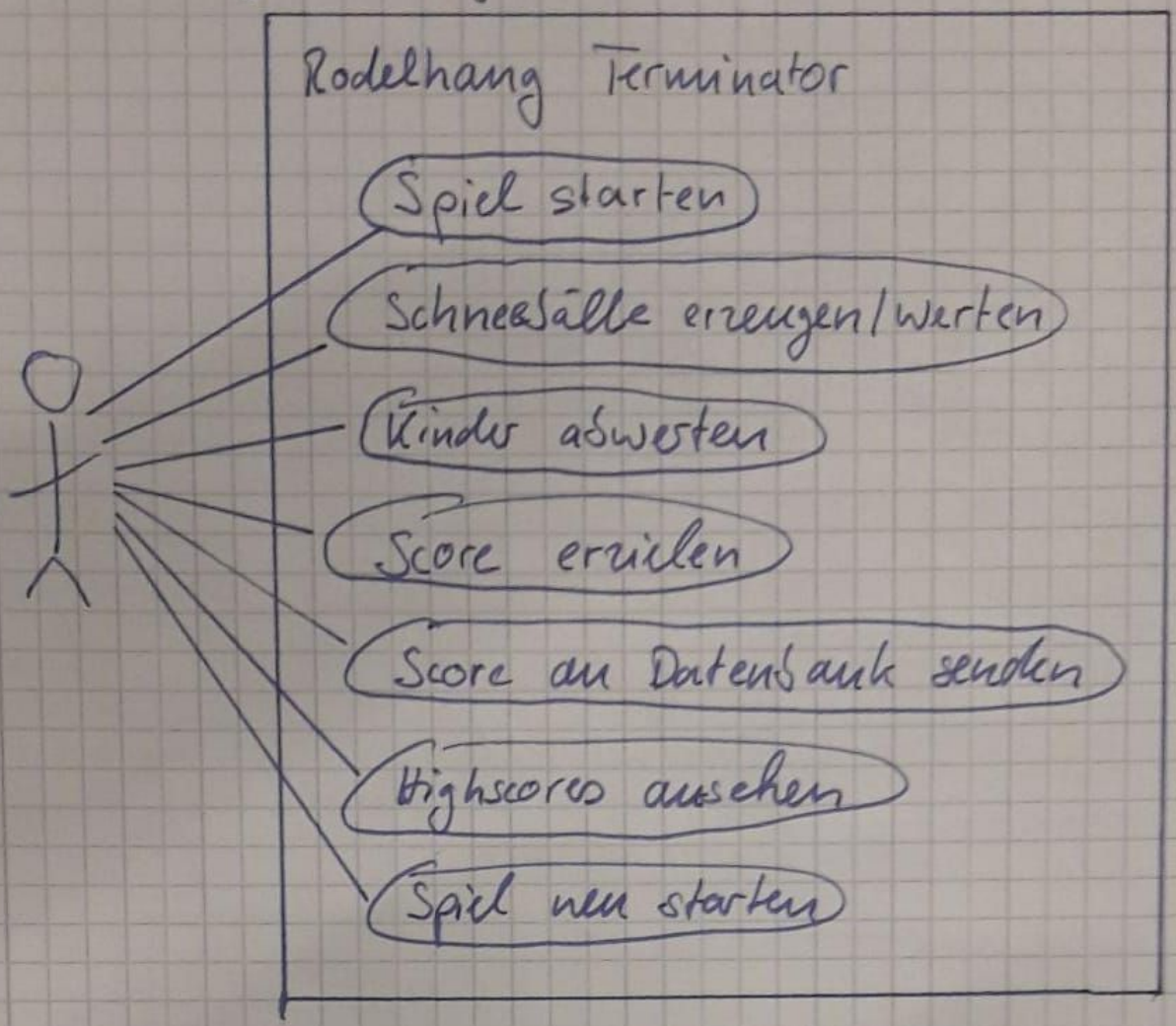
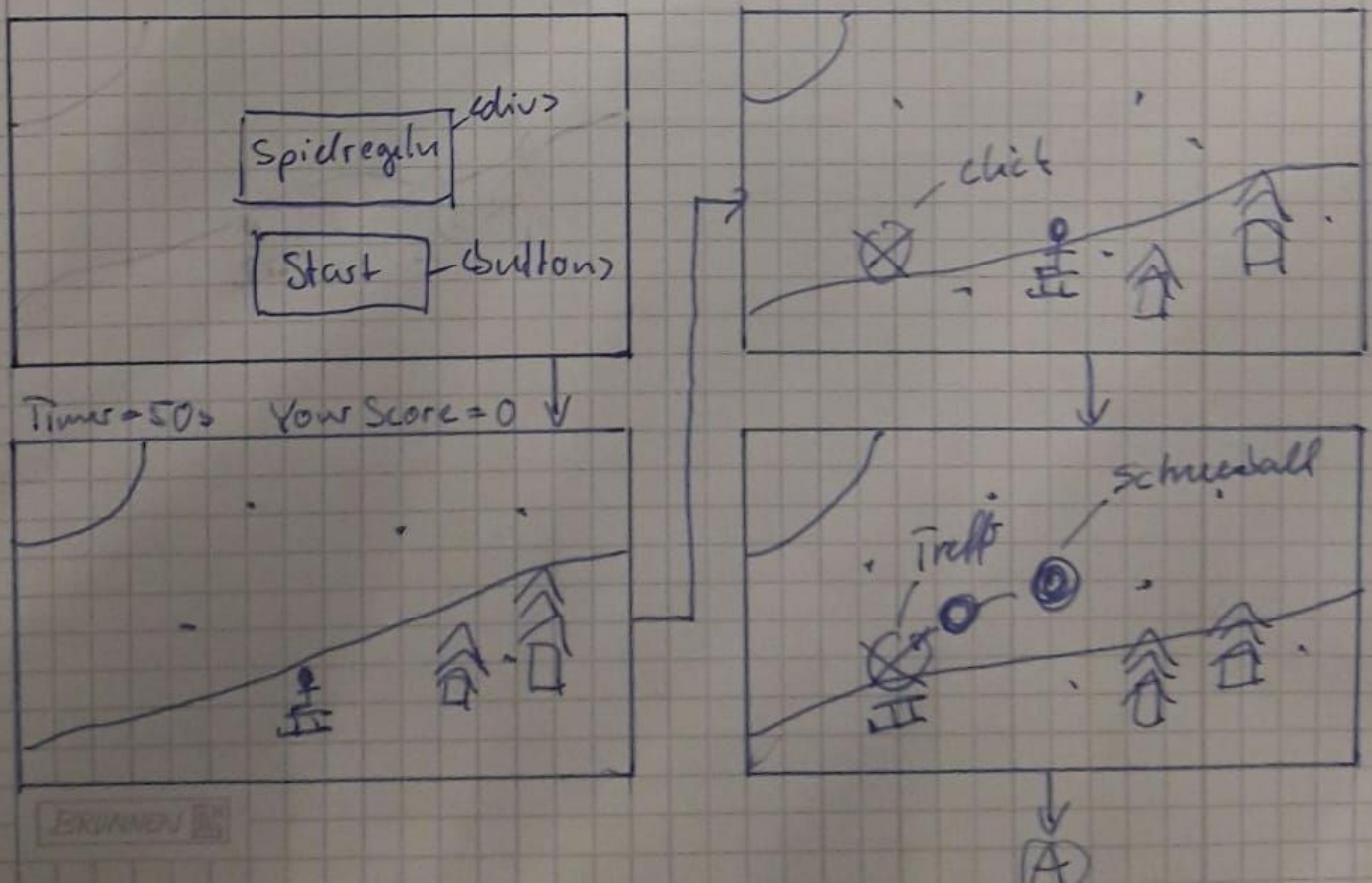
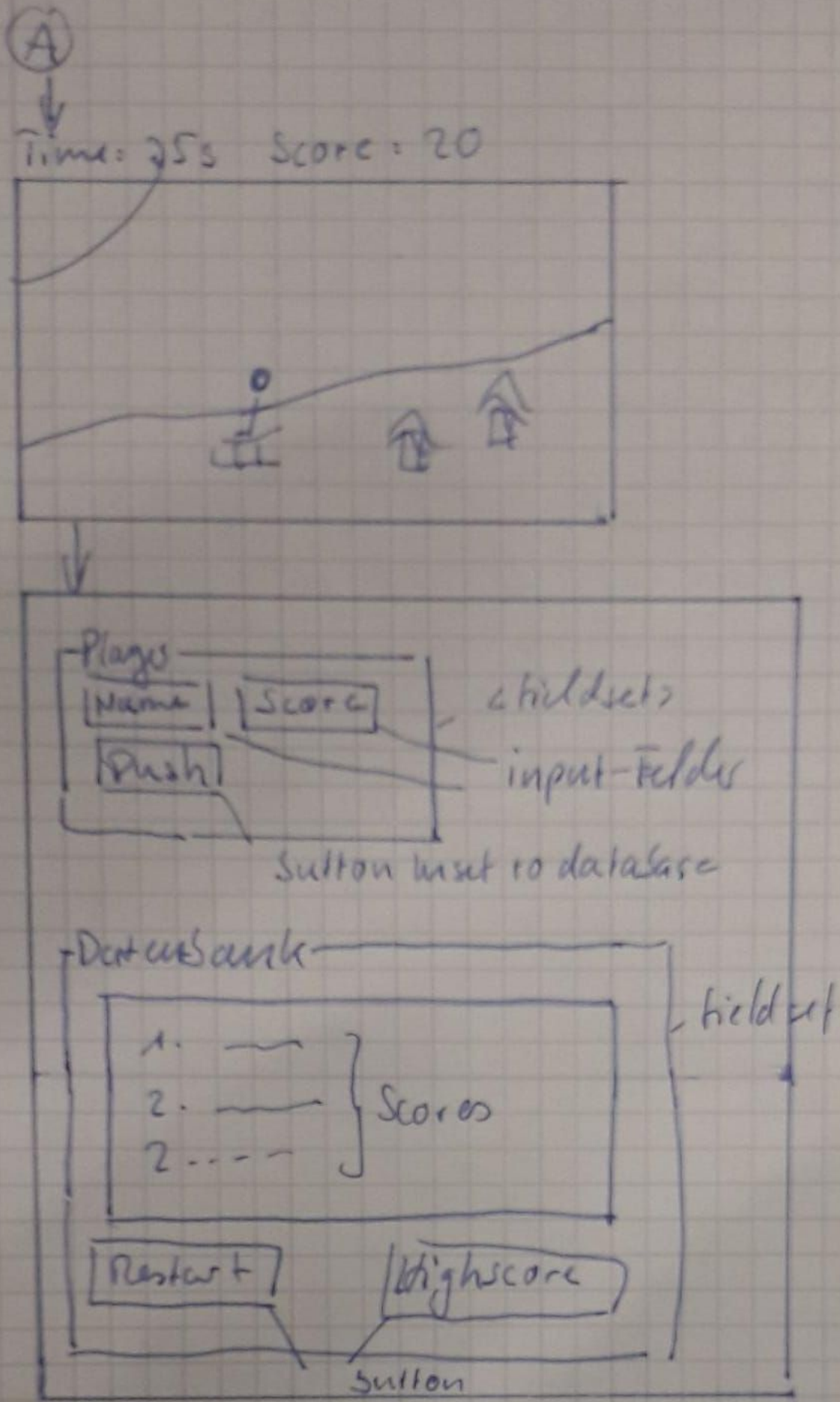
Anwendungsfalldiagramm:

