



Computer Graphics and Multimedia

Submitted By:

Subham Choudhary

Serial No: 02

11415002718

MSIT CSE-3

Submitted to:

Vimal Gaur

Soccer Strikers

How the game is Built:

- The basic logic stays in JAVASCRIPT

- The structure of UI is made with Frontend (HTML Canvas CSS JS=>JQUERY)

- and other data input in form of Player name happens with JAVASCRIPT (Node.js).

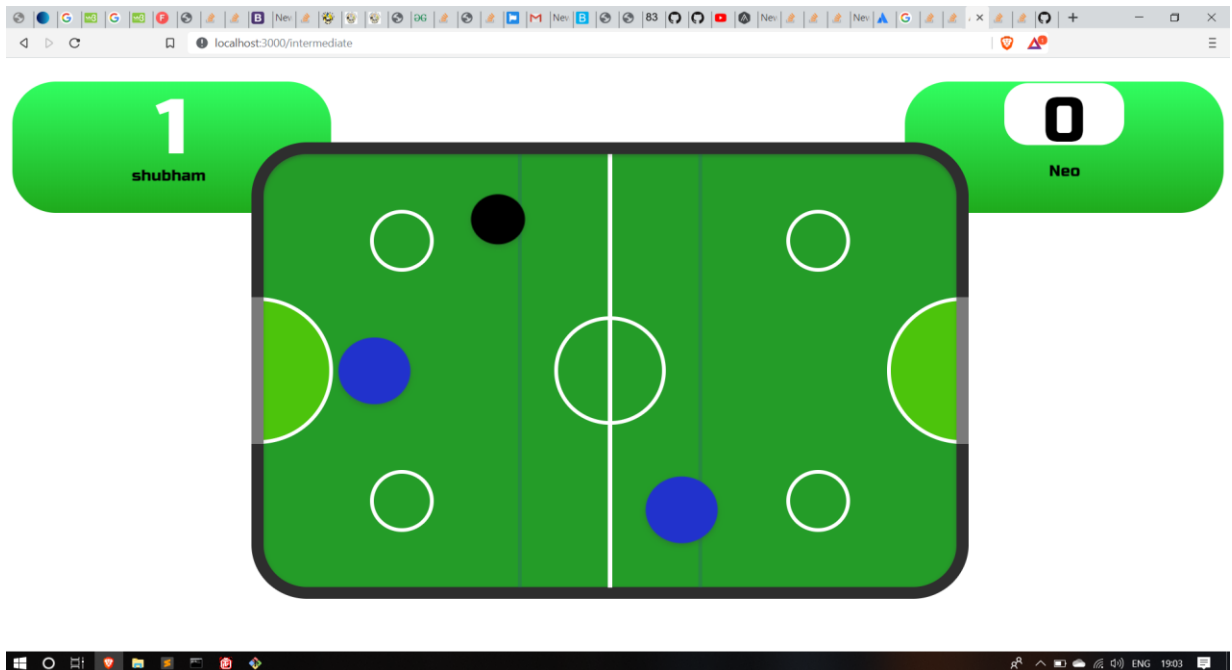
- The Principle is based on 2D collisions of canvas made objects.

- As physics have given formulae of elastic collisions occurs between two objects therefore when the 2D objects collide then there parameters are sent to the function formulae and whole thing return the resultant direction of the two balls and the puck also .

Brief About the Game:

1. Versus COM

It's like most games everyone would have played in their childhood where you have to control the player 1 movement through Arrow Keys and the second player will be dependent upon the ball movement. Have to score goal by striking the ball into scoring area.



2. Versus PLAYER

It's like most games everyone would have played in their childhood where you have to control the player 1 movement through Arrow Keys and the second player will be assigned other key values but on the same keyboard. Both players will have separate control and no player is bound to any movement done.



