# Introduction

## PREFACE

This article uses Processing 3.0 to build a Snake game, the single player version. The installation of the software is explained in the GETTING STARTED section. The basic gameplay and the Logic implementations are explained in the theory section. It is expected that you have a basic idea over the concepts of Java and Object-Oriented Programming (OOP) before you take on this article, for it may only help your own understanding of the subject matter.

The WALKTHROUGH section explains the coding part with the aid of screenshots and codes taken through in a step-by-step manner.

For any reference required to the methods used in the code, please refer to the REFERENCE SECTION.

Processing acts as a flexible cross-platform open-source graphical library and Integrated Development Environment (IDE) which has utilizations in the fields of GUI programming, electronic media arts and visual design.

The paradigm is object-oriented and was designed by Casey Reas and Ben Fry.

The OpenGL integration allows expandable scope for projects, while the interactive predefined output methods serve as a user-friendly interface for the programmer.

For more details, please visit: <https://processing.org/>

## GETTING STARTED

To get stared we must first download the software from the official website which is:

<https://processing.org/download>

and download the setup file for your preferred operating system.

Processing is available for Windows, Mac and Linux, and we shall go over on how to install them on each operating system.

Device Specific Installation instructions:

- On Windows you will be downloading a .zip file, extract it your preferred location on your hard disk. Once extracted, simply enter the folder and run processing.exe.

-On Mac you will be downloading a .zip file, extract it your preferred location on your hard disk. Once extracted, simply enter the folder and run processing.app.

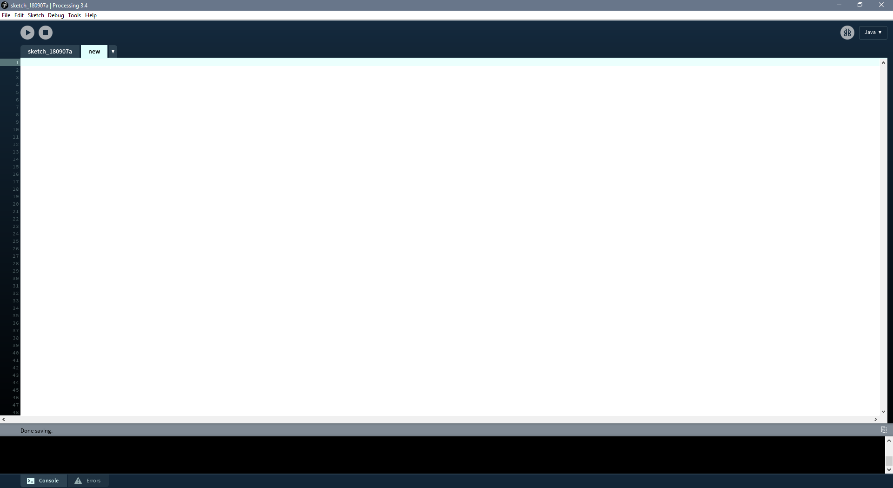
- On Linux you will be downloading the latest installation (processing-3.2.1-linux64.tgz, for example). After that, open a terminal and copy the installation to your home folder. Ince copied, extract using:

tar zxf processing-3.2.1-linux64.tgz

Run using:

~/processing-3.2.1/processing

The splash screen and application should now be displayed.



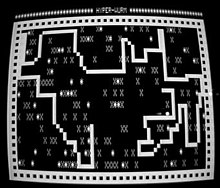
Processing window

Once you are done with your installation, you should be seeing a window similar to the one shown above. In-case if you have any trouble, please visit the wiki section in the oﬃcial website:

<https://github.com/processing/processing/wiki>

# THE SNAKE GAME

## THEORY

[](https://en.wikipedia.org/wiki/File:Snake_trs-80.jpg)[](https://en.wikipedia.org/wiki/File:Matopeli_telmac_1800.JPG)

Snake on a TRS-80 Snake on a Telmac 1800, [CHIP-8](https://en.wikipedia.org/wiki/CHIP-8)

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](https://en.wikipedia.org/wiki/Mobile_phone) in 1998, there was a resurgence of interest in the snake concept as it found a larger audience

[](https://en.wikipedia.org/wiki/File:Cgasnake.png)

Snake on a PC with a Color Graphics Adapter

### Gameplay

The player controls a dot, square, or object on a bordered plane. The snake has a specific length, so there is a moving tail a fixed number of units away from the head. The player loses when the snake runs into the screen border, a trail or itself. The player attempts to eat items by running into them with the head of the snake. Each item eaten makes the snake longer, so controlling is progressively more difficult.

### Logic Implemented

* The **Background** is done by using the *background()* function.
* The **Title** is done by using the *text()* function.
* The **Grid** is done by using two for loops which goes from 0 to the extremes of the viewport and draws lines at certain intervals. One for loop develops the vertical lines and the other draws the horizontal ones. This helps the player with the movement and spotting accuracy in the gameplay.
* The **SNAKE** is built using an array list of rectangles. The initial block (rectangle), also referred to as the head was assigned the color Red/Maroon. The rest are colored with a grey-ish tone.
* The **Movement** is done by using two integer arrays, using the keyPressed() function and integer variables. Depending on the final direction variable, the direction of the snake’s head changes (or doesn’t) and the rest of the snake is incremented in the same direction while with each frame, one block is decremented from the tail.
* The **Food** is a rectangle drawn at random coordinates using the random() function and the product of its random coordinates and the blockSize variable.
* The **Eating and Size Incrementation** is done using collision detection algorithm as well as adding of blocks in the array list, and thus increasing the number of rectangles to be drawn.
* The **Border** is put into effect by checking the head coordinate of the snake which must never exceed the extreme values of the viewport in anyway.
* The **GameOver Logic** is implemented by using the border as well as checking if the coordinate of the head is about to match with that of some other rectangle/body part of the snake. If yes, the constraint toggles TRUE and the game is reset.
* The **Score Board** shows score which is the total number of elements in the array list. The High score simply stores the highest value of the scores obtained after running the game.

