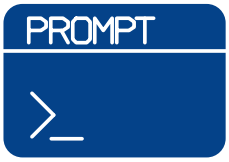
VIBE CODING



p5.js, VS Code & Copilot Free

Inhoud

1. **Intro**
    1. 1. Wat is een LLM?
    1. 2. Waar is een LLM goed in?
    1. 3. Mogelijke obstakels
    1. 4. Je eigen koers behouden
2. **De omgeving opzetten**
    2. 1. Stap 1: VS Code installeren
    2. 2. Stap 2: Copilot Free activeren
    2. 3. Stap 3: Live Server installeren
    2. 4. Stap 4: Een lege p5.js-sketch
    2. 5. Stap 5: Een vraag over je eigen code
    2. 6. Optioneel: regels voor het hele project
3. **Prompten**
    3. 1. Een goede prompt schrijven
    3. 2. Werken met de Agent
    3. 3. Stap voor stap werken
    3. 4. Als het misgaat
4. **Oefening 1: reCode Molnár, Nake, Nees of Noll**
    4. 1. Spelregels
    4. 2. Stappenplan
    4. 3. Reflectie
5. **Oefening 2: Eigen programma**
    5. 1. Casus A: Plottertekening
    5. 2. Casus B: interactief beeld en geluid
6. **Reflectie & verder**
    6. 1. Terugblik
    6. 2. Cheat sheet: prompts
    6. 3. Kritische kanttekeningen
    6. 4. Links

1. Intro

Copilot, ChatGPT, Claude en Gemini kunnen code schrijven, uitleggen en verbeteren op basis van gewone taal. Zo wordt de afstand tussen een **idee en een eerste programma of algoritmische tekening** een stuk kleiner.

Een Large Language Model begrijpt je idee niet zoals een mens dat doorgaans doet, maar voorspelt welke code bij je instructies past. Bij p5.js werkt dat vaak goed omdat er online veel voorbeelden te vinden zijn, maar het resultaat is al snel een vrij gemiddelde sketch. De uitdaging is dus om zelf voldoende richting te geven, kritisch te blijven en bij te sturen.

Een goede prompt is eigenlijk ook een **reeks instructies**. Denk bijvoorbeeld aan de *Wall Drawings* van Sol LeWitt: een idee wordt vertaald naar een set regels die iemand anders kan uitvoeren.

In deze workshop bouwen we verder op de basis van p5.js (setup(), draw(), vorm en kleur, variabelen en loops) en onderzoeken we hoe je een LLM als sparringpartner inzet, **zonder zelf de controle over je artistieke keuzes te verliezen**.

1.1. Wat is een LLM?

Een **Large Language Model (LLM)** is een AI-model dat getraind is op enorme hoeveelheden tekst, waaronder code. Het voorspelt telkens welk woord of stukje tekst waarschijnlijk **volgt** op wat jij schrijft. Dat klinkt simpel, maar het levert verrassend bruikbare code op.

Stel je een razendsnelle assistent voor die **alle p5.js-voorbeelden op het internet** gelezen heeft, maar zelf niets uitprobeert en altijd heel zeker klinkt.

1.2. Waar is een LLM goed in?

- Een eerste versie van een idee snel opzetten.
- Uitleggen wat een regel code doet.
- Foutmeldingen vertalen naar begrijpelijke taal.
- Variaties en alternatieven voorstellen.
- Code opschonen en herstructureren.

1.3. Mogelijke obstakels

- Het verzint soms functies die niet bestaan, de zogenaamde *hallucinaties*.
- Het kiest vaak voor het meest voorspelbare resultaat.
- Het haalt soms p5.js 1.x en 2.x door elkaar. Vandaag zijn de meeste voorbeelden die online staan nog geschreven voor versie 1.x.
- Het klinkt ook overtuigd wanneer het fout zit.
- Het kan niet altijd zien wat er op jouw scherm gebeurt. Sommige tools, zoals Copilot in VS Code, kunnen inmiddels wel zelfstandig een browserpagina openen, een screenshot maken en het resultaat visueel controleren.
- Een agent kan meer aanpassen dan je gevraagd hebt (andere bestanden, andere functies). Controleer wat er wijzigt.

