

# **Snake Game**

**Project report in partial fulfillment of the requirement for the award of the degree of  
Bachelor of Technology**

**In  
COMPUTER SCIENCE**

**Submitted By**

Simran Singh	University Roll No. 12019009002048
Sneha Agrawal	University Roll No. 12019009001154
Ahana Sen	University Roll No. 12019009001406
Ankush Khanra	University Roll No. 12019009022027
Snehasish Ghosh	University Roll No. 12019009002157
Siddhant Kumar	University Roll No. 12019009022129

**Under the guidance of**

Prof. (Dr.) Rajendrani Mukherjee & Prof. Suseta Datta

Department of Computer Science



**UNIVERSITY OF ENGINEERING & MANAGEMENT, KOLKATA**

University Area, Plot No. III – B/5, New Town, Action Area – III, Kolkata – 700160.

---

## CERTIFICATE

This is to certify that the project titled **Snake Game** submitted by **Simran Singh**(University Roll No. **12019009002048**),**Sneha Agrawal**(University Roll No. **12019009001154**),**Ahana Sen**(University Roll No. **12019009001406**),**Ankush Khanra** (University Roll No. **12019009022027**),**Snehasish Ghosh**(University Roll No. **12019009002157**) and **Siddhant Kumar**(University Roll No. **12019009022129**), students of UNIVERSITY OF ENGINEERING & MANAGEMENT, KOLKATA, in partial fulfillment of requirement for the degree of Bachelor of Computer Science, is a bonafide work carried out by them under the supervision and guidance of Prof. (Dr.) Rajendrani Mukherjee & Prof. Suseta Dutta during 5th Semester of academic session of 2020 - 2021. The content of this report has not been submitted to any other university or institute. I am glad to inform that the work is entirely original and its performance is found to be quite satisfactory.

---

Prof. (Dr.) Rajendrani Mukherjee  
& Prof. Suseta Dutta

Assistant Professor

Department of Computer Science

UEM, Kolkata

---

Prof. Sukalyan Goswami

Head of the Department

Department of Computer Science

UEM, Kolkata

## **ACKNOWLEDGEMENT**

We would like to take this opportunity to thank everyone whose cooperation and encouragement throughout the ongoing course of this project remains invaluable to us.

We are sincerely grateful to our guide Prof. (Dr.) Rajendrani Mukherjee & Prof. Suseta Dutta of the Department of Computer Science, UEM, Kolkata, for their wisdom, guidance and inspiration that helped us to go through with this project and take it to where it stands now.

We would also like to express our sincere gratitude to Prof. Sukalyan Goswami, HOD, Computer Science, UEM, Kolkata and all other departmental faculties for their ever-present assistance and encouragement.

Last but not the least, we would like to extend our warm regards to our families and peers who have kept supporting us and always had faith in our work.

Simran Singh

Sneha Agrawal

Ahana Sen

Ankush Khanra

Snehasish Ghosh

Siddhant Kumar

# TABLE OF CONTENTS

<b>ABSTRACT .....</b>	<b>6</b>
<b>CHAPTER – 1: INTRODUCTION .....</b>	<b>7</b>
<b>CHAPTER – 2: LITERATURE SURVEY .....</b>	<b>8-10</b>
<b>CHAPTER – 3: PROBLEM STATEMENT.....</b>	<b>11</b>
<b>CHAPTER – 4: PROPOSED SOLUTION.....</b>	<b>11-19</b>
<b>CHAPTER – 5: EXPERIMENTAL SETUP AND RESULT ANALYSIS ....</b>	<b>20-24</b>
<b>Experimental Setup.....</b>	<b>20</b>
<b>Result Analysis .....</b>	<b>21-24</b>
<b>CHAPTER – 6: CONCLUSION &amp; FUTURE SCOPE .....</b>	<b>25</b>
<b>Conclusion .....</b>	<b>25</b>
<b>Future Scope.....</b>	<b>25</b>
<b>BIBLIOGRAPHY .....</b>	<b>25</b>

---



## ***ABSTRACT***

*Snake is a computer action game in which you control a snake to move around a map and collect food. We develop a controller based on the arrow keys of our keyboard in this paper. The controller is responsible for deciding the moving direction of the snake. Simply speaking, it has to choose one from three possible actions: go straight, turn left, or turn right. These functions' scores are combined using a linear weighted sum, and the snake takes the action that results in the highest score. The results of our experiments show that our design method is capable of producing smart controllers.*

*This initiative intends to bring the fun and simplicity of the snake game to the masses. This project looks into adding a new dimension to the standard snake game in order to make it more engaging and hard. Because of its simplicity, this game is an excellent contender for a small project.*

# 1. INTRODUCTION

Snake's coding was incredibly challenging, with numerous faults occurring. Many systems had to be written in a variety of methods before a final functioning solution was discovered. Prior to the final version, for example, two alternative movement methods were employed; however, even the final version is faulty since vertical movement causes the snake's scale to shift. There were additional food-related difficulties, such as snake collision detection. While the final design produced a snake capable of eating food, the movement flaw allowed the food to cause additional size concerns.

Despite the fact that the game could not be properly played because no score could be given, it is nonetheless enjoyable. The snake listens to user input and moves across the screen as commanded, with the exception of a size problem while turning. If I had more time to work on this, the collision detection with movement would be the first thing I would solve. By resolving this issue, all other pieces of code that are presently not working would be able to run. The leader board would function since accurate scores would be entered, and the snake would expand because the food would cause it to increase by one number rather than various values based on direction. Furthermore, if the movement is fixed, the snake will perish when it collides with itself. In its current state, the snake moves as a matrix, making it impossible for it to kill itself because it would be impossible to move in any direction. The failure to build a proper movement system was the game's biggest letdown, since it was the root of all other difficulties.

