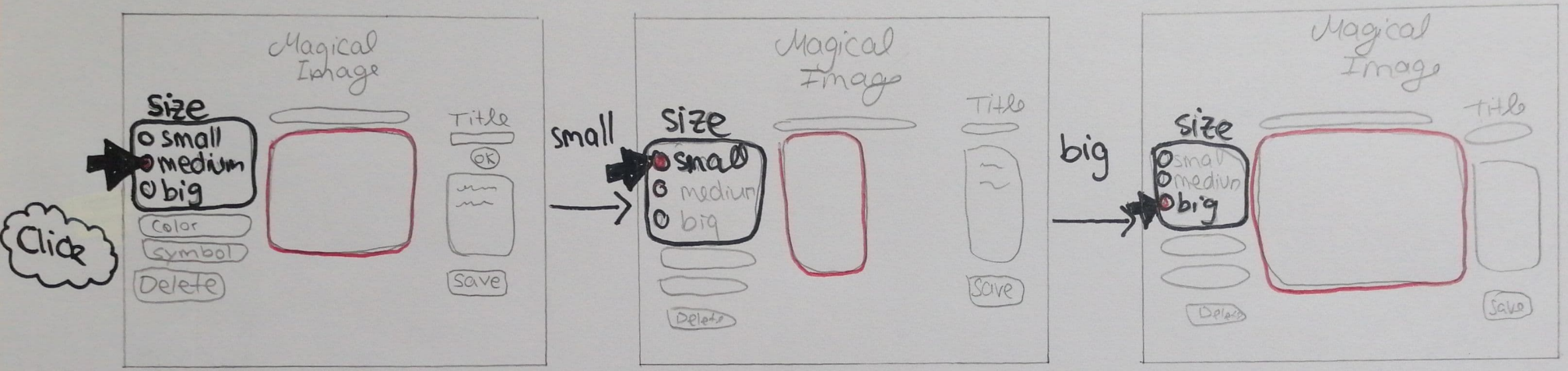
Magical Image - Skizze

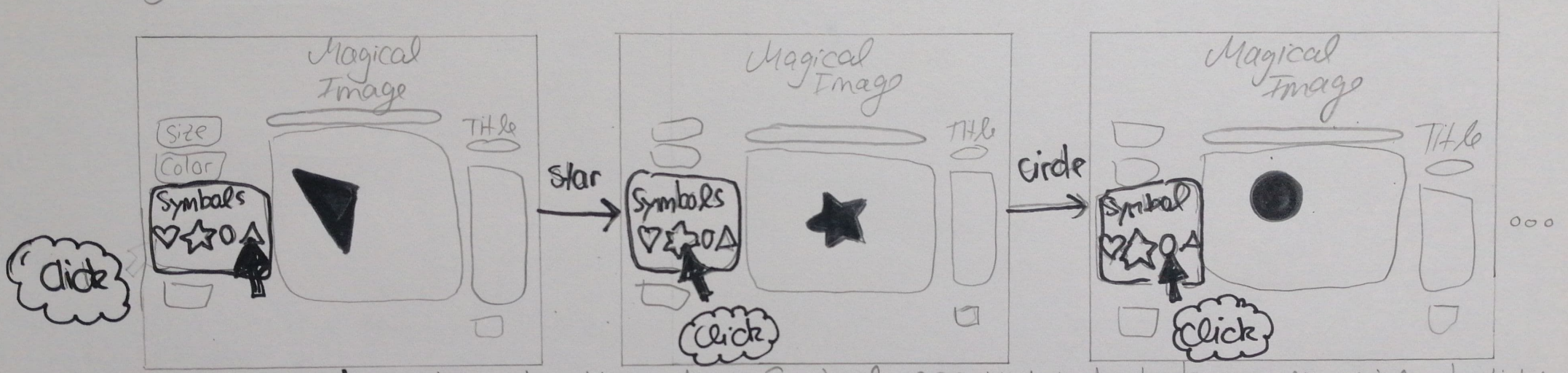
Der Nutzer kann die Größe der Zeichenfläche einstellen



Der Nutzer kann die Farbe der Zeichenfläche einstellen

ooo

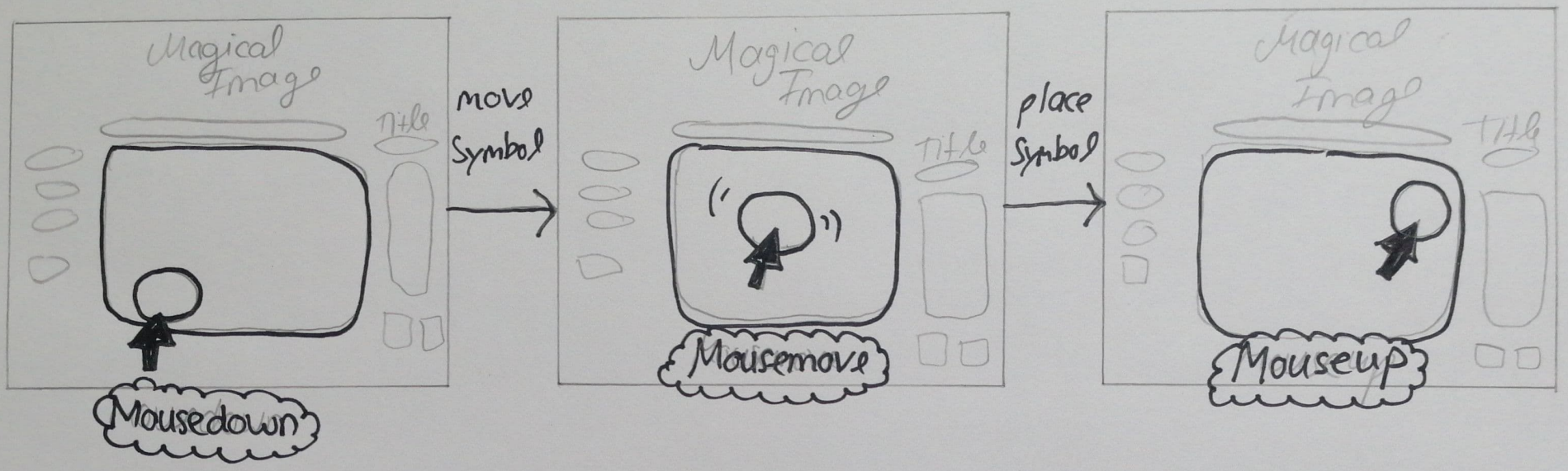
Der Nutzer kann zwischen 4 verschiedenen Symbolen wählen



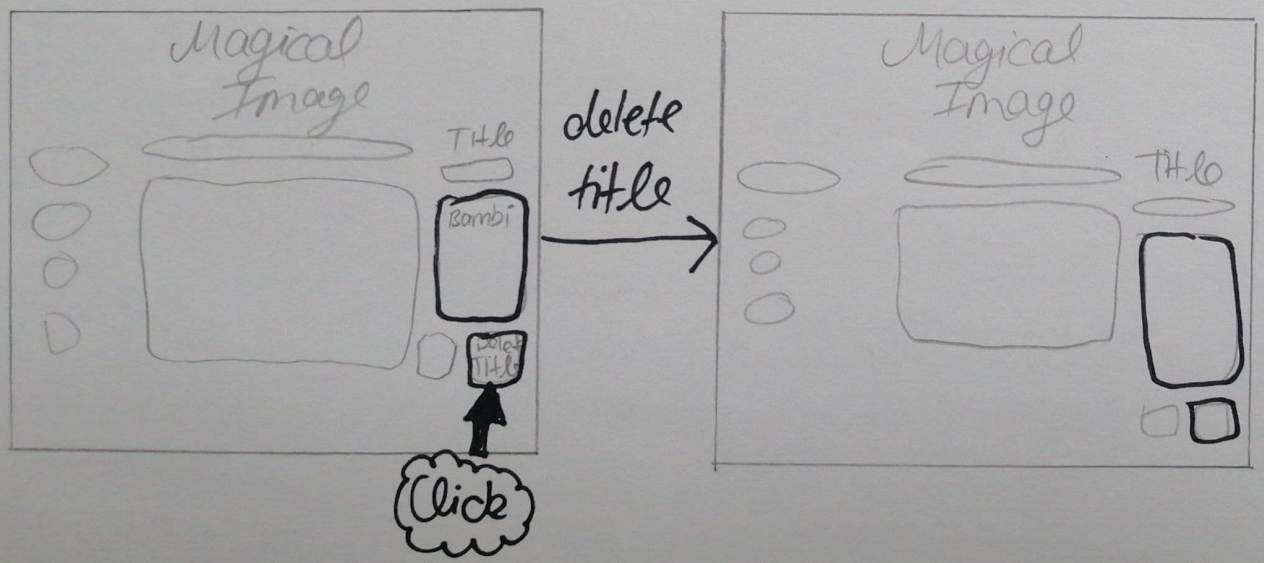
heart

Nachdem der User das Symbol angeklickt hat, kann er eine beliebige Farbe eingeben

Der Nutzer kann die Symbole per Drag & Drop individuell verschieben



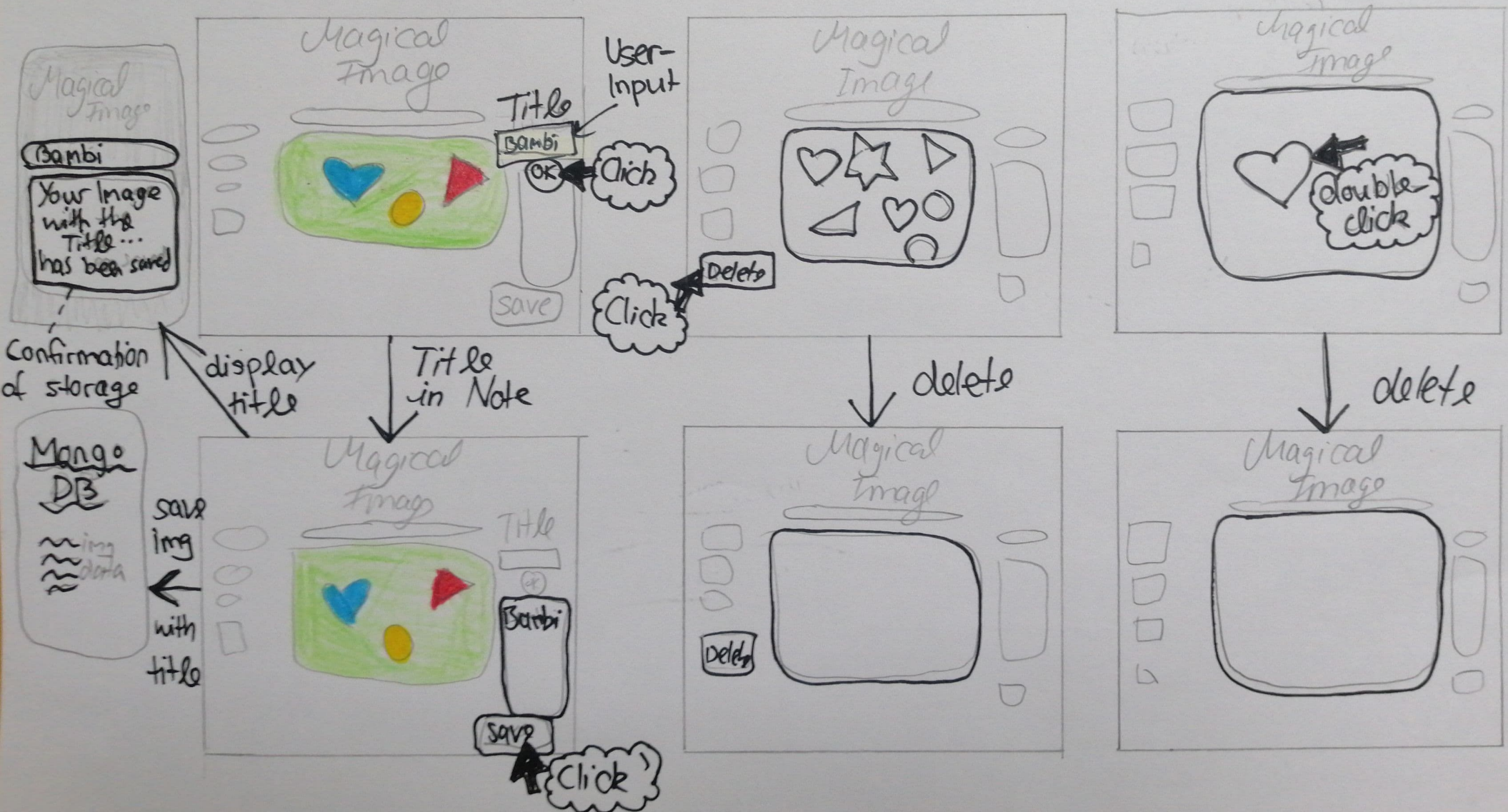
Der Nutzer kann den eingegeberen Titel wieder löschen

