

# Games maken en ervaren

Domein J: Keuzethema Programmeerparadigma's

Domein P: Keuzethema User experience

**handleiding docent**

# INHOUD

Algemeen	3
Inleiding	3
Materialen	3
Werkomgeving inrichten en delen	4
Overige informatie werkomgeving	4
In de les aan de slag met de werkomgeving	5
Wat bied ik aan mijn leerlingen aan?	5
De hoofdstukken	6
Tot slot	12

# ALGEMEEN

## Inleiding

Welkom bij de handleiding bij de module **Games maken en ervaren**. Deze module is een uitwerking van de domeinen **Domein J: Keuzethema Programmeerparadigma's** en **Domein P: Keuzethema User experience**. Er is gekozen voor een praktische insteek waar zowel havo als vwo mee uit de voeten kan. Leerlingen moeten veel zelf aan de slag, zonder dat te zware theorie hun in de weg zit.

## Materialen

Het lesmateriaal bevat de volgende onderdelen:

- PDF met theorie en vragen
- een complete werkomgeving (toelichting verderop in dit document) in de map *server* met:
  - navigatie-schil
  - uitgewerkte voorbeelden waar in theorie en opdrachten naar wordt verwezen
  - alle opdrachten en uitwerkingen (voor zover dit programmeeropdrachten betreft)
  - een separate map voor een eindopdracht
  - een docentenmap
- Basisopdrachten: te herkennen aan ✔  
Deze bevatten vragen die dichtbij de theorie en voorbeelden liggen. Ze dienen als *eenvoudige* instap en kunnen naar uw inschatting door betere leerlingen worden overgeslagen. In de testfase waren deze opdrachten verplicht voor havo en optioneel voor vwo.
- Steropdrachten en -paragrafen: te herkennen aan ☆  
Deze bevatten stof voor de betere havo-leerling en vwo-leerling, in te zetten ter differentiatie. Hieronder vallen ook zogenaamde obfuscator-opdrachten voor extra uitdaging.
- Toetsen behorende bij de stof van hoofdstuk 1.  
De toetsen met uitwerking vindt u in de map *toetsen*.

Het lesmateriaal is opgedeeld in drie hoofdstukken (H4 bevat geen lesstof). De studielast hiervan hangt erg af van de vraag hoe u het verplichte programmeerdeel uit het examenprogramma in de klas heeft gedaan.

### Hoofdstuk 1 kennismaken met P5

Dit hoofdstuk heeft als doel om leerlingen kennis te laten maken met P5 en indien nodig met Javascript. Het dient slechts als basis of herhaling en bevat nog geen inhoud van domein J of domein P.

### Hoofdstuk 2 objecten

Dit hoofdstuk leert de leerlingen stapsgewijs om te programmeren volgens het OO-paradigma. Dit hoofdstuk is een uitwerking van Domein J van het examenprogramma.

### Hoofdstuk 3 game design

In dit hoofdstuk ligt de nadruk minder op programmeren en code en meer op de manier waarop spellen worden ontwikkeld en welke uitwerking dit op een speler heeft. Dit hoofdstuk is een uitwerking van Domein P van het examenprogramma.

De betere, zelfstandige leerling kan de stof van hoofdstuk 1 en hoofdstuk 2 ook verwerken met behulp van:

### Hoofdstuk 4 gamification

Dit hoofdstuk bevat opdrachten waarmee de leerling zelf kan vaststellen of ze de stof van een bepaalde paragraaf beheersen door een bestaand programma na te maken. Elke opdracht is vertaald naar een *level* dat gehaald moet worden om verder te mogen. Als de kennis en vaardigheid daarvoor ontbreekt, zal de leerling zelf in de stof moeten kijken of een aantal opdrachten moeten oefenen om het level voor elkaar te krijgen.

Dit hoofdstuk biedt naast de basis- en steropdrachten een extra manier om te differentiëren.

