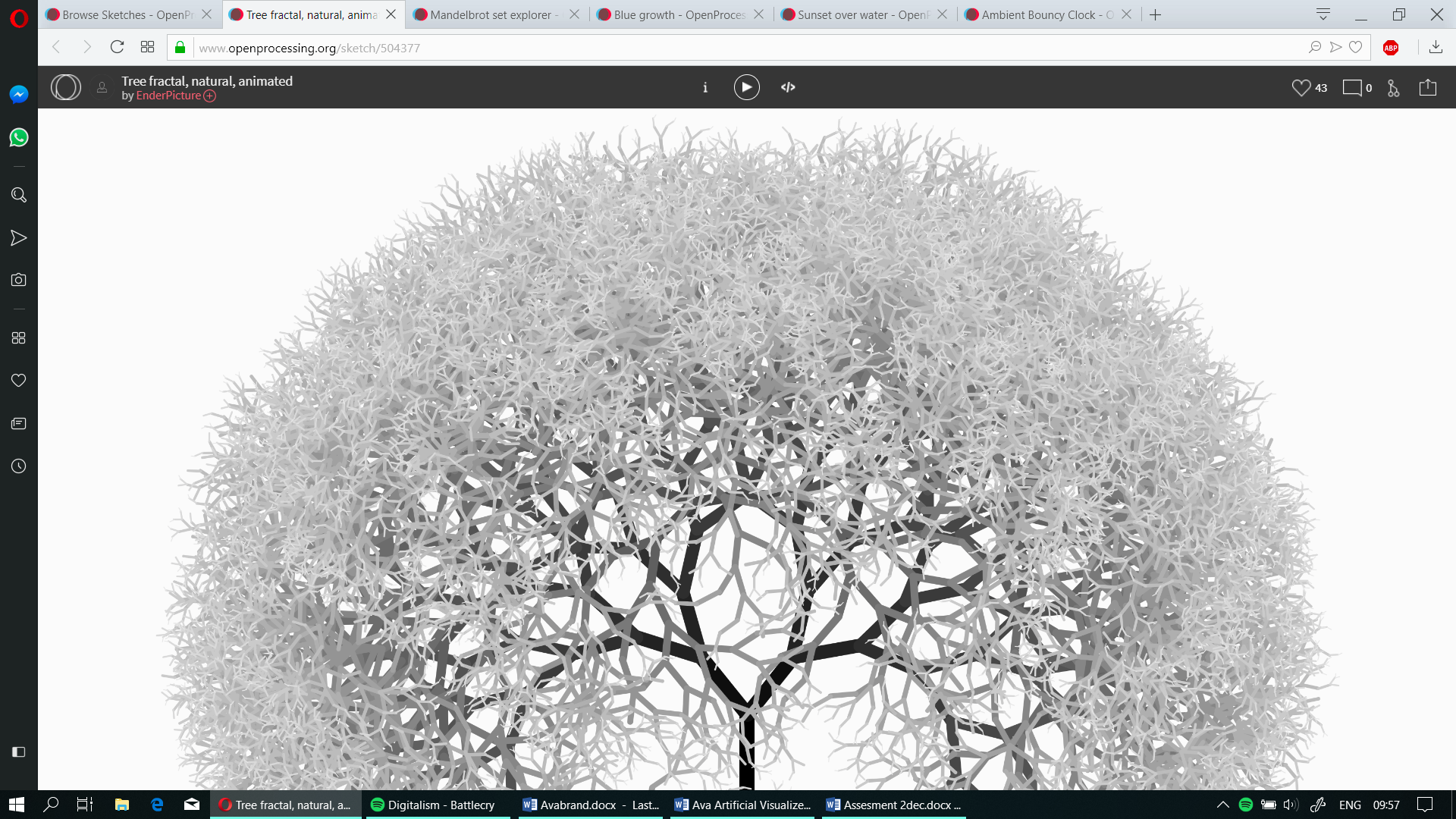
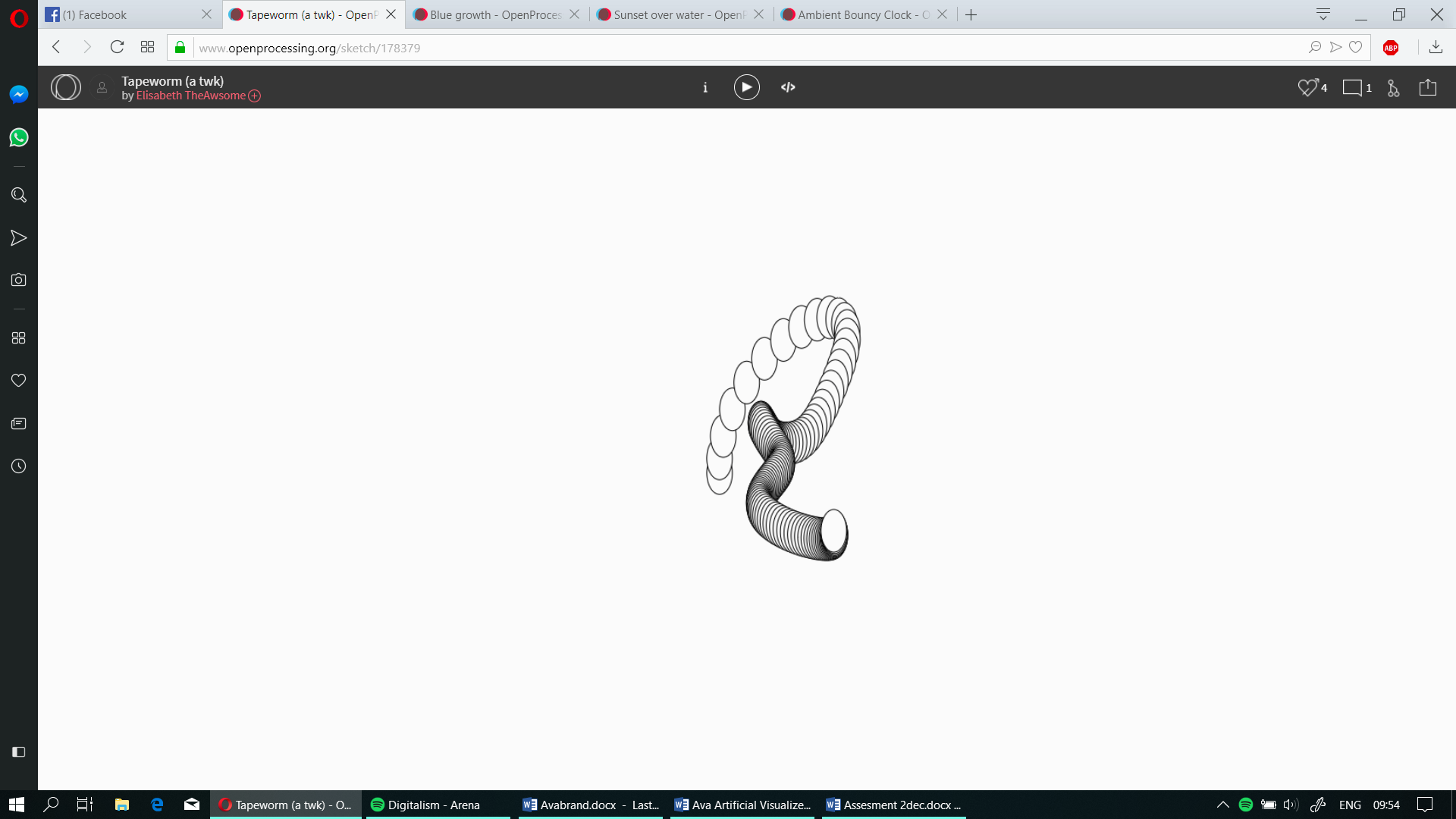
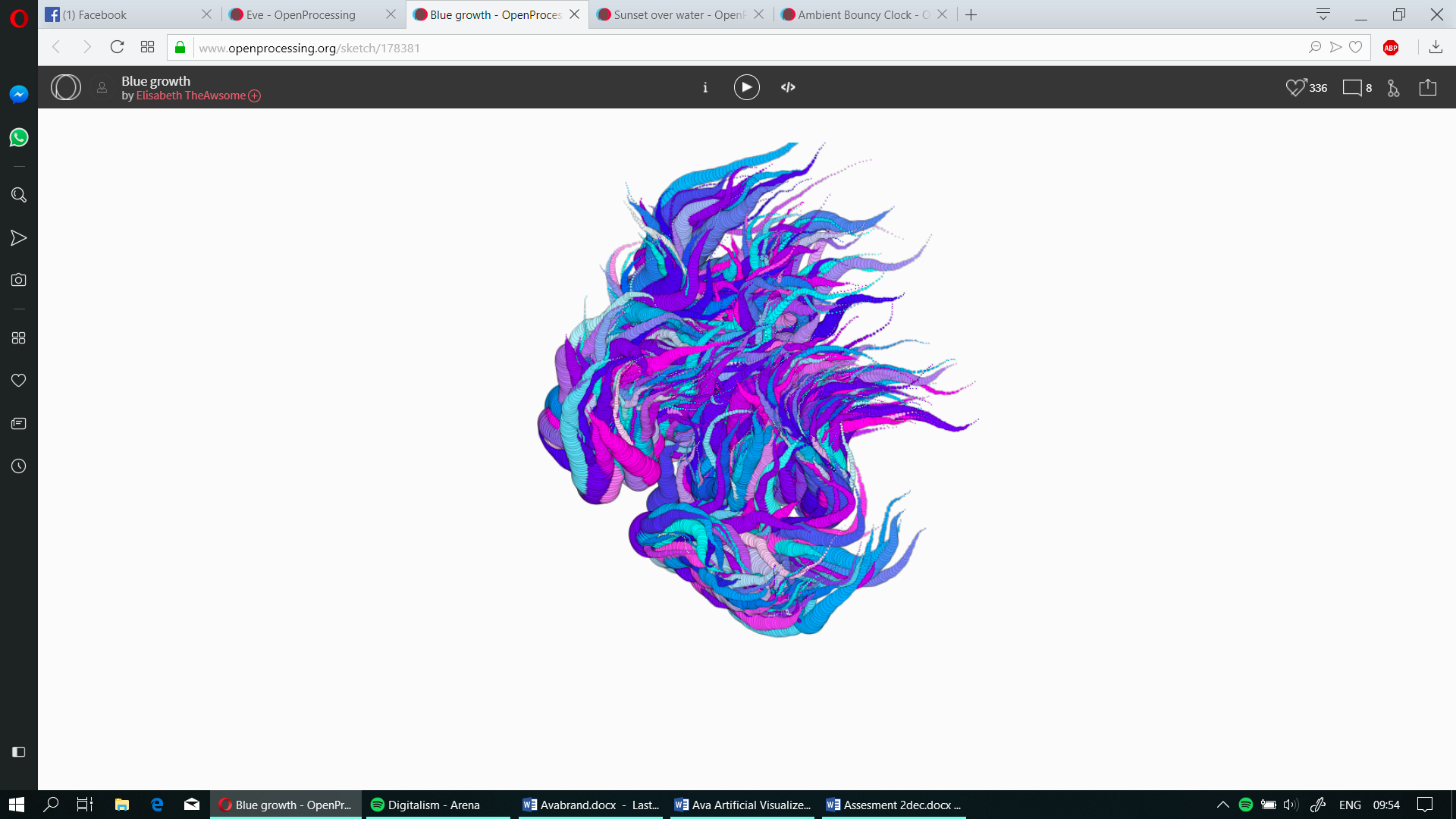