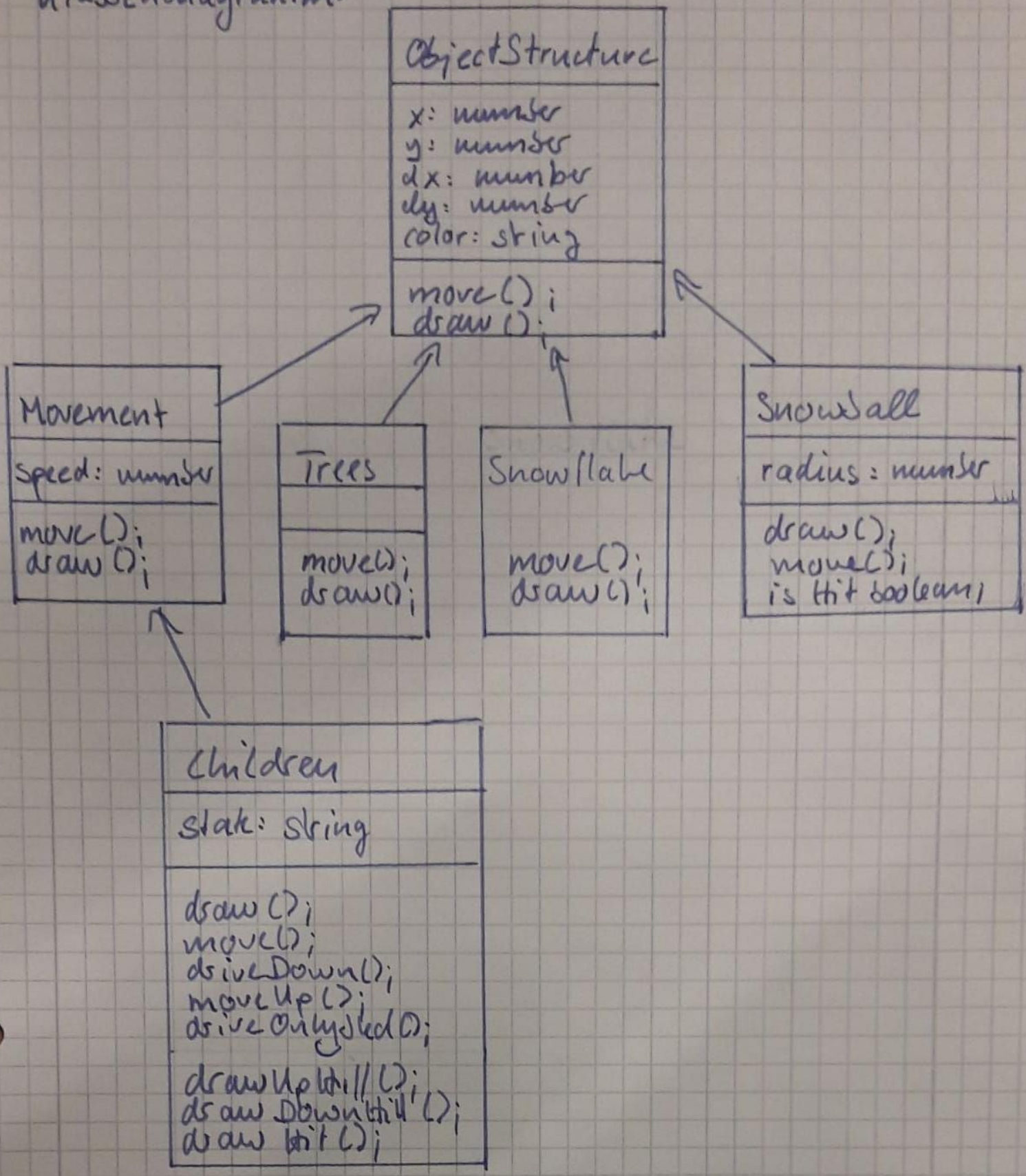


Desktop-Verlauf:

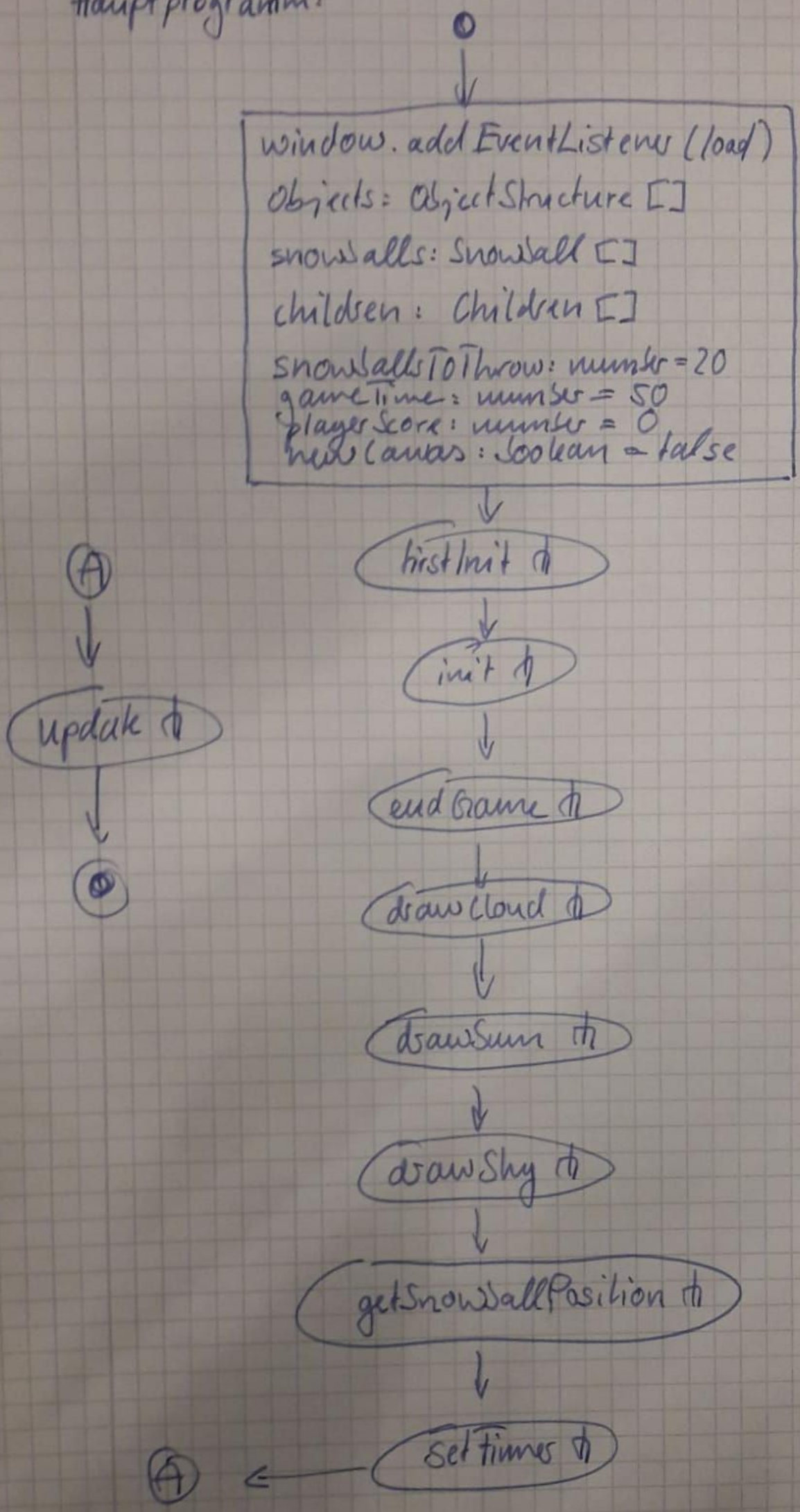



A
Time: 25s Score: 20
Player
Name
Score
Push
<fieldset>
input-Felder
Sutton insert to database
Datenbank
1. ——
2. ——
2. ——
Scores
fieldset
Restart
Highscore
Sutton

Klassendiagramm:



Hauptprogramm:



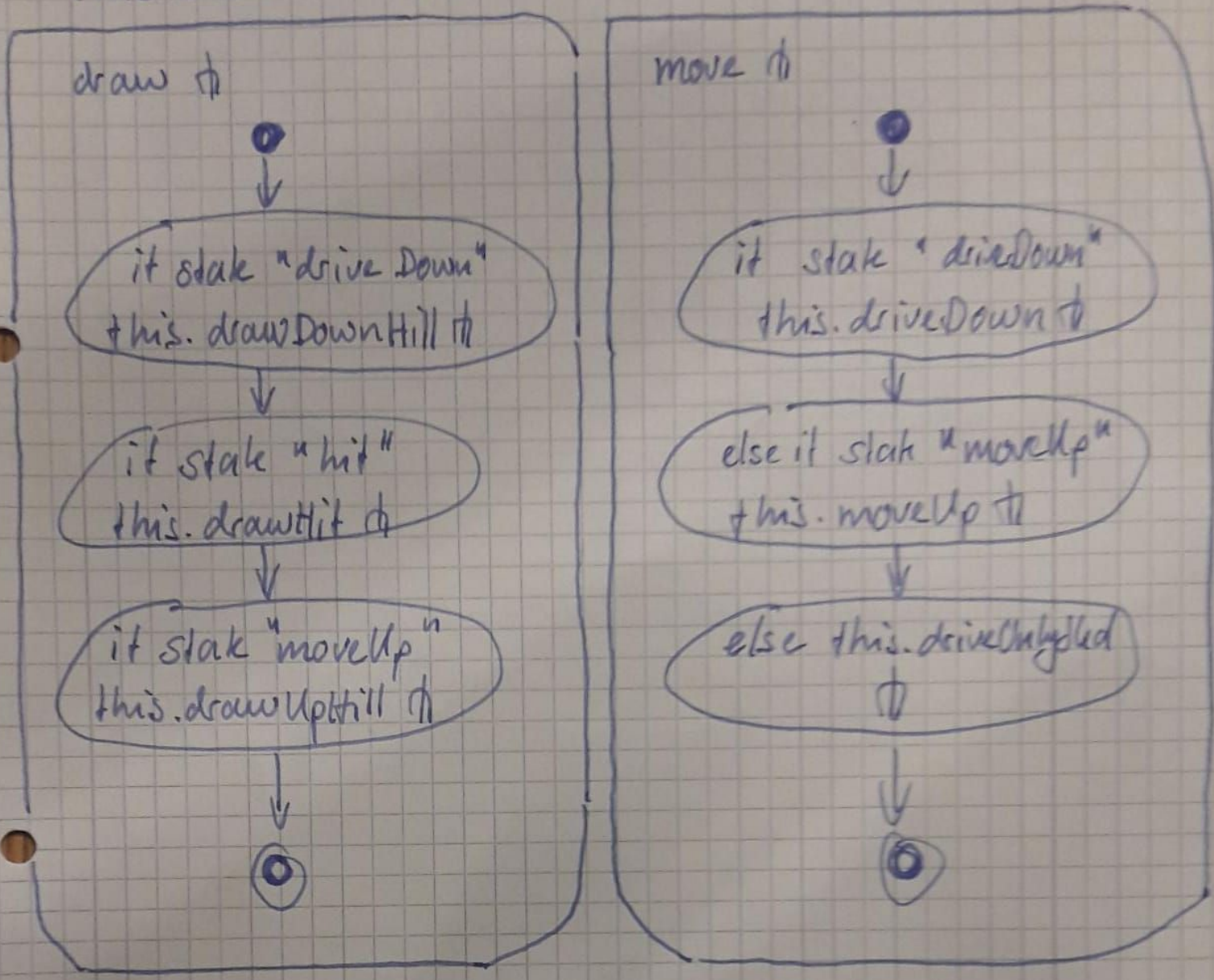
Klasse Children

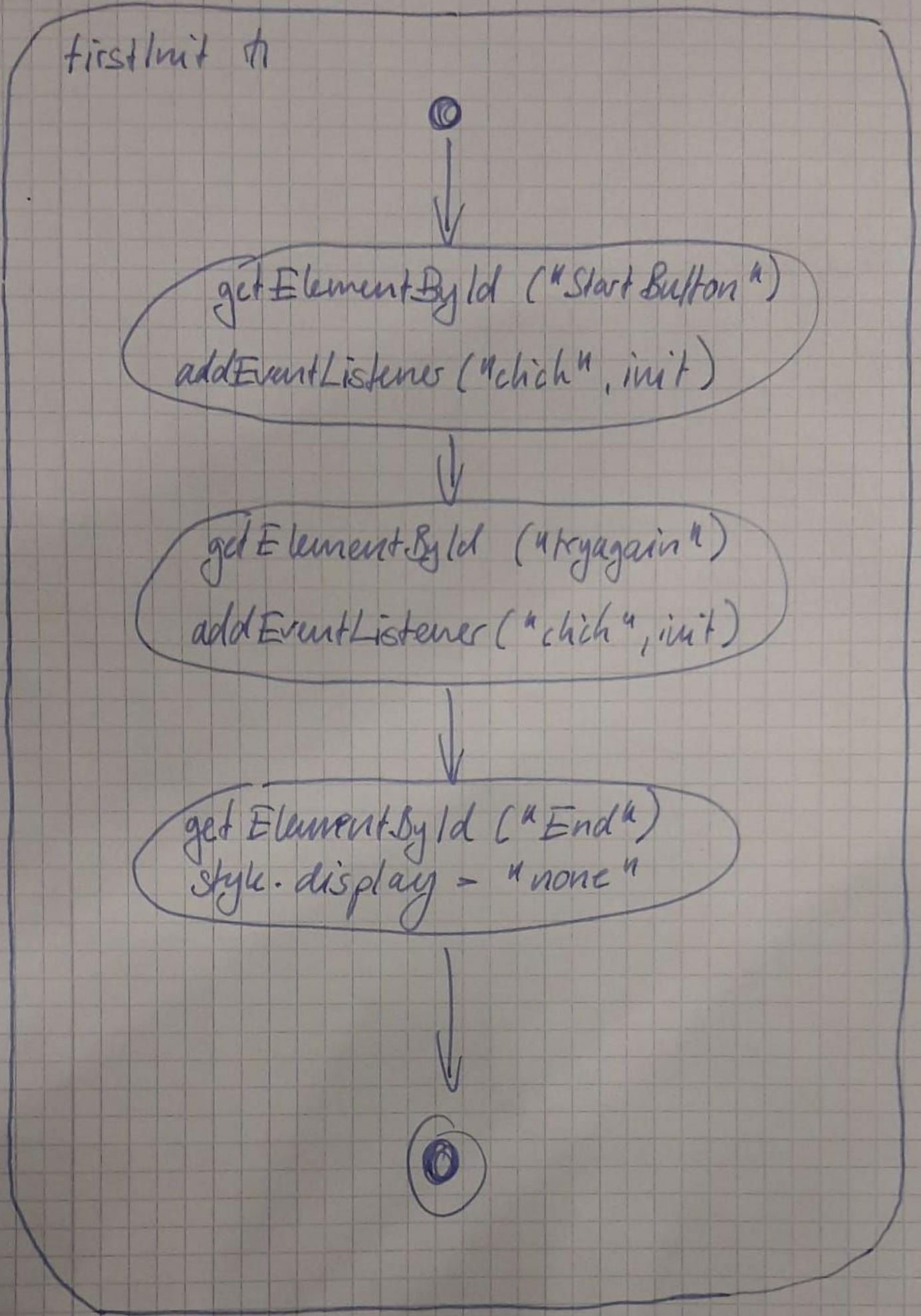
states:

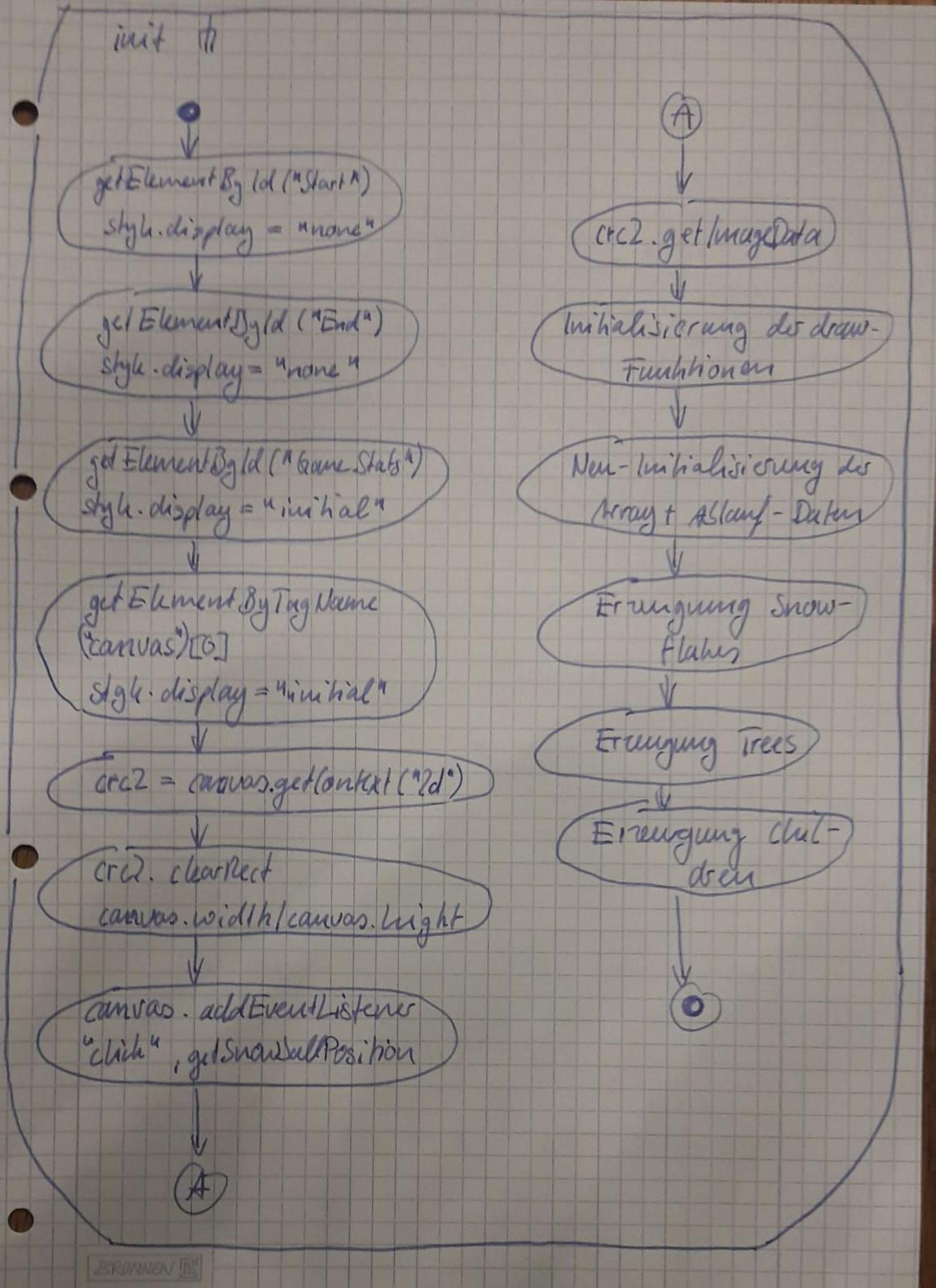
this.stak == "driveDown"

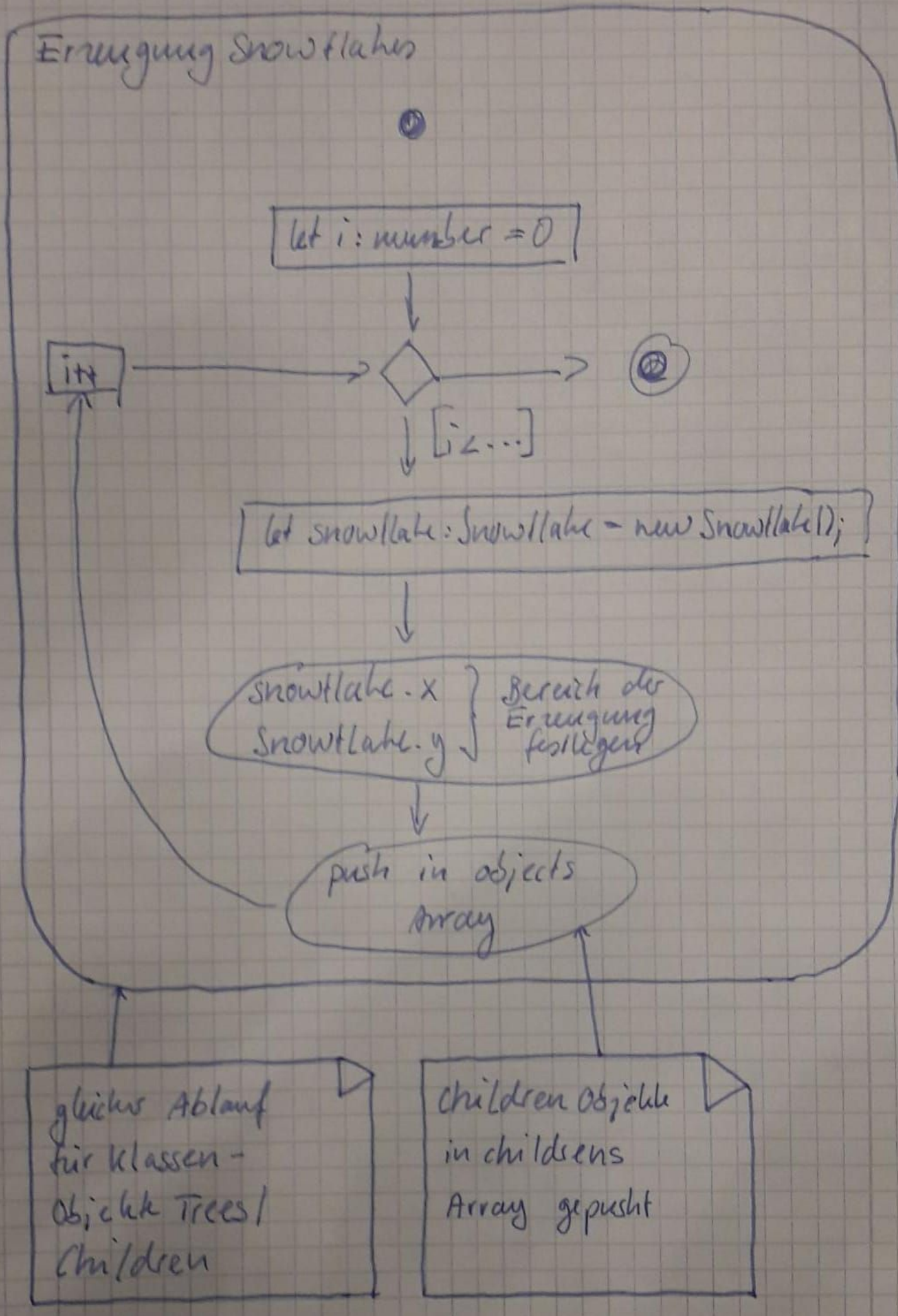
this.state == "moveUp"

