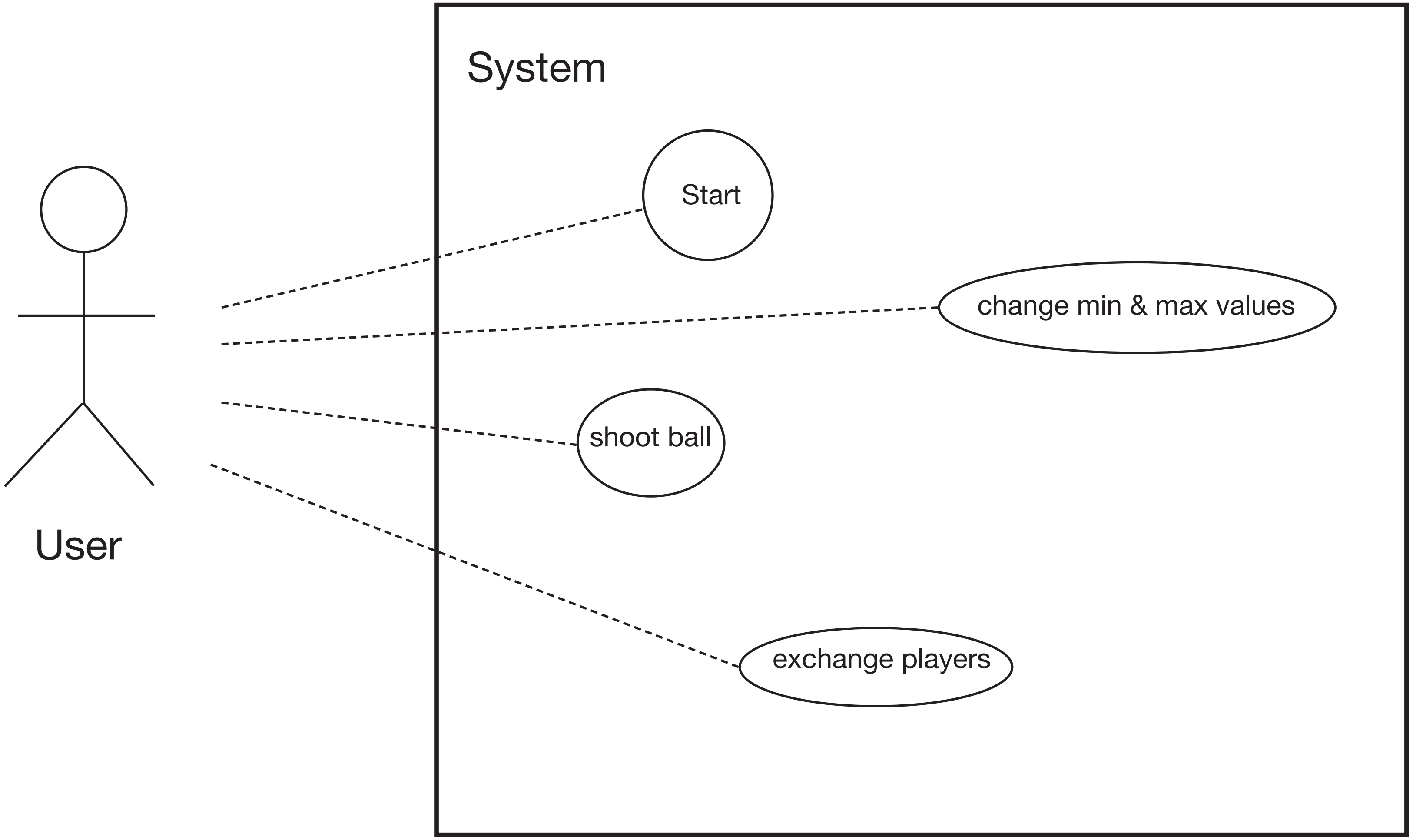
