H4 GAMIFICATION

4.1 Inleiding

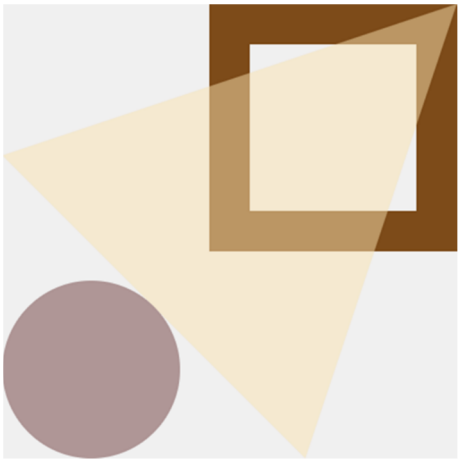
Je kunt deze module doorwerken door alle opgaven te maken, maar je kunt ook bewijzen dat je over voldoende *skills* beschikt, door bij elke paragraaf één opdracht te maken. Als je de opdracht correct hebt uitgevoerd, heb je een level gehaald en daarmee bewezen dat je de bijbehorende stof begrepen hebt.

De *gamification*-opdrachten vindt je onder de knop **GA**. De aanpak is omgekeerd aan die van de andere opdrachten. De uitwerkingen zijn hier niet beschikbaar, want jij maakt de uitwerkingen. Als je de opdracht selecteert, zie je wat je eindresultaat moet zijn. In het bijbehorende uitwerkingen-bestand is steeds al een beginnetje voor je gemaakt. Om te scoren moet jij het zelf afmaken.

4.2 Levels

Level 1 bij §1.1 vormen, kleuren & positie

Open *HGAO1.js* in de browser. Je ziet dan het eindresultaat van uit figuur 4.1. Jij moet deze figuur namaken.
Open *HGAO1U.js* in jouw *editor*. Hierin is al een beginnetje gemaakt.



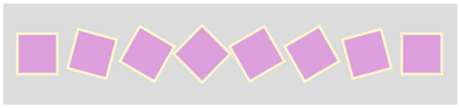
FIGUUR 4.1

Voer de volgende taken uit:

a. Voeg linksonder een cirkel in met een diameter van 176 pixels en RGB-kleurcode *175,150,150*.
b. Voeg rechtsboven een vierkant toe met zijdes van 225 pixels en een rand van 40 pixels dik met RGB-kleurcode *125,75,25*. Zorg dat het vierkant geen vulkleur heeft en achter de driehoek komt te staan.
c. Zorg dat de driehoek voor 50% doorzichtig is, zodat je het het kader van het vierkant weer volledig kunt zien.

Level 2 bij §1.2 verplaatsen & draaien

Open *HGAO2.js* in de browser. Je ziet het eindresultaat van figuur 4.2. Je moet deze figuur namaken.
Open *HGAO2U.js* in jouw *editor*. In de code zijn al een aantal keuzes gemaakt:



FIGUUR 4.2

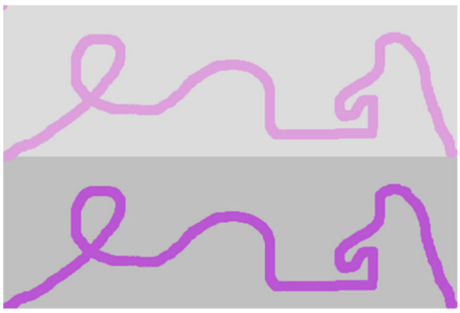
- `rectMode(CENTER)` en `angleMode(DEGREES)` worden in de *setup* aangeroepen.
- `translate(30 + 40,100);` zorgt er daarom voor dat het eerste vierkant (met zijdes van 80 pixels) 30 pixels van de linkerrand van het canvas wordt getekend.
- Om de volgende drie vierkanten te tekenen, wordt telkens dezelfde combinatie van twee regels (met `rect` en `translate`) gebruikt.