this.stak == "hit

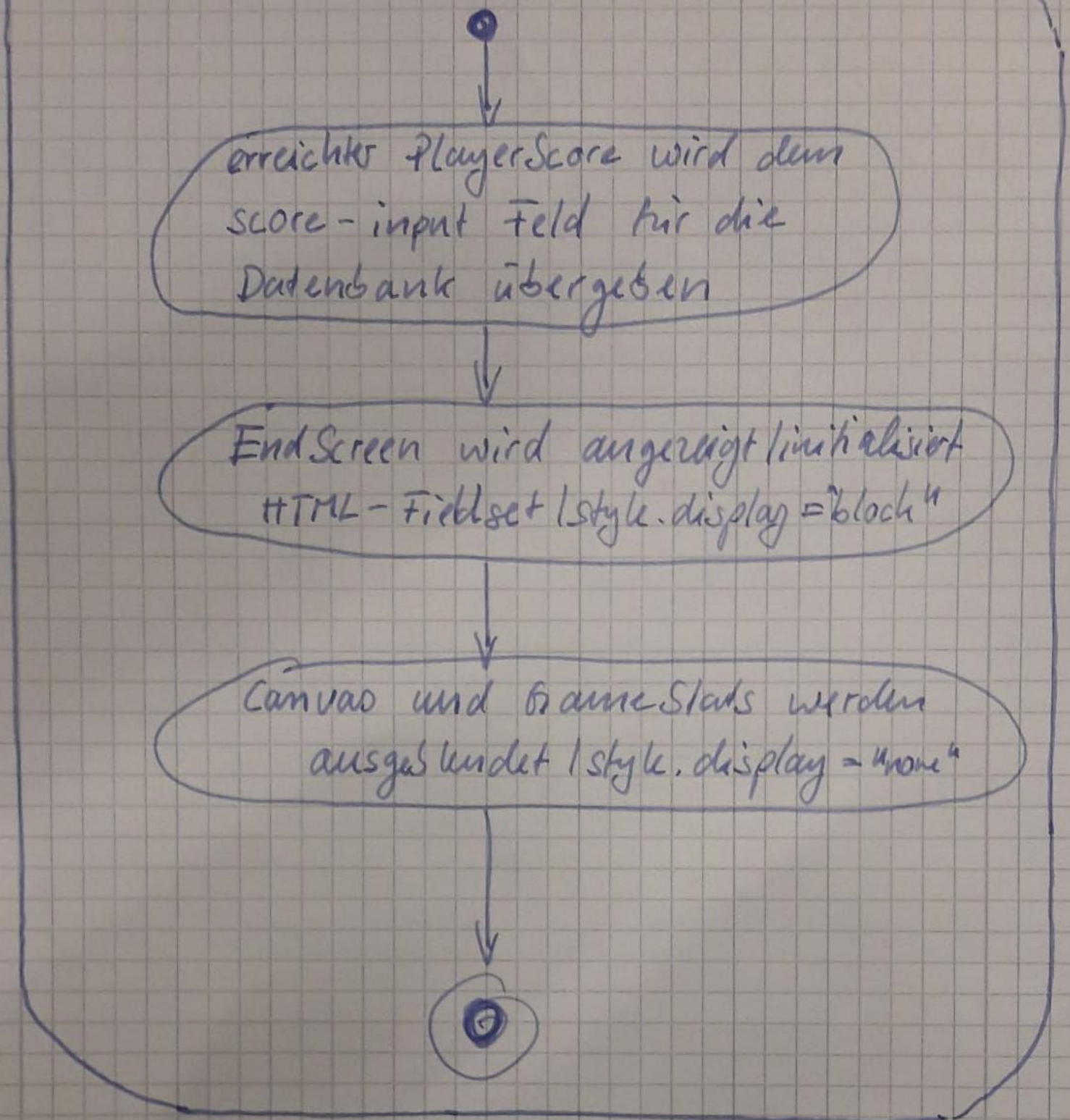


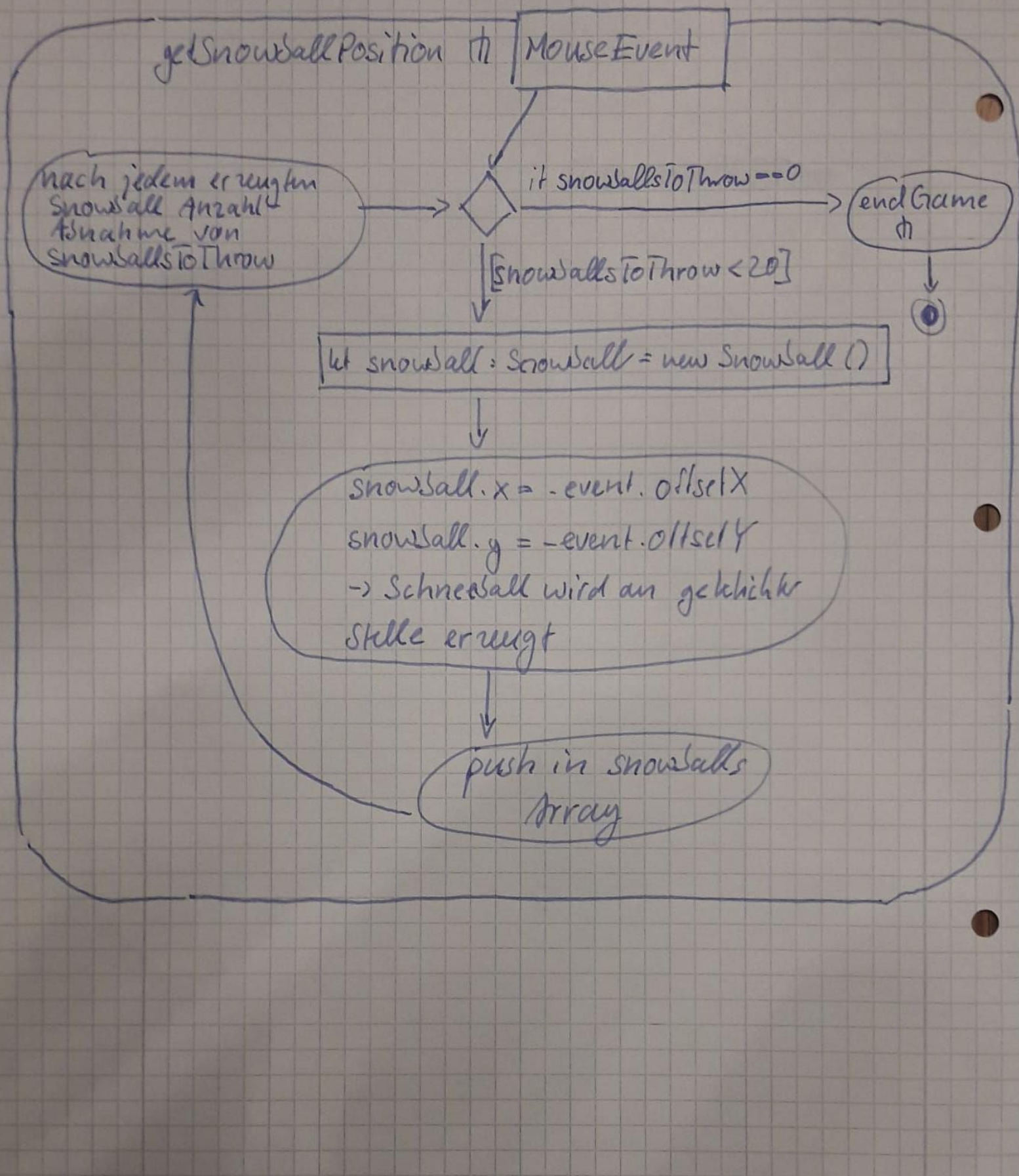

firstInit fn
getElementById ("StartButton")
addEventListener ("click", init)
getElementById ("tryagain")
addEventListener ("click", init)
getElementById ("End")
style.display = "none"

