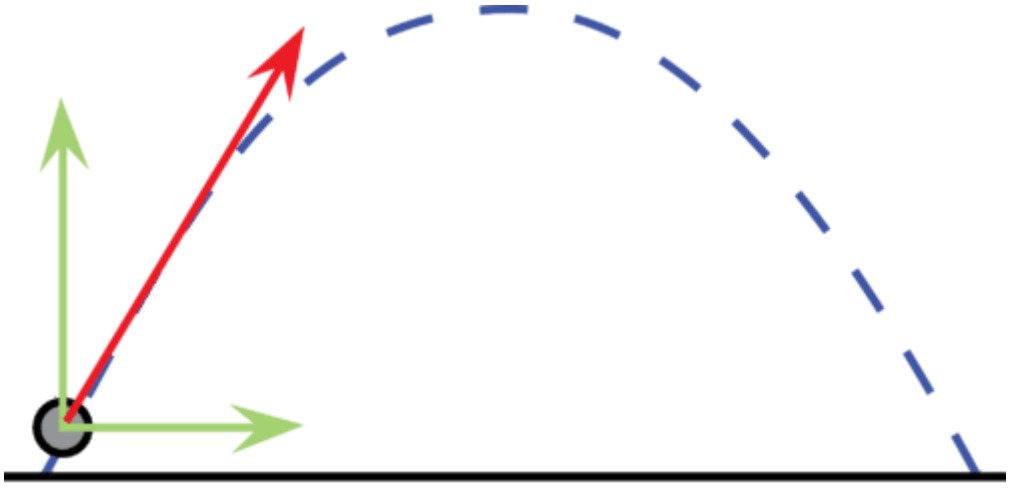
Modellering af projektil i computerspil vha. vektor matematik



Indholdsfortegnelse

Teori om vektorer

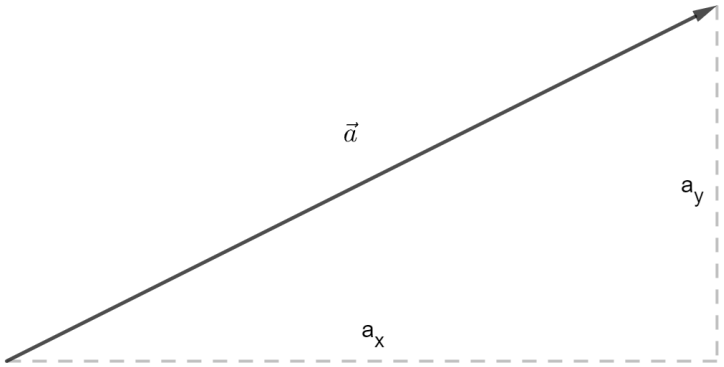
Hvordan skriver man en vektor

For at forstå hvordan et projektil bevæger sig, skal man først vide noget omkring vektorer.

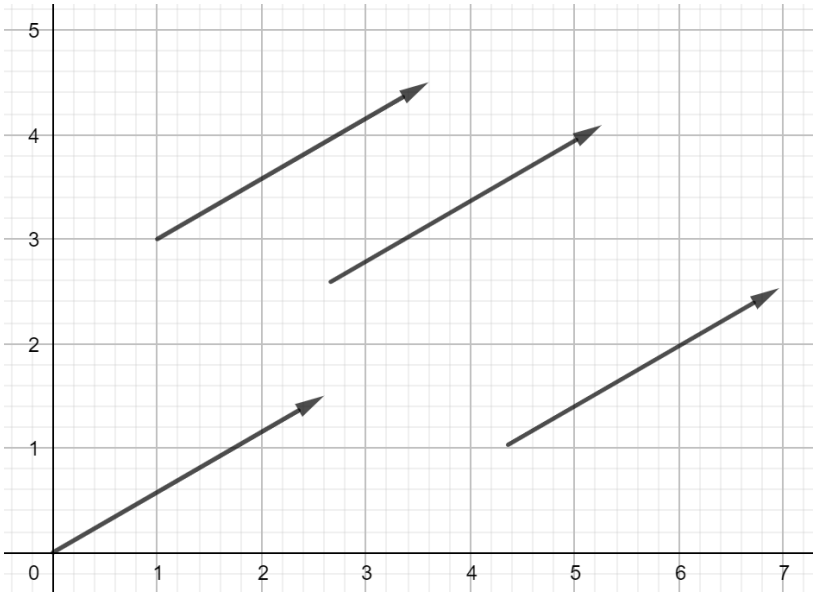
Den simpleste forklaring af en vektor, er en pil med en retning og en længde. Man skriver oftest en vektor som koordinater, disse koordinater fortæller hvor lang vektoren er i henholdsvis x-aksen og y-aksen retning. Men skriver koordinater lodret i parentes.

