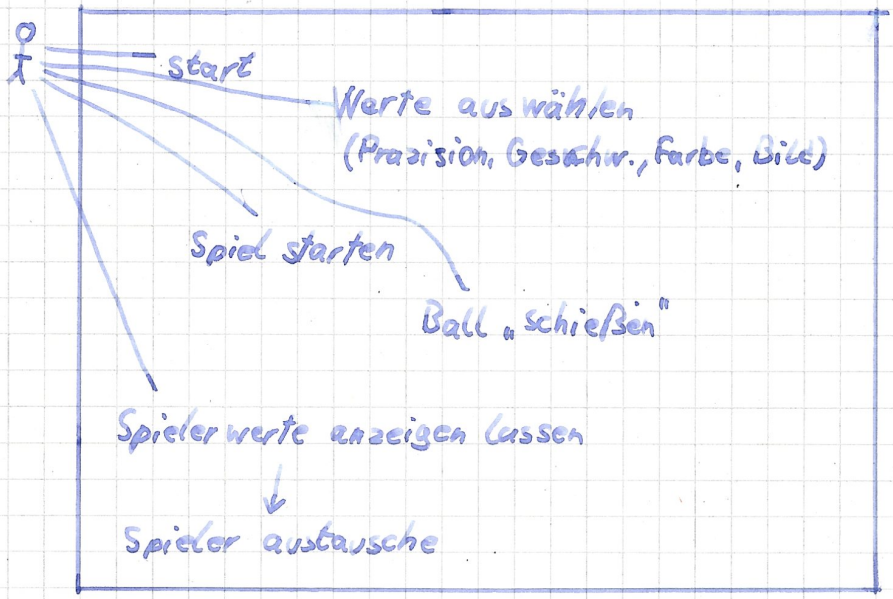
Abschlussarbeit EIA 2

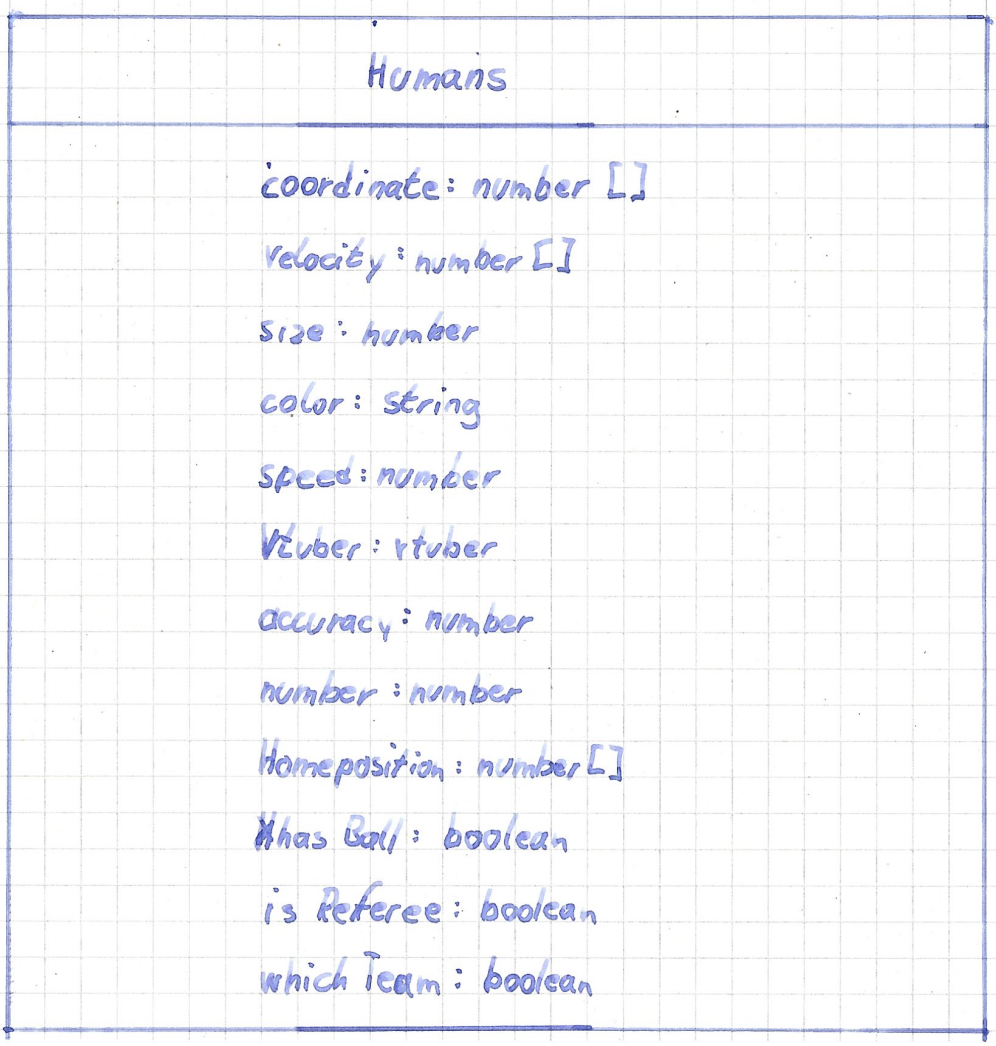
Edwin Brill, 267054

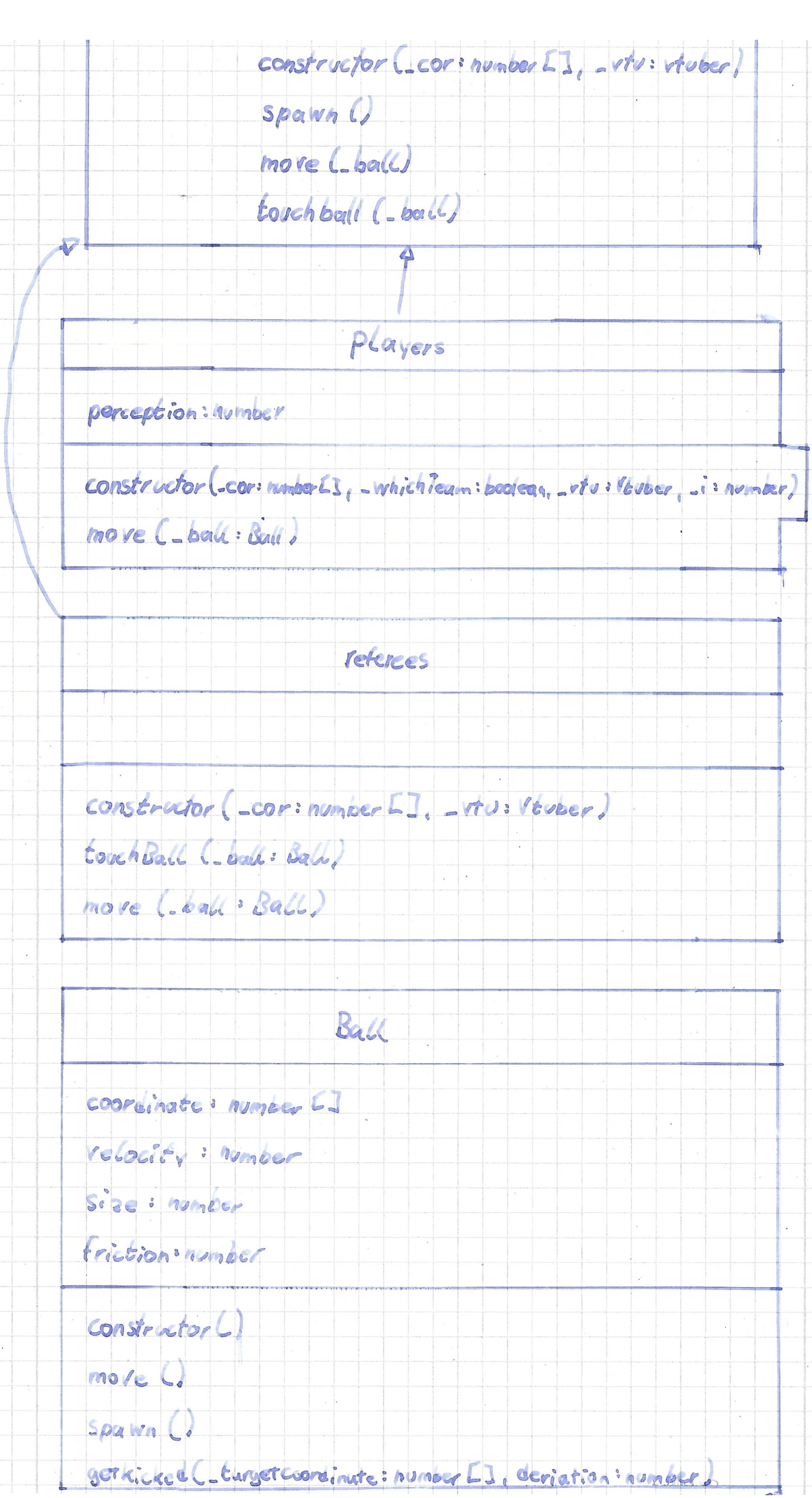
18.07.21

Use-Case Diagram



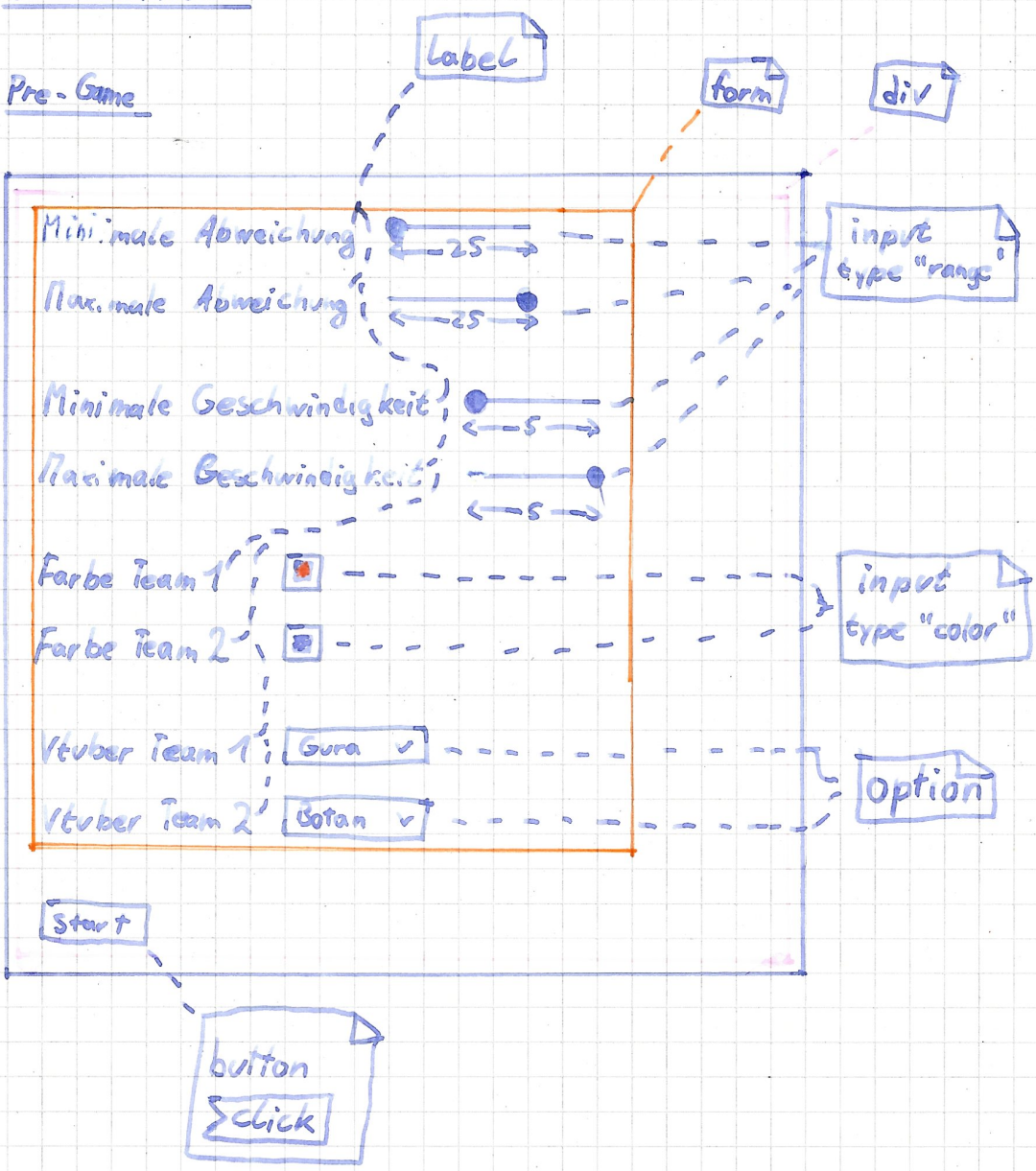
Class-Diagram



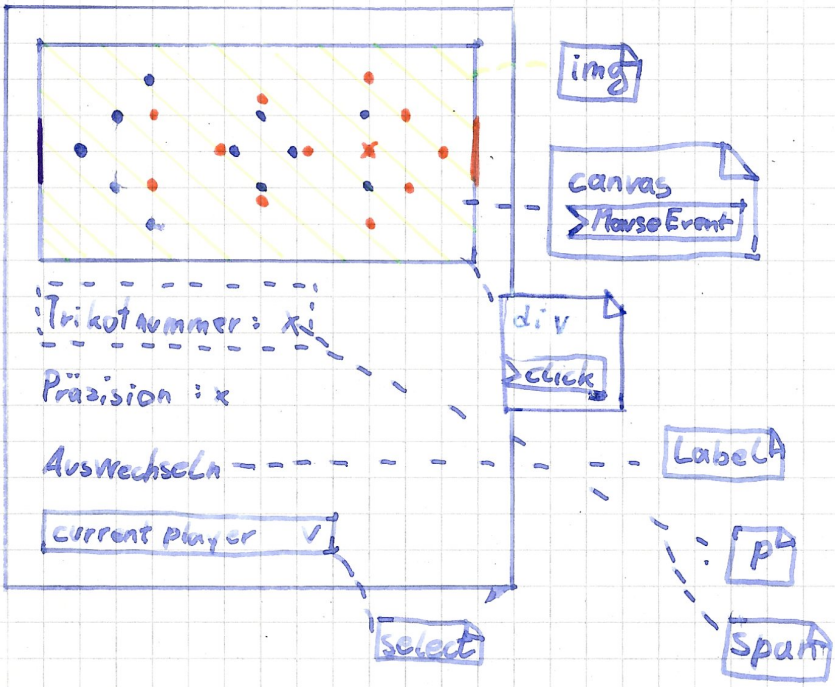

constructor (_cor: number[], _vtu: vtuber)
spawn ()
move (_ball)
touchball (_ball)
players
perception: number
constructor(_cor: number[], _whichTeam: boolean, _vtu: vtuber, _i: number)
move (_ball: Ball)
references
constructor (_cor: number[], _vtu: vtuber)
touchBall (_ball: Ball)
move (_ball: Ball)
Ball
coordinate: number[]
velocity: number
