# PROJECT REPORT

ON

# Shoot OR Die

**SUBMITTED BY:**

SURYANSH VERMA(201500726)

UMANG GUPTA(201500752)

RIPUNJAY YADAV(201500562)

Aryan Choudhary(201500147)

GLA UNIVERSITY ESTD. 2010

ऋते ज्ञानान्न मुक्तिः

GLA UNIVERSITY MATHURA

Recognised by UGC Under Section 2(f)

Accredited with A+ 3.46 Score Grade by NAAC

12-B Status from UGC

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**INSITUTE OF ENGINNERING & TECHNOLOGY**

**GLA UNIVERSITY, MATHURA**

# DECLARATION

I would like to express my special thanks of gratitude to my project guide Dr Manoj Varshney sir who gave me the golden opportunity to do this wonderful project on the topic Simple Shooting Game(Shoot OR Die), which connect us to a friends and I came to know about so many new things I am really thankful to them.

Secondly, I would also like to thank my friends who helped me a lot in finalizing this project within the limited time frame.

**Candidate Names:**

UMANG GUPTA(201500752)
RIPUNJAY YADAV(201500562)
SURYANSH VERMA(201500726)
Aryan Choudhary(201500147)

# CERTIFICATE

This is to certify that the above statements made by the candidate are correct to the best of my/our knowledge and belief.

**Project Supervisor**

Dr Manoj Varshney

Assistant Professor

Date: 24-April-2023

# Table of Content

# INTRODUCTION

Welcome to the world of shooting! In this simple yet addictive game, your goal is to shoot as many dots as possible while avoiding obstacles and trying to beat your high score.

The gameplay is easy to understand - you control a small dot shooter that moves back and forth at the bottom of the screen. Your objective is to shoot the dots that are moving across the screen from the top. Every time you successfully hit a dot, your score increases, and the game becomes more challenging as the dots move faster and obstacles appear on the screen.

As you progress through the levels, you'll encounter various power-ups that can help you in your quest to beat your high score. With its simple and intuitive gameplay, this dot shooting game is perfect for players of all ages and skill levels. So, grab your dot shooter and get ready for some addictive fun!

Dot shooting games are popular and entertaining games that have been around for years. They are simple, easy-to-play games that can be enjoyed by anyone, regardless of age or skill level. The objective of the game is straightforward - you need to shoot the dots that appear on the screen, while avoiding obstacles and enemies that may try to block your path.

One of the main appeals of dot shooting games is their addictive gameplay. As you progress through the levels, the game becomes more challenging, and you need to use your quick reflexes and strategic thinking to overcome the obstacles and continue to advance. Additionally, these games often feature colorful and attractive graphics, and catchy sound effects, which add to the overall excitement of the game.

In recent years, dot shooting games have become increasingly popular on mobile devices, making them accessible to a wider audience than ever before. Many of these games also offer social features, such as leaderboards and multiplayer modes, which allow players to compete with their friends and other players around the world.

Overall, dot shooting games are simple yet fun games that can provide hours of entertainment. So why not give it a try and see how high you can score!

# TECHNOLOGY USED

## HTML -

The **HyperText Markup Language** or **HTML** is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes, and other items. HTML elements are delineated by *tags*, written using angle brackets. HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. The inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.[2] A form of HTML, known as HTML5, is used to display video and audio, primarily using the <**canvas**> element, in collaboration with javascript.

.

## CSS -

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colours, layout, and fonts, thus making our web pages presentable to the users.

CSS is designed to make style sheets for the web. It is independent of HTML and can be used with any XML-based markup language. Now let's try to break the acronym:

- Cascading: Falling of Styles
- Style: Adding designs/Styling our HTML tags
- Sheets: Writing our style in different documents

## Java Script -

*JavaScript* was initially created to “make web pages alive”.
The programs in this language are called *scripts*. They can be written right in a web page’s HTML and run automatically as the page loads.
Scripts are provided and executed as plain text. They don’t need special preparation or compilation to run.
In this aspect, JavaScript is very different from another language called Java.
Today, JavaScript can execute not only in the browser, but also on the server, or actually on any device that has a special program called the java script.
The browser has an embedded engine sometimes called a “JavaScript virtual machine”.
Different engines have different “code names”. For example:

- V8 – in Chrome, Opera and Edge.
- Spider Monkey – in Firefox.
- There are other code names like “Chakra” for IE, “JavaScript”, “Nitro” and “Squirrel Fish” for Safari, etc.

