Asian Journal of Research in Computer Science

4(3): 1-8, 2019; Article no.AJRCOS.53088
ISSN: 2581-8260

Implementation of Java Based Racing Game, Pirate Race, Using Runnable Interface

K. Logiraj[1*]

[1]Department of Mathematics, Eastern University, Sri Lanka.

Author's contribution

The sole author designed, analysed, interpreted and prepared the manuscript.

Article Information

DOI: 10.9734/AJRCOS/2019/v4i330115
Editor(s):
(1) Dr. Young Lee, Associate Professor, Department of Electrical Engineering and Computer Science, Frank H. Dotterweich College of Engineering, Texas A&M University-Kingsville, USA.
(2) Dr. G. Sudheer, Professor, Department of Mathematics and Computer Science, G.V.P College of Engineering for Women, Madhurawada, India.
(3) Dr. Shivanand S. Gornale, Professor, Department of Computer Science, School of Mathematics and Computing Sciences, Rani Channamma University, Vidyasangam, NH - 4, Belagavi, India.
Reviewers:
(1) Osondu Oguike, University of Nigeria, Nigeria.
(2) Mohammed Farik, The University of Fiji, Fiji.
(3) Yulia, Petra Christian University, Indonesia.
Complete Peer review History: http://www.sdiarticle4.com/review-history/53088

Original Research Article

Received 04 October 2019
Accepted 11 December 2019
Published 19 December 2019

ABSTRACT

Nowadays Java has become the most popular programming language that has been designed to develop desktop applications that run on Java virtual machine (JVM) regardless of computer architecture. Particularly, Java is useful for developing Game Applications. Implementing these applications is an effective educational way to encourage Java Learners. This paper aims to develop a desktop racing game to motivate freshmen who self-identity as creative or who wants to implement their own 2D racing games rather than a prescribed activity. It brings fun and simplicity of the game 'Pirate Race' with new features. This game was developed with the Runnable Interface by extending the JFrame and the movement of the objects in the frame was controlled by keyboard events. 'Pirate Race', is a simple game application that targets Java Learners to understand how usable classes and interfaces can be handled in a relatively short time. The application presents a graphical user interface with 2D graphical images having different file types (GIF, PNG, JPG) and with a background sound. The application allows the user to move the ship by pressing the (up, down, left, right) keys of the keyboard. The user's goal is to compete with the other two pirate ships and finish the race with the first rank. When racing with opponents, the user must take correct

**Corresponding author: E-mail: logirajk@gmail.com;*

moves to prevent the ship from crashing with ice mountain that resides in the sea. The game contains simple controls that can be easily caught by children. Therefore, it is suitable for players of the ages three and up. This will be a challenging and interesting game for children who likes to play computer games.

Keywords: Java virtual machine; racing game; Jframe; game application; graphical user interface.

1. INTRODUCTION

Java is the most popular programming language in the computational world. Java was developed by James Gosling and Patrick Naughton at Sun Microsystems Inc. in 1991, later acquired by Oracle Corporation [1]. It's a simple programming language, while it's easy to write, compile and debug a program in Java. It allows the user to develop modular programs and reusable codes. There are more than three billion devices that run Java. It is used to build Mobile Applications, Desktop Applications, Web Applications, and Games and so on.

Runnable is an interface that is used to execute the instances of a class using a thread [2]. The java.lang package contains the interface which provides a standard set of rules for the instances of classes. A run method should be defined inside the class in order to create a thread [3]. This method with void as return type and it takes in no arguments. A separate flow of control in network programming is represented using these runnable classes. Also, it is used to perform multi-thread programming.

