Abschlussarbeit S21 EIA2

Mariia Kolkutova MKB2 266409

Anleitung EIA2 Abschlussarbeit S21

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.

Zusammengearbeitet haben wir Huu Thien Phan Ngoc, Mona Kabelka, Christina Däschner, Timur Yildirim und Mariia Kolkutova.

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 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 den Player Adjustment kann man allgemeine Änderungen vornehmen. Min and Mas Speed 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 min and mad precision. 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.

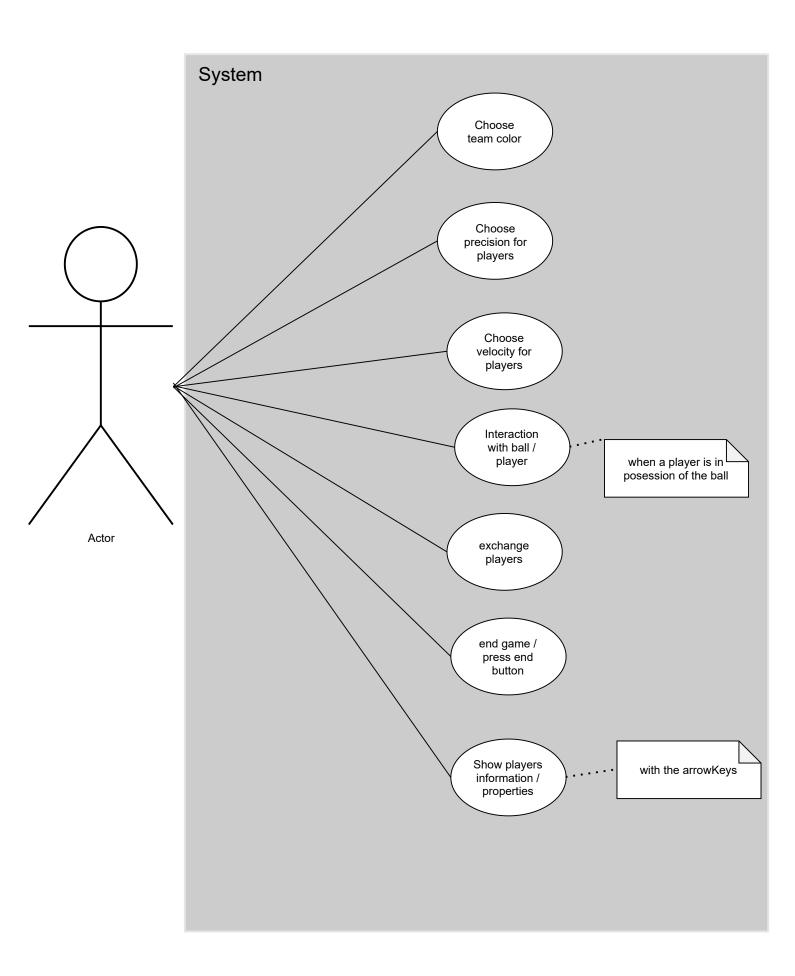
In der rechten Box kann man Auswechslungen durchführen. Sofern man auf einen Spieler klickt, werden die jeweiligen Attribute sowie sein Team angezeigt. Unter "player exchange" kann man einen der Spieler im Dropdown-Menü anklicken, dessen Attribute anschauen und vergleichen. Mit dem Knopf "change" kann man die Auswechslung dann durchführen.

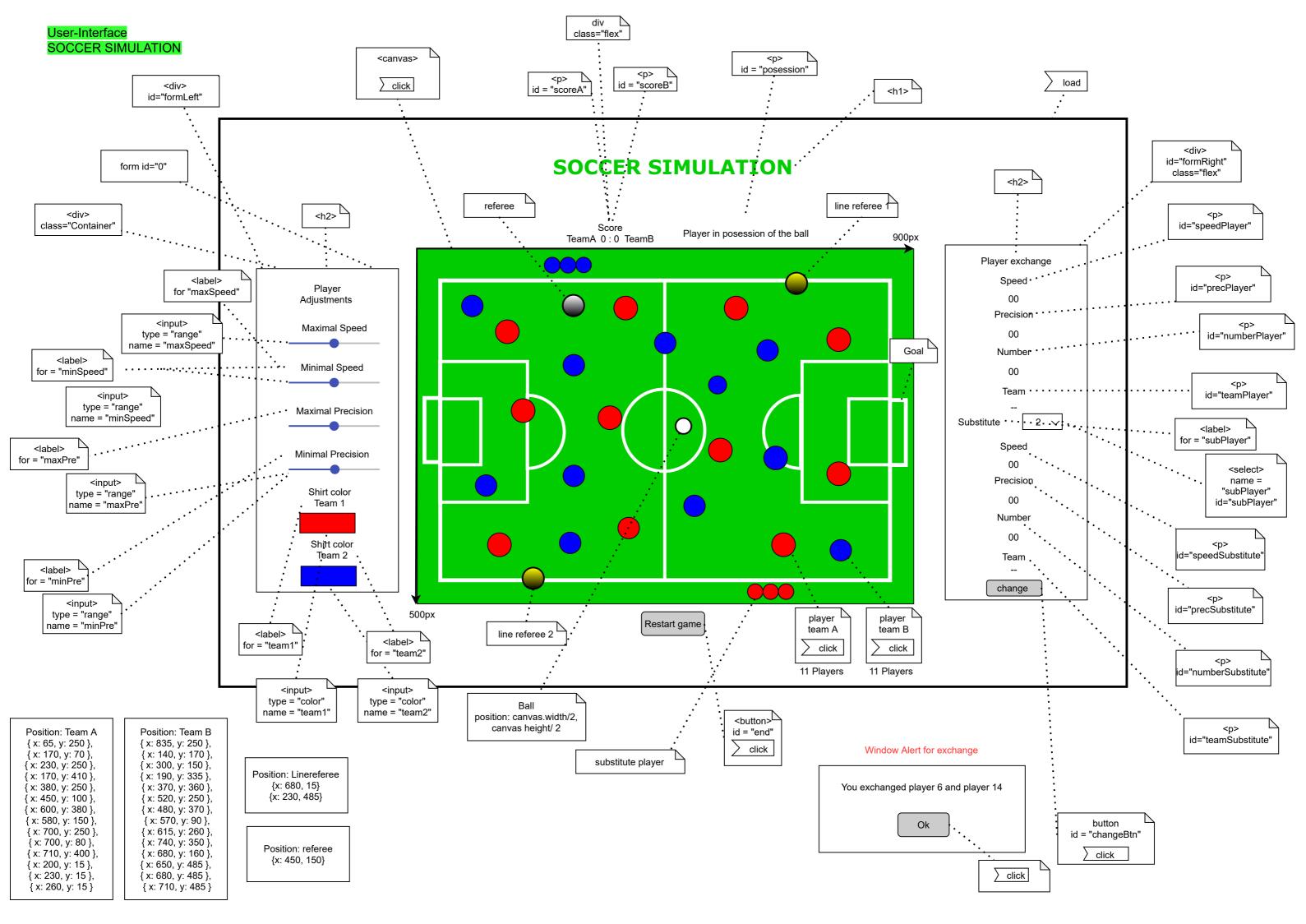
Um Spieler auszuwechseln, drücke Pfeiltasten links und rechts, somit werden Spieler aus dem Team A und B ausgewählt. Um diesen ausgewählten Spieler zu wechseln, wähle einer der drei Spieler aus dem Drop-Down Menü und drücke auf change.