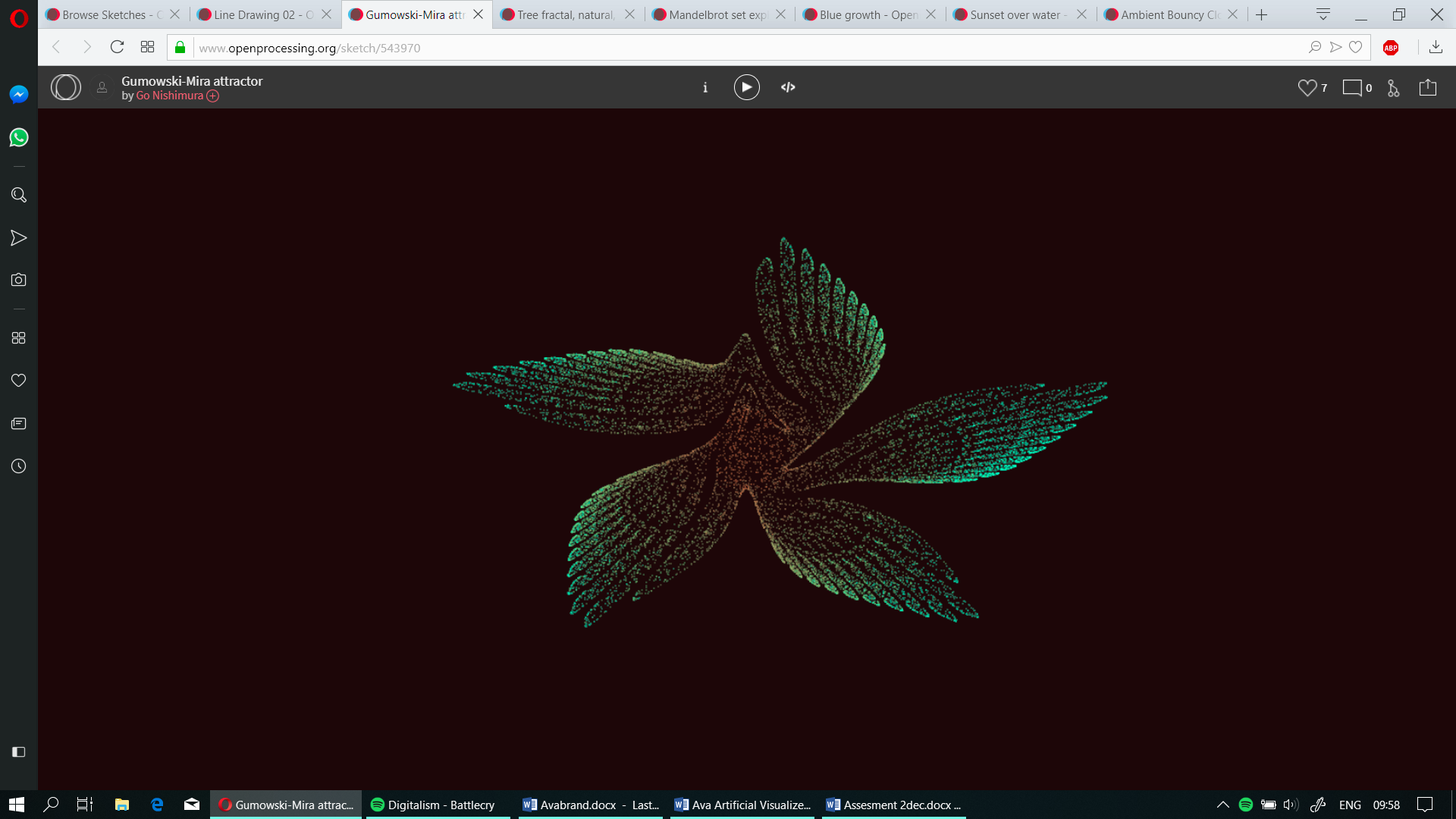
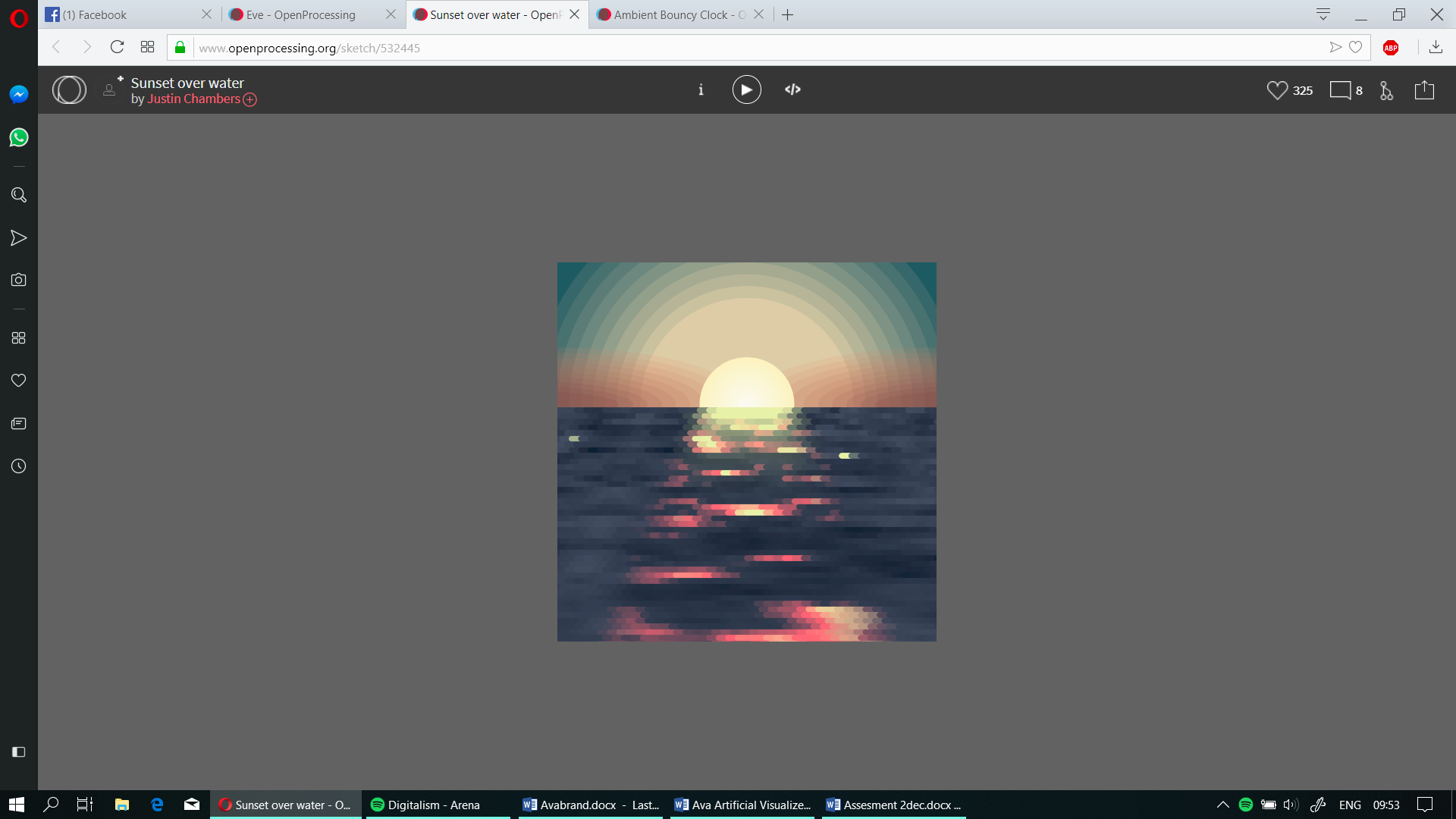
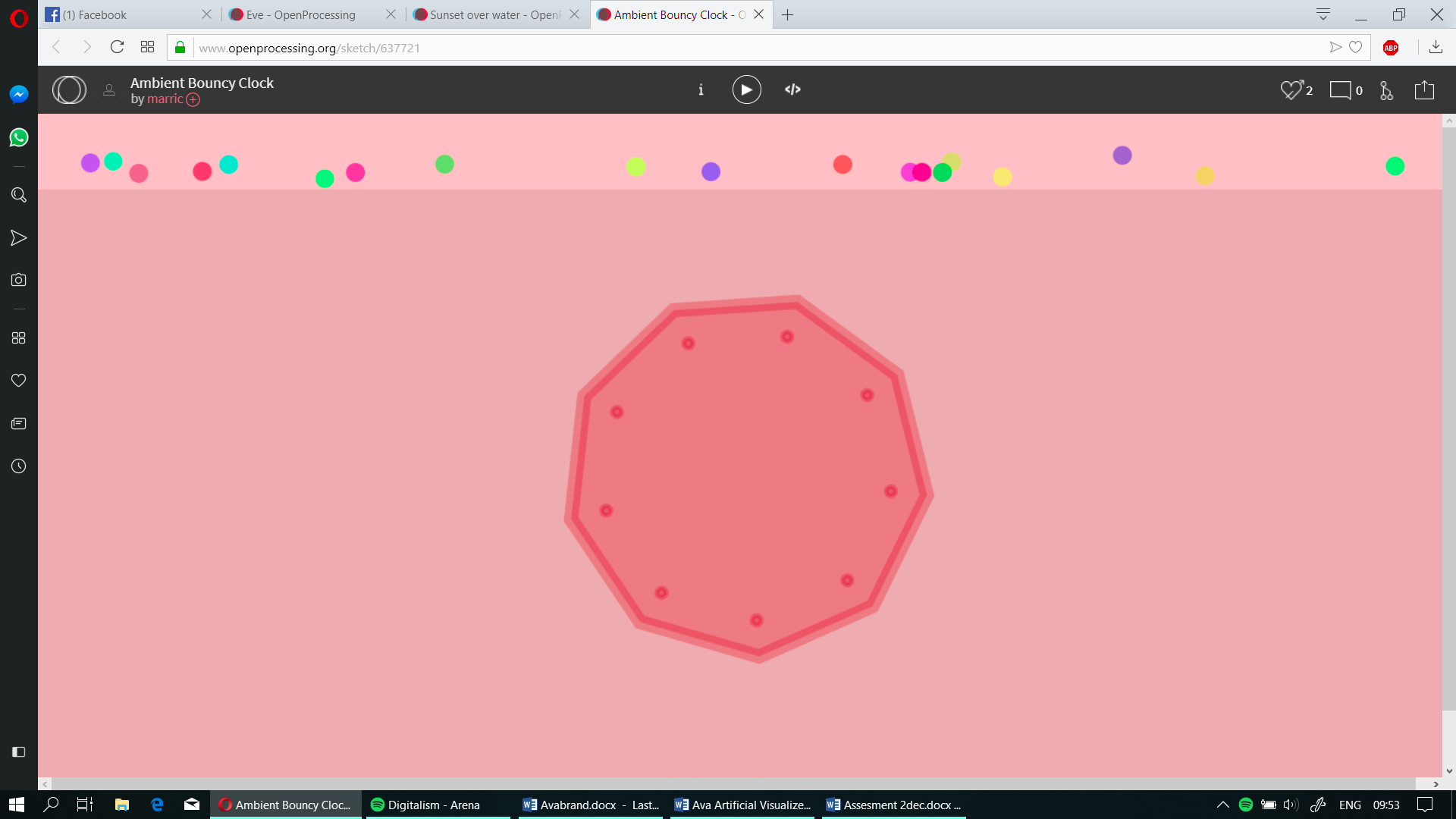
Algoritmische kunst

