SOCCER SIM

Model Construction

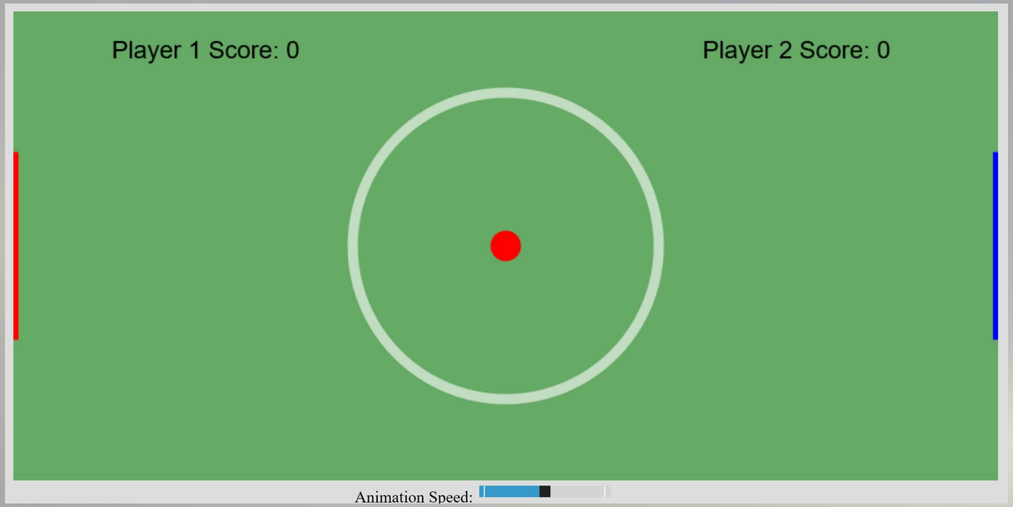
Agent-Based Modeling

Agent-Based Simulation

Our agent: Ball

Aim: Ensure the ball perform differently in various states and simulate the actual soccer match as a 2-dimensional representation.

Initial Model: (no longer in use)



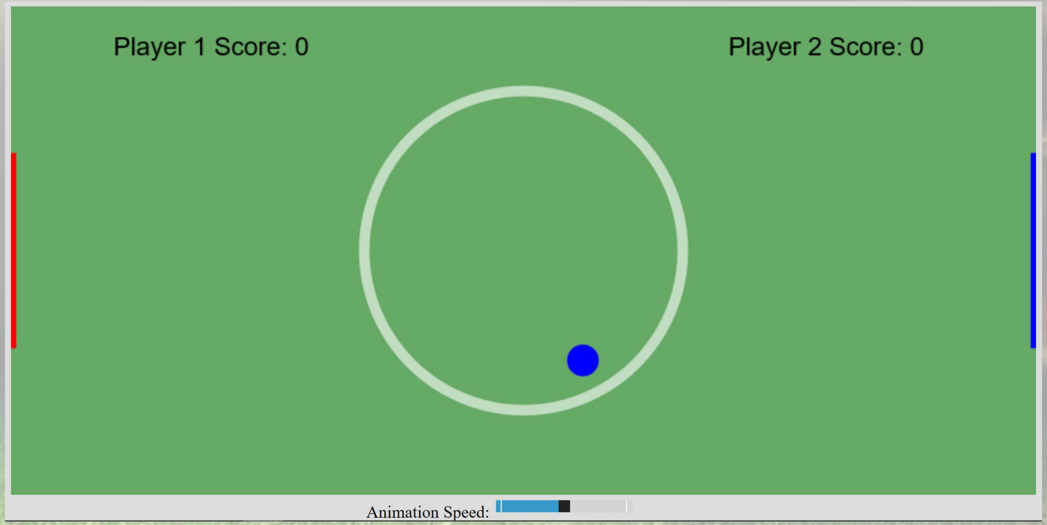
- A very simplified version of simulation using Canvas and 2d to simulate the ball movement.

Good:

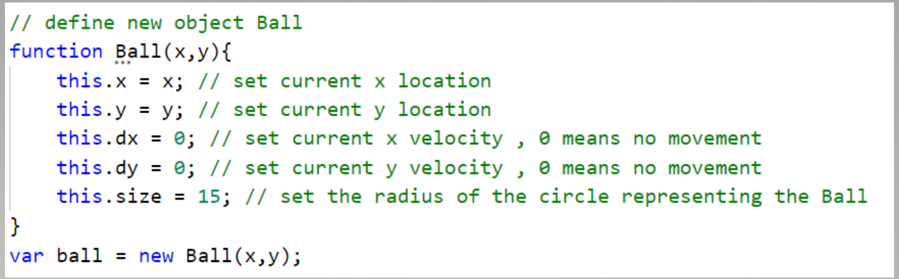
1. Better visualization as we are able to change colour of the "ball" (circle) when different team holds the ball

Limitations and challenges we faced:

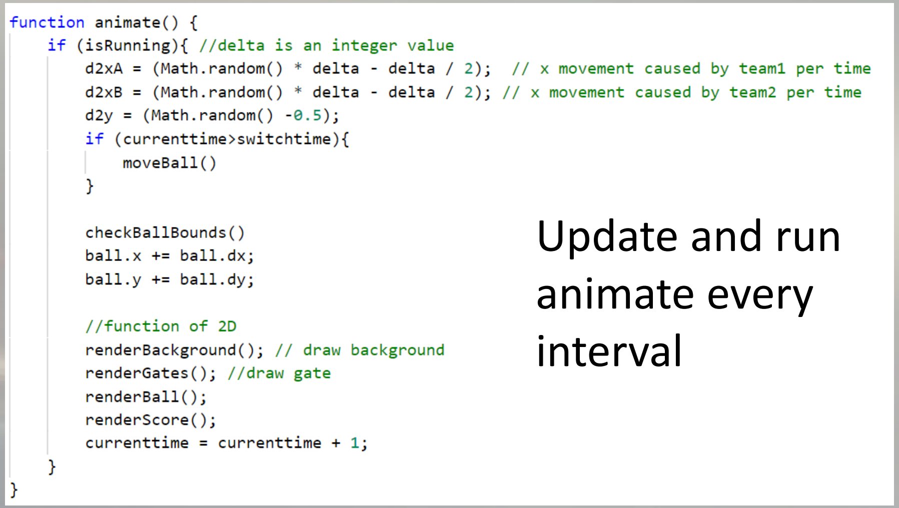
1. It doesn't show explicitly the current state of the ball.
2. We are unable to append the image of the ball, so the ball is shown as little circle.



Initial Model: (no longer in use)

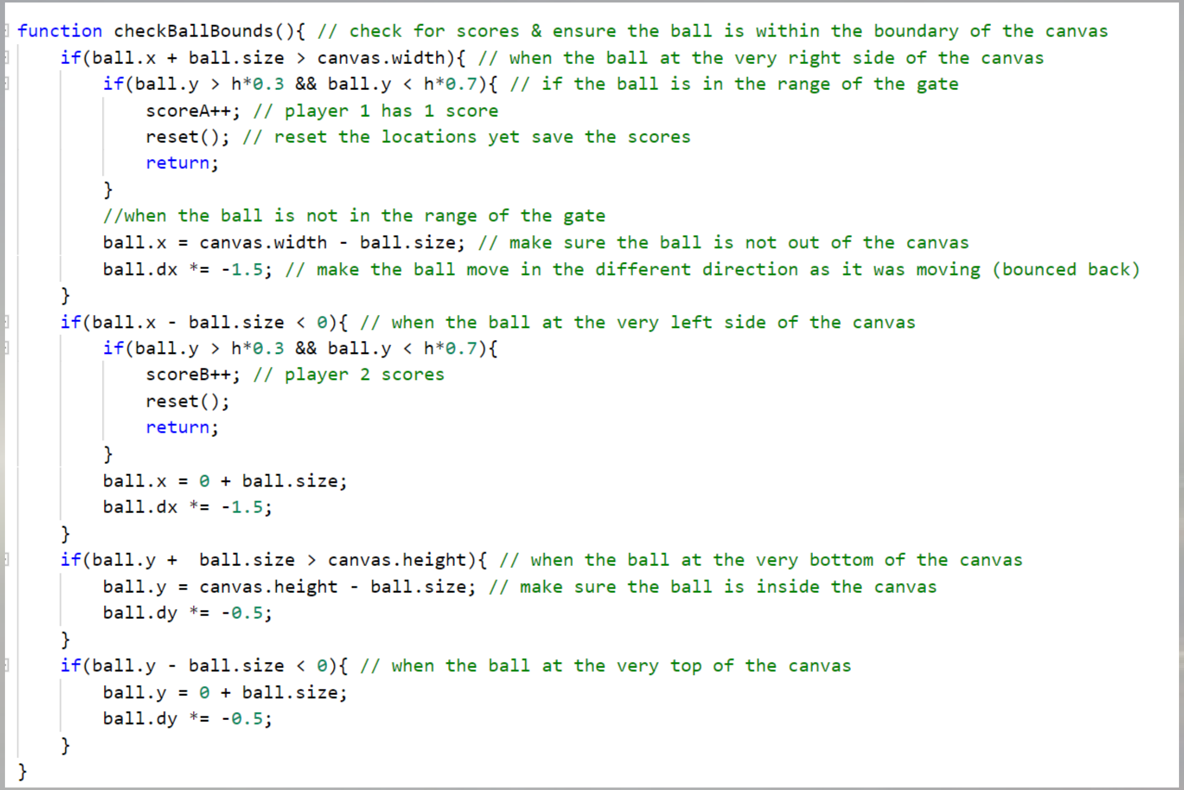


```
// define new object Ball
function Ball(x,y){
    this.x = x; // set current x location
    this.y = y; // set current y location
    this.dx = 0; // set current x velocity , 0 means no movement
    this.dy = 0; // set current y velocity , 0 means no movement
    this.size = 15; // set the radius of the circle representing the Ball
}
var ball = new Ball(x,y);
```



```
function animate() {
    if (isRunning){ //delta is an integer value
        d2xA = (Math.random() * delta - delta / 2);  // x movement caused by team1 per time
        d2xB = (Math.random() * delta - delta / 2); // x movement caused by team2 per time
        d2y = (Math.random() -0.5);
        if (currenttime>switchtime){
            moveBall()
        }

        checkBallBounds()
        ball.x += ball.dx;
        ball.y += ball.dy;

        //function of 2D
        renderBackground(); // draw background
        renderGates(); //draw gate
        renderBall();
        renderScore();
        currenttime = currenttime + 1;
    }
}
```

Update and run animate every interval



```
function checkBallBounds(){ // check for scores & ensure the ball is within the boundary of the canvas
    if(ball.x + ball.size > canvas.width){ // when the ball at the very right side of the canvas
        if(ball.y > h*0.3 && ball.y < h*0.7){ // if the ball is in the range of the gate
            scoreA++; // player 1 has 1 score
            reset(); // reset the locations yet save the scores
            return;
        }
        //when the ball is not in the range of the gate
        ball.x = canvas.width - ball.size; // make sure the ball is not out of the canvas
        ball.dx *= -1.5; // make the ball move in the different direction as it was moving (bounced back)
    }
    if(ball.x - ball.size < 0){ // when the ball at the very left side of the canvas
        if(ball.y > h*0.3 && ball.y < h*0.7){
            scoreB++; // player 2 scores
            reset();
            return;
        }
        ball.x = 0 + ball.size;
        ball.dx *= -1.5;
    }
    if(ball.y +  ball.size > canvas.height){ // when the ball at the very bottom of the canvas
        ball.y = canvas.height - ball.size; // make sure the ball is inside the canvas
        ball.dy *= -0.5;
    }
    if(ball.y - ball.size < 0){ // when the ball at the very top of the canvas
        ball.y = 0 + ball.size;
        ball.dy *= -0.5;
    }
}
```

Animate includes:

1. Moveball : decide who process the ball
2. TeamA /TeamB : when the ball is under the control of team A/B

3. checkBallBounds
- Increase the score of each team if the ball reached the gate
- Check the boundary to ensure the ball is within the field

Current Model:
Build up based on the initial model using svg and D3

How the current simulation works:
• Steps through time in increments using *“currentTime”*
• In each increment: update the agent ball
• The update describes agent behavior with nested switch statements based on modal analysis

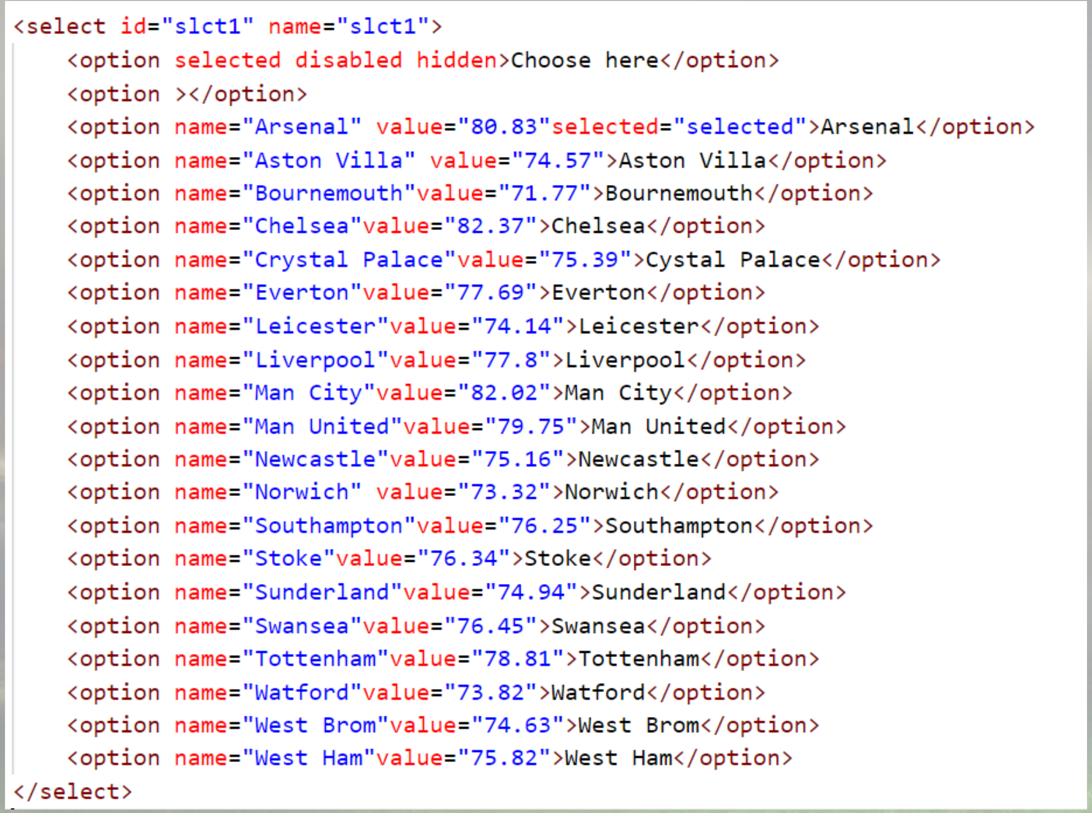
Functional programming like D3 is employed for displaying the soccer field and animating the movement of the ball