1.4. Je eigen koers behouden

Het blijft belangrijk om je eigen keuzes bewaken. Houd daarom deze drie regels in je achterhoofd:

1. **Run it.** Voer de code uit voordat je een stap verder gaat.
2. **Read it.** Lees de code, ook als je niet alles meteen begrijpt. Ga de wijzigingen na (de *diff*) en elk commando dat hij wil uitvoeren.
3. **Learn one thing.** Probeer bij elk antwoord minstens één nieuwe regel of functie te begrijpen. Vraag de LLM om uitleg als iets niet duidelijk is.

2. De omgeving opzetten

We werken met gratis tools. Je hebt nodig:

1. Visual Studio Code (VS Code)
2. een gratis GitHub-account (voor Copilot Free)
3. de VS Code-extensie Live Server
4. een browser (Chrome, Firefox, Edge of Safari)

> **Tip**
> Als je liever geen tijd besteedt aan het opzetten van een eigen werkomgeving, kan je ook meteen aan de slag via p5js.ai [1].

2.1. Stap 1: VS Code installeren

👉 https://code.visualstudio.com/

Download, installeer en start VS Code.

2.2. Stap 2: Copilot Free activeren

Maak eerst een gratis account aan op github.com [2] als je er nog geen hebt. Open daarna in VS Code de **Chat** in het **View**-menu **> Chat**, of gebruik **Ctrl+Alt+I** (Mac: ⇧⌘**I**). Meld je aan met je GitHub-account en kies de gratis versie. Lukt het niet? Zoek in de Command Palette op *Copilot*.

> **Note**
> **Het budget is beperkt.** Copilot Free bevat een beperkt aantal chatberichten per maand. Prompt dus bewust: één doordachte prompt is beter dan vijf slordige.
> **Raakt je budget op?** Gebruik dezelfde prompts in een gratis chat in de browser (Copilot op github.com, ChatGPT, Claude of Gemini) en kopieer de code zelf naar VS Code.

👉 https://github.com/features/copilot/plans

2.3. Stap 3: Live Server installeren

Open het tabblad **Extensions** (**Ctrl+Shift+X**, Mac: ⇧⌘**X**), zoek op *Live Server* (van Ritwick Dey) en klik op **Install**. Daarna verschijnt onderaan rechts een knop **Go Live**. Live Server toont je sketch in de browser en **herlaadt automatisch** telkens je een bestand opslaat.

👉 https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer

2.4. Stap 4: Een lege p5.js-sketch

In de p5.js web editor stond alles al klaar. Hier laten we Copilot dezelfde drie bestanden opzetten, zoals beschreven in p5.js-anatomie: HTML, CSS & JS.

- Maak op je computer een map en kies in VS Code **File > Open Folder**.
- Open de Copilot-chat en typ iets als dit:

> **PROMPT**
> Maak in de huidige map een minimale, werkende boilerplate voor p5.js 2.x. Gebruik de laatste stabiele versie in global mode. Maak precies drie bestanden aan: index.html, style.css, sketch.js. Zorg voor de correcte koppeling tussen de drie bestanden. Houd de code minimaal, voeg geen extra libraries of functionaliteiten toe. Het canvas in p5.js is 600 bij 600 pixels groot.

Vermoedelijk moet je een paar keer op **Allow** drukken en/of krijg je een gerichte vraag om te beantwoorden.

- Controleer of `index.html`, `style.css` en `sketch.js` zijn aangemaakt.
- Rechtsklik op `index.html` en kies **Open with Live Server** (of klik op **Go Live**).

Als alles goed gaat, zie je een grijs canvas. Voeg eventueel deze regel toe in `draw()`, onder `background(220)`, om iets zichtbaars en interactiefs te hebben:

```
circle(mouseX, mouseY, 30);
```

2.5. Stap 5: Een vraag over je eigen code

Voeg `sketch.js` toe als context (sleep het bestand in de chat, gebruik **Add Context** of typ `#` en kies het bestand) en stel dan deze vraag:

> **PROMPT**
> Wat doet background(220) in sketch.js? Leg het uit in twee zinnen en geef een voorbeeld met een andere waarde.