## Main Objects of the Game:

- a. Snake
  - i. Attributes of Snake Object- Length (Size) → 1 (Initially)
  - ii. Snake starts at the centre of the grid
  - iii. Snake's Length increases as it consumes more food
  - iv. Snake's speed increases:
    - I. With increasing length, the speed also increases proportionately
    - II. Linear Variation
- b. Food
  - i. Single unit increases the length of snake by 1.
  - ii. Each food unit appears at random locations on the grid.
  - iii. Food can even appear on the cells currently occupied by the snake.

## 2. LITERATURE SURVEY

There are several definitions of game. According to Kim (1995), a game is a concrete operation through which a student can experience a new concept before he can recognize it formally. Moreover, Erzoz (2000) says that game is an activity in which the interaction existed among the contestants with certain rules to gain a special objective. Meanwhile, Wright (1983) says that game is kind of activity under a certain rules to reach a certain goal with the element of fun. Silvers (1982) cited in Uberman (1998) says many teachers are enthusiastic about using games as a teaching device, yet they often perceive games as mere times-fillers. He also claims that many teachers often over look the fact that in relaxed atmosphere, real learning takes places, and the students can use the language they have been exposed to and have practiced earlier. However, while discussing game, appropriacy is need to be considered. They must be suitable with the students' level or age, or to the material that is to be introduced or practiced. 15 According to Wright(1983), games help the teacher to create contexts in which the language is useful and meaningful. Furthermore, Erzoz (2000) says that games are highly motivating since they are amusing and at the same time challenging. In addition, as well as Wright, Erzoz also says that the game encourage and increase cooperation. Game can be used at all stages of the progression from controlled to free practice, serving at one end of the range as a memory aid and repetition drill at the other as change to used the language freely and as a means to an end rather than an end in itself. They can also serve as a diagnostic tool for teacher, who can note areas of difficulty and take appropriate remedial action. (Haldfield, 1999).

The Snake variety of games dates back to the arcade game Blockade, developed and published by Gremlin in 1976. In 1978, Atari, Inc. released, as an unofficial port, an early home console version of the Blockade concept, titled Surround. Surround was one of the nine Atari 2600 (VCS) launch titles, and was also sold by Sears under the name Chase. That same year, a similar game was launched for the Bally Astro cade as Checkmate. The first known personal computer version of Snake, titled Worm, was programmed in 1978 by Peter Trefonas of the US on the TRS-80computer, and published by CLOAD magazine in the same year. This was followed shortly afterwards with versions from the same author for the Commodore PET and Apple II computers. A microcomputer port of Hustle was first written by Peter Trefonas in 1979 and published by CLOAD. This was later released by Milton Bradley for the TI-99/4A in 1980. There were several versions of Snake on the BBC Micro. 1982's Snake by Dave Bresnen was different in that the snake was controlled using the left and right arrow keys relative to the direction it was heading in. The snake increases in speed as it gets longer, and there are no "lives", making achieving a high score or reaching higher levels relatively difficult as one mistake means starting from the beginning. An analog joystick-controlled variant of Snake, called Anaconda, was included as a hidden mini game in Time Splitters 2, which featured free rotation



instead of a fixed 4- direction system, and multiple types of food. More recent versions include a Neopets version known as Meerca Chase. Snake can be played on YouTube videos that use the 2010 version of the player. When a video is selected, the user can press the 'left' key, the 'right' key, the 'home' key, or the 'end' key for any period of time before immediately pressing the 'up' key. The game will appear and can be played in the video screen. Until 2012, a version called Old Snake could be played inside Gmail.

Mobile gaming started with Snake in 1997 on the monochrome display of a Nokia 6110. Since then, improvements in hardware and game development have altered the user experience dramatically. An understanding of who currently plays and purchases mobile games is vital when designing and developing mobile games. According to de Boer et al., mobile gamers can be divided into two groups; hardcore gamers and casual gamers, stating that on mobile devices, casual gamers “make up the vast majority of the total gamers”. Graft agrees with this distinction, but notes that there is a dedicated mobile game fan base within the estimated 2 billion cell phone users worldwide. A survey conducted by the Interactive Digital Software Association states that 87% of the people asked said they played games for fun, and 74% said they wanted to be challenged. Jacobs writes on casual gaming: “I think mobile games have come of age,” Bruce Gibson, Research Director at Juniper Research, said. “The casual games sector is going to be the market driver, even though it may not be at the leading edge of mobile games technology. Casual games make most use of the inherent advantages of the mobile platform...” This categorizing of mobile gamers is useful in determining what types of games are successful on mobile devices. Games can be divided into genres, and certain genres are better suited to mobile devices than others. De Boer et al. categorise games into various genres and describe their suitability for mobile devices, specifically with Flash Lite in mind. De Boer et al. found that strategy and logic games, such as Minesweeper, work well on mobile devices, as well as role-playing, sport strategy and management games. Games that fall into the first-person shooter genre, were found to be “almost impossible” to create in Flash Lite. Action games and 2D racing games were said to be technically possible with Flash Lite, but the processing power required to achieve the desired speed was the limiting factor in these types of games. Telephia published the following table, showing the penetration rate of various game types among mobile game players in the U.K. These penetration rates of the various game genres support what De Boer et al. stated in terms of strategy and logic games being successful. However, arcade style games have been more successful, while sports and racing games have been less successful.