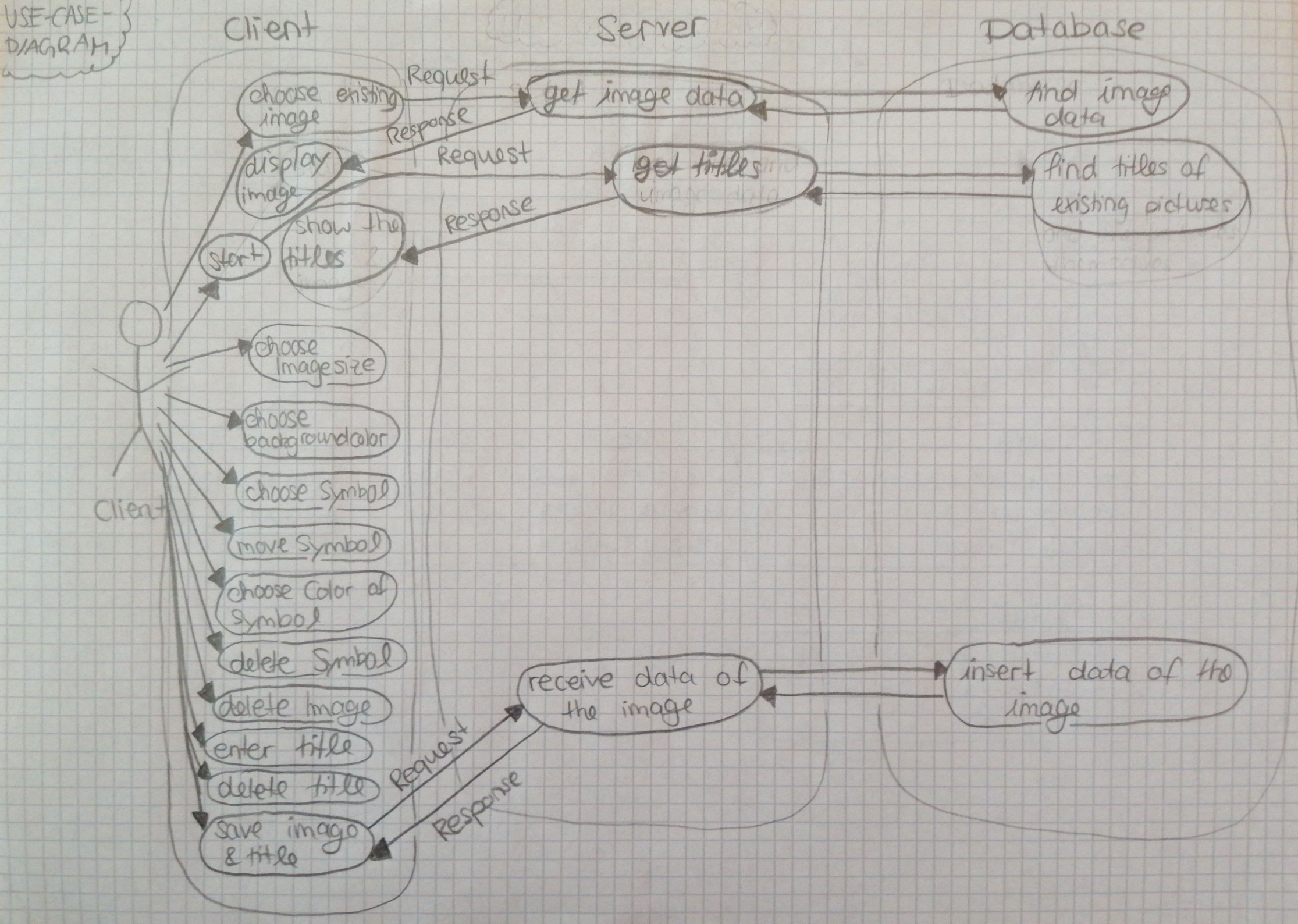


Der Nutzer kann im Input-Feld einen Titel für das fertige Bild eingeben und dadurch Bild + Titel abspeichern

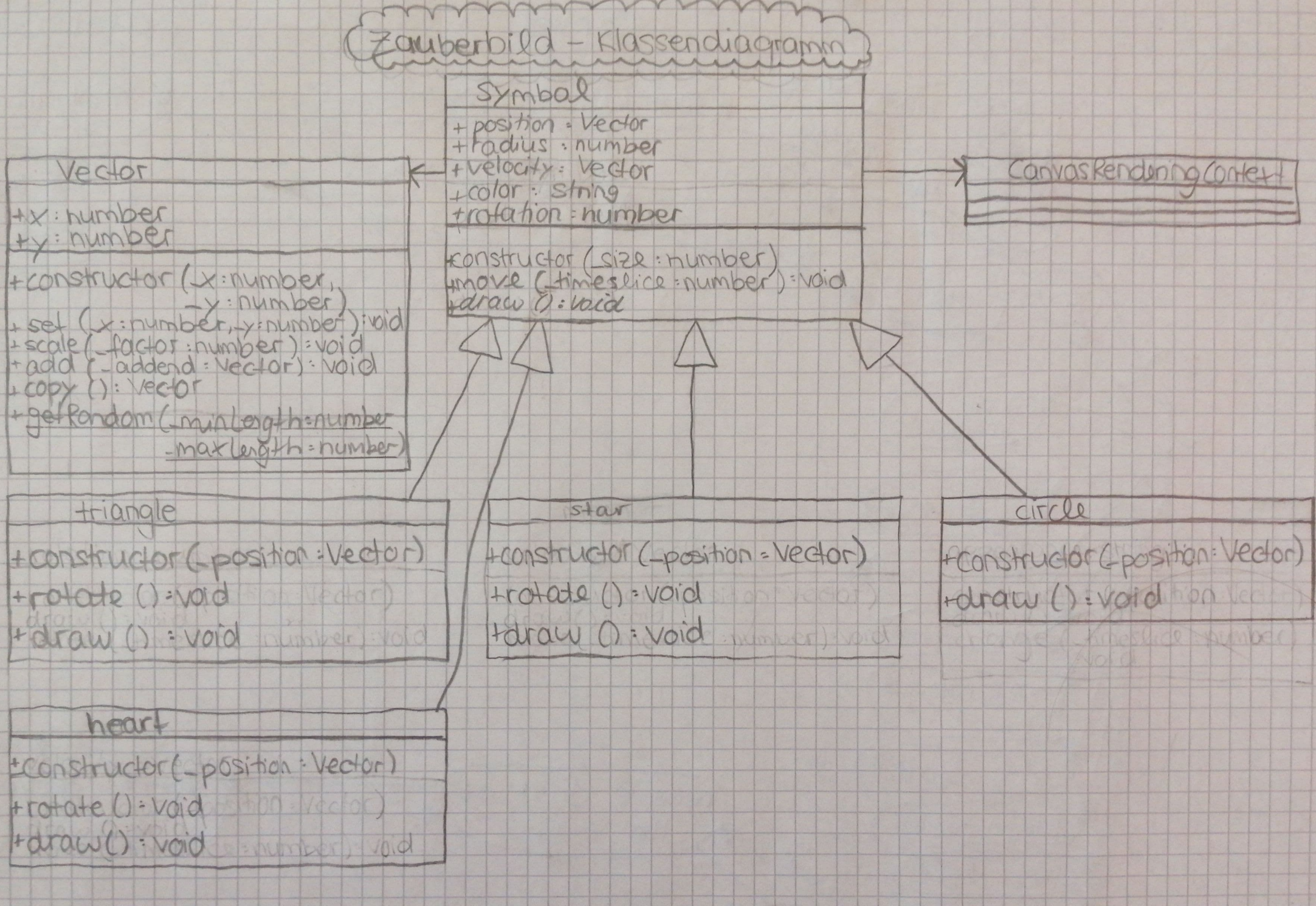
Durch Klicken des "Delete"-Buttons werden alle Symbole im Canvas gelöscht

Durch Doppelklick auf Symbol wird dieses gelöscht



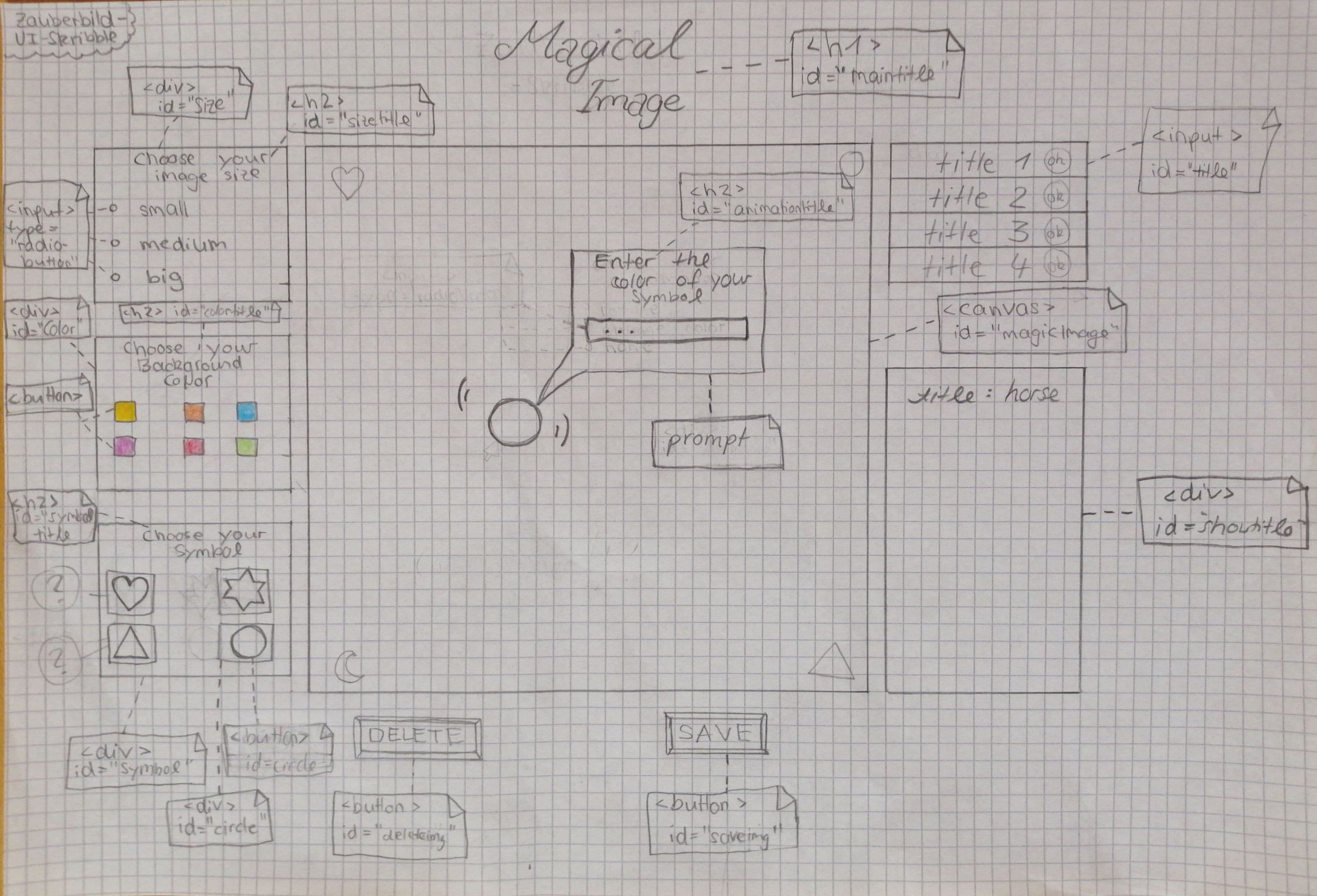

USE-CASE-DIAGRAM
Client
Server
Database
choose existing image
Request
Response
get image data
find image data
display image
Request
get titles
find titles of existing pictures
Response
start
show the titles
choose imagesize
choose backgroundcolor
choose Symbol
move Symbol
choose Color of Symbol
delete Symbol
delete Image
enter title
delete title
save image & title
Request
Response
receive data of the image
insert data of the image
Client

