Farmer vs Zombies Game Tutorial

Step 1: Set Up HTML Structure

Start by creating an HTML document to hold all the elements of your game, such as the canvas, description, and score.

HTML Structure includes:

- Basic HTML Document

- Canvas for rendering the game

- Score display section

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min
.css" rel="stylesheet" />
    <title>Farmer vs Zombie Game</title>
    <style>
        body {
            margin: 0;
            background: #000;
            overflow: hidden;
            color:white;
        }

        canvas {
            display: block;
            margin: 0 auto;
            background: url('./images/grass.jpg') no-repeat center
center;
            background-size: cover;
        }

        #score {
            color: white;
            padding: 10px;
            border: 4px double yellow;
            background: rgba(0, 0, 0, 0.7);
            font-size: 20px;
            font-family: Arial, sans-serif;
            text-align: center;
        }
```

```
    </style>
</head>
<body>
    <!-- Game Description and Canvas -->
    <section class="container-fluid">
        <div class="row justify-content-center text-center">
            <h1>Farmer vs Zombies</h1>
            <p>Farmer vs Zombies is an action-packed survival game where
you play as a fearless farmer dodging relentless zombies. Armed with only
your agility and a supply of ripe tomatoes, you must evade the undead and
take them down one throw at a time. Can you outlast the zombie horde and
defend your farm?</p>
        </div>

        <div class="row justify-content-center text-center">
            <div id="score" class="col-lg-1">Score: 0</div>
            <canvas id="gameCanvas" width="1280" height="720"></canvas>
        </div>
    </section>
```

Step 2: Add JavaScript for Game Logic

Declare Important Variables like Farmer, Zombie, Bullet properties. Handle the farmer's movement and bullet shooting using key events.

```
const canvas = document.getElementById('gameCanvas');
const ctx = canvas.getContext('2d');

const keys = {}; // Stores key presses

// Farmer properties
const farmer = {
    x: canvas.width / 2 - 40,
    y: canvas.height - 100,
    width: 100,
    height: 100,
    speed: 7,
    bullets: [] // Bullet storage
};

const farmerImg = new Image();
farmerImg.src = "https://purepng.com/public/uploads/large/purepng.com-
farmeragriculturefarmerraw-materialsraising-field-
cropslaborerclipartcartoon-1421526886903vjerk.png";

// Bullet properties
const bulletImg = new Image();
bulletImg.src = "https://purepng.com/public/uploads/large/purepng.com-red-
tomatoestomatosalad-fruitred-fruittomatoes-1701527316192n3ycv.png";
```

```
// Zombie properties
const zombieImg = new Image();
zombieImg.src =
"https://media1.giphy.com/media/lpsYPm3S4HmezfNMO6/giphy.gif?cid=6c09b952e
0m0de4p3hvxgz61iyqy2c42rkdmvhsxoqvgkkns&ep=v1_stickers_search&rid=giphy.gi
f&ct=s";
```

Movement: WASD keys to move the farmer.

Shooting: Spacebar to shoot tomatoes.

```
document.addEventListener('keydown', (e) => {
    keys[e.code] = true;
    if (e.code === 'Space') {
        shootBullet();
    }
});

document.addEventListener('keyup', (e) => {
    keys[e.code] = false;
});

function shootBullet() {
    farmer.bullets.push({
        x: farmer.x + farmer.width / 2 - 5,
        y: farmer.y,
        width: 40,
        height: 60
    });
}
```

Step 3: Create Game Mechanics

Set up zombie spawning with random positions and movement. Implement collision detection

between bullets, zombies, and the farmer.

```
const zombies = [];
let zombieSpawnInterval = 2000;
let lastZombieSpawn = Date.now();

function spawnZombie() {
    const size = Math.random() * 80 + 60;
    zombies.push({
        x: Math.random() * (canvas.width - size),
        y: -size,
        width: size,
        height: size,
```

```
        speed: Math.random() * 3 + 3
    });
}
```

Spawn Zombies: Random positions and speeds.

Collision detection between farmer and zombies, and bullet and zombies.