$$\vec{a} = \begin{pmatrix} x \\ y \end{pmatrix}$$

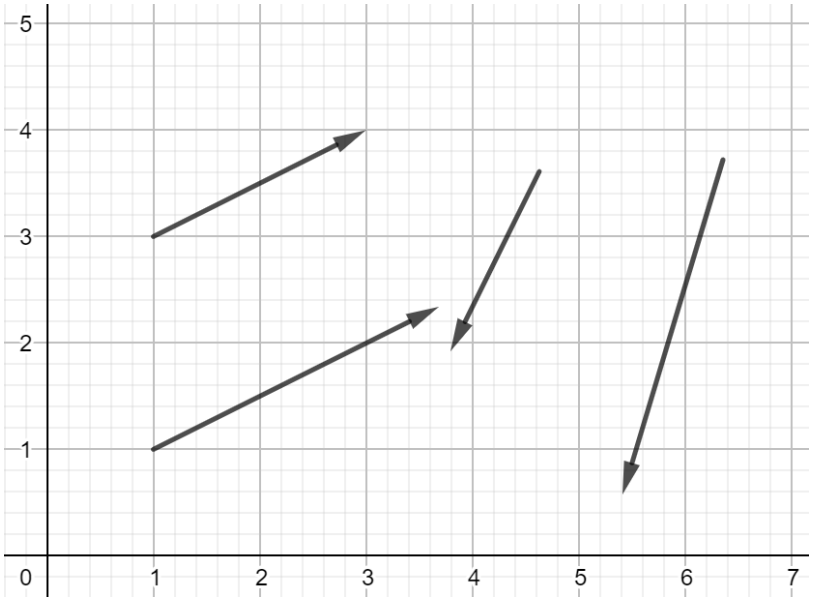
Vektorer kan beskrives på to forskellige måder, numerisk og visuelt. Numerisk betegnelse er den måde der er beskrevet ovenfor, hvor man skrive x og y komponenterne. Den anden måde at beskrive en vektor på er visuelt, dette kan gøres ved at forestille sig et koordinatsystem, og så tegne en linje ud af x-aksen som er x lang, og tegne en linje ud af y-aksen som er y lang. Derefter tegnes en linje fra $(0; 0)$ til $(a_x; a_y)$



Når man tegner en vektor i et koordinatsystem, så er det lige meget hvor vektoren har sit startpunkt, så længe at længden og retningen er den samme. Det betyder at alle disse vektorer er ens



Min disse vektorer er ikke ens, fordi de ikke har samme længde og retning



Matematik med vektorer

Inden vi kan begynde at regne med vektorer, skal vi forstå forskellen mellem vektorer og normale tal, og hvad de hedder. Et normalt tal bliver også kaldt en skalar, så for eksempel længden på en vektor er en skalar, men selve vektoren er stadig en vektor.