HTML 1

Agent-Based Modeling

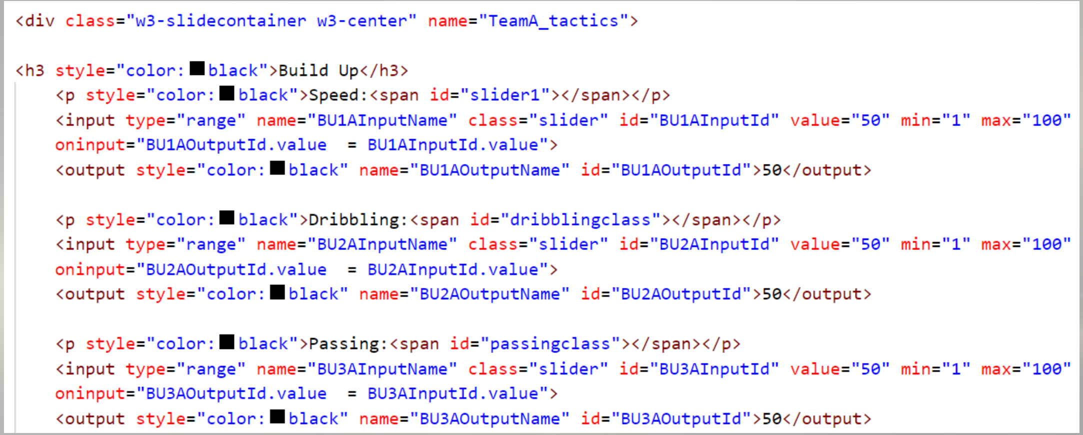
HTML: SoccerSim Tactics Page

Basic elements 1: drop down list of Team A or Team B



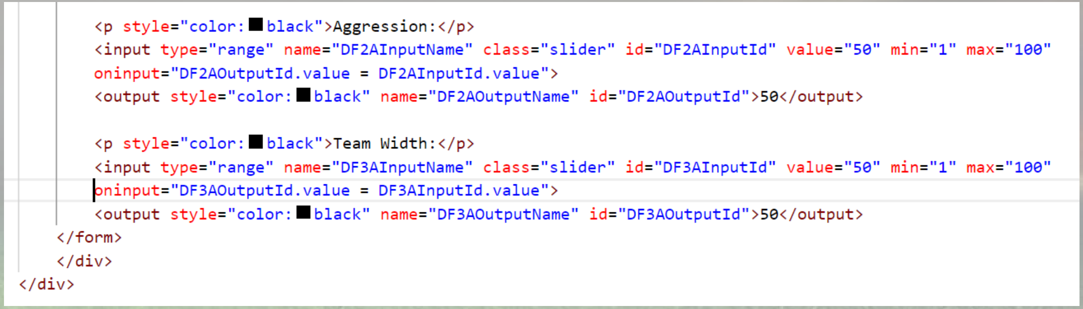
```html
<select id="slct1" name="slct1">
    <option selected disabled hidden>Choose here</option>
    <option ></option>
    <option name="Arsenal" value="80.83"selected="selected">Arsenal</option>
    <option name="Aston Villa" value="74.57">Aston Villa</option>
    <option name="Bournemouth"value="71.77">Bournemouth</option>
    <option name="Chelsea"value="82.37">Chelsea</option>
    <option name="Crystal Palace"value="75.39">Cystal Palace</option>
    <option name="Everton"value="77.69">Everton</option>
    <option name="Leicester"value="74.14">Leicester</option>
    <option name="Liverpool"value="77.8">Liverpool</option>
    <option name="Man City"value="82.02">Man City</option>
    <option name="Man United"value="79.75">Man United</option>
    <option name="Newcastle"value="75.16">Newcastle</option>
    <option name="Norwich" value="73.32">Norwich</option>
    <option name="Southampton"value="76.25">Southampton</option>
    <option name="Stoke"value="76.34">Stoke</option>
    <option name="Sunderland"value="74.94">Sunderland</option>
    <option name="Swansea"value="76.45">Swansea</option>
    <option name="Tottenham"value="78.81">Tottenham</option>
    <option name="Watford"value="73.82">Watford</option>
    <option name="West Brom"value="74.63">West Brom</option>
    <option name="West Ham"value="75.82">West Ham</option>
</select>
```

Basic elements 2: 9 sliders for Team A or B each



```html
<div class="w3-slidecontainer w3-center" name="TeamA_tactics">

<h3 style="color:■black">Build Up</h3>
    <p style="color:■black">Speed:<span id="slider1"></span></p>
    <input type="range" name="BU1AInputName" class="slider" id="BU1AInputId" value="50" min="1" max="100"
    oninput="BU1AOutputId.value  = BU1AInputId.value">
    <output style="color:■black" name="BU1AOutputName" id="BU1AOutputId">50</output>

    <p style="color:■black">Dribbling:<span id="dribblingclass"></span></p>
    <input type="range" name="BU2AInputName" class="slider" id="BU2AInputId" value="50" min="1" max="100"
    oninput="BU2AOutputId.value  = BU2AInputId.value">
    <output style="color:■black" name="BU2AOutputName" id="BU2AOutputId">50</output>

    <p style="color:■black">Passing:<span id="passingclass"></span></p>
    <input type="range" name="BU3AInputName" class="slider" id="BU3AInputId" value="50" min="1" max="100"
    oninput="BU3AOutputId.value  = BU3AInputId.value">
    <output style="color:■black" name="BU3AOutputName" id="BU3AOutputId">50</output>
```

⋮



```html
        <p style="color:■black">Aggression:</p>
        <input type="range" name="DF2AInputName" class="slider" id="DF2AInputId" value="50" min="1" max="100"
        oninput="DF2AOutputId.value = DF2AInputId.value">
        <output style="color:■black" name="DF2AOutputName" id="DF2AOutputId">50</output>

        <p style="color:■black">Team Width:</p>
        <input type="range" name="DF3AInputName" class="slider" id="DF3AInputId" value="50" min="1" max="100"
        oninput="DF3AOutputId.value = DF3AInputId.value">
        <output style="color:■black" name="DF3AOutputName" id="DF3AOutputId">50</output>
    </form>
    </div>
</div>
```

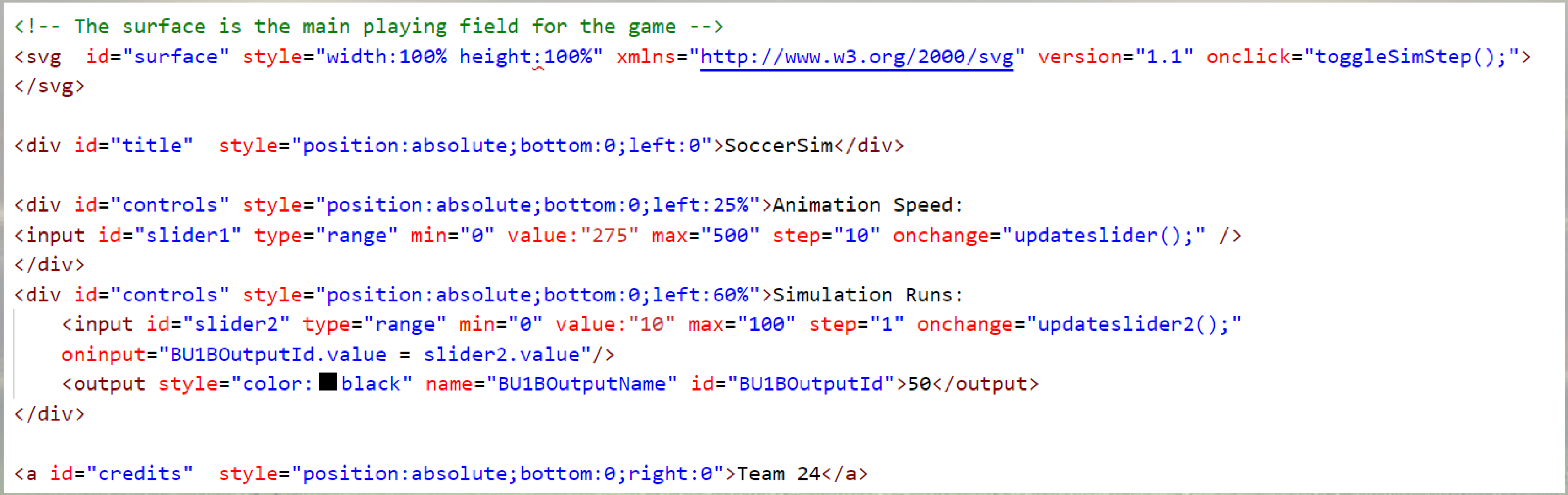
HTML 2

Agent-Based Modeling

HTML: SoccerSim Page

Basic elements 1: surface: an svg element for drawing graphs
Basic element 2: title: title label for the page

Basic elements 3: slider1: a slider bar to control the animation speed
Basic elements 4: slider2: a slider bar to control number of simulation runs, with number placed beside the bar



```
<!-- The surface is the main playing field for the game -->
<svg  id="surface" style="width:100% height:100%" xmlns="http://www.w3.org/2000/svg" version="1.1" onclick="toggleSimStep();">
</svg>

<div id="title"  style="position:absolute;bottom:0;left:0">SoccerSim</div>

<div id="controls" style="position:absolute;bottom:0;left:25%">Animation Speed:
<input id="slider1" type="range" min="0" value:"275" max="500" step="10" onchange="updateslider();" />
</div>
<div id="controls" style="position:absolute;bottom:0;left:60%">Simulation Runs:
    <input id="slider2" type="range" min="0" value:"10" max="100" step="1" onchange="updateslider2();"
    oninput="BU1BOutputId.value = slider2.value"/>
    <output style="color:■black" name="BU1BOutputName" id="BU1BOutputId">50</output>
</div>

<a id="credits"  style="position:absolute;bottom:0;right:0">Team 24</a>
```

HTML: SoccerSim Page

Imports to html at the beginning of the page

```
<!-- d3 is for data visualization -->
<script type="text/javascript" src="lib/d3.min.js"></script>
```

Load after the page elements have been created:

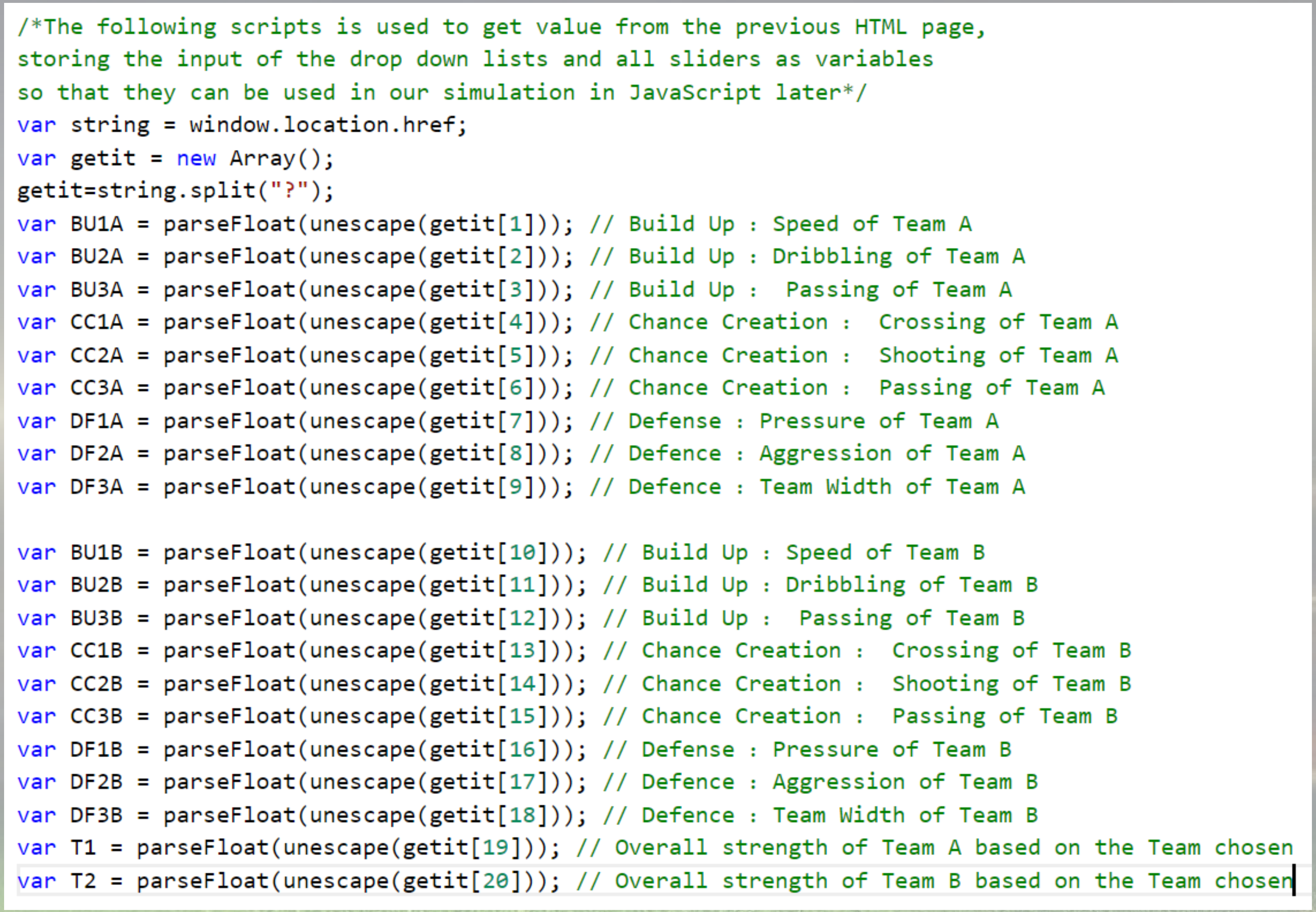
```
<!-- Anything below this line should be a popup window or dialog or a late-loading library -->
<script type="text/javascript" src="lib/SoccerSim.js"></script>
```

JavaScript

Agent-Based Modeling

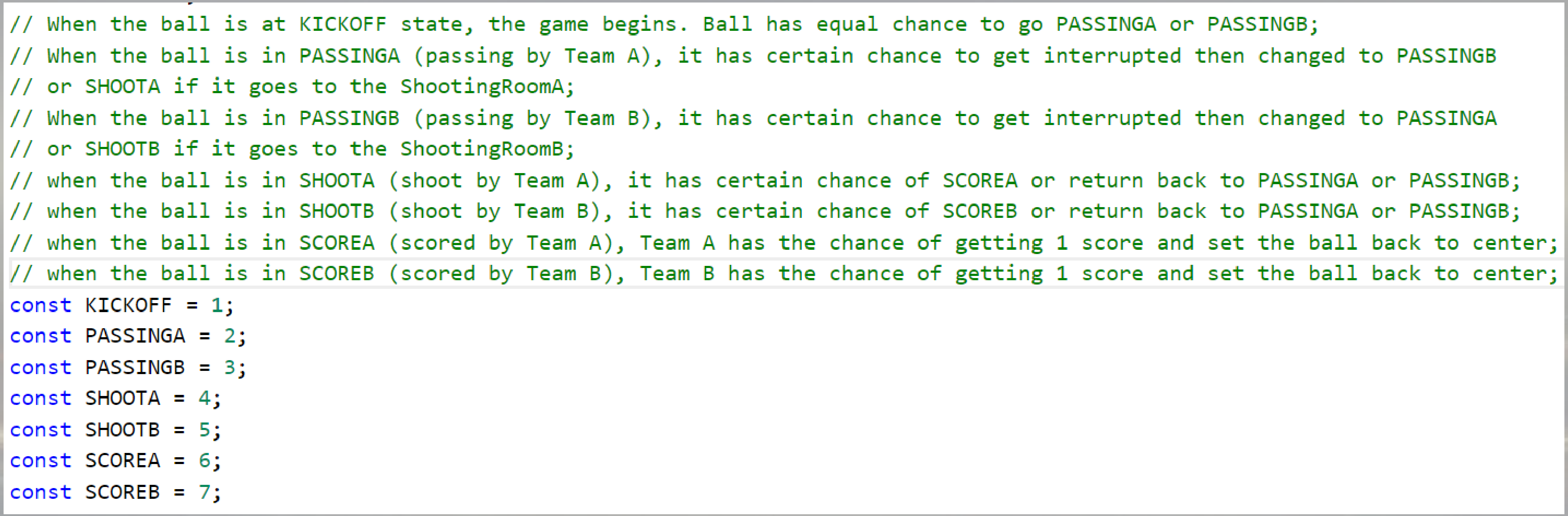
JavaScript:

Get Input from previous HTML page.



```
/*The following scripts is used to get value from the previous HTML page,
storing the input of the drop down lists and all sliders as variables
so that they can be used in our simulation in JavaScript later*/
var string = window.location.href;
var getit = new Array();
getit=string.split("?");
var BU1A = parseFloat(unescape(getit[1])); // Build Up : Speed of Team A
var BU2A = parseFloat(unescape(getit[2])); // Build Up : Dribbling of Team A
var BU3A = parseFloat(unescape(getit[3])); // Build Up :  Passing of Team A
var CC1A = parseFloat(unescape(getit[4])); // Chance Creation :  Crossing of Team A
var CC2A = parseFloat(unescape(getit[5])); // Chance Creation :  Shooting of Team A
var CC3A = parseFloat(unescape(getit[6])); // Chance Creation :  Passing of Team A
var DF1A = parseFloat(unescape(getit[7])); // Defense : Pressure of Team A
var DF2A = parseFloat(unescape(getit[8])); // Defence : Aggression of Team A
var DF3A = parseFloat(unescape(getit[9])); // Defence : Team Width of Team A

var BU1B = parseFloat(unescape(getit[10])); // Build Up : Speed of Team B
var BU2B = parseFloat(unescape(getit[11])); // Build Up : Dribbling of Team B
var BU3B = parseFloat(unescape(getit[12])); // Build Up :  Passing of Team B
var CC1B = parseFloat(unescape(getit[13])); // Chance Creation :  Crossing of Team B
var CC2B = parseFloat(unescape(getit[14])); // Chance Creation :  Shooting of Team B
var CC3B = parseFloat(unescape(getit[15])); // Chance Creation :  Passing of Team B
var DF1B = parseFloat(unescape(getit[16])); // Defense : Pressure of Team B
var DF2B = parseFloat(unescape(getit[17])); // Defence : Aggression of Team B
var DF3B = parseFloat(unescape(getit[18])); // Defence : Team Width of Team B
var T1 = parseFloat(unescape(getit[19])); // Overall strength of Team A based on the Team chosen
var T2 = parseFloat(unescape(getit[20])); // Overall strength of Team B based on the Team chosen
```

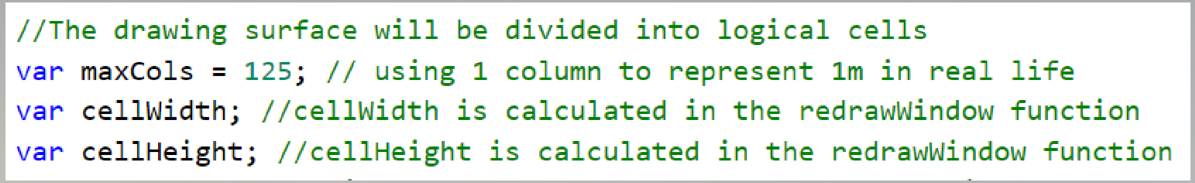
States and types:



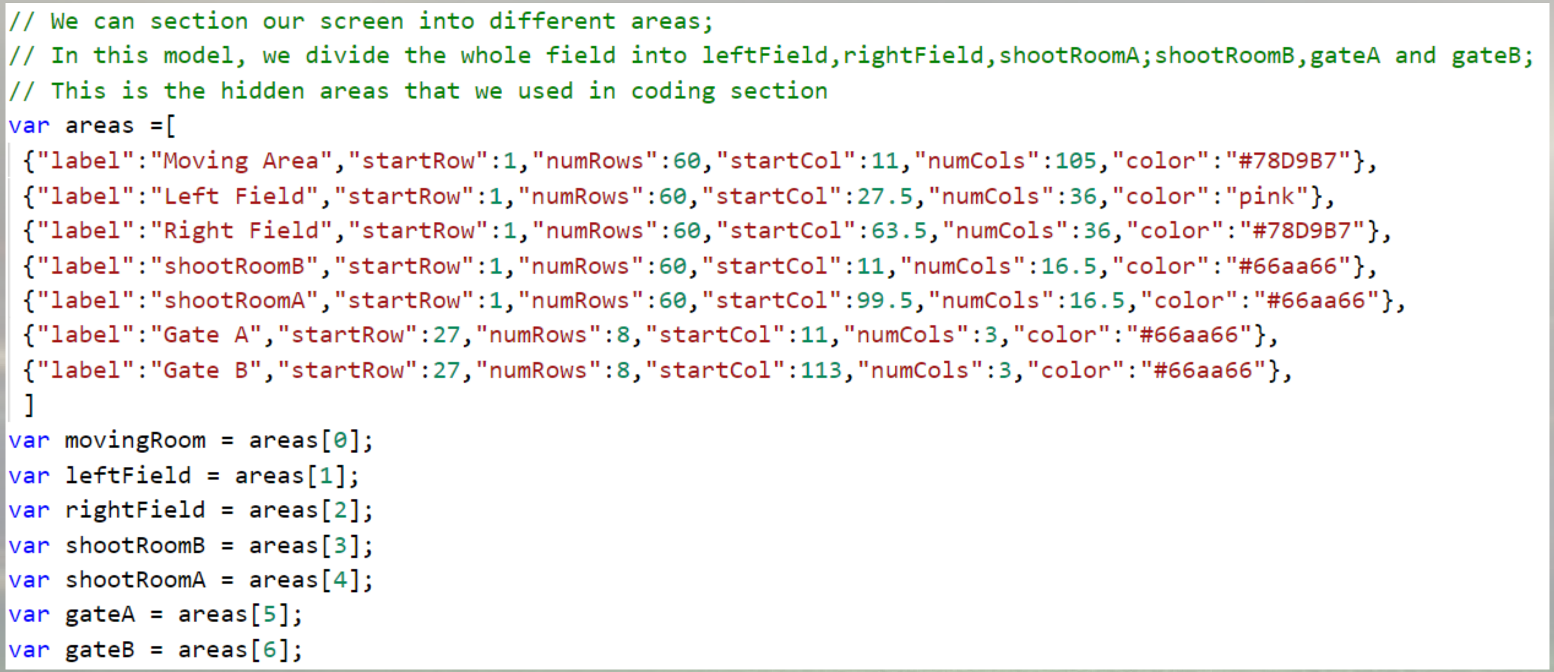
```
// When the ball is at KICKOFF state, the game begins. Ball has equal chance to go PASSINGA or PASSINGB;
// When the ball is in PASSINGA (passing by Team A), it has certain chance to get interrupted then changed to PASSINGB
// or SHOOTA if it goes to the ShootingRoomA;
// When the ball is in PASSINGB (passing by Team B), it has certain chance to get interrupted then changed to PASSINGA
// or SHOOTB if it goes to the ShootingRoomB;
// when the ball is in SHOOTA (shoot by Team A), it has certain chance of SCOREA or return back to PASSINGA or PASSINGB;
// when the ball is in SHOOTB (shoot by Team B), it has certain chance of SCOREB or return back to PASSINGA or PASSINGB;
// when the ball is in SCOREA (scored by Team A), Team A has the chance of getting 1 score and set the ball back to center;
// when the ball is in SCOREB (scored by Team B), Team B has the chance of getting 1 score and set the ball back to center;
const KICKOFF = 1;
const PASSINGA = 2;
const PASSINGB = 3;
const SHOOTA = 4;
const SHOOTB = 5;
const SCOREA = 6;
const SCOREB = 7;
```

The ball has 7 states.

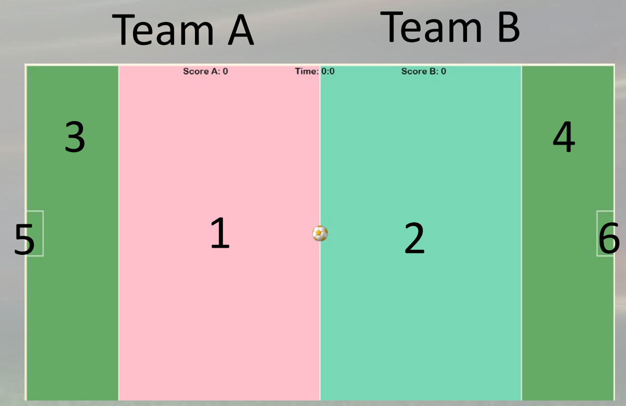
Location Information:



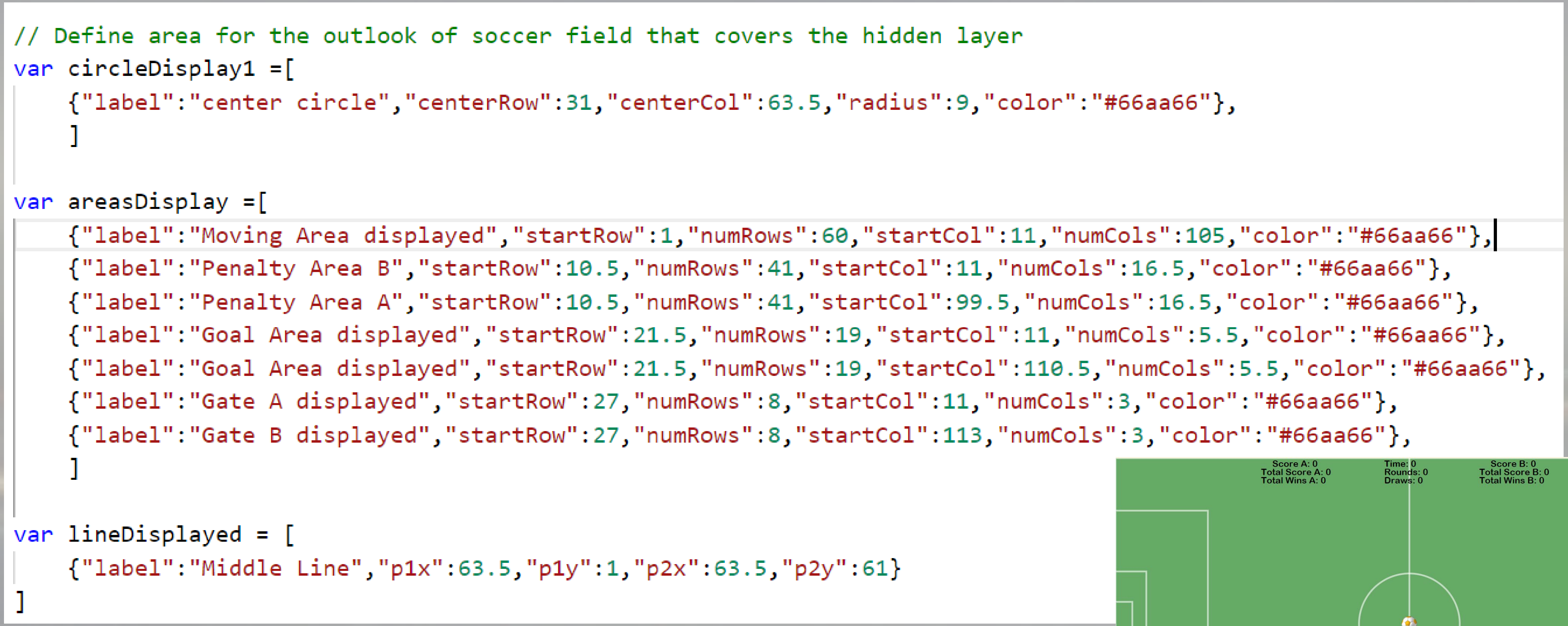
```
//The drawing surface will be divided into logical cells
var maxCols = 125; // using 1 column to represent 1m in real life
var cellWidth; //cellWidth is calculated in the redrawWindow function
var cellHeight; //cellHeight is calculated in the redrawWindow function
```



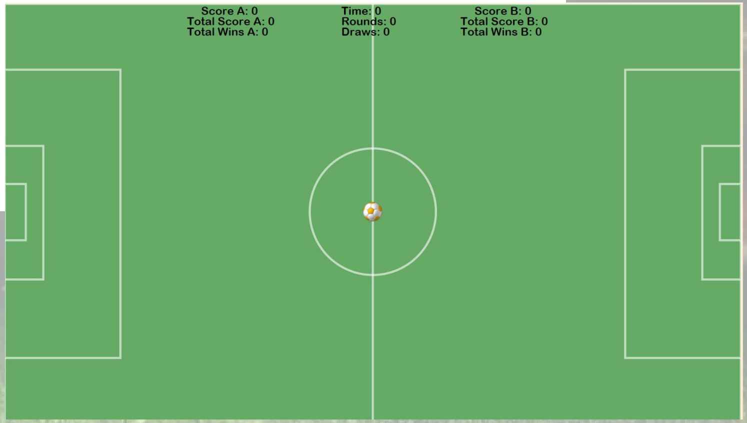
```
// We can section our screen into different areas;
// In this model, we divide the whole field into leftField,rightField,shootRoomA;shootRoomB,gateA and gateB;
// This is the hidden areas that we used in coding section
var areas =[
 {"label":"Moving Area","startRow":1,"numRows":60,"startCol":11,"numCols":105,"color":"#78D9B7"},
 {"label":"Left Field","startRow":1,"numRows":60,"startCol":27.5,"numCols":36,"color":"pink"},
 {"label":"Right Field","startRow":1,"numRows":60,"startCol":63.5,"numCols":36,"color":"#78D9B7"},
 {"label":"shootRoomB","startRow":1,"numRows":60,"startCol":11,"numCols":16.5,"color":"#66aa66"},
 {"label":"shootRoomA","startRow":1,"numRows":60,"startCol":99.5,"numCols":16.5,"color":"#66aa66"},
 {"label":"Gate A","startRow":27,"numRows":8,"startCol":11,"numCols":3,"color":"#66aa66"},
 {"label":"Gate B","startRow":27,"numRows":8,"startCol":113,"numCols":3,"color":"#66aa66"},
 ]
var movingRoom = areas[0];
var leftField = areas[1];
var rightField = areas[2];
var shootRoomB = areas[3];
var shootRoomA = areas[4];
var gateA = areas[5];
var gateB = areas[6];
```



Location Information:

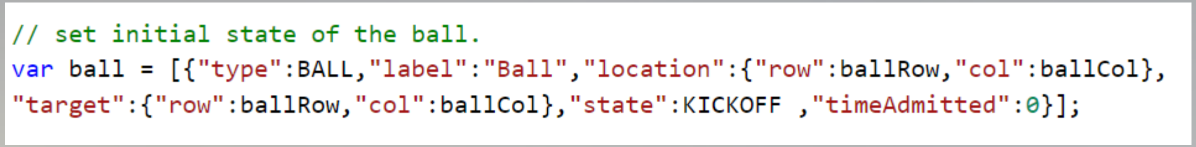


```
// Define area for the outlook of soccer field that covers the hidden layer
var circleDisplay1 =[
    {"label":"center circle","centerRow":31,"centerCol":63.5,"radius":9,"color":"#66aa66"},
    ]

var areasDisplay =[
    {"label":"Moving Area displayed","startRow":1,"numRows":60,"startCol":11,"numCols":105,"color":"#66aa66"},
    {"label":"Penalty Area B","startRow":10.5,"numRows":41,"startCol":11,"numCols":16.5,"color":"#66aa66"},
    {"label":"Penalty Area A","startRow":10.5,"numRows":41,"startCol":99.5,"numCols":16.5,"color":"#66aa66"},
    {"label":"Goal Area displayed","startRow":21.5,"numRows":19,"startCol":11,"numCols":5.5,"color":"#66aa66"},
    {"label":"Goal Area displayed","startRow":21.5,"numRows":19,"startCol":110.5,"numCols":5.5,"color":"#66aa66"},
    {"label":"Gate A displayed","startRow":27,"numRows":8,"startCol":11,"numCols":3,"color":"#66aa66"},
    {"label":"Gate B displayed","startRow":27,"numRows":8,"startCol":113,"numCols":3,"color":"#66aa66"},
    ]

var lineDisplayed = [
    {"label":"Middle Line","p1x":63.5,"p1y":1,"p2x":63.5,"p2y":61}
]
```

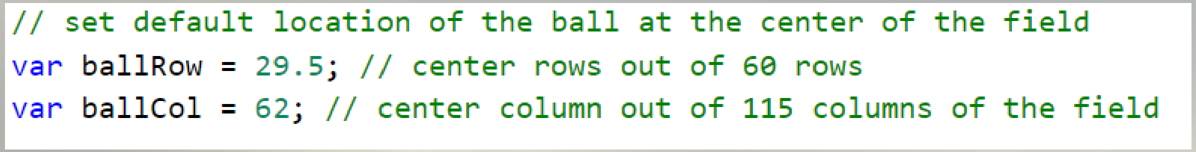


```
Score A: 0              Time: 0        Score B: 0
Total Score A: 0        Rounds: 0      Total Score B: 0
Total Wins A: 0         Draws: 0       Total Wins B: 0
```

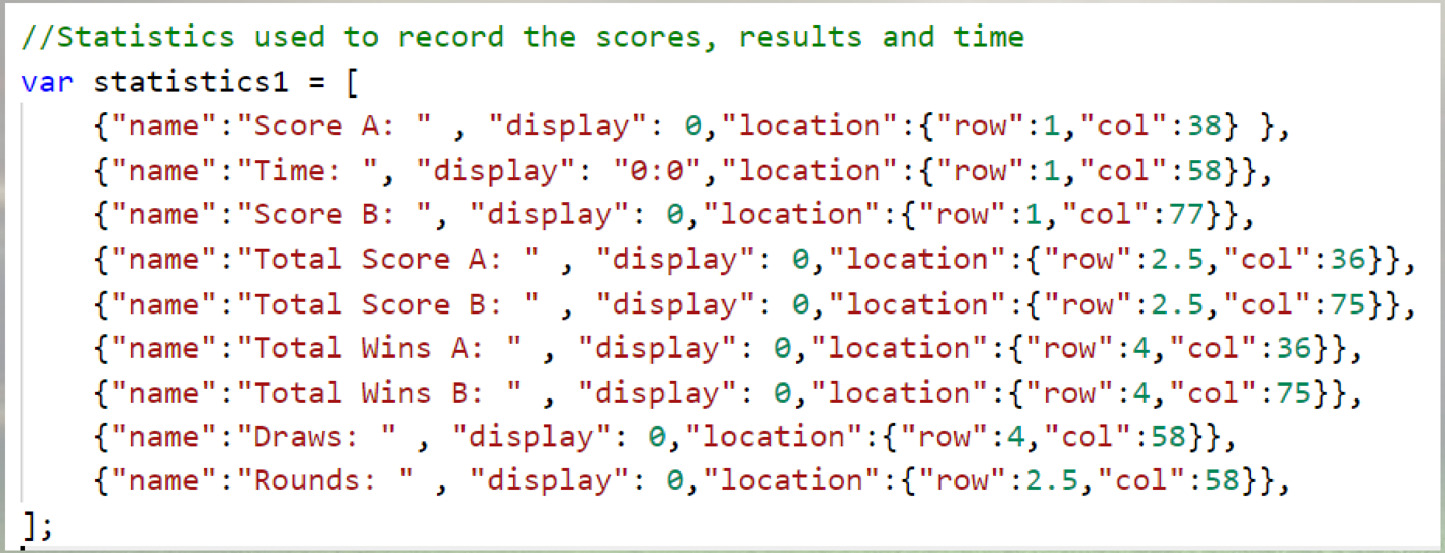
Location Information:



```
// set initial state of the ball.
var ball = [{"type":BALL,"label":"Ball","location":{"row":ballRow,"col":ballCol},
"target":{"row":ballRow,"col":ballCol},"state":KICKOFF ,"timeAdmitted":0}];
```



```
// set default location of the ball at the center of the field
var ballRow = 29.5; // center rows out of 60 rows
var ballCol = 62; // center column out of 115 columns of the field
```



```
//Statistics used to record the scores, results and time
var statistics1 = [
    {"name":"Score A: " , "display": 0,"location":{"row":1,"col":38} },
    {"name":"Time: ", "display": "0:0","location":{"row":1,"col":58}},
    {"name":"Score B: ", "display": 0,"location":{"row":1,"col":77}},
    {"name":"Total Score A: " , "display": 0,"location":{"row":2.5,"col":36}},
    {"name":"Total Score B: " , "display": 0,"location":{"row":2.5,"col":75}},
    {"name":"Total Wins A: " , "display": 0,"location":{"row":4,"col":36}},
    {"name":"Total Wins B: " , "display": 0,"location":{"row":4,"col":75}},
    {"name":"Draws: " , "display": 0,"location":{"row":4,"col":58}},
    {"name":"Rounds: " , "display": 0,"location":{"row":2.5,"col":58}},
];
```

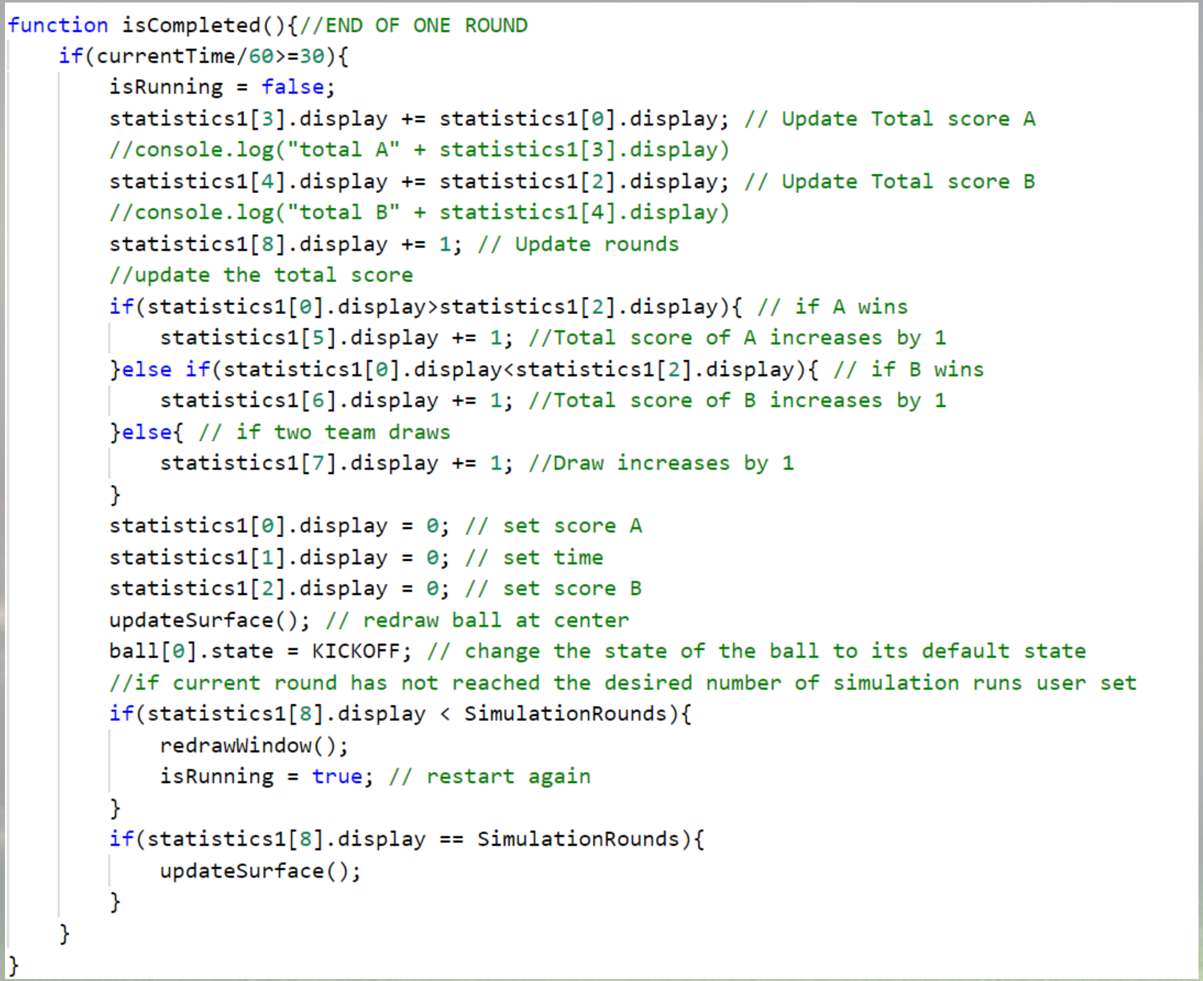
The Basic Simulation step

```
function simStep(){
    //This function is called by a timer; if running, it executes one simulation step
    //The timing interval is set in the page initialization function near the top of this file

    if (isRunning){ //the isRunning variable is toggled by toggleSimStep
        // Increment current time (for computing statistics)
        currentTime++;
        //display time
        statistics1[1].display = Math.floor(currentTime/20)+":"+(currentTime-Math.floor(currentTime/60)*60);
        updateSurface();
        //update the ball
        updateBall();
        //check whehter we have reached the end of the simulation
        isCompleted();
    }
}
```

We call this function repeatedly in the page’s initialization code:

```
simTimer = window.setInterval(simStep, animationDelay); // call the function simStep every animationDelay milliseconds
```

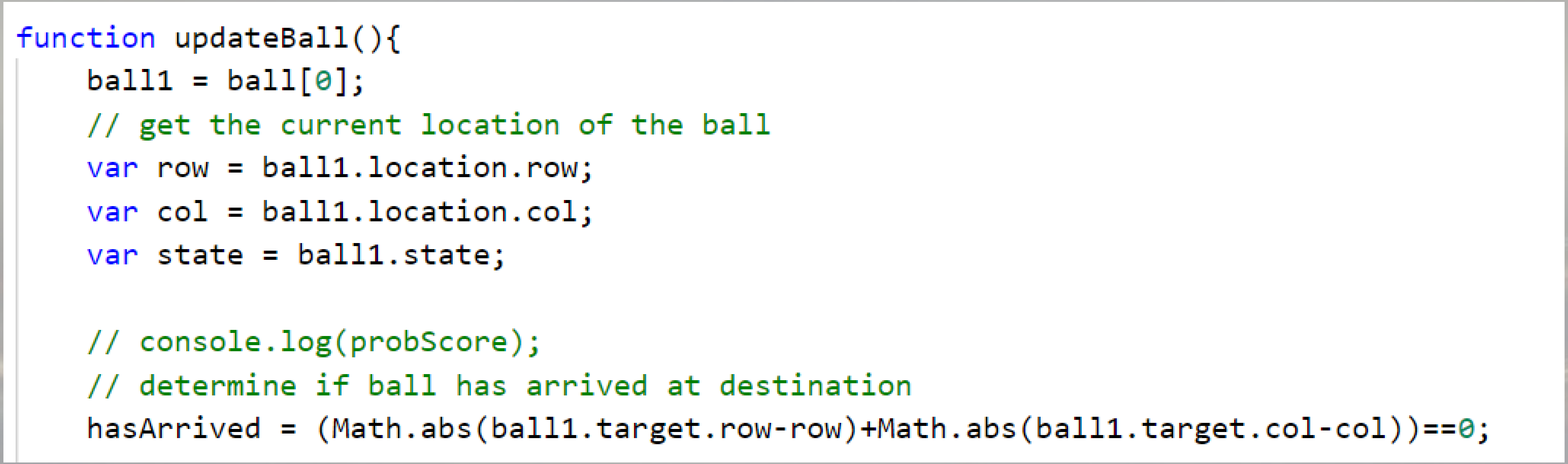
The Basic Simulation step



```
function isCompleted(){//END OF ONE ROUND
    if(currentTime/60>=30){
        isRunning = false;
        statistics1[3].display += statistics1[0].display; // Update Total score A
        //console.log("total A" + statistics1[3].display)
        statistics1[4].display += statistics1[2].display; // Update Total score B
        //console.log("total B" + statistics1[4].display)
        statistics1[8].display += 1; // Update rounds
        //update the total score
        if(statistics1[0].display>statistics1[2].display){ // if A wins
            statistics1[5].display += 1; //Total score of A increases by 1
        }else if(statistics1[0].display<statistics1[2].display){ // if B wins
            statistics1[6].display += 1; //Total score of B increases by 1
        }else{ // if two team draws
            statistics1[7].display += 1; //Draw increases by 1
        }
        statistics1[0].display = 0; // set score A
        statistics1[1].display = 0; // set time
        statistics1[2].display = 0; // set score B
        updateSurface(); // redraw ball at center
        ball[0].state = KICKOFF; // change the state of the ball to its default state
        //if current round has not reached the desired number of simulation runs user set
        if(statistics1[8].display < SimulationRounds){
            redrawWindow();
            isRunning = true; // restart again
        }
        if(statistics1[8].display == SimulationRounds){
            updateSurface();
        }
    }
}
```