### WALKTHROUGH

The basic setup:

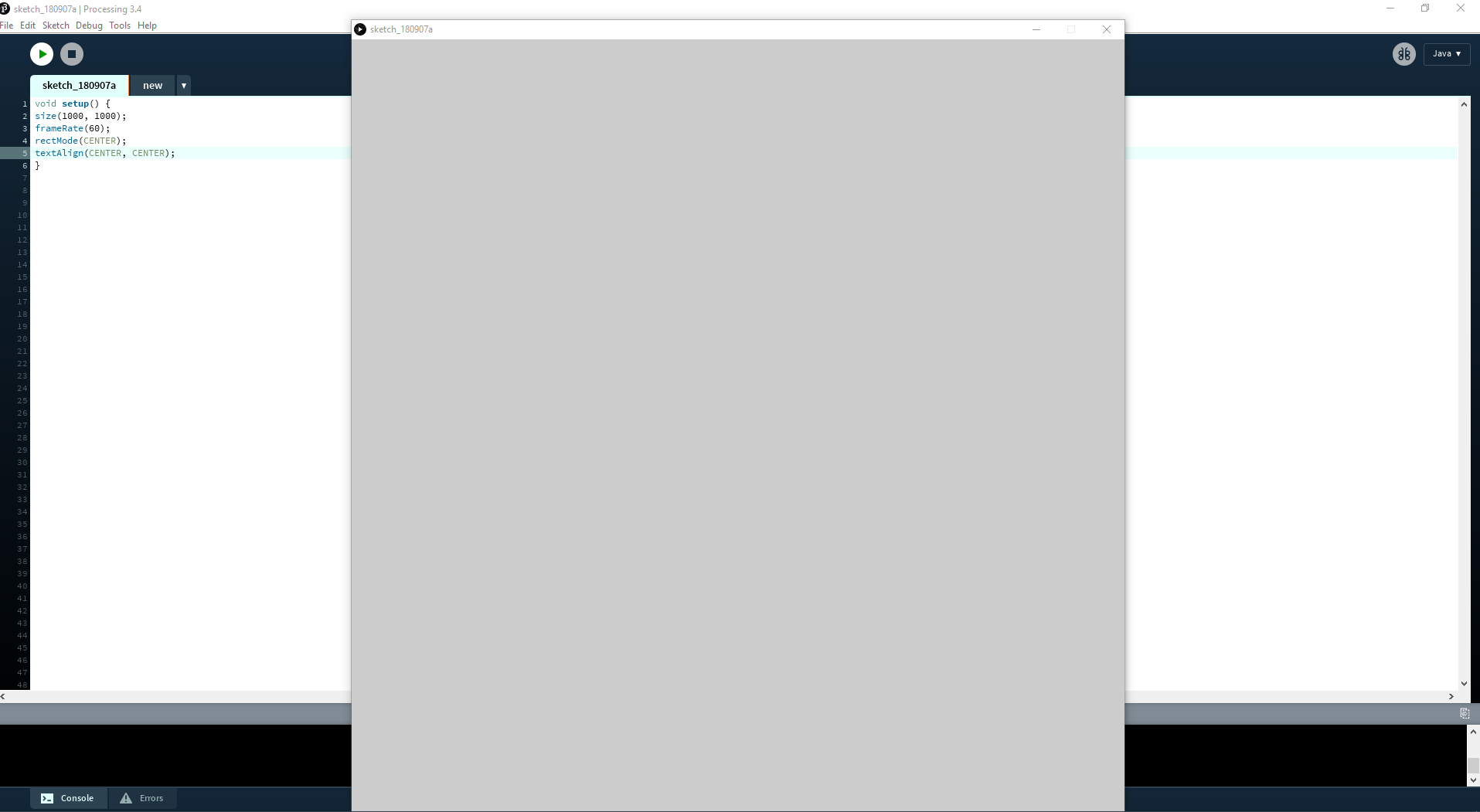
void setup() {

size(1000, 1000);

frameRate(60);

rectMode(CENTER);

textAlign(CENTER, CENTER);

}

Thus, we define the Viewport, the mode of the rect() function, the mode of the text function, as well as the framerate.