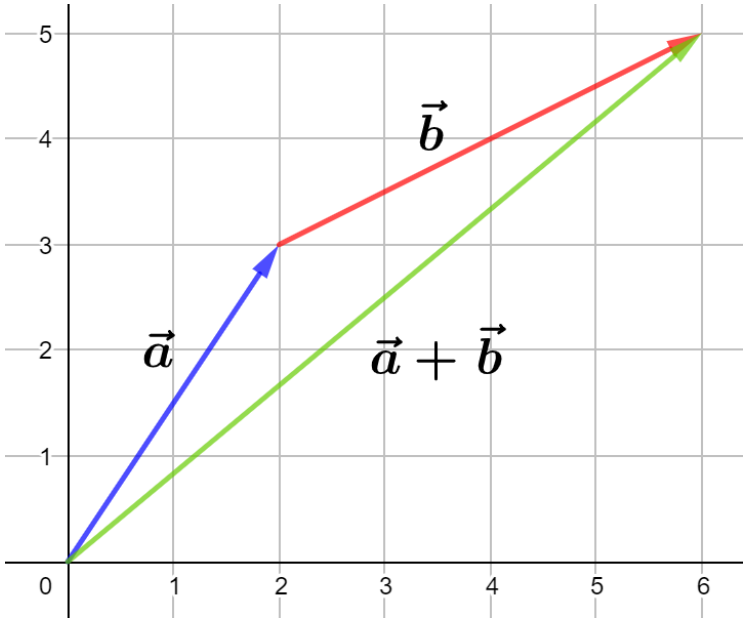
Addition

Du kan plusse to vektorer sammen, men du kan ikke plusse en vektor med en skalar.

Forestil dig at du har to vektorer $\vec{a} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$ og $\vec{b} = \begin{pmatrix} 4 \\ 2 \end{pmatrix}$, hvis du vil lægge dem sammen, skal du lægge vektorens komponenter sammen, det ser sådan her ud

$$\vec{a} + \vec{b} = \begin{pmatrix} 2 \\ 3 \end{pmatrix} + \begin{pmatrix} 4 \\ 2 \end{pmatrix}$$
$$\vec{a} + \vec{b} = \begin{pmatrix} 2+4 \\ 3+2 \end{pmatrix}$$
$$\vec{a} + \vec{b} = \begin{pmatrix} 6 \\ 5 \end{pmatrix}$$

Man kan også vise hvordan man lægger to vektorer sammen visuelt, så skal man forestille sig den første vektor starter i $(0; 0)$ og den anden vektor starter der hvor den første slutter, så tegner man en pil fra første vektors startpunkt til anden vektors slutpunkt, det ser sådan her ud.

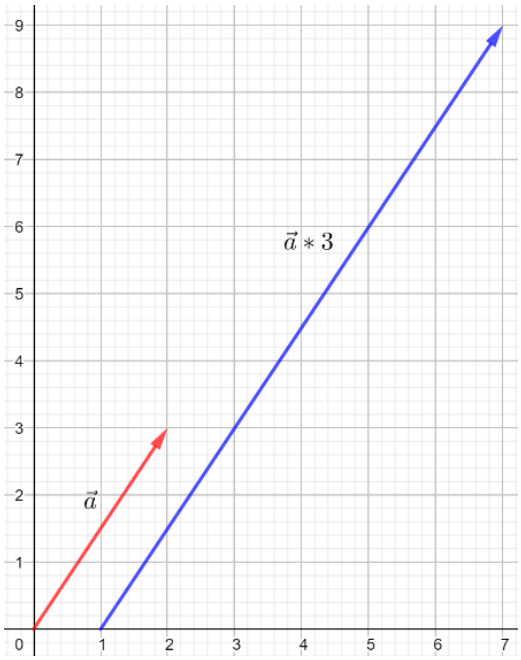


Multiplikation

Når du skal lave multiplikation, kan du ikke gange to vektorer sammen, men du kan godt gange en vektor med en skalar. Hvis du ganger en vektor med en skalar skal du gange hver af vektorens komponenter med skalaren. Numerisk ser den sådan her ud, vi har vektoren $\vec{a} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$, og skalaren 3

$$\vec{a} \cdot 3 = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \cdot 3$$
$$\vec{a} \cdot 3 = \begin{pmatrix} 2 \cdot 3 \\ 3 \cdot 3 \end{pmatrix}$$
$$\vec{a} \cdot 3 = \begin{pmatrix} 6 \\ 9 \end{pmatrix}$$

Hvis man skal vise hvordan man ganger visuelt så kan man forestille sig man tager sin vektor, og forlænger den x gange, i dette tilfælde forlænger vi vores vektor $\vec{a}$ 3 gange.