## Werkomgeving inrichten en delen

De map *server* bevat een groot aantal bestanden die naar uw keuze volledig of minder volledig met de leerlingen kunnen worden gedeeld. Een aantal aandachtspunten **voordat** u de map met leerlingen deelt:

- In de map *JS* staan per hoofdstuk de bijbehorende opgaven en uitwerkingen (met de toevoeging **U**). In de testfase hebben wij steeds de uitwerkingen met leerlingen gedeeld. Doet u dat liever niet, dan dient u deze bestanden te verwijderen voor delen.  
LET OP: als u wilt werken met *Gamification* (zie toelichting verderop), dan kan dit niet, omdat de opzet daar juist is dat de leerlingen het uitwerkingsbestand aanvullen.
- Als u de uitwerkingen niet wil delen, heeft u de optie om de knop *Toon Uitwerkingen* van de werkomgeving uit te schakelen (om te voorkomen dat er *dead links* ontstaan). Het bestand *SLOBF.js* in de map *JS* bevat hiertoe in regel 1 de setting *uitwerkingenIngeschakeld*. Deze kan op *false* worden gezet.  
LET OP: als u wilt werken met *Gamification* (zie toelichting verderop), dan kan dit niet, omdat de opzet daar juist is dat de leerlingen het uitwerkingsbestand aanvullen.
- Met betrekking tot *Gamification*: deze uitwerkingen staan niet in de map van H4 maar in de docentenmap in de subfolder *GA\_uitwerkingen*.
- De docentenmap bevat een losse *index.html* die gebruikt kan worden voor het geven van afzonderlijke demo's. De map *DEMO* bevat een aantal voorbeelden die gebruikt zijn bij presentaties voor docenten. Aan deze map kunt u uw eigen voorbeelden voor in de les toevoegen (en al dan niet delen met uw leerlingen).

## Overige informatie werkomgeving

De map *server* bevat een aantal mappen (met soms submappen), waarvan wij verwachten dat ze voor zich wijzen: *CSS*, *images*, *sounds*. Daarnaast:

- *index.html* bevat de schil. Leerlingen hoeven dit bestand niet in hun editor te openen en kunnen volledig met de Javascript-bestanden in de map *JS* werken.
- *bekijkJS.html* biedt een kans om zonder de navigatie-schil te werken en toont hoe een html-bestand kan worden gekoppeld aan een JS-bestand. Leerlingen die hier mee willen werken, moeten dan wel het html-bestand aanpassen als ze naar een andere opdracht willen.
- In *bekijkJS.html* staat een verwijzing naar *test.js* in de map *JS*. Dit bestand is in de testfase gebruikt om leerlingen kleine klassikale oefeningen te laten maken, op basis van door de docent getoonde voorbeelden. Dit kan dus didactisch worden ingezet.
- In de head van *bekijkJS.html* wordt een lokale koppeling gelegd met de P5-library (in de submap *P5*). Deze wordt logischerwijs niet geüpdatet. Het is mogelijk een koppeling met de actuele versie van P5 te leggen:  
<https://p5js.org/get-started>  
Wij hebben daar niet voor gekozen, omdat een aantal online omgevingen deze externe koppelingen niet toestaat.
- De submap *addons* van *P5* bevat een aantal aanvullende libraries die aan het eind van H3 worden gebruikt ter illustratie van de uitgebreide mogelijkheden. Naar eigen smaak kan deze map worden aangevuld met voorbeelden uit: (er zijn ook andere bronnen met particulier ontwikkelde libraries)  
<https://p5js.org/libraries>
- De map *mijnGame* bevat bestanden met een voorbeeldgame en (een kopie van) ondersteunende bestanden zoals *p5.minj.js*. De leerling kan deze map gebruiken om zijn eigen game in te ontwikkelen, bijvoorbeeld als praktische opdracht.  
Het voorbeeldspel kan via de knop in het menu worden bereikt.
- De omgeving bevat de PDF van het lesmateriaal (zie de knop PDF in het menu). Veel leerlingen vinden het echter prettig om een papieren versie van de module naast hun computer te hebben.

