Flood-pedestrian simulator

User guide

Contents:

Mohammad Shirvani

University of Sheffield

Last update: April 2021

1. Introduction

The flood-pedestrian simulator simulates crowd dynamics under immediate evacuation conditions with flowing floodwater in small urban areas. The simulator enables microscopic flood risk assessment on people at individual level, analysis of evacuation patterns of individuals, estimating the time for issuing emergency warnings, and finding potential safe destinations for immediate evacuation planning.

The simulator is capable of:

- incorporating each individual's realistic physical body characteristics, moving speeds and mobility states in and around the floodwater;
- factoring in autonomous decision making behaviour of people in going and following the others towards the safest destinations among multiple exit choices in outdoor spaces; and
- capturing the dynamic back interaction of people's crowding on the local floodwater dynamics.

Information about the approach and methodology for developing, evaluating and demonstrating the capabilities of the simulator is already documented in Shirvani et al. (2020), Shirvani et al. (2021) and Shirvani & Kesserwani (2021). This document provides a brief overview of the simulator's algorithmic structure (Sect. 2), aimed to offer step-by-step guidance for users to run the simulator on their own machine (Sect. 3) for two test cases reported in the afore-cited papers (see also the demo videos). Also, the outputs from the simulator are explained (Sect. 4), with guidance on how to modify the simulations by changing the environment parameters (Sect. 5) and on how to apply the simulator to new test cases using the available tools and models designed for this purpose (Sect. 6).

2. Brief overview of the flood-pedestrian simulator

The flood-pedestrian simulator is an agent-based model which dynamically couples a 'hydraulic model' to a 'pedestrian model' in one shared modelling framework, called FLAMEGPU. The FLAMEGPU framework allows simulation of multiple agents and their interactions on the Graphical Processing Units (GPUs) for parallel computations. In FLAMEGPU, CUDA simulation programs are generated automatically by processing three inputs as described in Fig. 1: a model file (*XMLModelFile.xm*l) defining agents' descriptive information (e.g. their type, numbers, etc.); a description of agent behaviour within a source code in C (*Functions.c*) for spatiotemporal update of the state of the agents responding to messages they receive from other agents; and, agents' input file (*input.xml*) for setting up their initial state.

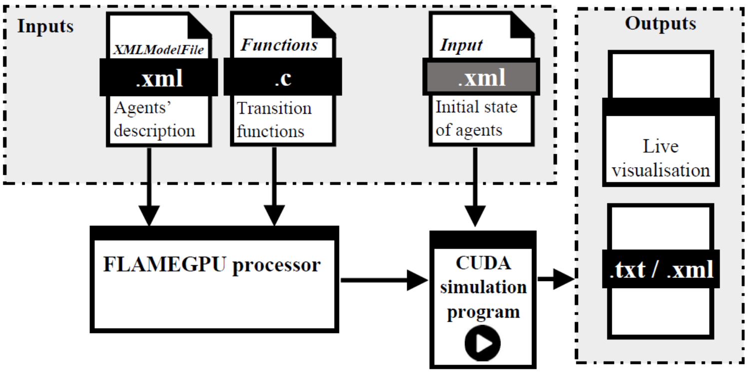


Figure 1. Illustration of the process of building and running an agent-based simulation program on FLAMEGPU via translating three user-devised input files (*XMLModelFile.xm*l, *Functions.c* and *input.xml*) into CUDA simulation program by the FLAMEGPU processor.

The pedestrian model was previously developed by Karmakharm et al. (2010) on FLAMEGPU. It is programmed based on the formulation of a social force model for people's dynamics including their movement patterns and their interaction with each other and their surrounding environment. The formulation of the social force model is embedded in *Functions.c* source code, allowing to update the state of two different agents that are specified in the *XMLModelFile.xm*l file: 'navigation agents' and 'pedestrian agents'. Navigation agents are of discrete type and they are fixed on a grid encoding the features of the environment layout into navigational vector fields for the movement of pedestrian agents. Pedestrian agents are of continuous type that could move continuously over the grid of navigation agents as they get updated in time and space. Detailed information about the pedestrian model can be found in Karmakharm et al. (2010). The hydrodynamic model is implemented on FLAMEGPU (Wang et al. 2011, Shirvani et al. 2021) on a fixed grid of discrete flood agents into the pedestrian model, to update all flood agents once at a time. The states of the flood agents are specified in *XMLModelFile.xm*l file where the initial navigation and pedestrian agents are also specified. The numerical formulations of the hydrodynamic model are embedded in the same *Functions.c* to simultaneously update the state of flood agents and pedestrian agents in space and time.

