Anleitung EIA2-Endabgabe

Willkommen beim Fußballsimulator 2021. Tauche als Trainer in die Welt des Fußballs ein und simuliere Spiele, um so neue Strategien zu entwickeln oder bereits geschehene Situationen besser analysieren zu können.

Primär muss die ZIP-Datei entpackt werden. Dafür sind Programme wie WinRar besonders hilfreich. Ist der Ordner entpackt, sind mehrere Ordner und Dateien drinnen. Mit Klick auf die HTML-Datei wird die Anwendung im Browser gestartet.

Zu sehen sind das Fußballfeld mit den Spielern und Schiedsrichtern. Oberhalb ist das Scoreboard sowie ein Startknopf, links und rechts sind Balken, um die Einstellungen zu ändern.

Um das Spiel zu starten, musst du auf einen gewünschten Punkt auf dem Spielfeld klicken, wo der Ball hingespielt werden soll. Spieler in der Nähe bewegen sich dann zum Ball. Der Ball bleibt stehen, sofern ein Spieler an ihn rangeht oder der Ball ausrollt. Nur bei ersterem Fall kann der Ball durch weiteres Klicken weitergespielt werden. Sind die Spieler nicht mehr „aufmerksam“ auf den Ball, kehren sie zurück zu ihrer Position. Erzielt man ein Tor, erscheint eine Benachrichtigung, welches Team das Tor erzielt hat. Mit Klick auf „Ok“ wird das Spiel wieder angepfiffen.

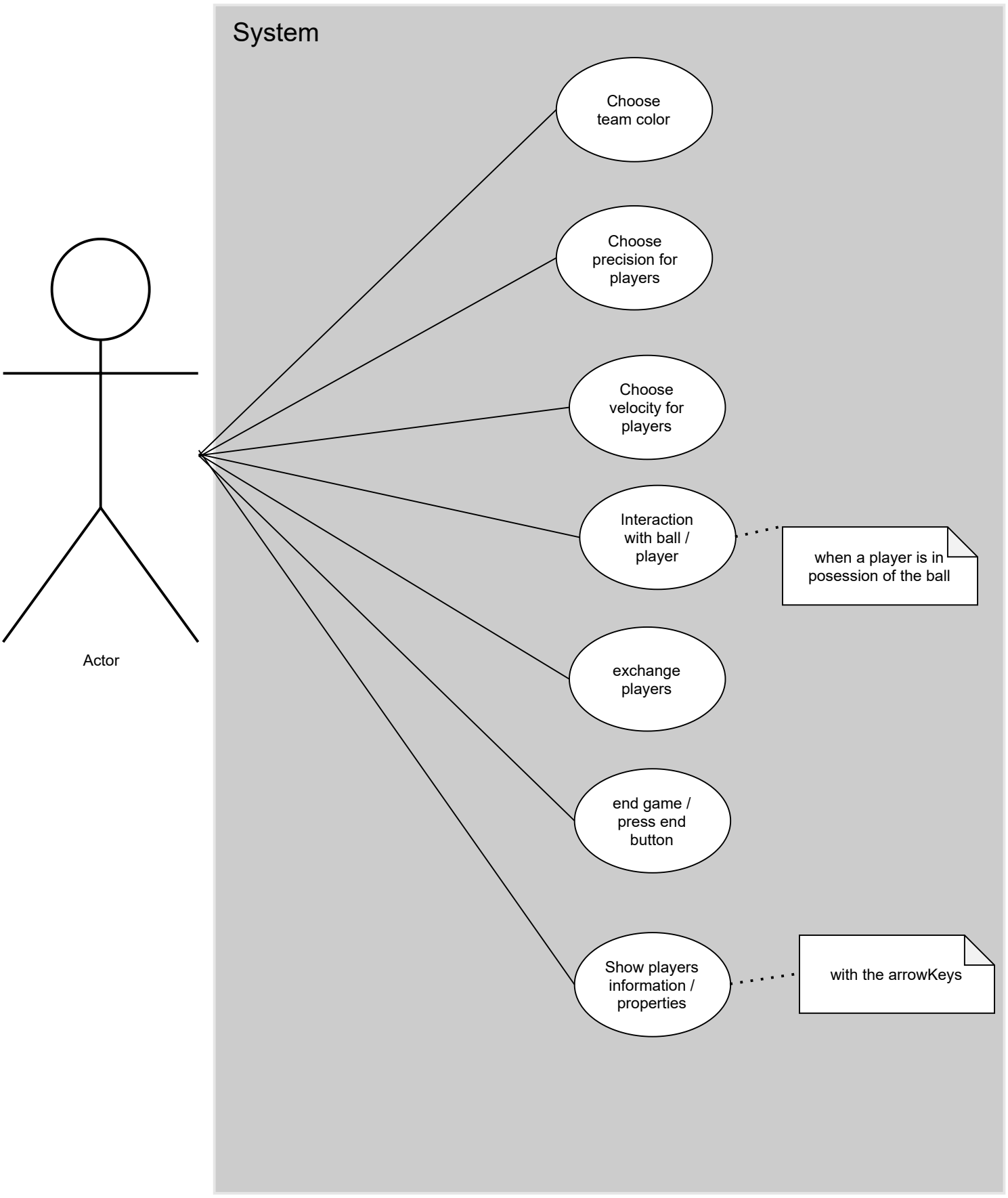
Links in der Tabelle kann man allgemeine Änderungen vornehmen. Mit Höchst- und Minimalgeschwindigkeit werden die Bewegungen der Spieler erhöht bzw. gesenkt. Hierbei muss schlicht der Balken nach links für geringer und rechts für höher geschoben werden. Gleiches gilt für die Präzision der Spieler bei Höchst- und Mindestpräzision. Hierbei wird die Pass- bzw. Schussgenauigkeit adjustiert. Die Farbpaletten drunter sind für das Ändern der Trikotfarben zuständig. Einstellungen werden hierbei nur übernommen, sofern man die Palette wieder schließt.

Auf der rechten Seite kann man Spieler auswechseln, die bereits am Fußballrand stehen. Mit den Pfeiltasten sucht man sich einen Spieler aus und wählt dann in der Tabelle im Dropdown-Menü einen Auswechselspieler aus. Auf den Button „Change“ werden diese dann ausgewechselt.

Möchte man das Spiel neu starten, drückt man auf den „Restart“ Button, unterhalb des Spielfeldes. Außerdem wird der Torzähler bei einem Tor aktualisiert und ständig der Ballbesitz angezeigt.

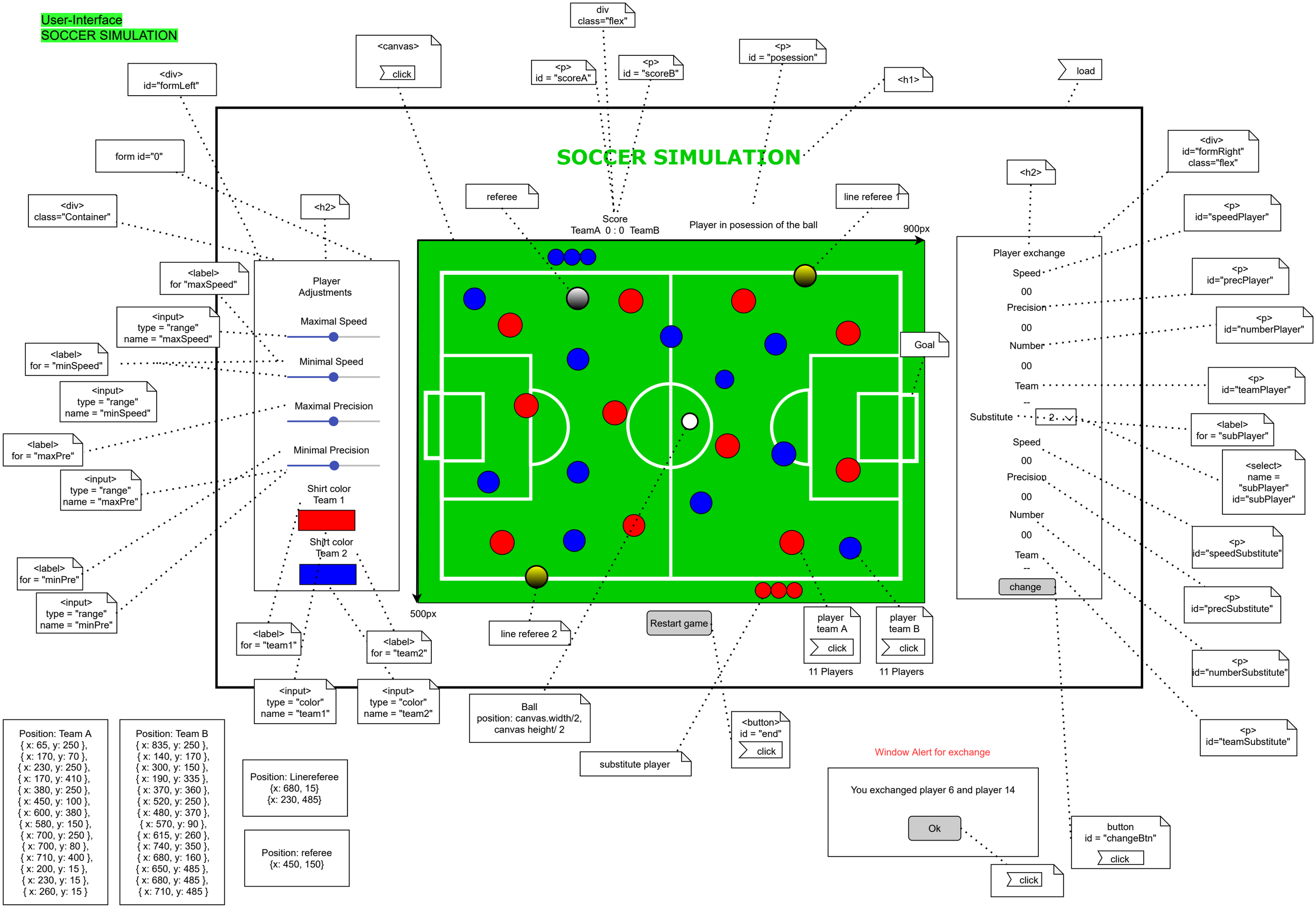
Die Endabgabe entstand durch die Zusammenarbeit mit Mona Kabelka, Christina Däschner, Mariia Kolkutova und Timur Yildirim.

Soccer Simulation: Use-Case-Diagram



User-Interface
SOCCER SIMULATION

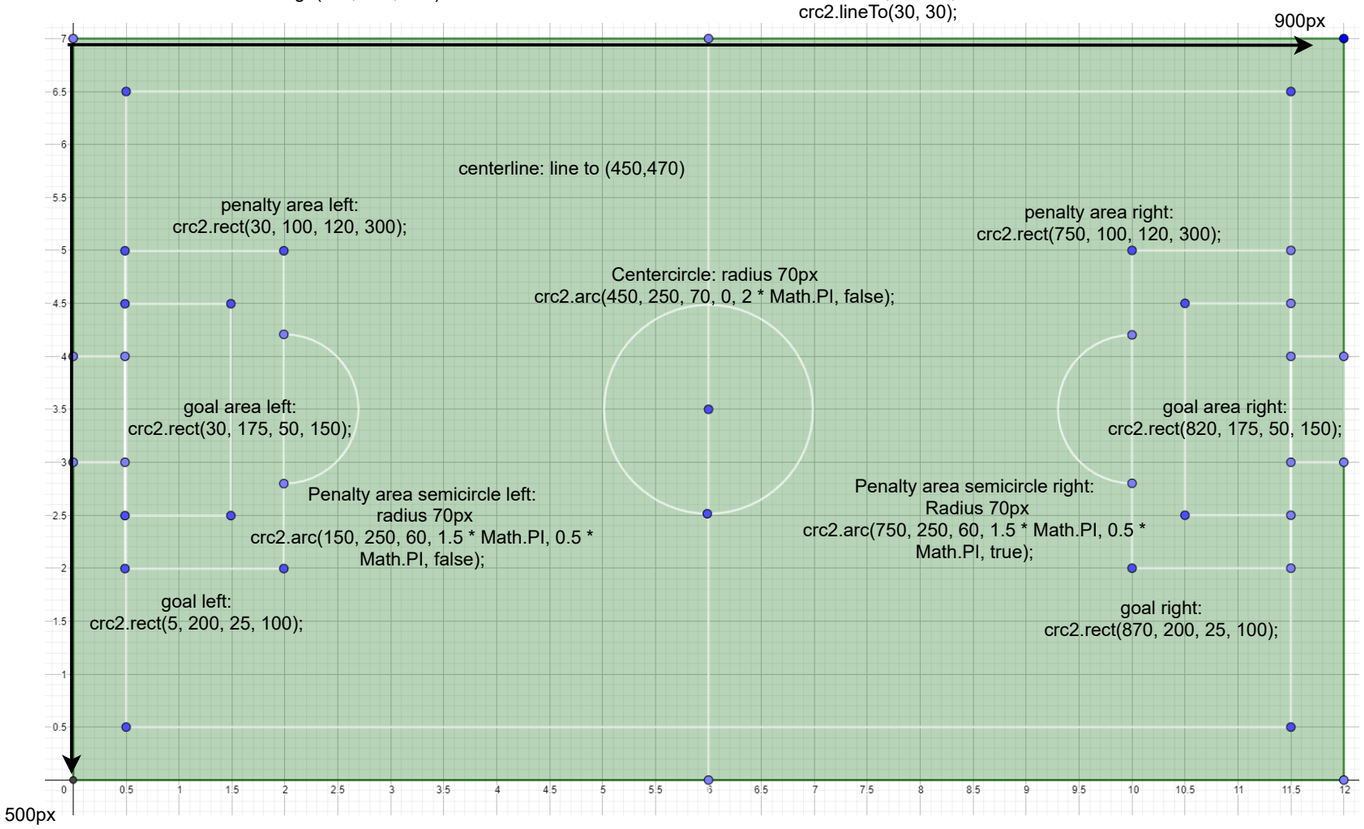
SOCCER SIMULATION



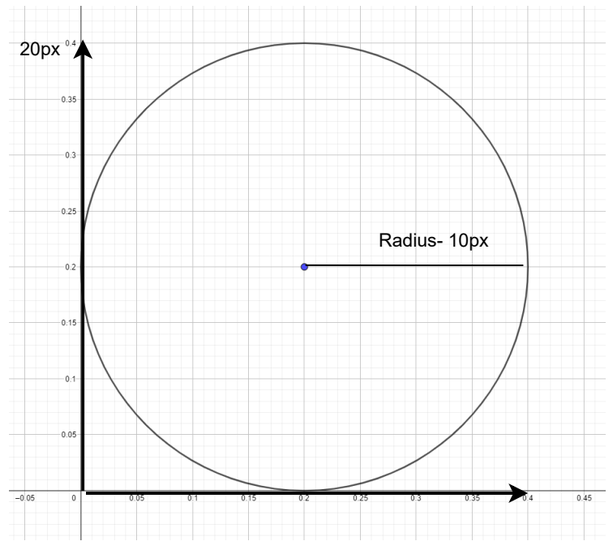
Measurements
with Geogebra

Soccerfield rgb(117, 159, 105)

