Creative Coding

WiSe 2019/20 – Grundlagen

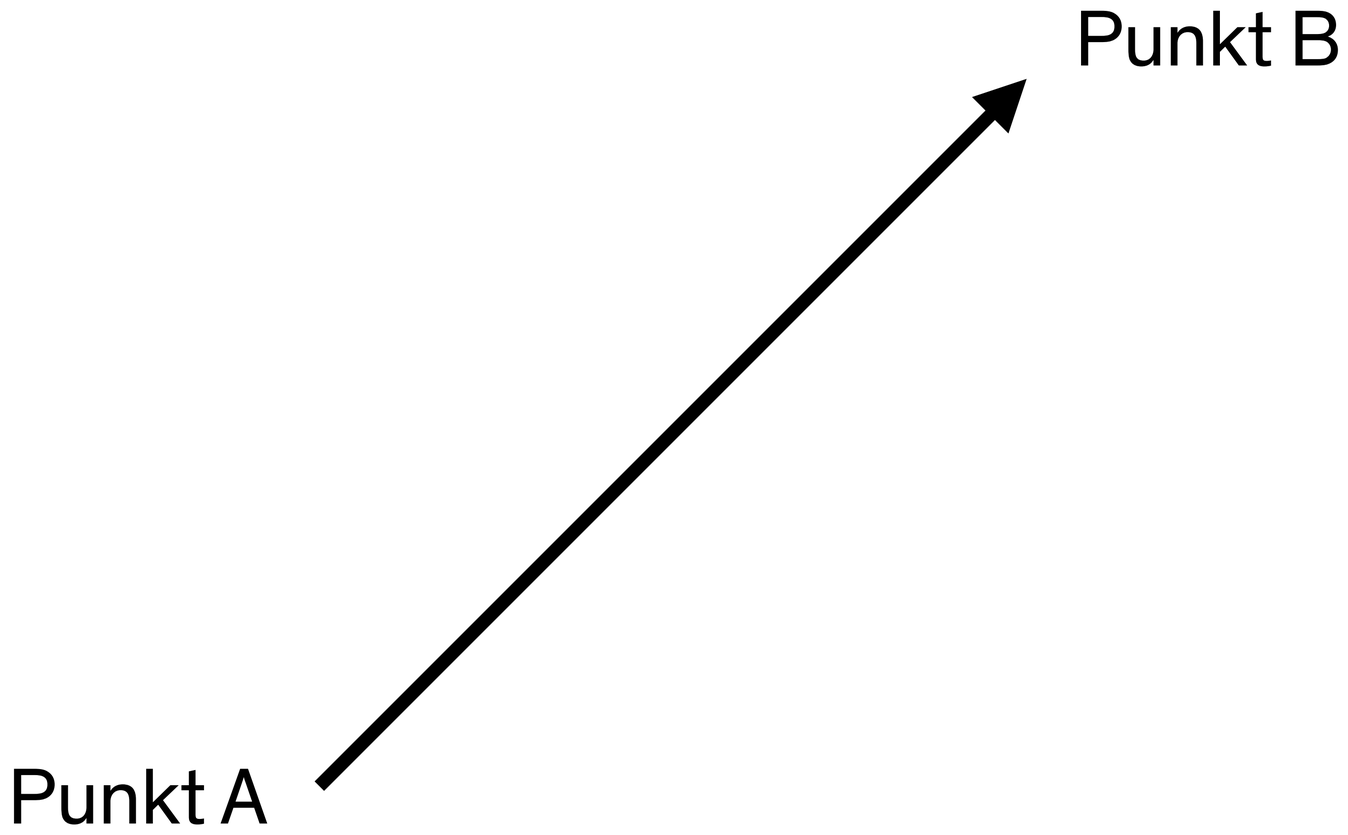
Aufgabe
Vektoren

Nehmt ein Raster von der letzten Aufgabe und erweitert es:
– Erstellt min. 3 Klassen mit Funktionalität
– benutzt ArrayListen
– vermeidet es auf den Canvas direkt zu zeichnen (PGraphics)
– bringt Bewegung in's Spiel (s. Auto, mouseX/ mouseY – was gibt es noch?)
- Wie kann das Raster aufgebrochen werden?

Aufgabe 3 – Bis 15.11.2019

Aufgabe
Vektoren

Vektoren



Vektoren

```
float x = 100;
float y = 100;
float xspeed = 1;
float yspeed = 3.3;

void setup() {
  size(640,600);
  background(255);
  noStroke();
}

void draw() {
  background(255);

  x = x + xspeed;
  y = y + yspeed;

  if( (x > width) || (x < 0) ) {
    xspeed = xspeed * -1;
  }
  if( (y > height) || (y < 0) ) {
    yspeed = yspeed * -1;
  }

  fill(0);
  ellipse(x,y,16,16);
}
```

Vektoren

```
float x = 100;
float y = 100;
float xspeed = 1;
float yspeed = 3.3;

void setup() {
  size(640,600);
  background(255);
  noStroke();
}

void draw() {
  background(255);

  x = x + xspeed;
  y = y + yspeed;

  if( (x > width) || (x < 0) ) {
    xspeed = xspeed * -1;
  }
  if( (y > height) || (y < 0) ) {
    yspeed = yspeed * -1;
  }

  fill(0);
  ellipse(x,y,16,16);
}
```

Vektoren

```
float x = 100;
float y = 100;
float xspeed = 1;
float yspeed = 3.3;

void setup() {
  size(640,600);
  background(255);
  noStroke();
}

void draw() {
  background(255);

  x = x + xspeed;
  y = y + yspeed;

  if( (x > width) || (x < 0) ) {
    xspeed = xspeed * -1;
  }
  if( (y > height) || (y < 0) ) {
    yspeed = yspeed * -1;
  }

  fill(0);
  ellipse(x,y,16,16);
}
```

