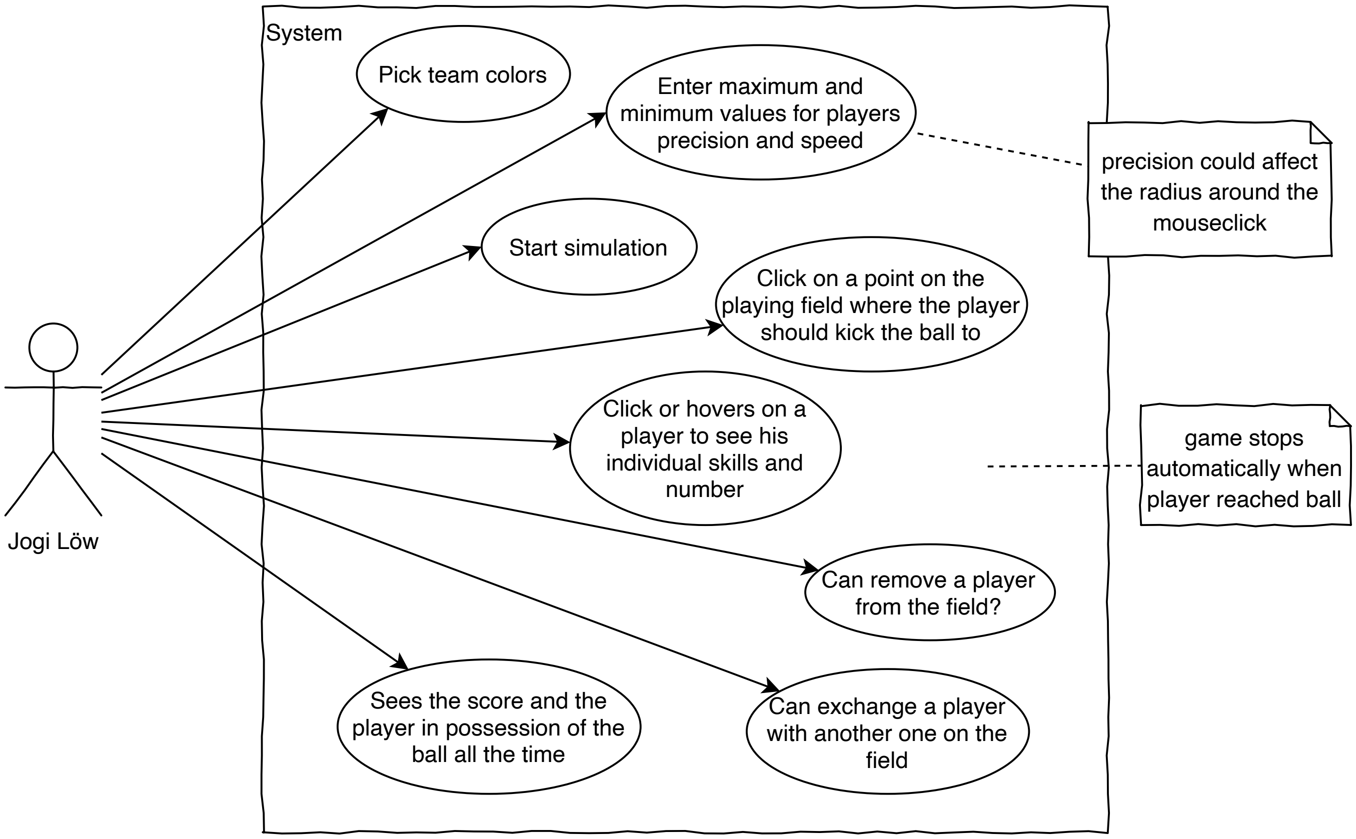
Soccer Training Simulation - Complete Documentation

The whole production of this application was developed in collaboration of Mona Stingl and Hannah Dürr with equal contribution of each team member to the final result.

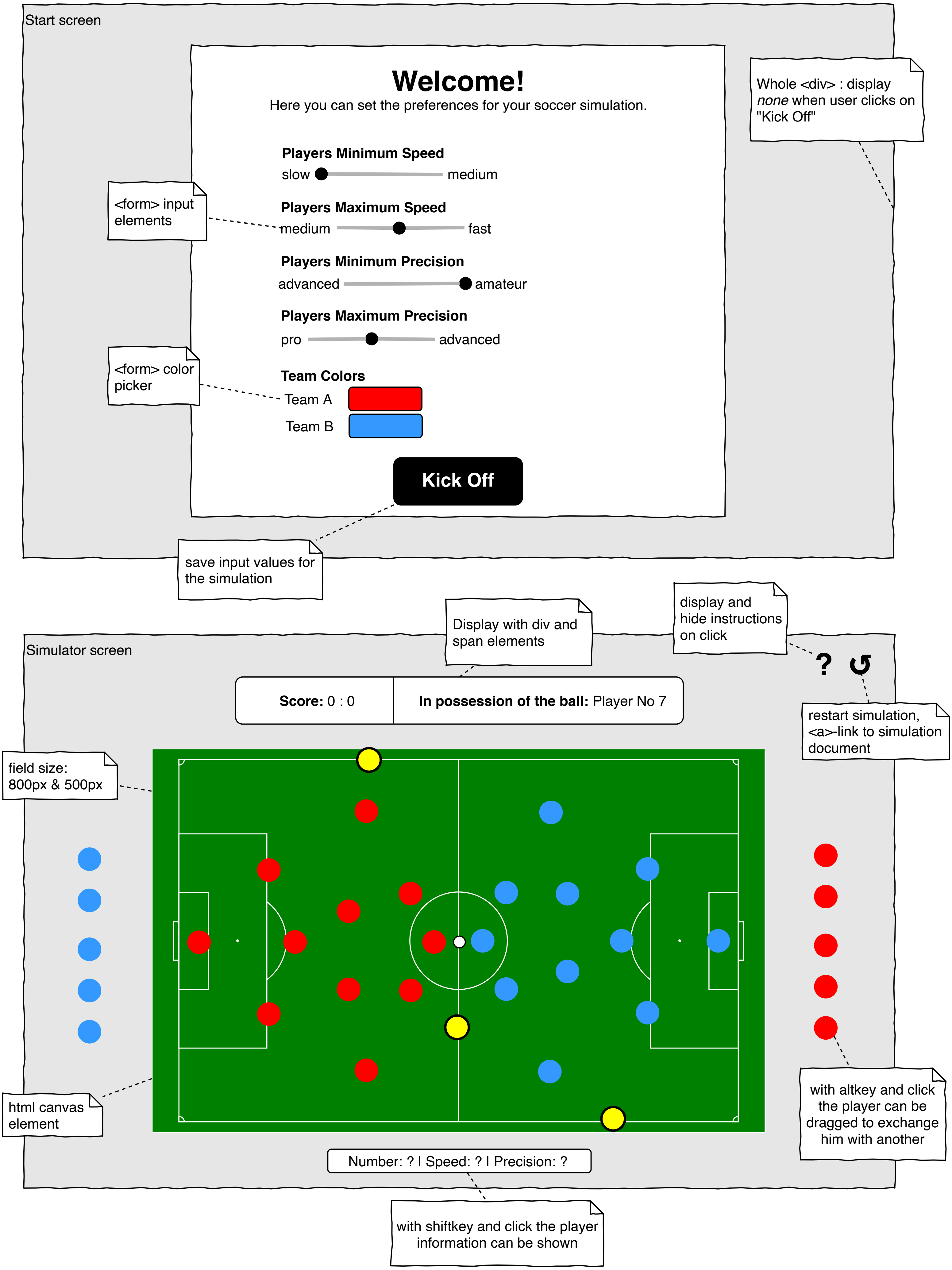
Date of completion: 19/07/2021

Content

Soccer Simulator Use Case Diagram


System
Pick team colors
Enter maximum and minimum values for players precision and speed
precision could affect the radius around the mouseclick
Start simulation
Click on a point on the playing field where the player should kick the ball to
Click or hovers on a player to see his individual skills and number
game stops automatically when player reached ball
Jogi Löw
Can remove a player from the field?
Sees the score and the player in possession of the ball all the time
Can exchange a player with another one on the field

Soccer Simulator User Interface Scribble



Start screen

Whole <div> : display *none* when user clicks on "Kick Off"

Welcome!

Here you can set the preferences for your soccer simulation.

Players Minimum Speed

slow ●──────── medium

<form> input elements

Players Maximum Speed

medium ─────●──── fast

Players Minimum Precision

advanced ────────● amateur

Players Maximum Precision

pro ─────●──── advanced

<form> color picker

Team Colors

Team A 🟥

Team B 🟦

Kick Off

save input values for the simulation

Simulator screen

Display with div and span elements

display and hide instructions on click

? ↺

restart simulation, <a>-link to simulation document

Score: 0 : 0 | **In possession of the ball:** Player No 7

field size: 800px & 500px

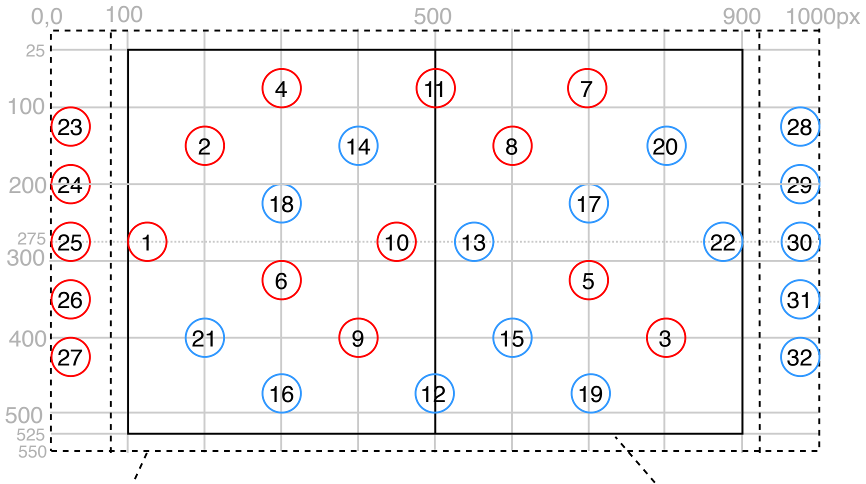
html canvas element

with altkey and click the player can be dragged to exchange him with another

Number: ? I Speed: ? I Precision: ?