Creating the background and gridlines:

int widthGame=50;

int heightGame=50;

int blockSize=20;

void setup() {

size(1000, 1000);

frameRate(60);

rectMode(CENTER);

textAlign(CENTER, CENTER);

}

void draw() {

background(0,50);

strokeWeight(1);

for (int i=0;i<widthGame;i++) {

stroke(200,50);

line((i-0.5)\*blockSize,0,(i-0.5)\*blockSize,height);}

for(int i=0;i<heightGame;i++){

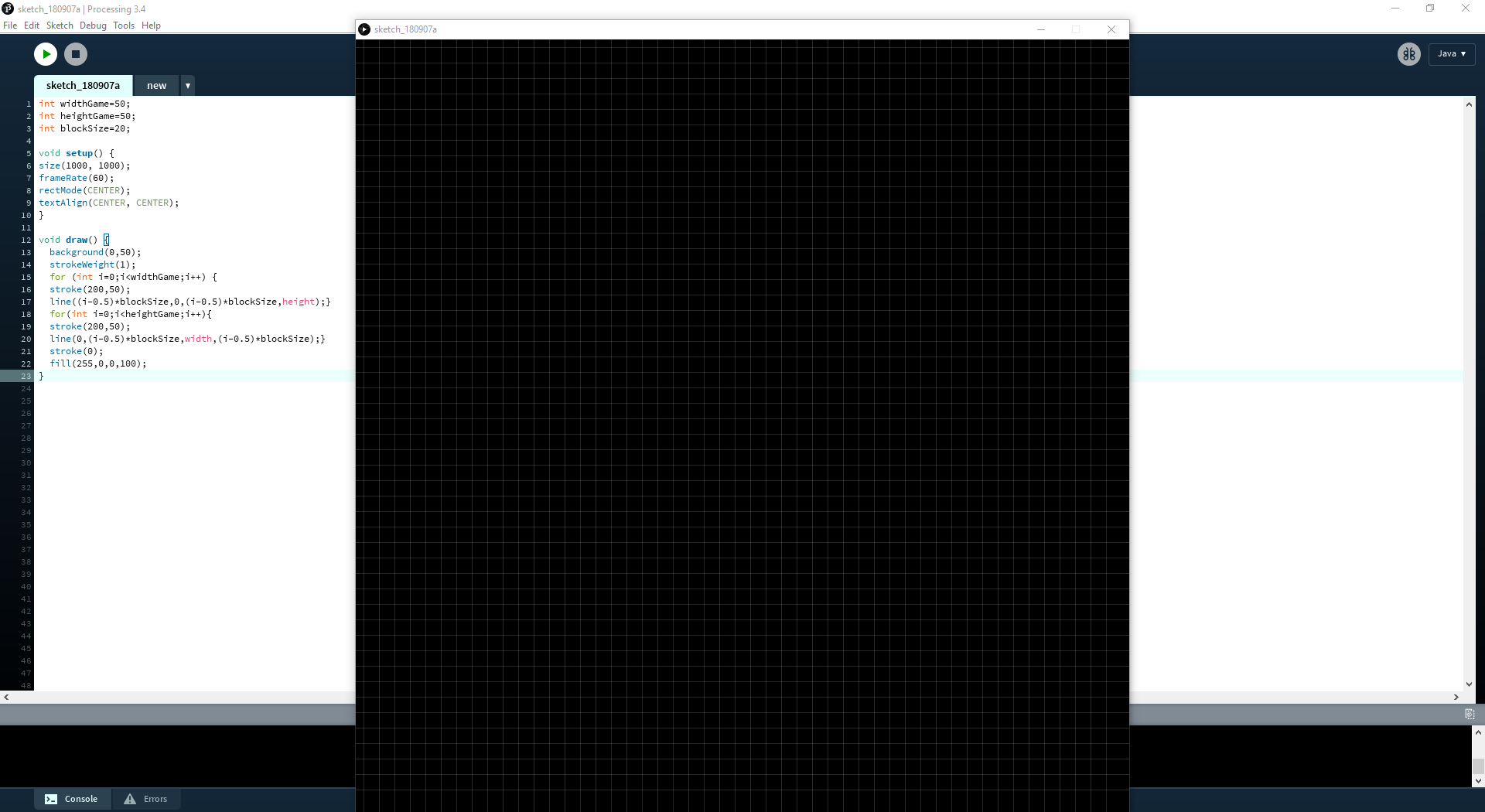
stroke(200,50);

line(0,(i-0.5)\*blockSize,width,(i-0.5)\*blockSize);}

stroke(0);

fill(255,0,0,100);

}



This Grid, as described before, acts as a guide while playing the game.

Drawing the snake with movement:

ArrayList<Integer> x = new ArrayList<Integer>(), y = new ArrayList<Integer>();

int widthGame=50;

int heightGame=50;

int blockSize=20;

int direction=floor(random(4));

int speedRatio=5;

int[]directionX={0,0,1,-1},directionY={1,-1,0,0};

void setup() {

size(1000, 1000);

x.add(6+(int)random(50));

y.add(6+(int)random(50));

frameRate(60);

rectMode(CENTER);

textAlign(CENTER, CENTER);

}

void draw() {

background(0,50);

strokeWeight(1);

for (int i=0;i<widthGame;i++) {

stroke(200,50);

line((i-0.5)\*blockSize,0,(i-0.5)\*blockSize,height);}

for(int i=0;i<heightGame;i++){

stroke(200,50);

line(0,(i-0.5)\*blockSize,width,(i-0.5)\*blockSize);}

stroke(0);

fill(255,0,0,100);

rect(x.get(0)\*blockSize, y.get(0)\*blockSize, blockSize, blockSize);

fill(255,100);

for (int i = 1 ; i < x.size(); i++)

rect(x.get(i)\*blockSize, y.get(i)\*blockSize, blockSize, blockSize);

if (frameCount%speedRatio==0) {

x.add(0, x.get(0) + directionX[direction]);

y.add(0, y.get(0) + directionY[direction]);

x.remove(x.size()-1);

y.remove(y.size()-1);

