

# Soccer Training Simulation - Complete Documentation

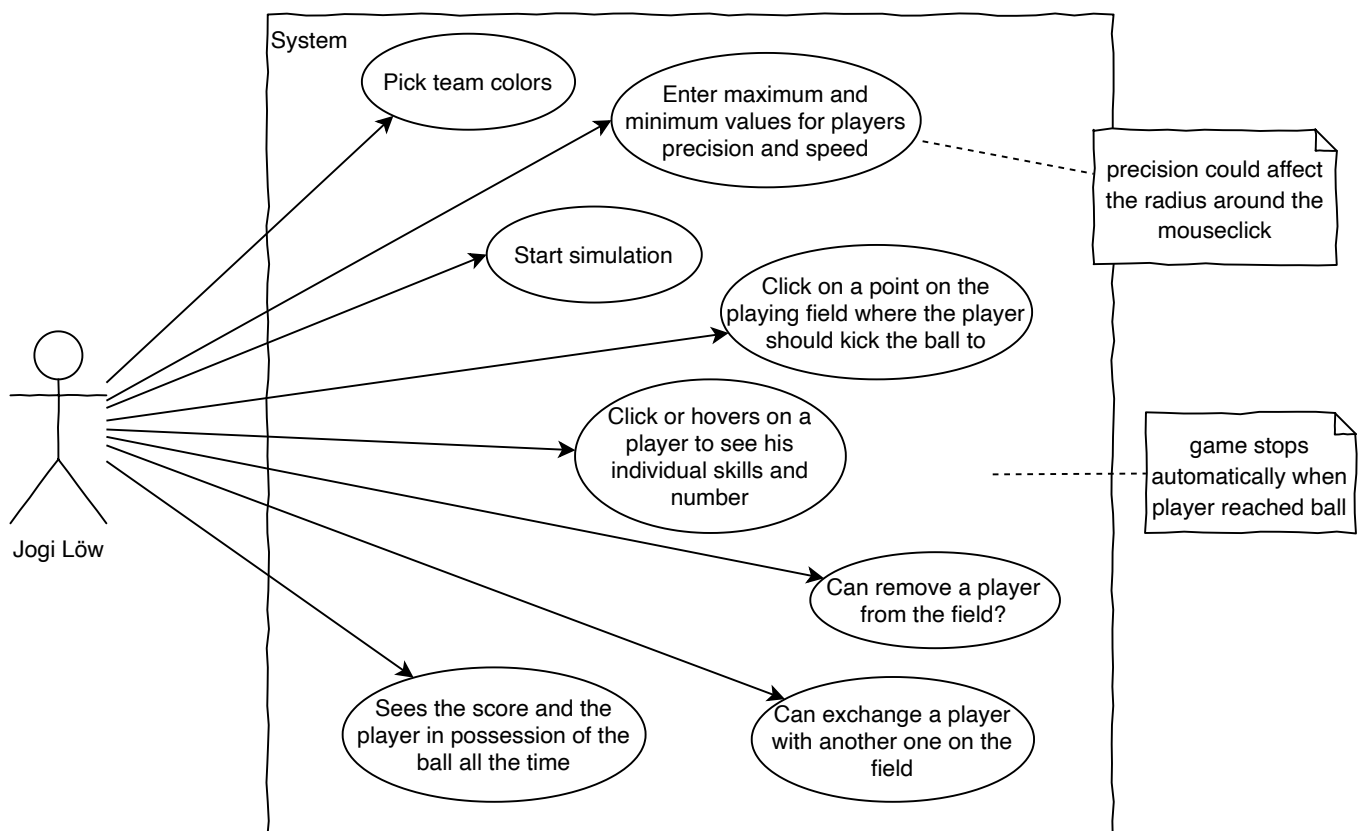
The whole production of this application was developed in collaboration of Mona Stingl and Hannah Dürr with equal contribution of each team member to the final result.

Date of completion: 19/07/2021

## Content

- Use Case Diagram
- User Interface Scribbles
- Class Diagram
- Activity Diagram
- Class Methods

## Soccer Simulator Use Case Diagram



# Soccer Simulator User Interface Scribble

Start screen

## Welcome!

Here you can set the preferences for your soccer simulation.

### Players Minimum Speed

slow  medium

### Players Maximum Speed

medium  fast

### Players Minimum Precision

advanced  amateur

### Players Maximum Precision

pro  advanced

### Team Colors

Team A

Team B

Kick Off

Whole <div> : display *none* when user clicks on "Kick Off"

<form> input elements

<form> color picker

save input values for the simulation

Display with div and span elements

display and hide instructions on click

Simulator screen

Score: 0 : 0

In possession of the ball: Player No 7

field size: 800px & 500px

html canvas element

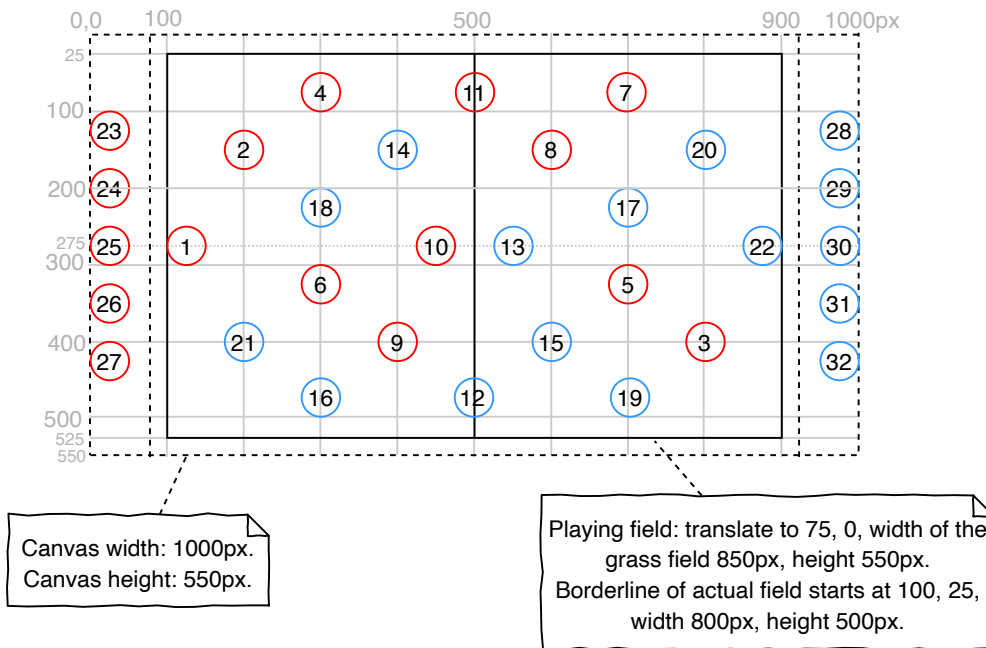
? ↺  
restart simulation, <a>-link to simulation document

with altkey and click the player can be dragged to exchange him with another

Number: ? | Speed: ? | Precision: ?