Nowadays people, especially children are interested in playing games on their smart devices such as desktop computers, smartphones, and tablets. It is quite difficult for newcomers to handle the classes and interfaces in developing Java Game applications. There has been substantial research work that focuses on handling the components of game systems that can imitate human game playing styles [4]. Microsoft developed a game development tool, Kodu Game Lab, which helps engage students in learning programming through making and playing games [5]. As a result, many racing games developed with various gameplay that are unique to the game itself. Hydro Thunder Hurricane is an arcade-style boat racing game and the sequel to Hydro Thunder. The primary mode of the gameplay involves a sixteen competitor race to the finish line [6]. In order to win the game, the player has to finish at first place. Therefore, the Player must make continuous adjustments to its ship's direction as opposed to simply driving in a straight line and turning only when an obstacle in the sea approaches. Racing games are the most interesting types of game for personal computer, which allows to entertain and attract children. 'Pirate Race' is a challenging game between user pirate ship and the other two competing pirate ships. The opponent ship movements are already defined in the program according to the pixel value of the coordinate in the frame.

As said above, the proposed application is a desktop computer game, targeting children and adults who wish to play racing games and making correct moves to win. The intention is also to popularize this racing game as a desktop application, because of its easy for players to control movements using a keyboard.

2. PIRATE RACE

Pirate Race is a racing game for desktop computers designed and implemented using Runnable Interfaces. Users can install the installation file of the game on their local computers and run it. The game consists of three pirate ships, include a user ship and two competing ships. The user ship movements are controlled by the keyboard events and the other two pirate ship movements are controlled automatically by the program. The user ship must be aware of the ice mountain in the sea during the race. If the ice mountain is on the user ship's path, the player must use the correct move to pass the ice mountain to avoid crashing on it. Meanwhile, if the user ship uses another direction to move the ship, the speed will be reduced and the ship will be delayed. Depending on the distance, the rank will be calculated for the three pirate ships. The ranking will be displayed on the screen when racing. If the user's ship crashed with an ice mountain using a wrong move, the game will be over.

There is a starting line and an end line in the game. The goal of the user ship is to finish the end line safely in a short time period. At the same time, if the user can not complete the target line on time, the user can replay the game.