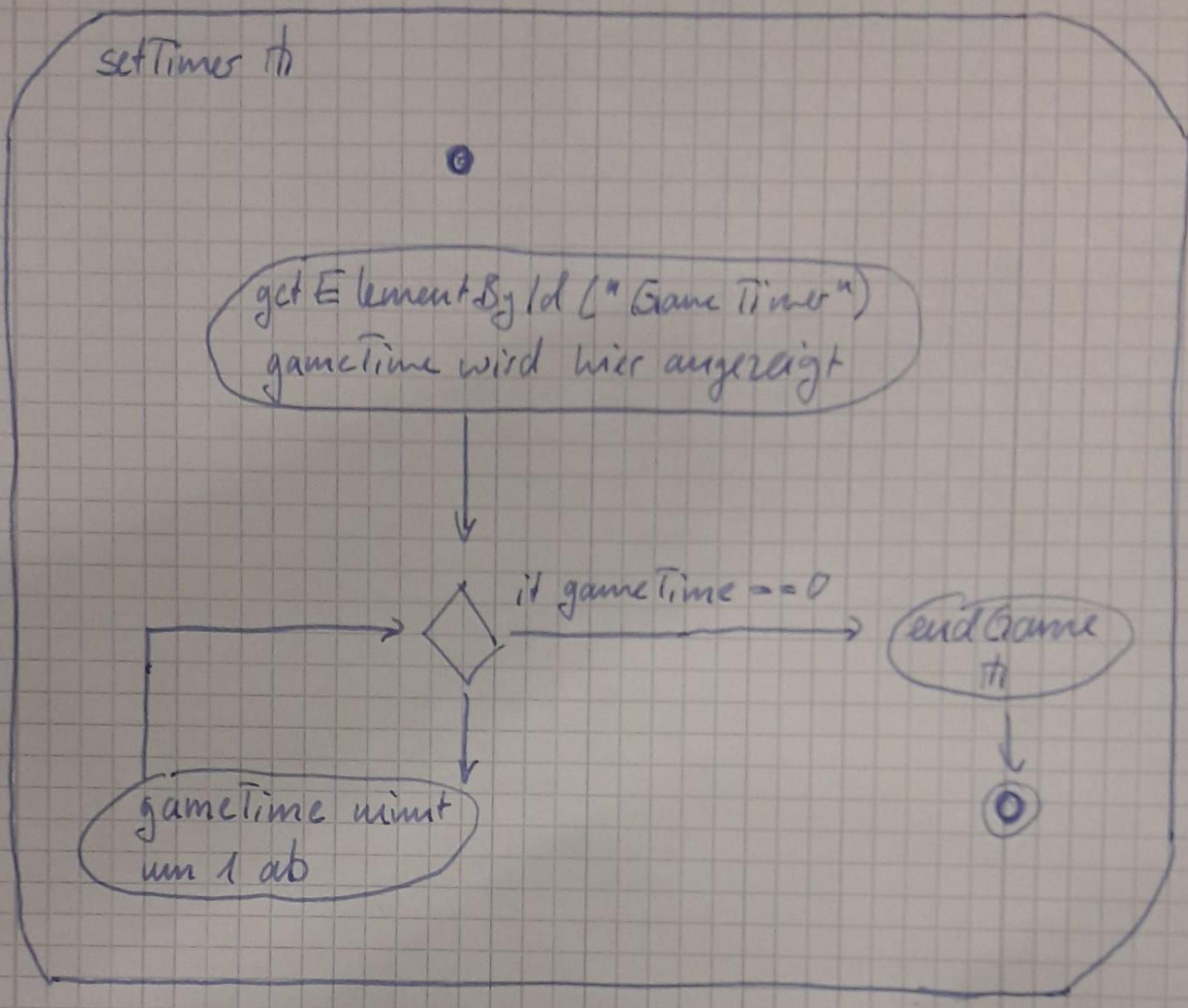

init
getElementById("Start")
style.display = "none"
getElementById("End")
style.display = "none"
getElementById("GameStats")
style.display = "initial"
getElementByTagName("canvas")[0]
style.display = "initial"
crc2 = canvas.getContext("2d")
crc2.clearRect
canvas.width/canvas.height
canvas.addEventListener
"click", getSnowballPosition
A
A
crc2.getImageData
Initialisierung der draw-Funktionen
Neu-Initialisierung des Array + Ablauf-Daten
Erzeugung Snow-Flakes
Erzeugung Trees
Erzeugung Chal-dren

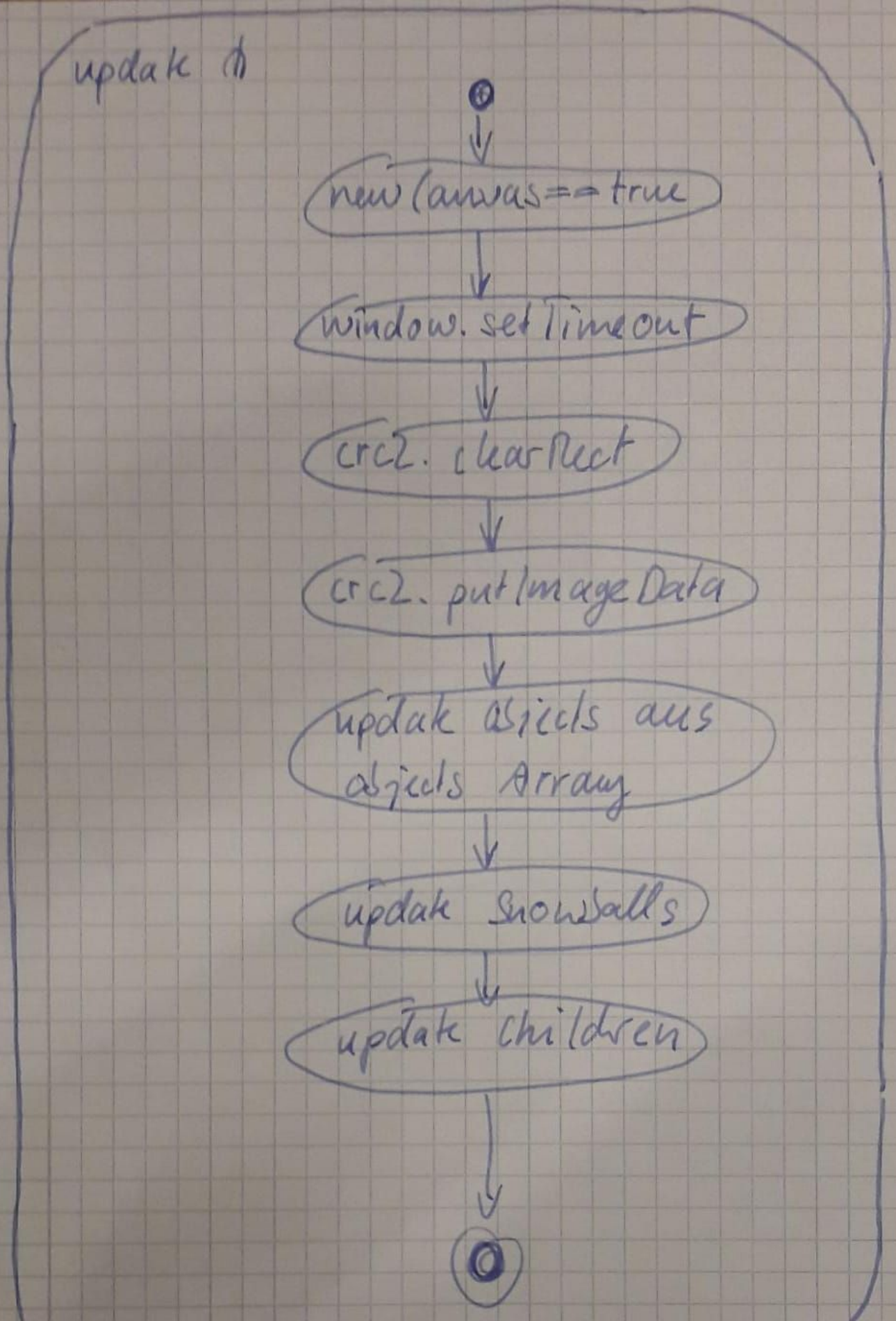

Erzeugung Snowflakes
let i: number = 0
i++
[i < ...]
let snowflake: Snowflake = new Snowflake();
snowflake.x
snowflake.y
Bereich der Erzeugung festlegen
push in objects Array
gleicher Ablauf für Klassen-Objekte Trees/ Children
children Objekte in childrens Array gepusht