size: number
friction: number
Constructor()
move ()
spawn ()
getKicked(_targetCoordinate: number[], deviation: number)

UI - Scribble

Pre - Game



In - Game



Actirity Diagram

Humans

constructor

_cor: number[]
_vtu: Vtuber

this.cor = _cor
this.color = "red"
this.size = ~20
this.vtuber = _vtu
this.velocity = [0,0]
this.hasball = false

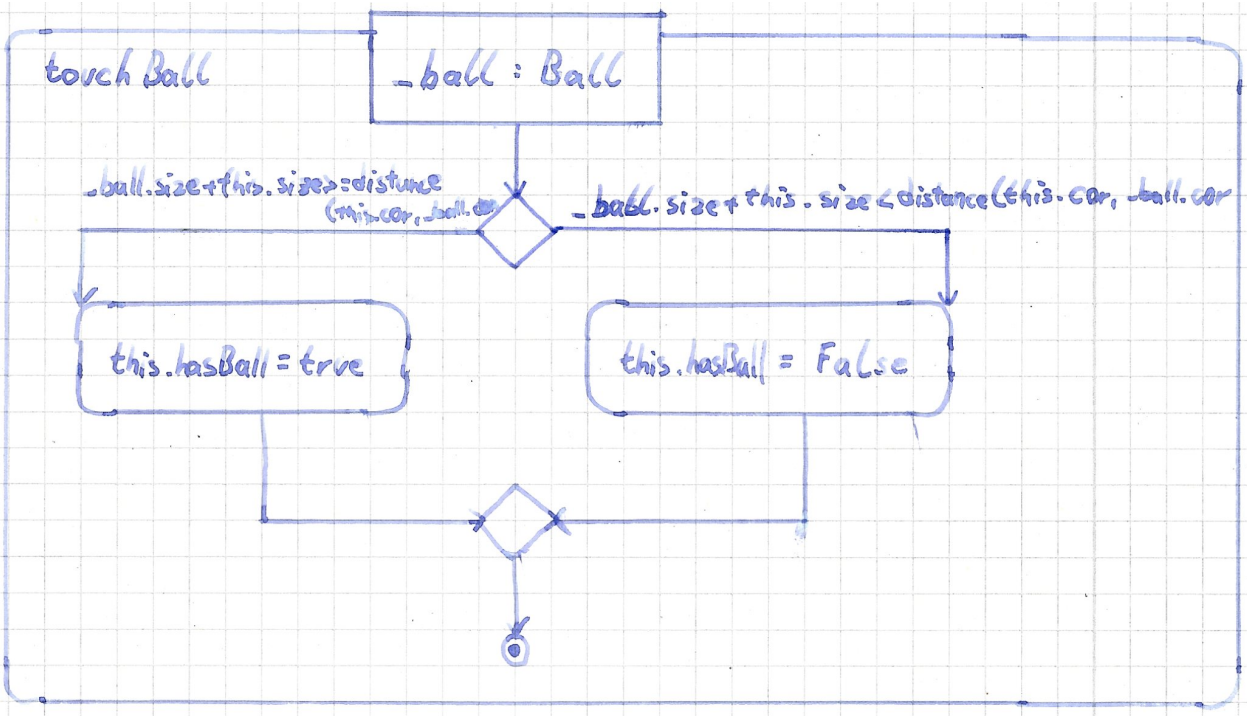
spawn

draw an arc with(this.cor[0], this.cor[1], this.size, 0, 2 * Math.PI) in this.color, strokestyle = this.color, Linewidth = ~10

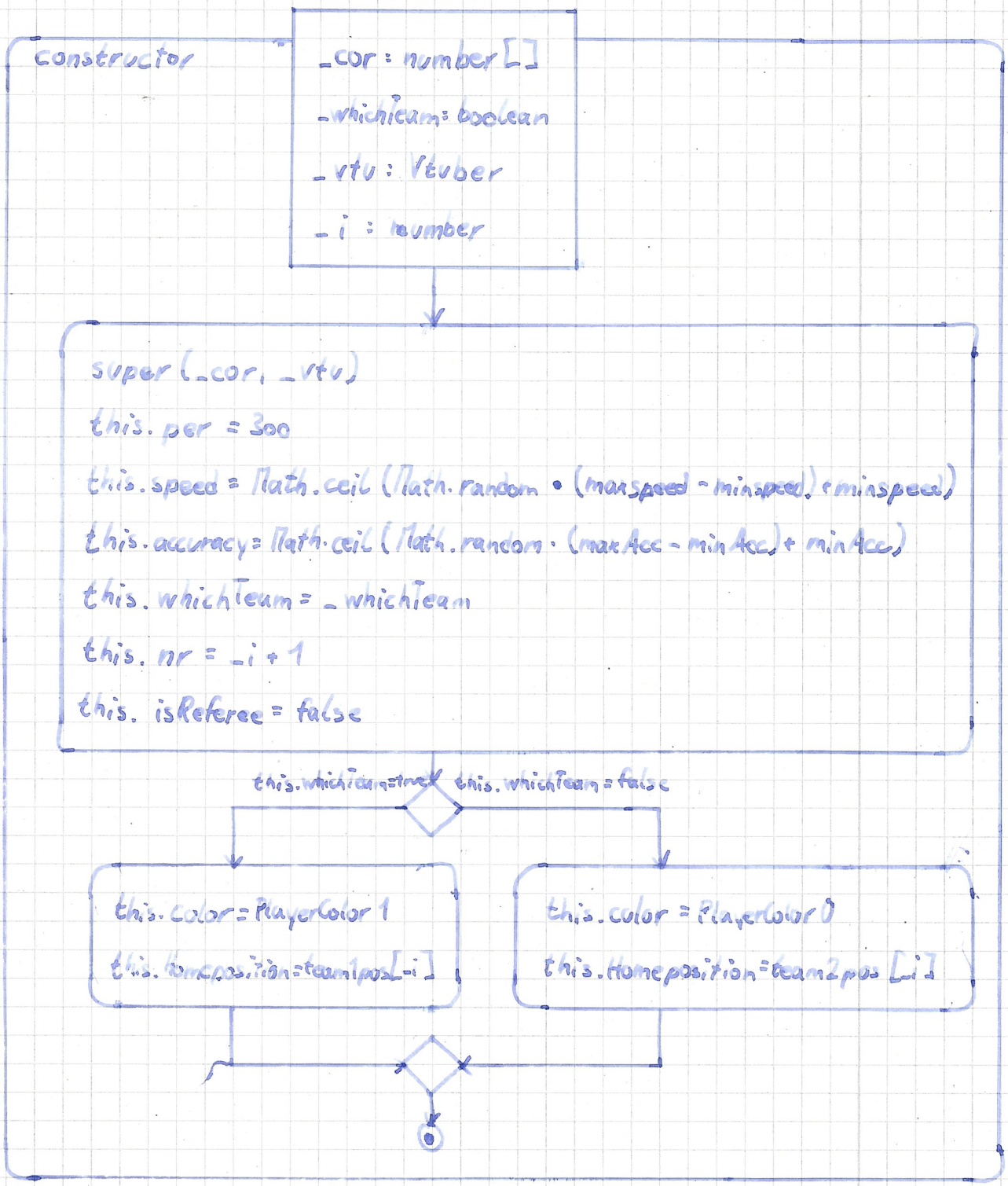
drawImage (this.vtuber.img, this.cor[0]-this.size, this.cor[1] - this.size)