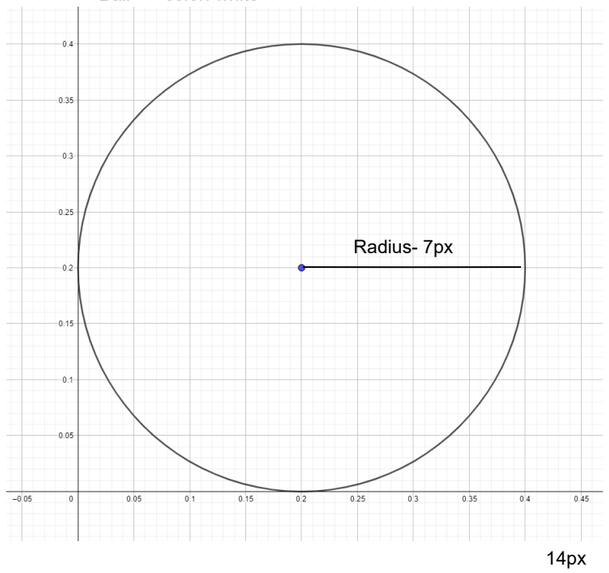
Outfieldline:
crc2.lineTo(870, 30);
crc2.lineTo(870, 470);
crc2.lineTo(30, 470);
crc2.lineTo(30, 30);