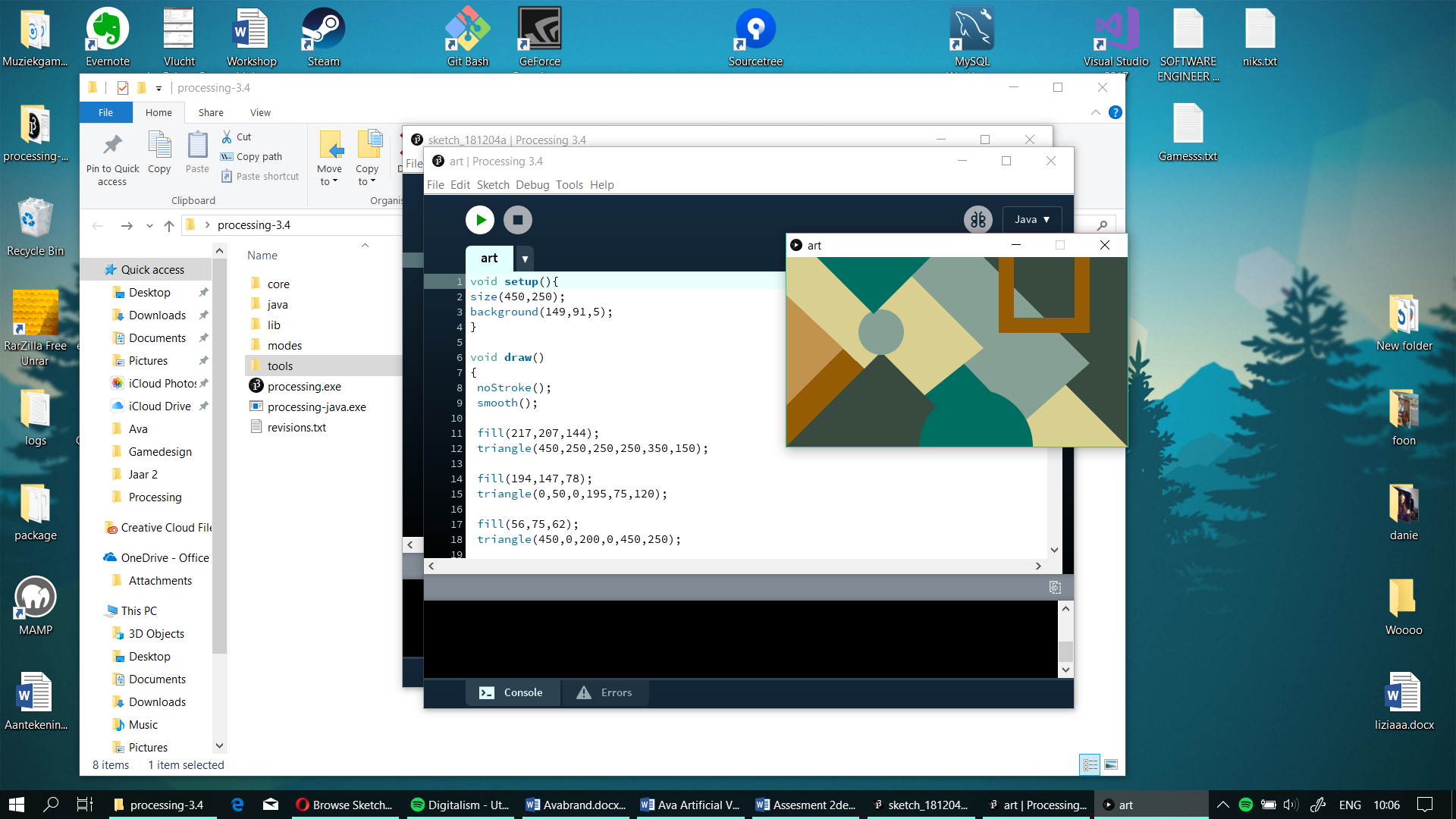
Uit Ava ben ik meer geïnteresseerd geraakt in algoritmische kunst. Het werd mij aangeraden om eens voorbeelden te zoeken van andere kunstenaren en ook gebruik te maken van de Processing applicatie. Voorheen heb ik met p5.js gewerkt. Het betere aan Processing is dat je meer kunt met hardware en de bugs beter worden weergegeven dan in p5.js (wat via de browser draait). Daarnaast wordt het ook meer gebruikt door andere ‘wiskundige’ kunstenaren.



