Name:– Amruta S. Bagde

Email:– amrutabagde7@gmail.com

Task – 1: Snake Game in Java

Table of Contents

Acknowledgement

I would like to express my sincere gratitude to Veritech Software Services for providing me with the opportunity to work on this project and gain valuable hands–on experience in Java development.

Introduction

Our project is an exciting Java game based on the classic "Snake." The game first appeared in arcades in the late 1970s and later became famous on Nokia phones in the late 1990s. In this game, players control a snake moving around a grid. The goal is to eat dots to make the snake longer while avoiding crashing into the snake's body or the grid's edges. As the snake gets longer, it becomes harder to maneuver. Despite its simple design, the Snake Game continues to be popular because it's easy to play and offers a lot of fun.

Our aim with this project is to recreate the magic of the Snake Game using Java programming. We'll use Processing GUI to enhance the graphics and interactive elements. By combining Java and Processing, we hope to create an engaging gaming experience that honors the classic while introducing it to new players. We're excited to share this journey with you as we explore the timeless appeal of retro gaming and inspire creativity in modern game development. Join us as we dive into the pixelated world of the Snake Game!

List the key features of the Snake Game project.

- Snake Movement
- Food Generation
- Scoring System
- Game Over Condition
- Speed Variation

Explain that the game is implemented using the Processing programming language, which provides a simple and intuitive way to create interactive graphics. Describe the main components of the game, including the snake (represented by a series of rectangles), food items, scoring system, and game over conditions.

Most of the games doesn't allow you to change from Right to Left or UP to Down. In one steo, but this is allowed. it is not the world's greatest game, but it does give you an idea of what you can achieve with a relatively simple java program and build a GUI with the help of "Processing". Which to extend the principles and create more interesting games of your own

Technologies

JAVA:–

- Java is a popular programming language developed by Sun Microsystems (now owned by Oracle Corporation) in 1995.
- It's high–level and object–oriented, designed to be easy to use and platform–independent.
- One of Java's key features is its platform independence, thanks to the Java Virtual Machine (JVM).
- This means Java programs can run on any device with a JVM installed, making it versatile and accessible.
- Java is widely used for building web applications, with millions of Java apps in use today.
- It's suitable for various devices and applications, including mobile apps, business software, and server–side technologies.
- Java is known for its speed, security, and reliability, making it ideal for handling large–scale projects and managing servers.
- It offers safety features like automatic memory management and strong typing.
- Java provides GUI frameworks like Swing, JavaFX, and Processing for creating user interfaces.
- These frameworks simplify the process of designing and implementing graphical elements in games and applications.
- The Snake Game, a classic arcade game, can be developed using Java with GUI frameworks like Processing.
- Java's platform independence, rich standard library, and support for object–oriented programming make it suitable for creating games like Snake.

Overall, Java's versatility, ease of use, platform independence, and robust features make it a preferred choice for a wide range of software development projects, including web applications and games like the Snake Game.

Processing IDE

Processing, created by Ben Fry and Casey Reas, teaches coding in a fun, visual way. It has its own easy–to–use language based on Java. Processing is a graphical library and a development environment (IDE). It provides a graphical interface that can be used to draw different shapes andd text. The Processing IDE is where you write, run, and fix your "sketches," which are like mini programs. These sketches have setup() and draw() functions for starting and updating.

You can draw shapes, pictures, and text using Processing's built–in functions. It works on Windows, Mac, and Linux, so anyone can use it. ArrayList cann be used to create a dynamically sized list and Pvector specifies a vector, and provides menthods to calculate the distance between vectors. Plus, there's a big community sharing projects and tutorials online. Overall, Processing is a great tool for making art, interactive projects, and learning to code visually.

Goals in creation of JAVA

The creation of the Java language aimed to achieve five key objectives:

- Ease of Use and Object–Oriented Design:
- Reliability and Safety:
- Cross–Platform Compatibility and Portability:
- Optimized Performance:
- Dynamic and Interpreted Execution with Threading Support:

System Specifications

Software Specifications:–

Operating System:–Linux
Coding Language:– JAVA
GUI Library:– Processing IDE

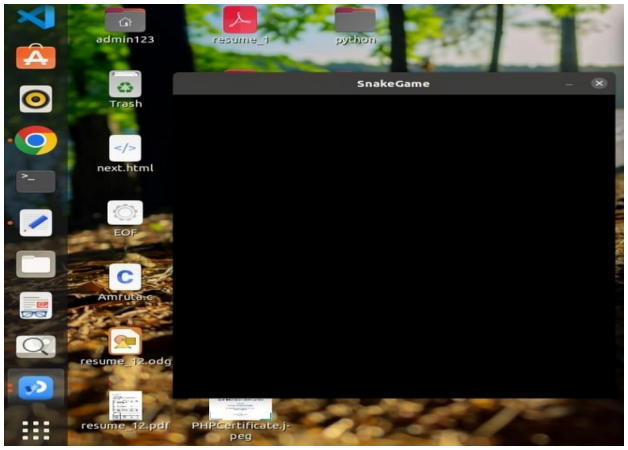
Hardware Specification:–

System:–Intel core i5
RAM:–7.7GB

Functionality

Creating Our Widnow

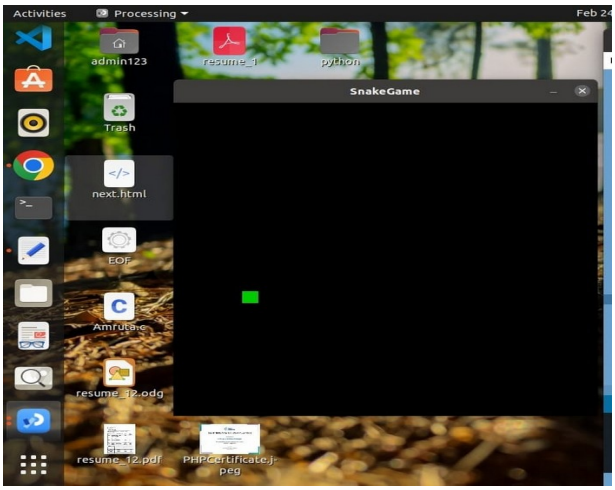
On that page, I'll demonstrate how to set up or create a window panel.




```
void setup()
{
  size(500, 500);
  background(0);
}
```

creating snake

On that page, I'll show how we create a snake, displayed as a rectangles on a grid by using using the draw() method.




```
ArrayList<Integer>x_pos= new ArrayList<Integer>();
ArrayList<Integer>y_pos= new ArrayList<Integer>();
int hgt= 24, wdt= 24;                  //windo
int block= 20;
void setup()
{
  x_pos.add(4);          //Initital Position
  y_pos.add(15);
}
void draw()
{

 fill(0, 204, 0);
 for(int i= 0; i<x_pos.size(); i+ + )
 rect(x_pos.get(i)*block, y_pos.get(i)*block, block, block);
}
```

Moving our Snake

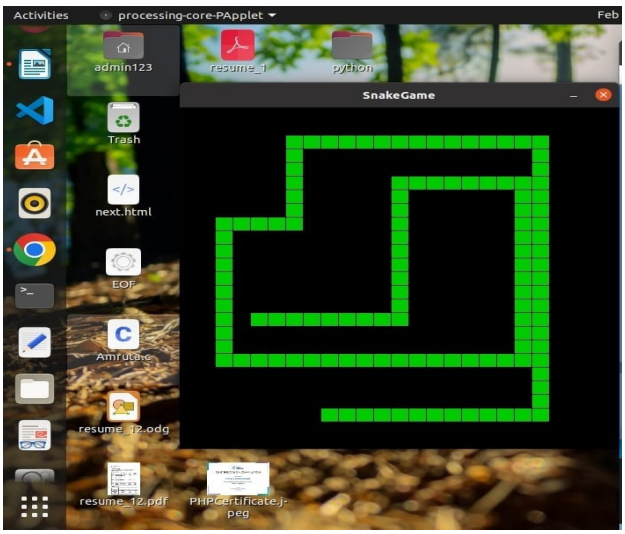
On that page, to explain how to move the snake within the window panel to eat food in any random direction:

```
int dir= 2;
int[]x_dir= {0, 0, 1, −1};                    //down, up, right, left
int[]y_dir= {1, −1, 0, 0};

if(frameCount % 8 = = 0)                //only one element is present
{
        x_pos.add(0, x_pos.get(0)+ x_dir[dir]);
        y_pos.add(0, y_pos.get(0)+ y_dir[dir]);
```

```
        //x_pos.remove(x_pos.size()-1);
        //y_pos.remove(y_pos.size()-1);
}
```