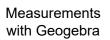
Möchte man das Spiel neu starten, drückt man auf den Knopf "Restart" oberhalb des Scoreboards.

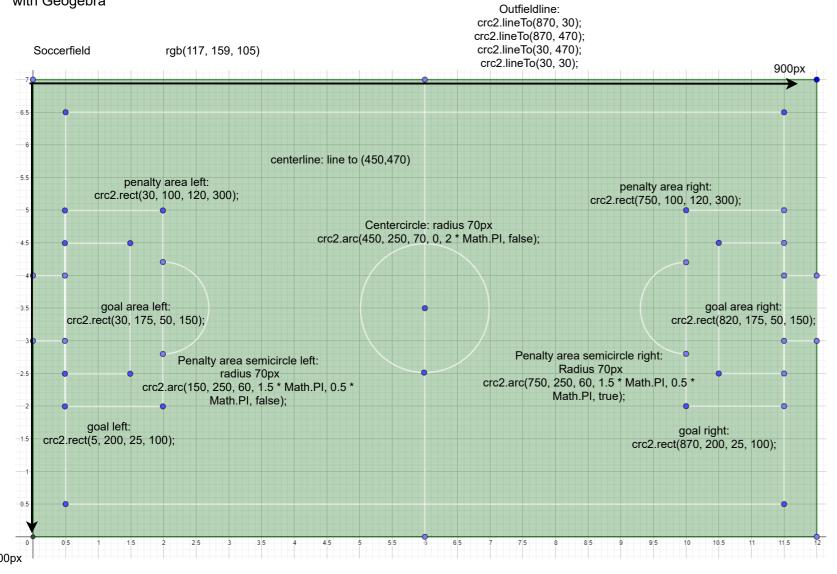
Konzept

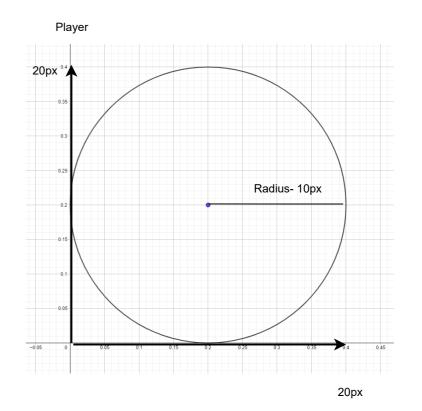
Soccer Simulation: Use-Case-Diagram

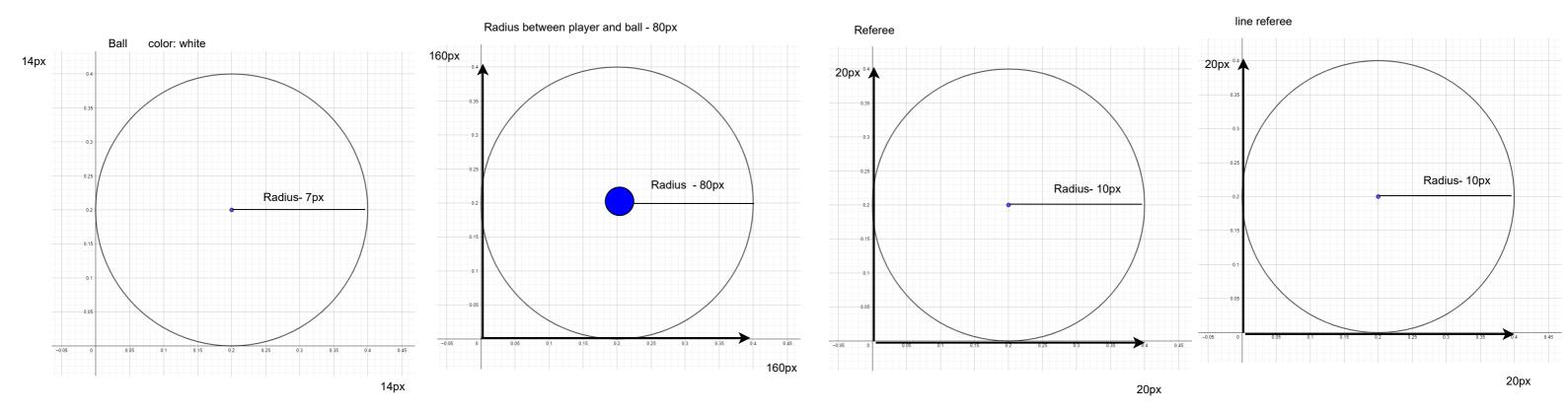




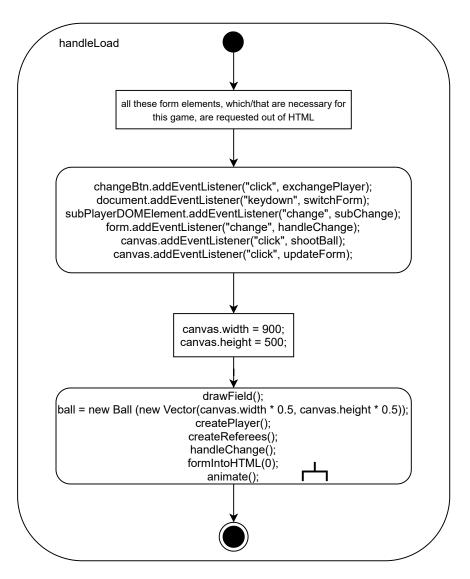


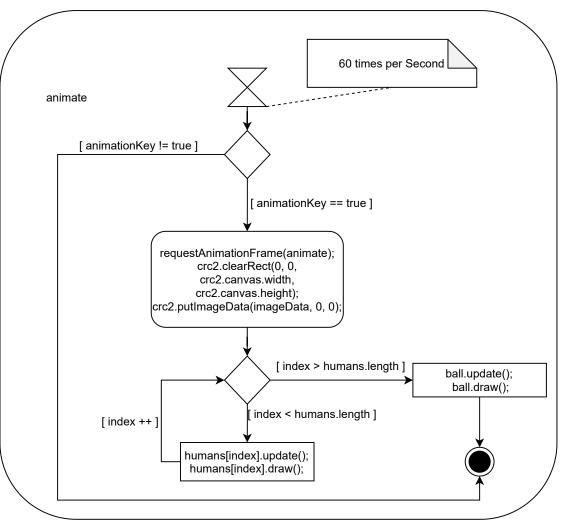


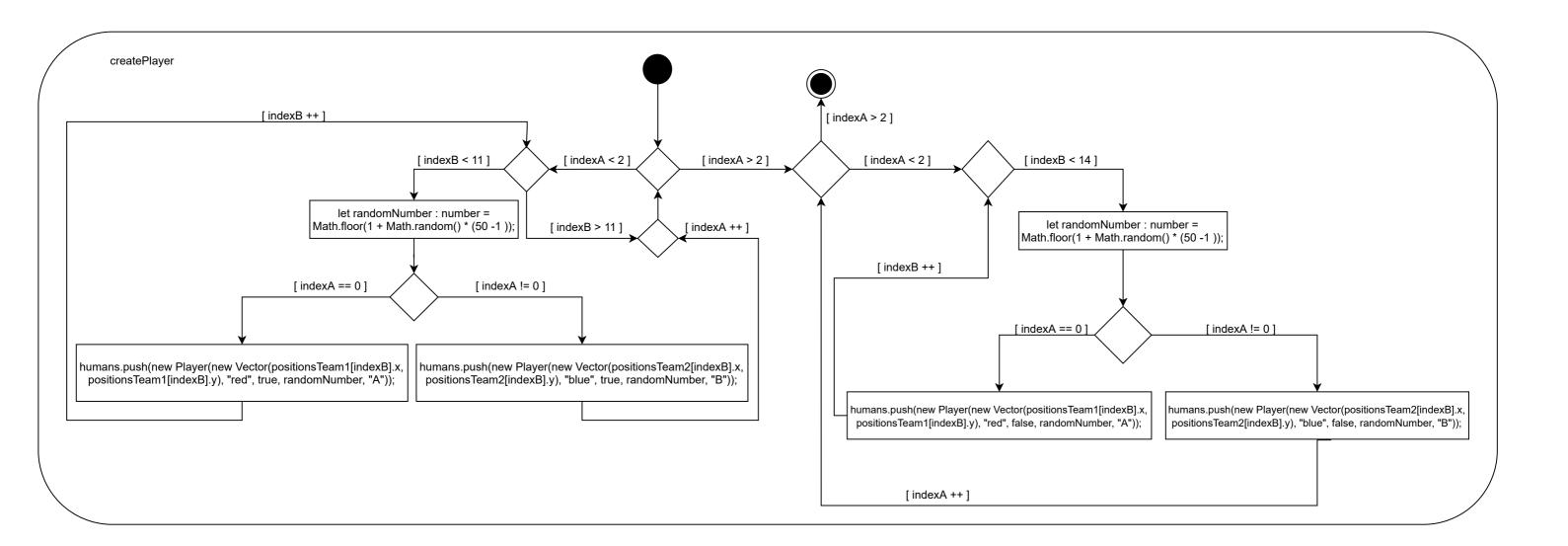


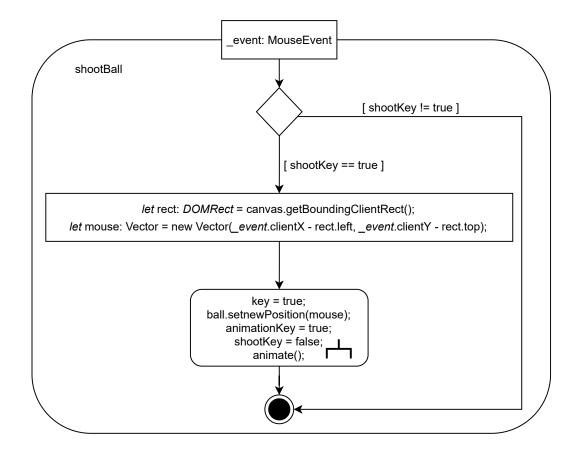


Activity-Diagram: Main let canvas: HTMLCanvasElement = <HTMLCanvasElement> document.querySelector("canvas"); let crc2 CanvasRenderingContext2D = <CanvasRenderingContext2D> canvas.getContext("2d"); export enum Task{}; let imageData: ImageData; change keyDown load let ball: Ball; let key: boolean; let animationKey: boolean = true; switchForm handleLoad H subChange H let shootKey: boolean = false; let humans: Human[] = []; let scoreA: number = 0; let scoreB: number = 0; click change let form: HTMLDivElement; let playerNumberDOMElement: HTMLParagraphElement; shootBall handleChange H let teamDOMElement: HTMLParagraphElement; let teamADOMElement: HTMLButtonElement; let teamBDOMElement: HTMLButtonElement; let speedPlayer: HTMLParagraphElement; let precPlayer: HTMLParagraphElement; let numberPlayer: HTMLParagraphElement; let teamPlayer: HTMLParagraphElement; let speedSub: HTMLParagraphElement; let precSub: HTMLParagraphElement; let numberSub: HTMLParagraphElement; let teamSub: HTMLParagraphElement; let subPlayerDOMElement: HTMLSelectElement; let scoreADOMElement: HTMLElement; let scoreBDOMElement: HTMLElement; let changeBtn: HTMLButtonElement; let scoreA: number = 0; let scoreB: number = 0; let posession: HTMLParagraphElement; createReferees players.push(new Referee(new Vector(450, 150), "white")); players.push(new Linereferee(new Vector(680, 15), "pink")); players.push(new Linereferee(new Vector(230, 485), "pink"));