Onderzoek

Ik ben gaan kijken op [www.openprocessing,org](http://www.openprocessing,org) naar voorbeelden van Processing / p5.js. De kunstwerken die boven te zien zijn spraken mij het meeste aan. Ik vind vooral de vormen heel erg mooi en sommige hebben ‘random’ functies, zijn op tijd gebaseerd of zijn interactief met de muis. Daarnaast heb ik ook de codes doorgelezen. Het valt me op dat er vooral gebruik wordt gemaakt van Arrays & for loopjes. Nu ga ik zelf ook wat creëren doormiddel van tutorials. De tutorials die op de site zijn te vinden van [www.processing.com](http://www.processing.com).

*Vormen*



Ik ben begonnen met het bekend worden van de verschillende vormen. Zo heb ik hier gebruikt gemaakt van quads, triangles, ellipse en de vertex shaper. Het verschil tussen de quads en een rect is dat je een vierkant schuin kunt laten gaan. Hierbij moet dan wel een wiskundige formule worden toegepast (Pythagoras) om de juiste locatie te vinden zodat de rechthoek nog steeds recht is. Dit heb ik niet netjes toegepast, maar vooral in me hoofd uitgerekend.

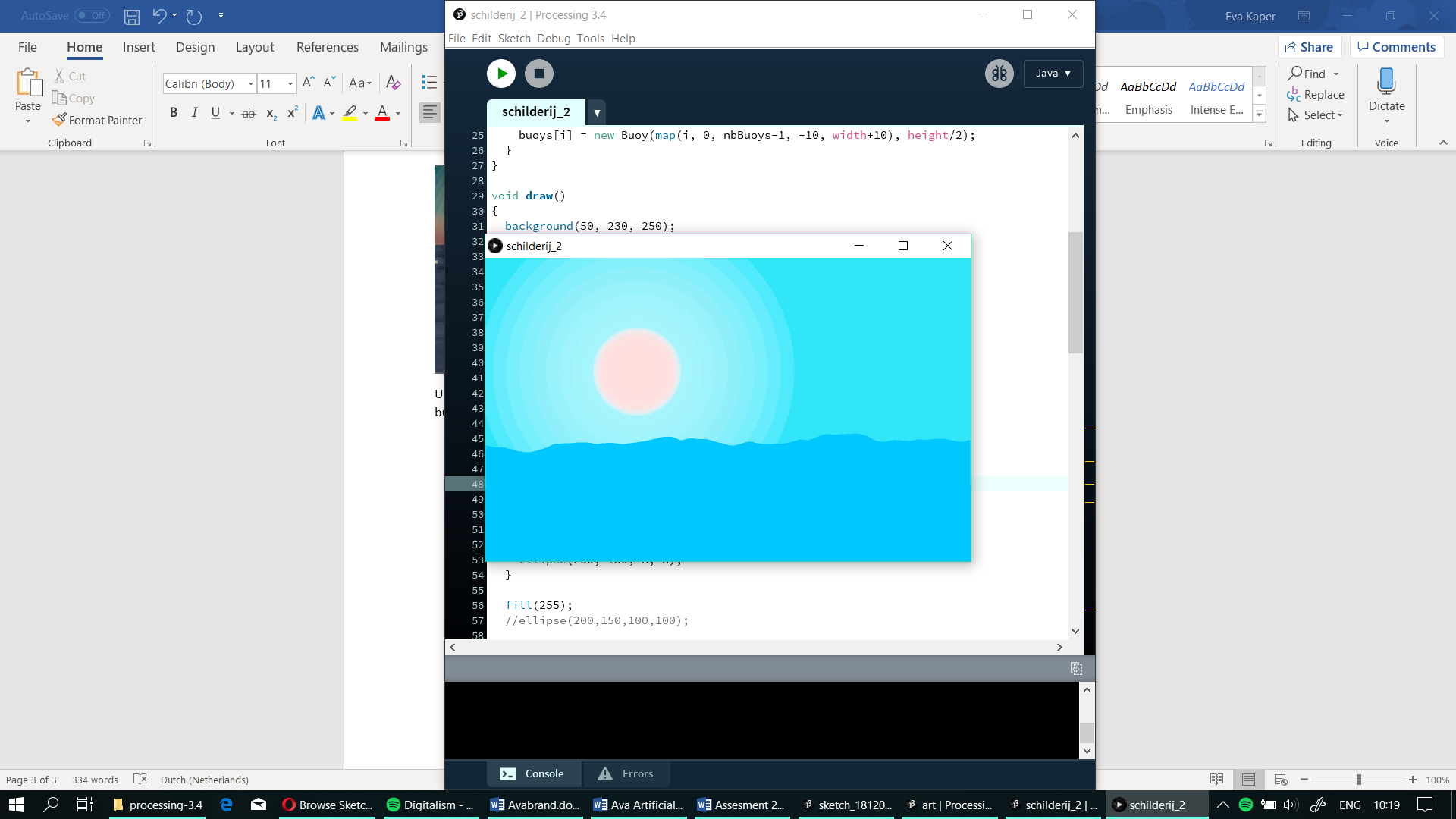
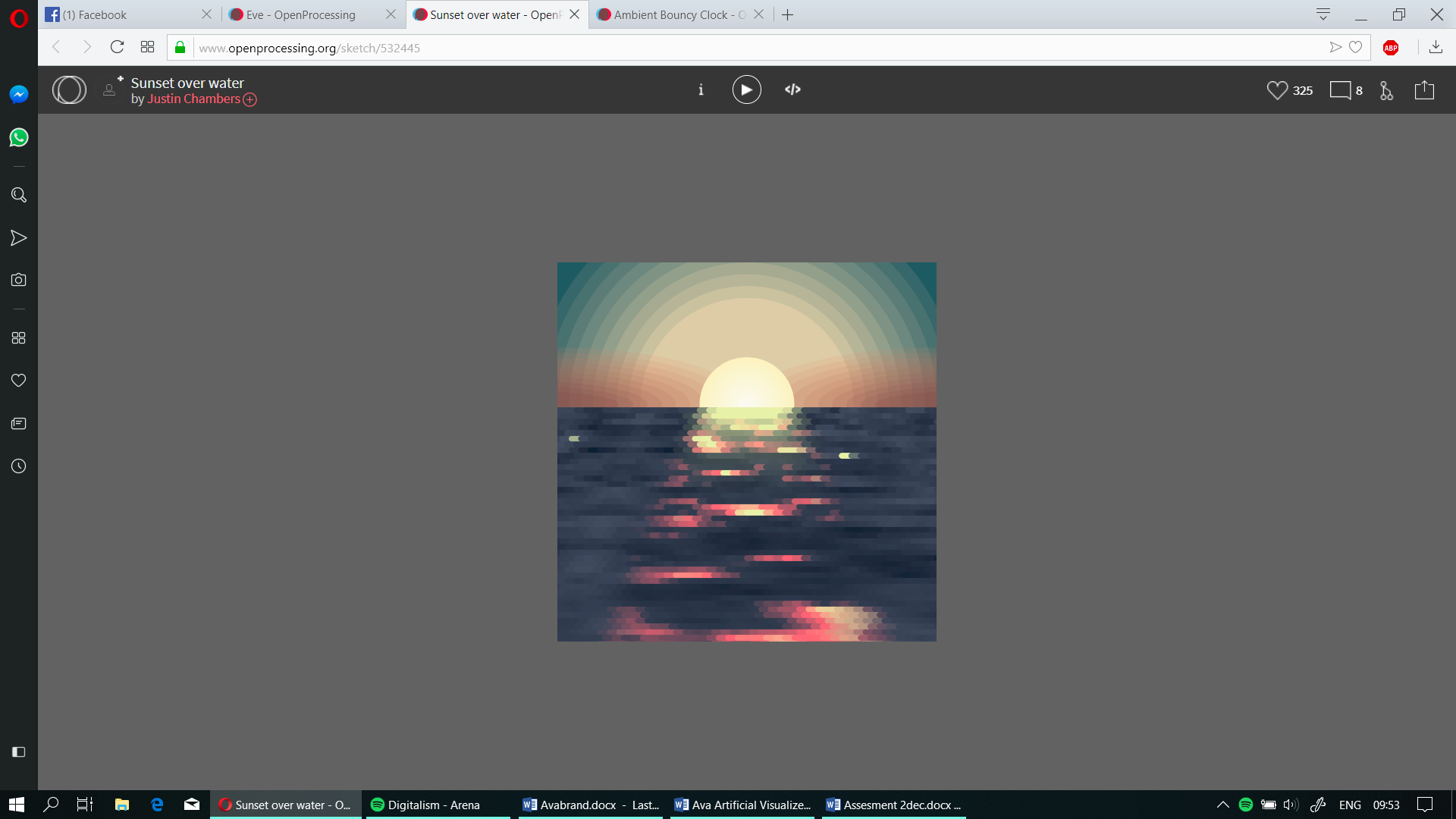
De code van een quad bevat (x1,y1,x2,y2,x3,y3)