Zauberbild - Klassendiagramm

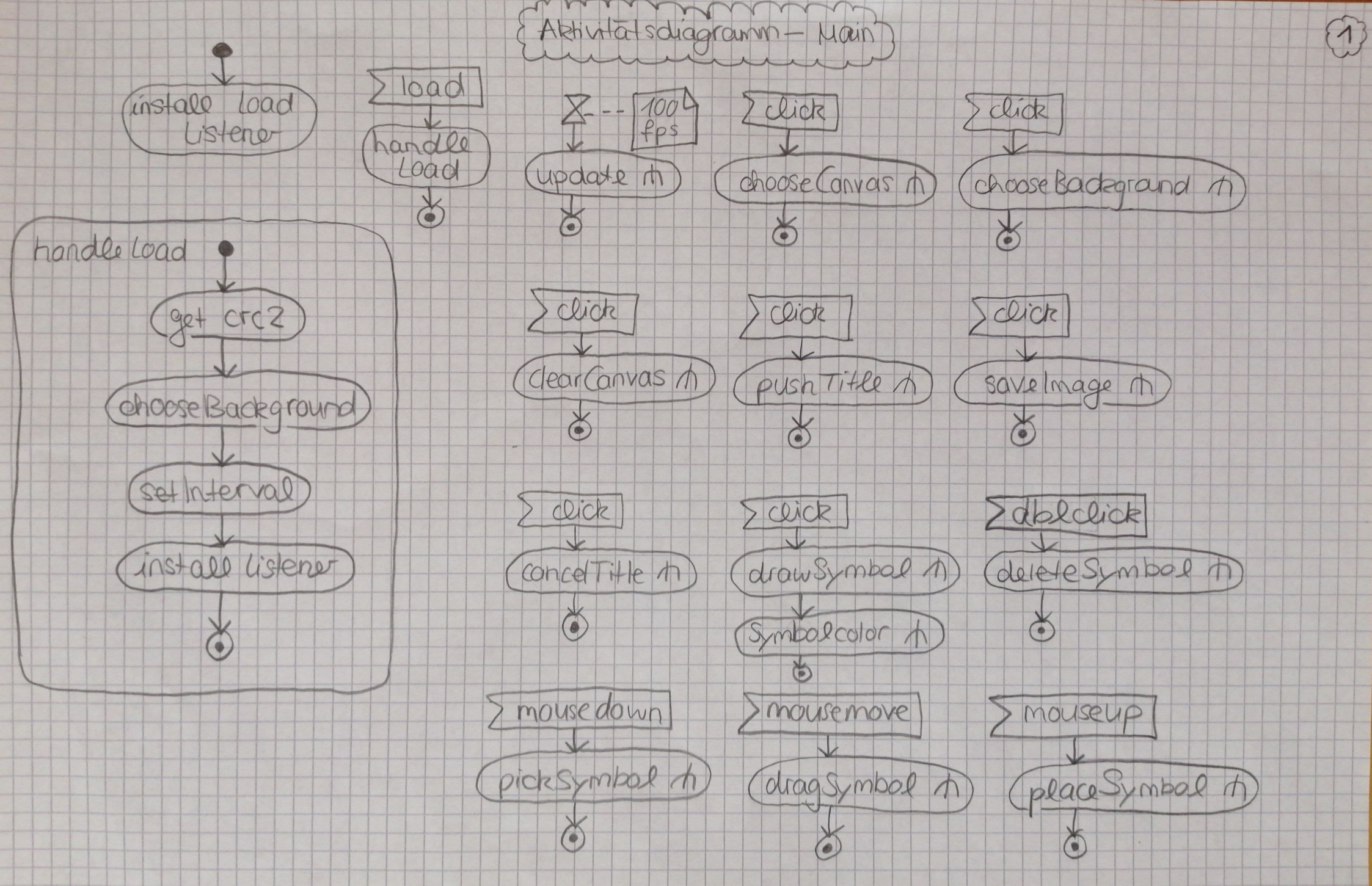


Zauberbild - UI-Skribble

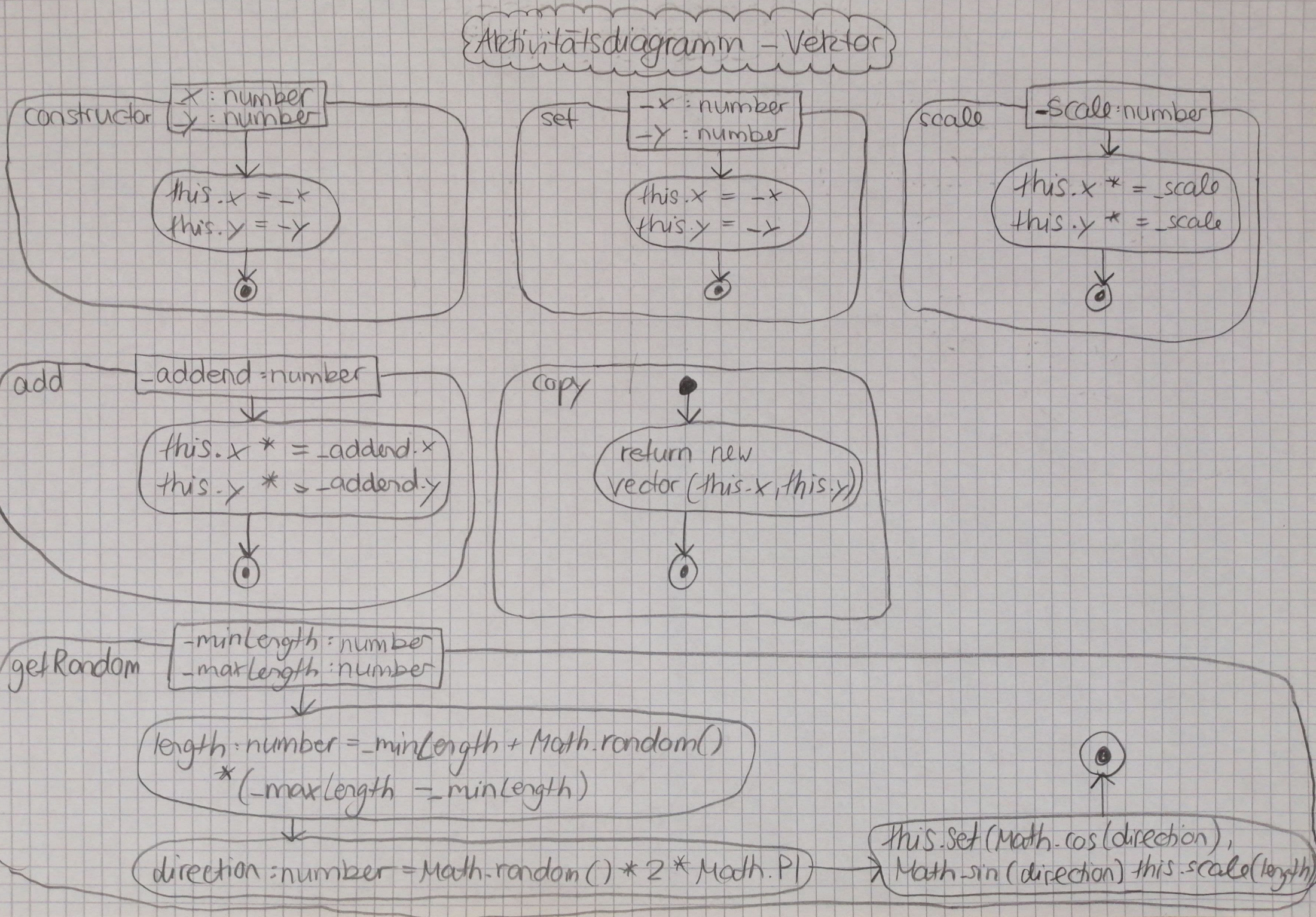
Magical Image



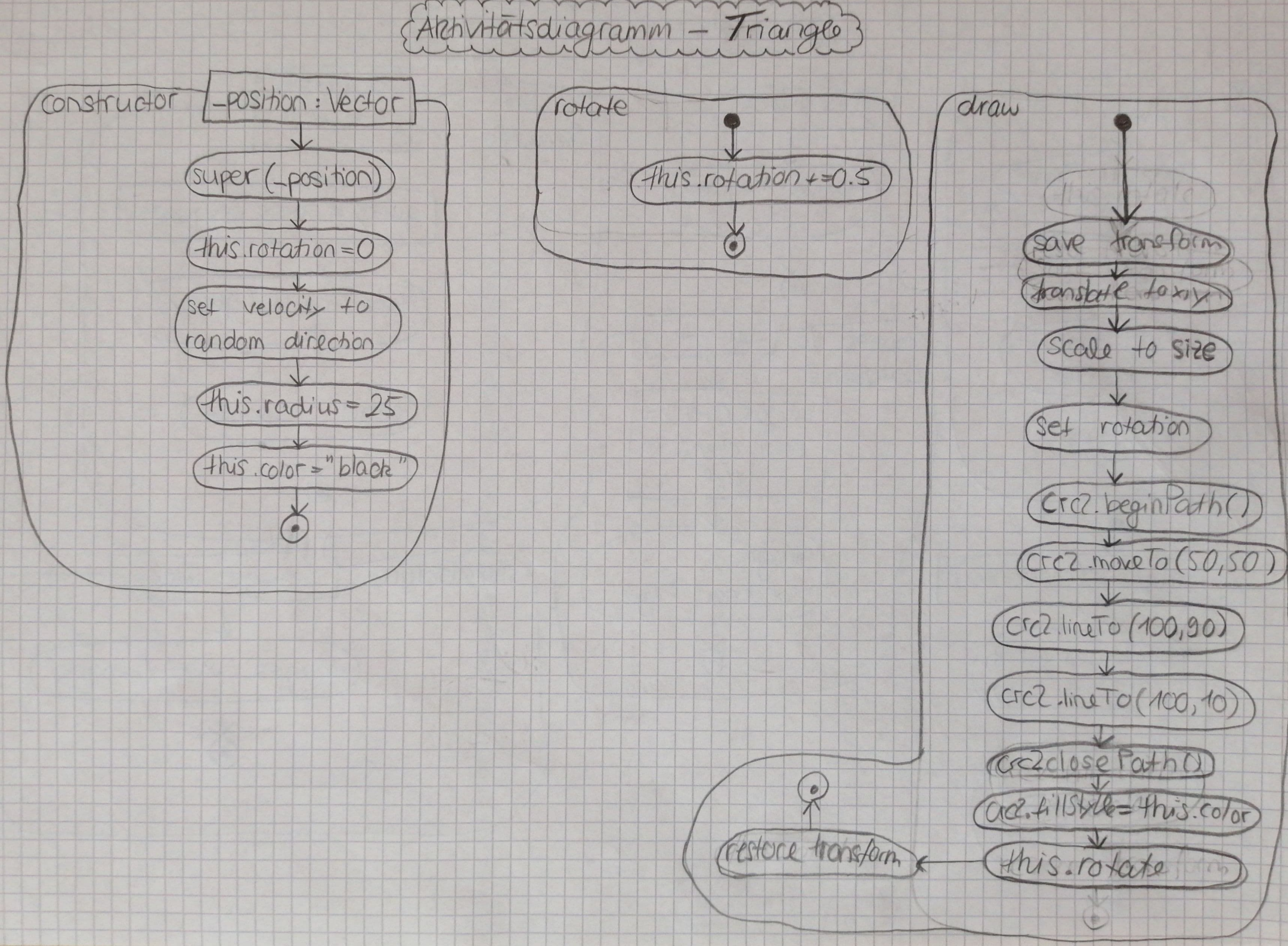
Aktivitätsdiagramm – Main



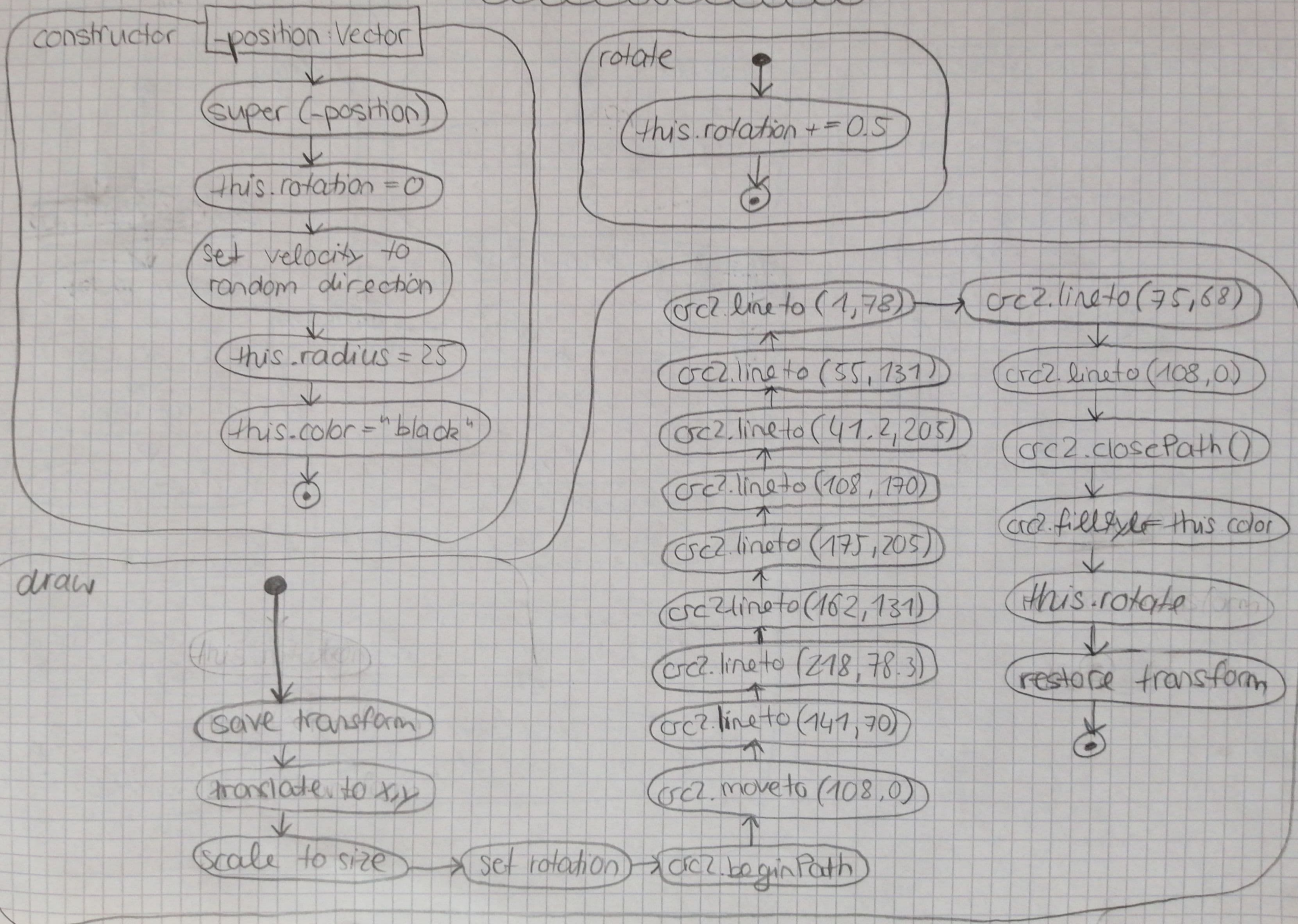
Aktivitätsdiagramm – Vektor



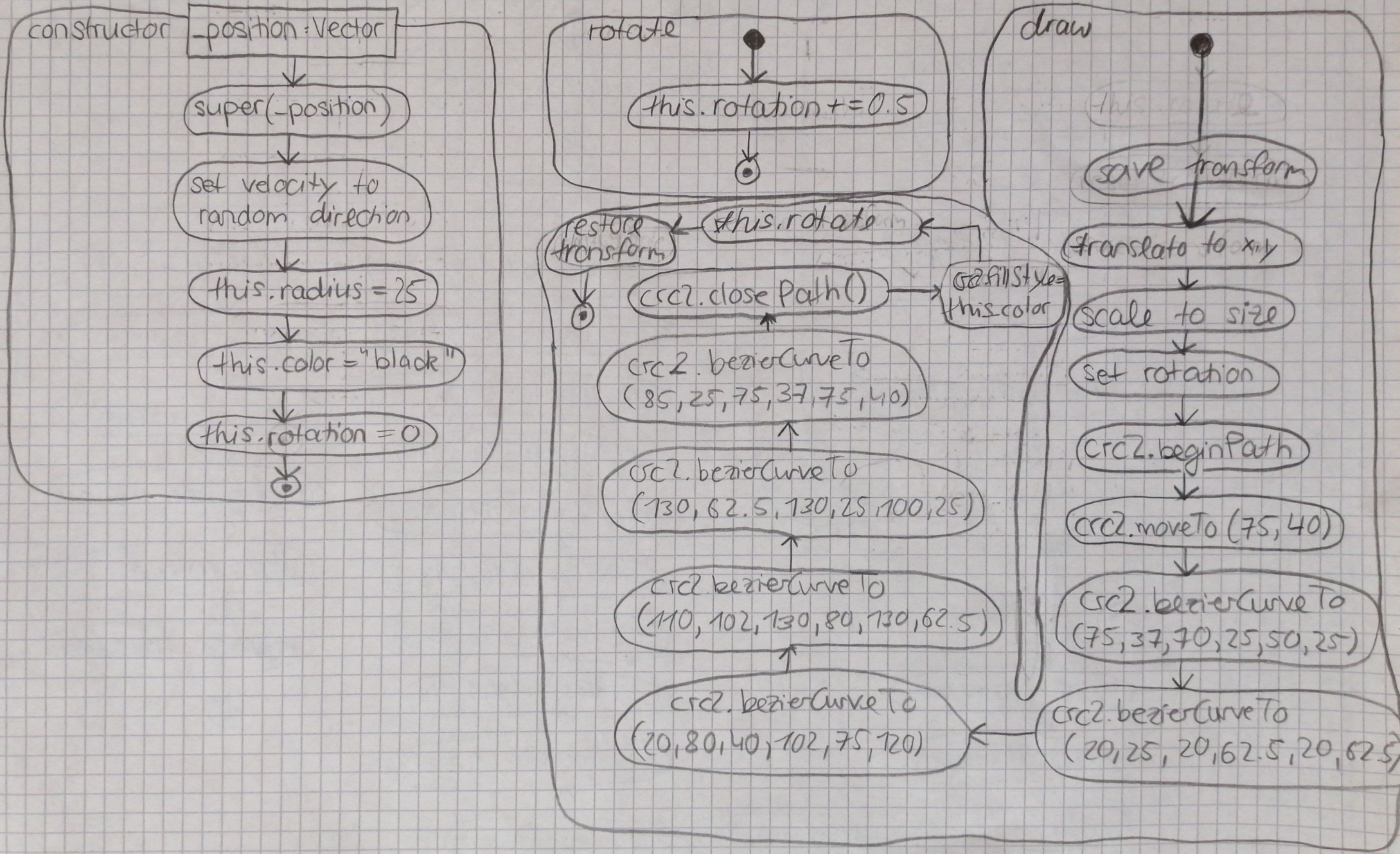
Aktivitätsdiagramm – Triangle



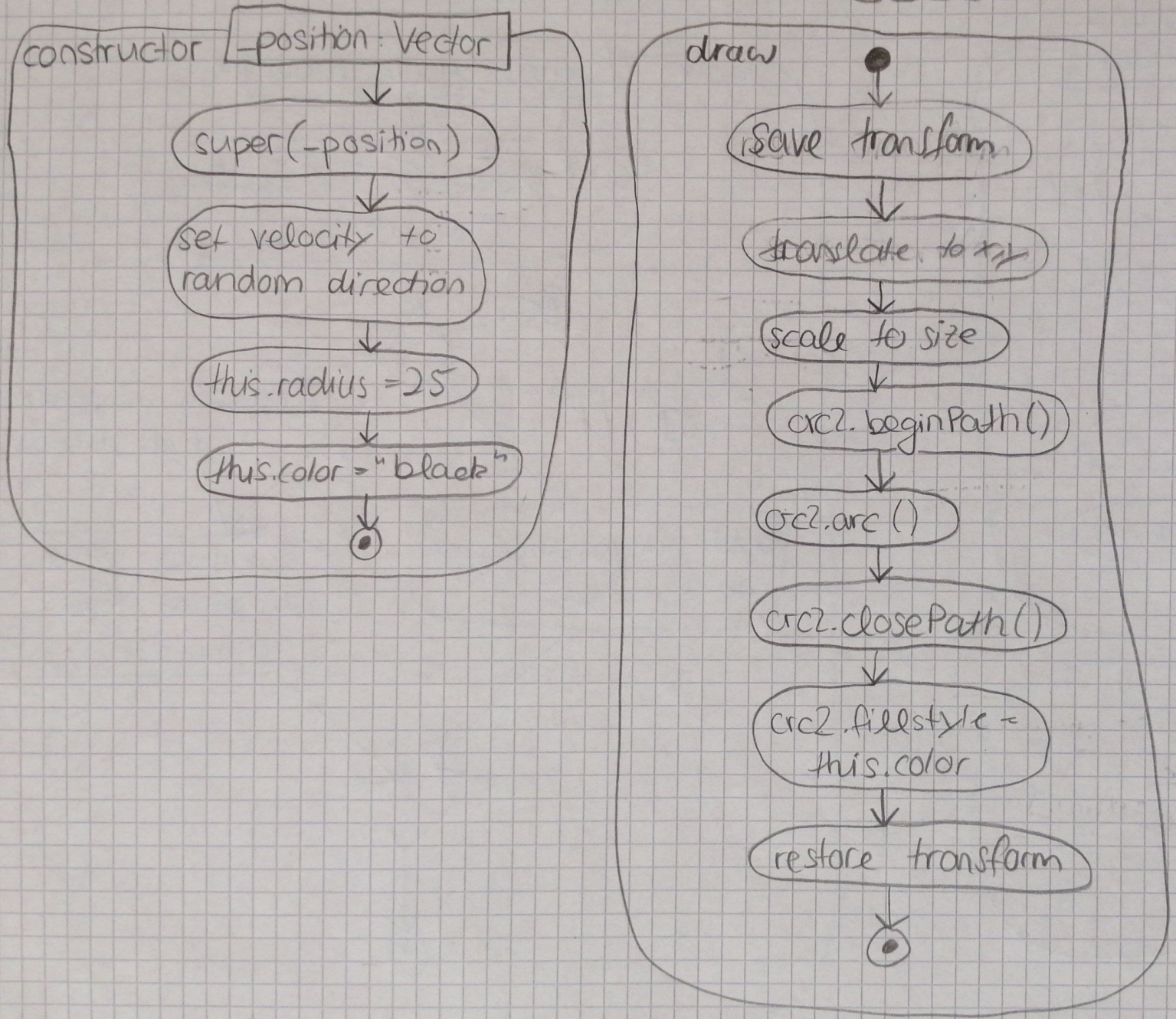
Aktivitätsdiagramm – Star



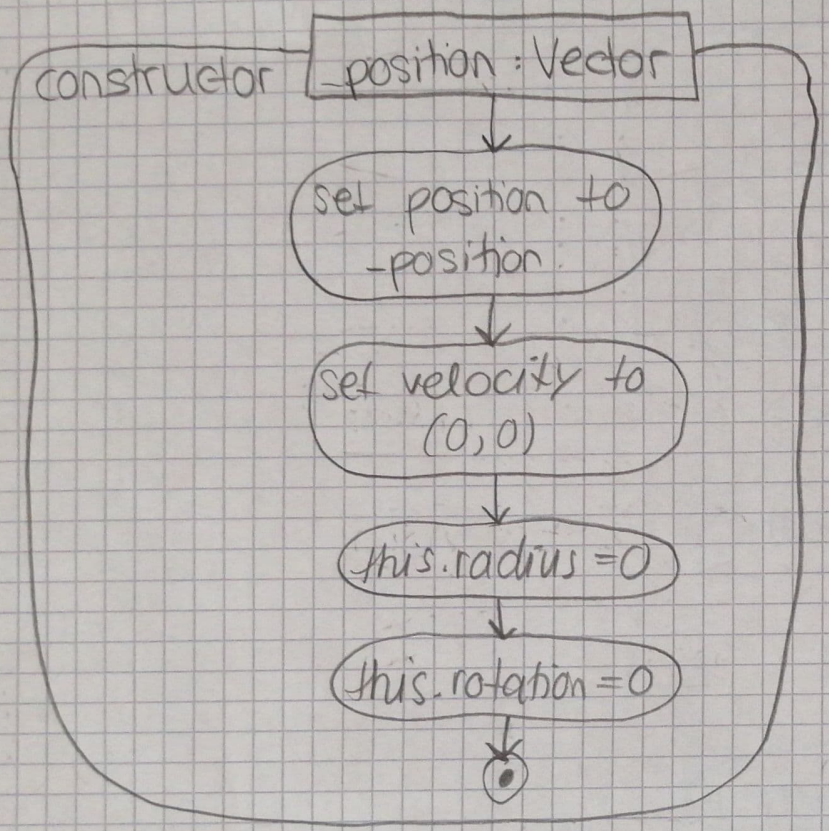
Aktivitätsdiagramm - Heart

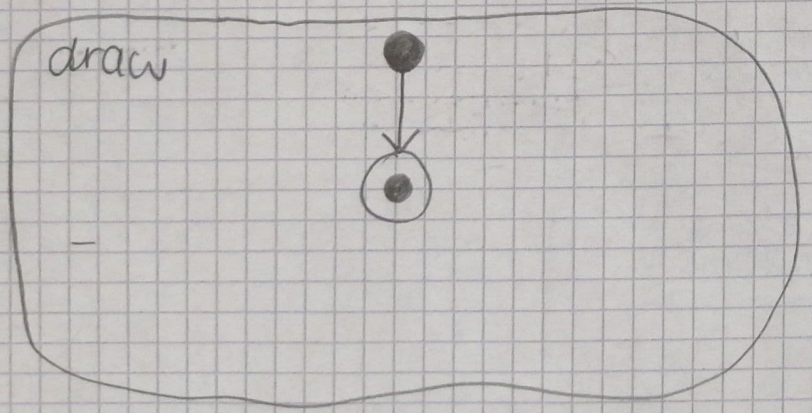


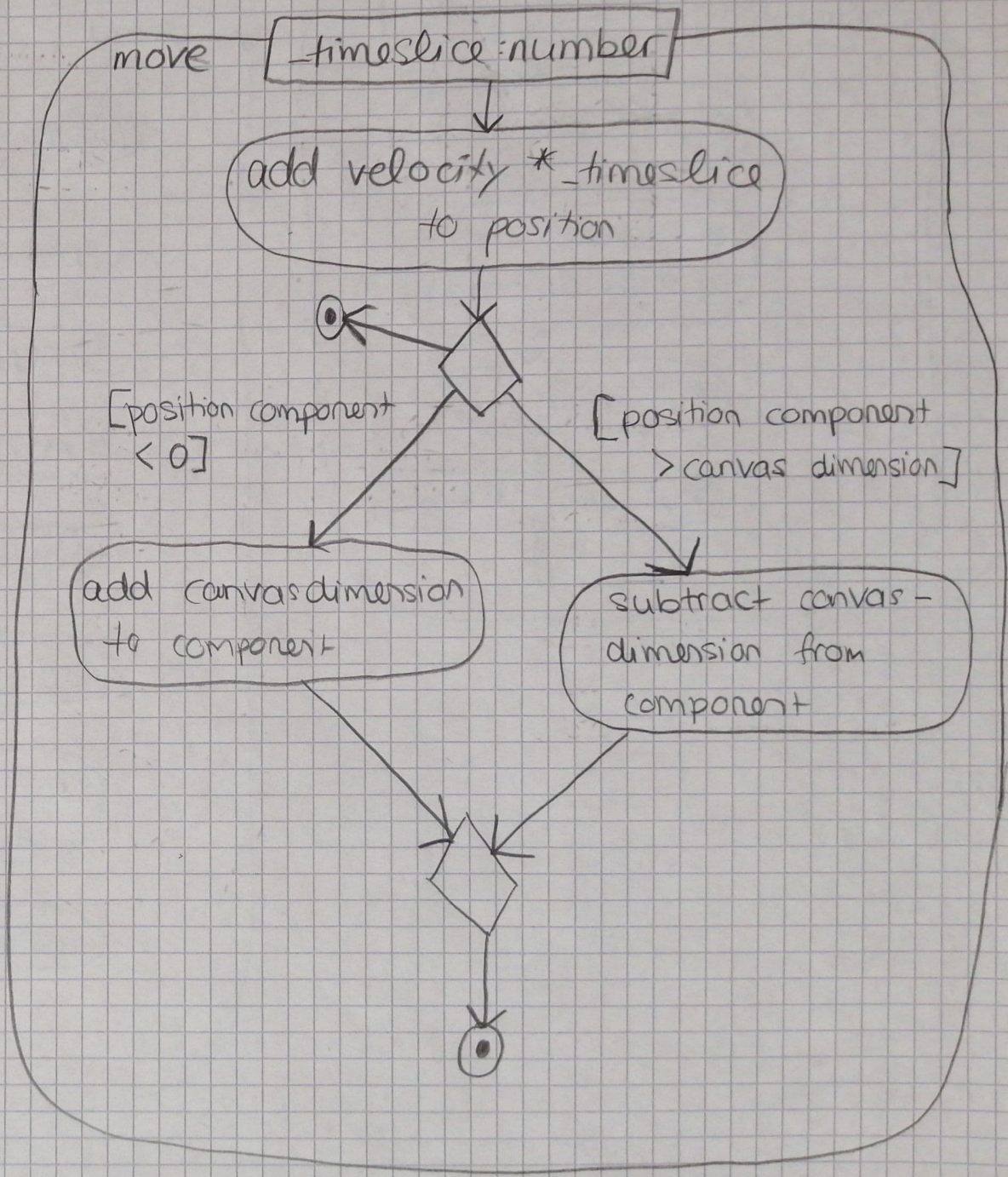
Aktivitätsdiagramm – Circle

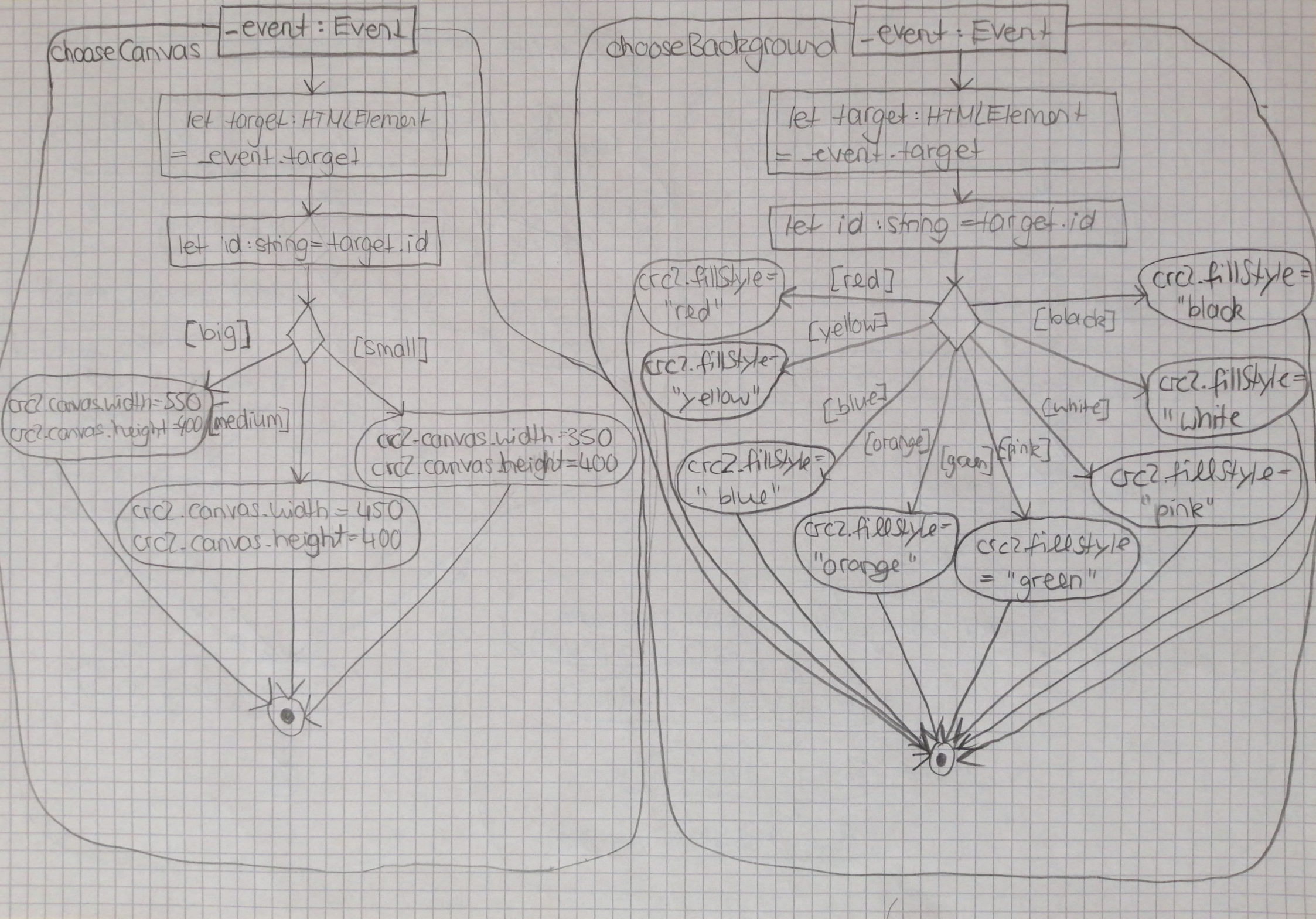


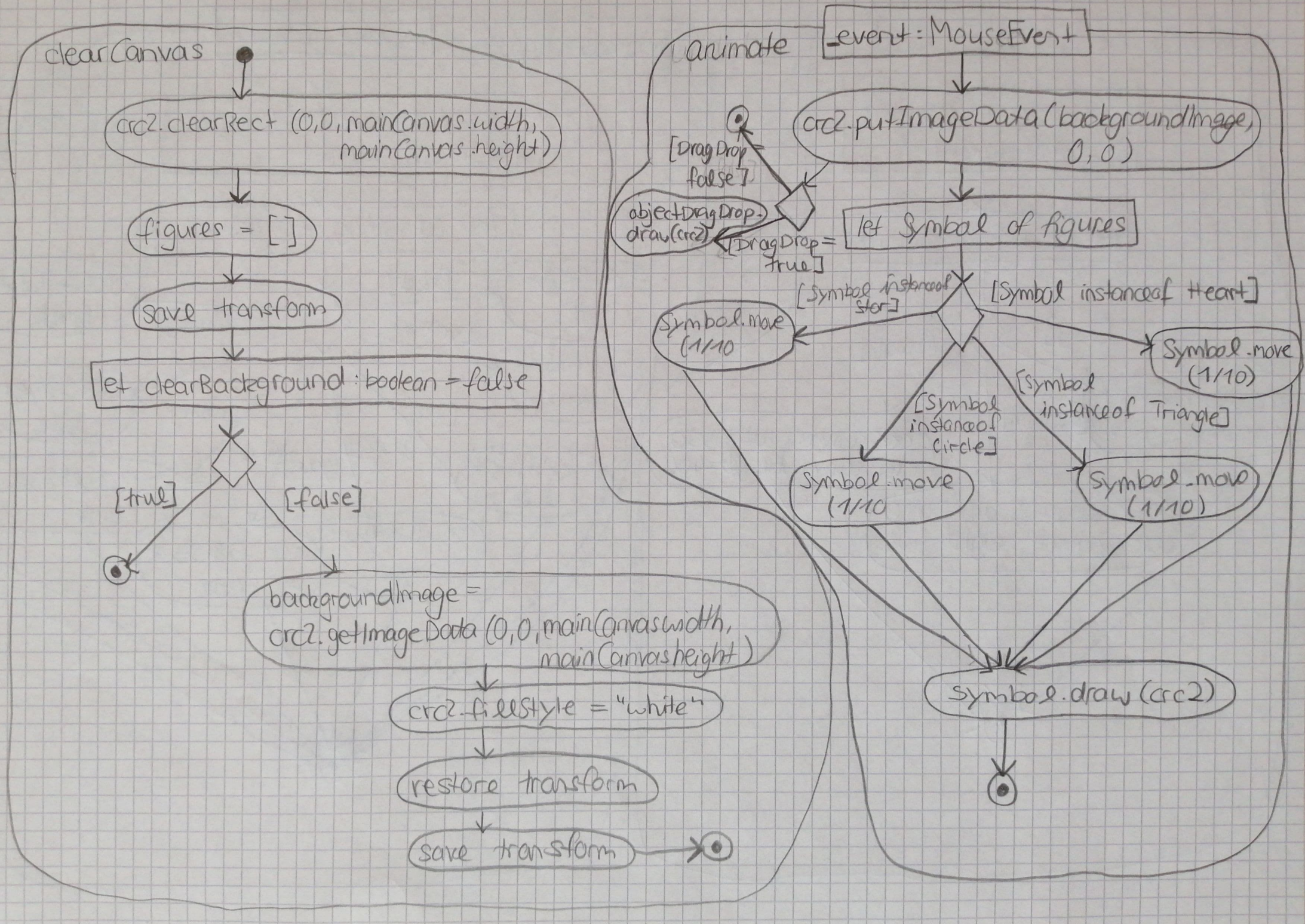