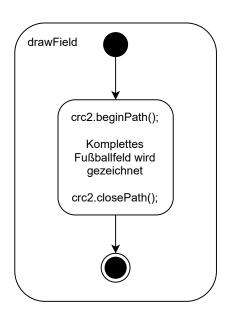




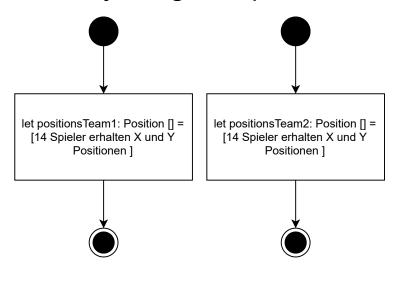




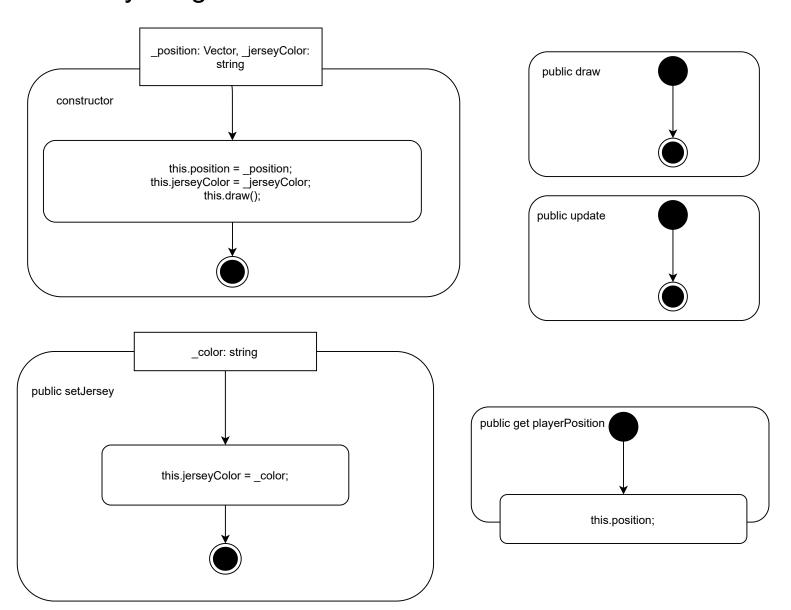
Activity-Diagram: footballField



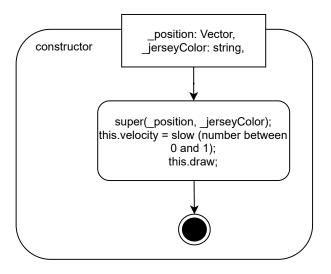
Activity-Diagram: position

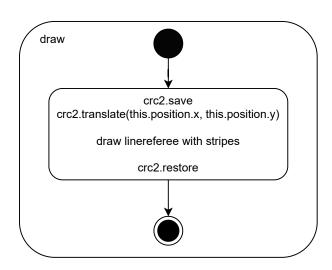


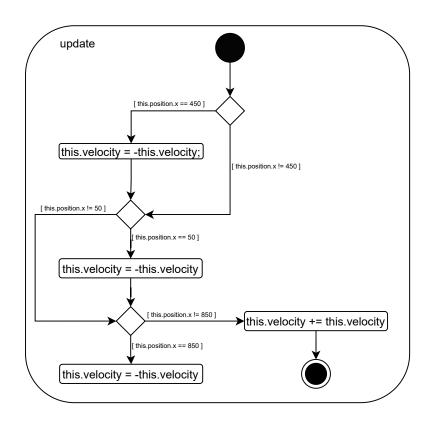
Activity-Diagram: human



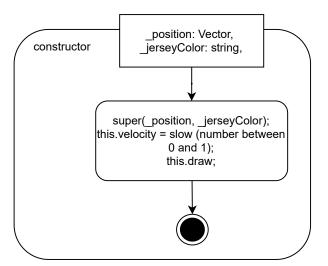
Activity-Diagram: lineReferee

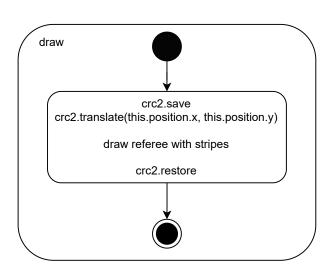


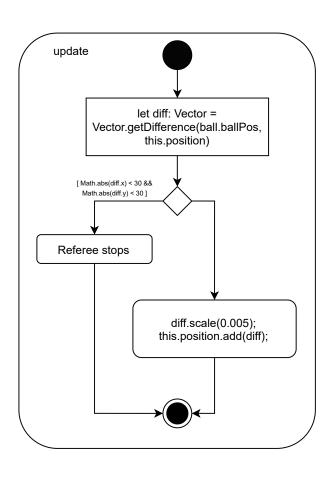




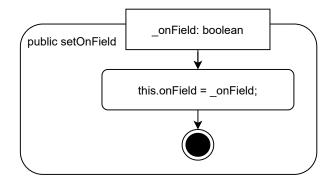
Activity-Diagram: Referee

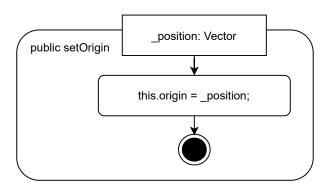


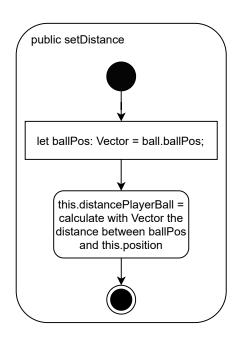


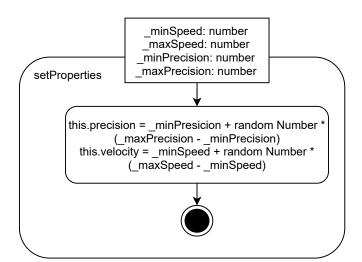


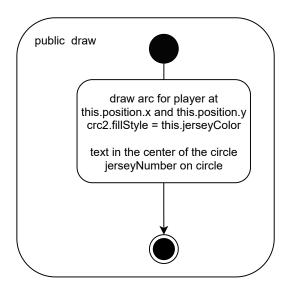
Activity-Diagram: Player position: Vector, _jerseyColor: string, _onField: boolean, jerseyNumber: number, constructor _team: string, task: Task = Task.lookForBall super(_position, _jerseyColor); origin: Vector; this.onField = _onField; this.velocity = 0.5; precision: number; this.jerseyNumber = _jerseyNumber; this.origin = this.position (origin); radius: number = 80; jerseyNumber: number; distancePlayerBall: number; this.team = _team; onField: boolean; team: string; newPosition: Vector; public get jerseyNumberPlayer public get distance public get playerSpeed this.jerseyNumber; this.velocity; this.distancePlayerBall; public get playerPrecision public get playerOnField public get playerTeam return this.precision; return this.onField; return this.team; public get playerOrigin _position: Vector private movePlayer let playerDifference: Vector = return this.origin; Vector.getDifference(_position, this.position); let playerDifference: Vector = Vector.getDifference(_position, this.position); let ratio: number = this.velocity / playerDistance playerDifference.scale(ratio); this.position.add(playerDifference); this.draw;