SOCCKER-STRIKERS

```
var p2score=0;
```

```
var p1score=0;
let hit = new Audio();
let wall = new Audio();
let userScore = new Audio();
let comScore = new Audio();
let stadium= new Audio();

hit.src = "sounds/hit.mp3";
wall.src = "sounds/wall.mp3";
comScore.src = "sounds/comScore.mp3";
userScore.src = "sounds/userScore.mp3";
stadium.src="sounds/stadium.mp3"
```

```
var p2=document.querySelector('.scorecardp2');
var p1=document.querySelector('.scorecardp1');
```

```
var board = document.getElementById("canvas"),
    boardContext=board.getContext('2d'),
    boardWidth = 770,
    boardHeight = 520,
    boardCenterX = boardWidth / 2,
    boardCenterY = boardHeight / 2,
    controllers = [],
    goal = document.getElementsByClassName('table__goal-crease'),
    goalHeight1=goal[1].clientHeight,
    goalPosTop1=(boardHeight-goalHeight1)/2,
    goalHeight2 = goal[0].clientHeight,
    goalPosTop2 = (boardHeight - goalHeight2) / 2,
    score = [];
```

```
board.width = boardWidth;
board.height = boardHeight;
board.focus();
```

```
function updateplayer2(b){
```

```

p2.textContent="\xa0\xa0\xa0\xa0\xa0\xa0"+b;
p2.style.transition="0.6s";
p2.style.transitionTimingFunction = "ease-out";

p2.style.color="white";
p2.style.font=" 100px Russo One";
p2.style.objectFit="cover";
}
function updateplayer1(a){

p1.textContent="\xa0\xa0\xa0\xa0\xa0\xa0"+a;
p1.style.transition="0.6s";
p1.style.transitionTimingFunction = "ease-out";
p1.style.marginTop='30px';
p1.style.color="white";
p1.style.font=" 100px Russo One";
p1.style.objectFit="cover";
}
function goalscreen(){
$('.goal-celebration').addClass('goal-celebration-active');
$('.fa-futbol').addClass('rotateme');
setTimeout(function(){
$('.goal-celebration').removeClass('goal-celebration-active');
$('.fa-futbol').removeClass('rotateme');

},2000)

}
function p1screen(){
$('.player1').addClass('player1-active');

setTimeout(function(){
$('.player1').removeClass('player1-active');

})

}
function p2screen(){
$('.player2').addClass('player2-active');

setTimeout(function(){
$('.player2').removeClass('player2-active');

```

```

    })

}

function Disc() {

    this.startingPosX = boardCenterX;
    this.startingPosY = boardCenterY;
    this.x = this.startingPosX;
    this.y = this.startingPosY;
    this.radius = 30;
    this.mass = 15;
    this.velocityX = 0;
    this.velocityY = 0;
    this.maxSpeed = 10;
    this.frictionX = 0.997;
    this.frictionY = 0.997;
    this.acceleration = 1;
    this.color = '#000000';

    this.keepControllerInBoard = function() {

        if (this.x > (boardWidth - this.radius) || this.x < this.radius) {

            if (this.x < this.radius) {
                this.velocityX = 2;
            } else {
                this.velocityX = -2;
            }
        }
    }

    if (this.y > (boardHeight - this.radius) || this.y < this.radius) {

        if (this.y < this.radius)
        {
            this.velocityY = 2;
        } else
        {
            this.velocityY = -2;
        }
    }
}

```



```

    }

}

if (controller.x > (boardCenterX - controller.radius) && controller.x < boardCenterX) {

    controller.velocityX = -3;
}

if (controllerTwo.x > boardCenterX && controllerTwo.x < (boardCenterX +
(controllerTwo.radius / 2))) {

    controllerTwo.velocityX = +3;
}
},

this.keepPuckInBoard = function() {

    // Determine if disc is to far right or left
    // Need to determine if goal scored on x axis as well
    if (this.x > (boardWidth - this.radius) || this.x < this.radius) {

        // Stop puck from getting stuck
        if (this.x > (boardWidth - this.radius)) {
            this.x = boardWidth - this.radius;
        } else {
            this.x = this.radius;
        }
    }

    // Check to see if goal scored
    if (this.y > (goalPosTop1 + puck.radius) && this.y < (goalPosTop1 + goalHeight1) -
puck.radius && this.x >= (boardWidth-2*puck.radius))
    { if(p1score<11) {
        userScore.play();
        p1score++;
        updateplayer1(p1score);
    }
    }
}

