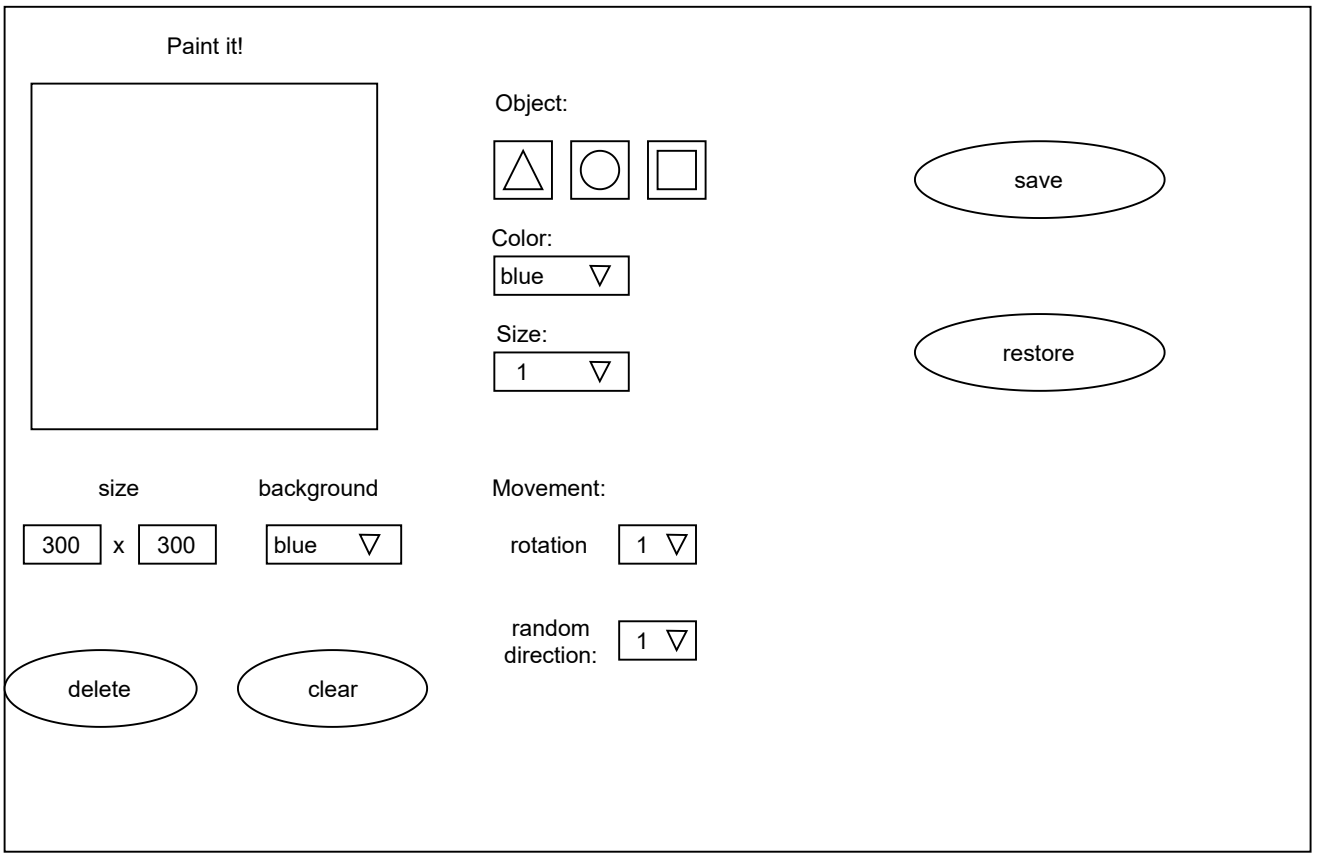
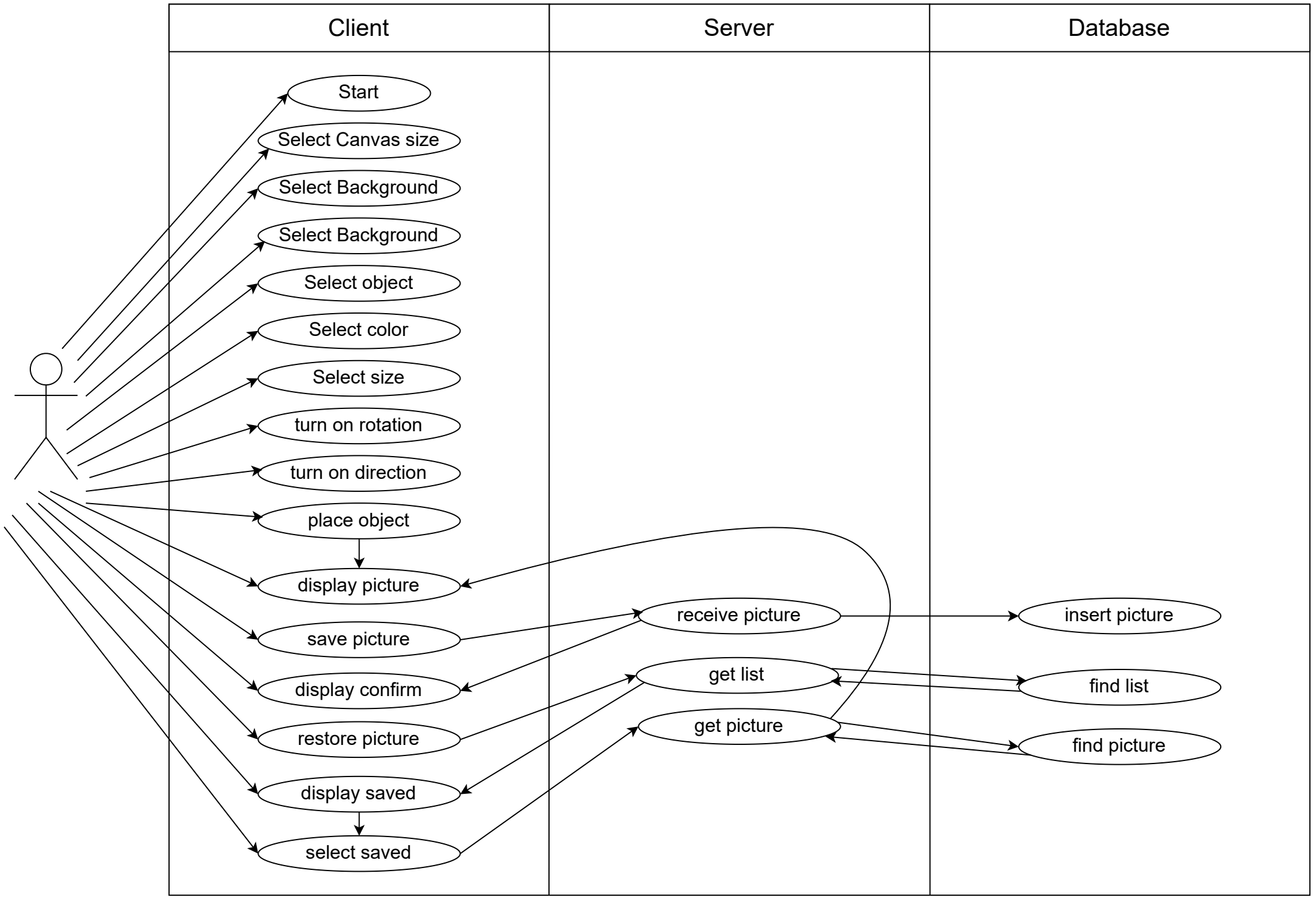
EIA 2 - Endabgabe
Konzept