Subtraktion

Når du trækker to vektorer fra hindanen, foregår det på samme som addition, så hvis vi forestiller os at vi har to vektorer, $\vec{a} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$ og $\vec{b} = \begin{pmatrix} 4 \\ 2 \end{pmatrix}$ så ser subtraktion sådan her ud numerisk

$$\vec{a} - \vec{b} = \begin{pmatrix} 2 \\ 3 \end{pmatrix} - \begin{pmatrix} 4 \\ 2 \end{pmatrix}$$
$$\vec{a} - \vec{b} = \begin{pmatrix} 2 - 4 \\ 3 - 2 \end{pmatrix}$$
$$\vec{a} - \vec{b} = \begin{pmatrix} -2 \\ 1 \end{pmatrix}$$

Division

Du kan ikke divider to vektorer, så du skal dividere en vektor med skalar. Numerisk ser den sådan her ud, vi har vektoren $\vec{a} = \begin{pmatrix} 6 \\ 4 \end{pmatrix}$, og skalaren 2.

$$\vec{a} = \begin{pmatrix} 6 \\ 4 \end{pmatrix} \div 2$$
$$\vec{a} = \begin{pmatrix} 6 \div 2 \\ 4 \div 2 \end{pmatrix}$$
$$\vec{a} = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

Analyse af 3D vektor

Vi har et projektil med en positionsvektor $\overrightarrow{V_p}$, tyngdekraftsvektor $\overrightarrow{V_t}$, hastighedsvektor $\overrightarrow{V_h}$

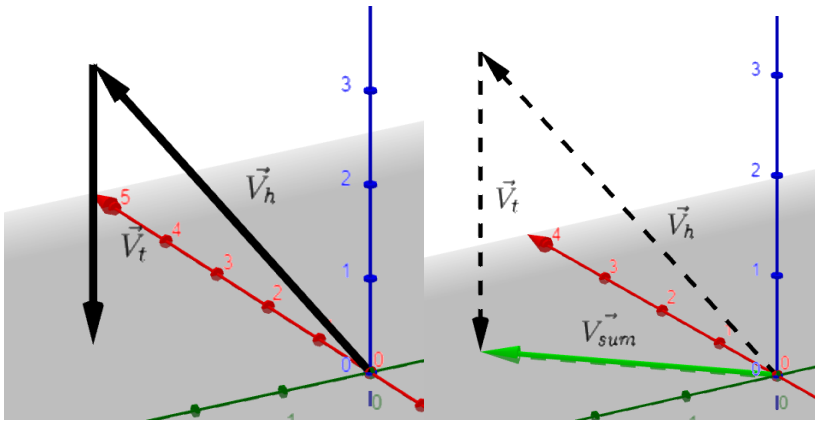
- Positionsvektoren fortæller hvor projektilet befinder sig
- Tyngdekraftsvektoren indeholder tyngdekraften som kun peger ned af
- Hastighedsvektoren er hvordan projektilet bevæger sig

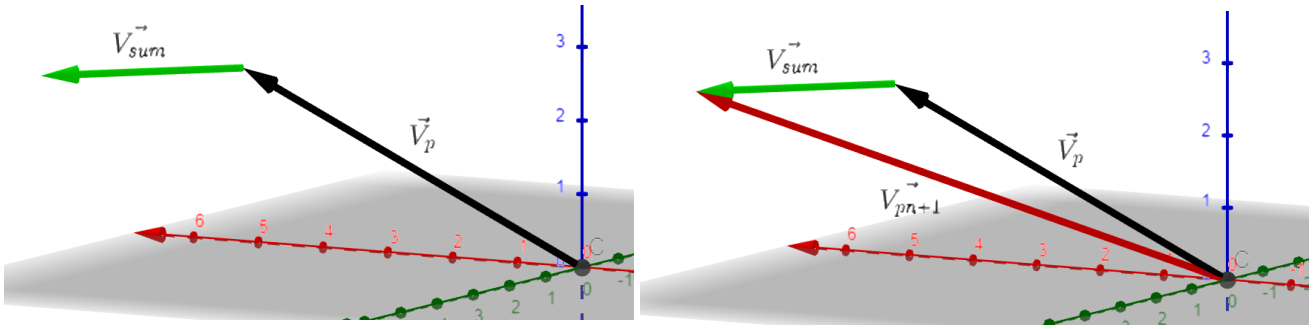
Vi starter med at finde summen af $\overrightarrow{V_t}\ og\ \overrightarrow{V_h}$, det resultere i den ændring vi skal lave af $\overrightarrow{V_p}$, vi vil kalde den resulterende kraft $\overrightarrow{V_{sum}}$

$$\overrightarrow{V_{sum}} = \overrightarrow{V_t} + \overrightarrow{V_h}$$
$$\overrightarrow{V_{sum}} = \begin{pmatrix} \overrightarrow{V_t}_x + \overrightarrow{V_h}_x \\ \overrightarrow{V_t}_y + \overrightarrow{V_h}_y \end{pmatrix}$$

