Empowering Creative Thinking Through Programming

Les 6: Interactivity & Physics

Wat gaan we doen

Interactiviteit (mouse)

Physics (natuurkunde)

Interactiviteit

- Interactiviteit door user input

- keyboard of mouse

- Wij gaan mouse doen

Waarom geen keyboard?

- ingewikkelder
(keyPressed() vs KeyUp/Down met herhaling etc)

- Waar zitten de keys op je telefoon??

Mouse

De positie van de muis (binnen je venster) is op te vragen met:

MouseX voor de horizontale positie
MouseY voor de verticale positie

```
1  // voorbeeld Mouse
2  // door Ton Markus En Vincent Booman, september 2019
3
4  void setup() {
5    size(500, 500);
6  }
7
8  void draw() {
9    println(mouseX);
10 }
11
12
```

There is no step 2…

‘teken een rechthoek waarvan de x-positie bepaald wordt door de mouseX’

```
3
4  void setup() {
5    size(500, 500);
6  }
7
8  void draw() {
9    background(100);
10   rect(mouseX, height/2,50,50);
11 }
12
13
```

2 uitdagingen

1: mouse positie wordt ‘geplakt’ aan linkerbovenkant rect()

2: beweging is onnatuurlijk (geen versnelling/vertraging)

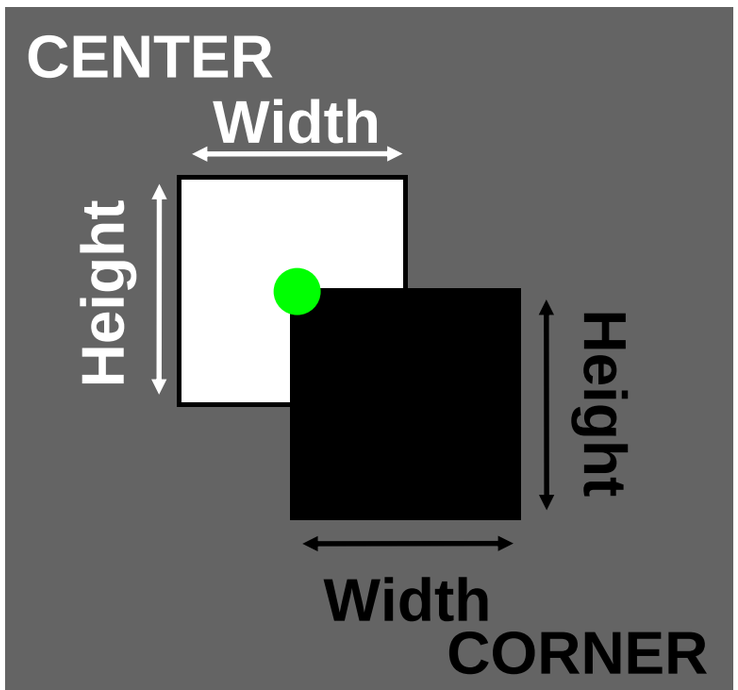
Uitdaging 1

De oplossing hiervoor het rectMode()

- stelt in waarvandaan de vorm getekend wordt

  - 2 smaken: CORNER en CENTER
    (Moeten in HOOFDLETTERS)

- rect(x, y, width, height)



In code:

```
7
8  void draw() {
9    background(100);
10
11   fill(255); // witte vulling
12   rectMode(CENTER); // teken de rechthoek vanuit het midden
13   rect(mouseX, mouseY,50,50); // teken rechthoek vanuit muispositie
14
15   fill(0); // zwarte vulling
16   rectMode(CORNER); // teken de rechthoek vanuit de linkerbovenhoek
17   rect(mouseX, mouseY,50,50); // teken rechthoek vanuit muispositie
18 }
19
```

