Rocket Jam

This is the construction manual for a browser game called Rocket Jam. Rocket Jam allows users to have an online experience of what they have been missing out in real life.
A new years Firework. This construction manual serves as a user guide and building instructions for developers of all levels.

Author
Rouven Sahertian
Matriculation number 259665
Student at Hochschule Furtwangen University

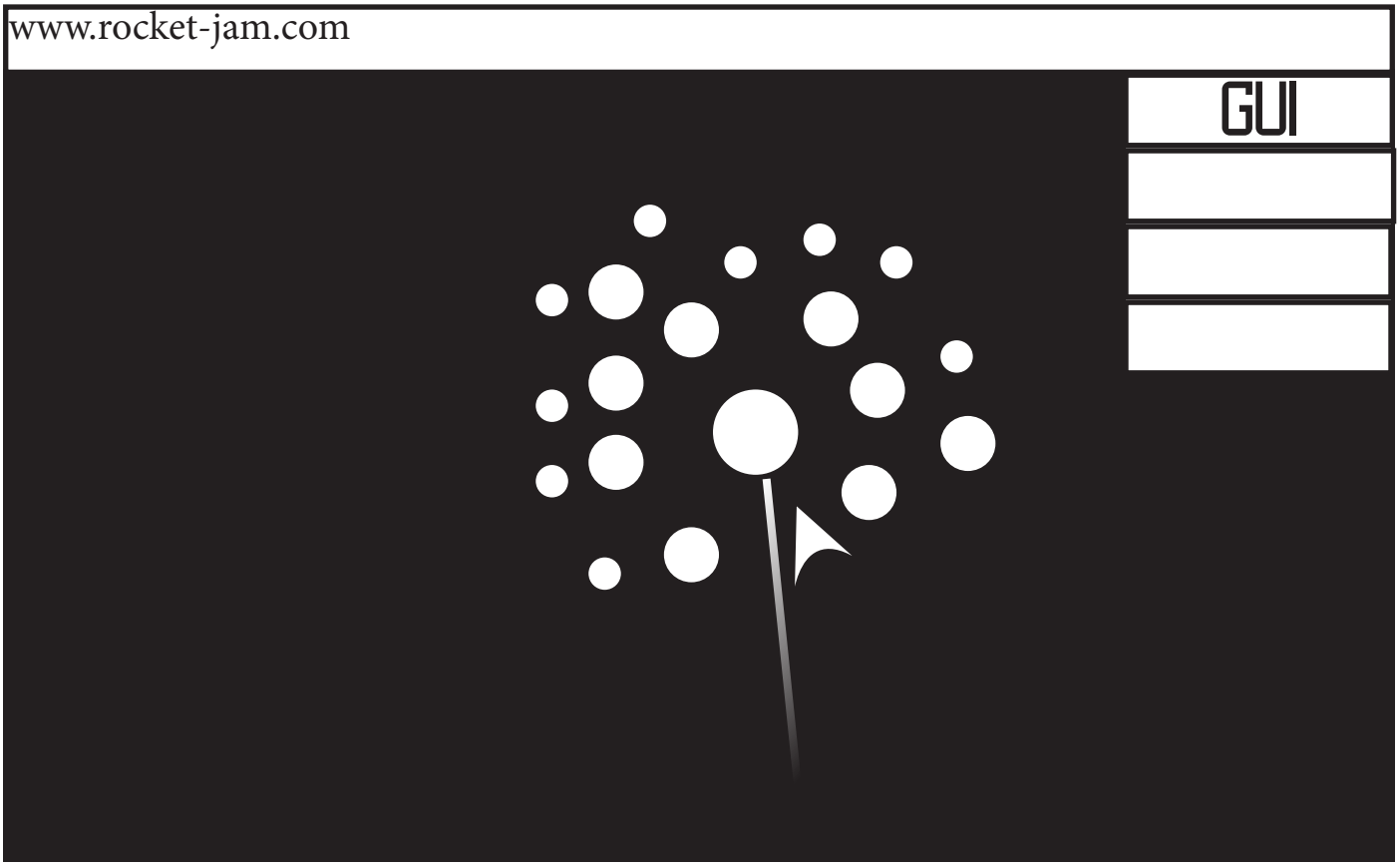
The Idea

is to create an application that allows users to modify they´re own rocket to create a digital firework. Users can choose between different options like colors, shapes, explosion size and radius as well as the amount of particles created when they´re rockets explode. All this works by combining HTML, CSS and Typescript / Javascript.

Device of choice

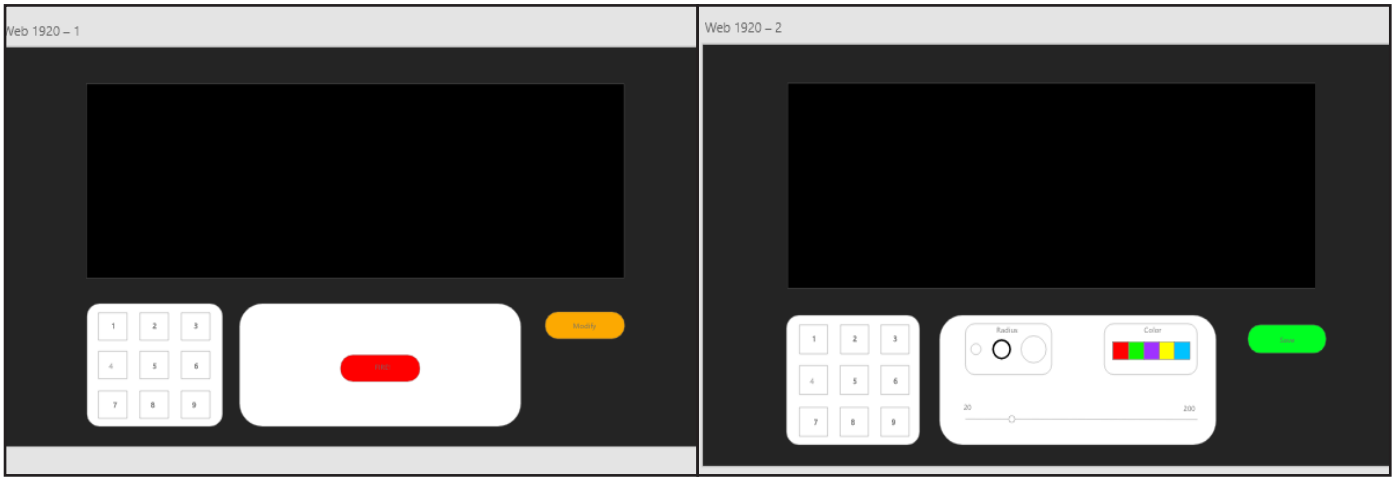
Rocket Jam is primarily designed to be used on a PC or Laptop for the sake of usability. The comfort of a big screen, a mouse in addition of a more likely stable internet connection surpass the usual advantages of a mobile device such as smartphones.. A firework can be an intense experience in real life,, using the full width of a Laptop screen, monitor or TV gives more space for visual effects to recreate the feeling of a real firework. Never the less, Rocket Jam works just as good on mobile devices as on bigger machines and can easily be optimised by developers on all levels.

User experience design



user interface

The first design was too complex, a canvas where the magic happens, a user interface seperated into different divs and too much space for logos etc. . Pictures are seen below.
A few steps later I figured out that using the full screen of any device used would be more useful, so now at any given time the canvas will use the full screen height and width. I then also came to the conclusion that the user interface should also be more compact and integrated into the website. This also saves the work of seperating the application into different pages.
The final design is seen above.



Although Rocket Jam is a just for fun application, its usage can be very diverse. The idea is to give users options which in combination with code will create flutter generative artworks.

The application is pretty much intuivtive but the results of the users customised rockets do need the right combination of options to create crasy and colorful explosions. So the results vary depending of the users combination of modifications.

Interaction guide

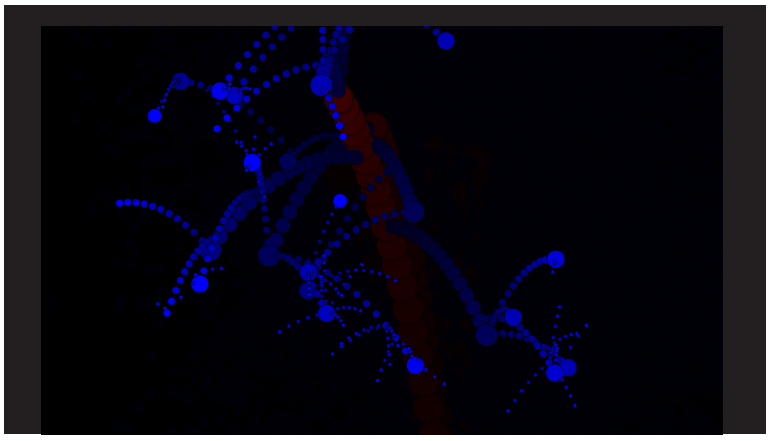
Rocket Jam

1. First off, the user has to load a given preset or create an individual rocket preset by using the GUI.

2. The preset has to be saved and then loaded before the rockets are being fired. If no preset is loaded the rocket fired is set by default and if another preset is loaded before the new preset is saved it will be erased..

