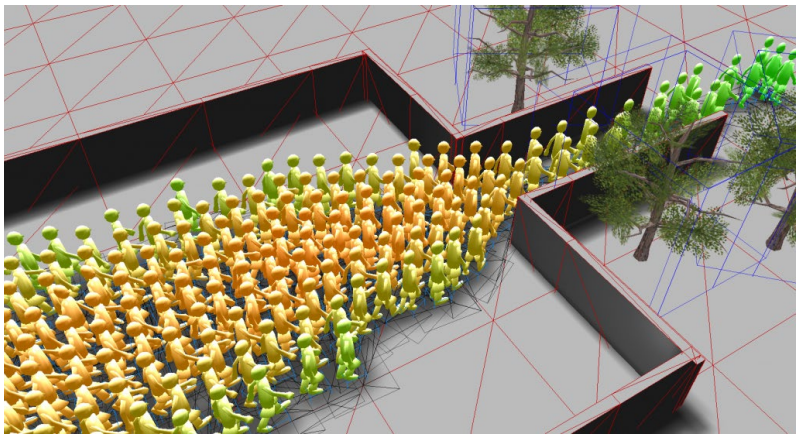
Simulating pedestrian crowds - 2021

Modeling and simulating the behavior of pedestrian crowds is an outstanding challenge in physics. It shares deep connections with statistical fluid mechanics, and sports primary societal relevance.

Examples of key questions in this context are: can we simulate the normal and evacuation behavior of crowds in a public facility? Which emergent behaviors characterize the system? Are there universal physical features in the motion of individuals?



Simulated crowd through a bottleneck (image from vadere.com).

A common modeling approach in mathematical-physics considers pedestrians as active interacting particles[1] (see Helbing and Molnar, *Social force model for pedestrian dynamics*, Phys. Rev. E, 1995).
As pedestrians move to reach a desired destination, they interact with one another and with the environment, e.g. to avoid collisions.

Let $x_i(t) = (X_i(t), Y_i(t))$ be the position[2] of pedestrian i at time $t > 0$ in a crowd of N individuals (i.e. $i = 1, \dots, N$). We model the motion with Newton-like Ordinary Differential Equations (ODE) as

$$\ddot{x_\iota} = F(x_i, \dot{x_\iota}) + \sum_{j \neq i, j=1}^{N} K(x_i, x_j) + E(x_i) \qquad (\blacksquare)$$

Positions are intended in the 2D plane, and the following are considered:

- F regulates propulsion[3] as $F = \frac{v_d(x_i) - \dot{x}_i}{\tau}$, where $v_d(x)$ is a "desired velocity field" and τ is a relaxation time.

[1] Active particles are characterized by the intrinsic ability to convert internal energy into motion.
[2] x_i indicates the 2-dimensional vector of components X_i, Y_i.
[3] Such a terms is also referred to as an active friction. Note that $v_d = 0$ yields a standard "passive" Stokes friction.

- K is a pairwise interaction kernel: $K(x_i, x_j) = A \exp\left(-\frac{\|x_i - x_j\|^2}{R^2}\right) \frac{x_i - x_j}{\|x_i - x_j\|} \theta(x_i - x_j, v_d)$, i.e. it is a decaying repulsion force from x_j to x_i, that is non-zero only when x_j is in the view cone of x_i ("social force"); R is a typical interaction radius (the view cone is given by $\theta(x_i - x_j, v_d)$ which is zero if the angle between v_d and $x_i - x_j$ is larger than e.g. 80 degrees and 1 otherwise).
- E takes into account the repulsion of objects and the "impermeability" of walls. $E = Bexp\left(-\frac{d}{R'}\right)\vec{n}$, where d is the distance between a pedestrian and a wall, $\vec{n}$ is the normal vector pointing towards a wall and R' is a distance scale.

In this project you are required to develop in a collaborative fashion a simple crowd simulator working in basic geometries, possibly containing objects.

Must have, in keywords:
Collaborative development via the *conversational development* paradigm
Unit testing & continuous integration
Docstring documentation
Expressive code, with proper naming choices for variables and functions
Implementation via numpy arrays
Plotting via matplotlib

Must have:

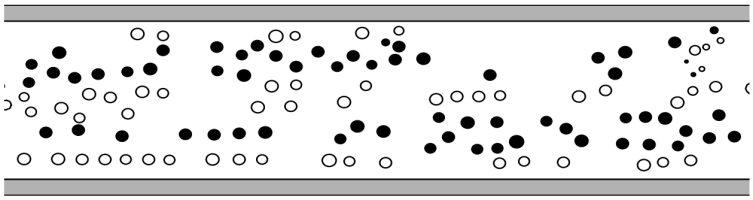
1. The simulation is carried out at discrete timesteps of size Δt: the n-th timestep is $t_n = n\Delta t$.
2. The state of the crowd is described by 4 numpy arrays[4] X, Y, U, V with length N
3. The integration of the ODE ■ is managed via the explicit Euler scheme:
   a. $X_{n+1} = X_n + \Delta t U_n$
   b. $Y_{n+1} = Y_n + \Delta t V_n$
   c. $U_{n+1} = U_n + \Delta t\, RHS_n^X$
   d. $V_{n+1} = V_n + \Delta t\, RHS_n^Y$

   The n subscripts indicate the state at time t_n. RHS_n^X indicates the X component of the right-hand side of the ODE ■.

---

1. [4] . $X[i]$ is therefore the X component of the position of the pedestrian i (and similarly Y). U, V are the X and Y components of the velocity. Remember that we need to integrate a 2nd order ODE. See, e.g. http://sites.science.oregonstate.edu/math/home/programs/undergrad/CalculusQuestStudyGuides/ode/second/so_num/so_num.html

4. Given two crowds walking in a corridor in opposite directions lane emerge spontaneously. Check it.

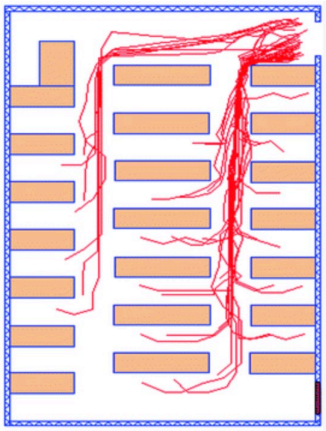


Spontaneous formation of lanes for two crowds walking in a corridor in opposite direction (solid dots vs. circles – image from Helbing 1995).

5. Basic documentation is provided in form of docstrings
6. CI performs simple unit testing

Nice to have:

7. Simulate the evacuation of a classroom.



Example of a classroom simulated evacuation
(image from Masria et al. InCIEC 2013 pp 423-434).

8. All simulation elements are objects: people, walls, desks…
9. The code is provided with a virtualenv that enables its execution (checked by CI)
10. Documentation is generated by Sphinx – and served through gitlab-pages.

Ultimate challenge:

11. The code is shipped with a docker container that can run it.