Programmeer de volgende stappen:

a. Geef de vier getoonde vierkanten een rand van 5 pixels dik van de kleur *lemonchiffon*.
b. Zorg dat elk vierkant ten opzichte van zijn voorganger (vanaf links gezien) 15° met de klok mee gedraaid is.
c. Gebruik `push` en `pop` om te zorgen dat `translate(80 + 30,0);` steeds alleen voor een horizontale verplaatsing zorgt.
d. Heb je een vast patroon gevonden dat zich herhaalt? Breid dan het aantal vierkanten uit naar 8 zodat figuur 4.2 ontstaat.

Level 3 bij §1.3 variabelen gebruiken

Open *HGAO3.js* in de browser. Beweeg je muis, om tot een eindresultaat te komen, vergelijkbaar met figuur 4.3. Je moet dit tekenspel namaken.
Open *HGAO3U.js* in jouw *editor*. Regel 9, 14 & 16 zijn leeg.



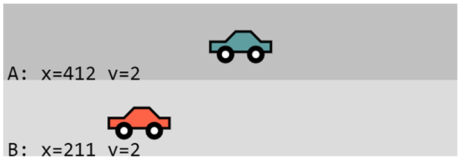
FIGUUR 4.3

Voer de volgende taken uit:

a. Voeg in regel 9 code toe die gebruik maakt van de variabele `hoogte` (zie regel 1) om een *silver* rechthoek te tekenen over de onderste helft van het canvas (zie figuur 4.3).
b. Voeg in regel 14 code toe die op de plaats waar de muis zich bevindt een cirkel met diameter 10 tekent.
c. Voeg in regel 16 een coderegel toe om het programma af te maken volgens het voorbeeld.

Level 4 bij §1.4 loopfunctie inzetten

Open *HGAO4.js* in de browser. Je ziet een race tussen twee auto's. In figuur 4.4 zie je daarvan een momentopname.
Open *HGAO4U.js* in jouw *editor*. In de code zijn al een aantal keuzes gemaakt:



FIGUUR 4.4

- Er zijn variabelen aangemaakt voor de coördinaten en de snelheid van de twee auto's.
- Er is een variabele `finish` gemaakt gedeclareerd, die aangeeft voor welke waarde van *x* de auto de finish heeft bereikt.
- Er zijn tekstregels gemaakt, om de actuele x-positie van beide auto's te kunnen weergeven. Hierbij is voor auto B de functie `round` gebruikt.
- Er is een functie `tekenAuto` geprogrammeerd, waarmee de twee auto's kunnen worden getoond.

Voer de volgende taken uit:

a. Roep de functie `tekenAuto` twee maal aan (regel 24 en 25 zijn hiervoor leeg gelaten) om auto A (met kleur *cadetblue*) en auto B (met kleur *tomato*) te tekenen.
b. Laat de auto's rijden. Gebruik hierbij de gedeclareerde variabelen voor de snelheid van de auto's.
c. Gebruik `constrain` om te zorgen dat de auto's niet verder kunnen rijden dan de door `finish` aangegeven x-positie.
d. Zorg dat behalve de x-positie ook de snelheid van de auto's wordt getoond (zie figuur 4.4).
e. In de eindversie gaat auto B steeds sneller. Declareer bovenaan een variabele `acceleratieB` met de waarde `0.01`.
f. Zorg dat de snelheid van B elke loop toeneemt met `acceleratieB`.
g. Zorg dat de snelheid van B met één cijfer achter de komma op het scherm wordt getoond.

Level 5 bij §1.5 functies maken

Open *HGAO5.js* in de browser en maak kennis met Toby (figuur 4.5).
Open HGAO5U.js in jouw editor. Vanaf regel 17 wordt Toby getekend.

FIGUUR 4.5

Voer de volgende taken uit:

a. Maak een functie `tekenToby` om Toby te tekenen. Zorg dat aan de functie een parameter `s` kan worden meegegeven, die aangeeft op welke schaal Toby moet worden getekend.
b. Roep `tekenToby` aan. Gebruik hierbij als argument de variabele `schaal` die bovenaan is gedeclareerd.
c. Vanaf regel 10 is een if-else-structuur geprogrammeerd. Pas deze aan, zodanig dat de variabele `schaal` met 1% toeneemt als er op de muis wordt geklikt en anders de waarde `1` krijgt.