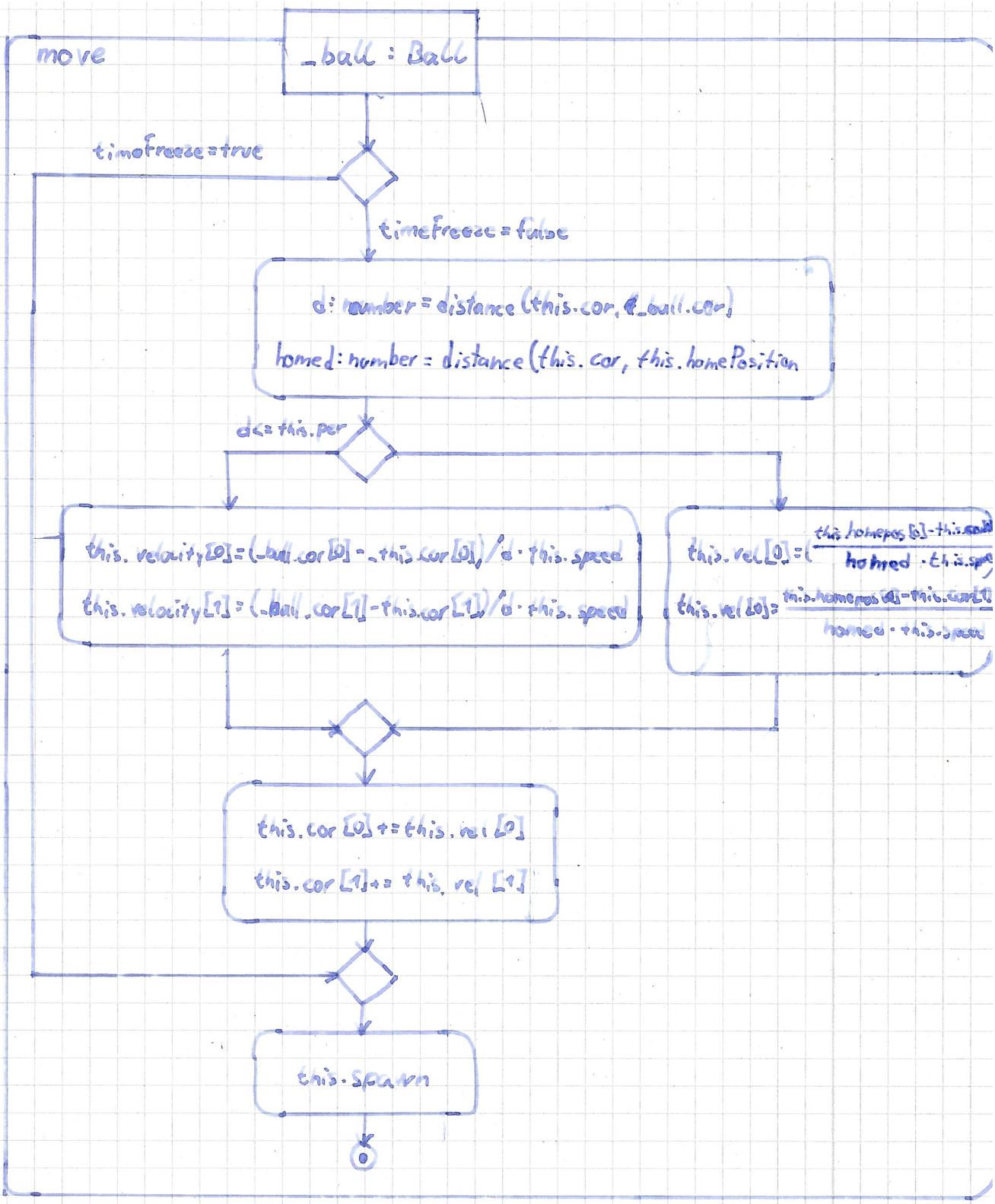
move

_ball : Ball

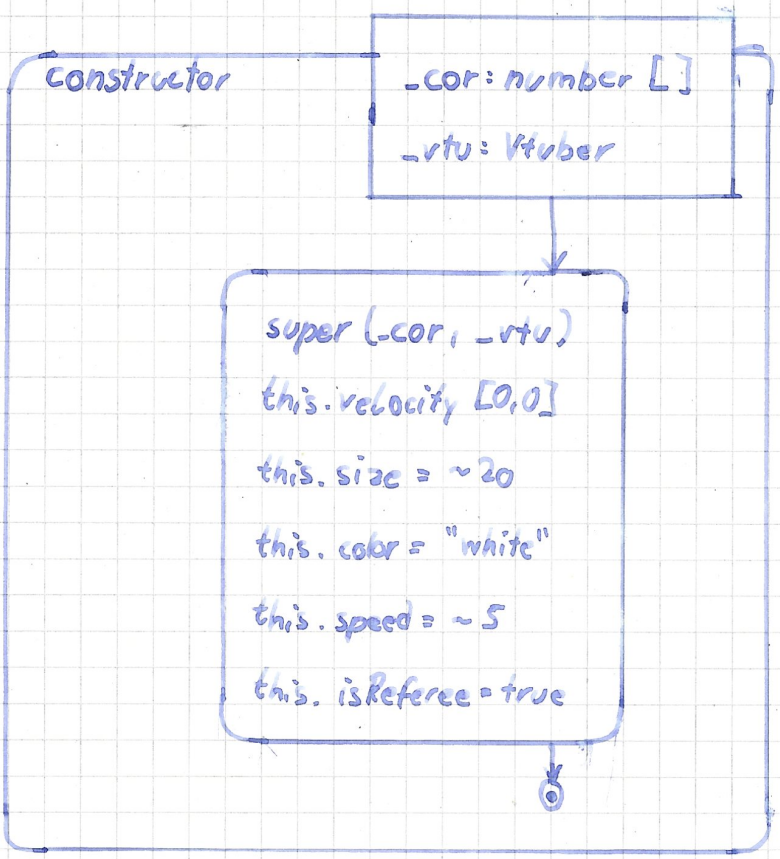

touchBall
_ball : Ball
_ball.size+this.size>=distance (this.car, _ball.car
_ball.size+this.size<distance(this.car, _ball.car
this.hasBall = true
this.hasBall = False

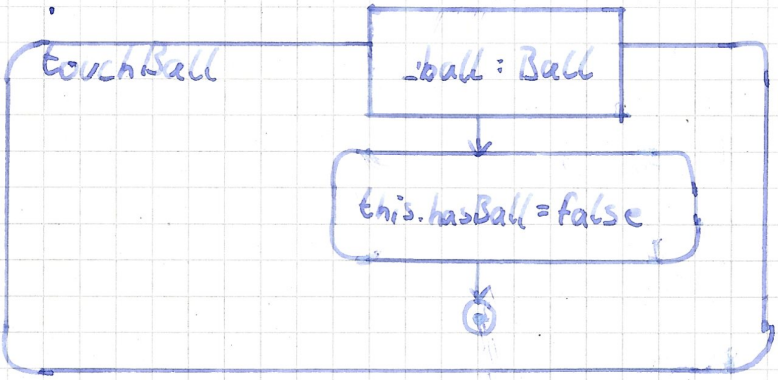
Players

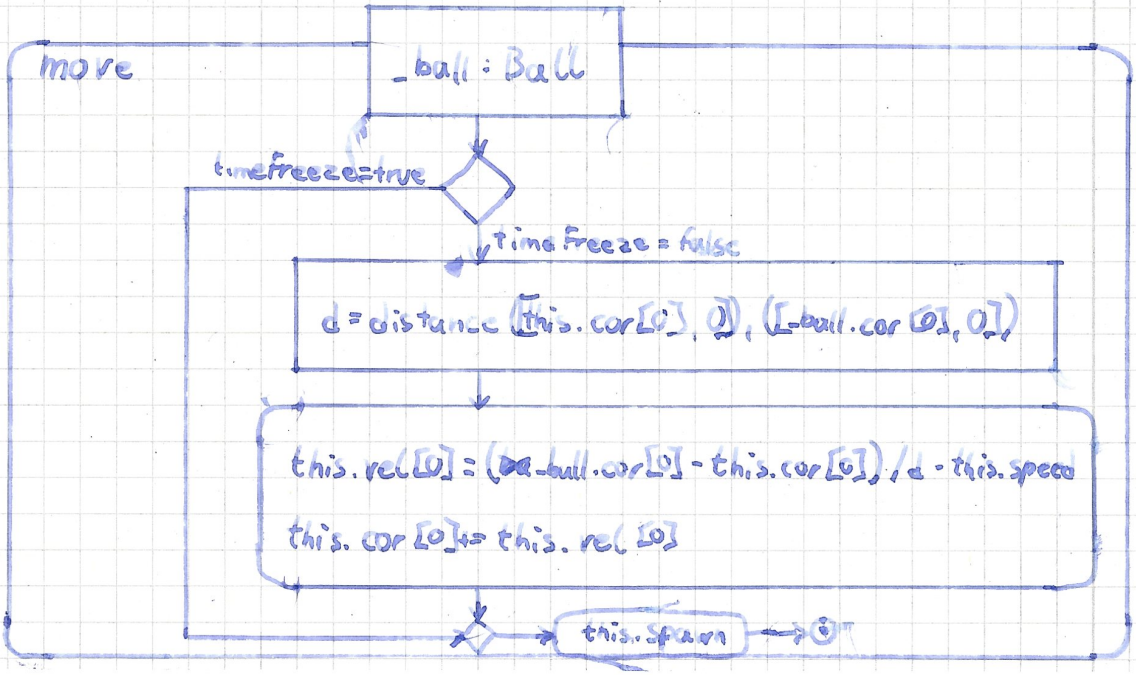

constructor
_cor : number[]
_whichTeam: boolean
_vtu : Vtuber
_i : number
super(_cor, _vtu)
this.per = 300
this.speed = Math.ceil(Math.random * (maxspeed - minspeed) + minspeed)
this.accuracy = Math.ceil(Math.random * (maxAcc - minAcc) + minAcc)
this.whichTeam = _whichTeam
this.nr = _i + 1
this.isReferee = false
this.whichTeam=true
this.whichTeam = false
this.color = PlayerColor 1
this.Homeposition = team1pos[_i]
this.color = PlayerColor 0
this.Homeposition = team2pos[_i]


move
_ball : Ball
timefreeze=true
timefreeze=false
d: number = distance (this.cor, _ball.cor)
homed: number = distance (this.cor, this.homePosition
d<=this.per
this.velocity[0]=(_ball.cor[0]-this.cor[0])/d*this.speed
this.velocity[1]=(_ball.cor[1]-this.cor[1])/d*this.speed
this.vel[0]=
this.vel[0]=
this.cor[0]+=this.vel[0]
this.cor[1]+=this.vel[1]
this.spawn

