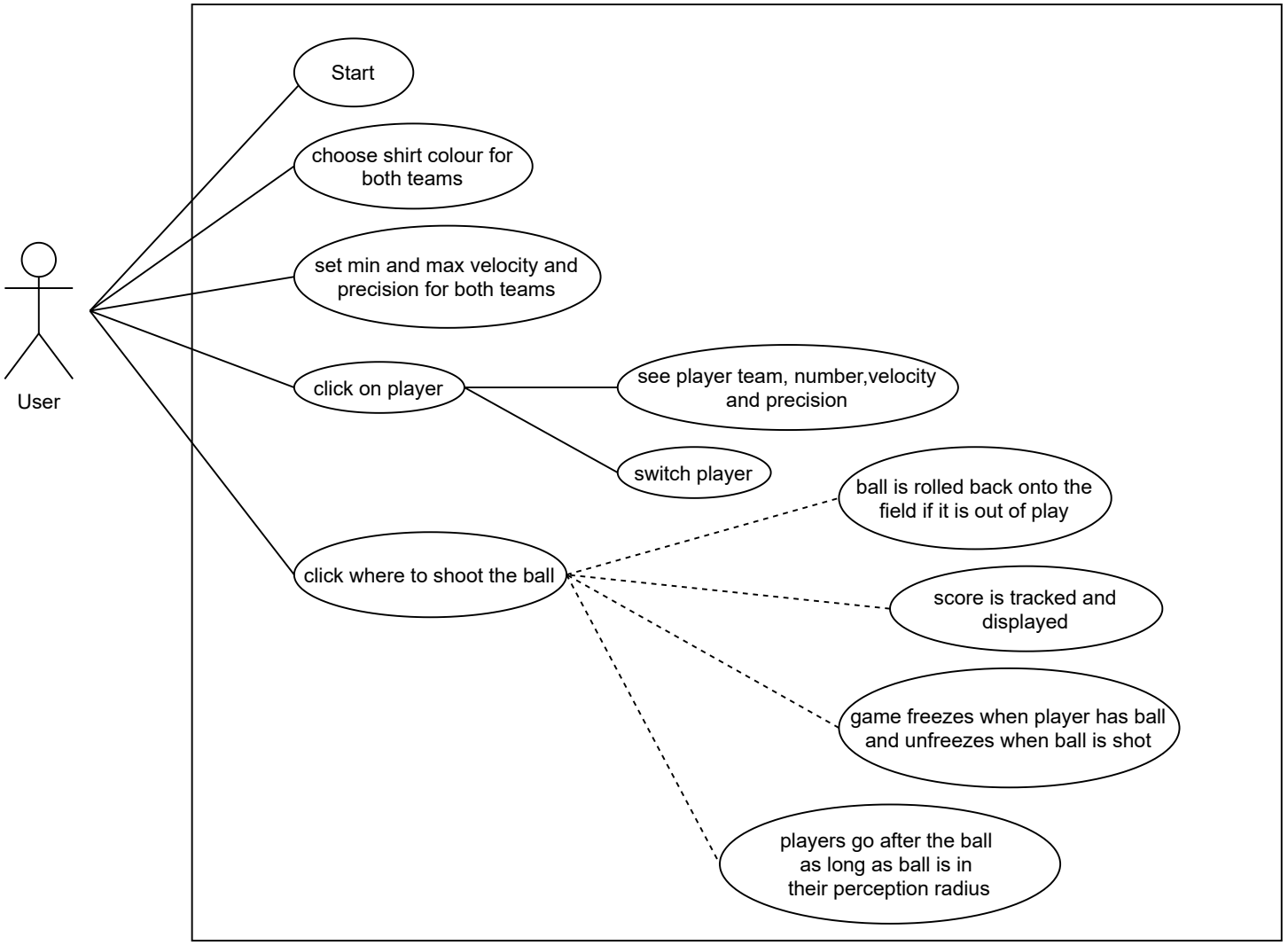
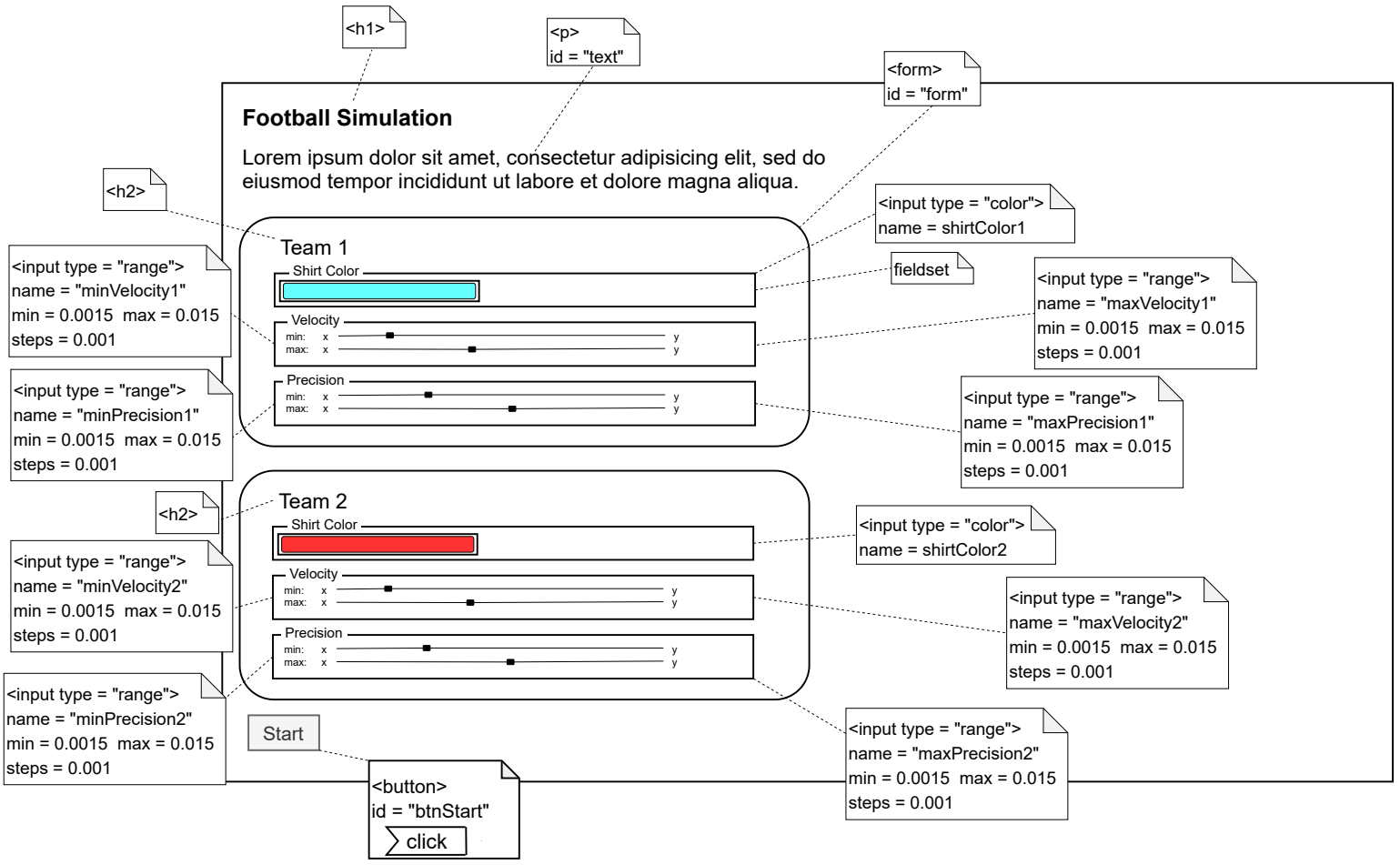
Abschlussarbeit SoSe 2021 - Football Simulation