User-Interface



User Interface



Klassendiagramm

crc2

Vector

x: number
y:number

constructor(_x:number,y:number)
set(_x:number, y:number):void
scale(_factor:number):void
add(addend:vector):void
copy():vector

Figure

position: Vector
velocity: Vector
rotation:number

constructor(_position: Vector,_parameter?:number,_rotation?:number)
move(_timeslice:number):void
draw():void
rotate():void

Circle

color:string, size:number

constructor(_color:string,_size:number)
draw():void

Triangle

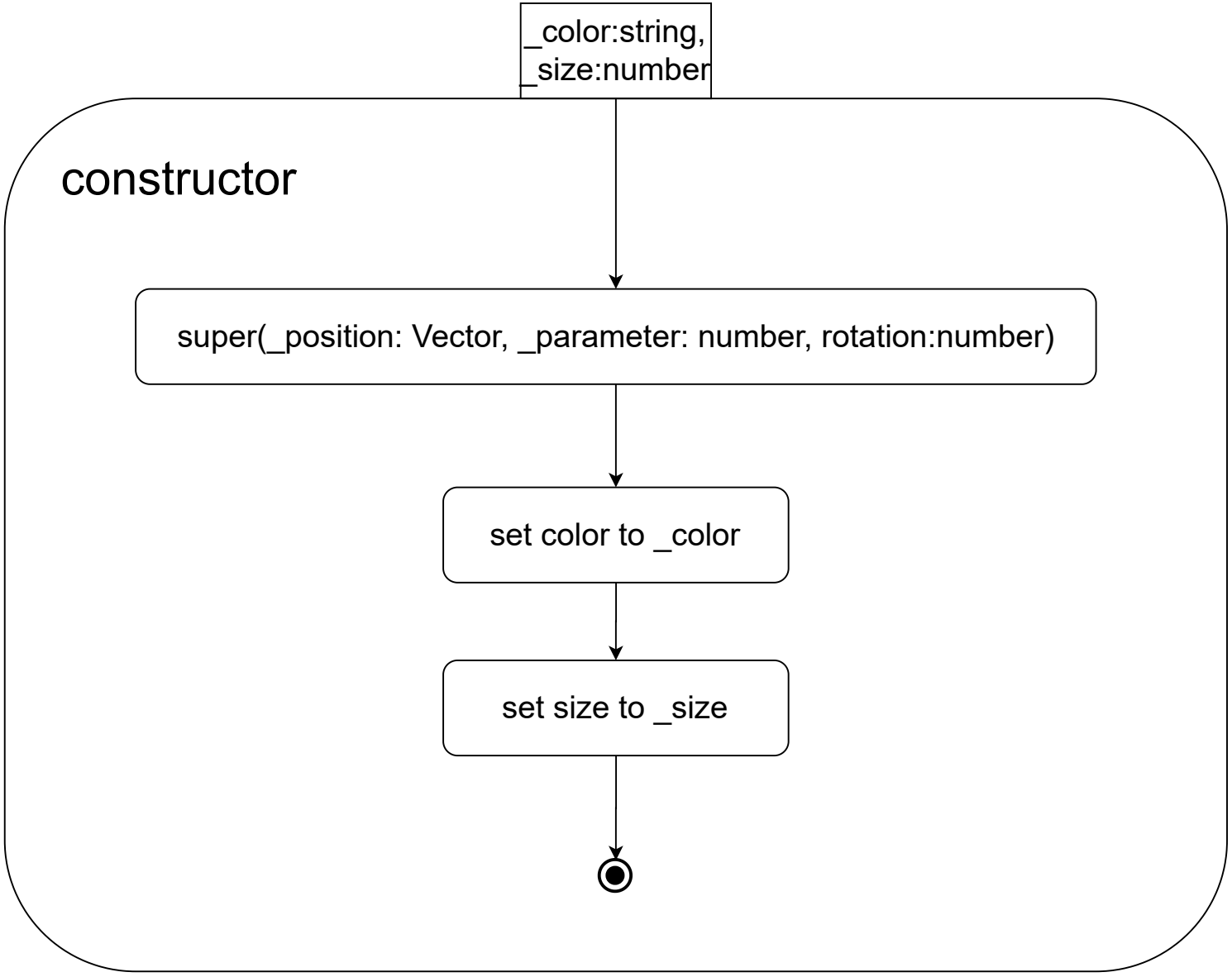
color:string, size:number

constructor(_color:string,_size:number)
draw():void

Circle

color:string, size:number

constructor(_color:string,_size:number)
draw():void


_color:string,
_size:number
constructor
super(_position: Vector, _parameter: number, rotation:number)
set color to _color
set size to _size