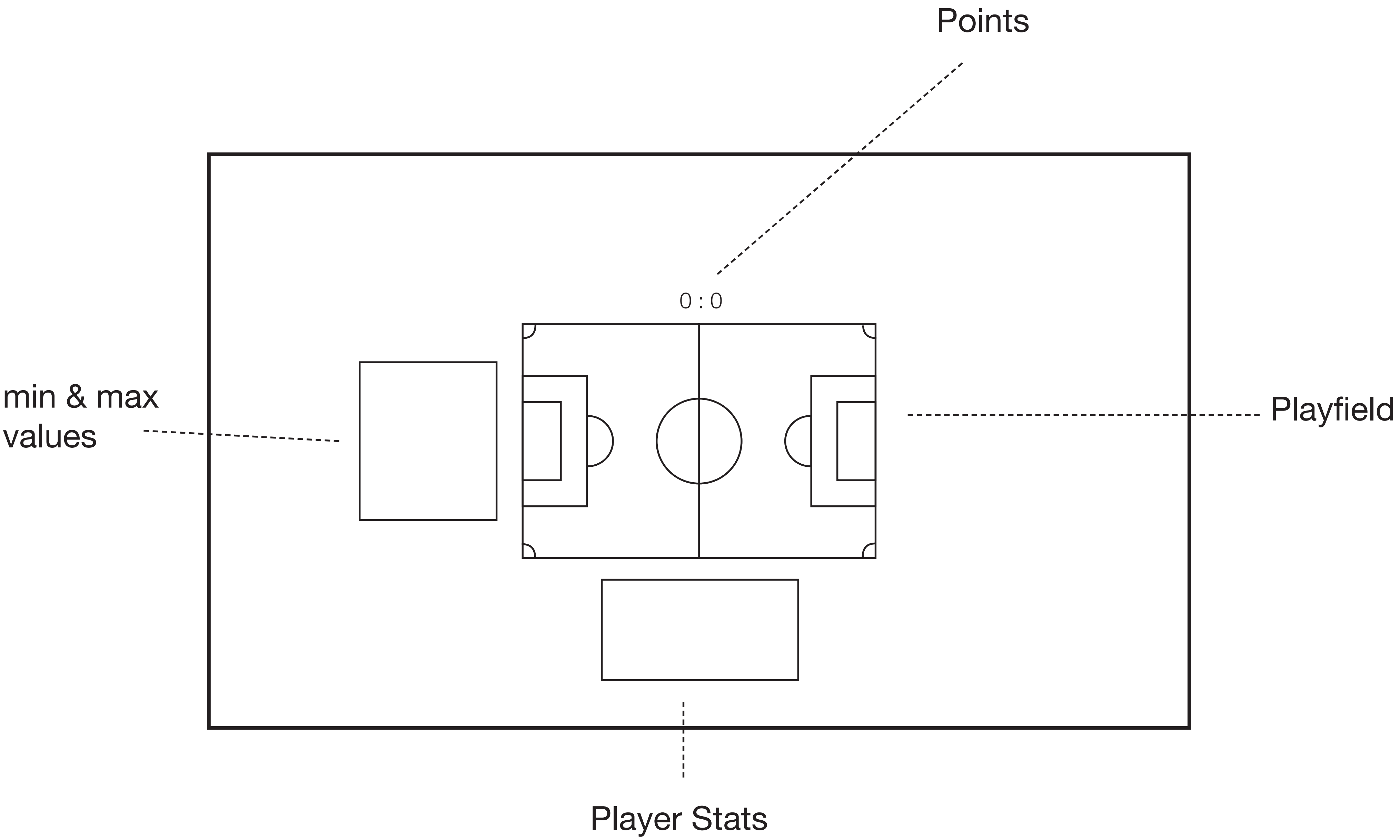