Aktivitätsdiagramm Symbol

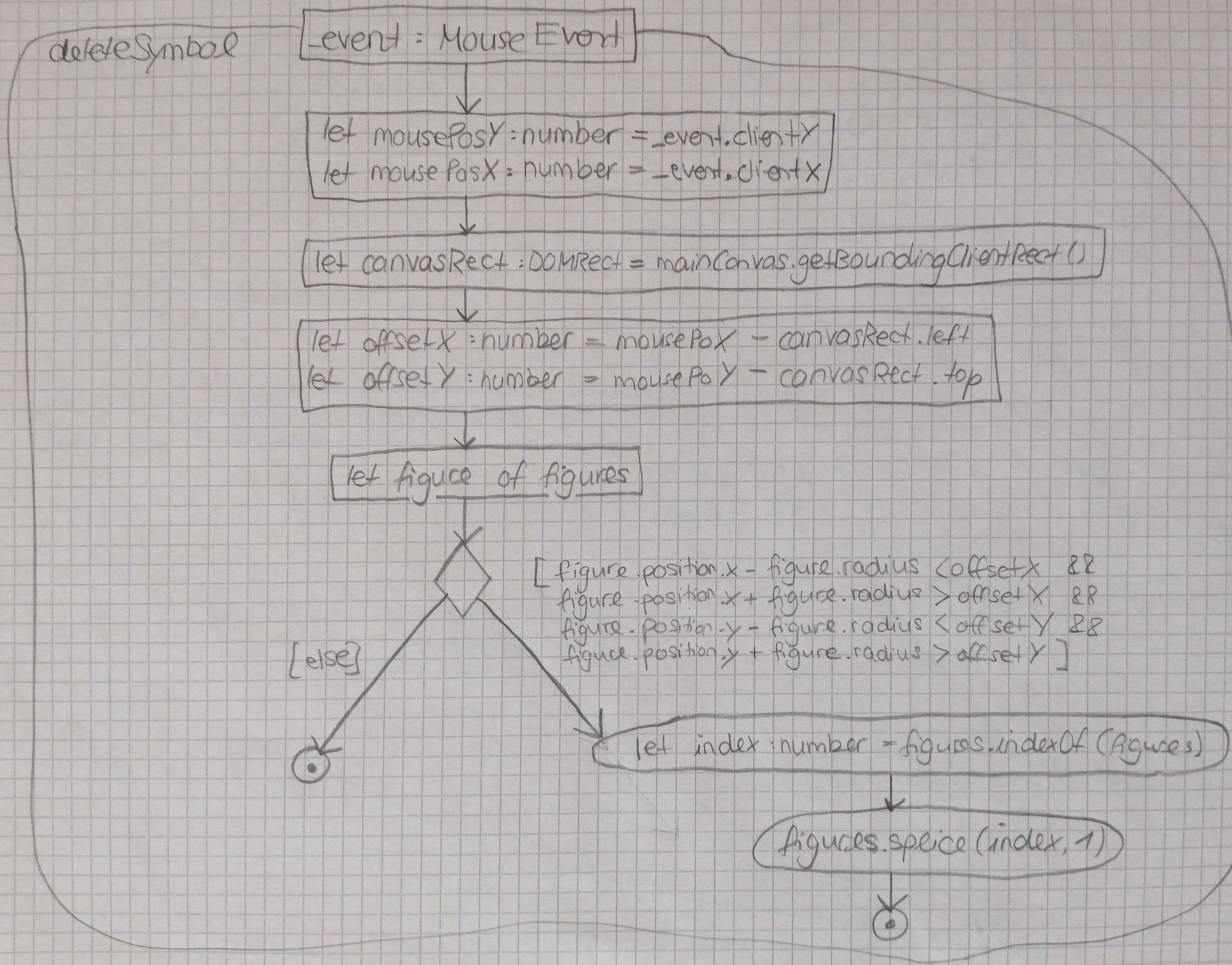


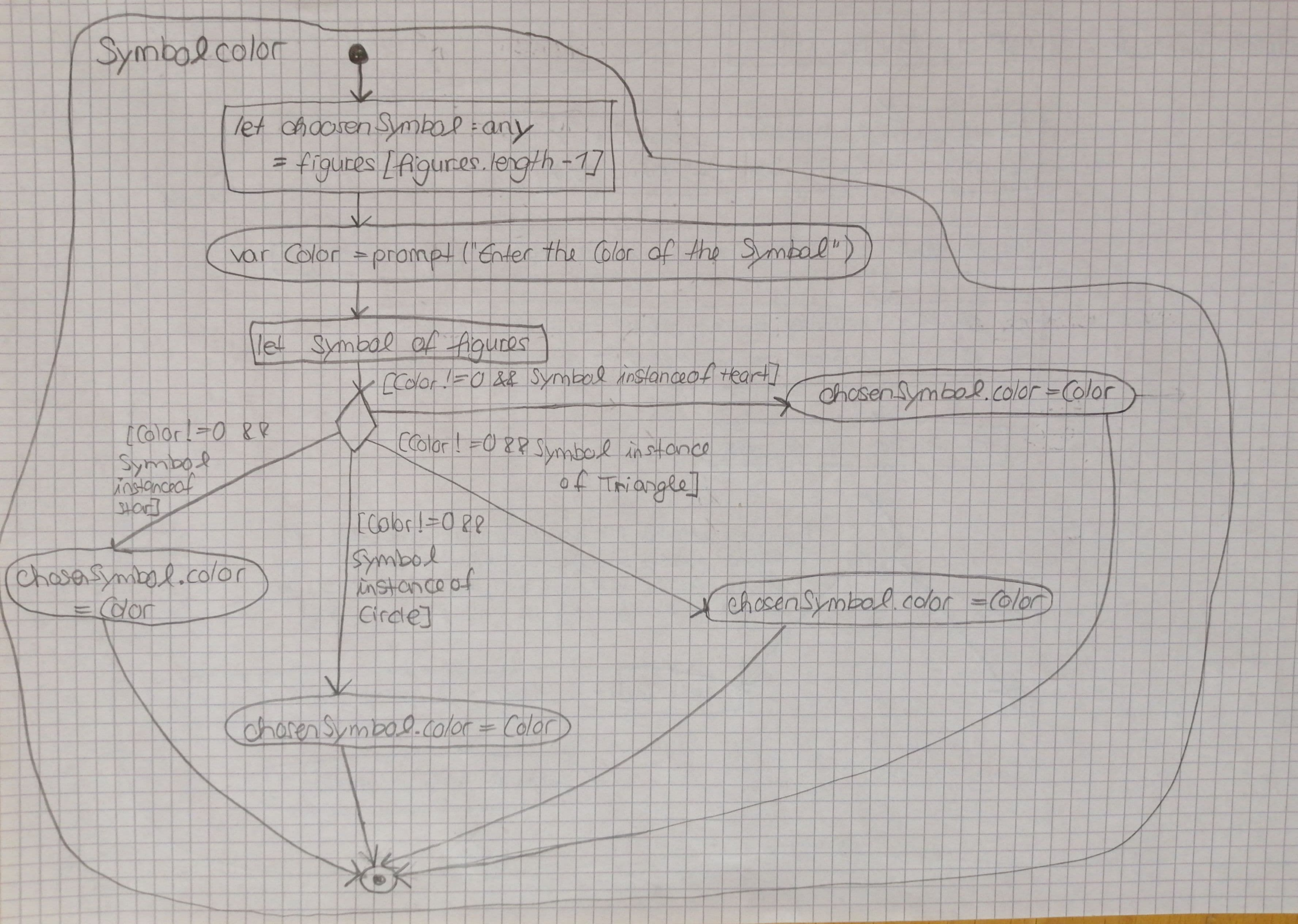


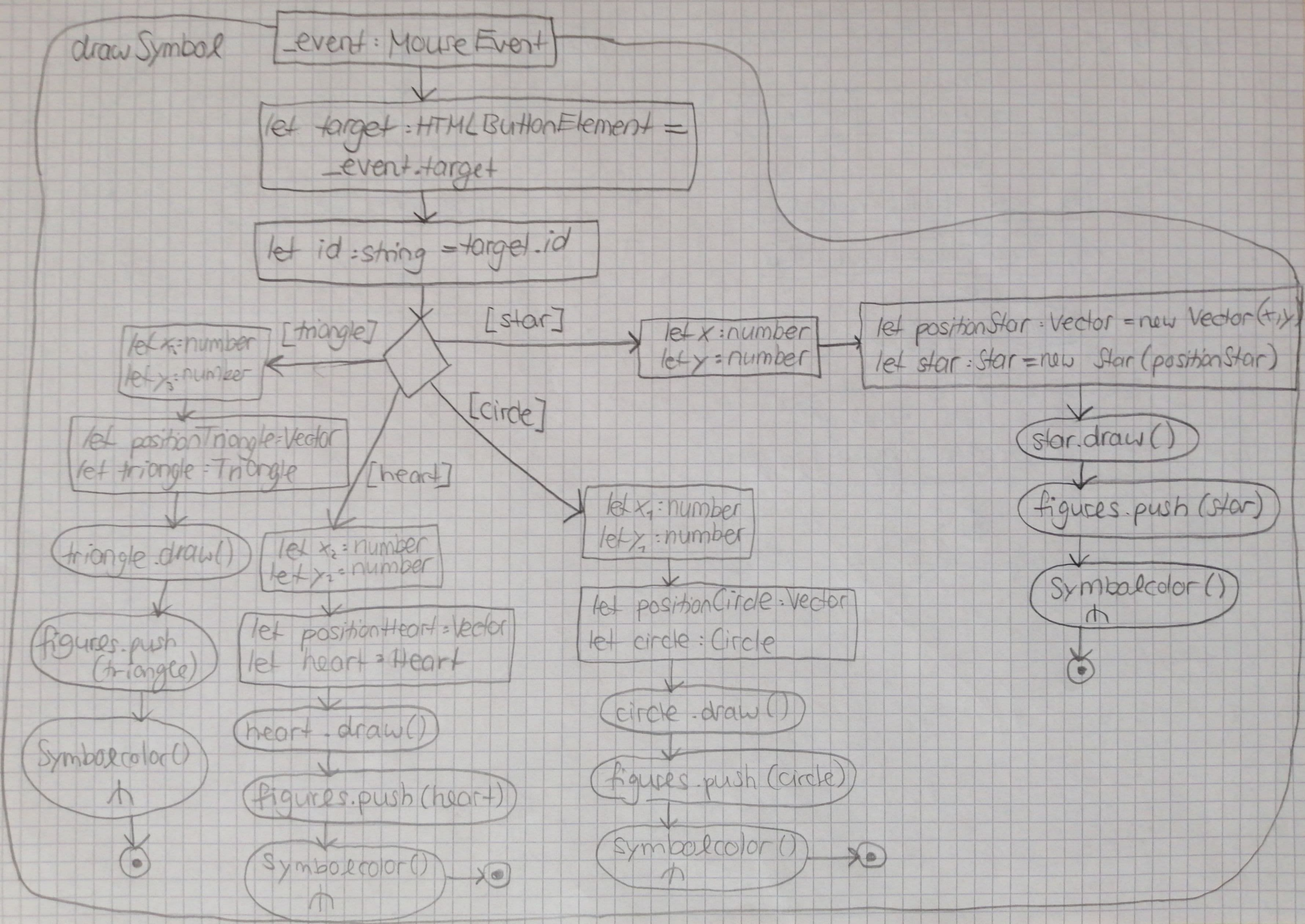


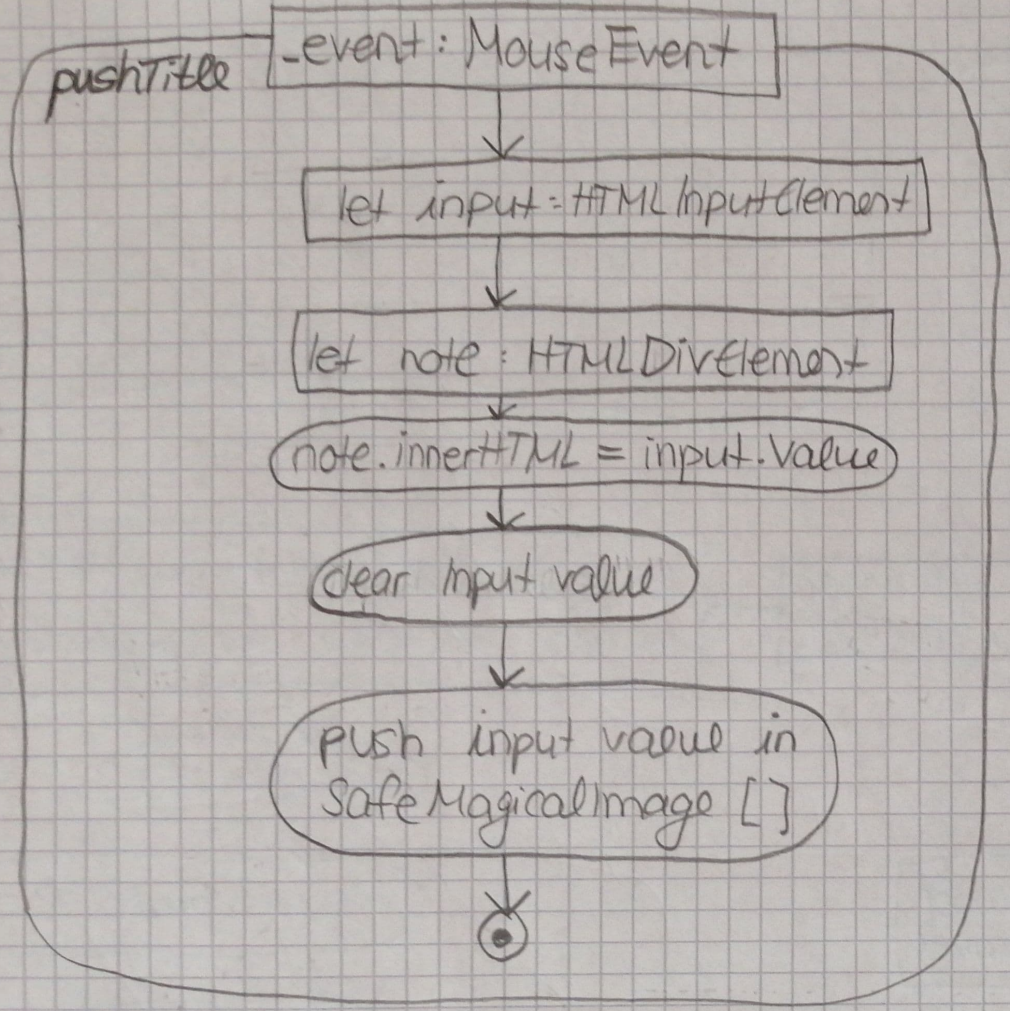

chooseCanvas
-event : Event
let target : HTMLElement = -event.target
let id : string = target.id
[big]
[small]
[medium]
crc2.canvas.width = 550
crc2.canvas.height = 400
crc2.canvas.width = 350
crc2.canvas.height = 400
crc2.canvas.width = 450
crc2.canvas.height = 400
chooseBackground
-event : Event
let target : HTMLElement = -event.target
let id : string = target.id
[red]
[yellow]
[blue]
[orange]
[green]
[pink]
[white]
[black]
crc2.fillStyle = "red"
crc2.fillStyle = "yellow"
crc2.fillStyle = "blue"