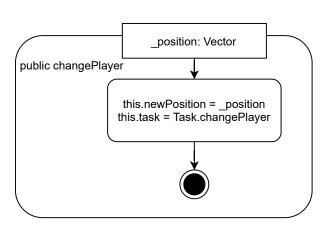


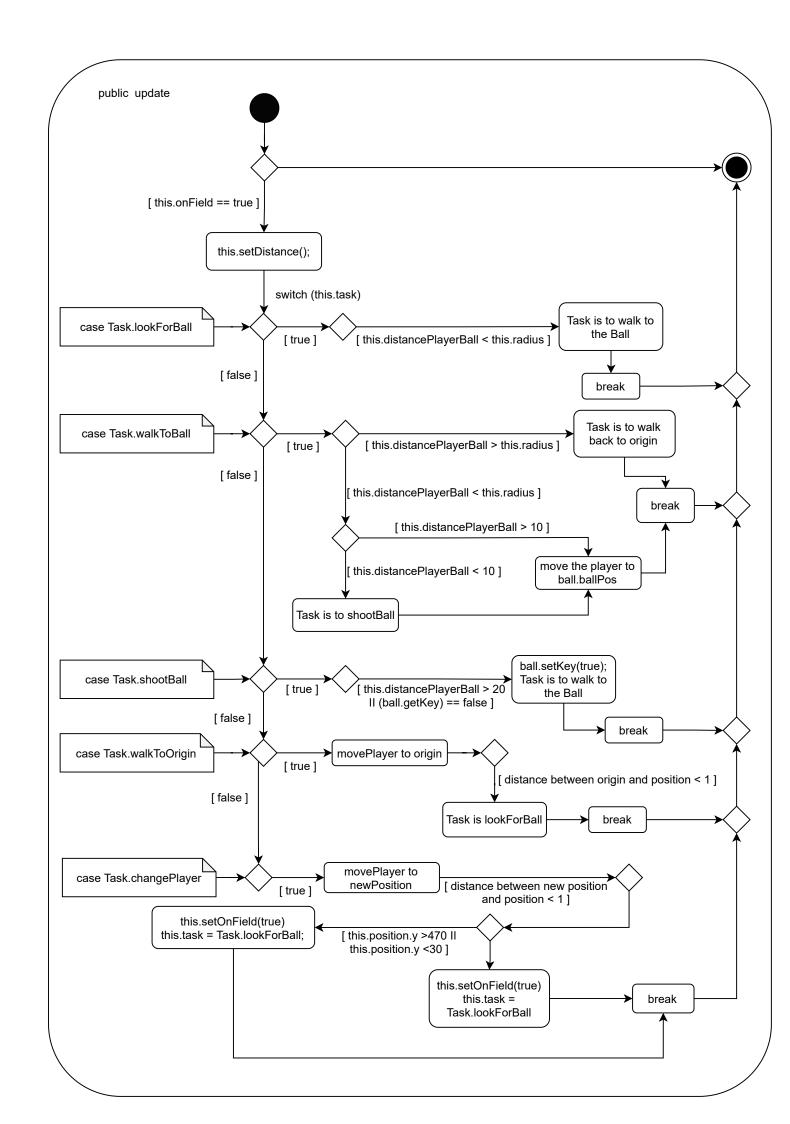




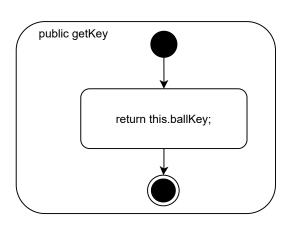


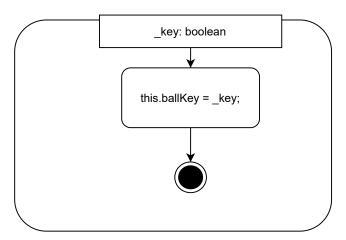


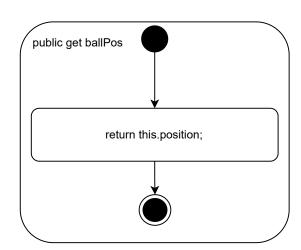


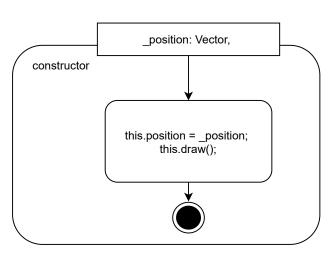


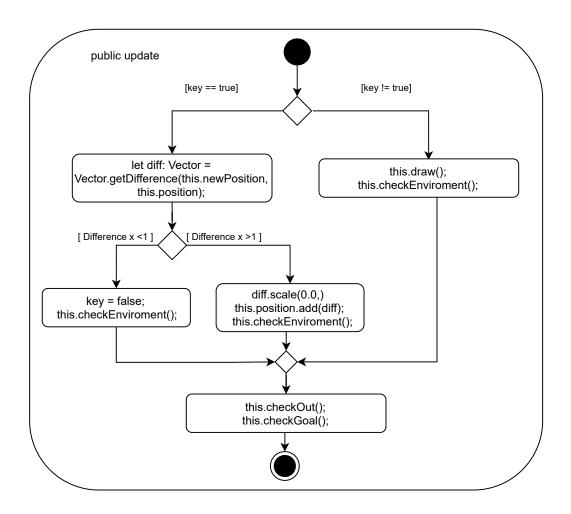
Activity-Diagram: Ball

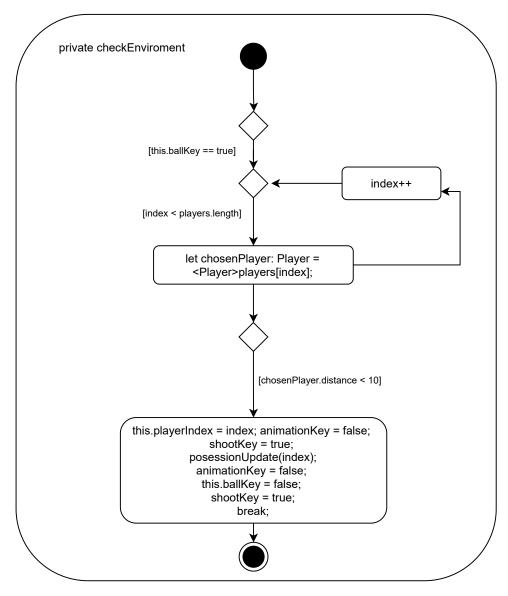


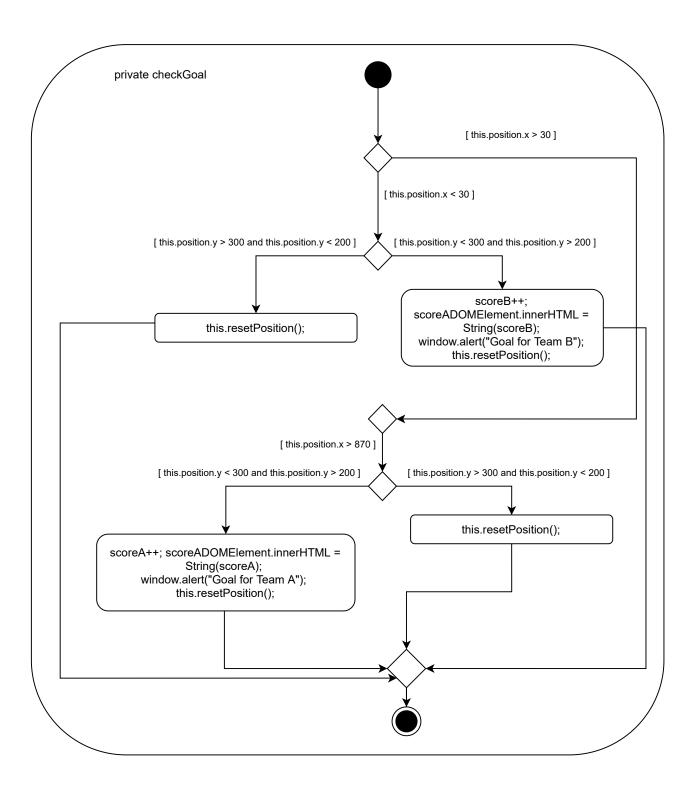


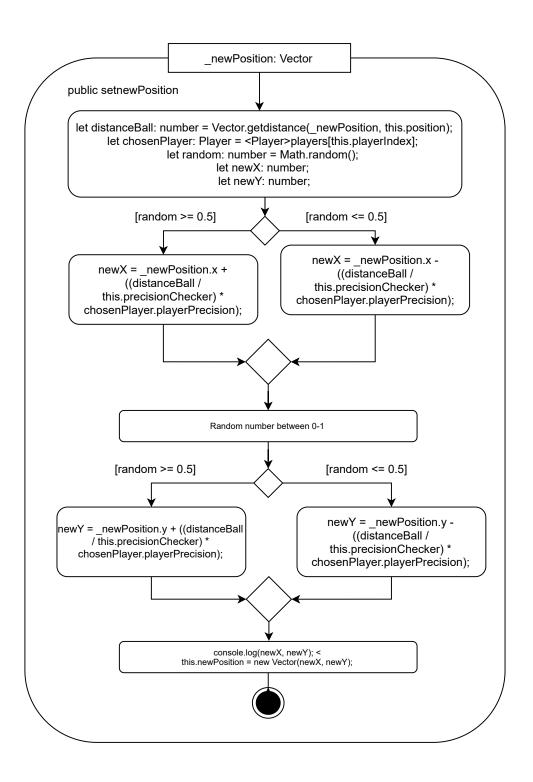




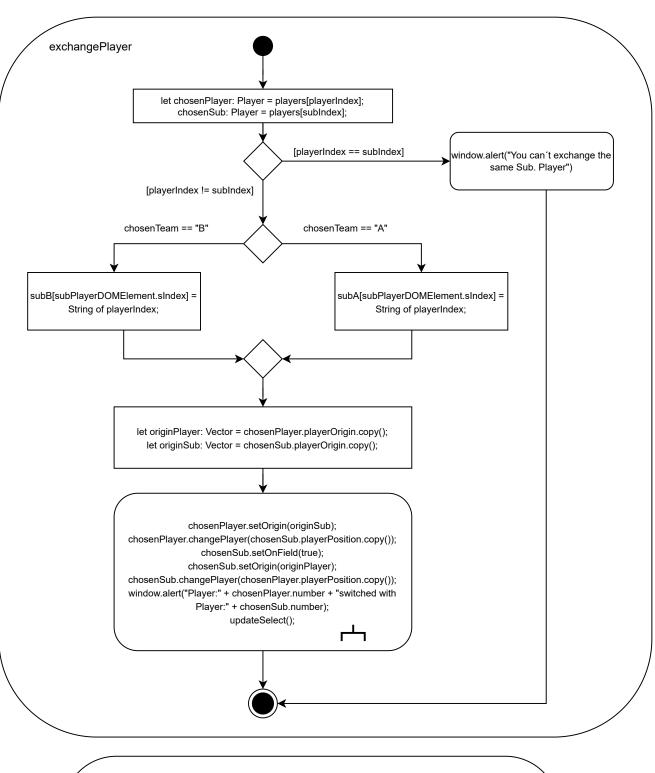