The *XMLModelFile.xm*l and *Functions.c* are both located in the '…\src\model' folder, alongside '…\src\dynamic' and '…\src\visualisation' folders. The contents of these folders are automatically generated and rewritten after each build of the simulator on FLAMEGPU according to *XMLModelFile.xm*l and *Functions.c*. The .xml input files are located in '..\iterations' folder and they contain the initial state of the flood and navigation agents and environment parameters for setting up the Shopping centre (ShopCent.xml) and Hillsborough stadium (HilStad.xml) test cases.

The following instructions will get you a copy of the source code of the simulator for running it on your local machine. By cloning/downloading the flood-pedestrian simulator repository contents on Github (accessible at: https://github.com/SahebSh/flood-pedestrian-simulator), the user will have access to the flood-pedestrian source code and agents description file (*XMLModelFile.xm*l and *Functions.c*) to build and run the Shopping centre and Hillsborough stadium test cases using the provided ShopCent.xml and HilStad.xml input files.

3. Step-by-step guide to run the flood-pedestrian simulator on windows

Step 1. Download and unzip/extract the flood-pedestrian-simulator.zip folder in the Github repository (accessible at: https://github.com/SahebSh/flood-pedestrian-simulator).

Note:

Before clone/download, consider the following things that need to be downloaded/installed on your machine.

Required software:

- FLAMEGPU v1.5: download FLAME-GPU-SDK.zip folder from the FLAMEGPU master repository (https://github.com/FLAMEGPU/FLAMEGPU) or, alternatively, directly from this link. Also, more information about FLAMEGPU and the latest Technical Report and User Guide could be found at http://www.flamegpu.com/.
- MS Visual Studio 2015 or earlier: you can download the latest version of the Visual Studio from Microsoft website available at https://visualstudio.microsoft.com/downloads/. You need to install Visual C++

components and .NET Framework toolkit during the installation of MS Visual Studio.

- CUDA Toolkit 10.1: You can download CUDA from the Nvidia developer download archive - alternatively you may use later versions, but it needs manual modifications to the solution file (that is explained in Step 3 below).

Required hardware:

- Nvidia Graphics card - the simulator should be able to run on any Nvidia Graphics Card with a minimum 2GB memory installed on a normal machine.

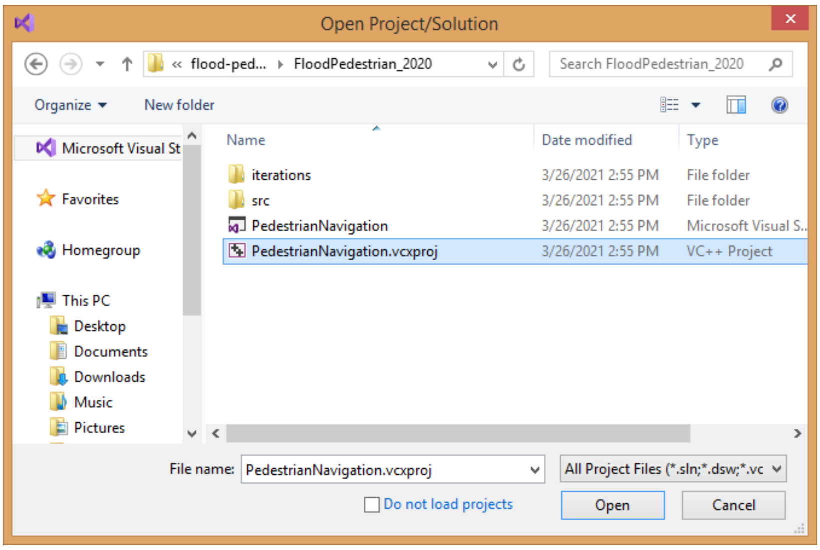
Step 2. Copy the *FloodPedestrian_2020* folder from the drive directory you selected when downloading the simulator and paste it to '..\FLAME-GPU-SDK\examples' folder along with the other examples that are made available by FLAMEGPU developers for practicing and learning purposes.