with shiftkey and click the player information can be shown

# Scribble for exact canvas values



```

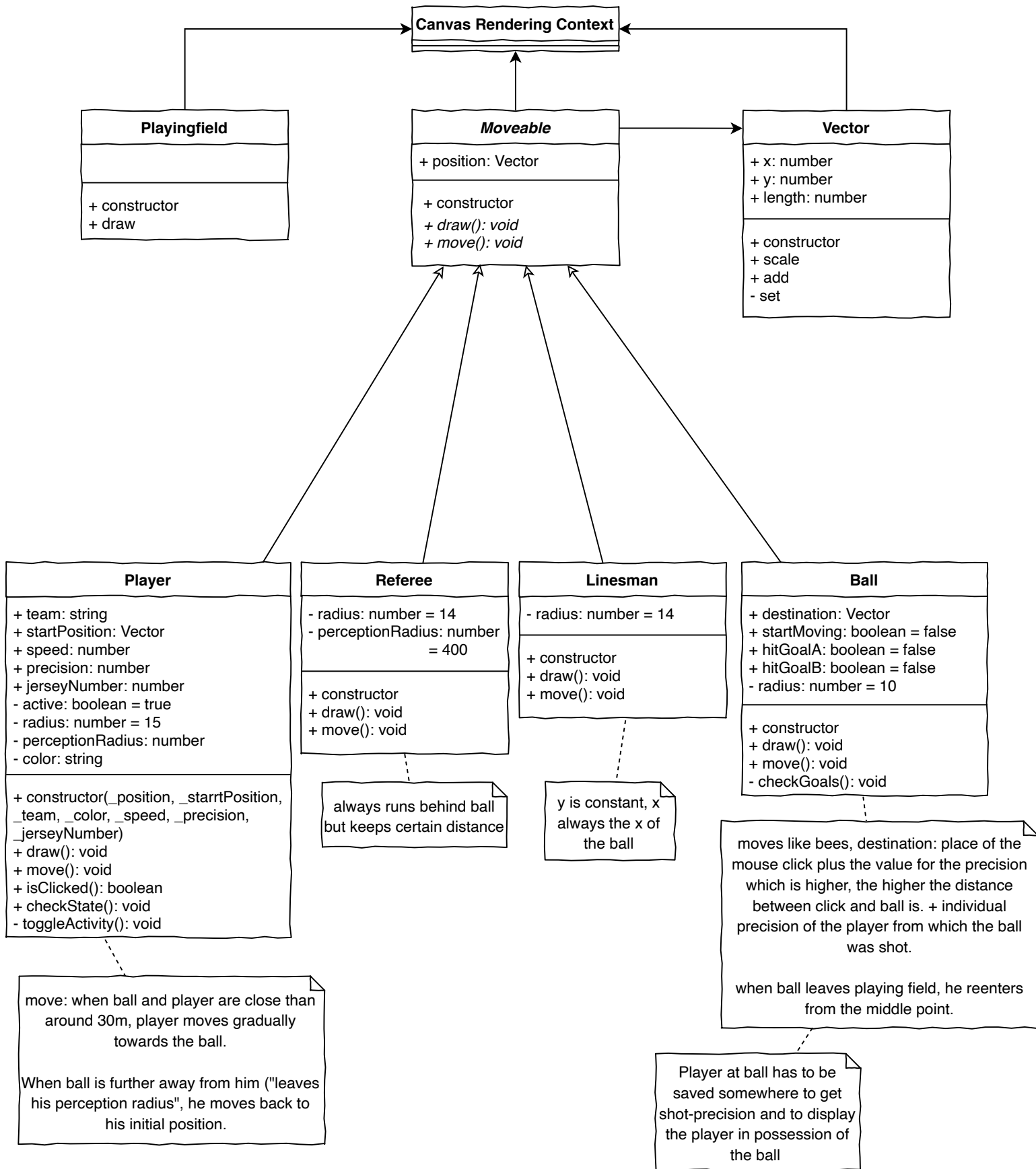
playerInformation [
  {x: 135, y: 275, team: "A"},
  {x: 180, y: 100, team: "A"},
  {x: 180, y: 450, team: "A"},
  {x: 300, y: 75, team: "A"},
  {x: 300, y: 225, team: "A"},
  {x: 300, y: 325, team: "A"},
  {x: 300, y: 475, team: "A"},
  {x: 400, y: 150, team: "A"},
  {x: 400, y: 400, team: "A"},
  {x: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

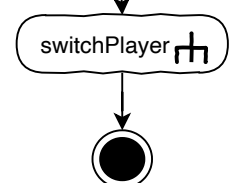
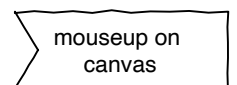
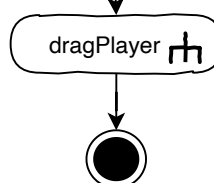
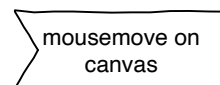
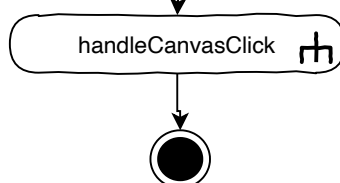
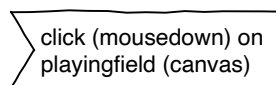
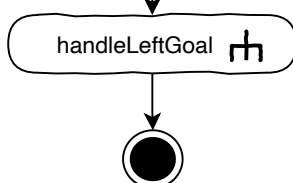
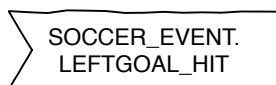
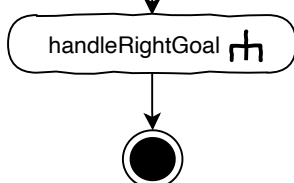