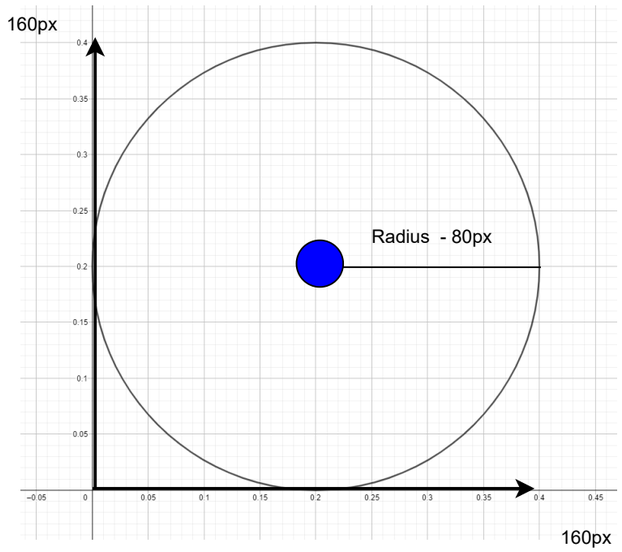
Player



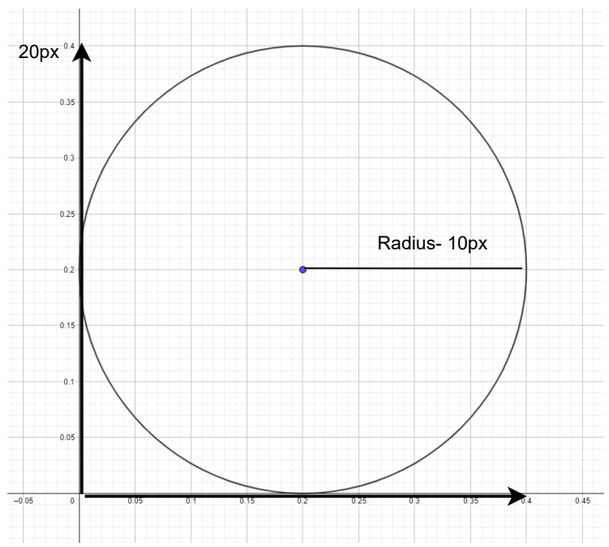
Ball color: white



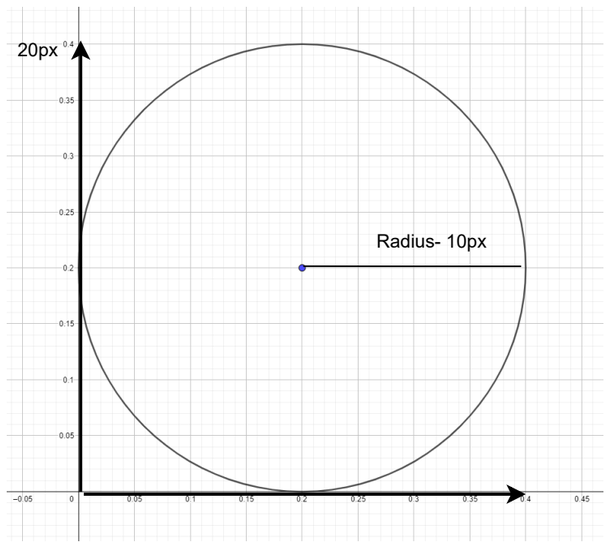
Radius between player and ball - 80px



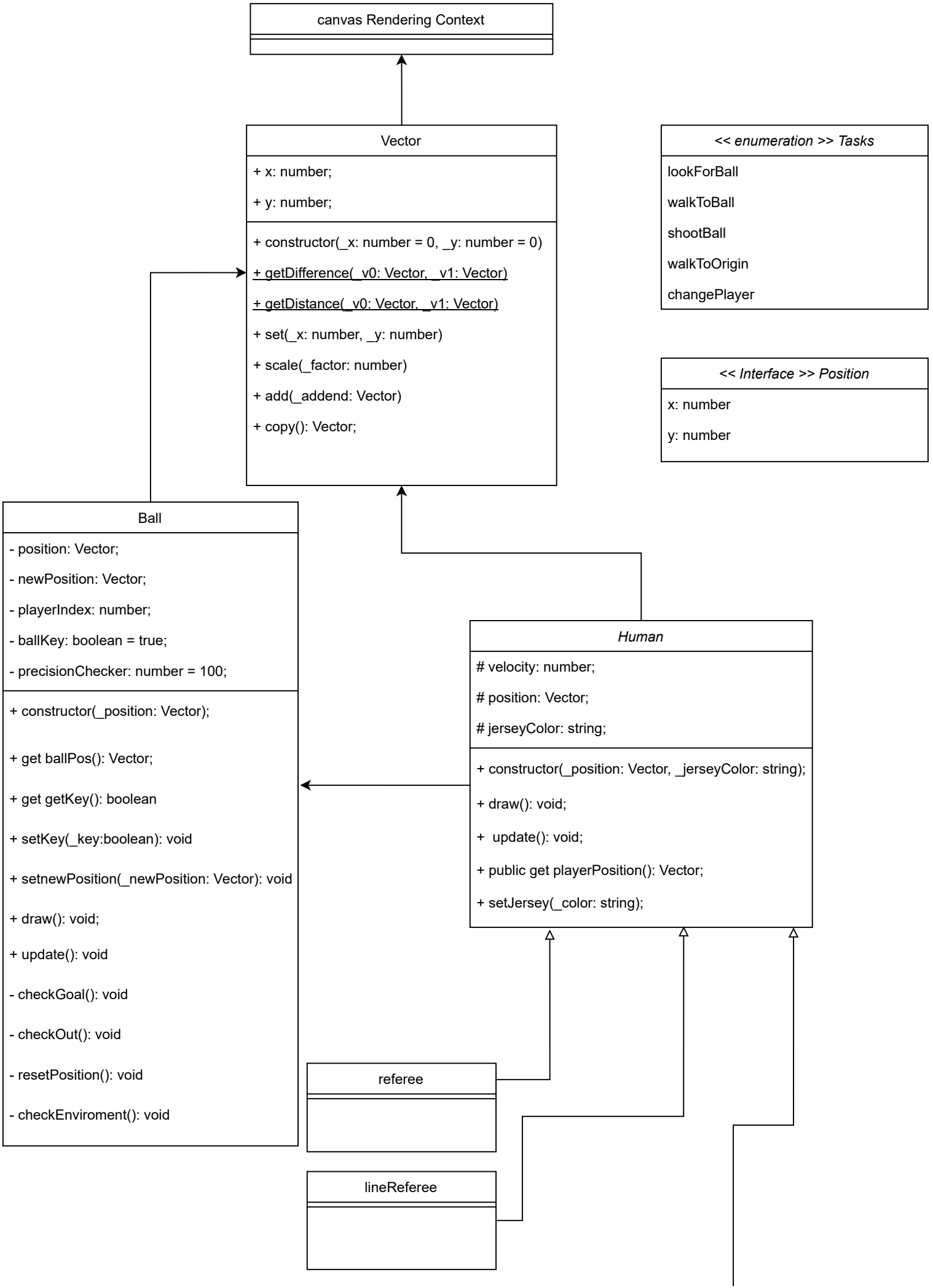
Referee

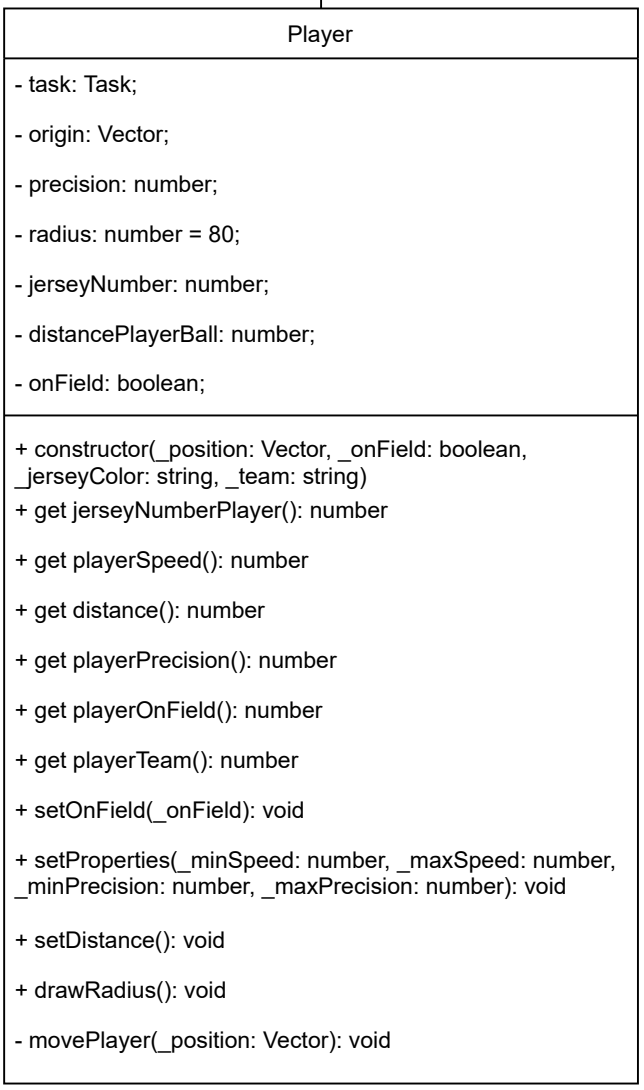


line referee

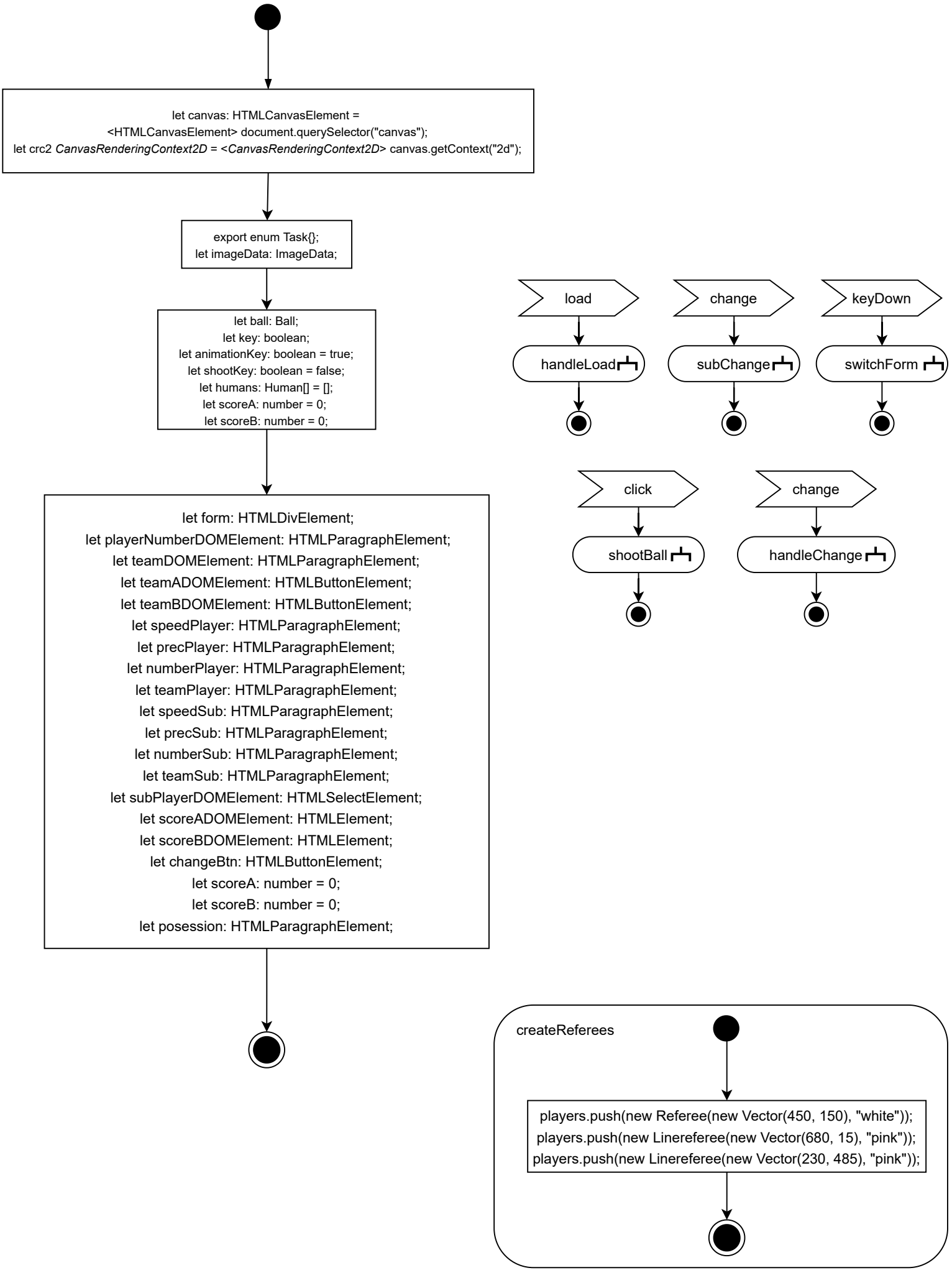


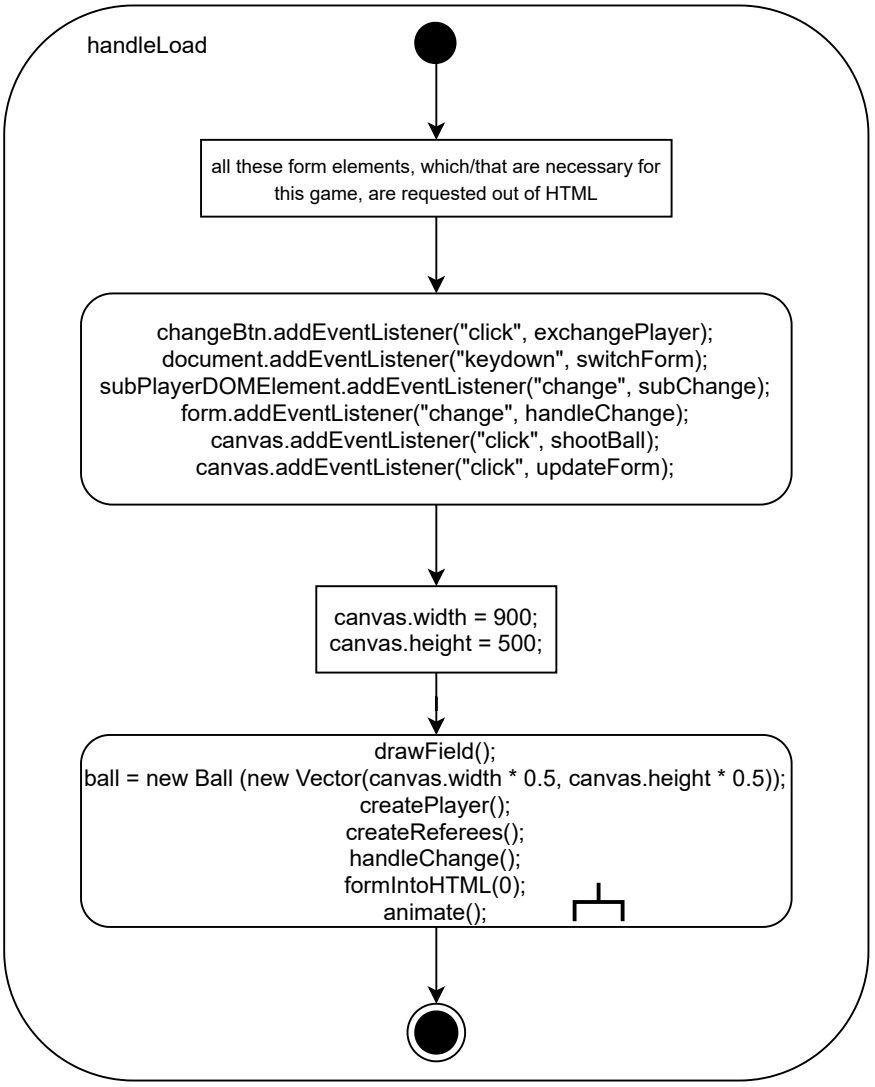
Class-Diagramm

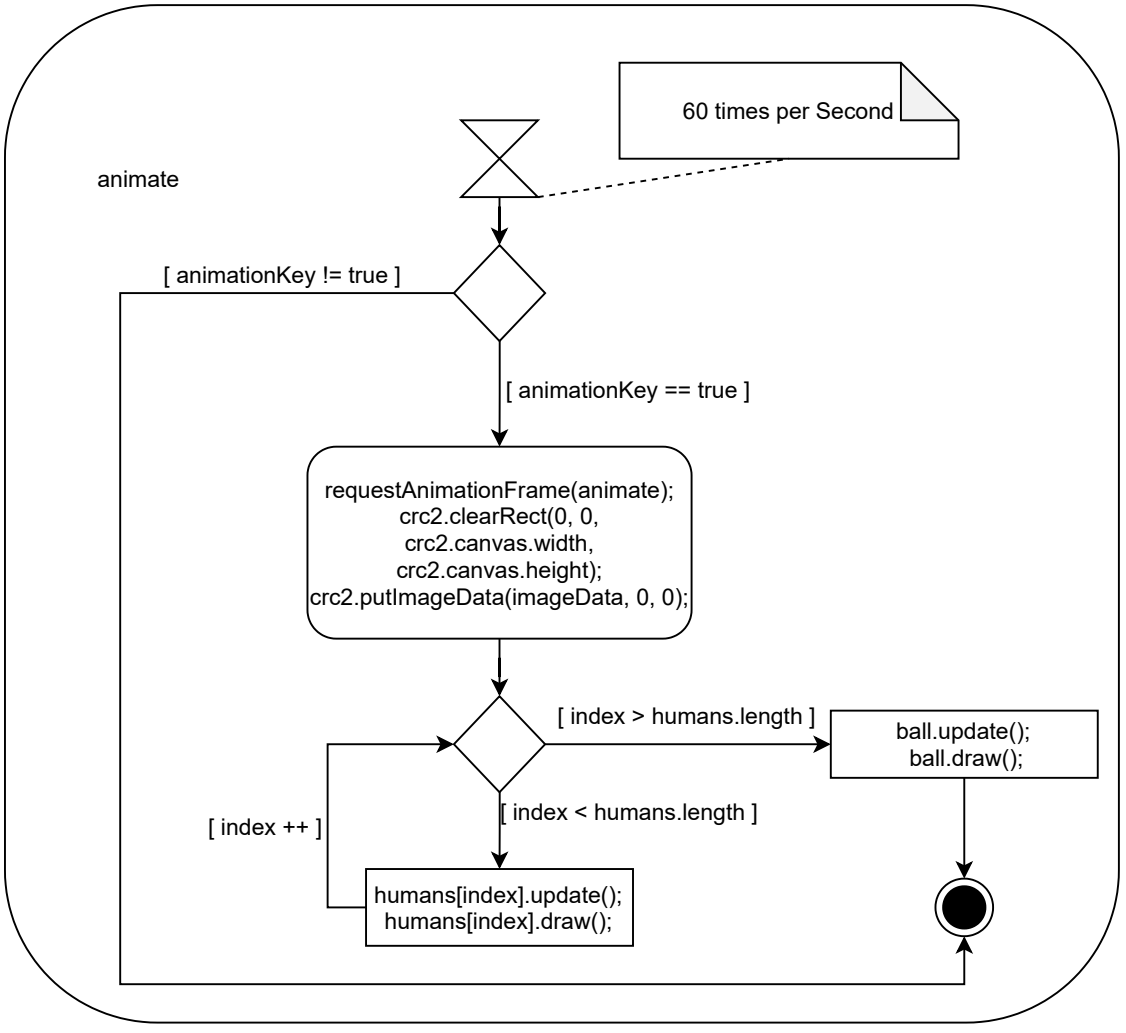


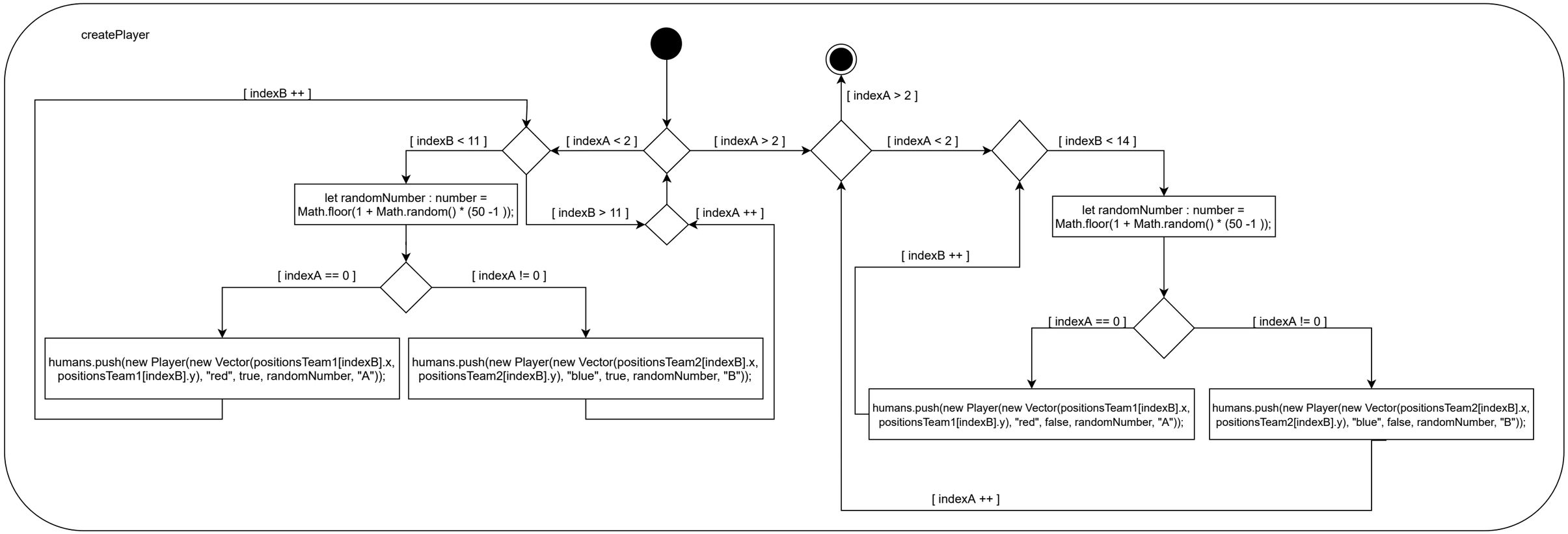

Player
- task: Task;
- origin: Vector;
- precision: number;
- radius: number = 80;
- jerseyNumber: number;
- distancePlayerBall: number;
- onField: boolean;
+ constructor(_position: Vector, _onField: boolean, _jerseyColor: string, _team: string)
+ get jerseyNumberPlayer(): number
+ get playerSpeed(): number
+ get distance(): number
+ get playerPrecision(): number
+ get playerOnField(): number
+ get playerTeam(): number
+ setOnField(_onField): void
+ setProperties(_minSpeed: number, _maxSpeed: number, _minPrecision: number, _maxPrecision: number): void
+ setDistance(): void
+ drawRadius(): void
- movePlayer(_position: Vector): void

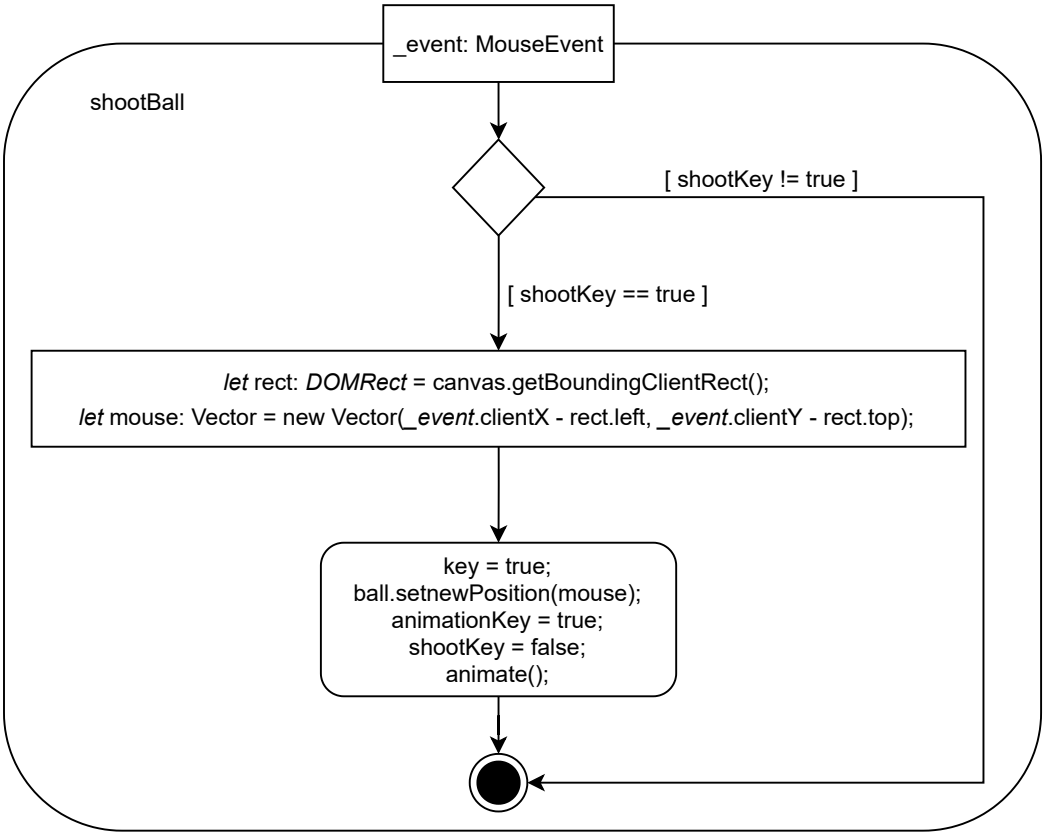
Activity-Diagram: main



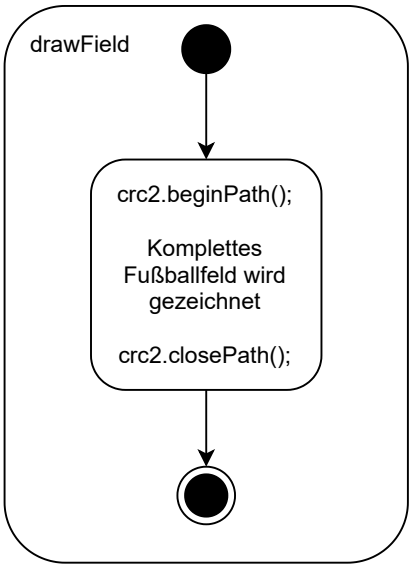

handleLoad
all these form elements, which/that are necessary for this game, are requested out of HTML
changeBtn.addEventListener("click", exchangePlayer);
document.addEventListener("keydown", switchForm);
subPlayerDOMElement.addEventListener("change", subChange);
form.addEventListener("change", handleChange);
canvas.addEventListener("click", shootBall);
canvas.addEventListener("click", updateForm);
canvas.width = 900;
canvas.height = 500;
drawField();
ball = new Ball (new Vector(canvas.width * 0.5, canvas.height * 0.5));
createPlayer();
createReferees();
handleChange();
formIntoHTML(0);
animate();