Controlling our Snake

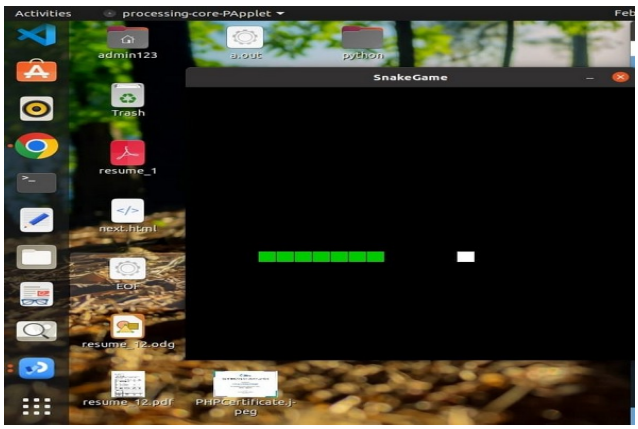
On that page, to enable keyboard input for directing the snake (up, down, left, right), you'll need to




```
void keyPressed()
 {
  int new_dir =  keyCode;
  if(keyCode= = DOWN)
       new_dir= 0;
  else if(keyCode= = UP)
       new_dir= 1;
  else if(keyCode= = LEFT)
       new_dir= 3;
  else if(keyCode= = RIGHT)
       new_dir= 2;
  else
       new_dir= -1;
   if(new_dir!= -1)
        dir= new_dir;
```

Creating Food

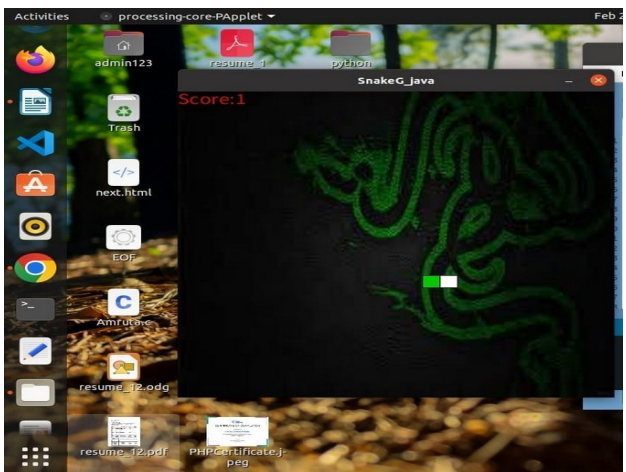
On that page, we will show how we can create a food block, and displayed as rectangles on a grid (located at random positions). and ensuring they do not appear on top of the snake.




```
boolean gamestatus =  false;
if(!gamestatus)
{
            fill(255);
            rect(f_x_pos*block, f_y_pos*block, block, block);
 }
```

Making Snake Eat Food

on that page, we will show When the snake's head intersects with the food's position, it means the snake has successfully consumed the food.



```
if (x_pos.get(0)= = f_x_pos && y_pos.get(0)= = f_y_pos)
{

        f_x_pos= (int)random(0,wdt);
        f_y_pos= (int)random(0,hgt);

 }
```

Displaying Score and Increasing Speed

On that page, we'll show the score displayed using the text() function in Processing. Additionally, the snake's speed increases as it grows longer.

```
textAlign(LEFT);
      textSize(25);                          //Show score
      fill(222,9,12);
      text("Score:"+  x_pos.size(),0,20);

f(x_pos.size()%2= = 0 && speed>= 2)
        {                                          //
          speed= speed−1;
          f_x_pos= (int)random(0,wdt);
          f_y_pos= (int)random(0,hgt);
        }
        else
        {
          f_x_pos= (int)random(0,wdt);
          f_y_pos= (int)random(0,hgt)
}
```

Dying Conditions

In a typical snake game, there are several conditions but we use only two dying condition(Collision with Walls and Collision with Itself) that can lead to the game ending.

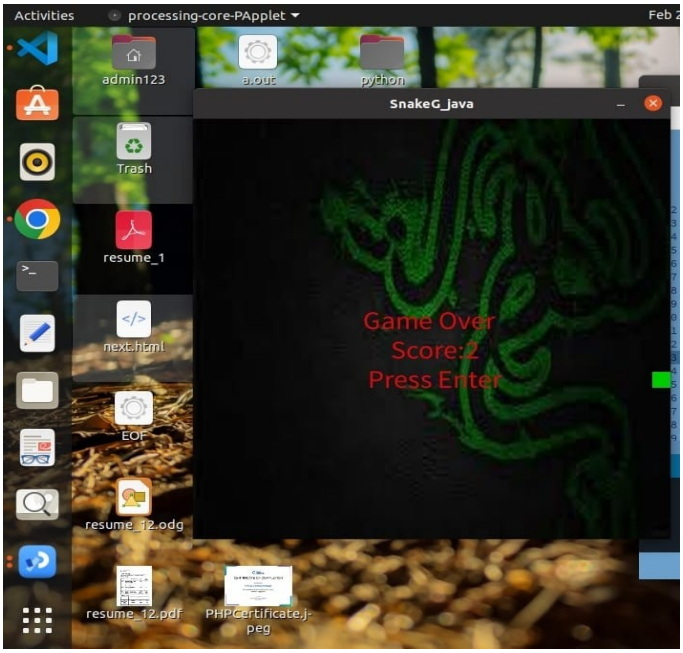
```
if(x_pos.get(0)<0 || y_pos.get(0)<0 || x_pos.get(0)>wdt ||
y_pos.get(0)>hgt)
{
        gamestatus= true;
}
else
{
  for(int j= 1; j<x_pos.size();j+ + )
```

```
  {
     if(x_pos.get(0)= = x_pos.get(j) && y_pos.get(0)= = y_pos.get(j))
     {
                gamestatus= true;
      }
   }
}
```