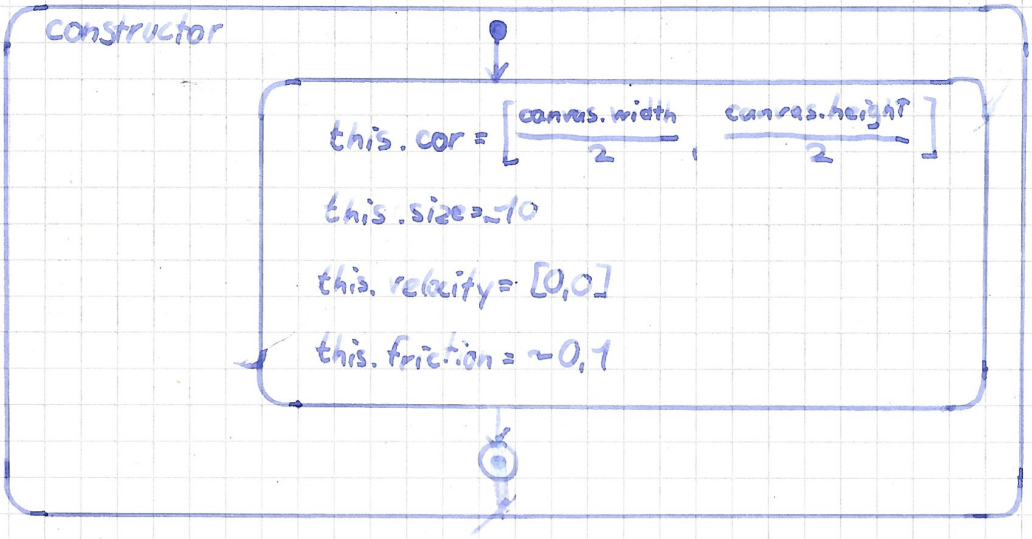
Referees

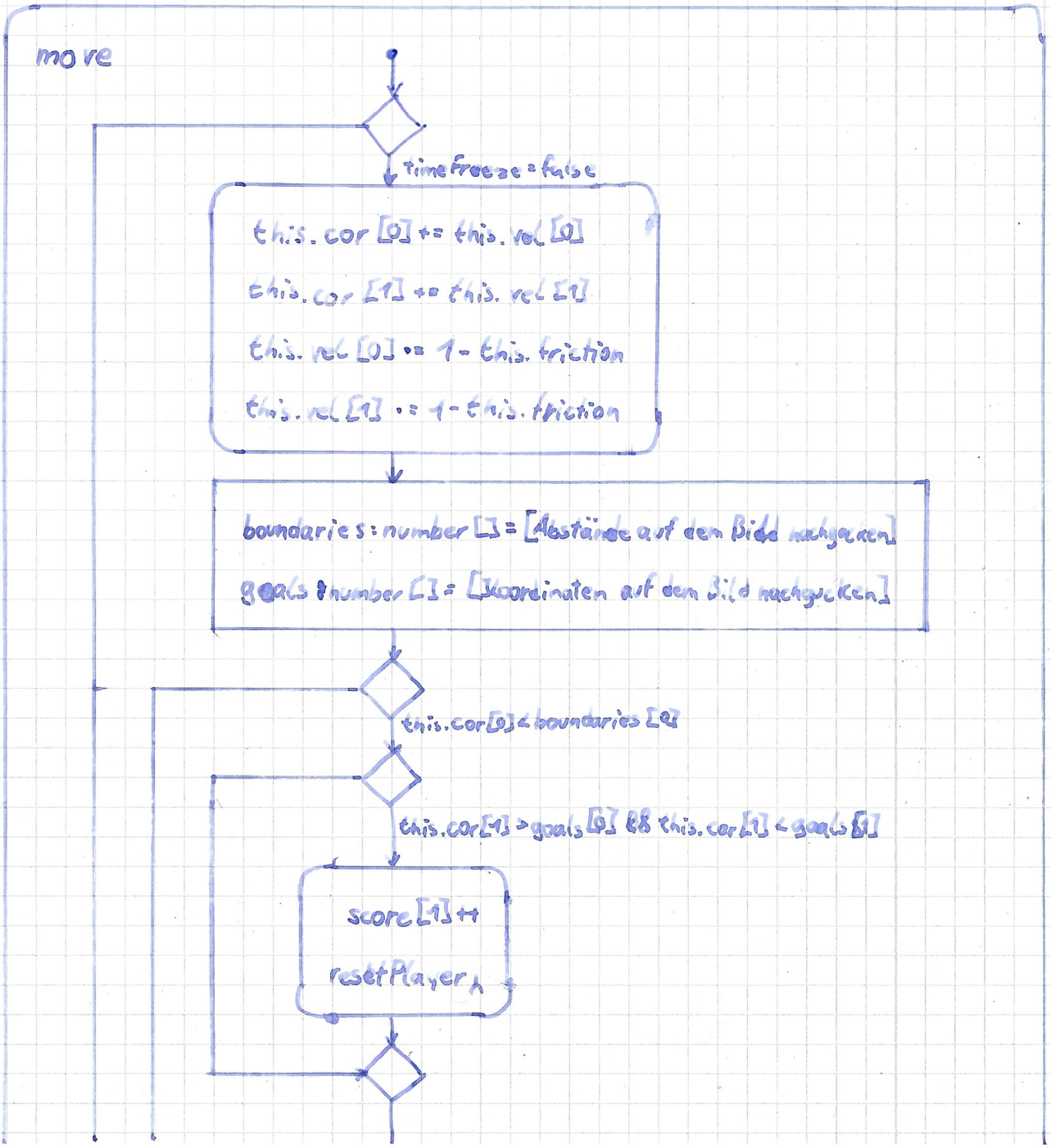

constructor
_cor: number[]
_vtu: Vtuber
super(_cor, _vtu)
this.velocity [0,0]
this.size = ~20
this.color = "white"
this.speed = ~5
this.isReferee = true

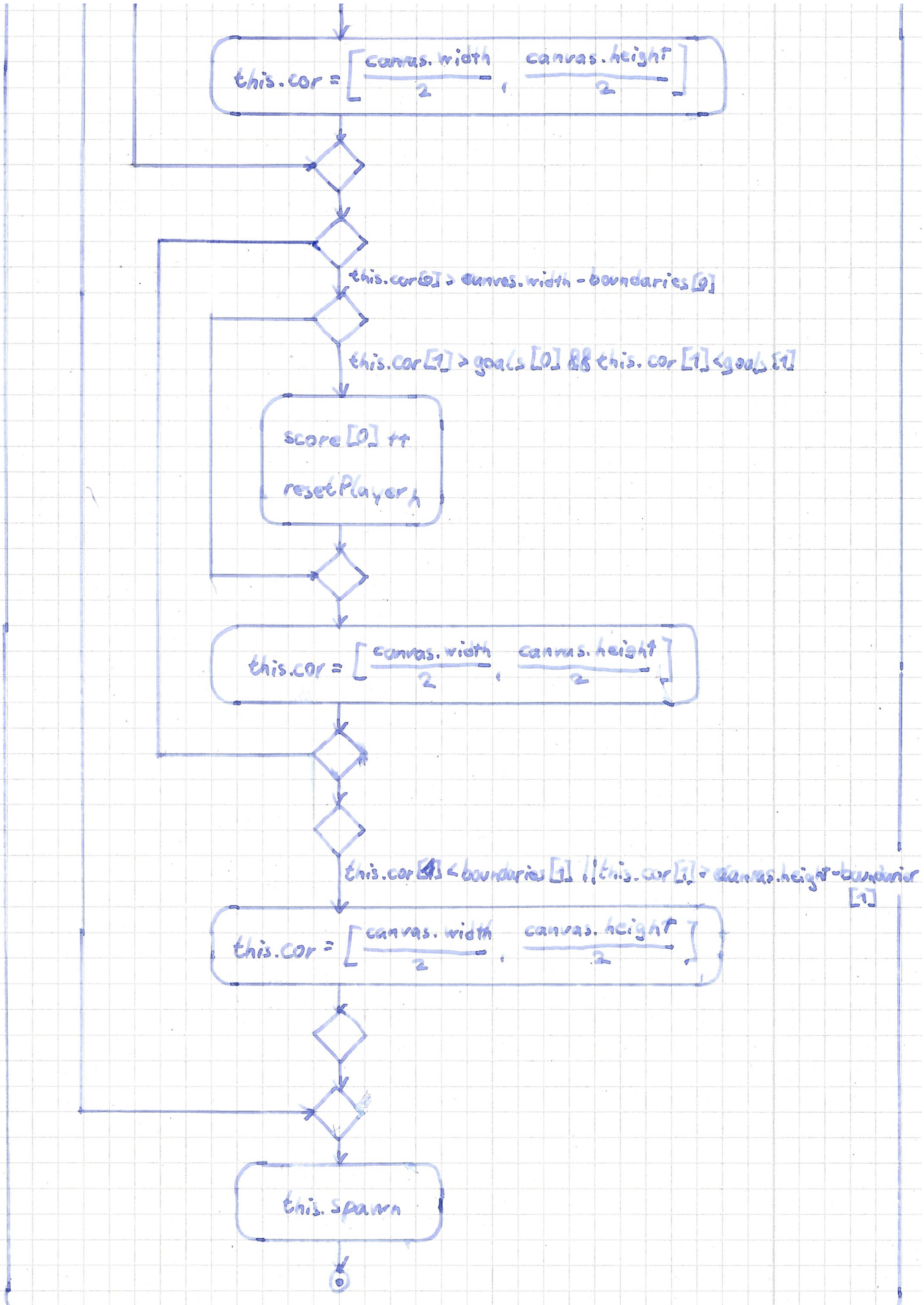

touchBall
_ball : Ball
this.hasBall = false

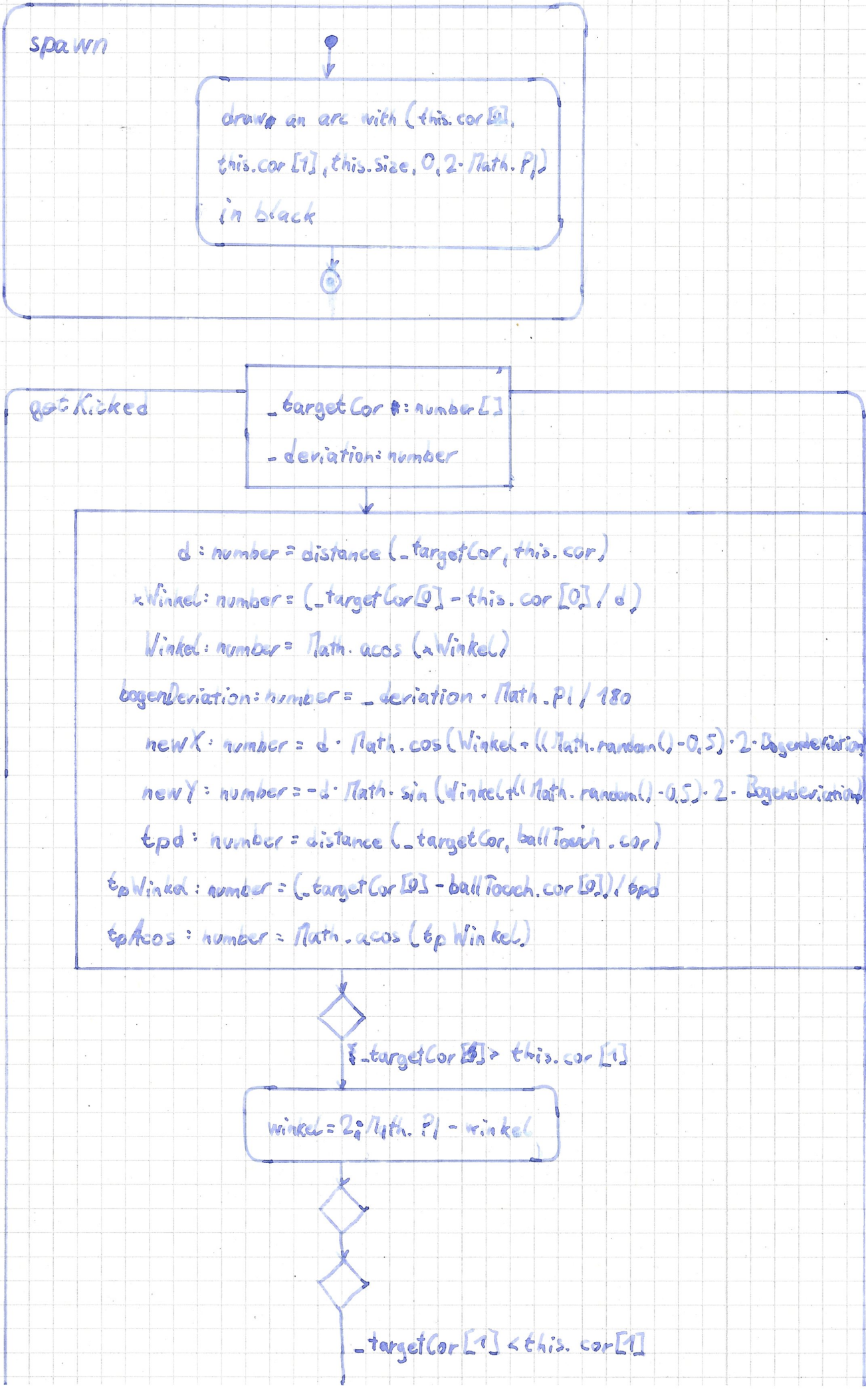

move
_ball : Ball
timeFreeze=true
timeFreeze = false
d = distance([this.cor[0], 0], [_ball.cor[0], 0])
this.vel[0] = (_ball.cor[0] - this.cor[0]) / d * this.speed
this.cor[0] += this.vel[0]
this.spawn