### ***Penetration Rate Per Game Type***

<b>Game Type</b>	<b>Penetration Rate</b>
1. Puzzle/Strategy	61%
2. Retro/Arcade	45%
3. Action/Adventure	42%
4. Card/Casino	39%
5. Trivia/Word	32%
6. Sports/Racing	30%

The future of mobile gaming definitely looks bright. According to Gartner the worldwide mobile gaming revenue will grow from \$2.9 billion in 2006 to \$9.6 billion in 2011, and is on target to reach \$4.3 billion in 2007, a 49.9 percent increase from 2006. Gartner accredit this increase mainly to the pervasiveness of mobile phones in many markets and the ease of game play on mobile phones. All sources are in agreement that the future of mobile gaming is spectacularly bright, and will continue to grow. However, the mobile game development community is divided on whether devices will become more powerful, and as Graft suggests, be able to play more complex and demanding 3D games with hardware acceleration, or, as Crooks suggests, whether other limiting factors, such as user input, will see games that rely on captivating game play and user interfaces suitable for cell phones. Morales and Nelson write that most successful mobile games in recent years have been 2D games with simple user controls, but are confident that 3D games will become the predominant type of game on mobile platforms. It was found that certain game genres have achieved more success than others in mobile gaming. This can be attributed mainly to the limited resources available on mobile devices, as well limited user input through the small keys on mobile phones. Games that have achieved success have been easily controllable and relatively slow in nature, e.g. puzzle and strategy games.

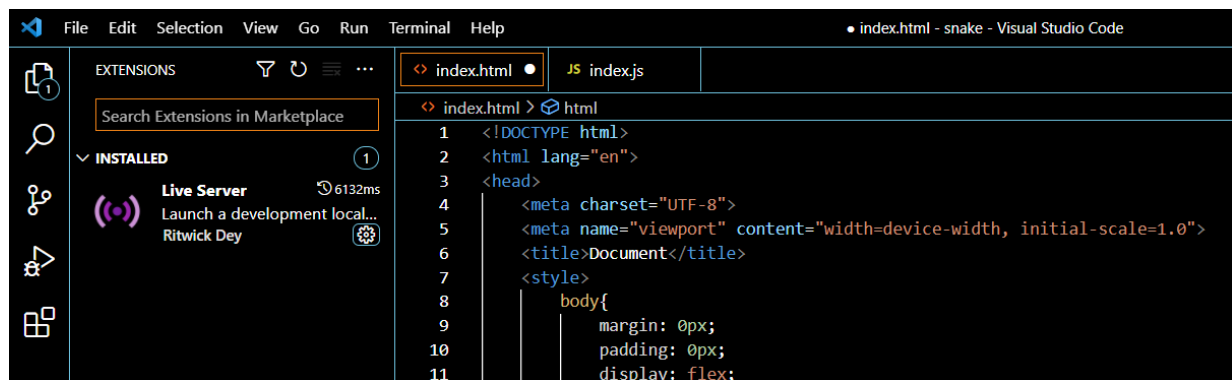
### 3. PROBLEM STATEMENT

This project aims to build a snake game which satisfies following requirements clearly:

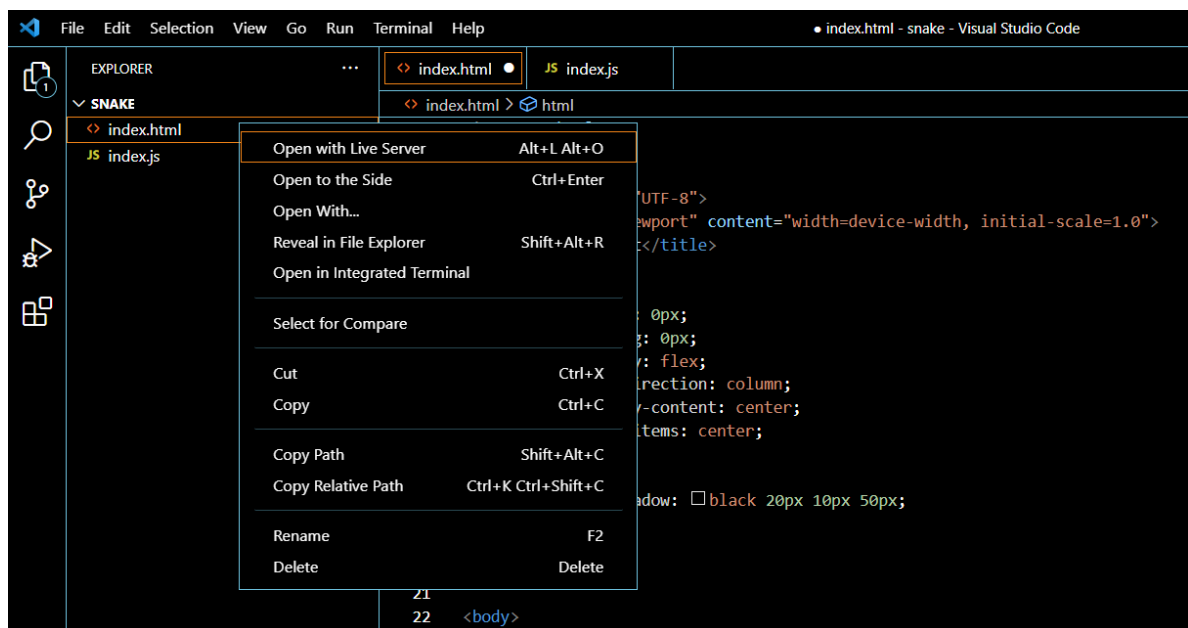
1. Snake moves with the help of arrow buttons.
2. It will die on touching any side of the wall.
3. Speed and size of snake must increase on eating a unit of food, thus increasing difficulty as the game progresses.