Krijg je een antwoord? Dan is de omgeving klaar. 🎉

2.6. Optioneel: regels voor het hele project

VS Code kan vaste projectinstructies meegeven aan GitHub Copilot. Zo hoef je bijvoorbeeld niet telkens opnieuw te vermelden welke p5.js-versie je gebruikt of dat je graag Nederlandstalige comments wilt toegevoegd zien.

Maak hiervoor in de hoofdmap van je project een map `.github` aan en voeg daarin het bestand `copilot-instructions.md` toe. Plaats daarin de onderstaande regels:

```
# Projectregels
- Gebruik p5.js in global mode (geen instance mode).
- Gebruik de syntax van p5.js 2.x. Gebruik geen functies of voorbeelden uit p5.js 1.x die niet compatibel zijn met versie 2.x.
- Schrijf de code in sketch.js. Pas andere bestanden enkel aan als ik daar uitdrukkelijk om vraag.
- Voeg geen extra libraries toe, behalve als ik er expliciet om vraag.
- Verwijder of hernoem geen bestanden zonder het eerst te vragen.
- Schrijf korte comments in het Nederlands.
- Vermeld bij elke wijziging kort wat je hebt aangepast.
```

3. Prompten

3.1. Een goede prompt schrijven

Taal: Nederlands, Engels of een mix werkt allemaal. Belangrijker dan de taal is dat je **concreet** bent en de **juiste woorden** gebruikt.

De opbouw van een prompt

Ingrediënt	Vraag	Voorbeeld
Context	Waar werk ik mee?	p5.js 2.x, global mode, sketch.js, canvas 600×600
Doel	Wat moet er te zien, te horen of te doen zijn?	Een raster van cirkels waarvan de grootte reageert op de muis
Regels	Wat mag wel of niet?	Enkel lijnen, geen fill, maximaal 3 kleuren, geen libraries
Antwoord	Hoe wil ik het antwoord?	Volledige sketch.js, korte Nederlandse comments, één wijziging tegelijk

Hoe duidelijker de regels, hoe bruikbaarder het resultaat. Dat lijkt op de regels uit de computertekening met potlood.

Slechte versus betere prompt

PROMPT:

Minder goed: Maak iets cools met cirkels waarbij de vibe goed zit ;-)

PROMPT:

Beter: Ik gebruik p5.js 2.x (global mode) in sketch.js. Het canvas is 600×600 met een zwarte achtergrond.
Teken een raster van 10×10 cirkels met geneste for-loops. De diameter van elke cirkel hangt af van de afstand tot mouseX en mouseY (gebruik dist() en map()). Alleen stroke, geen fill.
Geef de volledige sketch.js, met korte Nederlandse comments.

Terminologie gebruiken

Beschrijf je een **beeld**, vertaal het dan naar een **term**. De p5.js Reference helpt je daarbij.

Wat je wilt zeggen	Term of functie
willekeurig	`random()`
zachte, organische willekeur	`noise()` (Perlin noise)
een waarde herschalen	`map()`
afstand tussen twee punten	`dist()`
geleidelijk naar iets toe bewegen	`lerp()`
golven, schommelen, cirkelbeweging	`sin()` en `cos()`
herhalen in een raster	nested for loops
verplaatsen en draaien, zonder de rest te storen	`translate()`, `rotate()`, `push()` en `pop()`
een eigen vorm of pad tekenen	`beginShape()`, `vertex()`, `endShape()`
reageren op muis of toets	`mouseX`, `mouseY`, `mousePressed()`, `keyPressed()`

👉 https://p5.js.org/reference/

> **Tip**
> Je mag ook verwijzen naar kunstenaars en stijlen, zoals *"in de geest van Vera Molnár"* of *"strak en geometrisch, zoals een Bauhaus-affiche"*.

3.2. Werken met de Agent

In VS Code werken we met de **Agent-modus**. Je geeft een opdracht en de agent zoekt de juiste bestanden, past ze rechtstreeks aan, doet meerdere stappen na elkaar en vraagt toestemming voor acties zoals terminalcommando's. **Jij** beoordeelt het resultaat.