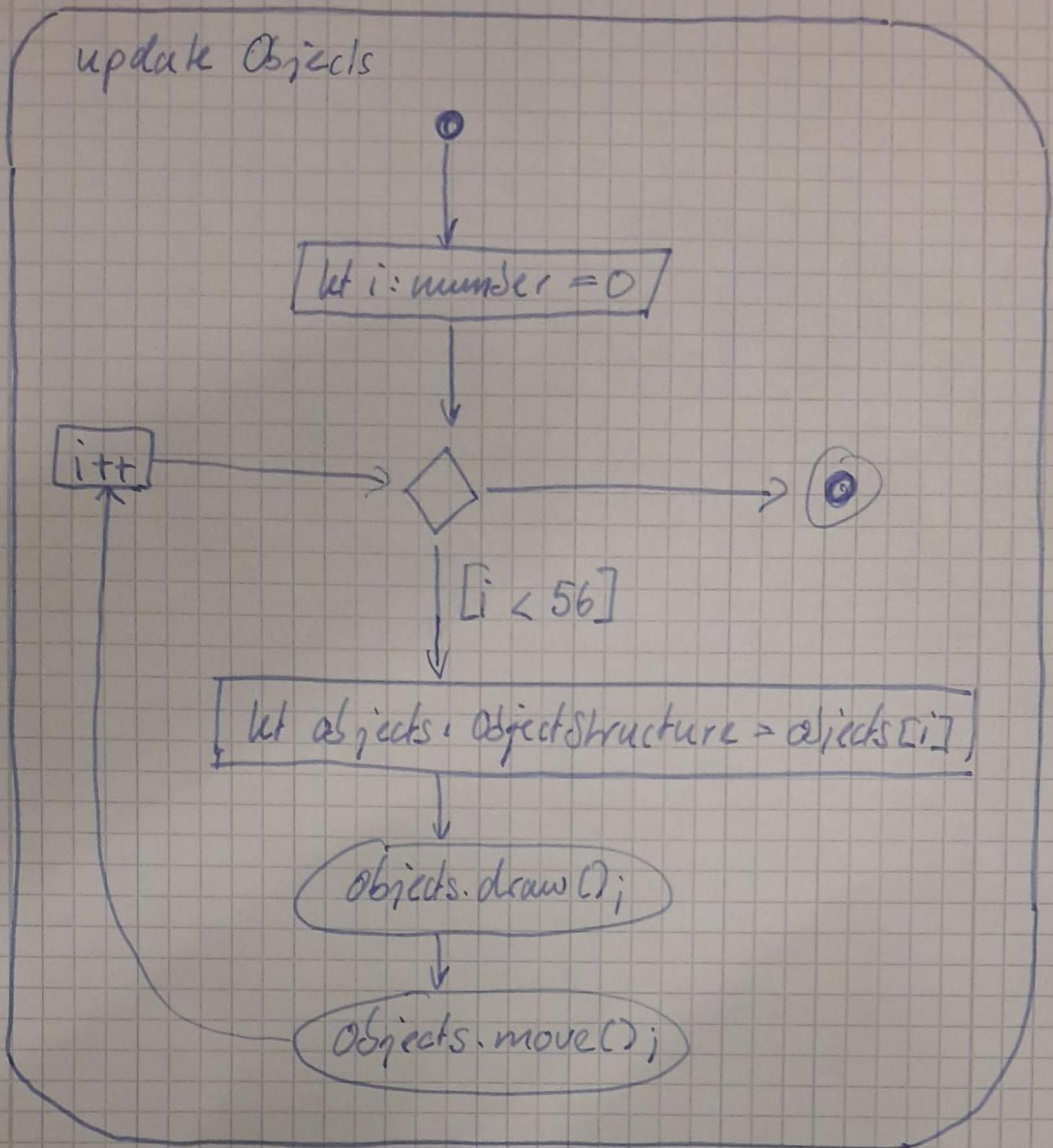
endGame ()

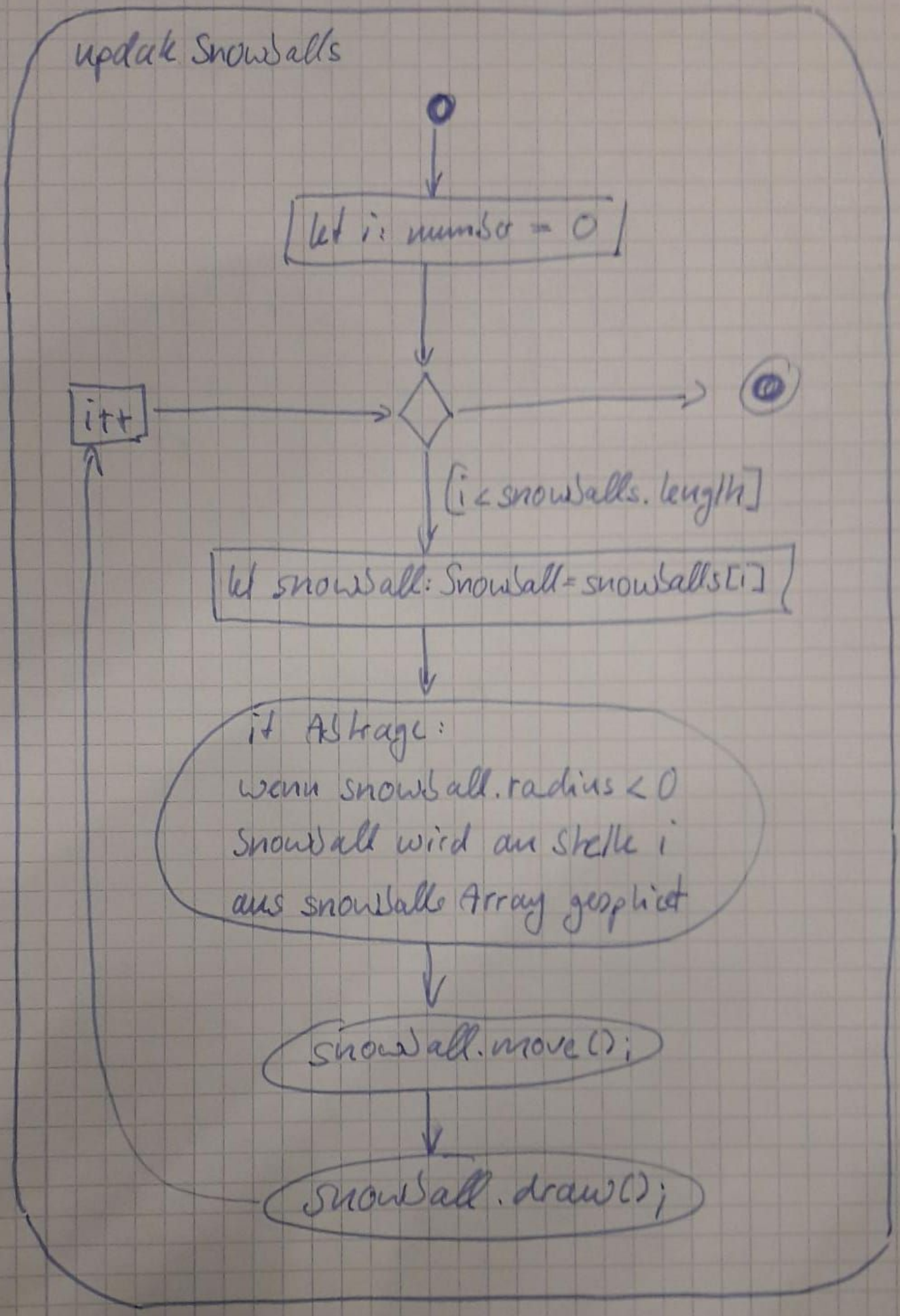

erreichter PlayerScore wird dem score-input Feld für die Datenbank übergeben
EndScreen wird angezeigt (initialisiert HTML-Fieldset (style.display = "block"
Canvas und GameStats werden ausgeblendet (style.display = "none"