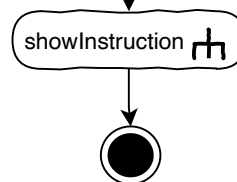
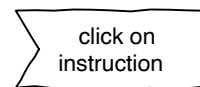
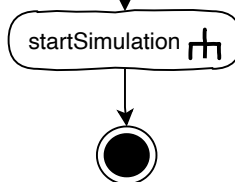
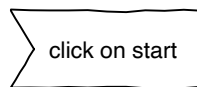
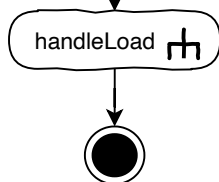
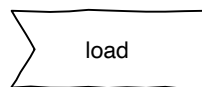
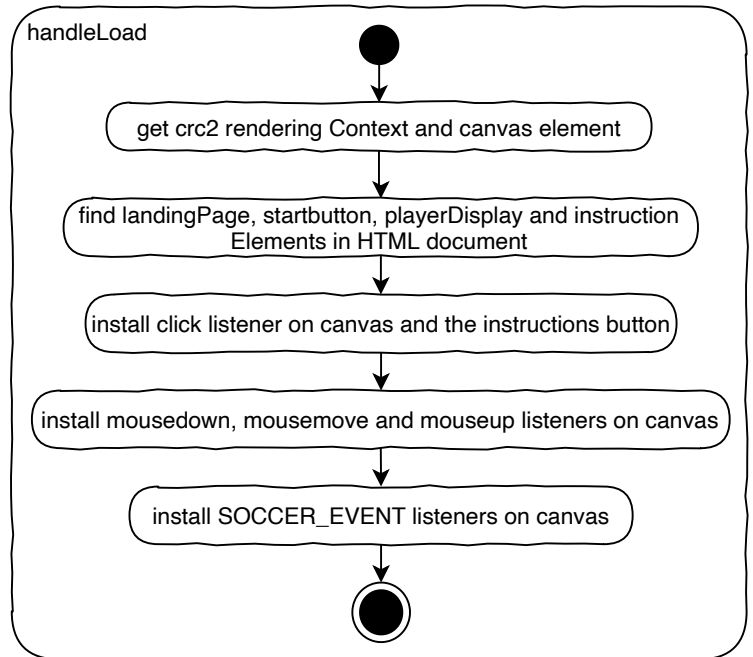
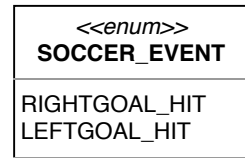
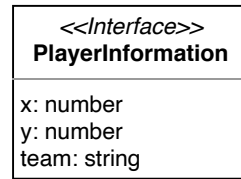
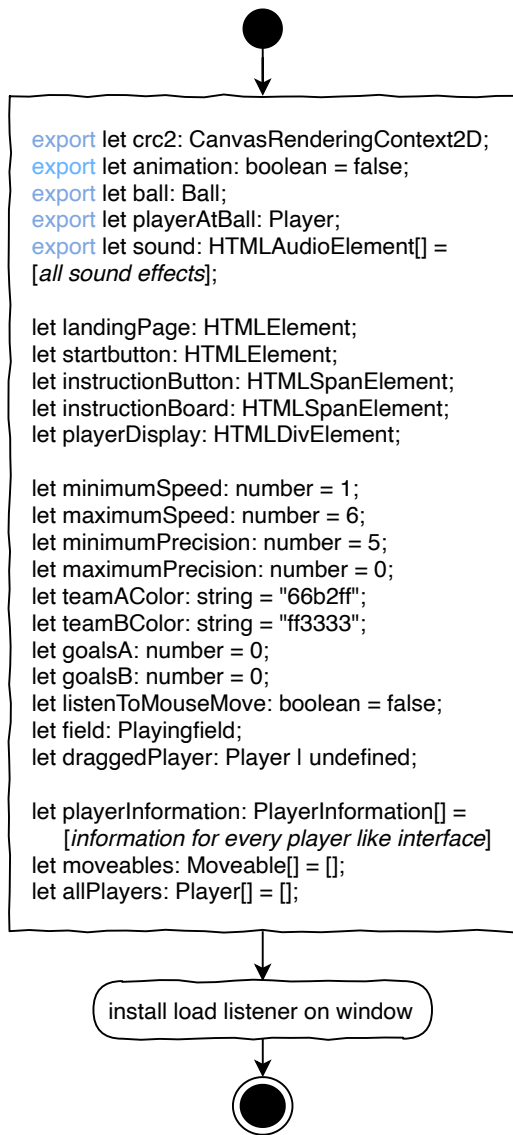
  {x: 25, y: 125, team: "A"},
  {x: 25, y: 200, team: "A"},
  {x: 25, y: 275, team: "A"},
  {x: 25, y: 350, team: "A"},
  {x: 25, y: 425, team: "A"},

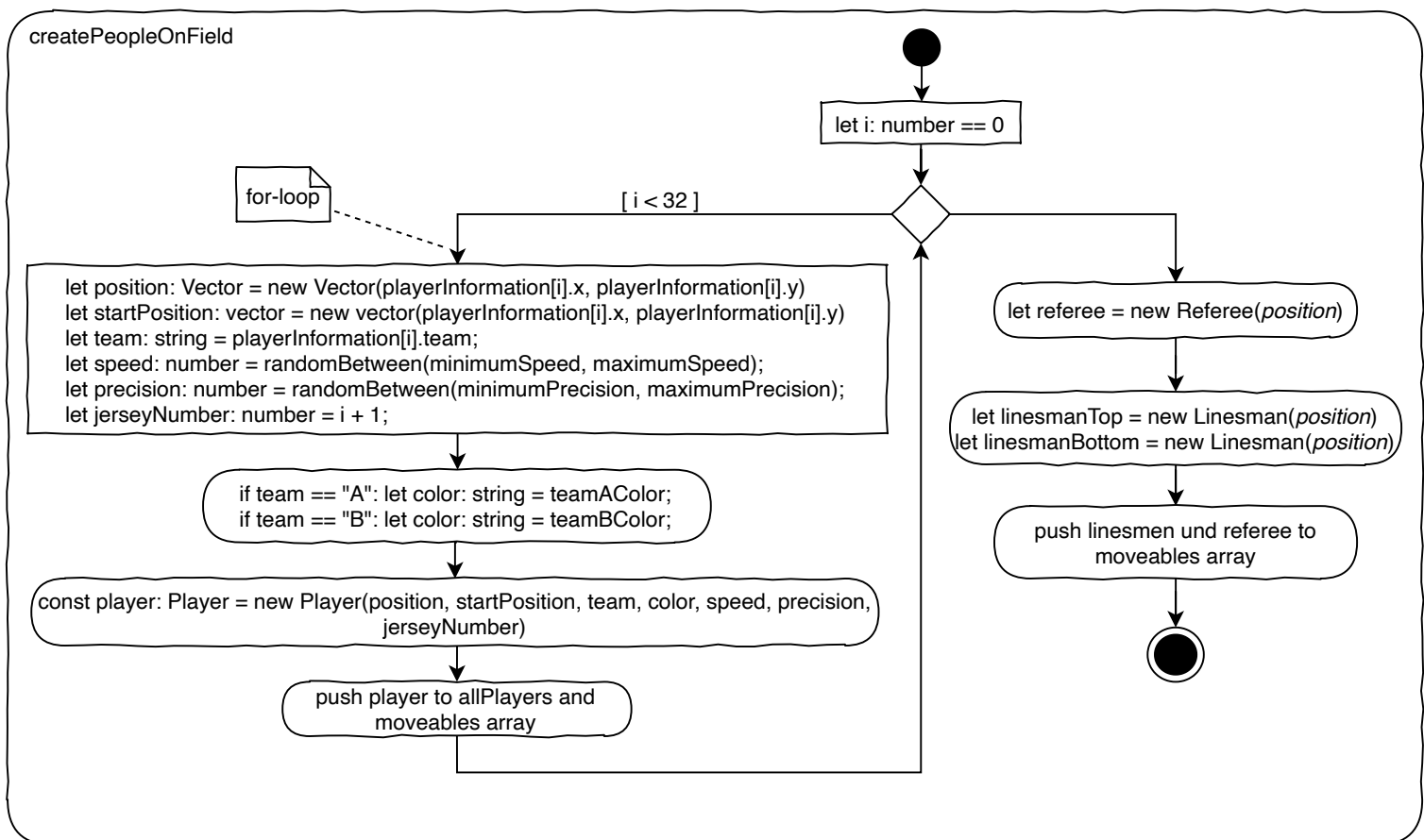
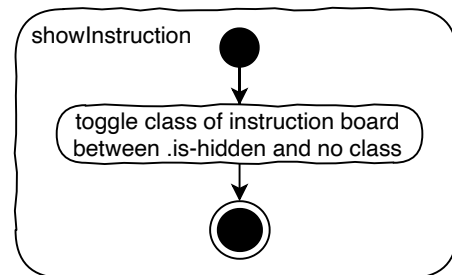
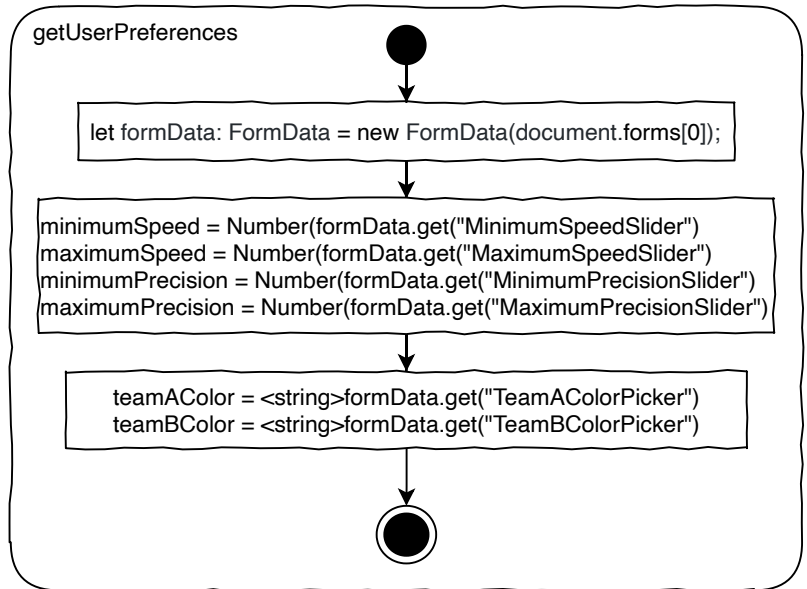
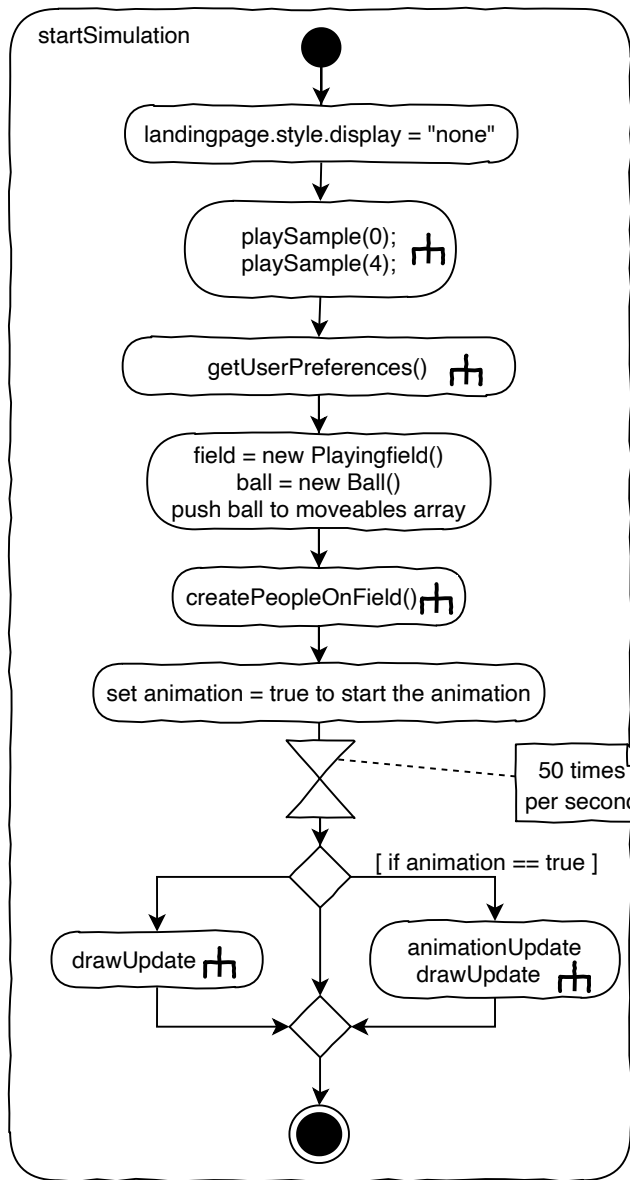
  {x: 975, y: 125, team: "B"},
  {x: 975, y: 200, team: "B"},
  {x: 975, y: 275, team: "B"},
  {x: 975, y: 350, team: "B"},
  {x: 975, y: 425, team: "B"},
]
    
```

