MeJ Makers 26’ Spring School

Multi-agent crowd simulation - Learning week in Pertuis activity

Pertuis, 30 March - 1st April 2026

Yves Papegay - yap@informathiques.fr
Laure Vallet - laure@informathiques.fr

Funded by the European Union

This document is the course material for the activity 10 (Multi-agent crowd simulation - Learning week in Pertuis) in the framework of the Erasmus+ KA210 Small-scale partnerships in school education project ***MeJ Makers****. 2023-2-FR01-KA210-SCH-000176068*

The electronic version is an interactive document delivered under the Wolfram Notebook format. Read it or interact with it can be done with the help of the freely available Wolfram Player.

Introduction

Multi-agent crowd simulation sits at the intersection of artificial intelligence and computer graphics, aiming to model, analyze, and reproduce the collective dynamics emerging from large numbers of interacting individuals. Rather than treating a crowd as an homogeneous continuum, the multi-agent paradigm represents each individual as an autonomous entity endowed with perception, decision-making capabilities, and behavioral rules. This bottom-up perspective enables the emergence of complex global phenomena—such as lane formation, congestion, or panic-induced stampedes—from simple local interactions.

The study of crowd simulation draws on foundational concepts from complex systems and statistical physics, where macroscopic patterns arise from microscopic interactions, as well as from behavioral modeling to capture heterogeneity in human intentions and responses. Applications span a wide spectrum: urban planning, evacuation analysis, architectural design, virtual environment generation for films and video games, and increasingly, the coordination of robot swarms and human-robot coexistence scenarios.

This course introduces the theoretical foundations, algorithmic frameworks, and practical implementations of multi-agent crowd simulation. By the end of the course, participants will be equipped to both understand existing models and design their own simulation frameworks tailored to specific applications.

An example

A simulation of an attraction/repulsion phenomenon

Getting in...

Practical implementation will be done in the JavaScript language, with the help of the p5.js library and of its web development framework.

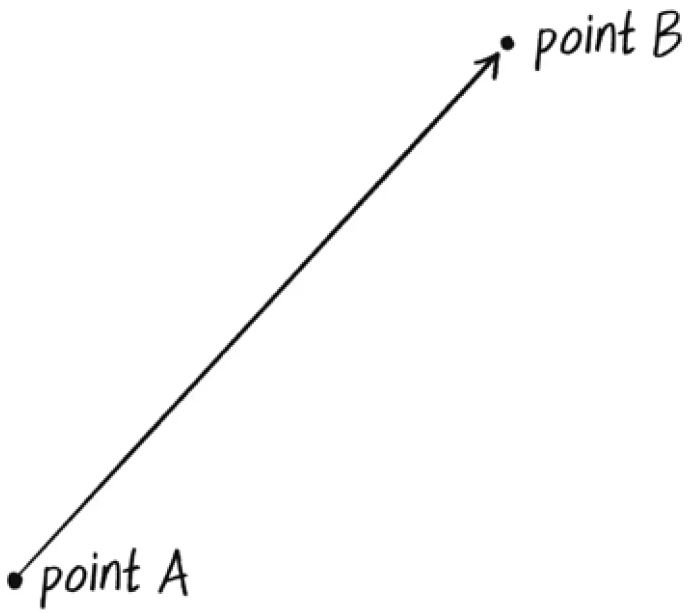
Credits

Code, drawings and examples in this course are extracted, inspired and/or adapted from the book “The Nature of Code” from Daniel Shiffman (https://natureofcode.com/).

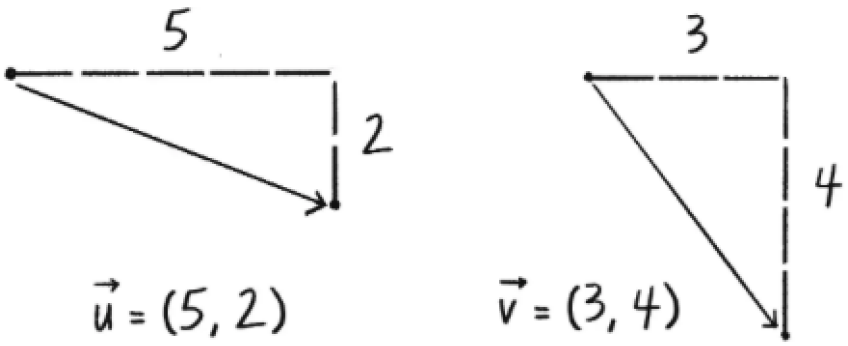
Bases of simulation : Vectors, Position, Velocity, Acceleration

