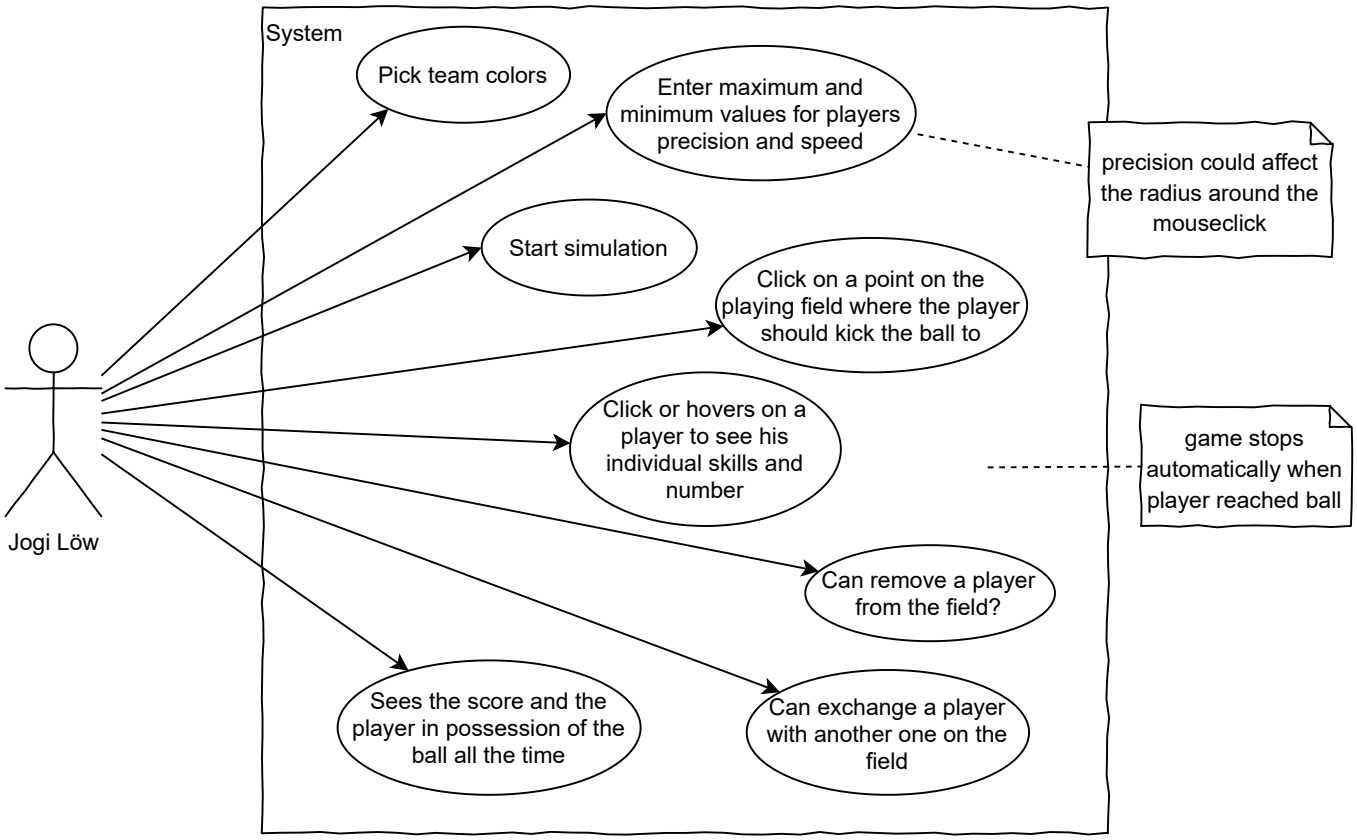
Soccer Training Simulation - Complete Documentation

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.

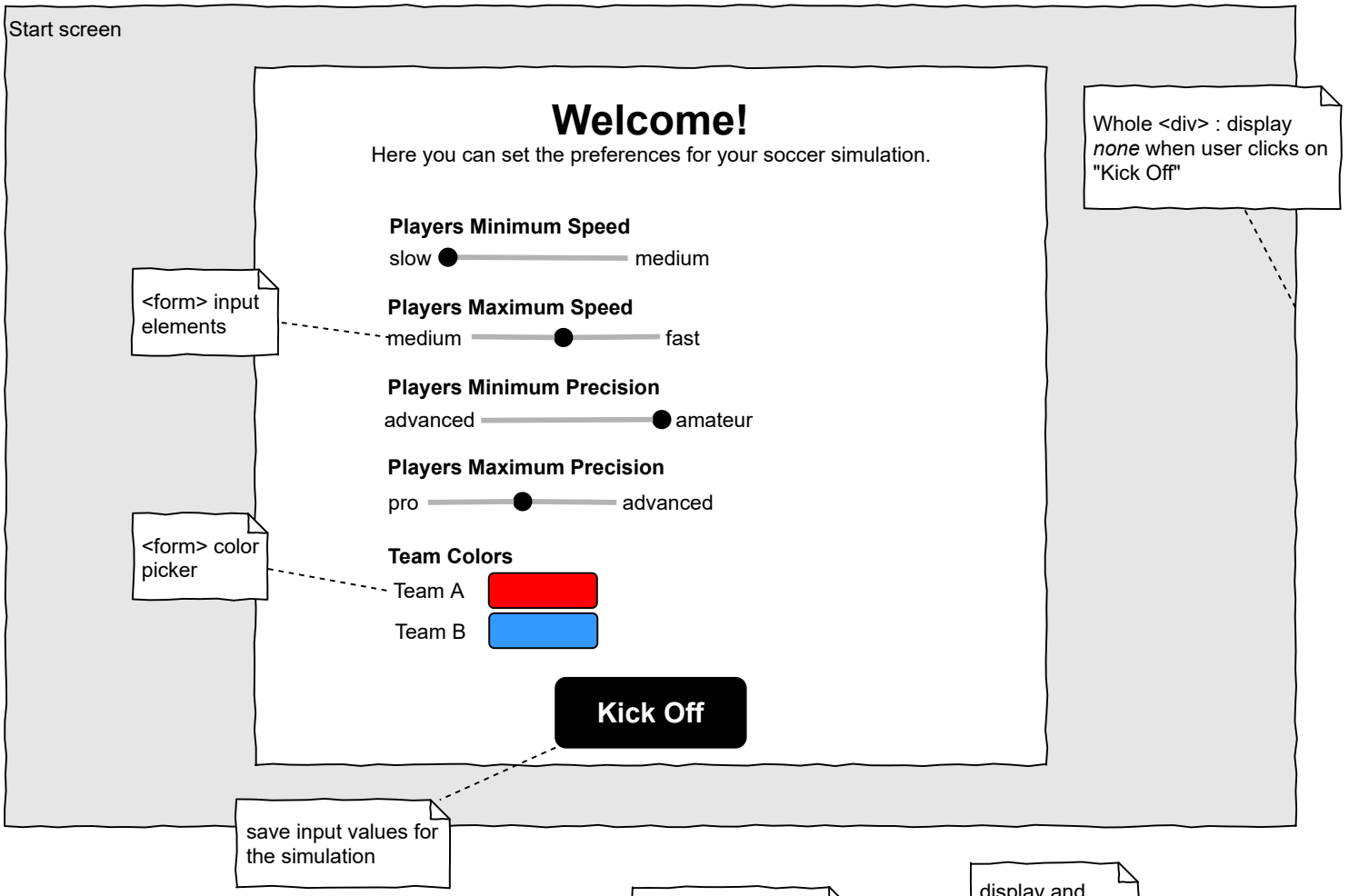
Date of completion: 15/07/2021

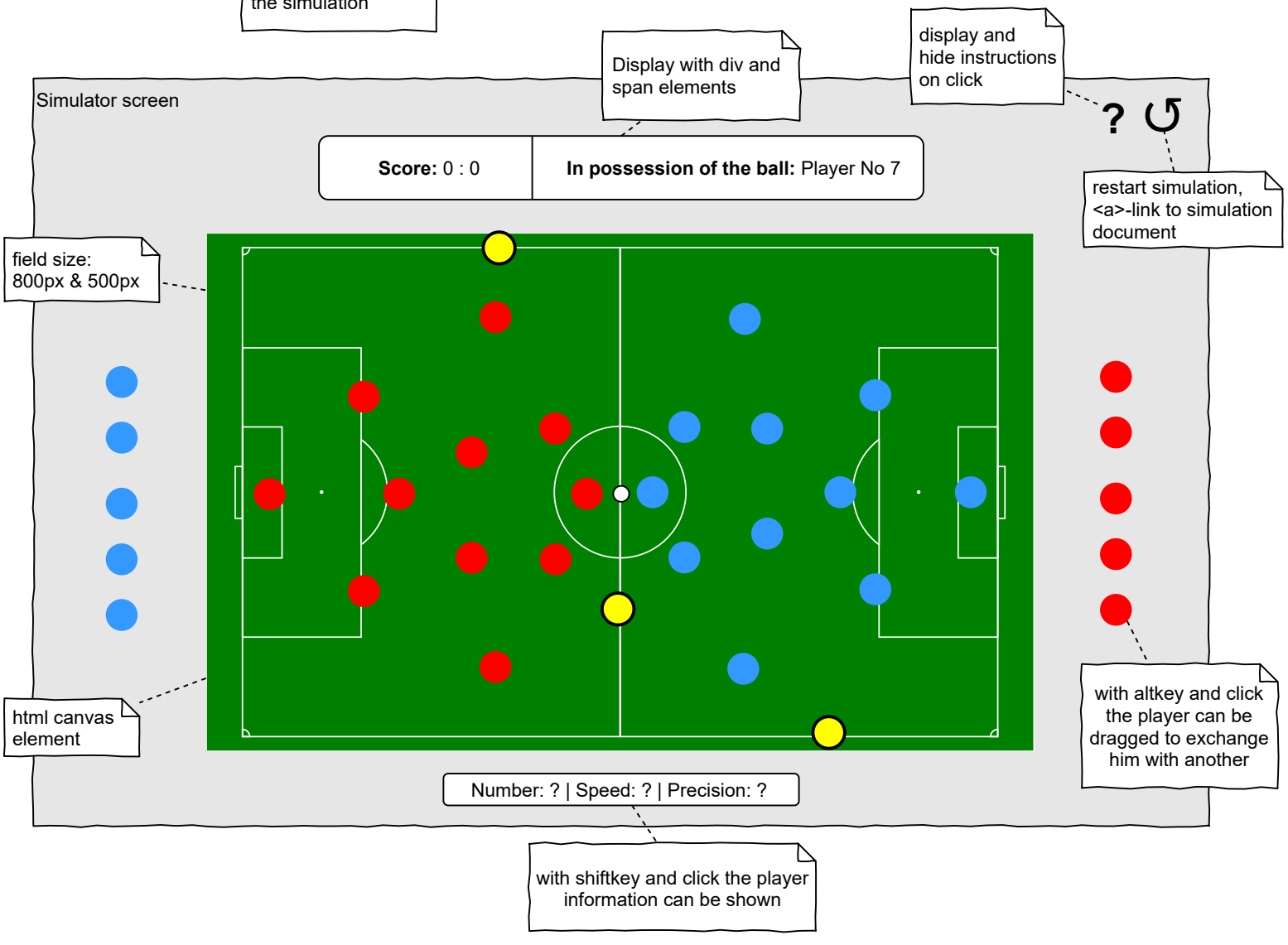
Content

Soccer Simulator Use Case Diagram

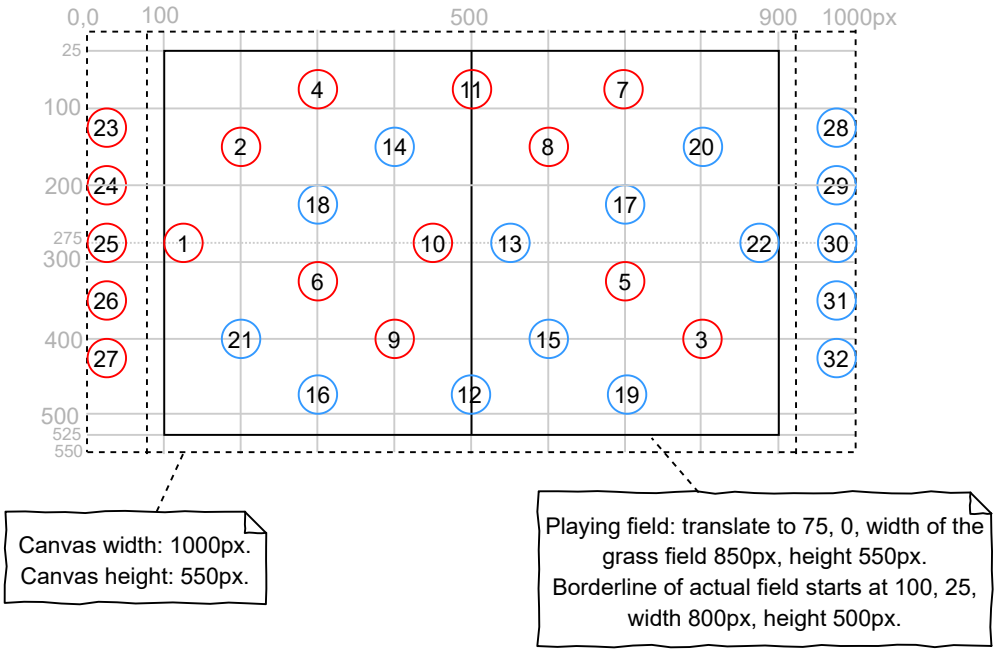


Soccer Simulator User Interface Scribble



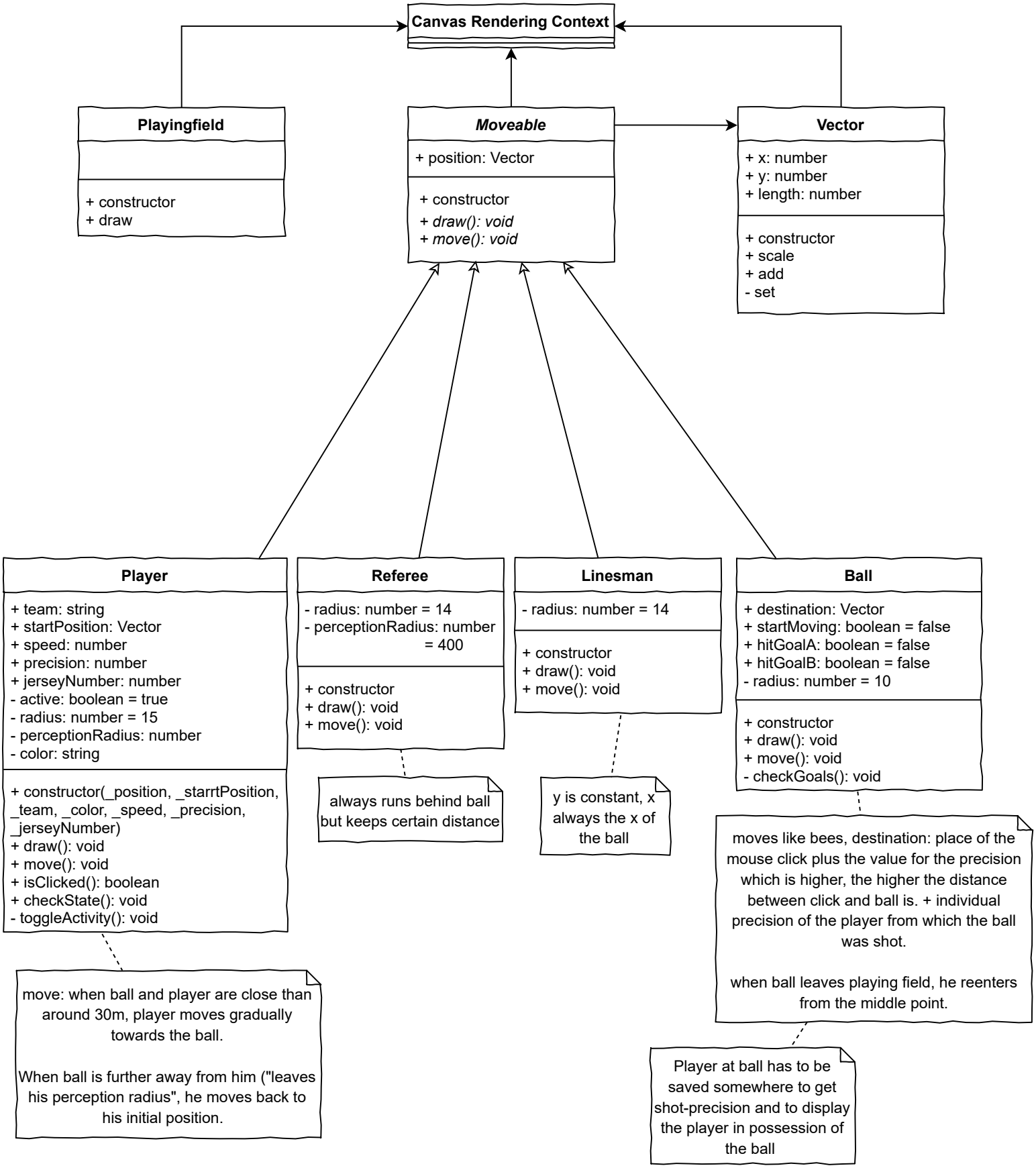


Scribble for exact canvas values

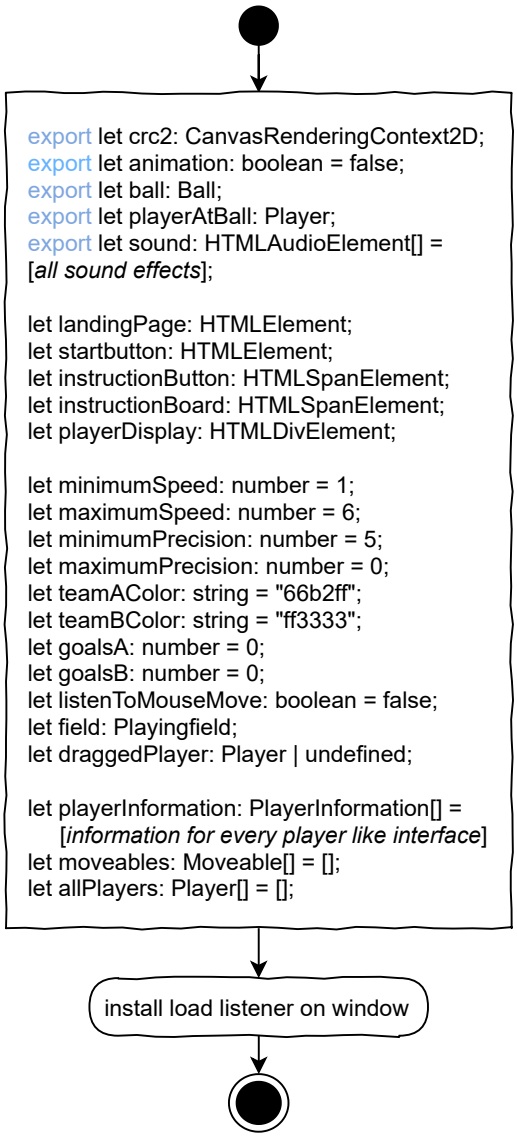


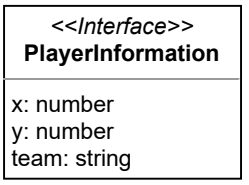

