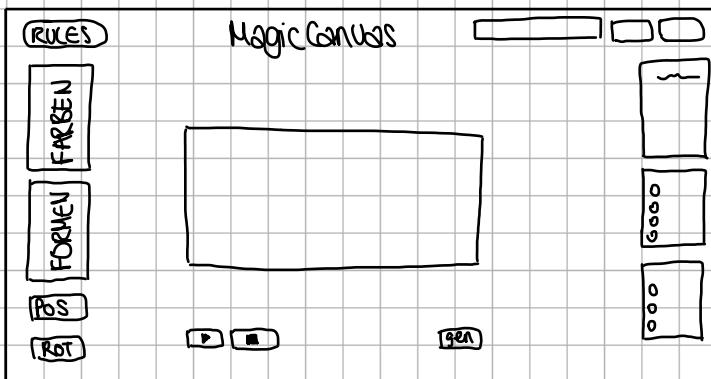


## Magic Canvas: Funktionale Analyse

### Plattform PC

- × Mehr Platz für verschiedene Auswahlmöglichkeiten
- × bessere Übersichtlichkeit
- × bessere Testbarkeit
- × Elemente können stärker in der Größe variieren
- × gezielte Auswahl (durch Hover möglich)

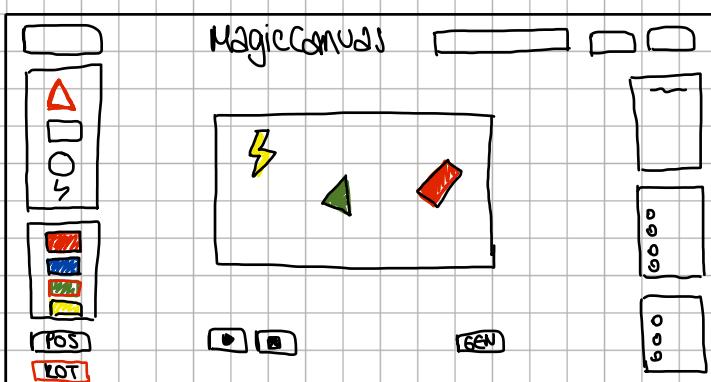
### Interaktionsverlauf



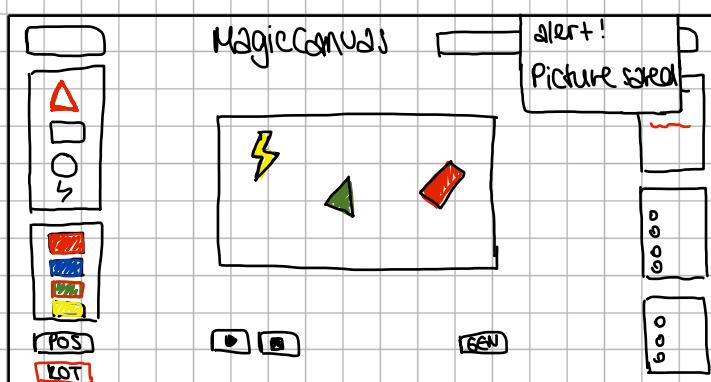
1. Home



2. Rules - Overlay



3. Choose the symbols & place



4. Save picture

## Infos, die zum Nutzer gelangen

- Alert-Nachricht, wenn das gezeichnete Bild gespeichert wurde
- Auflistung der gespeicherten Bilder zum Auswählen
- farbiges Feedback, wenn ein Button ausgewählt wurde und wenn ein Symbol, Farbe und Animation geklickt wird

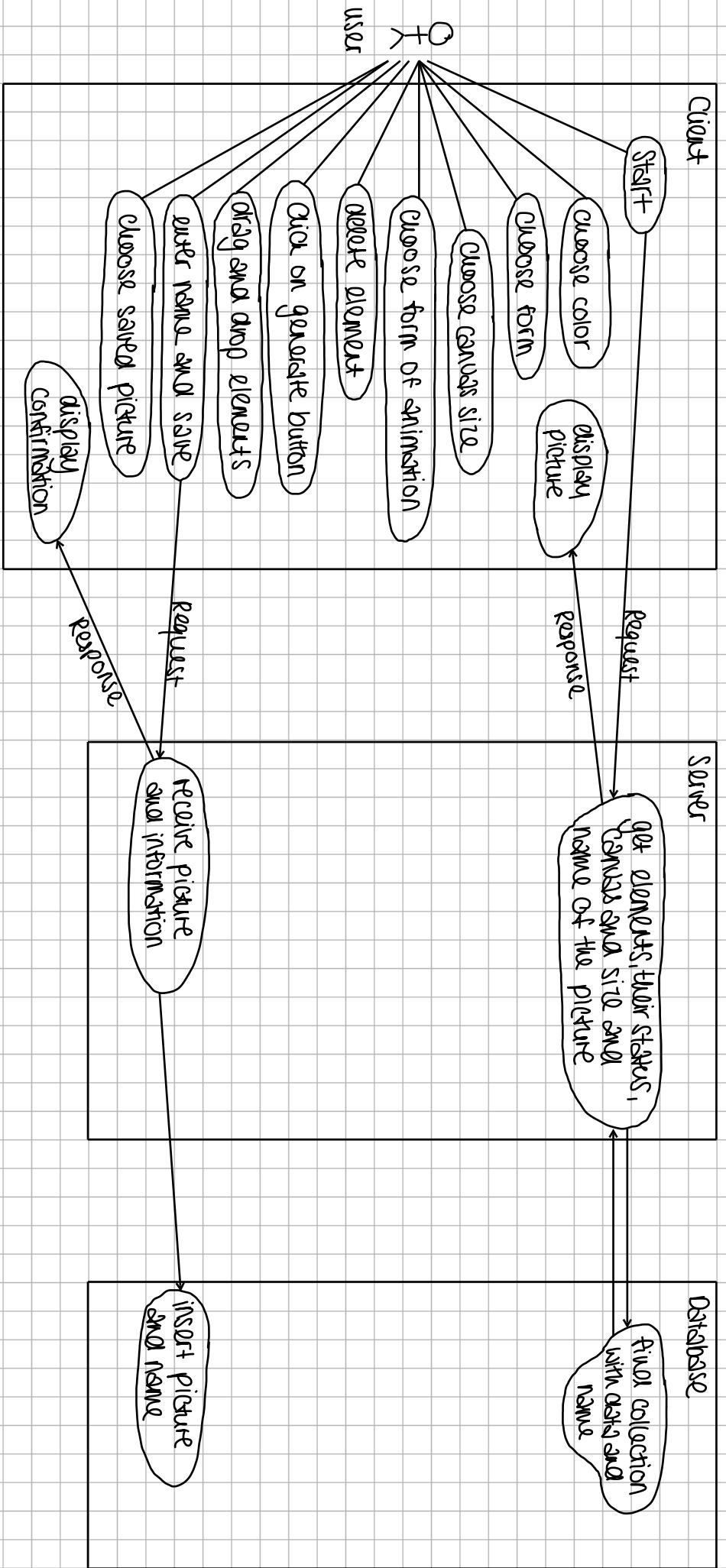
## Anleitung zur Interaktion mit der Anwendung (siehe rules-Button in der Anwendung)