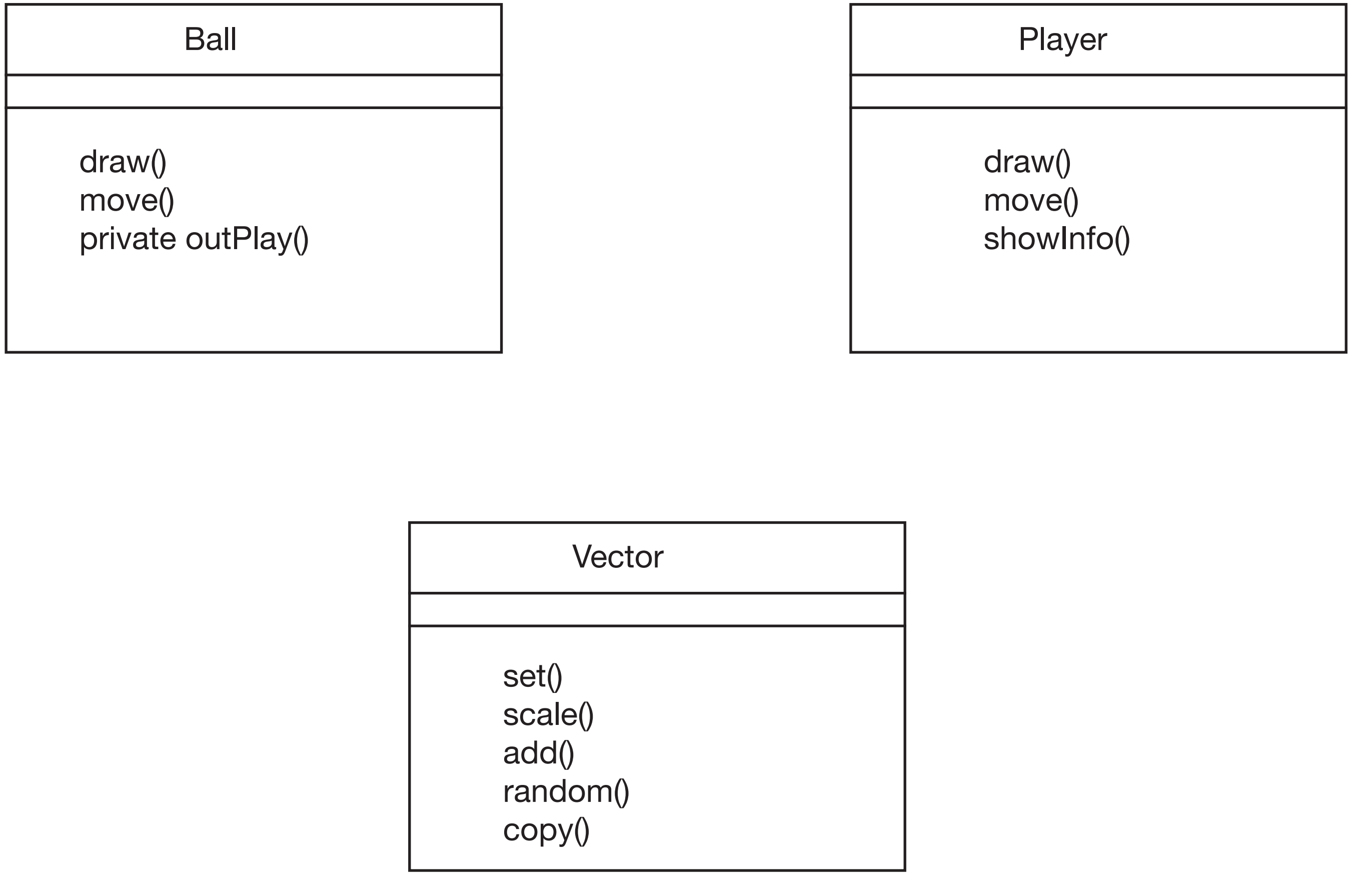
Use-Case Diagramm