```
playerInformation [
{x: 135, y: 275, team: "A"},
{x: 180, y: 100, team: "A"},
{x: 180, y: 450, team: "A"},
{x: 300, y: 75, team: "A"},
{x: 300, y: 225, team: "A"},
{x: 300, y: 325, team: "A"},
{x: 300, y: 475, team: "A"},
{x: 400, y: 150, team: "A"},
{x: 400, y: 400, team: "A"},
{x: 450, y: 275,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

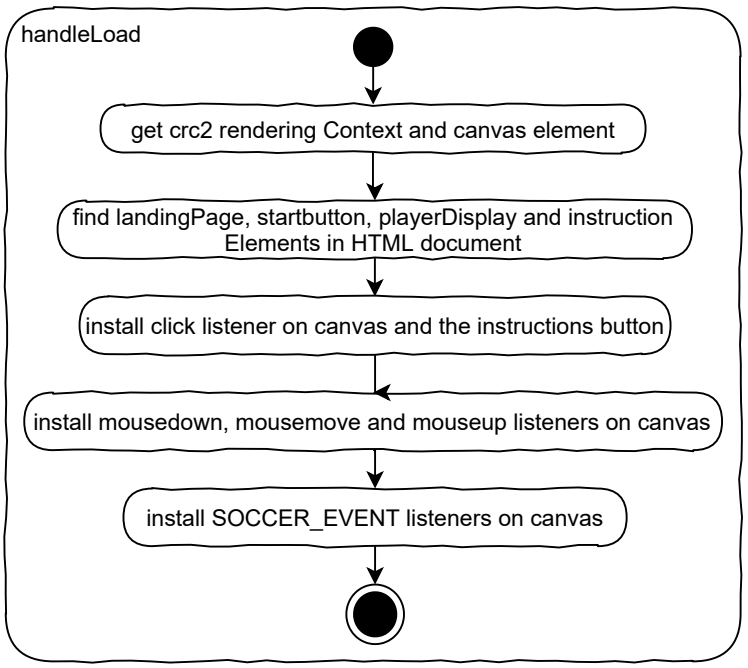
Soccer Simulator Class Diagram

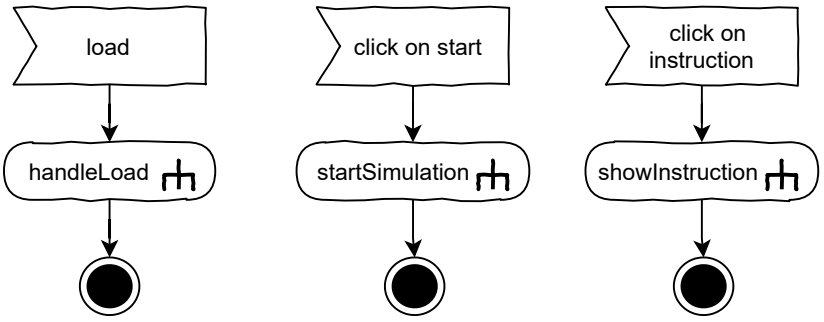


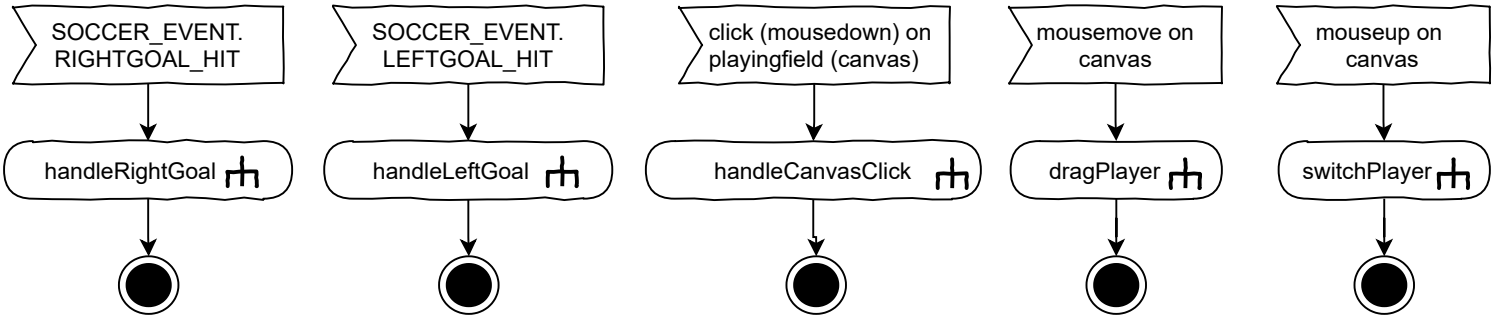
Soccer Simulator Activity Diagram