- 1 Wähle Canvas Größe und Hintergrund aus
- 2 Wähle eine Farbe und ein Symbol und optional eine der beiden Animationsformen
- 3 drücke Generiere, damit das Element links oben auf dem Canvas angezeigt wird
- 4 drücke optional start, wenn eine Animationsform ausgewählt wurde und stop um sie zu stoppen
- 5 verschiebe das Element, indem du draufklickst und gedrückt gehalten es an die gewünschte Position ziehest
- 6 Du kannst deinen canvas mit einem Klick schieben
- 7 um das Bild zu speichern, gebe einen Namen an und drücke auf save
- 8 Wenn es erfolgreich gespeichert wurde, erscheint ein Alert und dein Bild wird in "Latest Pictures" aufgelistet

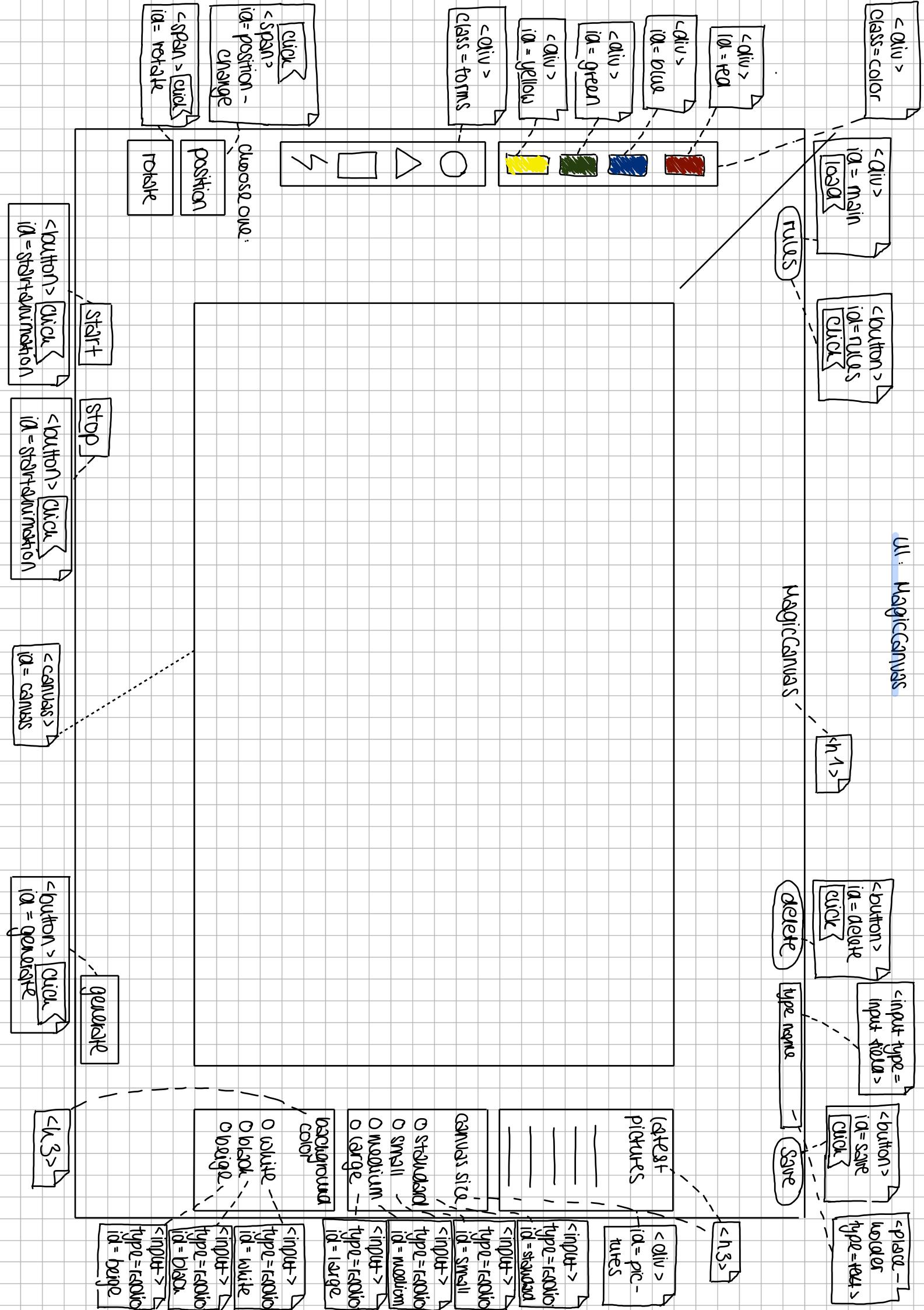
## Anleitung zur Installation der Anwendung