Determine whether to reiterate the match or end the game

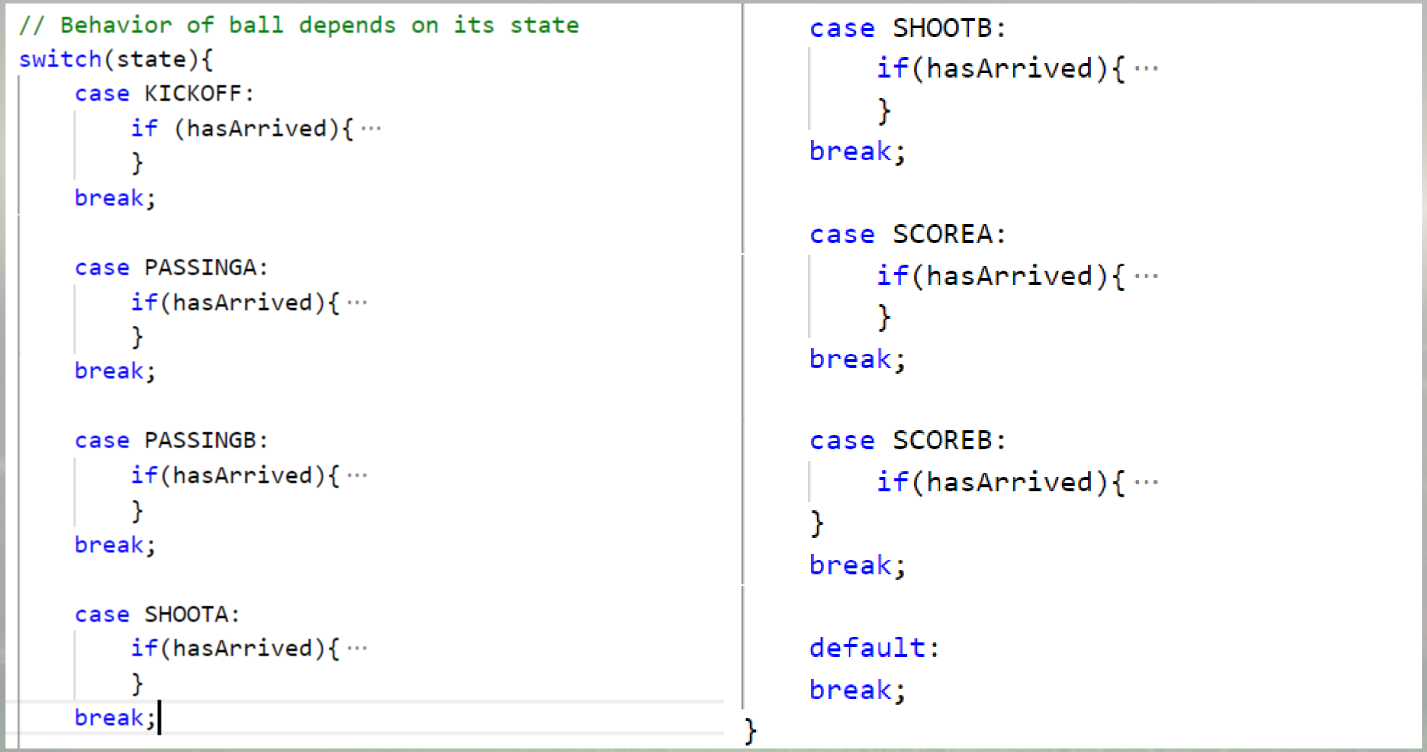
Update Agent:



```
function updateBall(){
    ball1 = ball[0];
    // get the current location of the ball
    var row = ball1.location.row;
    var col = ball1.location.col;
    var state = ball1.state;

    // console.log(probScore);
    // determine if ball has arrived at destination
    hasArrived = (Math.abs(ball1.target.row-row)+Math.abs(ball1.target.col-col))==0;
```

Structure of Update Code:

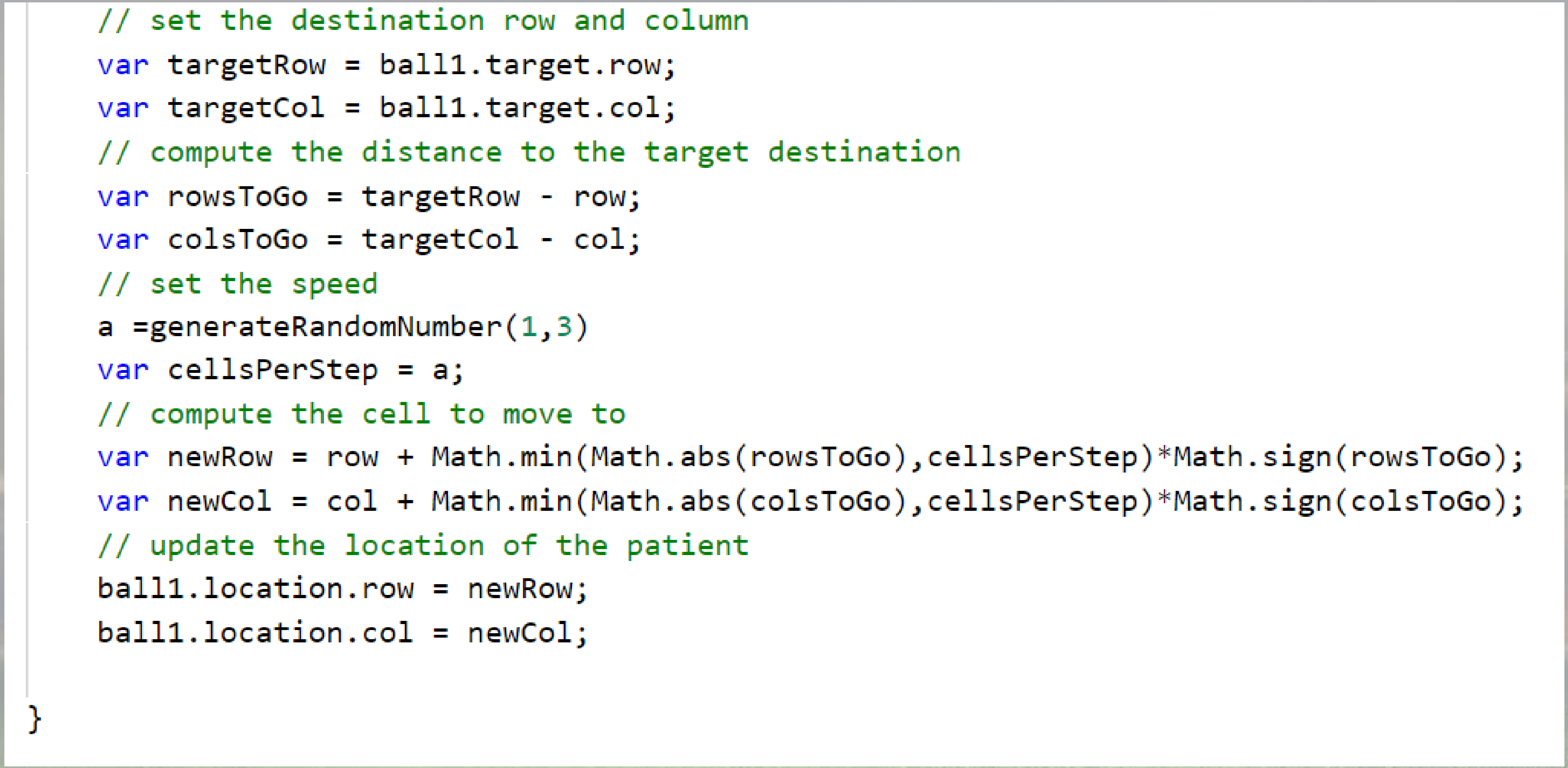
Nested switch statements are used to show various state explicitly and to handle all the different cases.



```
// Behavior of ball depends on its state
switch(state){
    case KICKOFF:
        if (hasArrived){⋯
        }
    break;

    case PASSINGA:
        if(hasArrived){⋯
        }
    break;

    case PASSINGB:
        if(hasArrived){⋯
        }
    break;

    case SHOOTA:
        if(hasArrived){⋯
        }
    break;
```

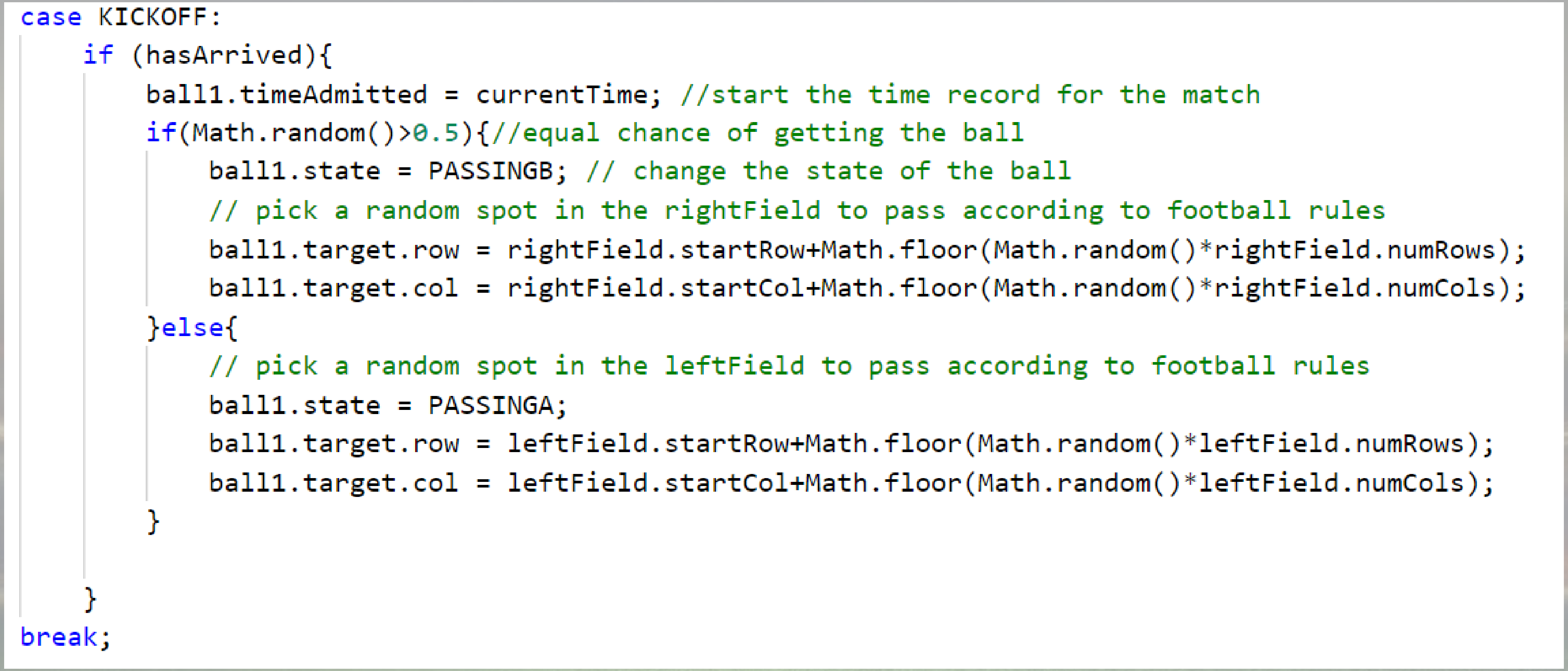
```
    case SHOOTB:
        if(hasArrived){⋯
        }
    break;

    case SCOREA:
        if(hasArrived){⋯
        }
    break;

    case SCOREB:
        if(hasArrived){⋯
    }
    break;

    default:
    break;
}
```

Code to handle movement:



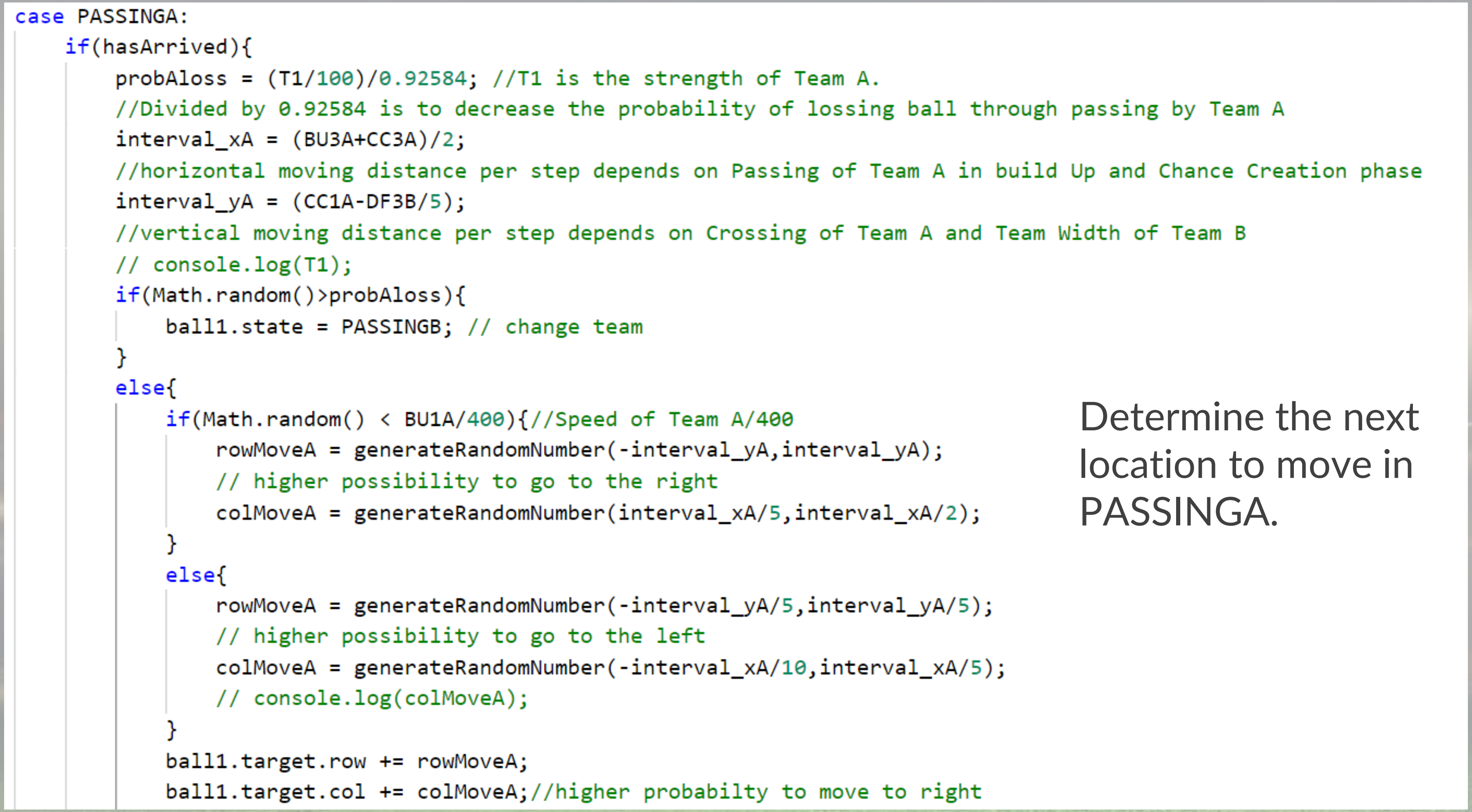
```
    // set the destination row and column
    var targetRow = ball1.target.row;
    var targetCol = ball1.target.col;
    // compute the distance to the target destination
    var rowsToGo = targetRow - row;
    var colsToGo = targetCol - col;
    // set the speed
    a =generateRandomNumber(1,3)
    var cellsPerStep = a;
    // compute the cell to move to
    var newRow = row + Math.min(Math.abs(rowsToGo),cellsPerStep)*Math.sign(rowsToGo);
    var newCol = col + Math.min(Math.abs(colsToGo),cellsPerStep)*Math.sign(colsToGo);
    // update the location of the patient
    ball1.location.row = newRow;
    ball1.location.col = newCol;

}
```