The terms above are good to remember because they are used in developer articles on the internet. We’ll use them too. For instance, if “a feature X is supported by V8”, then it probably works in Chrome, Opera and Edge.

# SYSTEM REQUIREMENTS

## Software Requirement-

**To build  web application –**

- 64-bit Windows 8/10/11
- VS code (version 1.73)
- XAMPP (version 8.1.10)

## Hardware Requirement –

- x86_64 CPU architecture; 2nd generation Intel Core or newer
- 2 GB RAM or more
- 4 GB of available disk space minimum

# IMPLEMENTATION

## Explanation of Source Code :-

This code is an HTML file with a linked JavaScript file. It creates a simple shooting game in a canvas element using the 2D context.

The HTML file contains a canvas element, a score div, and a menu div. The canvas element is set to fill the window. The score div displays the current score of the game. The menu div is used to display the game menu before the game starts.

The JavaScript file defines three classes, namely Player, Bullet, and Particle. Each class has a constructor method to initialize properties, a draw method to draw the shape in the canvas, and an update method to update the position of the shape.

The Player class represents the player's character, which is drawn as a circle in the canvas.

The Bullet class represents the bullets fired by the player, which are also circles. The bullets have a velocity vector and a speed value to control their movement.

The Particle class represents the particles that appear when a bullet hits an enemy. The particles also have a velocity vector, a speed value, and an alpha value to control their transparency.

The JavaScript file also defines several variables and functions to control the game's behavior. The update function is the main loop of the game, which updates the positions of the player, bullets, enemies, and particles, and checks for collisions.

The game starts when the user clicks the Start Game button in the menu. The menu is then hidden, and the game canvas and score div are displayed. The user controls the player using the mouse, clicking to fire bullets at the enemies. The game ends when the player is hit by an enemy or the time runs out, and the final score is displayed in the menu.

## Final Code :-

### Index.html:

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF">
        <meta http-equiv="X-UA-Campatible" content="IE-edge">
        <meta name="viewpoint" content="width=device-width, initial-
scale=1.0">
        <title>Shoot OR Die</title>
        <style>
            body{
                margin: 0px;
                overflow:hidden;
                font-family: Verdana, Geneva, Tahoma, sans-serif;
                background-color: black;
            }
            #score{
                color: white;
                position:fixed;
                padding: 5px;
                font-size: small;
                user-select: none;
            }
            #menu{
                position: fixed;
                top:0;
                bottom: 0;
                left: 0;
                right: 0;
                display: flex;
                justify-content: center;
                align-items: center;
                user-select: none;

            }
            #menu div{
                width: 400px;
                background-color: white;
                text-align: center;
                padding: 25px;
                border-radius: 2px;
            }
            #menu div h1,h4,button{
                margin:0px;
                margin: 5px auto;
            }
            #startgame_btn{
                padding: 10px;
                background-color: cornflowerblue;
                border: none;
                border-radius: 5px;
                width: 80%;
                cursor: pointer;
                font-weight: bold;
            }
            #startgame_btn:hover{
                opacity: 0.7;
            }
            h1{
                font-size: 55px;
            }
        </style>
    </head>
    <body>

        <div id="score">Score : <span id ="lscore">0</span></div>
        <div id="menu">
            <div>
            <h1 id="finalscore">0</h1>
            <h4>Points</h4>
            <button id="startgame_btn">Start Game</button>
        </div>
        </div>
        <canvas></canvas>
<script src="logic.js"></script>
    </body>
</html>
```

**logic.js:**