}

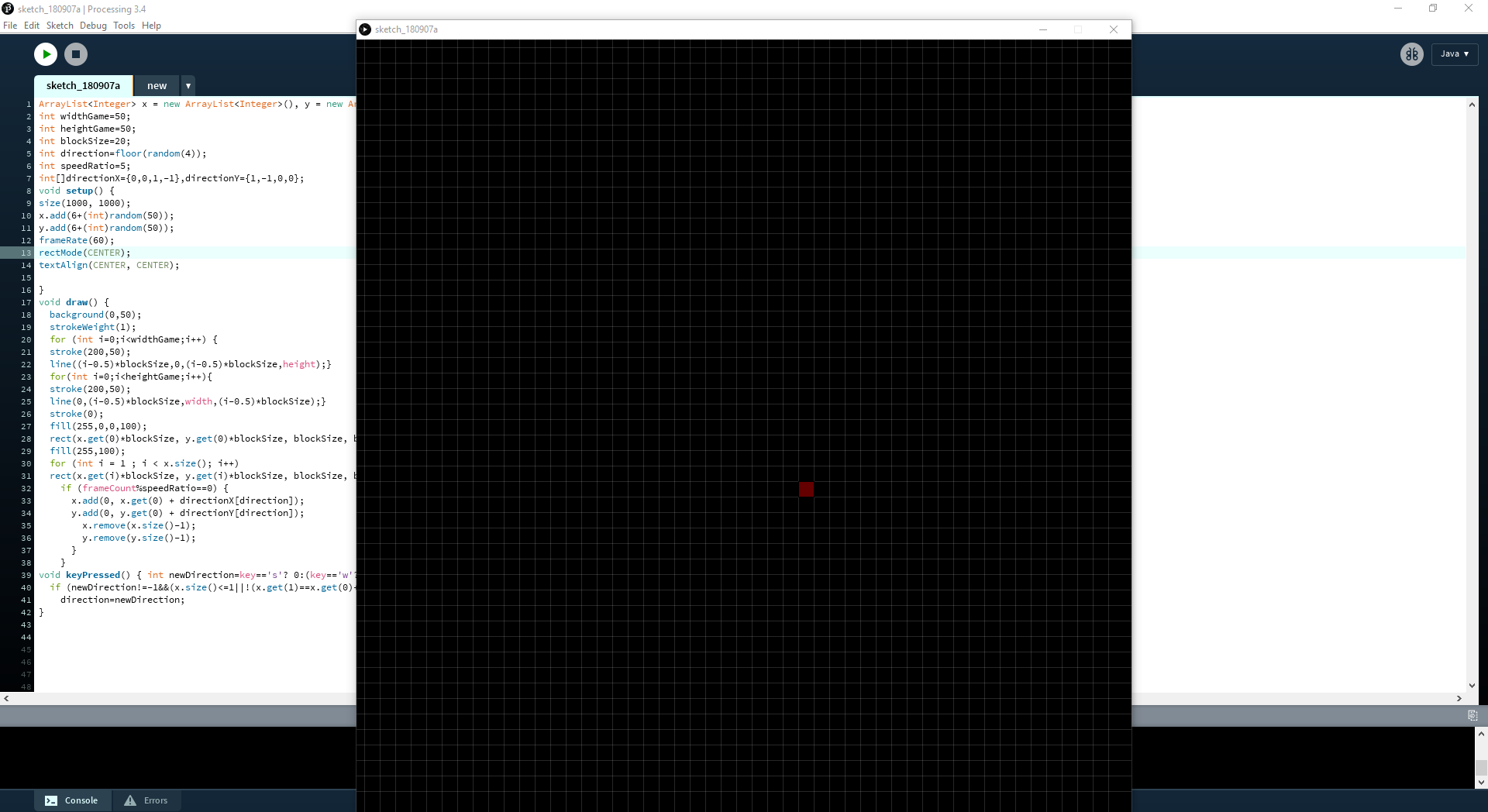
}

void keyPressed() { int newDirection=key=='s'? 0:(key=='w'?1:(key=='d'?2:(key=='a'?3:-1)));

if (newDirection!=-1&&(x.size()<=1||!(x.get(1)==x.get(0)+directionX[newDirection]&&y.get(1)==y.get(0)+directionY[newDirection])))

direction=newDirection;

}



The speedRatio decides the speed at which the screen is updated, so basically the speed of the game. A random function has been used to randomize the initial direction of the snake as well.