---

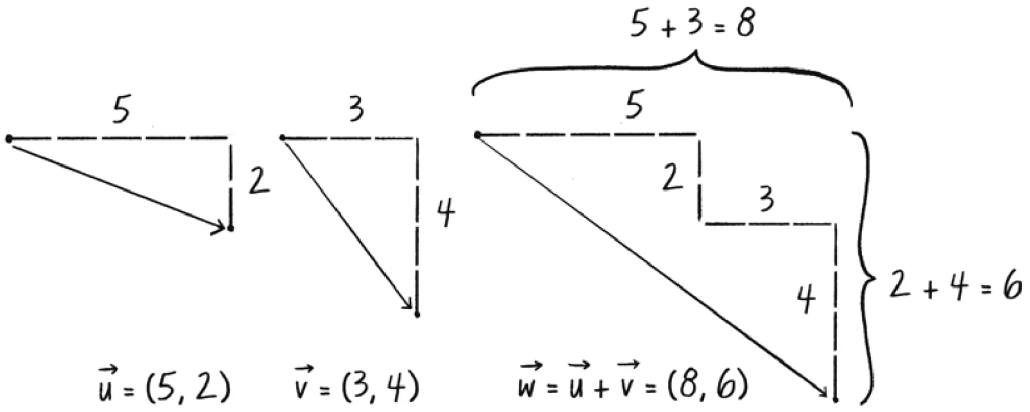
Vectors and Position



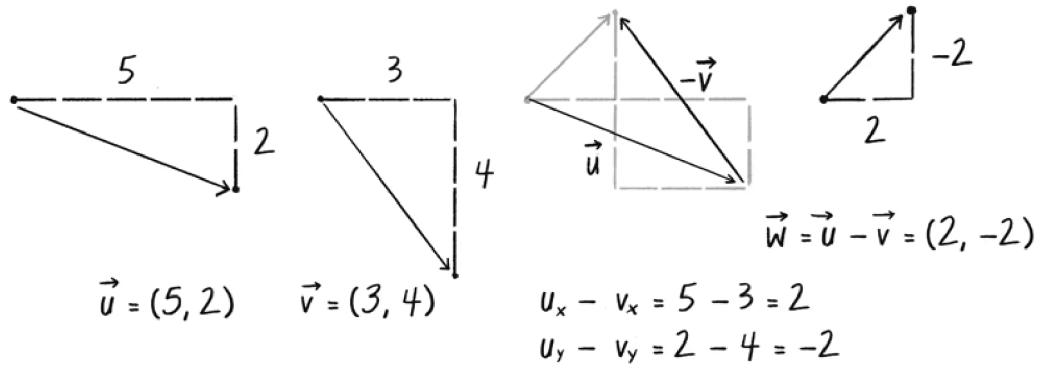
Components



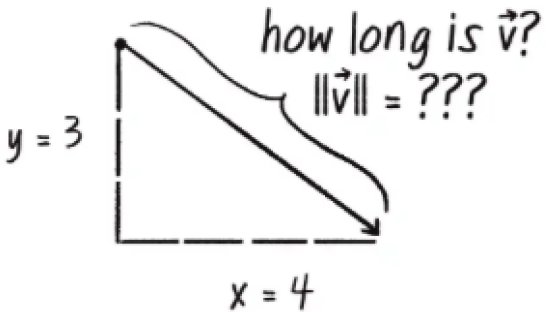
Addition



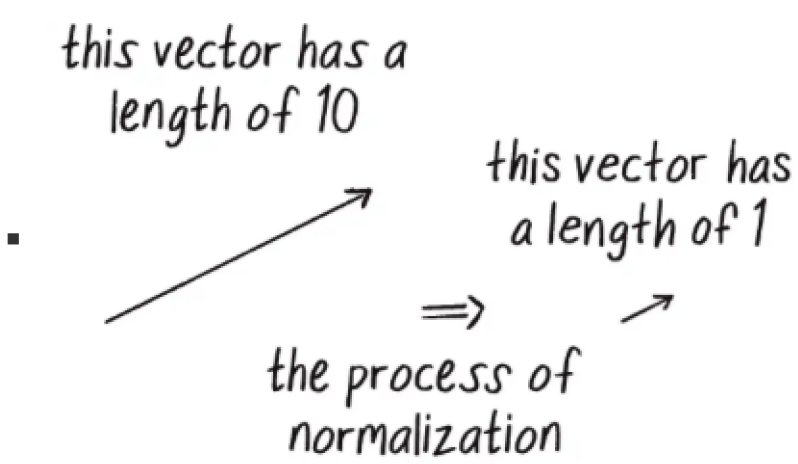
Subtraction



Magnitude



Normalization



Do It Yourself

Start using p5 web environment

P5 web environment is made of an editor allowing to develop and