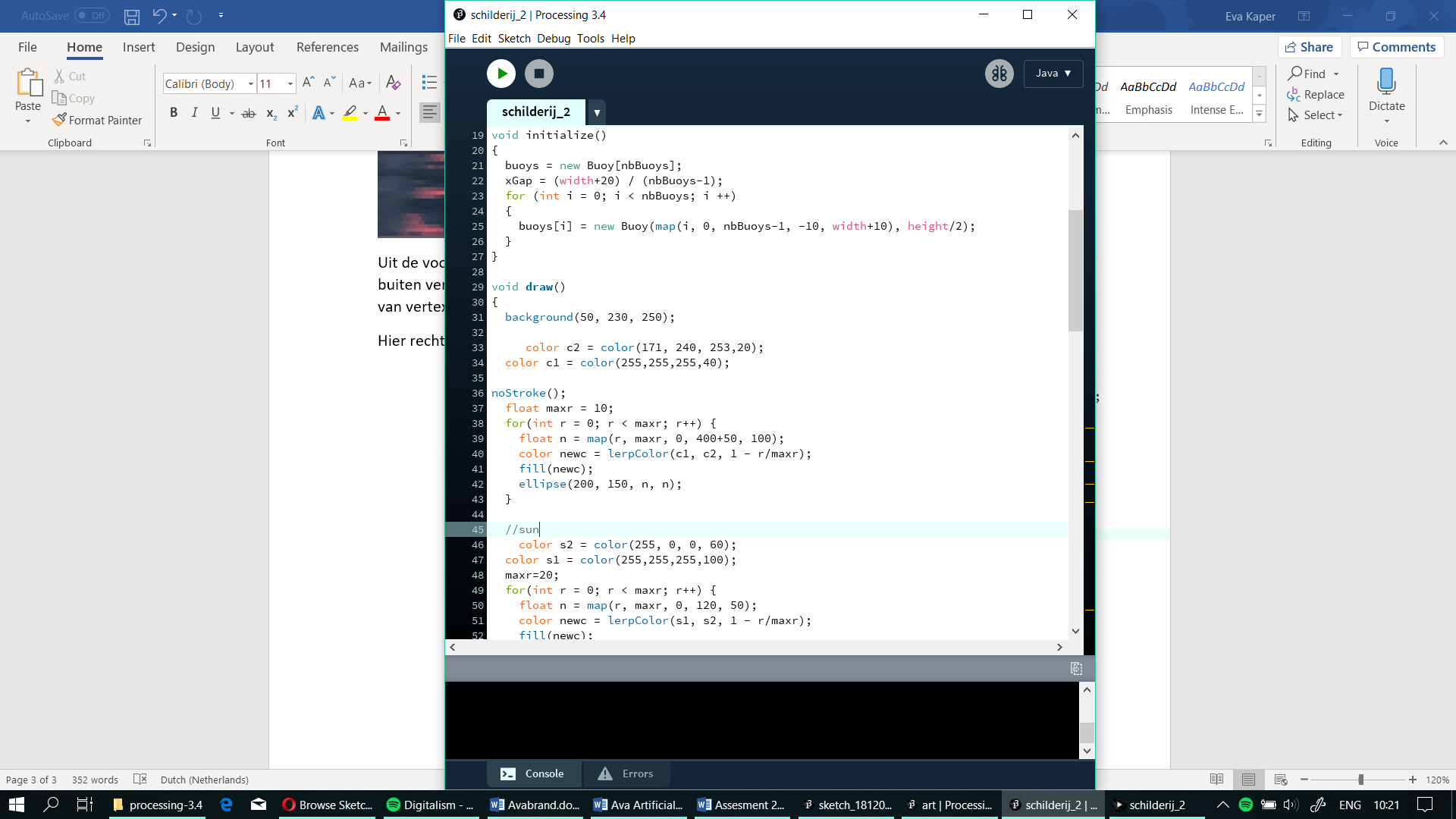
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | X=1 | X=2 | X=3 | X=4 | X=5 | X=6 | X=7 | X=8 | X=9 |
| Y=1 |  |  |  |  |  |  |  |  |  |
| Y=2 |  |  |  |  |  |  |  |  |  |
| Y=3 |  |  |  |  |  |  |  |  |  |
| Y=4 |  |  |  |  |  |  |  |  |  |
| Y=5 |  |  |  |  |  |  |  |  |  |

De x staat voor de x-as (horizontaal) en de y voor de y-as (verticaal). Alles wordt gemeten vanaf links boven. Dus als je (10,30) betekent dat de eerste punt gepositioneerd wordt op 10 naar rechts en 30 naar onderen.

Hier rechts is een voorbeeld te zien.

*Gradient & Vertex*

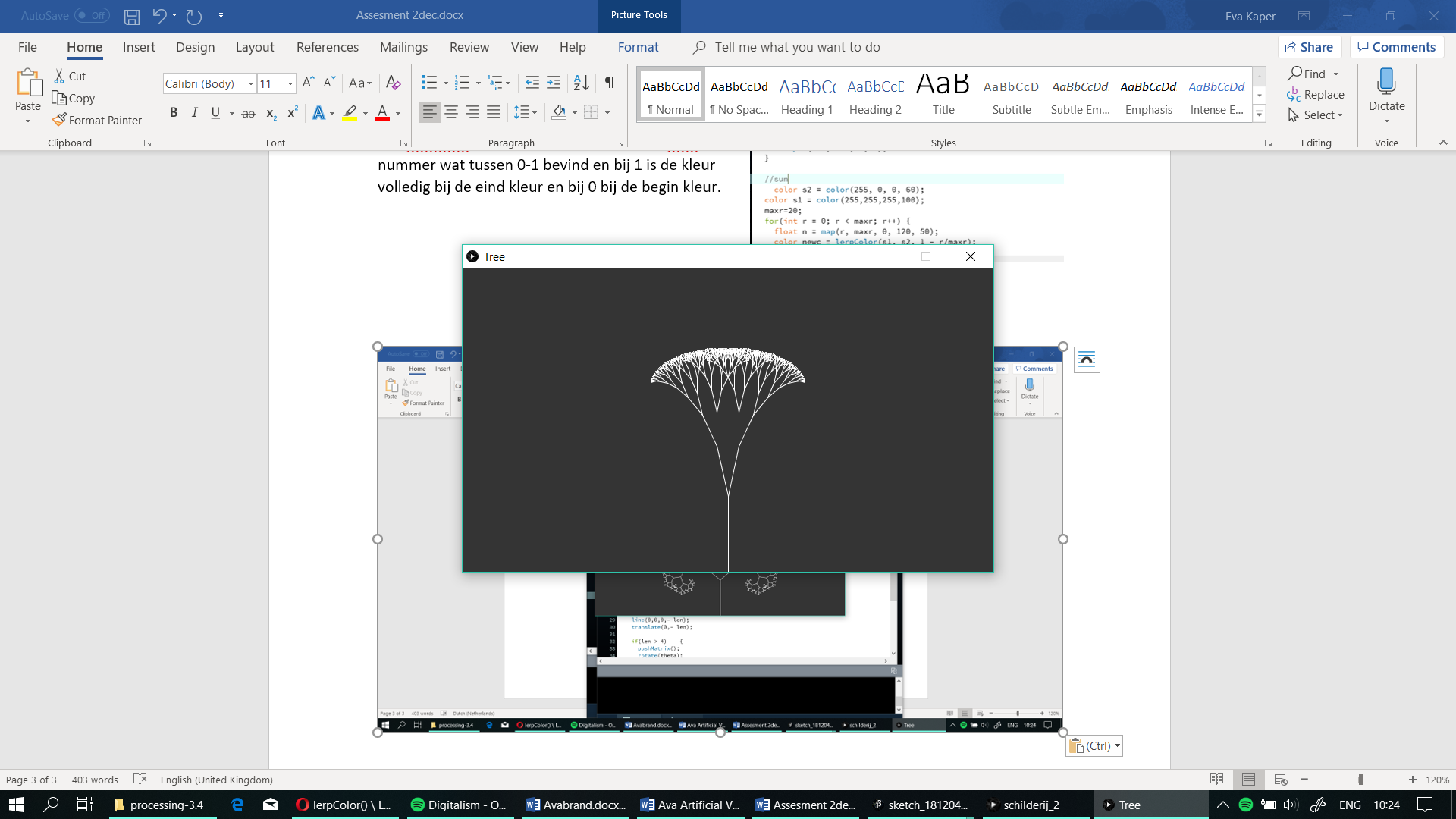
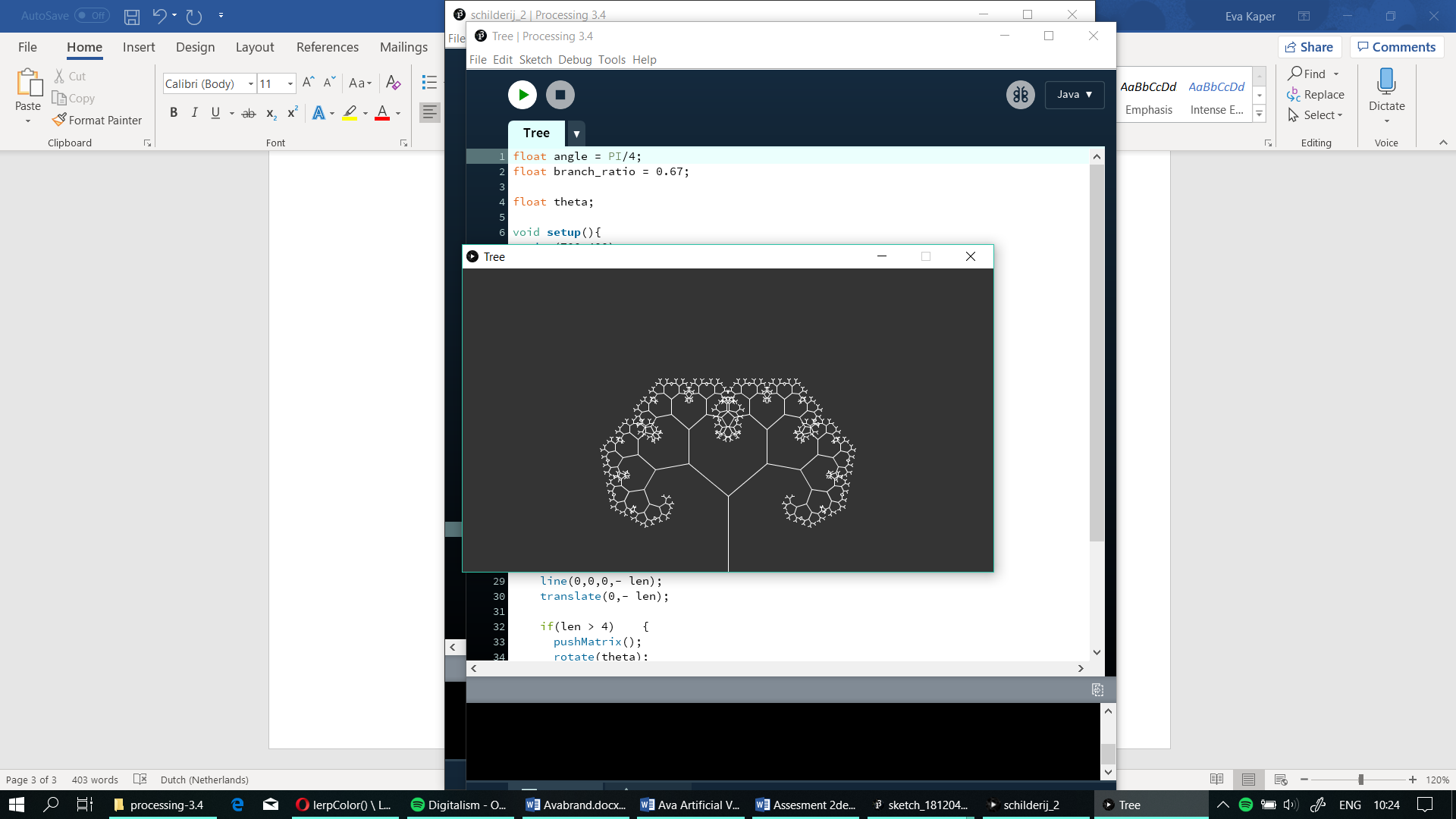
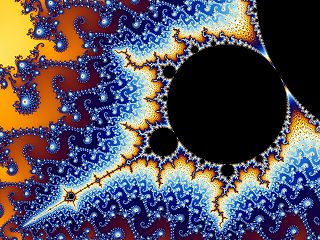