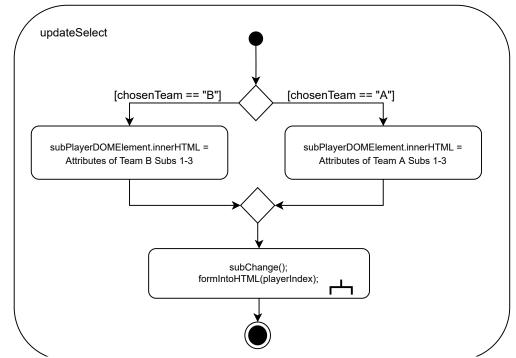


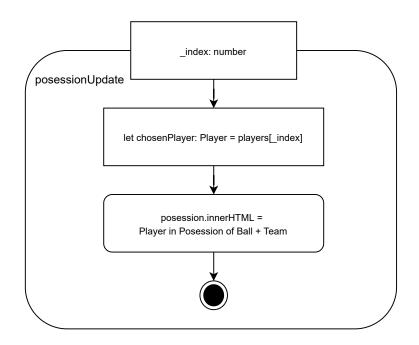


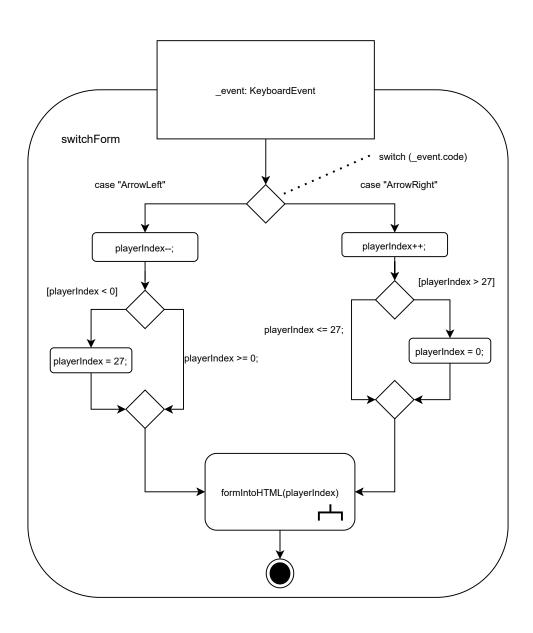


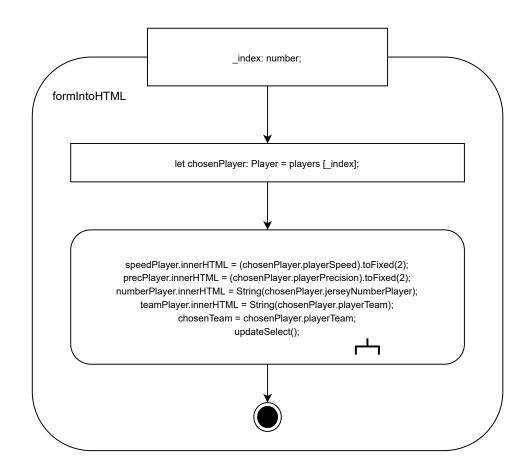
Activity-Diagram:-Forms let valuesGlobal: string []; let playerIndex: number = 0; let subIndex: number; let chosenTeam: string; let subA: string: [] = [22, 23, 24]; let subB: string [] = [25, 26, 27]; handleChange let formData: FormData = new FormData (document.forms[0]); valuesGlobal = []; push Data in valuesGlobal [index = 0][formData++] (durchläuft alle FormDatas) [index++] [index >= 28] [index < 28] let chosenPlayer: Player = players[index]; setProperties of chosenPlayer with valuesGlobal [index >= 25] set Jersey of set Jersey of [index >= 22] [index < 22] [index < 11] players[index] with players[index] with valuesGlobal[5]; valuesGlobal[4]; players[index].draw(); players[index].draw(); [index < 25] index >= 11 set Jersey of set Jersey of players[index] players[index] with with valuesGlobal[5]; valuesGlobal[4]; players[index].draw(); players[index].draw();