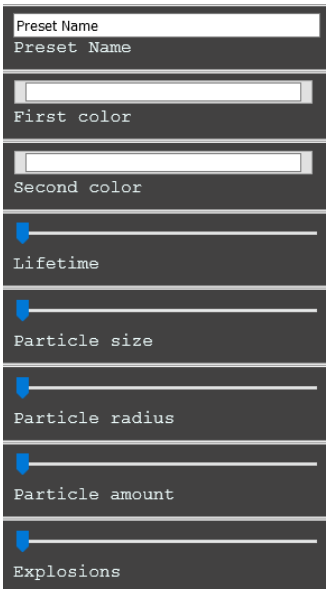
3. By clicking on the black canvas rockets are spawned.

It is recommended to shoot 1 or 2 rockets first to see how the creation comes out, if too many rockets are fired at the same time it might get a little bit messy.



CANVAS AND PARTICLES

Presetselector, save button, load button



GUI (Graphical User Inteface)

Installation guide

Rocket Jam, Heroku, MongoDB

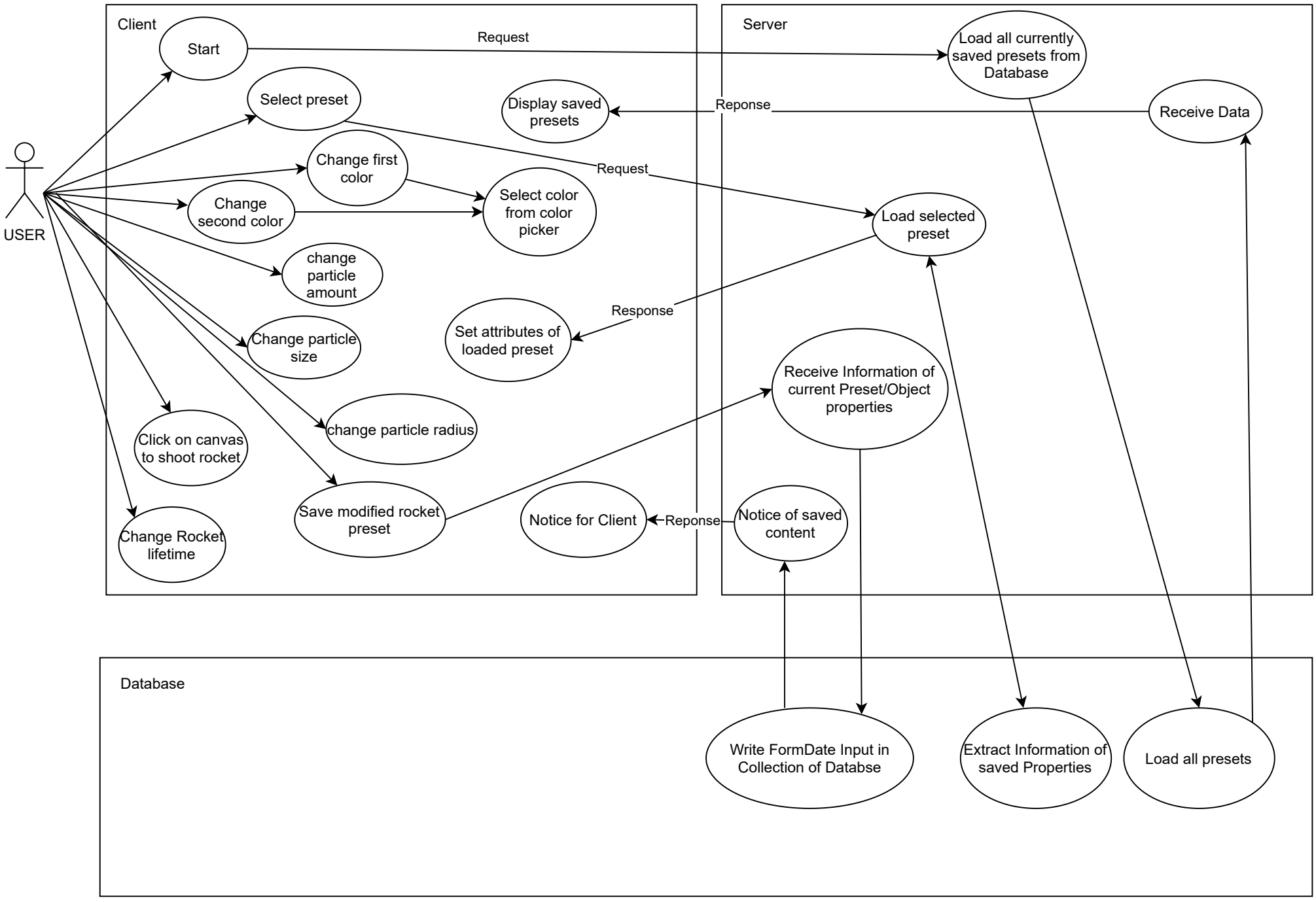
To play Rocket Jam no additional software is needed. Just open your browser using the link and let it bang. In order to install MongoDB you will have to create a user account and a cluster. The cluster contains the databasecollections in which data that is collected using an application, can be saved.

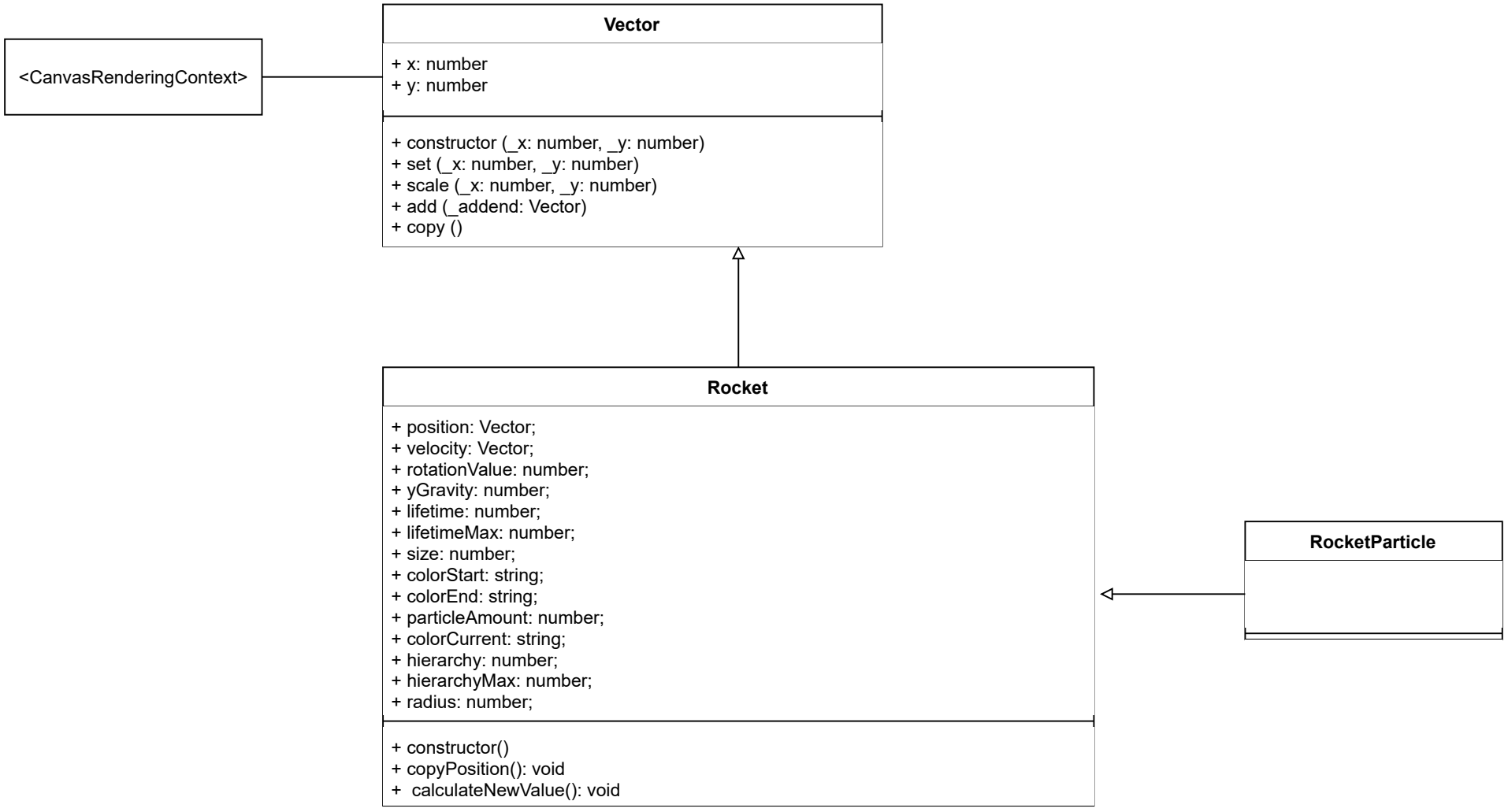
After connecting your application a link is generated that has to be placed into your server script. This string contains a user and password query which you have set before.
Only after allowing whitelist access with the IP 0.0.0.0. / 0 connections from all networks are granted.