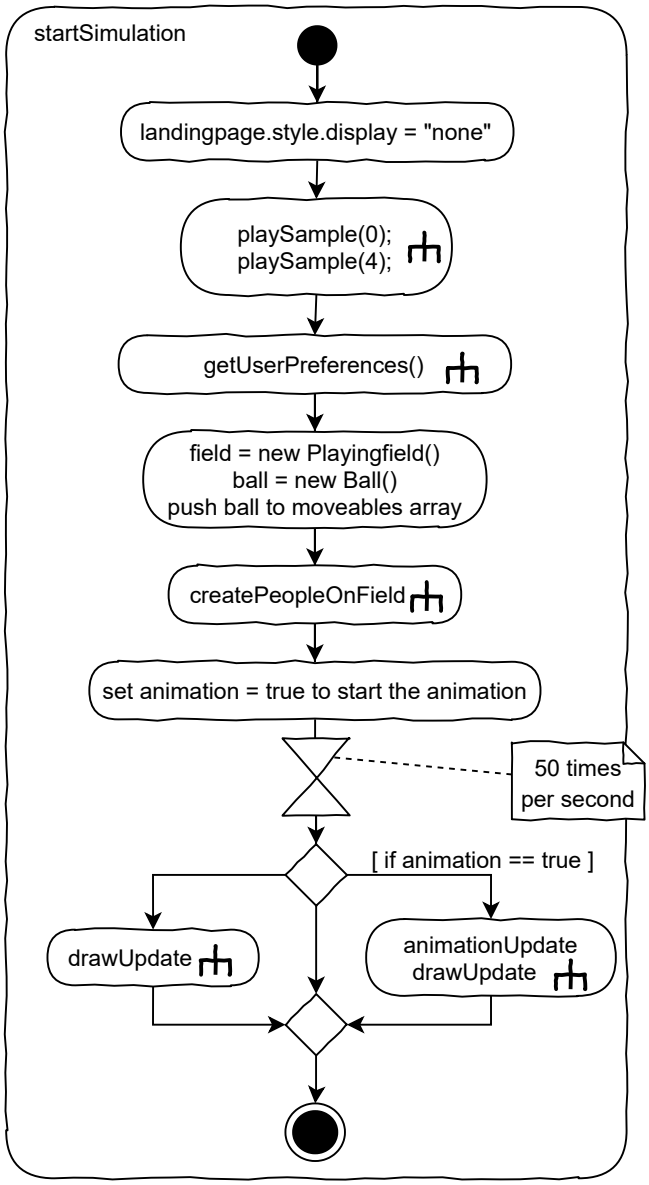


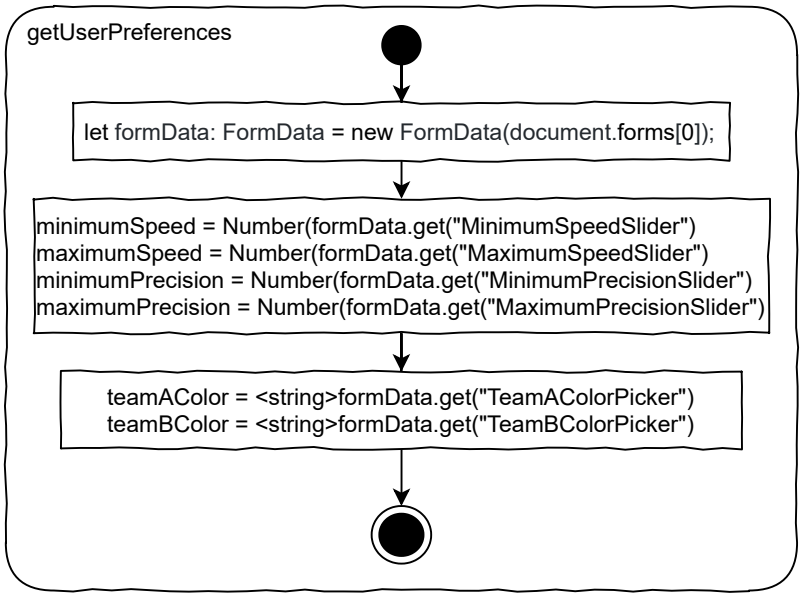


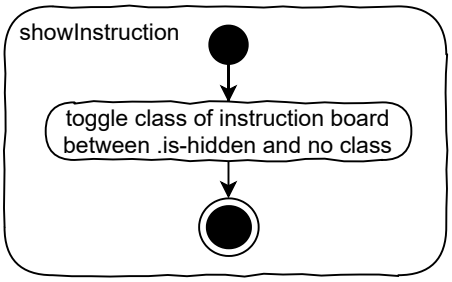


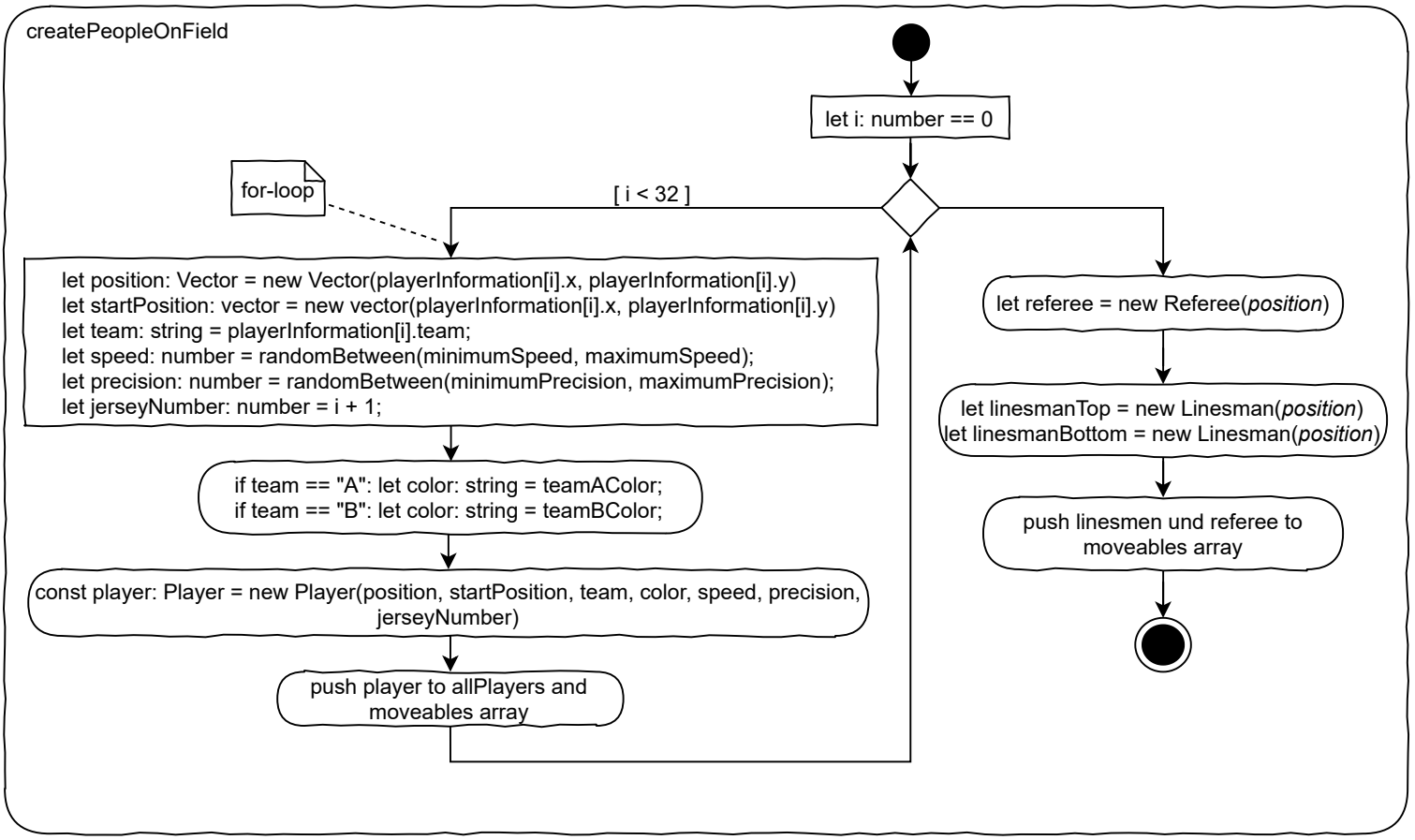


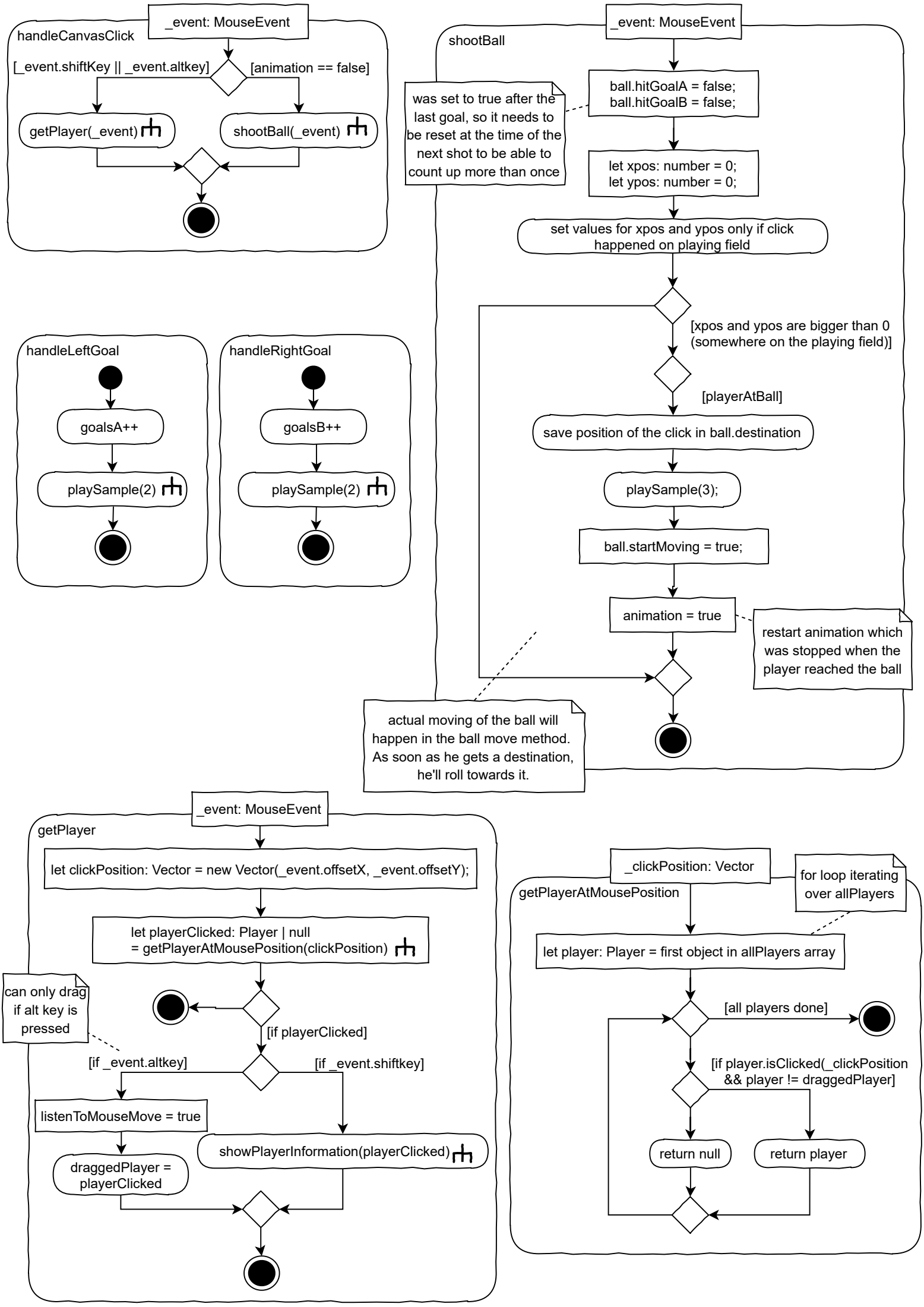


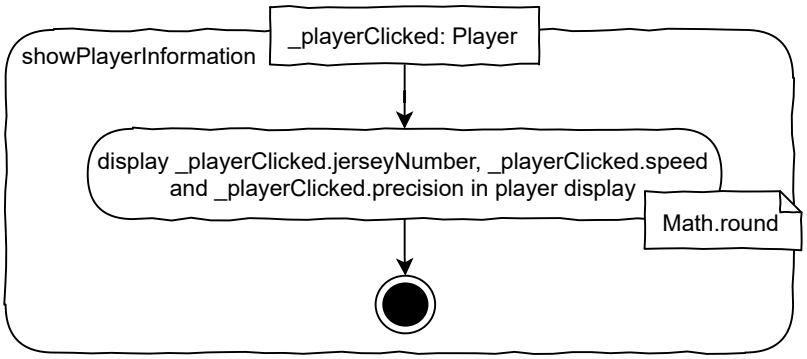

startSimulation
landingpage.style.display = "none"
playSample(0);
playSample(4);
getUserPreferences()
field = new Playingfield()
ball = new Ball()
push ball to moveables array
createPeopleOnField
set animation = true to start the animation
50 times
per second
[if animation == true]
