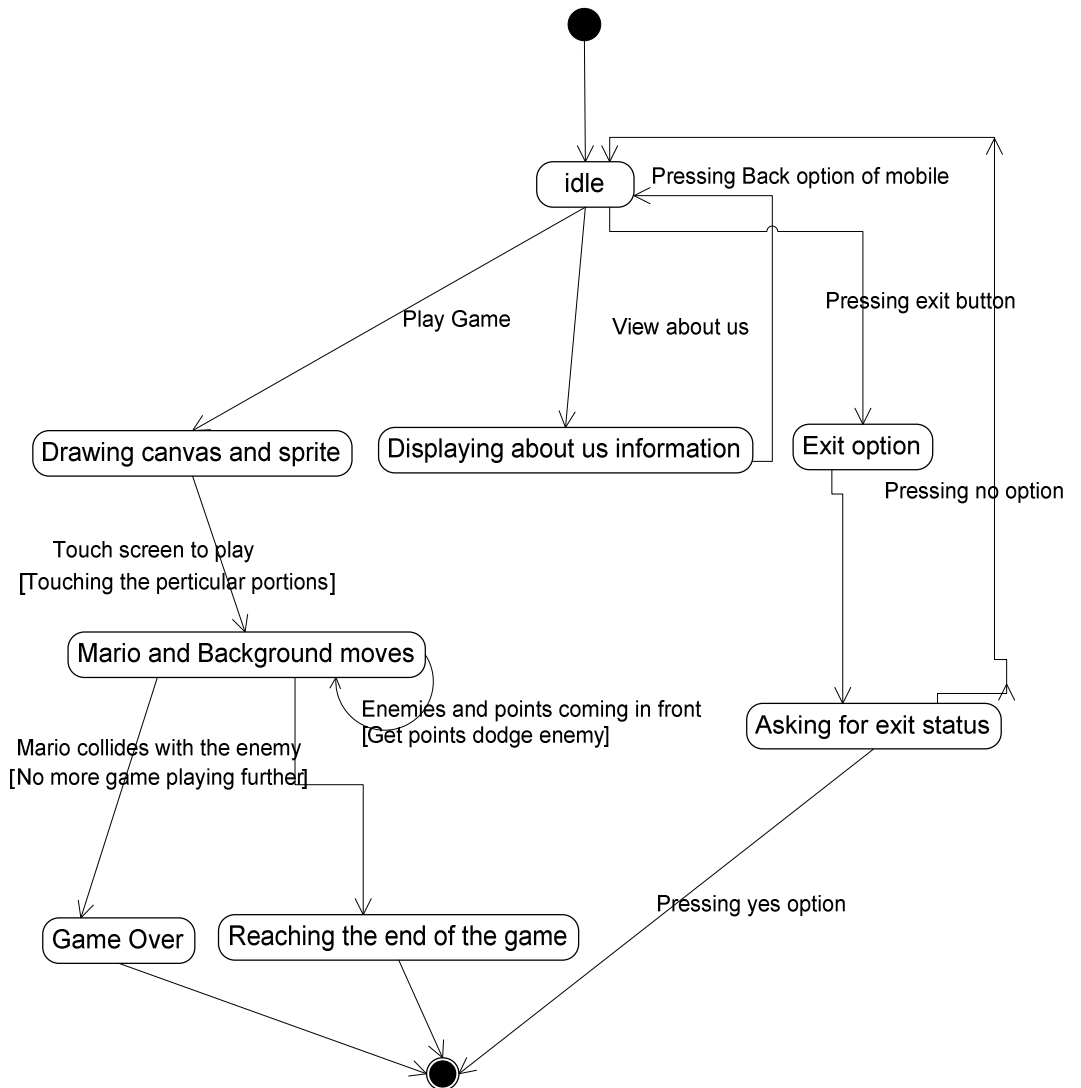
4.3 Sequence Diagram:-



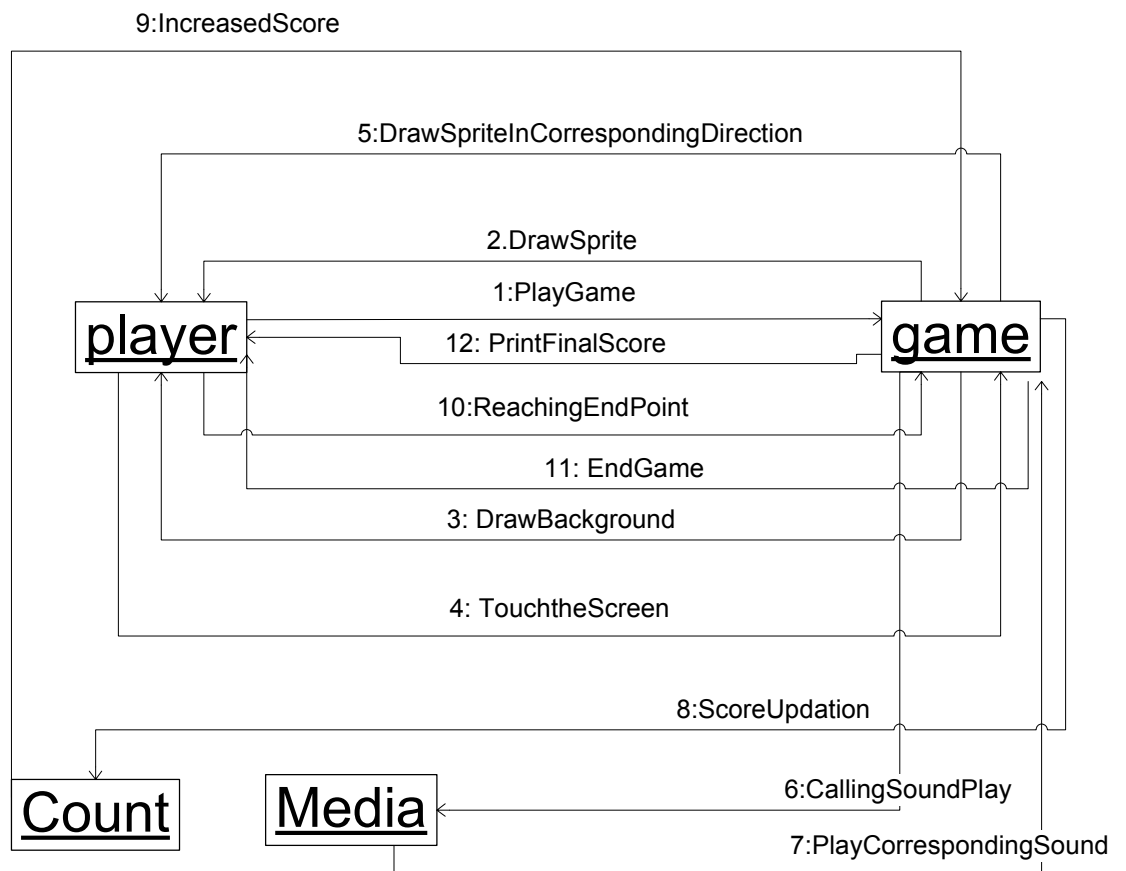
4.4 Activity Diagram:-



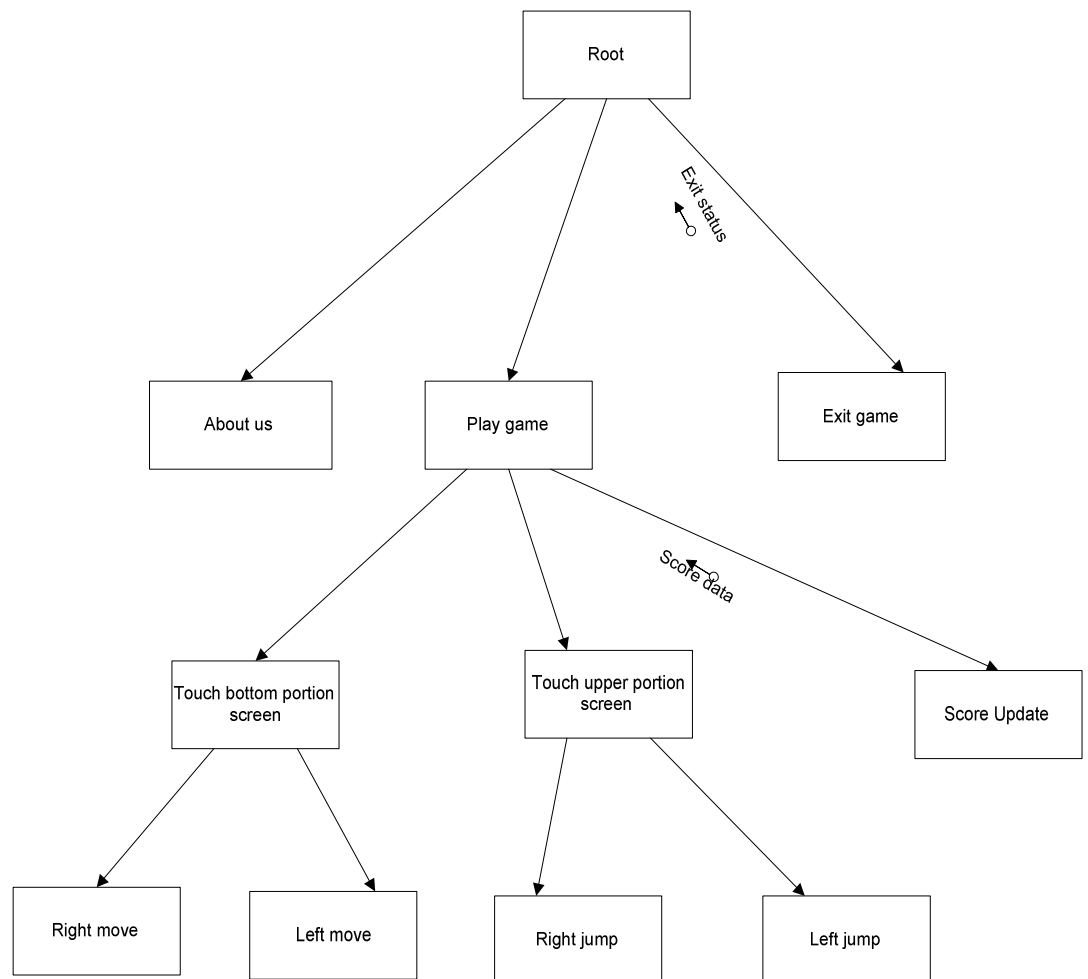
4.5 State Diagram:-



4.6 Collaboration Diagram:-



4.7 Structure Chart :-



5 Implementation

Package	Class	Method	Description
android.app.Activity	Activity	onCreate()	Do work on creation time of Activity
		setContentView()	Set content of screen.
android.media.MediaPlayer	MediaPlayer	create()	Get object of media file.
		start()	Start media file.
		setLooping()	Set true/false for looping media file.
android.view.View.*	OnClickListener	onClick()	Do activities on click of button.
android.view.SurfaceView	SurfaceView	onDraw()	Draw on given canvas.
		getWidth()	Get width of surface.
		getHeight()	Get height of surface.
		surfaceCreated()	Define activities on creation of surface.
		surfaceDestroyed()	Define activities on destroy of surface.
android.graphics.Canvas	Canvas	getWidth()	Get width of canvas.
		getHeight()	Get height of canvas.
		drawBitmap()	Draw specified bitmap at specified location.
java.lang.Thread	Thread	run()	Run the specific thread.

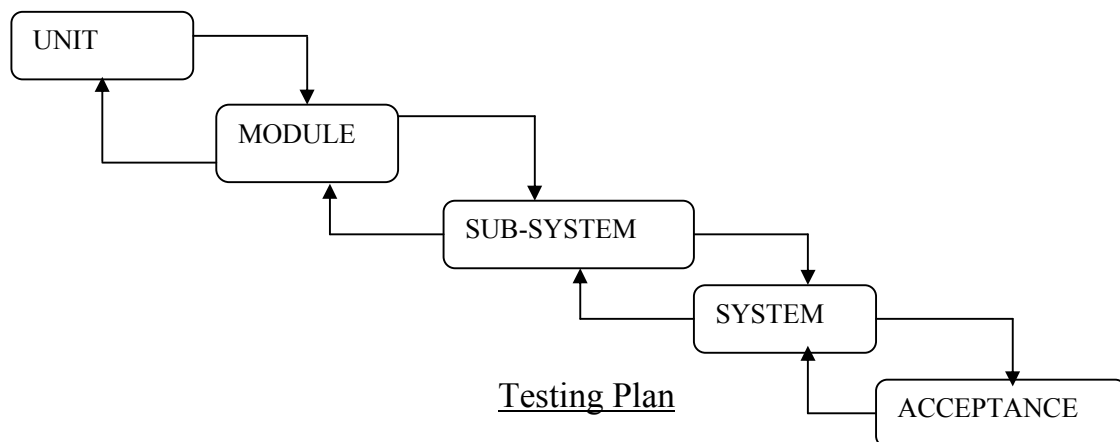