# In de les aan de slag met de werkomgeving

Bij een groot deel van de vragen is het de bedoeling dat leerlingen zelf programmeren. U heeft vast al een favoriete editor naar keuze. In de testfase is het volgende gebruikt:

- In de eerste testfase is gebruik gemaakt van Cloud9:  
<https://aws.amazon.com/cloud9>  
waarmee leerlingen hun eigen (virtuele) server kregen (als clone van de docent) om in te werken. Sinds de overname door Amazon is dit naar onze smaak een stuk minder toegankelijk geworden.
- In de tweede testfase is gebruik gemaakt van *GitPod* en combinatie van *GitHub*. Leerlingen maken hierbij een *fork* van de repository van de vakdocent in *GitHub* en gebruiken de omgeving van *GitPod* voor het doen van aanpassingen (die indien gewenst weer in *GitHub* opgeslagen kunnen worden (*push*). Deze combinatie vraagt wat meer investering in de les, maar daarna zijn de ervaringen positief (een tip dus).  
<https://github.com> && <https://gitpod.io>
- Als laatste: lokaal werken met *Notepad++*. Dit kan in de browser voor problemen zorgen, wanneer gewerkt wordt met externe bronnen als afbeeldingen en geluidsbestanden. Zie voor meer achtergrondinformatie:  
<https://developer.mozilla.org/nl/docs/Web/HTTP/CORS>

## Wat bied ik aan mijn leerlingen aan?

Er zijn vele manieren om dit lesmateriaal in de praktijk te gebruiken. Er is een overmaat aan materiaal: ook als u beide keuzedomeinen wilt aanbieden is het verstandig om keuzes te maken. Daarnaast kunt u didactische keuzes maken qua niveau en vorm (basis- en steropdrachten; gamification).

Zoals gezegd is hoofdstuk 1 bedoeld als basis om kennis te maken met Javascript en P5. In hoeverre dit nodig is voor uw leerlingen, hangt af van de voorkennis die ze eerder hebben opgedaan. Omdat dit voor elke school weer anders is, adviseren we om een planning op maat te maken, maar het hoofdstuk niet helemaal over te slaan. Leerlingen die een jaar geleden hebben geprogrammeerd, hebben ook behoefte aan herhaling om hun kennis en vaardigheid op te frissen.

Als uw klas een behoorlijke basis heeft, zou u ervoor kunnen kiezen om de stof klassikaal aan te bieden met *gamification*. Voor hoofdstuk 1 betreft dit zeven levels: voor elke paragraaf 1. Om te toetsen of ze voldoende niveau hebben kan, gebruik gemaakt worden van de bijgeleverde toetsen, al dan niet in formatieve vorm.

Wie dit materiaal alleen wil gebruiken voor domein J kan intensief aan de slag met hoofdstuk 2. Hoewel hoofdstuk 3 nog uitbreiding en verdieping geeft, ook als het gaat om structuur en strategie van programmeren, geeft hoofdstuk 2 voldoende basis om af te sluiten met een PO waarbij leerlingen gevraagd wordt om een game te maken. In de eerste fase van testen bestond hoofdstuk 3 nog niet. Niettemin bleken leerlingen (zowel havo-4 als vwo-4) in staat om op basis van hoofdstuk 2 een OO-geprogrammeerde game te maken. De betere leerlingen kunnen de stof ook volledig doorlopen m.b.v. de *gamification*-opdrachten.

Ter afwisseling en verdieping in de les kan, ook als alleen voor domein J wordt gekozen, H3 gebruikt worden om selectief aandacht te besteden aan een aantal aspecten van games. Ook is het mogelijk om leerlingen uit te dagen om zelfstandig te kijken naar de uitgewerkte voorbeelden. Nota bene: de meeste spellen in H3 zijn al volledig uitgewerkt, omdat hier weinig nadruk ligt op het programmeren zelf.

Wie dit materiaal alleen wil gebruiken voor domein P, kan hiervoor een aangepaste selectie uit de opdrachten halen, maar er zijn wel de nodige beperkingen. Bij het eerste deel van hoofdstuk 3 is daar bij de meeste opgaven wel enige kennis van object-georiënteerd programmeren vereist. Het zwaartepunt van domein P zit in de paragrafen 3.5 en 3.6. Daar wordt relatief weinig gevraagd naar het bestuderen of aanpassen van code. Wel moeten soms de *game-settings* worden aangepast. Dit kan in onze ogen echter ook prima, zonder verdere kennis van Javascript.