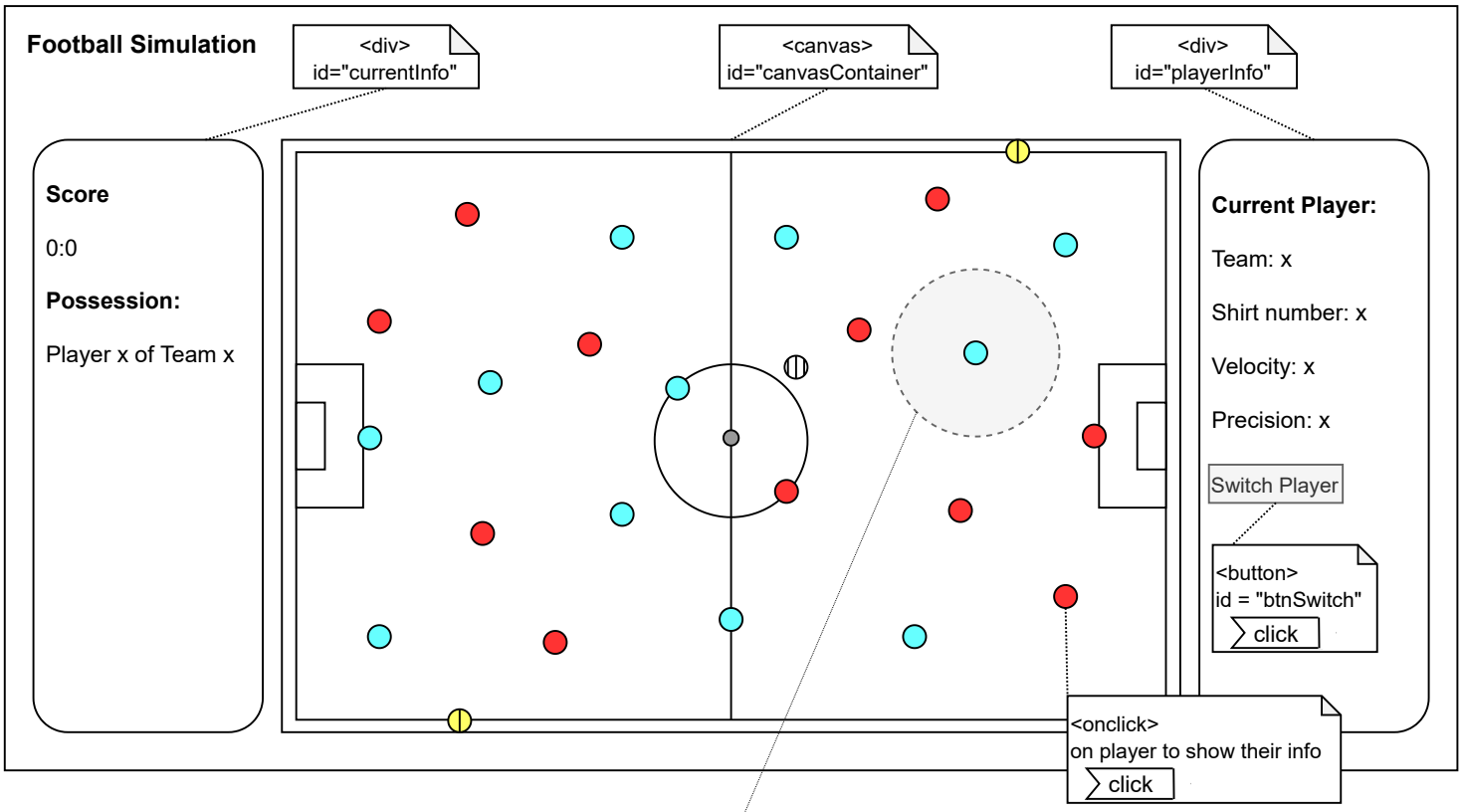
Use-Case-Diagramm

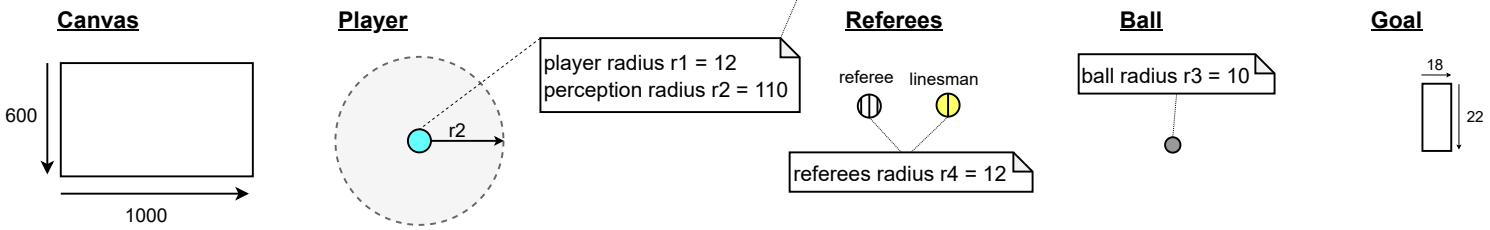


UI Scribble - Form

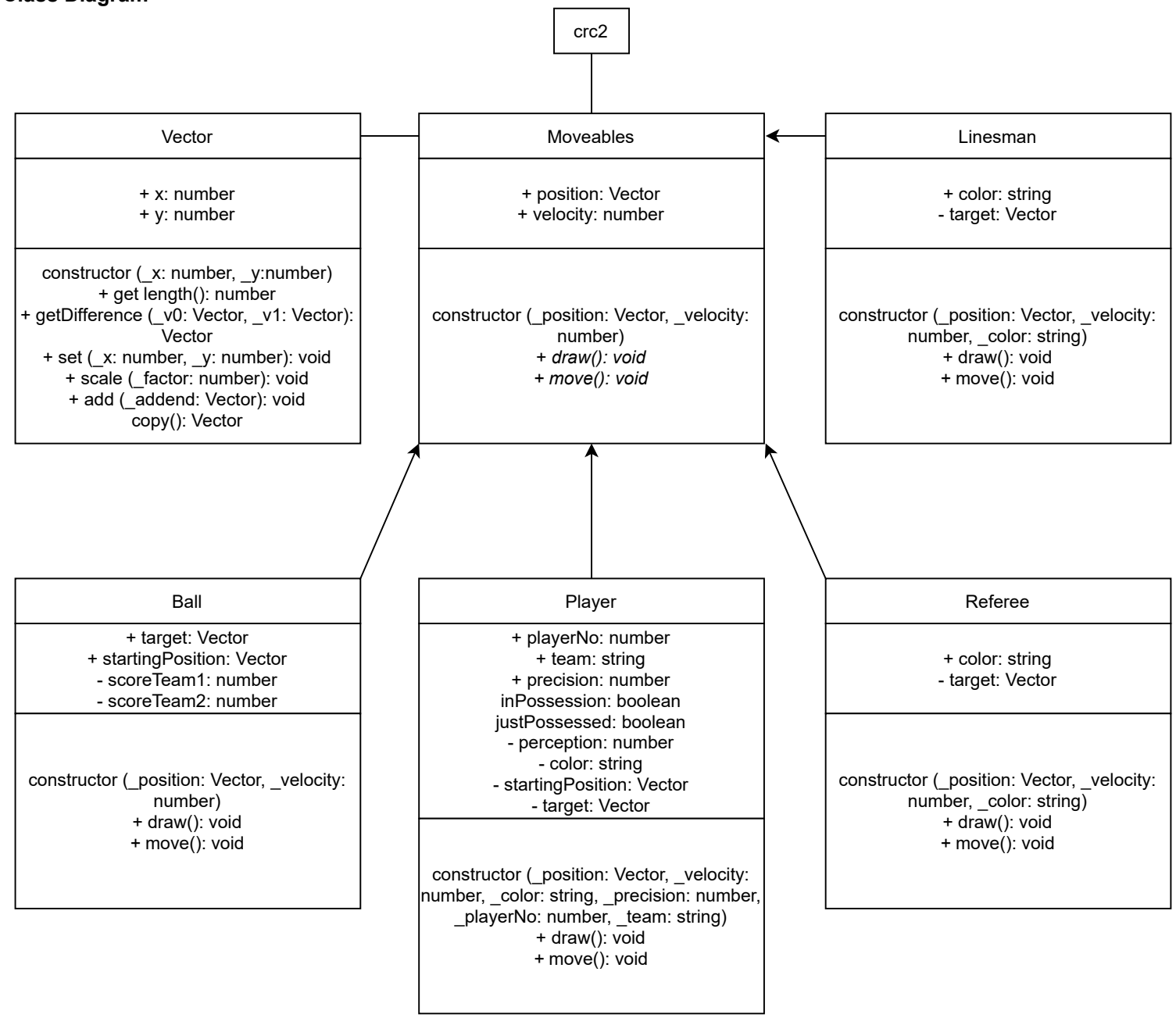


UI Scribble - Field

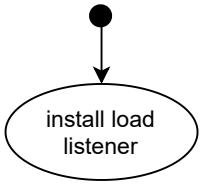




Class Diagram



Activity Diagram - Main



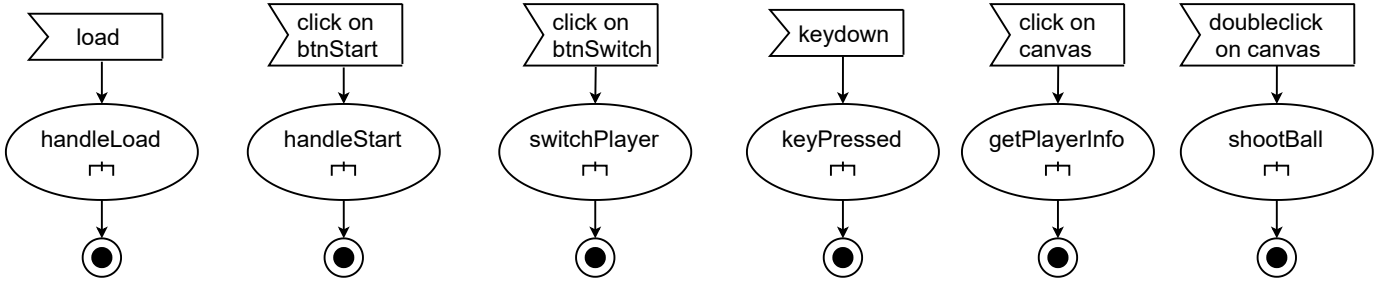
export let canvas:HTMLCanvasElement
export let crc2: CanvasRenderingContext2D
export let football: Ball
export let freeze: boolean = false
export let currentPlayer: Player
export let selectedPlayer: Player
export let currentPrecision: number
export letimageData: ImageData