Step 3. If you are using CUDA 10.1, then skip this step, otherwise, go to '..\flood-pedestrian-simulator-master\FloodPedestrian_2020' folder and right-click on PedestrianNavigation VC++ Project file and open it with any text and source code editor (e.g. Notepad). Then press Ctrl+f and look for CUDA 10.1 and replace it with the version of the CUDA Toolkit that you have installed on your system (e.g. replace CUDA 10.1 with CUDA 10.2). Then, save the changes.

Note:

The executable files required to run the flood-pedestrian simulator are not provided in the source repository; therefore the user is expected to build them locally on their own machine prior to any run attempt. The following steps will guide you through how to do it.

Step 4. Open Visual Studio, then click on '*Project/Solution..*' located in '*File*' > '*Open*' in the top menu and navigate to '..\FLAMEGPU\examples\FloodPedestrian_2020' folder and open 'PedestrianNavigation' VC++ Project file as shown in the screenshot below.



Name	Date modified	Type
iterations	3/26/2021 2:55 PM	File folder
src	3/26/2021 2:55 PM	File folder
PedestrianNavigation	3/26/2021 2:55 PM	Microsoft Visual S...
PedestrianNavigation.vcxproj	3/26/2021 2:55 PM	VC++ Project

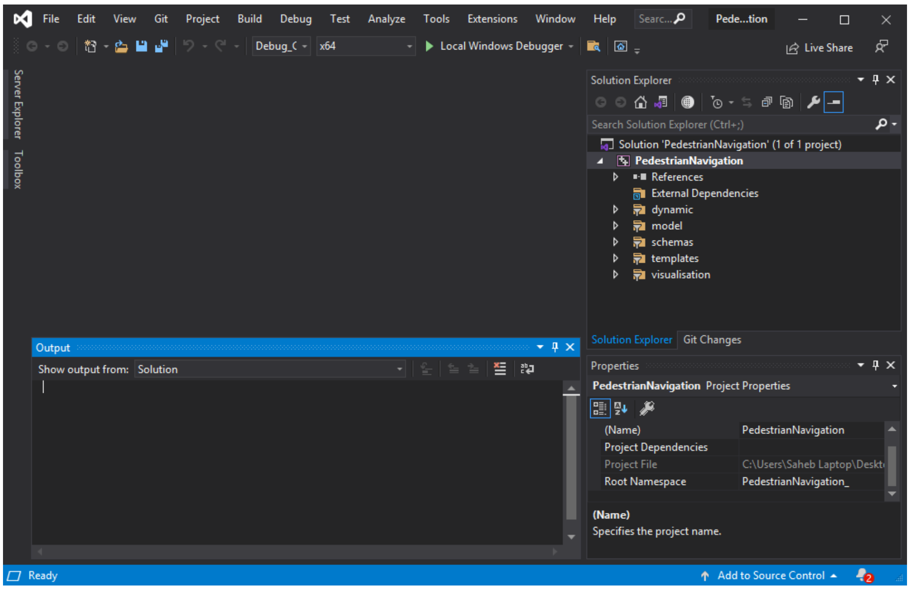
File name: PedestrianNavigation.vcxproj

All Project Files (*.sln;*.dsw;*.vc

Figure 2. Screenshot of the 'Open Project/Solution' window navigated to the location of the 'PedestrianNavigation' VC++ Project file to be opened in the Visual Studio (Step 4).

Note:

ensure that all the contents of the simulator are opened without any error/warning. Once the project is opened correctly, the user should see the contents of the simulator within the Solution Explorer on the right side of Visual Studio window as shown below in Figure 3.



(Name) PedestrianNavigation
Project Dependencies
Project File C:\Users\Saheb Laptop\Deskt
Root Namespace PedestrianNavigation_
(Name)
Specifies the project name.

Figure 3. Screenshot of the Visual Studio window showing the contents of the flood-pedestrian simulator inside the 'Solution Explorer' tab on the right side.