Our Snake game is now complete and proudly presents the final score attained by players.

```
ArrayList<Integer>x_pos= new ArrayList<Integer>();
ArrayList<Integer>y_pos= new ArrayList<Integer>();
int hgt= 24,wdt= 24;
int block= 20;
int dir= 2;
int[]x_dir= {0,0,1,−1};
int[]y_dir= {1,−1,0,0};
int f_x_pos= 15;
int f_y_pos= 15;
int speed= 10;
boolean gamestatus =  false;
PImage backgroundImage;

void setup()
{
   size(500,500);
   backgroundImage =  loadImage("images.jpeg");  // Load background image
   x_pos.add(4);
   y_pos.add(15);
}
void draw()
{
  image(backgroundImage,0,0,width,height);
  fill(0,204,0);
  for(int i= 0;i<x_pos.size();i+ + )
    rect(x_pos.get(i)*block,y_pos.get(i)*block,block,block);

  if(!gamestatus)
  {
     fill(255);
     rect(f_x_pos*block,f_y_pos*block,block,block);
     textAlign(LEFT);
     textSize(25);
     fill(222,9,12);
     text("Score:"+  x_pos.size(),0,20);

     if(frameCount % 8 = = 0)
     {
        x_pos.add(0,x_pos.get(0)+ x_dir[dir]);
        y_pos.add(0,y_pos.get(0)+ y_dir[dir]);
        if(x_pos.get(0)<0 || y_pos.get(0)<0 || x_pos.get(0)>wdt || y_pos.get(0)>hgt)
```

```
        {
          gamestatus= true;
        }
        else
        {
          for(int j= 1; j<x_pos.size();j+ + )
          {
            if(x_pos.get(0)= = x_pos.get(j) && y_pos.get(0)= = y_pos.get(j))
            {
                gamestatus= true;
            }
          }
        }
      if (x_pos.get(0)= = f_x_pos && y_pos.get(0)= = f_y_pos)
      {
        if(x_pos.size()%2= = 0 && speed>= 2)
        {
          speed= speed−1;
          f_x_pos= (int)random(0,wdt);
          f_y_pos= (int)random(0,hgt);

        }
        else
        {
          f_x_pos= (int)random(0,wdt);
          f_y_pos= (int)random(0,hgt);
        }
      }
        else
        {
         x_pos.remove(x_pos.size()−1);
         y_pos.remove(y_pos.size()−1);
        }
  }
}
      if(gamestatus= = true)
      {
         fill(222,9,12);
         textAlign(CENTER);
         textSize(30);
         text("Game Over \n Score:" +  x_pos.size()+ "\n Press Enter",500/2,500/2);
         if(keyCode = =  ENTER)
        {
          x_pos.clear();
          y_pos.clear();
          x_pos.add(15);
          y_pos.add(15);
          dir= 2;
          speed= 10;
          gamestatus =  false;
        }
      }
  }
```



```
 void keyPressed()
 {
  int new_dir =  keyCode;
  if(keyCode= = DOWN)
       new_dir= 0;
  else if(keyCode= = UP)
       new_dir= 1;
  else if(keyCode= = LEFT)
       new_dir= 3;
  else if(keyCode= = RIGHT)
       new_dir= 2;
  else
       new_dir= −1;
   if(new_dir!= −1)
        dir= new_dir;
}
```

Future Scope of the Project

- In the future, our Snake Game could get even cooler! We're thinking about adding stuff like power−ups and multiplayer mode so you can play with friends.
- you'll be able to customize your game with different looks and sounds. We'll also make sure it works smoothly on phones and lets you share your scores online.
- we might even bring it to other devices like consoles or VR headsets.
- It's all about making the game more fun and accessible for everyone!

Conclusion

Summarize the Snake Game project and its significance. Highlight the engaging and interactive experience it provides for players. Emphasize the simplicity of the mechanics and the challenge of the gameplay. Conclude by expressing the enjoyment and entertainment the Snake Game offers to players of all ages.