```
const canvas = document.querySelector('canvas')
const scoreboard = document.querySelector('#lscore')
const menu = document.querySelector('#menu')
const finalscore = document.querySelector('#finalscore')
const startbtn = document.querySelector('#startgame_btn')
canvas.width = window.innerWidth
canvas.height = window.innerHeight
const c = canvas.getContext('2d')

class Player{
    constructor(x,y,radius,color){
        this.x = x;
        this.y = y;
```

```
       this.radius = radius;
       this.color = color;
   }
   draw()
   {
       c.beginPath();
       c.arc(this.x,this.y,this.radius,0,Math.PI*2,false)
       c.fillStyle = this.color;
       c.fill();
   }

}
class Bullet{
   constructor(x,y,radius,color,velocity,speed){
       this.x = x;
       this.y = y;
       this.radius = radius;
       this.color = color;
       this.velocity = velocity;
       this.speed = speed;
   }
   draw()
   {
       c.beginPath();
       c.arc(this.x,this.y,this.radius,0,Math.PI*2,false)
       c.fillStyle = this.color;
       c.fill();
   }
   update(){
       this.draw();
       this.x = this.x + this.velocity.x*this.speed
       this.y = this.y + this.velocity.y*this.speed
   }
}
const friction=0.90;
class Particle{
   constructor(x,y,radius,color,velocity,speed,alpha){
       this.x = x;
       this.y = y;
       this.radius = radius;
       this.color = color;
       this.velocity = velocity;
       this.speed = speed;
       this.alpha = 1 ;

   }
   draw()
   {   c.save()
```

```
        c.globalAlpha = this.alpha
        c.beginPath();
        c.arc(this.x,this.y,this.radius,0,Math.PI*2,false)
        c.fillStyle = this.color;
        c.fill();
        c.restore()
    }
    update(){
        this.draw();
        this.velocity.x*=friction;
        this.velocity.y*=friction;
```

```
        this.x = this.x + this.velocity.x*this.speed
        this.y = this.y + this.velocity.y*this.speed
        this.alpha-=0.01
    }
}
```

```
class Enemy{
    constructor(x,y,radius,color,velocity,speed){
        this.x = x;
        this.y = y;
        this.radius = radius;
        this.color = color;
        this.velocity = velocity;
        this.speed = speed;
    }
    draw()
    {
        c.beginPath();
        c.arc(this.x,this.y,this.radius,0,Math.PI*2,false)
        c.fillStyle = this.color;
        c.fill();
    }
    update(){
        this.draw();
        this.x = this.x + this.velocity.x*this.speed
        this.y = this.y + this.velocity.y*this.speed
    }
}
```

```
let score = 0
let enemies = []
let bullets = []
let particles = []
function init(){
    score = 0
```

```
     scoreboard.innerHTML=0
     enemies = []
     bullets = []
     particles = []
 }
```

```
   function spawnEnemies(){
    setInterval(()=>{
        const radius=Math.random()*(38-6)+6

        let x
        let y
        if (Math.random()<0.5) {
            x=Math.random()<0.5?0-radius:canvas.width+radius;
            y=Math.random()*canvas.height;
        }else{
```

```
            y=Math.random()<0.5?0-radius:canvas.height+radius;
            x=Math.random()*canvas.width;
        }

         const color=`rgb(${Math.random()*(200-50)+50},${Math.random()*(200-
50)+50}
         ,${Math.random()*(200-50)+50})`
         const angle = Math.atan2(canvas.height/2-y,canvas.width/2-x)
    const velocity = {
        x:Math.cos(angle),
        y:Math.sin(angle)
    }
         const speed = 1
```

```
         enemies.push(new Enemy(x,y,radius,color,velocity,speed))
    },1000);
   }
```

```


 window.addEventListener("click",(event)=>{
    const angle = Math.atan2(event.clientY-canvas.height/2,event.clientX-
canvas.width/2)
    const velocity = {
        x:Math.cos(angle),
        y:Math.sin(angle)
    }
    const bullet = new
Bullet(canvas.width/2,canvas.height/2,4,'white',velocity,7);
    bullets.push(bullet);
 })
```