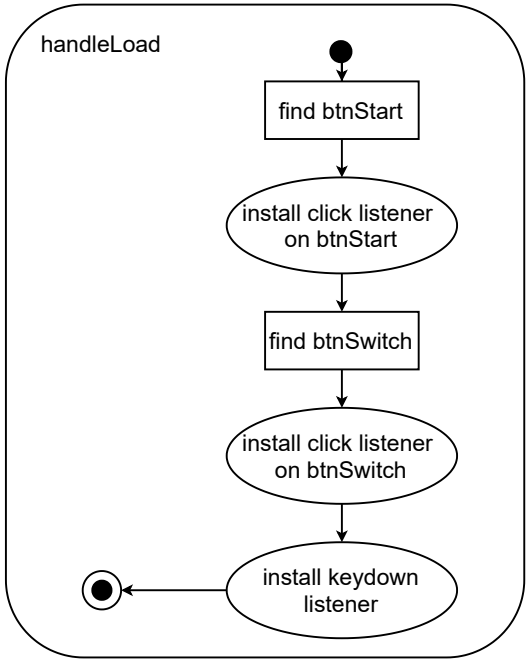
let firstPage: boolean = true
let btnStart: HTMLButtonElement
let btnSwitch: HTMLButtonElement
let formData: FormData
let timesSwitched1: number = 1
let timesSwitched2: number = 1

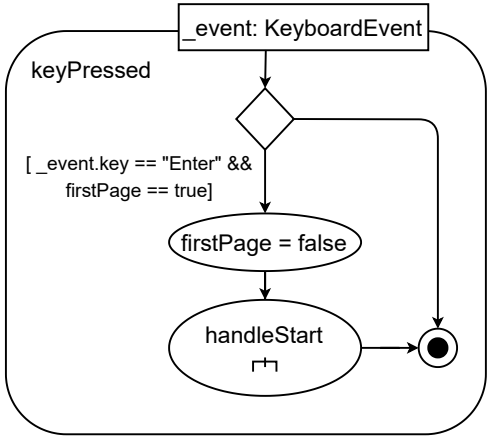
let shirtColor1: string
let minVelocity1: number
let maxVelocity1: number
let minPrecision1: number
let maxPrecision1: number

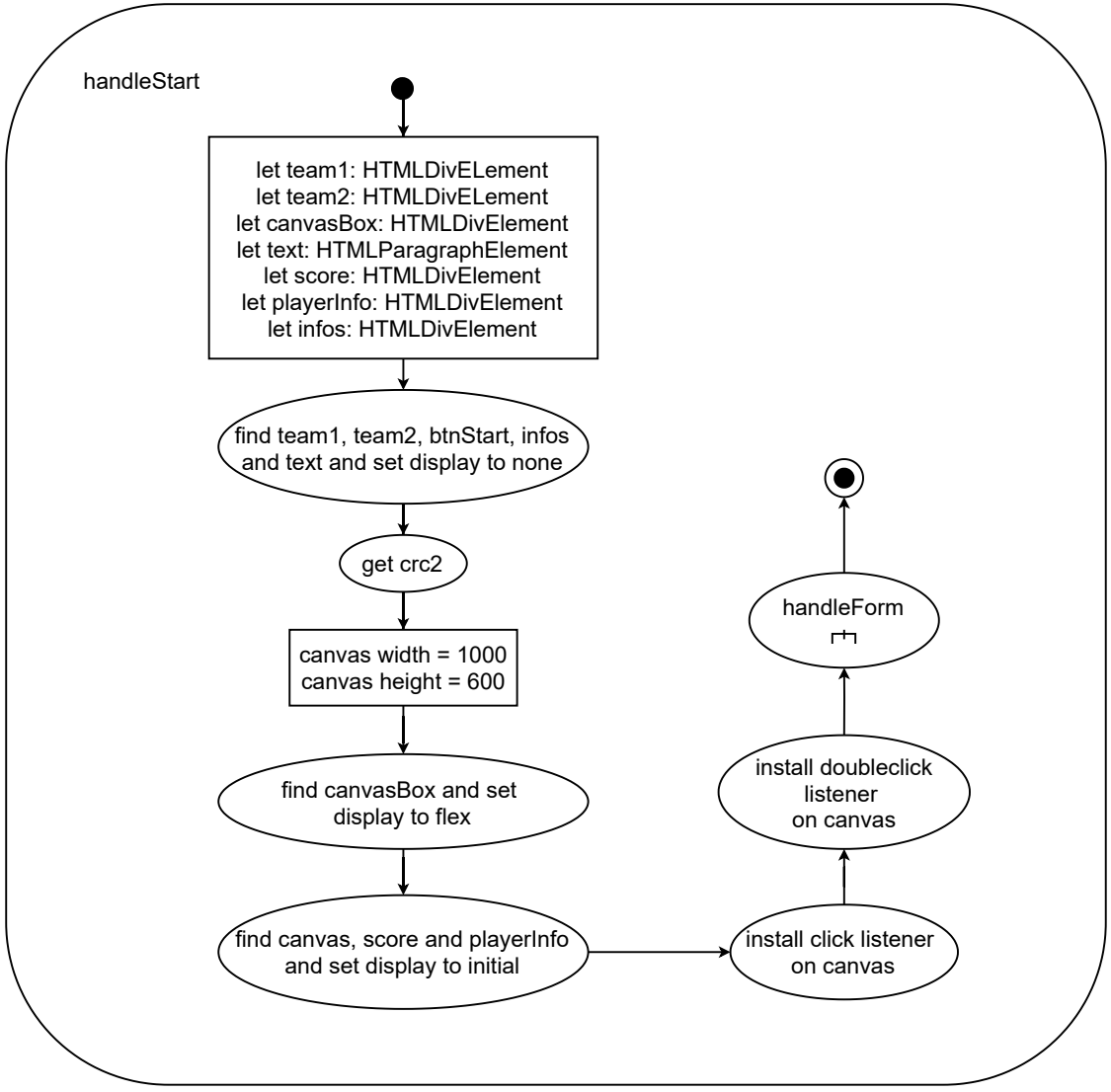
let shirtColor2: string
let minVelocity2: number
let maxVelocity2: number
let minPrecision2: number
let maxPrecision2: number

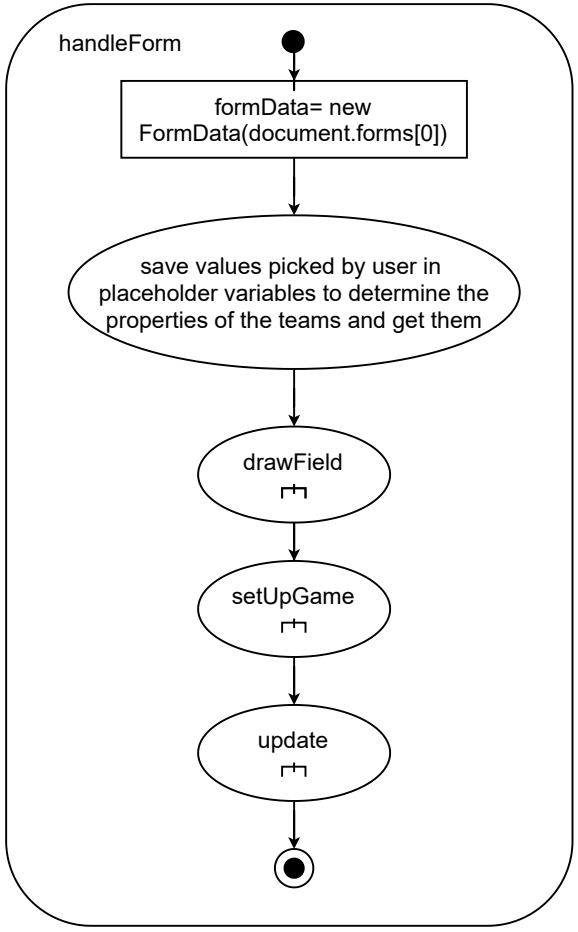
let refColor: string
let linesmanColor: string