Wil je eerst een plan of uitleg? Zeg het in je prompt: *"Leg eerst uit wat je gaat doen, nog niets aanpassen."*

Wijzigingen beoordelen

Open het aangepaste bestand: je ziet de wijzigingen als *diff* in de editor.

- **Keep** behoudt een wijziging (per stukje, per bestand of alles tegelijk).
- **Undo** draait ze terug.
- **Lees de diff voor je op *Keep* klikt.** Heeft de agent meer aangepast dan je vroeg? Zet dat terug met Undo en herhaal je opdracht strenger: *"Wijzig enkel ..."*.

Iets misgelopen? Gebruik Undo, **Ctrl+Z** (Mac: **⌘Z**) of de *checkpoints* in de chat, waarmee je terugkeert naar een eerder punt.

Toestemming voor commando's

Voor een terminalcommando vraagt VS Code eerst toestemming. Je kan het toestaan, weigeren of aanpassen. **Lees elk commando voor je het toestaat.** Weet je niet waarom het nodig is, vraag dan: *"Waarom heb je dit commando nodig, en kan het ook zonder?"*

In deze workshop heb je **geen terminalcommando's nodig**: onze sketch bestaat uit gewone HTML-, CSS- en JavaScript-bestanden. Weiger dus gerust, en wees extra alert bij:

Commando of actie	Waarom opletten?
`npm install …`, `pip install …`	Installeert code van het internet. Niet nodig hier.
`rm`, `del`, `Remove-Item`	Verwijdert bestanden, mogelijk meer dan bedoeld.
`sudo` of beheerdersrechten	Geeft volledige controle over je computer.
`curl … | sh` of een gedownload bestand uitvoeren	Voert onbekende code uit.
Paden buiten je projectmap (`..`, `~`, `C:\`)	De agent hoort enkel in je projectmap te werken.

Het **permissieniveau** onder het invoerveld bepaalt hoeveel de agent zonder te vragen mag doen. Laat het op **Manual Permissions** staan (de standaard).

> **Caution**
> **Assisted Permissions** laat de agent zelf het risico inschatten.
> **Allow All** slaat alle bevestigingen over.

Veilig werken

- Loopt de agent de verkeerde kant op? Klik op de **stopknop**, draai de wijzigingen terug met Undo en geef een kleinere, scherpere opdracht.
- Open in VS Code enkel je **projectmap**, niet je hele Documenten-map.

> **Warning**
> **Privacy.** Alles wat je in de chat plakt of als context meegeeft, wordt naar de AI-dienst gestuurd. Plak nooit wachtwoorden, API-sleutels of persoonlijke gegevens.

3.3. Stap voor stap werken

1. **Plan.** Vraag eerst een plan (Plan-modus, of: *"Geef eerst een plan en pas nog niets aan"*) en controleer welke bestanden de agent wil aanpassen.
2. **Uitvoeren.** Vraag **één ding tegelijk**: eerst het raster, dan de kleur, dan de beweging.
3. **Lezen.** Lees de diff en kies Keep of Undo.
4. **Testen.** Sla op, bekijk de sketch en controleer of het resultaat nog bij je bedoeling past.
5. **Bewaren.** Werkt het? Maak een *commit* voor je aan de volgende stap begint.

> **Tip**
> **Een echt vangnet:** open **View > Source Control** en kies **Initialize Repository**. Na elke werkende stap typ je een korte boodschap en klik je op **Commit**. De **Timeline** onderaan in de Explorer bewaart ook oudere versies.

Handige extra's

- Wil je iets anders proberen? Vraag: *"Geef me drie verschillende aanpakken, kort uitgelegd."*
- Snap je een stuk code niet? Selecteer het en vraag: *"Leg deze regel uit, ik ken alleen de basis van p5.js."*
- Start voor elke nieuwe opdracht een **nieuw gesprek**. In een lang gesprek raakt de context vol en verliest de LLM de draad.

3.4. Als het misgaat

Geef de LLM dezelfde informatie als aan een persoon: wat verwachtte je, wat zie je, wat zegt de foutmelding? Die vind je in de browser (**F12**, tabblad **Console**).