Adding Food:

ArrayList<Integer> x = new ArrayList<Integer>(), y = new ArrayList<Integer>();

int widthGame=50;

int heightGame=50;

int blockSize=20;

int direction=floor(random(4));

int appleX=12;

int appleY=10;

int speedRatio=5;

int[]directionX={0,0,1,-1},directionY={1,-1,0,0};

void setup() {

size(1000, 1000);

x.add(6+(int)random(50));

y.add(6+(int)random(50));

frameRate(60);

rectMode(CENTER);

textAlign(CENTER, CENTER);

}

void draw() {

background(0,50);

strokeWeight(1);

for (int i=0;i<widthGame;i++) {

stroke(200,50);

line((i-0.5)\*blockSize,0,(i-0.5)\*blockSize,height);}

for(int i=0;i<heightGame;i++){

stroke(200,50);

line(0,(i-0.5)\*blockSize,width,(i-0.5)\*blockSize);}

stroke(0);

fill(255,0,0,100);

rect(x.get(0)\*blockSize, y.get(0)\*blockSize, blockSize, blockSize);

fill(255,100);

for (int i = 1 ; i < x.size(); i++)

rect(x.get(i)\*blockSize, y.get(i)\*blockSize, blockSize, blockSize);

fill(200, 20, 100);

rect(appleX\*blockSize, appleY\*blockSize, blockSize, blockSize);

if (frameCount%speedRatio==0) {

x.add(0, x.get(0) + directionX[direction]);

y.add(0, y.get(0) + directionY[direction]);

if (x.get(0)==appleX && y.get(0)==appleY) {

appleX = (int)random(5, widthGame-5);

appleY = (int)random(5, heightGame-5);

}

else {

x.remove(x.size()-1);

y.remove(y.size()-1);

}

}

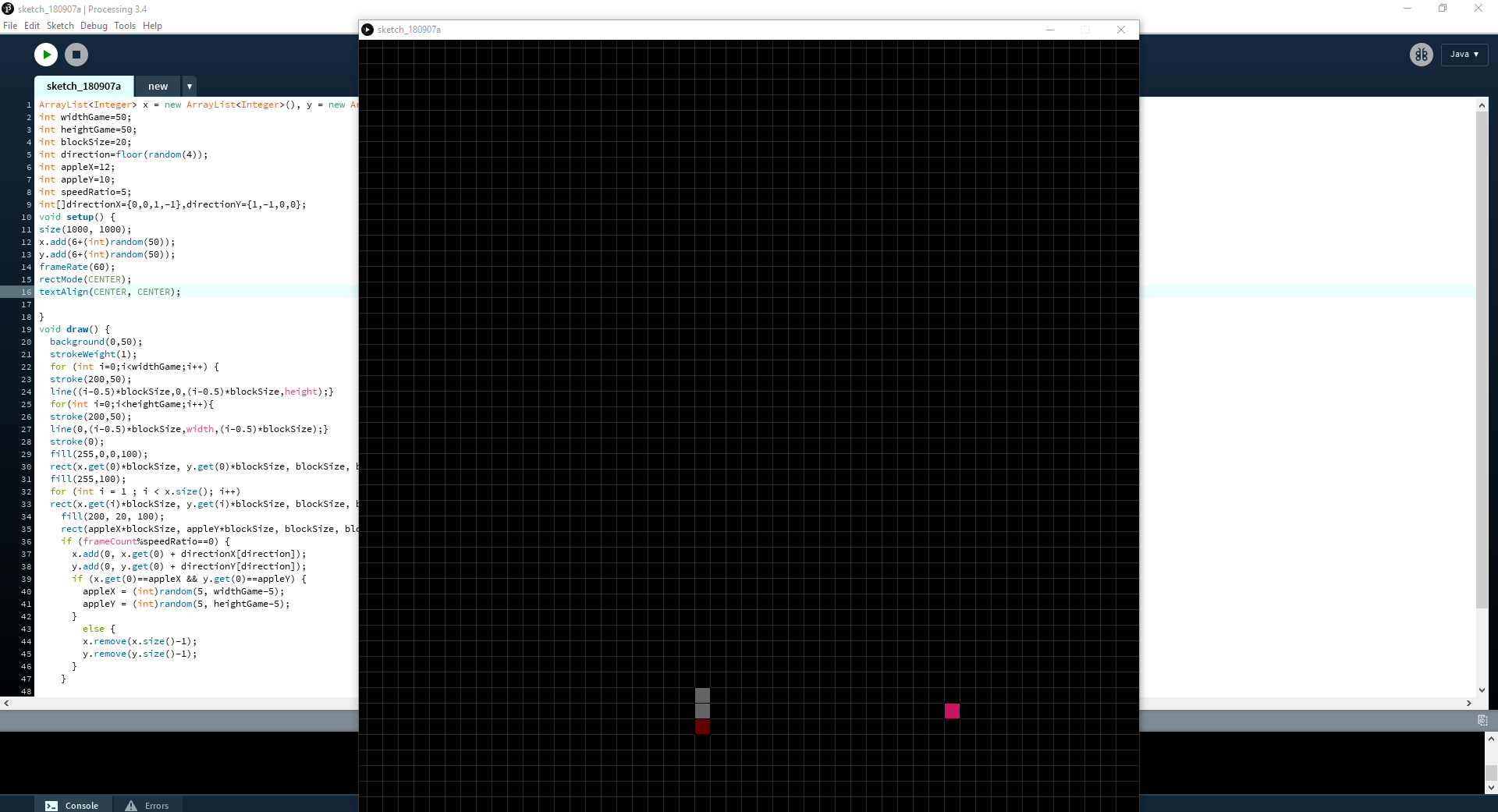
}

void keyPressed() { int newDirection=key=='s'? 0:(key=='w'?1:(key=='d'?2:(key=='a'?3:-1)));

if (newDirection!=-1&&(x.size()<=1||!(x.get(1)==x.get(0)+directionX[newDirection]&&y.get(1)==y.get(0)+directionY[newDirection])))

direction=newDirection;

}



Now the food appears randomly through out the grid excluding 5 blocks from the border, thus assuring less glitchy gameplay.