60 times per Second
animate
[animationKey != true]
[animationKey == true]
requestAnimationFrame(animate);
crc2.clearRect(0, 0,
crc2.canvas.width,
crc2.canvas.height);
crc2.putImageData(imageData, 0, 0);
[index > humans.length]
ball.update();
ball.draw();
[index < humans.length]
[index ++]
humans[index].update();
humans[index].draw();

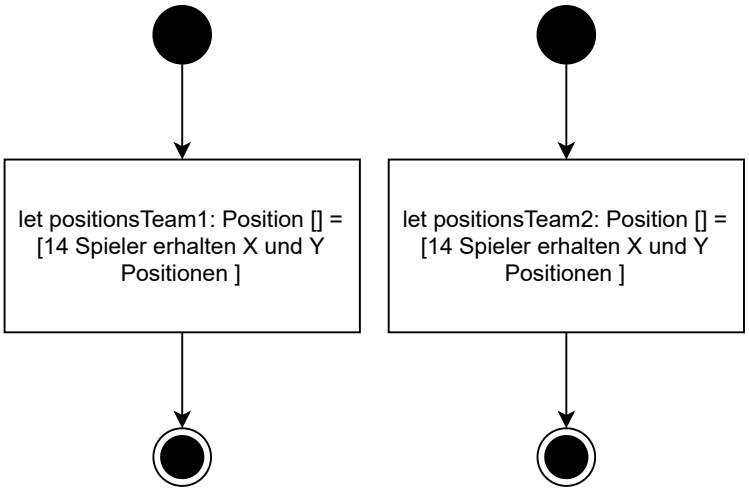

createPlayer
[indexB ++]
[indexB < 11]
[indexA < 2]
[indexA > 2]
[indexA > 2]
[indexA < 2]
[indexB < 14]
let randomNumber : number =
Math.floor(1 + Math.random() * (50 -1));
[indexB > 11]
[indexA ++]
let randomNumber : number =
Math.floor(1 + Math.random() * (50 -1));
[indexB ++]
[indexA == 0]
[indexA != 0]
[indexA == 0]
[indexA != 0]
humans.push(new Player(new Vector(positionsTeam1[indexB].x,
positionsTeam1[indexB].y), "red", true, randomNumber, "A"));
humans.push(new Player(new Vector(positionsTeam2[indexB].x,
positionsTeam2[indexB].y), "blue", true, randomNumber, "B"));
humans.push(new Player(new Vector(positionsTeam1[indexB].x,
positionsTeam1[indexB].y), "red", false, randomNumber, "A"));
humans.push(new Player(new Vector(positionsTeam2[indexB].x,
positionsTeam2[indexB].y), "blue", false, randomNumber, "B"));
[indexA ++]



