Spielanleitung:

Zu Beginn des Spiels kann der User einige Voreinstellungen auswählen. Den Teams kann jeweils durch Schreiben auf der Tastatur einen Namen gegeben werden. Außerdem können noch die Trikotfarben der Spieler, sowie die Shirtfarbe des Schiedsrichters und der Assistenten über Color-Picker gewählt werden. Im nächsten Schritt werden die Min- und Max-Werte der Geschwindigkeit und Präzision über einen Slider eingestellt und dann kann das Spiel gestartet werden.

Jedes Team besitzt 11 aktive Spieler und 6 Auswechselspieler. Die Auswechselspieler können per Drag & Drop (linke Maustaste & Shift gedrückt halten) mit einem aktiven Spieler ausgewechselt werden. Um die Spielerinformationen, der Name, Teamnummer, Geschwindigkeit und Präzision, anzeigen zu lassen, klicke bei gedrückter Alt-Taste auf einen aktiven Spieler. Des Weiteren wird der Spielstand, die Gegner und der Ballbesitzer angezeigt.

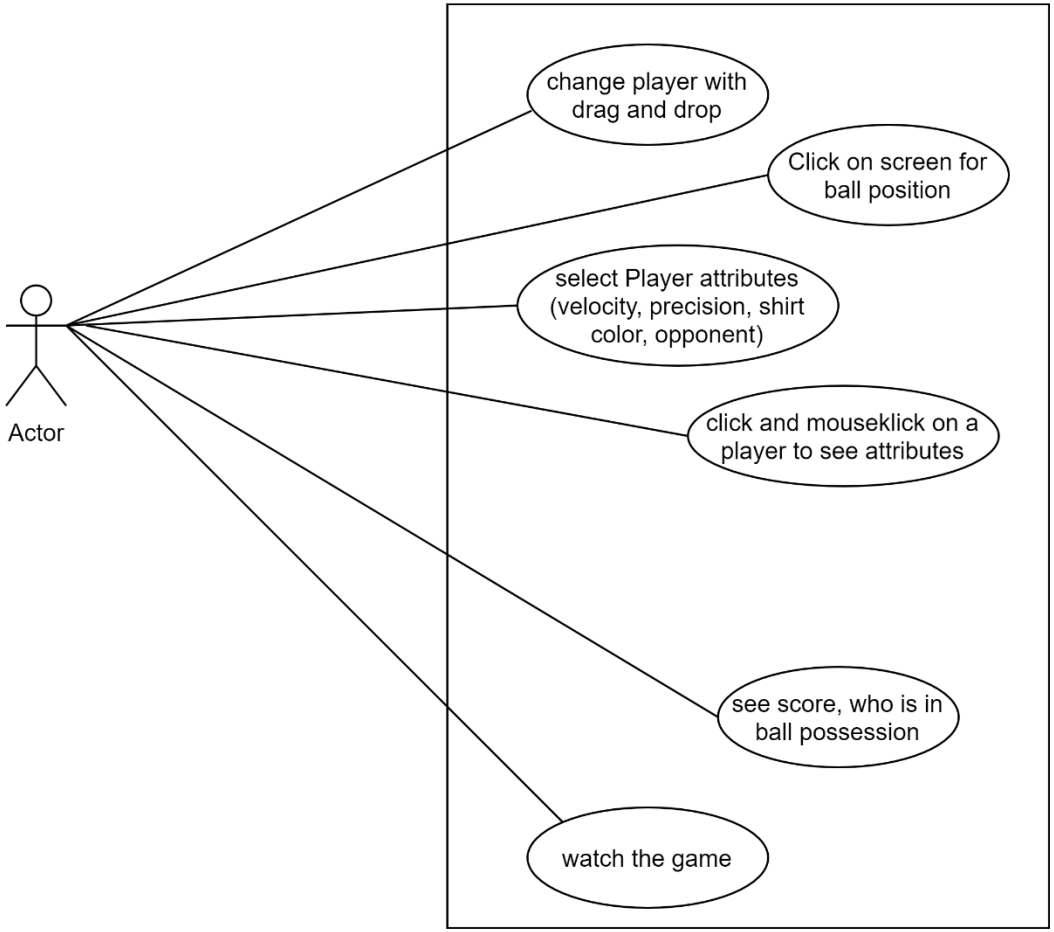
Um das Spiel zu starten, klicke auf die Leertaste. Nun kannst du durch Mausklick auf das Spielfeld eine neue Ballposition wählen. Die Spieler laufen selbstständig zum Ball. Befindet sich ein Spieler am Ball so stoppt die Animation und du kannst eine neue Ballposition wählen. Wird ein Tor erzielt, so ändert sich der Spielstand und die Spieler gehen zur Ausgangsposition zurück.

Anleitung zur Installation:

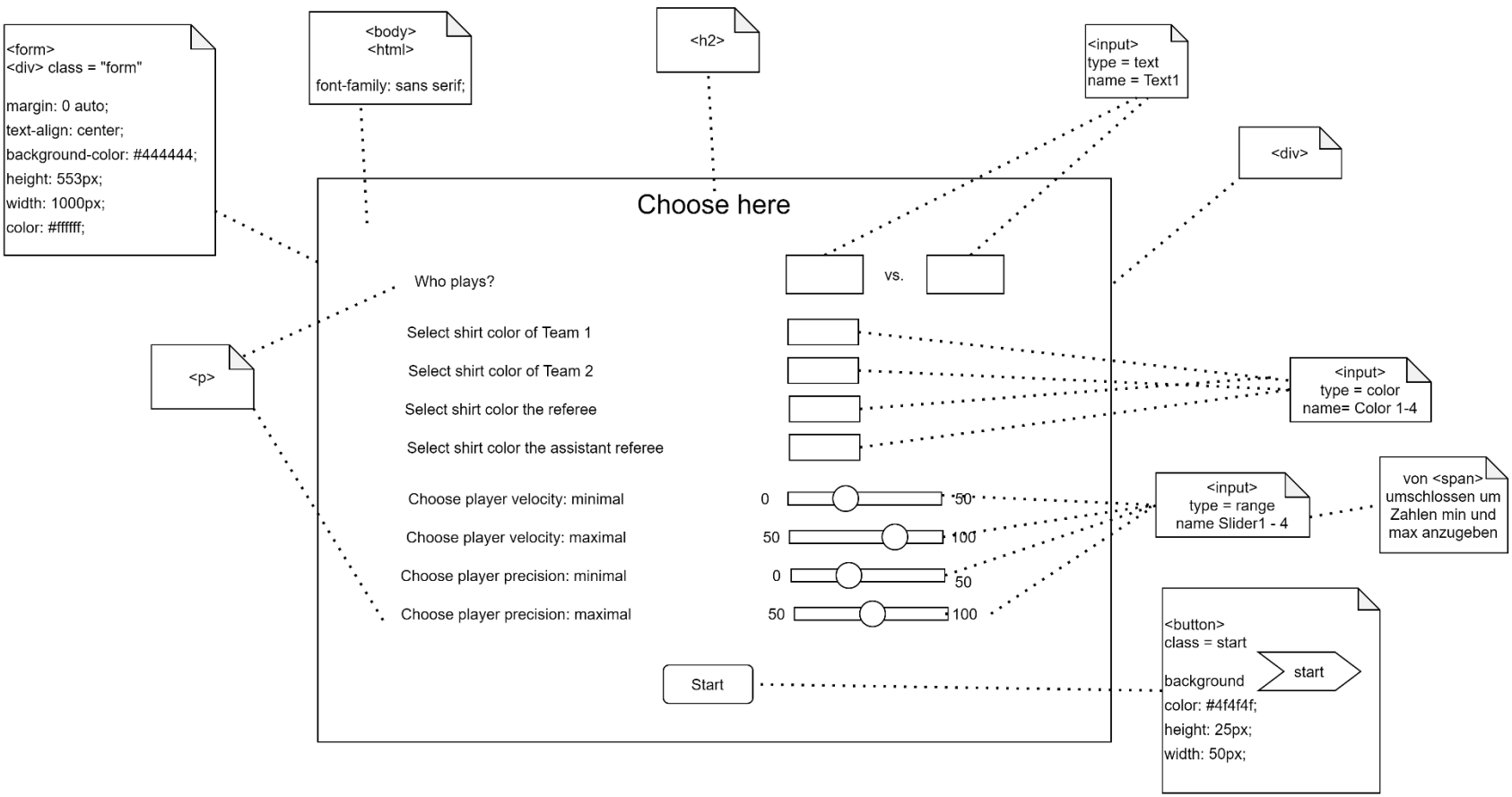
Laden Sie zunächst den ZIP-Ordner herunter und entpacke diesen mit der rechten Maustaste und der Option „Extrahieren“ bzw. „Entpacken“.

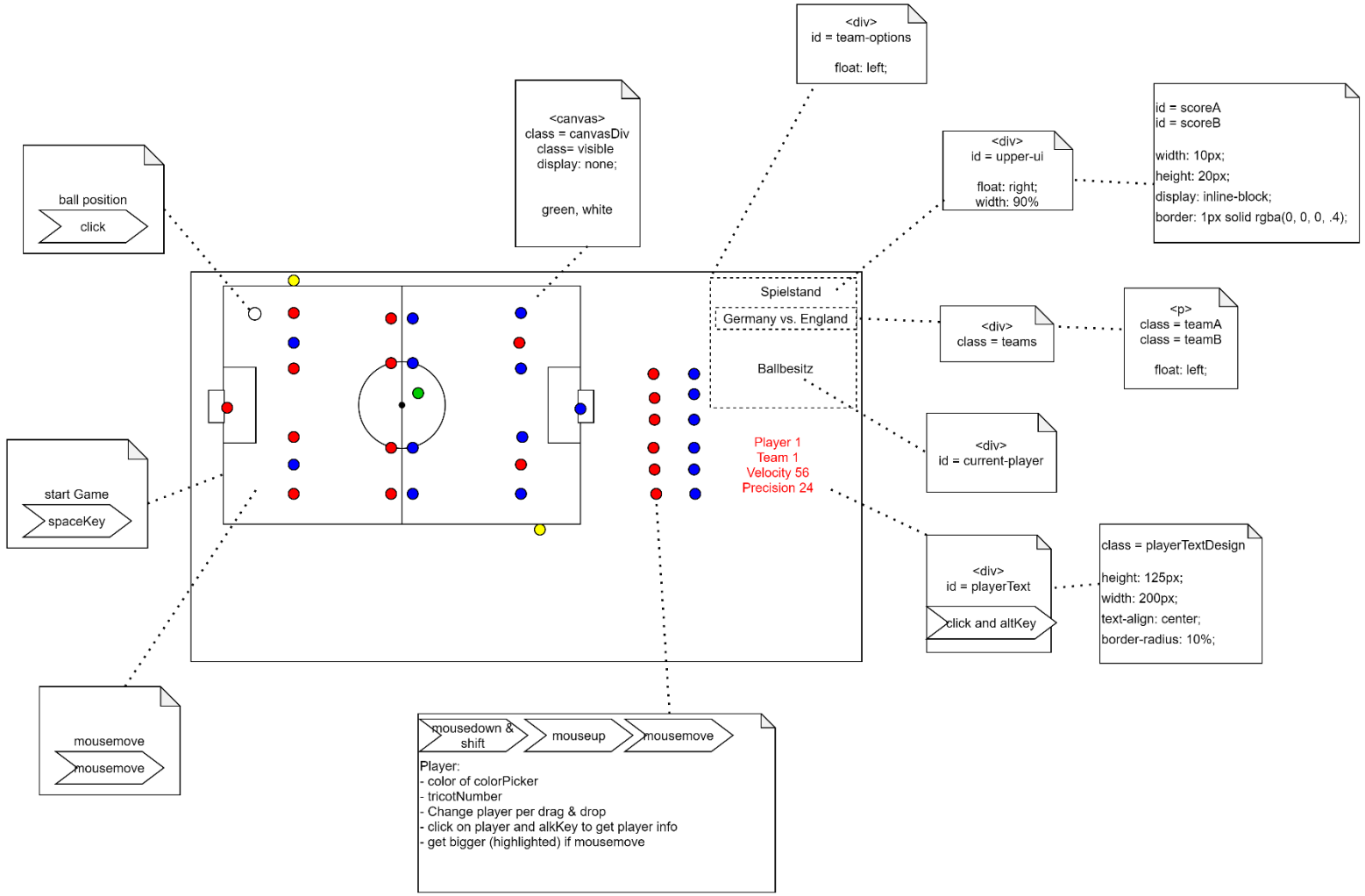
Öffnen Sie die HTML-Datei im Browser oder über den Live-Server in VS-Code. Um den Code zu sehen, öffnen Sie den Ordner in VS-Code.