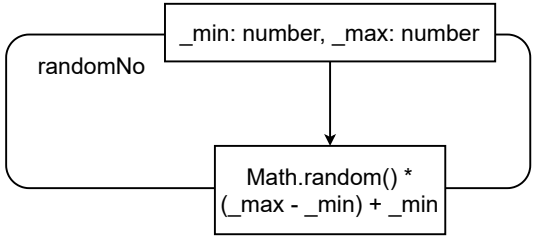


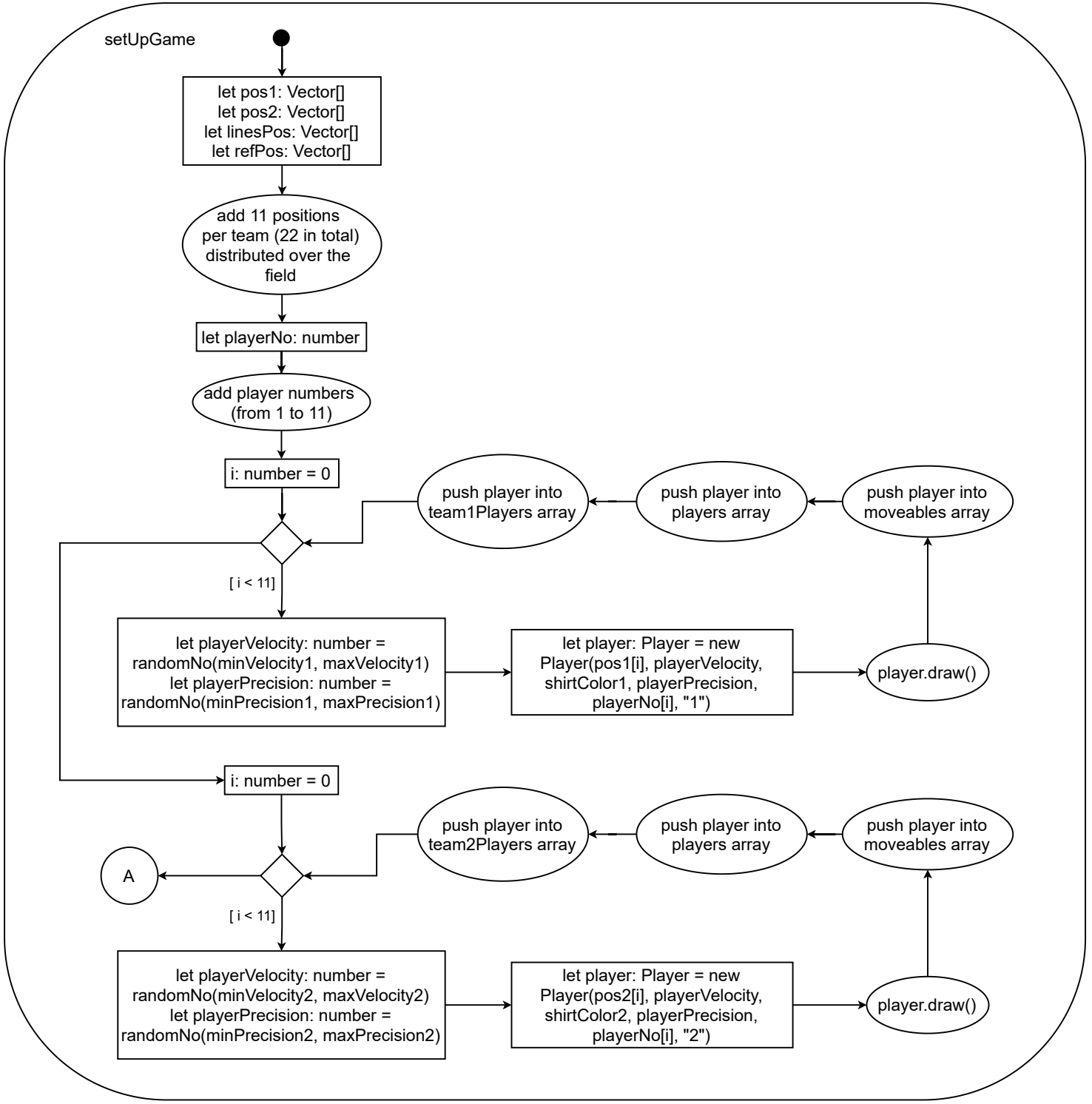


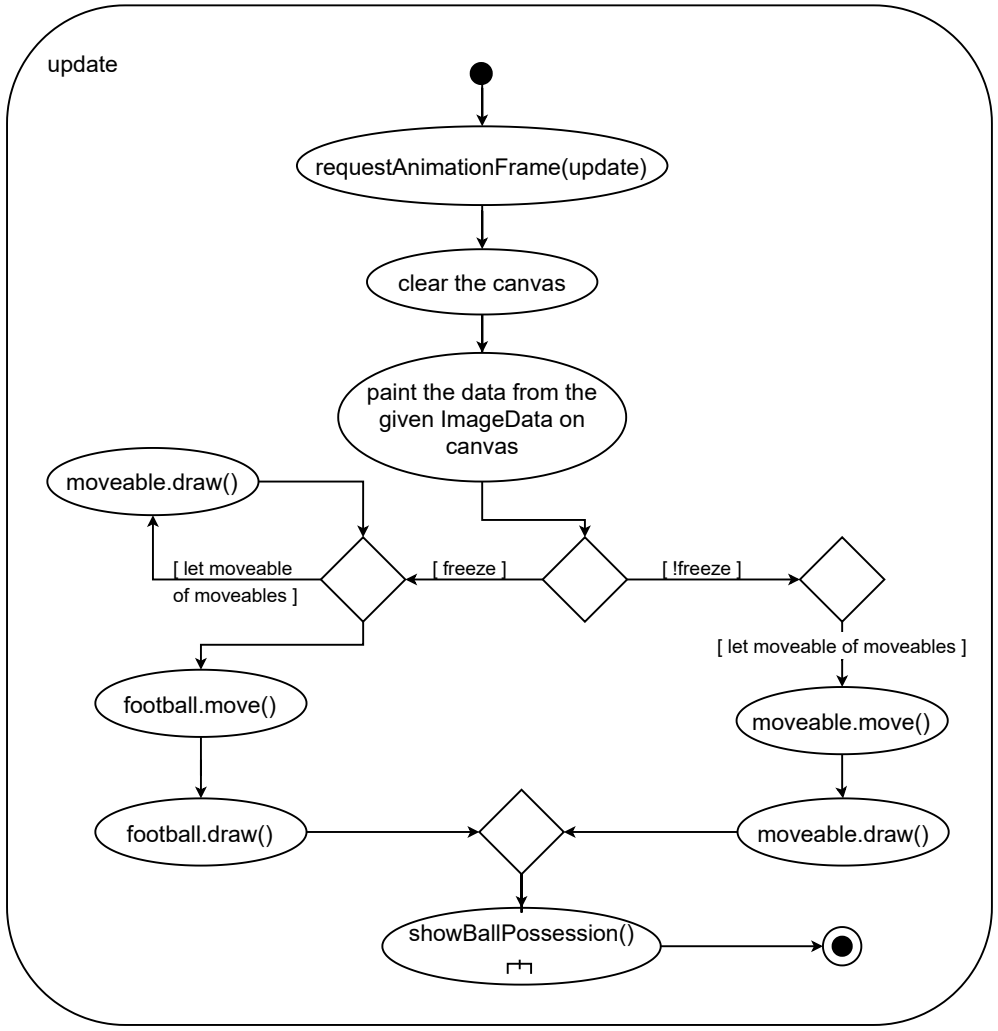


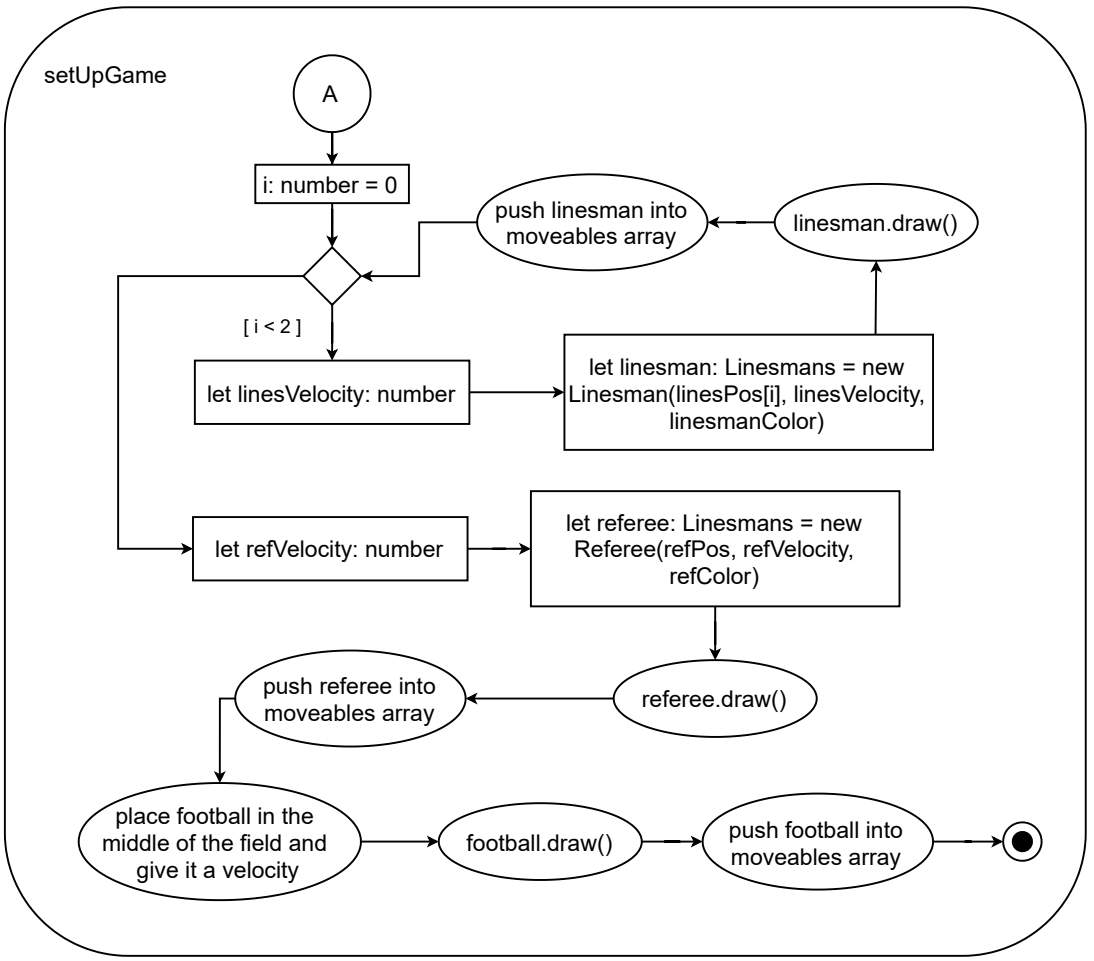


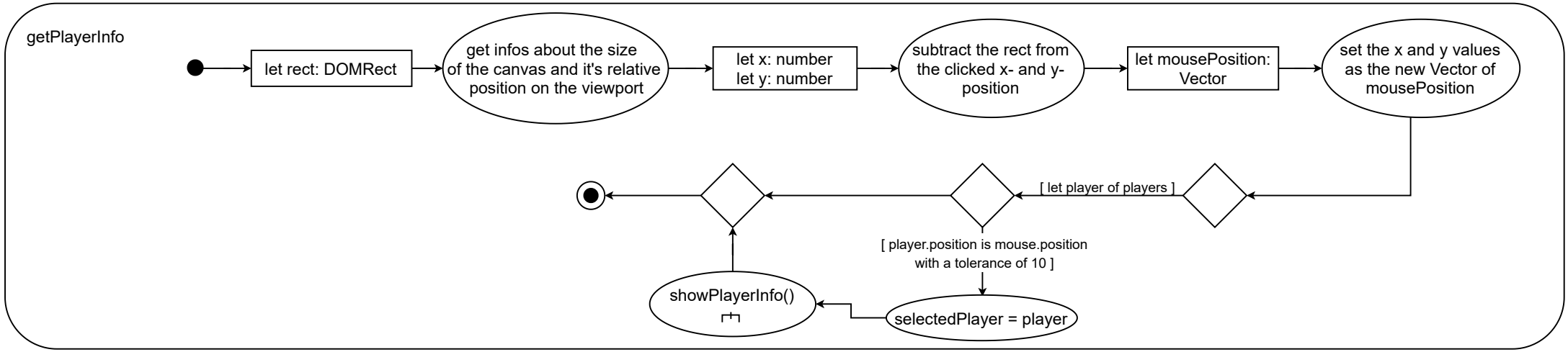

handleForm
formData= new FormData(document.forms[0])
save values picked by user in placeholder variables to determine the properties of the teams and get them
drawField
setUpGame
update

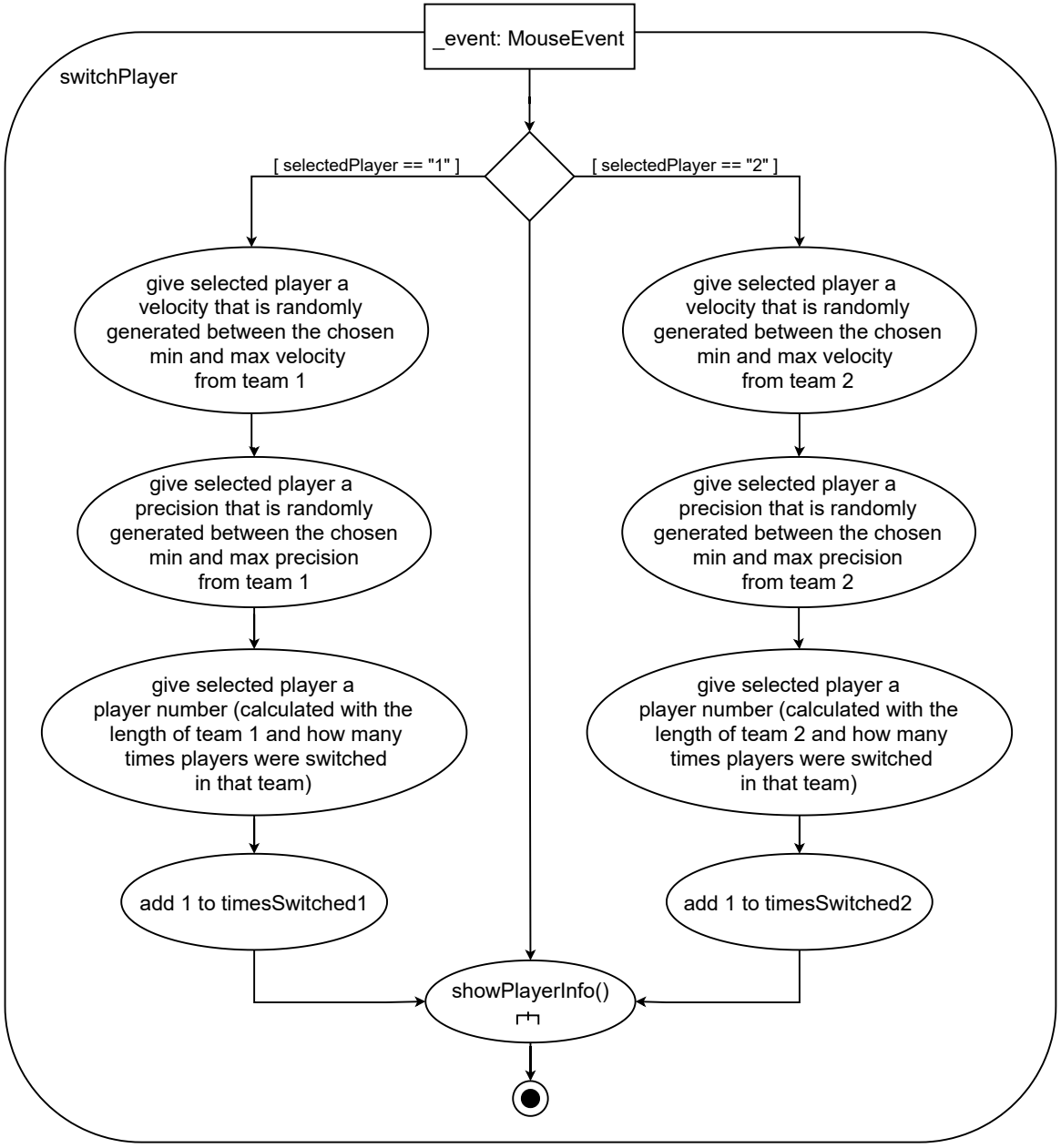

randomNo
_min: number, _max: number
Math.random() * (_max - _min) + _min

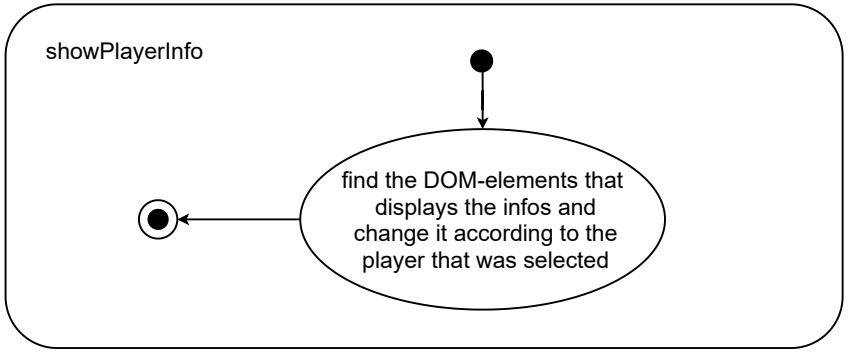

setUpGame
let pos1: Vector[]
let pos2: Vector[]
let linesPos: Vector[]
let refPos: Vector[]
add 11 positions per team (22 in total) distributed over the field
let playerNo: number
add player numbers (from 1 to 11)
i: number = 0
push player into team1Players array
push player into players array
push player into moveables array
[i < 11]
let playerVelocity: number = randomNo(minVelocity1, maxVelocity1)
let playerPrecision: number = randomNo(minPrecision1, maxPrecision1)
let player: Player = new Player(pos1[i], playerVelocity, shirtColor1, playerPrecision, playerNo[i], "1")
player.draw()
i: number = 0
push player into team2Players array
push player into players array
push player into moveables array
A
[i < 11]
let playerVelocity: number = randomNo(minVelocity2, maxVelocity2)
let playerPrecision: number = randomNo(minPrecision2, maxPrecision2)
let player: Player = new Player(pos2[i], playerVelocity, shirtColor2, playerPrecision, playerNo[i], "2")
player.draw()

