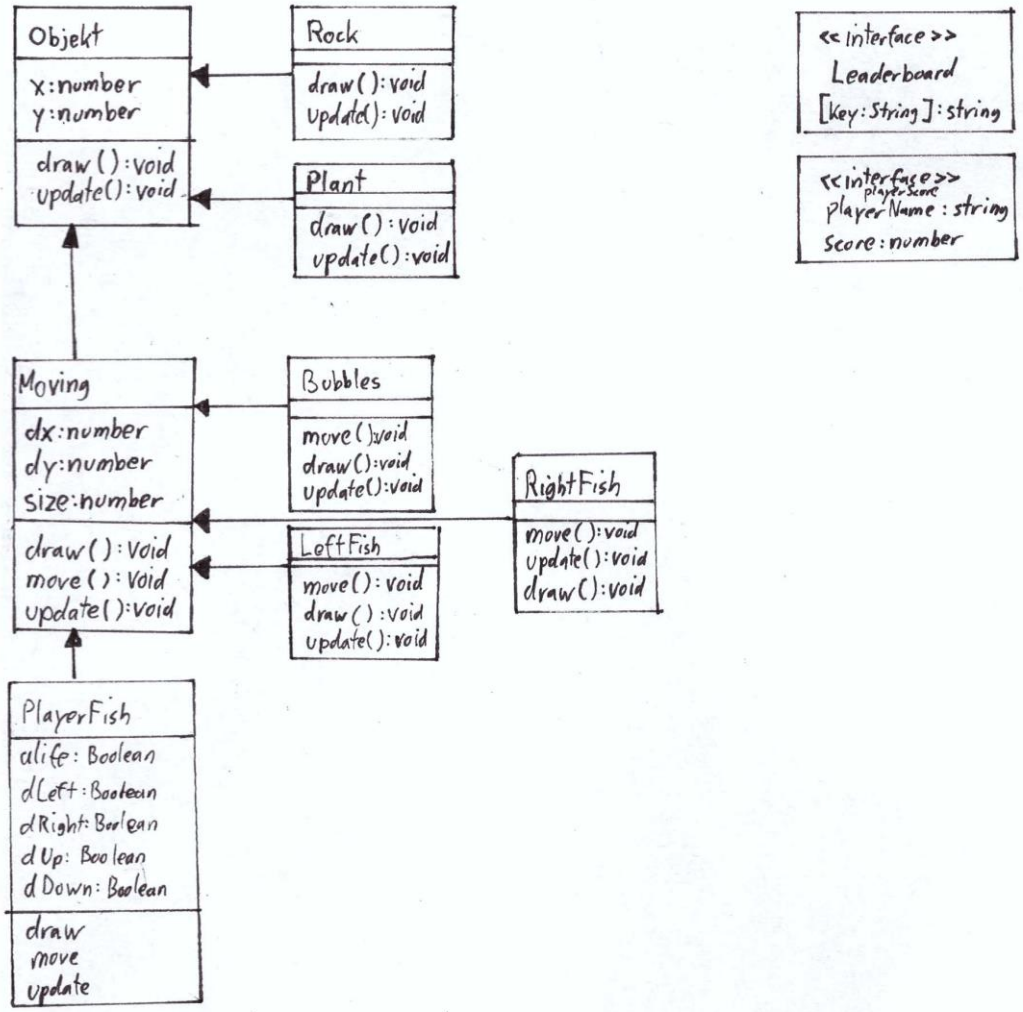
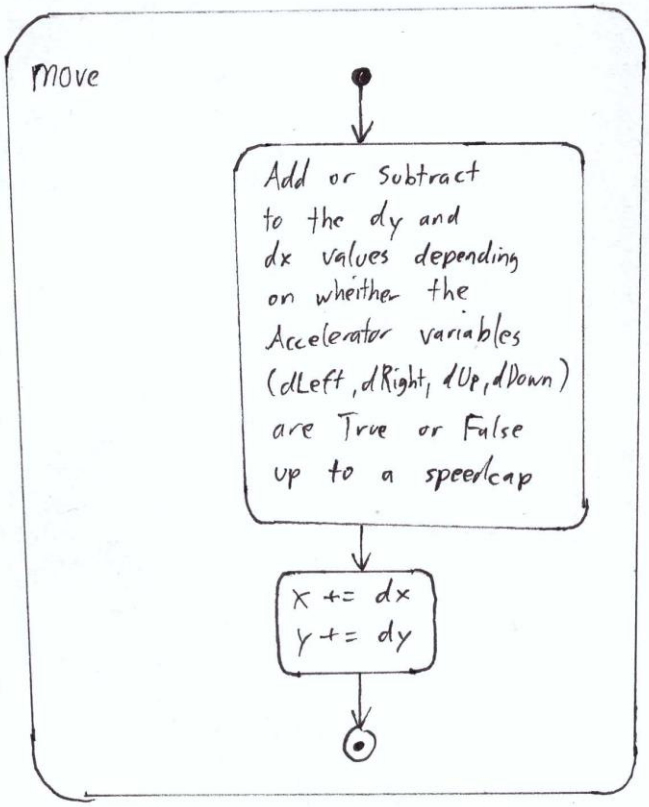
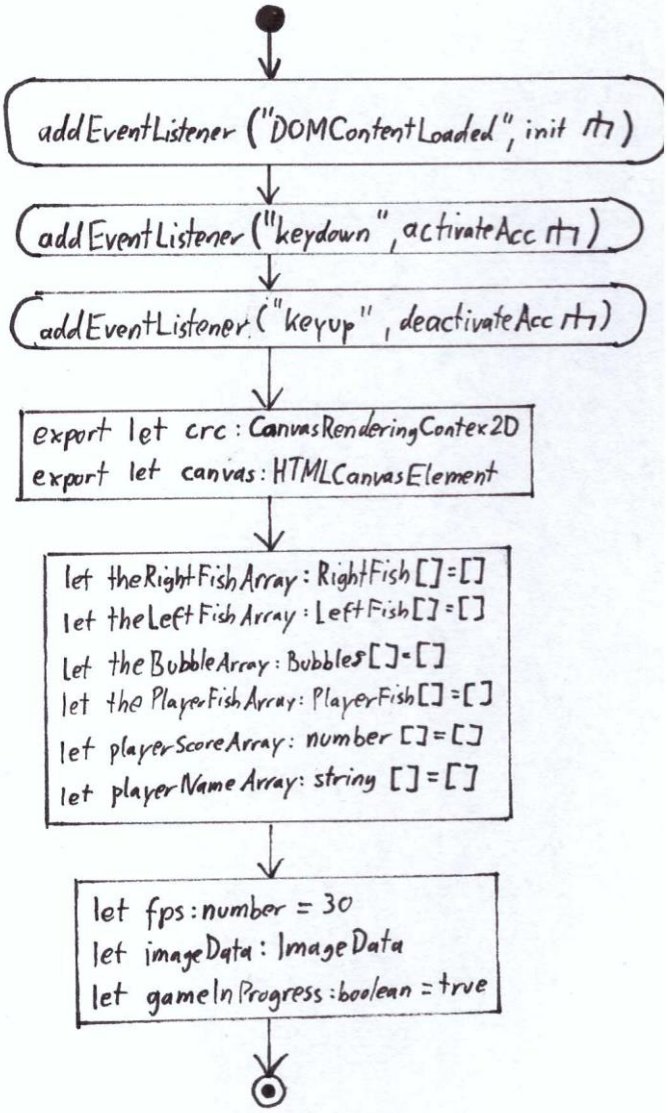