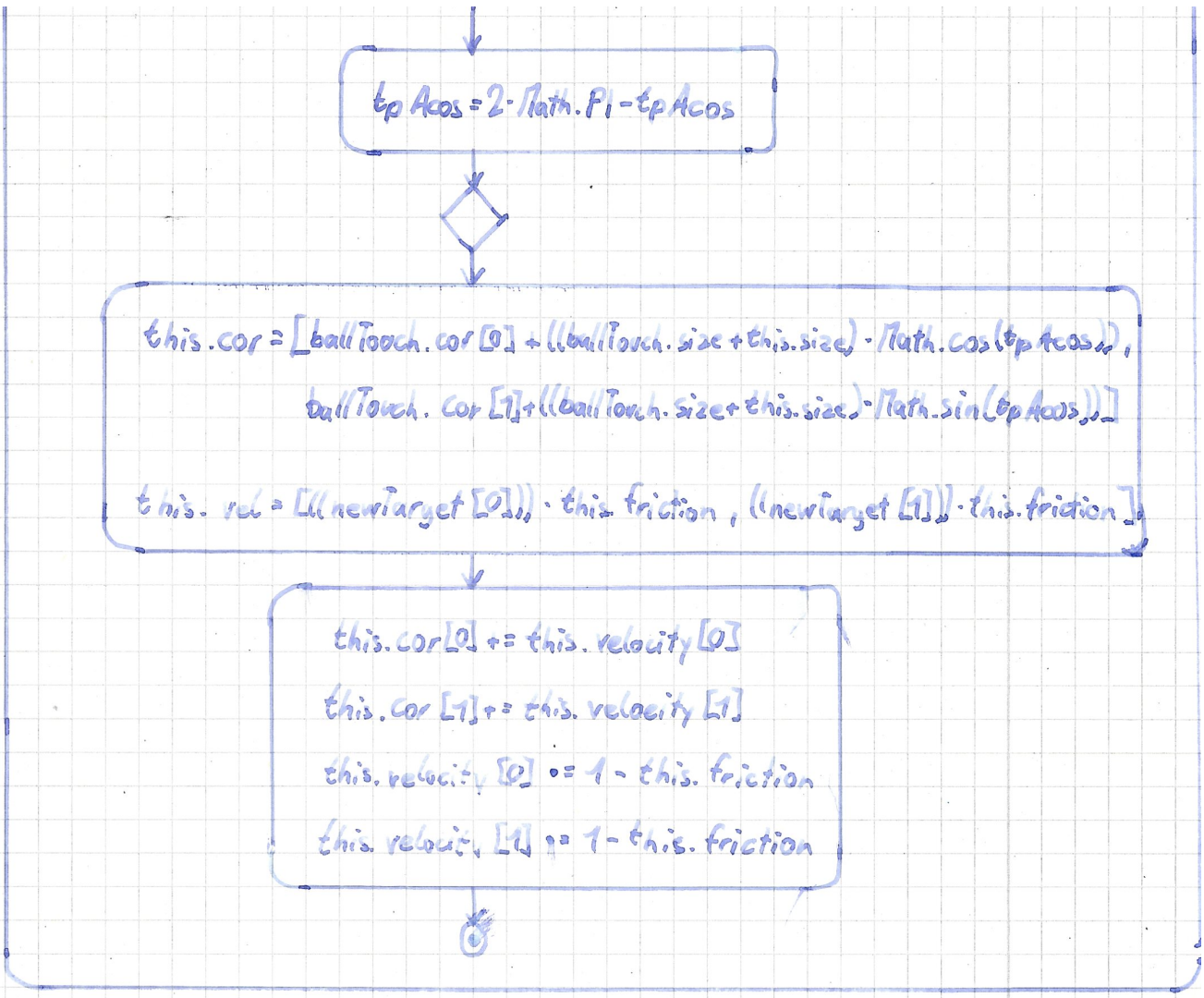
Ball


constructor
this.cor = [canvas.width / 2, canvas.height / 2]
this.size = 10
this.velocity = [0,0]
this.friction = 0,1


move
timeFreeze = false
this.cor[0] += this.vel[0]
this.cor[1] += this.vel[1]
this.vel[0] *= 1 - this.friction
this.vel[1] *= 1 - this.friction
boundaries: number[] = [Abstände auf dem Bild nachgucken]
goals: number[] = [Koordinaten auf dem Bild nachgucken]
this.cor[0] < boundaries[0]
this.cor[1] > goals[0] && this.cor[1] < goals[1]
score[1]++
resetPlayer()


this.cor = [canvas.width / 2, canvas.height / 2]
this.cor[0] > canvas.width - boundaries[0]
this.cor[1] > goals[0] && this.cor[1] < goals[1]
score[0]++
resetPlayer1
this.cor = [canvas.width / 2, canvas.height / 2]
this.cor[1] < boundaries[1] || this.cor[1] > canvas.height - boundaries[1]
this.cor = [canvas.width / 2, canvas.height / 2]
this.spawn


spawn
draw an arc with (this.cor[0], this.cor[1], this.size, 0, 2 * Math.PI) in black
getKicked
_targetCor: number[]
_deviation: number
d: number = distance(_targetCor, this.cor)
xWinkel: number = (_targetCor[0] - this.cor[0]) / d
Winkel: number = Math.acos(xWinkel)
bogenDeviation: number = _deviation * Math.PI / 180
newX: number = d * Math.cos(Winkel + (Math.random() - 0.5) * 2 * BogenDeviation)
newY: number = -d * Math.sin(Winkel + (Math.random() - 0.5) * 2 * BogenDeviation)
tpd: number = distance(_targetCor, ballTouch.cor)
tpWinkel: number = (_targetCor[0] - ballTouch.cor[0]) / tpd
tpAcos: number = Math.acos(tpWinkel)
_targetCor[0] > this.cor[0]
winkel = 2 * Math.PI - winkel
_targetCor[1] < this.cor[1]


tpAcos = 2 · Math.PI - tpAcos
this.cor = [ballTouch.cor[0] + ((ballTouch.size + this.size) · Math.cos(tpAcos)),
ballTouch.cor[1] + ((ballTouch.size + this.size) · Math.sin(tpAcos))]
this.vel = [((newTarget[0])) · this.friction, ((newTarget[1])) · this.friction]
this.cor[0] += this.velocity[0]
this.cor[1] += this.velocity[1]
this.velocity[0] *= 1 - this.friction
this.velocity[1] *= 1 - this.friction

main

canvas : HTMLCanvasElement

crc2 : CanvasRenderingContext2D

score : number []

Anzeige : HTMLHeadingElement

FieldDiv : HTMLDivElement = div von Canvas u. Bild

InputColor1 : HTMLInputElement

InputColor2 : HTMLInputElement

form : HTMLFormElement = form der Pre-Game Settings

PlayerColor1 : string

PlayerColor2 : string

Substitutes : Players []

idolStrings : string [] = [" Namen der Assets"]

Vtuber : Interface aus name : string und img : HTMLImgElement

IdolGroup : Vtuber []