- 1 Pull Github Repository mit allen Dateien und öffne sie mit einem geeigneten Programm (VS Code).
- 2 lege dir einen Heroku Account an mit der primary language Node.js und lege eine App an.
- 3 Verbinde Heroku mit deinem Github Repository (mit den Dateien der App).
- 4 ersetze die appurl in der main.js mit deiner App URL von Heroku
- 5 deploy den master branch auf Heroku
- 6 lege einen eigenen Account bei MongoDB an und Wähle "Create a Cluster" in Europe an
- 7 lege einen Testuser an und gebe ihm Read / Write Zugriff
- 8 Wähle "Connect your application" und ersetze den Connection String mit dem in der server.js (DatabaseURL)
- 9 lege eine Datenbank an
- 10 Erstelle ein Bild in der Anwendung und speichere es unter einem eindeutigen Namen
- 11 Finde den Namen des Bildes auf der rechten Seite.

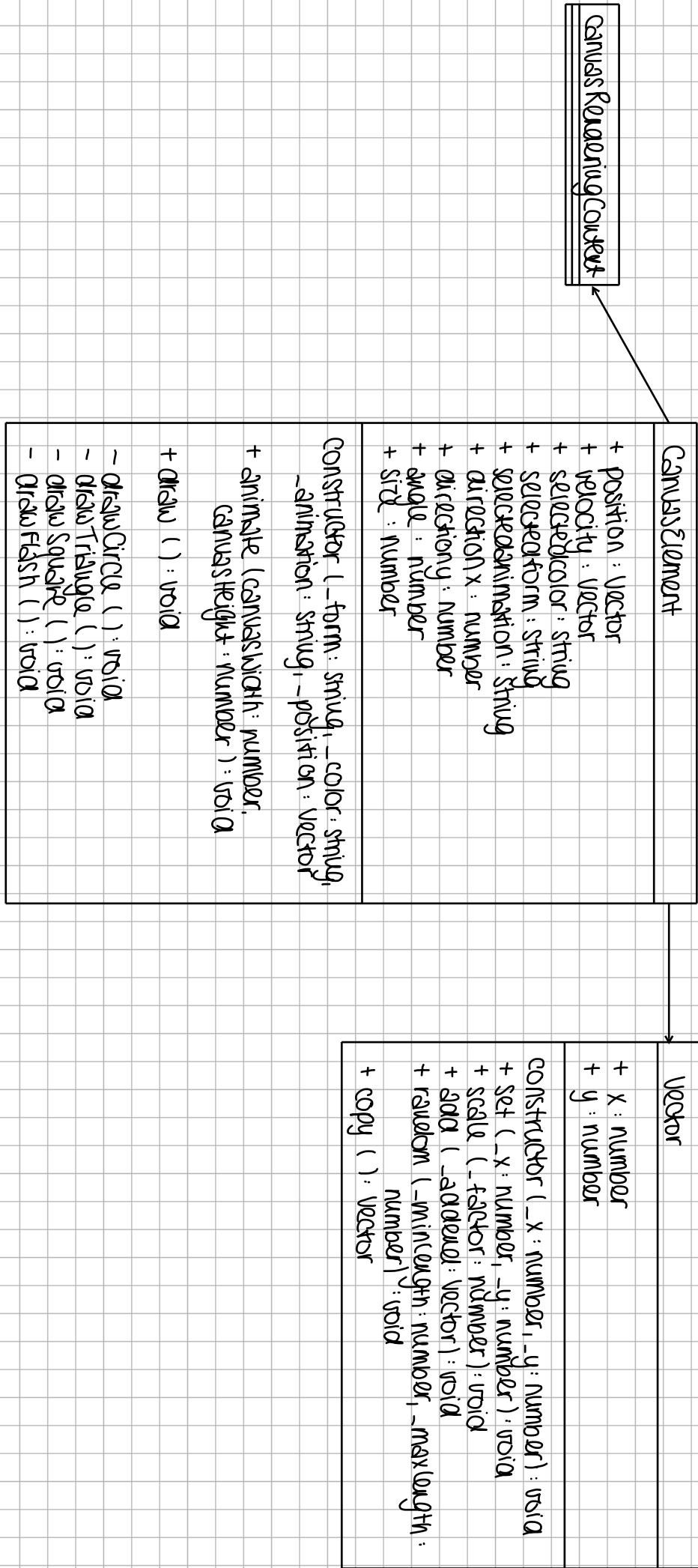
## Use-Case Diagram



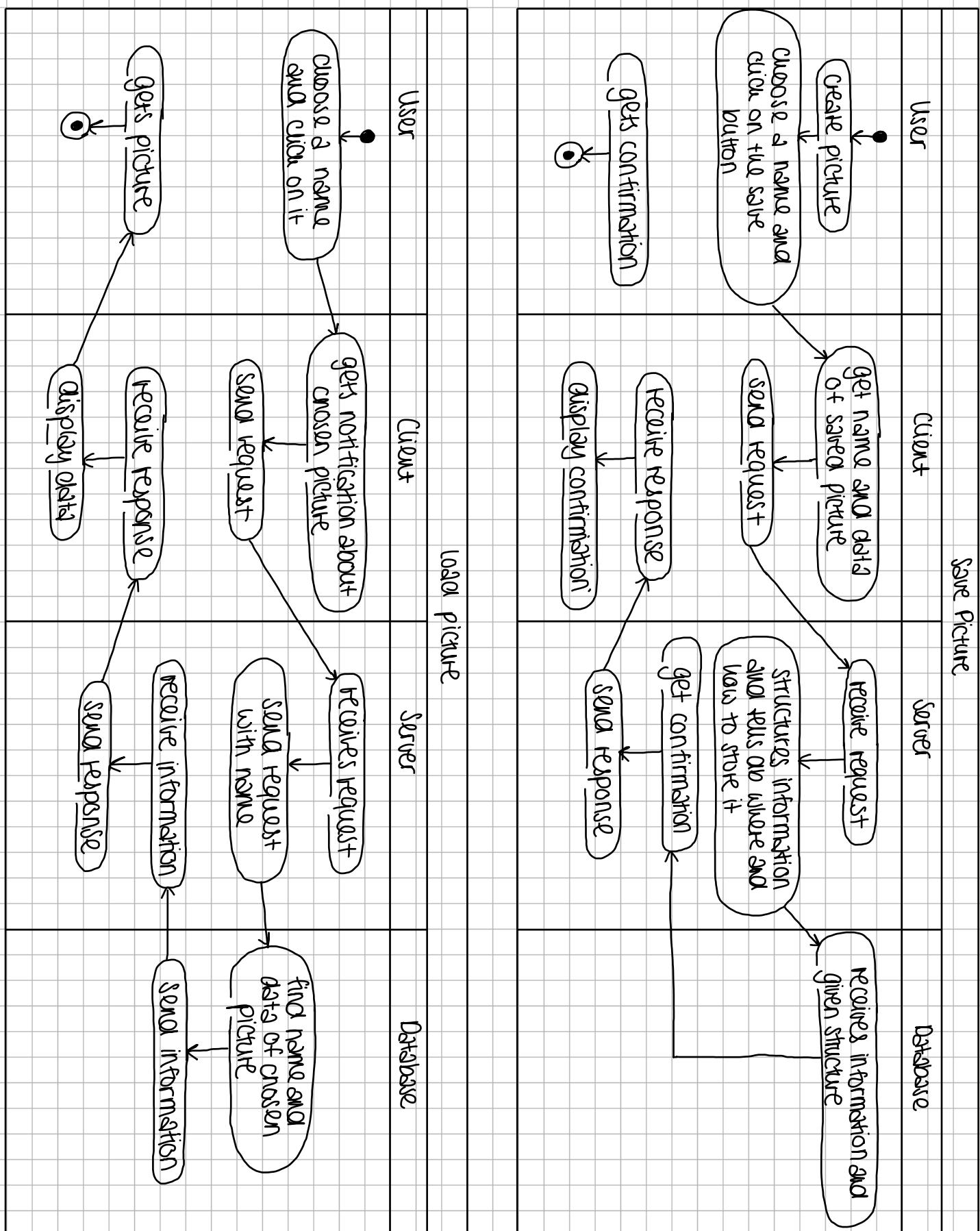
UI : MagicCanvas



## Magic canvas: Class diagram



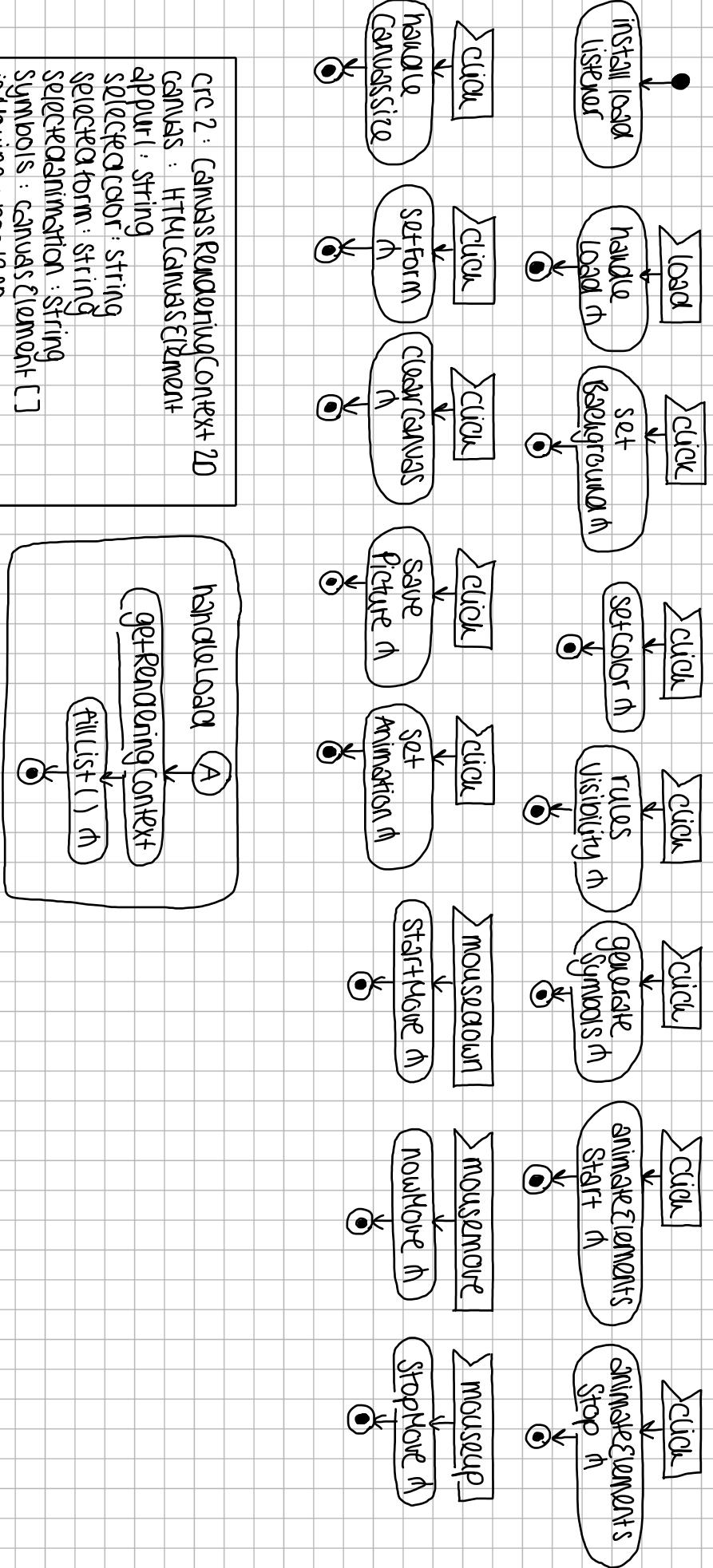
## MagicCanvas: swimming lanes

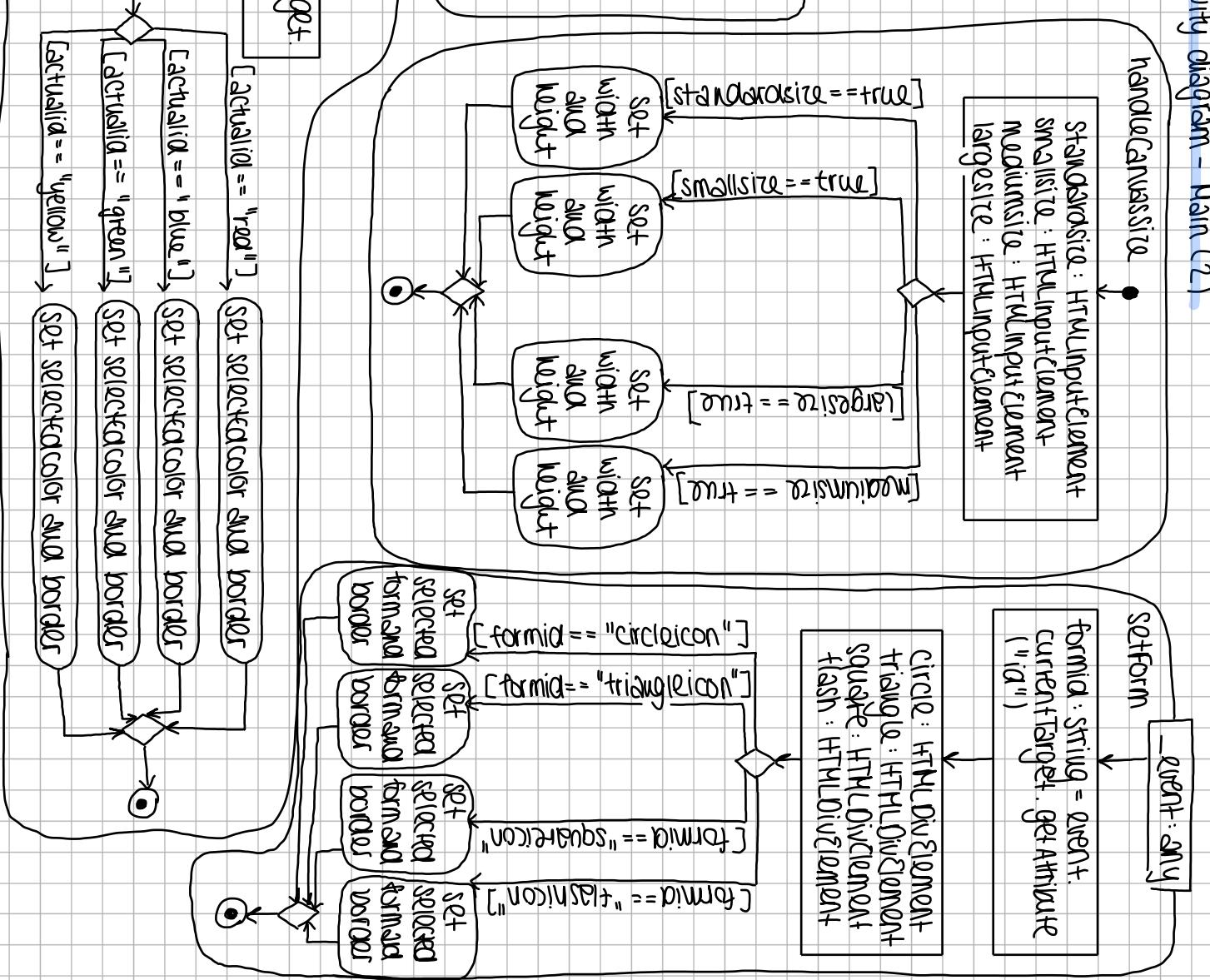
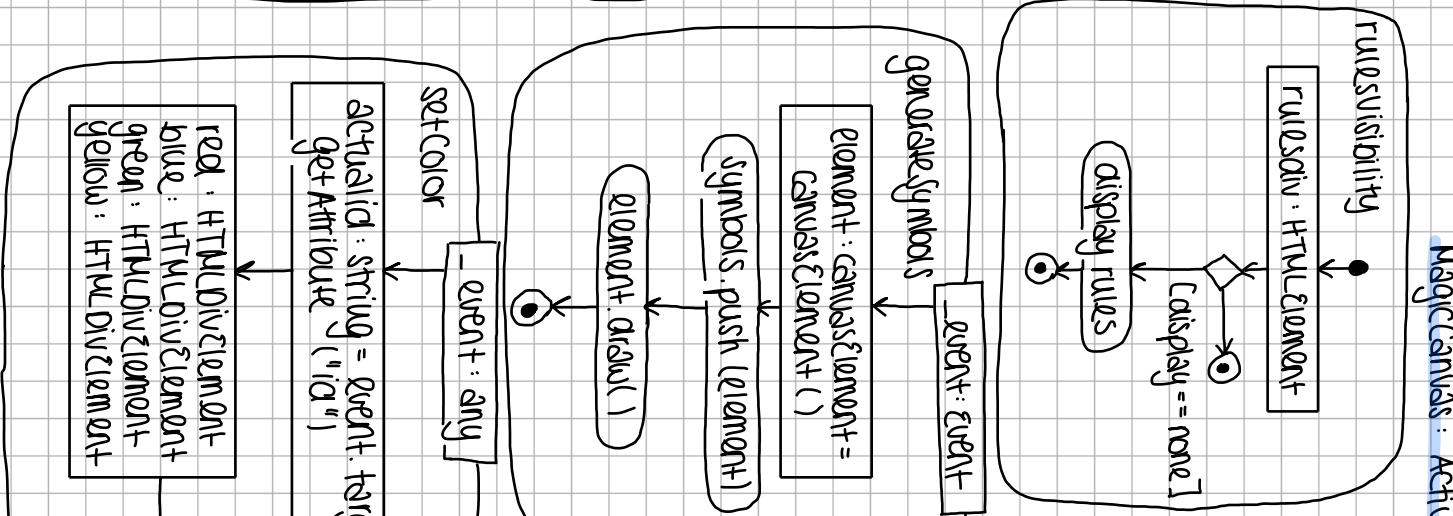
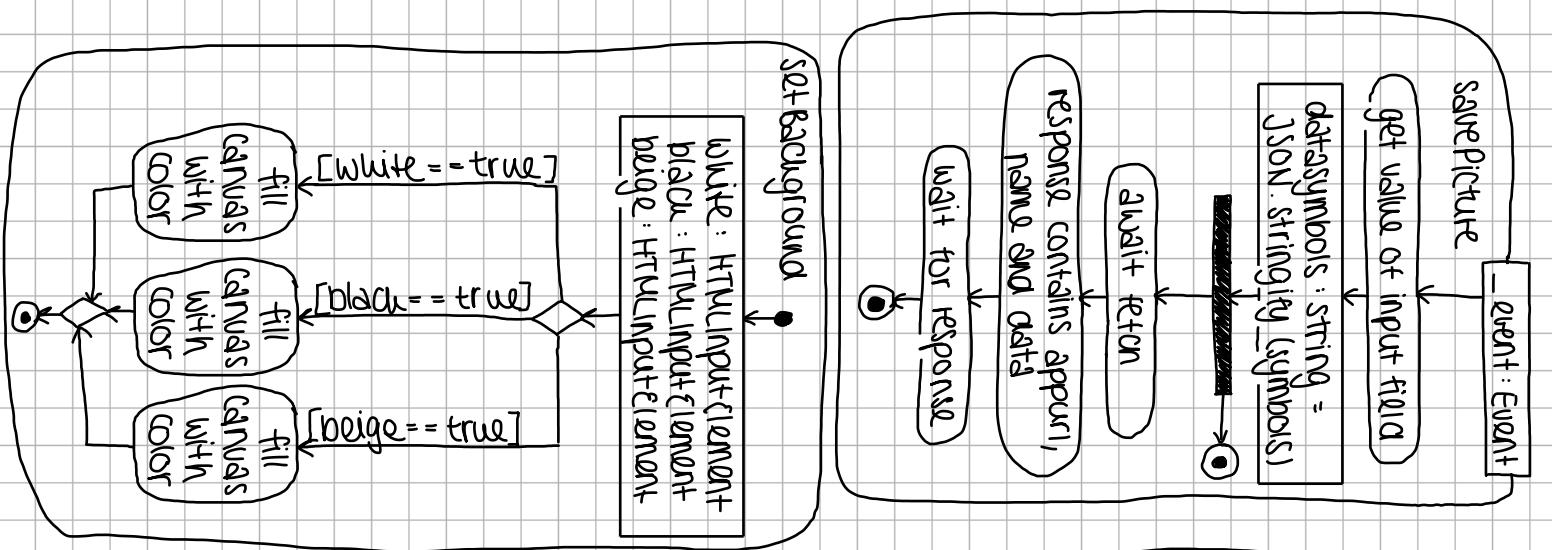