Setting up Heroku works similar, you create a user account and choose Node.js as your primary language before you create an app. You will need to connect your Github repository with this app in order for them to communicate properly. Rocket Jam is uploaded into a specific repository and BOOM. Within the code all paths have to be declared correct and folder structures have to be considered.

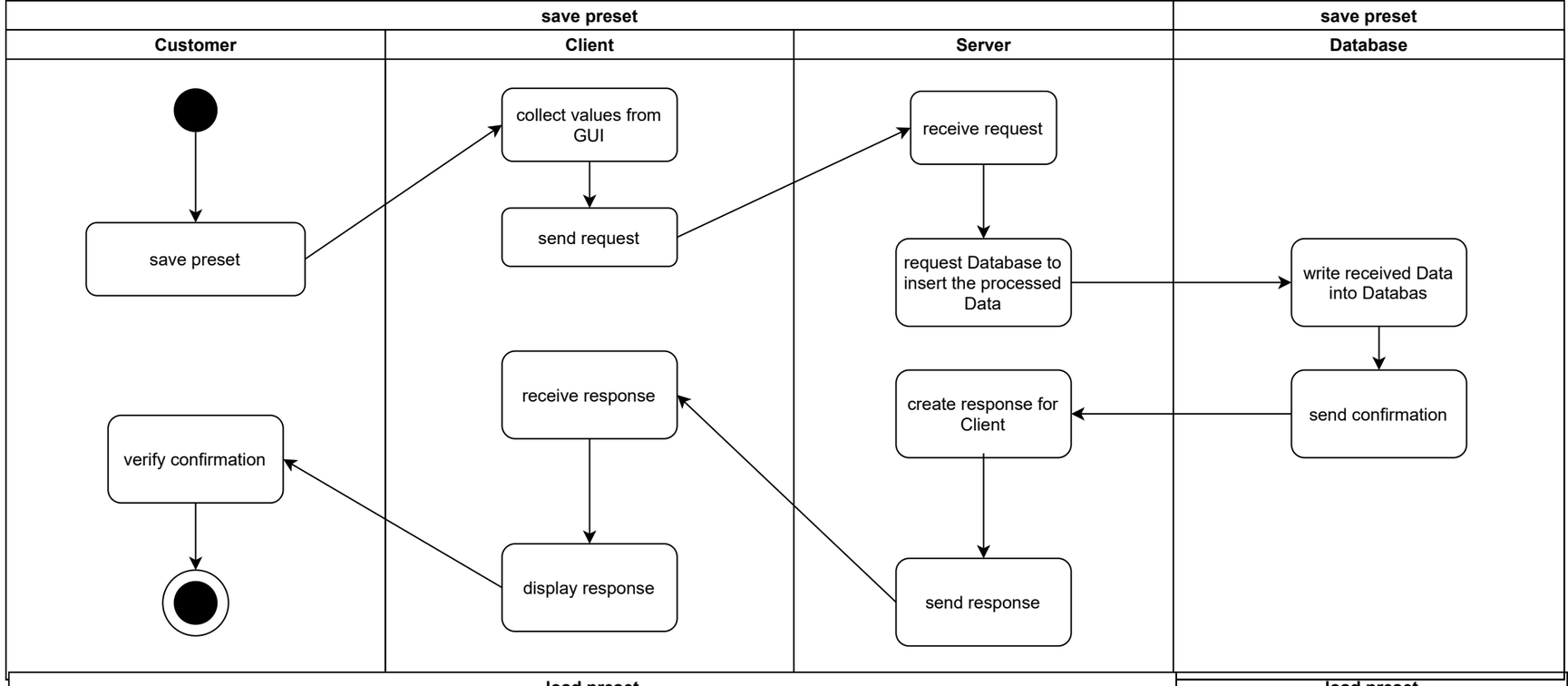
This procedure will work fine with any other application written in supported languages.

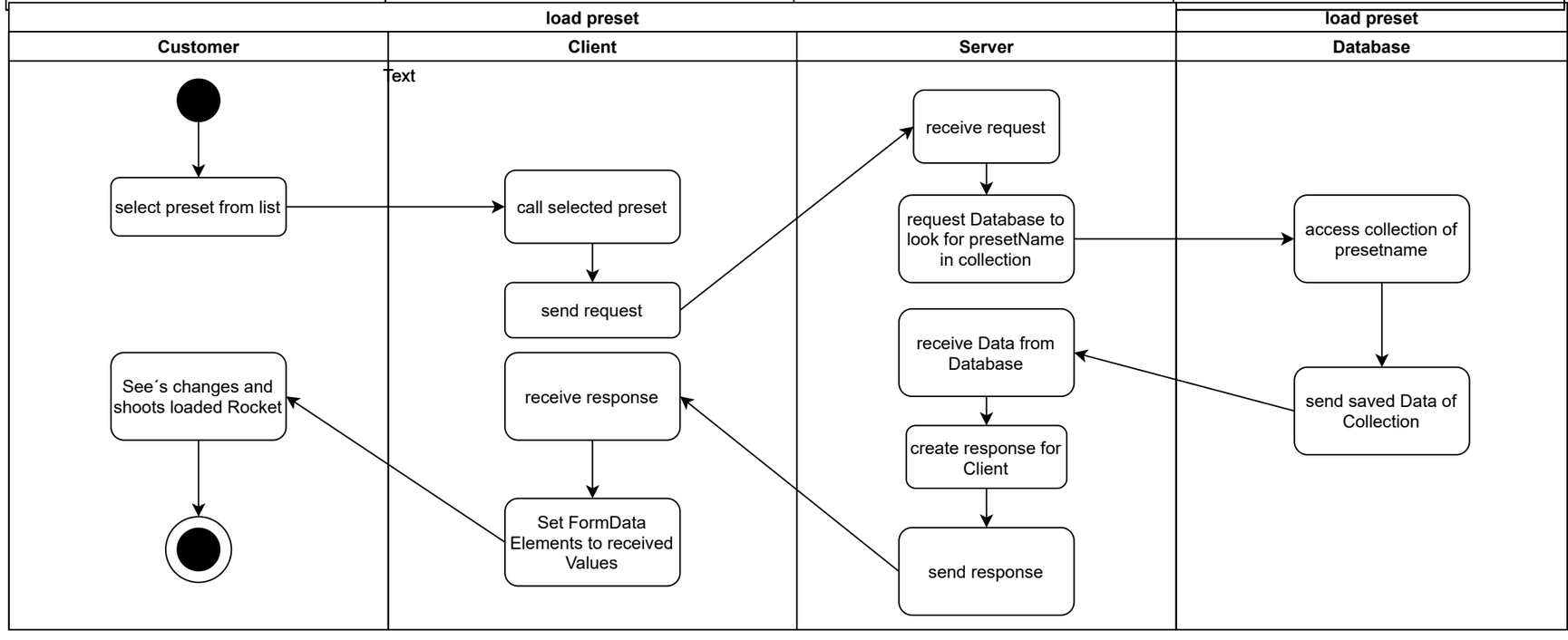
On the following pages you will find the use case diagram created at the beginning of this project followed by all class and activit diagrams.