Step 5. Select 'Release_Visualisation' from the building configuration mode located under the top menu of the Visual Studio user interface like below:

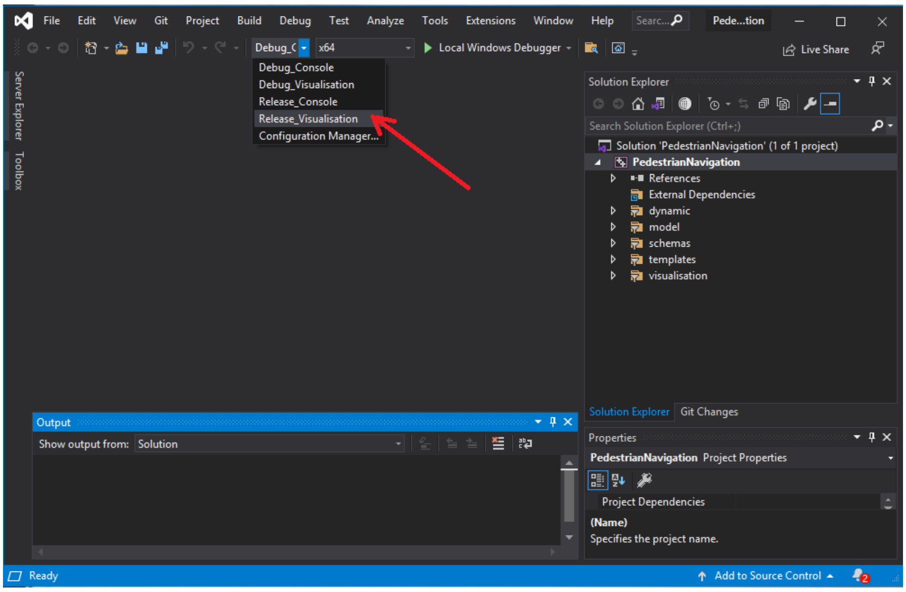


Figure 4. Screenshot of the Visual Studio window showing where to select 'Release_Visualisation' from the building configuration mode.

Step 6. Right-click on PedestrianNavigation in the ‘Solution Explorer’ on the right side of the Visual Studio and click on ‘Properties’.

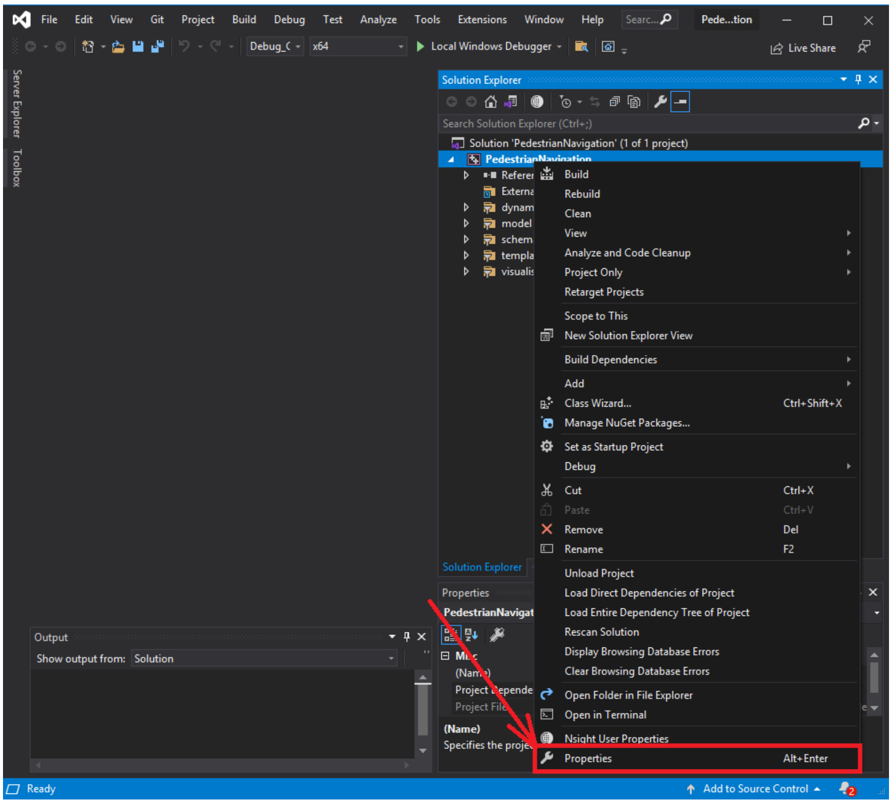
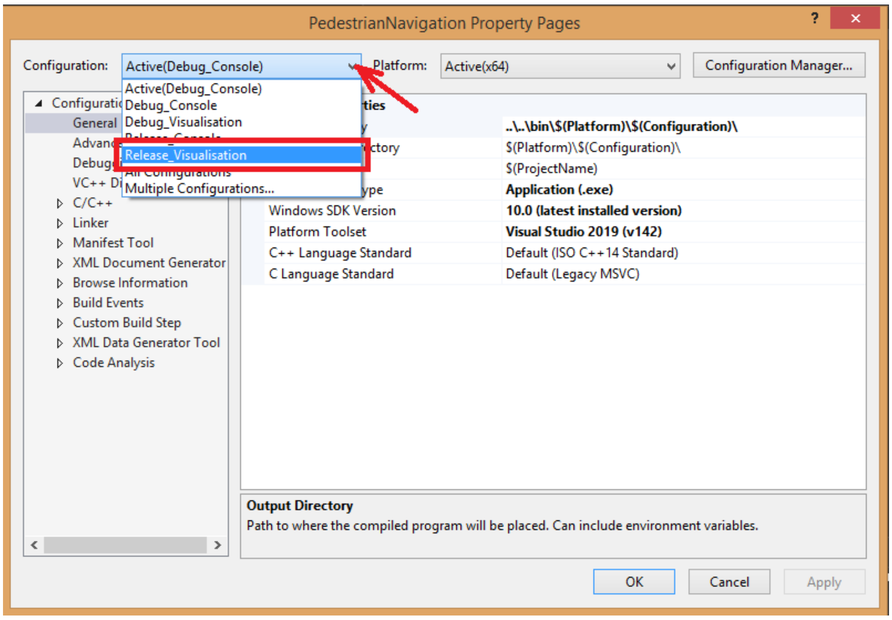