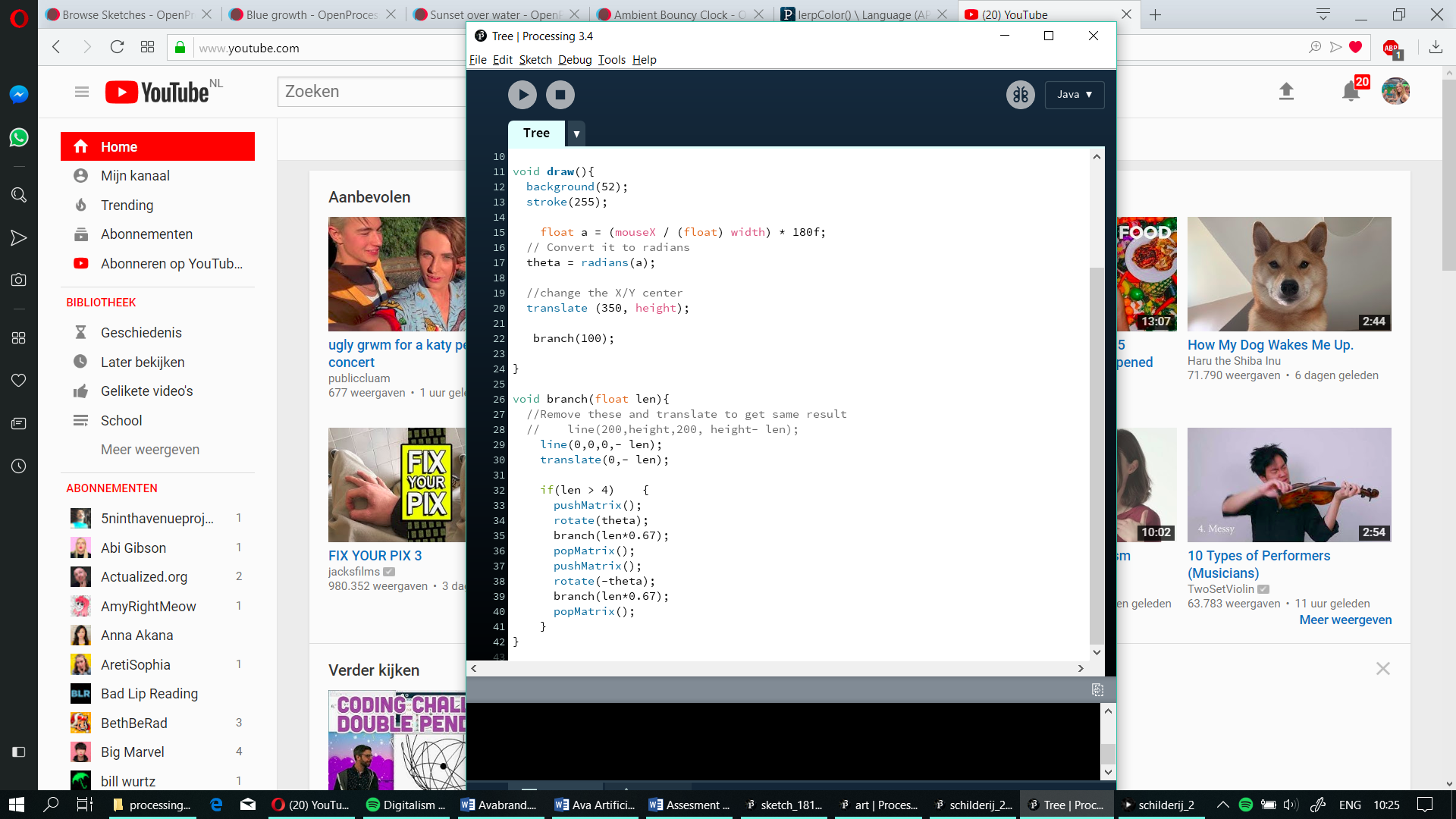
Uit de voorbeelden die ik had gevonden, vond ik een ontzettend mooi. De zon die pixel achtig naar buiten verkleurd. Het water heb ik gedaan doormiddel van vertex.

Hier rechts bevindt zich de code van de gradient kleur. Wat er wordt gedaan is dat er 2 kleuren zijn.

C1 is de begin kleur en C2 de eind kleur. Doormiddel van lerpColor(kleur1,kleur2,float); Pakt de float een nummer wat tussen 0-1 bevind en bij 1 is de kleur volledig bij de eind kleur en bij 0 bij de begin kleur. Dit wordt uitgetekend in de for-loop.