### 4. PROPOSED SOLUTION

We have built a Snake Game where the size increases on having food. We have added audio, collision detection, keyboard input and 'Game Over' as well as score display in our game. To present out game visually, we have installed a Live Server in the studio itself.



*Fig 1: From the extensions option, installed the Live Server*



*Fig 2: To present, right click on the index.html*

## Frontend [HTML]

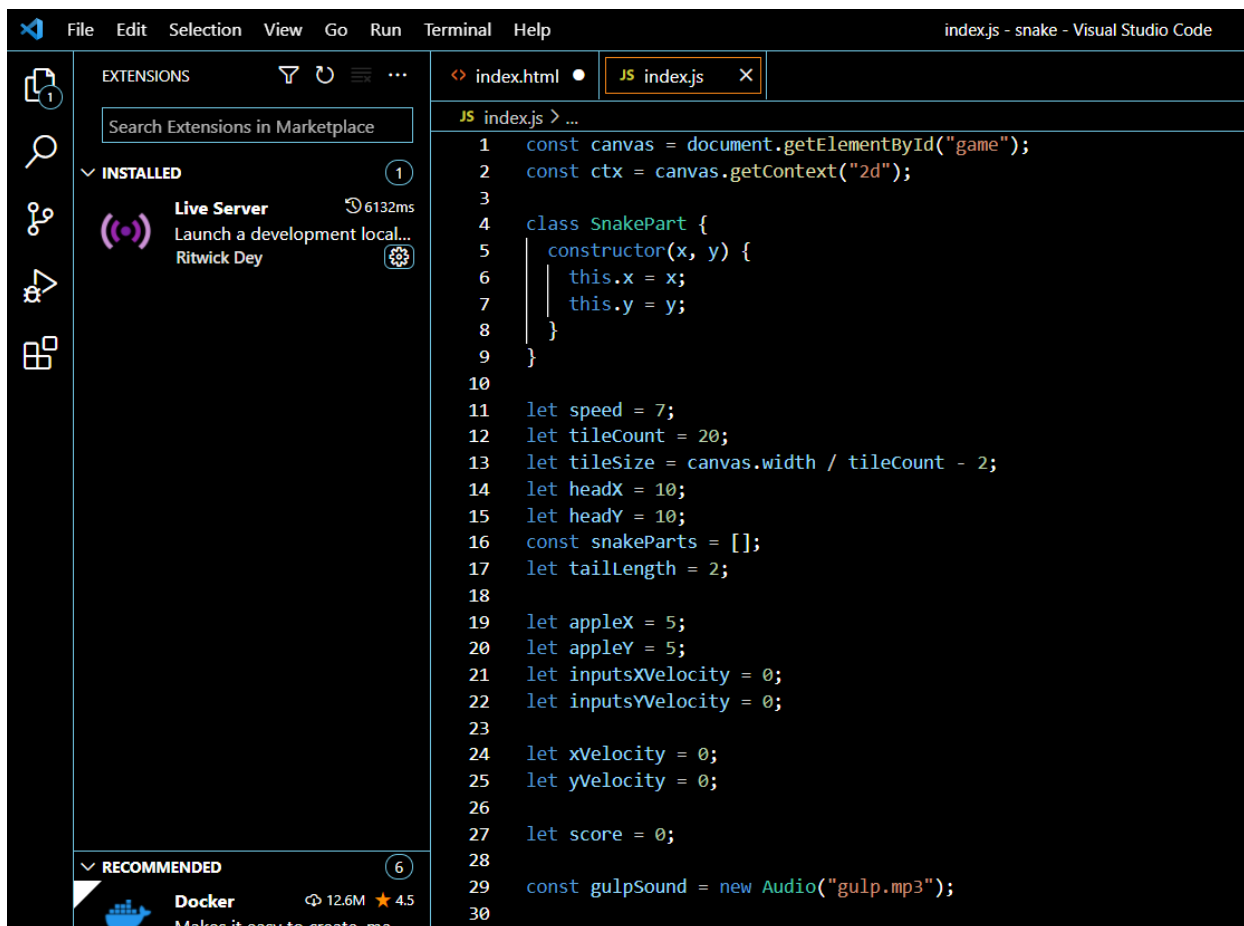
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7   <style>
8     body{
9       margin: 0px;
10      padding: 0px;
11      display: flex;
12      flex-direction: column;
13      justify-content: center;
14      align-items: center;
15    }
16    canvas{
17      box-shadow: 0px 0px 0px 0px;
18    }
19  </style>
20 </head>
21 <body>
22   <h1>Snake Game</h1>
23   <canvas id="game" width="400" height="400" />
24   <script src="index.js"></script>
25 </body>
26 </html>
```

Fig 3: HTML code [snippet from our code]

- `!html`: gives us the base template to run an HTML file in Visual Studio Code
- `<script src="index.js"></script>`: connects the frontend and backend
- `<h1>Snake Game</h1>`: Heading of the Game
- `<canvas>` is an HTML element which can be used to draw graphics via scripting. The CANVAS element allows you to add so much more interactivity to your web pages because now you can control the graphics, images, and text dynamically with a scripting language. The CANVAS element helps you turn images, photos, charts, and graphs into animated elements.
- The HTML `<style>` tag allows us to customize the text that appears on the page. This includes adjusting the font size, font family, font color, and so forth. Not only can we modify the text, but we can also change the style of a page's body or a section of it.
- The `<body>` tag defines the document's body. The `<body>` element contains all the contents of an HTML document, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
- The outer space of an element i.e. margin is the space outside the border. The inner space of an element i.e. padding is space inside the element's border.

- Display used to define the display of the different parts of a web page. Flex is used to set the length of flexible items. It is easy to position child elements and the main container. The margin doesn't collapse with the content margins. The order of any element can be easily changed without editing the HTML section.
- The flex-direction CSS attribute defines how flex elements are positioned in the flex container, including the main axis and direction (normal or reversed).
- The justify-content attribute is used to describe the flexible box container's alignment. It is the space between and surrounding content elements along the main axis of a browser-displayed flex container.
- The align-items attribute specifies the alignment of items within the flexible container or the specified window. It synchronizes the Flex Items throughout the axis. To override the align-items property, use the align-self property.
- The box-shadow CSS property adds shadow effects around an element's frame.

### *Backend [JavaScript]*



