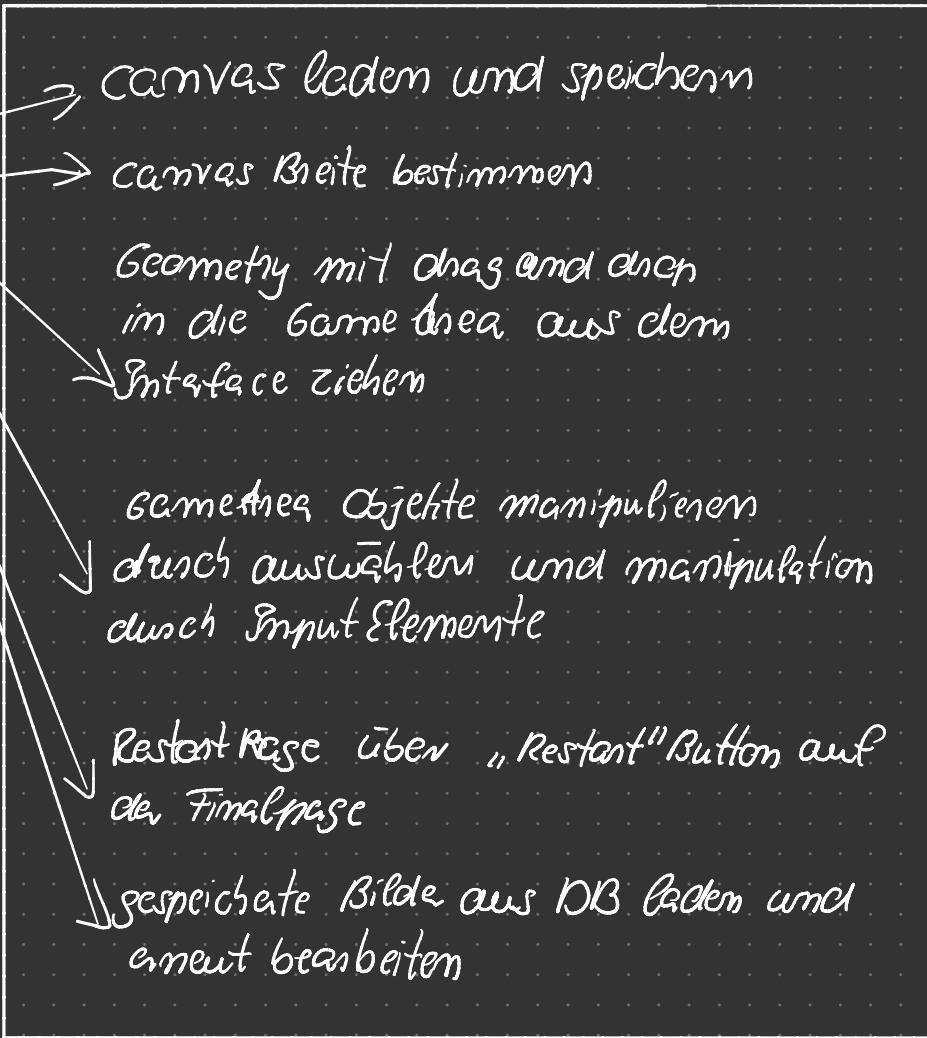
