DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institute affiliated to VTU, Belagavi, Approved by AICTE & ISO 9001:2008 Certified)

Accredited by National Assessment & Accreditation Council (NAAC) with 'A' grade, Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru-560078.

Mini Project Report

on

"Snake Xenzia"

Submitted By

Darshan H E [1DS18CS707]
Gautam Kumar [1DS18CS710]
R Srinath [1DS18CS729]
Shaik Iftekhar Ahmed [1DS18CS736]

[Third Semester B.E (CSE)]
in
Object Oriented Programming with Java

Under the guidance of
PROF. SHRAVYA AR
Assistant Professor
Dept. of CSE
DSCE, Bangalore

Department of Computer Science and Engineering
Dayananda Sagar College of Engineering
Bangelore-78

Abstract

Snake is an older classic video game. It was first created in late 70s. Later it was brought to PCs. Snake is a classic video game in which a player controls a Snake that must eat apples and avoid walls. Each time, you eat an apple, the Snake grows. The Snake's goal is to eat the maximum of apples. It's a game of kind infinite. Snake is also an ideal game to create when you want to learn to create 2D games in Java.
In this snake going to eat objects randomly emerging on screen and if successful in eating then it becomes larger in size and gains score. The player has to change the direction of the snake by pressing left, right, top, down arrows for getting the food. Addition feature is completion of the game in a given time which makes game addictive and has ability to mesmerize the player.

Table of Contents

Introduction

Snake is the common name for a video game concept where the player maneuvers a line which grows in length, with the line itself being a primary obstacle. The concept originated in the 1976 arcade game *Blockade*, and the ease of implementing *Snake* has led to hundreds of versions (some of which have the word *snake* or *worm* in the title) for many platforms. After a variant was preloaded on Nokia mobile phones in 1998, there was a resurgence of interest in the snake concept as it found a larger audience. There are over 300 *Snake*-like games for iOS alone.

Design/Implementation

The size of each of the joints of a snake is 10 px. The snake is controlled with the cursor keys. Initially, the snake has three joints. If the game is finished, the "Game Over" message is displayed in the middle of the board.

First we will define the constants used in our game.

```
private final int B_WIDTH = 300;
private final int B_HEIGHT = 300;
private final int DOT_SIZE = 10;
private final int ALL_DOTS = 900;
private final int RAND_POS = 29;
private final int DELAY = 140;
```

The B_WIDTH and B_HEIGHT constants determine the size of the board. The DOT_SIZE is the size of the apple and the dot of the snake. The ALL_DOTS constant defines the maximum number of possible dots on the board (900 = (300*300)/(10*10)). The RAND_POS constant is used to calculate a random position for an apple. The DELAY constant determines the speed of the game.

```
private final int x[] = new int[ALL_DOTS];
private final int y[] = new int[ALL_DOTS];
```

These two arrays store the x and y coordinates of all joints of a snake.

```
private void loadImages() {

    ImageIcon iid = new ImageIcon("src/resources/dot.png");
    ball = iid.getImage();

    ImageIcon iia = new ImageIcon("src/resources/apple.png");
    apple = iia.getImage();

    ImageIcon iih = new ImageIcon("src/resources/head.png");
    head = iih.getImage();
}
```

In the loadImages() method we get the images for the game. The ImageIcon class is used for displaying PNG images.

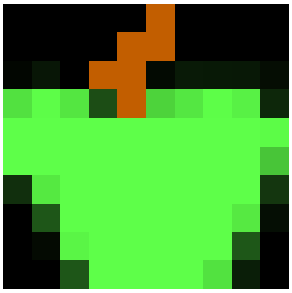

Figure 1.1: apple.png

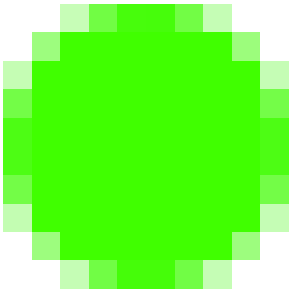

Figure 1.2: dot.png

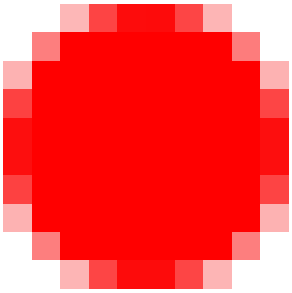

Figure 1.3: head.png

```
private void initGame() {

    dots = 3;

    for (int z = 0; z < dots; z++) {
        x[z] = 50 - z * 10;
        y[z] = 50;
    }

    locateApple();

    timer = new Timer(DELAY, this);
    timer.start();
}
```

In the initGame() method we create the snake, randomly locate an apple on the board, and start the timer.

```
private void checkApple() {

    if ((x[0] == apple_x) && (y[0] == apple_y)) {

        dots++;
        locateApple();
    }
}
```

If the apple collides with the head, we increase the number of joints of the snake. We call the locateApple() method which randomly positions a new apple object.