```

1  const canvas = document.getElementById("game");
2  const ctx = canvas.getContext("2d");
3
4  class SnakePart {
5    constructor(x, y) {
6      this.x = x;
7      this.y = y;
8    }
9  }
10
11  let speed = 7;
12  let tileCount = 20;
13  let tileSize = canvas.width / tileCount - 2;
14  let headX = 10;
15  let headY = 10;
16  const snakeParts = [];
17  let taillength = 2;
18
19  let appleX = 5;
20  let appleY = 5;
21  let inputsXVelocity = 0;
22  let inputsYVelocity = 0;
23
24  let xVelocity = 0;
25  let yVelocity = 0;
26
27  let score = 0;
28
29  const gulpSound = new Audio("gulp.mp3");
30

```

*Fig 4: Backend Code [snippet from our code]*

- The document method `getElementById()` returns an element object representing the element whose id property matches with the given value. This method is used to alter an element on our document and is commonly used in web design to change the value of any particular element or to obtain a specific element. If the method's given ID does not exist, it returns null.
- Context is linked to an Activity and its lifecycle in `getContext()`. Context can be thought of as a layer that exists behind Activity and will exist as long as Activity does. Context will die when the Activity dies. `getContext(2d)` returns the 2D rendering context for a `<canvas>` element's drawing surface.
- The X and the Y are the components that act as axes. The initial length is defined as 2 and the initial place of apple has been defined. After that the random function has been used.
- In JavaScript, `let` is a keyword that is used to declare a block scoped variable.

```

29 const gulpSound = new Audio("gulp.mp3");
30
31 //game loop
32
33 function drawGame() {
34   xVelocity = inputsXVelocity;
35   yVelocity = inputsYVelocity;
36   changeSnakePosition();
37   let result = isGameOver();
38   if (result) {
39     return;
40   }
41   clearScreen();
42   checkAppleCollision();
43   drawApple();
44   drawSnake();
45   drawScore();
46   if (score > 5) {
47     speed = 9;
48   }
49   if (score > 10) {
50     speed = 11;
51   }
52   setTimeout(drawGame, 1000 / speed);
53 }
54 function isGameOver() {
55   let gameOver = false;
56
57   if (yVelocity === 0 && xVelocity === 0) {
58     return false;
59   }
60
61   //walls
62   if (headX < 0) {

```

Fig 5: Backend Code [snippet from our code]

- We downloaded the gulp sound effect from the internet and then saved the file in mp3 format. Then we saved the sound file in the snake folder and called it within VS code as a separate file extension: gulp.mp3.
- The drawGame() function- All the game functions are defined in this function.
- xVelocity: This is the speed of the snake in the X-Axis.
- yVelocity: This is the speed of the snake in the Y-Axis.
- changeSnakePosition(): This function is called for the implementation of the different points occupied by the snake.
- isGameOver(): Displays 'Game Over' on the screen when the snake hits itself or the walls.
- drawApple(): This function is required to display the food on the screen on any random position.
- drawSnake(): This function is required to display the Snake on the screen on any random position.
- drawScore(): This function is required to display the score on the top-right corner of the screen.
- checkAppleCollision(): Whenever the position of the head of the Snake and the food is same the food changes its position and acquires a new random position.