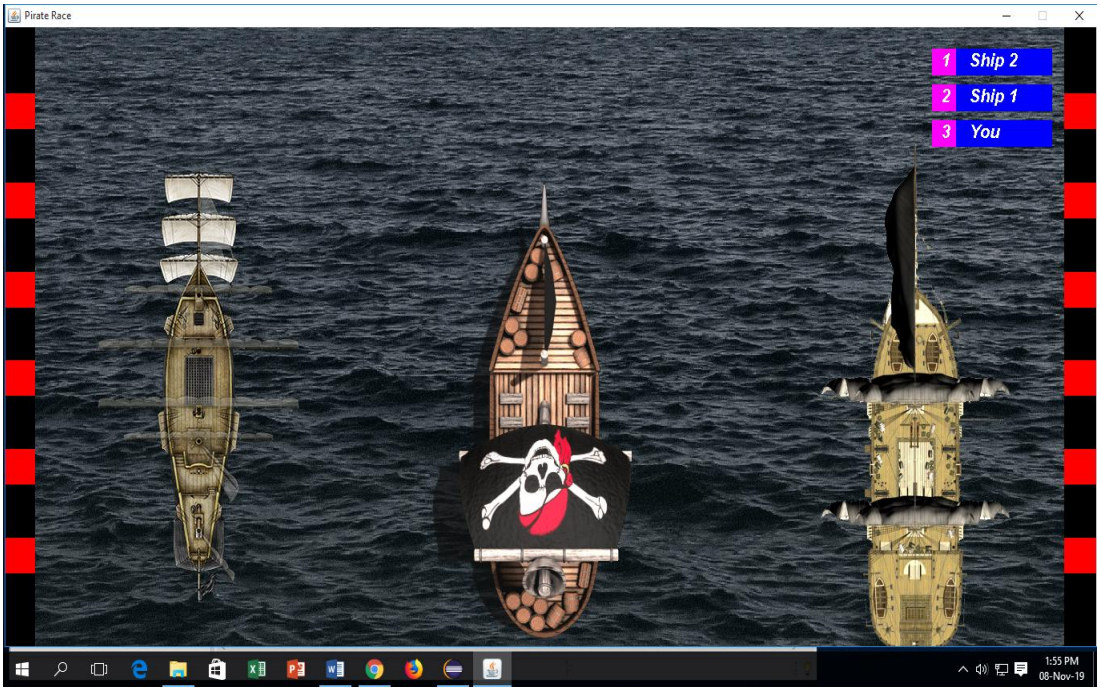

Fig. 1. The starting position of the implemented pirate race

3. GAME IMPLEMENTATION

Usually, a game consists of some basic components, like moving objects, backgrounds, friendly user interface, etc. In general, dealing with graphics is the most important task in race game programming. The Awt, Swing and Util packages are used to design Graphical User Interface (GUI) in games. Awt package is mainly used to design graphical user interface components such as buttons, labels, text field and other frames [7]. This application was implemented on eclipse using Java, because of its easy for coding. The implementation of Pirate Race can be done as follows:

3.1 Interface Design

The most important and fundamental component of the Awt package is JFrame. Actually, this component extends class Frame of package Awt. JFrame object manages the user interface with paint on the screen to receive information from the keyboard. The game initial JFrame properties are given below.

Algorithm:

//Setting the JFrame Window Size
Title of the JFrame=”Pirate Race”
JFrame Width=”1370”
JFrame Height=”725”

// Intialize the user ship position.
User Ship Initial X Coordinate Value=400
User Ship Initial Y Coordinate Value=100

3.2 Move Function

The movement of the user ship can be controlled by a method called move(). According to the keyboard events, the coordinate values are updated with corresponding directions. Each and every time of execution the run function invoke the move function. The code lines given below represents the user ship movement control statements.

Algorithm:

Function:move()

// Position of the user ship when racing.
X Coordinate newValue += eventValue in X Direction

```
 Y Coordinate newValue += eventValue in Y Direction

//Control the user ship movement within the window
If (X Coordinate newValue <= 100) then
        X Coordinate newValue=100;
If (X Coordinate newValue >= 800) then
        X Coordinate newValue=800;
If (Y Coordinate newValue <= 100) then
        Y Coordinate newValue=100;

If (Y Coordinate newValue <= 100) then
        Y Coordinate newValue=100;
```

3.3 Run Function

The Runnable Interface should have an undefined method run() with void as a return type to invokes in the thread that executes separately. It allows for implementing the game loop. The execution of the code can be delayed using a Thread.sleep(Delay Time).The codes are given below represent how the run() invoked in the thread.

```
public void run(){
      try {
           while(true)
              {
                 move();Thread.sleep(20);
              }
        }
      catch (Exception e)
        {
           System.err.println(e.getMessage());
        }
  }
```

3.4 Background Sound

The AudioInputStream object starts its own thread that runs until the program terminates and it loads an audio clip in the constructor to play it when the event fires. Initially create a “sound” directory in the “src” folder and put a file named “Name of the sound file.wav” in that directory. The program lines are given below.

```
URL url = this.getClass
().getClassLoader().getResource("sound/15-He's
A Pirate.wav");
AudioInputStream audioIn =
AudioSystem.getAudioInputStream(url);
Clip clip = AudioSystem.getClip();
Clip. Open (audioIn);
clip.start ();
```

3.5 An Abstract Adapter Class for Receiving Keyboard Events

Defining an abstract class Key Adapter for receiving keyboard events [8,9]. While moving the user ship on the Screen the coordinate values increase/ decrease according to the key pressed in the keyboard. The following codes are shows how to assign a value when pressing up, down, right, left keys.