draw

crc2.reset transform

crc2.translate to position

draw path

crc2.fillStyle = color

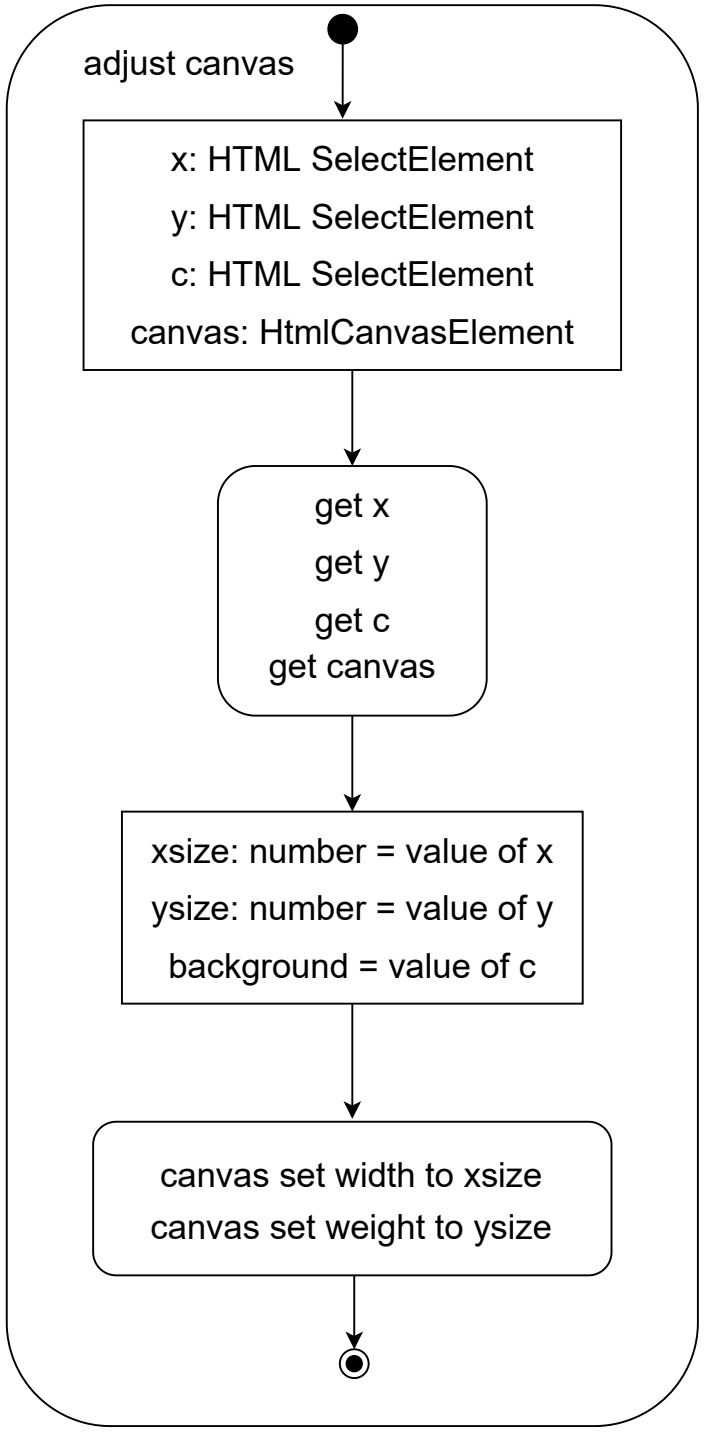
[size > 0]

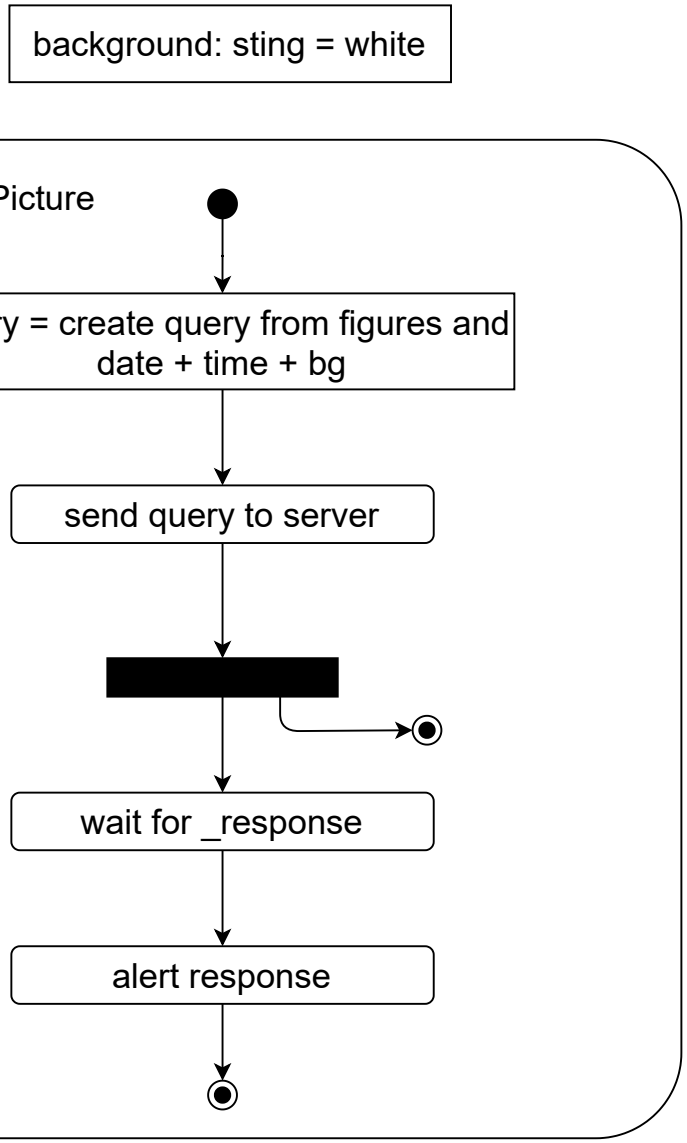
[size == 0]

crc2.scale (this.size, this.size)

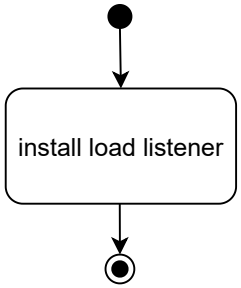
crc2.fill()

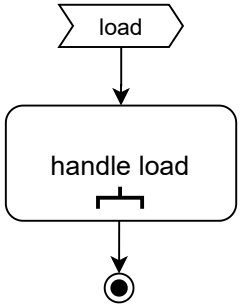
AD-Main

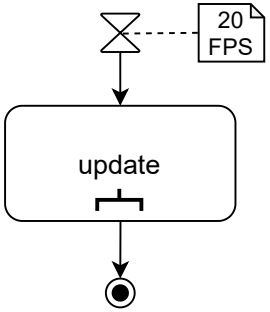

adjust canvas
x: HTML SelectElement
y: HTML SelectElement
c: HTML SelectElement
canvas: HtmlCanvasElement
get x
get y
get c
get canvas
xsize: number = value of x
ysize: number = value of y
background = value of c
canvas set width to xsize
canvas set weight to ysize