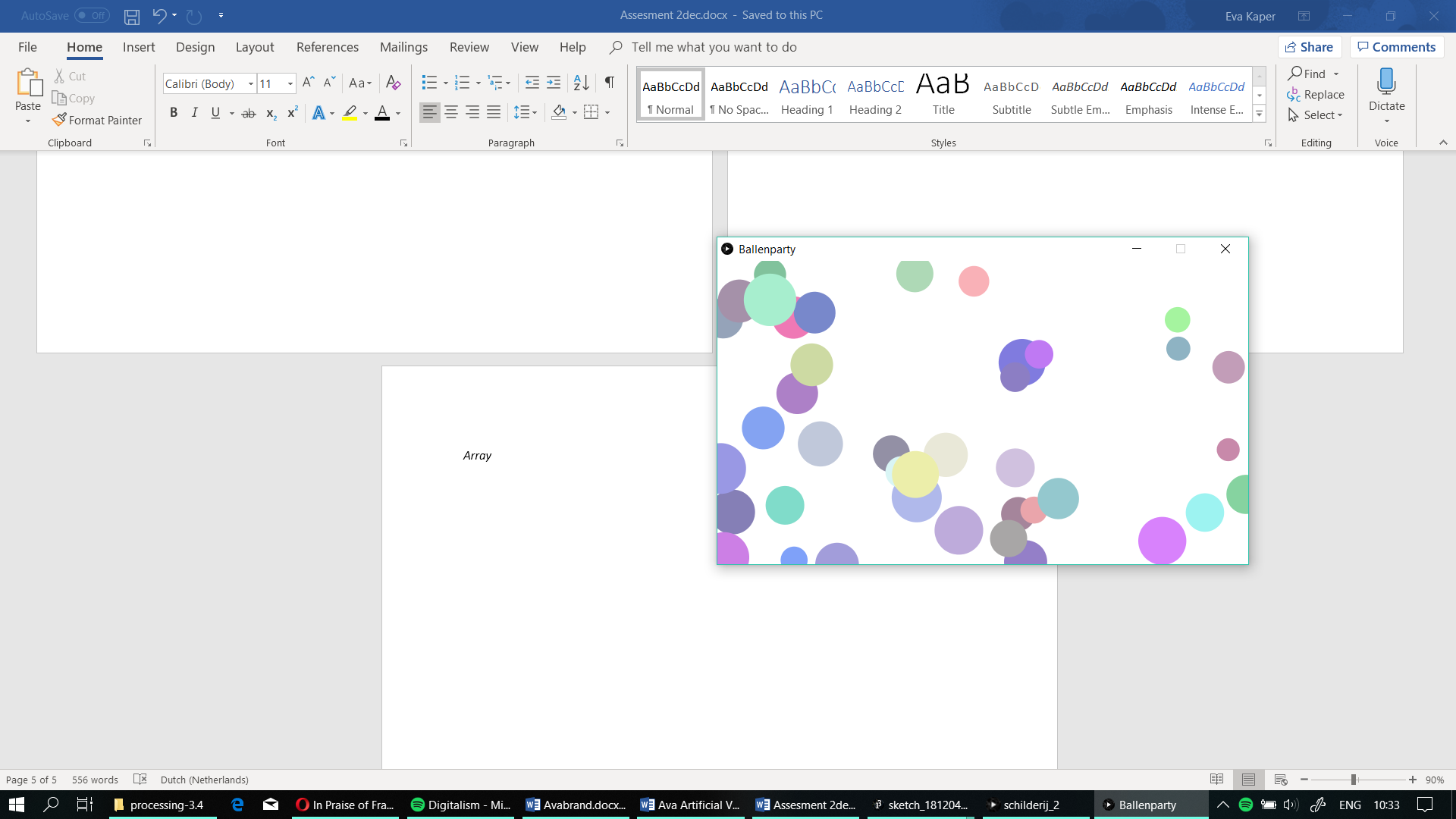
*Fractal Tree*

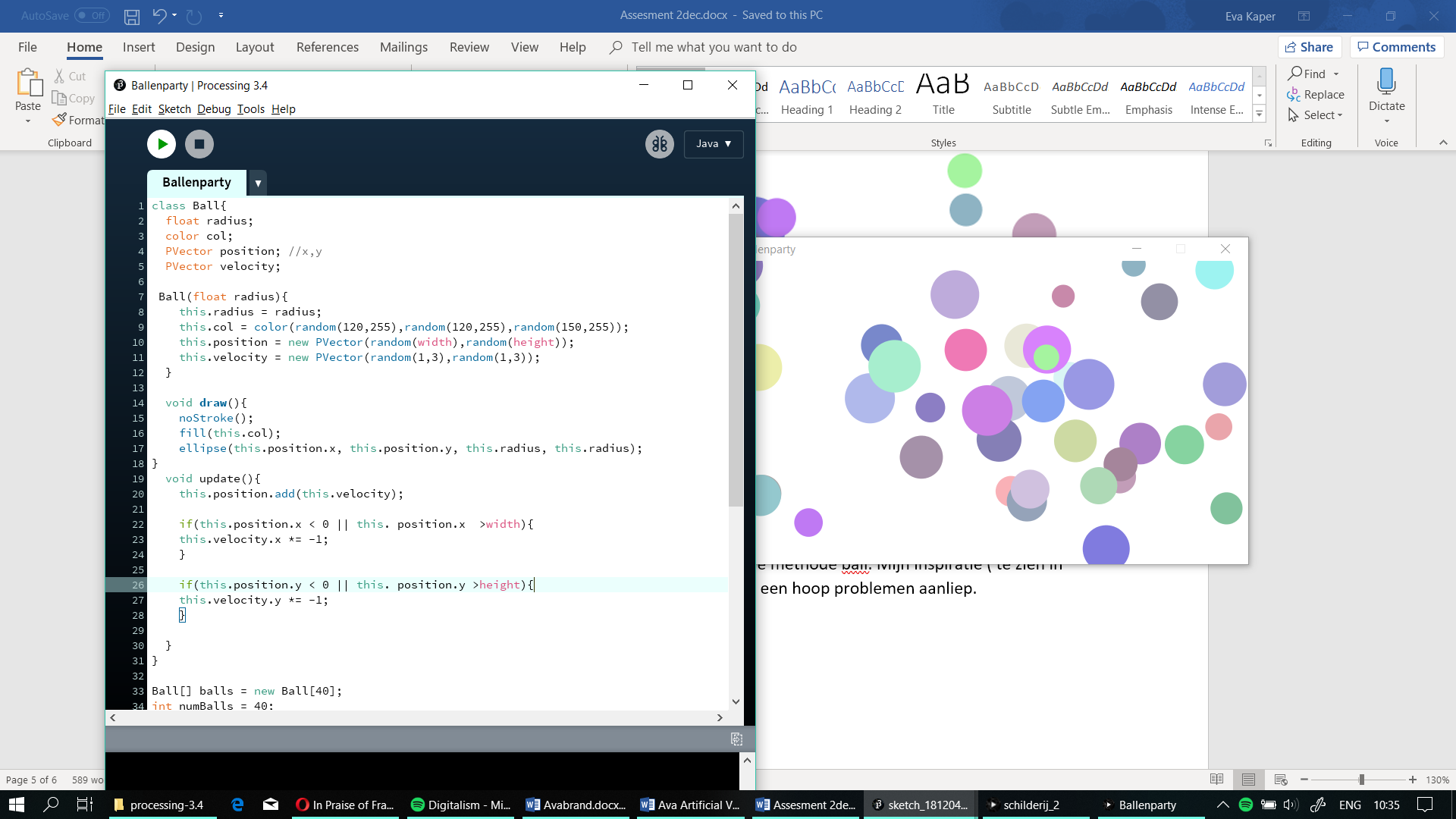
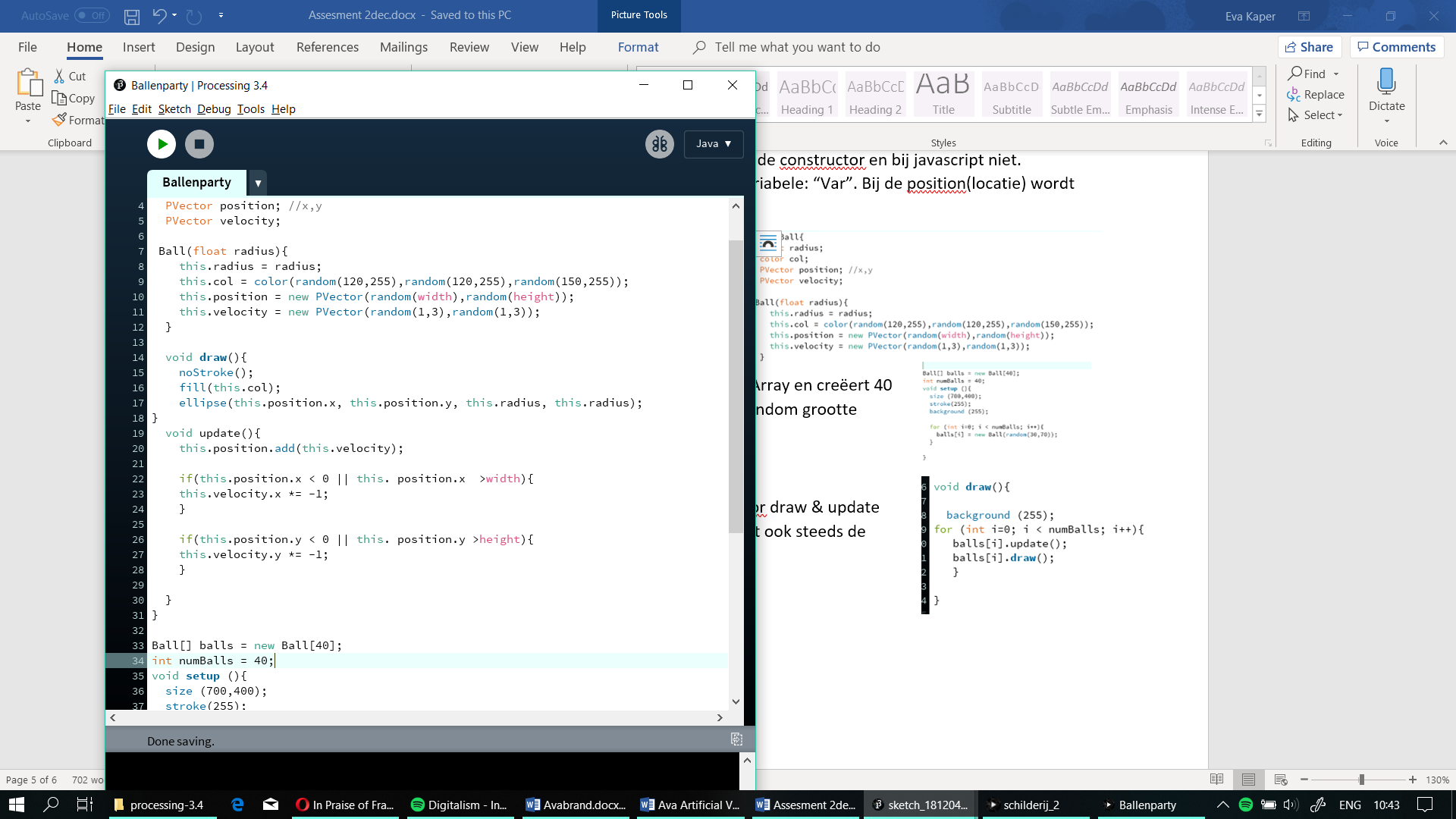


Doormiddel van de muis te bewegen, kun je de graden tussen de lines veranderen zodat de boom van vorm verandert.