Algorithm:

```
Functon:keypressed()

//Initialize

key=get the key pressed

if (key=Left Arrow Key) then
        eventValue in X Direction=-4;
if (key=Right Arrow Key) then
        eventValue in X Direction=+4;

if (key=Up Arrow Key) then
        eventValue in Y Direction=-4;

if (key=Down Arrow Key) then
        eventValue in Y Direction=+4;
```

```
Functon:keyreleased()

//Initialize
key=get the key released

if (key=Left Arrow Key) then
        eventValue in X Direction=0;
if (key=Right Arrow Key) then
        eventValue in X Direction=0;

if (key=Up Arrow Key) then
        eventValue in Y Direction=0;

if (key=Down Arrow Key) then
        eventValue in Y Direction=0;
```

3.6 Get the Image Resource from A Folder

An abstract class Toolkit is present in java.awt package [10]. It is a predefined method available in the toolkit class. This way the Toolkit object is created in the class. getImage() is a predefined abstract method present in Toolkit class [9]. The return type of this method is the Image class object. The following are Java codes for showing how to use getImage () for loading the image into the interface.

```
Toolkit toolkit = Toolkit.getDefaultToolkit ();
Image image=toolkit.getImage (this.getClass
().getClassLoader ().getResource
("images\\sea.gif"));
```

3.7 Winning Condition

The player has to finish the race at first place in order to win the race. At the start of the race, all three ships distance are initialized to zero. Then the user ship distance is incremented along with the up-key pressed in the keyboard by the player. The winning condition is checked in the finish line that depending on the maximum distance traveled by the ships. The following algorithm developed for the winning and losing condition of the User ship.

Algorithm:

Let at the Beginning of the race,

```
//Initialize

Ship1 Distance=0
Ship2 Distance=0
User Ship Distance=0

If (Ship1 Distance>10000 || Ship2 Distance >10000)
   {
      If (Ship1 Distance< (User Ship Distance) && Ship2 Distance< (User Ship Distance))
{
        Print ("You Won!!!! ");
                        Exit ()
                }

Else if (Ship1 Distance> (User Ship Distance) && Ship2 Distance< (User Ship Distance))
        {

        Print ("You Are Second!!!! ");
                        Exit ()
                }
        Else if (Ship1 Distance< (User Ship Distance) && Ship2 Distance> (User Ship Distance))

{
        Print ("You Are Second!!!! ");

                        Exit ()
}

Else if (Ship1 Distance> (User Ship Distance) && Ship2 Distance > (User Ship Distance))
```

```
{
        Print ("You Are Lost!!!!");
                Exit ()
}
}
```