> **PROMPT:**
> Ik verwachtte dat de cirkels groter worden als ik de muis naar rechts beweeg, maar ze blijven even groot.
> Foutmelding in de console: (plak hier de tekst).
> Hier is mijn code: (plak of voeg sketch.js toe). Wat gaat er fout en wat is de kleinste aanpassing om het op te lossen?

Veelvoorkomende problemen

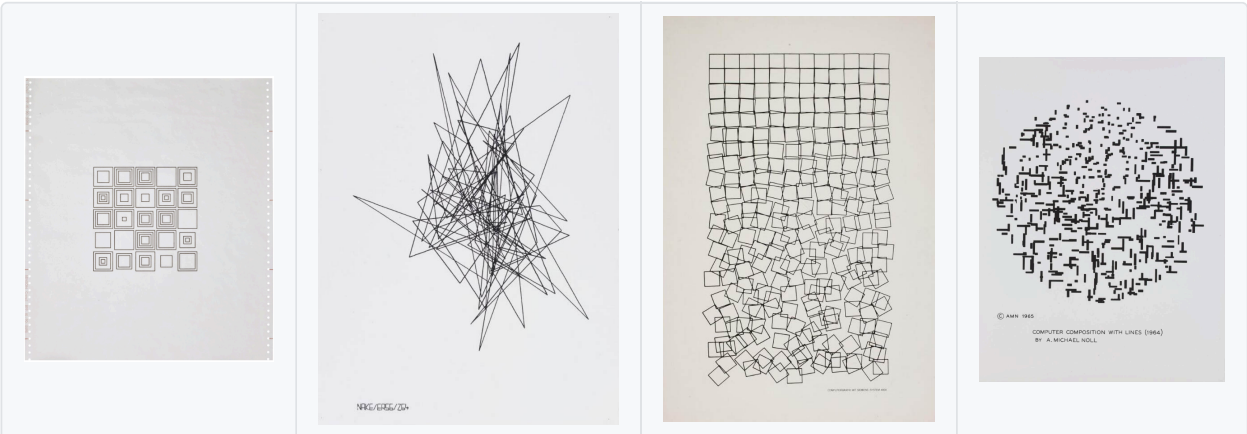
Probleem	Mogelijke oorzaak of oplossing
De pagina herlaadt niet	Controleer of het bestand **opgeslagen** is. Een stip op het tabblad betekent dat het nog niet bewaard is, en Live Server herlaadt pas na opslaan. Draait de Live Server? Klik op **Go Live**.
Wit scherm, geen canvas	Kijk in de console (**F12**). Staat in index.html het p5.js-script vóór `sketch.js` ?

4. Oefening 1: reCode Molnár, Nake, Nees of Noll

In de jaren zestig en zeventig lieten kunstenaars als Vera Molnár, Frieder Nake, Georg Nees en Michael Noll plotters tekeningen maken op basis van zelfgeschreven regels, vaak met een spel tussen orde en toeval. Het ReCode Project herwerkt zulke werken in moderne code.

Wij gaan iets vergelijkbaars doen met hulp van AI, maar zonder de originele code. We starten vanaf de tekening, ontcijferen de regels die erachter kunnen zitten en laten Copilot ze stap voor stap omzetten naar een werkende sketch.

Werk in duo en kies samen één van deze werken:



👉 Vera Molnár, *(Dés)Ordres*, 1974

👉 Frieder Nake, *Random Polygon (Zufälliger Polygonzug)*, 1965

👉 Georg Nees, *Schotter*, 1968

👉 A. Michael Noll, *Computer Composition with Lines*, 1964

4.1. Spelregels

- Je hoeft geen perfecte kopie te maken. Het doel is dat de regels dezelfde **soort** tekening opleveren.
- **Noem in je prompts de titel of de kunstenaar niet.** LLM's kennen deze werken vermoedelijk en geven dan een kant-en-klare versie. Dan hebben we niets geleerd.
- Je budget is beperkt, dus mik op ±7 à 10 doordachte prompts.
- Je kan ook inline hulp krijgen van de LLM terwijl je rechtstreeks in je code werkt. Je schrijft zelf code en gaat in op de suggesties die verschijnen, of vraagt de LLM om bepaalde stukken code uit te leggen, aan te passen of te verbeteren. Dat kost je wat minder credits.