6 Testing

Testing is the process carried out on software to detect the differences between its behavior and the desired behavior as stipulated by the requirements specifications. Testing is advantageous in several ways. Firstly, the defects found help in the spot corrected; testing gives an idea as to how reliable the software is. Secondly, over time, the record of defects found reveals the most common kinds of defects, which can be used for developing appropriate preventive measures such as training, proper design and reviewing.

6.1 Testing Plans

The testing sub-process includes the following activities in a phase dependent manner:

- a) Create Test Plans.
- b) Create Test Specifications.
- c) Review Test Plans and Test Specifications.
- d) Conduct tests according to the Test Specifications, and log the defects.
- e) Fix defects, if any.
- f) When defects are fixed continue from activity.



6.2 Testing Strategy

The development process repeats this testing sub-process a number of times for the following phases.

- a) Unit Testing.
- b) Integration Testing

Unit Testing tests a unit of code (module or program) after coding of that unit is completed. Integration Testing tests whether the various programs that make up a system, interface with each other as desired, fit together and whether the interfaces between the programs are correct. System Testing ensures that the system meets its stated design specifications. Acceptance Testing is testing by the users to ascertain whether the system developed is a correct implementation of the Software Requirements Specification.

Testing is carried out in such a hierarchical manner to ensure that each component is correct and the assembly/combination of components is correct. Merely testing a whole system at the end would most likely throw up errors in components that would be very costly to trace and fix.

We have performed both Unit Testing and System Testing to detect and fix errors. A brief description of both is given below.

Unit Testing

Objective:

The objective of Unit Testing is to test a unit of code (program or set of programs) using the Unit Test Specifications, after coding is completed. Since the testing will depend on the completeness and correctness of test specifications, it is important to subject these to quality and verification reviews.