SoccerGame: Use-Case-Diagram



SoccerGame: UI-Scribble





id = team-options
float: left;
<canvas>
class = canvasDiv
class= visible
display: none;
green, white

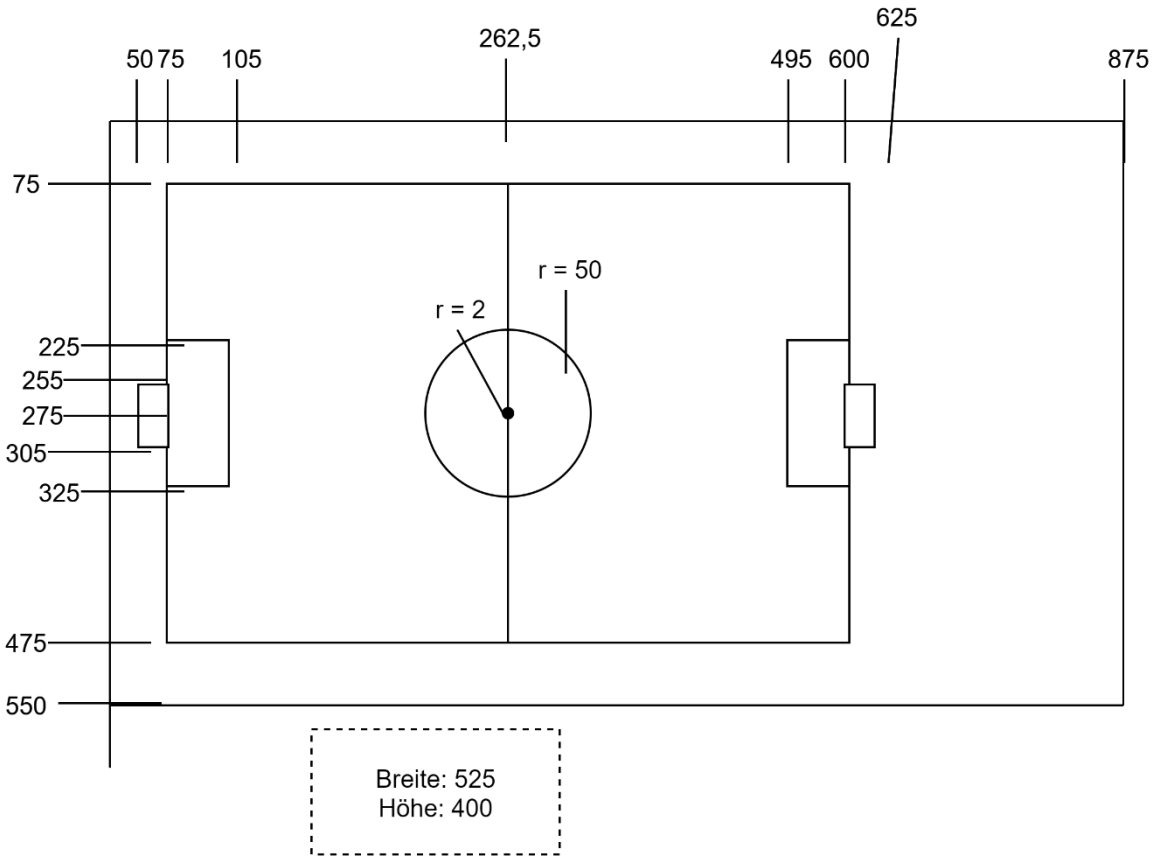
id = upper-ui
float: right;
width: 90%
id = scoreA
id = scoreB
width: 10px;
height: 20px;
display: inline-block;
border: 1px solid rgba(0, 0, 0, .4);
ball position
click
Spielstand
Germany vs. England
Ballbesitz

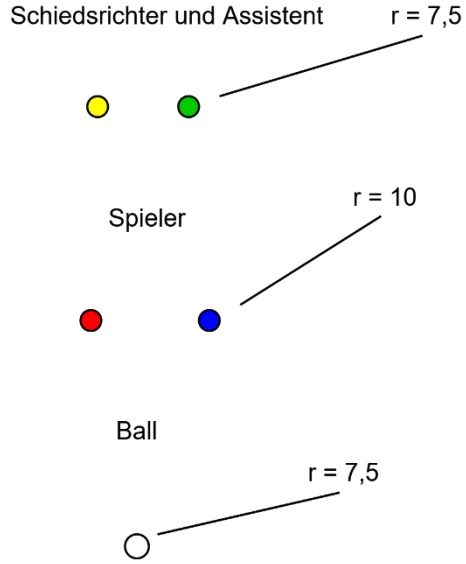
class = teams
<p>
class = teamA
class = teamB
float: left;
start Game
spaceKey
Player 1
Team 1
Velocity 56
Precision 24

id = current-player

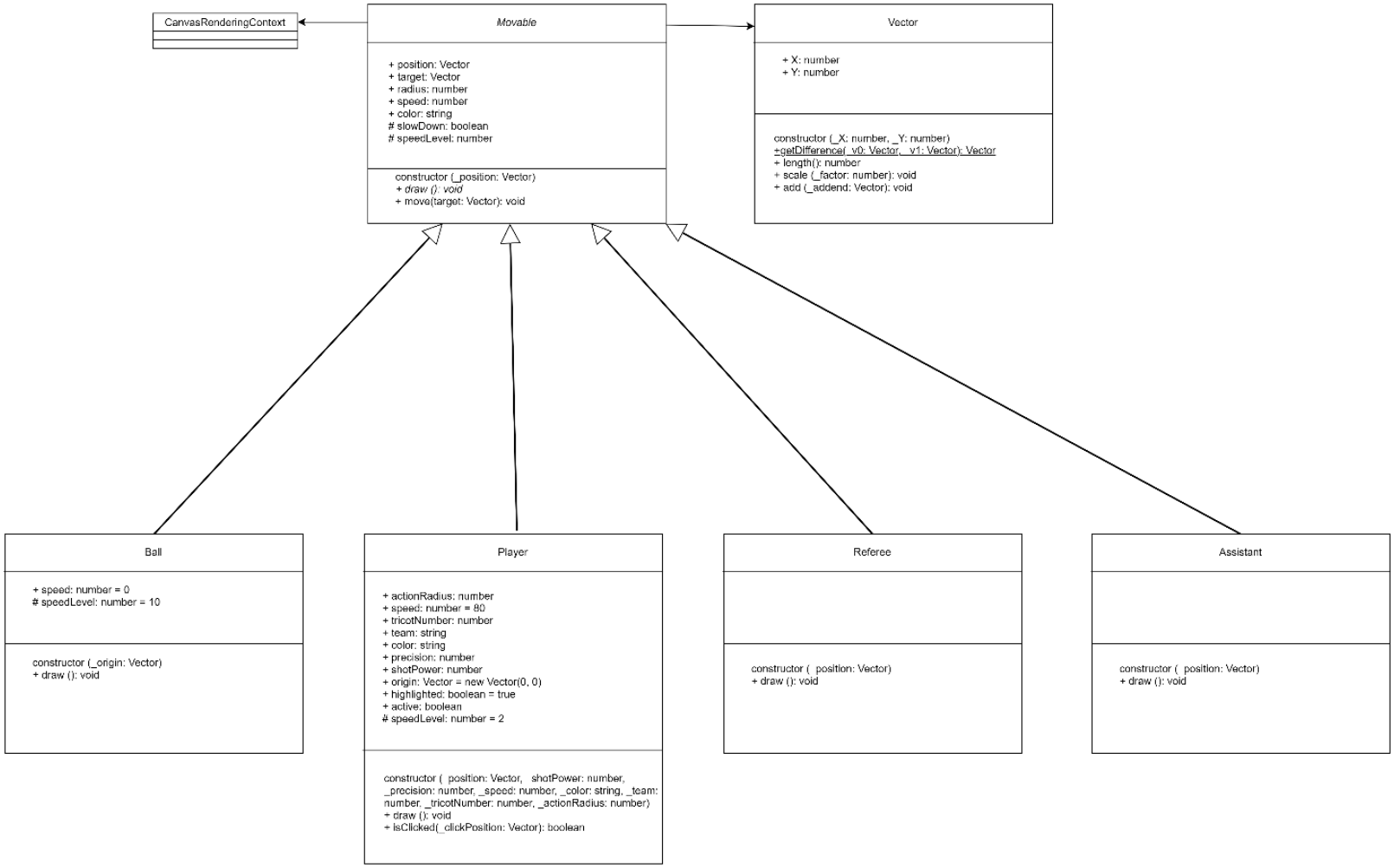
id = playerText
click and altKey
class = playerTextDesign
height: 125px;
width: 200px;
text-align: center;
border-radius: 10%;
mousemove
mousemove
mousedown & shift
mouseup
mousemove
Player:
- color of colorPicker
- tricotNumber
- Change player per drag & drop
- click on player and alkKey to get player info
- get bigger (highlighted) if mousemove

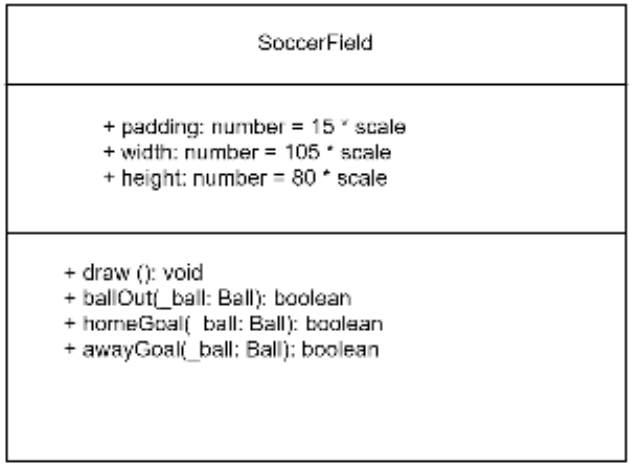
Spielfeld


50
75
105
262,5
495
600
625
875
75
225
255
275
305
325
475
550
r = 50
r = 2
Breite: 525
Höhe: 400