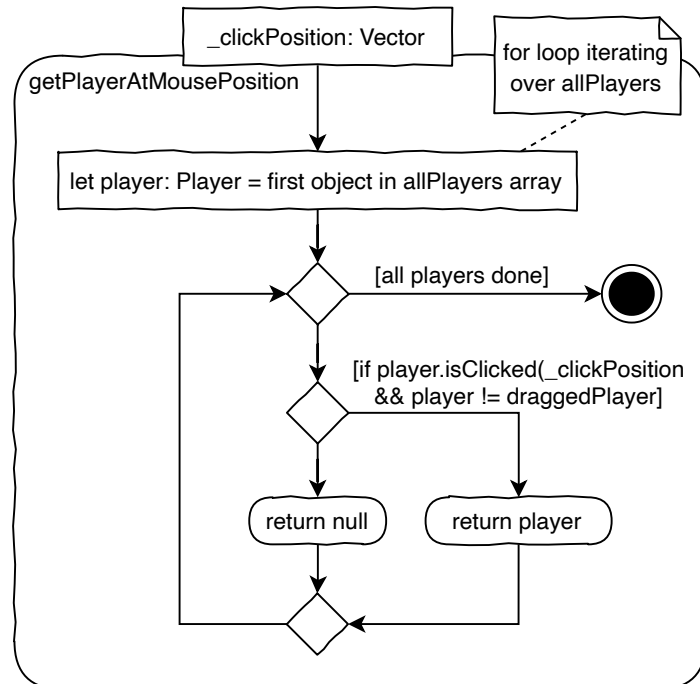
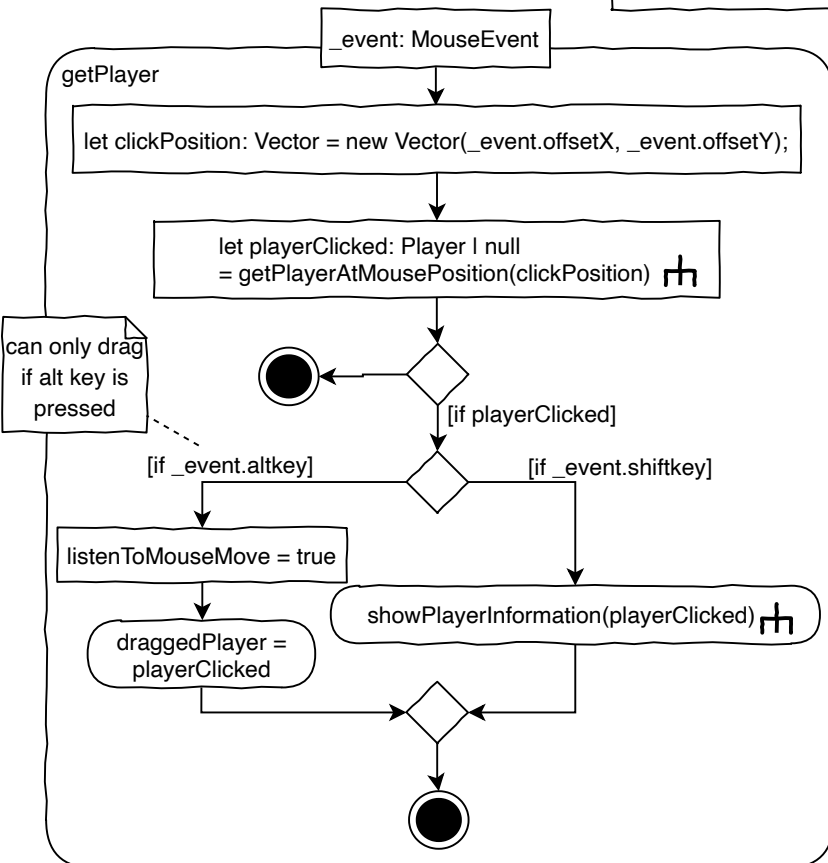
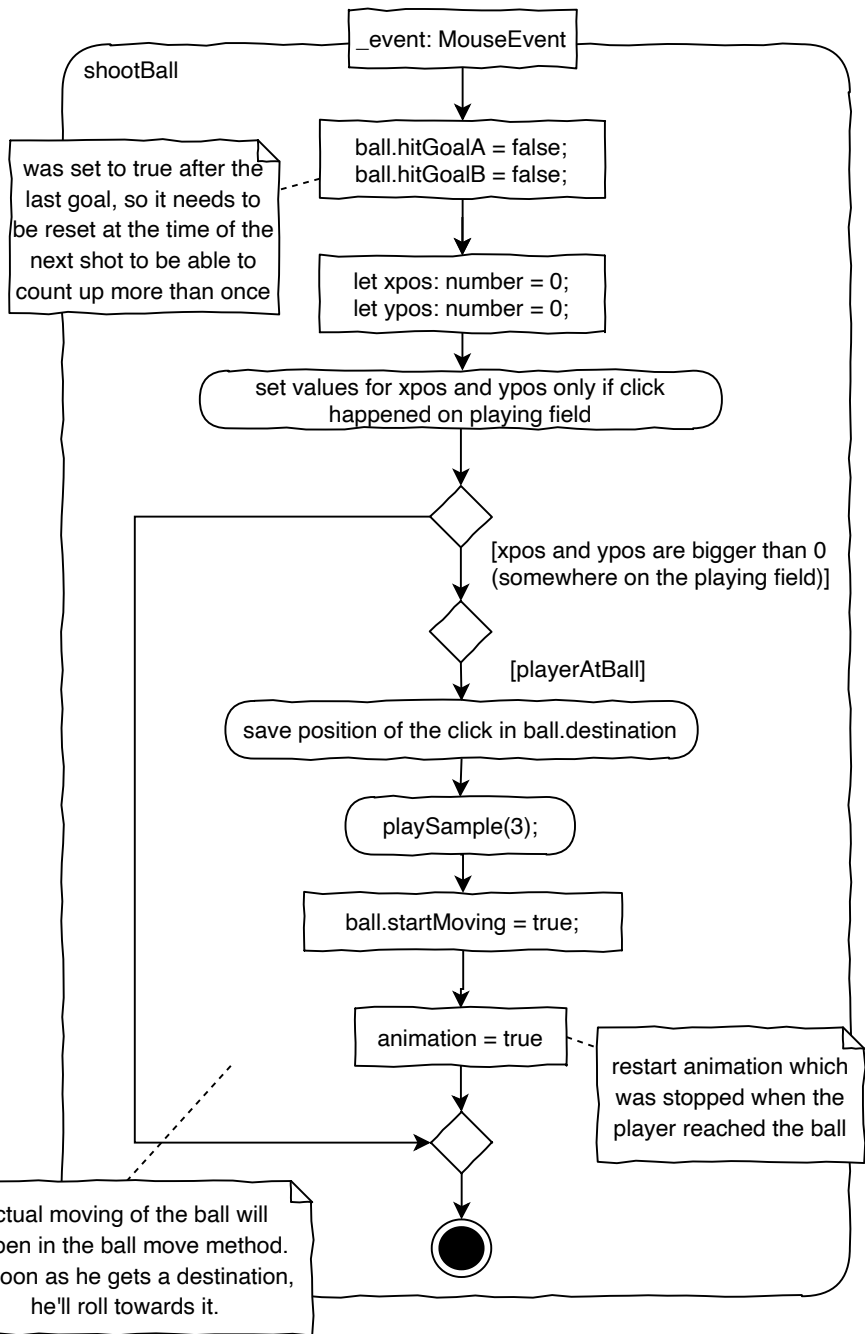
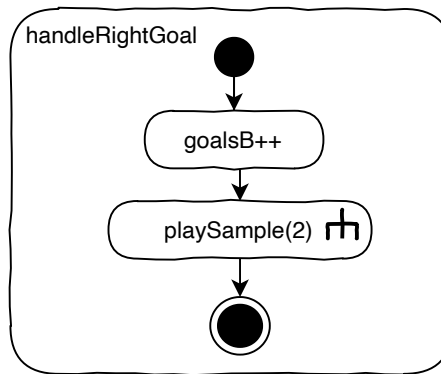
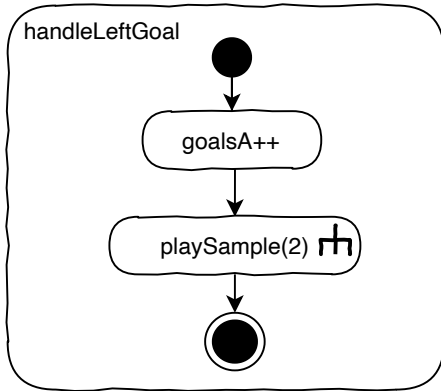
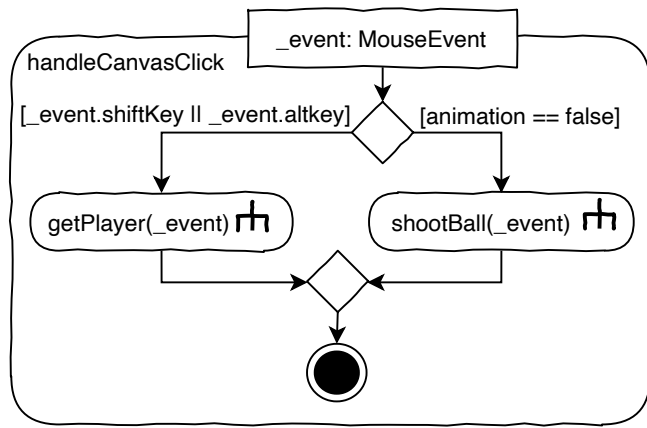
# Soccer Simulator Class Diagram