Input: Unit Test Specifications

Testing Process:

- Checking for availability of Code Walk-through reports which have documented the existence of and conformance to coding standards.
- Review of Unit Test Specifications.

Verify the Unit Test Specifications confirm to the program specifications.

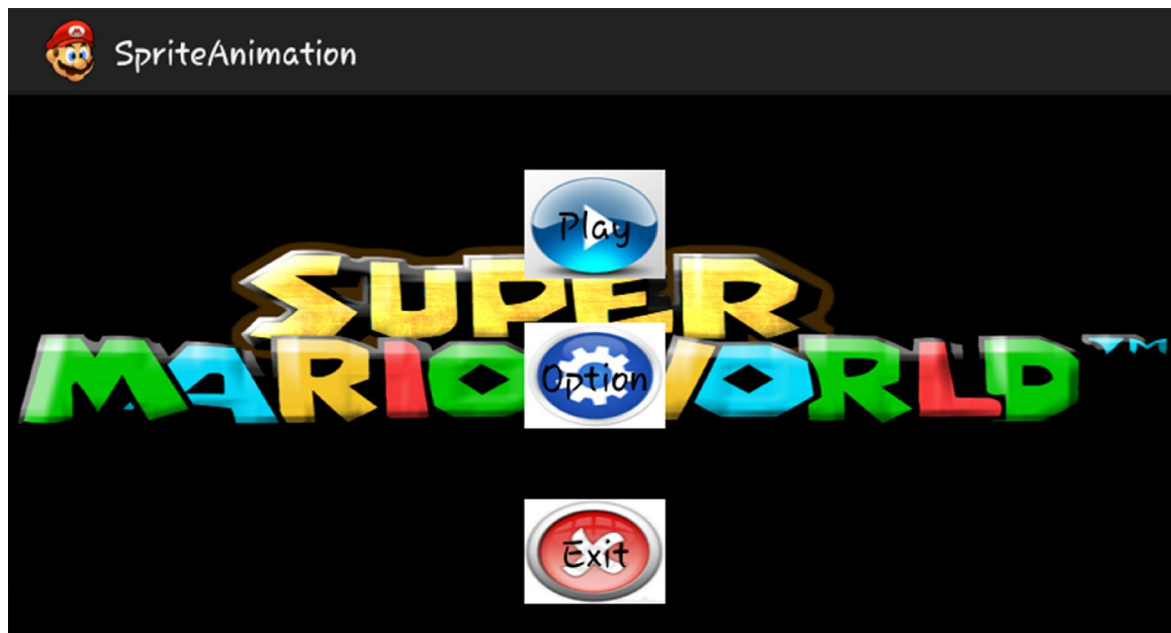
Verify that all boundary and null data conditions are included.

6.3 Test Cases

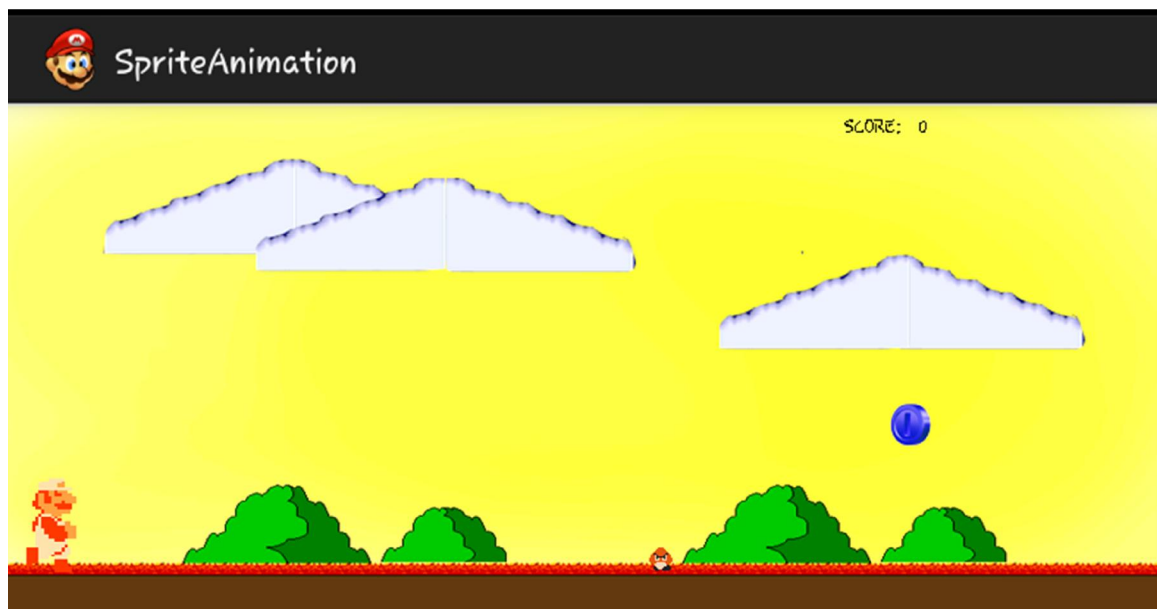
Test Case No.	Module	Functionality	Expected Result	Actual Result	Pass/Fail
1.	Mario Management	Move Forward	Successful Movement	Successful Movement	Pass
		Move Backward	Successful Movement	Successful Movement	Pass
		Jump Forward	Successful Jump	Successful Jump	Pass
		Jump Backward	Successful Jump	Successful Jump	Pass
2.	Moving Background	Background Moves	Background Moves As Mario Moves Forward	Successful Background Move	Pass
3	Hurdle Impact	Energetic Impact	Mario Eats Ice cream And gets Bigger in size if small	Successful Implementation	Pass
		Death	Mario deaths if collides with enemies if small in size	Successful Implementation	Pass
		Hazardous Impact	Looses energy if Mario is bigger in size otherwise Death	Successful Implementation	Pass
4.	Sound	Move Sound	If Mario moves then sound generated	Successful Sound generated	Pass
		Death Sound	If Mario dies then sound generated	Successful Sound generated	Pass
		Jump Sound	If Mario jumps then sound generated	Successful Sound generated	Pass