location = x, y

geschwindigkeit = xspeed, yspeed

beschleunigung = xacceleration, ya

wind = xwind, ywind

ziel = xtarget, ytarget

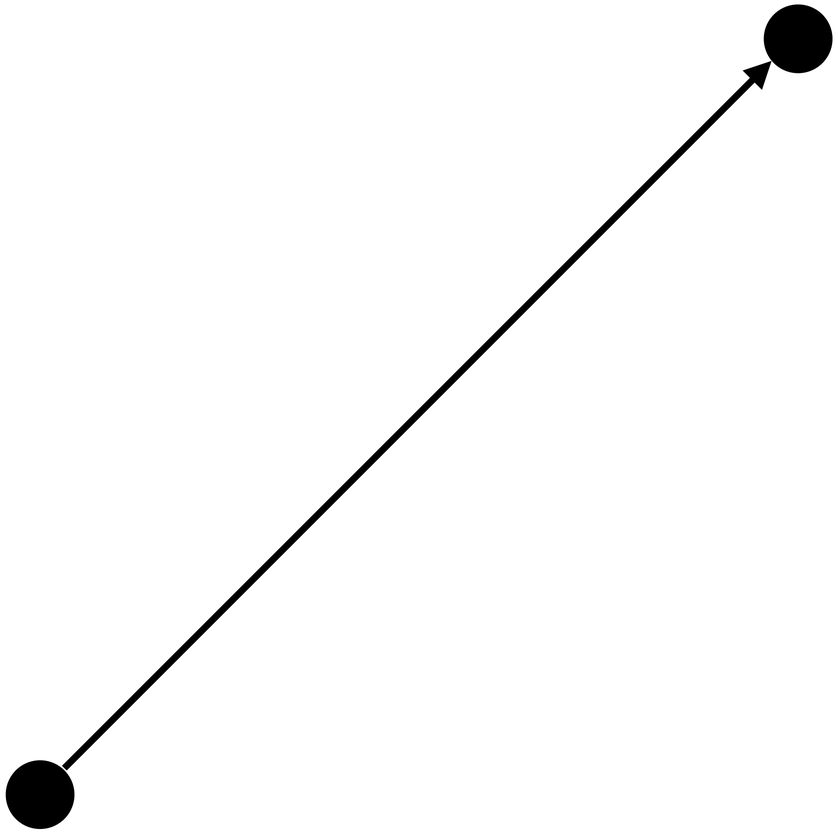
reibung = xfriction, yfriction

Vektoren

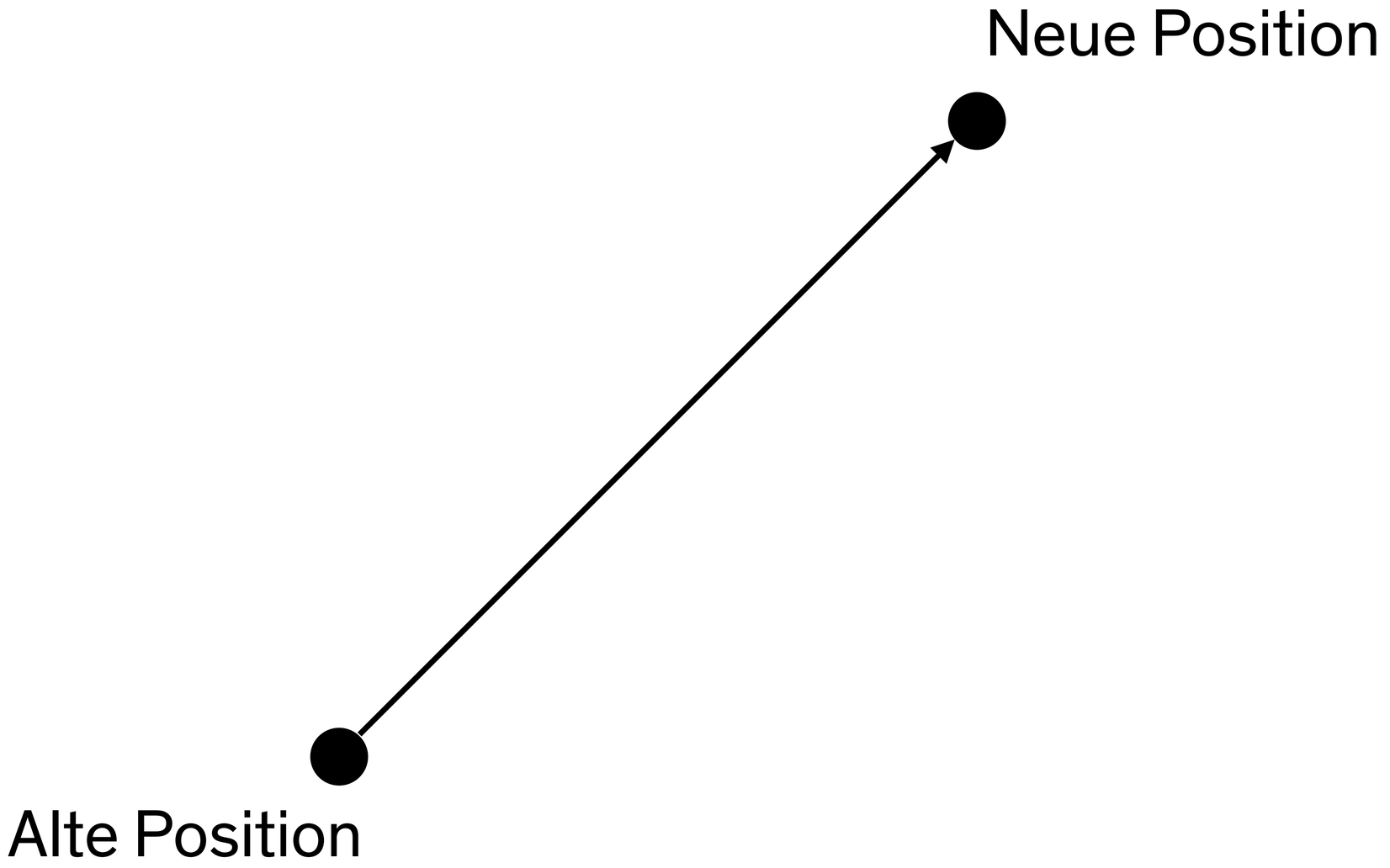
```
float x = 100;
float y = 100;
float xspeed = 1;
float yspeed = 3.3;

void setup() {
  size(640,600);
  background(255);
  noStroke();
}

void draw() {
  background(255);

  x = x + xspeed;
  y = y + yspeed;

  if( (x > width) || (x < 0) ) {
    xspeed = xspeed * -1;
  }
  if( (y > height) || (y < 0) ) {
    yspeed = yspeed * -1;
  }

  fill(0);
  ellipse(x,y,16,16);
}
```