Client
Start
Request
Server
Load all currently saved presets from Database
Select preset
Display saved presets
Reponse
Receive Data
Change first color
Request
USER
Change second color
Select color from color picker
Load selected preset
change particle amount
Response
Change particle size
Set attributes of loaded preset
Receive Information of current Preset/Object properties
Click on canvas to shoot rocket
change particle radius
Save modified rocket preset
Notice for Client
Reponse
Notice of saved content
Change Rocket lifetime
Database
Write FormDate Input in Collection of Databse
Extract Information of saved Properties
Load all presets

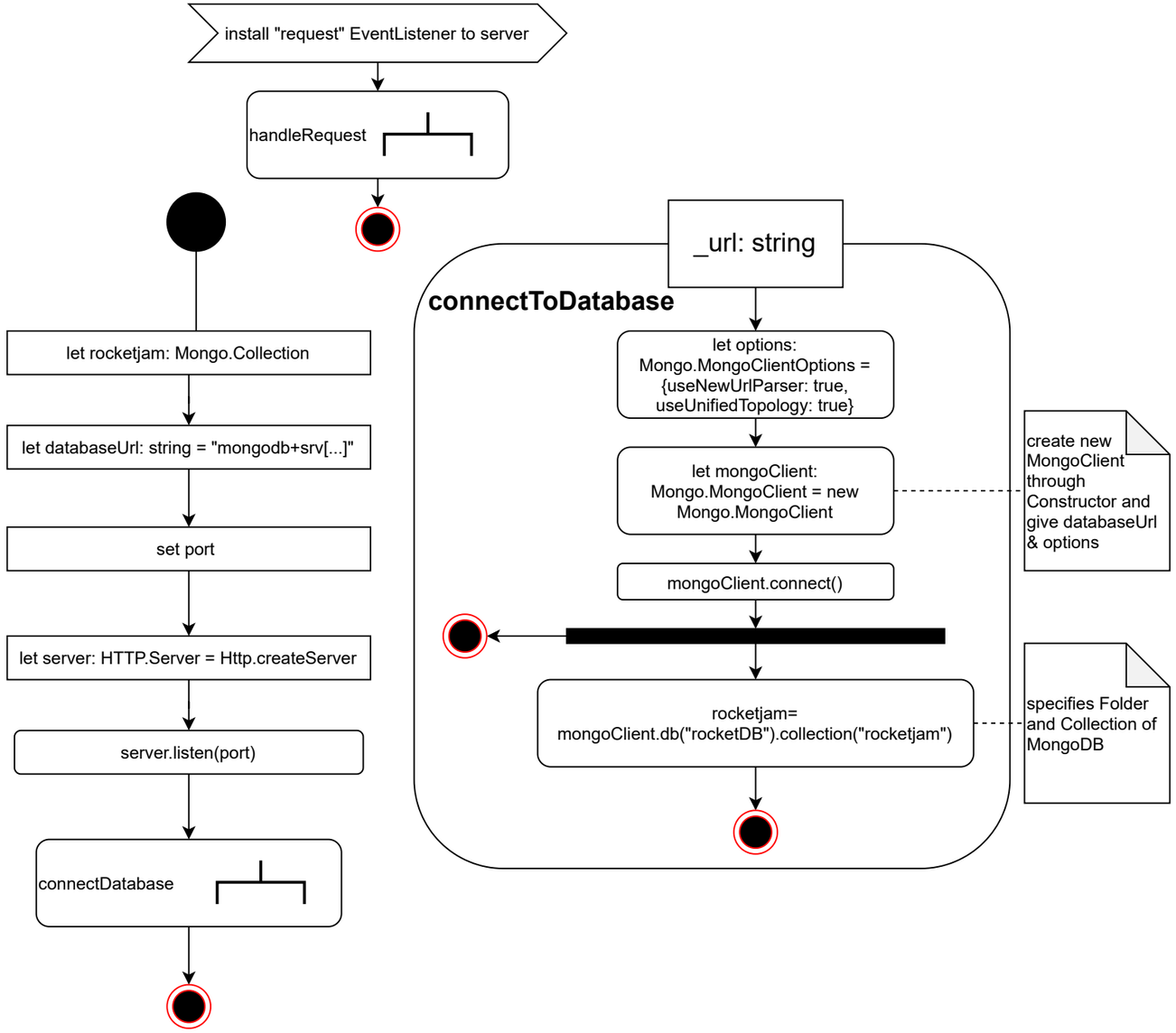

<CanvasRenderingContext>
Vector
+ x: number
+ y: number
+ constructor (_x: number, _y: number)
+ set (_x: number, _y: number)
+ scale (_x: number, _y: number)
+ add (_addend: Vector)
+ copy ()
Rocket
+ position: Vector;
+ velocity: Vector;
+ rotationValue: number;
+ yGravity: number;
+ lifetime: number;
+ lifetimeMax: number;
+ size: number;
+ colorStart: string;
+ colorEnd: string;
+ particleAmount: number;
+ colorCurrent: string;
+ hierarchy: number;
+ hierarchyMax: number;
+ radius: number;
+ constructor()
+ copyPosition(): void
+ calculateNewValue(): void
RocketParticle