crc2.fillStyle = "orange"
crc2.fillStyle = "green"
crc2.fillStyle = "pink"
crc2.fillStyle = "white"
crc2.fillStyle = "black"

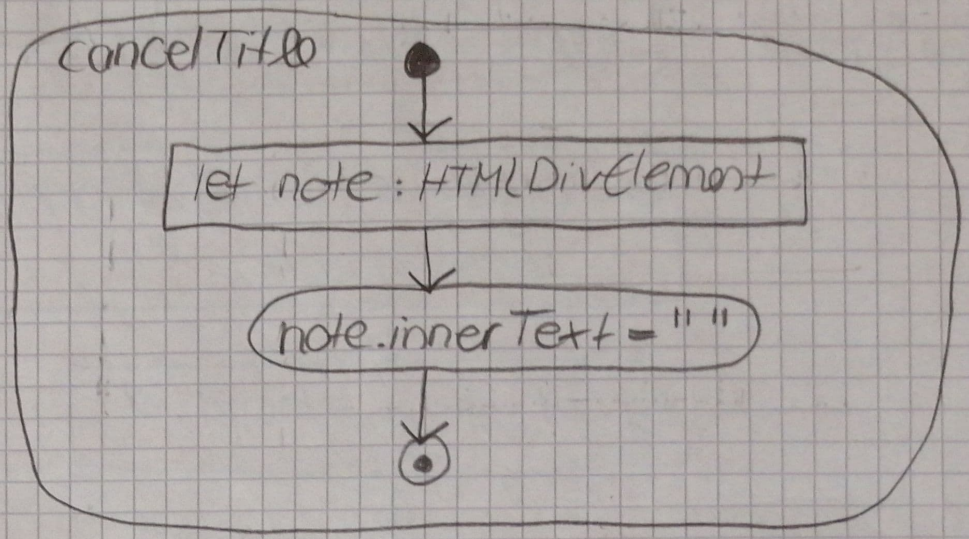

clearCanvas
crc2.clearRect (0,0, mainCanvas.width, mainCanvas.height)
figures = []
save transform
let clearBackground : boolean = false
[true]
[false]
backgroundImage = crc2.getImageData (0,0, mainCanvas.width, mainCanvas.height)
crc2.fillStyle = "white"
restore transform
save transform
animate
_event : MouseEvent
crc2.putImageData (backgroundImage, 0, 0)
[DragDrop = false]
objectDragDrop.draw(crc2)
[DragDrop = true]
let Symbol of figures
[Symbol instanceof Star]
[Symbol instanceof Heart]
[Symbol instanceof Circle]
[Symbol instanceof Triangle]