Vektoren

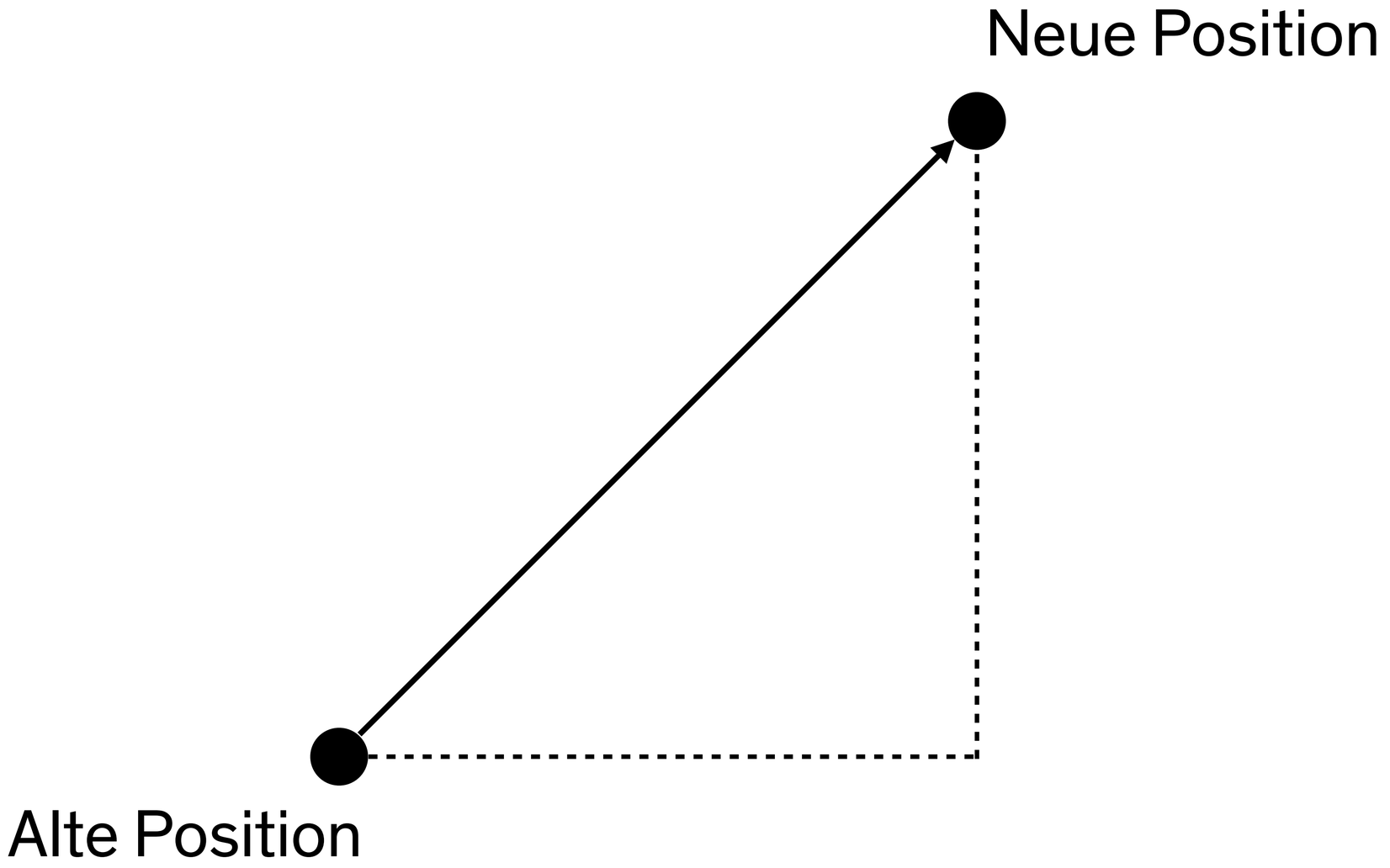
Vector location;
Vector speed;
Vector acceleration;
Vector wind;
Vector target;
Vector friction;