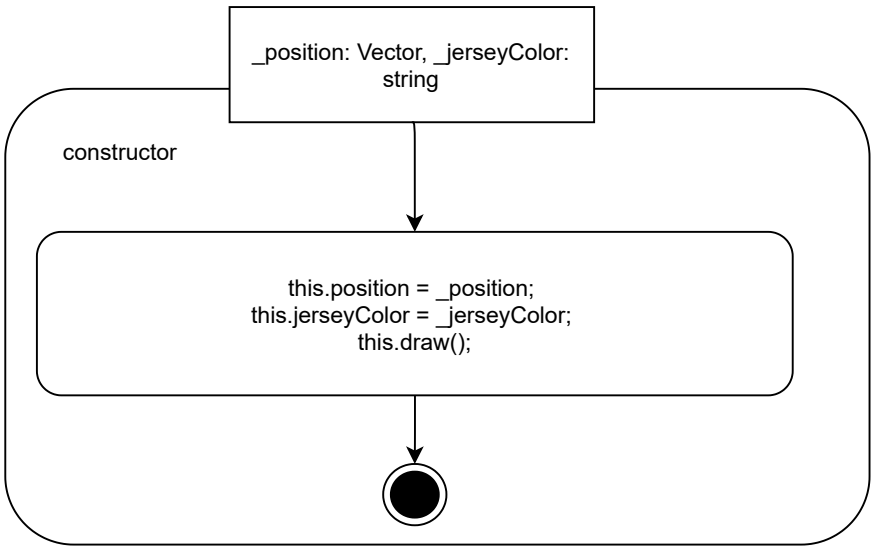
Activity-Diagram: footballField

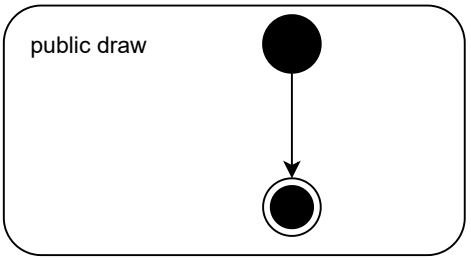


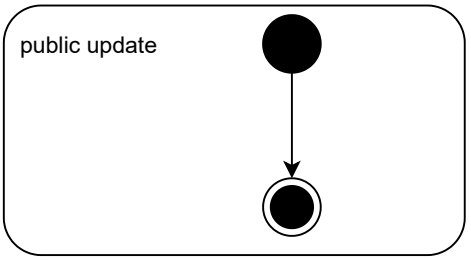
Activity-Diagram: position

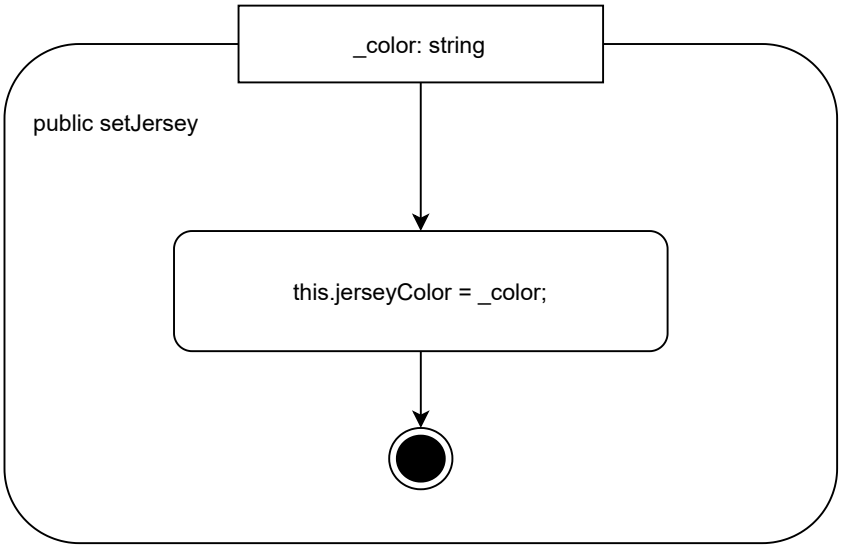


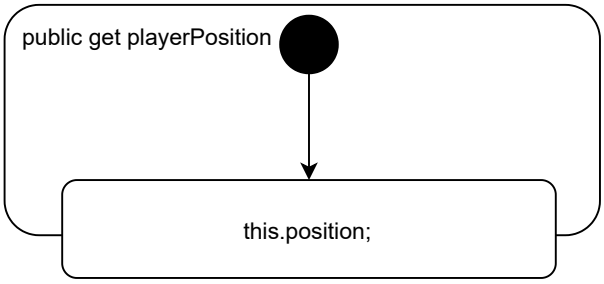
Activity-Diagram: human



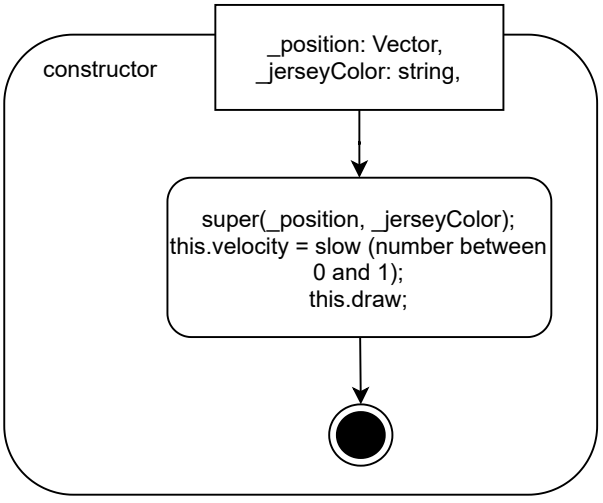


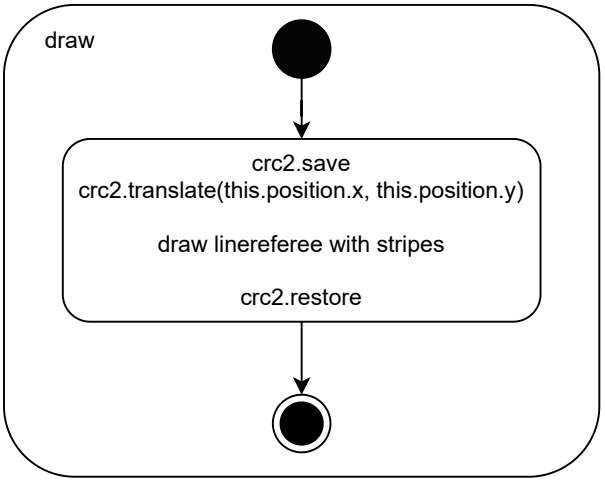


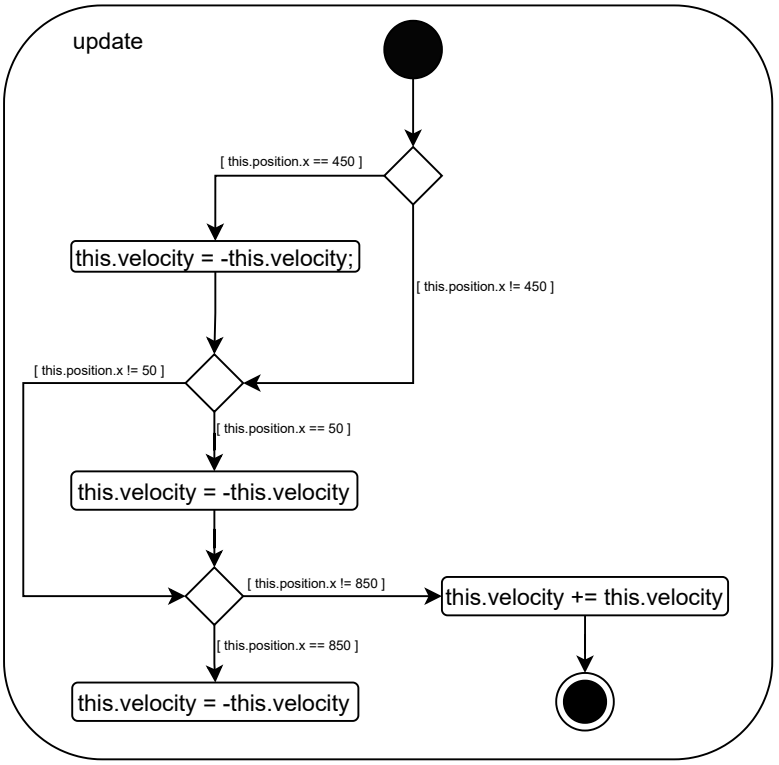




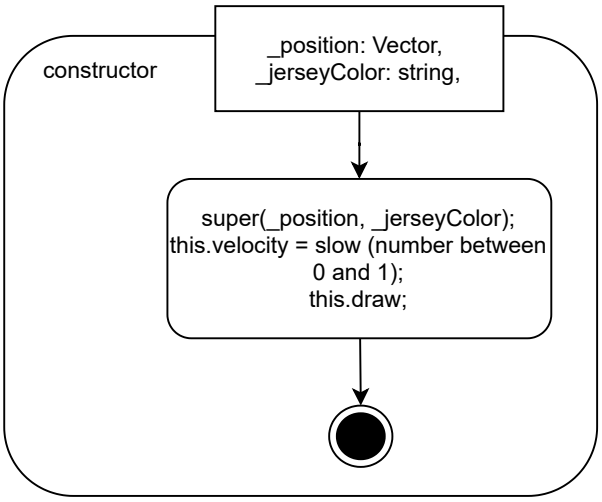
Activity-Diagram: lineReferee

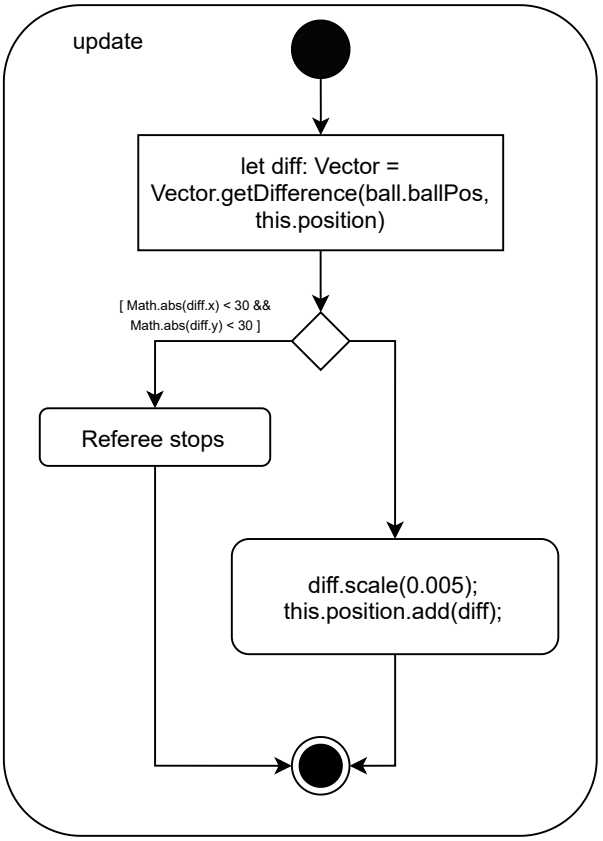


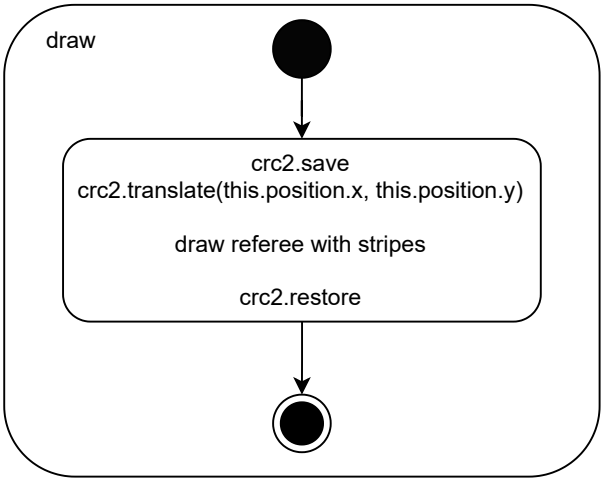




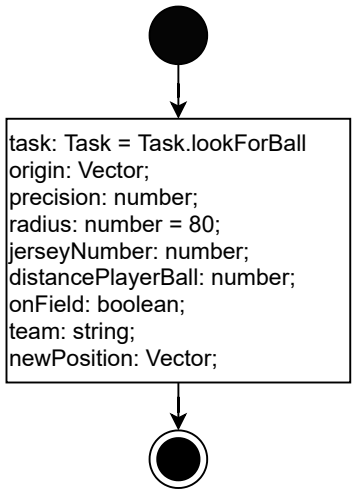
Activity-Diagram: Referee

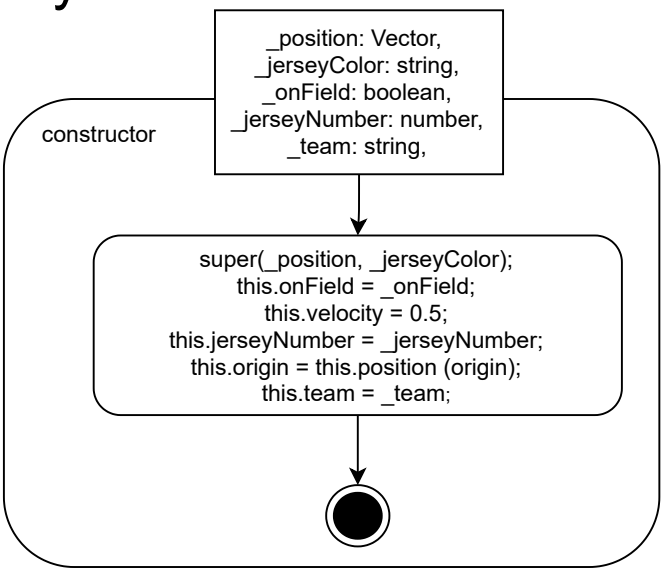


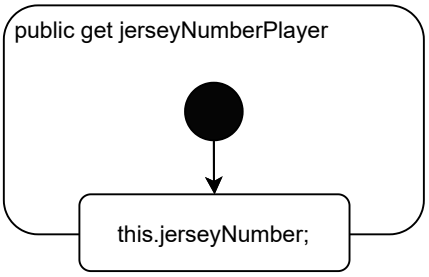


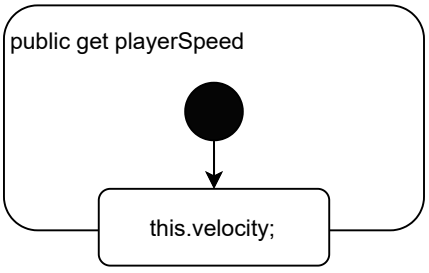


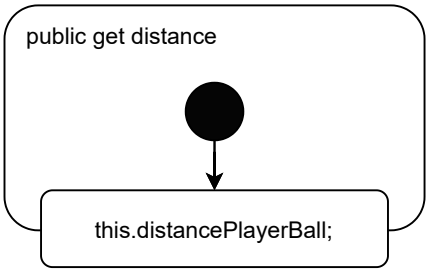
Activity-Diagram: player

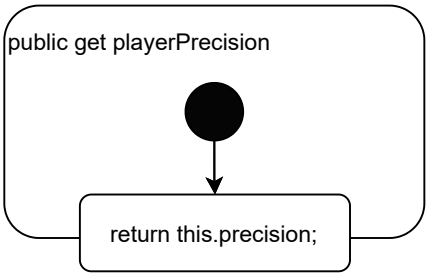


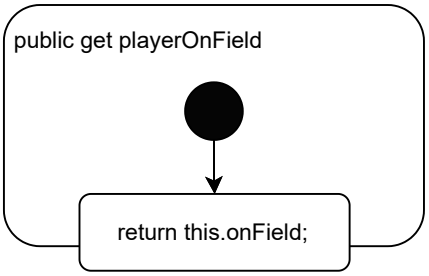


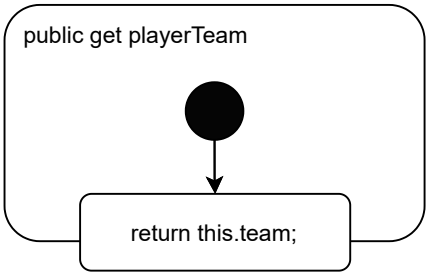


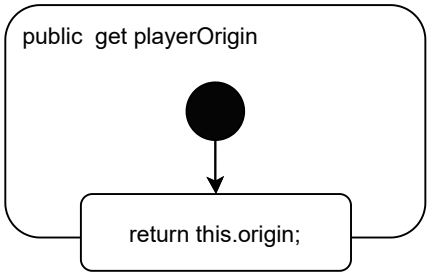


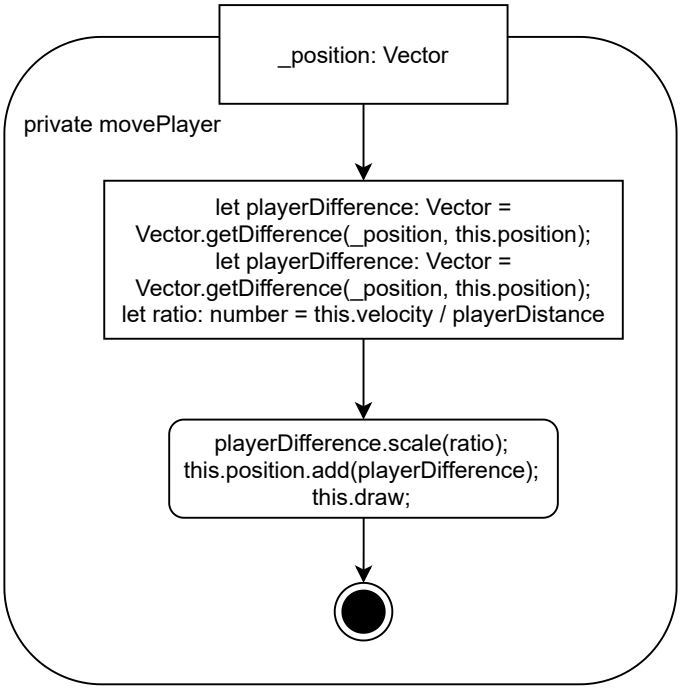


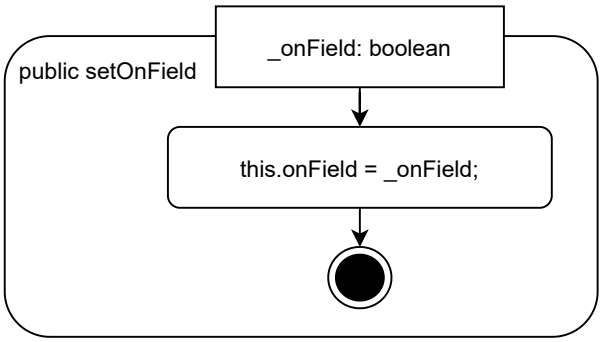


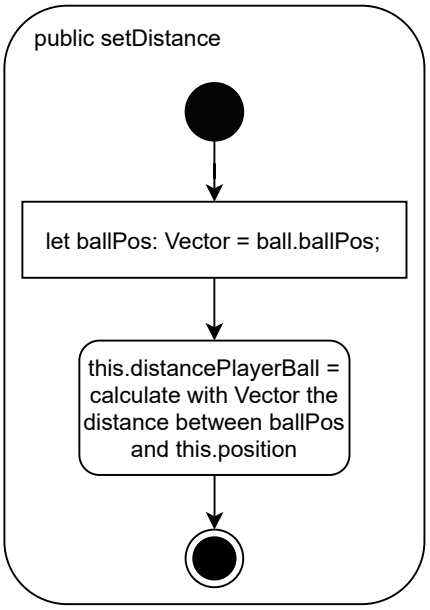


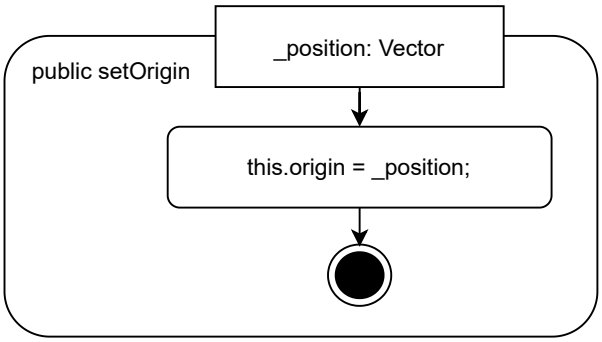


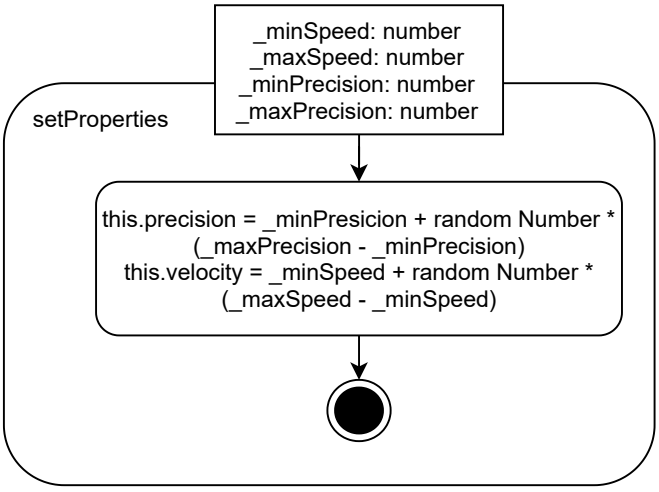


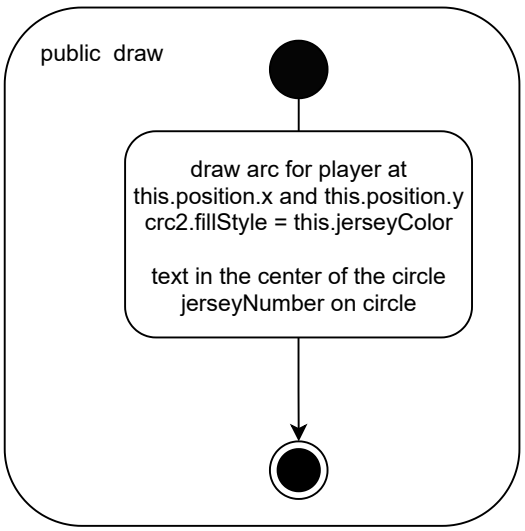


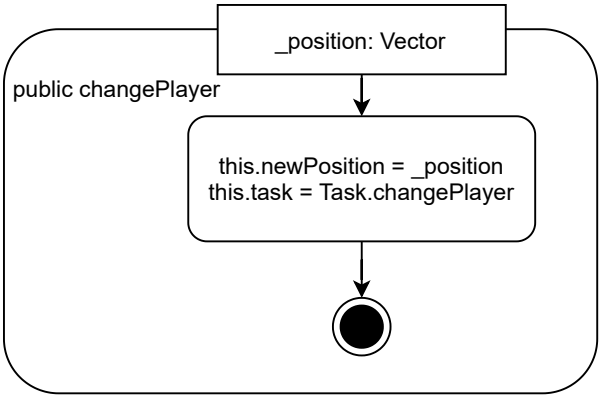

public setOnField
_onField: boolean
this.onField = _onField;

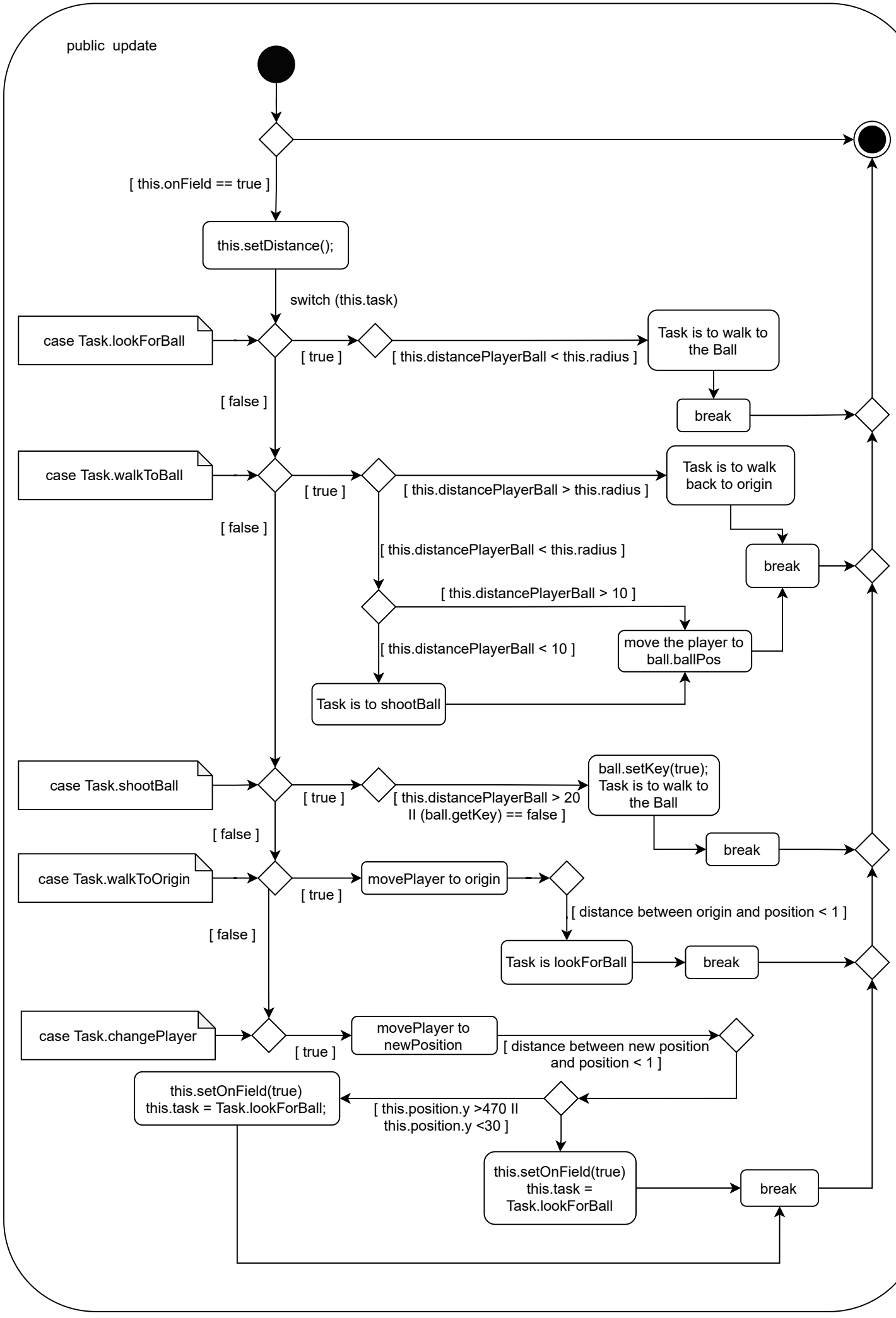

public setDistance
let ballPos: Vector = ball.ballPos;
this.distancePlayerBall = calculate with Vector the distance between ballPos and this.position


public setOrigin
_position: Vector
this.origin = _position;