```
const x = canvas.width/2;
const y = canvas.height/2;
const player = new Player(x,y,10,'silver')
let animationid
function animate(){
   animationid=requestAnimationFrame(animate)
   c.fillStyle='rgba(0,0,0,0.1)'
   c.fillRect(0,0,canvas.width,canvas.height)
   player.draw();
   particles.forEach((particle,index)=>{
       if(particle.alpha<=0){
           particles.splice(index,1)
       }else{
           particle.update();
       }

   })
   bullets.forEach((bullet,bindex)=>{
       bullet.update()
       if(bullet.x-bullet.radius<0 ||
           bullet.x-bullet.radius>canvas.width || bullet.y-bullet.radius<0 ||
           bullet.y-bullet.radius>canvas.height){
           setTimeout(()=>{
               bullets.splice(bindex,1)
```

```
           }, 0);
```

```
       }
   })
   enemies.forEach((enemy,eindex)=>{
       enemy.update()
       const dist = Math.hypot(player.x-enemy.x,player.y-enemy.y)
            if(dist-enemy.radius-player.radius<1){
               cancelAnimationFrame(animationid);
               menu.style.display='flex'
               finalscore.innerHTML = score
               startbtn.innerText = 'Restart Game'

           }
```

```
       bullets.forEach((bullet,bindex)=>{
            const dist = Math.hypot(bullet.x-enemy.x,bullet.y-enemy.y)
            if(dist-enemy.radius-bullet.radius<1){
```

```
               for(i=0;i<enemy.radius*2;i++){
                   particles.push(new Particle(bullet.x,bullet.y,Math.random
                       ()*2,enemy.color,{
                       x:Math.random()-0.5,
                        y:Math.random()-0.5
                    },9))
                }
                if(enemy.radius-10 > 10 ){
                    score+=50
```

```
                    enemy.radius-=10;
                    setTimeout(()=>{
                        bullets.splice(bindex,1)


                    }, 0);
```

```
                }else{
                    score+=150
                    setTimeout(()=>{
                        bullets.splice(bindex,1)
                    enemies.splice(eindex,1)