Level 6 bij §1.6 voorwaarden

Open *HGAO6.js* in de browser. Gebruik de pijltjestoetsen (links en rechts) om Toby veilig te laten landen tussen de obstakels zoals in figuur 4.6. Je moet dit spelletje namaken.

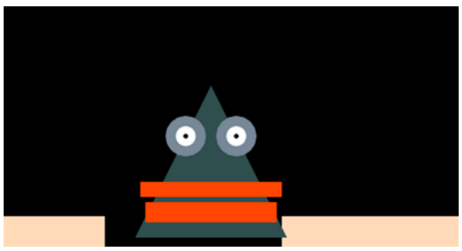
FIGUUR 4.6

Open *HGAO6U.js* in jouw *editor*. In de code zijn al een aantal keuzes gemaakt:

- Er zijn variabelen gedeclareerd voor de positie van Toby.
- Er zijn functies gemaakt om Toby en de obstakels te tekenen.

Voer de volgende taken uit:

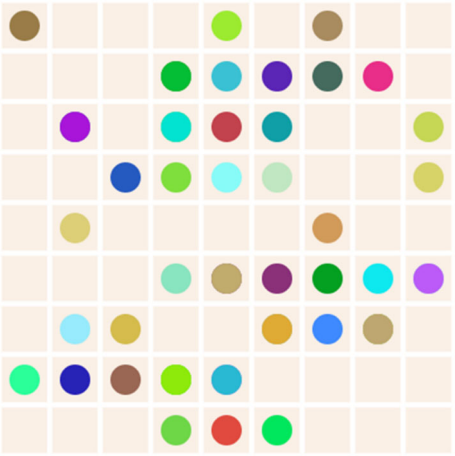
a. Zorg dat Toby valt. Elke loop moet hij één pixel naar beneden vallen.
b. Zorg dat de achtergrond wit wordt als Toby de onderkant van het canvas raakt. Roep op dat moment `noLoop` aan, zodat Toby stopt met bewegen.
c. Voeg code toe, zodat Toby tijdens het vallen naar links en rechts kan bewegen. Het drukken op een pijltjestoets moet steeds een stap van 5 pixels verplaatsing opleveren.
d. Beperk Toby's beweging: zorg dat Toby links en rechts altijd volledig binnen het canvas blijft.
e. Zorg dat de achtergrond zwart wordt als Toby een obstakel raakt. Roep op dat moment `noLoop` aan, zodat Toby stopt met bewegen. Zie figuur 4.7.



FIGUUR 4.7

Level 7 bij §1.7 vaste herhalingen

Open *HGAO7.js* in de browser. Je ziet een raster met een willekeurig stippenpatroon, zoals in figuur 4.8. Laad de pagina enkele keren opnieuw, zodat je ziet dat zowel de positie als de kleur van de stippen willekeurig is.



FIGUUR 4.8

Open *HGAO7U.js* in jouw *editor*. In de code zijn al een aantal keuzes gemaakt:

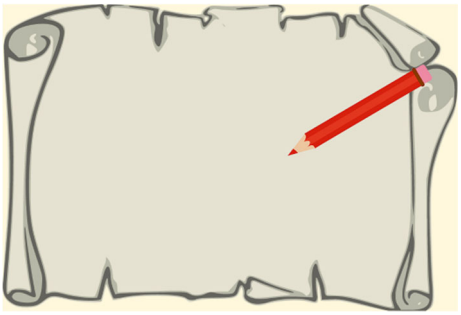
- Er is een variabele `aantal` gedeclareerd die aangeeft hoeveel stippen er moeten worden getekend.
- De functie `tekenRaster` tekent op dit moment één cel van het raster in kolom 4 van rij 1.
- De functie `tekenStip` tekent op dit moment één stip met kleur *darkgoldenrod* in kolom 4 van rij 1 met behulp van de parameters x en y.