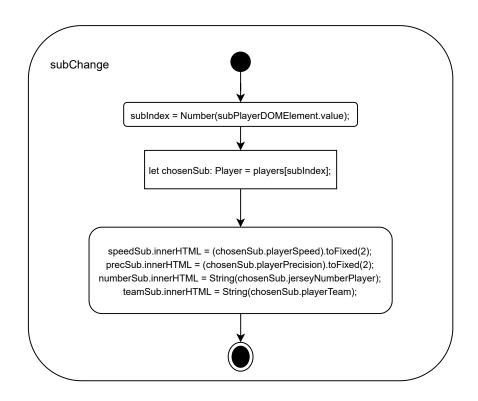




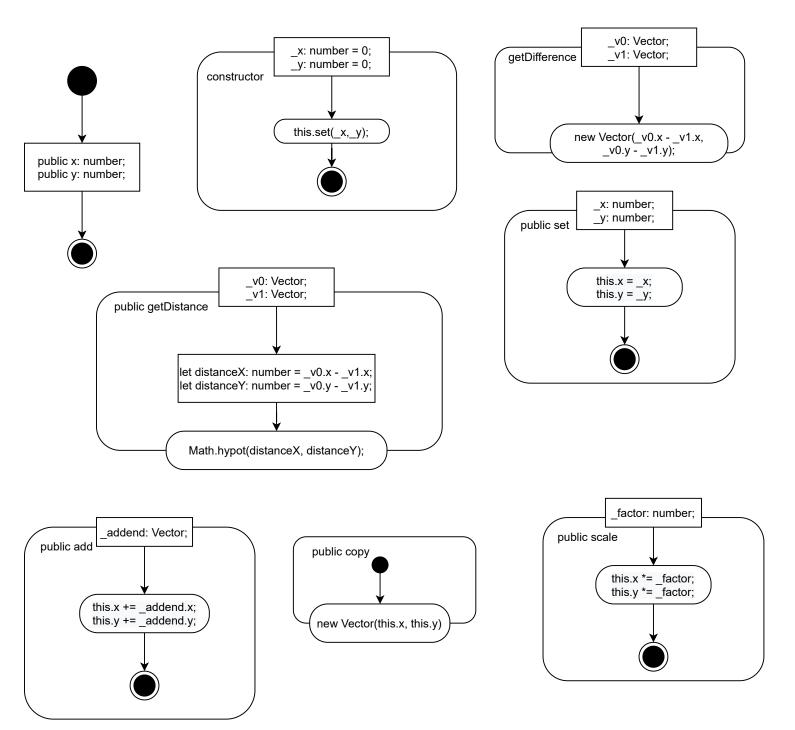


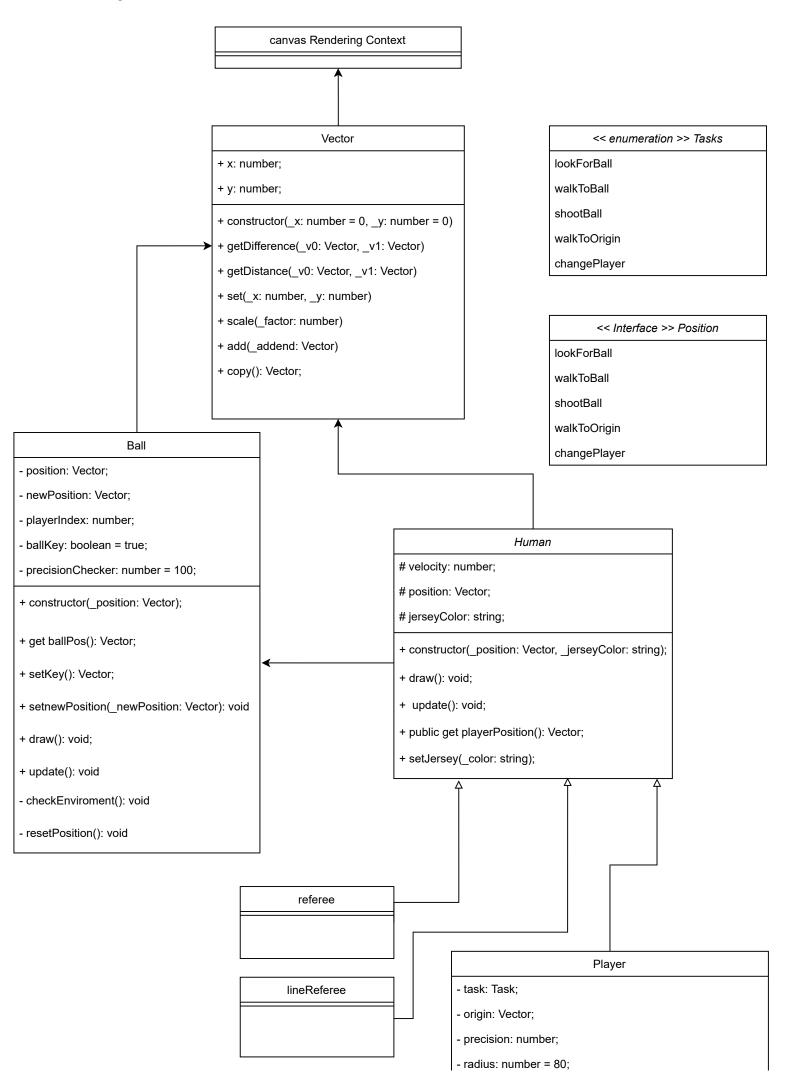






Activity Diagram Vector





- jerseyNumber: number;
 - onField: boolean;
 - + constructor(_position: Vector, _onField: boolean, _jerseyColor: string, _team: string)
 - + get jerseyNumberPlayer(): number
 - + get playerSpeed(): number

- distancePlayerBall: 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