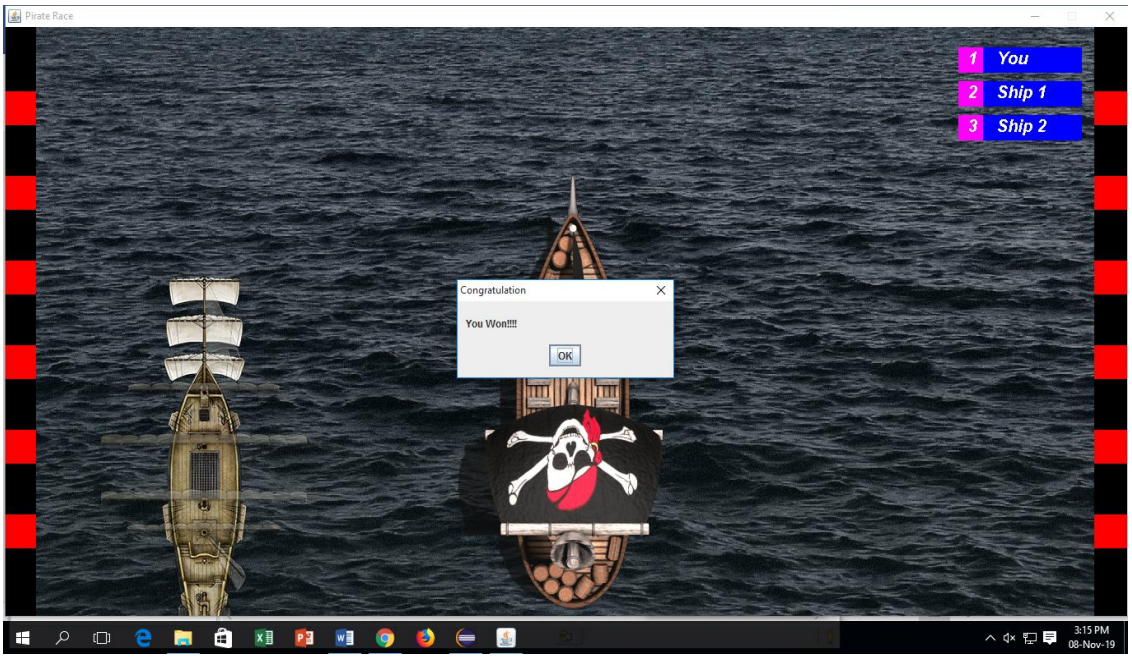

Fig. 2. The winning position of the pirate race

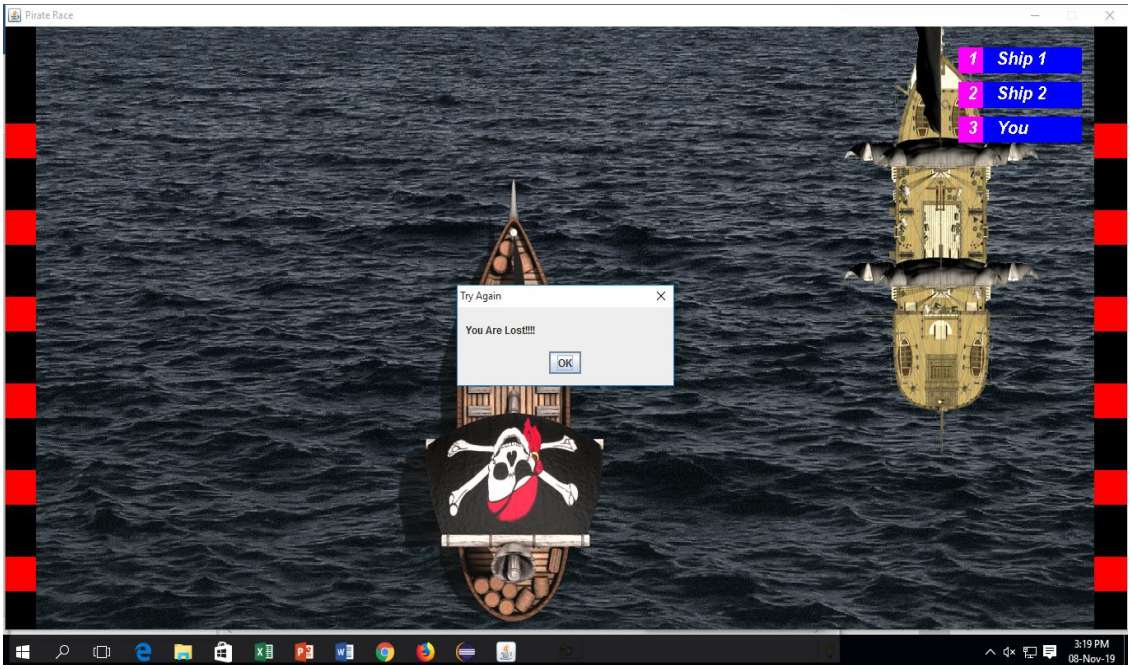

Fig. 3. The loosing position of the pirate race

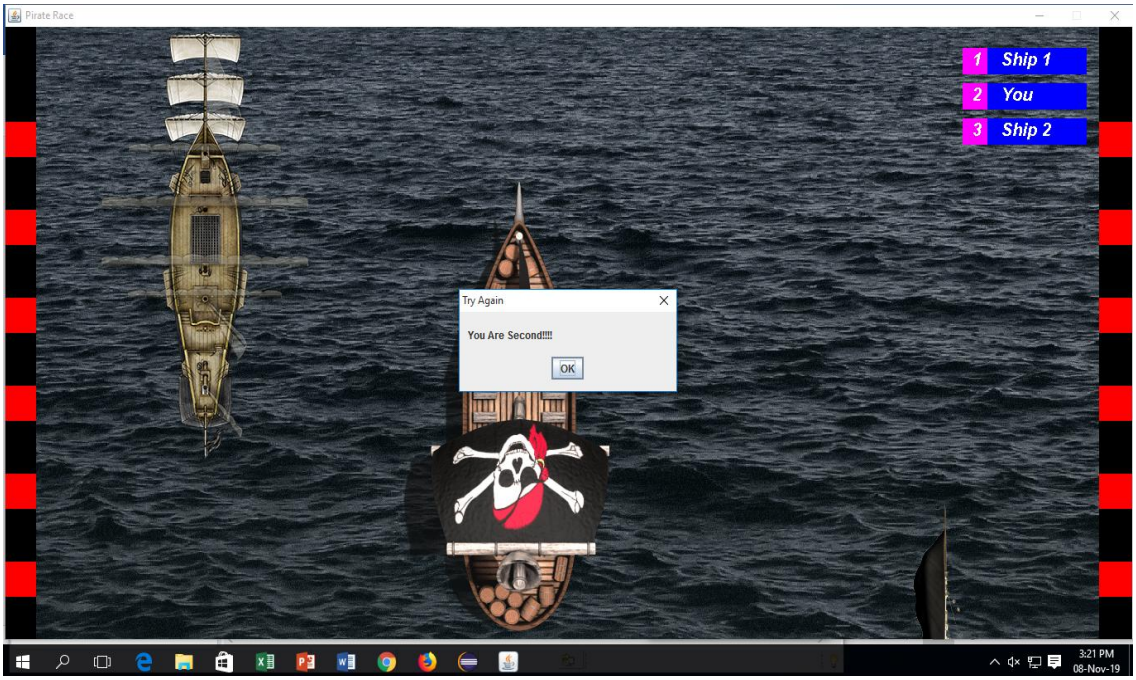

Fig. 4. The second position of the pirate race

3.8 Crashing With Ice Mountain

During a race, if a User Ship hits an Ice Mountain, the game comes to an end; to avoid this situation the ship needs to use correct moves to pass the Ice Mountain. In this paper, the crashing condition has been checked using the coordinate position of Ice Mountain reside in the screen and the user ship coordinates position on the screen while racing. If both coordinate positions are intercepted with each other, then hits the ice mountain becomes true otherwise it will be false. The following algorithm describe how the condition check using a Java if statement.