                    }, 0);
```

```
                }
                scoreboard.innerHTML=score

             }
```

```
        })
    })
```

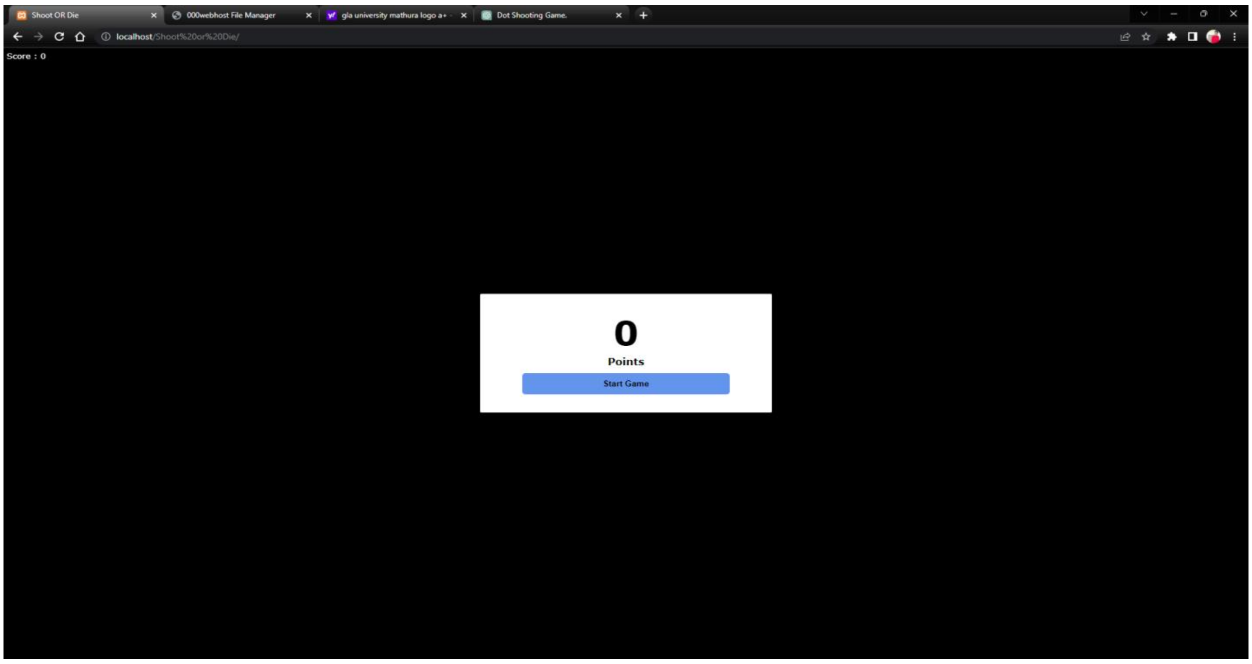
```

 }
```

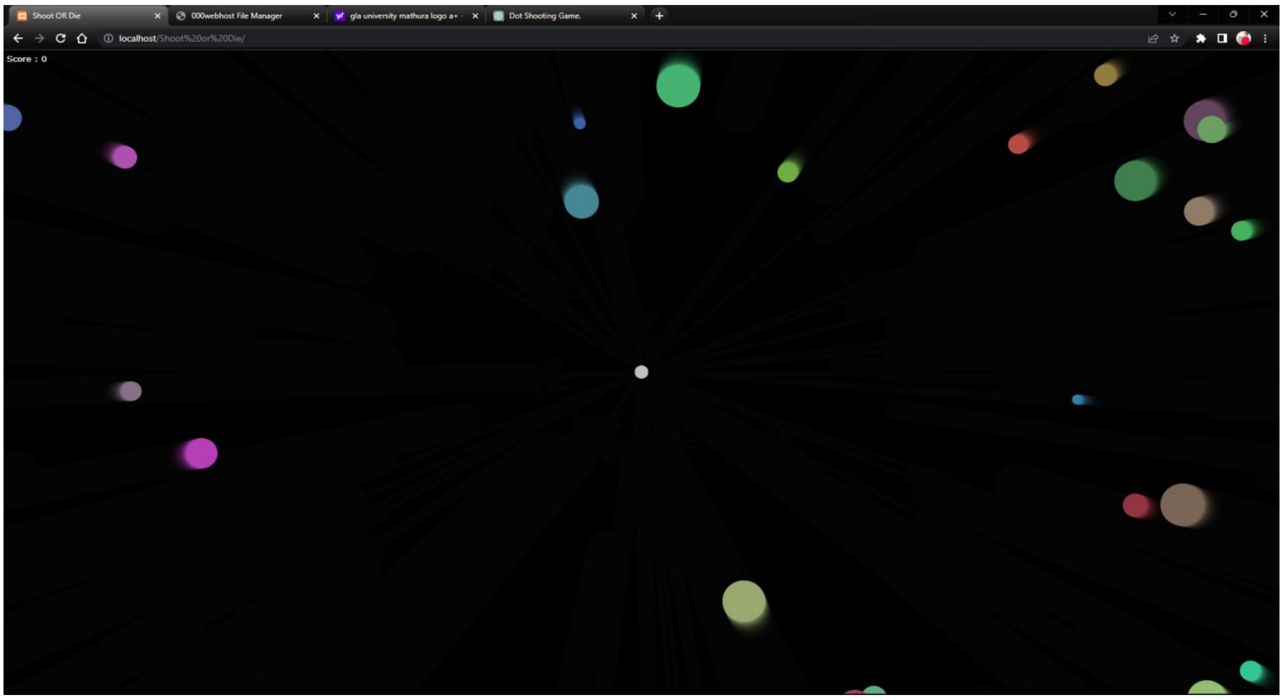
```
startbtn.addEventListener('click',()=>{
    init()
    menu.style.display='none'
    animate();
    spawnEnemies();
})
```

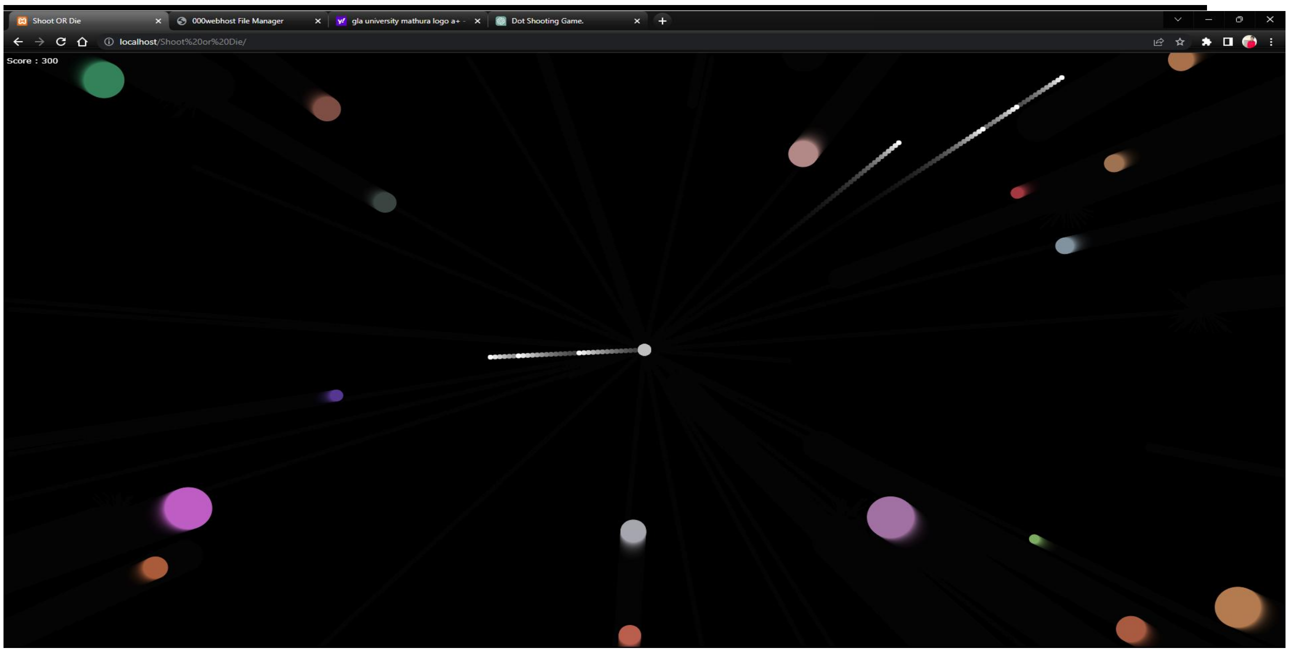
## WORKING

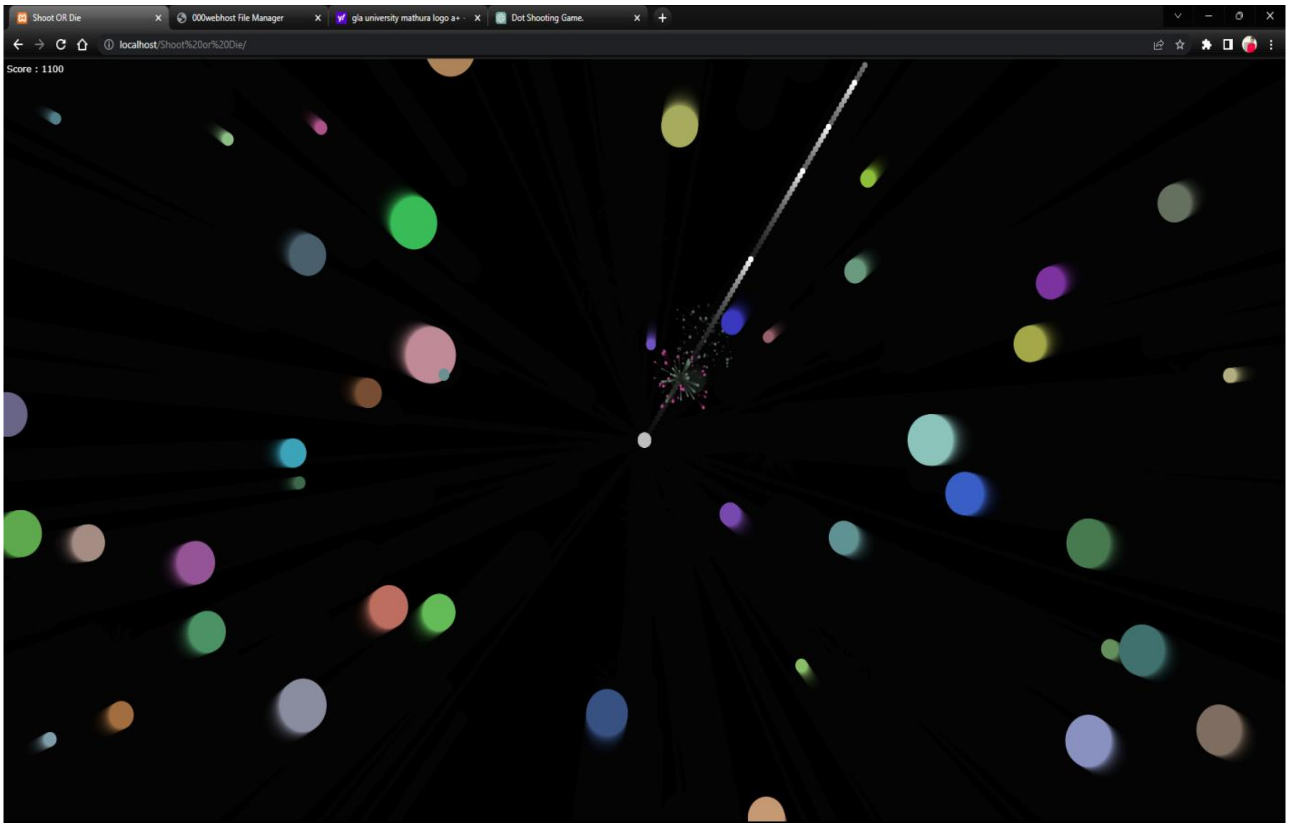
1. Game start with a menu page where the initial score is zero.
2. Then click on start game .
3. After that click wherever you want to shoot .
4. And burst the other colour dot .
5. If other dot touches you the game is over and score is updated.