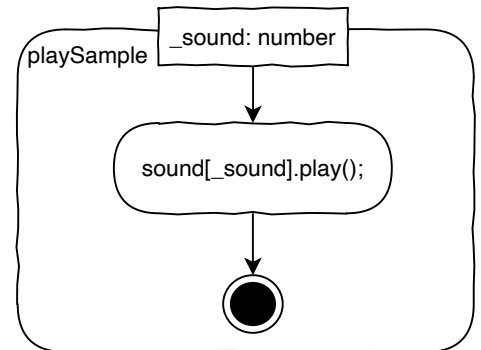
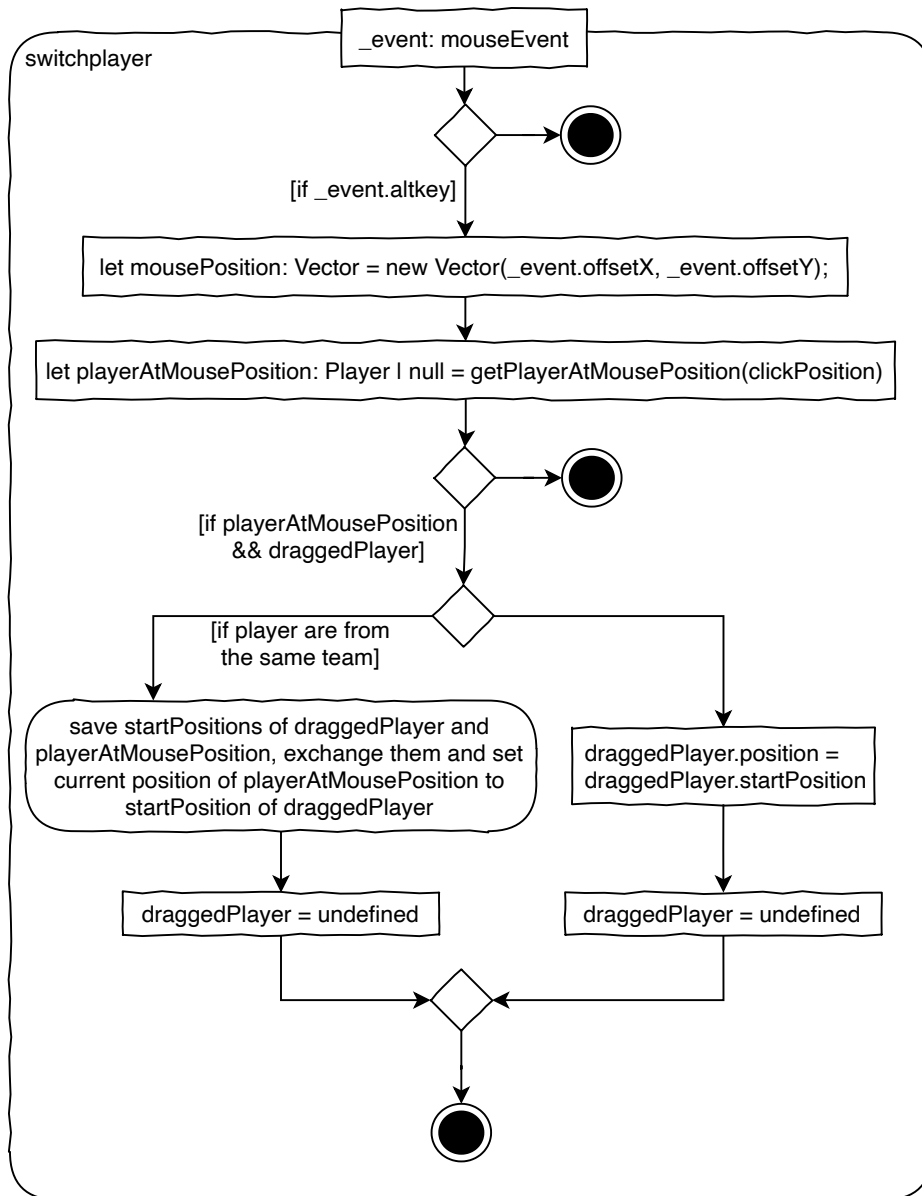
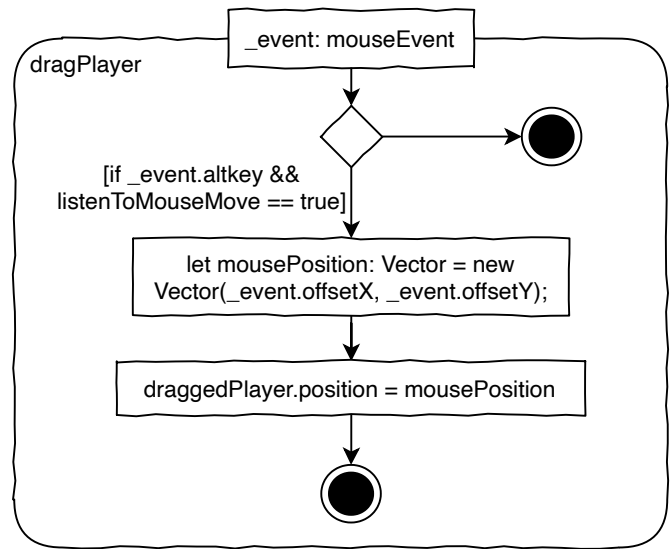
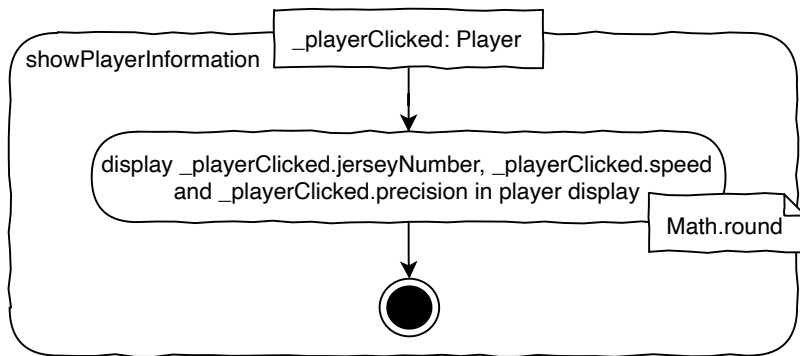