In the move() method we have the key algorithm of the game. To understand it, look at how the snake is moving. We control the head of the snake. We can change its direction with the cursor keys. The rest of the joints move one position up the chain. The second joint moves where the first was, the third joint where the second was etc.

```
for (int z = dots; z > 0; z--) {
    x[z] = x[(z - 1)];
    y[z] = y[(z - 1)];
}
```

This code moves the joints up the chain.

```
if (leftDirection) {
    x[0] -= DOT_SIZE;
}
```

This line moves the head to the left.

In the checkCollision() method, we determine if the snake has hit itself or one of the walls.

```
for (int z = dots; z > 0; z--) {

    if ((z > 4) && (x[0] == x[z]) && (y[0] == y[z])) {
        inGame = false;
    }
}
```

If the snake hits one of its joints with its head the game is over.

```
if (y[0] >= B_HEIGHT) {
    inGame = false;
}
```

The game is finished if the snake hits the bottom of the board.

```
setResizable(false);
pack();
```

The setResizable() method affects the insets of the JFrame container on some platforms. Therefore, it is important to call it before the pack() method. Otherwise, the collision of the snake's head with the right and bottom borders might not work correctly. [1]

Testing/Result and Analysis

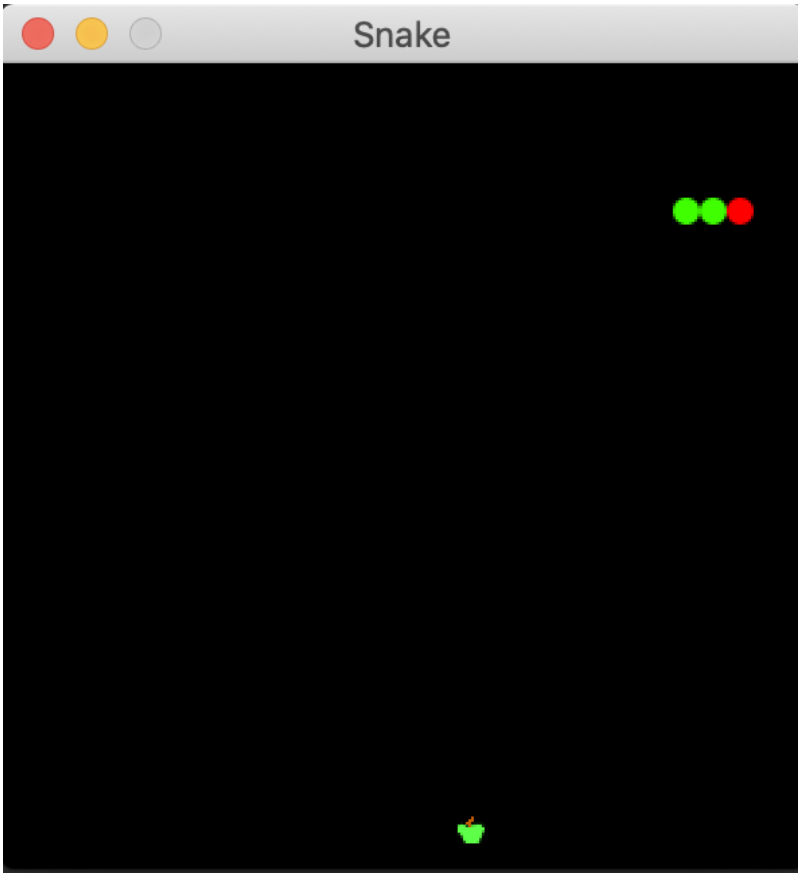

Figure 2: Initial Game jFrame

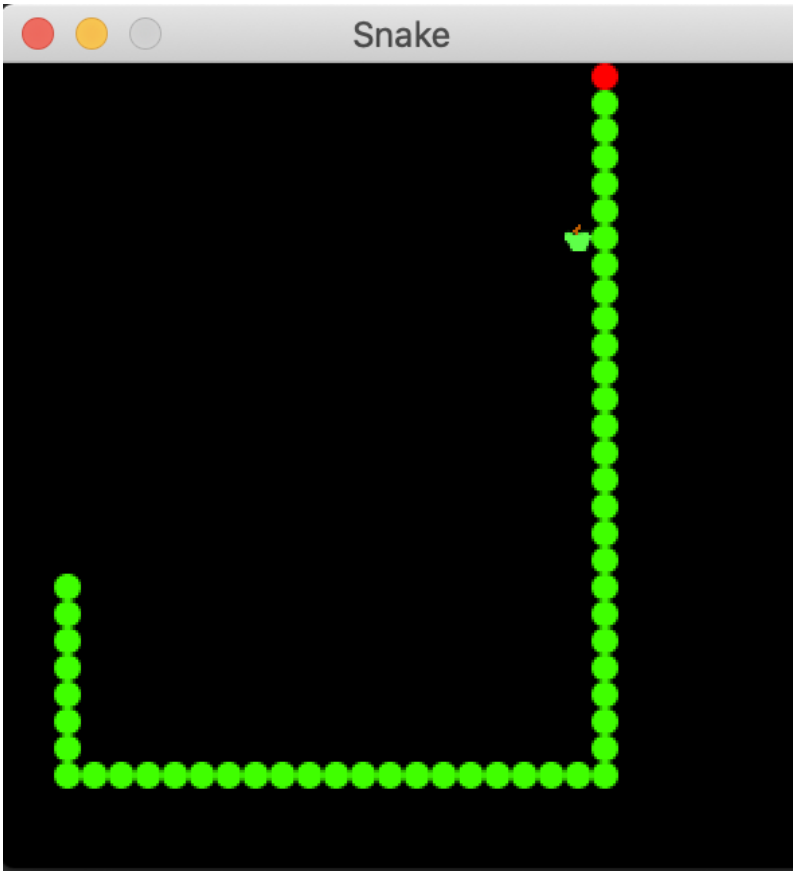

Figure 3: In-Game jFrame

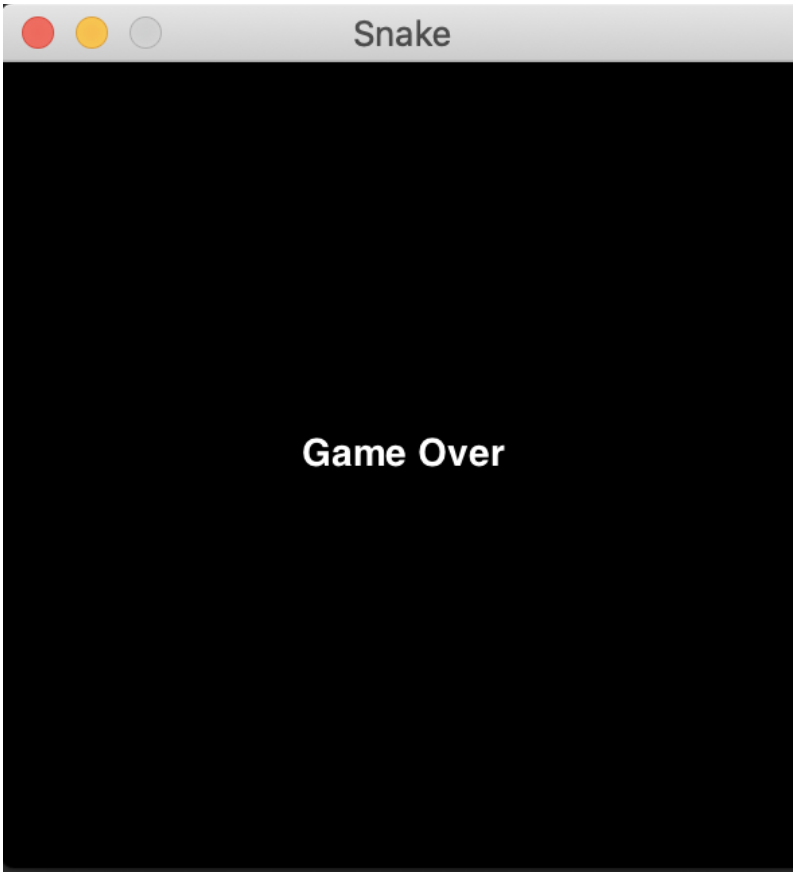

Figure 4: Game Over jFrame

Conclusions & Future Enhancements

We present a simple Snake game using Java programming language. Further enhancements are to be made in the future to make this game more interactive. We intend to add a scoreboard to track the score in the game and challenge others based on high score. Further, we would like to implement different difficulty levels for the game and control the speed of the snake during gameplay. And also would like to update other obstacles in the game to make the game much more interesting.

References

Works Cited

[1] Jan bodnar, "Java Snake Game," [Online]. Available: http://zetcode.com/tutorials/.