```

61 //walls
62 if (headX < 0) {
63   gameOver = true;
64 } else if (headX === tileCount) {
65   gameOver = true;
66 } else if (headY < 0) {
67   gameOver = true;
68 } else if (headY === tileCount) {
69   gameOver = true;
70 }
71
72 for (let i = 0; i < snakeParts.length; i++) {
73   let part = snakeParts[i];
74   if (part.x === headX && part.y === headY) {
75     gameOver = true;
76     break;
77   }
78 }
79
80 if (gameOver) {
81   ctx.fillStyle = "white";
82   ctx.font = "50px Verdana";
83   if (gameOver) {
84     ctx.fillStyle = "white";
85     ctx.font = "50px Verdana";
86     var gradient = ctx.createLinearGradient(0, 0, canvas.width, 0);
87     gradient.addColorStop(0, "magenta");
88     gradient.addColorStop(0.5, "blue");
89     gradient.addColorStop(1.0, "red");
90     // Fill with gradient
91     ctx.fillStyle = gradient;
92     ctx.fillText("Game Over!", canvas.width / 6.5, canvas.height / 2);
93   }
94   ctx.fillText("Game Over!", canvas.width / 6.5, canvas.height / 2);
95 }
96 return gameOver;
97 }

```

Fig 6: Backend Code [snippet from our code]

- To increase the speed of the Snake in the game, the if block has been defined.
- Ctx.fillStyle: fillStyle property of the Canvas 2D API specifies the color, gradient, or pattern to use inside shapes. The default style is #000 (black).
- addColorStop: This method specifies the color stops, and its position along the gradient. Gradient positions can be anywhere between 0 to 1. To use the gradient, set the fillStyle or strokeStyle property to the gradient, then draw the shape (rectangle, text, or a line).
- createLinearGradient: The createLinearGradient() method creates a linear gradient object. The gradient can be used to fill rectangles, circles, lines, text, etc. Tip: Use this object as the value to the strokeStyle or fillStyle properties.



```
98 function drawScore() {
99   ctx.fillStyle = "white";
100   ctx.font = "10px Verdana";
101   ctx.fillText("Score " + score, canvas.width - 50, 10);
102 }
103 function clearScreen() {
104   ctx.fillStyle = "black";
105   ctx.fillRect(0, 0, canvas.width, canvas.height);
106 }
107 function drawSnake() {
108   ctx.fillStyle = "green";
109   for (let i = 0; i < snakeParts.length; i++) {
110     let part = snakeParts[i];
111     ctx.fillRect(part.x * tileCount, part.y * tileCount, tileSize, tileSize);
112   }
113   snakeParts.push(new SnakePart(headX, headY)); //put an item at the end of the list next to the head
114   while (snakeParts.length > taillength) {
115     snakeParts.shift(); // remove the furthest item from the snake parts if have more than our tail size.
116   }
117   ctx.fillStyle = "orange";
118   ctx.fillRect(headX * tileCount, headY * tileCount, tileSize, tileSize);
119 }
120 function changeSnakePosition() {
121   headX = headX + xVelocity;
122   headY = headY + yVelocity;
123 }
124 function drawApple() {
125   ctx.fillStyle = "red";
126   ctx.fillRect(appleX * tileCount, appleY * tileCount, tileSize, tileSize);
127 }
128 function checkAppleCollision() {
129   if (appleX === headX && appleY === headY) {
130     appleX = Math.floor(Math.random() * tileCount);
131     appleY = Math.floor(Math.random() * tileCount);
132     taillength++;
133     score++;
134     gulpSound.play();
135   }
```

Fig 7: Backend Code [snippet from our code]

- **fillRect():** The CanvasRenderingContext2D. fillRect() method of the Canvas 2D API draws a rectangle that is filled according to the current fillStyle . This method draws directly to the canvas without modifying the current path, so any subsequent fill() or stroke() calls will have no effect on it.
- **snakeParts.push:** Appends new elements to the end of an array, and returns the new length of the array.
- **snakeParts.shift:** Removes the first element from an array and returns it. If the array is empty, undefined is returned and the array is not modified.
- **gulpSound.play():** Loads and starts playback of a media resource.