run JavaScript code inside a browser. The dashboard consists in an editor for HTML, CSS, an JavaScript files and a preview area where the HTML file is rendered when pressing the “Run” button.

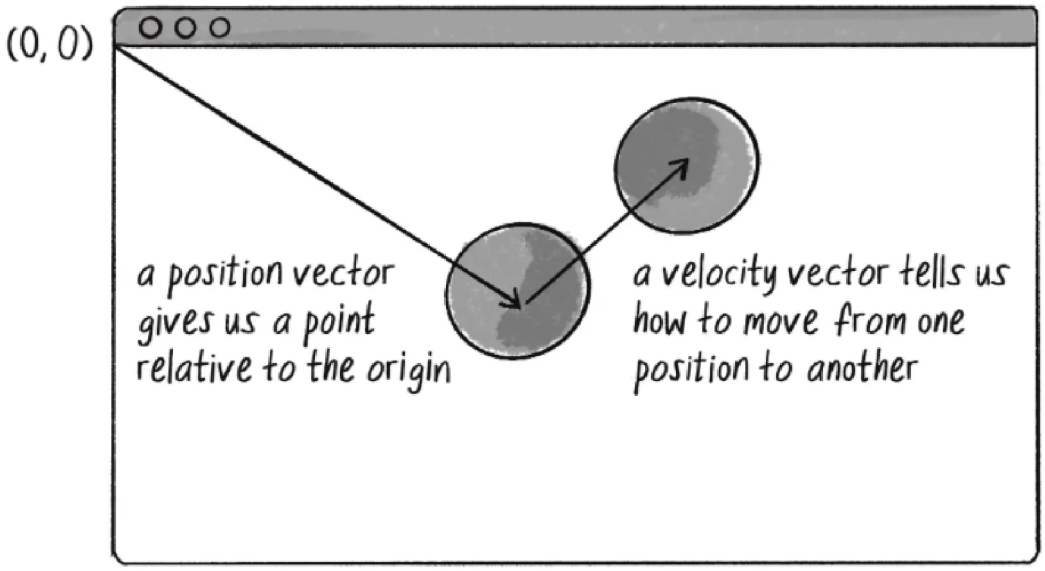
P5 web environment can be accessed at https://editor.p5js.org/. It is necessary to create an account and to log in to be able to save files.

