# How to run

This prototype was developed on the VSCode text editor for Windows 10 x64, using the MinGW x64 C++ compiler “g++”. The overall setup within Main features window creation followed by a purpose-built simulator implementation, built using only standard C++ libraries and these graphics libraries to display windows and simulation environments: OpenGL (3.3), GLSL, SDL2, GLM, and GLEW.

Testing was performed on a Ryzen 7 5800H CPU using an NVIDIA Geforce RTX 3060 graphics card, which was found to be sufficient to simulate in real-time for 400 particles at almost 50 epochs per second (1 second in the chosen default fine time step). Any lost simulation time when simulating 400 particles was gained back during the reduction in number throughout the simulated experiments before termination was reached.

To run this prototype the following files must be present alongside the executable within the same directory: SDL2.dll (SDL2), glew32.dll (GLEW), libgcc\_s\_dw2-1.dll (Windows), and libwinpthread-1.dll (Windows). Also, absence of an NVIDIA GPU means visualisation within the window’s OpenGL context is not guaranteed to display the entire environment. The prototype can be executed using default settings which adds analysis data to “output.txt”, a file that is generated if not present and otherwise appended to, or can be executed with an alternative output filename as an argument via console.

To compile this prototype, the following installations must be present within the local MinGW program as well as the “g++” command being assigned as a system path variable: glew-2.1.0 (GLEW), glm-0.9.9.7 (GLM), and SDL2-2.0.16 (SDL2), all found freely online. Executing the “compile.bat” batch file automatically links all required files to produce the prototype. The required code files should include the following: “contact.h”, “debug.cpp”, “debug.h”, “generators.h”, “input.h”, “main.cpp”, “main.h”, “obstacles.cpp”, “obstacles.h”, “particles.cpp”, “particles.h”, “path.h”, “simulation.cpp”, “simulation.h”, “sph.cpp”, “sph.h”, “window.cpp”, and “window.h”.

# Design

The following simulator is later implemented into a prototype to be used in proceeding experiments. The original simulator, UMANS, was developed as a platform to compare a variety of different crowd simulation techniques. This simulator features some main components of the UMANS while incorporating the SPH crowding behaviour of the later paper.

## Simulator

The simulation implements a numerical, forward-Euler update loop which moves pedestrians represented as particles about an environment. Within the environment, there are obstacles which can influence the movement of other particles as well as other neighbouring particles. The source of randomness within the simulation is represented exclusively by the seeded per-pedestrian-influence, which uses a random number stream to assign each pedestrian a linear pseudo-random value corresponding to their forcefulness amongst their neighbours.

The method of movement involves accumulating forces for each particle representing different forms of social and instinctive behaviours: SPH; Contact Forces; Navigation Policy. These techniques have been shown to provide reliable dense-crowd behaviour with improved, more intuitive parametric control of overall crowd properties including density. Each module produces an acceleration, , , and respectively which are used equations 1, 2 and 3 to calculate each particle’s next acceleration, velocity and position.

is the simulation time step. The coarse time step, , is used for updating the non-interactive navigation module less frequently within the simulation loop; the latest calculated acceleration term value is used for multiple subsequent simulation epochs. Time steps are measured in seconds and are simulated in real-time, provided the software and hardware can process each step under 1 second.

## Simulation Overview

The environment is a two-dimensional space measured in metres in which pedestrians can move around. Within the environment there are pedestrians and obstacles. Pedestrians are represented by the Particles class which holds arrays of each pedestrian attribute and handles movement by giving access of these attributes to other modules for implicit calculations, while obstacles are represented in the same manner within the Obstacles class which specifies boxes of static particles as well as empty boxes. These boxes comprise of 4 walls which are used to repel pedestrian particles away from the obstacle via the Contact module, whilst static particles within the box allow pedestrian particles to navigate around obstacles via the Hydrodynamics module.

### Data Structures

Tables 1, 2 and 3 show default values for each simulation property of particles, obstacles and obstacle boxes respectively, except for temporary values or non-scalar values, involving spawning regions or potentially non-trivial values like goal position, which heavily depend on the scenario. Most default values will be assigned during experimentation based on the original literature’s most stable or real-world-representative discovered values, to be disturbed during the experiments.