Voer de volgende taken uit:

a. Pas de functie `tekenRaster` aan, zodat een raster van 9 × 9 cellen wordt getoond. Gebruik hiervoor een vaste herhaling.
b. Pas regel 11 aan, zodat de stip in het midden van een willekeurige cel in het raster wordt geplaatst.
c. Gebruik een vaste herhaling om meerdere stippen in het raster te plaatsen. Gebruik hierbij de variabele `aantal`.
d. Pas de functie `tekenStip` aan, zodat de stippen een willekeurige kleur krijgen.
e. Verklaar dat je bijna nooit werkelijk 50 stippen te zien krijgt.
f. EXTRA: Wil je een scherm met dansende stippen? Haal dan de regel met `noLoop` weg. Voeg eventueel `frameRate(10);` toe aan de *setup*, om te voorkomen dat je hoofdpijn krijgt…

Level 8 bij §2.2 sprites

Open *HGAO8.js* in de browser. Je ziet een stuk perkament en een roodpotlood, dat meebeweegt met de muis. Merk op dat de cursor samenvalt met de punt van het potlood tot een zekere grens: de potloodpunt blijft altijd binnen bepaalde marges op het canvas.



FIGUUR 4.9

Open *HGAO8U.js* in jouw *editor*. In de code zijn al een aantal keuzes gemaakt:

- Er zijn variabelen gedeclareerd (`p` & `perkament`) die bedoeld zijn om een afbeeldingsobject in op te slaan
- Er zijn variabelen gedeclareerd (`pX` & `pY`) die straks de coördinaten bevatten waar het potlood `p` moet worden getoond
- Er zijn variabelen gedeclareerd (`margeHorizontaal` & `margeVerticaal`) die aangeven hoe dicht de potloodpunt bij de randen van het canvas mag komen (in pixels).
- De gebruikte afbeeldingen bevinden zich in de map *images* in de sublocatie:
  `backgrounds/perkament.svg`
  `sprites/potlood_400.png`
- Er is een coderegel gemaakt, om de afbeelding van het perkament te laden

Voer de volgende taken uit:

a. Voeg code toe om de perkament-afbeelding vooraf te laden in het object `perkament`.
b. Haal de `//` weg zodat `background(perkament);` wordt uitgevoerd.
   (Laat de andere regel met `background` gewoon staan!)
c. Voeg code toe om de afbeelding van het potlood vooraf te laden in het object `p`.
d. Maak gebruik van de variabelen `pX` en `pY` om het potlood op in het canvas te tonen.
e. Voeg coderegels toe, zodanig dat de waardes `pX` en `pY` afhangen van de positie van de muis.
f. Gebruik de eigenschap *hoogte* van het potlood-object om te zorgen dat de cursor samenvalt met de potloodpunt.
g. Gebruik `margeHorizontaal` en `margeVerticaal` en de functie `constrain` om `pX` & `pY` in te perken, zodanig dat de potloodpunt links en rechts 125 pixels en boven en onder 50 pixels van de rand van het canvas blijft.

Level 9 bij §2.3 arrays

Open *HGAO9.js* in de browser. Als je de muis beweegt zie je een reeks met stippen die het pad van de muis volgt zoals in figuur 4.10. Om precies te zijn wordt er bij de cursor steeds een nieuwe stip gemaakt. Telkens als dat gebeurt, verdwijnt er aan het eind van de reeks een stip, zodat het totaal aantal stippen gelijk blijft.

FIGUUR 4.10

Open *HGAO9U.js* in jouw *editor*. In de code zijn al een aantal keuzes gemaakt:

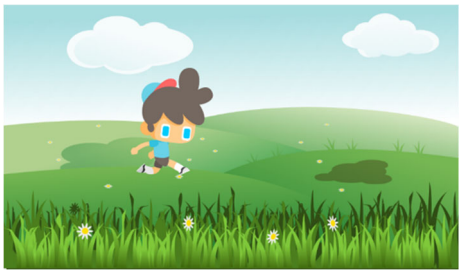
- Er zijn twee arrays `stipX` & `stipY` gedeclareerd, waarin de x- en y-coördinaten van de stippen worden opgeslagen
- Er wordt nu één stip getekend op de coördinaten [60,60] en een RGB-kleur waarvan de hoeveelheid rood eveneens 60 is