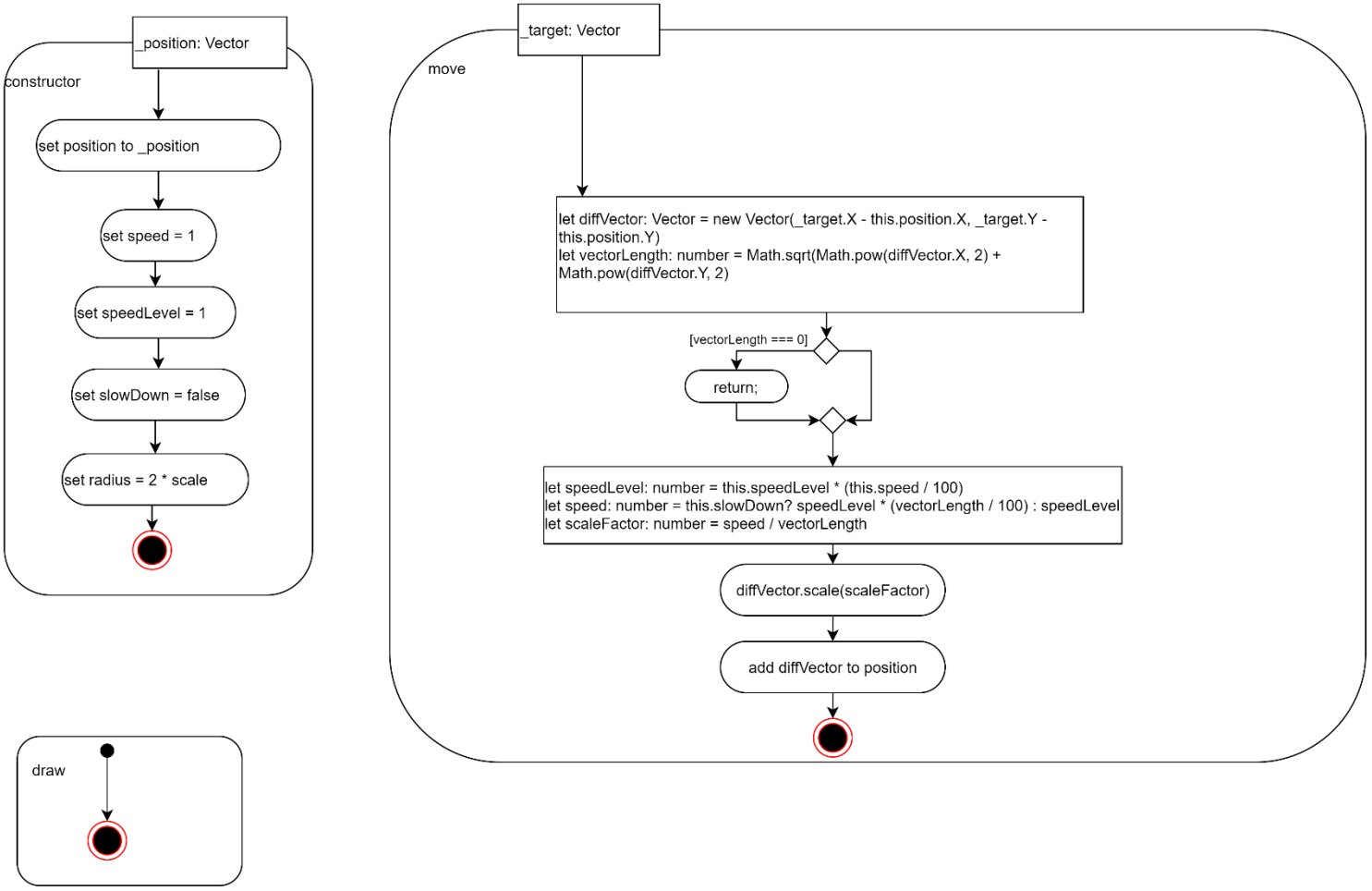

Schiedsrichter und Assistent
r = 7,5
Spieler
r = 10
Ball
r = 7,5

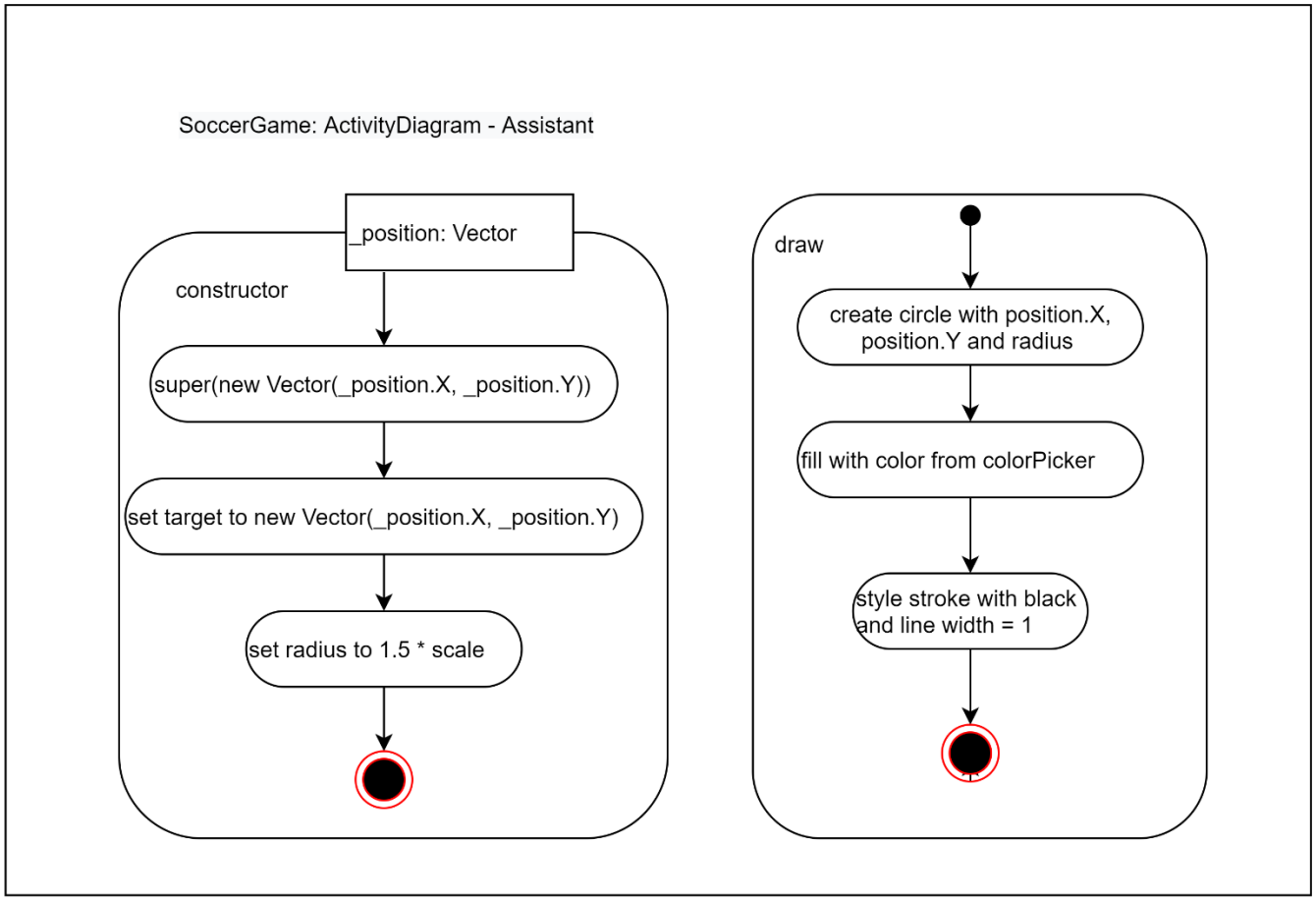
SoccerGame: Class Diagram



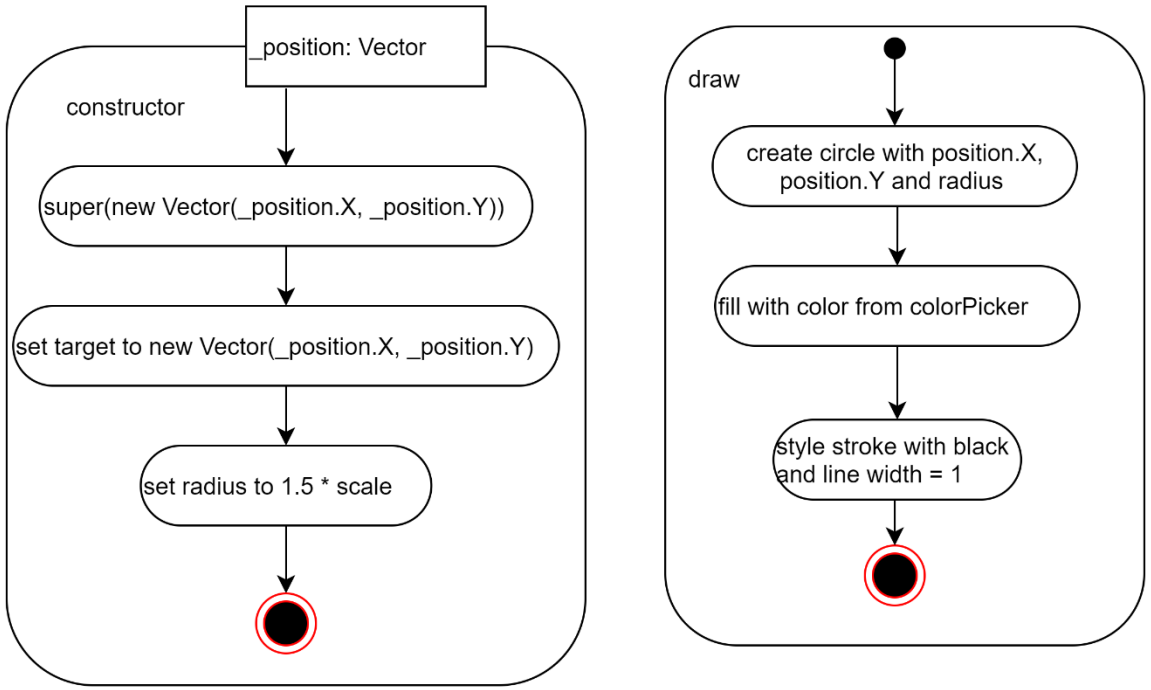

SoccerField
+ padding: number = 15 * scale
+ width: number = 105 * scale
+ height: number = 80 * scale
+ draw (): void
+ ballOut(_ball: Ball): boolean
+ homeGoal(_ball: Ball): boolean
+ awayGoal(_ball: Ball): boolean