Figure 5. Screenshot of the Visual Studio window showing where to find ‘Properties’ after right-clicking on PedestrianNavigation in the ‘Solution Explorer’ on the right side.

Step 7. On top of the ‘PedestrianNavigation Property Pages’ window choose ‘Release_Visualisation’ configuration mode:

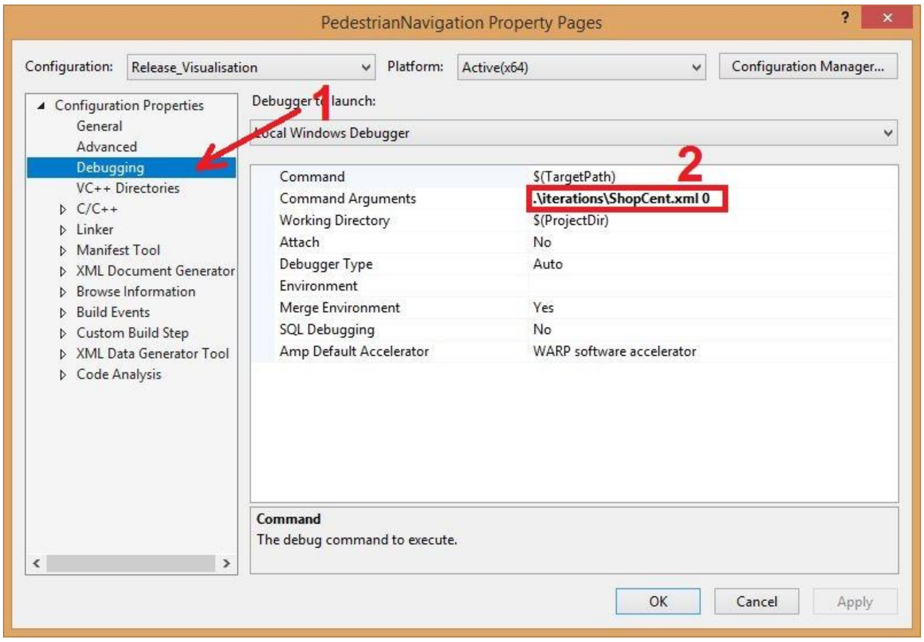


..\..\bin\$(Platform)\$(Configuration)\
ctory $(Platform)\$(Configuration)\
$(ProjectName)
ype Application (.exe)
Windows SDK Version: 10.0 (latest installed version)
Platform Toolset: Visual Studio 2019 (v142)
C++ Language Standard: Default (ISO C++14 Standard)
C Language Standard: Default (Legacy MSVC)

Output Directory
Path to where the compiled program will be placed. Can include environment variables.

Figure 6. Screenshot of the ‘PedestrianNavigation Property Pages’ window showing where to choose ‘Release_Visualisation’ from configuration mode options.