Wij adviseren om deze module af te sluiten met een praktische opdracht. Ook als hoofdstuk 3 niet klassikaal behandeld is, zouden leerlingen hierin zelfstandig kunnen *shoppen* afhankelijk van hun plan.

# DE HOOFDSTUKKEN

Hieronder volgt per hoofdstuk een aantal opmerkingen over de gemaakte (didactische) keuzes en uitwerkingen.

## H1 kennismaken met P5

Dit hoofdstuk laat de leerlingen kennis maken met Javascript en P5. Hierbij wordt een groot deel van de basisstructuren uit het programmeren in context herhaald. Denk hierbij aan variabelen, keuze, functies en herhaling. Gezien het doel van deze module, hebben we dit zoveel mogelijk beperkt en ligt er geen nadruk op algoritmes. Zo hebben we er bewust voor gekozen om geen *while*-loop in het materiaal opgenomen. Wel is er een game-loop en wordt kennis gemaakt met een aantal vormen van interactie t.b.v. de ontwikkeling van games.

Programmeren kan op vele manieren. U zult ontdekken dat we bij de ontwikkeling van deze module de nodige keuzes hebben gemaakt (ook qua indeling) die in uw ogen ook *anders* of *beter* zouden kunnen. Dit was voortdurend onderwerp van gesprek, waarbij er een spanningsveld bleek tussen zaken als correctheid, leesbaarheid / begrijpelijkheid, volledigheid en haalbaarheid. Om één concreet voorbeeld te noemen:

Als de variabele *test* een boolean is, voldoet de regel: *if (test) { }*

Ter verduidelijking beginnen we in de module met booleans met het uitgebreidere *if (test == true) { }*

De laatste paragraaf over recursie dient ter verdieping. Recursie wordt soms aangeduid als deel-paradigma en kan in die zin onder domein J geplaatst worden.

Bij dit hoofdstuk zijn geen uitwerkingen in deze handleiding. Het gros van de opdrachten leidt tot een uitwerking binnen het Javascript-bestand. Deze zijn meegeleverd in de basis die aan leerlingen wordt uitgedeeld. De overige vragen gaan grotendeels over waarnemingen in code of uitvoer.

## H2 objecten

Dit hoofdstuk leert leerlingen stapsgewijs denken en programmeren met objecten. Ook hier is het doel dat leerlingen snel zelf resultaat kunnen boeken. Om dat te bereiken hebben we gepoogd de codes zo toegankelijk mogelijk te houden. Qua objecten is de volgorde:

- i. inleiding met bestaande objecten
- ii. losse objecten programmeren
- iii. objecten maken op basis van een klasse
- iv. array van objecten
- v. overerving (alleen in een steropdracht)

Om eenheid van programmeerstijl te houden, werken we vanaf het moment dat klassen zijn geïntroduceerd altijd met een klasse, ook als hiervan maar één instantie wordt gemaakt. Overerving komt slechts beperkt aan bod in deze module, met als argument: *te veel en te complex*. Overerving wordt vooral functioneel bij grotere en complexere spellen / programma's. Dat komt de leesbaarheid en haalbaarheid niet ten goede. Daarom is ook in hoofdstuk drie verder geen gebruik gemaakt van overerving, ook niet wanneer dit wellicht functioneel zou zijn. Hiermee wordt voorkomen dat overerving voorkennis moet zijn voor het vervolg van de module.

Ook voor dit hoofdstuk hebben we een groot aantal overige keuzes moeten maken. Zo doen we niet moeilijk over afgeleide attributen, zoals *this.straal = this.diameter* en vermijden we *getters* en *setters*.

Bij dit hoofdstuk zijn geen uitwerkingen in deze handleiding. Het gros van de opdrachten leidt tot een uitwerking binnen het Javascript-bestand. Deze zijn meegeleverd in de basis die aan leerlingen wordt uitgedeeld. De overige vragen gaan grotendeels over waarnemingen in code of uitvoer.