Når vi har fundet den resulterende kraft, kan vi lægge den til positionsvektoren, for at finde positionen efter en iteration

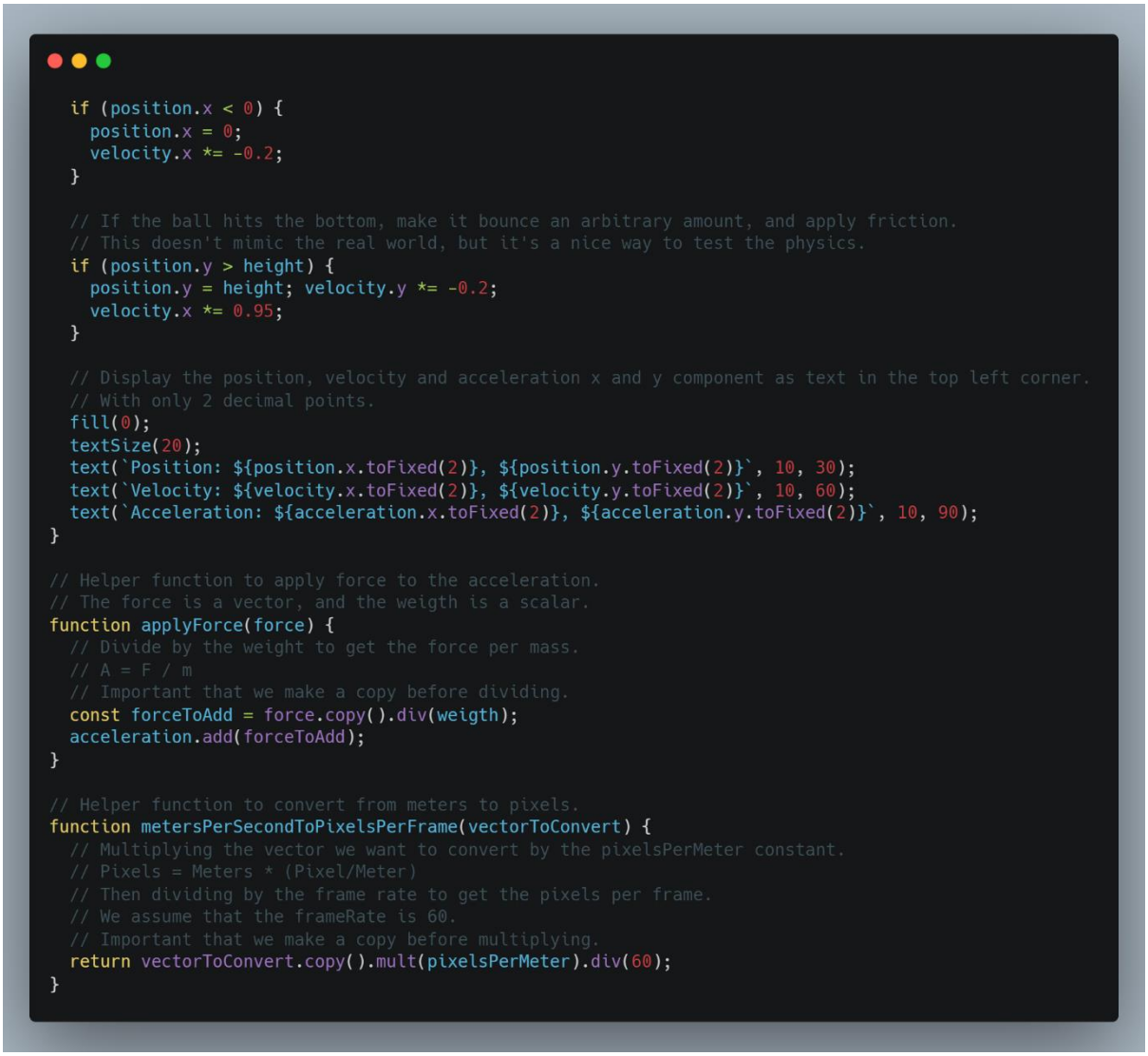
$$\overrightarrow{V_{p\,n+1}} = \overrightarrow{V_p} + \overrightarrow{V_{sum}}$$
$$\overrightarrow{V_{p\,n+1}} = \begin{pmatrix} \overrightarrow{V_p}_x + \overrightarrow{V_{sum}}_x \\ \overrightarrow{V_p}_y + \overrightarrow{V_{sum}}_y \end{pmatrix}$$