with shiftkey and click the player information can be shown

Scribble for exact canvas values

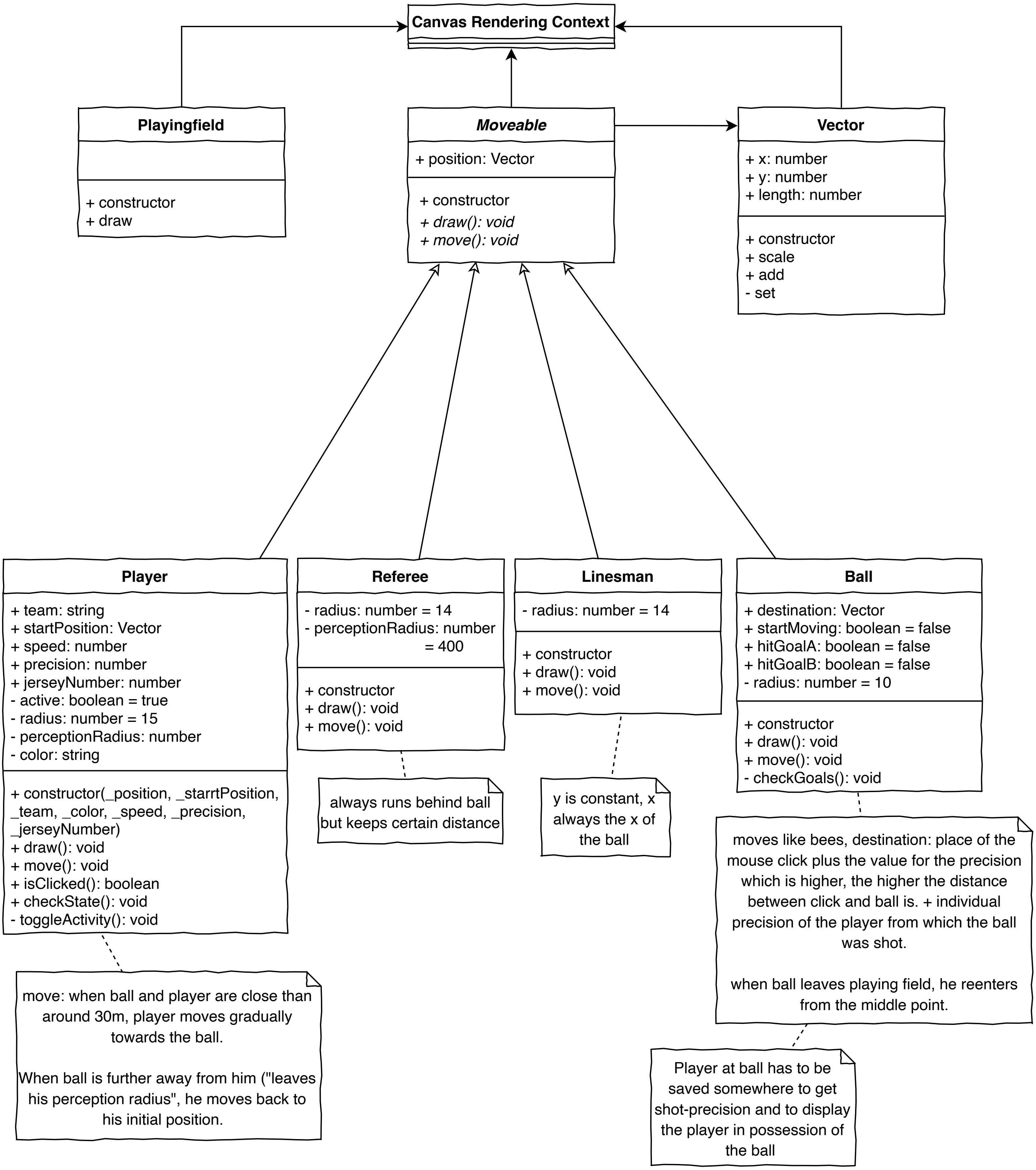


Canvas width: 1000px.
Canvas height: 550px.

Playing field: translate to 75, 0, width of the grass field 850px, height 550px. Borderline of actual field starts at 100, 25, width 800px, height 500px.

```
playerInformation [
{x: 135, y: 275, team: "A"},
{x: 180, y: 100, team: "A"},
{x: 180, y: 450, team: "A"},
{x: 300, y: 75, team: "A"},
{x: 300, y: 225, team: "A"},
{x: 300, y: 325, team: "A"},
{x: 300, y: 475, team: "A"},
{x: 400, y: 150, team: "A"},
{x: 400, y: 400, team: "A"},
{x: 450, y: 275, team: "A"},
{x: 500, y: 75, team: "A"},

{x: 500, y: 475, team: "B"},
{x: 550, y: 275, team: "B"},
{x: 600, y: 150, team: "B"},
{x: 600, y: 400, team: "B"},
{x: 700, y: 75, team: "B"},
{x: 700, y: 225, team: "B"},
{x: 700, y: 325, team: "B"},
{x: 700, y: 475, team: "B"},
{x: 820, y: 100, team: "B"},
{x: 820, y: 450, team: "B"},
{x: 865, y: 275, team: "B"},

{x: 25, y: 125, team: "A"},
{x: 25, y: 200, team: "A"},
{x: 25, y: 275, team: "A"},
{x: 25, y: 350, team: "A"},
{x: 25, y: 425, team: "A"},

{x: 975, y: 125, team: "B"},
{x: 975, y: 200, team: "B"},
{x: 975, y: 275, team: "B"},
{x: 975, y: 350, team: "B"},
{x: 975, y: 425, team: "B"},
]
```

Soccer Simulator Class Diagram



always runs behind ball but keeps certain distance

y is constant, x always the x of the ball

moves like bees, destination: place of the mouse click plus the value for the precision which is higher, the higher the distance between click and ball is. + individual precision of the player from which the ball was shot.

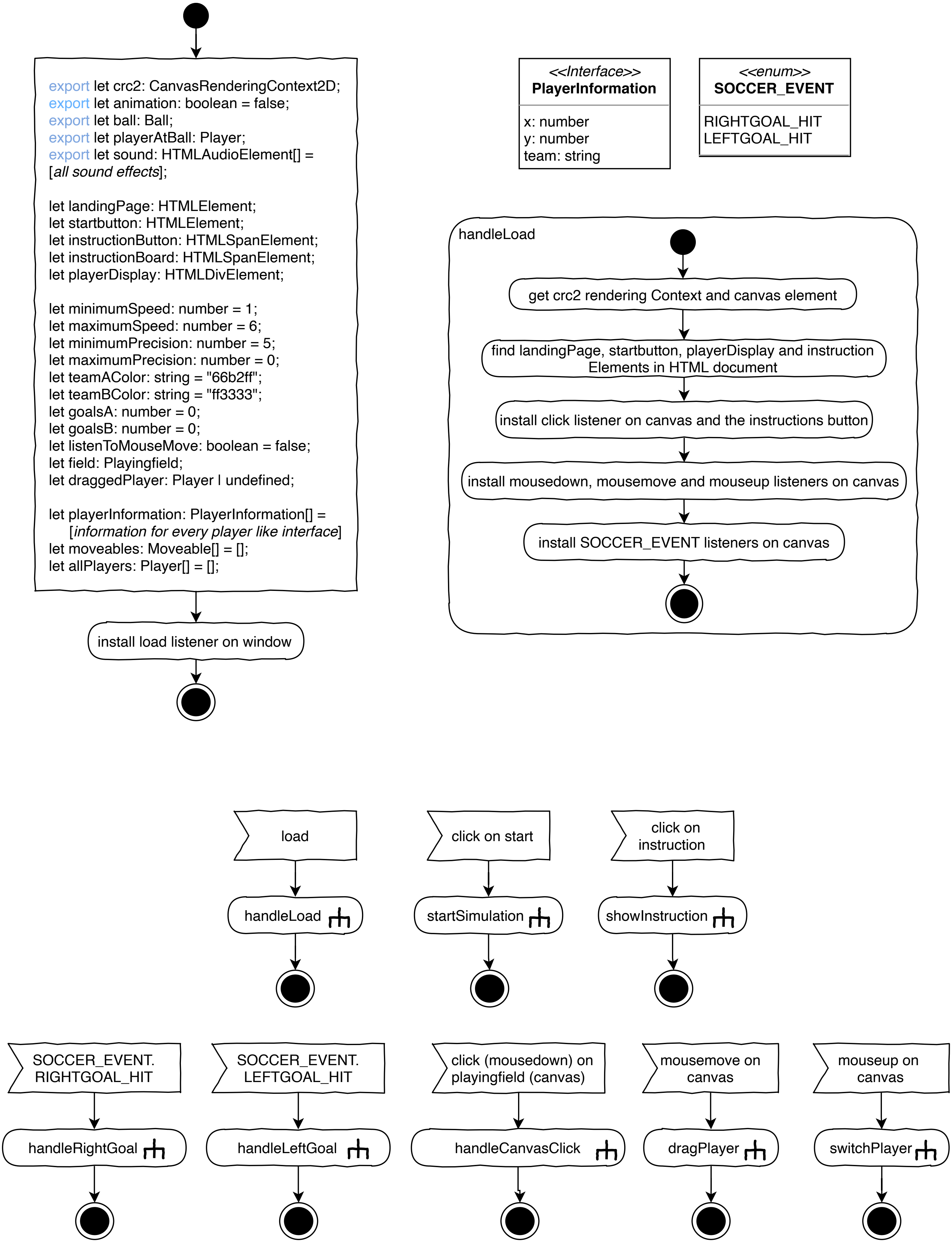
when ball leaves playing field, he reenters from the middle point.

Player at ball has to be saved somewhere to get shot-precision and to display the player in possession of the ball

move: when ball and player are close than around 30m, player moves gradually towards the ball.

When ball is further away from him ("leaves his perception radius", he moves back to his initial position.