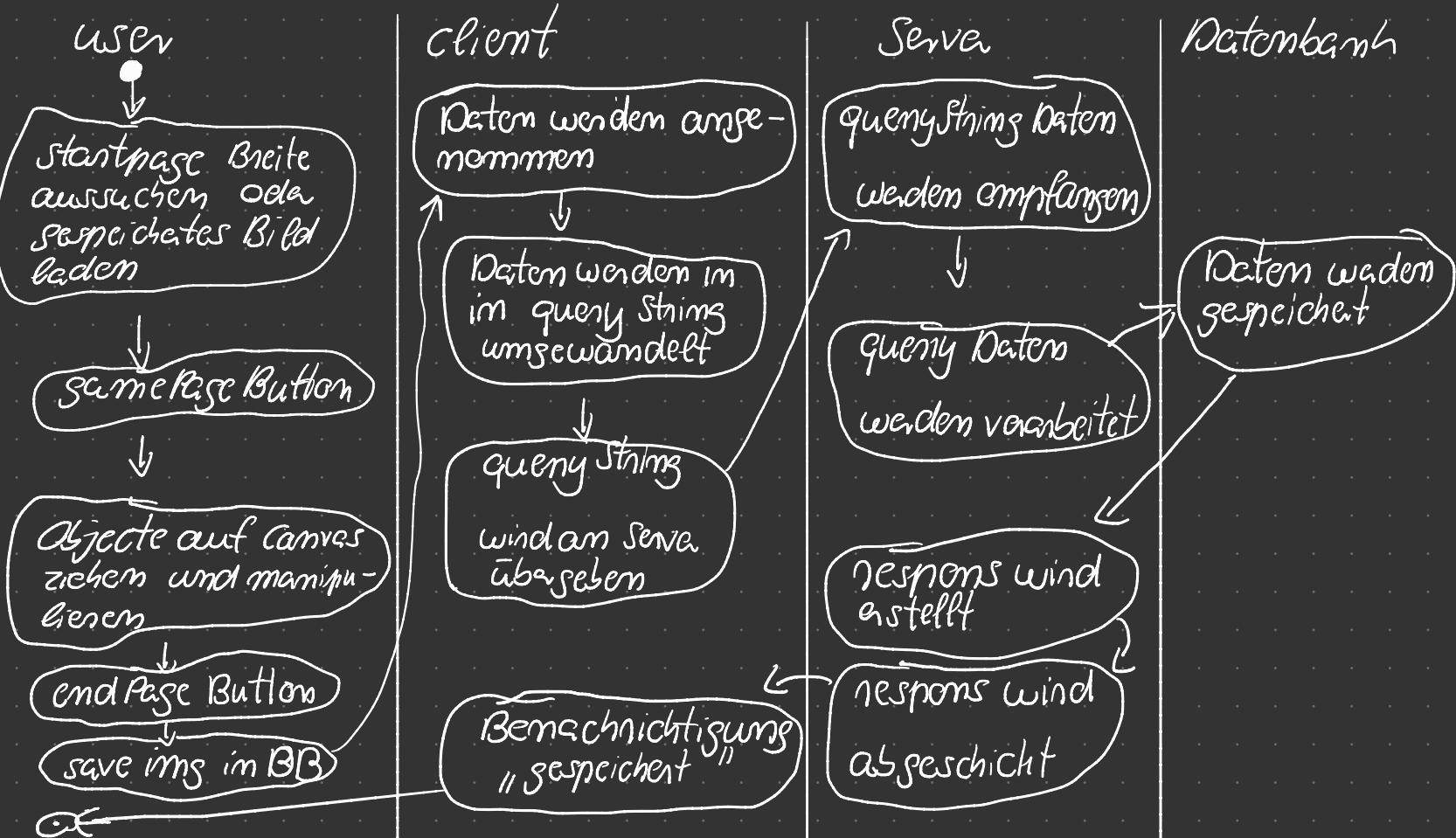