drawUpdate
animationUpdate
drawUpdate

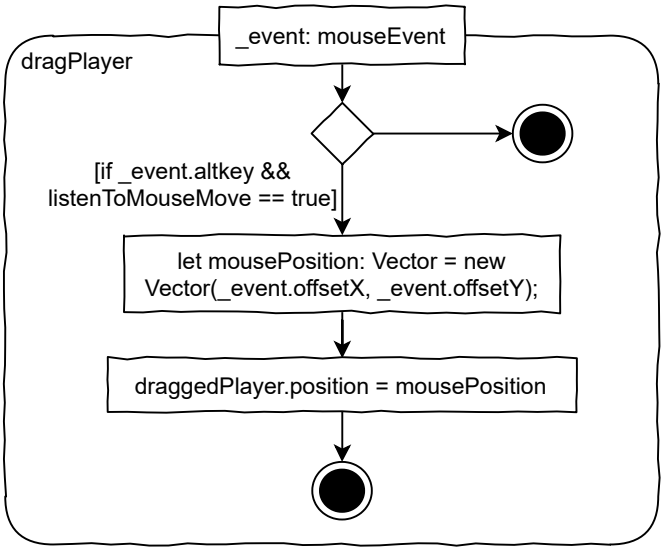

getUserPreferences
let formData: FormData = new FormData(document.forms[0]);
minimumSpeed = Number(formData.get("MinimumSpeedSlider")
maximumSpeed = Number(formData.get("MaximumSpeedSlider")
minimumPrecision = Number(formData.get("MinimumPrecisionSlider")
maximumPrecision = Number(formData.get("MaximumPrecisionSlider")
teamAColor = <string>formData.get("TeamAColorPicker")
teamBColor = <string>formData.get("TeamBColorPicker")

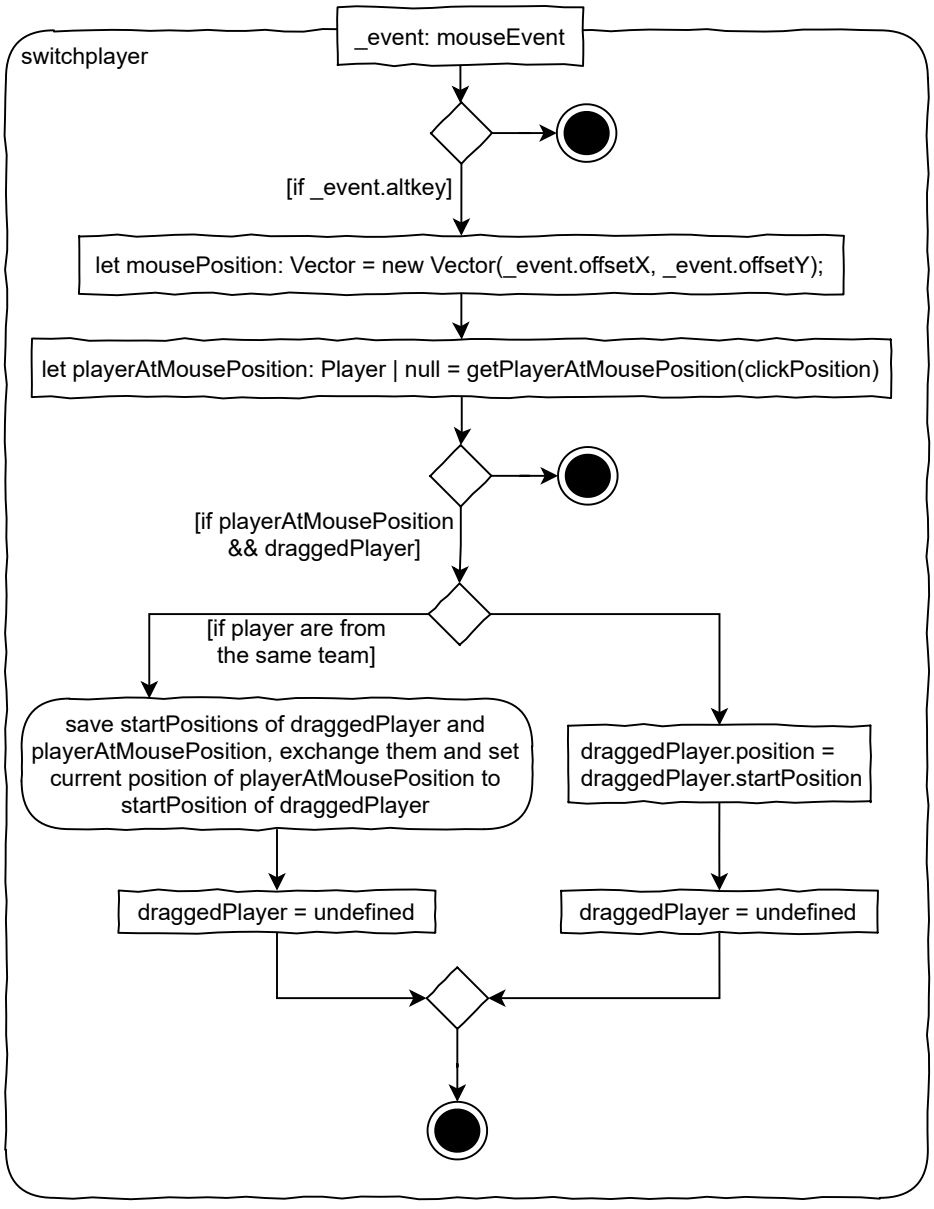