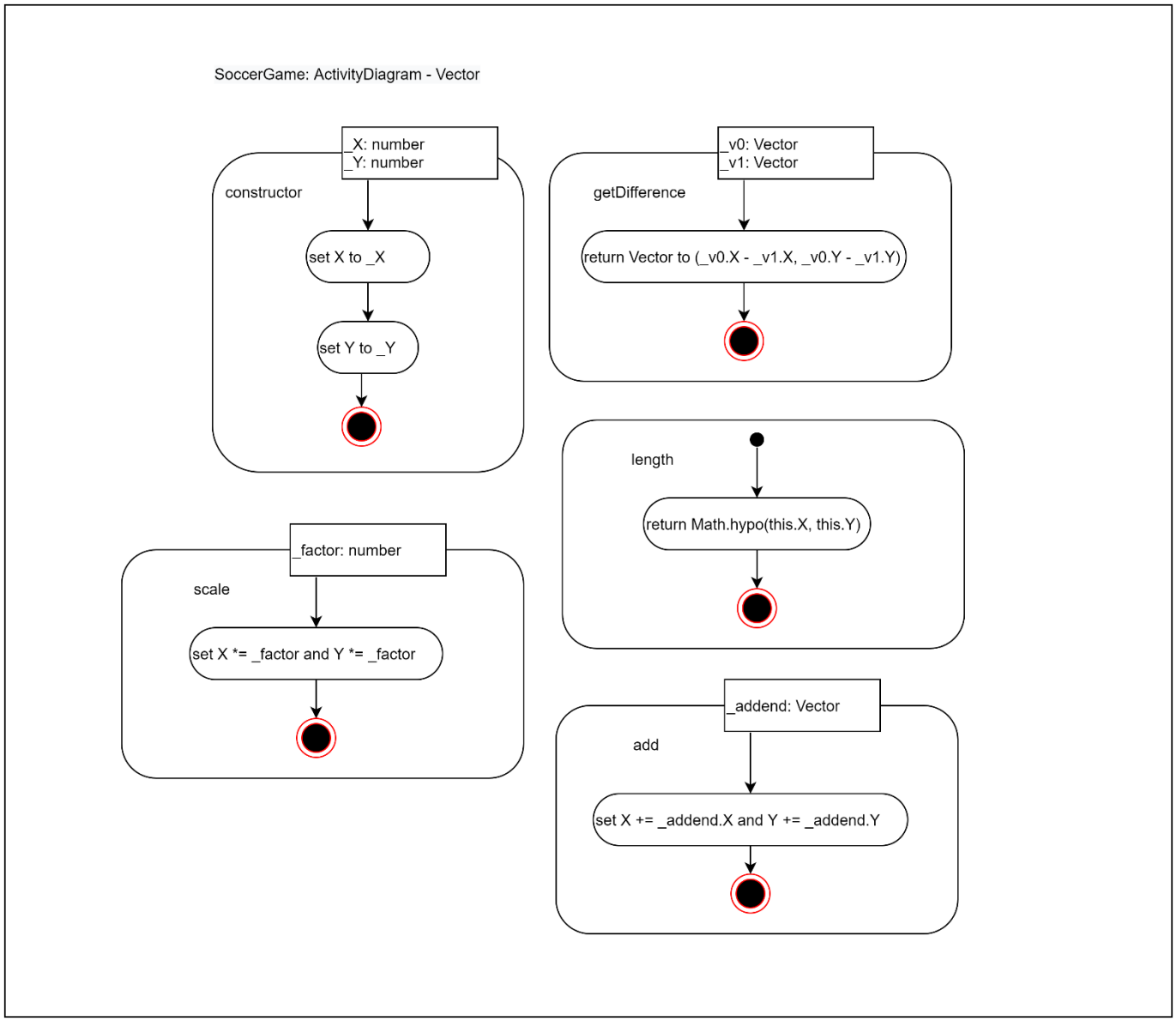
SoccerGame: ActivityDiagram - Movable

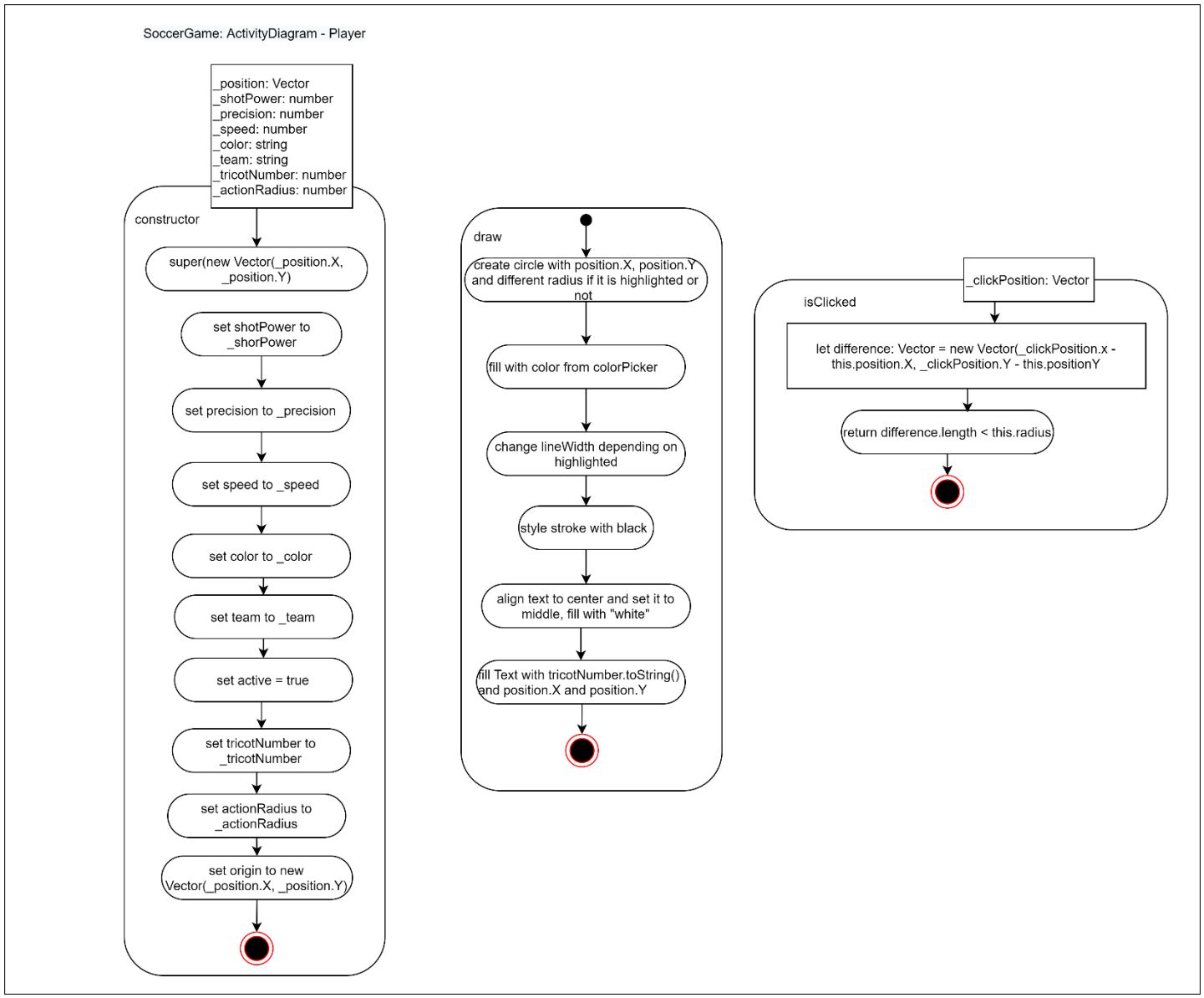


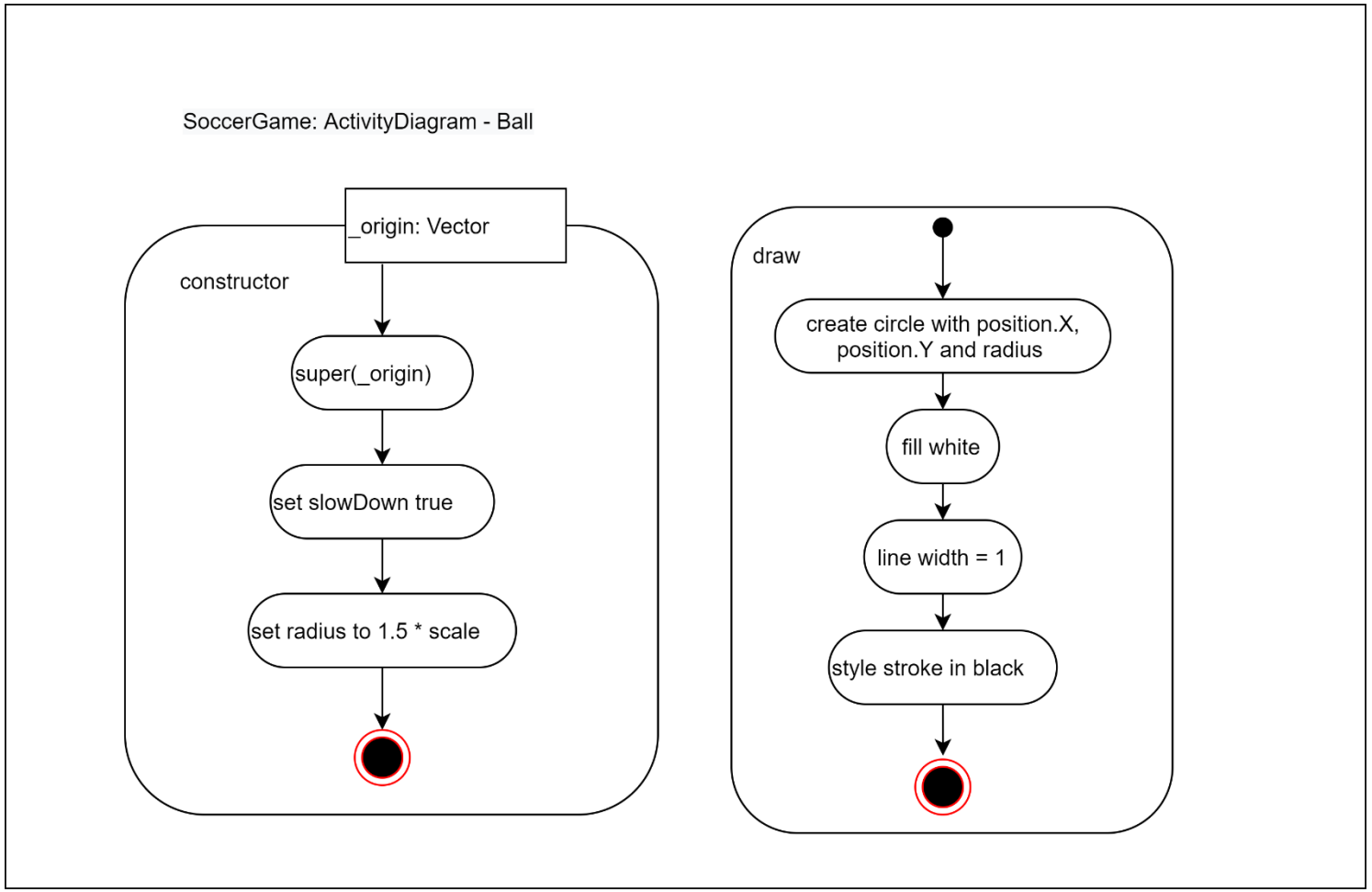

SoccerGame: ActivityDiagram - Assistant
_position: Vector
constructor
super(new Vector(_position.X, _position.Y))
set target to new Vector(_position.X, _position.Y)
set radius to 1.5 * scale
draw
create circle with position.X, position.Y and radius
fill with color from colorPicker
style stroke with black and line width = 1

SoccerGame: ActivityDiagram - Referee

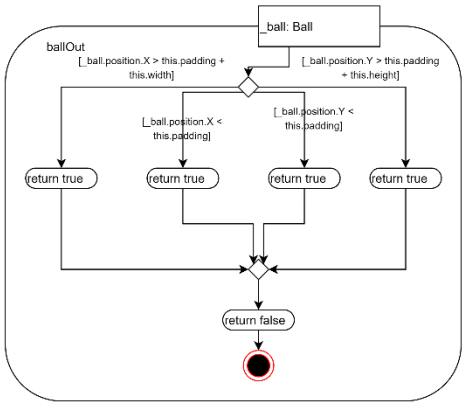


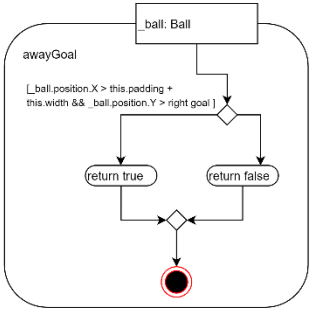

SoccerGame: ActivityDiagram - Vector
_X: number
_Y: number
constructor
set X to _X
set Y to _Y
_v0: Vector
_v1: Vector
getDifference
return Vector to (_v0.X - _v1.X, _v0.Y - _v1.Y)
length
return Math.hypo(this.X, this.Y)
_factor: number
scale
set X *= _factor and Y *= _factor
_addend: Vector
add
set X += _addend.X and Y += _addend.Y

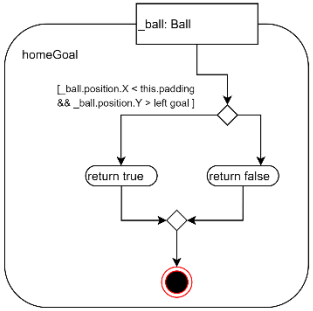

SoccerGame: ActivityDiagram - Player
_position: Vector
_shotPower: number
_precision: number
_speed: number
_color: string
_team: string
_tricotNumber: number
_actionRadius: number
constructor
super(new Vector(_position.X, _position.Y)
set shotPower to _shorPower
set precision to _precision
set speed to _speed
set color to _color
set team to _team
set active = true
set tricotNumber to _tricotNumber
set actionRadius to _actionRadius
set origin to new Vector(_position.X, _position.Y)
draw
create circle with position.X, position.Y and different radius if it is highlighted or not
fill with color from colorPicker
change lineWidth depending on highlighted
style stroke with black
align text to center and set it to middle, fill with "white"
fill Text with tricotNumber.toString() and position.X and position.Y
isClicked
_clickPosition: Vector
let difference: Vector = new Vector(_clickPosition.x - this.position.X, _clickPosition.Y - this.positionY
return difference.length < this.radius