Step 8. First, click on ‘Debugging’ under the ‘Configuration Properties’ tab on the left side of the ‘PedestrianNavigation Property Pages’ window, then within ‘Command Arguments’ type the directory address of the input file. For example to run the simulator for the shopping centre test case, type ‘.\iterations\ShopCent.xml’ followed by the device number (usually is 0) as shown in the screenshot below, then click on ‘OK’ to exit the ‘PedestrianNavigation Property Pages’ window.



Configuration: Release_Visualisation
Platform: Active(x64)
Debugger to launch: Local Windows Debugger

Command	$(TargetPath)
Command Arguments	.\iterations\ShopCent.xml 0
Working Directory	$(ProjectDir)
Attach	No
Debugger Type	Auto
Environment	
Merge Environment	Yes
SQL Debugging	No
Amp Default Accelerator	WARP software accelerator

Command
The debug command to execute.

Figure 7. Screenshot of the ‘PedestrianNavigation Property Pages’ window showing where to: (1) click on ‘Debugging’ under the ‘Configuration Properties’ and (2) type the directory address of the input file.

Step 9. Right-click on PedestrianNavigation in the ‘Solution Explorer’ again and click on ‘Build’ like below in the screenshot below (Fig. 8). The FLAMEGPU then generates the CUDA simulation program code that is executable on your machine. This may take a couple of minutes and if there is no error generated, the user could press Ctrl+F5 to run the simulation under visualisation mode.