showInstruction
toggle class of instruction board
between .is-hidden and no class

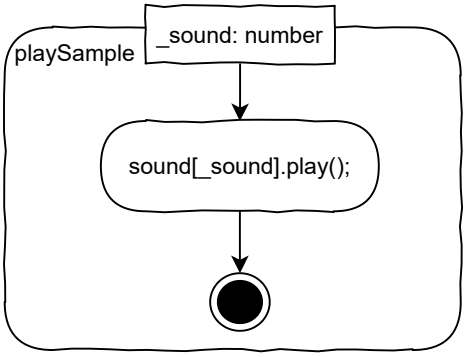

createPeopleOnField
let i: number == 0
for-loop
[i < 32]
let position: Vector = new Vector(playerInformation[i].x, playerInformation[i].y)
let startPosition: vector = new vector(playerInformation[i].x, playerInformation[i].y)
let team: string = playerInformation[i].team;
let speed: number = randomBetween(minimumSpeed, 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


handleCanvasClick
_event: MouseEvent
[_event.shiftKey || _event.altkey]
[animation == false]
getPlayer(_event)
shootBall(_event)
shootBall
_event: MouseEvent
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
ball.hitGoalA = false;
ball.hitGoalB = false;
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)
handleRightGoal
goalsB++
playSample(2)
getPlayer
_event: MouseEvent
let clickPosition: Vector = new Vector(_event.offsetX, _event.offsetY);
let playerClicked: Player | null
= getPlayerAtMousePosition(clickPosition)
can only drag if alt key is pressed