sel1 : HTMLSelectElement

sel2 : HTMLSelectElement

minAcc : number

maxAcc : number

minSpeed : number

maxSpeed : number

ball : Ball

timeFreeze : boolean

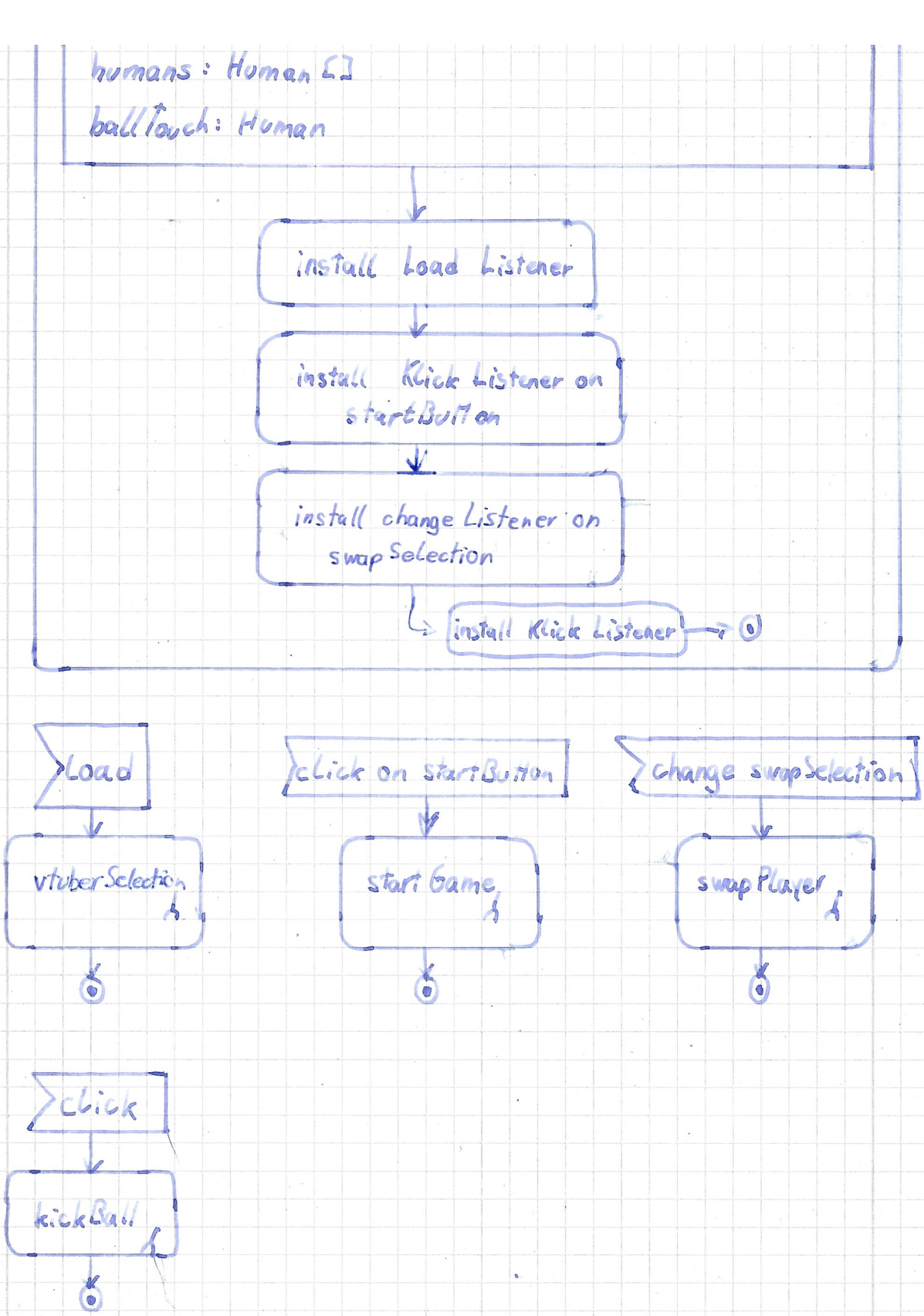
vTuber1 : Vtuber

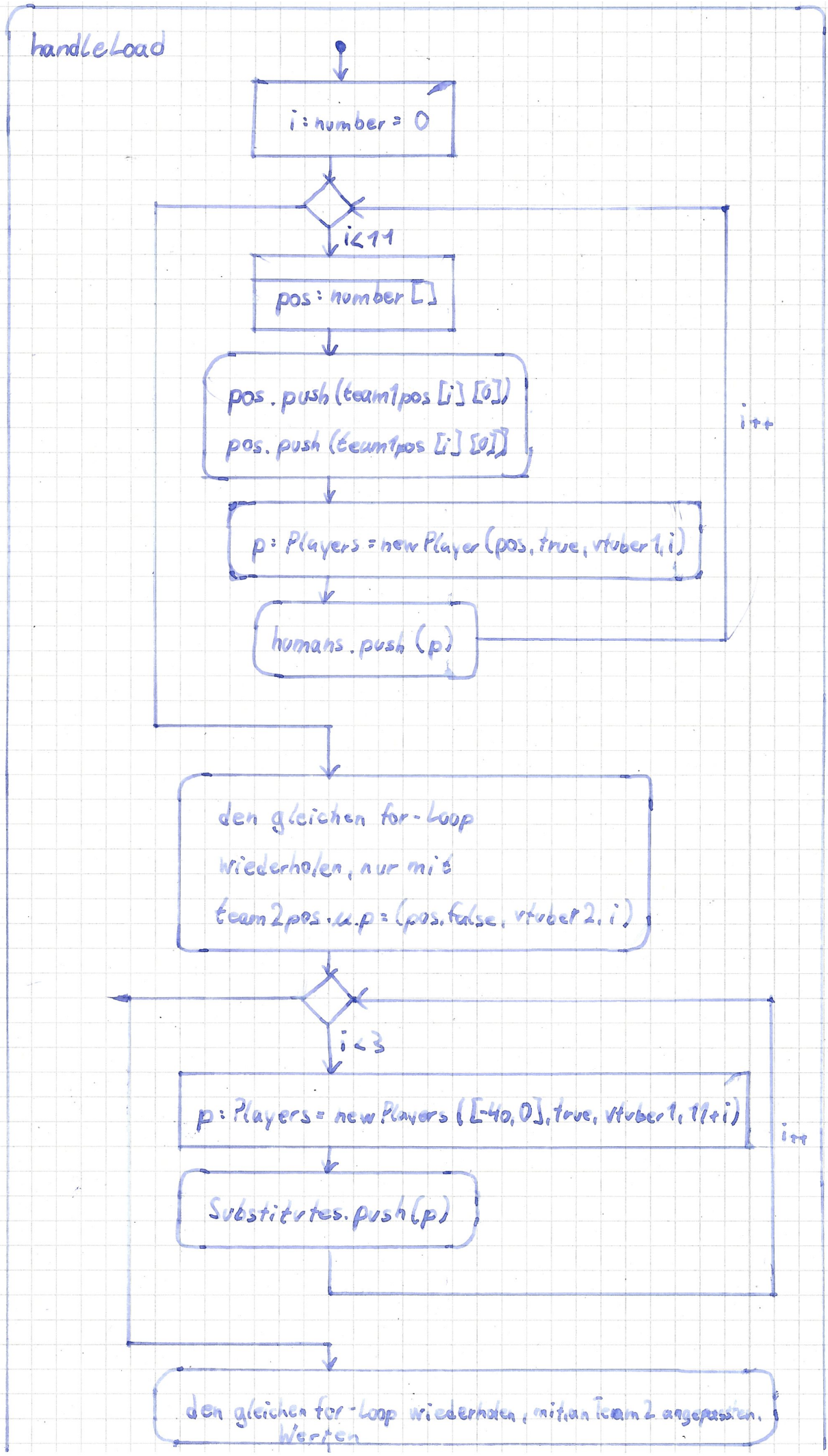
vTuber2 : Vtuber

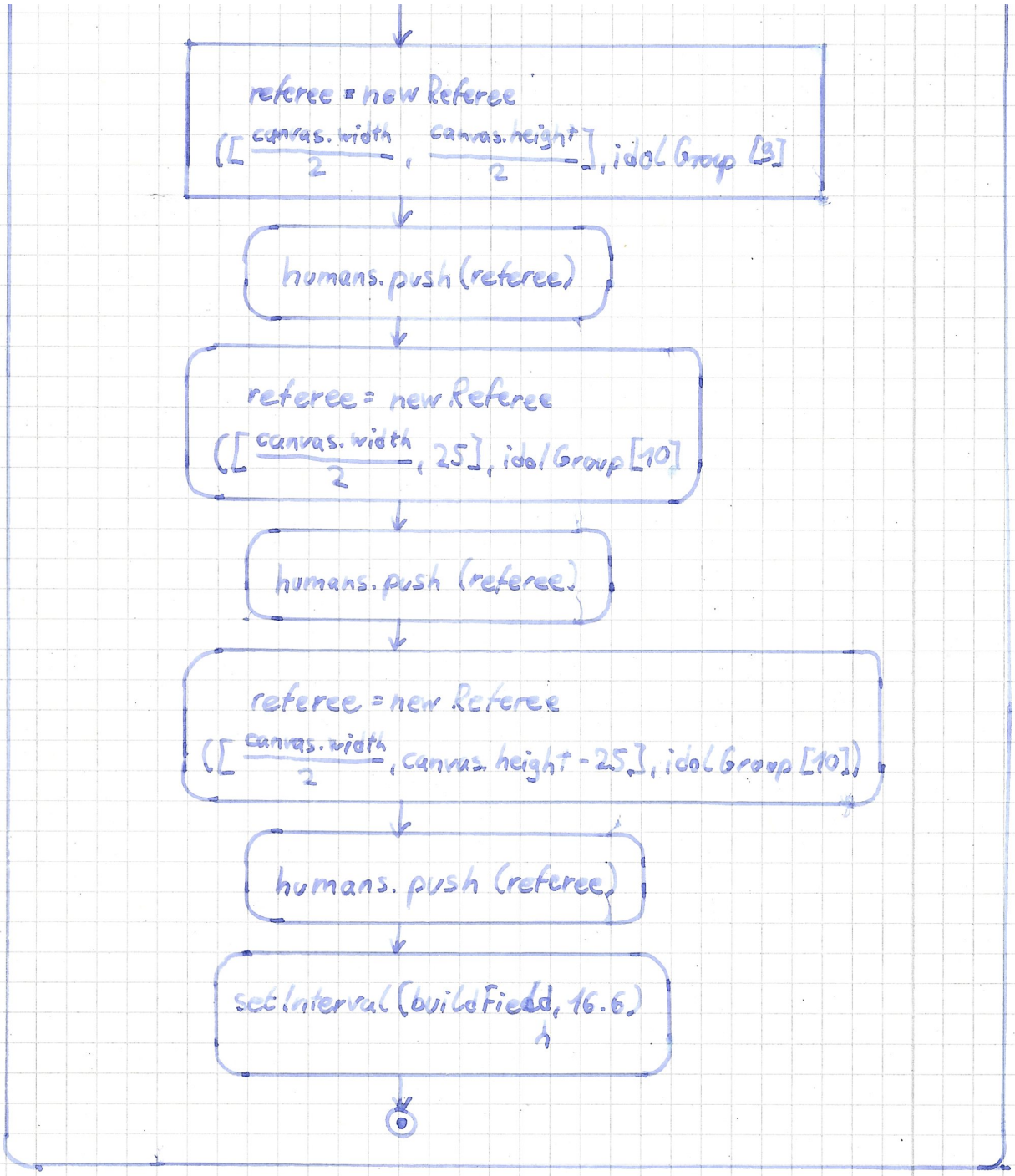
swapIndex : number

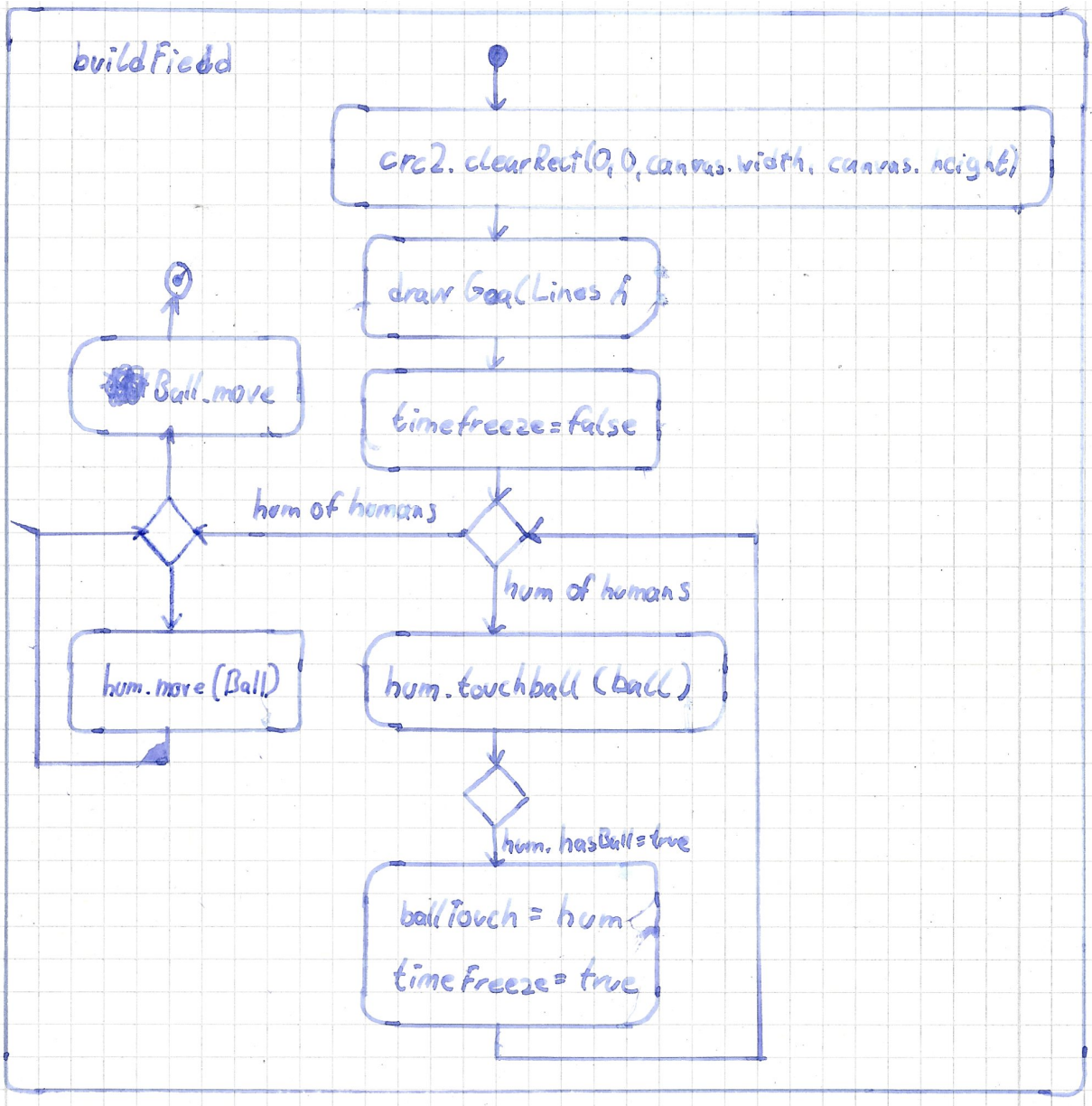
swapSelection : HTMLSelectElement

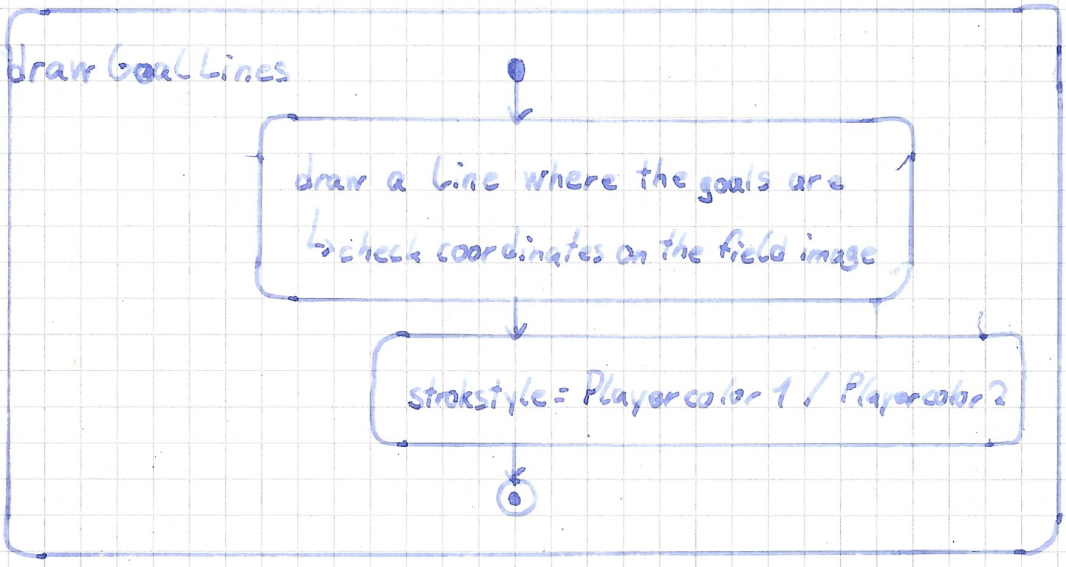
startButton : HTMLButtonElement = Button zum Spielstar-