Handle State KICKOFF:



```
case KICKOFF:
    if (hasArrived){
        ball1.timeAdmitted = currentTime; //start the time record for the match
        if(Math.random()>0.5){//equal chance of getting the ball
            ball1.state = PASSINGB; // change the state of the ball
            // pick a random spot in the rightField to pass according to football rules
            ball1.target.row = rightField.startRow+Math.floor(Math.random()*rightField.numRows);
            ball1.target.col = rightField.startCol+Math.floor(Math.random()*rightField.numCols);
        }else{
            // pick a random spot in the leftField to pass according to football rules
            ball1.state = PASSINGA;
            ball1.target.row = leftField.startRow+Math.floor(Math.random()*leftField.numRows);
            ball1.target.col = leftField.startCol+Math.floor(Math.random()*leftField.numCols);
        }


    }
break;
```

Handle State PASSINGA (Part 1):



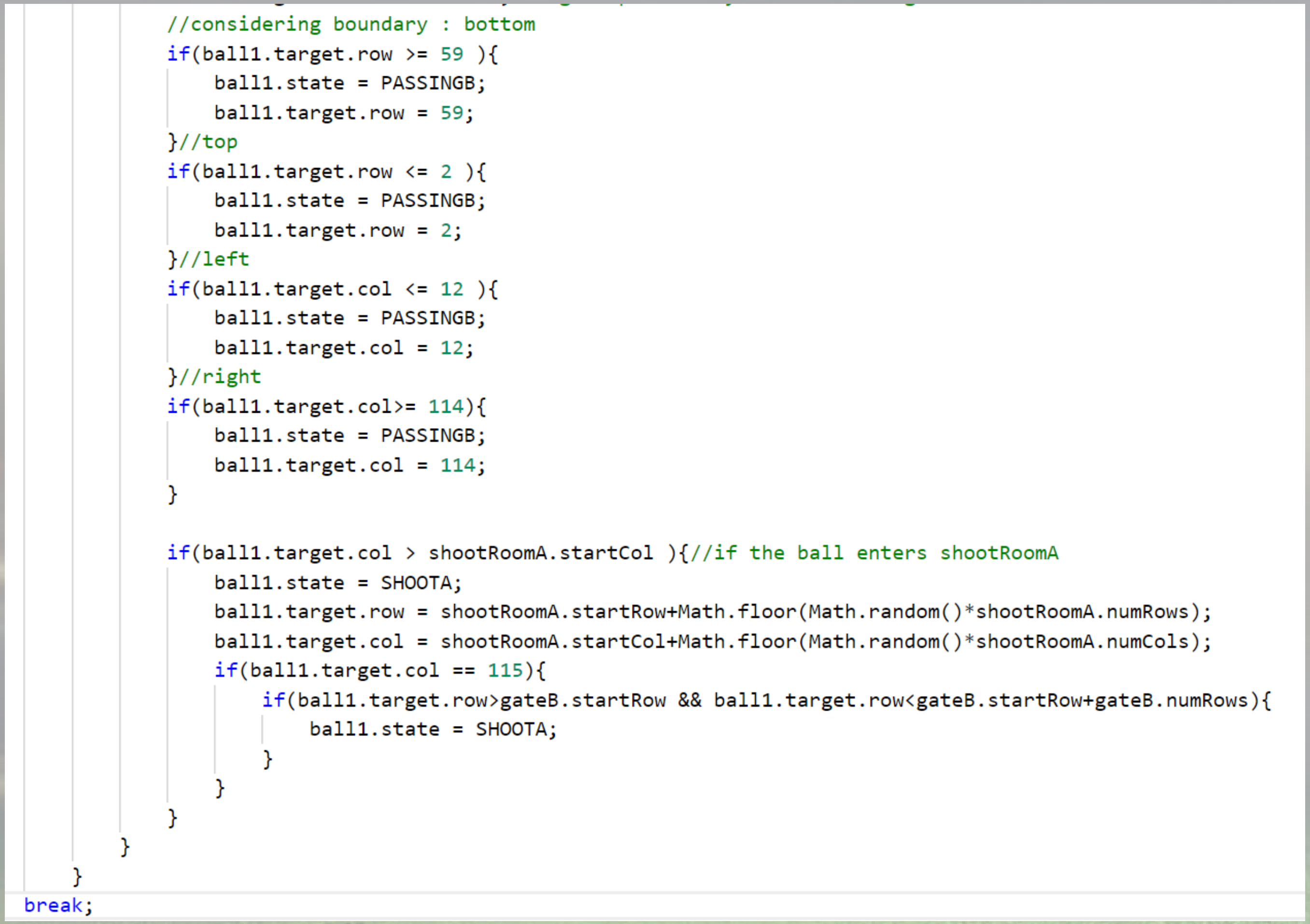
```
case PASSINGA:
    if(hasArrived){
        probAloss = (T1/100)/0.92584; //T1 is the strength of Team A.
        //Divided by 0.92584 is to decrease the probability of lossing ball through passing by Team A
        interval_xA = (BU3A+CC3A)/2;
        //horizontal moving distance per step depends on Passing of Team A in build Up and Chance Creation phase
        interval_yA = (CC1A-DF3B/5);
        //vertical moving distance per step depends on Crossing of Team A and Team Width of Team B
        // console.log(T1);
        if(Math.random()>probAloss){
            ball1.state = PASSINGB; // change team
        }
        else{
            if(Math.random() < BU1A/400){//Speed of Team A/400
                rowMoveA = generateRandomNumber(-interval_yA,interval_yA);
                // higher possibility to go to the right
                colMoveA = generateRandomNumber(interval_xA/5,interval_xA/2);
            }
            else{
                rowMoveA = generateRandomNumber(-interval_yA/5,interval_yA/5);
                // higher possibility to go to the left
                colMoveA = generateRandomNumber(-interval_xA/10,interval_xA/5);
                // console.log(colMoveA);
            }
            ball1.target.row += rowMoveA;
            ball1.target.col += colMoveA;//higher probabilty to move to right
```

Determine the next location to move in PASSINGA.

Ball tends to move more to the right side as it is controlled by Team A who want to score at the right side.

The equations we used here are based on our researches and data analysis.

Handle State PASSINGA (Part 2):



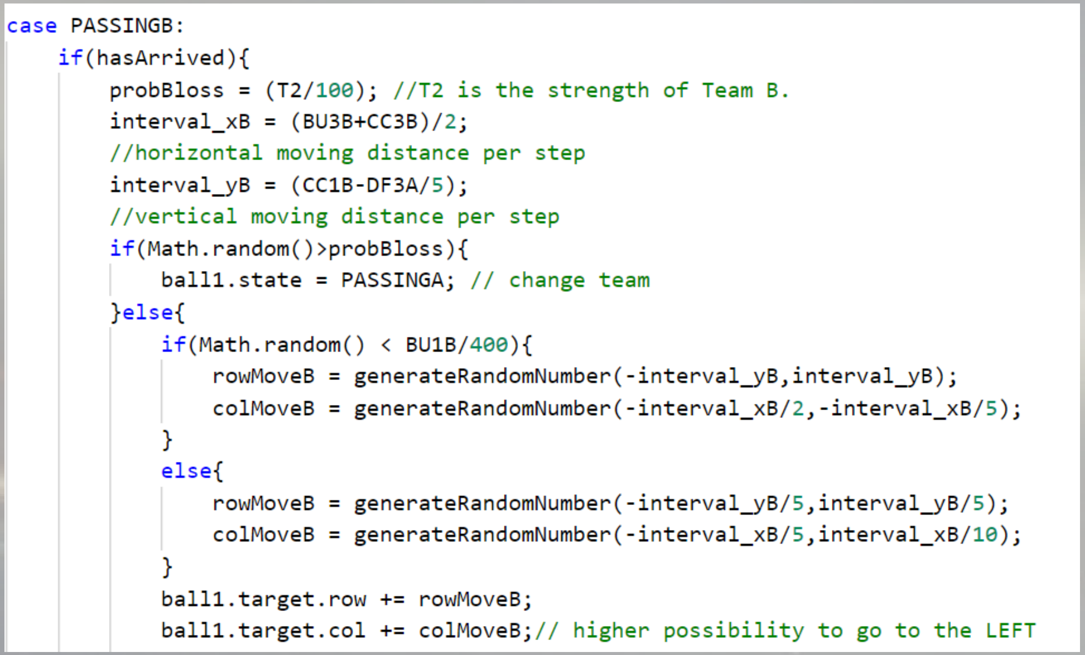
```
            //considering boundary : bottom
            if(ball1.target.row >= 59 ){
                ball1.state = PASSINGB;
                ball1.target.row = 59;
            }//top
            if(ball1.target.row <= 2 ){
                ball1.state = PASSINGB;
                ball1.target.row = 2;
            }//left
            if(ball1.target.col <= 12 ){
                ball1.state = PASSINGB;
                ball1.target.col = 12;
            }//right
            if(ball1.target.col>= 114){
                ball1.state = PASSINGB;
                ball1.target.col = 114;
            }

            if(ball1.target.col > shootRoomA.startCol ){//if the ball enters shootRoomA
                ball1.state = SHOOTA;
                ball1.target.row = shootRoomA.startRow+Math.floor(Math.random()*shootRoomA.numRows);
                ball1.target.col = shootRoomA.startCol+Math.floor(Math.random()*shootRoomA.numCols);
                if(ball1.target.col == 115){
                    if(ball1.target.row>gateB.startRow && ball1.target.row<gateB.startRow+gateB.numRows){
                        ball1.state = SHOOTA;
                    }
                }
            }
        }
    }
break;
```

Ensure the ball is within the boundary.

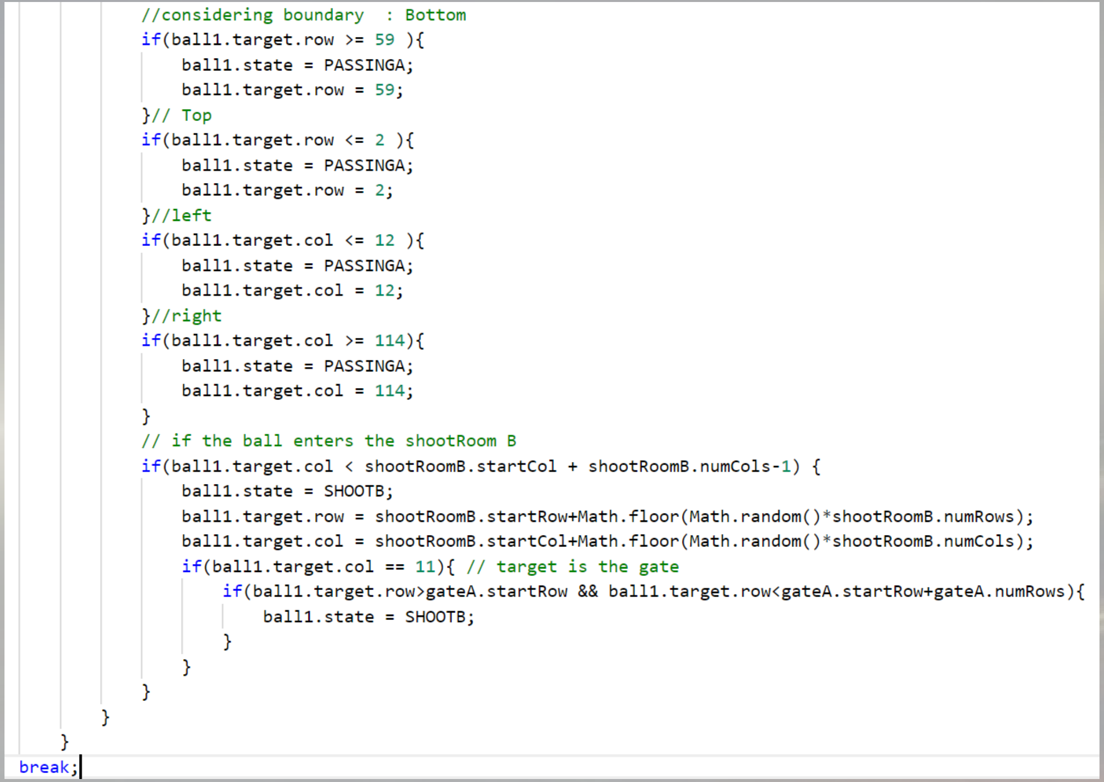
If it is out of the boundary, change ball possessing team.