Symbol.move (1/10)
Symbol.move (1/10)
Symbol.move (1/10)
Symbol.move (1/10)
Symbol.draw (crc2)

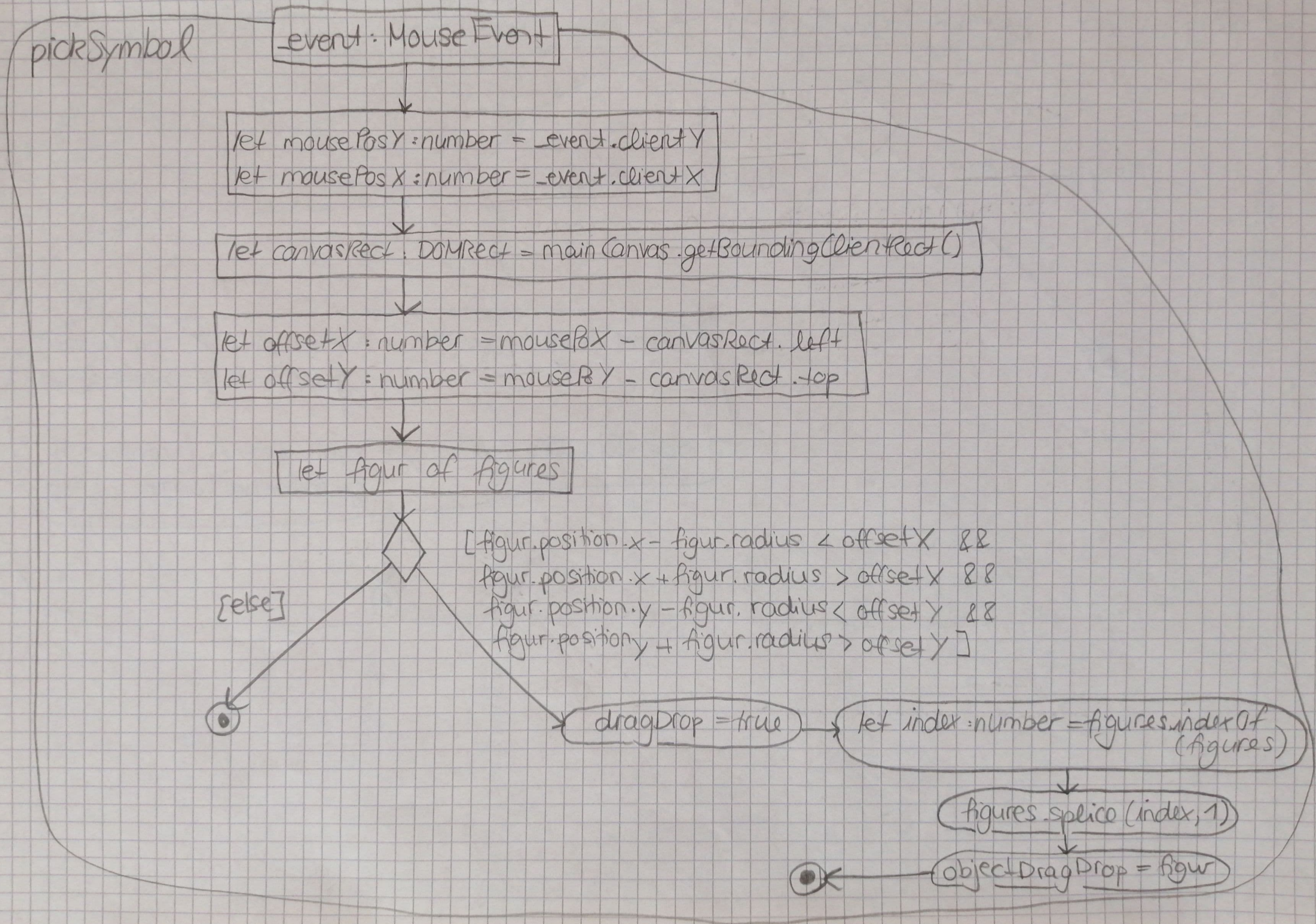

deleteSymbol
_event : MouseEvent
let mousePosY : number = _event.clientY
let mousePosX : number = _event.clientX
let canvasRect : DOMRect = mainCanvas.getBoundingClientRect()
let offsetX : number = mousePosX - canvasRect.left
let offsetY : number = mousePosY - canvasRect.top
let figure of figures
[figure.position.x - figure.radius < offsetX &&
figure.position.x + figure.radius > offsetX &&
figure.position.y - figure.radius < offsetY &&
figure.position.y + figure.radius > offsetY]
[else]
let index : number = figures.indexOf(figure)
figures.speice(index, 1)