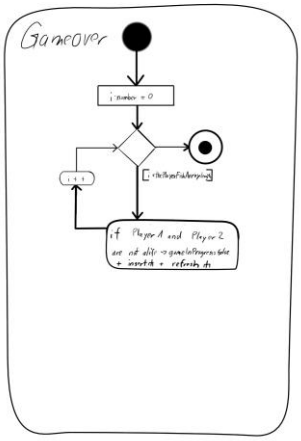
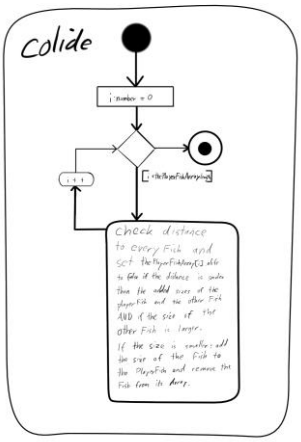
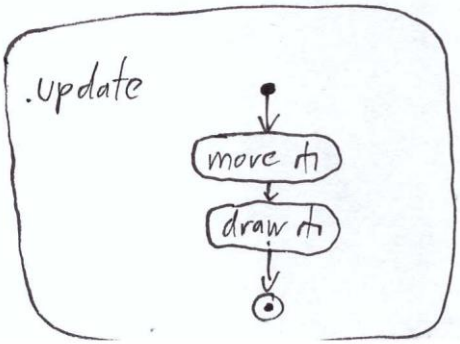