Change the state of the ball if the targets is shootRoomA

Handle State PASSINGB: *similar to PASSINGA*



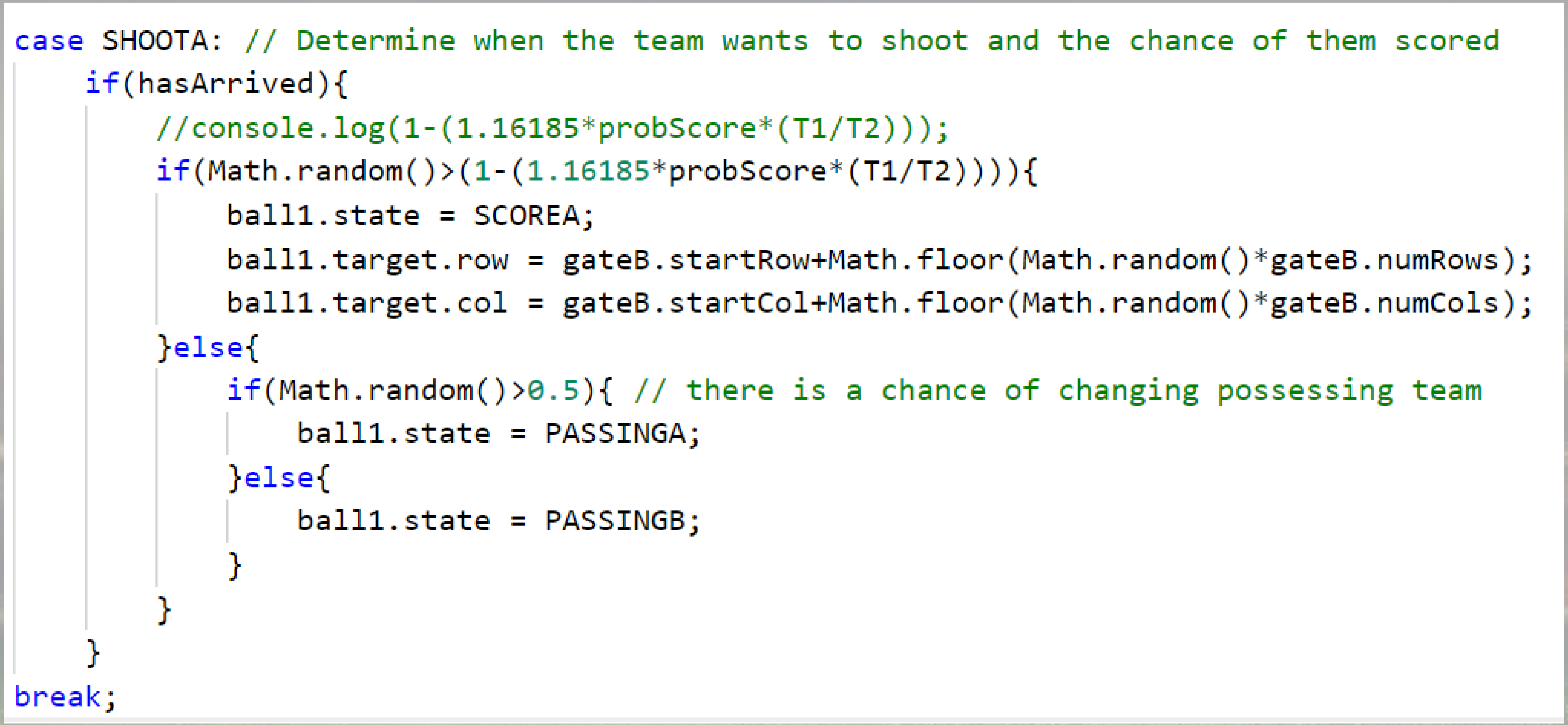
```
case PASSINGB:
    if(hasArrived){
        probBloss = (T2/100); //T2 is the strength of Team B.
        interval_xB = (BU3B+CC3B)/2;
        //horizontal moving distance per step
        interval_yB = (CC1B-DF3A/5);
        //vertical moving distance per step
        if(Math.random()>probBloss){
            ball1.state = PASSINGA; // change team
        }else{
            if(Math.random() < BU1B/400){
                rowMoveB = generateRandomNumber(-interval_yB,interval_yB);
                colMoveB = generateRandomNumber(-interval_xB/2,-interval_xB/5);
            }
            else{
                rowMoveB = generateRandomNumber(-interval_yB/5,interval_yB/5);
                colMoveB = generateRandomNumber(-interval_xB/5,interval_xB/10);
            }
            ball1.target.row += rowMoveB;
            ball1.target.col += colMoveB;// higher possibility to go to the LEFT
```



```
            //considering boundary  : Bottom
            if(ball1.target.row >= 59 ){
                ball1.state = PASSINGA;
                ball1.target.row = 59;
            }// Top
            if(ball1.target.row <= 2 ){
                ball1.state = PASSINGA;
                ball1.target.row = 2;
            }//left
            if(ball1.target.col <= 12 ){
                ball1.state = PASSINGA;
                ball1.target.col = 12;
            }//right
            if(ball1.target.col >= 114){
                ball1.state = PASSINGA;
                ball1.target.col = 114;
            }
            // if the ball enters the shootRoom B
            if(ball1.target.col < shootRoomB.startCol + shootRoomB.numCols-1) {
                ball1.state = SHOOTB;
                ball1.target.row = shootRoomB.startRow+Math.floor(Math.random()*shootRoomB.numRows);
                ball1.target.col = shootRoomB.startCol+Math.floor(Math.random()*shootRoomB.numCols);
                if(ball1.target.col == 11){ // target is the gate
                    if(ball1.target.row>gateA.startRow && ball1.target.row<gateA.startRow+gateA.numRows){
                        ball1.state = SHOOTB;
                    }
                }
            }
        }
    }
break;
```

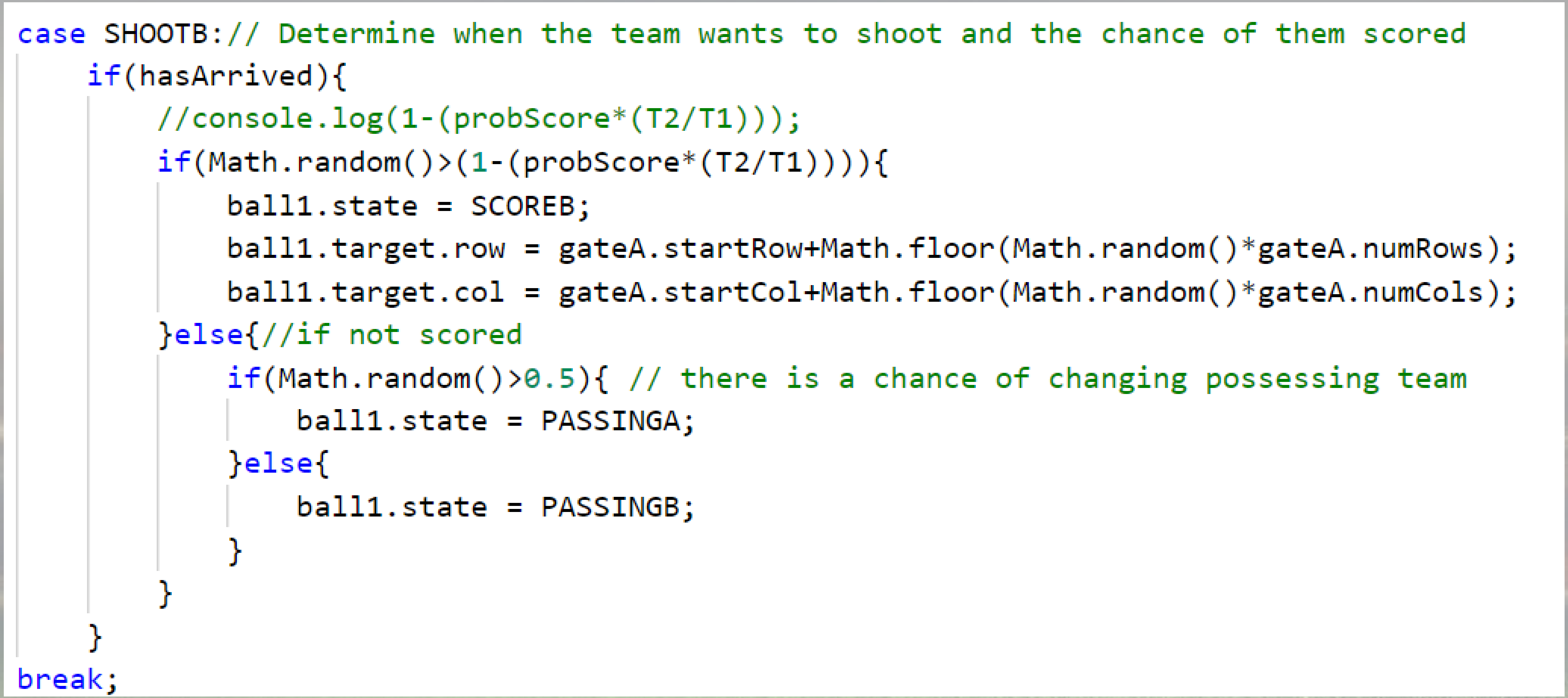
Determine the next location to move in PASSINGB.
Ball tends to move more to the right side as it is controlled by Team B who want to score at the right side.
The equations we used here are based on our researches and data analysis.

Handle State SHOOTA:



```
case SHOOTA: // Determine when the team wants to shoot and the chance of them scored
    if(hasArrived){
        //console.log(1-(1.16185*probScore*(T1/T2)));
        if(Math.random()>(1-(1.16185*probScore*(T1/T2)))){
            ball1.state = SCOREA;
            ball1.target.row = gateB.startRow+Math.floor(Math.random()*gateB.numRows);
            ball1.target.col = gateB.startCol+Math.floor(Math.random()*gateB.numCols);
        }else{
            if(Math.random()>0.5){ // there is a chance of changing possessing team
                ball1.state = PASSINGA;
            }else{
                ball1.state = PASSINGB;
            }
        }
    }
break;
```

Handle State SHOOTB: *similar to SHOOTA*

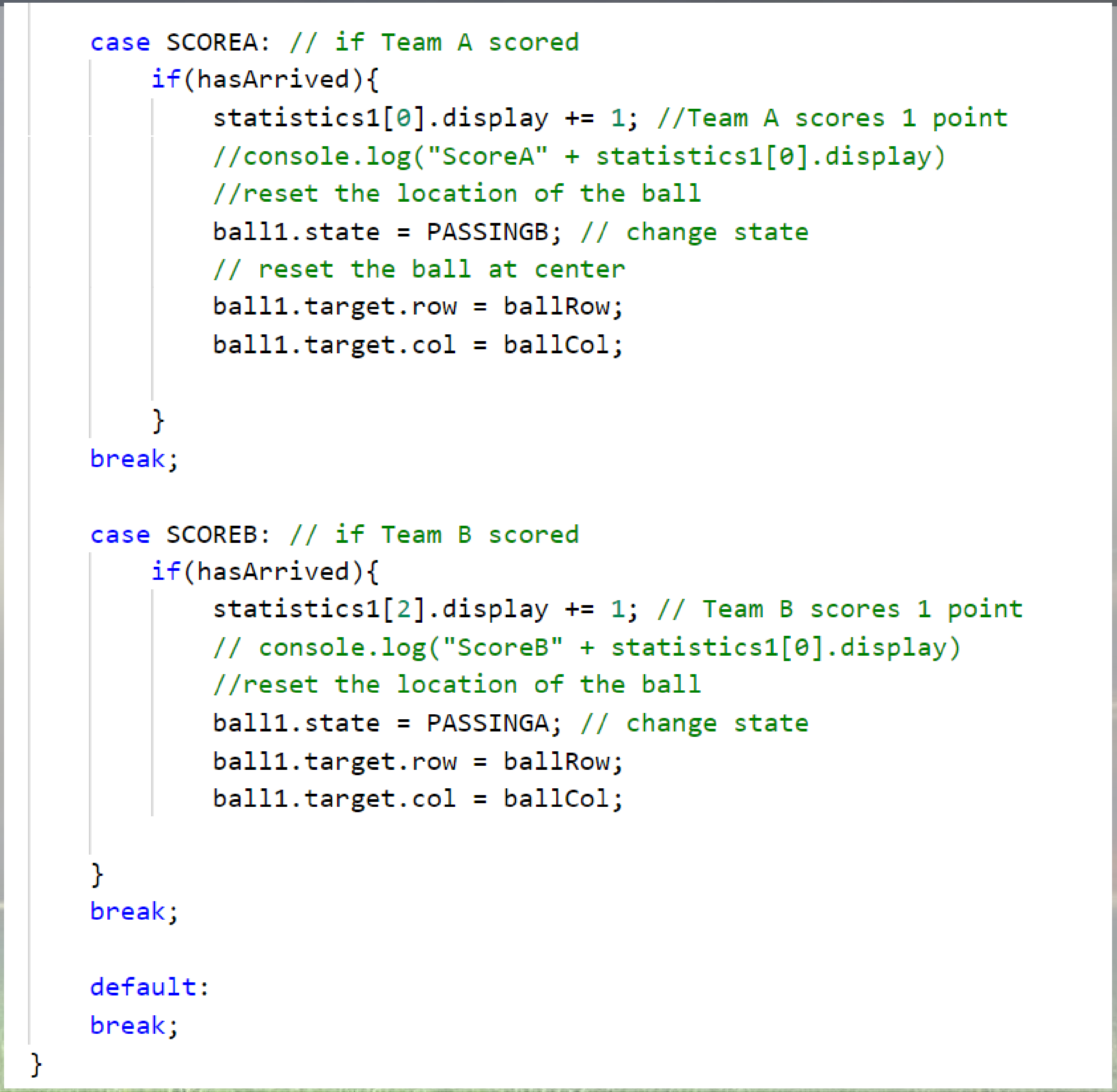


```
case SHOOTB:// Determine when the team wants to shoot and the chance of them scored
    if(hasArrived){
        //console.log(1-(probScore*(T2/T1)));
        if(Math.random()>(1-(probScore*(T2/T1)))){
            ball1.state = SCOREB;
            ball1.target.row = gateA.startRow+Math.floor(Math.random()*gateA.numRows);
            ball1.target.col = gateA.startCol+Math.floor(Math.random()*gateA.numCols);
        }else{//if not scored
            if(Math.random()>0.5){ // there is a chance of changing possessing team
                ball1.state = PASSINGA;
            }else{
                ball1.state = PASSINGB;
            }
        }
    }
break;
```

Handle State SCOREA & SCOREB:

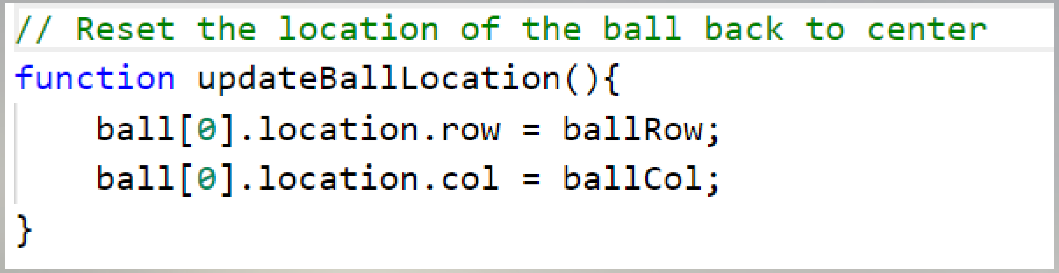
Update scores of both team is they scored

Change the possessing team and reset the ball to the centre

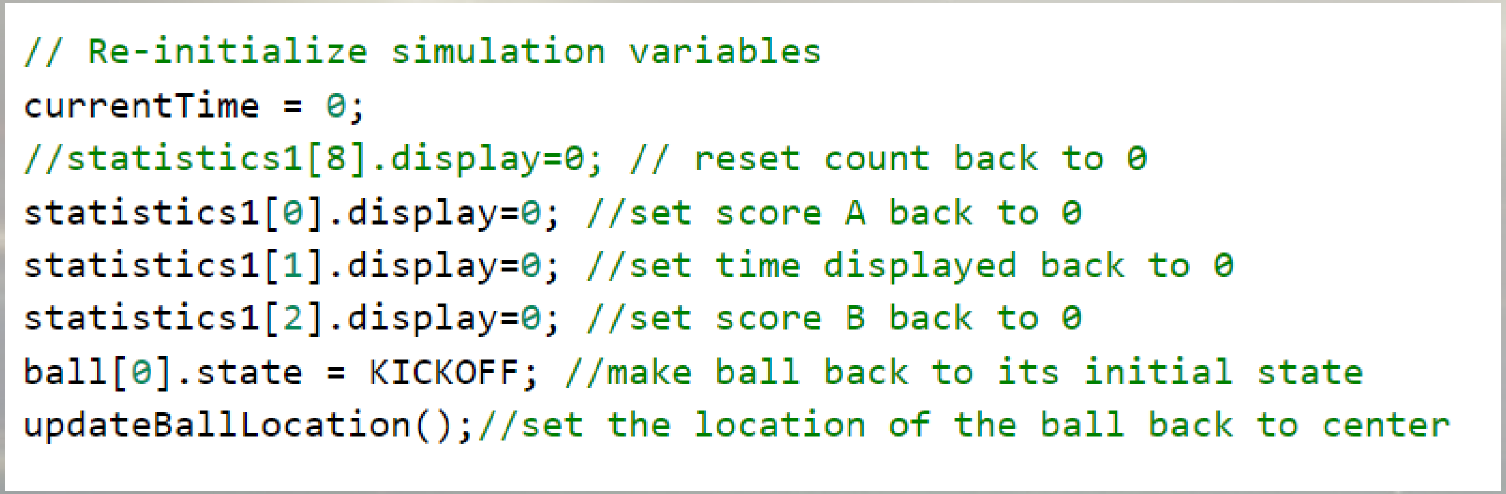


```
    case SCOREA: // if Team A scored
        if(hasArrived){
            statistics1[0].display += 1; //Team A scores 1 point
            //console.log("ScoreA" + statistics1[0].display)
            //reset the location of the ball
            ball1.state = PASSINGB; // change state
            // reset the ball at center
            ball1.target.row = ballRow;
            ball1.target.col = ballCol;

        }
    break;

    case SCOREB: // if Team B scored
        if(hasArrived){
            statistics1[2].display += 1; // Team B scores 1 point
            // console.log("ScoreB" + statistics1[0].display)
            //reset the location of the ball
            ball1.state = PASSINGA; // change state
            ball1.target.row = ballRow;
            ball1.target.col = ballCol;

    }
    break;

    default:
    break;
}
```