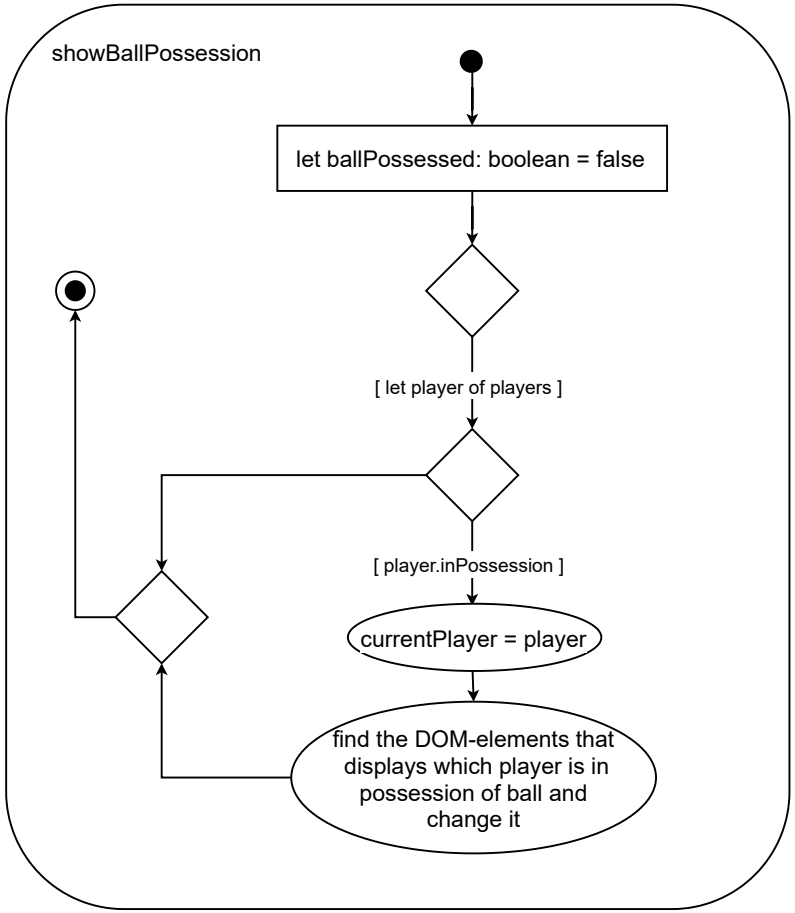

update
requestAnimationFrame(update)
clear the canvas
paint the data from the given ImageData on canvas
moveable.draw()
[let moveable of moveables]
[freeze]
[!freeze]
[let moveable of moveables]
football.move()
moveable.move()
football.draw()
moveable.draw()
showBallPossession()

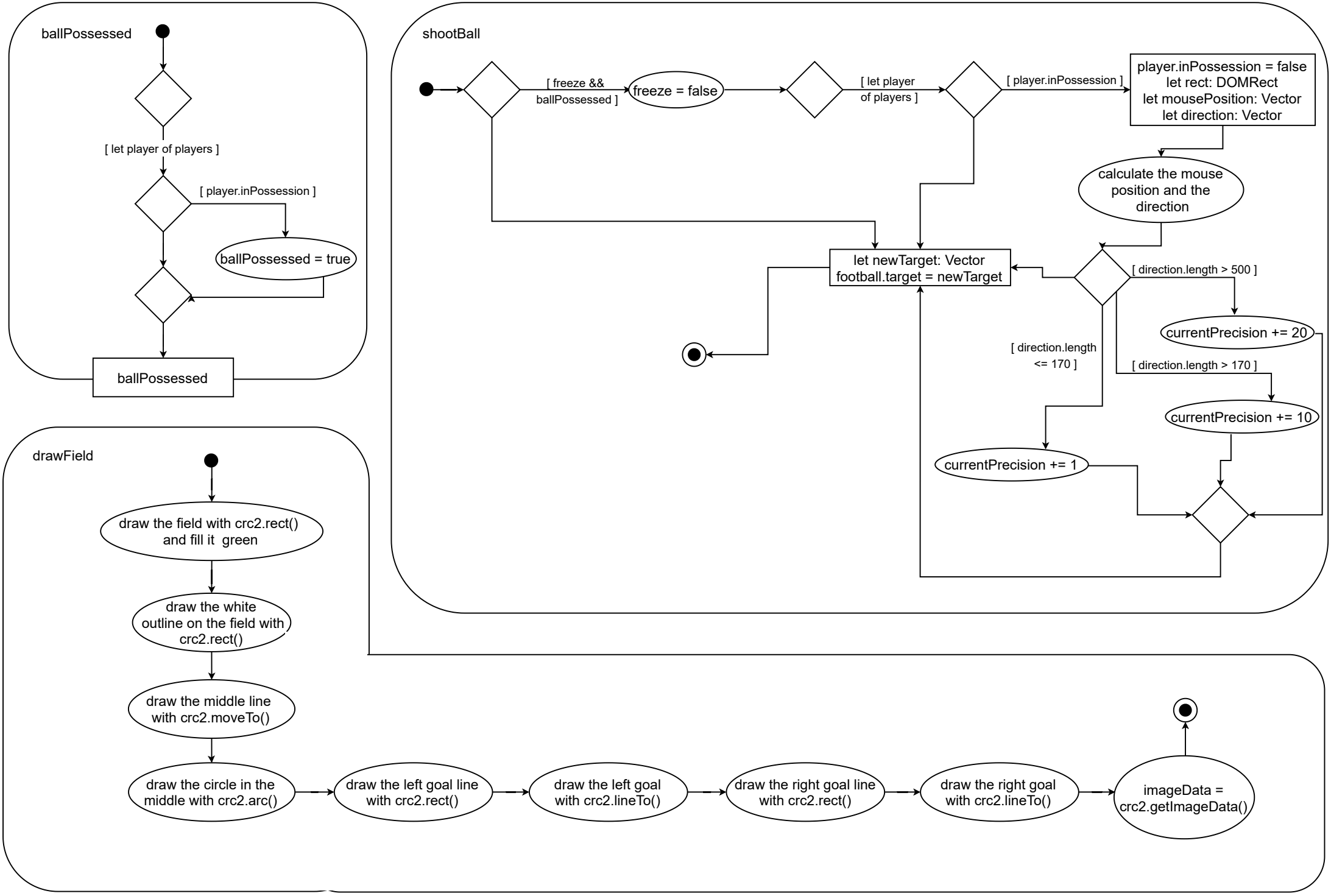

setUpGame
A
i: number = 0
push linesman into moveables array
linesman.draw()
[i < 2]
let linesVelocity: number
let linesman: Linesmans = new Linesman(linesPos[i], linesVelocity, linesmanColor)
let refVelocity: number
let referee: Linesmans = new Referee(refPos, refVelocity, refColor)
push referee into moveables array
referee.draw()
place football in the middle of the field and give it a velocity
football.draw()
push football into moveables array


getPlayerInfo
let rect: DOMRect
get infos about the size of the canvas and it's relative position on the viewport
let x: number
let y: number
subtract the rect from the clicked x- and y-position
let mousePosition: Vector
set the x and y values as the new Vector of mousePosition
[let player of players]
[player.position is mouse.position with a tolerance of 10]
showPlayerInfo()
selectedPlayer = player