Objekt
x:number
y:number
draw():void
update():void
Rock
draw():void
updated():void
Plant
draw():void
update():void
<<interface>>
Leaderboard
[key:String]:string
<<interfase>>
playerScore
playerName:string
Score:number
Moving
dx:number
dy:number
size:number
draw():void
move():void
update():void
Bubbles
move()void
draw():void
update():void
RightFish
move():void
update():void
draw():void
LeftFish
move():void
draw():void
update():void
PlayerFish
alife:Boolean
dLeft:Boolean
dRight:Boolean
dUp:Boolean
dDown:Boolean
draw
move
update

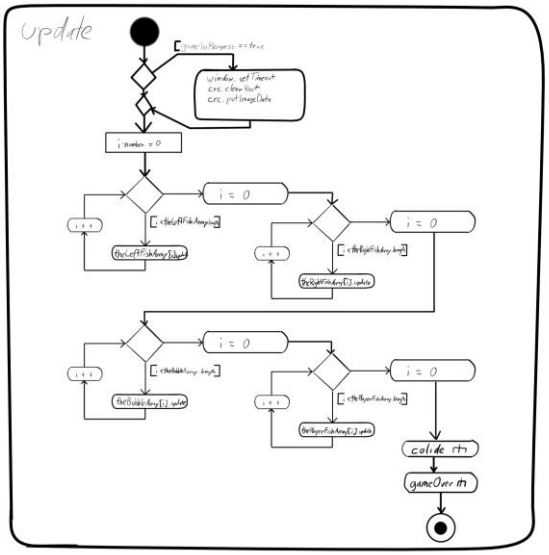

move
Add or Subtract to the dy and dx values depending on whether the Accelerator variables (dLeft, dRight, dUp, dDown) are True or False up to a speedcap
x += dx
y += dy

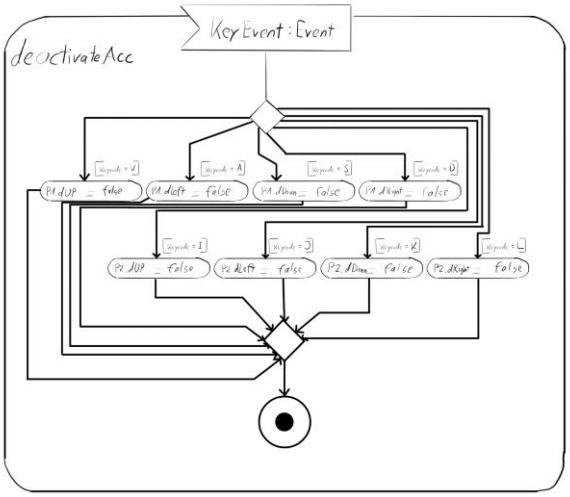

addEventListener ("DOMContentLoaded", init fn)
addEventListener ("keydown", activateAcc fn)
addEventListener ("keyup", deactivateAcc fn)
export let crc : CanvasRenderingContex2D
export let canvas : HTMLCanvasElement
let theRightFishArray : RightFish [] = []
let theLeftFishArray : LeftFish [] = []
let theBubbleArray : Bubbles [] = []
let thePlayerFishArray : PlayerFish [] = []
let playerScoreArray : number [] = []
let playerNameArray : string [] = []
let fps : number = 30
let imageData : ImageData
let gameInProgress : boolean = true