```

```

        goalscreen();
        puck = new Disc(boardCenterX, boardCenterY);
    }
    else{
        p1screen();

    }

}
else if(this.y>(goalPosTop2 + puck.radius) && this.y < (goalPosTop2 + goalHeight2) -
puck.radius)
{ if(p2score<11){
    comScore.play();
    p2score++;
    updateplayer2(p2score);
    goalscreen();
    puck = new Disc(boardCenterX, boardCenterY);
}
else{
    p2screen();

}
}
else {
    // Reverse X direction
    this.velocityX = -this.velocityX;
}
}

// Determine if disc is to far up or down
if (this.y > (boardHeight - this.radius) || this.y < this.radius) {

    // Stop puck from getting stuck
    if (this.y > (boardHeight - this.radius)) {
        this.y = boardHeight - this.radius;
    } else {
        this.y = this.radius;
    }

    // Reverse direction
    this.velocityY = -this.velocityY;
}

```

```
}
```

```
// i took help From Github and Youtube to configure exact formulae for Collision for 2D objects.
```

```
this.discCollision = function() {
```

```
    for (var i = 0; i < controllers.length; i++) {
```

```
        var distanceX = this.x - controllers[i].x,
```

```
            distanceY = this.y - controllers[i].y,
```

```
            // Multiply each of the distances by this
```

```
            // Squareroot that number, which gives you the distance between the two disc's
```

```
            distance = Math.sqrt(distanceX * distanceX + distanceY * distanceY),
```

```
            // Add the two disc radius together
```

```
            addedRadius = this.radius + controllers[i].radius;
```

```
        // Check to see if the distance between the two circles is smaller than the added radius
```

```
        // If it is then we know the circles are overlapping
```

```
        if (distance < addedRadius) {
```

```
            hit.play();
```

```
        //calculate angle, sine, and cosine
```

```
        var angle = Math.atan2(distanceY, distanceX),
```

```
            sin = Math.sin(angle),
```

```
            cos = Math.cos(angle),
```

```
            //rotate controllers[i]'s position
```

```
            pos0 = {
```

```
                x: 0,
```

```
                y: 0
```

```
            },
```

```
            //rotate this's position
```

```
            pos1 = rotate(distanceX, distanceY, sin, cos, true),
```

```
            //rotate controllers[i]'s velocity
```

```
            vel0 = rotate(controllers[i].velocityX, controllers[i].velocityY, sin, cos, true),
```

```
            //rotaty
```

```
            vel1 = rotate(this.velocityX, this.velocityY, sin, cos, true),
```

```
            //collision reaction
```

```
            velocityXTotal = vel0.x - vel1.x;
```

```
            vel0.x = ((controllers[i].mass - this.mass) * vel0.x + 2 * this.mass * vel1.x) /  
                (controllers[i].mass + this.mass);
```

```

        vel1.x = velocityXTotal + vel0.x;

        //update position - to avoid objects becoming stuck together
        var absV = Math.abs(vel0.x) + Math.abs(vel1.x),
            overlap = (controllers[i].radius + this.radius) - Math.abs(pos0.x - pos1.x);

        pos0.x += vel0.x / absV * overlap;
        pos1.x += vel1.x / absV * overlap;

        //rotate positions back
        var pos0F = rotate(pos0.x, pos0.y, sin, cos, false),
            pos1F = rotate(pos1.x, pos1.y, sin, cos, false);

        //adjust positions to actual screen positions
        this.x = controllers[i].x + pos1F.x;
        this.y = controllers[i].y + pos1F.y;
        controllers[i].x = controllers[i].x + pos0F.x;
        controllers[i].y = controllers[i].y + pos0F.y;

        //rotate velocities back
        var vel0F = rotate(vel0.x, vel0.y, sin, cos, false),
            vel1F = rotate(vel1.x, vel1.y, sin, cos, false);

        controllers[i].velocityX = vel0F.x;
        controllers[i].velocityY = vel0F.y;

        this.velocityX = vel1F.x;
        this.velocityY = vel1F.y;
    }
}

// Draw disc
this.draw = function() {

    boardContext.shadowColor = 'rgba(50, 50, 50, 0.25)';
    boardContext.shadowOffsetX = 0;
    boardContext.shadowOffsetY = 3;
    boardContext.shadowBlur = 6;

    boardContext.beginPath();
    boardContext.arc(this.x, this.y, this.radius, 0, 2 * Math.PI, false);