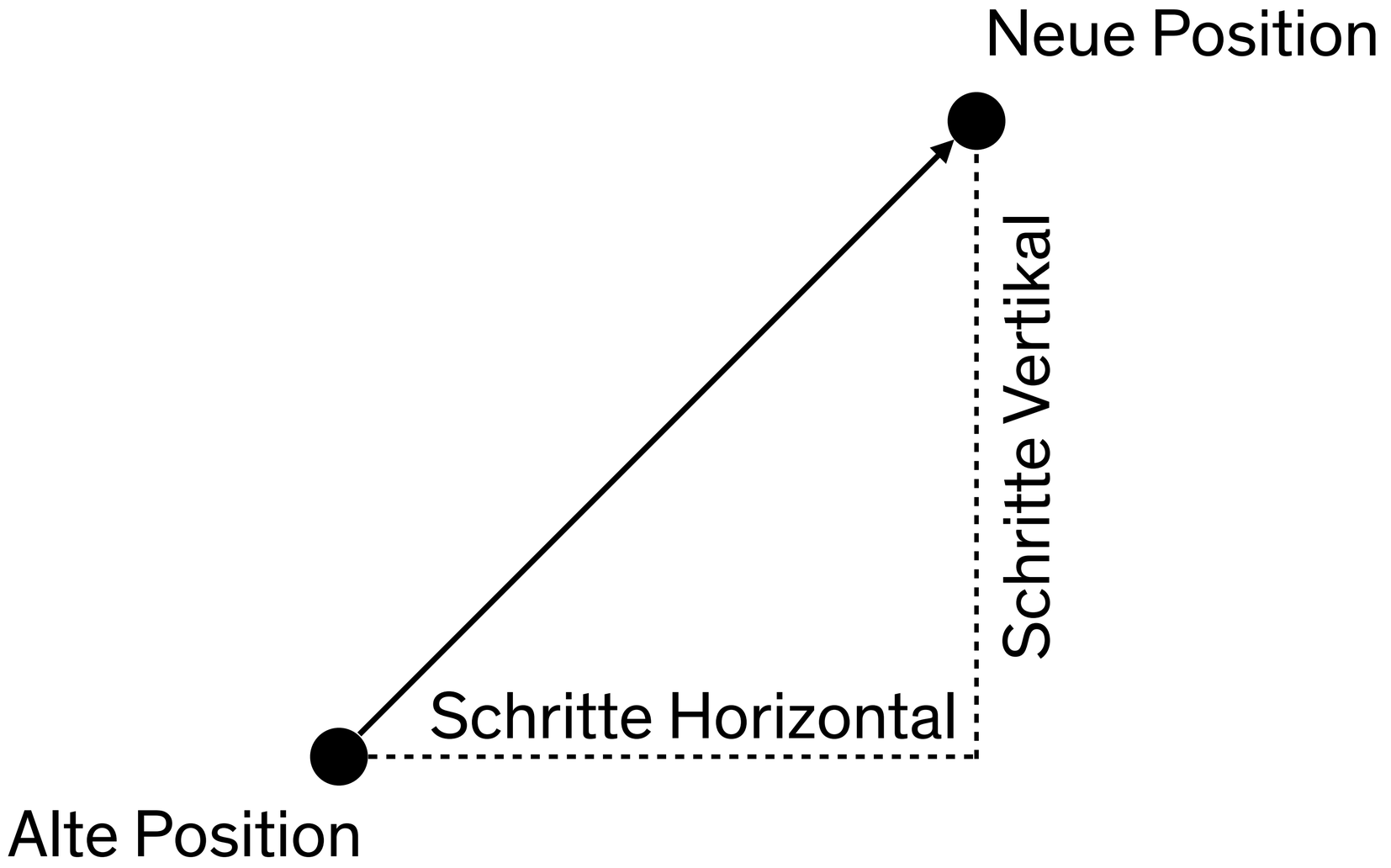
Vektoren



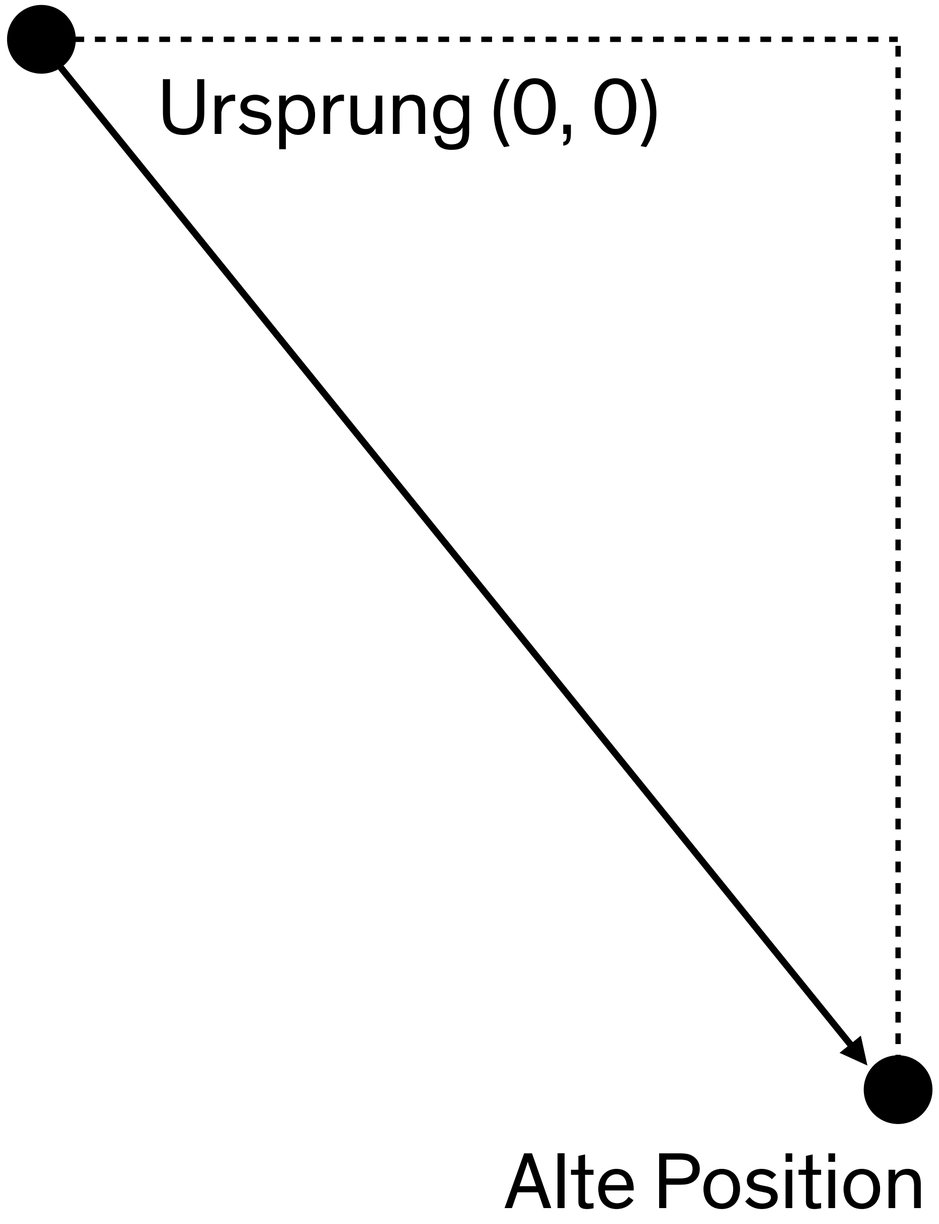
Vektoren



Vektoren


Neue Position
Schritte Vertikal
Schritte Horizontal
Alte Position

Vektoren



Vektoren

```
class PVector {
  public float x;
  public float y;

  PVector(float x_, float y_) {
    x = x_;
    y = y_;
  }
}
```

Vektoren

```
float x = 100;
float y = 100;
float xspeed = 1;
float yspeed = 3.3;


PVector location = new PVector(100, 100);
PVector velocity = new PVector(1, 3.3); // vorher "speed"
```