Soccer Simulator Activity Diagram

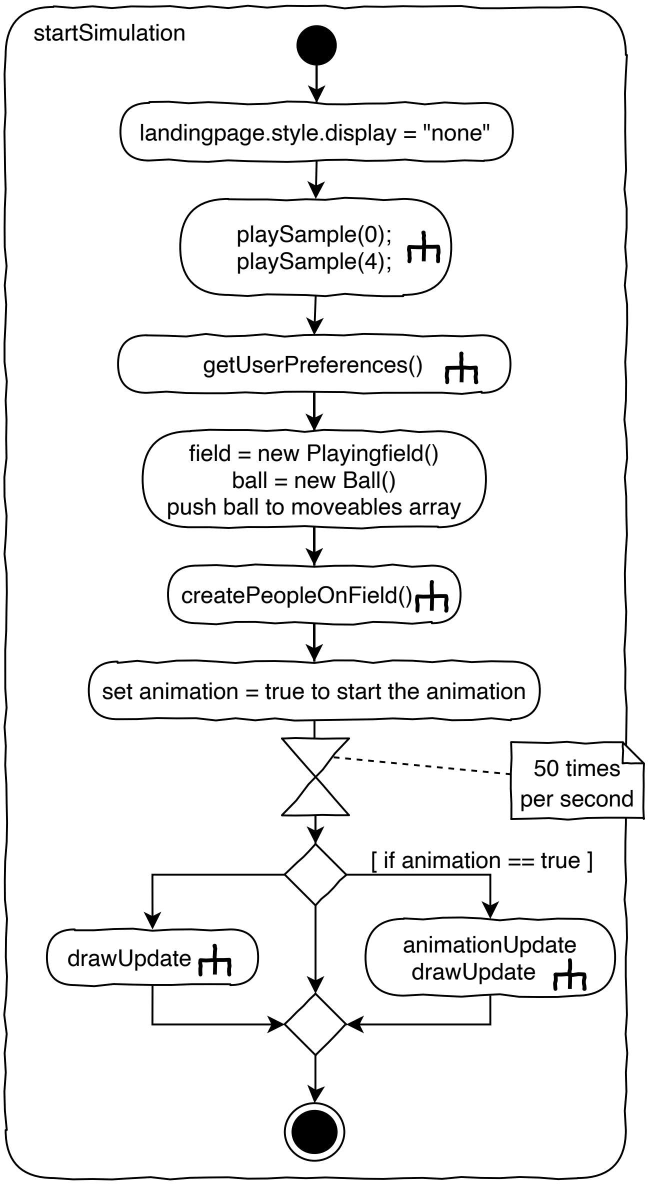


```
export let crc2: CanvasRenderingContext2D;
export let animation: boolean = false;
export let ball: Ball;
export let playerAtBall: Player;
export let sound: HTMLAudioElement[] =
[all sound effects];

let landingPage: HTMLElement;
let startbutton: HTMLElement;
let instructionButton: HTMLSpanElement;
let instructionBoard: HTMLSpanElement;
let playerDisplay: HTMLDivElement;

let minimumSpeed: number = 1;
let maximumSpeed: number = 6;
let minimumPrecision: number = 5;
let maximumPrecision: number = 0;
let teamAColor: string = "66b2ff";
let teamBColor: string = "ff3333";
let goalsA: number = 0;
let goalsB: number = 0;
let listenToMouseMove: boolean = false;
let field: Playingfield;
let draggedPlayer: Player | undefined;

let playerInformation: PlayerInformation[] =
    [information for every player like interface]
let moveables: Moveable[] = [];
let allPlayers: Player[] = [];
```

```
<<Interface>>
PlayerInformation
x: number
y: number
team: string
```

```
<<enum>>
SOCCER_EVENT
RIGHTGOAL_HIT
LEFTGOAL_HIT
```

install load listener on window
handleLoad
get crc2 rendering Context and canvas element
find landingPage, startbutton, playerDisplay and instruction Elements in HTML document
install click listener on canvas and the instructions button
install mousedown, mousemove and mouseup listeners on canvas
install SOCCER_EVENT listeners on canvas
load
handleLoad
click on start
startSimulation
click on instruction
showInstruction
SOCCER_EVENT. RIGHTGOAL_HIT
handleRightGoal
SOCCER_EVENT. LEFTGOAL_HIT
handleLeftGoal
click (mousedown) on playingfield (canvas)
handleCanvasClick
mousemove on canvas
dragPlayer
mouseup on canvas
switchPlayer



startSimulation

landingpage.style.display = "none"

playSample(0);
playSample(4);

getUserPreferences()

field = new Playingfield()
ball = new Ball()
push ball to moveables array

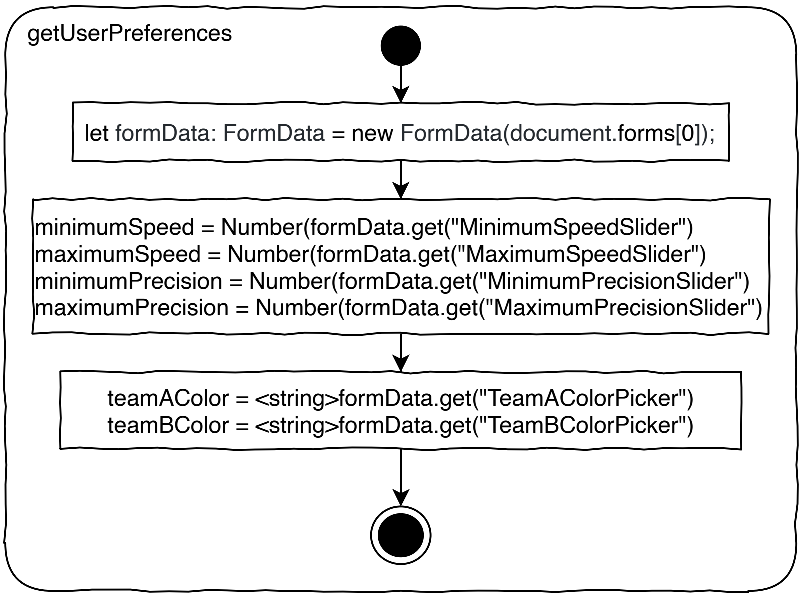
createPeopleOnField()

set animation = true to start the animation

50 times per second

drawUpdate

animationUpdate
drawUpdate

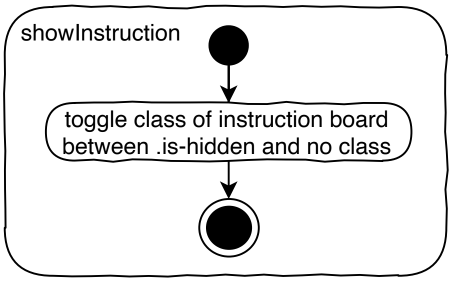


getUserPreferences

```
let formData: FormData = new FormData(document.forms[0]);
```

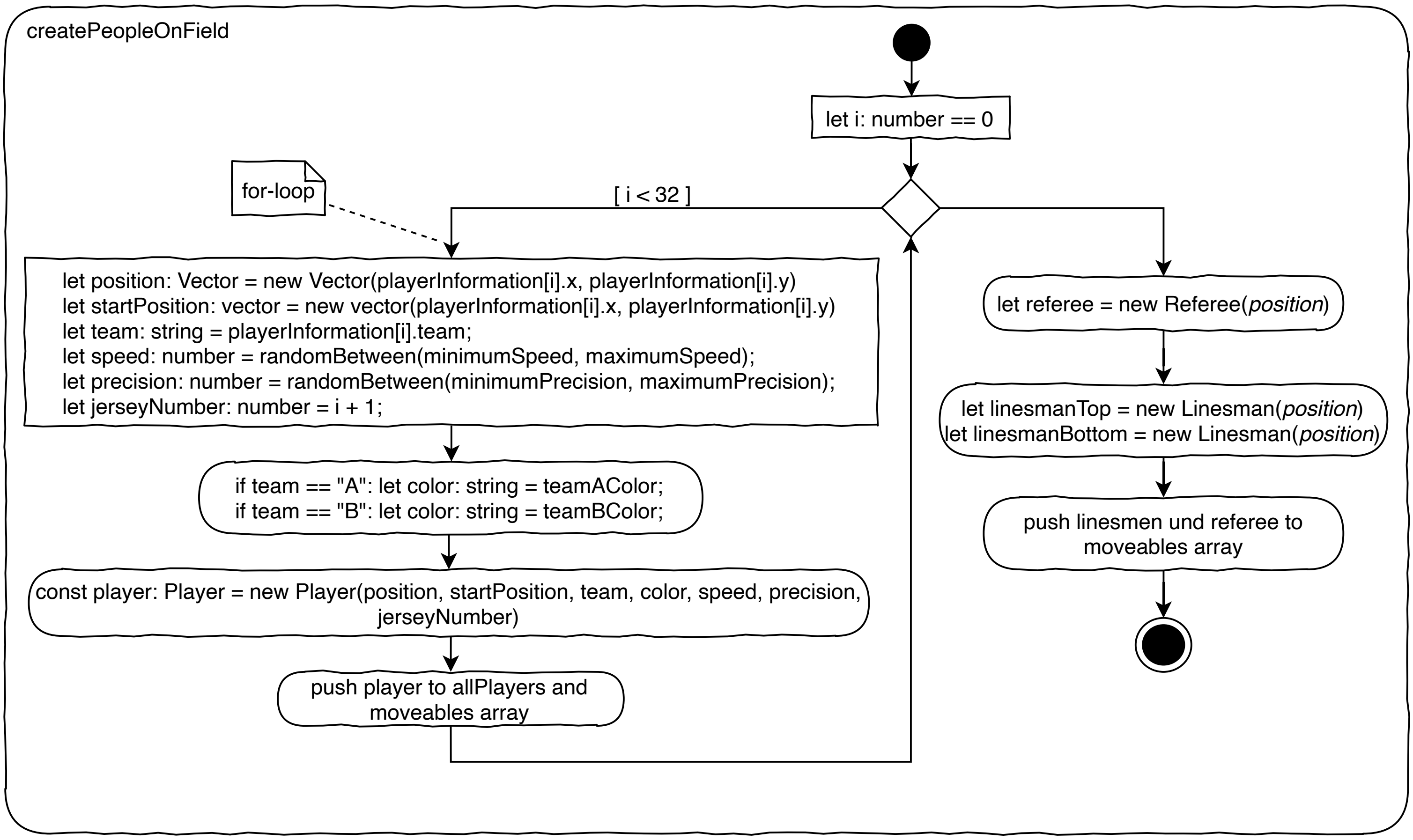
```
minimumSpeed = Number(formData.get("MinimumSpeedSlider")
maximumSpeed = Number(formData.get("MaximumSpeedSlider")
minimumPrecision = Number(formData.get("MinimumPrecisionSlider")
maximumPrecision = Number(formData.get("MaximumPrecisionSlider")
```

```
teamAColor = <string>formData.get("TeamAColorPicker")
teamBColor = <string>formData.get("TeamBColorPicker")
```



showInstruction

toggle class of instruction board between .is-hidden and no class



createPeopleOnField

```
let i: number == 0
```

```
let position: Vector = new Vector(playerInformation[i].x, playerInformation[i].y)
let startPosition: vector = new vector(playerInformation[i].x, playerInformation[i].y)
let team: string = playerInformation[i].team;
let speed: number = randomBetween(minimumSpeed, maximumSpeed);
let precision: number = randomBetween(minimumPrecision, maximumPrecision);
let jerseyNumber: number = i + 1;
```

```
if team == "A": let color: string = teamAColor;
if team == "B": let color: string = teamBColor;
```