Simulation af 2D fysik

```
// Create the physics variables, but don't initialize them yet.
let position;
let velocity;
let acceleration;

// Initialize the weight variable.
// Used to determine how much to apply to the acceleration.
const weigth = .5;

// Setup variables used for change meters to pixels.
let pixelsPerMeter;
const screenWidthInMeters = 100; // m

// Create the gravity force, and initalize it later,
let gravity;

function setup() {
  createCanvas(windowWidth, windowHeight);

  // Initialize the physics variables.
  position = createVector(width / 2, height / 2); // Start the ball in the middle of the screen.
  velocity = createVector(); // Initialize velocity to 0,
  acceleration = createVector(); // Initialize acceleration to 0,

  // Calculate the pixels per meter.
  pixelsPerMeter = width / screenWidthInMeters;

  // Initialize the gravity force. Using the default gravity of earth
  gravity = createVector(0, 9.82);
}

function draw() {
  // Draw a white background.
  background(255);

  // If the mouse is clicked, apply a force to the ball, towards the mouse.
  if(mouseIsPressed){
    applyForce(createVector(mouseX - position.x, mouseY - position.y).div(40));
  }

  // Draw a circle at the position.
  // The circle is the mass.
  fill(0);
  noStroke();
  ellipse(position.x, position.y, 20, 20);

  // Take one step in time.
  position.add(metersPerSecondToPixelsPerFrame(velocity)); // Apply velocity to position in the correct unit.
  velocity.add(acceleration); // Apply acceleration to velocity.
  velocity.mult(0.98); // Apply drag.
  acceleration.mult(0); // Reset acceleration to 0.

  // Apply gravity downwards.
  // And ignore the applyForce function, beacuse gravity is a constant.
  acceleration.add(gravity)

  // Check the bounds of the screen.
  // And make sure the ball doesn't go off the screen.

  // If the ball hits the walls, make it bounce an arbetrary amount. and set the position to the wall.
  if (position.x > width) {
    position.x = width;
    velocity.x *= -0.2;
  }
```



```
  if (position.x < 0) {
    position.x = 0;
    velocity.x *= -0.2;
  }

  // If the ball hits the bottom, make it bounce an arbitrary amount, and apply friction.
  // This doesn't mimic the real world, but it's a nice way to test the physics.
  if (position.y > height) {
    position.y = height; velocity.y *= -0.2;
    velocity.x *= 0.95;
  }

  // Display the position, velocity and acceleration x and y component as text in the top left corner.
  // With only 2 decimal points.
  fill(0);
  textSize(20);
  text(`Position: ${position.x.toFixed(2)}, ${position.y.toFixed(2)}`, 10, 30);
  text(`Velocity: ${velocity.x.toFixed(2)}, ${velocity.y.toFixed(2)}`, 10, 60);
  text(`Acceleration: ${acceleration.x.toFixed(2)}, ${acceleration.y.toFixed(2)}`, 10, 90);
}

// Helper function to apply force to the acceleration.
// The force is a vector, and the weigth is a scalar.
function applyForce(force) {
  // Divide by the weight to get the force per mass.
  // A = F / m
  // Important that we make a copy before dividing.
  const forceToAdd = force.copy().div(weigth);
  acceleration.add(forceToAdd);
}

// Helper function to convert from meters to pixels.
function metersPerSecondToPixelsPerFrame(vectorToConvert) {
  // Multiplying the vector we want to convert by the pixelsPerMeter constant.
  // Pixels = Meters * (Pixel/Meter)
  // Then dividing by the frame rate to get the pixels per frame.
  // We assume that the frameRate is 60.
  // Important that we make a copy before multiplying.
  return vectorToConvert.copy().mult(pixelsPerMeter).div(60);
}
```