Git & VSC

- enable sourcemap in tsconfig for better debugging
- Linter helps retain cleaner code structure
- open in integrated Terminal -> vsc -watch for the .ts-file
- git-ignore folders -> simple folder path
- "!" in empty HTML-file -> automatic HTML-structure
- right click -> "Format Document"-option
- Ctrl + Shift + 7 -> Toggle Comment line
- clear commit messages and small commit steps

Typescript

- namespaces: spread-out sections of code, even across multiple files
  Syntax: namespace Example { }
  ↳ also lets you section off code for managing variables

- "export"-keyword in front of declarations lets other files use that variable/function/object

- for...of-loop: a shorter way to e.g. cycle through an array
  Syntax: for (object of array) { }

not to be confused with the for...in-loop, which only gives out the entries numbers instead of the array entries themselves

- null: short for "null pointer". A value that makes it possible for complex variables to be empty without it being undefined (→ an empty reference)
  Syntax: let objekt : Object | null = null

- indexOf: returns the index of a given object in an array
  Syntax: array.indexOf(object)

- splicing an array:
  Syntax: array.splice (

Classes & Objects

Class	↔	Object
Blueprint	↔	House

A class is a structure that lets you define objects properties and functions. An instance of a class is then called an object.

Syntax

```
class Vector {

    x: number;  > class attributes
    y: number;

    scale(_factor: number): void {
        this.x *= _factor;
        this.y *= _factor;
    }

    add(_addend: Vector): void {
        this.x += _addend.x;
        this.y += _addend.y;
    }
}
```

class methods (things the object can inherently do)

Constructors

A constructor "builds" an instance of a class and can (for example) be told which information are needed to construct an object or other specifications.
In case of the "Vector"-class, a constructor could take in initial values for the created Vector:

```
class Vector {
 ↑attributes
    constructor(_x, _y) {
        this.set(_x, _y)
    }

    set(_x, _y): void {
        this.x = _x;
        this.y = _y;
 ↓methods
```

```
let v1: Vektor = new Vektor(5, 3)
```

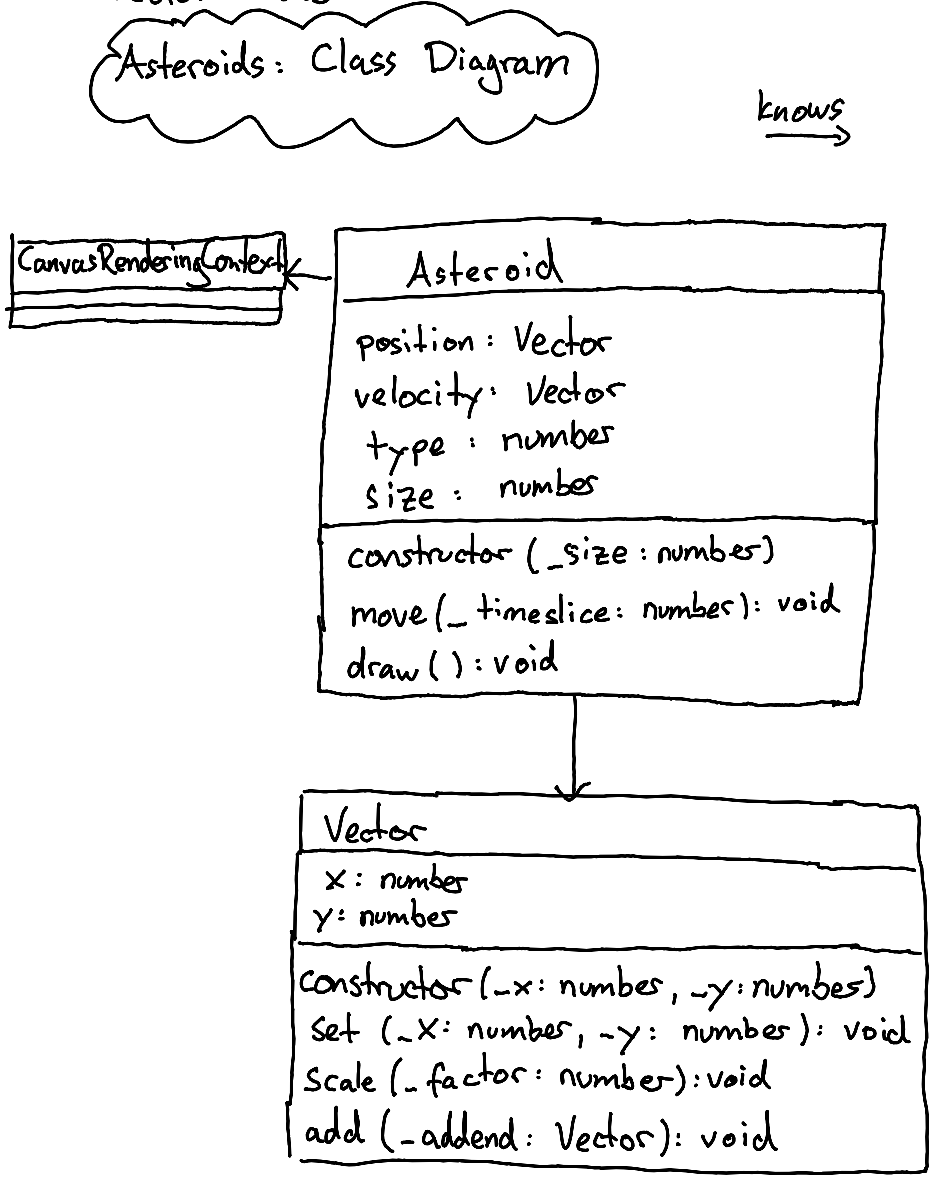
using the constructor to instance a new vector