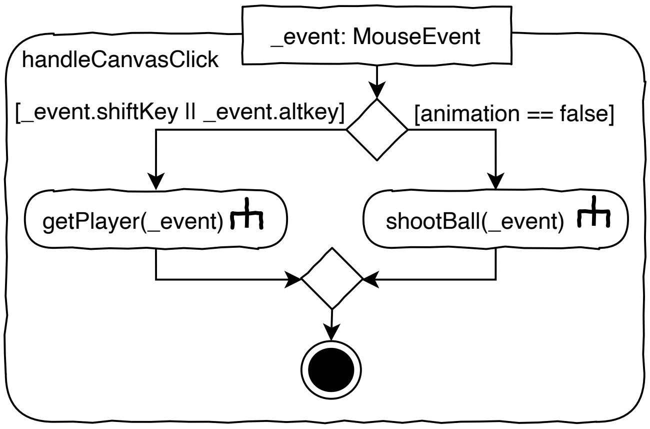
```
const player: Player = new Player(position, startPosition, team, color, speed, precision, jerseyNumber)
```

push player to allPlayers and moveables array

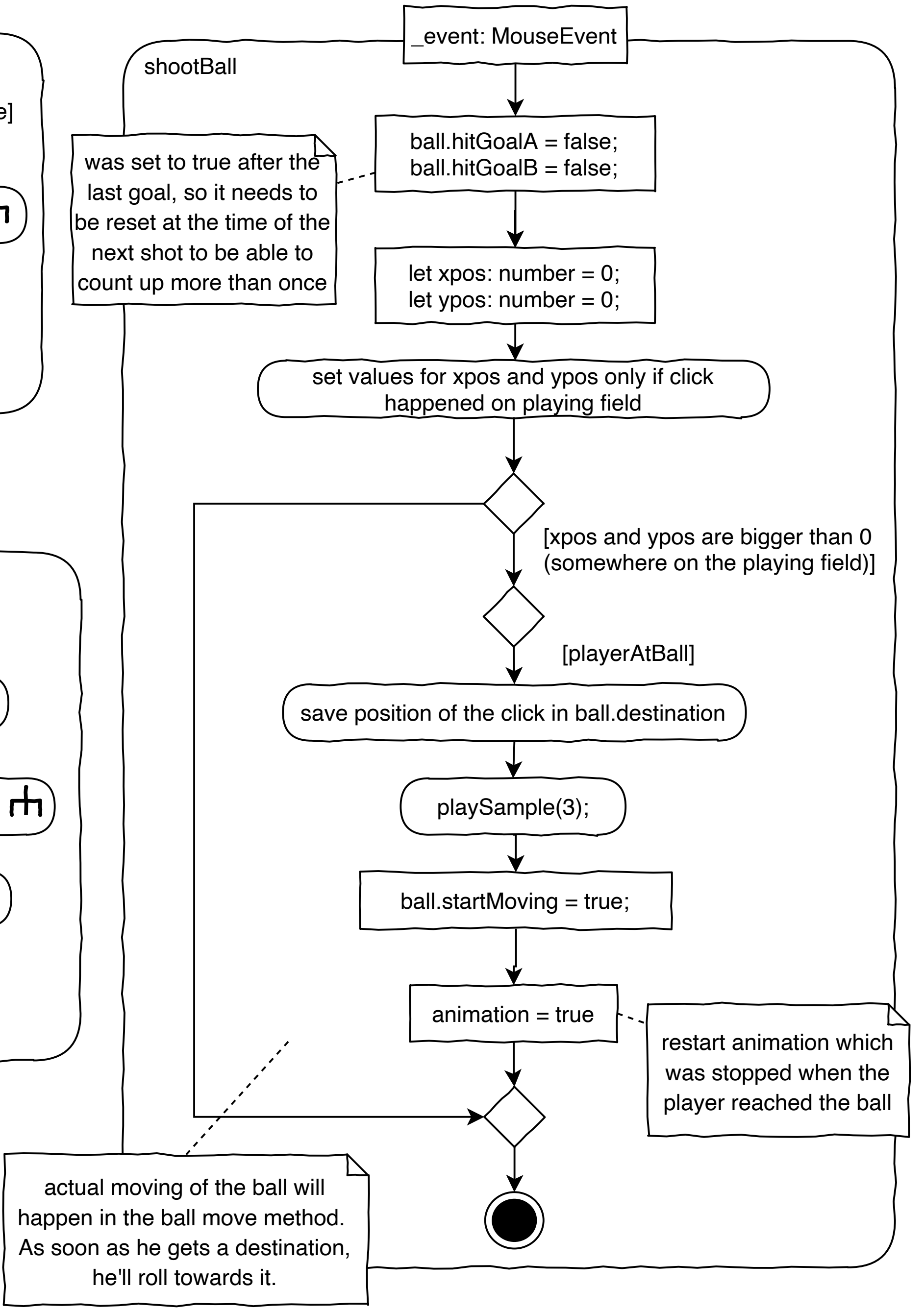
let referee = new Referee(*position*)

let linesmanTop = new Linesman(*position*)
let linesmanBottom = new Linesman(*position*)

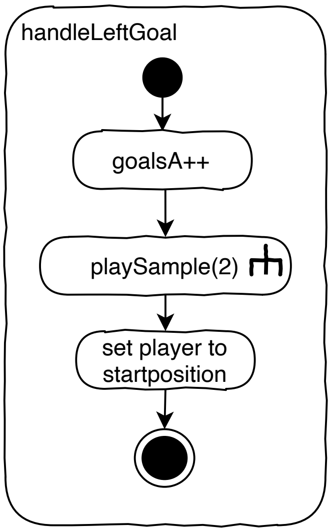
push linesmen und referee to moveables array



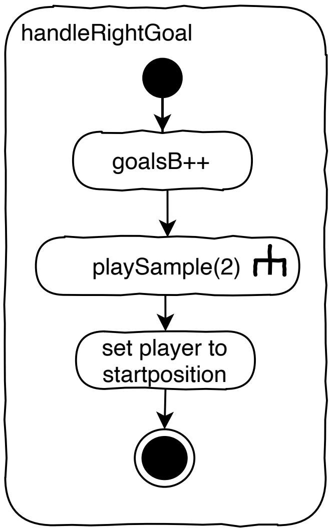
_event: MouseEvent
handleCanvasClick
[_event.shiftKey || _event.altkey]
[animation == false]
getPlayer(_event)
shootBall(_event)



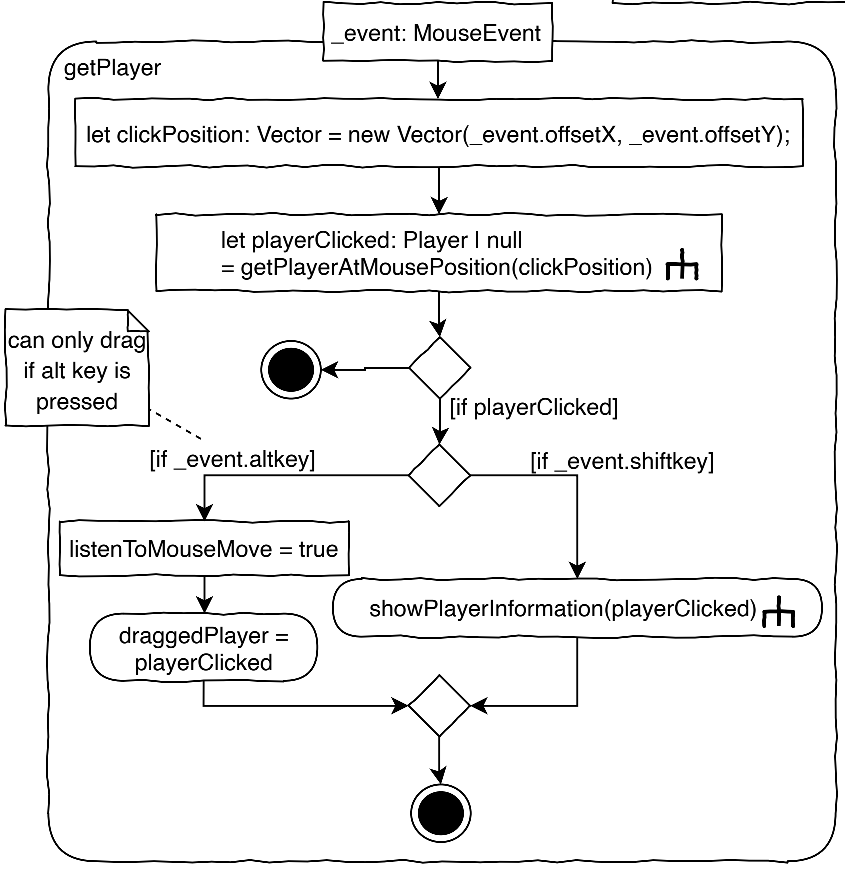
_event: MouseEvent
shootBall
ball.hitGoalA = false;
ball.hitGoalB = false;
was set to true after the last goal, so it needs to be reset at the time of the next shot to be able to count up more than once
let xpos: number = 0;
let ypos: number = 0;
set values for xpos and ypos only if click happened on playing field
[xpos and ypos are bigger than 0 (somewhere on the playing field)]
[playerAtBall]
save position of the click in ball.destination
playSample(3);
ball.startMoving = true;
animation = true
restart animation which was stopped when the player reached the ball
actual moving of the ball will happen in the ball move method. As soon as he gets a destination, he'll roll towards it.