# Use Case Diagramm:

Benutzen



# Dom übergeleitendes Aktivitätsdiagramm



## Start Page

wähle ein gespeichertes Bild

- Bild 1
- Bild 2
- Bild 3

oder die Breite der neuen Leinwand mit dem Slider

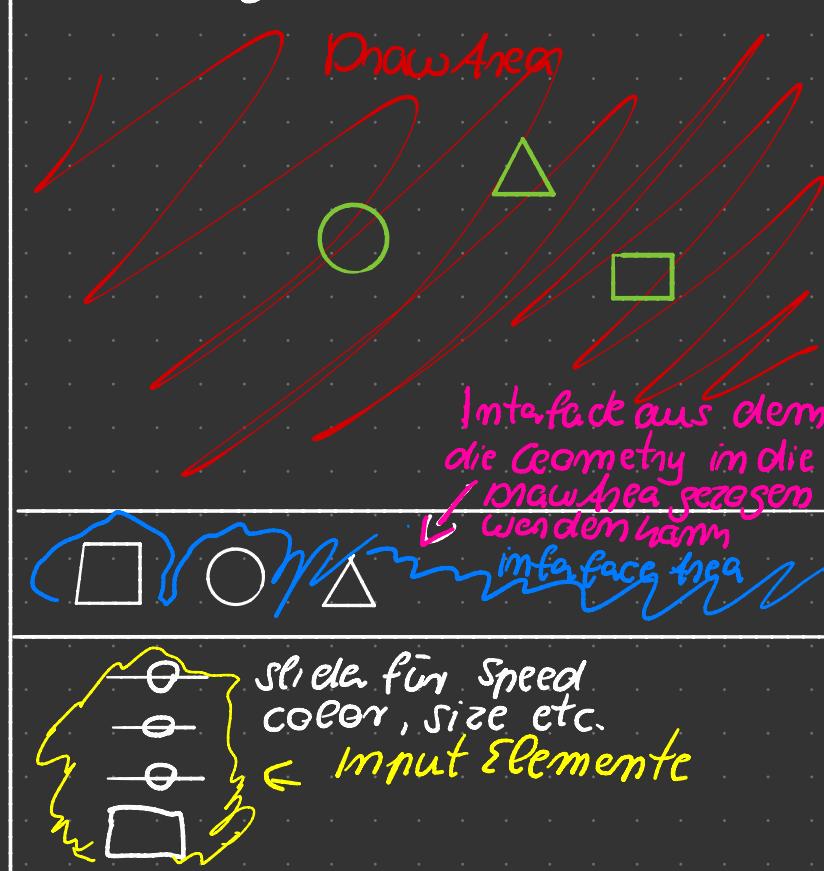
Breite:



Button To Game page

Speichert die Auswahl des Users und startet GamePage

# Game Page



Geometry kann ausgewählt und durch Input Elemente verändert werden

Button zum Final Page  
↓

end game

Breite variabel min 500px für Interface

# Final Page

Häufig benötigten Daten eim. Textfelder für DB Infos

name:

Datum:

ruft GamePage auf  
↓ ⇒ Neustart

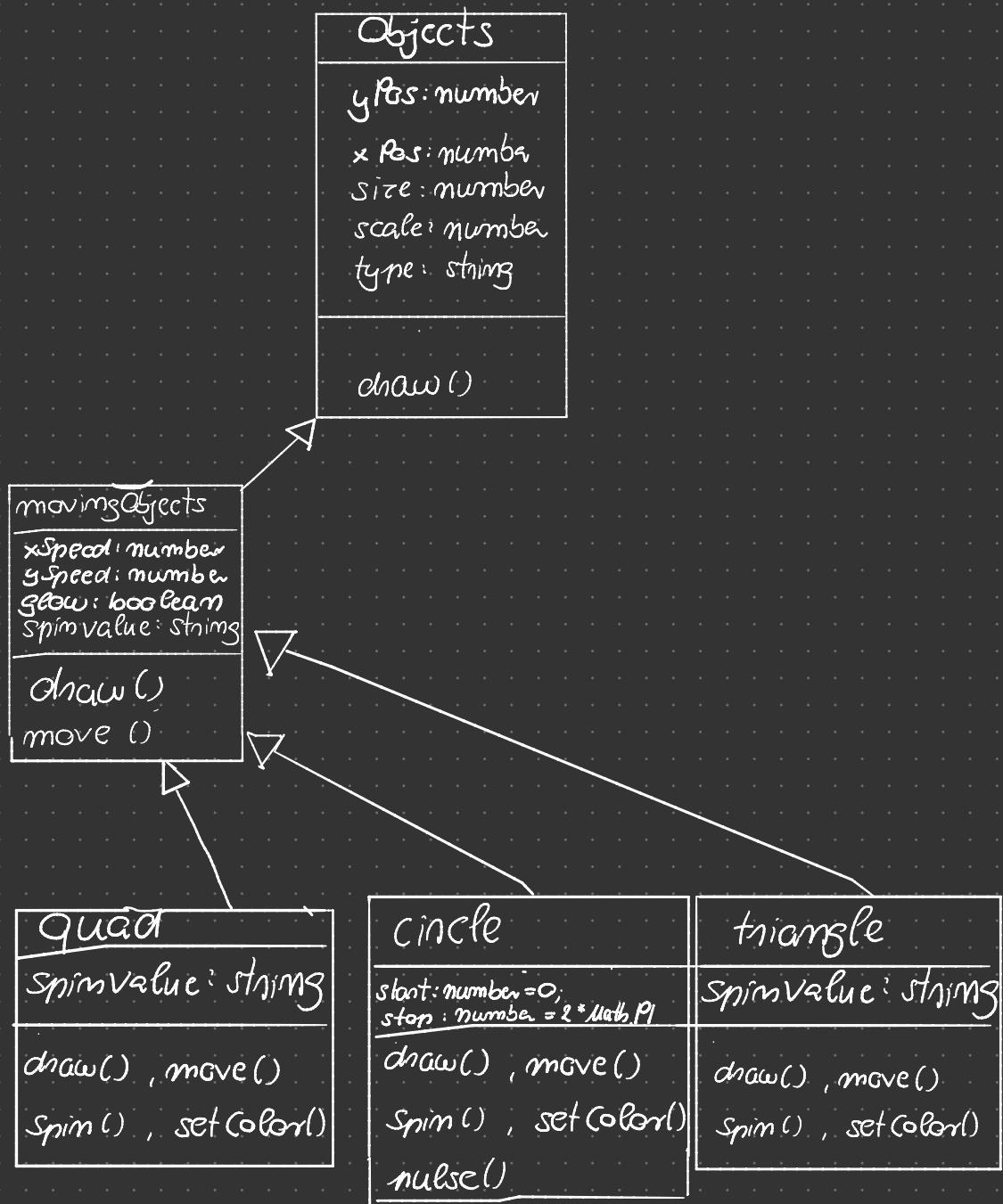
Absenden Button



Restart Button

Blieddaten werden  
im DB gespeichert

# Klassen diagram



```
export let crc: CanvasRenderingContext2D
```

```
export let canvas: HTMLCanvasElement
```

```
let imgData
```

```
export let interface: HTMLCanvasElement
```

```
export let canvasWidth: number
```

```
export let canvasHeight: number
```

```
export let quads: Quad[]
```

```
/* */ let circles: Circle[]
```

```
/* */ let triangles: Triangle[]
```

```
/* */ let movingObjects: MovingObject[] let scale: "
```

```
/* */ let objects: Object[]
```

```
/* */ let interfaceObjects: MovingObject[] let color: "
```

```
/* */ let spin: "
```

```
let StartPage: HTMLElement
```

```
let gamePage: "
```

```
let finalPage: "
```

```
let StartButton: HTMLButtonElement
```

```
let finalButton: "
```

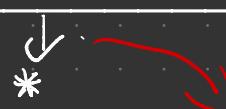
```
let sliderXSpeed: HTMLInputElement
```

```
let sliderYSpeed: "
```

```
let scale: "
```