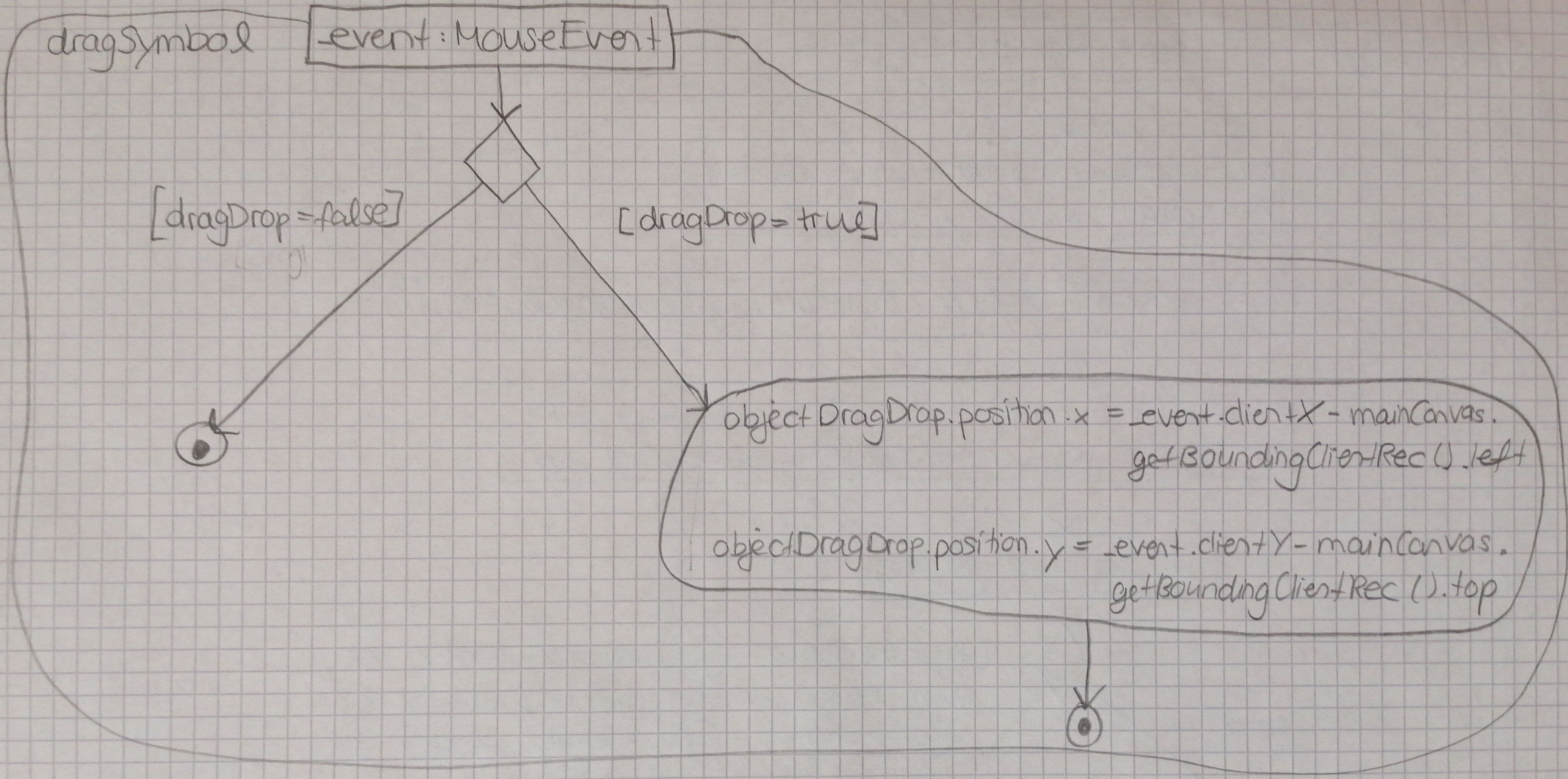

Symbol color
let chosenSymbol : any
= figures [figures.length - 1]
var Color = prompt ("Enter the Color of the Symbol")
let Symbol of figures
[Color != 0 && Symbol instanceof Heart]
chosenSymbol.color = Color
[Color != 0 && Symbol instanceof Star]
[Color != 0 && Symbol instance of Triangle]
[Color != 0 && Symbol instance of Circle]
chosenSymbol.color = Color
chosenSymbol.color = Color
chosenSymbol.color = Color

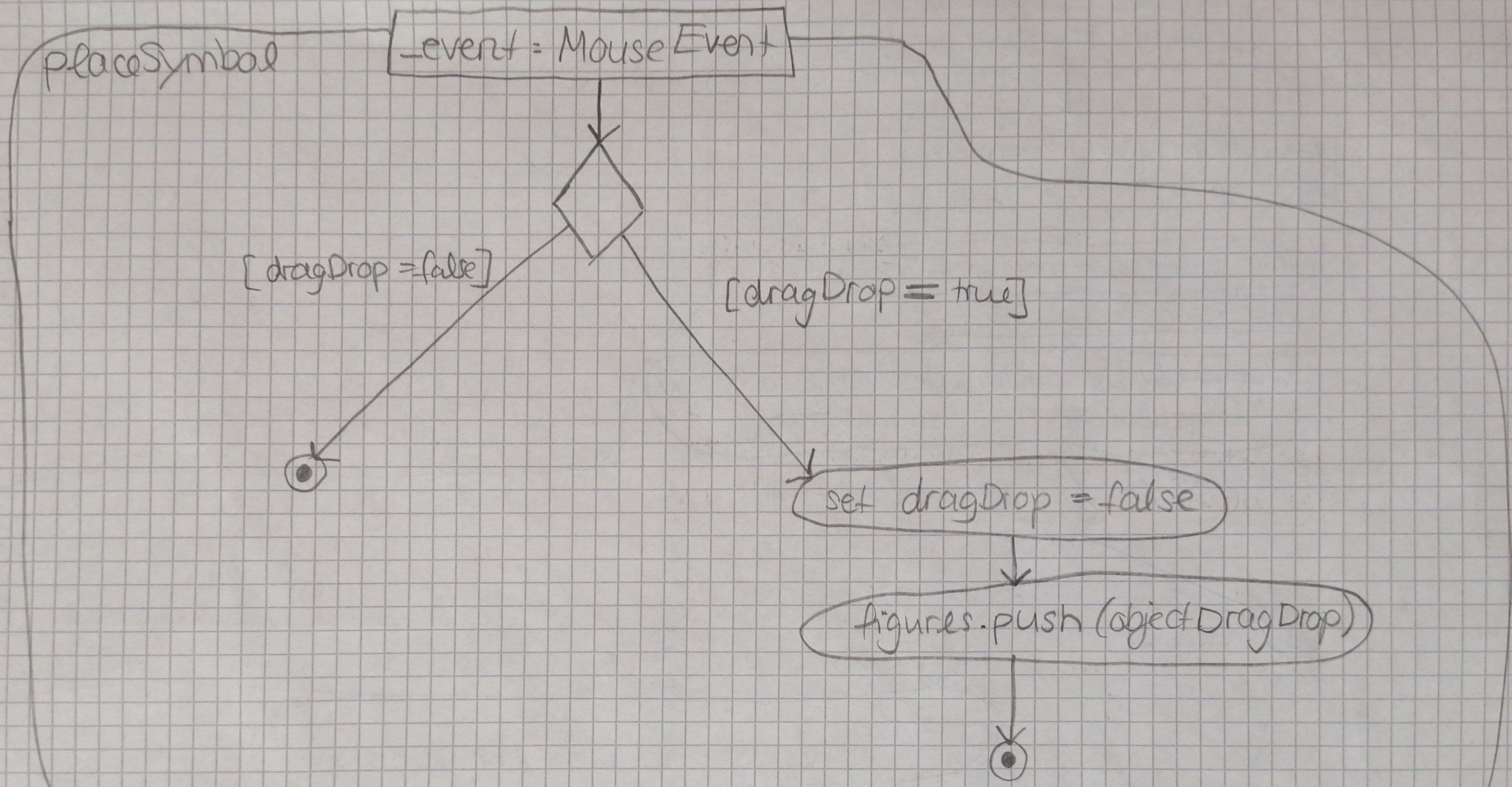

drawSymbol
_event : MouseEvent
let target : HTMLButtonElement = _event.target
let id : string = target.id
[triangle]
[star]
[heart]
[circle]
let x1 : number
let y3 : number
let positionTriangle : Vector
let triangle : Triangle
triangle.draw()
figures.push (triangle)
Symbolcolor()
let x2 : number
let y2 : number
let positionHeart : Vector
let heart : Heart
heart.draw()
figures.push (heart)
Symbolcolor()
let x1 : number
let y1 : number
let positionCircle : Vector
let circle : Circle
circle.draw()
figures.push (circle)
Symbolcolor()
let x : number
let y : number
let positionStar : Vector = new Vector (x,y)
let star : Star = new Star (positionStar)
star.draw()
figures.push (star)
Symbolcolor()


pushTitle
-event : MouseEvent
let input : HTMLInputElement
let note : HTMLDivElement
note.innerHTML = input.value
clear input value
push input value in SafeMagicalImage []

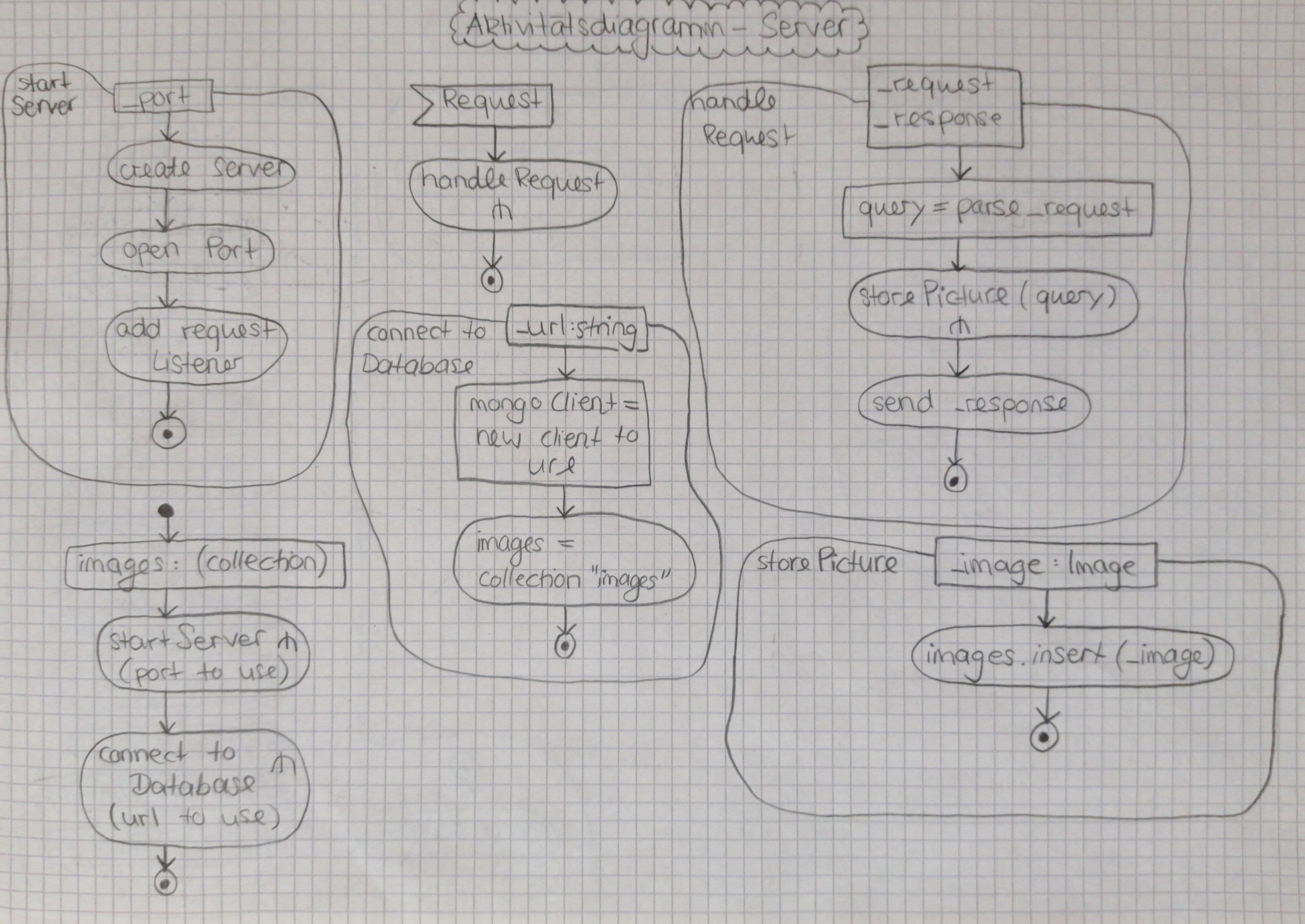

cancelTitle
let note : HTMLDivElement
note.innerText = ""