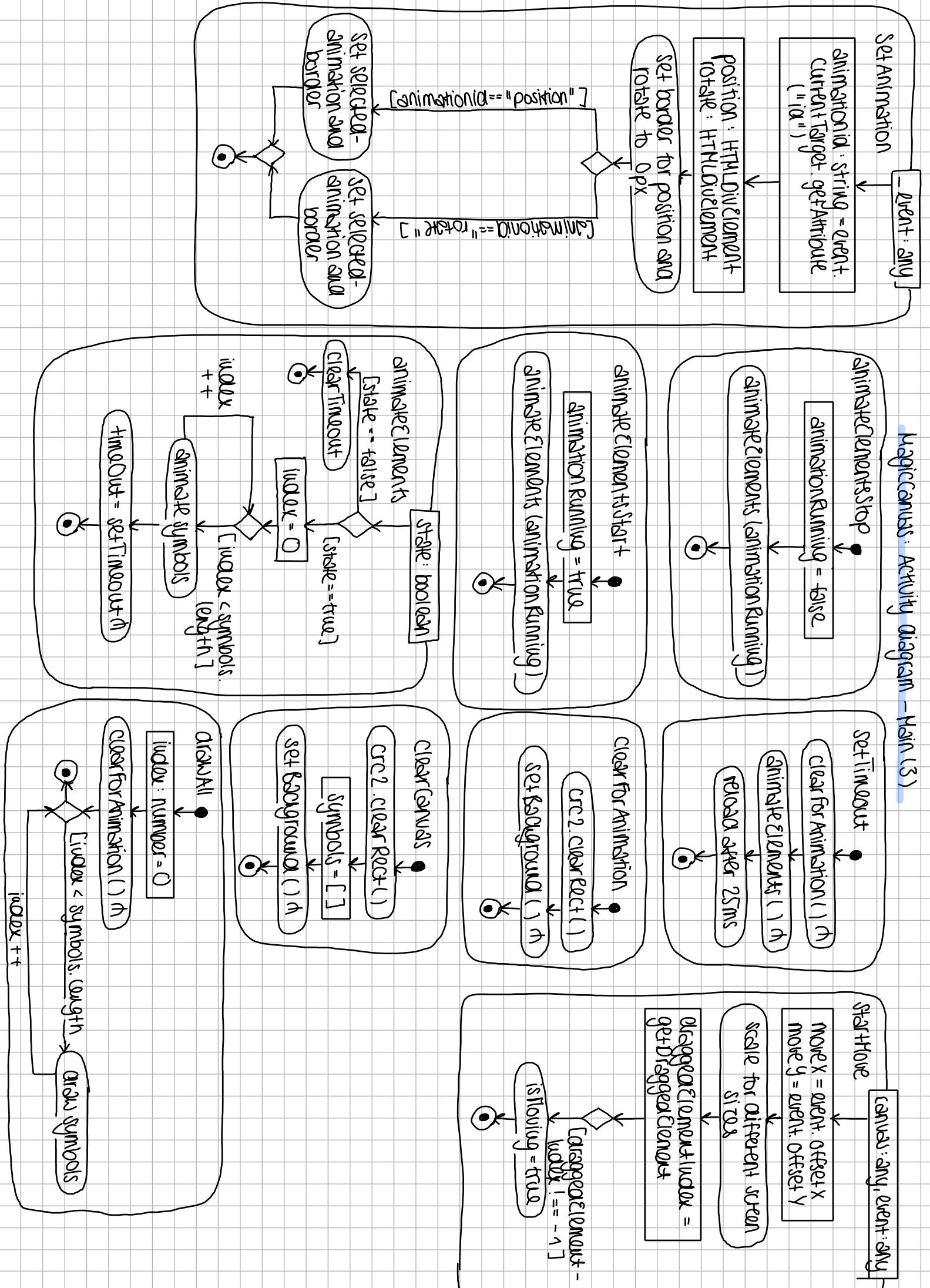
## Magic canvas: activity diagram - Main (1)

```

crc2 : CanvasRenderingContext2D
canvas : HTMLCanvasElement
appurl : string
selectedColor : string
selectAction : string
selectAnimation : string
symbols : canvasElement []
isShowing : boolean
moveX : number
moveY : number
draggedElementIndex : number
timeOut : any
xpos : number
ypos : number
index : number
  