Adding the gameOver constraints:

ArrayList<Integer> x = new ArrayList<Integer>(), y = new ArrayList<Integer>();

int widthGame=50;

int heightGame=50;

int blockSize=20;

int direction=floor(random(4));

int appleX=12;

int appleY=10;

int speedRatio=5;

int[]directionX={0,0,1,-1},directionY={1,-1,0,0};

boolean gameOver=false;

int highScore=0;

void setup() {

size(1000, 1000);

x.add(6+(int)random(50));

y.add(6+(int)random(50));

frameRate(60);

rectMode(CENTER);

textAlign(CENTER, CENTER);

}

void draw() {

background(0,50);

strokeWeight(1);

for (int i=0;i<widthGame;i++) {

stroke(200,50);

line((i-0.5)\*blockSize,0,(i-0.5)\*blockSize,height);}

for(int i=0;i<heightGame;i++){

stroke(200,50);

line(0,(i-0.5)\*blockSize,width,(i-0.5)\*blockSize);}

stroke(0);

fill(255,0,0,100);

rect(x.get(0)\*blockSize, y.get(0)\*blockSize, blockSize, blockSize);

fill(255,100);

for (int i = 1 ; i < x.size(); i++)

rect(x.get(i)\*blockSize, y.get(i)\*blockSize, blockSize, blockSize);

if (!gameOver) {

fill(200, 20, 100);

rect(appleX\*blockSize, appleY\*blockSize, blockSize, blockSize);

if (frameCount%speedRatio==0) {

x.add(0, x.get(0) + directionX[direction]);

y.add(0, y.get(0) + directionY[direction]);

if(x.get(0) < 0 || y.get(0) < 0 || x.get(0) >= widthGame || y.get(0) >= heightGame)

gameOver = true;

for(int i=1;i<x.size();i++)

if(x.get(0)==x.get(i)&&y.get(0)==y.get(i))

gameOver=true;

if (x.get(0)==appleX && y.get(0)==appleY) {

appleX = (int)random(5, widthGame-5);

appleY = (int)random(5, heightGame-5);

}

else {

x.remove(x.size()-1);

y.remove(y.size()-1);

}

}

}

else {

fill(255,0,0,255);

textSize(40);

textAlign(CENTER);

text("GAME OVER. Press SPACE to continue.",width/2,height/2);

fill(255,0,0,255);

for (int i = 0 ; i < x.size(); i++)

rect(x.get(i)\*blockSize, y.get(i)\*blockSize, blockSize, blockSize);

if(keyPressed&&key==' ') {

x.clear();

y.clear();

x.add(25);

y.add(25);

gameOver = false;

direction=floor(random(4));