Vektoren

```
x = x + xspeed;
y = y + yspeed;
```

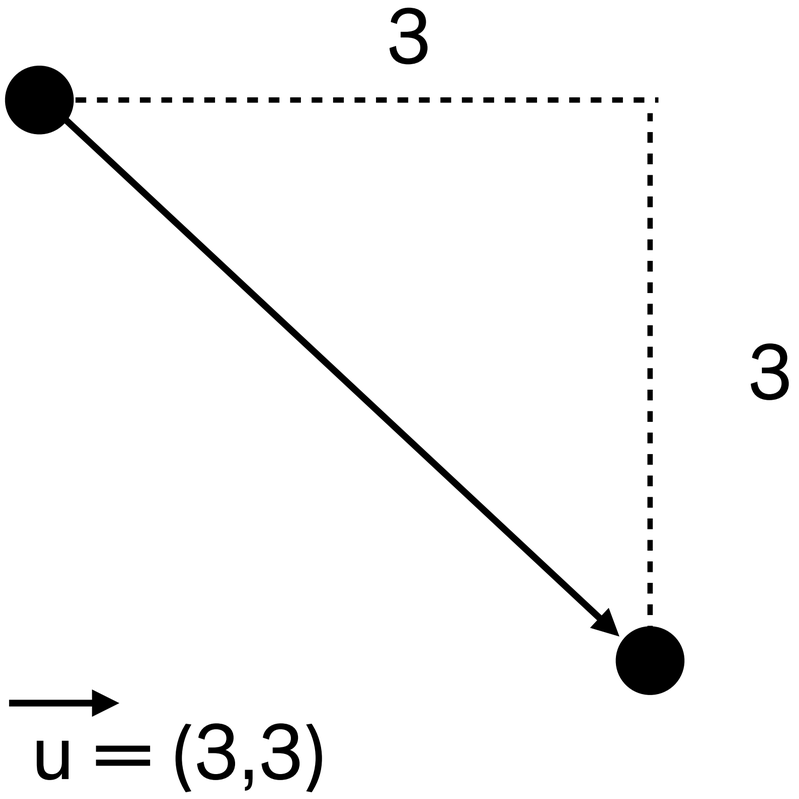
```
location = location + velocity;
```

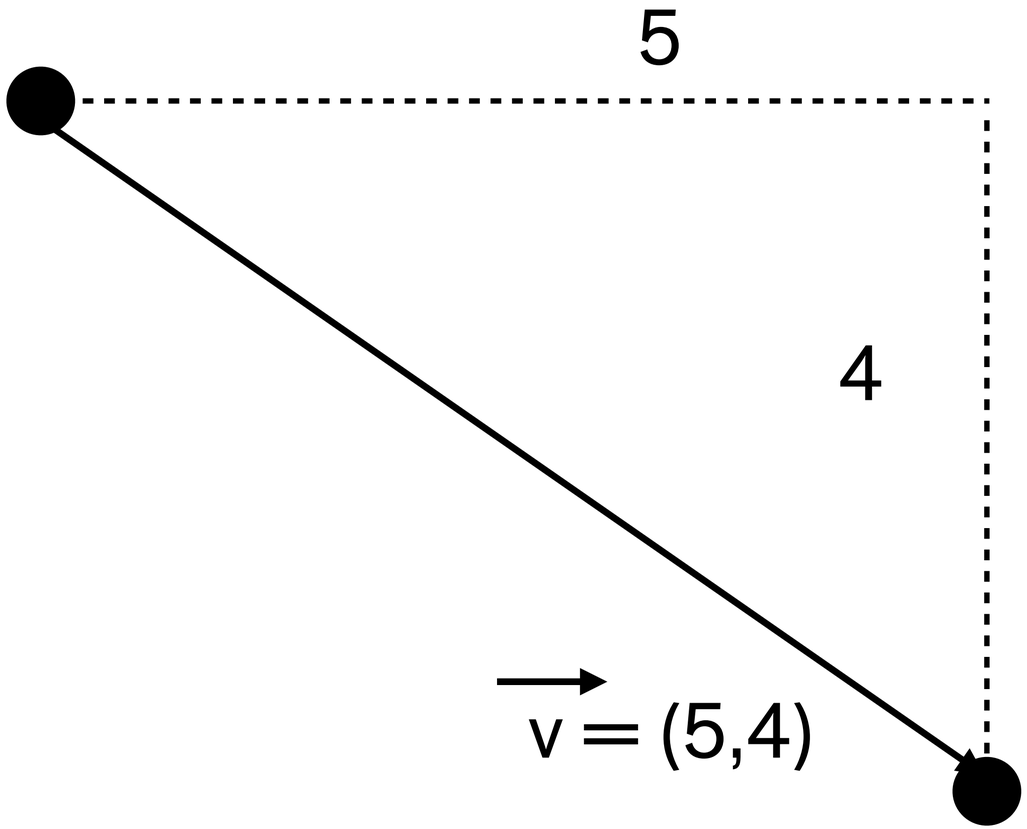
Vektoren

```
x = x + xspeed;
y = y + yspeed;
```