```







## MagicCanvas: Activity diagram - Main (4)

nowMore [canvas: any, event: any]

moreX = offsetX  
moreY = offsetY

[isDrawing == true]

index++  
index: number = 0  
foundIndex: number = -1

[draggedElementOut  
walk: i == -1]

set position of element to  
mouse position

[drawWall () ▲]

○

stopMore [canvas: any, event: any]

[isDrawing == true]

set position of elementOut  
to mouse position

[draggedElementOut  
i == -1]

[draw symbols]

moreX = 0  
moreY = 0  
isMoving = false

○

getDraggedElement

moveX: number, moveY: number

[index < symbols.length]

return foundIndex  
index < symbols.length

[check if mouse is on  
element]

[foundIndex = index] → ○

fillList

○

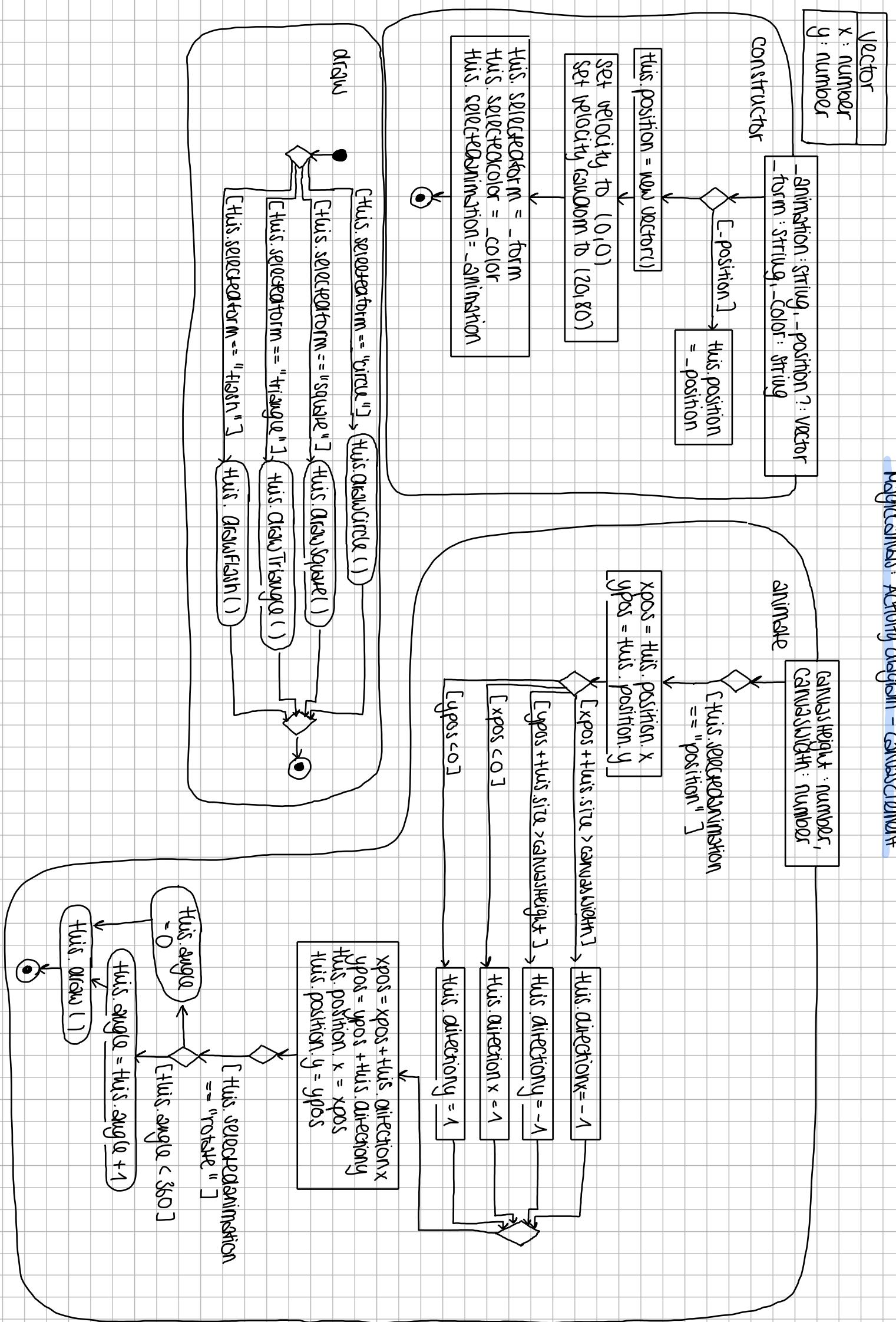
[await getItch]