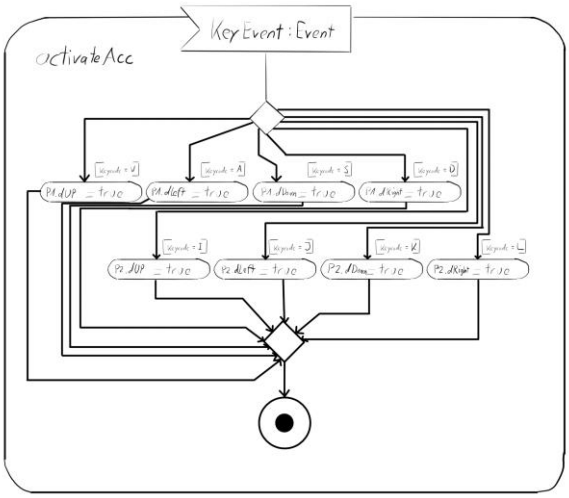

Gameover
i : number = 0
i++
[< thePlayerFishArray.length]
if PlayerA and Player2 are not alive → gameInProgress false + noteA + refresh th

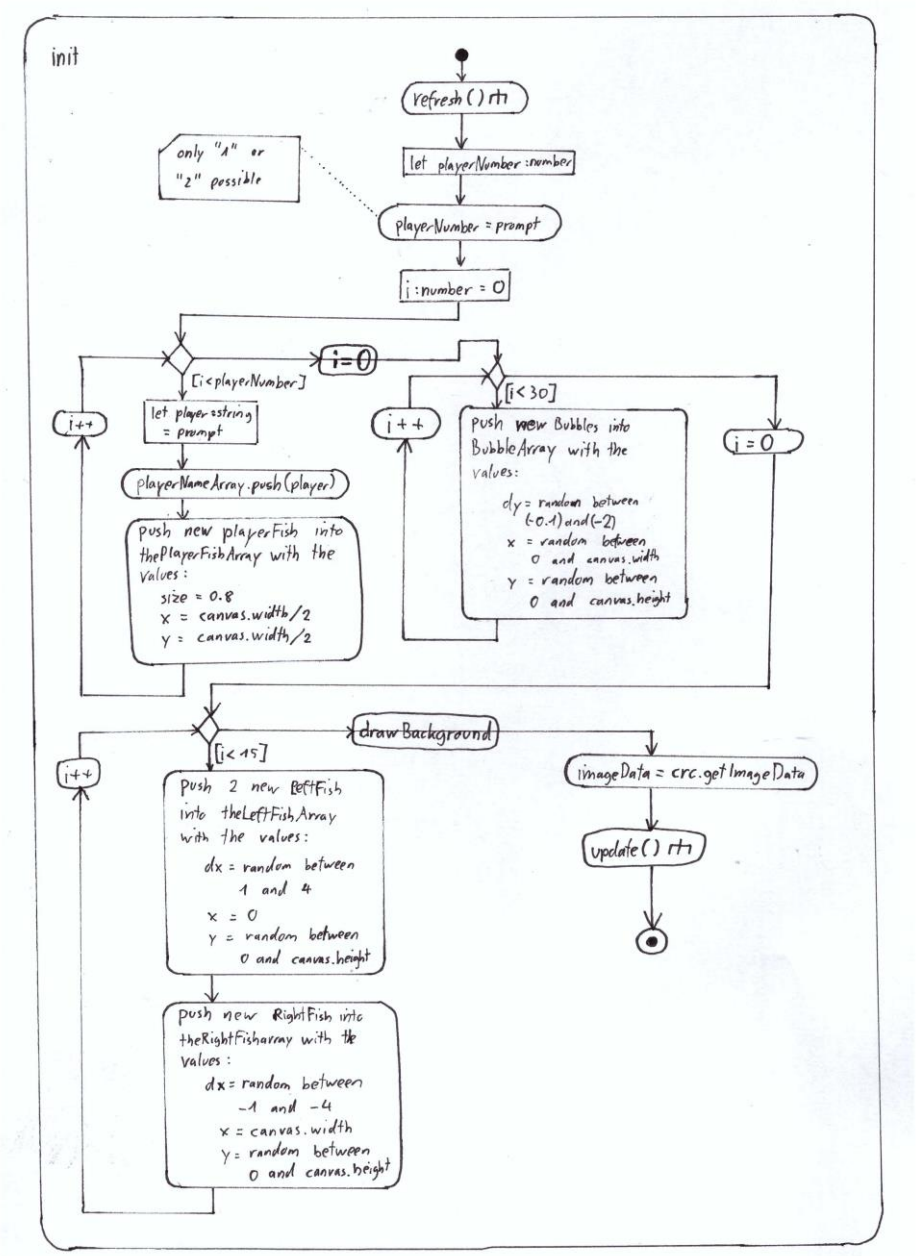

Colide
i : number = 0
i++
[< thePlayerFishArray.length]
check distance to every Fish and set thePlayerFishArray[i] alive to false if the distance is under then the added size of the player Fish and the other Fish AND if the size of the other Fish is larger.
If the size is smaller: add the size of the Fish to the PlayerFish and remove the Fish from its Array.