7 Screenshots

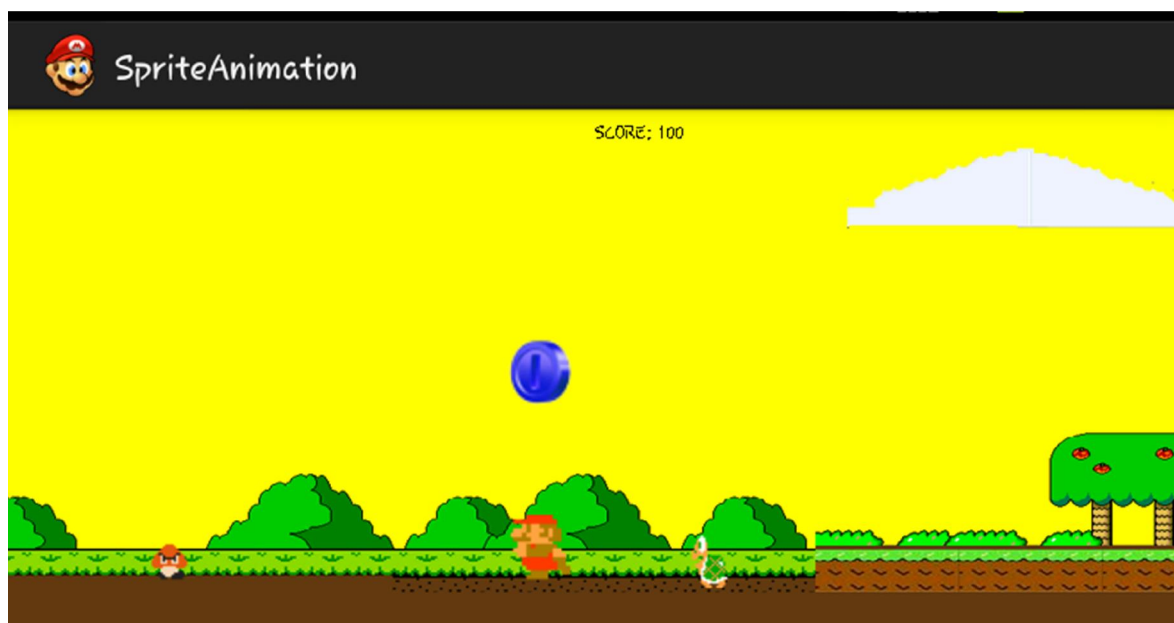
7.1 Initial Screen:



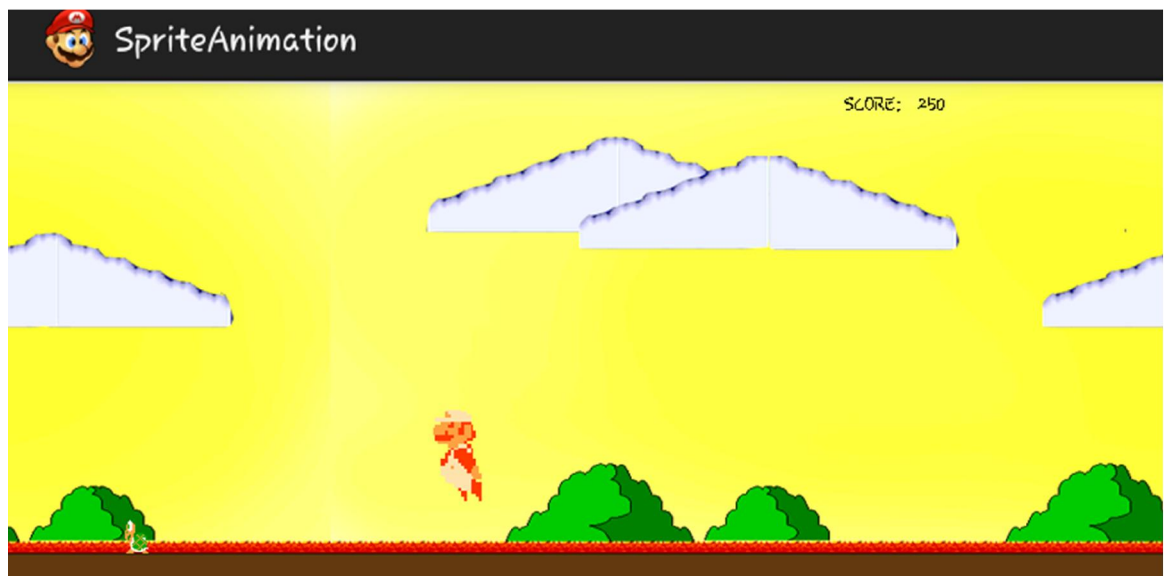
7.2 Press Play:



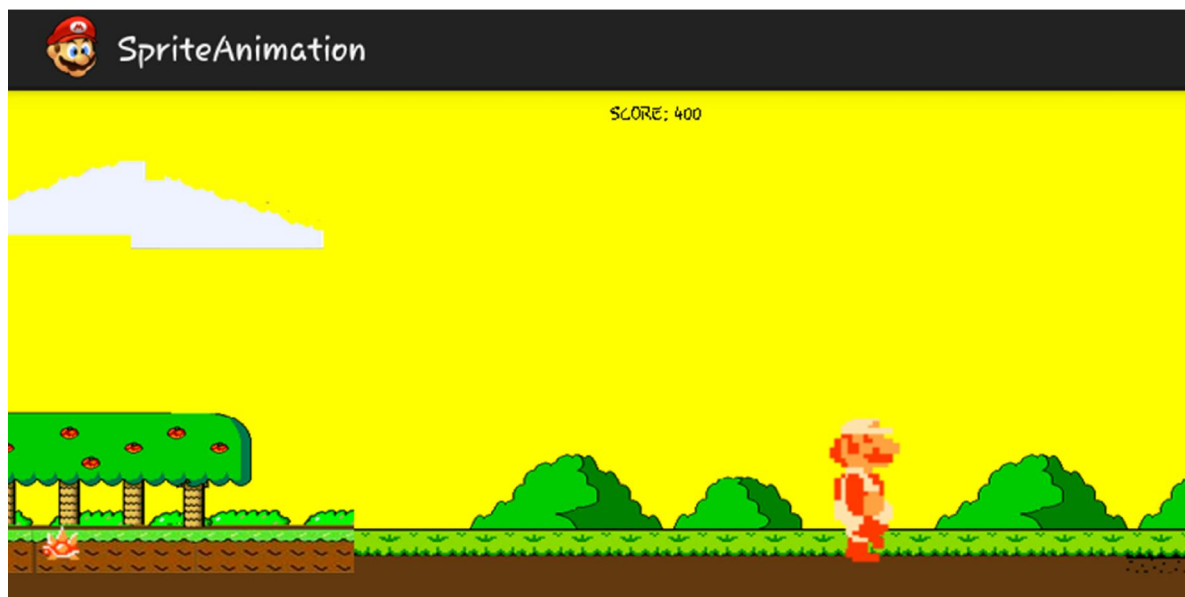
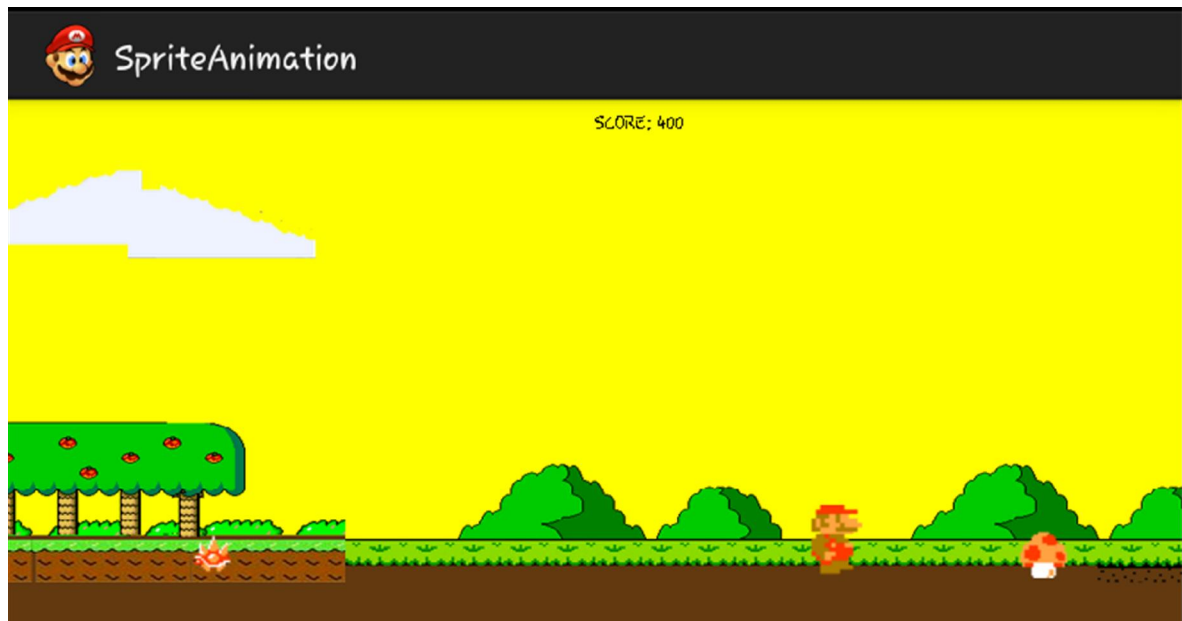
7.3 Move Mario:



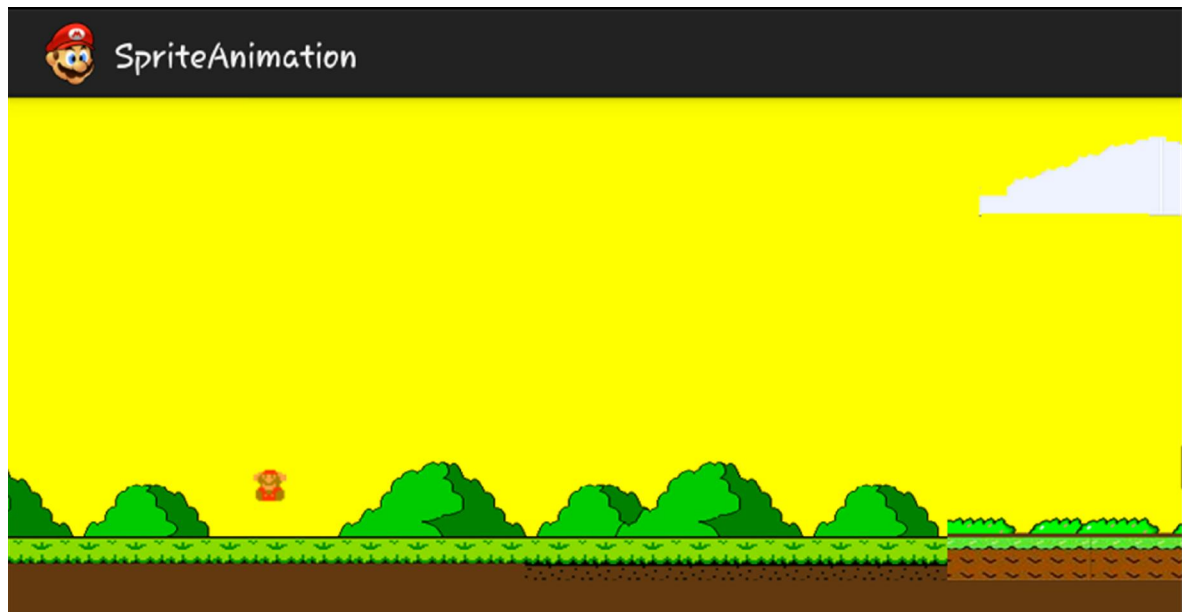
7.4 Jump Mario:



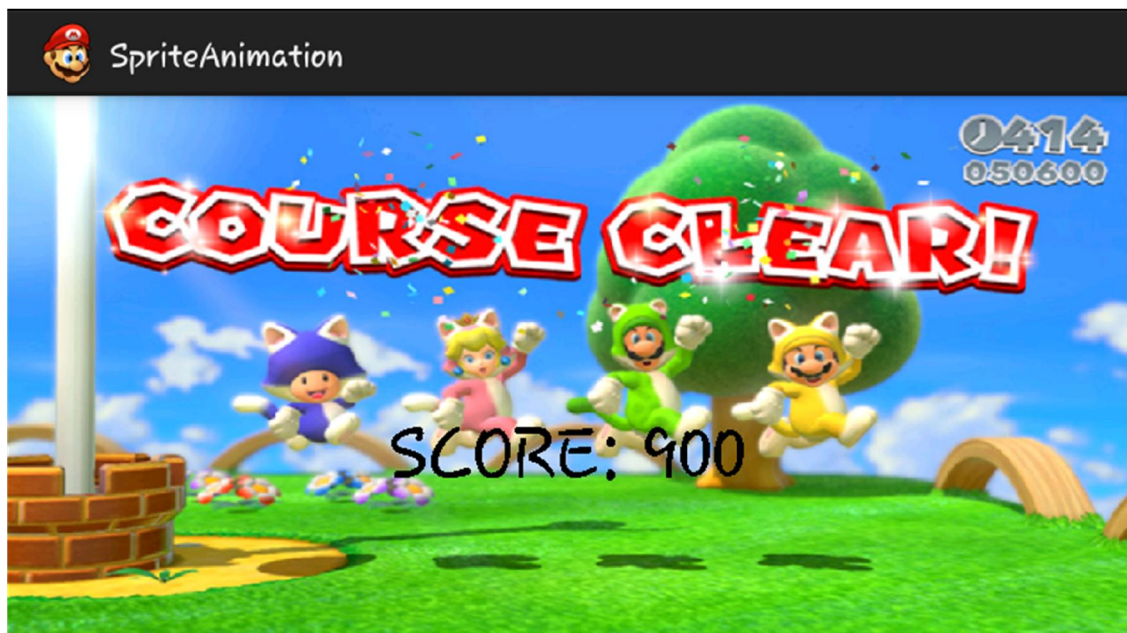
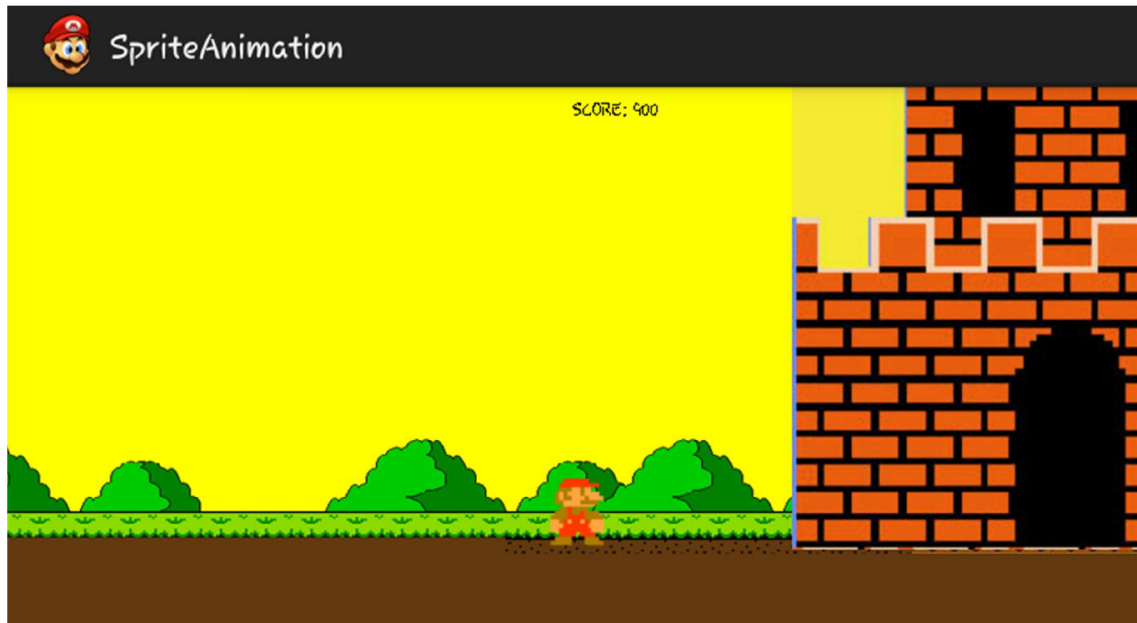
7.5 Energetic Impact:



7.6 Die:




7.7 Stage Clear:



7.8 Other Functionalities:

Press Options Button:

 **SpriteAnimation**


How to play:


You must play this game in the touch screen mobile and mobile should be facilitated with multiple touch.

R1. Mario Management

- Move:** **Move Forward** : User touches the right lower part of the screen and mario walks in the forward direction.
- Move Backward**: User touches the left lower part of the screen and mario walks in the backward direction.
- Jump:** **Jump Forward**: User touches the upper right part of the screen and mario jumps in the forward direction.
- Jump Backward**: User touches the upper left part of the screen and mario jumps in the backward direction.
- Moving Background**: User touches the screen's forward or backward direction parts lower or upper parts and Background moves accordingly in the backward direction.
- Hurdle impact**: Energetic impact and mario eats ice-cream and mario gets energy and grows up.
- Death**: Mario collides with the villains moving towards him in his way. Mario gets destroyed.
- Hazardous impact**: Mario collides with the unnecessary obstacles moving towards him in his way and mario loses his energy or gets smaller.

Press Exit Button:


 **SpriteAnimation**



Are you sure you want to close this app ??

No

Yes



8 Conclusion

As discussed earlier we had developed a user friendly attractive GUI based android game which is most recently played. The game includes mario's forward backward movements, mario's jump movements with attractive sound functionality. Game's graphics are more attractive to the users. The game provides a perfect package for the game users to play the basics of any game. This game provide a better quality and save time in real life. This software is developed to provide much better quality of service. But, no software can be termed as 'Perfect' in the real sense and there always remains scope of further improvement & so that helps to develop new version.

Functionality of application:-

- Read information about game.
- Play Game
- Exit Game

9 Limitation and Future Enhancement

Limitation:-

- There is no time limitation for complete the stage.
- We cannot pause the game after start.
- Only one stage of game is available.

Future Enhancement:-

- Include Time Limit.
- Include pause option.
- Add new stages in game.

10 Bibliography

Websites

- www.stackoverflow.com
- www.lynda.com
- www.thenewboston.org
- www.edu4java.com
- www.wildbunny.co.uk
- www.spritters-resource.com
- www.developer.android.com