User Interface Diagramm



Class Diagramm



Ball

draw()
move()
private outPlay()

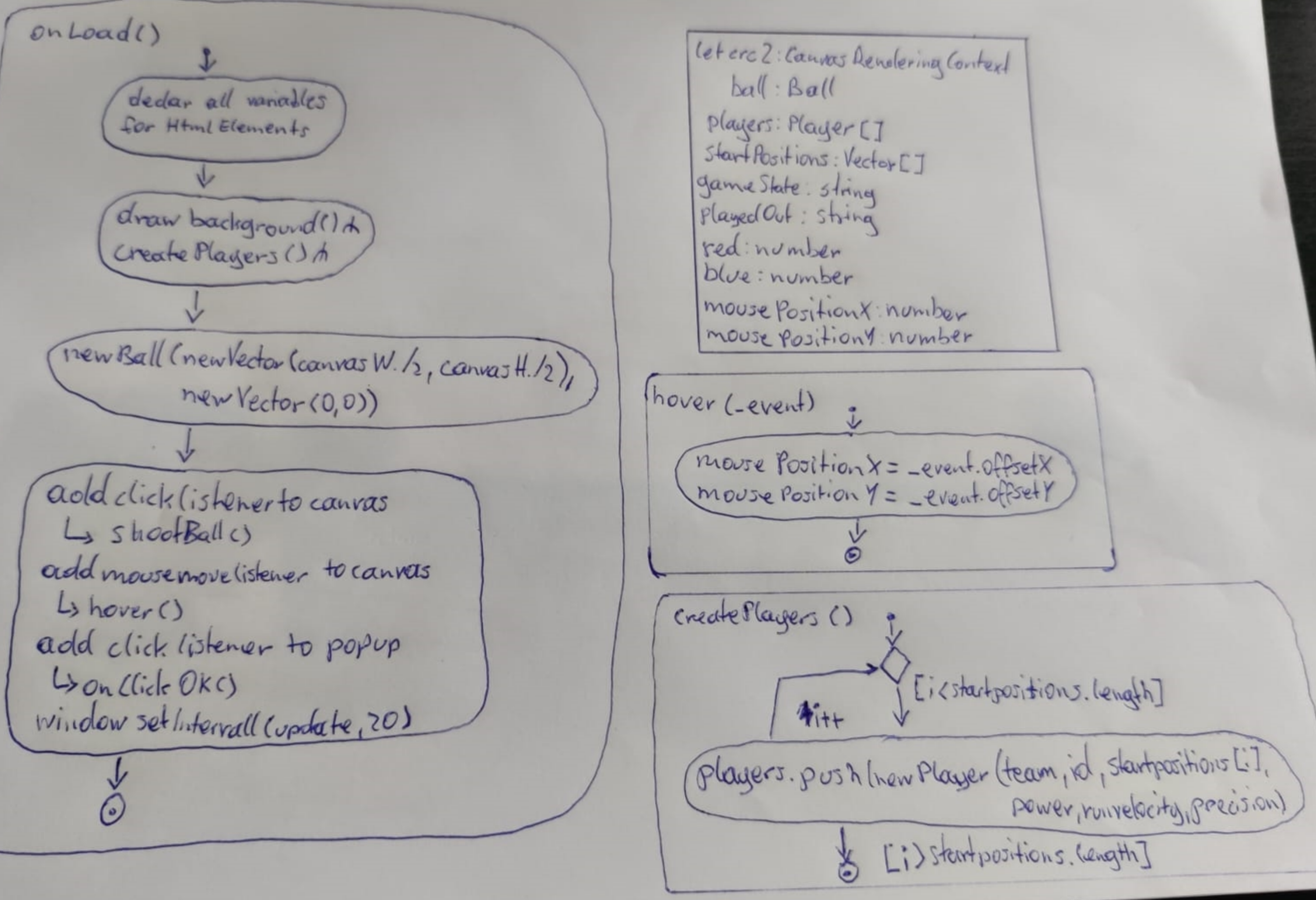
Player

draw()
move()
showInfo()