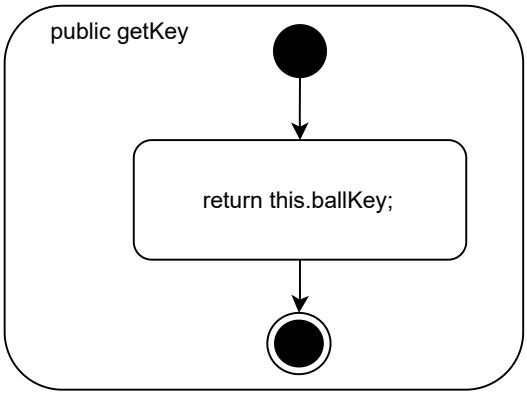

setProperties
_minSpeed: number
_maxSpeed: number
_minPrecision: number
_maxPrecision: number
this.precision = _minPresicion + random Number * (_maxPrecision - _minPrecision)
this.velocity = _minSpeed + random Number * (_maxSpeed - _minSpeed)

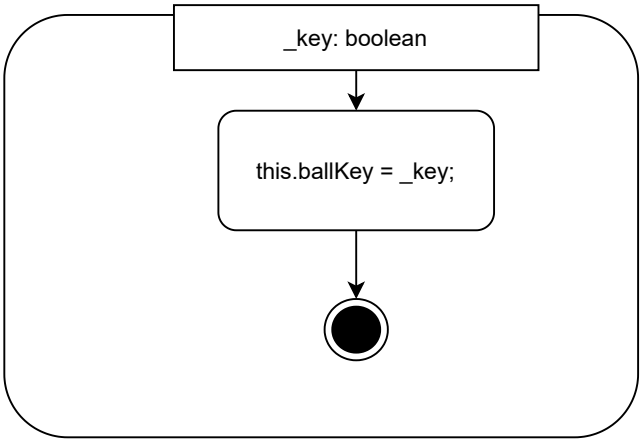

public draw
draw arc for player at this.position.x and this.position.y
crc2.fillStyle = this.jerseyColor
text in the center of the circle
jerseyNumber on circle

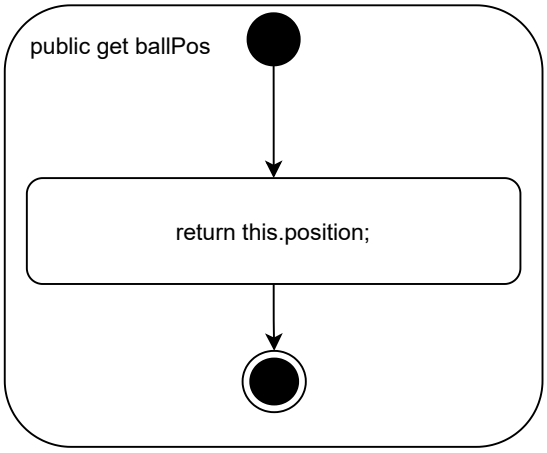

public changePlayer
_position: Vector
this.newPosition = _position
this.task = Task.changePlayer

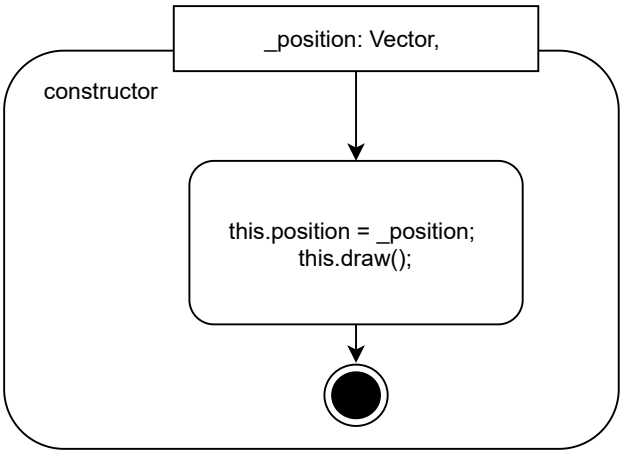

public update
[this.onField == true]
this.setDistance();
switch (this.task)
case Task.lookForBall
[true]
[this.distancePlayerBall < this.radius]
Task is to walk to the Ball
break
[false]
case Task.walkToBall
[true]
[this.distancePlayerBall > this.radius]
Task is to walk back to origin
break
[false]
[this.distancePlayerBall < this.radius]
[this.distancePlayerBall > 10]
move the player to ball.ballPos
[this.distancePlayerBall < 10]
Task is to shootBall
case Task.shootBall
[true]
[this.distancePlayerBall > 20 II (ball.getKey) == false]
ball.setKey(true); Task is to walk to the Ball
break
[false]
case Task.walkToOrigin
[true]
movePlayer to origin
[distance between origin and position < 1]
Task is lookForBall
break
[false]
case Task.changePlayer
[true]
movePlayer to newPosition
[distance between new position and position < 1]
this.setOnField(true) this.task = Task.lookForBall;
[this.position.y >470 II this.position.y <30]
this.setOnField(true) this.task = Task.lookForBall
break