# Soccer Simulator Activity Diagram





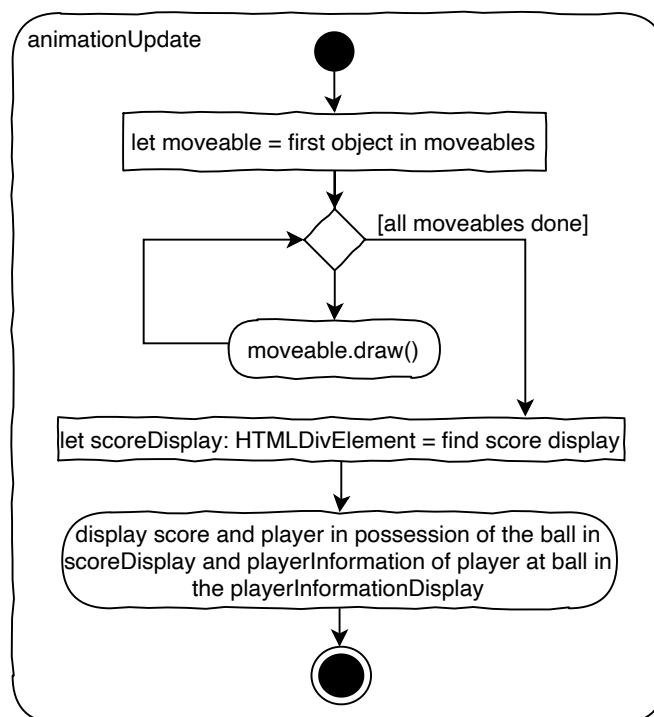
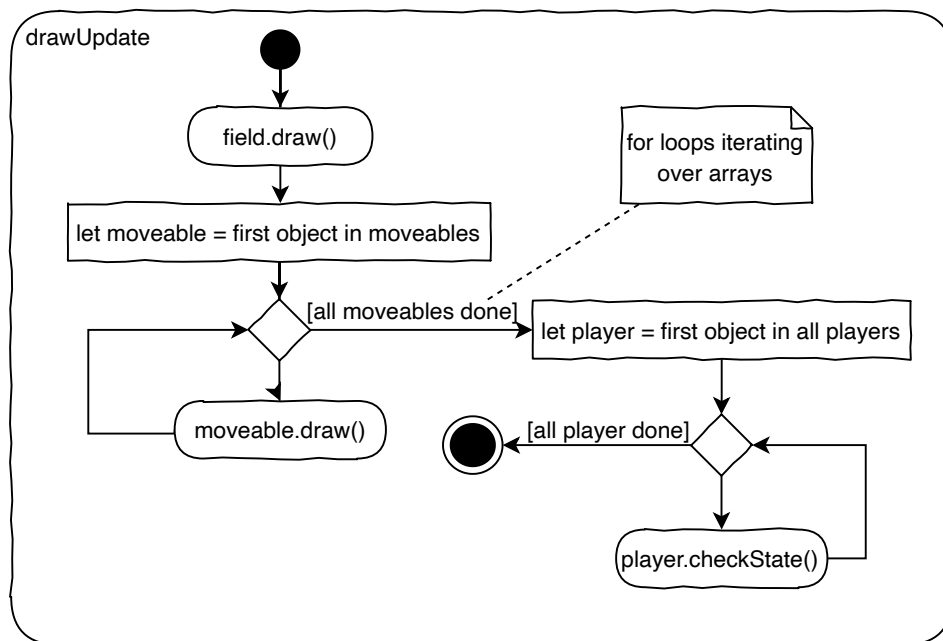




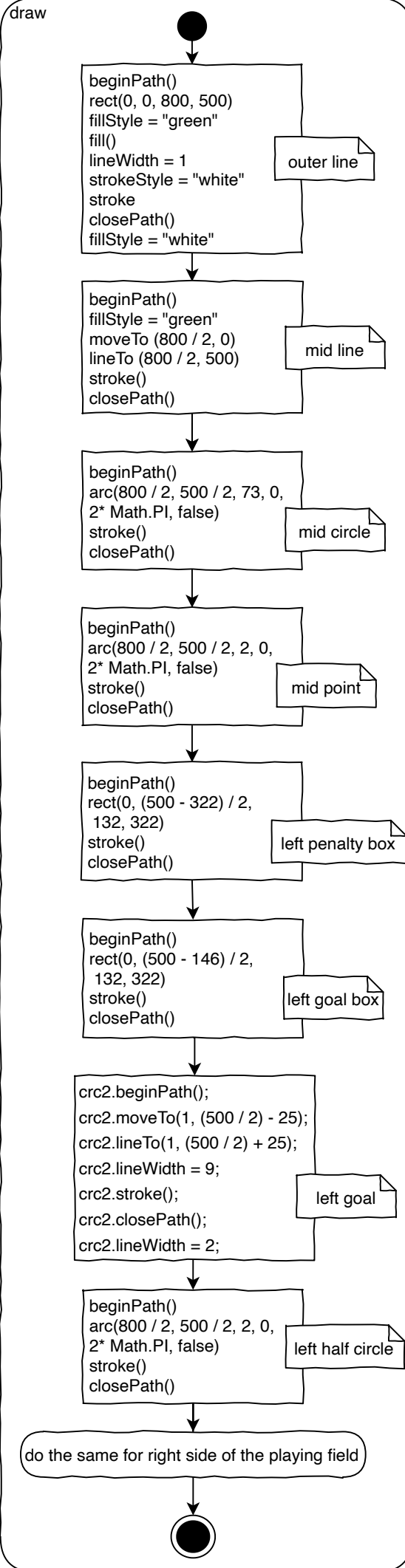
when draggedPlayer is overlapping with a field player at releasing the mouse, they switch their positions.