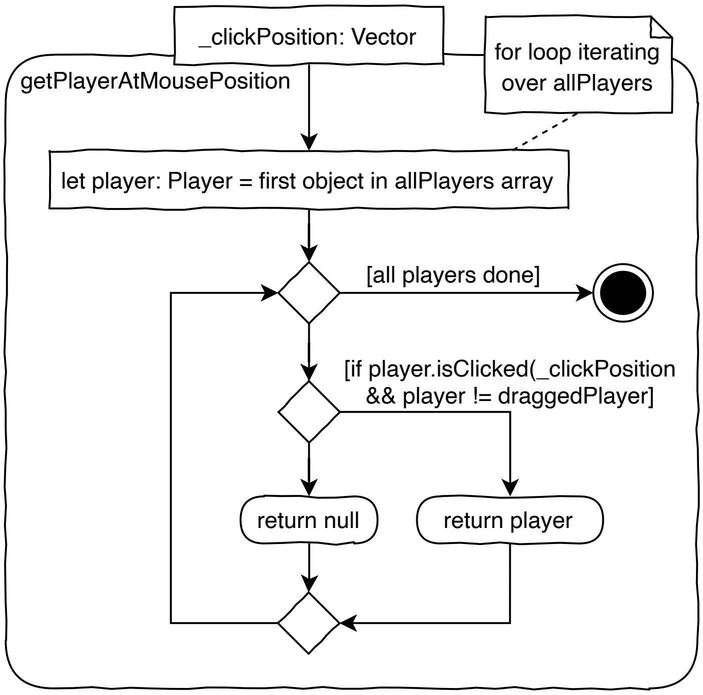
handleLeftGoal
goalsA++
playSample(2)
set player to startposition



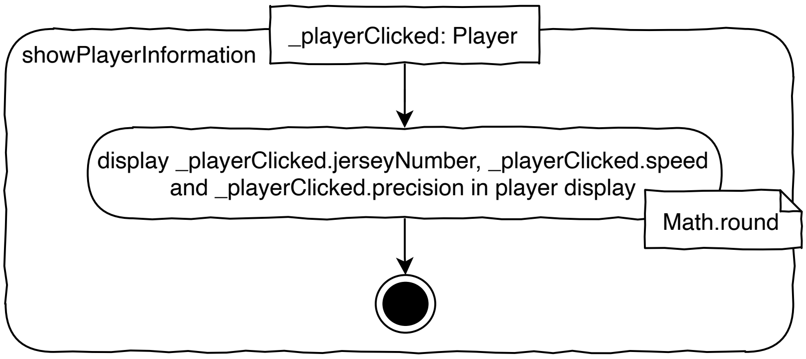
handleRightGoal
goalsB++
playSample(2)
set player to startposition

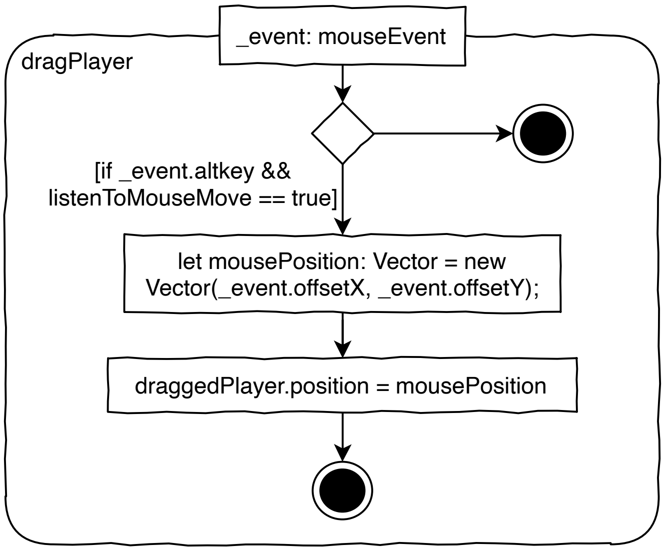


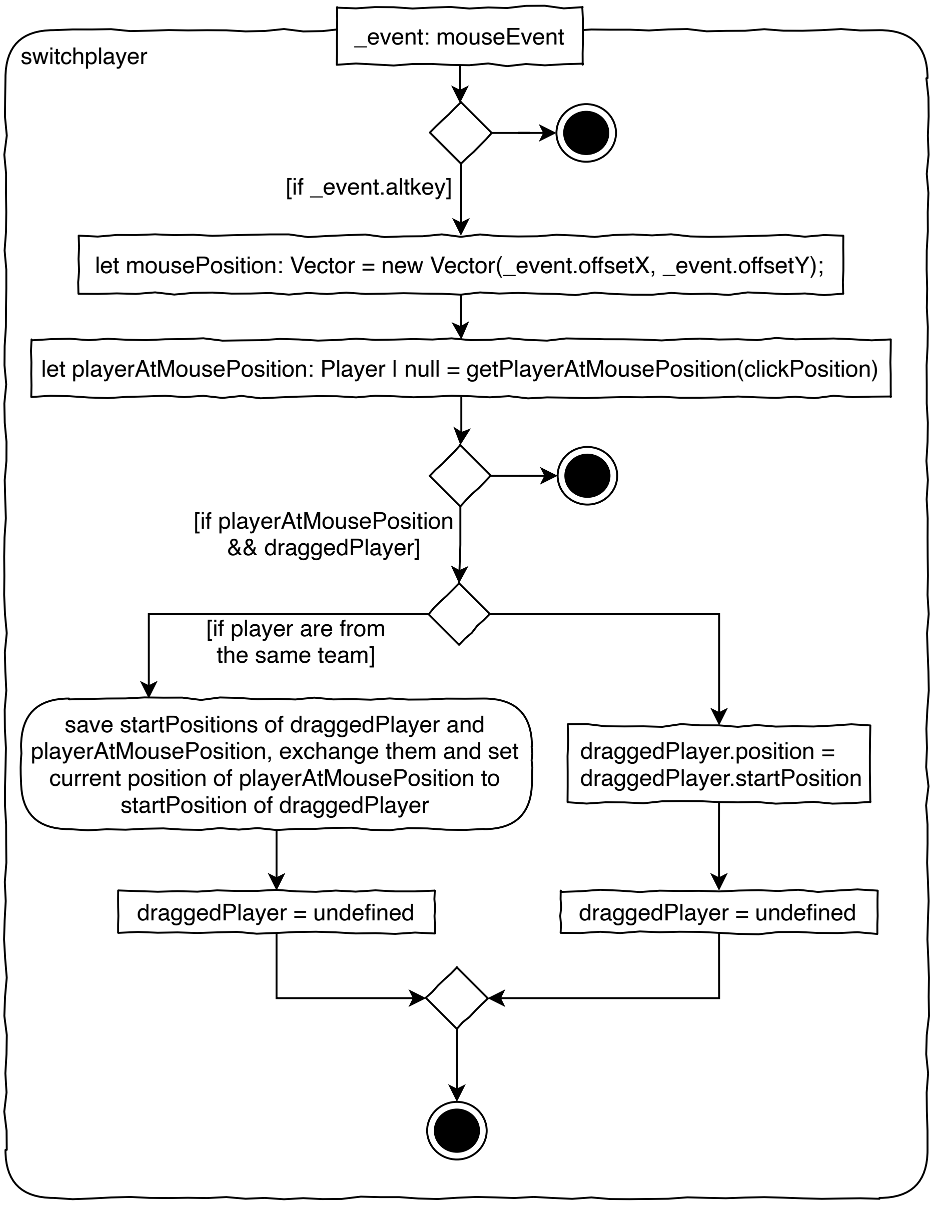
_event: MouseEvent
getPlayer
let clickPosition: Vector = new Vector(_event.offsetX, _event.offsetY);
let playerClicked: Player | null
= getPlayerAtMousePosition(clickPosition)
[if playerClicked]
can only drag if alt key is pressed
[if _event.altkey]
[if _event.shiftkey]
listenToMouseMove = true
draggedPlayer = playerClicked
showPlayerInformation(playerClicked)

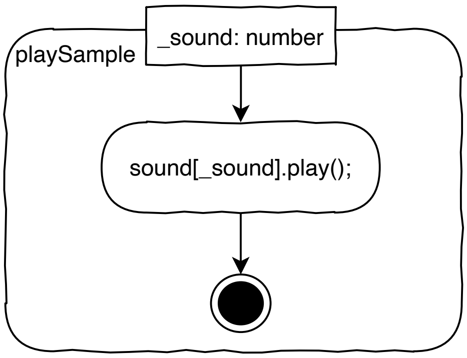


_clickPosition: Vector
getPlayerAtMousePosition
for loop iterating over allPlayers
let player: Player = first object in allPlayers array
[all players done]
[if player.isClicked(_clickPosition && player != draggedPlayer]
return null
return player


showPlayerInformation
_playerClicked: Player
display _playerClicked.jerseyNumber, _playerClicked.speed and _playerClicked.precision in player display
Math.round

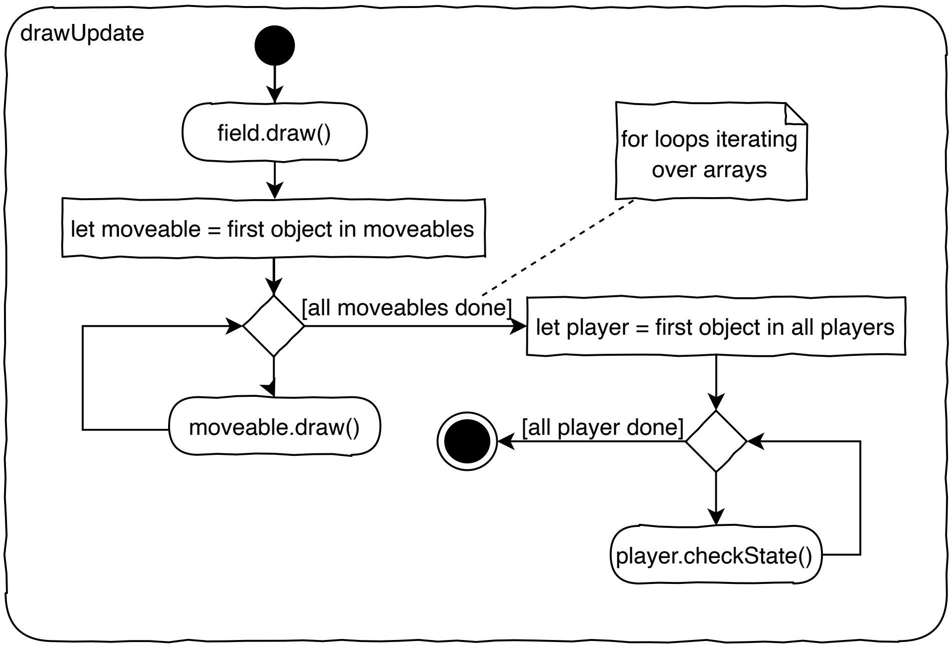

dragPlayer
_event: mouseEvent
[if _event.altkey && listenToMouseMove == true]
let mousePosition: Vector = new Vector(_event.offsetX, _event.offsetY);
draggedPlayer.position = mousePosition