## H3 game design

Dit hoofdstuk laat leerlingen kennis maken met talloze aspecten op het gebied van *user experience* en interactie in de context games. Daarnaast is er aandacht voor verdieping als het gaat om de structuur van spellen met *design patterns*, bijvoorbeeld door het gebruik van *flowcharts*, en het gebruik van bijzondere objecten als *spel* en *speler*. Doel is dat leerlingen patronen in spelontwerpen leren herkennen en leren begrijpen waarom ontwikkelaars bepaalde keuzes in spellen maken c.q. welke invloed ze hebben op de manier waarop een spel wordt ervaren.

Terloops zijn in dit hoofdstuk nog enkele wat meer geavanceerde technieken te zien:

- klassen in afzonderlijke bestanden
  - opgave 30 gebruikt het door ons geschreven *laadJavaScriptFile* om de klassen *Timer* en *Button* te laden. In de module wordt hier geen bijzondere aandacht aan geschonken. Als docent kunt u ervoor kiezen om aandacht te schenken aan hergebruik van klassen (of het gemak wanneer meerdere ontwikkelaars samenwerken)
  - opgave 33 en 34 gebruiken *Timer* opnieuw.
  - Het spel in *mijnGame* gebruikt eveneens meerdere klassen (uit de submap *class*), maar hier is dit in de head van het bijbehorende html-bestand afgehandeld.
- libraries
  - opgave 34 demonstreert het gebruik van *p5.sound* (met behulp van *laadJavaScriptFile*)
  - Het spel in *mijnGame* gebruikt *p5.sound* en *p5.shape* (via de head van de html)
- spel op volledig scherm
  - Het spel in *mijnGame* gebruikt volledig scherm (en aanpassing bij aanpassing grootte browserscherm). Wij nemen aan dat dit voor leerlingen een aantrekkelijke vorm is, maar voor de opdrachten hebben we gekozen voor een canvas i.c.m. een vaste menu-structuur.

De flowcharts in dit hoofdstuk zijn gemaakt met [www.diagrams.net](http://www.diagrams.net) (draw.io) dat integreert met *Google Docs*. Mocht u zelf het lesmateriaal willen uitbreiden of van leerlingen vragen om zelf diagrammen te ontwikkelen, dan is dit een laagdrempelige tool.

In de laatste paragraaf van dit hoofdstuk worden suggesties gedaan voor het maken van een eigen game. Hier staat ook een lijst met bruikbare links voor leerling en docent.

## H3 game: uitwerkingen

In tegenstelling tot de **U**-bestanden van hoofdstuk 1 en hoofdstuk 2, zijn de **U**-bestanden van H3 niet altijd uitwerkingen, omdat in veel opdrachten niet gevraagd wordt om iets te programmeren. Wel bevatten deze bestanden soms andere game-settings van hetzelfde spel. Ook kunnen ze dienen als backup, wanneer een leerling settings heeft aangepast.

Bij een deel van de vragen in dit hoofdstuk gaat het om waarnemen of het geven van een mening. Van een selectie van de overige vragen heven we hieronder een mogelijke uitwerking.

### Opdracht 1

2. Bij het spel kun je door te klikken extra ballen maken die op de grond stuiten. De uitdaging is om zoveel mogelijk ballen te maken en ze, voordat ze een tweede keer stuiteren, wederom allemaal aan te klikken.

Als je er eentje mist ben je af. Je score is het aantal ballen waarvoor het gelukt is.

3. kenmerk:

- speler (-s): Ja: niet in de code, maar je kunt het spel spelen
- interactie / keuze: Ja: je bestuurt het spel met de muis en bepaalt zelf hoeveel ballen je maakt
- spelregels: Ja: ze zijn impliciet (staan nergens), maar het spel laat je op basis van regels winnen
- uitdaging / doel / competitie: Ja: de uitdaging is om het trucje voor zoveel mogelijk ballen te doen
- resultaat / score / uitkomst: Ja: er is een scorebord en een eindscherm
- voortgang: Ja: tijdens het spel kun je lezen hoeveel ballen je succesvol hebt aangeklikt.
- levels / niveau: Nee.
- terugkoppeling (tekst, beeld, geluid): Ja, in de vorm van een scorebord en een eindscherm
- abstractie (versimpeling t.o.v. de echte wereld): Ja: bijna volledig: je ziet alleen wat cirkels.