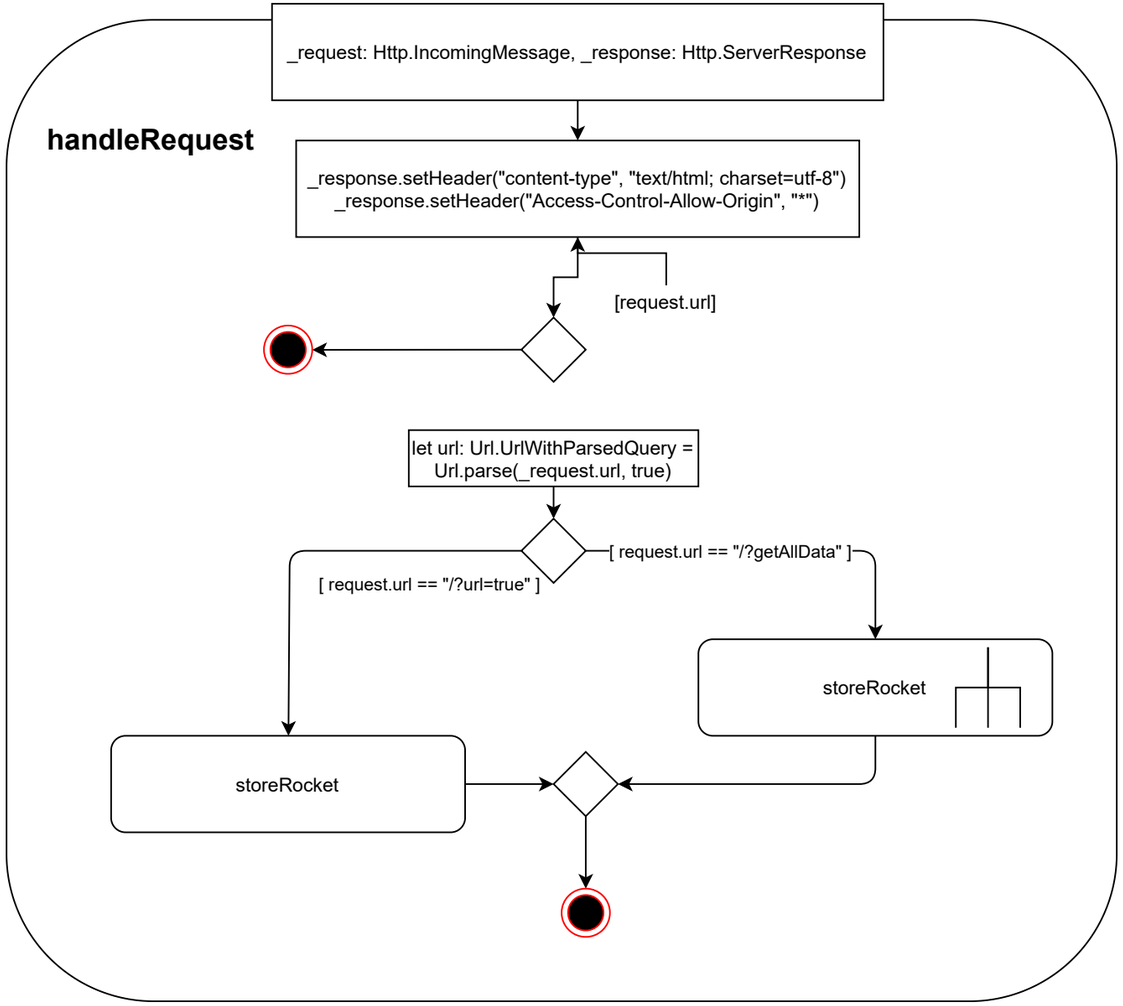
Cross-Domain Tracking



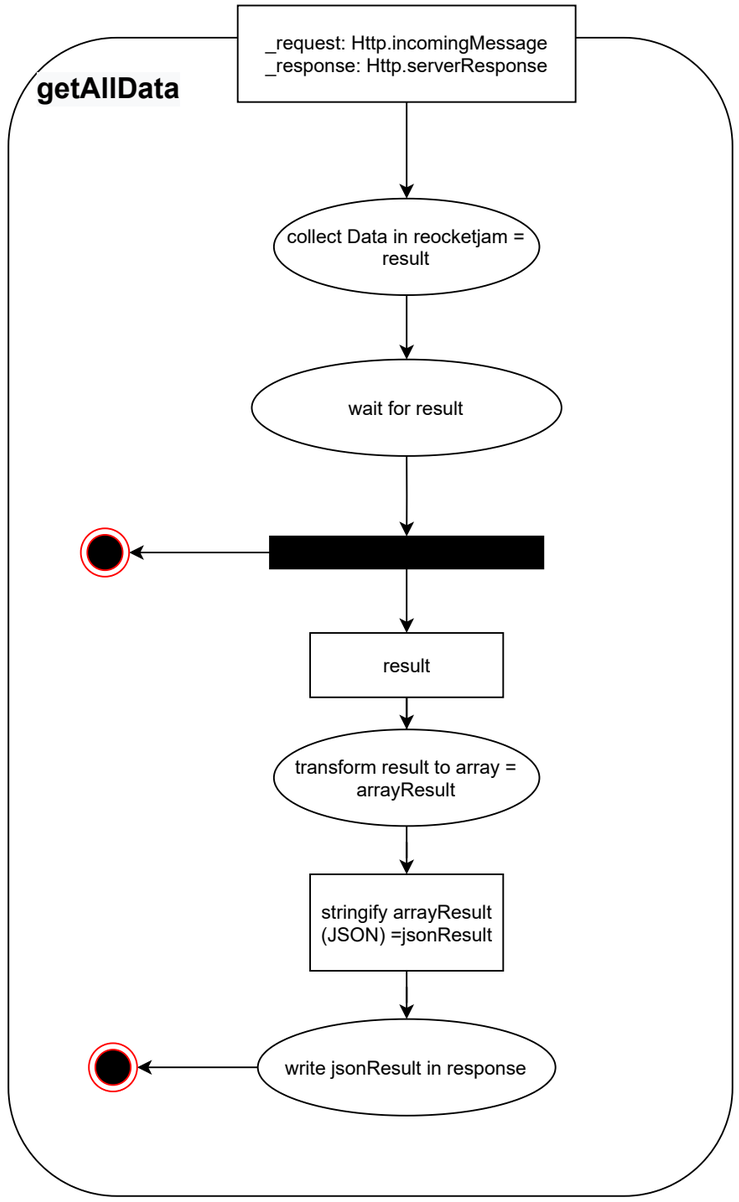


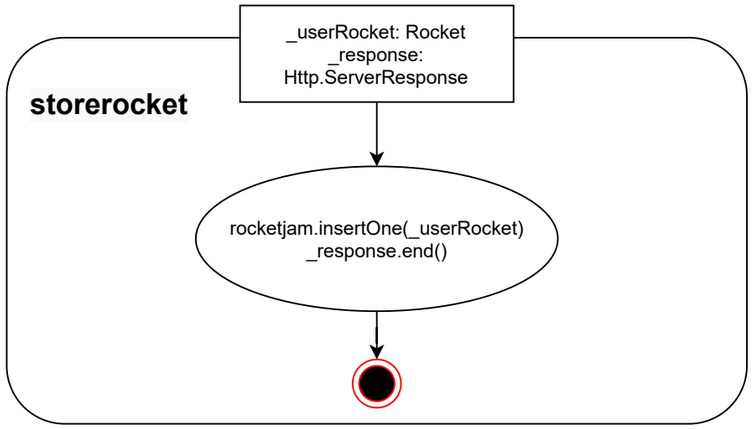
Server Activity Diagram



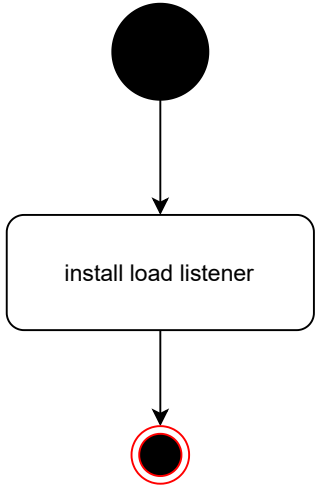


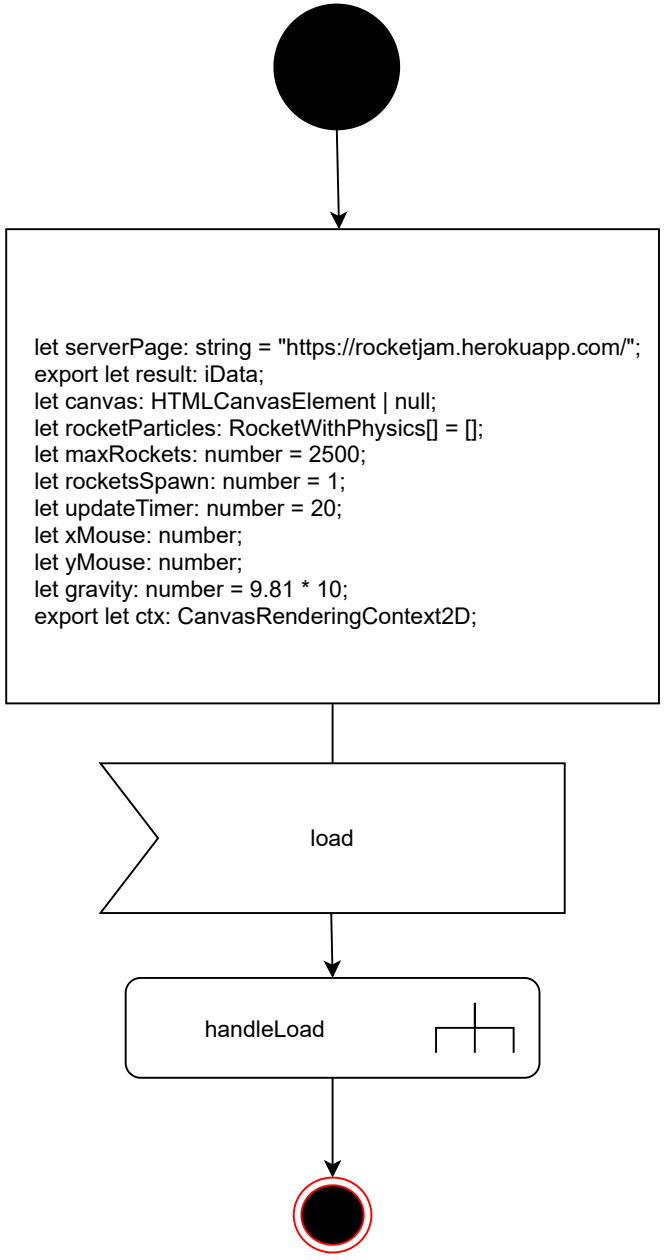
Server Activity Diagram



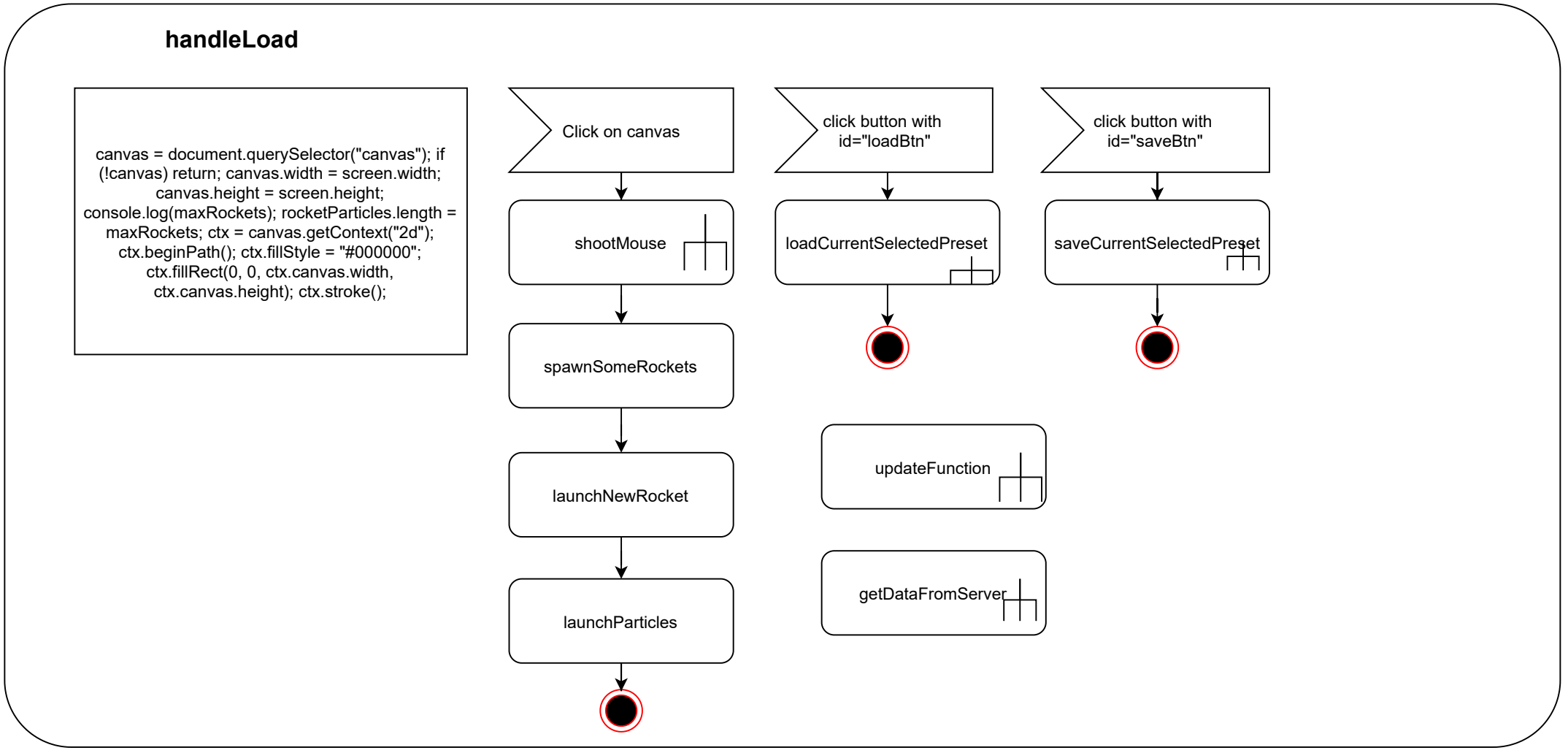


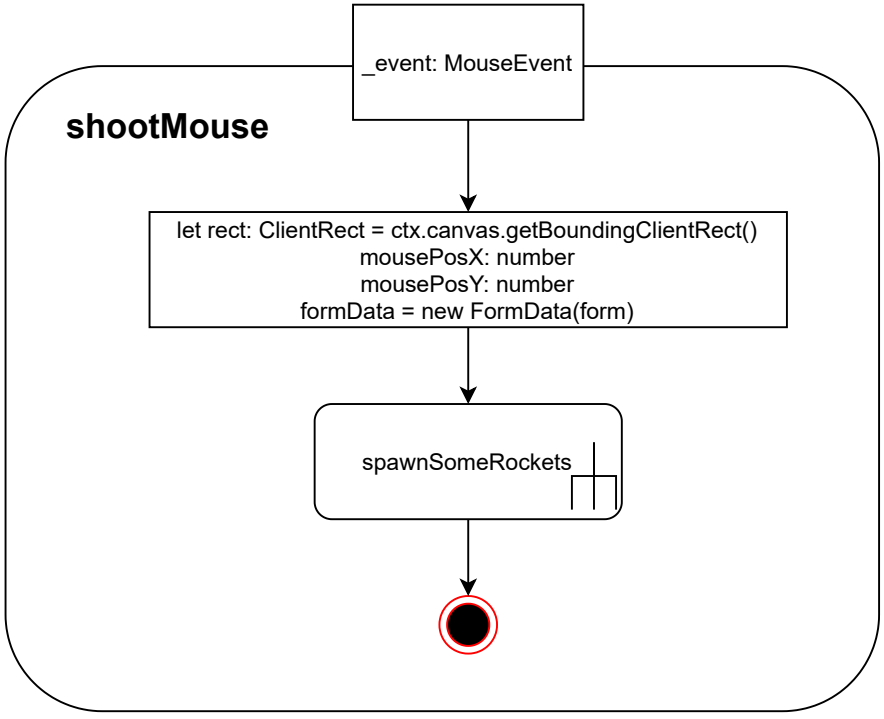
Main Activity Diagram

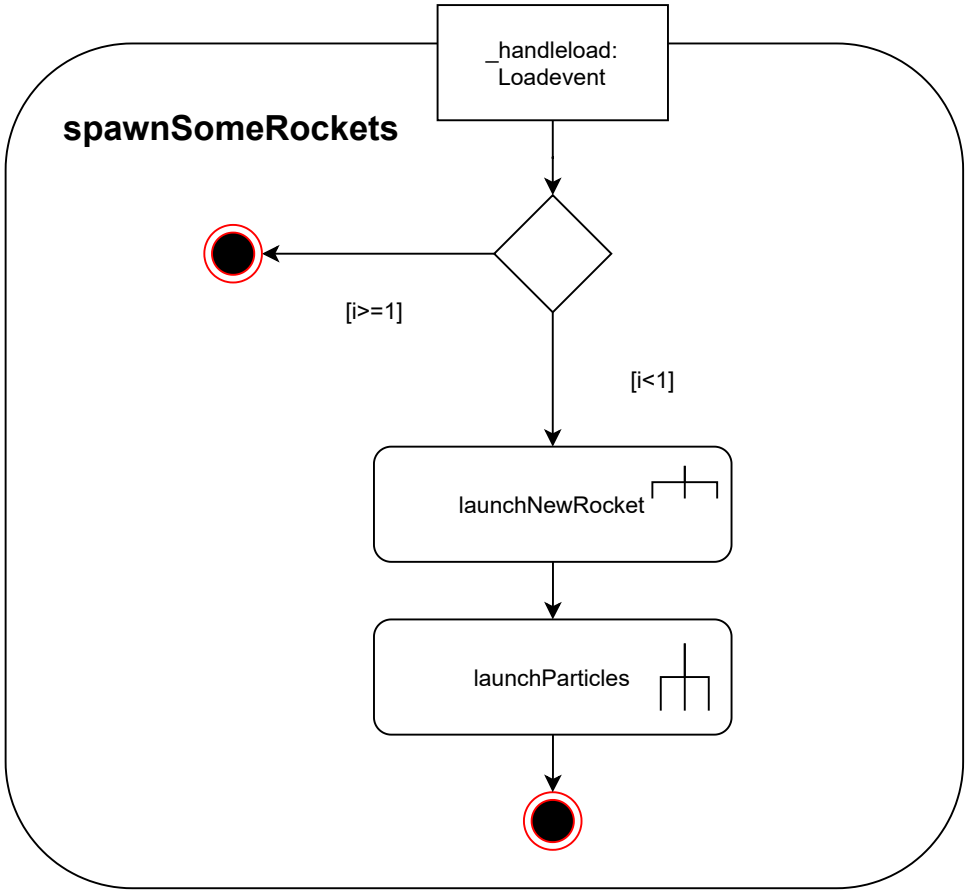


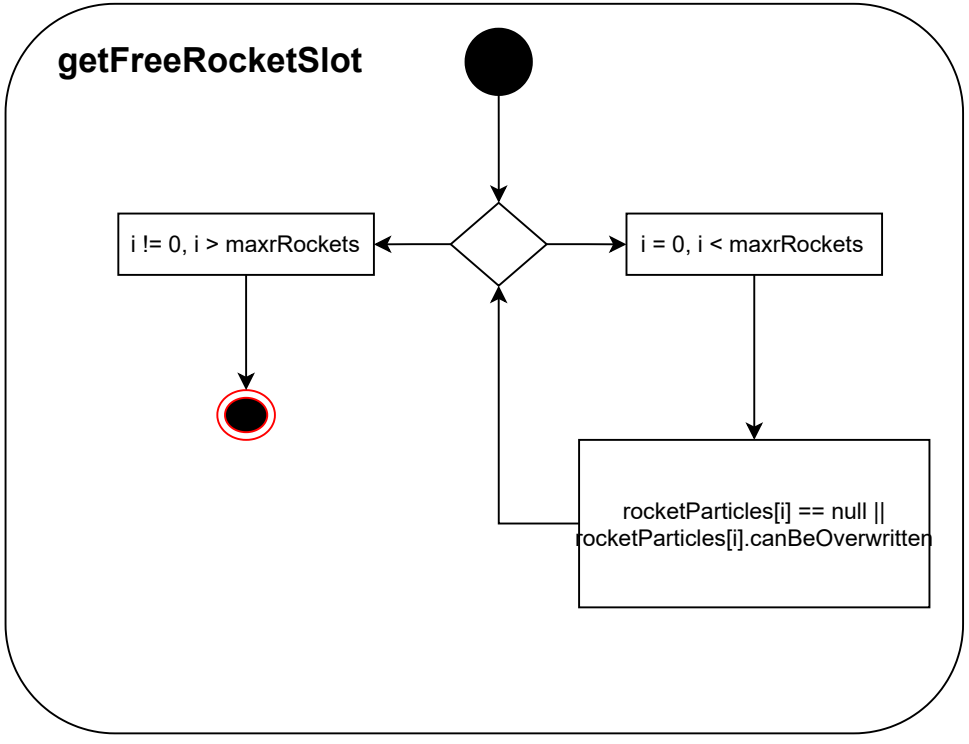


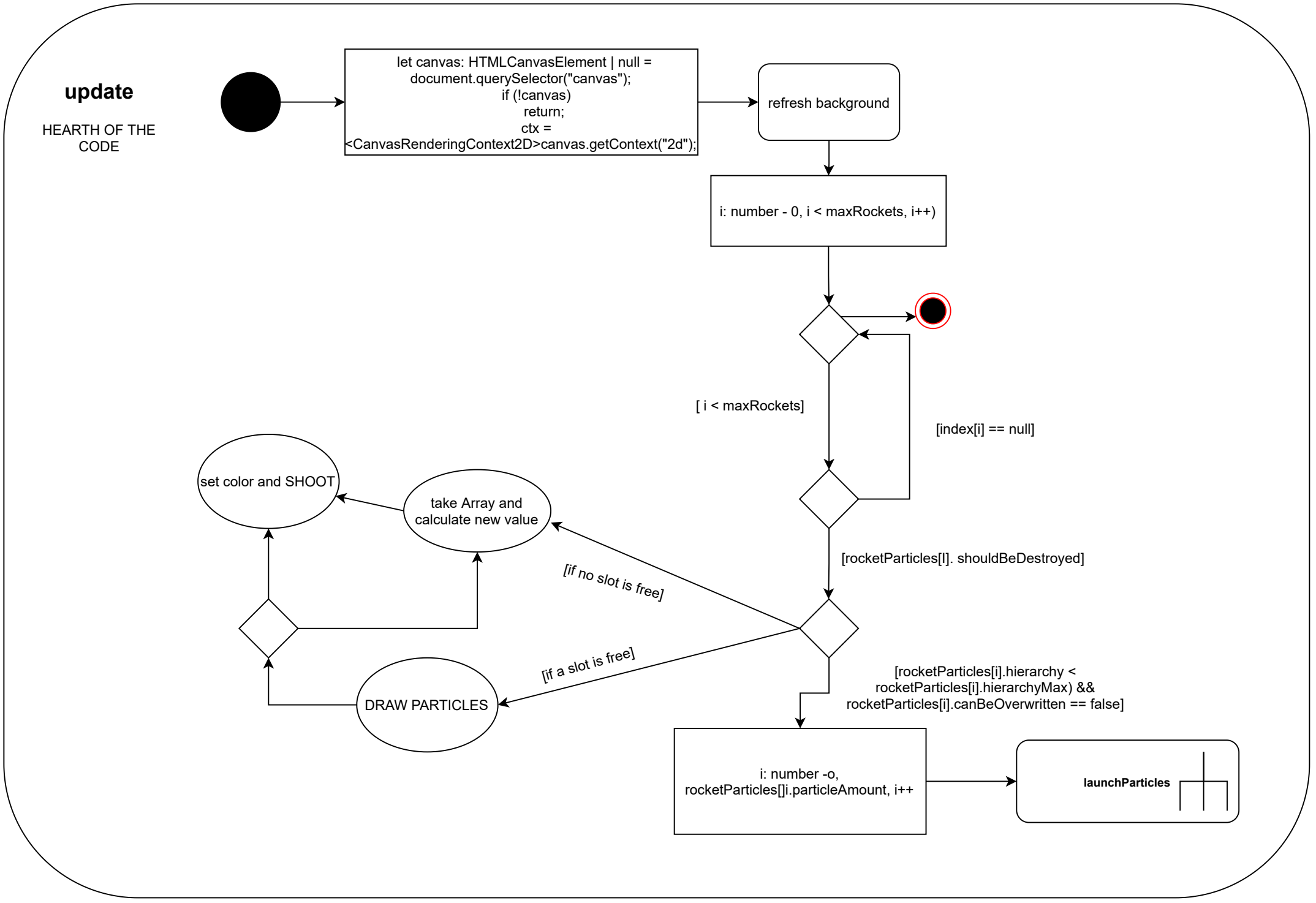
Main Activity Diagram

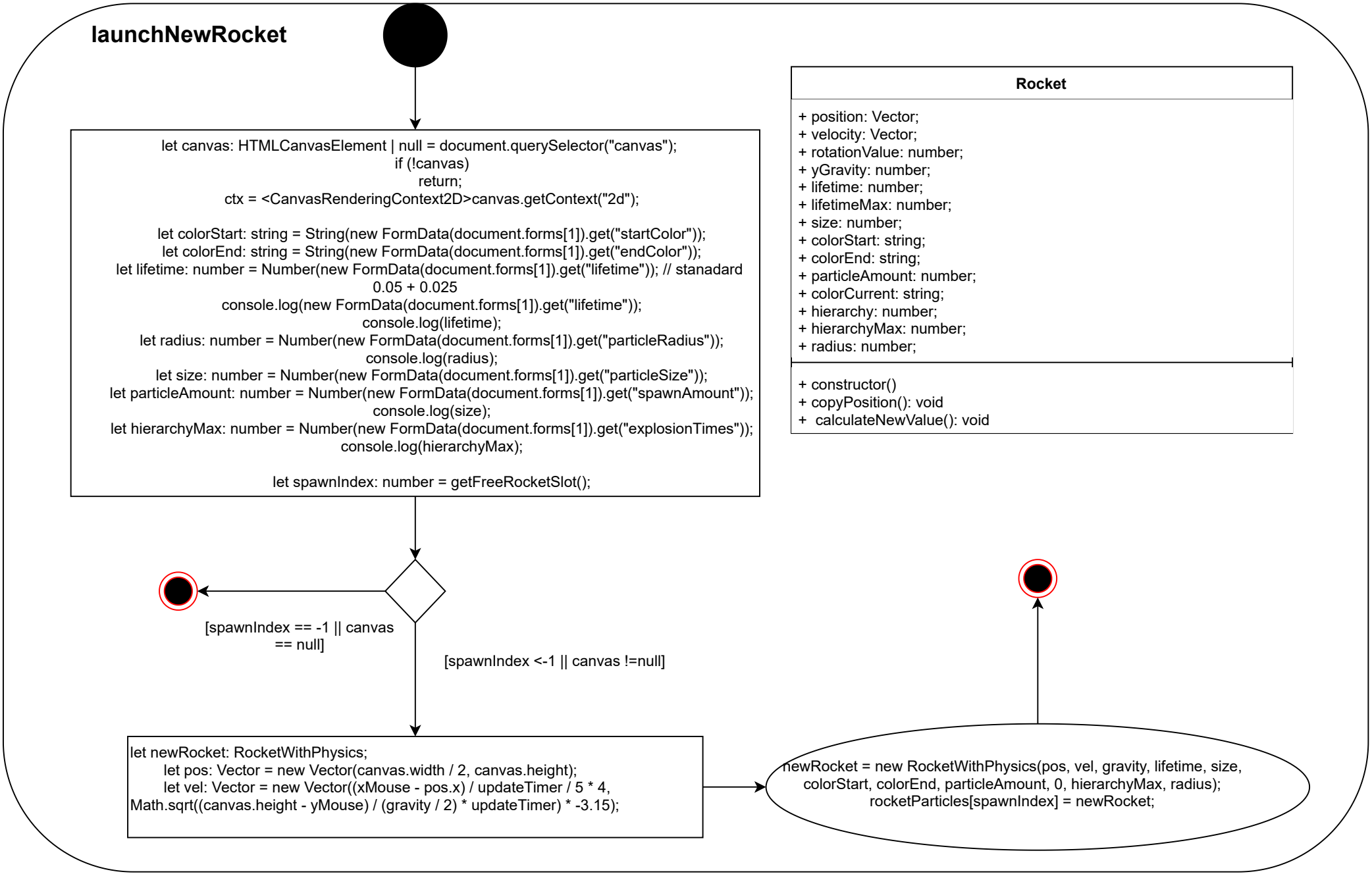


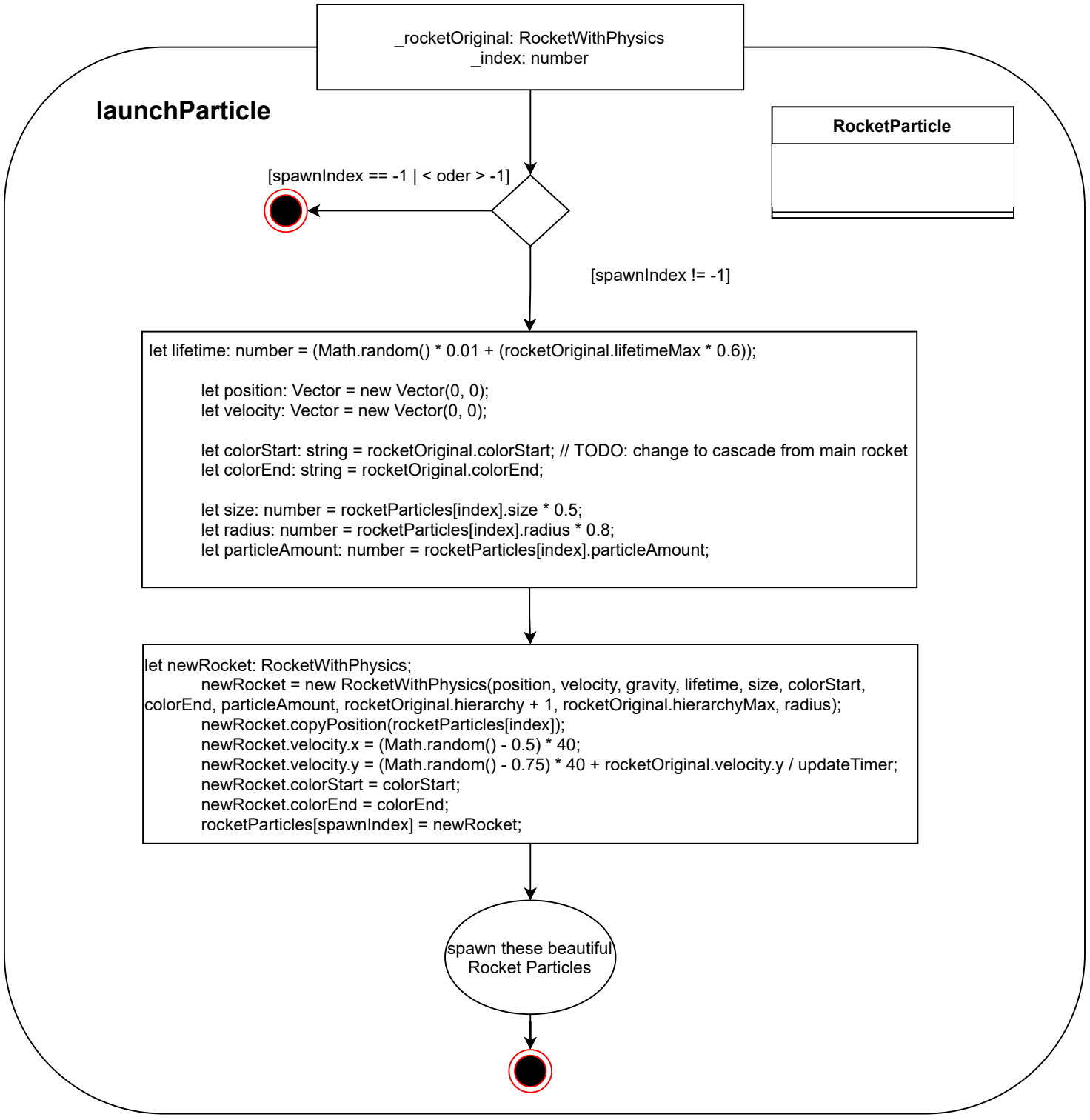

shootMouse
_event: MouseEvent
let rect: ClientRect = ctx.canvas.getBoundingClientRect()
mousePosX: number
mousePosY: number
formData = new FormData(form)
spawnSomeRockets


spawnSomeRockets
_handleload:
Loadevent
[i>=1]
[i<1]
launchNewRocket
launchParticles


getFreeRocketSlot
i != 0, i > maxrRockets
i = 0, i < maxrRockets
rocketParticles[i] == null ||
rocketParticles[i].canBeOverwritten


update
HEARTH OF THE CODE
let canvas: HTMLCanvasElement | null = document.querySelector("canvas"); if (!canvas) return; ctx = <CanvasRenderingContext2D>canvas.getContext("2d");