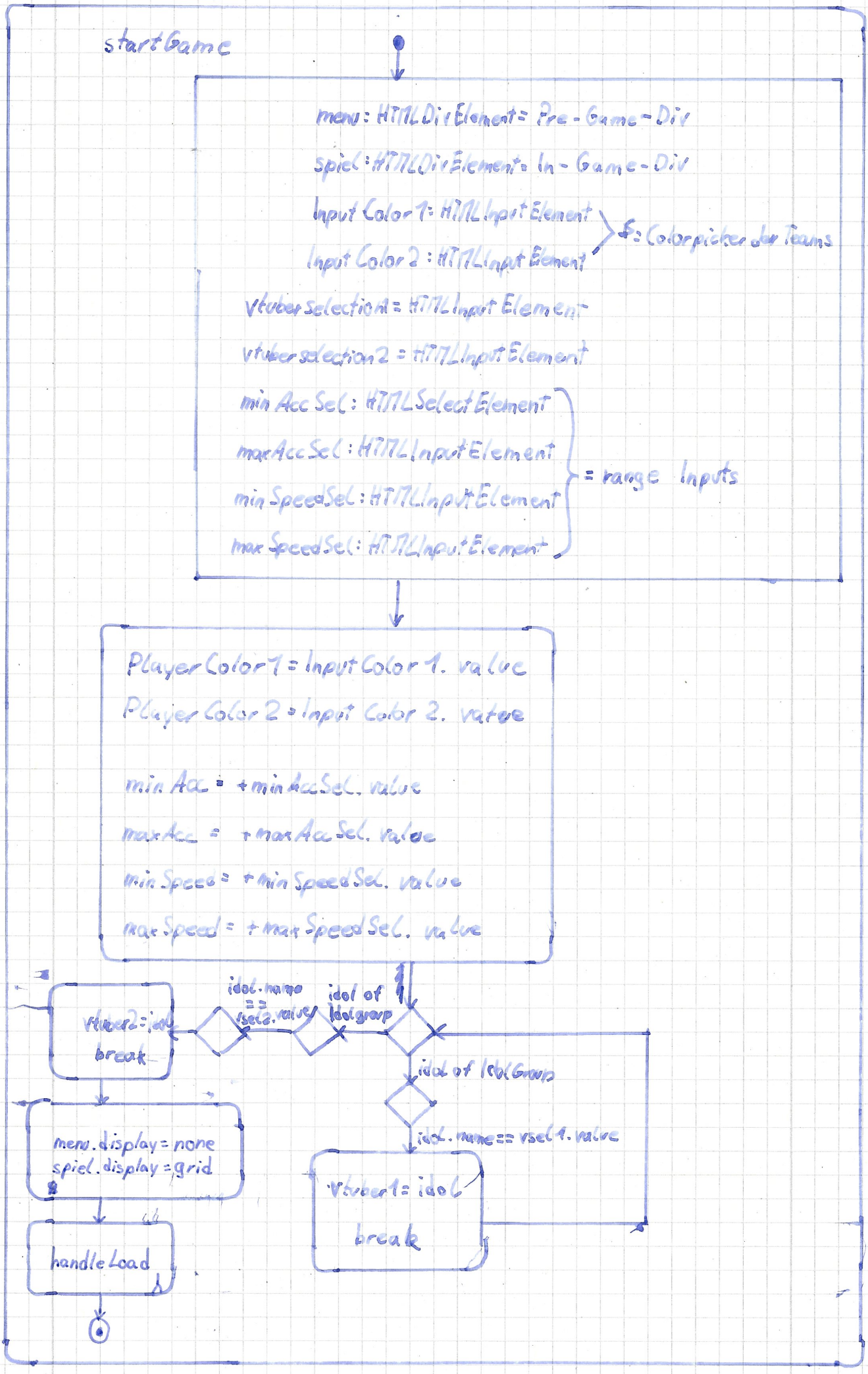

humans : Human []
ballTouch : Human
install Load Listener
install KClick Listener on startButton
install change Listener on swapSelection
install KClick Listener
Load
vtuberSelection
click on startButton
startGame
change swapSelection
swapPlayer
click
kickBall

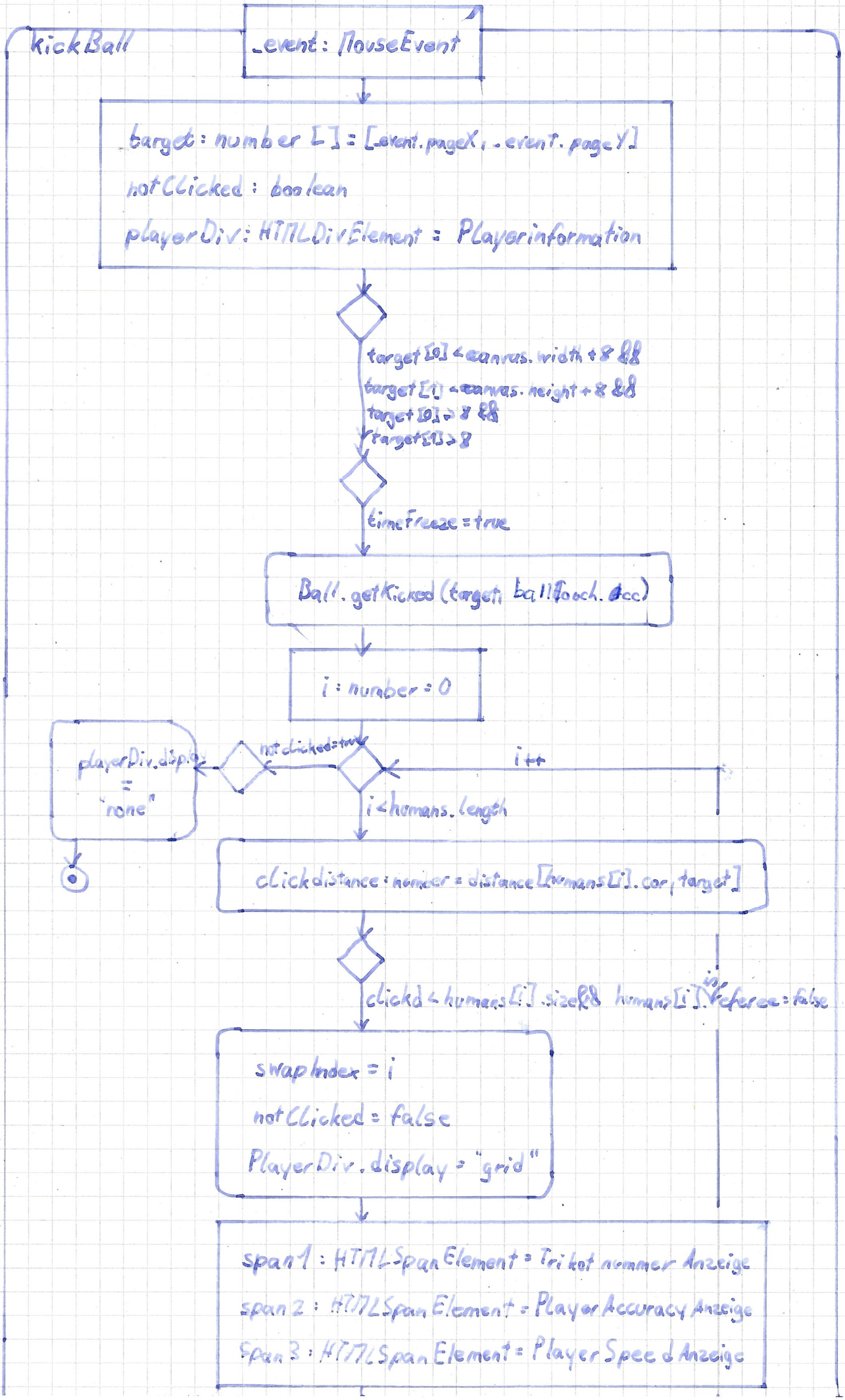

handleLoad
i : number = 0
i<11
pos : number [ ]
pos . push (team1pos [i] [0]),
pos. push (team1pos [i] [0])
i++
p : Players = new Player (pos, true, vtuber1, i)
humans . push (p)
den gleichen for-Loop
wiederholen, nur mit
team2pos . u.p = (pos, false, vtuber2, i)
i<3
p : Players = new Players ([-40, 0], true, vtuber1, 11+i)
i++
Substitutes. push (p)
den gleichen for-Loop wiederholen, mit nun Team 2 angepassten Werten

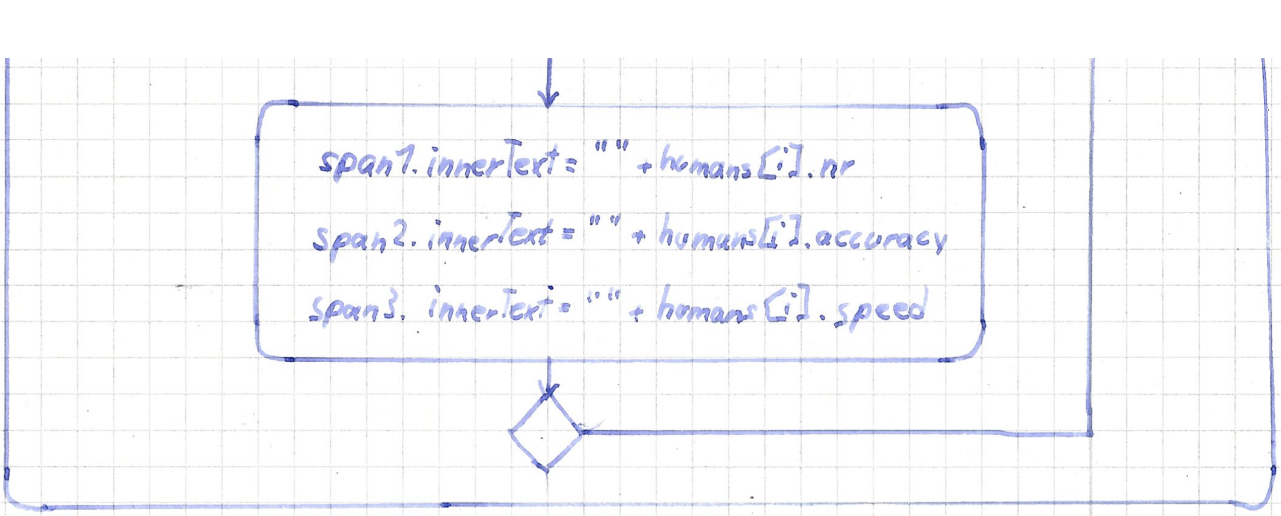

referee = new Referee
([canvas.width/2, canvas.height/2], idolGroup[3]
humans.push(referee)
referee = new Referee
([canvas.width/2, 25], idolGroup[10]
humans.push(referee)
referee = new Referee
([canvas.width/2, canvas.height-25], idolGroup[10])
humans.push(referee)
setInterval(buildField, 16.6)

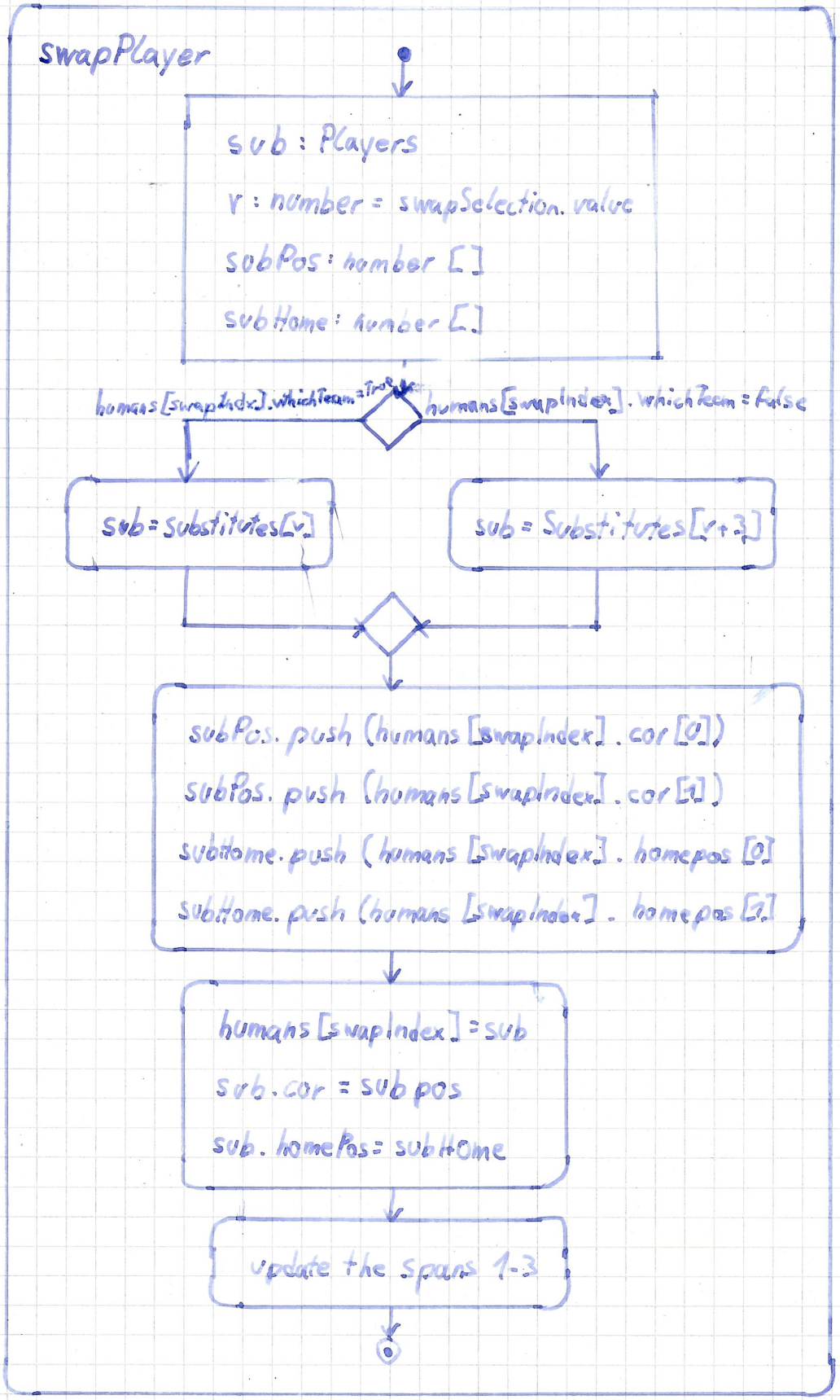

buildField
ctx2.clearRect(0, 0, canvas.width, canvas.height)
drawGoalLines()
timeFreeze=false
Ball.move
num of humans
num of humans
hum.move(Ball)
hum.touchBall(Ball)
hum.hasBall=true
ballTouch = hum
timeFreeze=true