Data Initialization:



```
// Reset the location of the ball back to center
function updateBallLocation(){
    ball[0].location.row = ballRow;
    ball[0].location.col = ballCol;
}
```

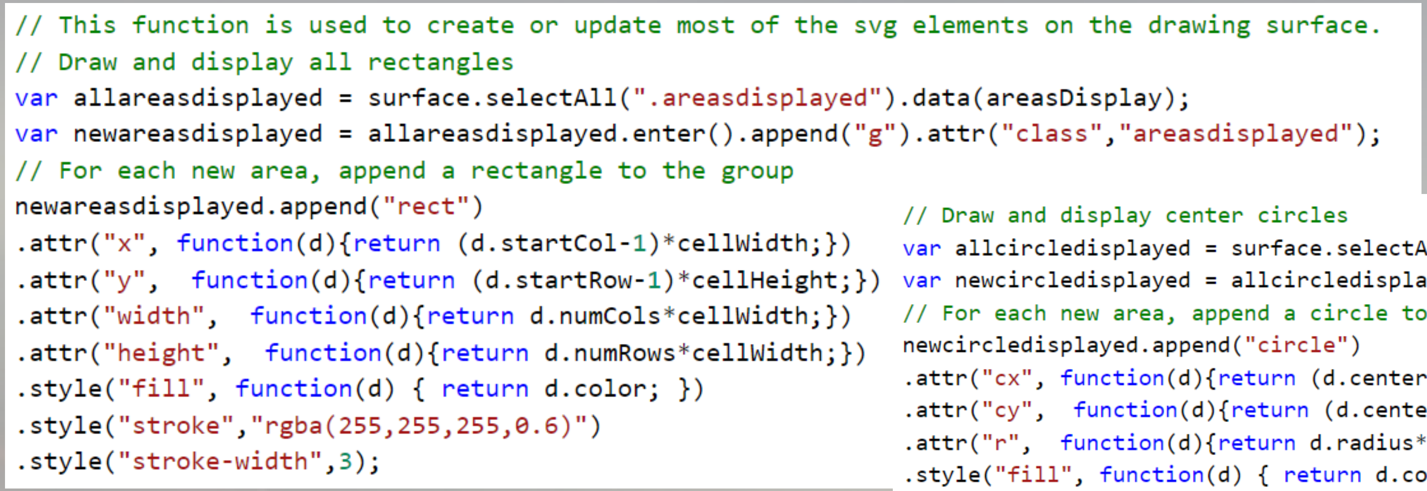


```
// Re-initialize simulation variables
currentTime = 0;
//statistics1[8].display=0; // reset count back to 0
statistics1[0].display=0; //set score A back to 0
statistics1[1].display=0; //set time displayed back to 0
statistics1[2].display=0; //set score B back to 0
ball[0].state = KICKOFF; //make ball back to its initial state
updateBallLocation();//set the location of the ball back to center
```

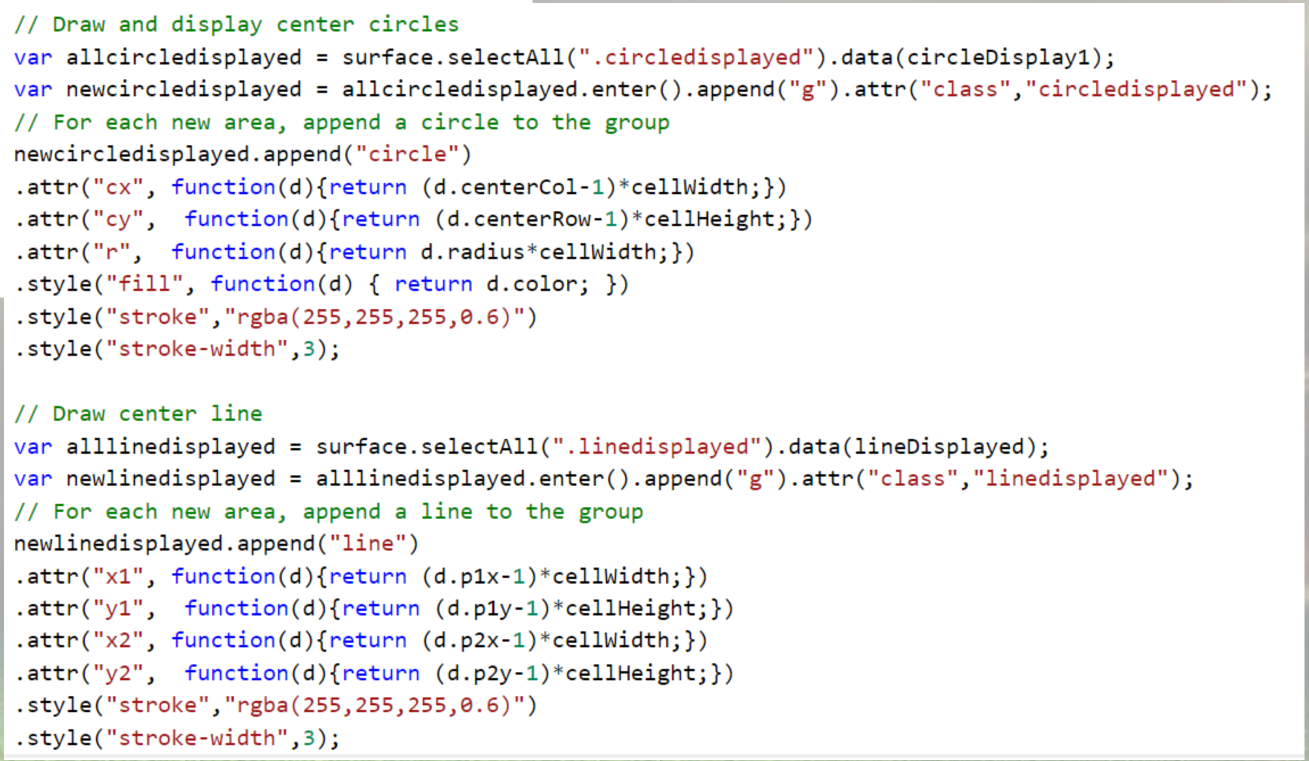
D3 Part of JavaScript

Agent-Based Modeling

Display the soccer field:



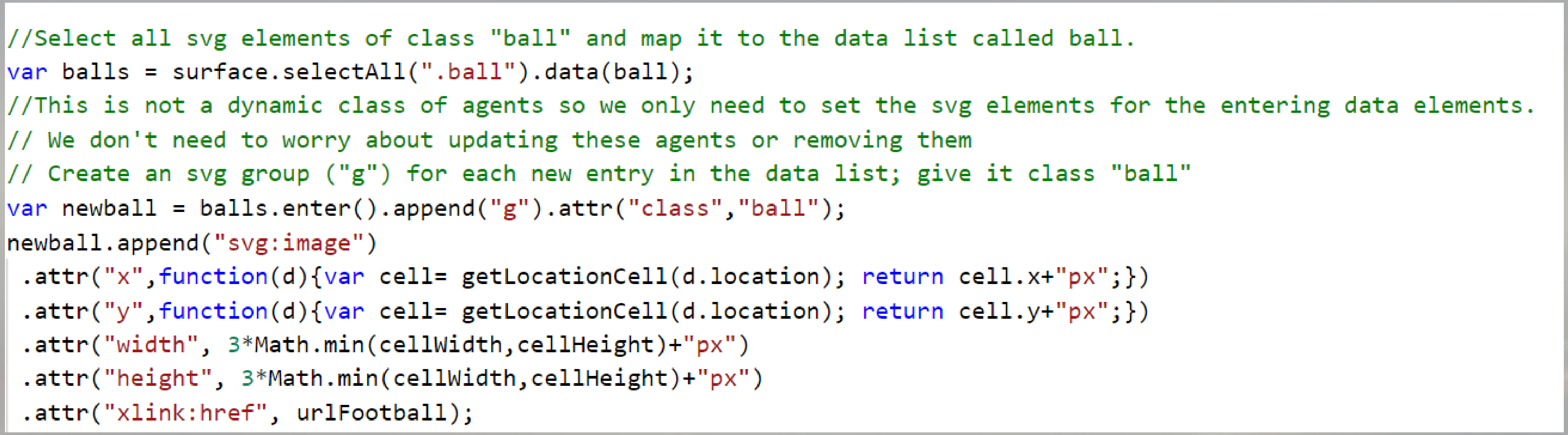
```
// This function is used to create or update most of the svg elements on the drawing surface.
// Draw and display all rectangles
var allareasdisplayed = surface.selectAll(".areasdisplayed").data(areasDisplay);
var newareasdisplayed = allareasdisplayed.enter().append("g").attr("class","areasdisplayed");
// For each new area, append a rectangle to the group
newareasdisplayed.append("rect")
.attr("x", function(d){return (d.startCol-1)*cellWidth;})
.attr("y",  function(d){return (d.startRow-1)*cellHeight;})
.attr("width",  function(d){return d.numCols*cellWidth;})
.attr("height",  function(d){return d.numRows*cellWidth;})
.style("fill", function(d) { return d.color; })
.style("stroke","rgba(255,255,255,0.6)")
.style("stroke-width",3);
```



```
// Draw and display center circles
var allcircledisplayed = surface.selectAll(".circledisplayed").data(circleDisplay1);
var newcircledisplayed = allcircledisplayed.enter().append("g").attr("class","circledisplayed");
// For each new area, append a circle to the group
newcircledisplayed.append("circle")
.attr("cx", function(d){return (d.centerCol-1)*cellWidth;})
.attr("cy",  function(d){return (d.centerRow-1)*cellHeight;})
.attr("r",  function(d){return d.radius*cellWidth;})
.style("fill", function(d) { return d.color; })
.style("stroke","rgba(255,255,255,0.6)")
.style("stroke-width",3);

// Draw center line
var alllinedisplayed = surface.selectAll(".linedisplayed").data(lineDisplayed);
var newlinedisplayed = alllinedisplayed.enter().append("g").attr("class","linedisplayed");
// For each new area, append a line to the group
newlinedisplayed.append("line")
.attr("x1", function(d){return (d.p1x-1)*cellWidth;})
.attr("y1",  function(d){return (d.p1y-1)*cellHeight;})
.attr("x2", function(d){return (d.p2x-1)*cellWidth;})
.attr("y2",  function(d){return (d.p2y-1)*cellHeight;})
.style("stroke","rgba(255,255,255,0.6)")
.style("stroke-width",3);
```

Initially there was no elements of class “areasdisplayed”, “circledisplayed”, or “linedisplayed” so we used the entering list of data elements to create the matching list of svg elements and append rectangle, circle and line accordingly.

Display the ball :



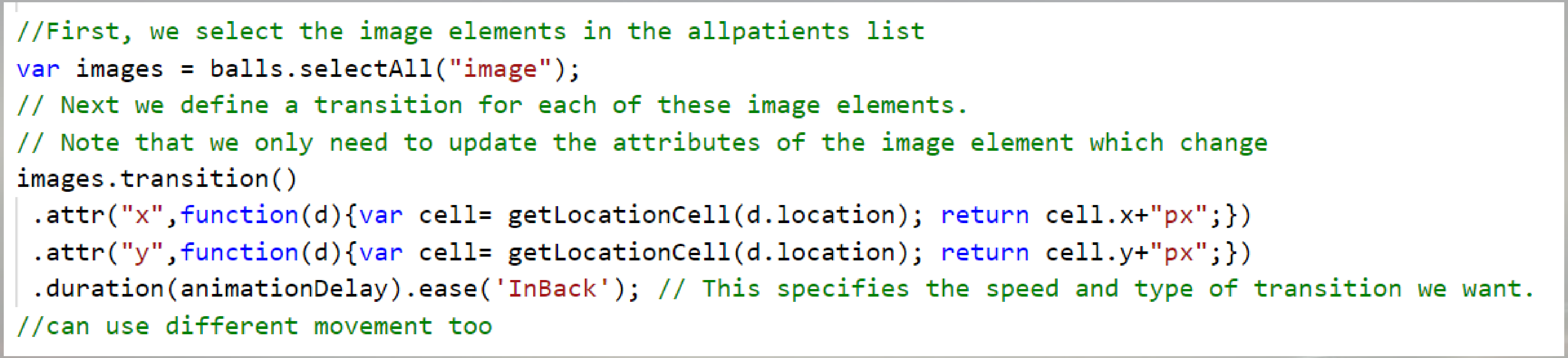
```
//Select all svg elements of class "ball" and map it to the data list called ball.
var balls = surface.selectAll(".ball").data(ball);
//This is not a dynamic class of agents so we only need to set the svg elements for the entering data elements.
// We don't need to worry about updating these agents or removing them
// Create an svg group ("g") for each new entry in the data list; give it class "ball"
var newball = balls.enter().append("g").attr("class","ball");
newball.append("svg:image")
 .attr("x",function(d){var cell= getLocationCell(d.location); return cell.x+"px";})
 .attr("y",function(d){var cell= getLocationCell(d.location); return cell.y+"px";})
 .attr("width", 3*Math.min(cellWidth,cellHeight)+"px")
 .attr("height", 3*Math.min(cellWidth,cellHeight)+"px")
 .attr("xlink:href", urlFootball);
```

We append an svg group, element g to the svg element and assign ball to the class attribute.

There will be one new group for each element of the enter() array.

We append the image of the football to the group element.

Update location of the ball (Animate the movement) :



```
//First, we select the image elements in the allpatients list
var images = balls.selectAll("image");
// Next we define a transition for each of these image elements.
// Note that we only need to update the attributes of the image element which change
images.transition()
 .attr("x",function(d){var cell= getLocationCell(d.location); return cell.x+"px";})
 .attr("y",function(d){var cell= getLocationCell(d.location); return cell.y+"px";})
 .duration(animationDelay).ease('InBack'); // This specifies the speed and type of transition we want.
//can use different movement too
```

The ball's location is updated by the simulation every animation Delay milliseconds.
Transition() function here makes it appears as a continuous movement.

Display the Statistics :

```
// The simulation should serve some purpose
// so we will compute and display the score of Team A and Team B;
// We created the array "statistics1" for this purpose.
// Here we will create a group for each element of the statistics array (two elements)
var allstatistics1 = surface.selectAll(".statistics1").data(statistics1);
var newstatistics1 = allstatistics1.enter().append("g").attr("class","statistics1");
// For each new statistic group created we append a text label
newstatistics1.append("text")
.attr("x", function(d) { var cell= getLocationCell(d.location); return (cell.x+cellWidth)+"px"; })
.attr("y", function(d) { var cell= getLocationCell(d.location); return (cell.y+cellHeight/2)+"px"; })
.attr("dy", ".65em")
.text("");

// The data in the statistics array are always being updated.
// So, here we update the text in the labels with the updated information.
allstatistics1.selectAll("text").text(function(d) {
    return d.name+ d.display;
});
```