refresh background
i: number - 0, i < maxRockets, i++)
[i < maxRockets]
[index[i] == null]
[rocketParticles[I]. shouldBeDestroyed]
[if no slot is free]
[if a slot is free]
set color and SHOOT
take Array and calculate new value
DRAW PARTICLES
[rocketParticles[i].hierarchy < rocketParticles[i].hierarchyMax) && rocketParticles[i].canBeOverwritten == false]
i: number -o, rocketParticles[]i.particleAmount, i++
launchParticles


launchNewRocket
let canvas: HTMLCanvasElement | null = document.querySelector("canvas");
if (!canvas)
return;
ctx = <CanvasRenderingContext2D>canvas.getContext("2d");
let colorStart: string = String(new FormData(document.forms[1]).get("startColor"));
let colorEnd: string = String(new FormData(document.forms[1]).get("endColor"));
let lifetime: number = Number(new FormData(document.forms[1]).get("lifetime")); // stanadard 0.05 + 0.025
console.log(new FormData(document.forms[1]).get("lifetime"));
console.log(lifetime);
let radius: number = Number(new FormData(document.forms[1]).get("particleRadius"));
console.log(radius);
let size: number = Number(new FormData(document.forms[1]).get("particleSize"));
let particleAmount: number = Number(new FormData(document.forms[1]).get("spawnAmount"));