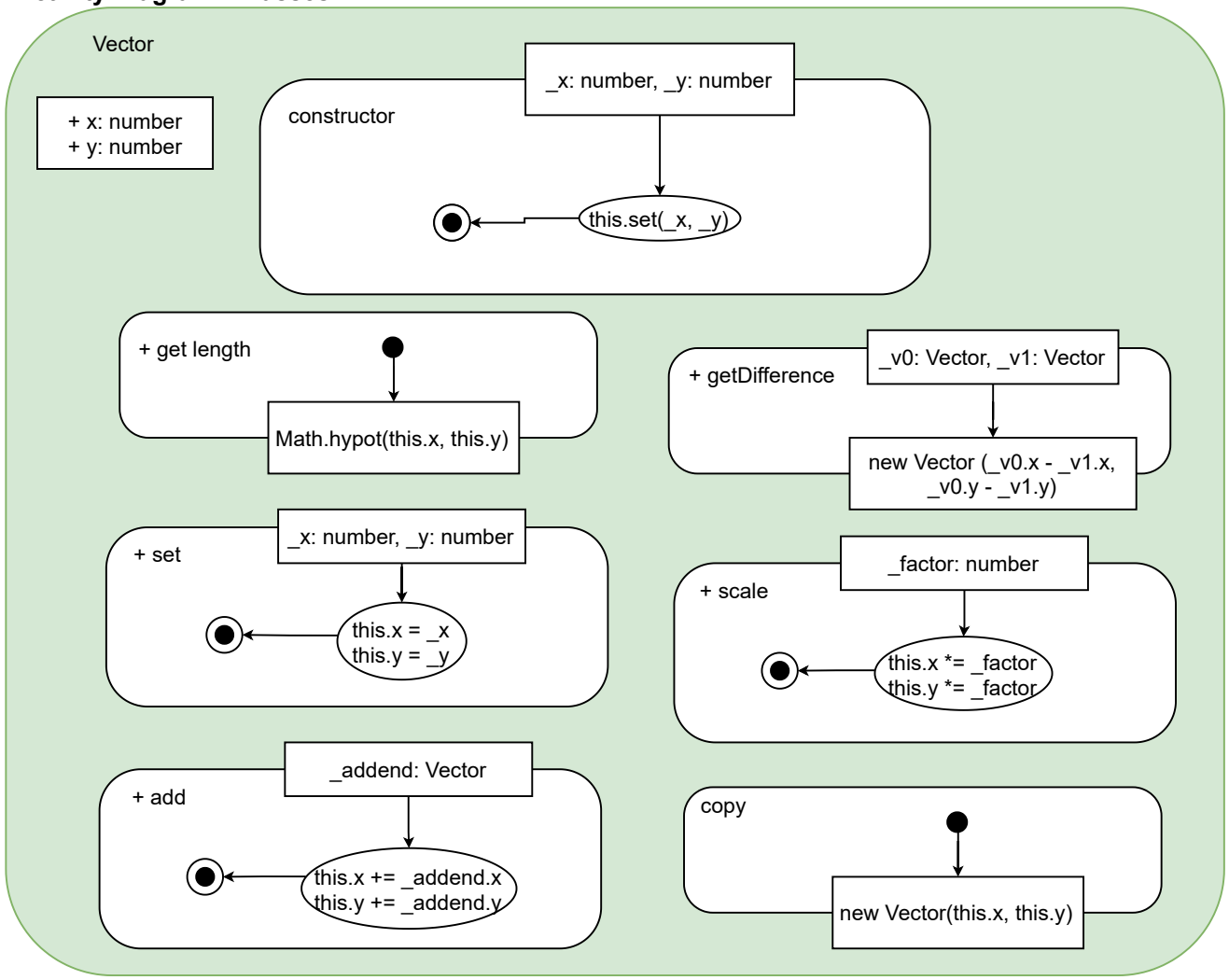

switchPlayer
_event: MouseEvent
[selectedPlayer == "1"]
[selectedPlayer == "2"]
give selected player a velocity that is randomly generated between the chosen min and max velocity from team 1
give selected player a velocity that is randomly generated between the chosen min and max velocity from team 2
give selected player a precision that is randomly generated between the chosen min and max precision from team 1
give selected player a precision that is randomly generated between the chosen min and max precision from team 2
give selected player a player number (calculated with the length of team 1 and how many times players were switched in that team)
give selected player a player number (calculated with the length of team 2 and how many times players were switched in that team)
add 1 to timesSwitched1
add 1 to timesSwitched2
showPlayerInfo()

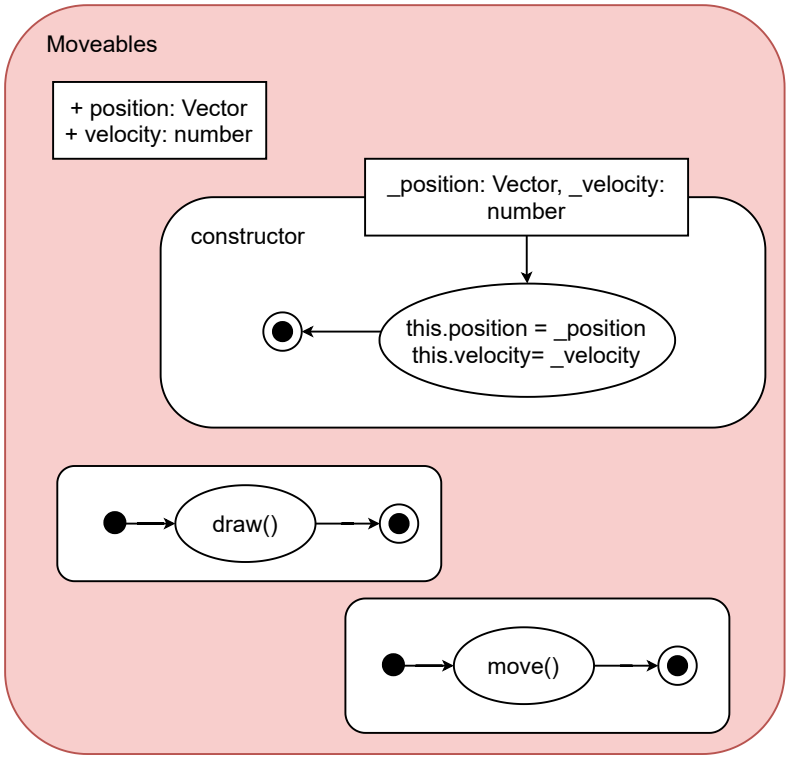

showPlayerInfo
find the DOM-elements that displays the infos and change it according to the player that was selected

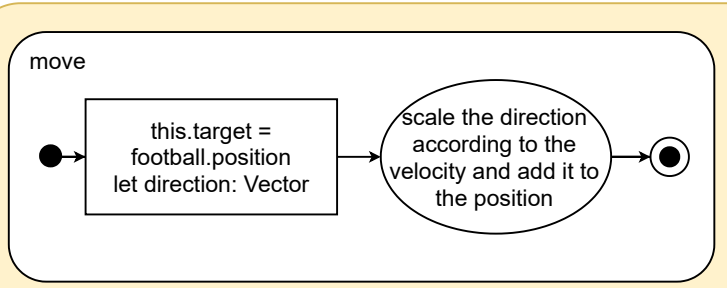

showBallPossession
let ballPossessed: boolean = false
[let player of players]
[player.inPossession]
currentPlayer = player
find the DOM-elements that displays which player is in possession of ball and change it

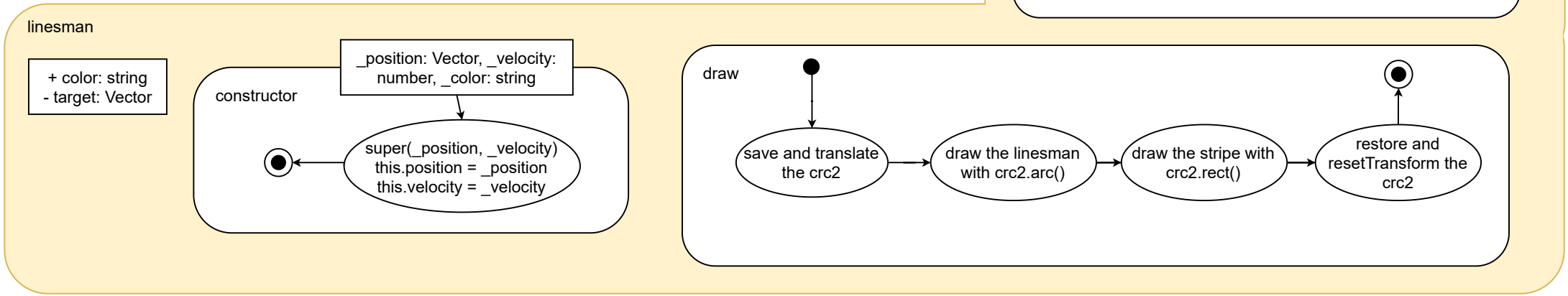

ballPossessed
[let player of players]
[player.inPossession]
ballPossessed = true
ballPossessed
shootBall
[freeze && ballPossessed]
freeze = false
[let player of players]
[player.inPossession]
player.inPossession = false
let rect: DOMRect
let mousePosition: Vector
let direction: Vector
calculate the mouse position and the direction
let newTarget: Vector
football.target = newTarget
[direction.length > 500]
currentPrecision += 20
[direction.length <= 170]
[direction.length > 170]
currentPrecision += 10
currentPrecision += 1
drawField
draw the field with crc2.rect() and fill it green
draw the white outline on the field with crc2.rect()
draw the middle line with crc2.moveTo()
draw the circle in the middle with crc2.arc()
draw the left goal line with crc2.rect()
draw the left goal with crc2.lineTo()
draw the right goal line with crc2.rect()
draw the right goal with crc2.lineTo()
imageData = crc2.getImageData()