```

```

        boardContext.fillStyle = this.color;
        boardContext.fill();

    }

    // Move disc with physic's applied
    this.move = function() {

        // Apply friction
        this.velocityX *= this.frictionX;
        this.velocityY *= this.frictionY;

        // Update position
        this.x += this.velocityX;
        this.y += this.velocityY;
    }

    // Play against a computer

};

// Run game functions
function updateGame() {
    stadium.play();
    // Clear board
    boardContext.clearRect(0, 0, boardWidth, boardHeight);
    // Draw & contain puck
    puck.draw();
    puck.move();
    puck.discCollision();
}

```

```

    puck.keepPuckInBoard();
    // Controllers
    controller.draw();
    controller.move();
    controller.keepControllerInBoard();
    controllerTwo.draw();
    //controllerTwo.computerPlayer();
    controllerTwo.move();
    controllerTwo.keepControllerInBoard();

    // Loop
    requestAnimationFrame(updateGame);

}

// Keyboard events
function moveController2(key) {

    // Up
    if (key === 38 && controllerTwo.velocityY < controllerTwo.maxSpeed) {
        controllerTwo.velocityY -= controllerTwo.acceleration;
    }

    // Down
    if (key === 40 && controllerTwo.velocityY < controllerTwo.maxSpeed) {
        controllerTwo.velocityY += controllerTwo.acceleration;
    }

    // Right
    if (key === 39 && controllerTwo.velocityX < controllerTwo.maxSpeed) {
        controllerTwo.velocityX += controllerTwo.acceleration;
    }

    // Left, decrease acceleration
    if (key === 37 && controllerTwo.acceleration < controllerTwo.maxSpeed) {
        controllerTwo.velocityX -= controllerTwo.acceleration;
    }

}

```

```

function moveController(key) {

    // Up
    if (key === 87 && controller.velocityY < controller.maxSpeed) {
        controller.velocityY -= controller.acceleration;
    }

    // Down
    if (key === 83 && controller.velocityY < controller.maxSpeed) {
        controller.velocityY += controller.acceleration;
    }

    // Right
    if (key === 68 && controller.velocityX < controller.maxSpeed ) {
        controller.velocityX += controller.acceleration;
    }

    // Left, decrease acceleration
    if (key === 65 && controller.acceleration < controller.maxSpeed ) {
        controller.velocityX -= controller.acceleration;
    }

}

function rotate(x, y, sin, cos, reverse) {
    return {
        x: (reverse) ? (x * cos + y * sin) : (x * cos - y * sin),
        y: (reverse) ? (y * cos - x * sin) : (y * cos + x * sin)
    };
}

// Events
document.addEventListener("keydown", function(e) {
    moveController(e.keyCode);
    moveController2(e.keyCode);

});

```

```
// Add puck
var puck = new Disc();

// Add contro
var controller = new Disc();
controller.color = '#2132CC';
controller.radius += 10;
controller.acceleration = 5;
controller.startingPosX = 125;
controller.mass = 50;
controller.x = controller.startingPosX;

var controllerTwo = new Disc();
controllerTwo.color = '#2132CC';
controllerTwo.radius += 10;
controllerTwo.mass = 50;
controllerTwo.startingPosX = (boardWidth - 155);
controllerTwo.acceleration = 5;
controllerTwo.x = controllerTwo.startingPosX;
controllers.push(controller, controllerTwo);

updateGame();
```


HELPS AND REFERENCES

-Stack Overflow -Github -Youtube