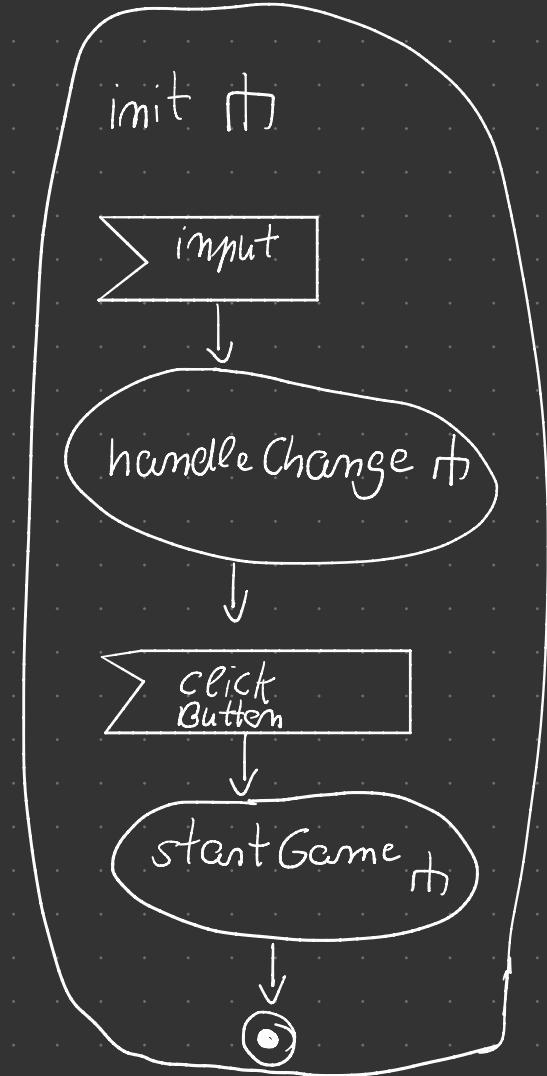
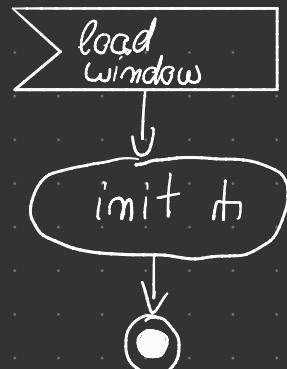
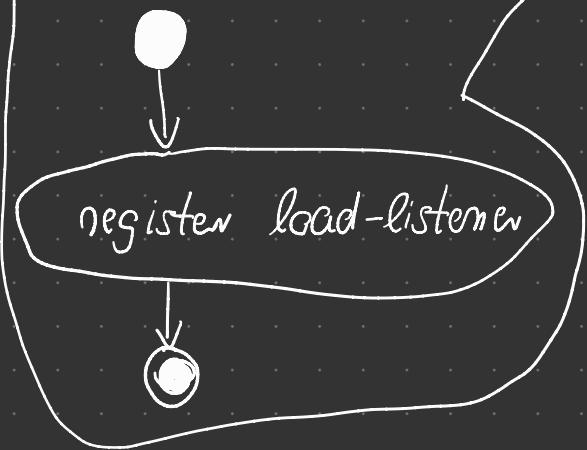
```
let color: "
```

```
let spin: "
```



Main.ts

← \*



start game

display Game Page

CV Elemente erstellen

mouse down

[mousedown trigger]

mouseDownCheck it

mouse over

[if mouseover trigger]

SelectArea it

imgData = crc.getImageData

setInterval( update it, 20)



update()

