# Classic Console Snake Game

Alexis Lyndon Galaura
BSIT - 1
Xavier University–Ateneo de Cagayan
**alexislyndon@gmail.com**

Tomas Jubile Libago
BSIT - 1
Xavier University–Ateneo de Cagayan
**libagotj09@gmail.com**

Youssef Hisham
BSIS - 1
Xavier University–Ateneo de Cagayan
**bojohisham69@gmail.com**

## 1. INTRODUCTION

### a. Overview

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 along with the walls. The snake grows in length as more and more food(or apples as it's commonly known) are eaten. Until a time where the snake covers the whole board and the player wins.

### b. Objectives

**a.**

The group's objective for this project is to take the Classic Nokia Snake Game as a model and mimic its features so that it's playable in Windows console.

### c. Scope and Limitations

The group will take the Classic Nokia keypad Snake game as a model for our project and will try to implement the features found therein.

The group will stay true to the teachings in class and will not use a Graphical User Interface. It will run on command prompt (console).

While the Classic snake game requires only 1 keypress for a movement to be executed. It is not possible in console where a user is required to press enter for an input to be read. Therefore, the group will stick to what the console is capable.

**b.**

### d. Functionalities

The game uses WASD for the snake's movements. W moves the snake up, A moves the snake to the left, S moves the snake downwards, and D moves the snake to the right. Just like the Classic Nokia game, the user can only control the Head section of the snake while the rest of the snake's body follows the head.

Like the Classic Nokia game, every after about half a second(500ms) the snake's head will have moved exactly 1 square to the direction its facing. This feature will be reflected in our program and will be used defined to simulate difficulty.

To make the snake game somewhat unique, a feature has been added wherein the snake shrinks after eating a certain food which we call *badfood*.

The game will also feature a wall-less mode much like in the later revisions of the Classic Nokia game - where the snakes head will pop out the opposite site after hitting the board's border.

## 2. PROGRAM DESIGN AND IMPLEMENTATION

### a. Pseudocode

```
addFood, setSnakePOS via Constructor
while(!gameover) {
        THREAD - 0 s = sc.next()
        MAIN THREAD check if(food eaten)
                                addFood
        MAIN THREAD moveSnake(char s) - head
moves one square
        MAIN THREAD moveSnake(char s) - whole
body follows
        MAIN THREAD check if(snake is dead)
                                return gameover
}
```

### b. Data Structures and Algorithms Discussion and Code Snippets

The data structures used were Arrays, Linked Lists, and Multi-threading. Arrays were used for the board and the placement of food/badfood and the initial position of the snake. Linked Lists were used for the coordinates of the food/badfood and the snake. Multithreading was used in order to move the snake in the direction the user wanted while a parallel thread runs to update the board.

Algorithm Discussion

The board is created by creating a new object of class Linkedstack via *constructor*. The constructor also sets the initial position of the food/badfood and snake *nodes* that sit on top of the array.

The main method then calls the method Mayn() from class Linkedstack to check if any food were eaten. Since its the first iteration, the Mayn() will move on to the last line and call method Coordinatesmove(char input) which facilitates the movement of the snake's head depending on the user's input. It is also in Coordinatesmove(char input) method that the snake's body will follow where the head goes. Therefore the important node to watch for is the *head* node.

When everything has move to its final state, Coordinatesmove(char input) returns false if the snake haven't eaten itself. In the main() method, boolean getout serves to receive Coordinatesmove(char input) return value. So long as boolean getout is false, the game will continue looping back to Mayn().

When the group first conceptualized this project. The main problem was user input. Using the scanner class was sufficient but it would mean that the whole program would then wait for the user to press enter before moving on - that would make for a lousy snake game.

To get around this, the group used Multithreading. A separate thread was created with the help of the Thread class which is built into Java. The separate thread(separate from the main thread) will wait on the user to press enter but will not affect the main thread which continuously loops and recycles the last input of the user which will make the snake move continuously until another move key is pressed.

To make the snake's head move this is one of the codes used:

```
int tempx=head.x;
int tempy=head.y;
else if(move=='d') { //move right
      if((this.head.x+1    ==    15)    &&
(this.walls == false)) {
this.head.x = 0;//jump to the other side
- walls off
      }
      else {
            head.x=head.x+1;
      }
}
```

Now that the head has moved, we now make the body of the snake follow the head:

```
int tempx1, tempy2;
for(temp = head.next; temp != null; temp
= temp.next) {
            tempx1=temp.x;
            tempy2=temp.y;
            temp.x=tempx;
            temp.y=tempy;
            tempx=tempx1;
            tempy=tempy2;
}
```

A problem encountered by the group was that the snake could turn 180 degrees. This code snippet prevents this:

```
t = sc.next().charAt(0);//user input
if(t=='a' && s=='d'){
      t = 'd';
}else{//first run goes here directly
      s = t;
}
```

## 3. CONCLUSION

The game works well enough to be played at low refresh rates. Upping the refresh rate to 90ms makes the game painful to watch in console as the elements of the game start display abnormally. This is because the console can't scroll fast enough to display more than 10 game frames per second. The group assumes that this kind of instability doesn't happen with games with a GUI. Therefore the Console Snake Game is limited to a 90ms update or refresh rate.

## 4. REFERENCES


Linux, HTML, PHP, and JavaScript Basics. (2015). *Web Programming for Business*,1-37. doi:10.4324/9780203582084-1

BlenderBlender 210k36341404, Peoropeoro 17.7k1473131, Bjornarsbjornars 8981712, & JohnJohn 5. (n.d.). How does sleep() work? Retrieved from https://stackoverflow.com/questions/4911739/how-does-sleep-work

https://www.tutorialspoint.com/java/java_documentation.htm

http://www.drjava.org/docs/user/ch10.html

https://www.baeldung.com/javadoc

http://mycodefixes.blogspot.com/2012/03/generate-javadoc-through-command-line.html

https://corochann.com/javadoc-coding-rule-of-link-linkplain-see-372.html#Basic_rule

https://www.geeksforgeeks.org/multithreading-in-java/

https://www.tutorialspoint.com/java/java_multithreading.htm

generate javadocs

https://www.javatpoint.com/java-thread-start-method