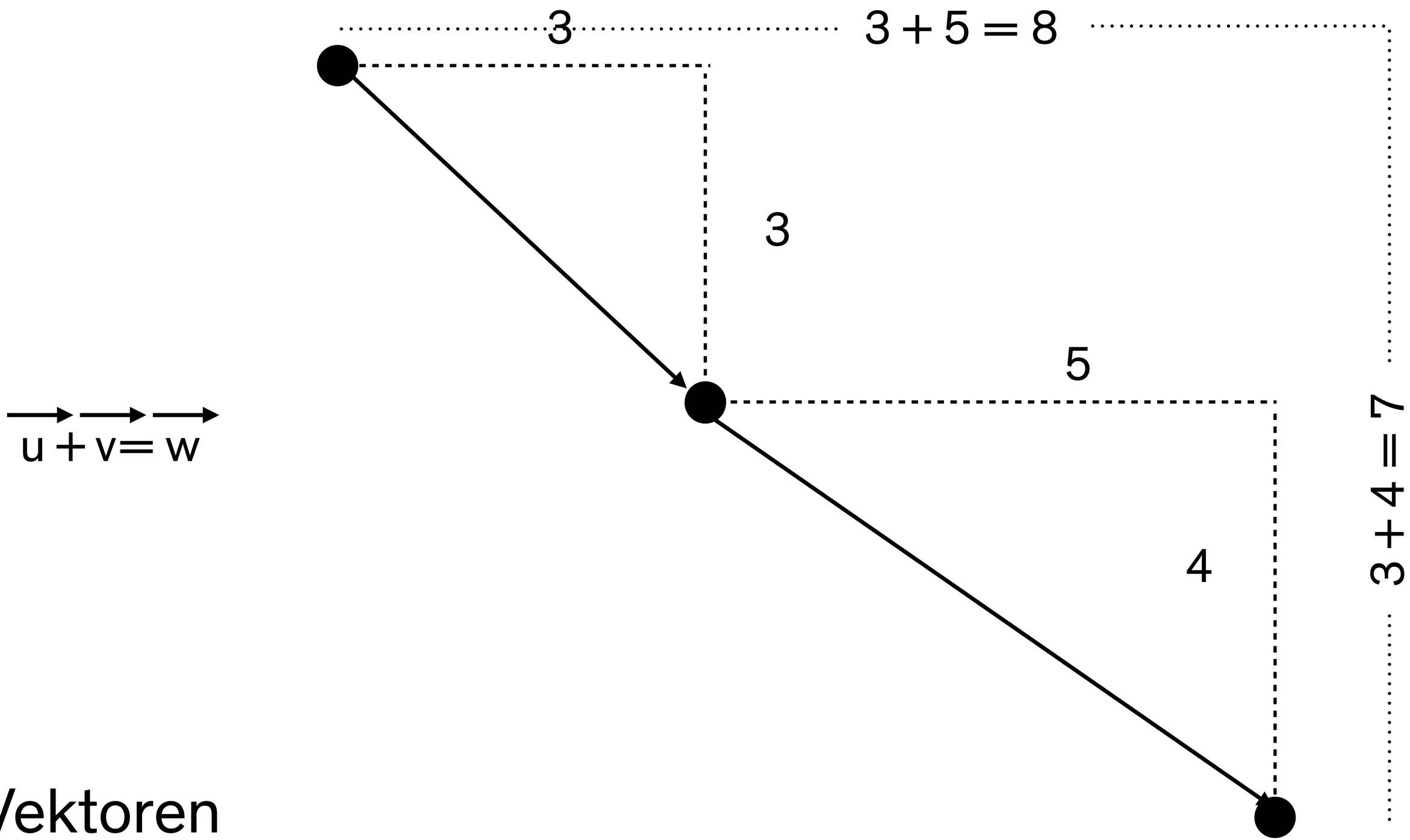
```
location = location + velocity;
```

Vektoren

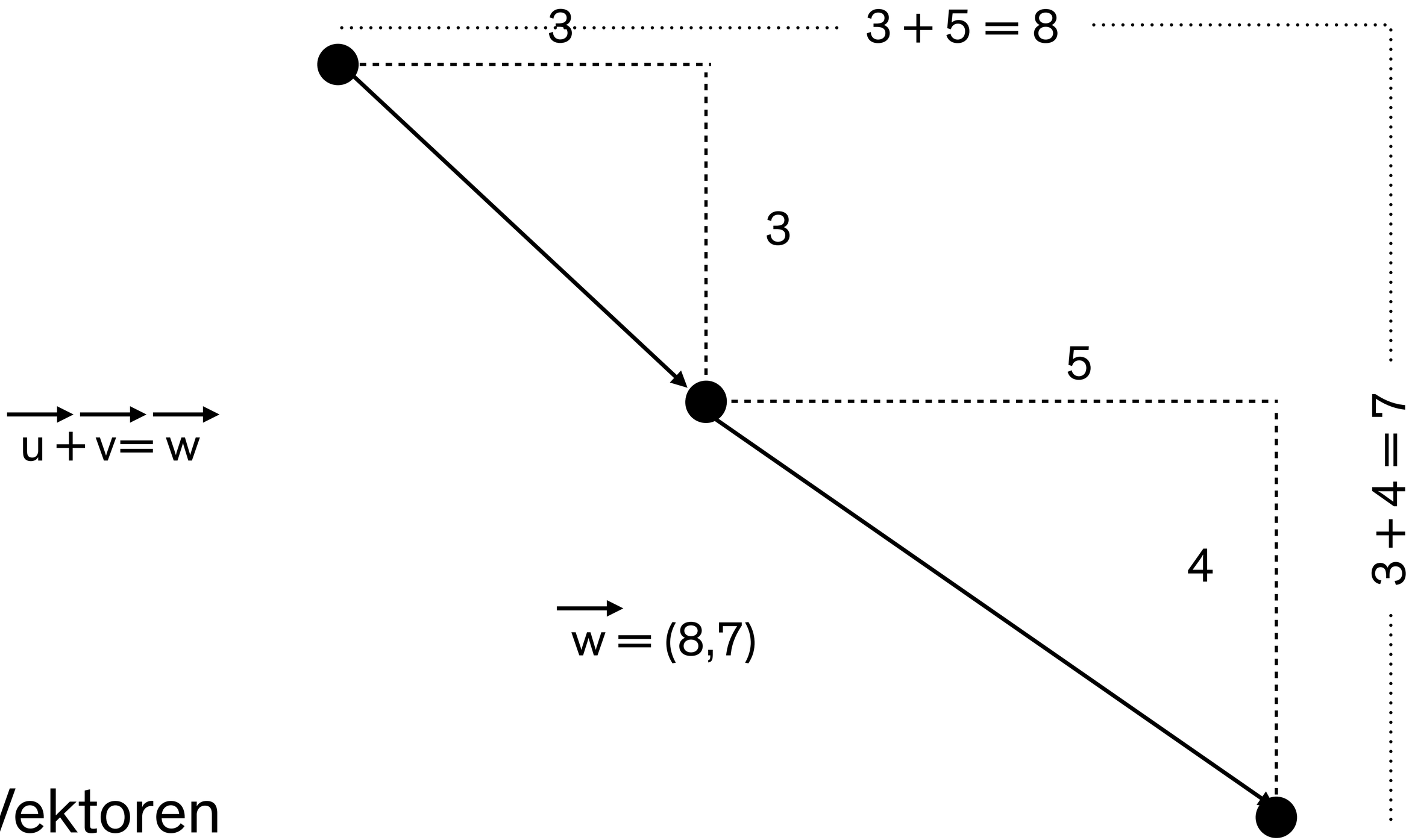




Vektoren



Vektoren



Vektoren

```
x = x + xspeed;
y = y + yspeed;
```

```
location = location + velocity;
location.add(velocity);
```

Vektoren

```
PVector location = new PVector(100, 100);
PVector velocity = new PVector(1, 3.3);

void setup() {
  size(640,600);
  background(255);
  noStroke();
}

void draw() {
  background(255);

  location.add(velocity);

  if ((location.x > width) || (location.x < 0)) {
    velocity.x = velocity.x * -1;
  }
  if ((location.y > height) || (location.y < 0)) {
    velocity.y = velocity.y * -1;
  }

  fill(0);
  ellipse(location.x,location.y,16,16);
}
```

Vektoren

add() — add vectors

sub() — subtract vectors

mult() — scale the vector with multiplication

div() — scale the vector with division

mag() — calculate the magnitude of a vector

setMag() – set the magnitude of a vector

normalize() — normalize the vector to a unit length of 1

limit() — limit the magnitude of a vector

heading() — the 2D heading of a vector expressed as an angle

rotate() — rotate a 2D vector by an angle

lerp() — linear interpolate to another vector

dist() — the Euclidean distance between two vectors (considered as points)