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

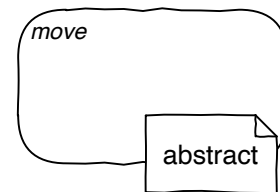
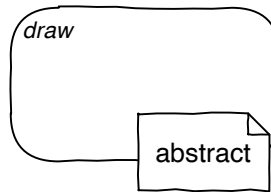
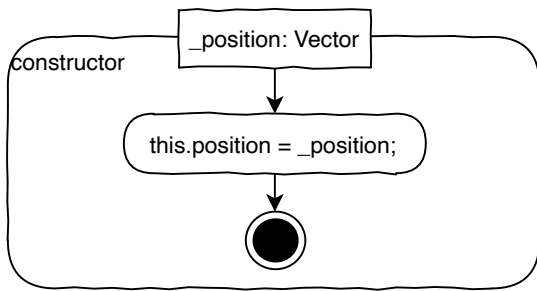




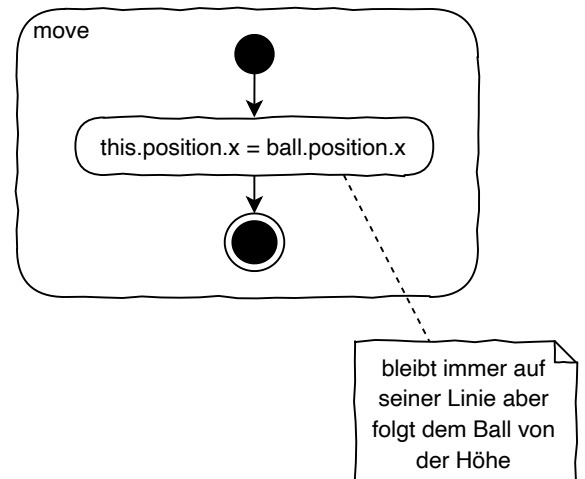
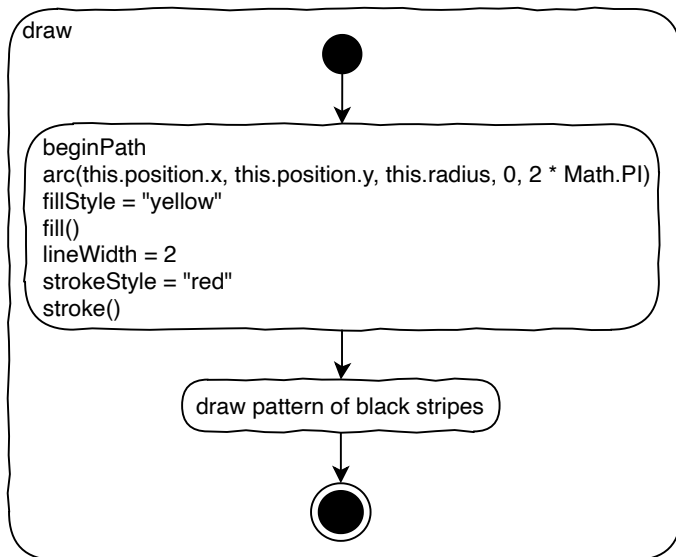
# Playing Field Methods



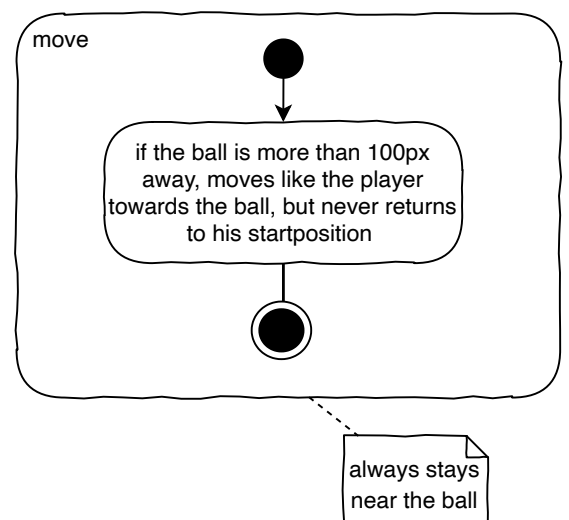
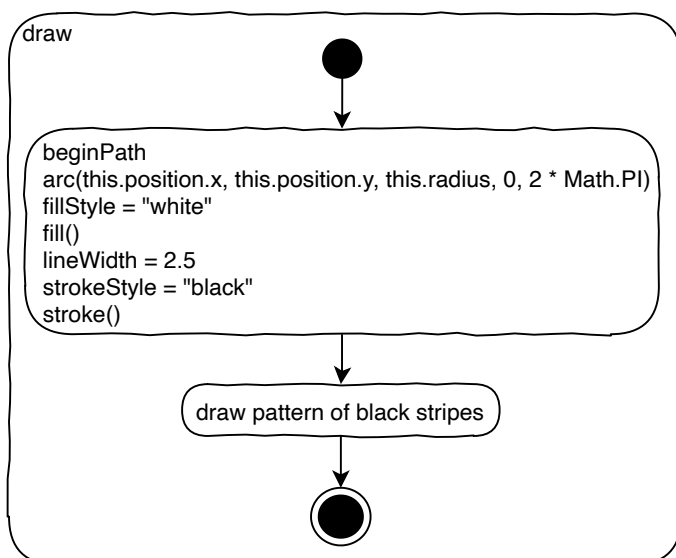
## Moveable Methods