}

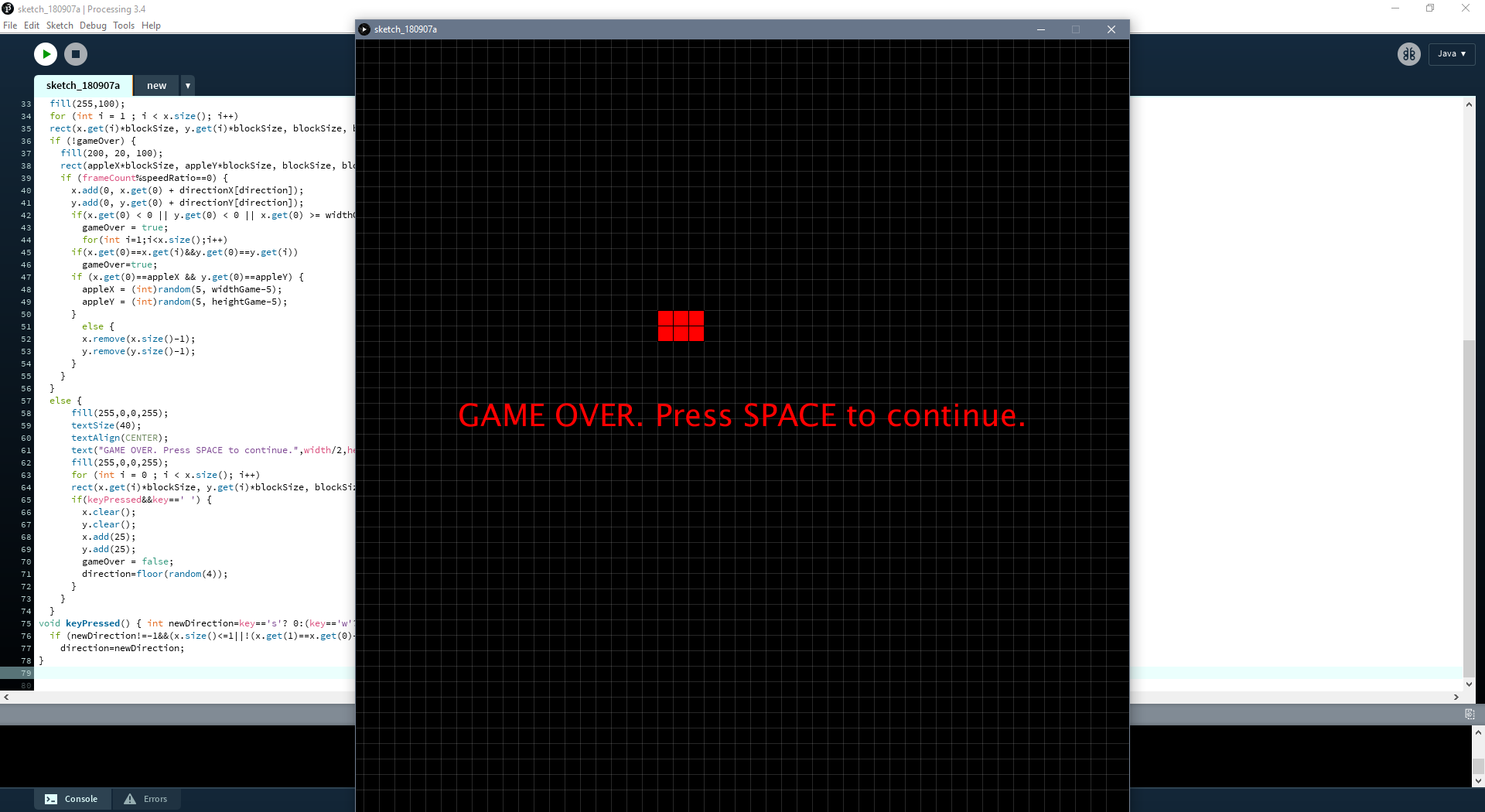
}

}

void keyPressed() { int newDirection=key=='s'? 0:(key=='w'?1:(key=='d'?2:(key=='a'?3:-1)));

if (newDirection!=-1&&(x.size()<=1||!(x.get(1)==x.get(0)+directionX[newDirection]&&y.get(1)==y.get(0)+directionY[newDirection])))

direction=newDirection;

}

Thus, whenever the snake eats itself or goes off the screen, the gameOver constraint toggles and this screen appears. (This particular screen is for the snake eating itself in the given location, though.)

And finally, adding Labels with the score board:

ArrayList<Integer> x = new ArrayList<Integer>(), y = new ArrayList<Integer>();

int widthGame=50;

int heightGame=50;

int blockSize=20;

int direction=floor(random(4));

int appleX=12;

int appleY=10;

int speedRatio=5;

int[]directionX={0,0,1,-1},directionY={1,-1,0,0};

boolean gameOver=false;

int highScore=0;

void setup() {

size(1000, 1000);

x.add(6+(int)random(50));

y.add(6+(int)random(50));

frameRate(60);

rectMode(CENTER);

textAlign(CENTER, CENTER);

}

void draw() {

background(0,50);

drawLabels();

strokeWeight(1);

for (int i=0;i<widthGame;i++) {

stroke(200,50);

line((i-0.5)\*blockSize,0,(i-0.5)\*blockSize,height);}

for(int i=0;i<heightGame;i++){

stroke(200,50);

line(0,(i-0.5)\*blockSize,width,(i-0.5)\*blockSize);}

stroke(0);

fill(255,0,0,100);

rect(x.get(0)\*blockSize, y.get(0)\*blockSize, blockSize, blockSize);

fill(255,100);

for (int i = 1 ; i < x.size(); i++)

rect(x.get(i)\*blockSize, y.get(i)\*blockSize, blockSize, blockSize);

if (!gameOver) {

fill(200, 20, 100);

rect(appleX\*blockSize, appleY\*blockSize, blockSize, blockSize);

if (frameCount%speedRatio==0) {

x.add(0, x.get(0) + directionX[direction]);

y.add(0, y.get(0) + directionY[direction]);

if(x.get(0) < 0 || y.get(0) < 0 || x.get(0) >= widthGame || y.get(0) >= heightGame)

gameOver = true;

for(int i=1;i<x.size();i++)

if(x.get(0)==x.get(i)&&y.get(0)==y.get(i))

gameOver=true;

if (x.get(0)==appleX && y.get(0)==appleY) {

appleX = (int)random(5, widthGame-5);

appleY = (int)random(5, heightGame-5);

}

else {

x.remove(x.size()-1);

y.remove(y.size()-1);

}

}

}

else {

fill(255,0,0,255);

textSize(40);

textAlign(CENTER);

text("GAME OVER. Press SPACE to continue.",width/2,height/2);

fill(255,0,0,255);

for (int i = 0 ; i < x.size(); i++)

rect(x.get(i)\*blockSize, y.get(i)\*blockSize, blockSize, blockSize);

if(keyPressed&&key==' ') {

x.clear();

y.clear();

x.add(25);

y.add(25);

gameOver = false;

direction=floor(random(4));

}

}

}

void keyPressed() { int newDirection=key=='s'? 0:(key=='w'?1:(key=='d'?2:(key=='a'?3:-1)));

if (newDirection!=-1&&(x.size()<=1||!(x.get(1)==x.get(0)+directionX[newDirection]&&y.get(1)==y.get(0)+directionY[newDirection])))

direction=newDirection;

}

void drawLabels(){

fill(150);

textSize(65);

textAlign(RIGHT);

text( "Snake Game", width/1.2, 80);

fill(150);

textSize(20);

text( "ABHIRUP CHAKRAVARTY", width/1.2, 140);

strokeWeight(10);

stroke(200);

fill(255);

rect(90, 70, 400, 150);

fill(0);

textAlign(CENTER);

textSize(17);

text( "Score: " + x.size(), 70, 60);

if (x.size()>highScore)