console.log(size);
let hierarchyMax: number = Number(new FormData(document.forms[1]).get("explosionTimes"));
console.log(hierarchyMax);
let spawnIndex: number = getFreeRocketSlot();
[spawnIndex == -1 || canvas == null]
[spawnIndex <-1 || canvas !=null]
let newRocket: RocketWithPhysics;
let pos: Vector = new Vector(canvas.width / 2, canvas.height);
let vel: Vector = new Vector((xMouse - pos.x) / updateTimer / 5 * 4, Math.sqrt((canvas.height - yMouse) / (gravity / 2) * updateTimer) * -3.15);
newRocket = new RocketWithPhysics(pos, vel, gravity, lifetime, size, colorStart, colorEnd, particleAmount, 0, hierarchyMax, radius);
rocketParticles[spawnIndex] = newRocket;
Rocket
+ position: Vector;
+ velocity: Vector;
+ rotationValue: number;
+ yGravity: number;
+ lifetime: number;
+ lifetimeMax: number;
+ size: number;
+ colorStart: string;
+ colorEnd: string;
+ particleAmount: number;
+ colorCurrent: string;
+ hierarchy: number;
+ hierarchyMax: number;
+ radius: number;
+ constructor()
+ copyPosition(): void
+ calculateNewValue(): void


_rocketOriginal: RocketWithPhysics
_index: number
launchParticle
RocketParticle
[spawnIndex == -1 | < oder > -1]
[spawnIndex != -1]