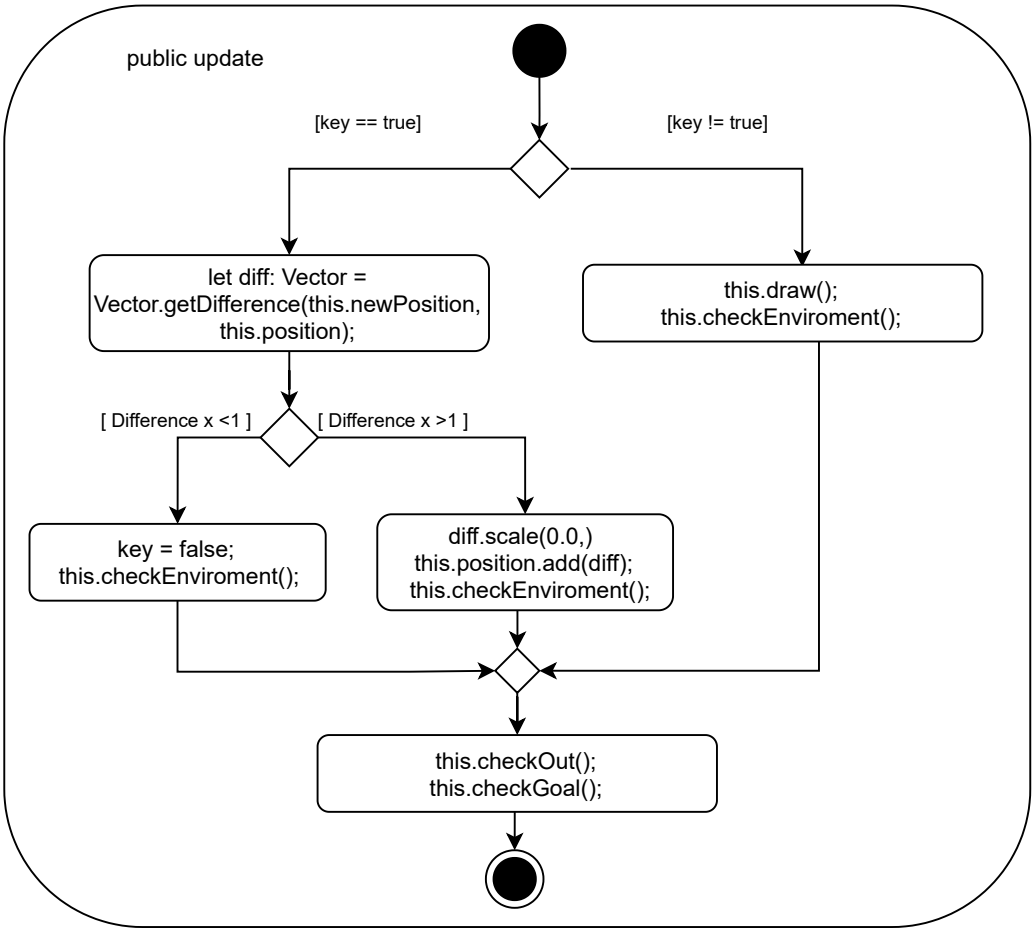
Activity-Diagram: ball

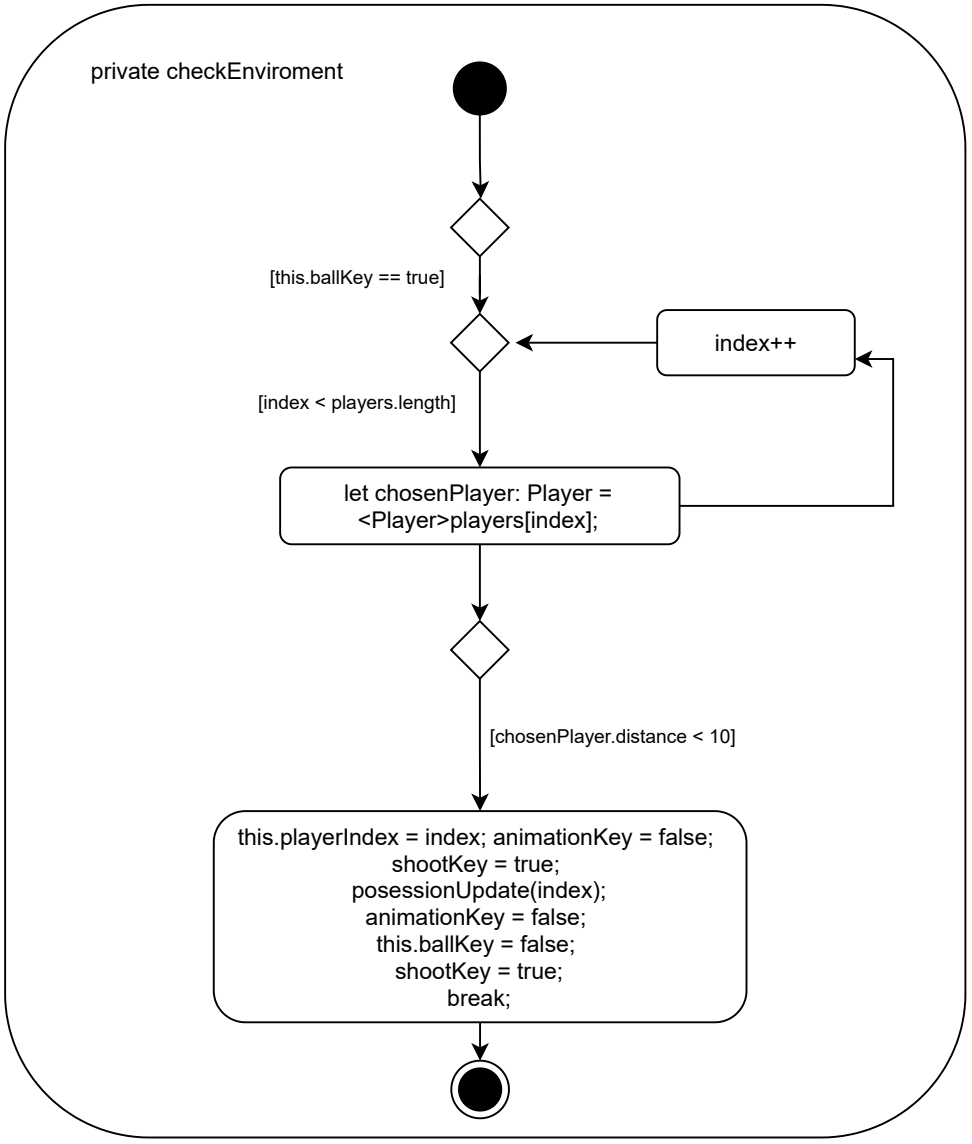


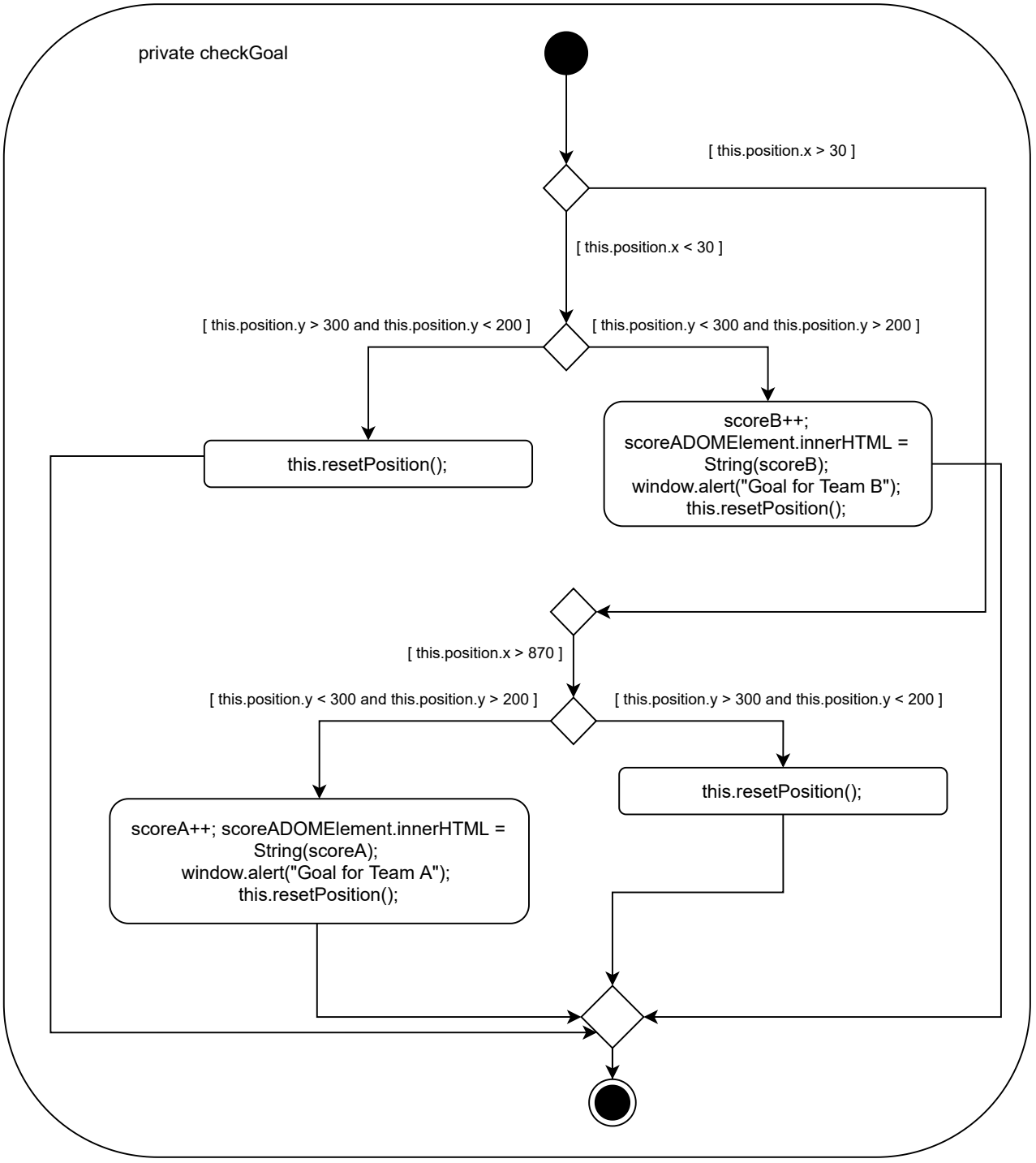


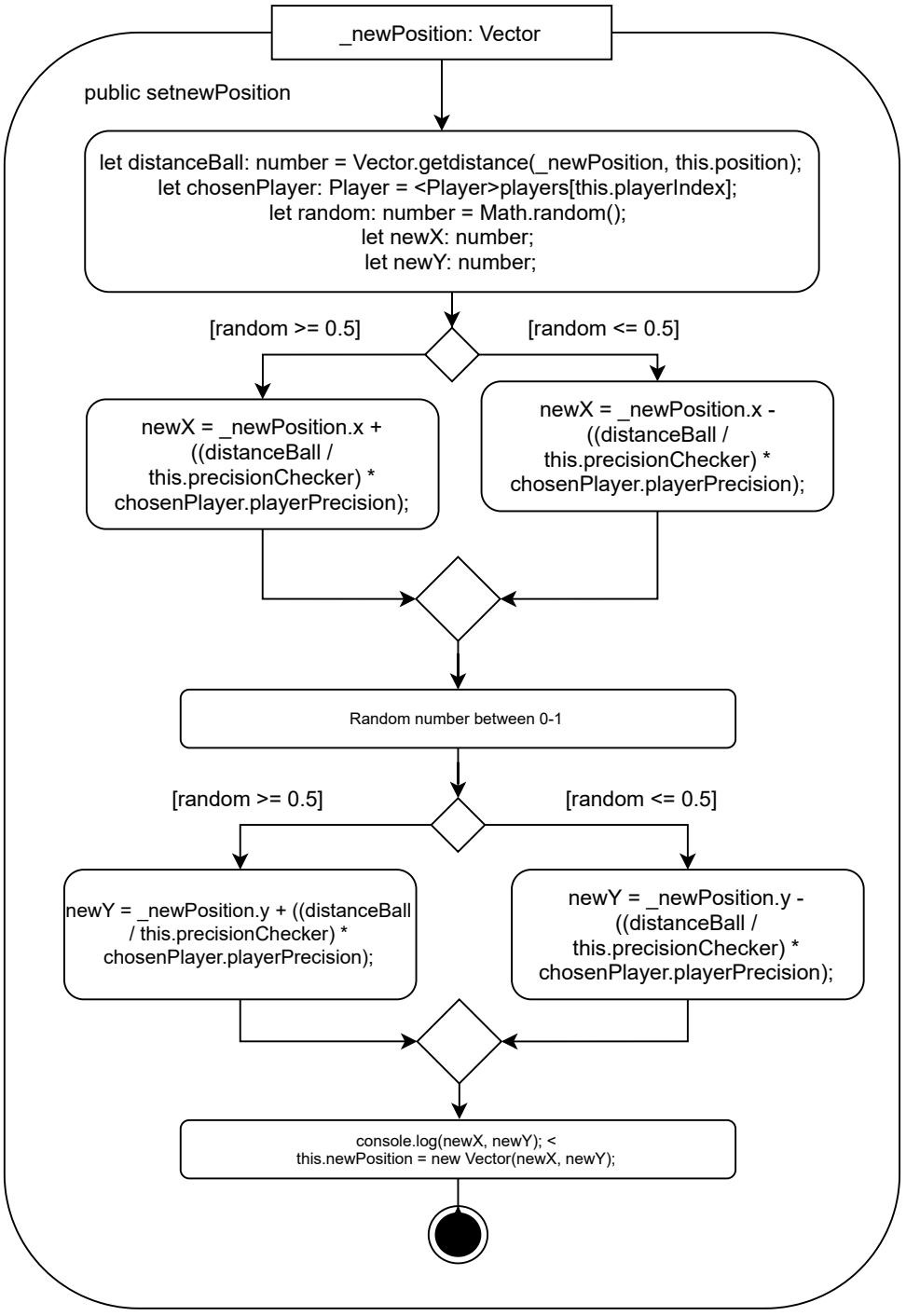




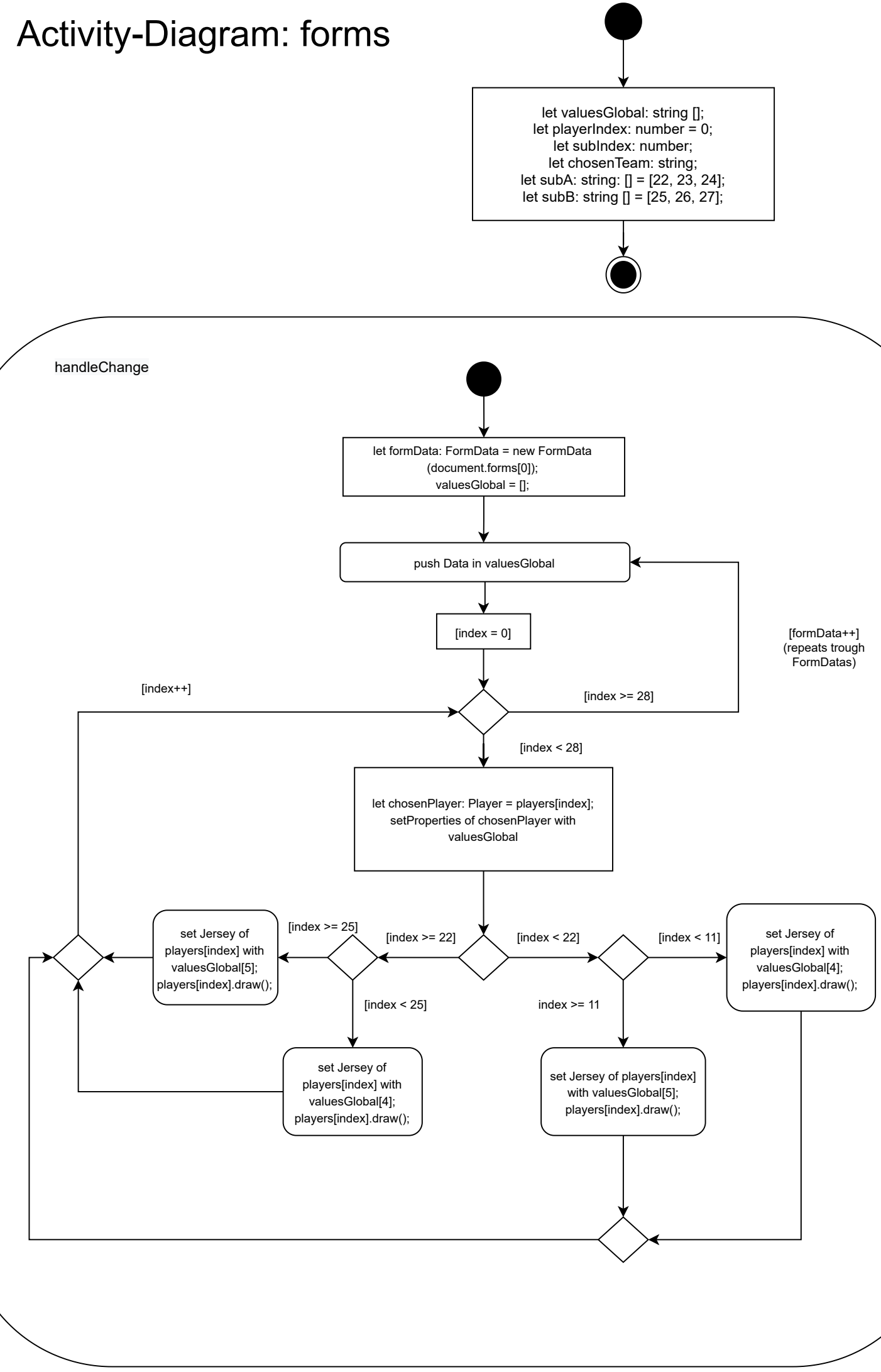

public update
[key == true]
[key != true]
let diff: Vector = Vector.getDifference(this.newPosition, this.position);
this.draw();
this.checkEnviroment();
[Difference x <1]
[Difference x >1]
key = false;
this.checkEnviroment();
diff.scale(0.0,)
this.position.add(diff);
this.checkEnviroment();
this.checkOut();
this.checkGoal();

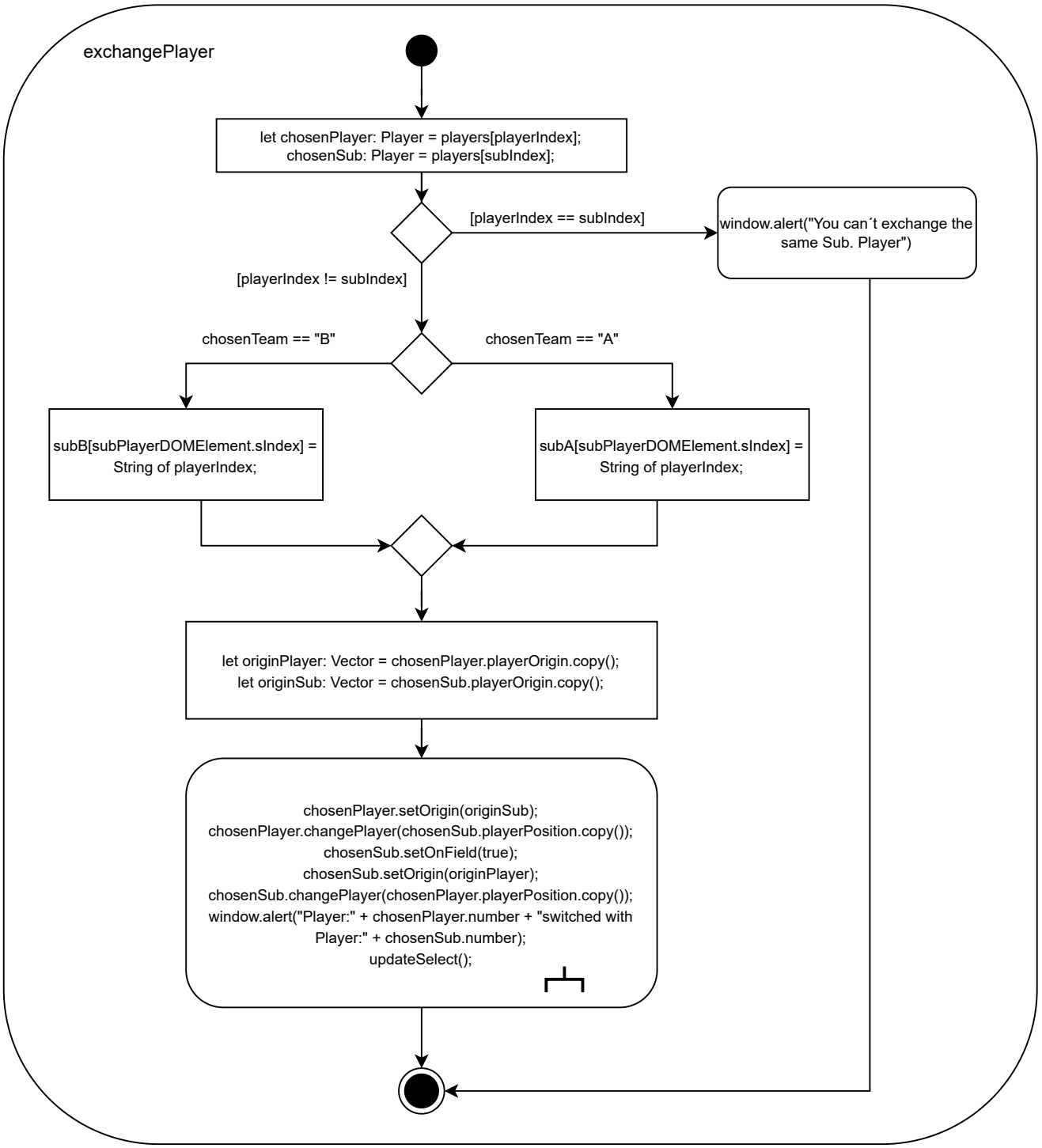

private checkEnviroment
[this.ballKey == true]
index++
[index < players.length]
let chosenPlayer: Player = <Player>players[index];
[chosenPlayer.distance < 10]
this.playerIndex = index; animationKey = false;
shootKey = true;
posessionUpdate(index);
animationKey = false;
this.ballKey = false;
shootKey = true;
break;