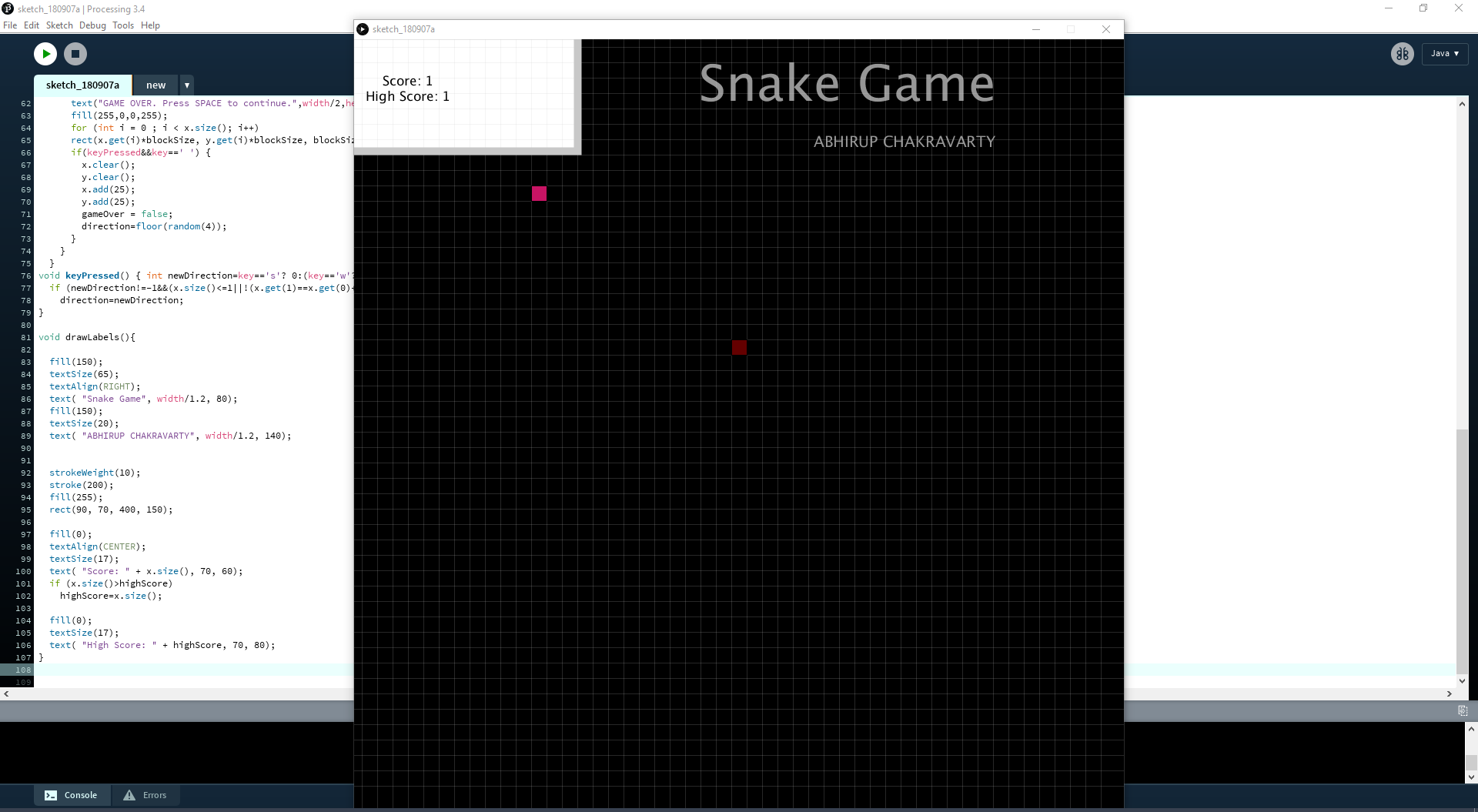
highScore=x.size();

fill(0);

textSize(17);

text( "High Score: " + highScore, 70, 80);

}



Now in this design, the initial score is given off as the total length (inclusive of the head). If you want it to be initialized as 0, then all you have to do is subtract the score with 1. Also, the snake can travel throughout the screen, inclusive over and through the labels and the score board. If you want to change that, all you have to do is redefine the borers and set thicker demarcations to signify the border.

# ACKNOWLEDGEMENT

I am grateful to Sibi Chakkaravarthy Sethur, Department of Computer Science and

Engineering in Vellore Institute of Technology- Andhra Pradesh (VIT-AP) University,

Amaravati for his valuable support.

I am also grateful to Hari Seetha, Head, Department of Computer Science and

Engineering in Vellore Institute of Technology- Andhra Pradesh (VIT-AP) University,

Amaravati for her valuable support.

# AUTHOR

I, Abhirup Chakravarty, am a second-year student of Department of Computer Science and

Engineering in Vellore Institute of Technology- Andhra Pradesh (VIT-AP) University,

Amaravati. My present interests include Machine Learning, 2D game designing as well as cryptography. If you ever wish to reach out to me, this is my official email address:

abhirup.chakravarty@vitap.ac.in