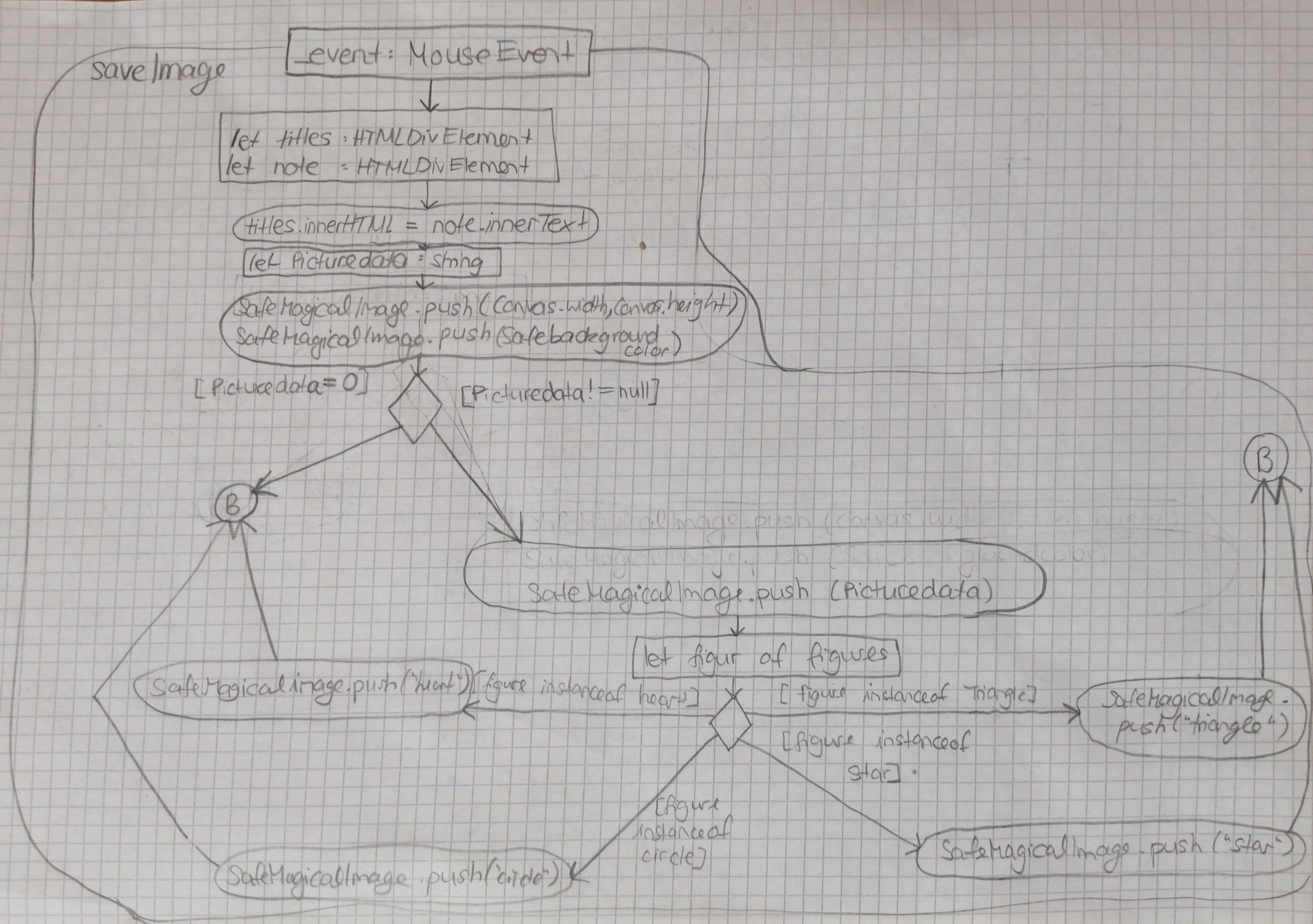

pickSymbol
_event : MouseEvent
let mousePosY : number = _event.clientY
let mousePosX : number = _event.clientX
let canvasRect : DOMRect = mainCanvas.getBoundingClientRect()
let offsetX : number = mousePosX - canvasRect.left
let offsetY : number = mousePosY - canvasRect.top
let figur of figures
[figur.position.x - figur.radius < offsetX &&
figur.position.x + figur.radius > offsetX &&
figur.position.y - figur.radius < offsetY &&
figur.positionY + figur.radius > offsetY]
[else]
dragDrop = true
let index : number = figures.indexOf(figures)
figures.splice(index, 1)
objectDragDrop = figur


dragSymbol
_event : MouseEvent
[dragDrop = false]
[dragDrop = true]
objectDragDrop.position.x = _event.clientX - mainCanvas.
getBoundingClientRec().left
objectDragDrop.position.y = _event.clientY - mainCanvas.
getBoundingClientRec().top

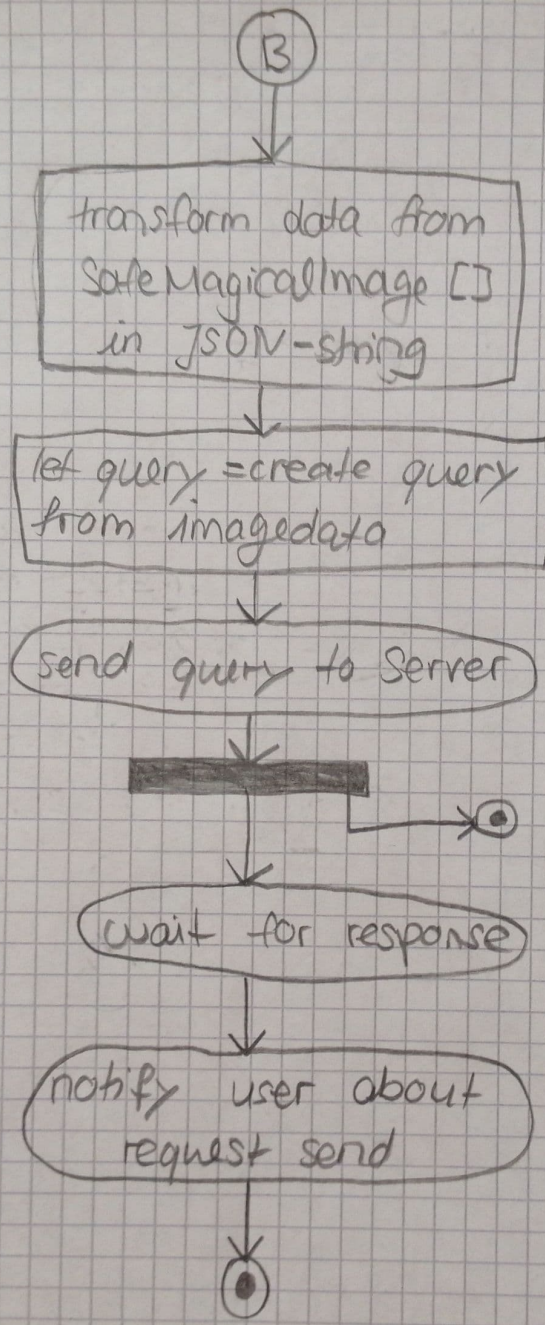

placeSymbol
_event : MouseEvent
[dragDrop = false]
[dragDrop == true]
set dragDrop = false
figures.push(objectDragDrop)

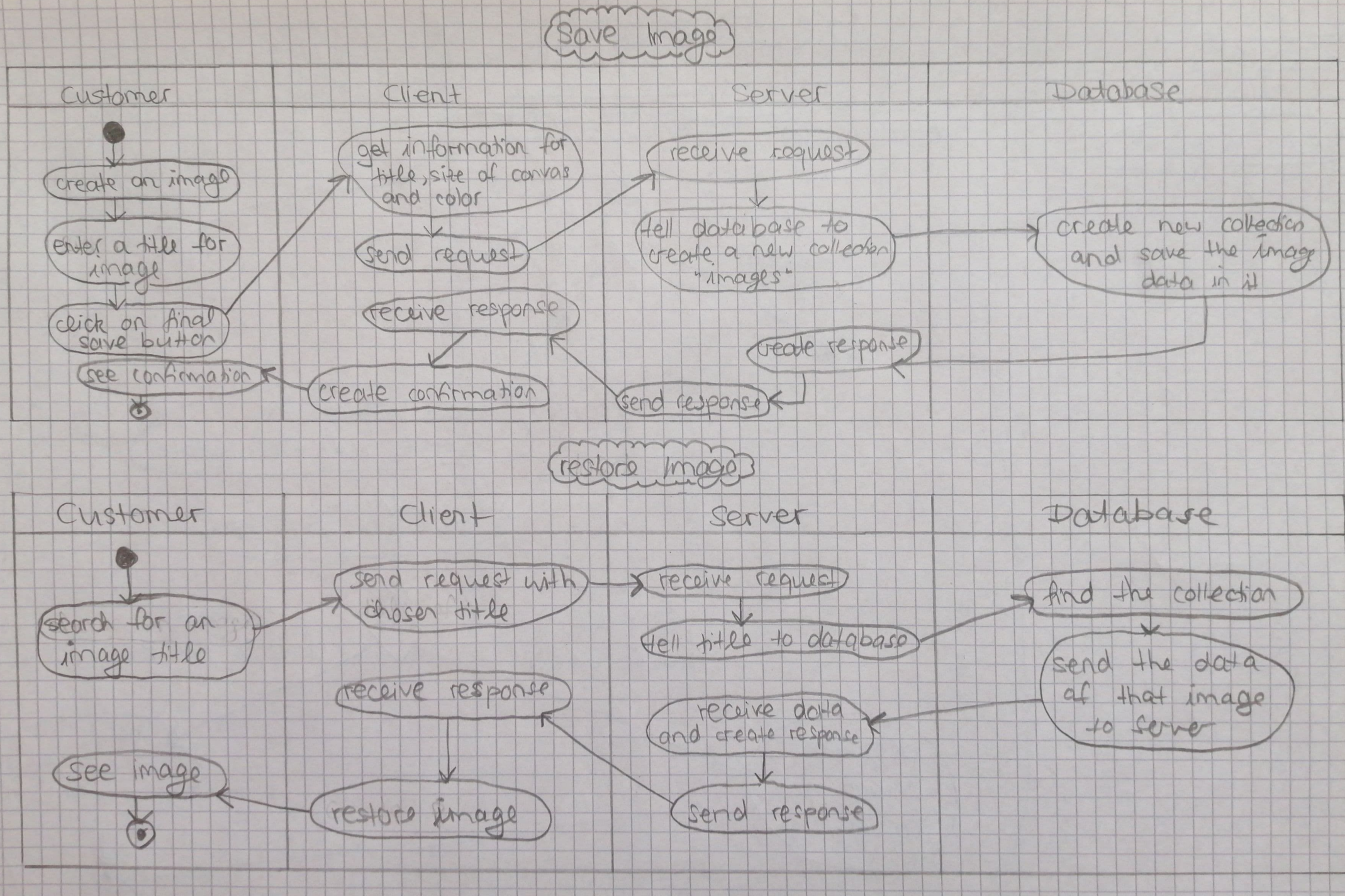
Aktivitätsdiagramm - Server

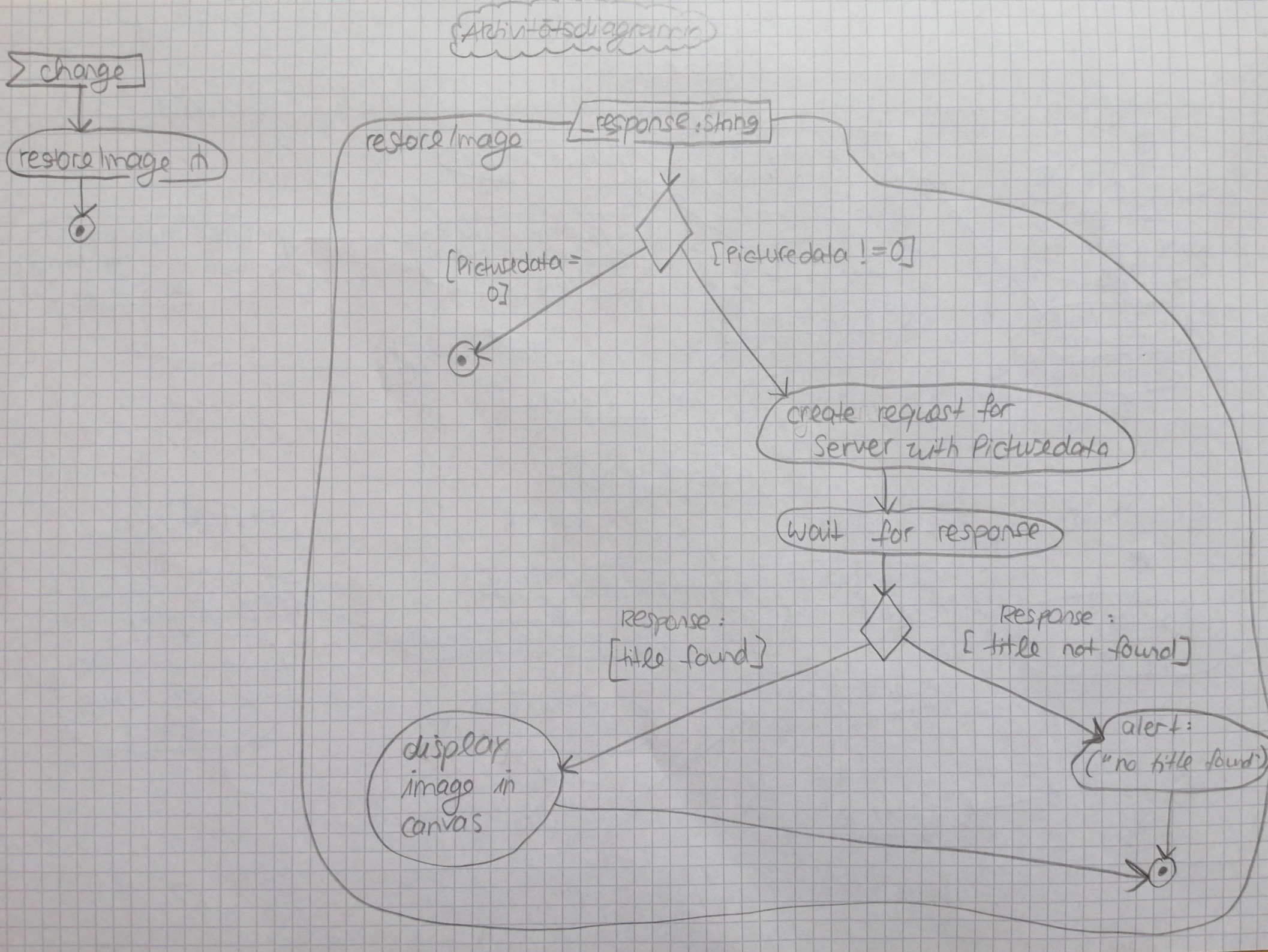



saveImage
_event : MouseEvent
let titles : HTMLDivElement
let note : HTMLDivElement
titles.innerHTML = note.innerText
let Picturedata : string
SafeMagicalImage.push(Canvas.width, Canvas.height)
SafeMagicalImage.push(Safebackground color)
[Picturedata=0]
[Picturedata!=null]
B
B
SafeMagicalImage.push (Picturedata)
let figur of figures
SafeMagicalImage.push("heart")
[figure instanceof heart]
[figure instanceof Triangle]
SafeMagicalImage.push("triangle")
[figure instanceof star]
[figure instanceof circle]
SafeMagicalImage.push("circle")
SafeMagicalImage.push("star")

saveImage


B
transform data from SafeMagicalImage [] in JSON-string
let query = create query from imagedata
send query to Server
wait for response
notify user about request send


Save Image
Customer
Client
Server
Database
create an image
enter a title for image
click on final save button
see confirmation
get information for title, size of canvas and color
send request
receive response
create confirmation
receive request
tell database to create a new collection "images"
create new collection and save the image data in it
create response
send response
restore image
Customer
Client
Server
Database
search for an image title
send request with chosen title
receive response
see image
restore image
receive request
tell title to database
receive data and create response
send response
find the collection
send the data of that image to server


Aktivitätsdiagramm
change
restoreImage()
restoreImage
_response:string
[Picturedata = 0]
[Picturedata != 0]
create request for Server with Picturedata
wait for response
Response: [title found]
Response: [title not found]
display image in canvas
alert: ("no title found")