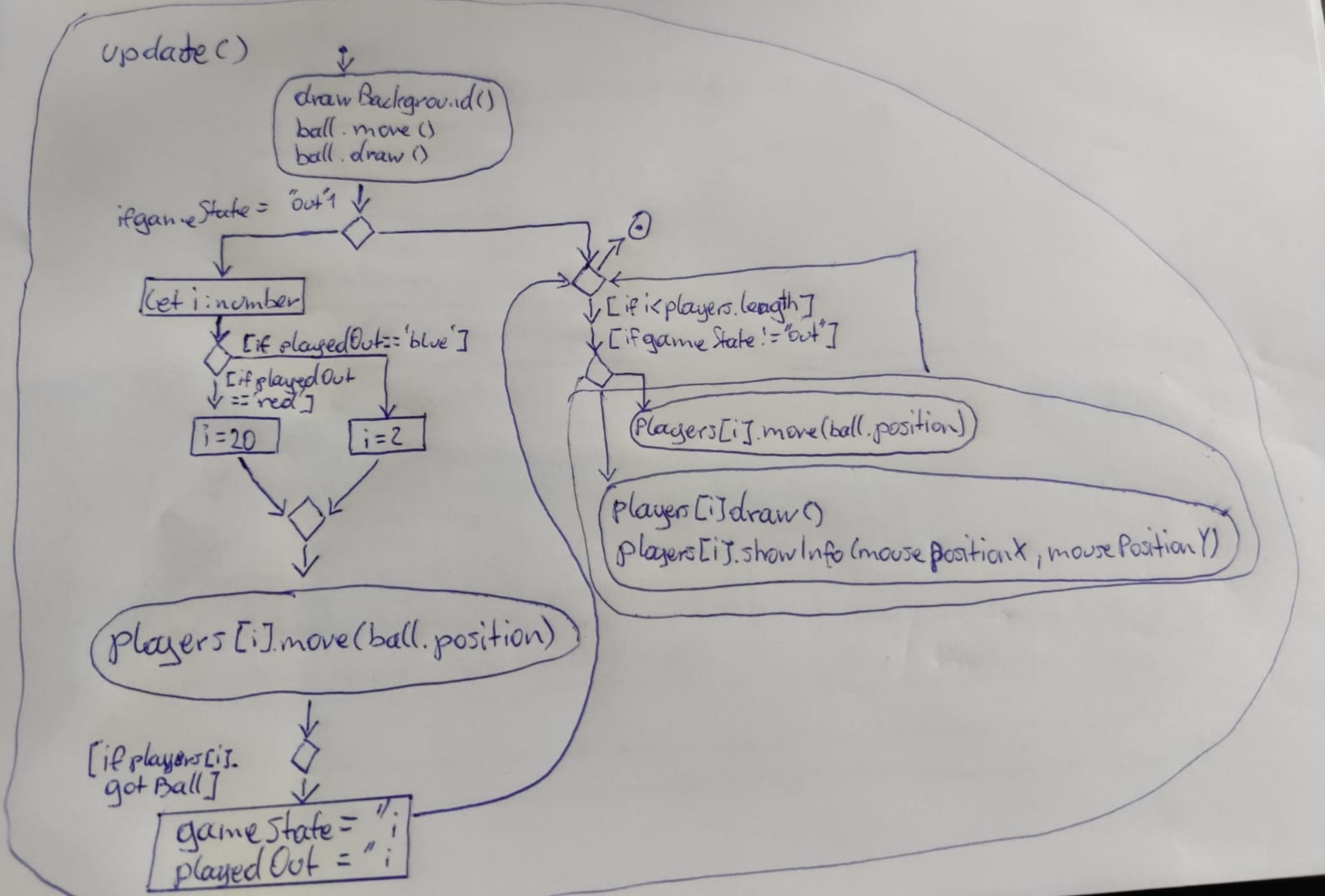
Vector

set()
scale()
add()
random()
copy()