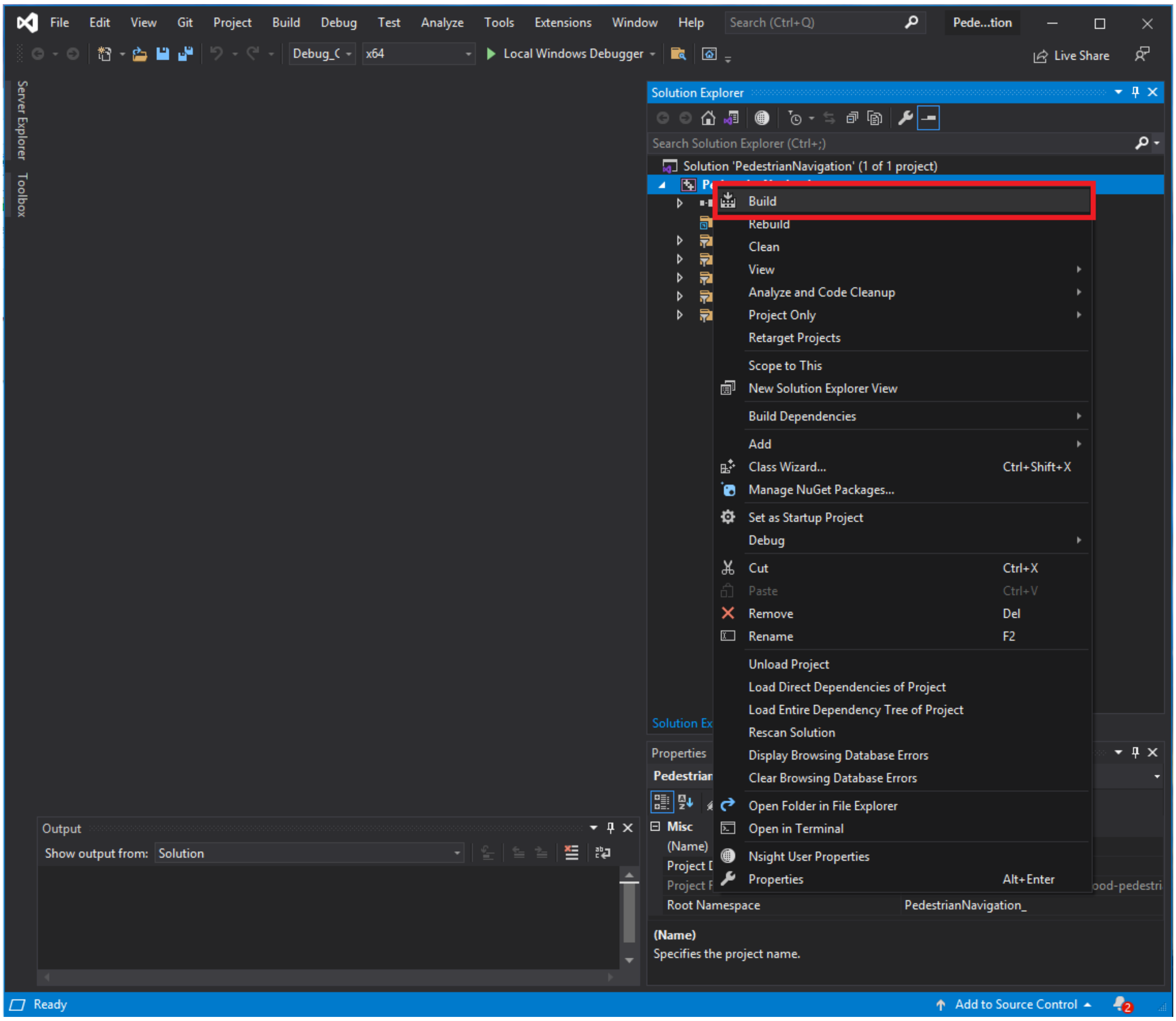


Figure 8. Screenshot of the Visual Studio window showing where to select 'Build' after right-clicking on PedestrianNavigation in the 'Solution Explorer' on the right side.

4. Simulation outputs

Apart from the FLAMEGPU's built-in visualisation that is immediately popped up on your screen after running the simulations, the simulator is programmed to output dynamic information of flood agents and pedestrian agents at each iteration of the simulation. These outputs are generated in three ways:

1) **The console command window** provides live information about the simulation time, the number of pedestrians at different flood risk states and floodwater flow. The information represented in the console command window is self explanatory and therefore not explained here. However, all the information shown on the console window is also copied into output files explained below.

2.a) **'output.txt'** is a text file that stores the dynamic information of pedestrians and floodwater at each simulation iteration. This file is automatically generated at the first iteration of simulation in the '../FloodPedestrian_2020/iterations' folder and it gets updated dynamically during the later iterations. This information is stored within 21 columns over rows corresponding to the simulation iteration. From left to right, table 1 outlines the description of the information as numbers in each column that the user can find at the end of the simulation after opening the 'output.txt' file.

Table 1. Description of the pedestrians and floodwater information stored dynamically in ‘output.txt’ file.

Col.	**Description**
1	Simulation time (s)
2	Total number of pedestrian agents generated during the simulation time
3	Remaining number of pedestrian agents yet to be generated
4	Total number of pedestrian agents moving in the area
5	Total number of pedestrian agents at no flood risk (in dry zones, HR = 0)
6	Total number of pedestrian agents at low flood risk (HR < 0.75)
7	Total number of pedestrian agents at medium flood risk (0.75 < HR < 1.5)
8	Total number of pedestrian agents at high flood risk (1.5 < HR < 2.5)
9	Total number of pedestrian agents at highest flood risk (HR > 2.5)
10	Maximum HR over the entire domain at the particular simulation time
11	Maximum depth of floodwater over the entire domain (m)
12	Maximum velocity of floodwater over the entire domain (m/s)
13	Maximum reached HR during the entire simulation
14	Maximum number of pedestrian agents that were at the highest flood risk
15	Maximum number of pedestrian agents that were at the low flood risk
16	Maximum number of pedestrian agents that were at the medium flood risk
17	Maximum number of pedestrian agents that were at the high flood risk
18	Total number of pedestrian agents at risk of sliding-only condition
19	Total number of pedestrian agents at risk of toppling-only condition
20	Total number of pedestrian agents at risk of toppling-and-sliding condition
21	Total number of pedestrian agents at risk of instability (regardless of condition)

2.b) ‘output_exits.txt’ is also a text file that stores the record of the number of pedestrian agents going towards a particular destination at each iteration of the simulation. The exits are predetermined by the user specifying from where the pedestrians enter and/or exit the domain. This output is useful only when the emergency exit of pedestrians is not predefined (like the Hillsborough Stadium case study) where the ‘autonomous change of direction’ condition is enabled to allow pedestrian agents to autonomously navigate into new pathways while moving within a flooded zone. The **output_exits.txt** file has 11 columns and rows corresponding to the simulation iteration. From left to right, the first column shows the simulation time and the second to the eleventh column show the total number of pedestrians going towards Exit1 to Exit10 respectively. The simulator is configured to have 10 predefined exits. This will be explained in more detail in (Sect 5).