Voer de volgende taken uit:

a. Gebruik een herhaling en de arrays `stipX` en `stipY` om de tien stippen op het scherm te tonen. Gebruik voor de hoeveelheid rood van de RGB-vulkeur de x-coördinaat van de stip.
b. Gebruik `push` om tijdens elke loop van de *draw* een waarde aan de arrays `stipX` en `stipY` toe te voegen. De nieuwe waardes moeten overeenkomen met actuele x- en y-coördinaat van de muis.
c. Met `shift` verwijder je het eerste element uit een array. Gebruik deze functie om te zorgen dat elke loop van de *draw* de *oudste* stip van de lijst (aan het eind van de keten) verdwijnt.

Level 10 bij §2.4 arrays van plaatjes

Open *HGAO10.js* in de browser. Je ziet *Flatboy* springen door een groen graslandschap (figuur 4.11). Het beeldmateriaal is rechtenvrij gedownload via www.gameart2d.com (voor *Flatboy*) en www.publicdomainfiles.com (voor het graslandschap). Kijk eens rond op deze websites. Wie weet doe je inspiratie op voor het gebruik in een eigen spel!



FIGUUR 4.11

Open *HGAO10U.js* in jouw *editor*. In de code zijn al een aantal keuzes gemaakt:

- Er zijn variabelen gemaakt voor de x-positie (`flatboyX`) en de breedte en hoogte waarin de sprite moet worden getoond (`flatboyBreedte` & `flatboyHoogte`)
- Het bovengenoemde landschap is geladen met `preload` en wordt door `background` gebruikt.

Voer de volgende taken uit:

a. Declareer `animatie` als lege array en gebruik `preload` om hierin alle beeldjes van een springende *Flatboy* te laden. De gebruikte beeldjes bevinden zich in de map *images* in de sublocatie: `sprites/flatboy/Jump(1).png` (1 t/m 15)
b. Zorg dat (alleen) het eerste beeldje van de array `animatie` getoond wordt op [200,100]. Gebruik `flatboyBreedte` & `flatboyHoogte` voor het juiste formaat van de afbeelding.
c. Zorg dat *Flatboy* gaat springen met behulp van alle beeldjes in de array `animatie`. Gebruik de modulus-functie om te zorgen dat de animatie zich steeds herhaalt.
d. Zorg dat *Flatboy* gaat bewegen vanaf de positie `flatboyX` met een snelheid van 5 pixels per frame.
e. Zorg dat *Flatboy* weer op x-positie `-140` wordt geplaatst als deze rechts 'uit beeld' verdwenen is.

Level 11 bij §2.6 objecten maken

Open *HGAO11.js* in de browser. Je ziet een trein die door een sneeuwlandschap rijdt. Met de pijltjes-toetsen links en rechts van je toetsenbord kun je de snelheid van de trein veranderen.

FIGUUR 4.12

Open *HGAO11U.js* in jouw *editor*. In de code zijn al een aantal keuzes gemaakt:

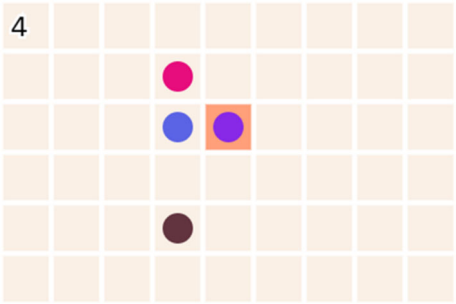
- Er is een object `spoor` gemaakt. Hierover moet de trein gaan rijden.
- In de `preload` zijn afbeeldingen geladen voor zowel de achtergrond als de trein.