4. a. Als je handig bent met je muis, heb je zeker voordeel.

b. Het maakt zeker uit op welke plek je klikt om ballen te maken, dus strategie is van belang.

c. Een ervaren speler kan bewust spelen en zal geen grote factor geluk ervaren.

5. Als je bovenin het canvas op een vaste breedte snel achter elkaar klikt, komen alle ballen weer naar je toe en kun je ze eenvoudig voor een tweede keer aanklikken.

9. Training (simulator Max Verstappen, opleiding chirurgen), onderwijs (educatieve game), inkomsten, wetenschappelijk onderzoek.

### Opdracht 3

16. Eenvoudig te spelen in tweetallen, kan ook in groepsverband, op papier kun je zelf de opdracht voor een ander bedenken wat allerhande emoties zorgt, weinig materialen voor nodig, kort, eenvoudige spelregels, iedereen kan meedoen, etc.

17. De computer controleert hier de juiste invoer en of een letter in het woord zit, de computer verzorgt de feedback, je kunt (in deze versie) in de computerversie niet zelf een woord kiezen.

18. invoerscherm voor een nieuwe opdracht, lijst met bestaande opdrachten waar random uit wordt gekozen, levels met oplopende moeilijkheidsgraad, overzicht van letters die zijn geprobeerd, blokkeren van dubbel straffen als je per ongeluk twee keer dezelfde verkeerde letter raadt, geluidjes afhankelijk van goed of fout raden.

### Opdracht 6

32. Op het moment dat je goed gokt, word je beloond met een hogere score. Als je dichtbij score 0 zit, geeft dat een gevoel van spanning, als je een paar keer succes hebt stimuleert dit om door te gaan, omdat je nieuwsgierig wordt hoe ver je kunt komen.

34. Het gevoel dat er meer te halen is. Het gevoel dat het de volgende keer beter kan gaan. Bij een ander hebben gezien dat die hoger scoorde. Directe feedback in scherm: je hebt voortdurend resultaat.



## Opdracht 7

40. Bij de start van het spel ziet de speler een geheime code in de vorm van vier zwarte cirkels. Deze verbergt een kleurcode die kan bestaan uit rode, groene, blauwe, paarse, gele of oranje cirkels in alle mogelijke combinaties (dus ook met meerdere keren dezelfde kleur). De speler voorspelt een code door achtereenvolgens vier keer te klikken op gekleurde cirkels met eerder genoemde kleuren. Na elke vierde klik toont het spel een nieuwe rij onder de code waar te zien is of en zo ja welke kleur goed geraden was. Dit proces herhaalt zich totdat de code geraden is met een maximum van vijf pogingen. Na vijf foute pogingen verschijnt *verloren* :(. Als de code geraden is verschijnt *gewonnen* :).

41. Allerhande spellen tegen de computer waarbij er geen of één tegenzet of verandering van toestand is, zoals schaken of dammen, stratego, Nim (opdracht 8), etc.

43. Voor de eerste beurten kun je er bijvoorbeeld voor kiezen om een code van vier dezelfde kleuren te kiezen, zodat je die kleuren in de volgende stappen kunt buitensluiten.

44. Een veel gebruikte strategie is om eerst op zoek te gaan naar de klinkers of meest voorkomende letters (zoals *e* en *n*). Als je een variant van Galgje speelt waarbij je zelf een te raden woord mag kiezen, kun je dit omgekeerd gebruiken door zeldzamere letters en combinaties te gebruiken, zoals *angstschreeuw* in opgave 3: het Nederlandse woord met de meeste medeklinkers achter elkaar.

45. Als je bij Codekraker een de tactiek van vraag 43 gebruikt heb je niet genoeg beurten om met zekerheid de code vast te stellen. Als je pech hebt bevat de code geen van de kleuren die je systematisch hebt onderzocht. Bij Galgje kan de tactiek van 44 ook zorgen voor weinig resultaat. Het is hier wel de vraag of dit te maken heeft met pech. In tegenstelling tot Codekraker, worden woorden bij Galgje doorgaans bewust en niet random gekozen.