```
138 document.body.addEventListener("keydown", keyDown);
139
140 function keyDown(event) {
141     //up
142     if (event.keyCode == 38 || event.keyCode == 87) {
143         //87 is w
144         if (inputsYVelocity == 1) return;
145         inputsYVelocity = -1;
146         inputsXVelocity = 0;
147     }
148
149     //down
150     if (event.keyCode == 40 || event.keyCode == 83) {
151         // 83 is s
152         if (inputsYVelocity == -1) return;
153         inputsYVelocity = 1;
154         inputsXVelocity = 0;
155     }
156
157     //left
158     if (event.keyCode == 37 || event.keyCode == 65) {
159         // 65 is a
160         if (inputsXVelocity == 1) return;
161         inputsYVelocity = 0;
162         inputsXVelocity = -1;
163     }
164
165     //right
166     if (event.keyCode == 39 || event.keyCode == 68) {
167         //68 is d
168         if (inputsXVelocity == -1) return;
169         inputsYVelocity = 0;
170         inputsXVelocity = 1;
171     }
172 }
173
174 drawGame();
```

Fig 8: Backend Code [snippet from our code]

- `HTMLElement.addEventListener<"keydown">(type: "keydown", listener: (this: HTMLElement, ev: KeyboardEvent) => any, options?: boolean | AddEventListenerOptions): void (+1 overload)`
- Appends an event listener for events whose type attribute value is type. The callback argument sets the callback that will be invoked when the event is dispatched.
- The options argument sets listener-specific options. For compatibility this can be a boolean, in which case the method behaves exactly as if the value was specified as options's capture.
- When set to true, options's capture prevents callback from being invoked when the event's eventPhase attribute value is BUBBLING\_PHASE. When false (or not

present), callback will not be invoked when event's eventPhase attribute value is CAPTURING\_PHASE. Either way, callback will be invoked if event's eventPhase attribute value is AT\_TARGET.

- When set to true, options's passive indicates that the callback will not cancel the event by invoking preventDefault(). This is used to enable performance optimizations described in § 2.8 Observing event listeners.
- When set to true, options's once indicates that the callback will only be invoked once after which the event listener will be removed.
- If an AbortSignal is passed for options's signal, then the event listener will be removed when signal is aborted.
- The event listener is appended to target's event listener list and is not appended if it has the same type, callback, and capture.
- keyDown(event): The keydown event is fired when a key is pressed. Unlike the keypress event, the keydown event is fired for all keys, regardless of whether they produce a character value. The keydown and keyup events provide a code indicating which key is pressed, while keypress indicates which character was entered.
- The keycode for left arrow is 37.
- The keycode for up arrow is 38.
- The keycode for right arrow is 39.