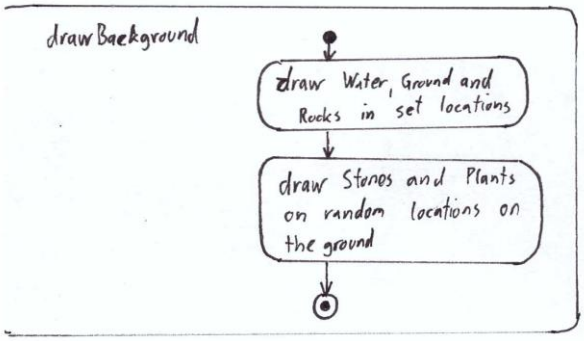

.Update
move fn
draw fn

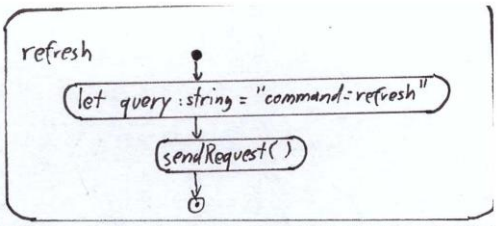

Update
[gameInProgress == true]
window.setTimeout
crc.clearRect
crc.putImageData
i : number = 0
i = 0
i++
[< theLeftFishArray.length]
theLeftFishArray[i].update
[< theRightFishArray.length]
theRightFishArray[i].update
[< theBubbleArray.length]
theBubbleArray[i].update
[< thePlayerFishArray.length]
thePlayerFishArray[i].update
colide fn
gameOver fn

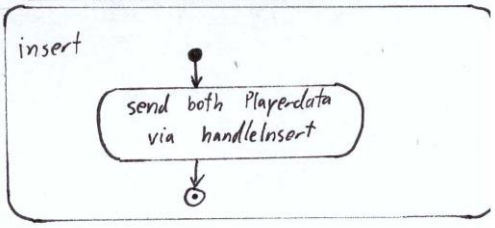

KeyEvent : Event
deactivateAcc
keycode = W
keycode = A
keycode = S
keycode = D
P1.dUP = false
P1.dLeft = false
P1.dDown = false
P1.dRight = false
keycode = I
keycode = J
keycode = K
keycode = L
P2.dUP = false
P2.dLeft = false
P2.dDown = false
P2.dRight = false


KeyEvent : Event
activateAcc
keycode = W
keycode = A
keycode = S
keycode = D
P1.dUP = true
P1.dLeft = true
P1.dDown = true
P1.dRight = true
keycode = I
keycode = J
keycode = K
keycode = L
P2.dUP = true
P2.dLeft = true
P2.dDown = true
P2.dRight = true


init
refresh() rtn
let playerNumber :number
only "1" or "2" possible
playerNumber = prompt
i :number = 0
i=0
[i<playerNumber]
let playerstring = prompt
playerNameArray.push(player)
push new playerFish into thePlayerFishArray with the values:
size = 0.8
x = canvas.width/2
y = canvas.width/2
i++
[i<30]
push new Bubbles into BubbleArray with the values:
dy = random between (-0.1) and (-2)
x = random between 0 and canvas.width
y = random between 0 and canvas.height
i++
i = 0
drawBackground
[i<15]
push 2 new LeftFish into theLeftFishArray with the values:
dx = random between 1 and 4
x = 0
y = random between 0 and canvas.height
push new RightFish into theRightFisharray with the values:
dx = random between -1 and -4
x = canvas.width
y = random between 0 and canvas.height
i++
imageData = crc.getImageData
update() rtn


drawBackground
draw Water, Ground and Rocks in set locations
draw Stones and Plants on random locations on the ground


refresh
let query : string = "command:refresh"
sendRequest( )


insert
send both Playerdata via handleInsert