4.2. Stappenplan

Stap 1: Kijken en decoderen

Zonder computer. Schrijf **5 tot 8 regels** op, in eigen woorden en met getallen waar het kan.

Kijkvragen:

- Wat zijn de **basiselementen**? Hoeveel zijn er, en hoe staan ze geordend?
- Wat is **overal gelijk** (constanten) en wat **verschilt** (variabelen)?
- Waar zit het **toeval** of **de willekeur**: in positie, grootte, draaiing, aanwezigheid, richting of lengte? Heeft dat toeval **grenzen**?
- Verandert er iets **geleidelijk** over de tekening?

- Wat zijn de **vorm en verhoudingen** van het geheel (marges, formaat)?

> **Tip**
>
> Zoom in, tel en meet (verhoudingen volstaan).
> Splits je lijst in *wat je zeker weet* en *wat je gokt*.
> Maak desnoods een kleine schets met de maten.
> Gebruik de regels en tips uit §3. Prompten

Stap 2: Het canvas en kader

Laat Copilot enkel een leeg kader opzetten: canvasformaat, achtergrond, lijnkleur, geen fill.

> **PROMPT:**
>
> Ik wil een tekening maken in p5.js op een witte achtergrond. Zet een basisstructuur op met de volgende eigenschappen: Canvas van 600 op 900 pixels in portretmode
> Witte achtergrond
> Uitsluitend zwarte contourlijnen, zonder vulling (`noFill()`)
> Lijndikte van 1 punt
> De tekening blijft overal minstens 50 pixels verwijderd van de randen van het canvas

In de prompt hierboven is bewust een fout opgenomen. Als alles goed gaat, vraagt Copilot om verduidelijking over de afmetingen en de oriëntatie van het canvas.

Copilot zal je tijdens het werken ook bijkomende vragen stellen of om toestemming vragen om bepaalde acties uit te voeren. Lees steeds goed wat er gevraagd of goedgekeurd moet worden voordat je instemt.

> **Tip**
> Je kan verwijzen naar één of meerdere files door deze toe te voegen of vast te pinnen in de chat.

Stap 3: De basis

Bouw eerst **zonder willekeur** (indien zinvol) de basiselementen en hun ordening.
Controleer de tekening naast het origineel voor je verder gaat.

> **PROMPT:**
>
> Het is aan jou :-)

Stap 4: Toeval toevoegen

Voeg **één** toevalsregel per prompt toe. Test na elke stap.
Gebruik een seed, zodat je een resultaat kunt terugvinden. Zit het toeval in een verloop of begrenzing, voeg dat dan als aparte stap toe.

Stap 5: Evalueren en bijsturen

Leg je tekening naast het origineel. Wat klopt er niet?
Pas **één parameter** aan en bekijk het effect. Vraag Copilot welke getallen die parameter bepalen, in plaats van het te laten herschrijven.

Stap 6: Exporteren

Voeg SVG-export toe met p5.plotSvg (versie 0.3.0, p5.js 2.x). Als je op een toets drukt, wordt de tekening één keer geëxporteerd als SVG-bestand, zodat je het later kan laten tekenen door een XY-plotter.

> **PROMPT:**
> Ik ben tevreden met het resultaat en wil de tekening kunnen exporteren. Gebruik hiervoor de bibliotheek p5.plotSvg.
> Pas de bestaande code in sketch.js aan zodat ik de tekening als SVG kan exporteren.
> Geef me de code die ik aan index.html moet toevoegen om de bibliotheek p5.plotSvg te laden. Vermeld ook waar in het document ik deze regel moet plaatsen.
> Leg kort uit hoe ik de SVG-export kan exporteren.
> Behoud de bestaande tekening en gebruik compatibele code voor p5.js 2.x. Controleer of de voorgestelde code overeenkomt met p5.plotSvg.

4.3. Reflectie

Hoeveel prompts heb je nodig gehad? Had je stappen kunnen overslaan?