Clear Rect

putImgsData

$i < \text{Object.length}$

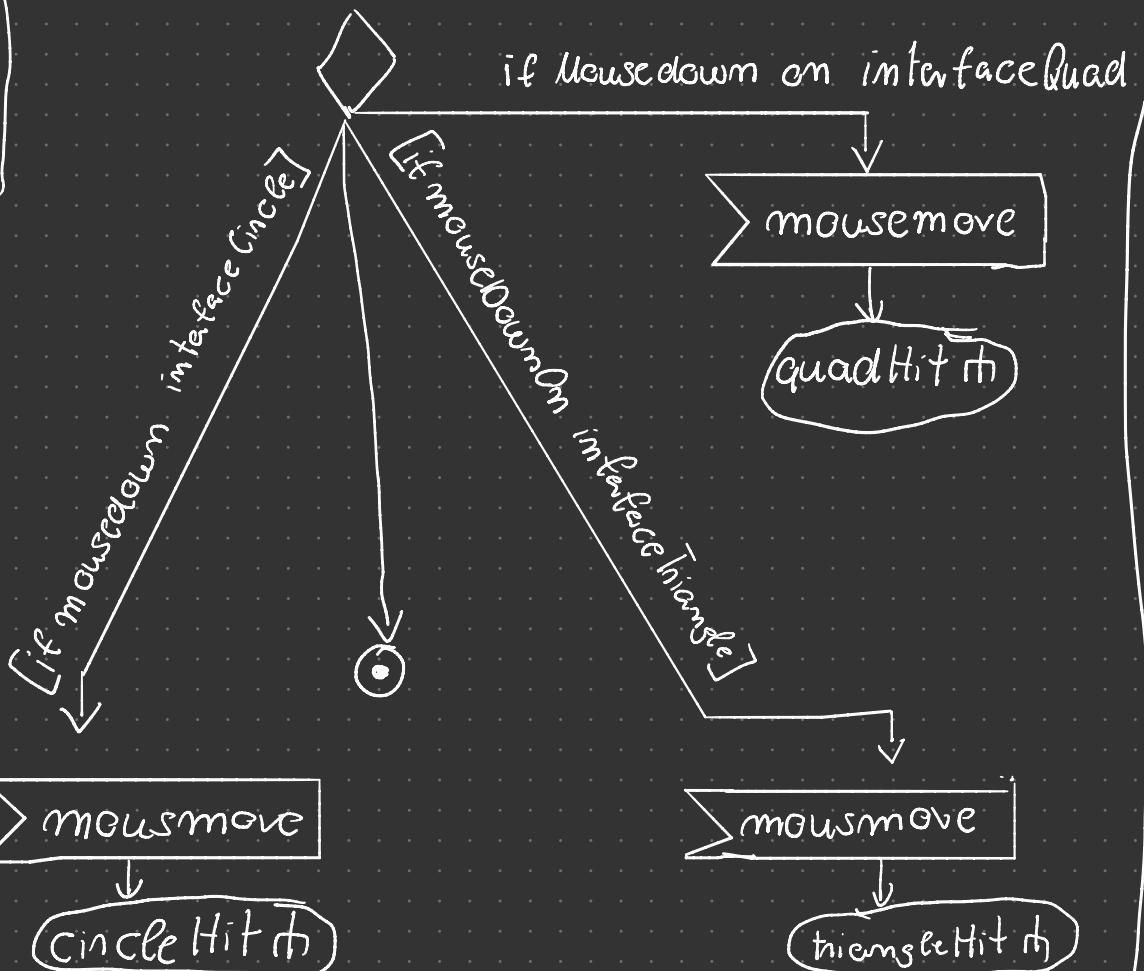
$c_i.x = x$

`movingObjects[i].draw()`  
`movingObjects[i].move()`

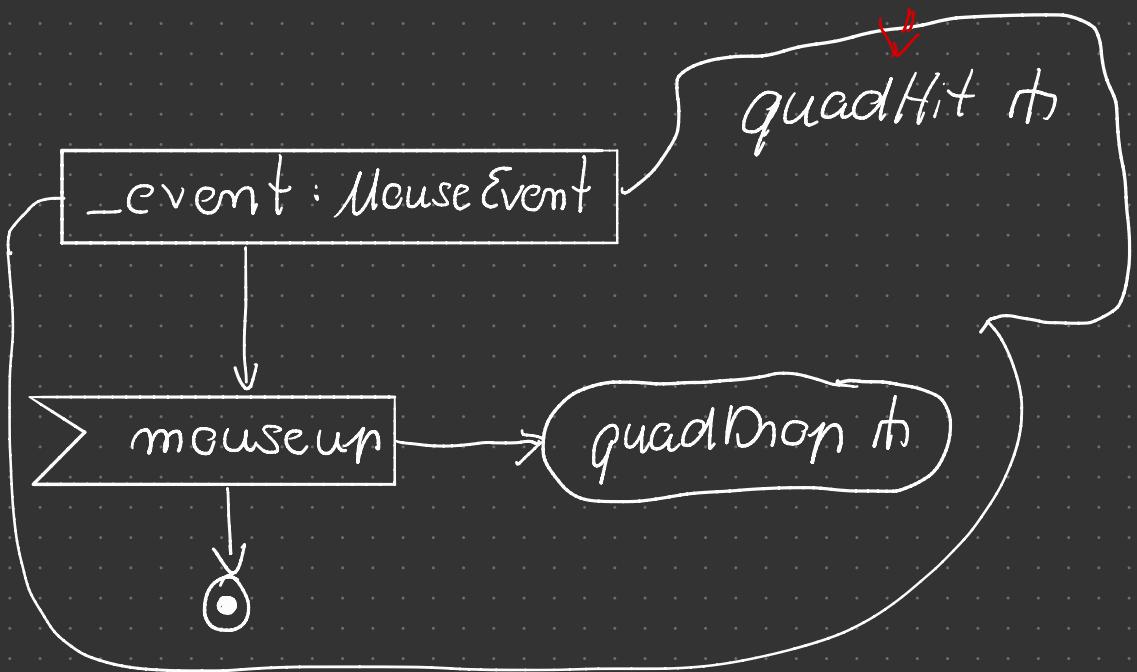


- event: MouseEvent

mousedownCheck in



Funktion gibt es auch  
für Triangle und circle  
gleiches Prinzip



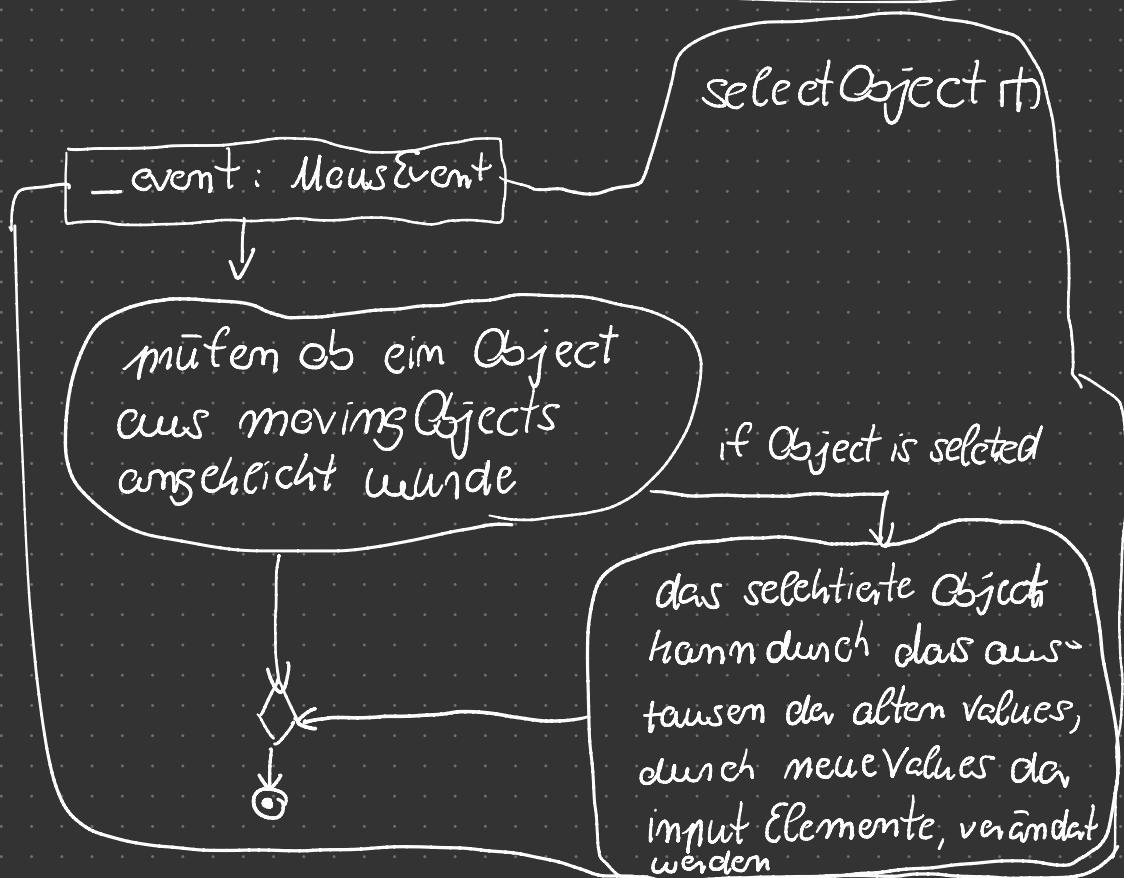