[response contains appri  
and action = select]

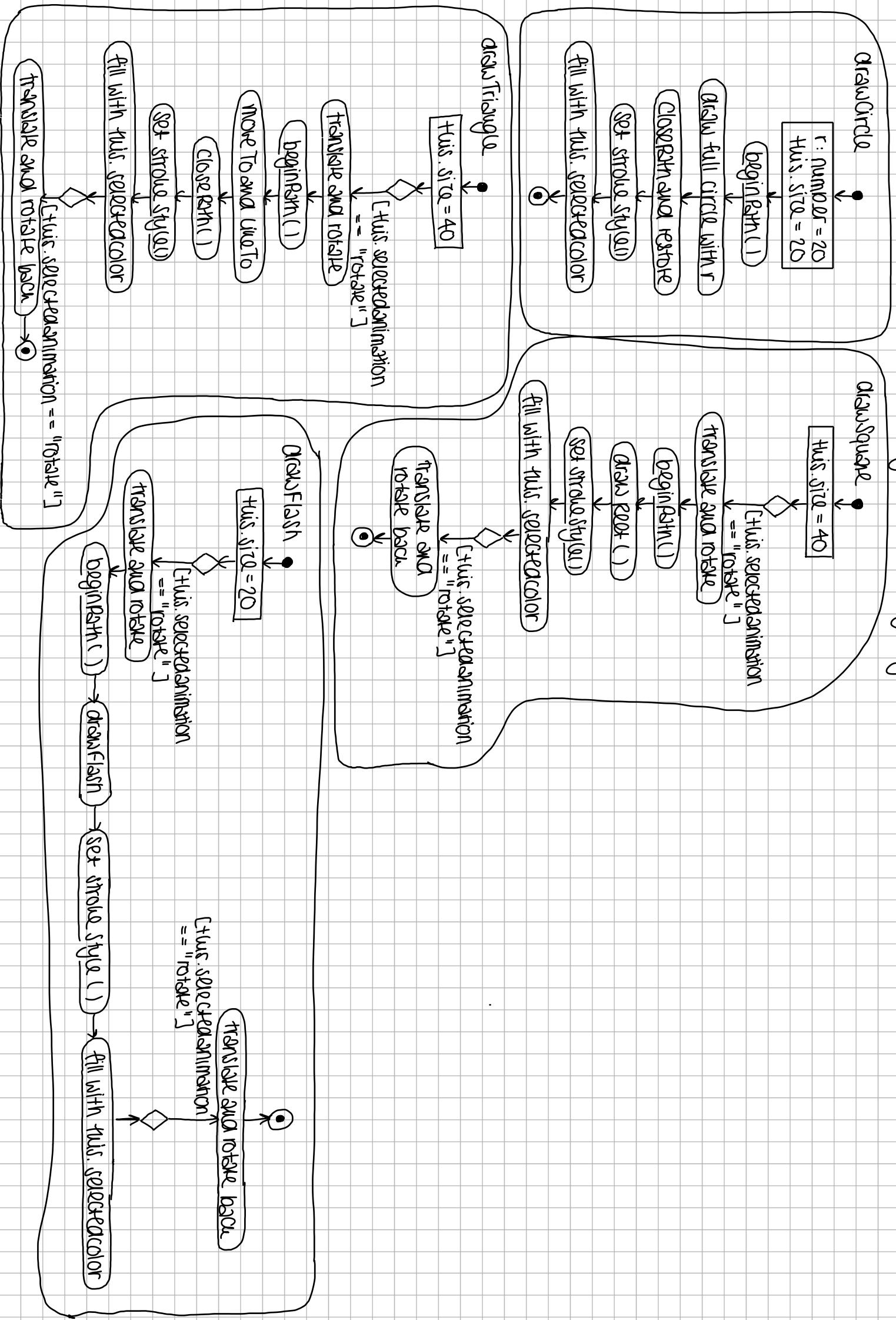
[wait for response]

○

## MagicDraw: Activity diagram - canvasElement



## MagicCanvas: Activity diagram – canvas comment (2)



## Magic canvas: Activity diagram – Server (1)