## Opdracht 9

51. In deze versie kun je terugkijken welke pogingen je hebt gedaan (in plaats dat je alleen maar de goed geraden cirkels ziet). De goed geraden cirkels verschijnen apart bovenaan op de plaats van de opdracht.

52. De extra informatie (voordeel voor de speler) zit hem in het feit dat je niet zelf hoeft te onthouden welke foute pogingen je hebt ondernomen.

## Opdracht 19

111. Je kunt rustig nadenken over een stap. De vijand neemt slechts één stap nadat jij een stap hebt gezet.

112. Als je snel handelt, kun je meerdere stappen doen, terwijl de vijand slechts één stap zet.

113. Voorbeeld 24 is meer een strategiespel: er is rust waardoor je tijd kunt nemen voor de beste beslissing. Bij voorbeeld 25 ervaar je tijdsdruk, omdat de tegenstander blijft bewegen als jij even nadent. Het stimuleert tot sneller handelen.

114. Update wordt uitgevoerd als jij een stap hebt gedaan. In eerste instantie wordt gecheckt of jij daarbij op een bom of vijand stapt. Vervolgens beweegt de vijand tijdens de update. Deze beweging kan ervoor zorgen dat je alsnog wordt geraakt, zodat een tweede check noodzakelijk is.

117. In de *Draw* met behulp van *frameCount*: de vijand doet om de vijf frames (loops van de gameloop) een stap.

## Opdracht 20

120. Je hebt een *echte* tegenstander. Dat geeft competitie en interactie: je gaat dingen tegen elkaar zeggen. Het sociale aspect werkt voor veel mensen plezier verhogend.

## Opdracht 21

131. Bij tennis kun je, hoewel de bal maar op één plek is, beide tegelijkertijd het spel besturen. Hierbij ontstaat bovendien telkens een nieuwe spelsituatie, terwijl je bij vier op een rij langzaam bouwt aan een zich steeds verder ontwikkelende toestand. Bij tennis bepaalt het spel grotendeels het tempo. Bij vier op een rij moet je wachten op de tegenstander.

132. Monopoly, Stratego, etc. etc. Het grootste deel van de bordspellen is sequentieel.

133. Minecraft, Fortnite, Realm Royale.

136. De computer.

137. Met *//* wordt de kolom gemarkeerd waar de laatste speler zijn steen geplaatst heeft. Zonder *//* wordt gemarkeerd boven welke kolom je staat (voor het plaatsen van een steen).

### Opdracht 23

156a. Een fout maken *kost* je minder: de schade is beperkter, omdat je op hetzelfde niveau opnieuw mag proberen, zonder dat je de investering van het halen van het huidige level opnieuw moet doen.

### Opdracht 24

163. Elk level zijn er minder platforms. Vanaf level 2 wordt elk platform dat je aangeraakt hebt steeds kleiner, zodat je minder tijd hebt. Vanaf level 3 worden alle platforms steeds kleiner, zodat je nog minder tijd hebt.

164. Oranje: je maakt geen contact met een platform, waardoor je niet kunt springen. Groen: je kunt wel springen. Rood: je bent af. Grijs: je hebt contact gemaakt met het platform. Zwart: je hebt nog geen contact gemaakt met dit platform.

### Opdracht 25

166. Je kunt alleen springen als je contact maakt. Hoe ver je kunt springen hangt af van hoe hoog je zelf zit, maar ook hoe hoog je doelplatform zit. Dit moet je goed timen, omdat ze niet allemaal op dezelfde manier bewegen. Deze factor wordt belangrijker, naarmate er minder platforms zijn of je eigen platform zo klein wordt dat je er dreigt af te vallen.

167a. Door levels te halen.

167b. Door te springen en door platforms aan te raken. (Platforms overslaan loont dus.)

168. Als het goed is wel: je wilt nu met zo min mogelijk springen en platform-contacten de overkant bereiken.

### Opdracht 26

173. Geluidjes bij succes en bij fouten, een tekst als je voor de tweede keer een foute combinatie probeert.

174. Laat de score afhangen van het aantal pogingen, idem de tijd die je neemt om de volledige puzzel op te lossen.

### Opdracht 27

182. Flipperkast, Tetris

183a. Door voldoende zure regendruppels op te vangen.