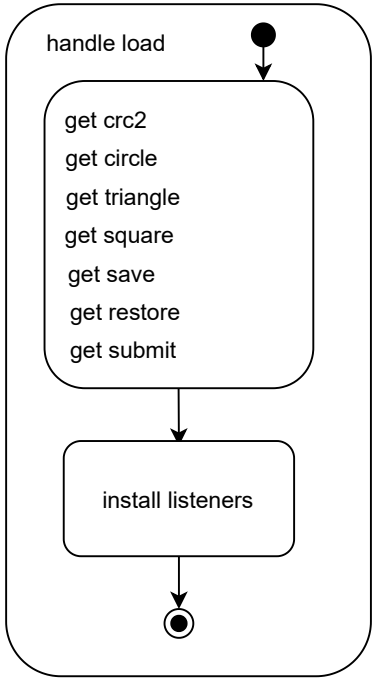

background: sting = white
async SavePicture
query = create query from figures and
date + time + bg
send query to server
wait for _response
alert response

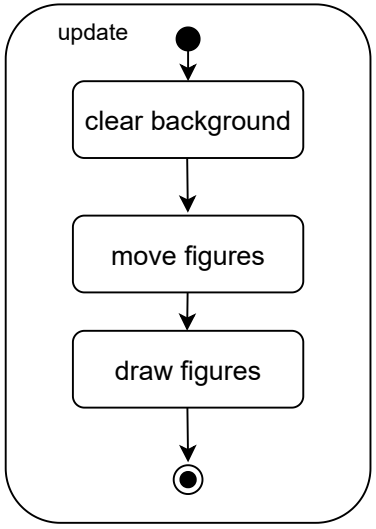
AD-Main

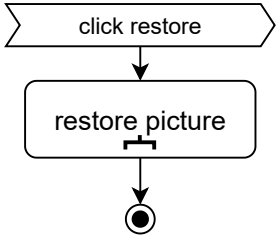




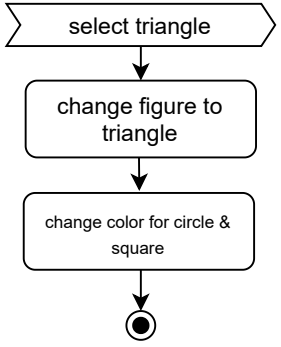


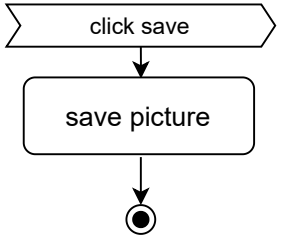


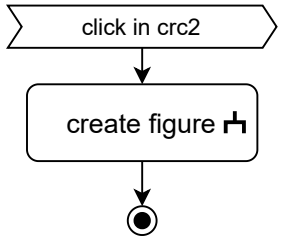


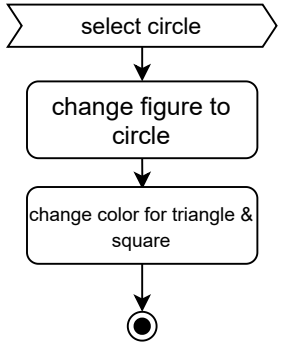


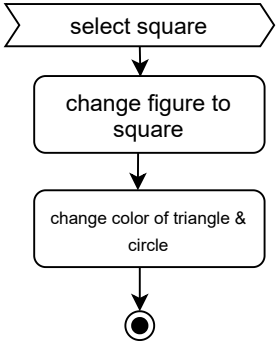
crc2: HTMLCanvasElement
figure: string
figures: Figure[ ]
circle: HTMLButtonElement
triangle: HTMLButtonElement
square: HTMLButtonElement
save: HTMLButtonElement
restore: HTMLButtonElement
submit: HTMLButtonElement