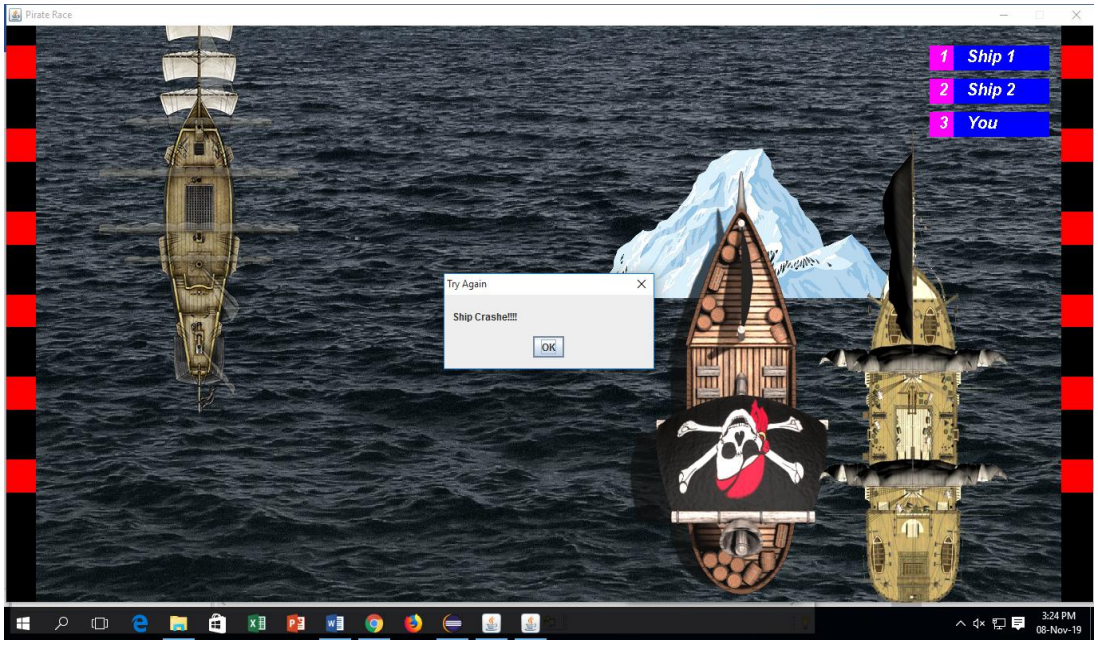

Fig. 5. The crashing position of the pirate race

Algorithm:

Input: User ship X, Y coordinate, Ice Mountain X, Y coordinate, Ice Mountain Width and Height.

If (User ship X coordinate < (Ice Mountain X Coordinate + Ice Mountain Width) && User ship Y Coordinate < (Ice Mountain Y Coordinate + Ice Mountain Height) && User ship X coordinate > (Ice Mountain X Coordinate) && User ship Y coordinate > (Ice Mountain Y Coordinate)) {

Print ("Ship Crashed!!!! ");

Exit ()

}

4. CONCLUSION

I expect this paper will be an easy approach to designing games in java. The Pirate Race was tested by over twenty users, who all reported that they thoroughly enjoyed the game. It was observed most of the users who were able to win the game with easy controls. This suggests that not only did the game provide an entertaining game experience, it also provided an understandable way to implement Java games using a runnable interface. It will grab the attention of a much wider body of java game developers who willing to develop their own game and allow newcomers to the field of game programming to feel comfortable.

In general, it can be concluded that the Runnable Interface supported the efficient development of the Pirate Race game. The interface supports implementing the game with background sound, 2D images with different file types and the key events. Making decisions based on multiple conditional cases has definitely a strong advantage over hard computing techniques and the paper will motivate the java learners in the world of game programming.

COMPETING INTERESTS

Author has declared that no competing interests exist.

REFERENCES

1. Doke ER, Hardgrave BC, Johnson RA, Doke ER, Hardgrave BC, Johnson RA. An introduction to object-oriented programming. COBOL programmers Swing with Java. 2011;21-40.
2. Clingman D, Kendall S, Mesdaghi S. Practical Java Game Programming; Charles River Media; 1 edition; 2004.
3. BA, PJ, Bhosale KA. Research paper on java interactional development environment programming tool. IARJSET. 2017;4(4):121–124.
4. Bjork S, Lundgren S, Holopainen J. Game design patterns, in: Lecture Note of the Game Design track of Game Developers Conference 2003, March 4–8, San Jose, CA, USA; 2003.
5. Fowler A, Fristoe T, MacLaurin M. Kodu game lab: A programming environment. The Computer Games Journal: Whitsun 2012, TuDo-cs Ltd. 2012;17-22.
6. McWhertor Michael. hydro thunder returns with all-new xbox live arcade sequel. Kotaku; 2010.
(Retrieved August 24, 2010)
7. Jain S. Developing Games (March 26, 2010). in Java for Beginners. Dev. Games Java Beginners. 2016;4(ii):693–696.
8. Dong Y, Ying. Design and evaluation of Java game programming environment: Major report, Concordia University; Last Modified; 2018.
9. Mads Hansen, Jacob Dinesen Gronhund. Java 2D Games, Roskilde University; 2012.
10. David Fox, Roman Verhovsek, Micro Java Game Development, Addison-Wesley Professional; 2002.

Peer-review history:
The peer review history for this paper can be accessed here:
http://www.sdiarticle4.com/review-history/53088