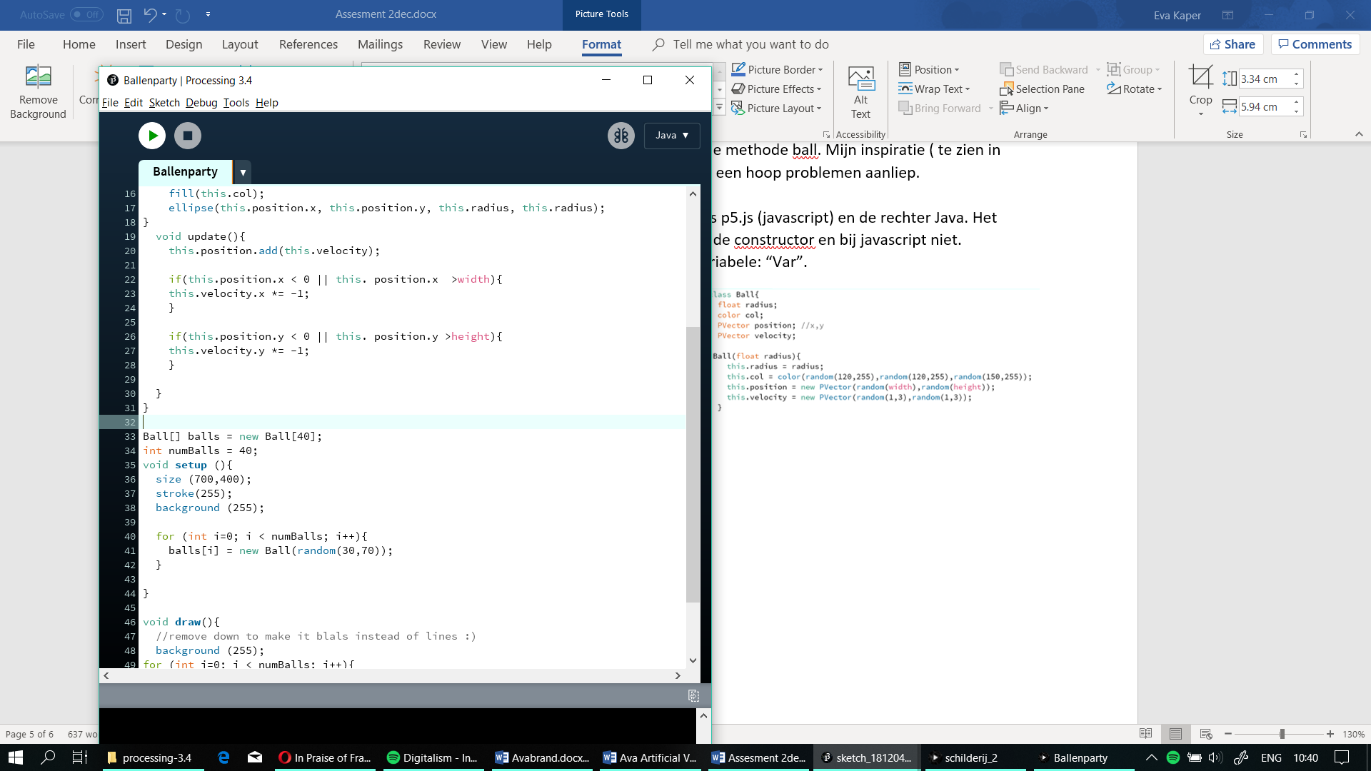
Een fractal betekend dat er uit een oorspronkelijk object steeds meer objecten ontstaan en steeds verder gaat tot in de oneindigheid. Rechts is een voorbeeld te zien, het meest beroemde fractal werk: Mandelbrot set. Ik heb de boom gemaakt doormiddel van een tutorial ( in de bronnen).

Wat er in de code wordt gedaan, is dat er een soort van loop wordt gedaan doormiddel van een if statement. In de draw() sectie wordt een methode opgeroepen en daarin wordt het begin punt weergeven (100). Deze 100 moet worden gebroken tot 4. De len wordt steeds per 0.67 vermenigvuldigd. Tot dit lager dan 4 is. Wanneer je de muis beweegt, zal de boom ook zijn graden tussen de takken veranderen.

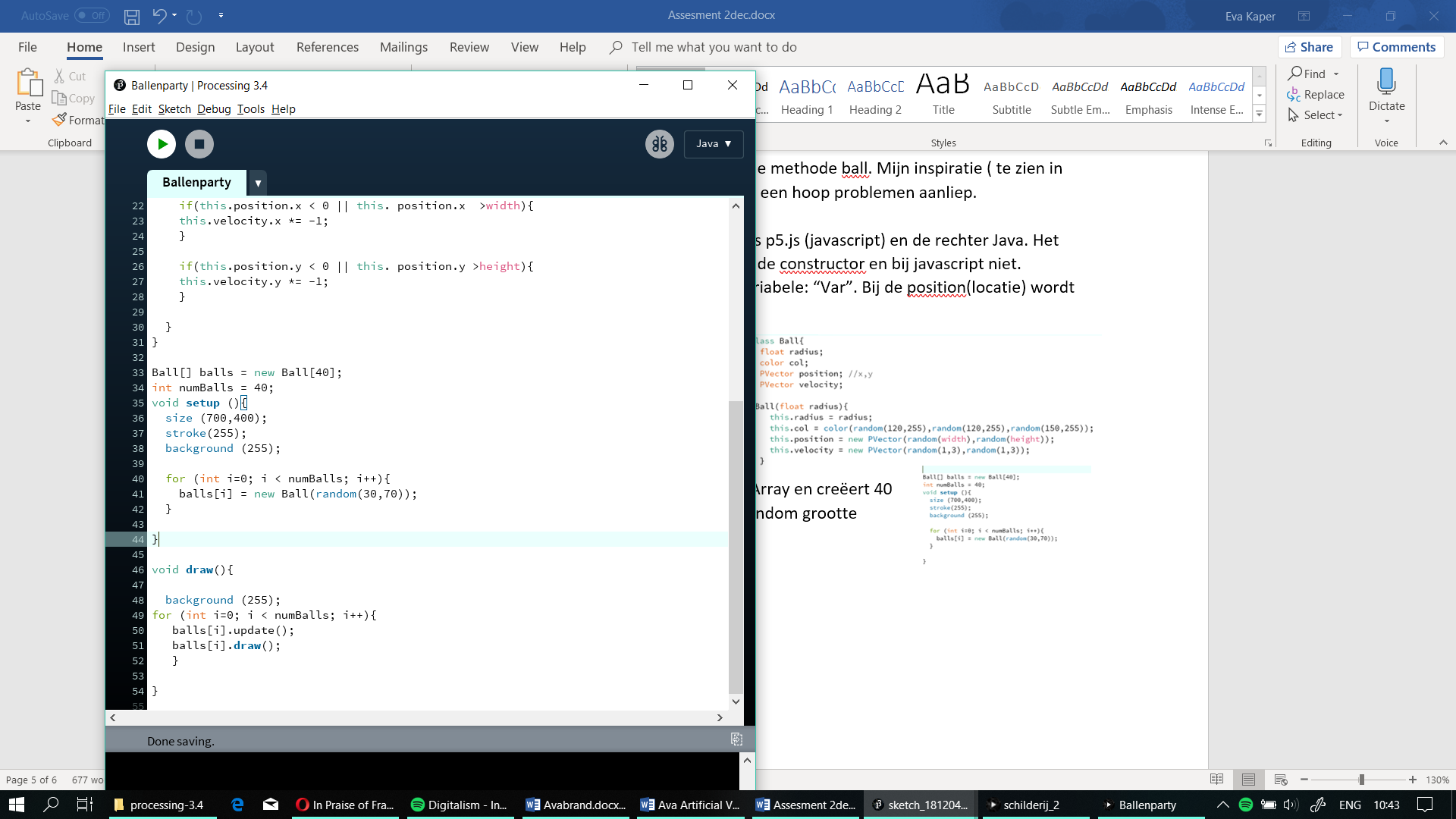
*Array*

Dit is een array van objecten die 50 ballen creëert via de methode ball. Mijn was gemaakt in p5.js waardoor ik tegen een hoop problemen aanliep.   
Hier zijn 2 klassen met een constructor er in. De linker is p5.js (javascript) en de rechter Java. Het verschil is dat ik bij Java de variabele moet aangeven in de constructor en bij javascript niet. Daarnaast kun je in Java geen gebruik maken van de variabele: “Var”. Bij de position(locatie) wordt er een random locatie & sneldheid(1-3) gegeven.

In de ball klassen bevind zich een draw methode voor de kleur (die random wordt gegeven). En de positie (ook random). Deze werden aangemaakt in de Ball constructor. In de update wordt de beweging gegeven zodat de ballen van de zijkant afweerkaatsen.



Bij de setup wordt de klassen Ball opgeroepen tot een Array en creëert 40 nieuwe ballen. Deze ballen krijgen in de for-loop een random grootte gegeven tussen de 30-70.



Bij de draw (buiten de ball klassen) wordt de constructor draw & update opgeroepen zodat ze blijven bewegen. Daarnaast wordt ook steeds de achtergrond opgeroepen.

Conclusie

Als conclusie om Ava verder uit te werken moet ik sowieso de code netter maken. Daarnaast om het meer interactief te maken wil ik met verschillende vormen gaan werken. Zo dacht ik om misschien lijntjes te creëren tussen de punten zodat er meer beweegt. Daarnaast ook de range van de frequentie aanpassen. Ik ga het waarschijnlijk ook verder uitwerken in Processing omdat ik het leuker vind. Ik ga sowieso verder met processing omdat het erg leuk is om er dingen mee te maken. Zo heb ik ook geprobeerd om eigen tekeningen er in te verwerken (gradient).

Bronnen

Fractal-tree

<https://www.youtube.com/watch?v=0jjeOYMjmDU>

Fractal

<https://blogs.scientificamerican.com/roots-of-unity/in-praise-of-fractals-and-poetry/>

Tijd gebaseerde

<https://www.openprocessing.org/sketch/637721>

Sunset

<https://www.openprocessing.org/sketch/532445>

Blue growth

<https://www.openprocessing.org/sketch/178381>

Gumowski

<https://www.openprocessing.org/sketch/543970>

Random tree fractal

<https://www.openprocessing.org/sketch/504377>

Lerp color

<https://processing.org/reference/lerpColor_.html>

Arrays

<https://processing.org/tutorials/arrays/>

Objects

<https://processing.org/tutorials/objects/>