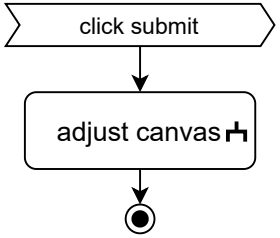


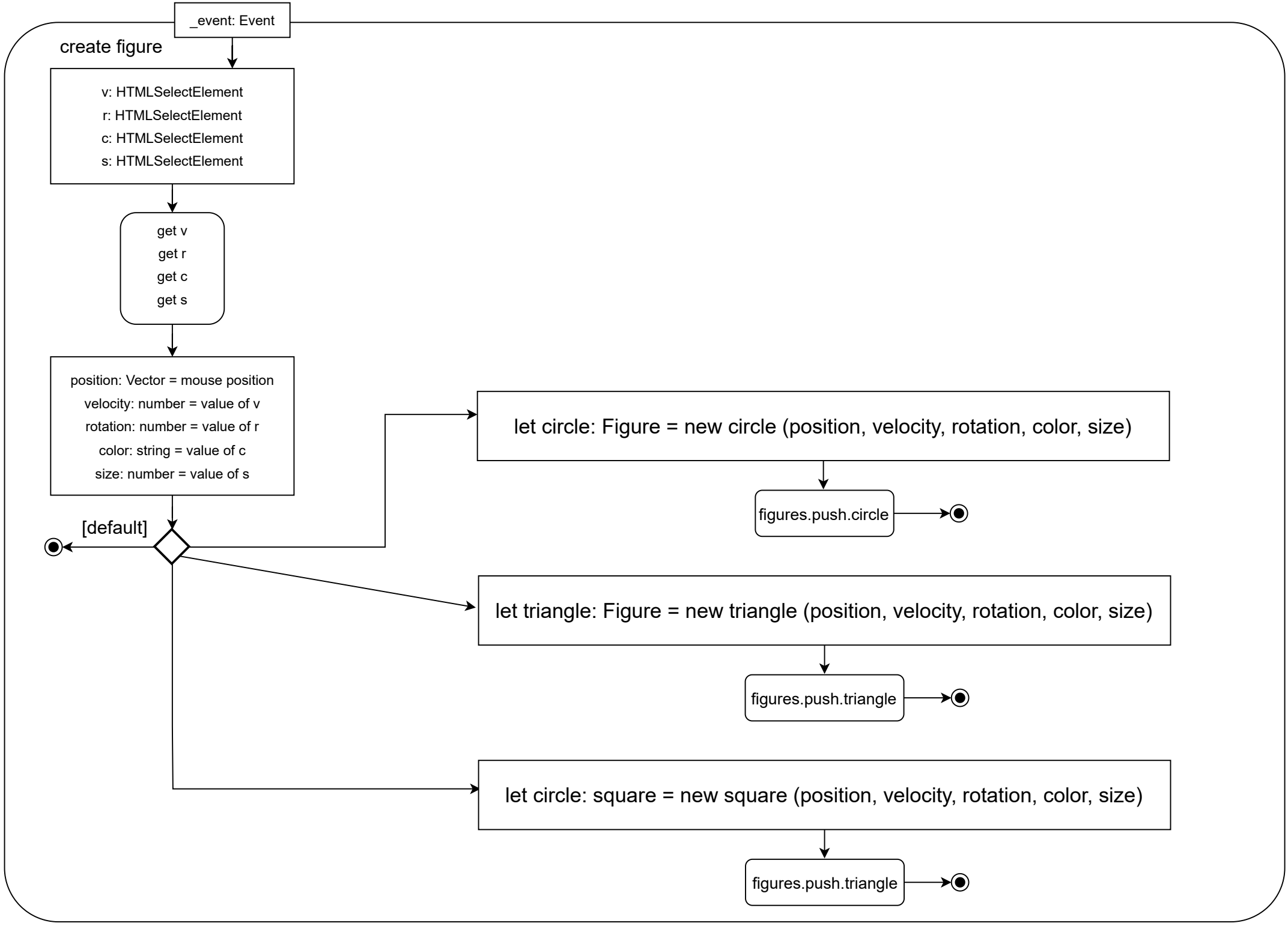




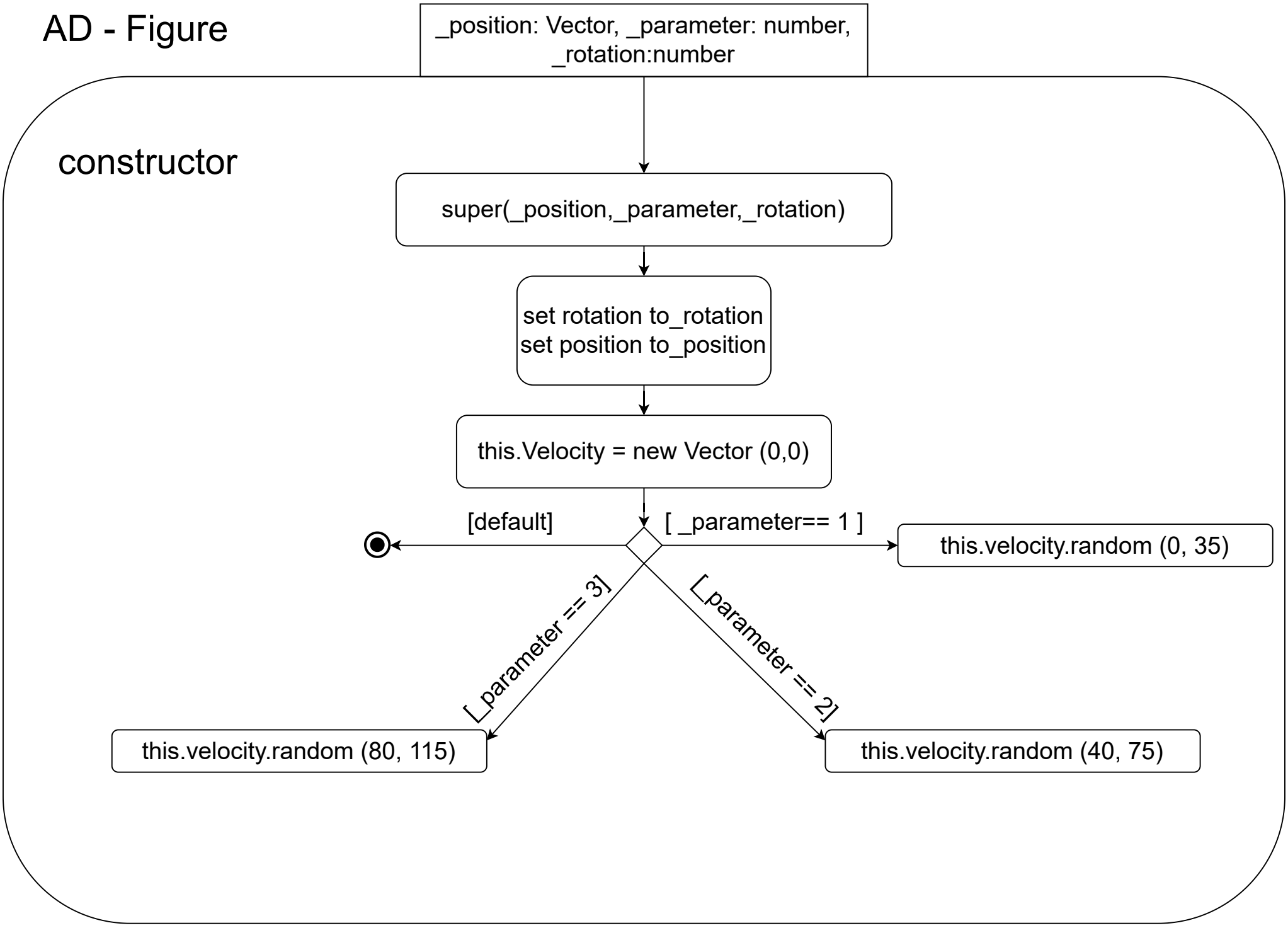


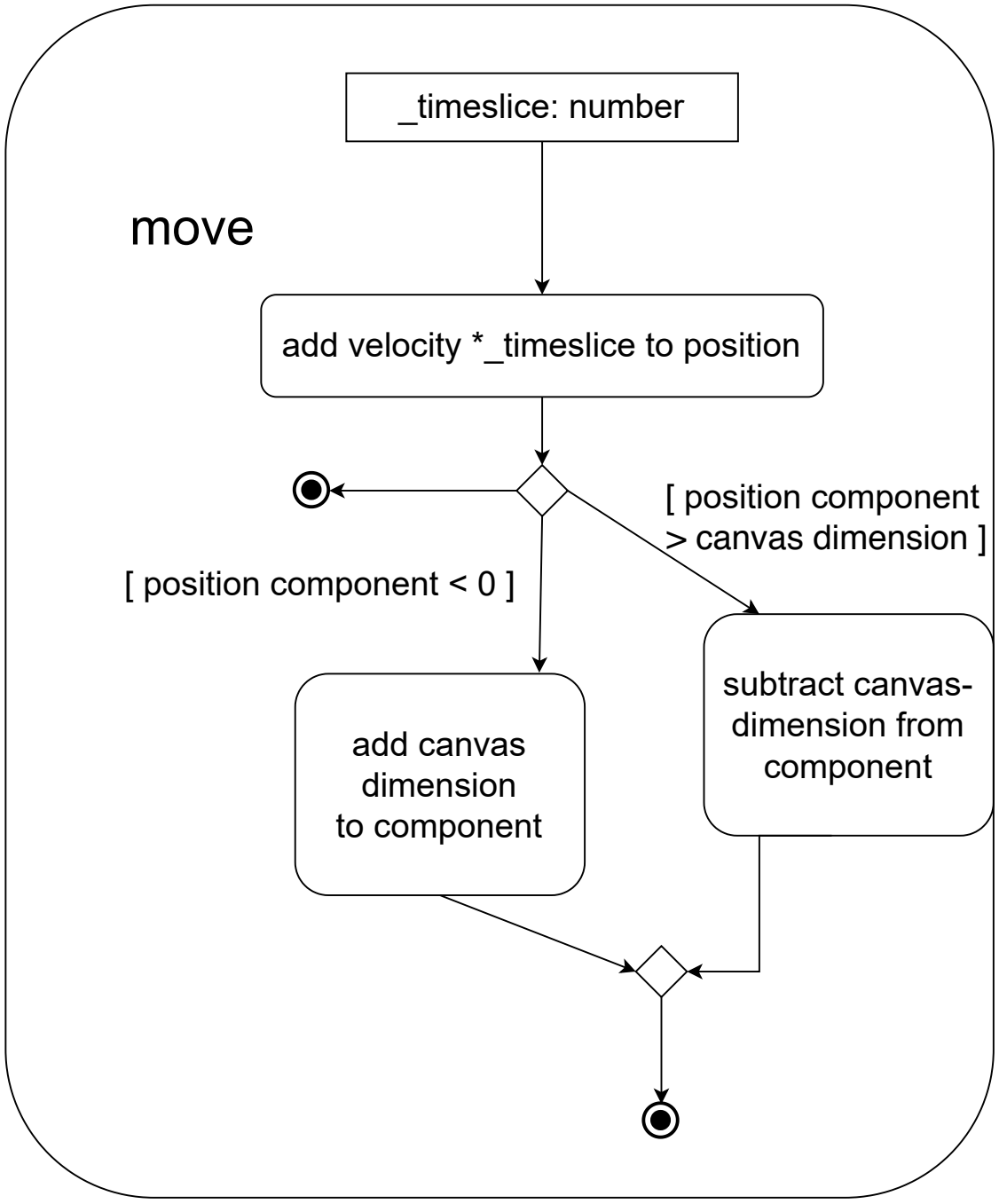


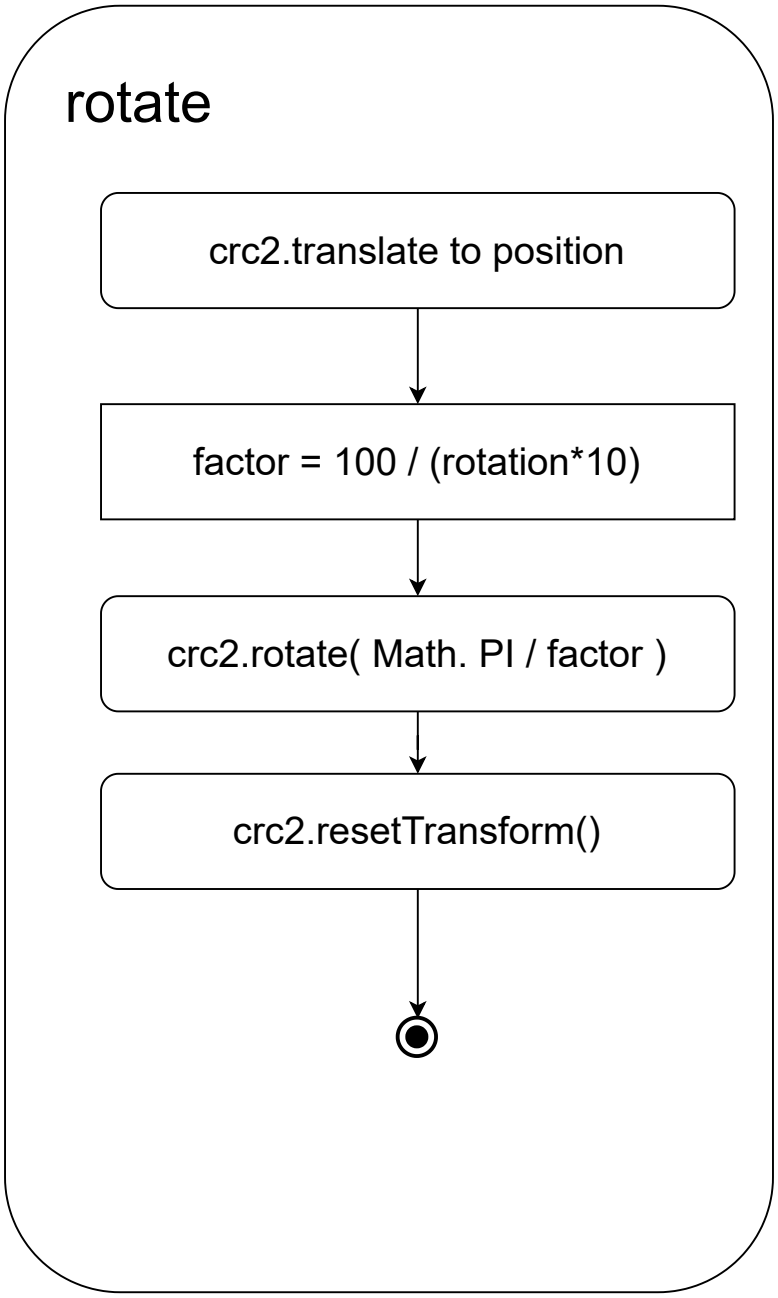


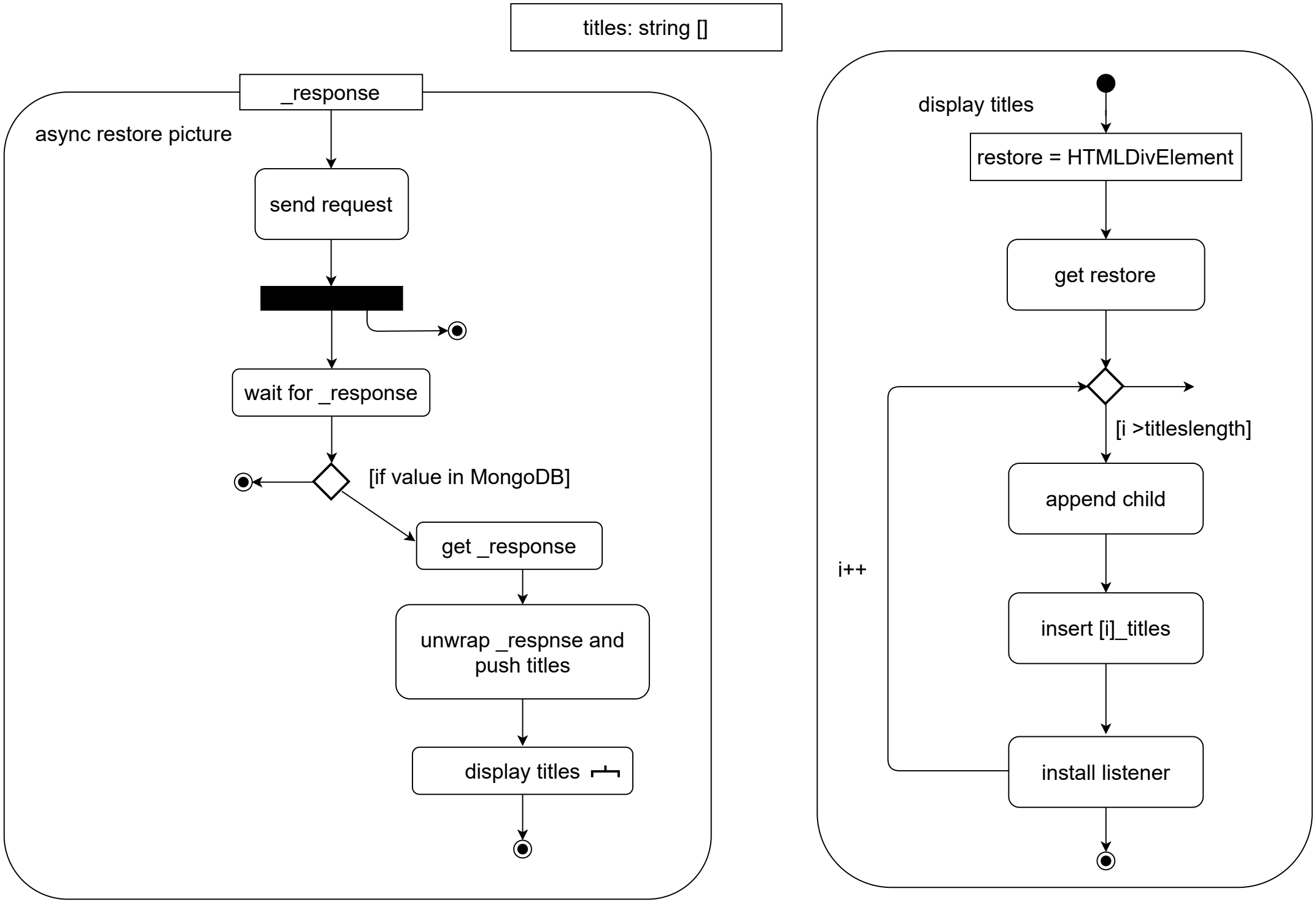

_event: Event
create figure
v: HTMLSelectElement
r: HTMLSelectElement
c: HTMLSelectElement
s: HTMLSelectElement
get v
get r
get c
get s
position: Vector = mouse position
velocity: number = value of v
rotation: number = value of r
color: string = value of c
size: number = value of s
[default]
let circle: Figure = new circle (position, velocity, rotation, color, size)
figures.push.circle
let triangle: Figure = new triangle (position, velocity, rotation, color, size)
figures.push.triangle
let circle: square = new square (position, velocity, rotation, color, size)
figures.push.triangle