let lifetime: number = (Math.random() * 0.01 + (rocketOriginal.lifetimeMax * 0.6));
let position: Vector = new Vector(0, 0);
let velocity: Vector = new Vector(0, 0);
let colorStart: string = rocketOriginal.colorStart; // TODO: change to cascade from main rocket
let colorEnd: string = rocketOriginal.colorEnd;
let size: number = rocketParticles[index].size * 0.5;
let radius: number = rocketParticles[index].radius * 0.8;
let particleAmount: number = rocketParticles[index].particleAmount;
let newRocket: RocketWithPhysics;
newRocket = new RocketWithPhysics(position, velocity, gravity, lifetime, size, colorStart,
colorEnd, particleAmount, rocketOriginal.hierarchy + 1, rocketOriginal.hierarchyMax, radius);
newRocket.copyPosition(rocketParticles[index]);
newRocket.velocity.x = (Math.random() - 0.5) * 40;
newRocket.velocity.y = (Math.random() - 0.75) * 40 + rocketOriginal.velocity.y / updateTimer;
newRocket.colorStart = colorStart;
newRocket.colorEnd = colorEnd;
rocketParticles[spawnIndex] = newRocket;
spawn these beautiful
Rocket Particles

Vector Activity Diagram

Vector
+ x: number + y: number
+ constructor (_x: number, _y: number) + set (_x: number, _y: number) + scale (_factor) + add (_addend: Vector) + copy ()