Particles are generated within a given area with equal spacing between each other. Each are assigned a randomised disc radius and relative mass using the previously mentioned linear, discrete random number stream, which represents their influence on neighbouring particles within the Contact and Hydrodynamics modules respectively. Density relates to the area-density calculated within the Hydrodynamics module using relative mass.

Obstacles are specified using boxes the walls enclose, where the static particles are generated inside as uniform columns and rows of touching particles using static values for and to represent a consistent boundary for the dynamic pedestrian particles.

All parameters for the simulation excluding scenario-specific parameters are summarised in table 4.

## Components

### SPH

The Hydrodynamics component is used to compute the density of particles then provide hydrodynamic forces to enhance the movement fluidity. Overall the force is found by calculating the particle’s density about all other particles representing people and obstacles, then calculating the pressure and viscosity force about the surrounding particles within the smoothing length, which is the radial influence of particles.

Equation 4 for acceleration is a particle-discretisation of a continuous field derived from the Navier Stokes partial differential equations of incompressible, viscous fluid motion and represents the overall output of this component, where is the density, is the pressure force, and is the viscosity force.

The SPH-approximation of these three terms is represented by equations 5 and 6. is a smoothing kernel which usually decreases as the distance between two particles is increased, and and are used in the gradient and Laplacian of . These equations are applied to density, pressure and viscosity to make equations 7, 8 and 9. Smoothing kernels of equations 10, 11 and 12 are used for the approximation of acceleration.

Each update these terms are calculated to determine the component acceleration for every simulated dynamic particle. Apparent from these equations and kernel functions density is a particle property, pressure enacts a force pushing away from a neighbouring particle, and viscosity enacts a force pushing against the particle’s current direction of motion.

Some assumptions are added to the fundamental model to ensure more acceptable crowd behaviours. is dynamically calculated for particles before calculating the pressure using a moving average of previous densities, to imitate the tendency of crowds to gradually accept higher densities using equation 13. Negative pressure is ignored to avoid drawing people to dense crowds and obstacles, done by altering the original pressure force term into equation 14. Any simulation-wide particles outside the radial influence of the smoothing length are completely ignored. Obstacle particles influence other particles but can only compute their density and pressure using equation 15.

### Contact Forces

The Contact component is used to compute forces between touching particles, for enforcing boundaries of particles and walls. To compute the nearest point between particles and walls the “nearestPointFromLine” function is borrowed from UMANS.

The overall contact force is computed as a sum of the forces between each particle and wall using equations 16, 17 and 18, where and and are the resulting particle-particle and particle-wall contact forces respectively. is a vector to from the nearest boundary point of the wall calculated using equations 19 and 20, where is the closest point on to , and are the outer edges of , and is an intermediate value denoting the relative distance between and of the point along which is closest to .

Additional operations are performed in order of priority, to prevent deeply-penetrating pedestrians from becoming trapped inside the box or multiple walls forcing the pedestrian away from a corner doubly: Directly-adjacent contacts are prioritised to ensure the closest wall provides the contact force for the pedestrian particle; A single wall is chosen for corner contacts to ensure only one contact force is applied to the pedestrian particle; For pedestrian particles with their centre inside a wall the force equation is altered, to reverse the contact’s direction and apply the correct force using equation 21, where the wall with the lowest contact distance is chosen, so that the contact of lowest penetration will force the particle out through the closest wall.

### Navigation Policy

The Pathing component is the final force to compute in the total acceleration equation. For simplicity the social forces collision avoidance isn’t implemented leaving only the goal force, the component responsible for veering the particles towards their goal locations.

The overall goal force is simply computed as a relaxed redirection towards the goal at a preferred speed using equation 22. All particles that reach the goal are deleted via a Sink, when the condition given in equation 23 holds true after the latest movement procedure. Finally particles may be introduced at the start of a coarse time step before the forces are calculated, by creating a new particle positioned randomly within some defined area called a Source

### Visualisation

Particles and obstacles are visualised for mid-development testing and observation. Particles are drawn as circles with a radius matching their contact range and obstacle boxes are drawn as grey blocks. Each particle is coloured dynamically based on SPH density using equations 24 and 25, which allows it to shift through blue, into green, into red, then finally into pink as the crowd becomes highly concentrated.