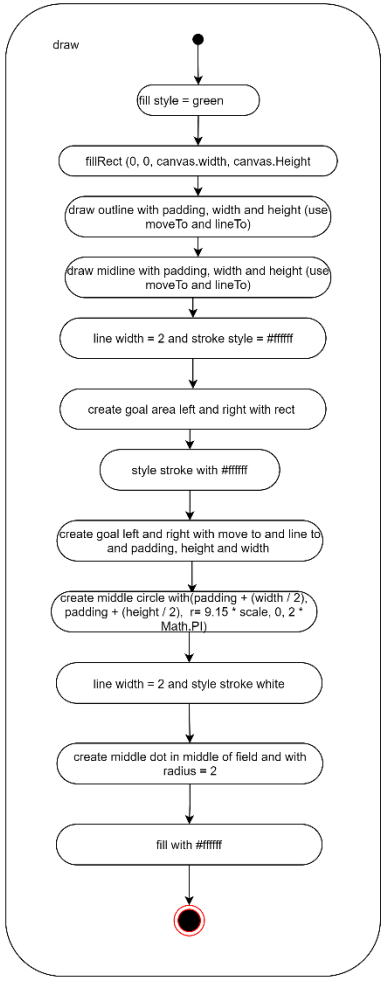

SoccerGame: ActivityDiagram - Ball
_origin: Vector
constructor
super(_origin)
set slowDown true
set radius to 1.5 * scale
draw
create circle with position.X, position.Y and radius
fill white
line width = 1
style stroke in black