Twee keer exact dezelfde opdracht

Voorbeeld 2

Uitdaging 2

De oplossing hiervoor is natuurkunde

Of eigenlijk de wiskunde achter de natuurkunde

We gaan nu hele moeilijke wiskunde doen om te zorgen dat de player op natuurlijke wijze beweegt…

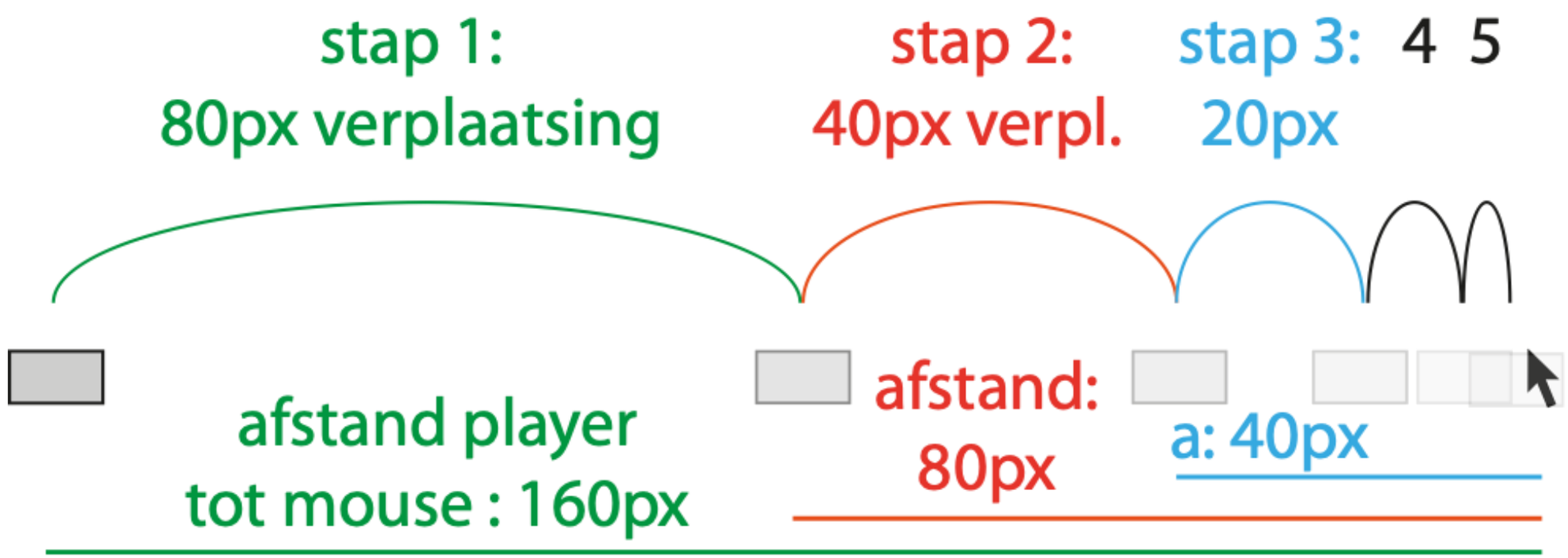
DELEN

/

Om precies te zijn:

De afstand tussen de mouseX en de x-positie van de player delen.

Visueel:

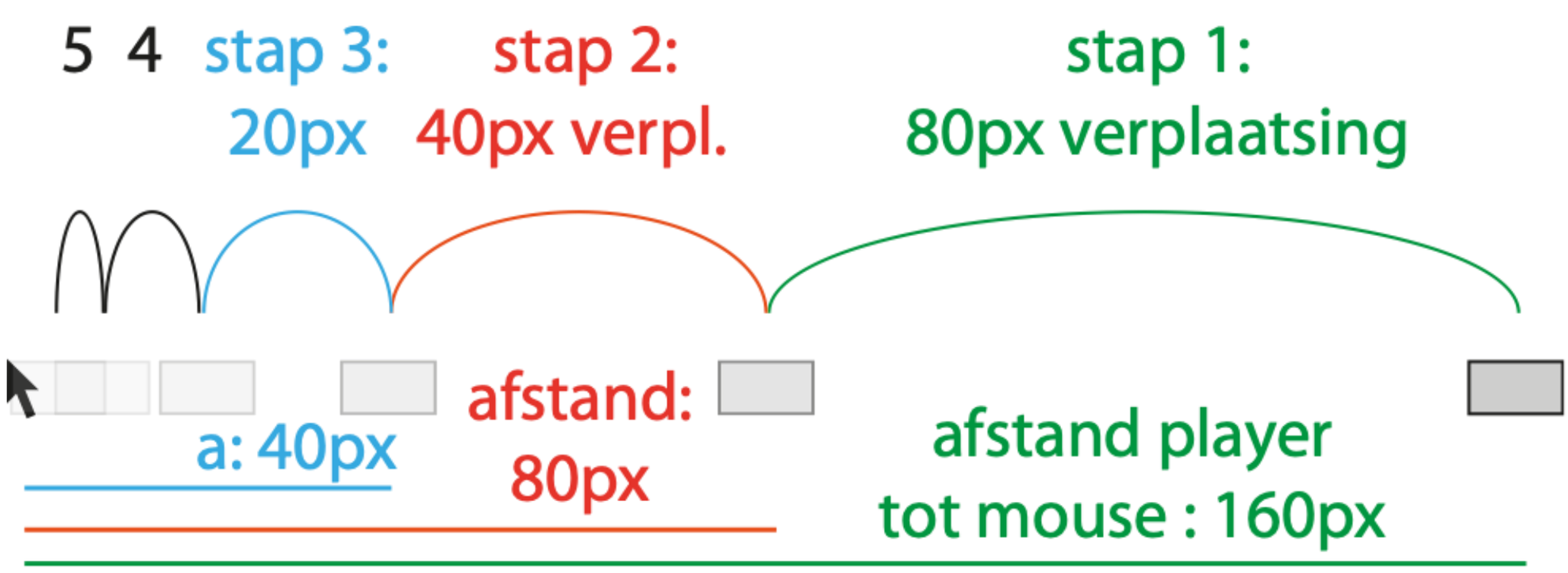


Afstand / 2

Resultaat: kleinere stappen maakt afnemende snelheid

Het werkt ook de andere kant op:

Visueel:



Afstand / 2

Resultaat: kleinere stappen maakt afnemende snelheid

In code

voorbeeld_3

```
1  // voorbeeld massatraaaaagheid
2  // door Ton Markus En Vincent Booman, september 2019
3  int player_x_pos = width/2; // zet de om te beginnen midden in in het venst
4  int vertraging = 10; // hoe hoger het getal, hoe trager
5
6  void setup() {
7    size(500, 500);
8    rectMode(CENTER); // teken de rechthoek vanuit het midden
9   // noCursor();
10 }
11
12 void draw() {
13   background(100);
14
15   int afstandTussenMouseEnPlayer = mouseX - player_x_pos;
16   int playerSpeed = afstandTussenMouseEnPlayer/vertraging;
17   player_x_pos = player_x_pos + playerSpeed;
18   rect(player_x_pos, 100, 50, 50); // teken rechthoek vanuit muispositie
19 }
20
21
```

Variabele om traagheid te berekenen (hard getal in code is niet netjes)

Bereken voor ieder frame de nieuwe snelheid

Deze twee moeten je bekend voorkomen

Lesopdracht

A

- Maak een rect() die op 1/3 vanaf de **linker**kant van het venster verticaal met de muispositie meebeweegt. De horizontale positie blijft altijd gelijk

B

- Maak nog een rect() die op 1/3 vanaf de **rechter**kant van het venster op natuurlijke wijze (versnelling/vertraging) verticaal met de muispositie meebeweegt. De horizontale positie blijft altijd gelijk