Vraag **nu** aan Copilot welke regels de kunstenaar waarschijnlijk heeft gebruikt en vergelijk die met jullie lijst. Waar verschillen jullie? Is dat een fout of een andere lezing?

5. Oefening 2: Eigen programma

Het doel van deze oefening is een eigen programma te maken dat een stuk verder gaat dan onze huidige kennis van p5.js.

Er zijn twee mogelijke opties.

1. **A. Plottertekening**: Een generatieve tekening die uit enkel lijnen bestaat, en die je kunt plotten op een AxiDraw
2. **B. Interactief beeld en geluid**: Een sketch die muisbeweging en / of het indrukken van toetsen vertaalt naar beeld én geluid, als een klein instrument

5.1. Casus A: Plottertekening

We maken een eigen generatieve tekening die uitsluitend uit lijnen bestaat en die we kunnen exporteren als SVG-bestand, klaar voor een XY-plotter. Het formaat is A4, staand of liggend. Om je tekening complexer en boeiender te maken, kan je experimenteren met arceringen, meerdere lagen of een tweede penkleur.

Regels bedenken op papier

Denk eerst na over de regels die je tekening bepalen. Hoe ontstaan de lijnen? Wat bepaalt hun richting, lengte of kromming? Is er een herhaling van figuren, ...
Schrijf je ideeën uit op papier en maak eventueel een schets. Vertrek vanuit een eenvoudig systeem en onderzoek hoe je daarmee interessante patronen kan genereren.

Werk verder uit

Vertaal je regels naar code in p5.js met Copilot. Begin eenvoudig en experimenteer met parameters zoals lijndichtheid, richting en herhaling. Zorg ervoor dat je tekening reproduceerbaar is met een vaste seed en houd rekening met het uiteindelijke A4 of A3 formaat en de mogelijkheden van de plotter.

Naar de plotter

> **PROMPT:**
> Voeg SVG-export toe met p5.plotSvg (compatibel met p5.js 2.x). Wanneer ik op de toets "s" druk, wordt de huidige tekening één keer geëxporteerd als "plot_" + seed + ".svg".

Exporteer het SVG-bestand en controleer het in de browser of in Inkscape.

👉 https://github.com/golanlevin/p5.plotSvg

Aanpassen, verfijnen en variëren

Bekijk het resultaat en experimenteer verder. Voeg bijvoorbeeld een tweede laag in een andere penkleur toe, gebruik arceringen rond bepaalde vormen of beperk het aantal pen-lifts om sneller te plotten. Probeer verschillende variaties en ontdek hoe kleine aanpassingen je tekening veranderen.

5.2. Casus B: interactief beeld en geluid

Liever iets interactiefs? Dan bouwen we een klein audiovisueel instrument.

We kunnen ons waarschijnlijk al iets voorstellen bij het genereren van bewegend beeld. De mogelijkheden om geluid af te spelen of te genereren zijn misschien wat moeilijker in te beelden. Bekijk daarom even de p5.sound-library referentie en deze track met video-tutorials van Dan Shiffman.

Zoek je eerder wat directe inspiratie bij (vroege) interactieve werken bekijk dan eventueel even deze referenties.

👉 *Bubble Harp*, van Scott Snibbe uit 1997 met een recente muzikale remake als de Passion Pit Gossamer App
👉 *Single Cell*, van LIA uit 2001
👉 *Sonic Wire Sculptor* van Amit Pitaru uit 2003
👉 *BallDroppings* van Josh Nimoy uit 2009
👉 *Yellowtail* van Golan Levin uit 1998, met een remake in p5js *(zonder geluid)*
👉 de internet-werken van Rafaël Rozendaal *(zonder geluid)*
👉 De installatie Communion van Field.io met Matt & Simon Pyke uit 2011

Regels formuleren

Schrijf op wat jouw instrument doet. Bijvoorbeeld: *de muispositie bepaalt de toonhoogte en het volume, een toets wisselt tussen vormen, het beeld laat een spoor achter dat langzaam vervaagt.*

Zorg er ook voor dat de sketch in fullscreen kan bespeeld worden.