SoccerGame: ActivityDiagram - SoccerField

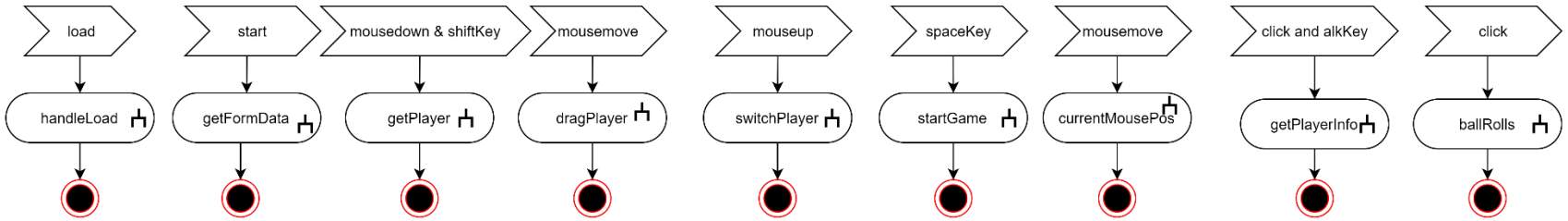


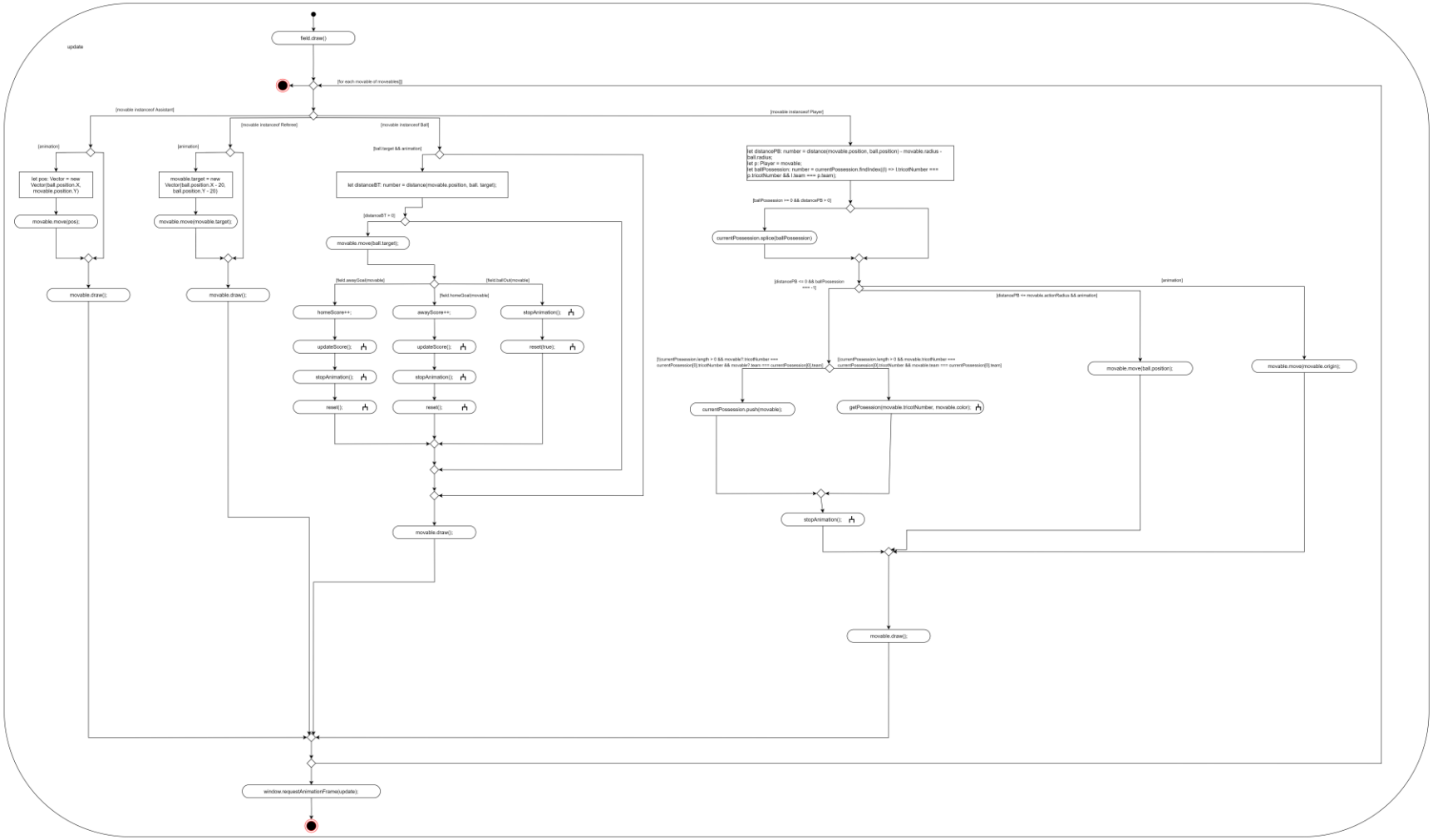


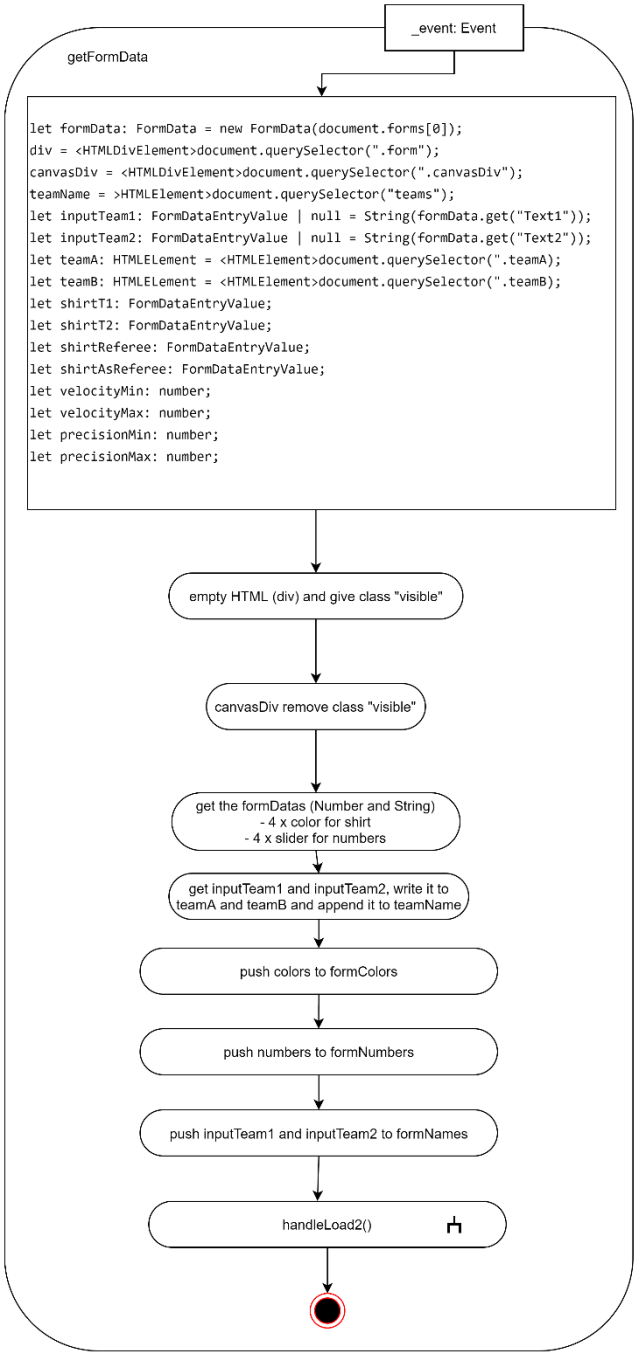




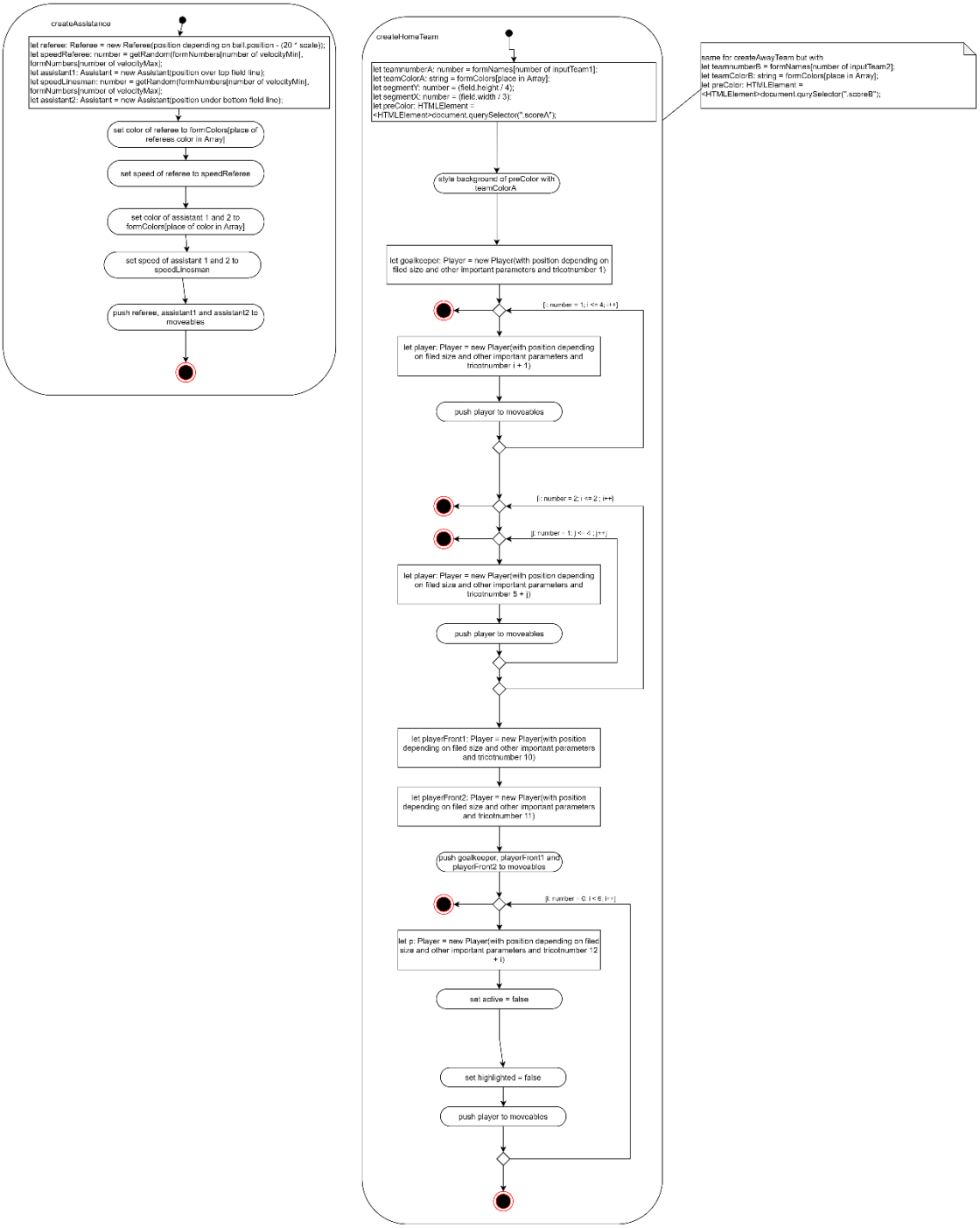
SoccerGame: ActivityDiagram

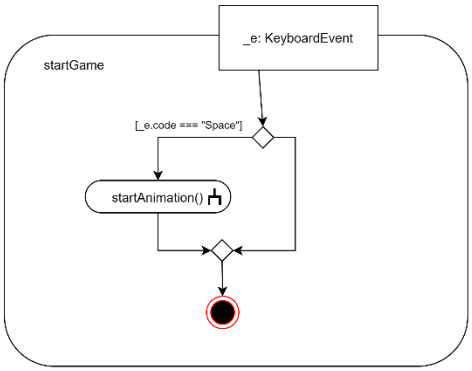


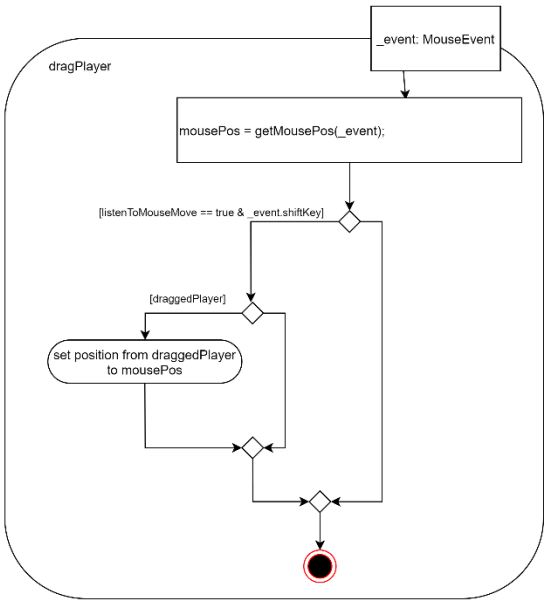

update
field.draw()
[for each movable of moveables[]]
[movable instanceof Assistant]
[movable instanceof Referee]
[movable instanceof Ball]
[movable instanceof Player]
[animation]
let pos: Vector = new Vector(ball.position.X, movable.position.Y)
movable.move(pos);
movable.draw();
[animation]
movable.target = new Vector(ball.position.X - 20, ball.position.Y - 20)
movable.move(movable.target);
movable.draw();
[ball.target && animation]
let distanceBT: number = distance(movable.position, ball. target);
[distanceBT > 0]
movable.move(ball.target);
[field.awayGoal(movable]
[field.homeGoal(movable]
[field.ballOut(movable]
homeScore++;
updateScore();
stopAnimation();
reset();
awayScore++;
updateScore();
stopAnimation();
reset();
stopAnimation();
reset(true);
movable.draw();
let distancePB: number = distance(movable.position, ball.position) - movable.radius - ball.radius;
let p: Player = movable;
let ballPossession: number = currentPossession.findIndex((l) => l.tricotNumber === p.tricotNumber && l.team === p.team);
[ballPossession >= 0 && distancePB > 0]
currentPossession.splice(ballPossession)
[distancePB <= 0 && ballPossession === -1]
[distancePB <= movable.actionRadius && animation]
[animation]
[!(currentPossession.length > 0 && movable?.tricotNumber === currentPossession[0].tricotNumber && movable?.team === currentPossession[0].team)]
[(currentPossession.length > 0 && movable.tricotNumber === currentPossession[0].tricotNumber && movable.team === currentPossession[0].team)]
currentPossession.push(movable);
getPossesion(movable.tricotNumber, movable.color);
movable.move(ball.position);
movable.move(movable.origin);
stopAnimation();
movable.draw();
window.requestAnimationFrame(update);

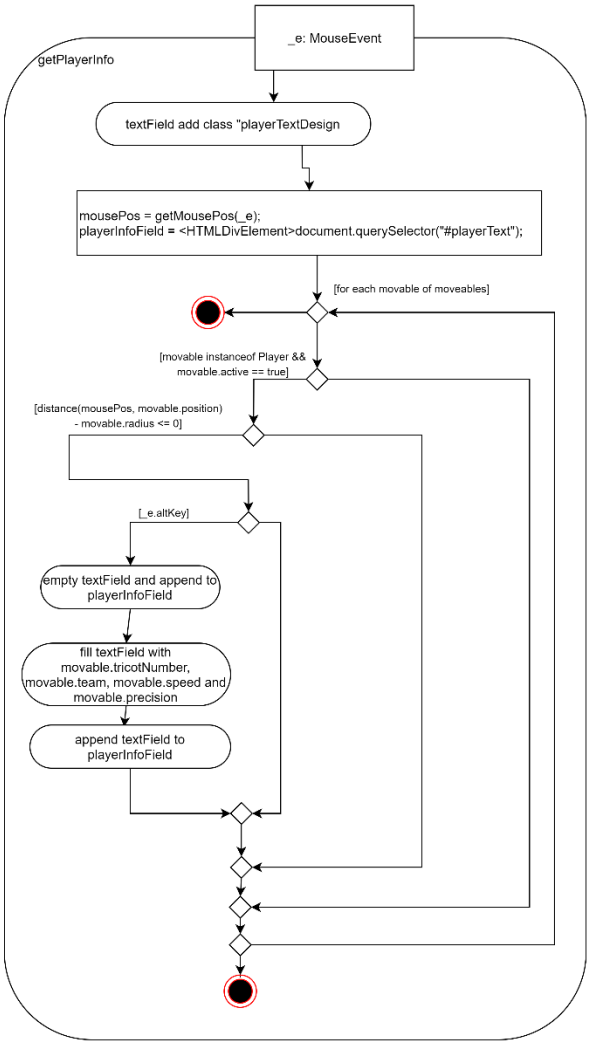

_event: Event
getFormData
let formData: FormData = new FormData(document.forms[0]);
div = <HTMLDivElement>document.querySelector(".form");
canvasDiv = <HTMLDivElement>document.querySelector(".canvasDiv");
teamName = >HTMLElement>document.querySelector("teams");
let inputTeam1: FormDataEntryValue | null = String(formData.get("Text1"));
let inputTeam2: FormDataEntryValue | null = String(formData.get("Text2"));
let teamA: HTMLELement = <HTMLElement>document.querySelector(".teamA);
let teamB: HTMLELement = <HTMLElement>document.querySelector(".teamB);
let shirtT1: FormDataEntryValue;
let shirtT2: FormDataEntryValue;
let shirtReferee: FormDataEntryValue;
let shirtAsReferee: FormDataEntryValue;
let velocityMin: number;
let velocityMax: number;
let precisionMin: number;
let precisionMax: number;
empty HTML (div) and give class "visible"
canvasDiv remove class "visible"
get the formDatas (Number and String)
- 4 x color for shirt
- 4 x slider for numbers
get inputTeam1 and inputTeam2, write it to teamA and teamB and append it to teamName
push colors to formColors
push numbers to formNumbers
push inputTeam1 and inputTeam2 to formNames
handleLoad2()

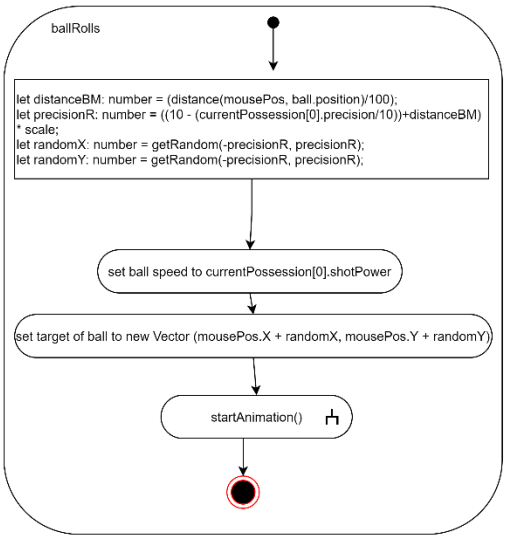

export let crc2: CanvasRenderingContext2D;
export let canvas: HTMLCanvasElement;
export let scale: number = 5;
let div: HTMLDivElement;
let canvasDiv: HTMLDivElement;
let currentPlayer: HTMLDivElement;
let teamName: HTMLElement;
let divPlayer: HTMLElement;
divPlayer = <HTMLElement>document. createElement("p");
let textField: HTMLElement;
textField = document.createElement("div");
let playerInfoField: HTMLDivElement;
let scoreElement: HTMLSpanElement =
<HTMLSpanElement>document.getElementById("score");
scoreElement = document.createElement("span");
let scoreTeamA: HTMLSpanElement = document.createElement("span");
let scoreTeamB: HTMLSpanElement = document.createElement("span");
let homeScore: number = 0;
let awayScore: number = 0;
let moveables: Moveable[] = [];
let currentPossession: Player[] = [];
let formColors: string[] = [];
let formNumbers: number[] = [];
let formNames: string[] = [];
let draggedPlayer: Player | null;
let field: SoccerField;
let ball: Ball;
let mousePos: Vector;
let listenToMouseMove: boolean = false;
let animation: boolean = false;
install load listener on windows