Implementation

When implementing classes into a program, each class should have a seperate .ts-file (except maybe really small classes only used in one part of a program), with the "export"-Keyword in front of the "class"-Keyword.
Don't forget to link those new scripts in your html-file!
That way, the code is much more organized and lets you find specific parts faster.

Class Diagrams

Example diagram based on the popular arcade game "Asteroids", working with the prior implemented "Vector"-class:



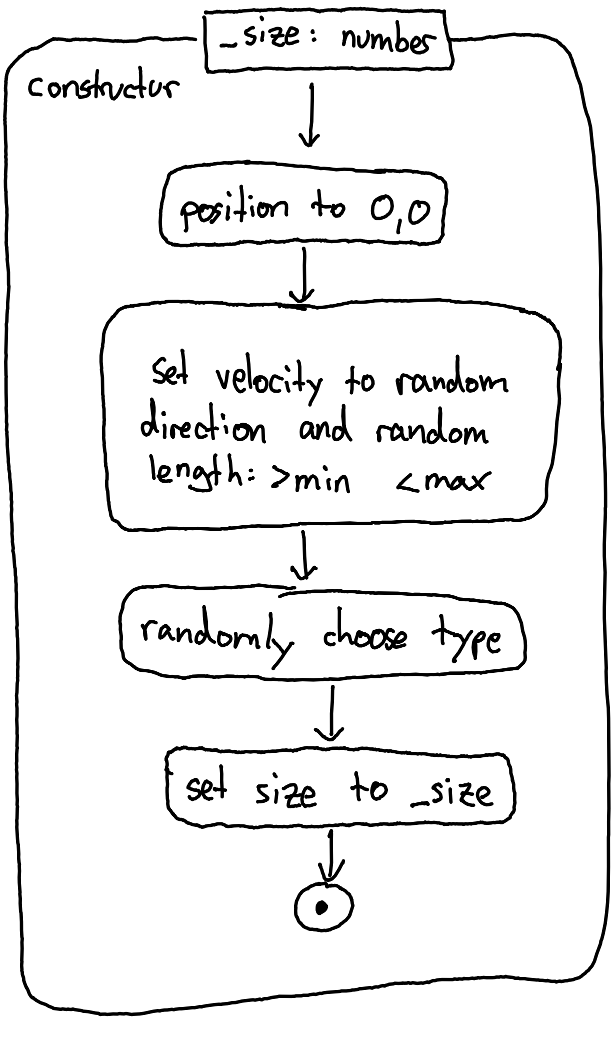
Asteroids: Class Diagram
knows
CanvasRenderingContext
Asteroid
position: Vector
velocity: Vector
type: number
size: numbes
constructor (_size: numbes)
move (_timeslice: numbes): void
draw (): void
Vector
x: number
y: numbes
constructor (_x: numbes, _y: numbes)
set (_x: number, _y: numbes): void
Scale (_factor: number): void
add (_addend: Vector): void