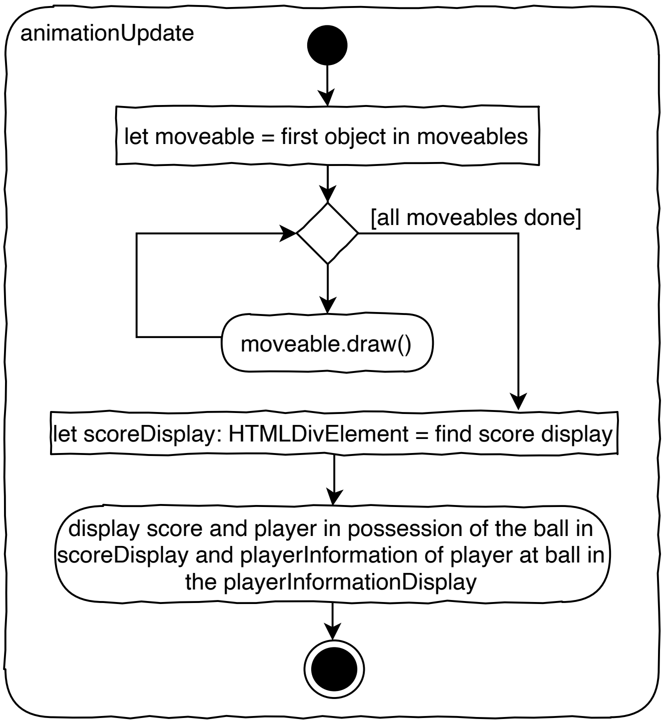

switchplayer
_event: mouseEvent
[if _event.altkey]
let mousePosition: Vector = new Vector(_event.offsetX, _event.offsetY);
let playerAtMousePosition: Player | null = getPlayerAtMousePosition(clickPosition)
[if playerAtMousePosition && draggedPlayer]
[if player are from the same team]
save startPositions of draggedPlayer and playerAtMousePosition, exchange them and set current position of playerAtMousePosition to startPosition of draggedPlayer
draggedPlayer.position = draggedPlayer.startPosition
draggedPlayer = undefined
draggedPlayer = undefined



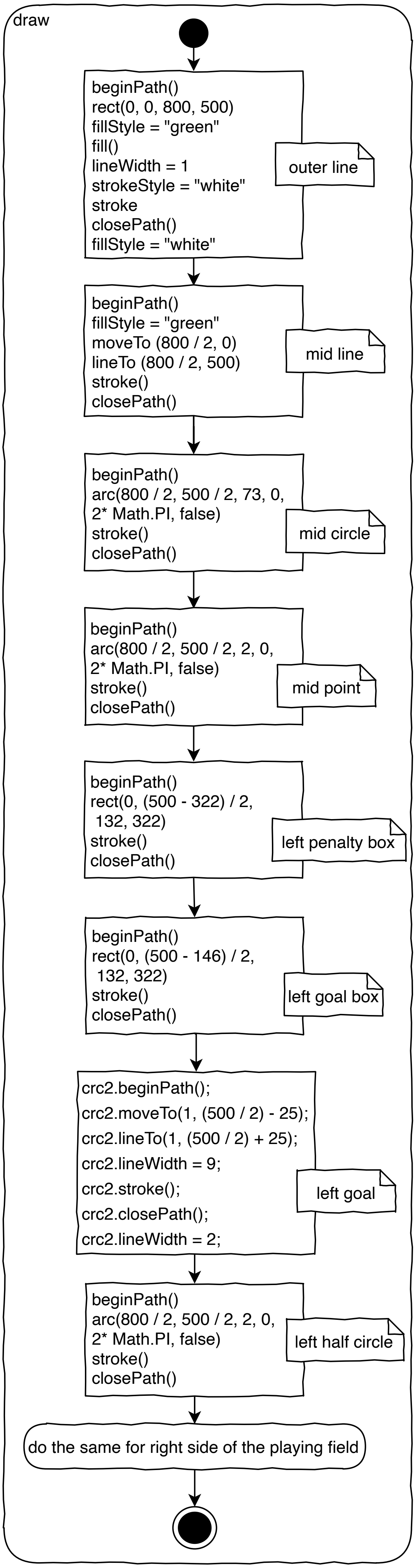
when draggedPlayer is overlapping with a field player at releasing the mouse, they switch their positions.

If there's no player underneath the dragged player, the dragged player jumps back to its start position


drawUpdate
field.draw()
for loops iterating over arrays
let moveable = first object in moveables
[all moveables done]
let player = first object in all players
moveable.draw()
[all player done]
player.checkState()


animationUpdate
let moveable = first object in moveables
[all moveables done]
moveable.draw()
let scoreDisplay: HTMLDivElement = find score display
display score and player in possession of the ball in scoreDisplay and playerInformation of player at ball in the playerInformationDisplay

Playing Field Methods



```
beginPath()
rect(0, 0, 800, 500)
fillStyle = "green"
fill()
lineWidth = 1
strokeStyle = "white"
stroke
closePath()
fillStyle = "white"
```

```
beginPath()
fillStyle = "green"
moveTo (800 / 2, 0)
lineTo (800 / 2, 500)
stroke()
closePath()
```

```
beginPath()
arc(800 / 2, 500 / 2, 73, 0,
2* Math.PI, false)
stroke()
closePath()
```

```
beginPath()
arc(800 / 2, 500 / 2, 2, 0,
2* Math.PI, false)
stroke()
closePath()
```

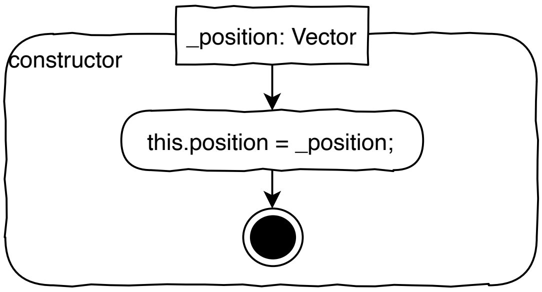
```
beginPath()
rect(0, (500 - 322) / 2,
 132, 322)
stroke()
closePath()
```

```
beginPath()
rect(0, (500 - 146) / 2,
 132, 322)
stroke()
closePath()
```

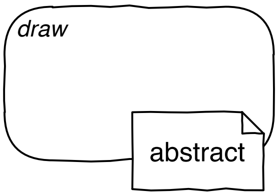
```
crc2.beginPath();
crc2.moveTo(1, (500 / 2) - 25);
crc2.lineTo(1, (500 / 2) + 25);
crc2.lineWidth = 9;
crc2.stroke();
crc2.closePath();
crc2.lineWidth = 2;
```

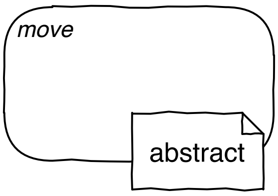
```
beginPath()
arc(800 / 2, 500 / 2, 2, 0,
2* Math.PI, false)
stroke()
closePath()
```

Moveable Methods

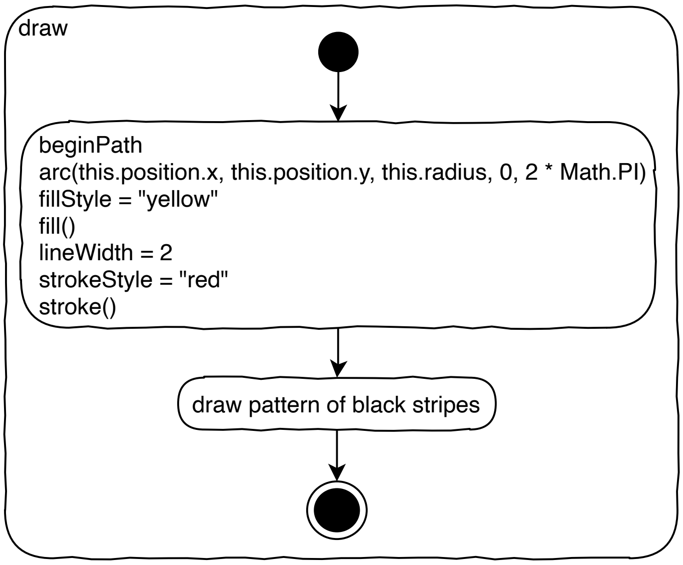


```
this.position = _position;
```



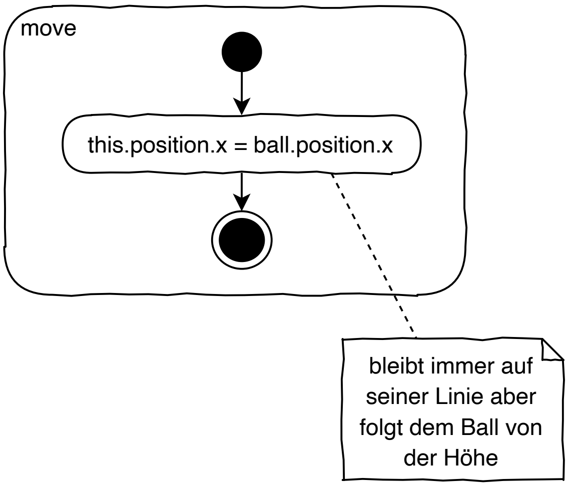


Linesman Methods



```
beginPath
arc(this.position.x, this.position.y, this.radius, 0, 2 * Math.PI)
fillStyle = "yellow"
fill()
lineWidth = 2
strokeStyle = "red"
stroke()
```