183b. Door de druppels die de grond raken.

### Opdracht 29

194. Als de appels langzaam vallen heb je tijd om de meeste appels te vangen en laat je die appels *lopen* die alleen en niet in groepjes vallen. Hiermee optimaliseer je de vangst en minimaliseer je het verlies. Als de appels snel vallen heb je geen tijd om de uiterste punten van het canvas te bezoeken en is de beste strategie om vanuit het midden te opereren en daar naar terug te keren als je een appel gevangen hebt. Zo optimaliseer je de kans om een appel te halen.

195. De hoeveelheid appels en de snelheid waarmee ze vallen. Daarnaast zorgt de highscore voor stress, zowel als je hem bijna hebt gehaald, als wanneer je weet dat je een nieuwe highscore hebt bereikt (en hem daarna zo goed mogelijk wil afronden).

196. Zie antwoord 195. Verder stimuleert het om opnieuw te beginnen als je het gevoel hebt er bij in de buurt te kunnen komen. Het geeft ook een element van competitie als je met meerdere mensen speelt.

197. Bij proberen in de klas, bleek dit niet echt. Vooral de highscore en het aantal levens worden gebruikt.

### Opdracht 30

201. Als je steeds dezelfde puzzels krijgt, wordt het al gauw saai. Bovendien treedt een leereffect op: een bepaalde puzzel krijgt dan voor jou een lagere moeilijkheidsgraad.

202. Beeld: voor jonge kinderen zijn andere beelden aantrekkelijker dan voor oudere spelers. Spelregels: hoe ouder je bent, hoe meer en complexere spelregels je aan kunt.

### **Opdracht 32**

216. Als de timer aan zijn eind is, wordt de plek waar je dan staat genomen als plek van jouw keuze.

217. Hoe sneller je positie kiest en klikt, des te meer punten kun je verdienen.

219. Tolerantie: als je de tolerantie bij een volgend level verlaagd, moet je steeds nauwkeuriger de juiste kleur zien te vinden; diameter: als je de diameter bij elk level laat afnemen, wordt de precieze plek steeds belangrijker; snelheid: als je sneller kunt bewegen, kan een handige speler een snelle keuze maken en daarmee tijd besparen en punten winnen; tijd: door elk level de beschikbare tijd te verkorten, ontstaat stress en heb je minder tijd om te kijken, wat het spel lastiger maakt.

# TOT SLOT

## interessante links

Deze module heeft P5 als fundament. Hoewel de ontwikkeling hiervan al jaren gaande is, is de 1.0-versie in 2020 gepubliceerd, bijna gelijk met de oplevering van deze module:

<https://medium.com/processing-foundation/p5-js-1-0-is-here-b7267140753a>

Bovenstaande link geeft achtergronden bij de ontwikkelingen rond P5 die zelf weer zijn basis kent in Processing (<https://processing.org> && <https://processingfoundation.org>) en een hoop links naar uitbreidingen. De P5-gemeenschap is groot en groeiende. We geven hier geen uitputtend overzicht, maar zetten een aantal interessante links voor u op een rij:

website P5 <https://p5js.org>

online editor <https://editor.p5js.org> && <http://1023.io/p5-inspector>

Kunst met P5 <http://www.generative-gestaltung.de/2>

Bluetooth <http://1023.io/p5blejs>


Spraak <https://idmnyu.github.io/p5.js-speech>

Sprites + PS <https://gist.github.com/jessefreeman/870172>

## debuggen

Leerlingen kunnen met dit materiaal volledig in de browser werken. Natuurlijk maken ze daarbij fouten. Wij adviseren om bij de analyse van het probleem gebruik te maken van de console (F12 in Chrome browser.)

Zie: <https://javascript.info/debugging-chrome>

Natuurlijk zijn de foutmeldingen (met regelnummers) hier bijzonder bruikbaar, maar maak ook gebruik van de *live expressions* (via ) om te zien hoe waarden van variabelen veranderen tijdens de uitvoer.