angleBetween() — find the angle between two vectors

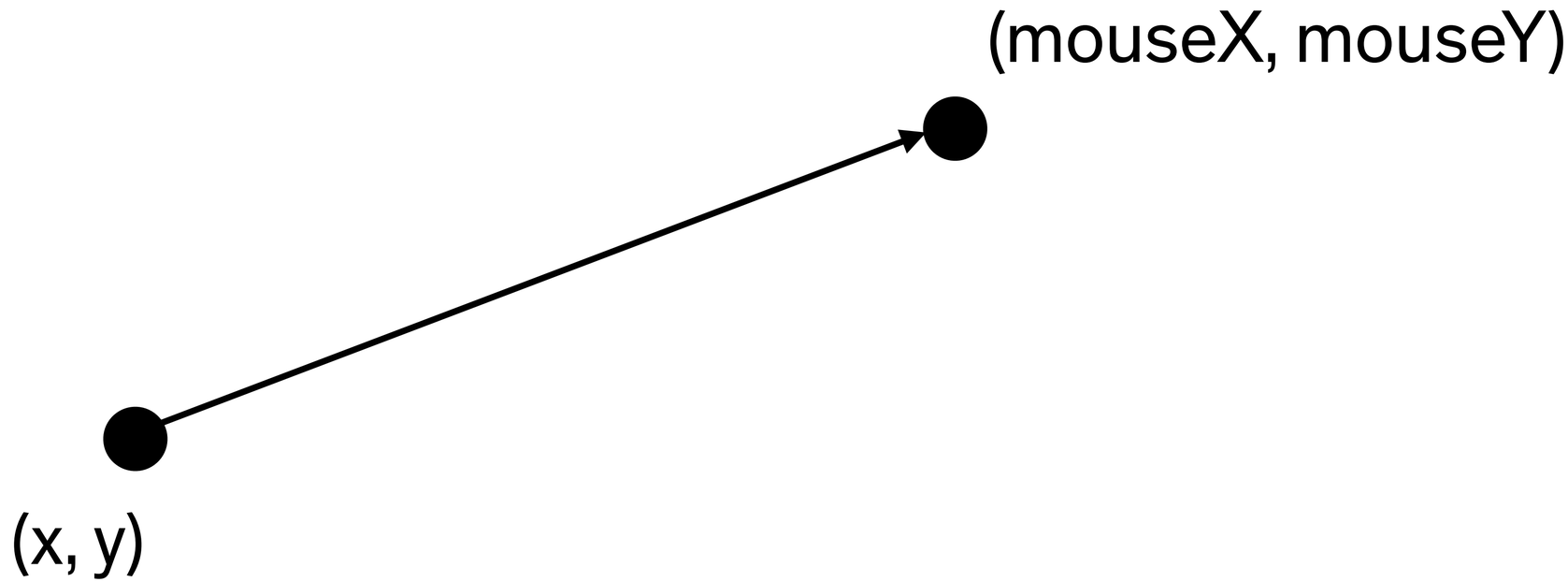
dot() — the dot product of two vectors

cross() — the cross product of two vectors (only relevant in three dimensions)

