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
- "export"-keyword in front of declarations lets other files find that variable/function/object
- Ctrl + Shift + 7 -> Toggle Comment line
- namespaces: spread-out sections of code, even across multiple files
  Syntax: namespace Example { }

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;
    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 attributes

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

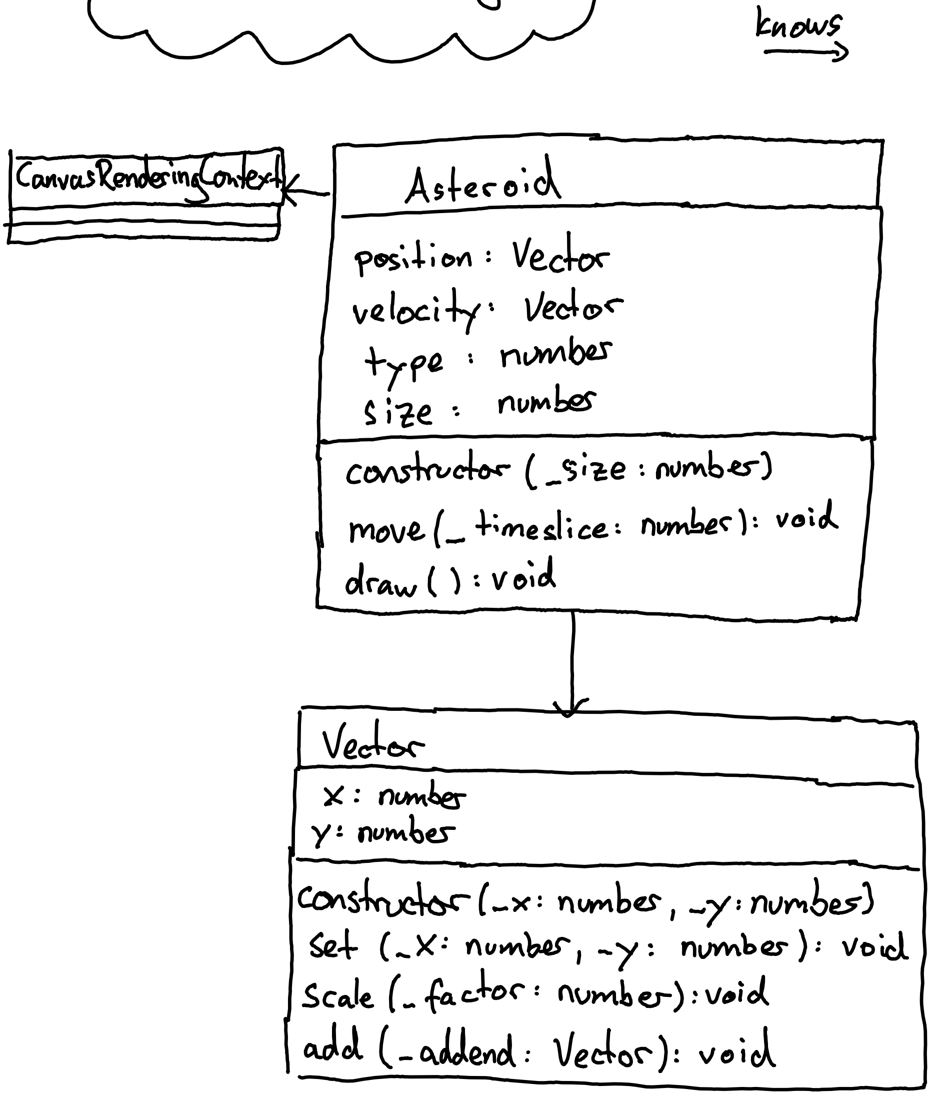
Implementation

When implementing classes into a program, each class should have a seperate .ts-file (except maybe reeally 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 yoor 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

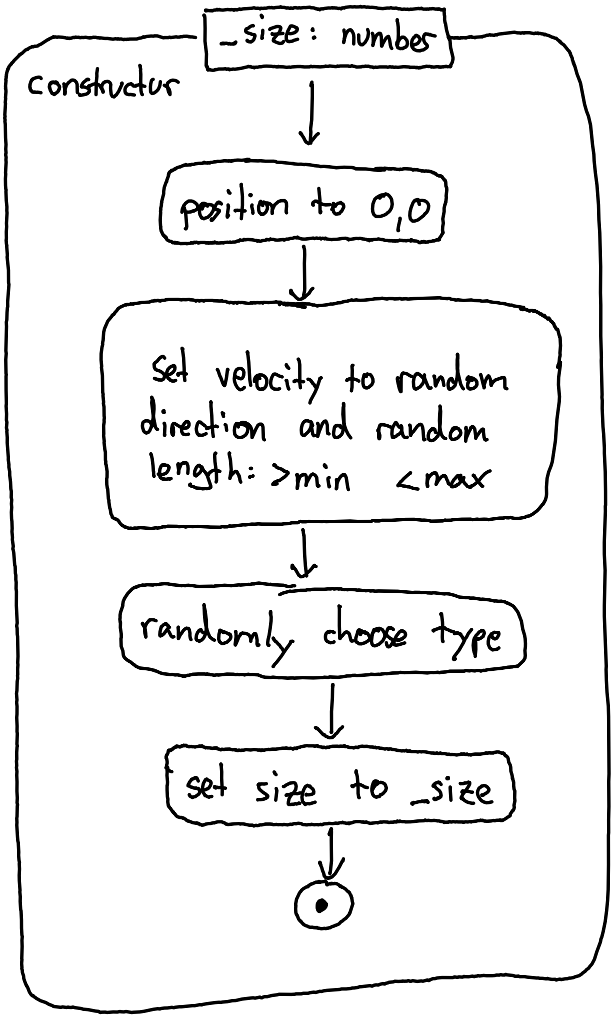

knows
CanvasRenderingContext
Asteroid
position : Vector
velocity : Vector
type : number
size : numbes
constructor (_size : numbes)
move (_timeslice : numbes) : void
draw ( ) : void
Vector
x : number
y : numbes
constructor (_x : numbes, _y : numbes)
Set (_X : number, _y : numbes) : void
Scale (_factor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

5 Questions to consider when creating classes:

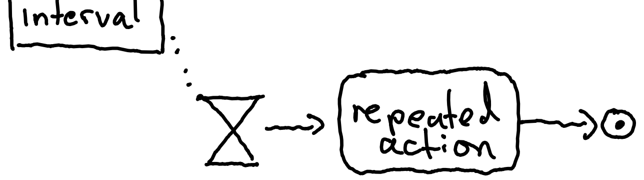
1. What does it have? (Attributes)
2. What can it do? (Methods)
3. What does it know? (Outside information)
4. Who holds it?
5. What is it?

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.