[if playerClicked]
[if _event.altkey]
[if _event.shiftkey]
listenToMouseMove = true
draggedPlayer = playerClicked
showPlayerInformation(playerClicked)
getPlayerAtMousePosition
_clickPosition: Vector
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



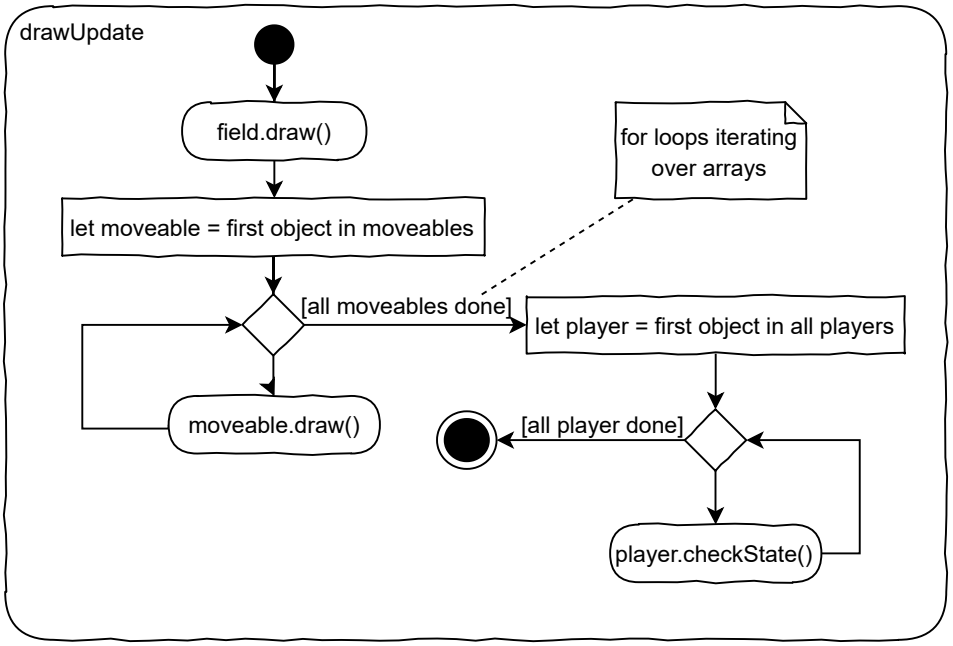


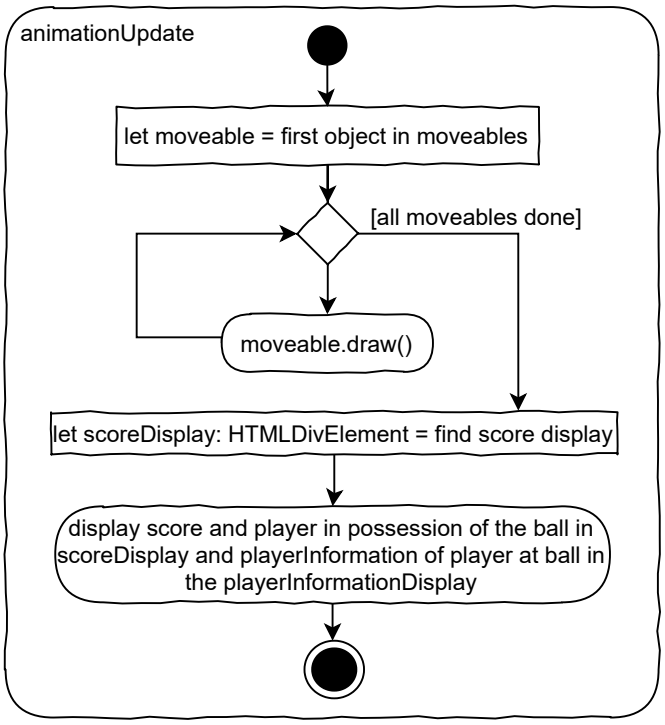




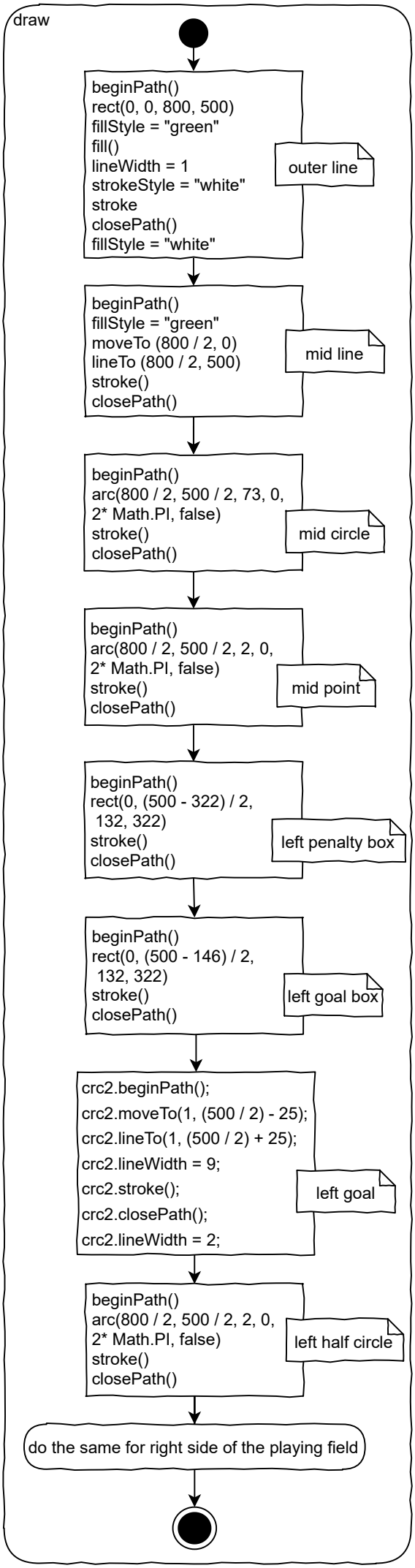
when draggedPlayer is overlapping with a field player at releasing the mouse, they switch their positions.

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