random2D() – make a random 2D vector

random3D() – make a random 3D vector

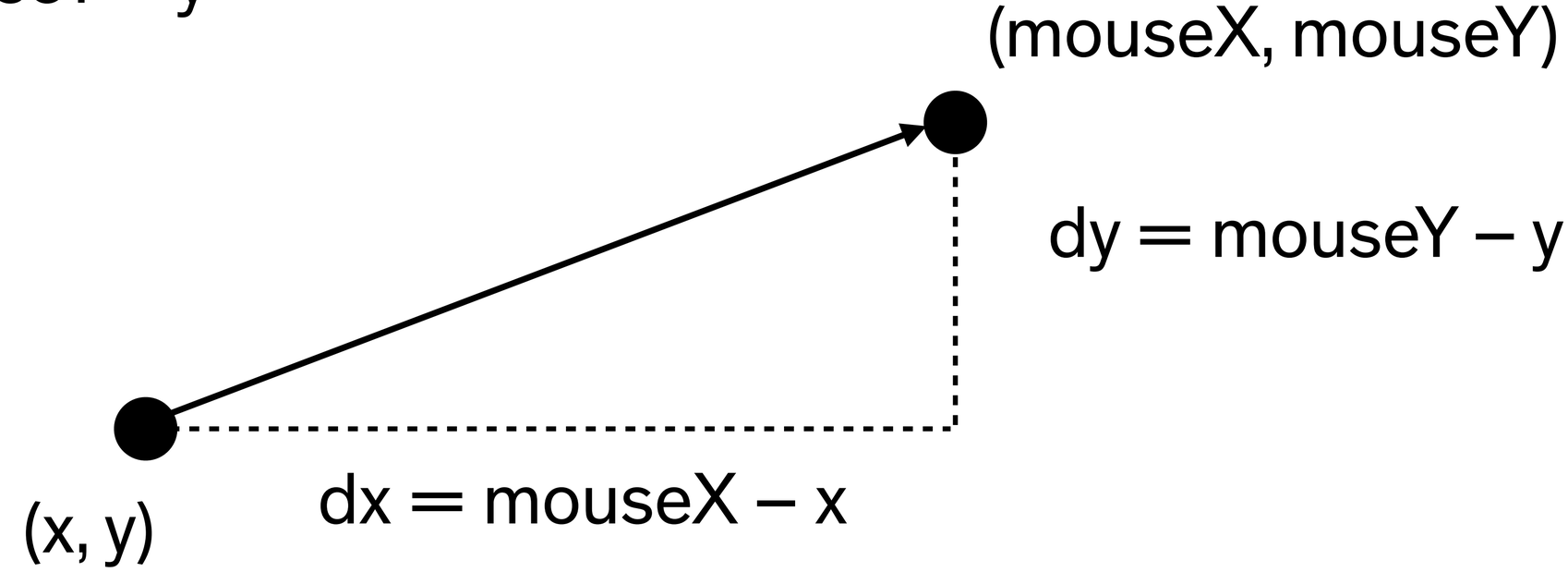
Vektoren



Richtungsvektor

dx = mouseX – x

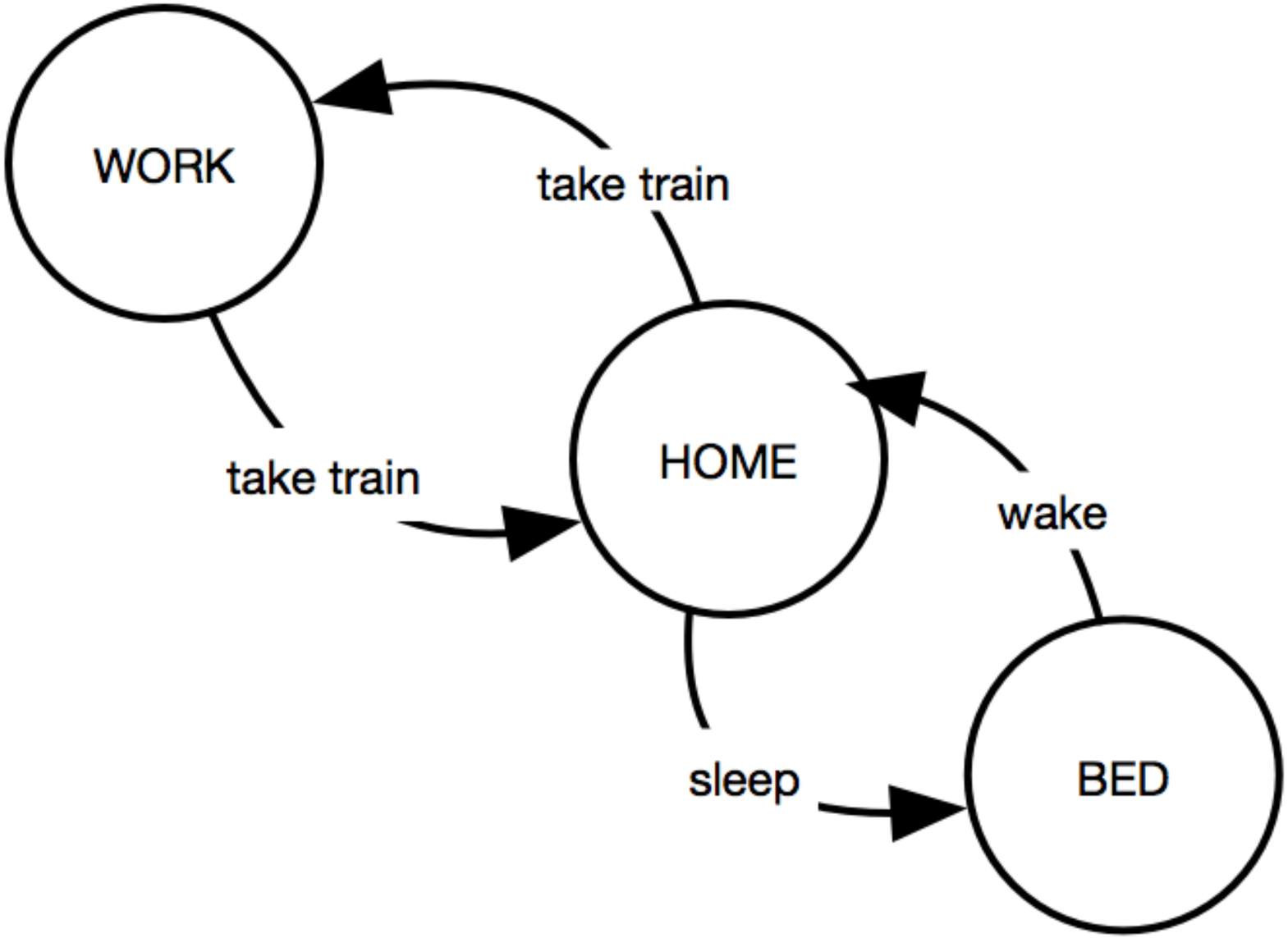
dy = mouseY – y



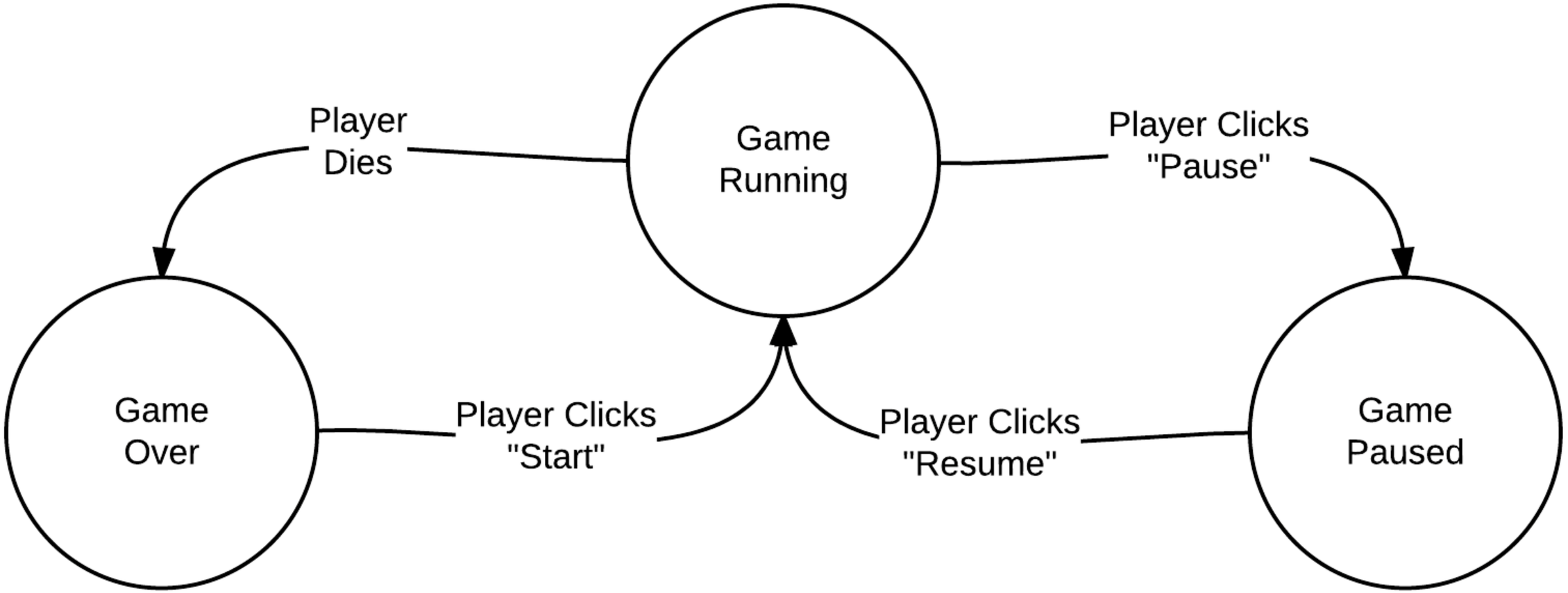
Richtungsvektor

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

void draw() {
  // unser hauptcode
}
```

Endlicher Zustandsautomat



Endlicher Zustandsautomat



Endlicher Zustandsautomat

```
final int WARTEN = 0;
final int BEARBEITEN = 1;
final int AUSGABE = 2;
int state = 0;

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

void draw() {
  stateMachine();
}
```

```
void stateMachine() {
  switch( state ) {
    case WARTEN:
      state = BEARBEITEN;
    break;

    case BEARBEITEN:
      state = AUSGABE;
    break;

    case AUSGABE:
      state = WARTEN;
    break;
  }
}
```

Endlicher Zustandsautomat

```
final int WARTEN = 0;
final int BEARBEITEN = 1;
final int AUSGABE = 2;
int state = 0;

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

void draw() {
  stateMachine();
}
```

```
void stateMachine() {
  switch( state ) {
    case WARTEN:
      // neuer code hier
      // oder funktion aufrufen
      state = BEARBEITEN;
    break;

    case BEARBEITEN:
      // neuer code hier
      state = AUSGABE;
    break;

    case AUSGABE:
      // neuer code hier
      state = WARTEN;
    break;
  }
}
```

Endlicher Zustandsautomat

https://github.com/ndsh/ws_cc_2019

Git Repository

Erweitert das Beschleunigungs-Maus-Beispiel:
– Implementiert eine simple State-Machine (+ "non-blocking") für die Agenten
– Erschafft mindestens zwei States, die sich auf Länge / Distanz des Vektors beziehen
– Lasst die Agenten Spuren "zeichnen" (auf PGraphics Objekt)
– Erschafft einen weiteren Agenten (*Coward*), der vor der Maus "flüchtet" sobald sie zu nahe kommt, sich aber sonst ruhig verhält.
*** Tauscht gerne geom. Formen aus**
*** Was ist mit nur orthogonalen Bewegungen? Probiert es aus!**
*** Einsatz von PVector in 3D (*Challenge?*)**

Aufgabe 4 – Bis 06.12.2019

Dokumentiert die Ausgabe + Code (alles als Zip)

Benutzt auch das Tutorium von Elias.

Beim nächsten Mal kurze (interaktive) Vorstellung!

https://processing.org/reference/

Aufgabe 4 – Bis 06.12.2019

Fragen?