- The keycode for down arrow is 40.

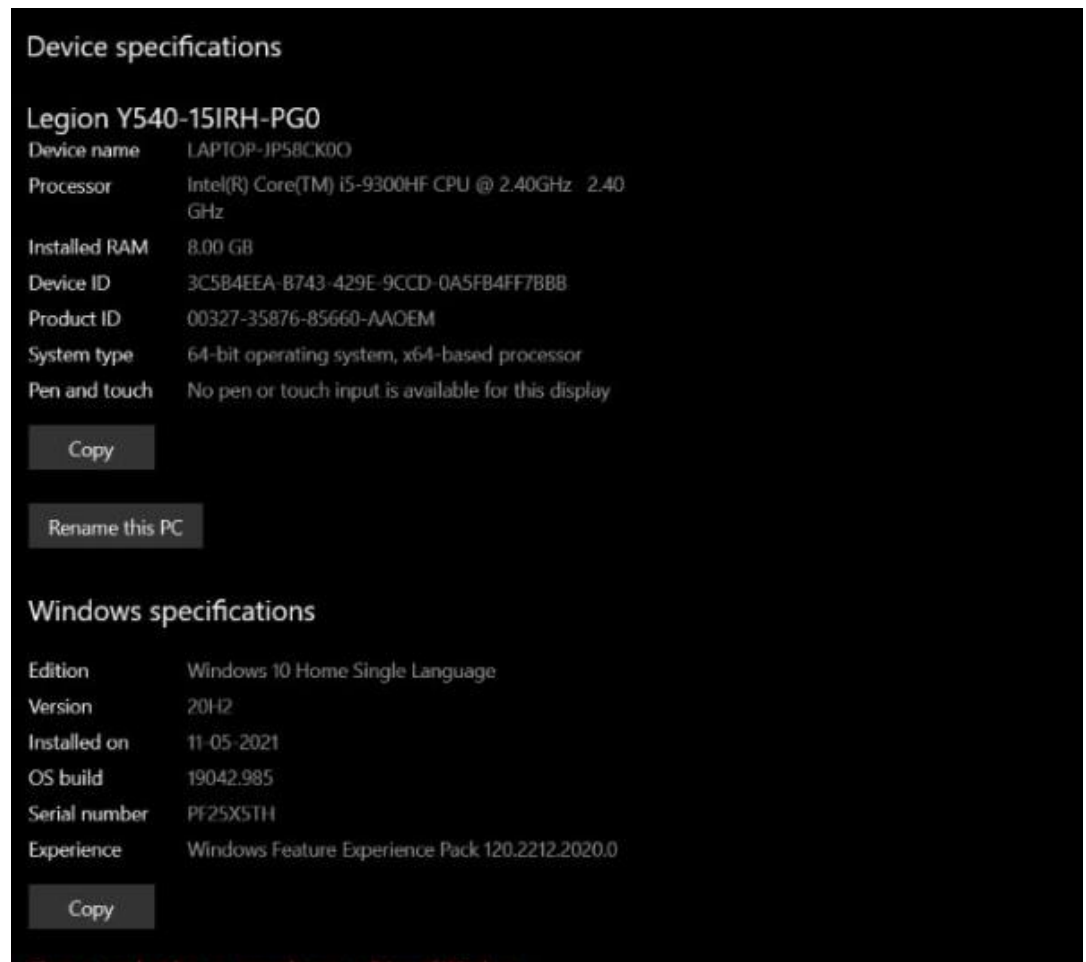
## 5. EXPERIMENTAL SETUP AND RESULTS ANALYSIS

### Experimental Setup

Technologies Used: HTML (Frontend), JavaScript

Platform: Visual Source Code

Hardware:



*Fig 9: Hardware features (where source code has been implemented)*

## Result Analysis

### SNAPSHOTS

#### Start of Game

Snake starts from the centre of the grid with a score of zero.

*Initial Score=0*

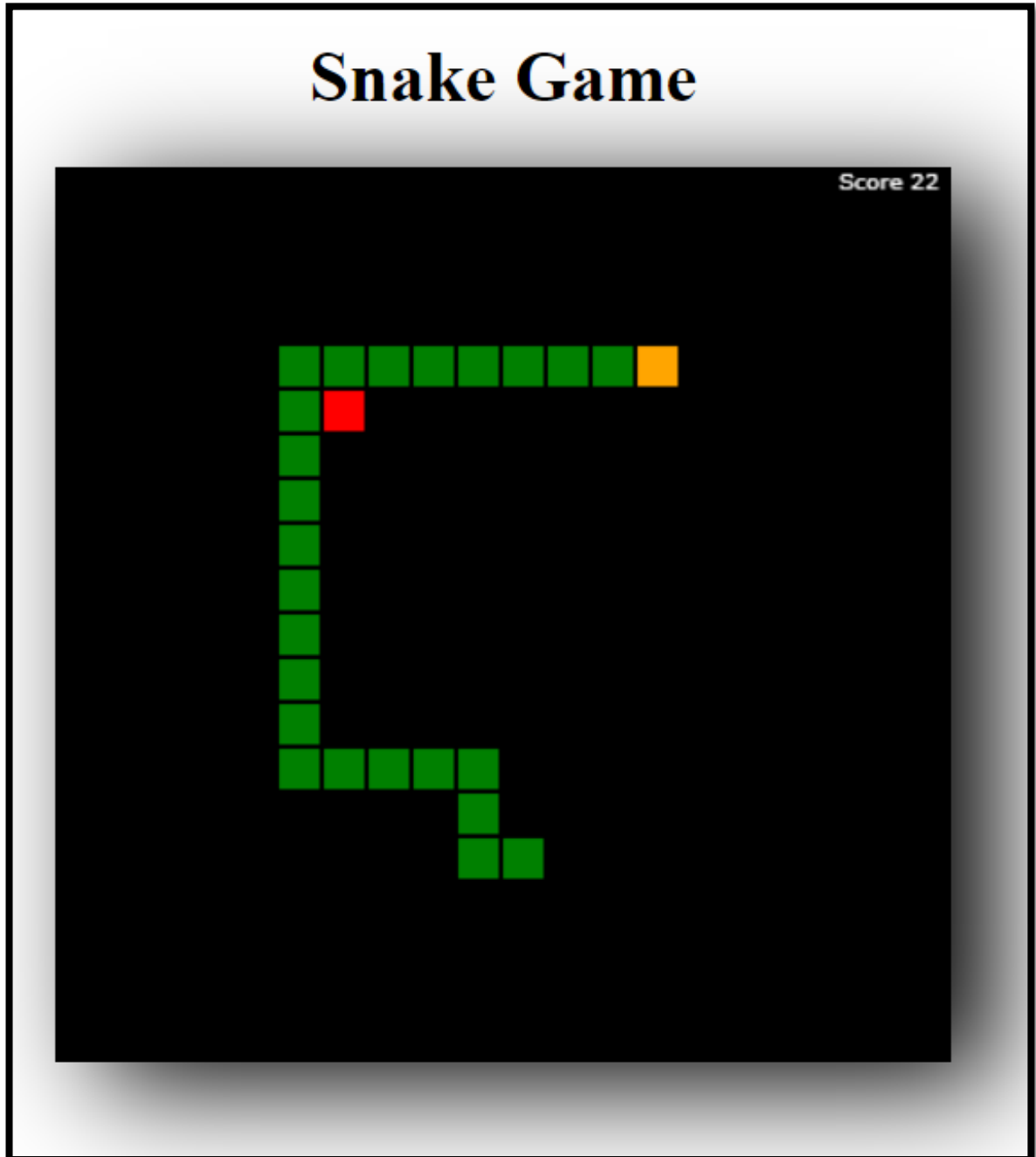


*Fig 10: The screen when we open the game*

### Growth of Snake

As the snake eats more food, it's length and speed increases. Speed has been modeled as a linear function of length or score.

*Score=22*



*Fig 11: While playing the game*

### Game Over

When the snake touches any part of it's own body or any side of the wall, GAME OVER is displayed and the total score is displayed.

*Score=22*



*Fig 12: Touching the wall the game ends*

# Snake Game



*Fig 13: When the snake touches its own body, the game ends*



## 6. CONCLUSION AND FUTURE SCOPE

### a. Conclusion

We were successful in creating a version of traditional snake game. We learned several project management techniques used by professionals to develop large scale project. The experience of working in team and integration of modules developed independently, with just requirement specifications, is a very important achievement for the team.

### b. Future Scope

- The graphics can be made more realistic-- instead of box it can represent an actual snake getting elongated.
- More than one user can play and compete.
- Some bonus objects can be introduced in the game such as speed booster or point multiplier etc.

## BIBLIOGRAPHY

1. Nokia Forum, Flash Lite for S60 - An Emerging Global Ecosystem, May 2006
2. Canalys, 64 million smart phones shipped worldwide in 2006, <http://www.canalys.com/pr/2007/r2007024.htm>, 2007.
3. Crooks II, C. E., Mobile Device Game Development, Charles River Media, Boston, MA, USA, 2007
4. Symbian, Fast Facts, <http://www.symbian.com/about/fastfacts/fastfacts.html>, 2007
5. Graft, K., Analysis: History of Cell Phone Gaming, Business Week Online, [http://www.businessweek.com/innovate/content/jan2006/id20060122\\_077129.htm](http://www.businessweek.com/innovate/content/jan2006/id20060122_077129.htm), 2006.
6. Theesa, Essential Facts About the Computer and Video Game Industry, <http://www.theesa.com/archives/files/Essential%20Facts%202006.pdf>, 2006.
7. Telephia, Puzzle/Strategy and Retro/Arcade Mobile Games are the Most Popular Among U.K. 3G Subscribers, [http://www.telephia.com/html/insights\\_071206.html](http://www.telephia.com/html/insights_071206.html), July 2006.
8. <https://www.hmdglobal.com/press-releases/iconic-snake-game-is-back#:~:text=Snake%20first%20appeared%20in%201997,developed%20and%20published%20by%20Gremlin>.
9. [https://en.wikipedia.org/wiki/Snake\\_\(video\\_game\\_genre\)](https://en.wikipedia.org/wiki/Snake_(video_game_genre))