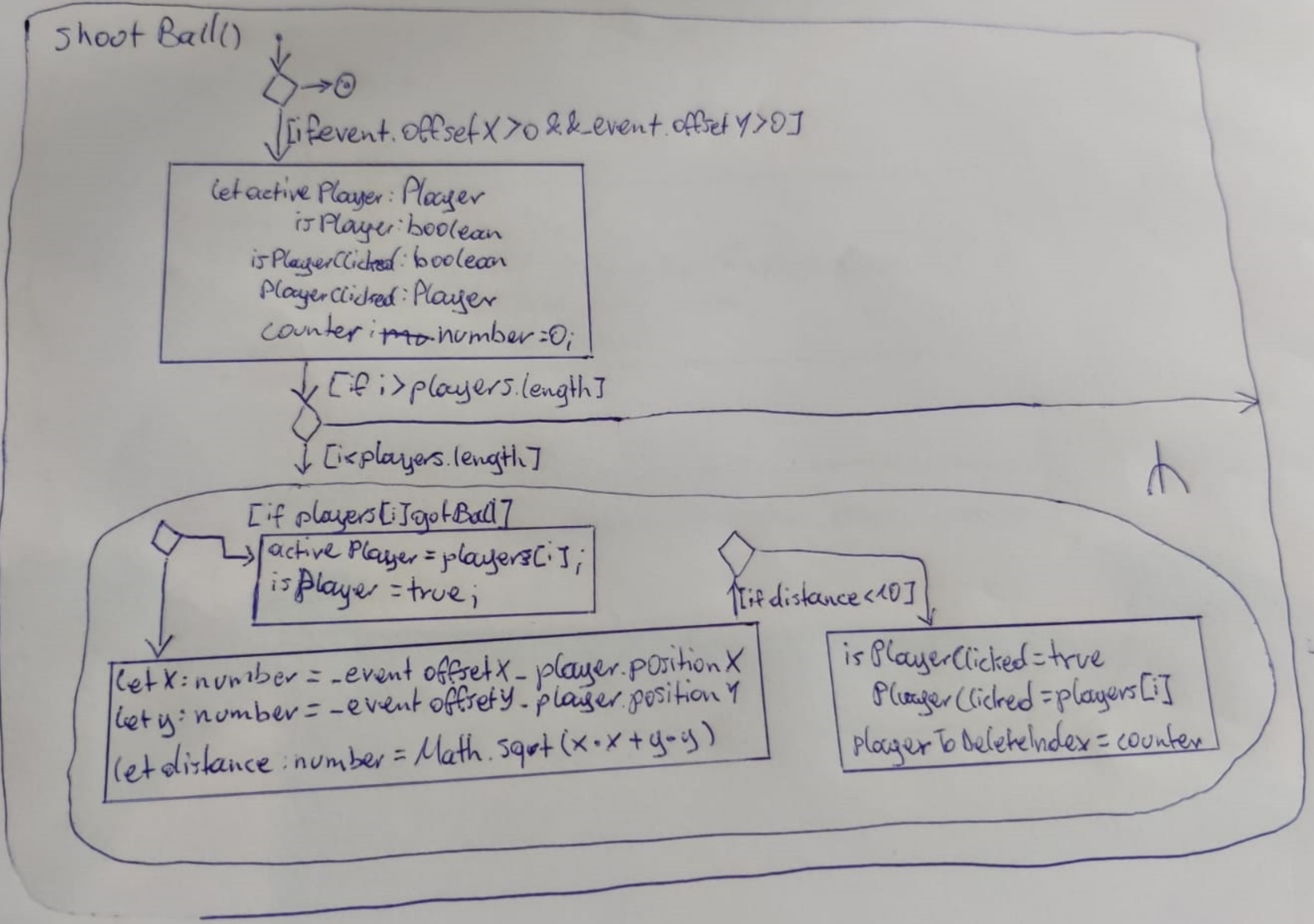
Activity Diagramm: Football-Simulator / Main