### Additional Components

The final components used in experimentation don’t contain complex calculations or affect the simulation so they are listed briefly here.

The input component sets all parameters and initialises the simulation before it runs.

The debug component periodically extracts various statistical information from the simulation useful for analysis and experimentation and is capable of varying a parameter mid-run. Statistics include updated data over a time frame including average and maximum values, and simulation data like run length and remaining contents, for model validation and experimentation.

# Implementation

## Class Structure

<full class structure and diagrams>

## Modules

<each module>

## Pseudocode

<pseudocode>

# Experimentation

## Parameters Used

<tables>

## Results

<graphs>

# Appendices

## Tables

### Model Design

1: Particles Data Structure

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Symbol | Default Value | Description |
| Simulation |  |  | Position |
|  |  | Velocity |
|  |  | Acceleration |
|  |  | Disc Radius |
|  |  | Mass |
| SPH |  |  | Density |
|  |  | Pressure |
|  |  | Average Density |
|  |  | Dynamic Rest Density |
|  |  | Acceleration Term |
| Contact Forces |  |  | Acceleration Term |
| Navigation Policy |  |  | Goal Position |
|  |  | Acceleration Term |

2: Obstacles Data Structure

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Symbol | Default Value | Description |
| Simulation |  |  | Position |
|  |  | Disc Radius |
|  |  | Mass |
| SPH |  |  | Density |
|  |  | Pressure |
|  |  | Average Density |
|  |  | Dynamic Rest Density |

3: Obstacle Box Data Structure

|  |  |  |
| --- | --- | --- |
| Symbol | Default Value | Description |
|  |  | Position |
|  |  | Size |

4: Model Parameters

|  |  |  |
| --- | --- | --- |
| Component | Symbol | Description |
| Simulation |  | Fine Time Step |
|  | Coarse Time Step |
|  | Maximum Particle Speed |
|  | Maximum Particle Acceleration |
| SPH |  | External Forces |
|  | Smoothing Length |
|  | Ideal Gas Constant |
|  | Viscosity |
|  | Minimum Rest Density |
|  | Maximum Rest Density |
|  | Dynamic Rest Density Time Window |
| Contact Forces |  | Neighbour Particle Contact Force Strength |
|  | Neighbour Wall Contact Force Strength |
| Navigation Policy |  | Goal Force Strength |
|  | Preferred Particle Journey Speed |
|  | Particle Speed Relaxation Time |
| The remaining components use ticks or epochs-until-next-update in place of simulation time. Their parameters aren’t shown here since they don’t directly affect the running of the simulation. | | |

### Simulator Design

5: Simulator Classes

### Experimentation

6: Scenario Parameters

7: Experimental Variables

8: Tracked Variables

9: Base Case Model Parameter Values

10: Base Case Scenario Parameter Values

11: Tracked Variable Expected Behaviours

12: Experiment 1

13: Experiment 2

14: Experiment 3

15: Experiment 4

## Equations

Equations given throughout the design provide the majority of functionality within the simulation, which are summarised here:

Simulation

1. Acceleration Update

2. Velocity Update

3. Position Update

SPH

4. Acceleration Term

5. Smoothed Approximation

6. Approximation Derivatives

7. Density Approximation Term

8. Pressure Approximation Term

9. Viscosity Approximation Term

10. Density Smoothing Kernel “Poly6”

11. Pressure Smoothing Kernel “spiky”

12. Viscosity Smoothing Kernel “Müller”

13. Dynamic Particle Density

14. Non-Negative Pressure Term

15. Obstacle Particle Density Term

Contact Forces

16. Acceleration Term

17. Particle Contact Force

18. Obstacle Wall Contact Force

19. Nearest Point On Wall

20. Relative Wall Point Position

21. Inter-Obstacle Contact Force

Navigation Policy

22. Acceleration Term

23. Sink Removal Condition

Visualisation

24. Particle Colour

25. Colour Density

## Graphs

### Base Case

### Experiment 1

### Experiment 2

### Experiment 3

### Experiment 4