Voer de volgende taken uit:

a. Maak een object `trein` met de attributen `x` (waarde: `100`), `y` (`100`), `snelheid` (`-10`), `schaal` (`5`) en `sprite` (`null`).
b. Voeg een methode `toon` toe die er bij aanroepen voor zorgt dat de afbeelding van de trein wordt getoond op het coördinaat [x,y] van het object `trein`. Gebruik hiervoor het attribuut `sprite`. LET OP: `sprite` heeft nu nog de 'waarde' `null`. Hoe los je dat op?
c. Verwijder de `//` voor de regel `trein.toon();` in de *draw* zodat de trein zichtbaar wordt.
d. De trein is op dit moment te groot. Gebruik het attribuut `schaal` van `trein` om de trein `5` maal kleiner te tonen (dan het oorspronkelijke afbeeldingsbestand).
e. Voeg een methode `beweeg` toe die er voor zorgt dat de x-positie van `trein` veranderd, door er de waarde van het attribuut `snelheid` bij op te tellen.
f. Breid `beweeg` uit, zodat de snelheid van de trein met 1 afneemt als op de linker pijltoets wordt gedrukt en met 1 toeneemt als op de rechter pijltoets wordt gedrukt.
g. De trein mag niet te hard rijden: zorg dat de snelheid tussen de -25 en 25 blijft.
h. Als de trein het canvas aan een kant verlaat, moet hij weer aan de andere kant verschijnen. Breid `beweeg` uit, zodat ook aan deze eis wordt voldaan.
i. Pas de beginpositie van `trein` aan, zodat hij in eerste instantie vanaf rechts het canvas binnenrijdt.

Level 12 bij §2.7 object dat antwoordt

Open *HGAO12.js* in de browser. Je ziet het raster van een geluksspelletje. Als je met je muis klikt, verschijnt een stip. Klik je opnieuw, dan verschijnt er een extra stip.



FIGUUR 4.13

Het aantal stippen staat linksboven. Dat aantal is ook jouw score, want de bedoeling is om te blijven klikken, zonder dat er een stip op de door jou geselecteerde cel komt en dat wordt steeds moeilijker als er meer stippen worden geplaatst. Je kunt er voor kiezen om je muis ingedrukt te houden, of om steeds een nieuwe positie te zoeken. Als je af bent (zoals in figuur 4.13) dan stopt het spel. Herlaad de pagina voor een nieuw spel (b.v. met F5).

Open *HGAO12U.js* in jouw *editor*. Het grootste gedeelte van het spel hebben we voor je geprogrammeerd. Het is de bedoeling dat je jezelf 'in de code inleeft' door deze goed te bestuderen. We noemen alvast:

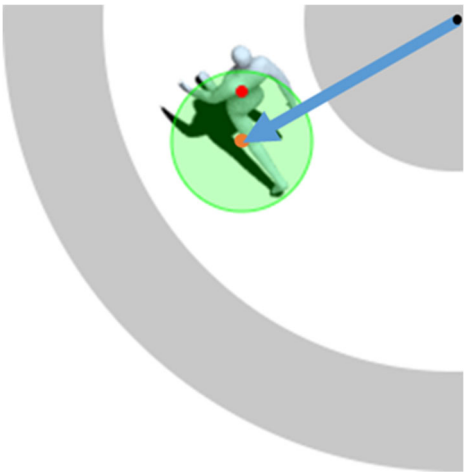
De methode `plaatsStip` wordt aangeroepen als er met de muis wordt geklikt en plaatst dan willekeurig stippen binnen het raster (die worden getoond met `tekenStip`). Na het tekenen van een stip wordt gecontroleerd of de stip zich op dezelfde plek bevindt als de zojuist geplaatste stip. Hiervoor is de methode `controleerRaak`. Deze methode moet `true` antwoorden als de stip zich op de plaats van de met de muis geselecteerde cel bevindt en anders `false`. De methode `controleerRaak` is nog niet af.

Voer de volgende taken uit:

a. Breid de methode `controleerRaak` uit, zodat aan bovenstaande eis wordt voldaan.
b. Had je al ontdekt dat je vals kan spelen? Je kan namelijk ook op de muis klikken, als die zich niet in het canvas bevindt. Pas het spel aan, zodat er alleen een nieuwe stip geplaatst wordt, als de muis zich in het speelveld (dus: op het canvas) bevindt.