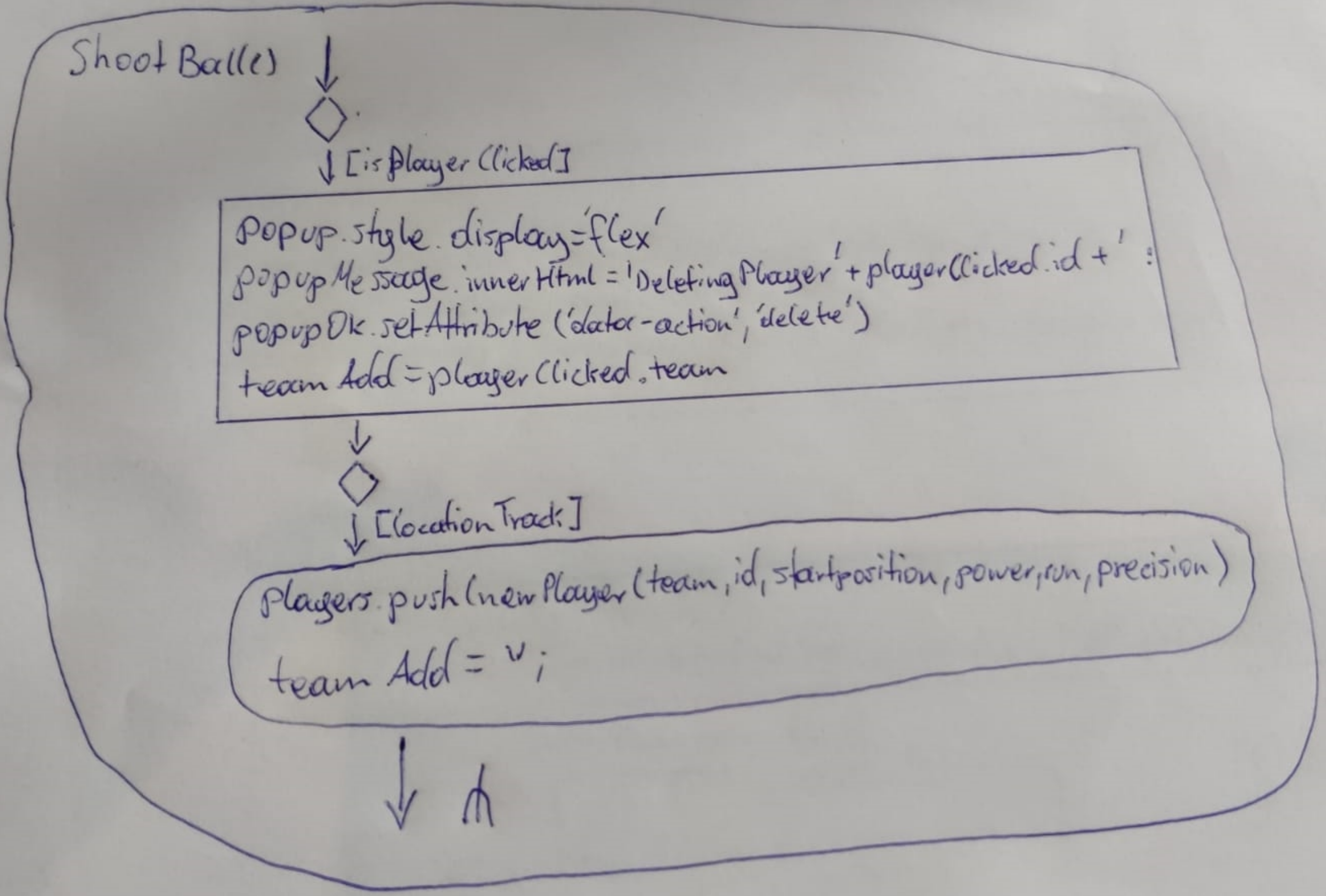

onLoad()
declar all variables for Html Elements
drawbackground()
createPlayers()
newBall(newVector(canvasW./2, canvasH./2), newVector(0,0))
add click listener to canvas
↳ shootBall()
add mousemove listener to canvas
↳ hover()
add click listener to popup
↳ onClickOK()
window setIntervall(update, 20)
let crc2: CanvasRenderingContext
ball: Ball
players: Player[]
startPositions: Vector[]
gameState: string
playedOut: string
red: number
blue: number
mousePositionX: number
mousePositionY: number
hover(_event)
mousePositionX = _event.offsetX
mousePositionY = _event.offsetY
createPlayers()
[i<startpositions.length]
i++
players.push(newPlayer(team, id, startpositions[i], power, runvelocity, precision)
[i>startpositions.length]