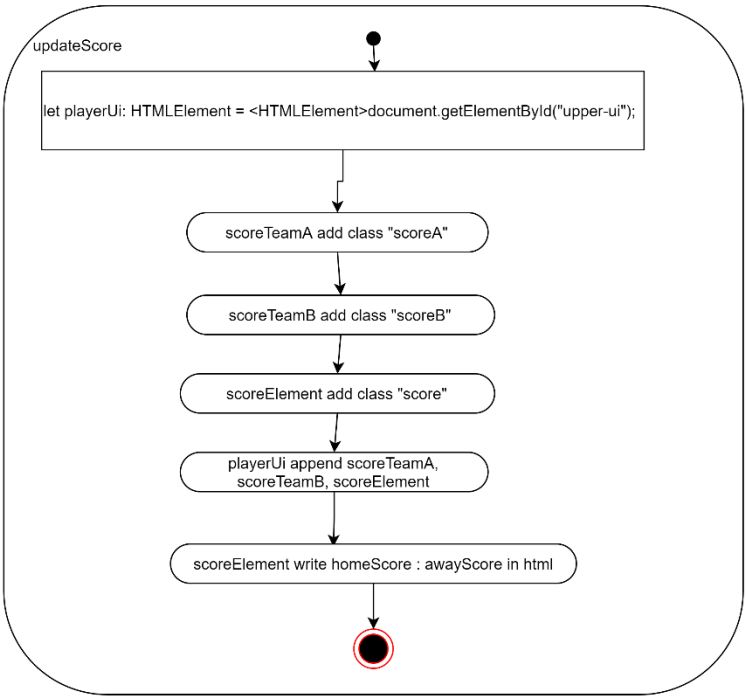

createAssistance
let referee: Referee = new Referee(position depending on ball.position - (20 * scale));
let speedReferee: number = getRandom(formNumbers[number of velocityMin], formNumbers[number of velocityMax]);
let assistant1: Assistant = new Assistant(position over top field line);
let speedLinesman: number = getRandom(formNumbers[number of velocityMin], formNumbers[number of velocityMax]);
let assistant2: Assistant = new Assistant(position under bottom field line);
set color of referee to formColors[place of referees color in Array]
set speed of referee to speedReferee
set color of assistant 1 and 2 to formColors[place of color in Array]
set speed of assistant 1 and 2 to speedLinesman
push referee, assistant1 and assistant2 to moveables
createHomeTeam
let teamnumberA: number = formNames[number of inputTeam1];
let teamColorA: string = formColors[place in Array];
let segmentY: number = (field.height / 4);
let segmentX: number = (field.width / 3);
let preColor: HTMLElement = <HTMLElement>document.querySelector(".scoreA");
same for createAwayTeam but with
let teamnumberB = formNames[number of inputTeam2];
let teamColorB: string = formColors[place in Array];
let preColor: HTMLElement = <HTMLElement>document.qurySelector(".scoreB");
style background of preColor with teamColorA
let goalkeeper: Player = new Player(with position depending on filed size and other important parameters and tricotnumber 1)
[i: number = 1; i <= 4; i++]
let player: Player = new Player(with position depending on filed size and other important parameters and tricotnumber i + 1)
push player to moveables
[i: number = 2; i <= 2 ; i++]
[j: number = 1; j <= 4 ; j++]
let player: Player = new Player(with position depending on filed size and other important parameters and tricolnumber 5 + j)
push player to moveables
let playerFront1: Player = new Player(with position depending on filed size and other important parameters and tricotnumber 10)
let playerFront2: Player = new Player(with position depending on filed size and other important parameters and tricotnumber 11)
push goalkeeper, playerFront1 and playerFront2 to moveables
[i: number = 0; i < 6; i++]
let p: Player = new Player(with position depending on filed size and other important parameters and tricolnumber 12 + i)
set active = false
set highlighted = false
push player to moveables

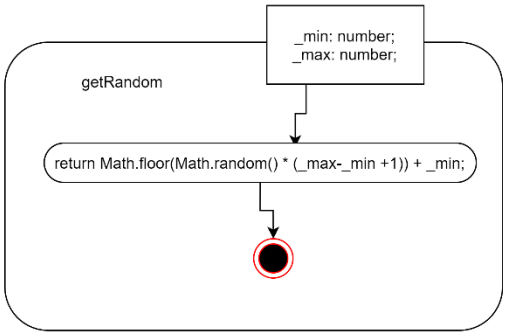

startGame
_e: KeyboardEvent
[_e.code === "Space"]
startAnimation()

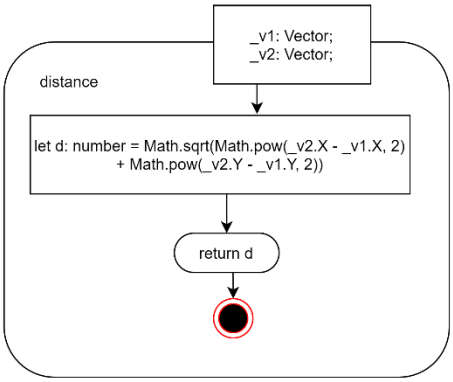

dragPlayer
_event: MouseEvent
mousePos = getMousePos(_event);
[listenToMouseMove == true & _event.shiftKey]
[draggedPlayer]
set position from draggedPlayer to mousePos

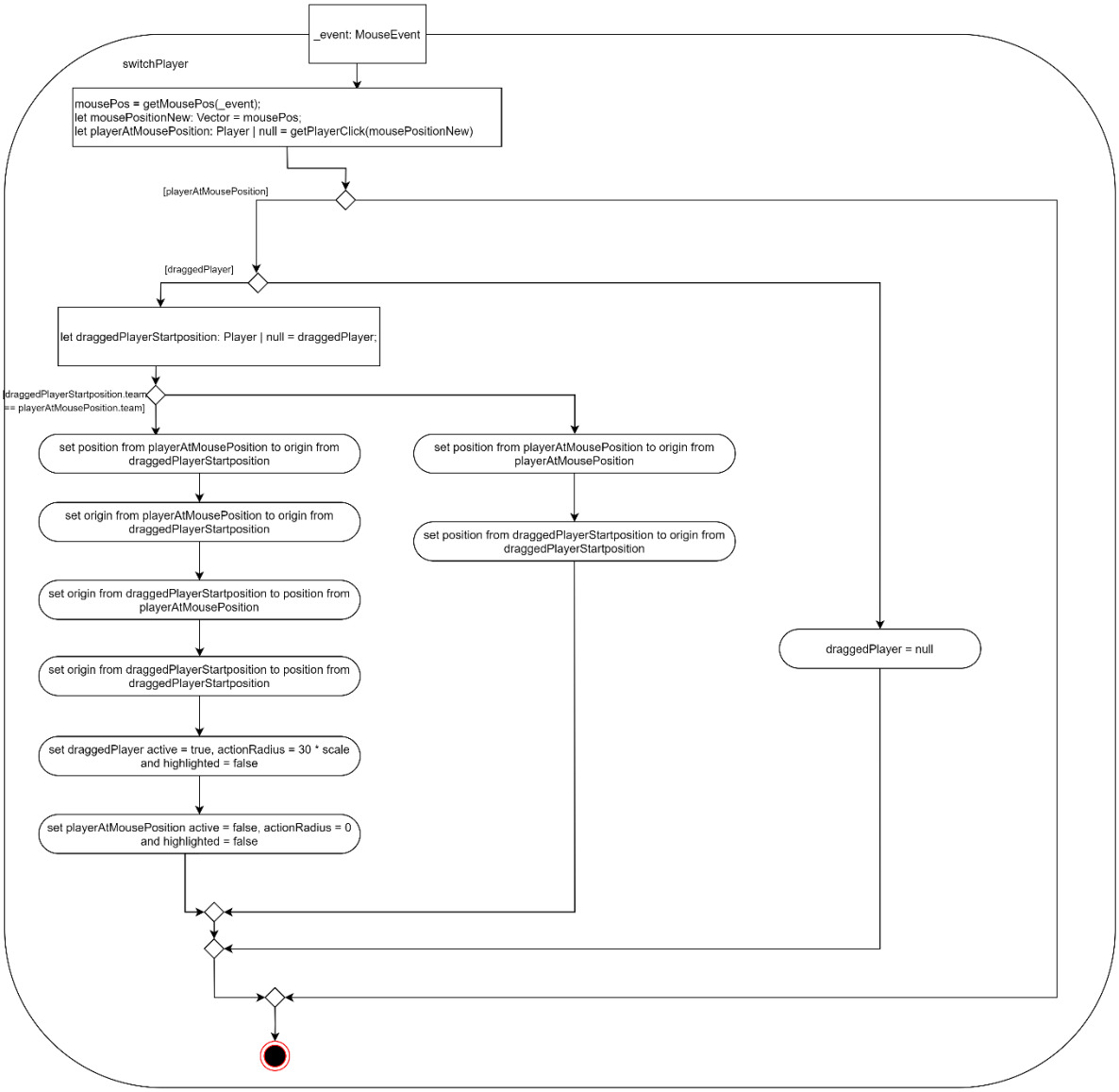

getPlayerInfo
_e: MouseEvent
textField add class "playerTextDesign
mousePos = getMousePos(_e);
playerInfoField = <HTMLDivElement>document.querySelector("#playerText");
[for each movable of moveables]
[movable instanceof Player && movable.active == true]
[distance(mousePos, movable.position) - movable.radius <= 0]
[_e.altKey]
empty textField and append to playerInfoField
fill textField with movable.tricotNumber, movable.team, movable.speed and movable.precision
append textField to playerInfoField

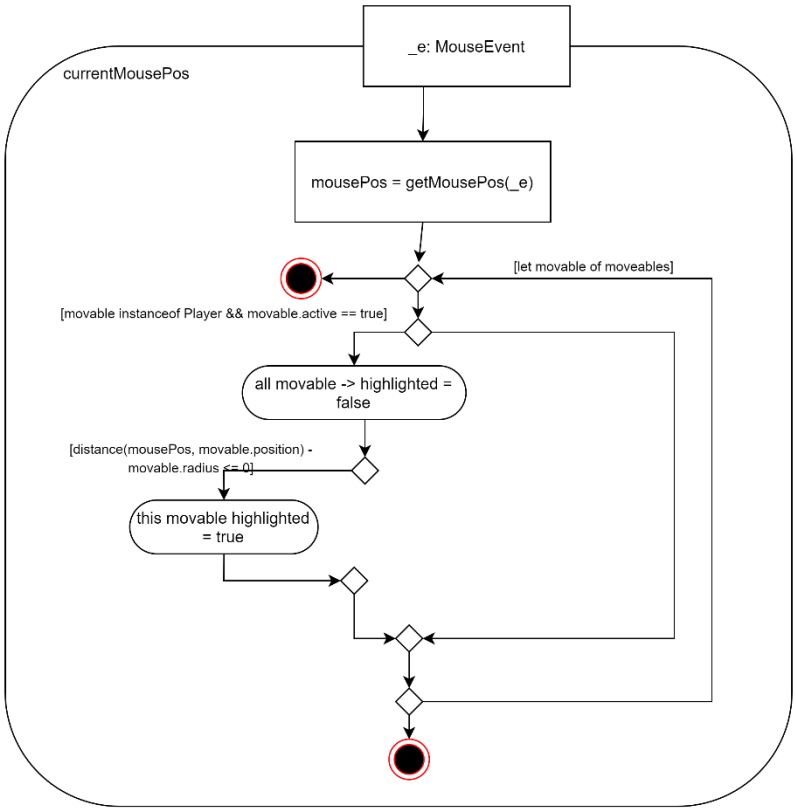

ballRolls
let distanceBM: number = (distance(mousePos, ball.position)/100);
let precisionR: number = ((10 - (currentPossession[0].precision/10))+distanceBM) * scale;
let randomX: number = getRandom(-precisionR, precisionR);
let randomY: number = getRandom(-precisionR, precisionR);
set ball speed to currentPossession[0].shotPower
set target of ball to new Vector (mousePos.X + randomX, mousePos.Y + randomY)
startAnimation()

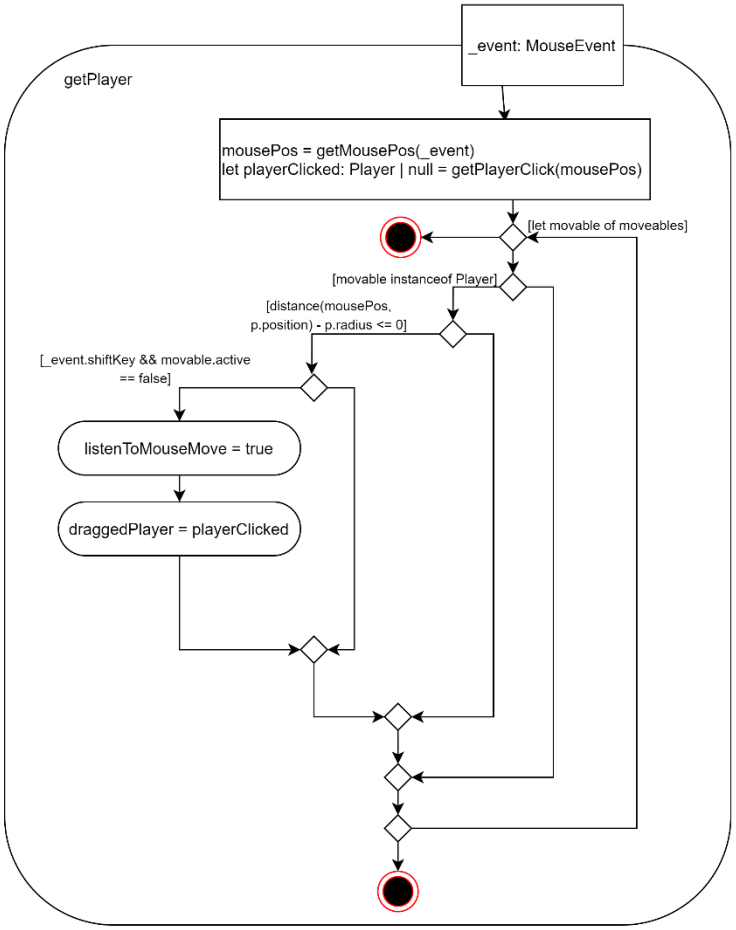

updateScore
let playerUi: HTMLElement = <HTMLElement>document.getElementById("upper-ui");
scoreTeamA add class "scoreA"
scoreTeamB add class "scoreB"
scoreElement add class "score"
playerUi append scoreTeamA, scoreTeamB, scoreElement
scoreElement write homeScore : awayScore in html