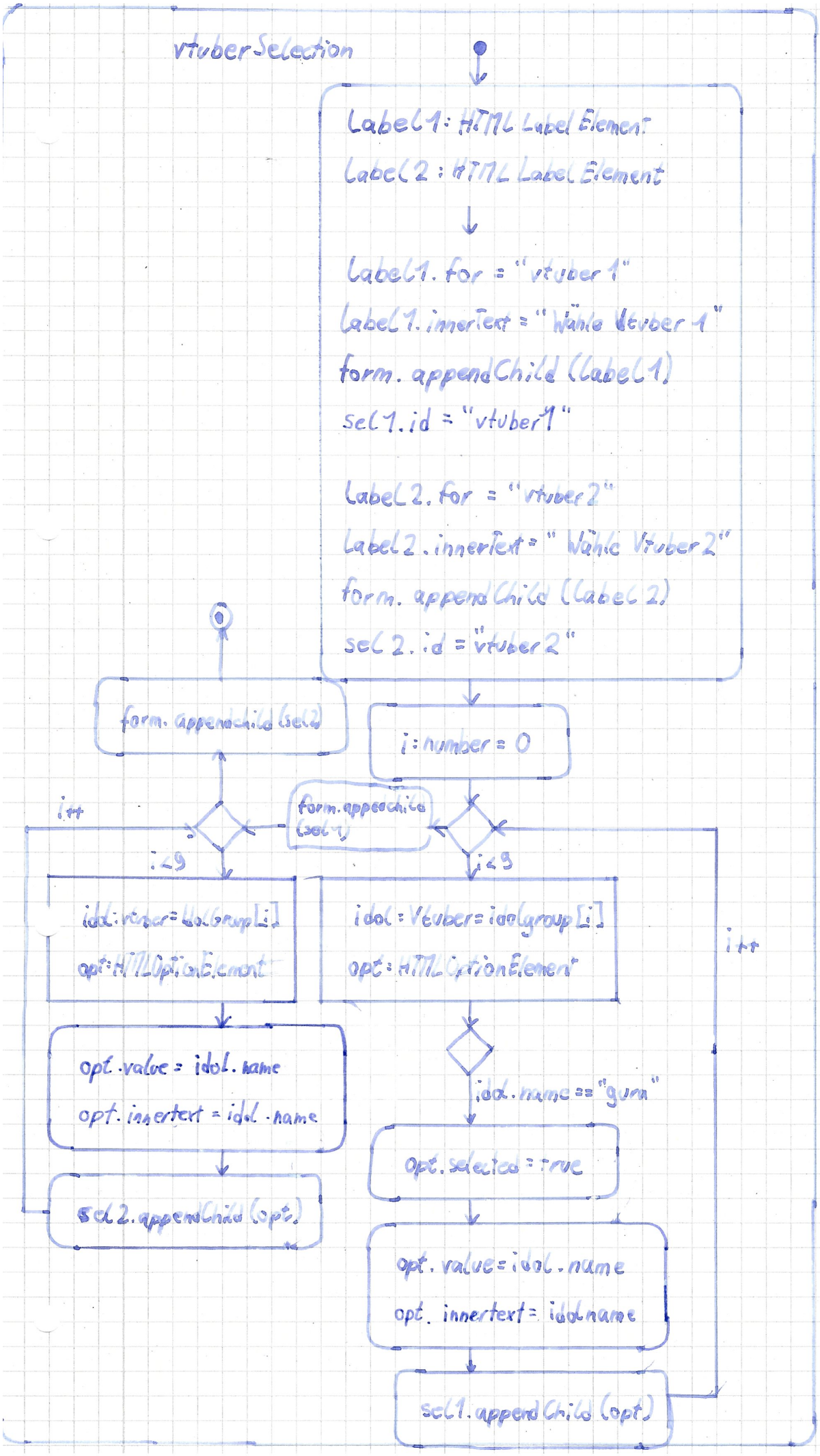

drawGoalLines
draw a Line where the goals are
check coordinates on the field image
strokStyle = Playercolor1 / Playercolor2

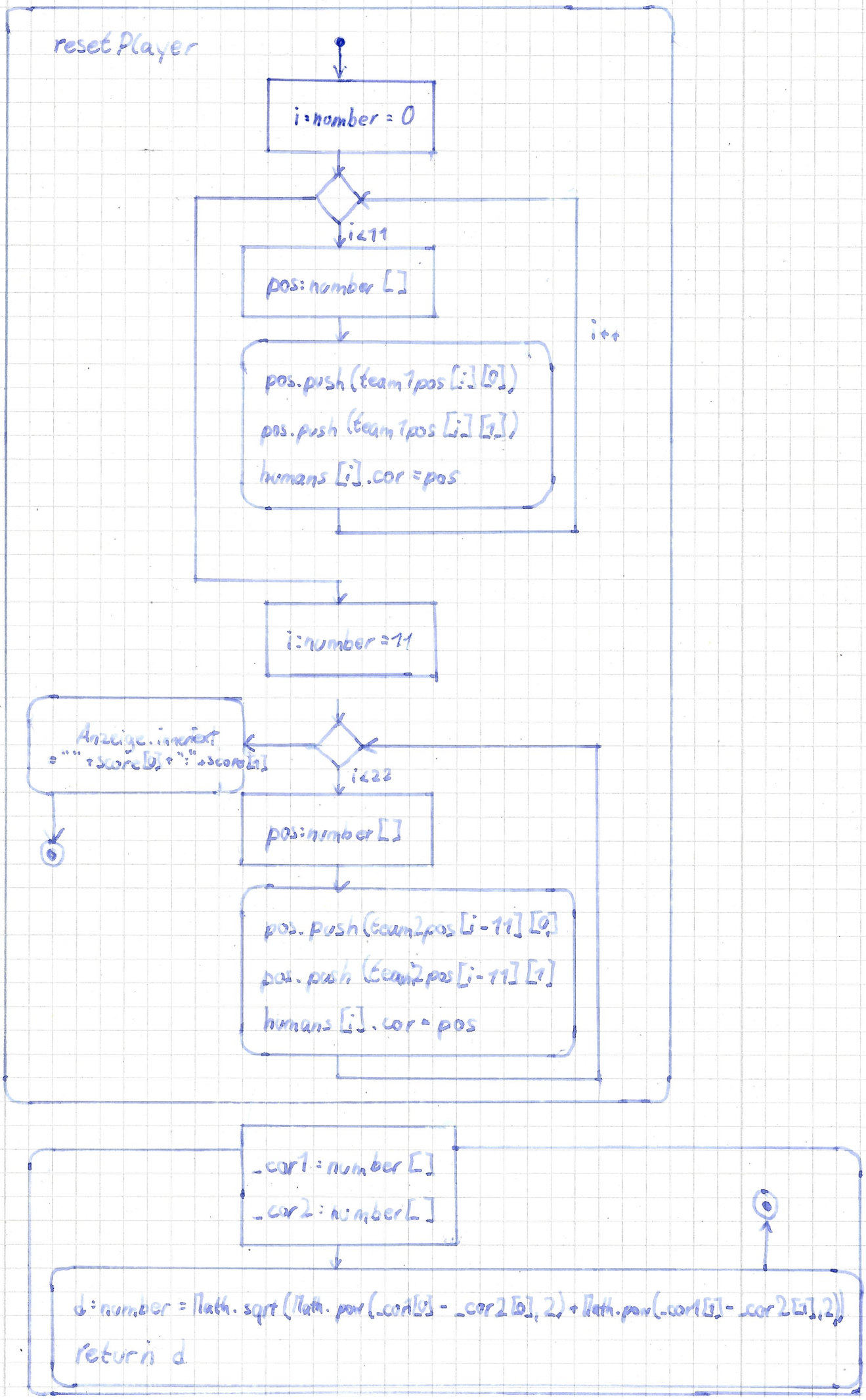

startGame
menu: HTMLDivElement = Pre-Game-Div
spiel: HTMLDivElement = In-Game-Div
InputColor1: HTMLInputElement
InputColor2: HTMLInputElement
} = Colorpicker der Teams
Vtuberselection1 = HTMLInputElement
vtuberselection2 = HTMLInputElement
minAccSel: HTMLSelectElement
maxAccSel: HTMLInputElement
minSpeedSel: HTMLInputElement
maxSpeedSel: HTMLInputElement
} = range Inputs
PlayerColor1 = InputColor1.value
PlayerColor2 = InputColor2.value
minAcc = +minAccSel.value
maxAcc = +maxAccSel.value
minSpeed = +minSpeedSel.value
maxSpeed = +maxSpeedSel.value
idol.name == vsel2.value
idol of Idolgroup
vtuber2 = idol
break
idol of IdolGroup
idol.name == vsel1.value
vtuber1 = idol
break
menu.display = none
spiel.display = grid
handleLoad


kickBall
_event: MouseEvent
target: number [] = [_event.pageX, _event.pageY]
notClicked: boolean
playerDiv: HTMLDivElement = Playerinformation
target[0] < canvas.width + 8 &&
target[1] < canvas.height + 8 &&
target[0] > 8 &&
target[1] > 8
timeFreeze = true
Ball.getKicked(target, ballTouch.dcc)
i: number = 0
playerDiv.display = "none"
notClicked=true
i++
i<humans.length
clickdistance: nomber = distance[humans[i].cor, target]
clickd < humans[i].sizefd
humans[i].referee = false
swapIndex = i
notClicked = false
PlayerDiv.display = "grid"
span1: HTMLSpanElement = Trikot nommer Anzeige
span2: HTMLSpanElement = Player Accuracy Anzeige
Span3: HTMLSpanElement = Player Speed Anzeige


span1.innerText = "" + humans[i].nr
span2.innerText = "" + humans[i].accuracy
span3.innerText = "" + humans[i].speed


swapPlayer
sub : Players
r : number = swapSelection.value
subPos : number []
subHome : number []
humans[swapIndex].whichTeam = True
humans[swapIndex].whichTeam = False
sub = substitutes[r]
sub = Substitutes[r+3]
subPos.push (humans[swapIndex].cor[0])
subPos.push (humans[swapIndex].cor[1])
subHome.push (humans[swapIndex].homepos[0]
subHome.push (humans[swapIndex].homepos[1]
humans[swapIndex] = sub
sub.cor = sub pos
sub.homePos = subHome
update the spans 1-3


vtuberSelection
Label1: HTML Label Element
Label2 : HTML Label Element
Label1.for = "vtuber1"
Label1.innerText = "Wähle Vtuber 1"
form.appendChild (Label1)
sel1.id = "vtuber1"
Label2.for = "vtuber2"
Label2.innerText = "Wähle Vtuber2"
form.appendChild (Label2)
sel2.id = "vtuber2"
form.appendchild (sel2)
i: number = 0
i++
form.appendchild (sel1)
i<9
i<9
idol: Vtuber = idol[group[i]]
opt: HTMLOptionElement
idol: Vtuber = idol[group[i]]
opt: HTMLOptionElement
i++
opt.value = idol.name
opt.innertext = idol.name
idol.name == "gura"
sel2.appendChild (opt)
opt.selected = true
opt.value = idol.name
opt.innertext = idol.name
sel1.appendChild (opt)


resetPlayer
i:number = 0
i<11
pos:number[]
pos.push(team1pos[i][0])
pos.push(team1pos[i][1])
humans[i].cor=pos
i++
i:number=11
Affichage.innerText = "" + score[0] + ":" + score[1]
i<22
pos:number[]
pos.push(team2pos[i-11][0])
pos.push(team2pos[i-11][1])
humans[i].cor=pos
_cor1:number[]
_cor2:number[]
d:number = Math.sqrt(Math.pow(_cor1[0] - _cor2[0], 2) + Math.pow(_cor1[1] - _cor2[1], 2))
return d