Level 13 bij §2.8 klasse van objecten

Open *HGAO13.js* in de browser. Je ziet het bovenaanzicht van een man in een baan. Met de linker- en rechterpijl-toets kun je hem laten wandelen. Als de man op een grijs deel terecht komt, kleuren de randen van de baan donkergrijs. Dit gebeurt slechts bij benadering:



FIGUUR 4.14

De sprite heeft een lastige vorm. Daarom wordt met behulp van de groene cirkel (zie figuur 4.14) bepaald of de man één van de randen raakt. De blauwe pijl is de onderlinge afstand tussen het middelpunt van de baan en het middelpunt van de cirkel. We zeggen dat de man 'goed loopt' zolang deze afstand groter is dan de straal van de kleine cirkel en kleiner dan de straal van de grote cirkel.

Open *HGAO13U.js* in jouw *editor*. Een groot deel staat al klaar:

- Er is een object `speler` met attributen, sprites en methodes.
- Ten behoeve van het ontwikkelen is code toegevoegd, zodat je de man met een muisklik op zijn plaats kan laten staan.

Voer de volgende taken uit:

a. Schrijf het object `speler` om naar een klasse `Man`. Zorg er hierbij voor dat je, als je de `//` voor regel 98 weghaalt, een instantie van de klasse `Man` met de juiste waarden van de attributen maakt.
b. Maak een klasse `Circuit` met in ieder geval het attribuut `kleur` en de methode `teken`. Als de `//` voor regel 97 wordt weggehaald, moet een instantie van `Circuit` worden gemaakt met als straal van de binnenste cirkel `75` en een breedte (van de witte baan) van `100`.
c. Verwijder alle `//` en `/* */` uit de code. Dit zorgt er onder andere voor dat de methode `raakt` van het object `speler` wordt aangeroepen. Deze methode geef op dit moment altijd `false` als antwoord. Breid de methode uit, zodat het eindresultaat van *HGAO13* ontstaat.

Level 14 bij §2.9 array van objecten

In dit laatste level verkennen we het basisprincipe van een *shooter*. Open *HGAO14.js* in de browser. Jij bestuurt een kanon (met de pijltjestoetsen) en moet hiermee de vijand (figuur 4.15) raken. Elke keer dat je op de spatiebalk drukt, vuur je één kogel op de vijand af. Die is niet meteen 'dood': pas als je hem vijf keer raakt is het gelukt. Hoeveel kogels heb jij daar voor nodig? Hoe minder hoe beter! Lukt het jou om slechts vijf kogels te gebruiken?

FIGUUR 4.15

Open *HGAO14U.js* in jouw *editor*. Een groot deel van de code staat al klaar:

- Er is een klasse `Vijand` met onder meer een methode `wordtGeraakt` waaraan een object k (een kogel) wordt meegegeven. Als de kogel de vijand raakt, antwoord de methode met `true`.
- Er is een klasse `Kogel` waarmee nieuwe kogels kunnen worden gemaakt.
- Er is een klasse `Kanon` met onder andere het attribuut `kogels` en de methode `schiet`. De array `kogels` is nu nog leeg. Elke keer als op de spatiebalk wordt gedrukt, moet een nieuwe instantie van de klasse `Kogel` worden gemaakt die wordt toegevoegd aan de array `kogels`.

Voer de volgende taken uit:

a. Breid de methode `schiet` van de klasse `Kanon` uit, zodat er een instantie van de klasse `Kogel` aan de array `kogels` wordt toegevoegd als op de spatiebalk wordt gedrukt. LET OP: de kogels moeten wel de juiste x- en y-coördinaten meekrijgen!

b. De klasse `Kogel` heeft een methode `teken` om een kogel in het canvas te tonen. Breid de *draw* met een herhaling, zodanig dat alle kogels in de array `kogels` te zien zijn.

c. Breid de herhaling uit, zodat de kogels ook gaan bewegen.

d. De methode `wordtGeraakt` van de klasse `Vijand` is al klaar voor gebruik, maar wordt nog niet aangeroepen. Zet deze methode in binnen *de draw* en zorg er hierbij voor dat er een eindscherm (vergelijkbaar met *HGAO14*; zie figuur 4.16) verschijnt, als de vijand geen levens meer heeft.

FIGUUR 4.16