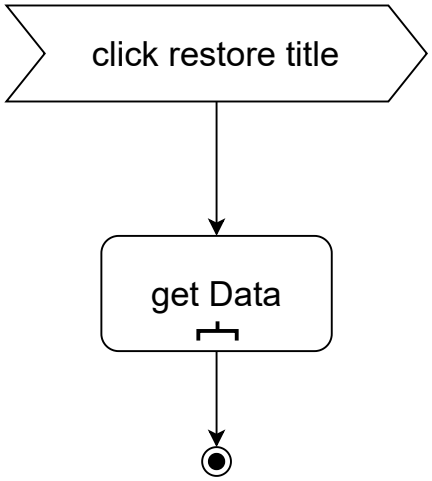
AD - Figure

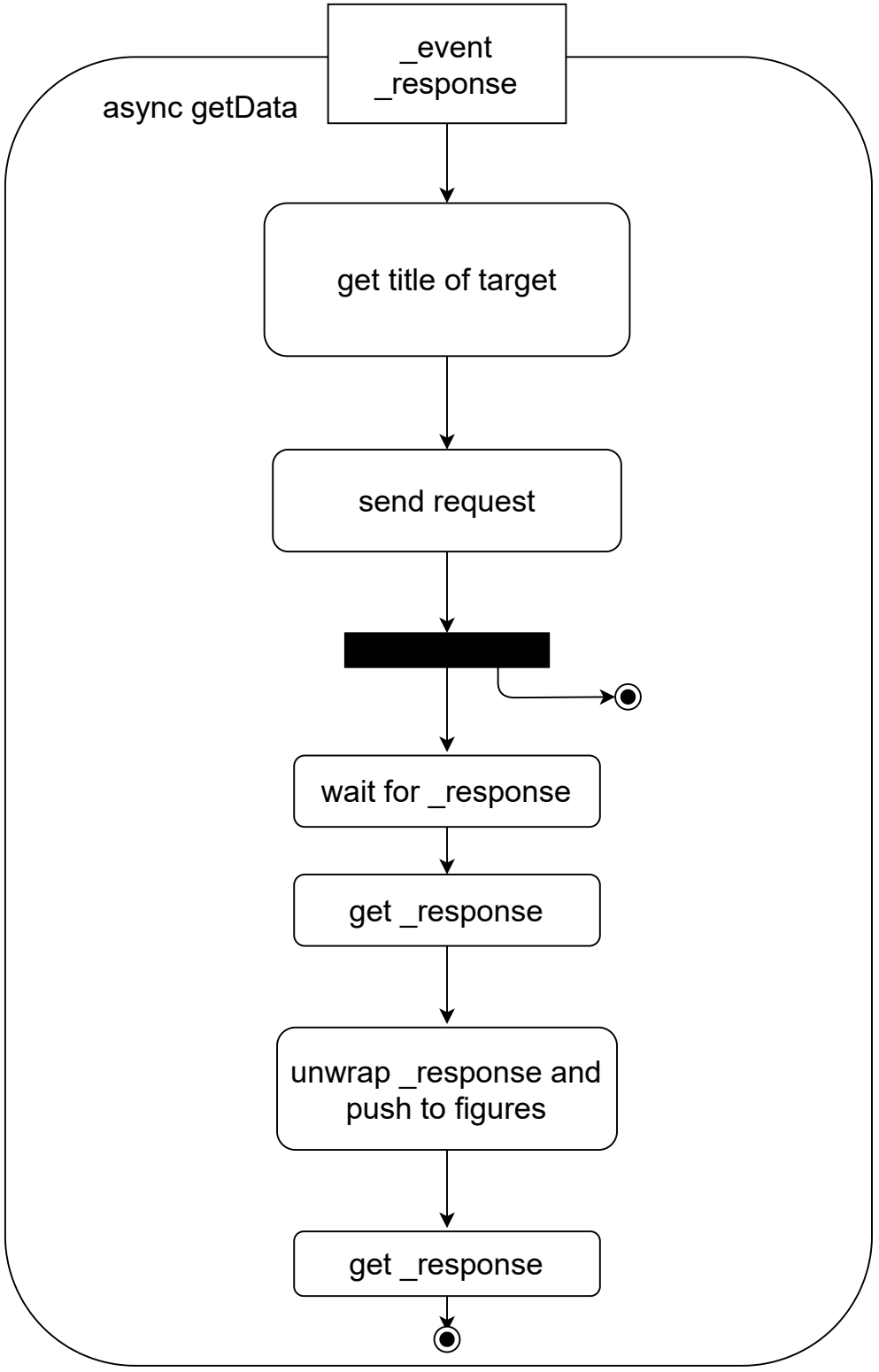

_position: Vector, _parameter: number,
_rotation:number
constructor
super(_position,_parameter,_rotation)
set rotation to_rotation
set position to_position
this.Velocity = new Vector (0,0)
[default]
[_parameter== 1]
this.velocity.random (0, 35)
[_parameter == 3]
[_parameter == 2]
this.velocity.random (80, 115)
this.velocity.random (40, 75)

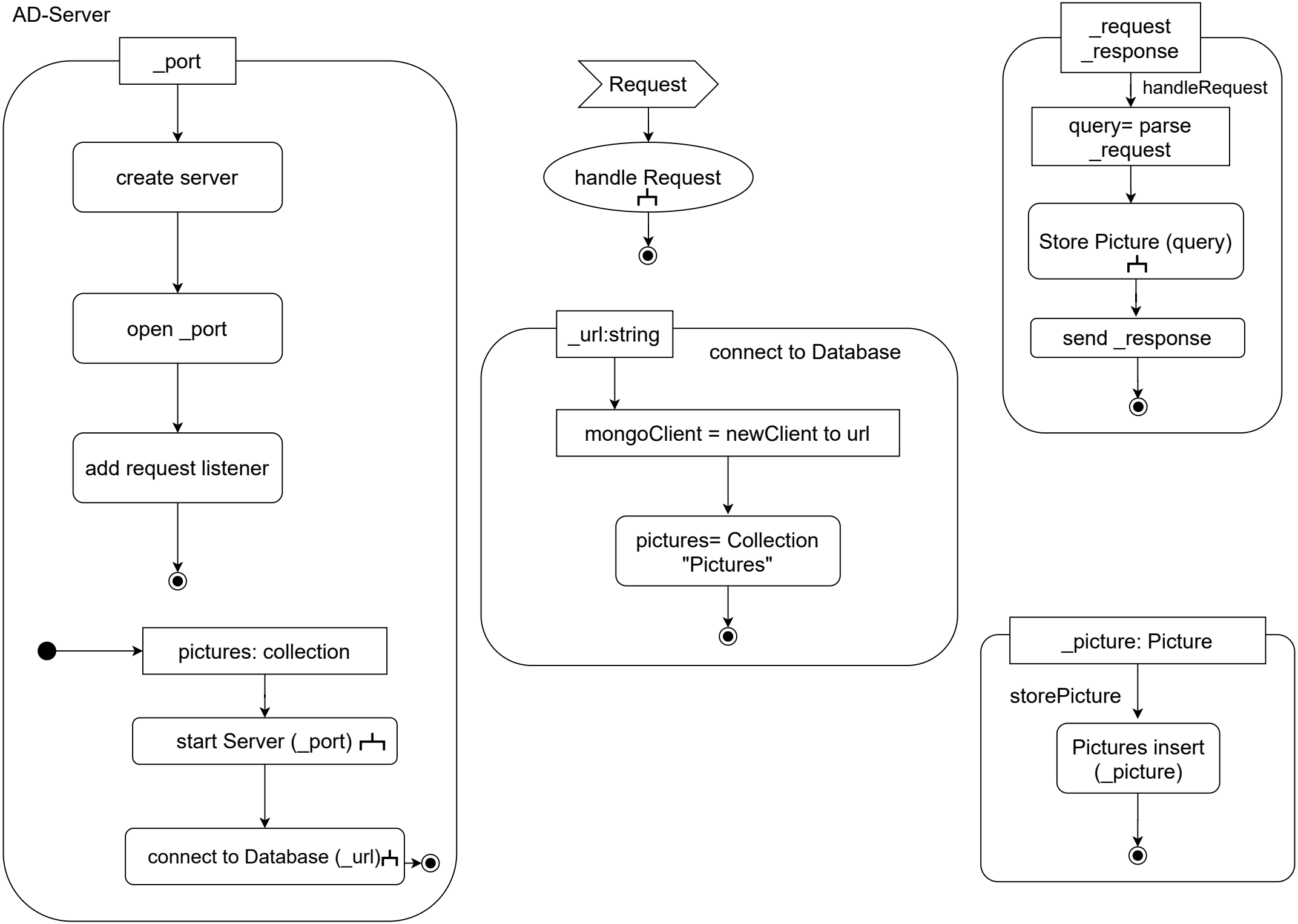

move
_timeslice: number
add velocity *_timeslice to position
[position component < 0]
[position component > canvas dimension]
add canvas dimension to component
subtract canvas-dimension from component


rotate
crc2.translate to position
factor = 100 / (rotation*10)
crc2.rotate(Math. PI / factor)
crc2.resetTransform()


titles: string []
_response
async restore picture
send request
wait for _response
[if value in MongoDB]
get _response
unwrap _respnse and
push titles
display titles
display titles
restore = HTMLDivElement
get restore
[i >titleslength]
append child
i++
insert [i]_titles
install listener


click restore title
get Data


async getData
_event
_response
get title of target
send request
wait for _response
get _response
unwrap _response and
push to figures
get _response


AD-Server
_port
create server
open _port
add request listener
pictures: collection
start Server (_port)
connect to Database (_url)
Request
handle Request
_url:string
connect to Database
mongoClient = newClient to url
pictures= Collection "Pictures"
_request _response
handleRequest
query= parse _request
Store Picture (query)
send _response
_picture: Picture
storePicture
Pictures insert (_picture)