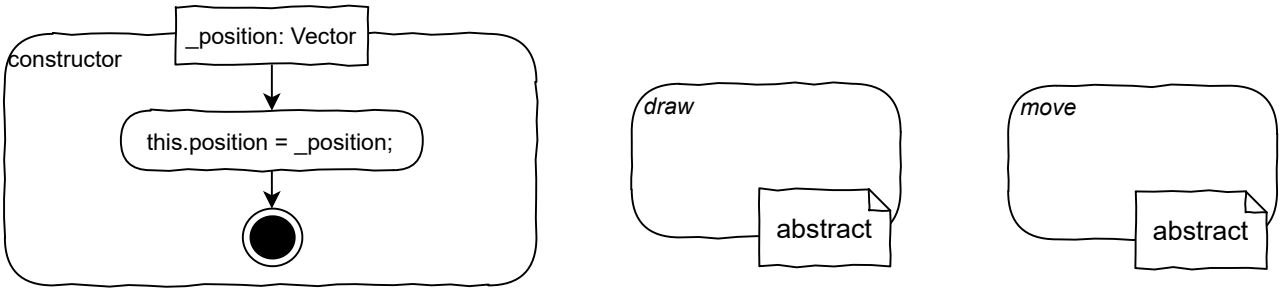

drawUpdate
field.draw()
for loops iterating over arrays
let moveable = first object in moveables
[all moveables done]
let player = first object in all players
moveable.draw()
[all player done]
player.checkState()


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

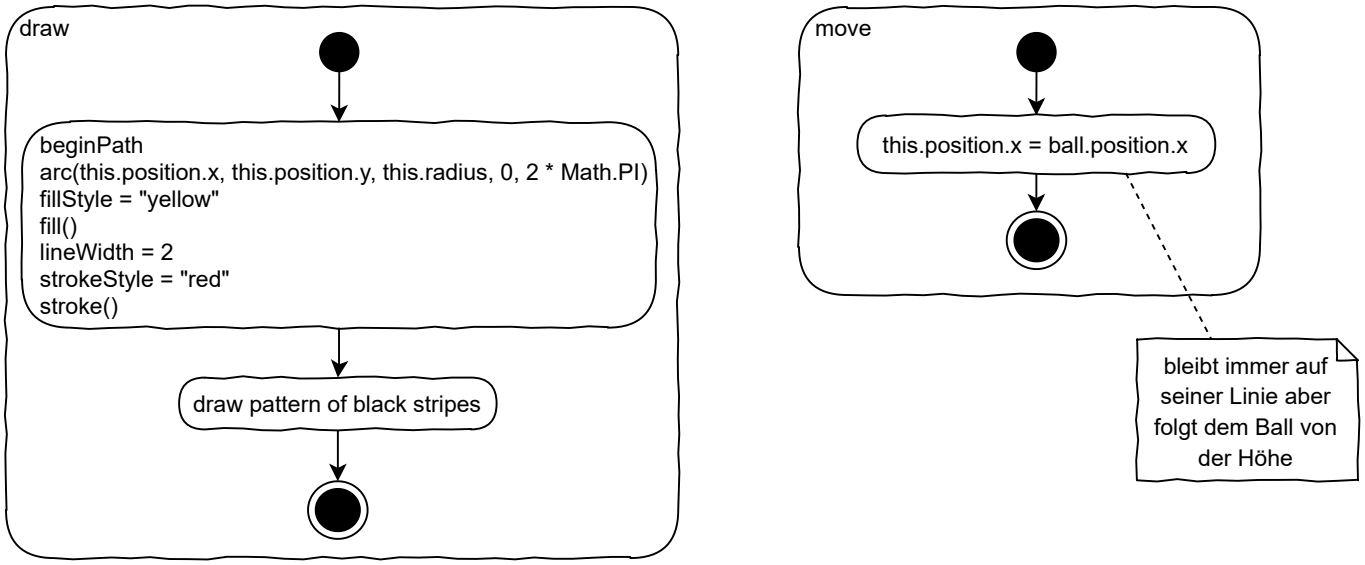
Playing Field Methods



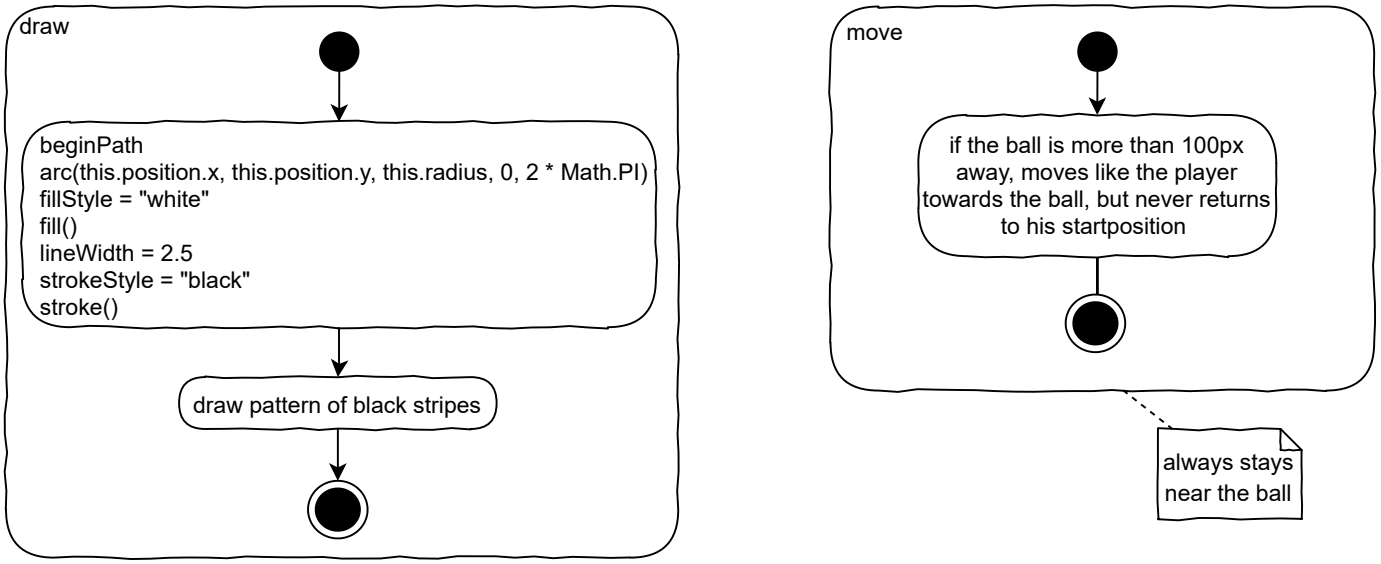
Moveable Methods



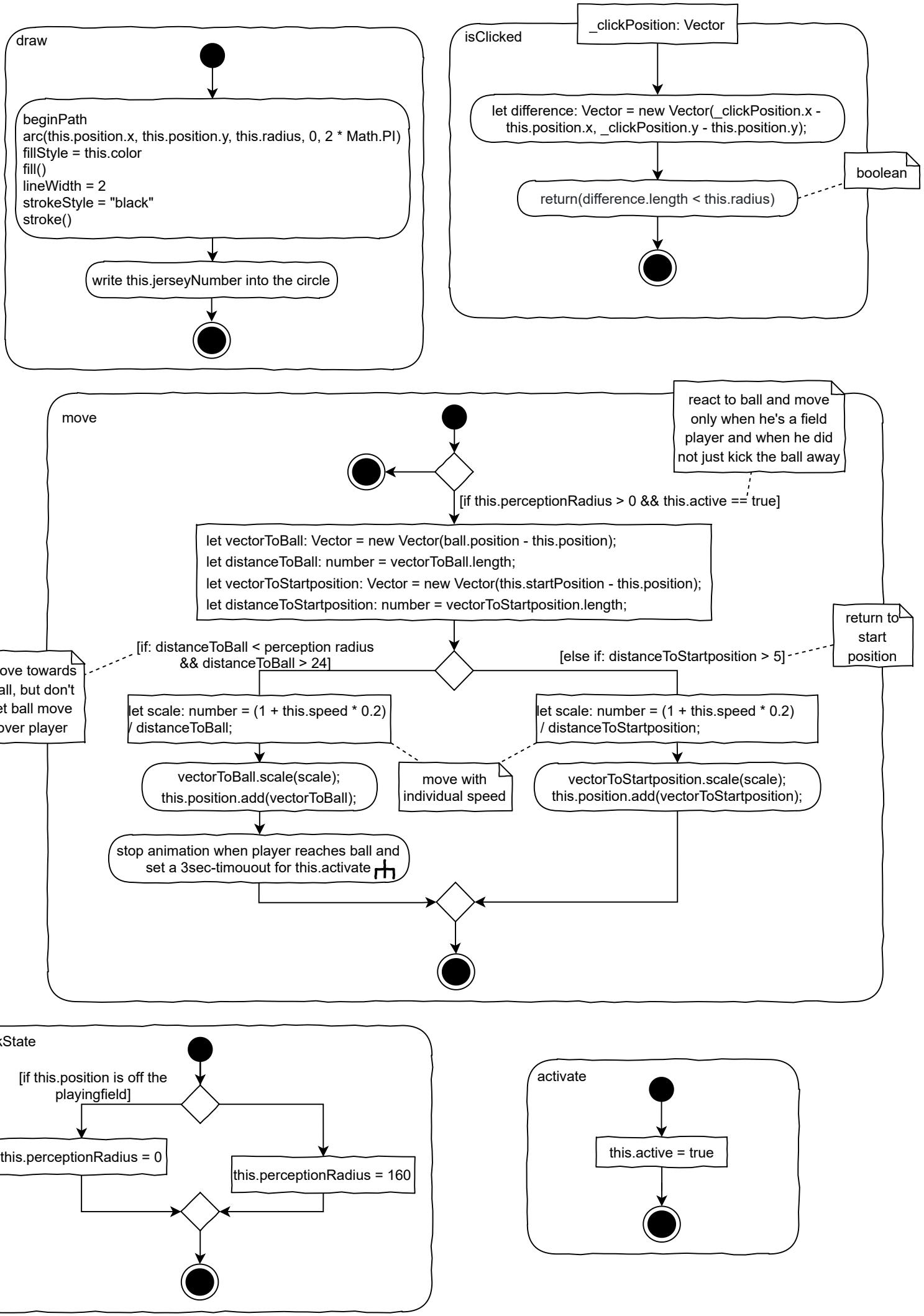
Linesman Methods



Referee Methods



Player Methods



Ball Methods