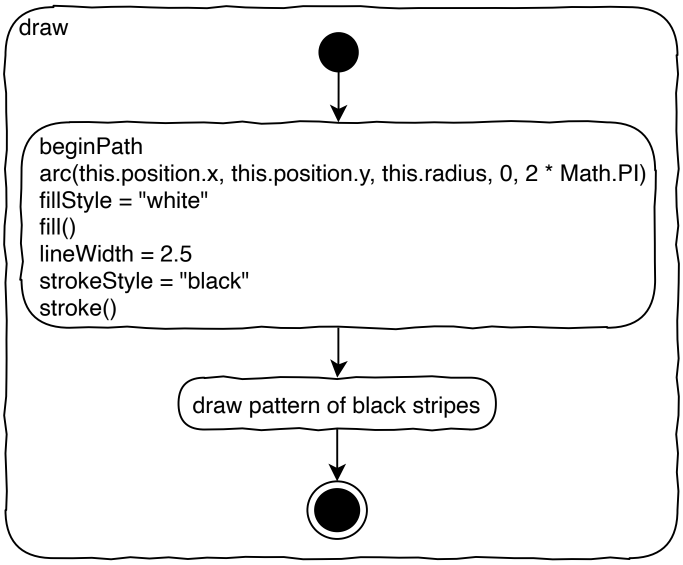
draw pattern of black stripes



```
this.position.x = ball.position.x
```

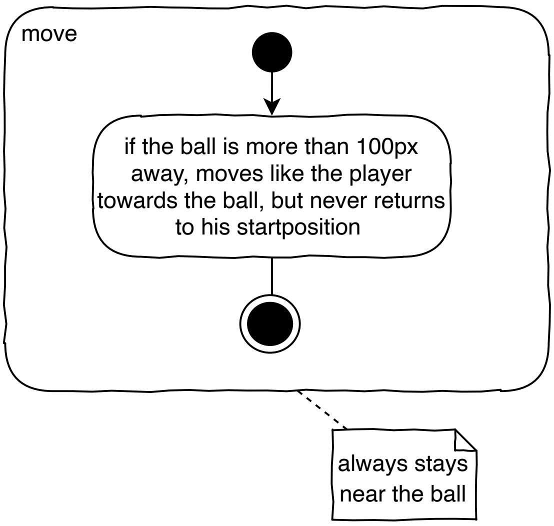
bleibt immer auf seiner Linie aber folgt dem Ball von der Höhe

Referee Methods



```
beginPath
arc(this.position.x, this.position.y, this.radius, 0, 2 * Math.PI)
fillStyle = "white"
fill()
lineWidth = 2.5
strokeStyle = "black"
stroke()
```

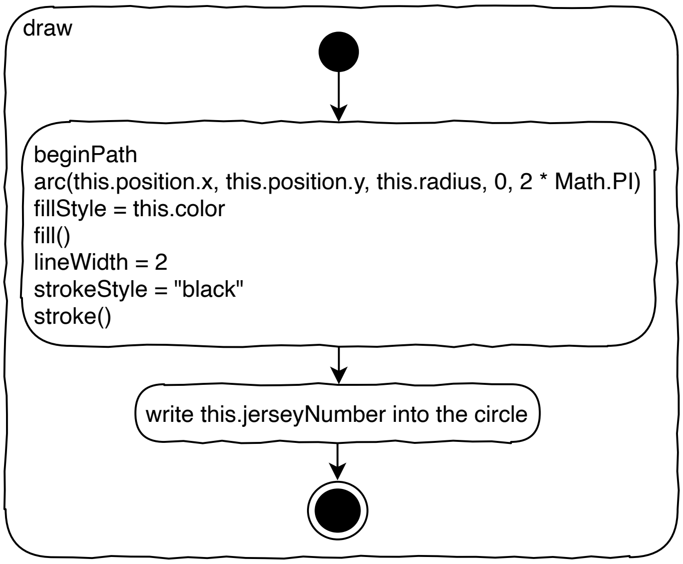
draw pattern of black stripes



if the ball is more than 100px away, moves like the player towards the ball, but never returns to his startposition

always stays near the ball

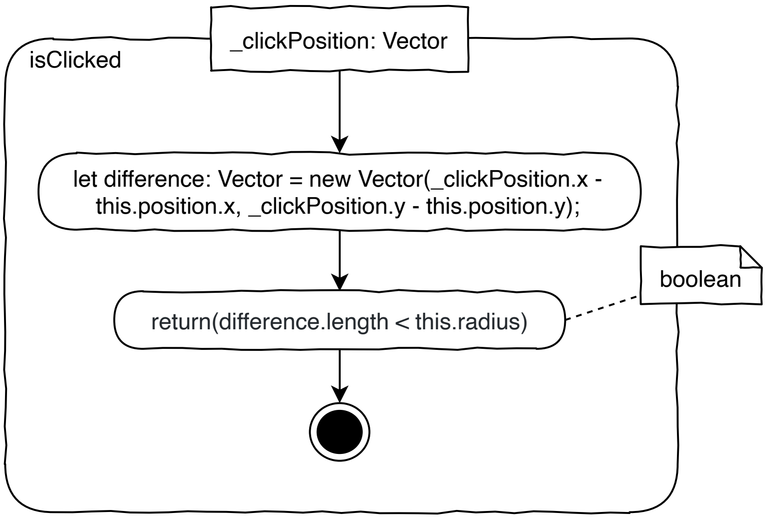
Player Methods



draw

beginPath
arc(this.position.x, this.position.y, this.radius, 0, 2 * Math.PI)
fillStyle = this.color
fill()
lineWidth = 2
strokeStyle = "black"
stroke()

write this.jerseyNumber into the circle



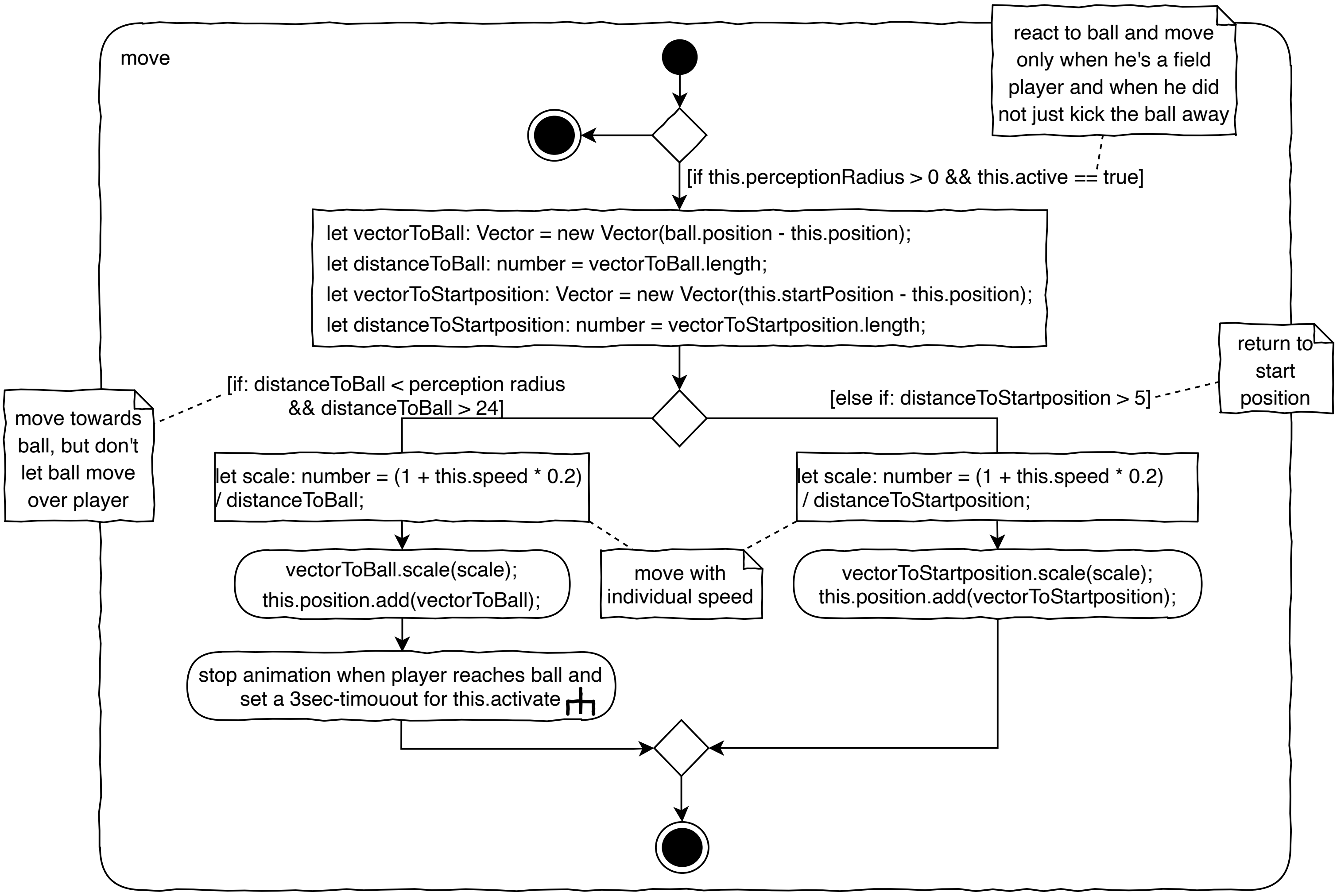
isClicked

_clickPosition: Vector

let difference: Vector = new Vector(_clickPosition.x - this.position.x, _clickPosition.y - this.position.y);

return(difference.length < this.radius)

boolean



move

react to ball and move only when he's a field player and when he did not just kick the ball away

[if this.perceptionRadius > 0 && this.active == true]

let vectorToBall: Vector = new Vector(ball.position - this.position);
let distanceToBall: number = vectorToBall.length;
let vectorToStartposition: Vector = new Vector(this.startPosition - this.position);
let distanceToStartposition: number = vectorToStartposition.length;

move towards ball, but don't let ball move over player

[if: distanceToBall < perception radius && distanceToBall > 24]

[else if: distanceToStartposition > 5]

return to start position

let scale: number = (1 + this.speed * 0.2) / distanceToBall;

