Pong, opdracht 2

Stap 1

Neem het volgende over in een leeg processing project:

```
int prevUpdate;

void setup() {
  size(1000, 600);
}

void draw() {
  float dt = (millis() - prevUpdate) / 1000F;
  prevUpdate = millis();

  // Code...
}
```

Het extra meegegeven stukje code geeft aan hoe lang het geleden is dat er voor het laatst een frame is getekend. Het resultaat hiervan staat in `dt` *en is aangegeven in seconden.*

Teken een cirkel. Houd de positie en snelheid van de cirkel bij in variabelen genaamd `cirkelPositieX`, `cirkelPositieY`, `cirkelSnelheidX` en `cirkelSnelheidY`. Tel voor het tekenen de snelheid van de cirkel, vermenigvuldigd met `dt`, op bij de positie.

Nu zal de cirkel elke seconde het aantal pixels afleggen dat is aangegeven door de snelheid. Dus als de cirkel een `cirkelSnelheidX` *heeft van 100, en een* `cirkelSnelheidY` *van 50, dan zal de cirkel na een seconde 100 pixels in de x-richting afleggen en 50 pixels in de y-richting. Na twee seconden zullen dit in totaal 200 en 100 zijn. Na drie seconden in totaal 300 en 150, enzovoort.*

Start de cirkel in de linkeronderhoek van het canvas en beweeg deze in vijf seconden naar de rechterbovenhoek.

Stap 2

Vervang de `cirkelSnelheidX` en `cirkelSnelheidY` variabelen met `cirkelSnelheid` en `cirkelRichting`. Vervang `cirkelSnelheidX` in je oude berekening met `cos(cirkelRichting) * cirkelSnelheid`. Vervang `cirkelSnelheidY` met `sin(cirkelRichting) * cirkelSnelheid`.

Hoeken en richtingen in programmeren zijn vrijwel altijd in radialen. Dat is in processing ook het geval. Dit betekent dat een heel ronde niet 360 graden is, maar 2 * π radialen. Het symbool π is een vast getal met een waarde van ongeveer 3,14. Een heel rondje is dus ongeveer 6,28 radialen.

Zorg nu dat de richting wijst vanaf de positie van de cirkel naar de muis. Dit kun je als volgt berekenen:

```
cirkelRichting = atan2(mouseY - cirkelPositieY, mouseX - cirkelPositieX);
```

Let op, bij de atan2 operatie is de eerste parameter de y-waarde en de tweede de x-waarde.

Zet de snelheid vervolgens op een waarde naar keuze. Als het goed is beweegt de cirkel naar je muis met de aangegeven snelheid. Als de cirkel bij de muis is aangekomen, trilt de cirkel een beetje. Als je de snelheid negatief maakt, vlucht de cirkel van je muis.

Stap 3

Geef de cirkel nu een willekeurige richting naar keuze. Zorg dat de cirkel stuitert als de cirkel een van de randen van het canvas raakt. Om te kijken of de cirkel een van de verticale randen raakt kun je de volgende voorwaarden gebruiken:

```
cirkelPositieX – cirkelRadius < 0

cirkelPositieX + cirkelRadius > width
```

Bereken zelf de voorwaarden voor de horizontale randen. Om de cirkel om te draaien als deze tegen een verticale rand komt kun je de volgende formule gebruiken:

```
cirkelRichting = PI - cirkelRichting;
```

Voor de horizontale randen is de formule:

```
cirkelRichting = TWO_PI - cirkelRichting;
```

Stap 4

Soms komt de cirkel vast te zitten in een rand. Dit komt doordat na het omdraaien en bewegen, de cirkel nog steeds de rand aanraakt. De cirkel zal dan weer omdraaien en bewegen naar de rand toe. Dit blijft zich herhalen en zo komt de cirkel uiteindelijk niet uit de rand.

Los dit probleem op door de volgende expressie uitvoeren als de cirkel tegen de bovenrand aan komt:

```
cirkelPositieX = cirkelRadius;
```

De expressie voor de onderrand is:

```
cirkelPositieX = width - cirkelRadius;
```

Doe hetzelfde voor de linker- en rechterrand. Bedenkt de expressies zelf.