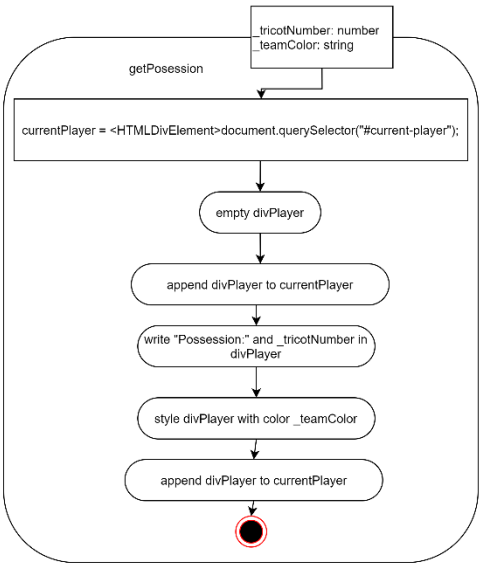

_min: number;
_max: number;
getRandom
return Math.floor(Math.random() * (_max-_min +1)) + _min;

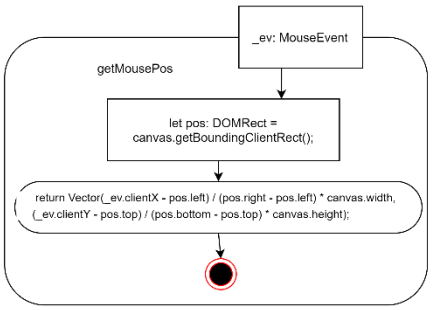

_v1: Vector;
_v2: Vector;
distance
let d: number = Math.sqrt(Math.pow(_v2.X - _v1.X, 2) + Math.pow(_v2.Y - _v1.Y, 2))
return d

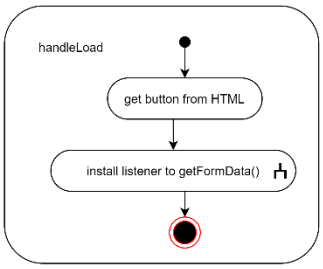

_event: MouseEvent
switchPlayer
mousePos = getMousePos(_event);
let mousePositionNew: Vector = mousePos;
let playerAtMousePosition: Player | null = getPlayerClick(mousePositionNew)
[playerAtMousePosition]
[draggedPlayer]
let draggedPlayerStartposition: Player | null = draggedPlayer;
[draggedPlayerStartposition.team == playerAtMousePosition.team]
set position from playerAtMousePosition to origin from draggedPlayerStartposition
set origin from playerAtMousePosition to origin from draggedPlayerStartposition
set origin from draggedPlayerStartposition to position from playerAtMousePosition
set origin from draggedPlayerStartposition to position from draggedPlayerStartposition
set draggedPlayer active = true, actionRadius = 30 * scale and highlighted = false
set playerAtMousePosition active = false, actionRadius = 0 and highlighted = false
set position from playerAtMousePosition to origin from playerAtMousePosition
set position from draggedPlayerStartposition to origin from draggedPlayerStartposition
draggedPlayer = null

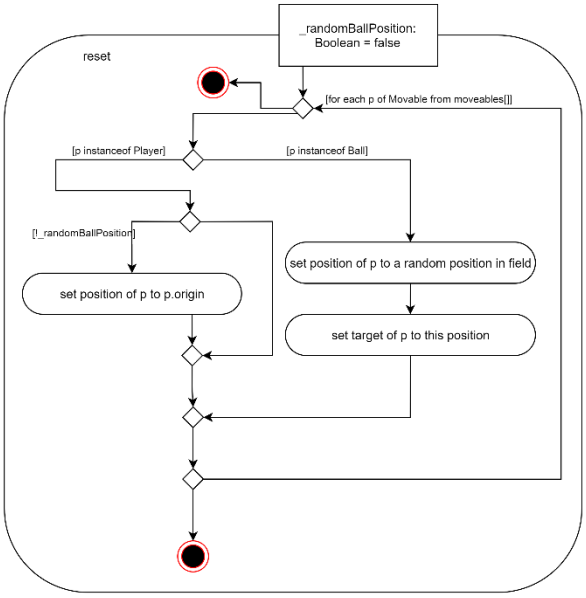

currentMousePos
_e: MouseEvent
mousePos = getMousePos(_e)
[let movable of moveables]
[movable instanceof Player && movable.active == true]
all movable -> highlighted = false
[distance(mousePos, movable.position) - movable.radius <= 0]
this movable highlighted = true

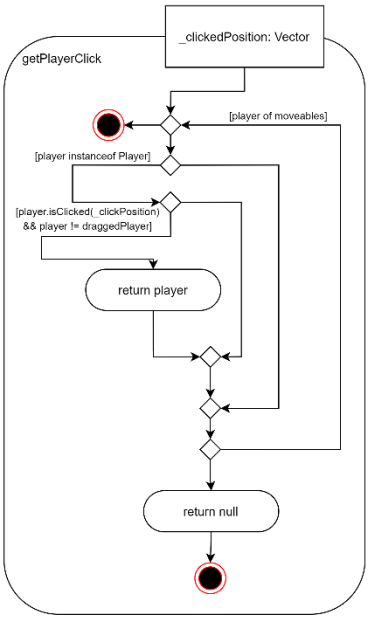

getPlayer
_event: MouseEvent
mousePos = getMousePos(_event)
let playerClicked: Player | null = getPlayerClick(mousePos)
[let movable of moveables]
[movable instanceof Player]
[distance(mousePos, p.position) - p.radius <= 0]
[_event.shiftKey && movable.active == false]
listenToMouseMove = true
draggedPlayer = playerClicked

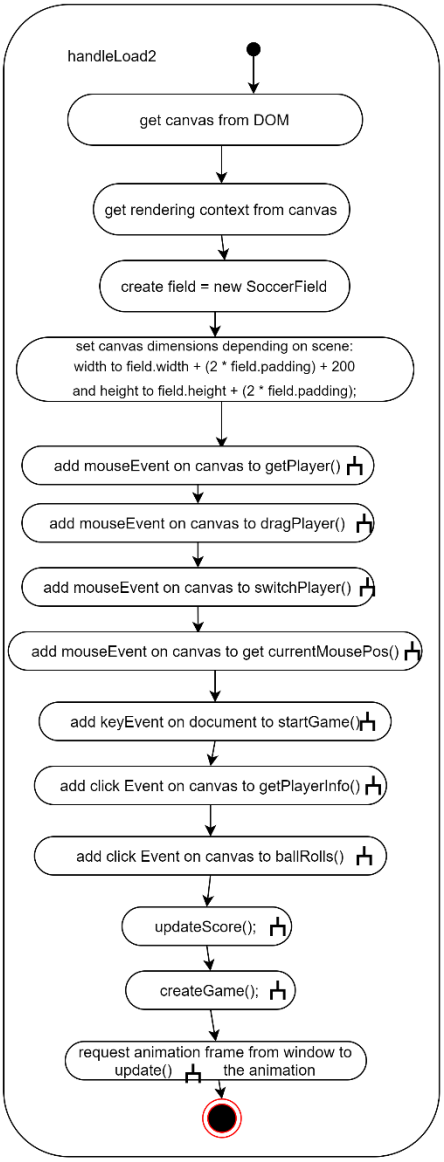

_tricotNumber: number
_teamColor: string
getPosession
currentPlayer = <HTMLDivElement>document.querySelector("#current-player");
empty divPlayer
append divPlayer to currentPlayer
write "Possession:" and _tricotNumber in divPlayer
style divPlayer with color _teamColor
append divPlayer to currentPlayer

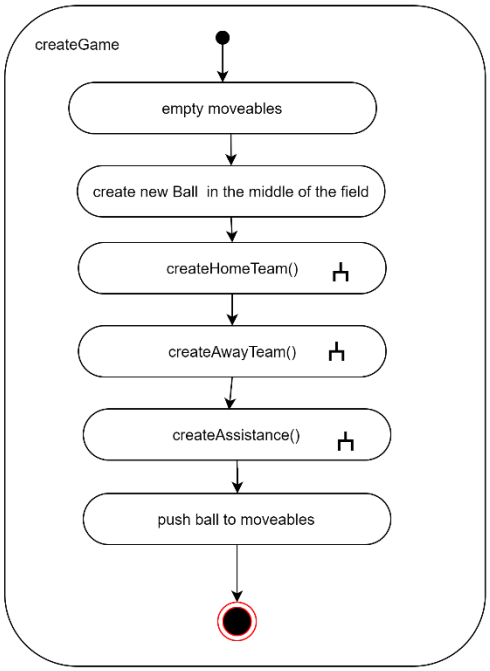

_ev: MouseEvent
getMousePos
let pos: DOMRect = canvas.getBoundingClientRect();
return Vector(_ev.clientX - pos.left) / (pos.right - pos.left) * canvas.width, (_ev.clientY - pos.top) / (pos.bottom - pos.top) * canvas.height);

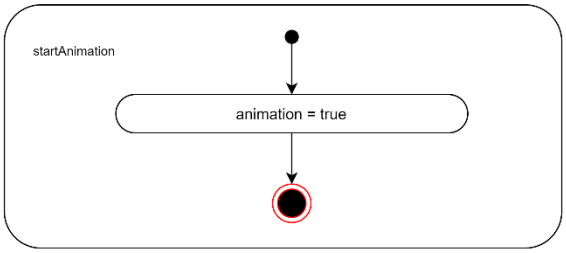

handleLoad
get button from HTML
install listener to getFormData()

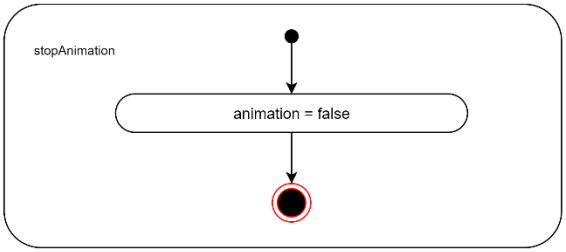

_randomBallPosition: Boolean = false
reset
[for each p of Movable from moveables[]]
[p instanceof Player]
[p instanceof Ball]
[!_randomBallPosition]
set position of p to p.origin
set position of p to a random position in field
set target of p to this position


_clickedPosition: Vector
getPlayerClick
[player of moveables]
[player instanceof Player]
[player.isClicked(_clickPosition) && player != draggedPlayer]
return player
return null


handleLoad2
get canvas from DOM
get rendering context from canvas
create field = new SoccerField
set canvas dimensions depending on scene:
width to field.width + (2 * field.padding) + 200
and height to field.height + (2 * field.padding);
add mouseEvent on canvas to getPlayer()
add mouseEvent on canvas to dragPlayer()
add mouseEvent on canvas to switchPlayer()
add mouseEvent on canvas to get currentMousePos()
add keyEvent on document to startGame()
add click Event on canvas to getPlayerInfo()
add click Event on canvas to ballRolls()
updateScore();
createGame();
request animation frame from window to update()
the animation


createGame
empty moveables
create new Ball in the middle of the field
createHomeTeam()
createAwayTeam()
createAssistance()
push ball to moveables


startAnimation
animation = true


stopAnimation
animation = false