De prompts

1. **Beeld en muis:** "Teken een spoor van ..."
2. **Toetsen:** "Met de toetsen 1, 2 en 3 wissel ik ..."
3. **Geluid:** "Voeg een `p5.Oscillator` toe ..."
4. **Koppeling beeld en geluid:** "Laat elke vorm een ander golftype hebben (sine, triangle, square) ..."
5. **Verfijnen:** "Voeg een ..."

Bij geluid zijn de eerste versies vaak hard of schel. Laat je volume dus laag beginnen en pas het op het einde aan.

> ⚠ **Warning**
> Voeg in index.html de **p5.sound**-bibliotheek toe, na het p5.js-script: `<script src="https://cdn.jsdelivr.net/npm/p5.sound@0.4.1/dist/p5.sound.min.js"></script>`
> Laat je Agent dit doen? Controleer dan de diff.

> ⓘ **Note**
> **Geen geluid?** Browsers starten geluid pas na een klik of toetsaanslag. Vraag in je prompt om een startscherm met *"klik om te starten"* (in p5.sound bestaat daarvoor `userStartAudio()`).

6. Reflectie & verder

6.1. Terugblik

Bespreek in tweetallen en daarna met de hele groep:

- Welke prompt werkte het best? Waarom?
- Waar ging het mis, en hoe heb je dat opgelost?
- Waar nam je zelf het stuur over van de LLM?
- Wat zag je in het resultaat dat je niet had gevraagd? Is dat een fout of een geschenk?
- Wat is volgens jou **jouw** deel van de sketch?
- Wanneer liet je de agent zijn gang gaan en wanneer greep je in? Welke wijzigingen of commando's heb je geweigerd, en waarom?

6.2. Cheat sheet: prompts

Situatie	**Voorbeeldprompt**
Starten	"Ik gebruik p5.js 2.x (global mode). Canvas $grootte$. Ik wil $doel$. Regels: $lijst$. Geef de volledige sketch.js met korte Nederlandse comments."
Bijsturen	"Verander enkel dit. Laat de rest ongewijzigd."
Begrijpen	"Leg deze regel uit. Ik ken de basis van p5.js, maar nog geen $term$."
Fout	"Ik verwachtte x, ik zie y. Foutmelding: $tekst$. Wat is de kleinste aanpassing?"
Ideeën	"Geef me drie verschillende manieren om x te doen, kort uitgelegd."
Opschonen	"Maak deze code overzichtelijker zonder het resultaat te veranderen. Leg uit wat je aanpast."
Controleren	"Gebruikt deze code enkel functies uit p5.js 2.x?"
Agent starten	"Geef eerst een plan en pas nog niets aan. Wijzig enkel sketch.js en voer geen terminalcommando's uit."
Scope bewaken	"Je hebt ook $bestand$ aangepast. Dat vroeg ik niet. Zet dat terug en wijzig enkel dit."
Commando	"Waarom heb je dit terminalcommando nodig, en kan het ook zonder?"
Terugdraaien	"Draai je laatste wijziging terug en leg uit wat er misging."

6.3. Kritische kanttekeningen

Een LLM roept ook vragen op die we niet oplossen, maar wel kort bespreken: **Van wie is een sketch die je samen met een LLM maakt?** Waar komt de code vandaan waarop het model getraind is? Wat betekent het energieverbruik? En wat gebeurt er met onze vormtaal als iedereen dezelfde tool gebruikt?

6.4. Links

👉 https://docs.github.com/en/copilot/tutorials/vibe-coding
👉 https://code.visualstudio.com/docs/copilot/chat/chat-agent-mode
👉 https://code.visualstudio.com/docs/agents/approvals

1. p5js.ai is verbonden met Xelsed.AI, een initiatief dat AI inzet voor creatief programmeren en toegankelijker programmeeronderwijs, onder meer voor mensen met dyslexie. ↵

2. GitHub is een online platform waar je code kunt bewaren, delen en samen met anderen aan projecten kunt werken. Het houdt ook de verschillende versies van je code bij, zodat je wijzigingen kunt opvolgen en eerdere versies kunt herstellen. ↵