5 Questions to consider when creating classes:

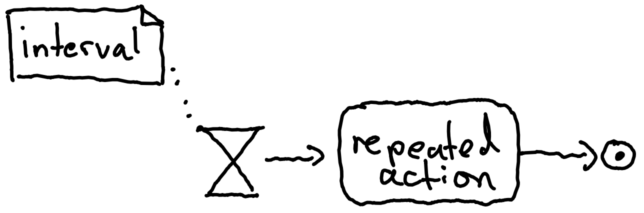
1. What does it have? (Attributes)
2. What can it do? (Methods)
3. What does it know? (Outside information)
4. Who holds it?
5. What is it?

Activity Diagram 2.0

Based on a class diagram, the methods of a class can now be described in detail via an activity diagram:



For the main program, a continuous game loop can be represented with this hourglass-symbol:



This symbol can also be used for other time-controlled activities.

The Gameloop

-> The Program that continously runs to create a Gameplay-Experience for the user. That includes stuff like animation and re-checking user input

Despite its name, a Game Loop cannot be implemented via a simple Loop (e.g. a for-Loop). We need a function that is repeatedly called, with defined time intervals in between to control the speed of the Game Loop.

Time Signals

In Typescript, there are three different ways to implement time intervals for calling a function:

window.setTimeout(handler, time, extras...)
The handler-function is called after the specified time (in millseconds) has passed. Extra parameters can be handed over to the handler-function. Within the handler-function, another setTimeout for the same function can be set to create an infinite Loop

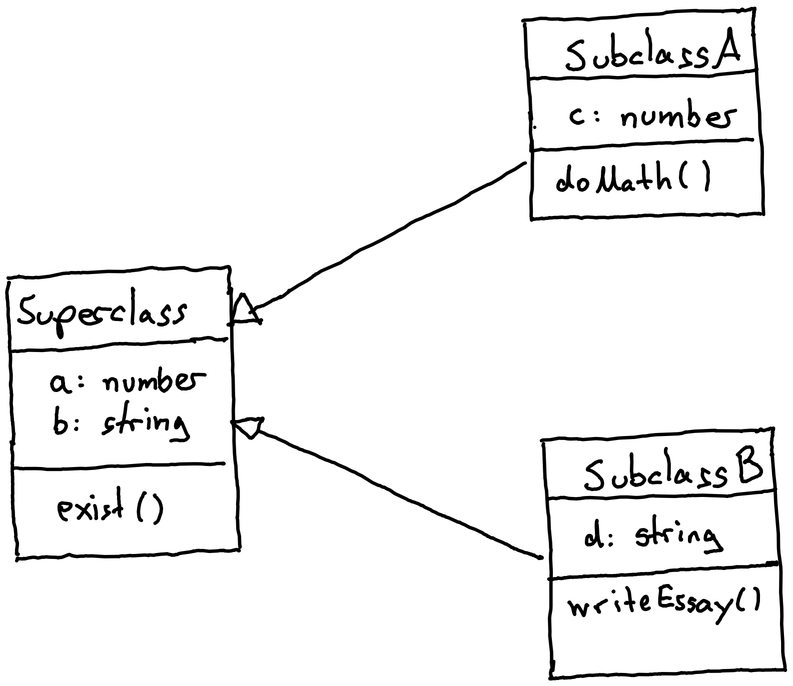
window.setInterval(handler, time, extras...)
Similar to the setTimeout-function, but setInterval periodially repeats the handler-function at the given interval (in milliseconds) without having to call it again

window.requestAnimationFrame(handler)
The handler-function is periodically called in an interval which the browser sets (Chrome tries to reach 60fps).
Gives a lot less control to the programmer.

Inhesitance

Classes can have Subclasses (also referred to as child classes), which inherit their traits and additionally gain new, more specific traits.
Classes that have Subclasses extended from them are called Superclasses (or parent classes).
This can go multiple levels deep.

In a class diagram, these relationships are expressed via an arrow with an empty arrowhead:



Subclasses are always more specialized than their Superclass, while Supesclasses are always more generalized than their Subclasses