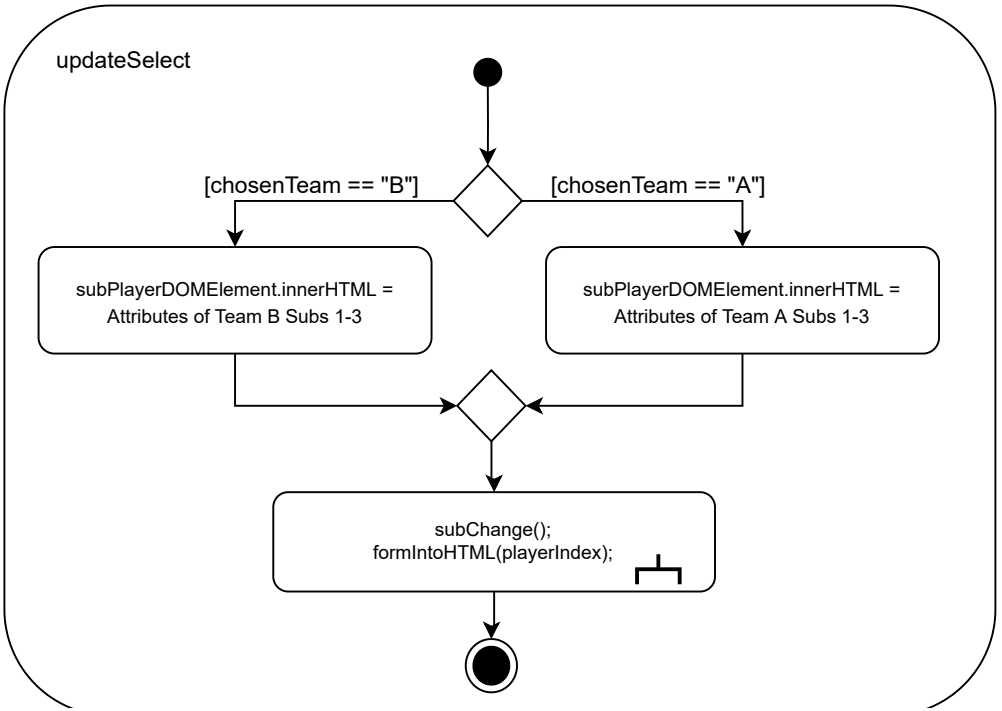

private checkGoal
[this.position.x > 30]
[this.position.x < 30]
[this.position.y > 300 and this.position.y < 200]
[this.position.y < 300 and this.position.y > 200]
this.resetPosition();
scoreB++;
scoreADOMElement.innerHTML =
String(scoreB);
window.alert("Goal for Team B");
this.resetPosition();
[this.position.x > 870]
[this.position.y < 300 and this.position.y > 200]
[this.position.y > 300 and this.position.y < 200]
this.resetPosition();
scoreA++; scoreADOMElement.innerHTML =
String(scoreA);
window.alert("Goal for Team A");
this.resetPosition();

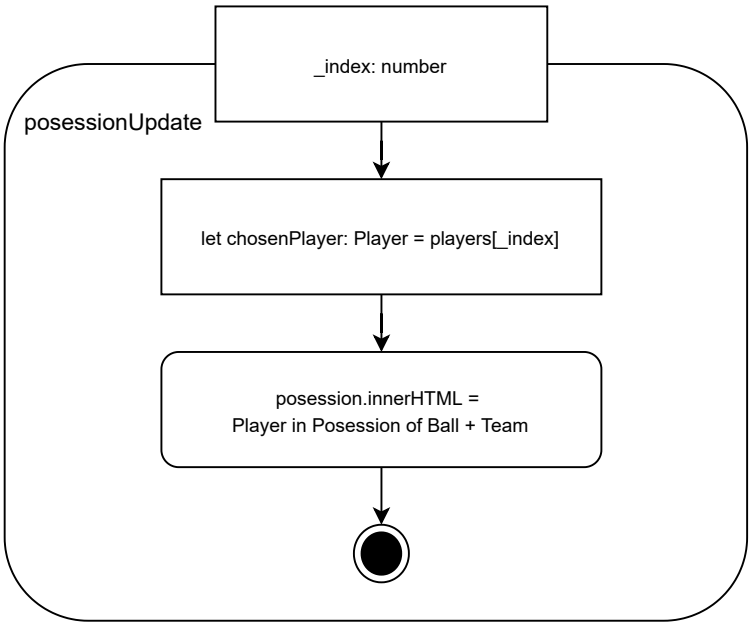

_newPosition: Vector
public setnewPosition
let distanceBall: number = Vector.getdistance(_newPosition, this.position);
let chosenPlayer: Player = <Player>players[this.playerIndex];
let random: number = Math.random();
let newX: number;
let newY: number;
[random >= 0.5]
[random <= 0.5]
newX = _newPosition.x + ((distanceBall / this.precisionChecker) * chosenPlayer.playerPrecision);
newX = _newPosition.x - ((distanceBall / this.precisionChecker) * chosenPlayer.playerPrecision);
Random number between 0-1
[random >= 0.5]
[random <= 0.5]
newY = _newPosition.y + ((distanceBall / this.precisionChecker) * chosenPlayer.playerPrecision);
newY = _newPosition.y - ((distanceBall / this.precisionChecker) * chosenPlayer.playerPrecision);
console.log(newX, newY); <
this.newPosition = new Vector(newX, newY);

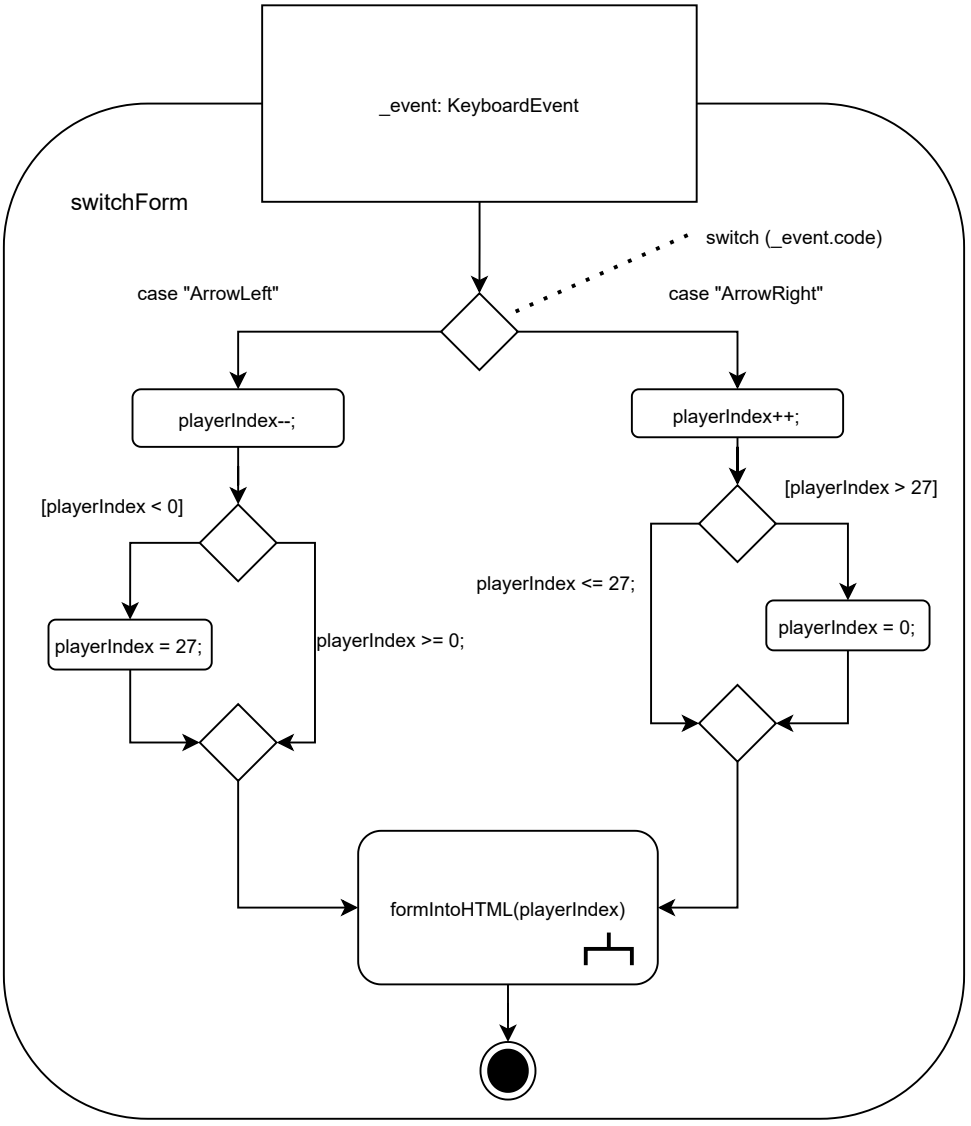
Activity-Diagram: forms

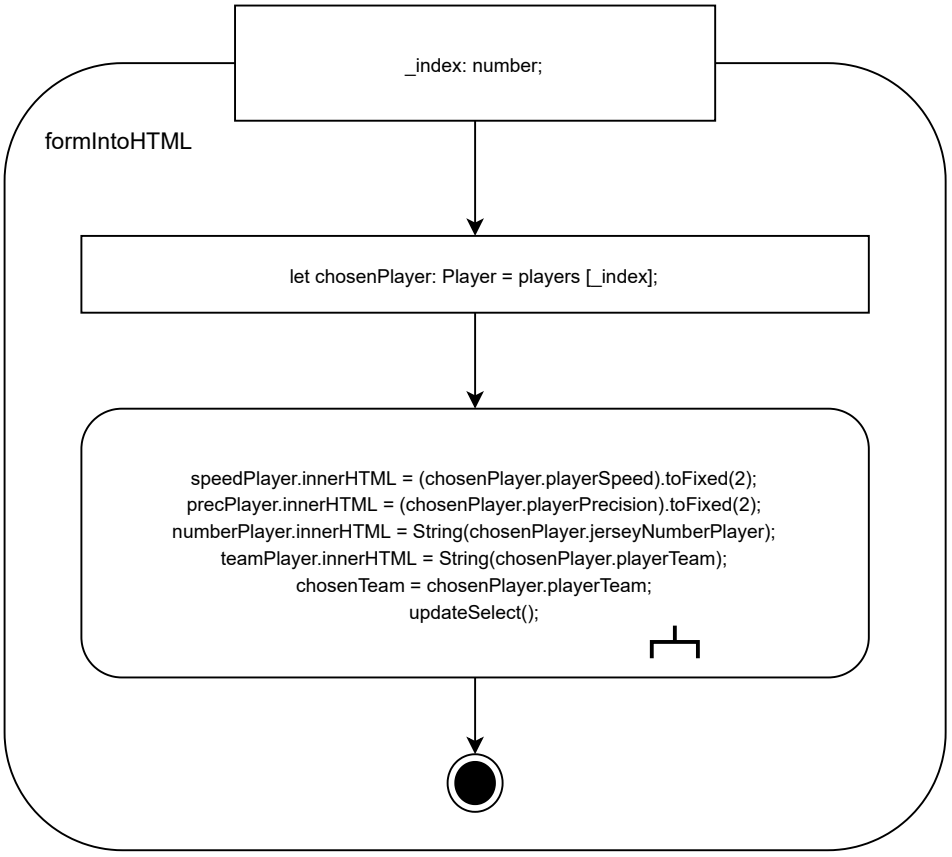


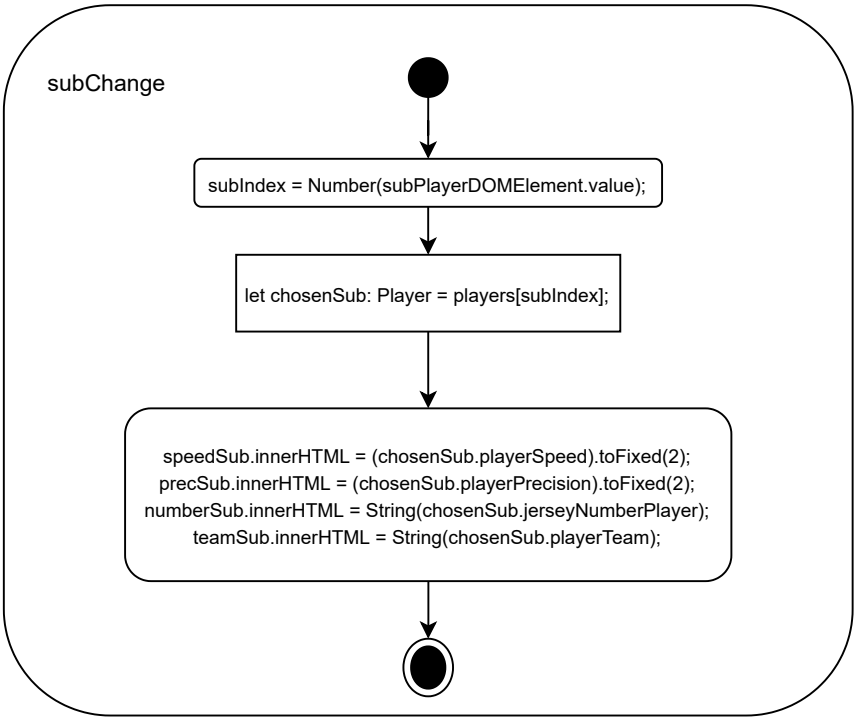

exchangePlayer
let chosenPlayer: Player = players[playerIndex];
chosenSub: Player = players[subIndex];
[playerIndex == subIndex]
window.alert("You can´t exchange the same Sub. Player")
[playerIndex != subIndex]
chosenTeam == "B"
chosenTeam == "A"
subB[subPlayerDOMElement.sIndex] = String of playerIndex;
subA[subPlayerDOMElement.sIndex] = String of playerIndex;
let originPlayer: Vector = chosenPlayer.playerOrigin.copy();
let originSub: Vector = chosenSub.playerOrigin.copy();
chosenPlayer.setOrigin(originSub);
chosenPlayer.changePlayer(chosenSub.playerPosition.copy());
chosenSub.setOnField(true);
chosenSub.setOrigin(originPlayer);
chosenSub.changePlayer(chosenPlayer.playerPosition.copy());
window.alert("Player:" + chosenPlayer.number + "switched with Player:" + chosenSub.number);
updateSelect();


updateSelect
[chosenTeam == "B"]
[chosenTeam == "A"]
subPlayerDOMElement.innerHTML = Attributes of Team B Subs 1-3
subPlayerDOMElement.innerHTML = Attributes of Team A Subs 1-3
subChange();
formIntoHTML(playerIndex);


posessionUpdate
_index: number
let chosenPlayer: Player = players[_index]
posession.innerHTML =
Player in Posession of Ball + Team


switchForm
_event: KeyboardEvent
switch (_event.code)
case "ArrowLeft"
case "ArrowRight"
playerIndex--;
playerIndex++;
[playerIndex < 0]
[playerIndex > 27]
playerIndex = 27;
playerIndex >= 0;
playerIndex <= 27;
playerIndex = 0;
formIntoHTML(playerIndex)


_index: number;
formIntoHTML
let chosenPlayer: Player = players [_index];
speedPlayer.innerHTML = (chosenPlayer.playerSpeed).toFixed(2);
precPlayer.innerHTML = (chosenPlayer.playerPrecision).toFixed(2);
numberPlayer.innerHTML = String(chosenPlayer.jerseyNumberPlayer);
teamPlayer.innerHTML = String(chosenPlayer.playerTeam);
chosenTeam = chosenPlayer.playerTeam;
updateSelect();


subChange
subIndex = Number(subPlayerDOMElement.value);
let chosenSub: Player = players[subIndex];
speedSub.innerHTML = (chosenSub.playerSpeed).toFixed(2);
precSub.innerHTML = (chosenSub.playerPrecision).toFixed(2);
numberSub.innerHTML = String(chosenSub.jerseyNumberPlayer);
teamSub.innerHTML = String(chosenSub.playerTeam);

Activity-Diagram: vector