Activity Diagranm : Football-Simulator / Main



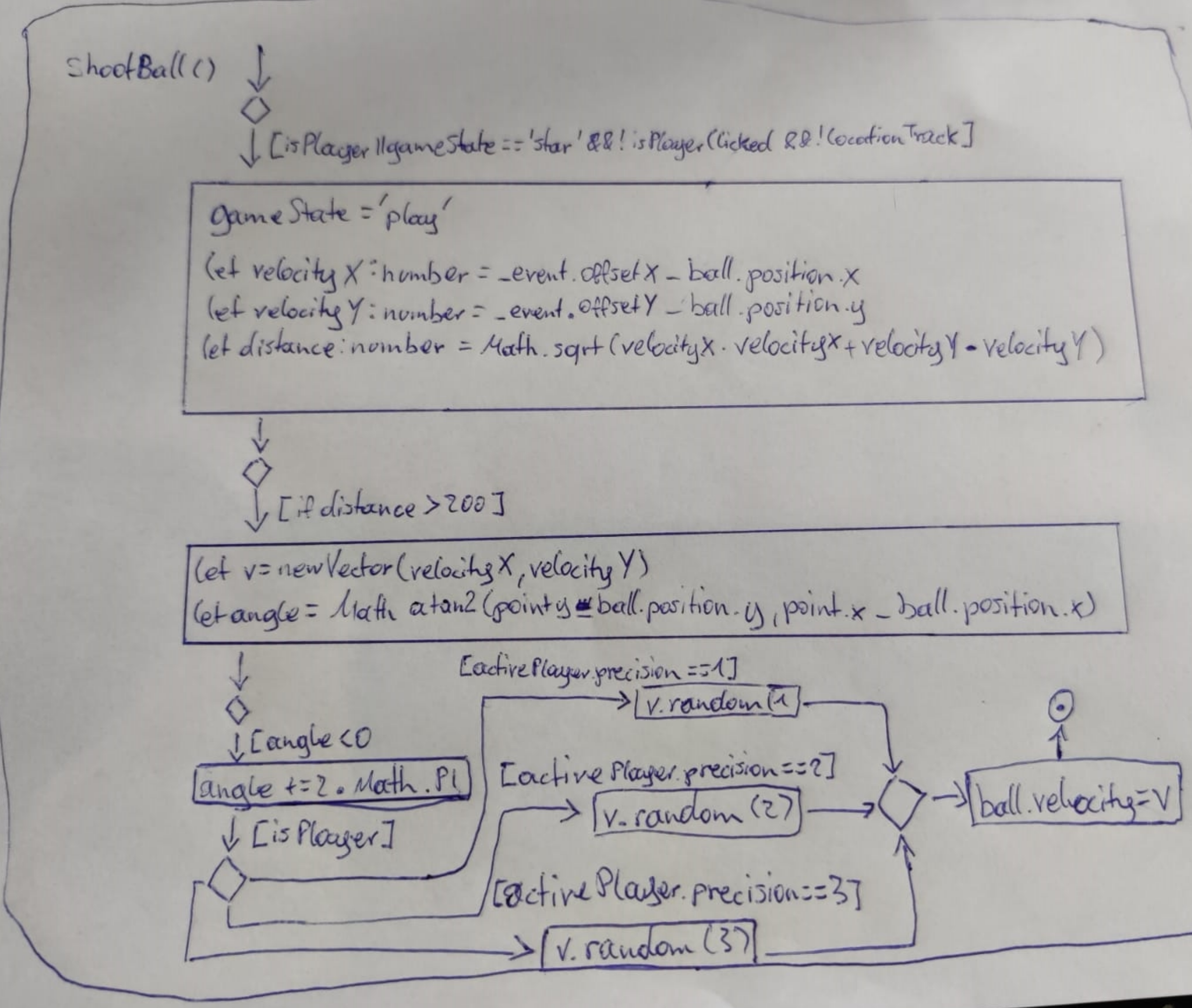
Activity Diagramm: Football-Simulator / Main


Shoot Ball()
[ifevent.offsetX>0 &&_event.offsetY>0]
let activePlayer: Player
isPlayer: boolean
isPlayerClicked: boolean
PlayerClicked: Player
counter: number = 0;
[if i > players.length]
[i < players.length]
[if players[i]gotBall]
activePlayer = players[i];
isPlayer = true;
let x: number = _event offsetX _ player.positionX
let y: number = _event offsetY _ player.positionY
let distance: number = Math.sqrt(x·x + y·y)
[if distance < 10]
isPlayerClicked = true
PlayerClicked = players[i]
playerToDeleteIndex = counter



```
popup.style.display='flex'
popupMessage.innerHtml = 'DeletingPlayer' + playerClicked.id + ' :
popupOk.setAttribute('data-action', 'delete')
teamAdd = playerClicked.team
```

```
players.push(new Player(team, id, startposition, power, run, precision)
teamAdd = ";
```



ShootBall()
[isPlayer||gameState=='star'&&!isPlayerClicked &&!locationTrack]
[if distance>200]
[angle<0
[isPlayer]
[activePlayer.precision==1]
[activePlayer.precision==2]
[activePlayer.precision==3]

```
gameState='play'
let velocityX:number=_event.offsetX_ball.position.x
let velocityY:number=_event.offsetY_ball.position.y
let distance:number = Math.sqrt(velocityX.velocityX+velocityY.velocityY)
```

```
let v=newVector(velocityX,velocityY)
let angle=Math atan2(pointy_ball.position.y,point.x_ball.position.x)
```

```
angle+=2.Math.PI
```

```
v.random(1)
```

```
v.random(2)
```

```
v.random(3)
```

```
ball.velocity=V
```