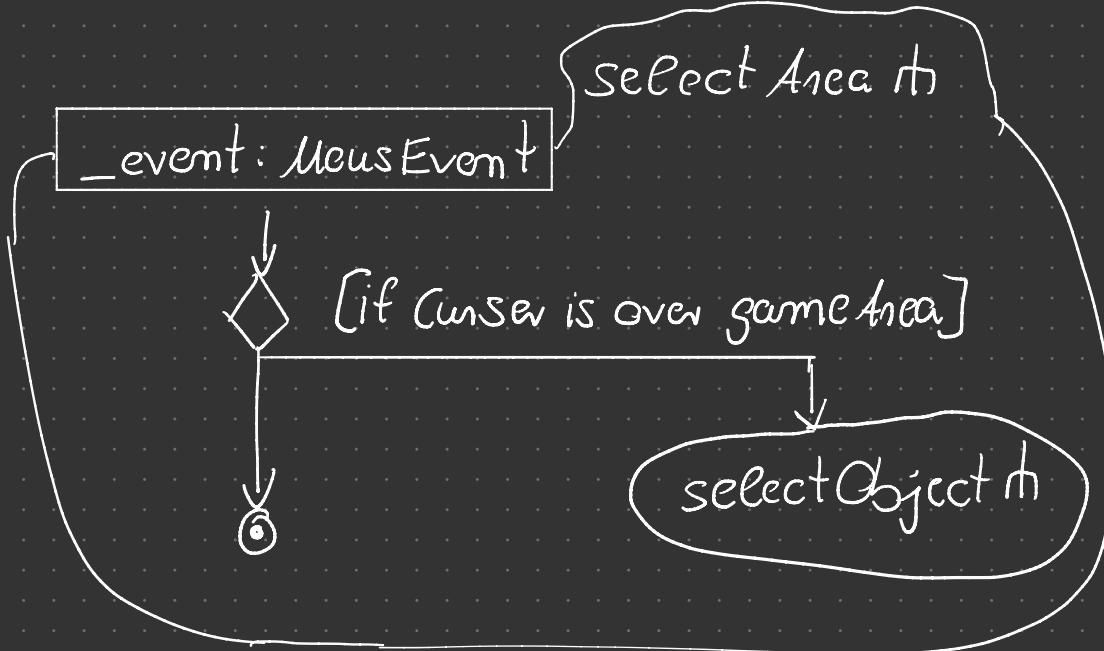
quadDrop it

Funktion gibt es auch  
für Triangle und circle  
gleiches Prinzip

- event : Mouse Event

let newQuad: new Quad

movingObjects.push(newQuad)



# Server.ts

```
import ure from "url"  
import Http from "Http"  
import Mongo from "mongoDb"  
let savedImg: Mongocollection  
let Database: string
```

startServer(  
 -port



createServer

open-port

request



handleRequest(  
 -request  
 -response

-request  
-response

query = parse\_request



split url in Anays



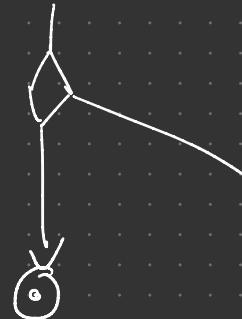
DOM content loaded

get/ImagesFromServer()

handle Change GetImages()

Images im RadioButtons anzeigen

[if RadioButton is picked]



Bildinfo aus DB laden,  
in Anzug umwandeln  
und anzeigen