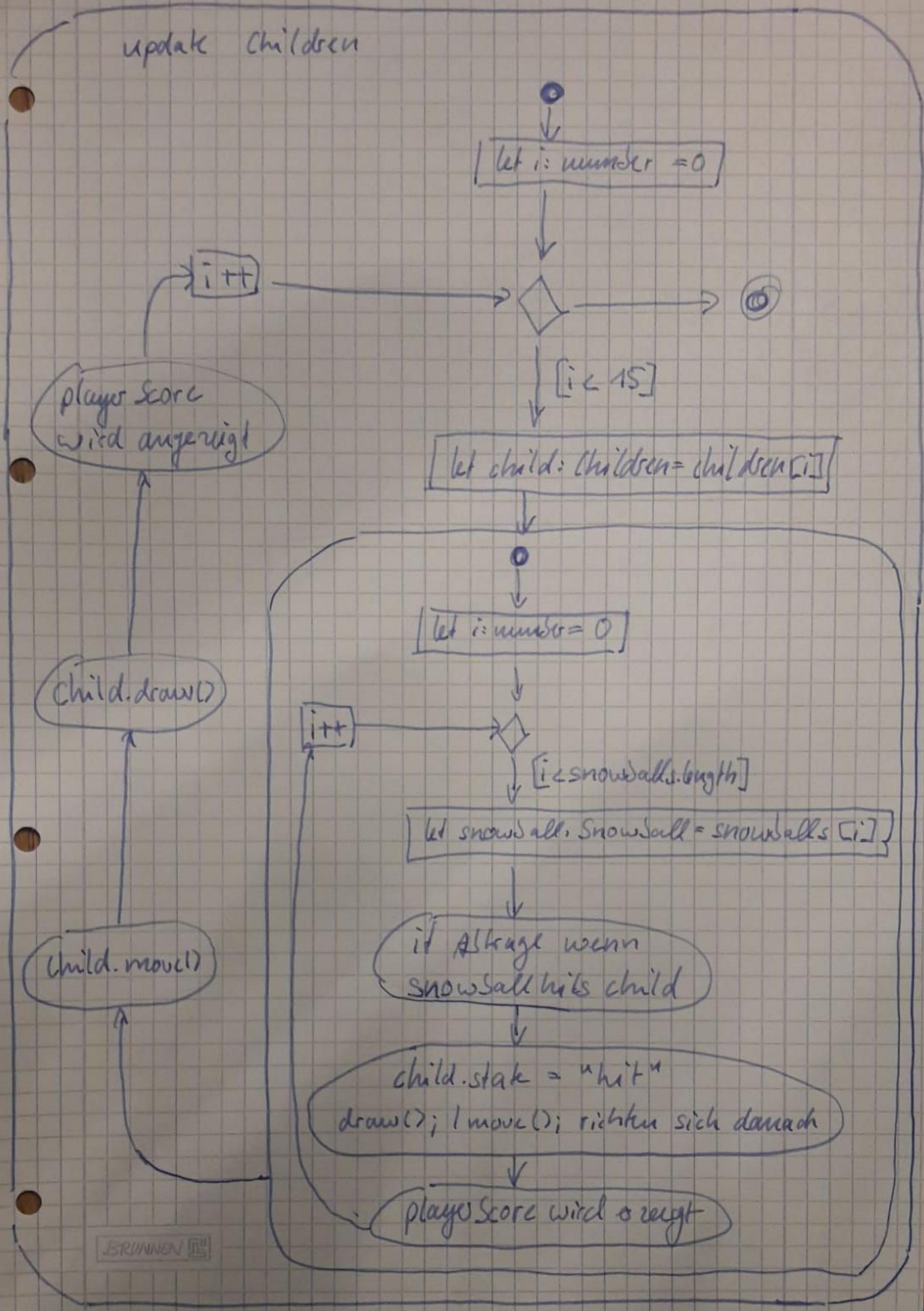

getSnowballPosition
MouseEvent
nach jedem erzeugten Snowball Anzahl Abnahme von snowballsToThrow
if snowballsToThrow == 0
endGame
[snowballsToThrow < 20]
let snowball: Snowball = new Snowball()
snowball.x = -event.offsetX
snowball.y = -event.offsetY
-> Schneeball wird an geklickter Stelle erzeugt
push in snowballs Array

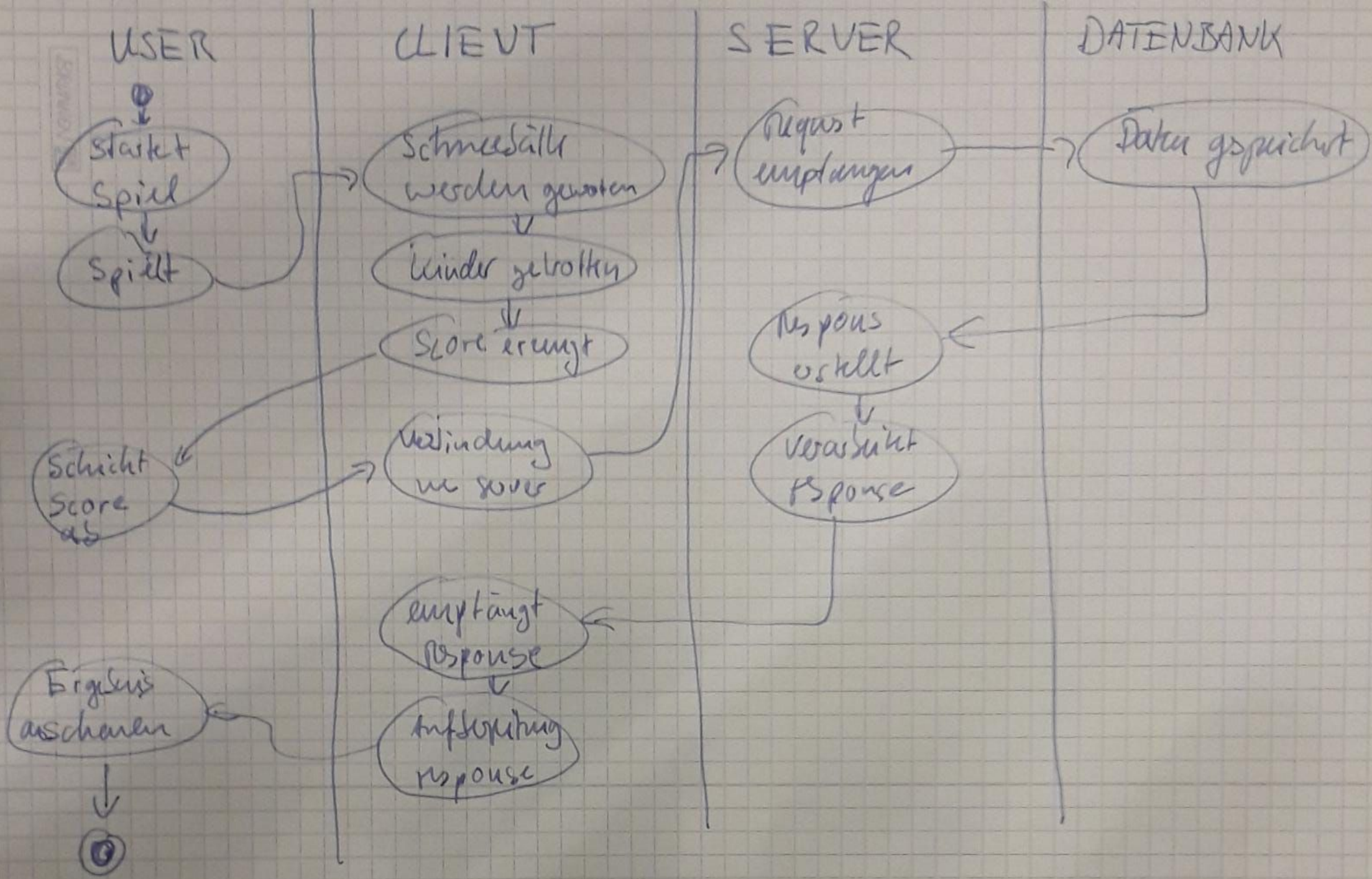

setTimer fn
getElementById("GameTimer")
gameTime wird hier angezeigt
if gameTime == 0
endGame fn
gameTime minut um 1 ab


update ()
newCanvas == true
window.setTimeout
crc2.clearRect
crc2.putImageData
update objects aus objects Array
update Snowballs
update children


update Objects
let i: number = 0
i++
[i < 56]
let objects: ObjectStructure = objects[i]
objects.draw();
objects.move();


updak Snowballs
let i: number = 0
i++
[i < snowballs.length]
let snowball: Snowball = snowballs[i]
if Abfrage:
wenn snowball.radius < 0
Snowball wird an Stelle i
aus snowballs Array gesplict
snowball.move();
snowball.draw();


update Children
let i: number = 0
i++
[i < 15]
player Score wird angezeigt
let child: Children = children[i]
let i: number = 0
child.draw()
i++
[i < snowballs.length]
let snowball: Snowball = snowballs[i]
child.move()
if Abfrage wenn snowball hits child
child.state = "hit"
draw(); / move(); richten sich danach
player Score wird erhöht


USER
CLIENT
SERVER
DATENBANK
startet Spiel
spielt
Schneeball werden geworfen
Ziel wird getroffen
Score erzeugt
Schicht Score ab
Verbindung zu Server
Request empfangen
Daten gespeichert
Response erstellt
verarbeitet Response
empfängt Response
Auftauchung Response
Ergebnis anschauen