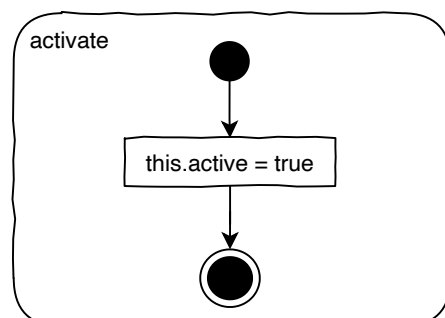
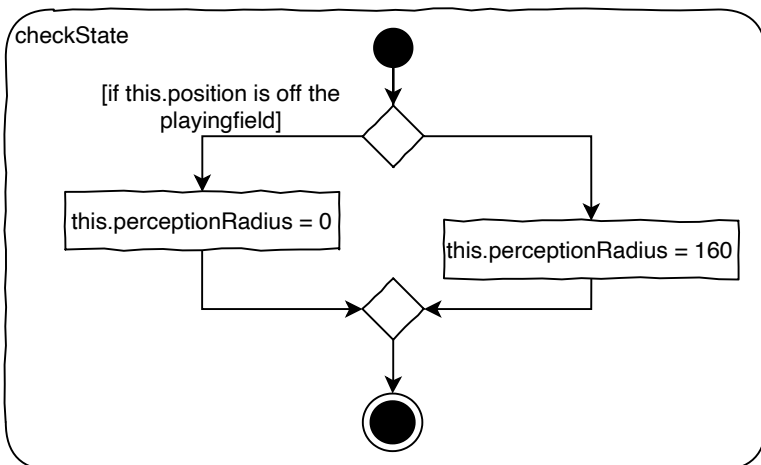
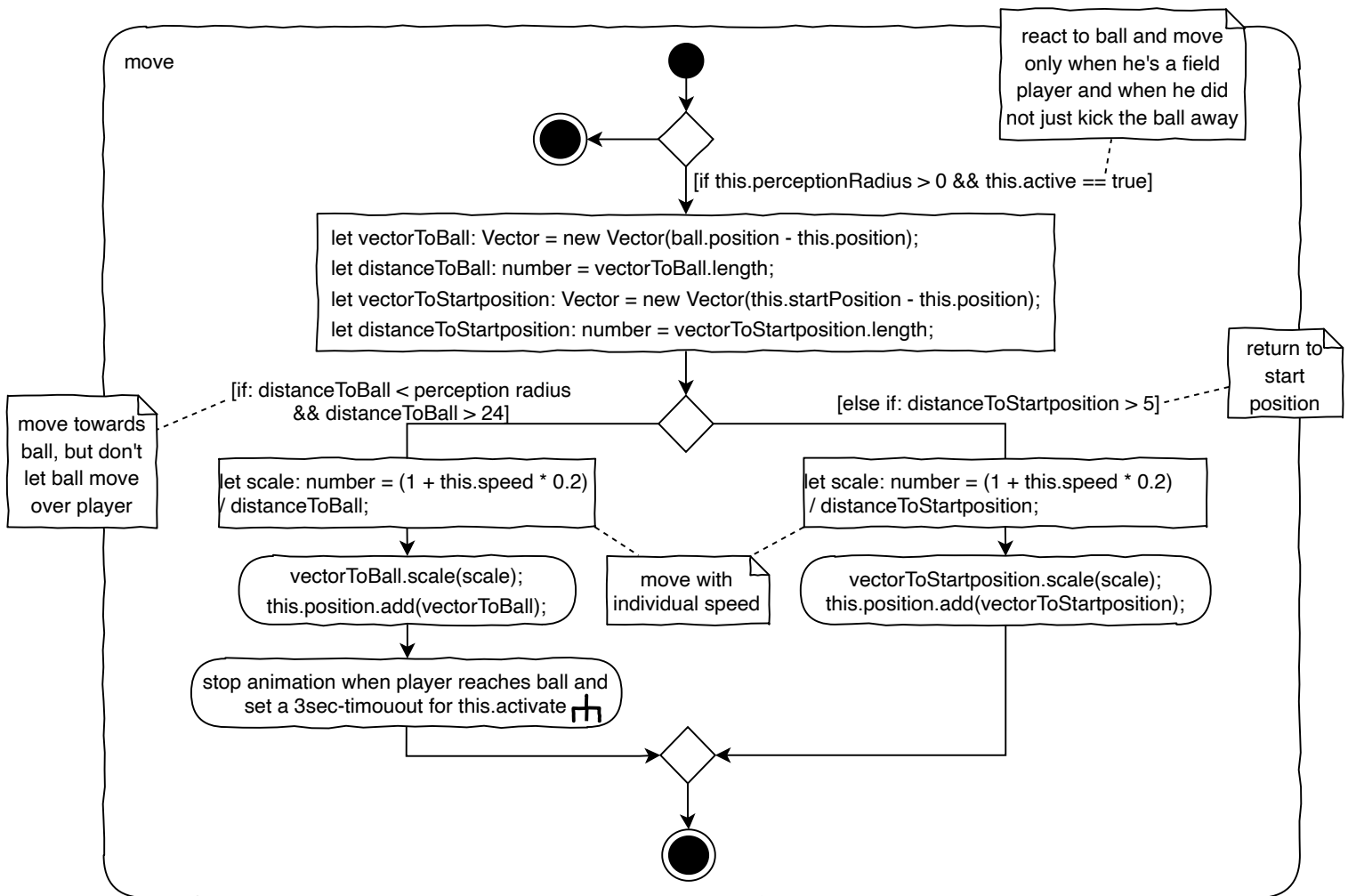
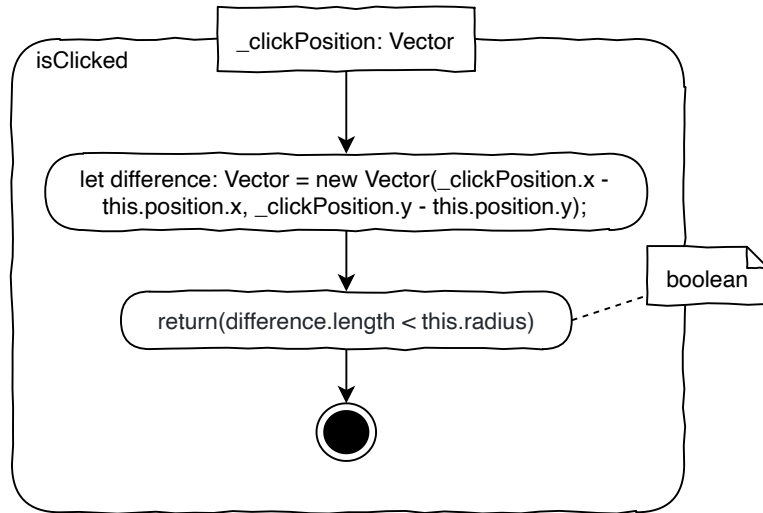
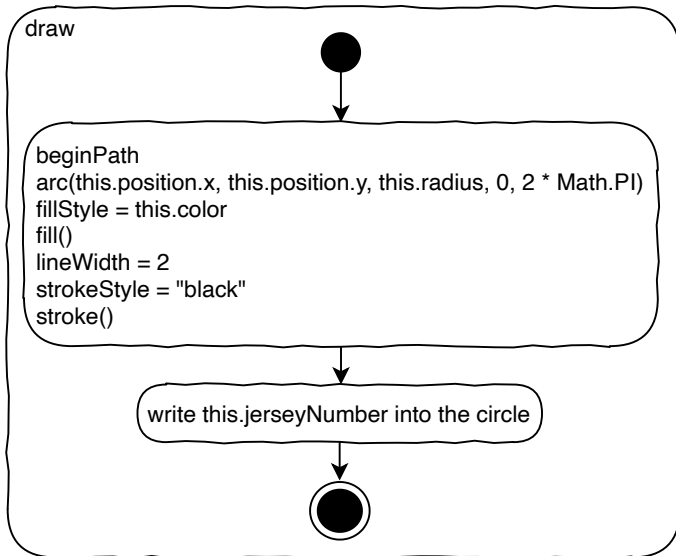
## Linesman Methods



## Referee Methods



# Player Methods



# Ball Methods

draw

draw a circle, fill white and stroke black. Then create a soccer ball pattern with lines and circles



checkGoals

[position in area of left goal  
&& hitGoalA == false]

let event: CustomEvent = new  
CustomEvent(SOCCER\_EVENT.LEFTGOAL\_HIT);

crc2.canvas.dispatchEvent(event)

this.hitGoalA = true

[position in area of right goal  
&& hitGoalB == false]

let event: CustomEvent = new  
CustomEvent(SOCCER\_EVENT.RIGHTGOAL\_HIT);

crc2.canvas.dispatchEvent(event)

this.hitGoalB = true

move



[if this.destination]

let direction: Vector = new Vector  
(this.destination.x - this.position.x,  
this.destination.y - this.position.y)

calculate the  
distance between  
the click and the  
ball

let distance: number = 0

use the distance  
as a factor for the  
precision with  
which the ball will  
reach the  
destination

[if this.startMoving == true]

distance = (playerAtBall.precision / 2) \* (0.1 \* direction.length)

distance += random-0.5 \* (0.25 \* direction.length)

this.startMoving = false

direction.scale(1/50)

to match the 50fps  
of the animation

[if distance < 150]

this.position.add(direction \* 2)

this.position.add(direction)

[this.position x and y  
bigger/smaller  
than the playingfield]

this.position && this.destination = new  
Vector (500, 275)

[this.position x is close  
to a goal area]

playSample(1); 

this.checkGoals() 