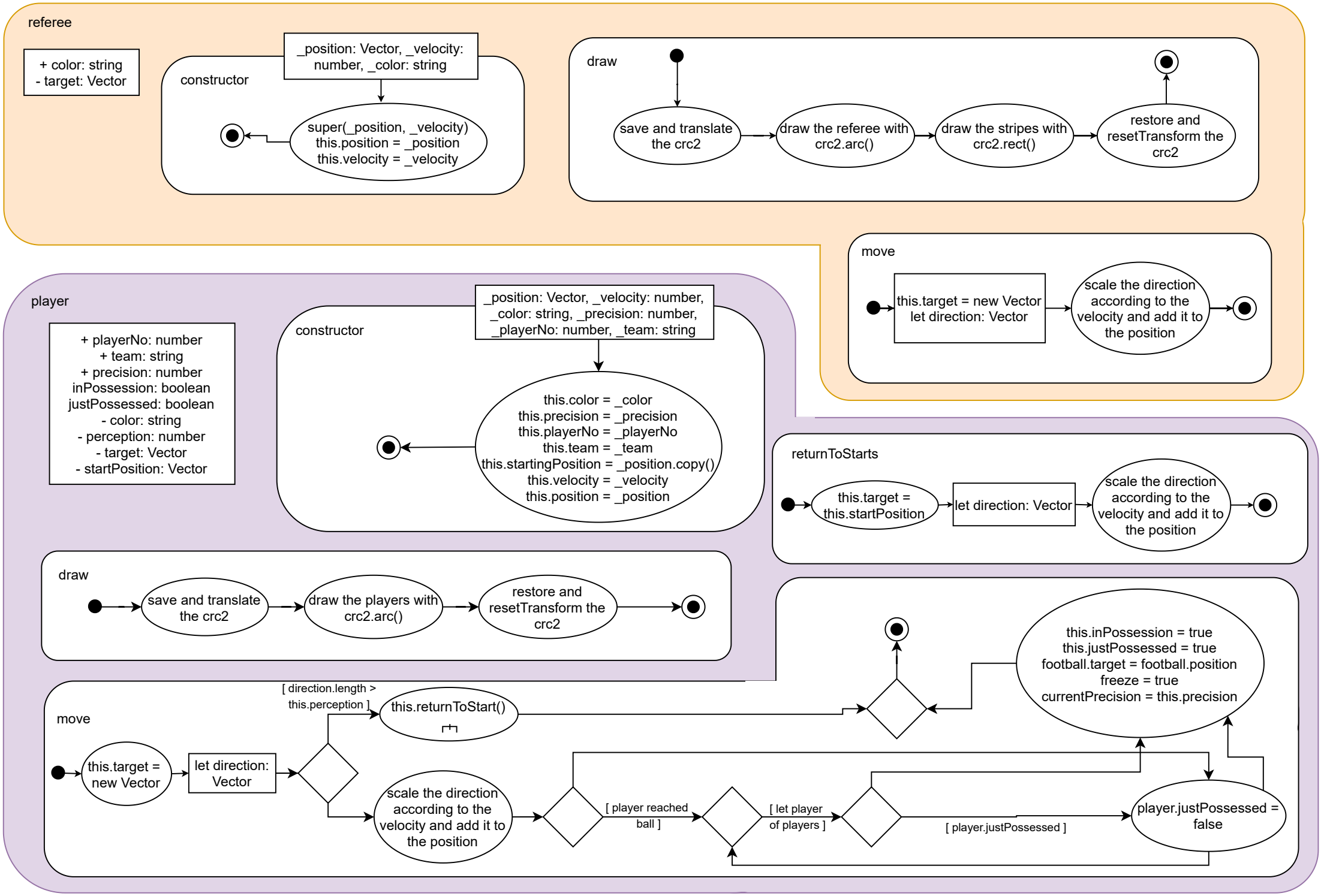
Activity Diagram - Classes

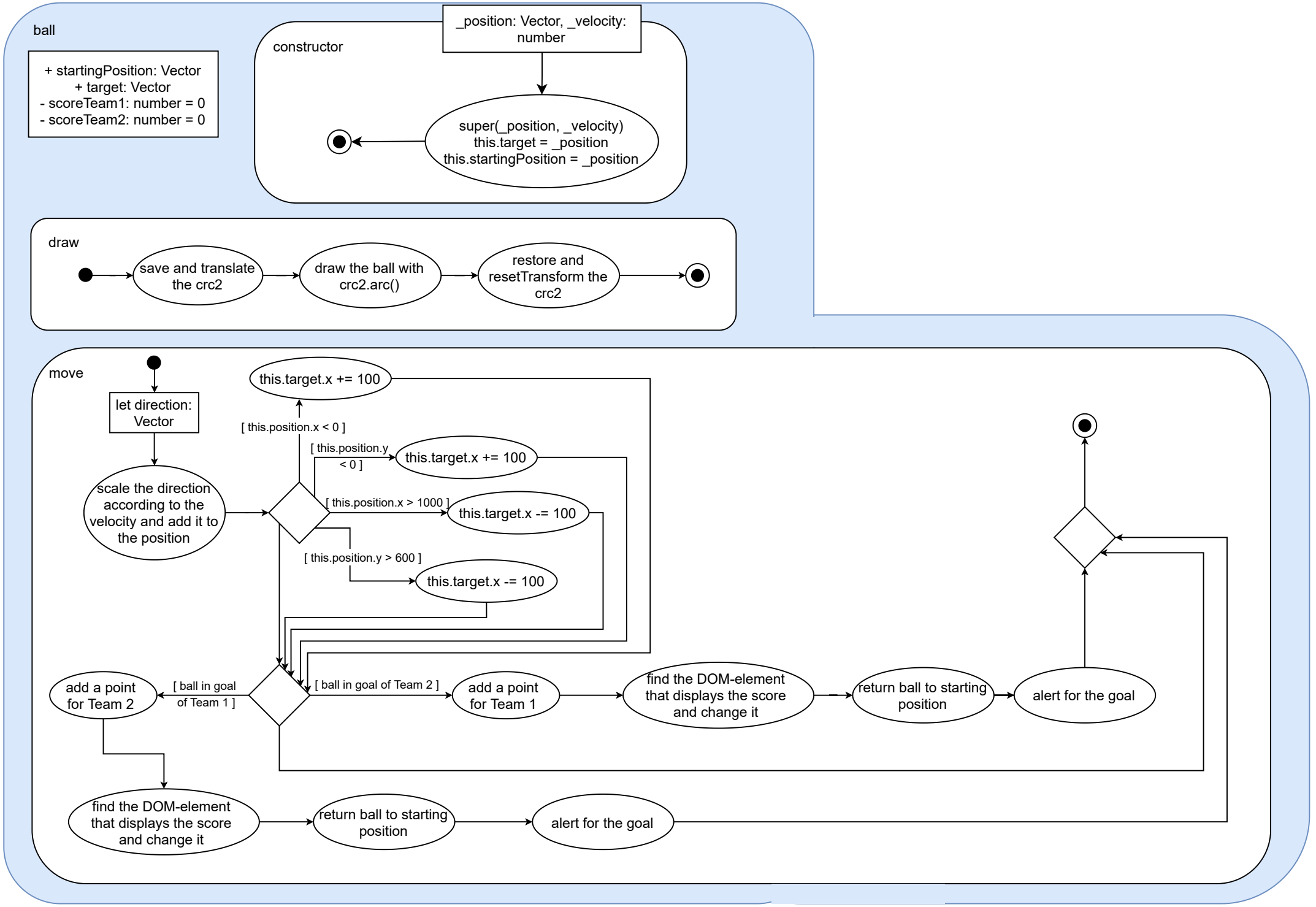









referee
+ color: string
- target: Vector
constructor
_position: Vector, _velocity: number, _color: string
super(_position, _velocity)
this.position = _position
this.velocity = _velocity
draw
save and translate the crc2
draw the referee with crc2.arc()
draw the stripes with crc2.rect()
restore and resetTransform the crc2
move
this.target = new Vector
let direction: Vector
scale the direction according to the velocity and add it to the position
player
+ playerNo: number
+ team: string
+ precision: number
inPossession: boolean
justPossessed: boolean
- color: string
- perception: number
- target: Vector
- startPosition: Vector
constructor
_position: Vector, _velocity: number, _color: string, _precision: number, _playerNo: number, _team: string
this.color = _color
this.precision = _precision
this.playerNo = _playerNo
this.team = _team
this.startingPosition = _position.copy()
this.velocity = _velocity
this.position = _position
returnToStarts
this.target = this.startPosition
let direction: Vector
scale the direction according to the velocity and add it to the position
draw
save and translate the crc2
draw the players with crc2.arc()
restore and resetTransform the crc2
move
this.target = new Vector
let direction: Vector
[direction.length > this.perception]
this.returnToStart()
scale the direction according to the velocity and add it to the position
[player reached ball]
[let player of players]
[player.justPossessed]
player.justPossessed = false
this.inPossession = true
this.justPossessed = true
football.target = football.position
freeze = true
currentPrecision = this.precision


ball
+ startingPosition: Vector
+ target: Vector
- scoreTeam1: number = 0
- scoreTeam2: number = 0
constructor
_position: Vector, _velocity: number
super(_position, _velocity)
this.target = _position
this.startingPosition = _position
draw
save and translate the crc2
draw the ball with crc2.arc()
restore and resetTransform the crc2
move
let direction: Vector
scale the direction according to the velocity and add it to the position
[this.position.x < 0]
this.target.x += 100
[this.position.y < 0]
this.target.x += 100
[this.position.x > 1000]
this.target.x -= 100
[this.position.y > 600]
this.target.x -= 100
[ball in goal of Team 1]
add a point for Team 2
[ball in goal of Team 2]
add a point for Team 1
find the DOM-element that displays the score and change it
return ball to starting position
alert for the goal
find the DOM-element that displays the score and change it
return ball to starting position
alert for the goal