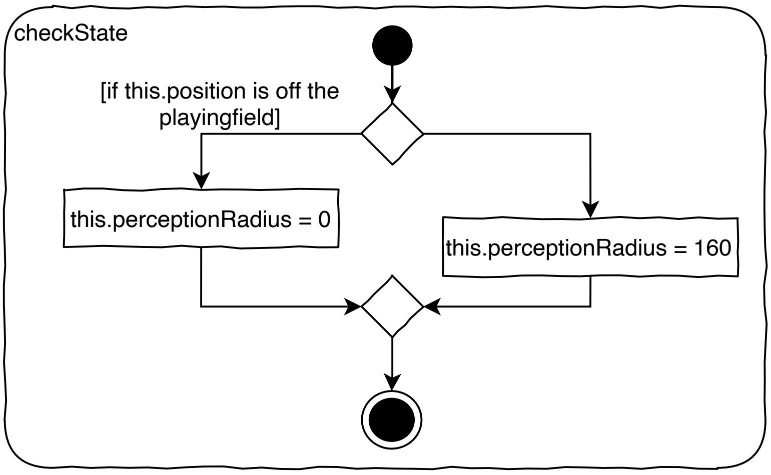
let scale: number = (1 + this.speed * 0.2) / distanceToStartposition;

move with individual speed

vectorToBall.scale(scale);
this.position.add(vectorToBall);

vectorToStartposition.scale(scale);
this.position.add(vectorToStartposition);

stop animation when player reaches ball and set a 3sec-timouout for this.activate

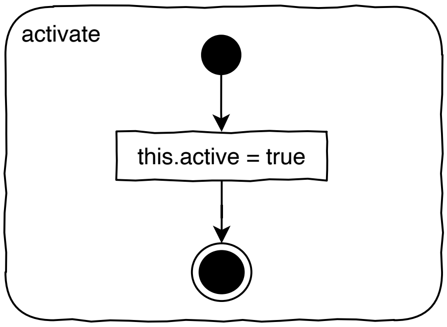


checkState

[if this.position is off the playingfield]

this.perceptionRadius = 0

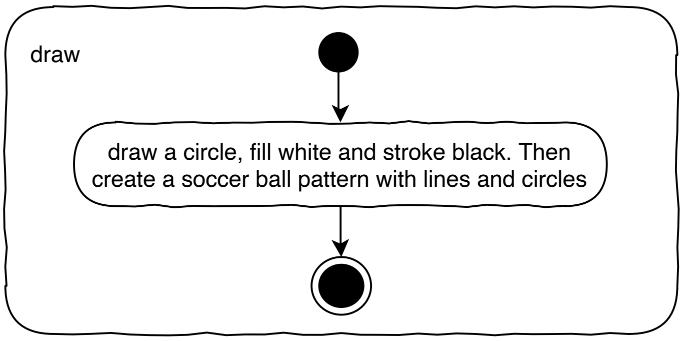
this.perceptionRadius = 160

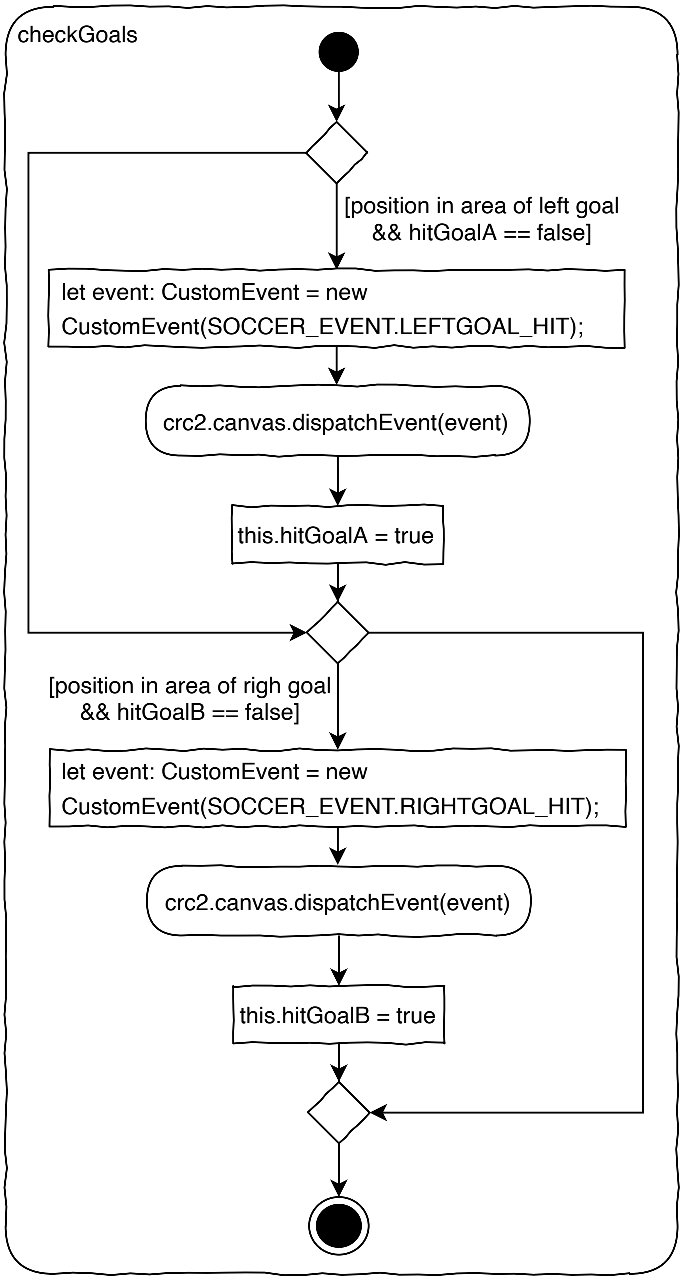


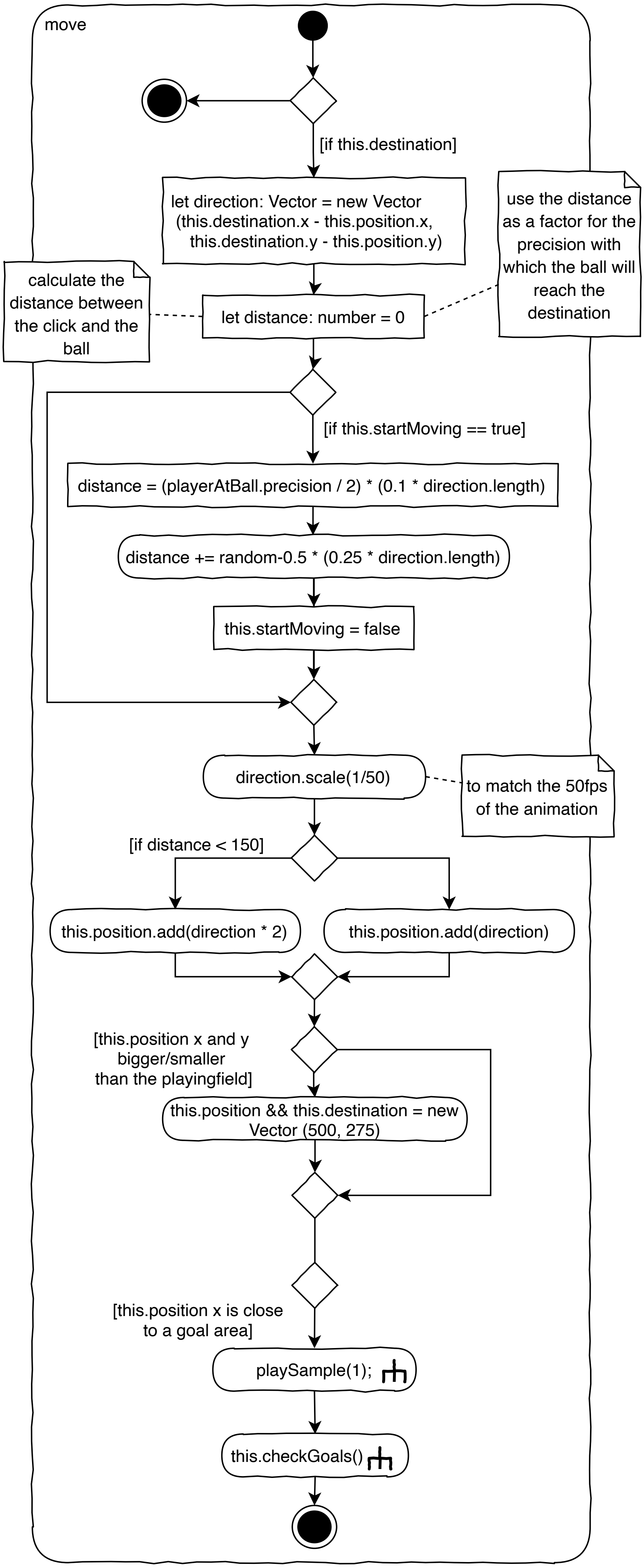
activate

this.active = true

Ball Methods


draw
draw a circle, fill white and stroke black. Then create a soccer ball pattern with lines and circles


checkGoals
[position in area of left goal && hitGoalA == false]
let event: CustomEvent = new CustomEvent(SOCCER_EVENT.LEFTGOAL_HIT);
crc2.canvas.dispatchEvent(event)
this.hitGoalA = true
[position in area of righ goal && hitGoalB == false]
let event: CustomEvent = new CustomEvent(SOCCER_EVENT.RIGHTGOAL_HIT);
crc2.canvas.dispatchEvent(event)
this.hitGoalB = true


move
[if this.destination]
let direction: Vector = new Vector (this.destination.x - this.position.x, this.destination.y - this.position.y)
use the distance as a factor for the precision with which the ball will reach the destination
calculate the distance between the click and the ball
let distance: number = 0
[if this.startMoving == true]
distance = (playerAtBall.precision / 2) * (0.1 * direction.length)
distance += random-0.5 * (0.25 * direction.length)
this.startMoving = false
direction.scale(1/50)
to match the 50fps of the animation
[if distance < 150]
this.position.add(direction * 2)
this.position.add(direction)
[this.position x and y bigger/smaller than the playingfield]
this.position && this.destination = new Vector (500, 275)
[this.position x is close to a goal area]
playSample(1);
this.checkGoals()