6. Player can restart the game by clicking on restart button

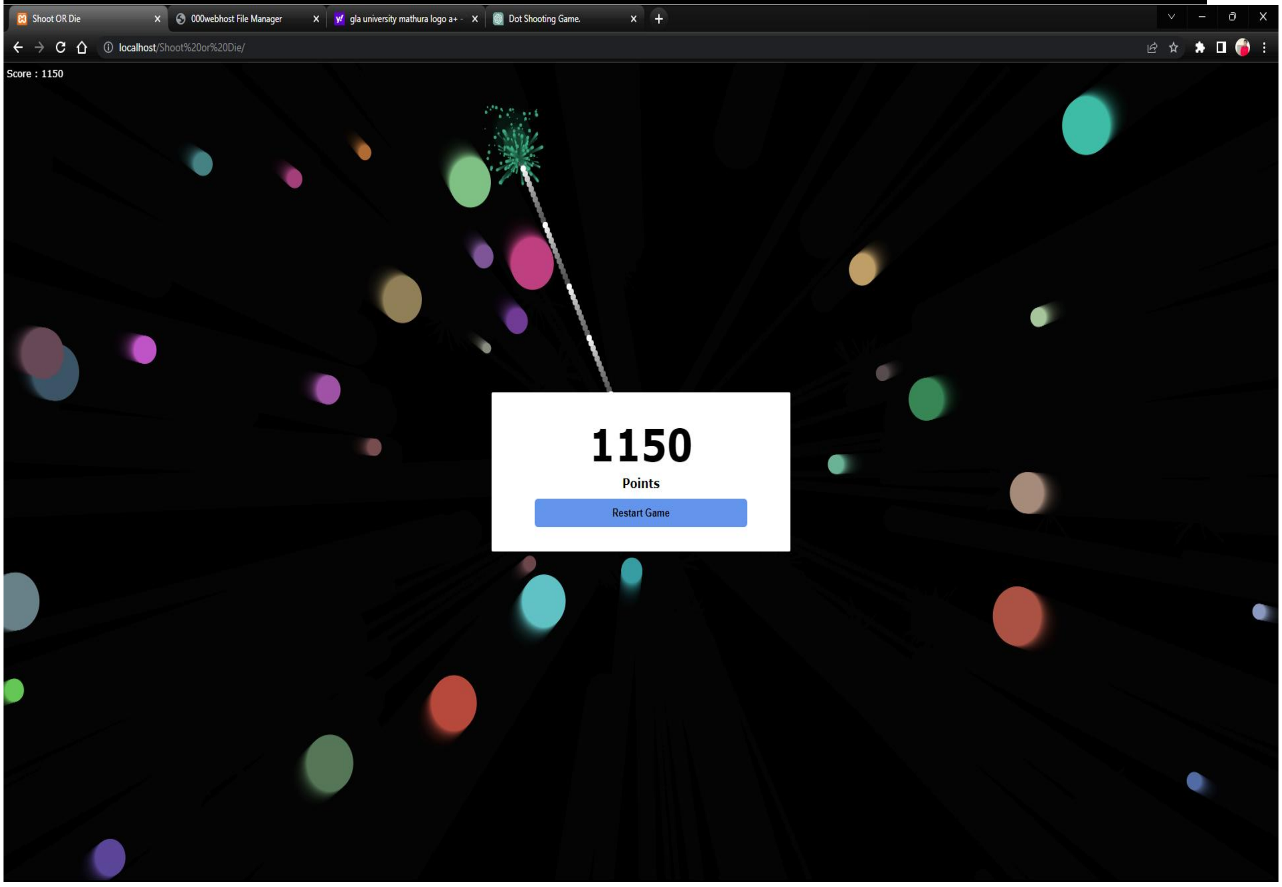
## -- GAME SCREENSHOOTS --




Shoot OR Die
000webhost File Manager
gla university mathura logo a+ -
Dot Shooting Game.
localhost/Shoot%20or%20Die/
Score : 0


Shoot OR Die
000webhost File Manager
gla university mathura logo a+ -
Dot Shooting Game.
localhost/Shoot%20or%20Die/
Score : 300


Shoot OR Die
000webhost File Manager
gla university mathura logo a+ -
Dot Shooting Game.
localhost/Shoot%20or%20Die/
Score : 1100


Shoot OR Die
000webhost File Manager
gla university mathura logo a+ -
Dot Shooting Game.
localhost/Shoot%20or%20Die/
Score : 1150
1150
Points
Restart Game

# CONCLUSION AND FUTURE WORK

## Conclusion:

In conclusion, the simple dot shooting game is a fun and addictive game that challenges players to test their aim and reflexes. The game involves shooting dots that move across the screen and trying to score as many points as possible. The game can be played by people of all ages and skill levels, and can be a great way to pass the time or to compete against friends. With its simple yet engaging gameplay, the simple dot shooting game is sure to provide hours of entertainment.

## Future work:-

The future goal of the simple shooting shooting game could be to add more features and levels to make the game more challenging and engaging for players. The game could also be adapted to different platforms, such as mobile devices or gaming consoles, to reach a wider audience. Additionally, the game could be integrated with social media platforms to allow players to compete and share their scores with friends. Another potential goal could be to explore virtual and augmented reality technologies to create an even more immersive gaming experience. Overall, the future of the simple dot shooting game holds many possibilities for further development and growth.

## GITHUB:

https://github.com/Suryanshverma213202/shoot-or-

die.git

## Game Is Live on:

https://shootordie143.000webhostapp.com/