Discover the structure of a sketch

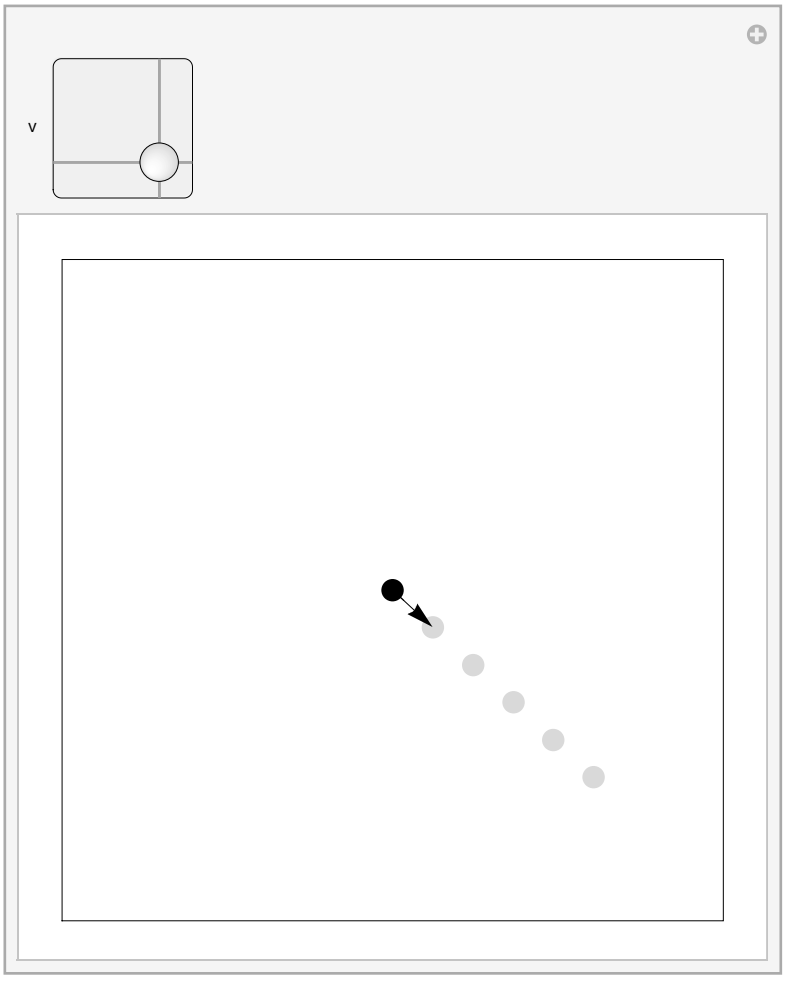
- Go to the collection of today’s course : https://editor.p5js.org/ypapegay/collections/cfchgd7GC
- Load the first sketch : “One Circle”

Velocity

Velocity is the vector difference between position at a given time and position at the next unit of time.



Out[ ]=



Do It Yourself

Implement motion

Based on the “One Circle” sketch, implement motion by defining a velocity and adding it to the position in each frame. (solution is sketch “One Circle Moving”)

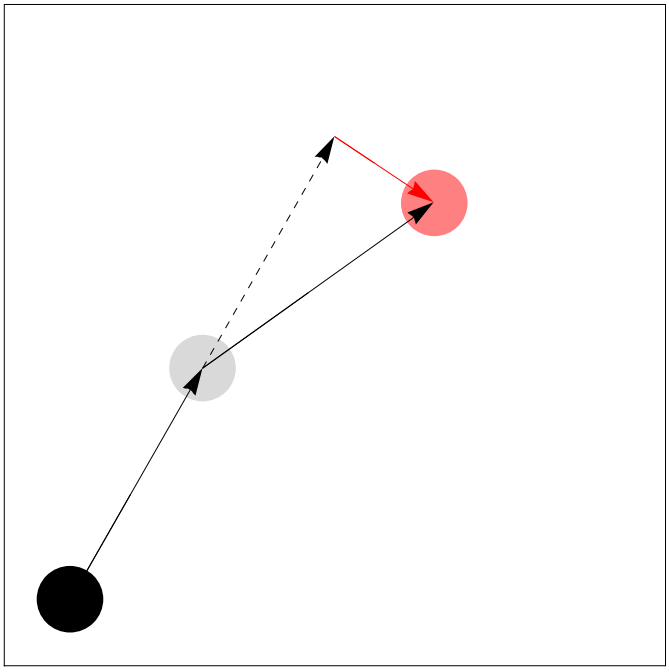
Force the circle to stay in the canvas

Based on the “One Circle Moving on a Torus” sketch, change the behavior of the circle when arriving to boundary to mimic a bouncing ball. (solution is sketch “One Bouncing Ball”)

Acceleration

Acceleration is the vector difference between velocity at a given time and velocity at the next unit of time.

Out[ ]=



Do It Yourself

Implement acceleration

Based on the “One Bouncing Ball” sketch, implement gravity by defining an acceleration and adding it to the velocity in each frame. (solution is sketch “One Bouncing Ball with Gravity”)

Implement a random direction acceleration

Based on the “One Bouncing Ball with Gravity” sketch, and starting in the center of the canvas, implement a random acceleration (solution is sketch “One Ball with Random Acceleration”)

Understand an acceleration depending on the mouse position

Play with the “One Ball accelerated by the Mouse” sketch, and understand by playing with the house the behavior of the acceleration in a motion.

Adding a bit of Physics to simulation

Reaching a target, Following a Path

Flocking : Separation, Alignment, Cohesion