```
function isColliding(rect1, rect2) {
    return !(rect2.x > rect1.x + rect1.width ||
        rect2.x + rect2.width < rect1.x ||
        rect2.y > rect1.y + rect1.height ||
        rect2.y + rect2.height < rect1.y);
}
```

Step 4: Game Update and Draw

Create update and draw functions to handle the game state: farmer's movement, zombie spawning,

and bullet movement.

```
function update() {


    if (keys['KeyA'] && farmer.x > 0) farmer.x -= farmer.speed;


    if (keys['KeyD'] && farmer.x + farmer.width < canvas.width) farmer.x
+= farmer.speed;


    if (keys['KeyW'] && farmer.y > 0) farmer.y -= farmer.speed;


    if (keys['KeyS'] && farmer.y + farmer.height < canvas.height) farmer.y
+= farmer.speed;







    // Bullet movement


    for (let i = farmer.bullets.length - 1; i >= 0; i--) {
```

```
        const bullet = farmer.bullets[i];
        bullet.y -= bulletSpeed;
        if (bullet.y < -bullet.height) farmer.bullets.splice(i, 1);
    }

    // Spawn zombies
    if (Date.now() - lastZombieSpawn > zombieSpawnInterval) {
        spawnZombie();
        lastZombieSpawn = Date.now();
    }

    // Move zombies
    for (let i = zombies.length - 1; i >= 0; i--) {
        const zombie = zombies[i];
        zombie.y += zombie.speed;
        if (zombie.y > canvas.height) zombies.splice(i, 1);
    }
```

```
    // Collision check (bullet-zombie)
    for (let i = zombies.length - 1; i >= 0; i--) {
        for (let j = farmer.bullets.length - 1; j >= 0; j--) {
            if (isColliding(zombies[i], farmer.bullets[j])) {
                zombies.splice(i, 1);
                farmer.bullets.splice(j, 1);
                break;
            }
        }
    }
    // Collision check (farmer-zombie)
    for (let i = 0; i < zombies.length; i++) {
        if (isColliding(farmer, zombies[i])) {
            alert("Game Over!");
            document.location.reload();
        }
    }
}
```

```
function draw() {

    ctx.clearRect(0, 0, canvas.width, canvas.height);

    ctx.drawImage(farmerImg, farmer.x, farmer.y, farmer.width,
farmer.height);




    for (const bullet of farmer.bullets) {

        ctx.drawImage(bulletImg, bullet.x, bullet.y, bullet.width,
bullet.height);

    }




    for (const zombie of zombies) {

        ctx.drawImage(zombieImg, zombie.x, zombie.y, zombie.width,
zombie.height);

    }


}
```

Game Update: Update positions and handle interactions.

Draw: Draw farmer, zombies, and bullets on the canvas.

Step 5: Add Scoring

Increase the score when a bullet hits a zombie and display it on the screen.

Score increases on zombie destruction.

Score is displayed in the score section.

```
let score = 0;
const originalIsColliding = window.isColliding;
window.isColliding = function (rect1, rect2) {
    const collided = originalIsColliding(rect1, rect2);
    if (collided) {
        const isBullet = rect1.width <= 60 || rect2.width <= 60;
        const isZombie = rect1.width >= 80 || rect2.width >= 80;
        if (isBullet && isZombie) {
            score++;
            const scoreDisplay = document.getElementById("score");
            if (scoreDisplay) scoreDisplay.textContent = "Score: " +
score;
        }
    }
    return collided;
};
```

Step 6: Start the Game Loop

Create a game loop using requestAnimationFrame to keep updating and rendering the game state

continuously.

Game Loop: Continuously update the game state and draw the elements.

Start the loop with: requestAnimationFrame(gameLoop);

```
function gameLoop() {
    update();
    draw();
    requestAnimationFrame(gameLoop);
}

requestAnimationFrame(gameLoop);
```