3) ‘(*simulation_time)*flood.csv’ and **‘(*simulation_time)*ped.csv’** are also text files generated at each simulation iteration in ‘../FloodPedestrian_2020/iterations’ folder. (*simulation_time*) denotes the simulation time (in seconds) at each iteration once it is generated. ‘(*simulation time*)flood.csv’ provides information about flood agents over their grid stored as a matrix with multiple rows relevant to their numbers and 8 columns as described in Table 2. ‘(*simulation time*)ped.csv’ also follows a similar format but with 12 columns containing the information of pedestrians as outlined in Table 3. Note that the user can generate these files in regular time intervals (in second) via assigning non-zero value to *outputting_time_interval* parameter located in .xml input file in ‘../FloodPedestrian_2020/iterations’ folder in advance (this will be explained in Sect. 5).

Table 2. Description of the information of one flood agent (one row) stored in '(*simulation_time*)flood.csv' file.

Description	**Col.**
The coordinate of flood agent in x-axis direction as a positive integer	1
The coordinate of flood agent in y-axis direction as a positive integer	2
The depth of floodwater (m) at the location of the flood agent	3
The velocity magnitude of floodwater (m/s) at the location of the flood agent	4
HR of floodwater at the location of the flood agent	5
Average velocity of floodwater along x-axis direction (m/s)	6
Average velocity of floodwater along y-axis direction (m/s)	7
Topography height (m) at the location of the flood agent	8

Table 3. Description of the Information of one pedestrian agent stored in ‘(*simulation_time*)ped.csv’ file. Each row represents the information of one pedestrian agent at a particular simulation time.

Description	**Col.**
The coordinate of pedestrian agent in x-axis direction as a floating-point number	1
The coordinate of pedestrian agent in y-axis direction as a floating-point number	2
The flood risk state of pedestrian agent based on HR of floodwater at their location: **0:** pedestrian agent is at no flood risk (in dry zones, HR = 0) **1:** pedestrian agent is at low flood risk (HR < 0.75) **2:** pedestrian agent is at medium flood risk (0.75 < HR < 1.5) **3:** pedestrian agent is at high flood risk (1.5 < HR < 2.5) **4:** pedestrian agent is at highest flood risk (HR > 2.5)	3
HR of floodwater at the location of the pedestrian agent	4
The depth of floodwater at the location of the pedestrian agent (m)	5
The velocity magnitude of floodwater at the location of the pedestrian agent (m/s)	6
The moving speed of pedestrian agent at the particular simulation time (m/s)	7
The body height of the pedestrian agent (m)	8
The body mass of the pedestrian agent (kg)	9
The gender of the pedestrian agent (**1**: female , **2**: male)	10
The age of the pedestrian agent (in years)	11
The destination of the pedestrian agent at that particular simulation time. It can vary between 1 and 10 that corresponds to the number of the predetermined exits.	12

Note:

- A copy of the output files from the simulations reported in Shirvani & Kesserwani (2021) is uploaded into Zenodo directory (accessible at: https://doi.org/10.5281/zenodo.4576906), where the user could see how the outputs from one run of the flood-pedestrian simulator look like.
- A live video of the simulations reported in Shirvani & Kesserwani (2021) that was captured from FLAMEGPU built-in visualisation window is also uploaded into the TIB AB Portal accessible at https://doi.org/10.5446/51547.

5. Modifying the simulations

A set of parameters is initialised in the <environment> element of .xml input file (located in the ‘../FloodPedestrian_2020/iterations’ folder) to enable users implement different actions by only changing the initial values assigned to these parameters. These parameters are set as environment constants accessible to all the agents. Table 4 guides the users through what actions can be performed and how to implement them in the simulator by assigning a set of possible initial values.

Note:

- Changing the values of the environment parameters in the .xml file does not require further compilation and building of the executable program. This enables the users to practice and study other scenarios and initial conditions immediately after assigning new values to these parameters.
- Ensure that the parameters' name in the .xml file is not changed while assigning values to them as this will disable the functionality of that particular parameter.
- Always have a copy of the .xml input file before any modification. Doing this will help the user to have the original structure and naming for later restoration of the data.

Table 4. Description of the actions and related parameters including their format, unit and possible values that could be assigned to them.

Action	Parameter	Format	Unit	Possible values
To set a time limit for outputting the simulation results. **Note:** by default, the results are automatically generated in each simulation time step until the termination of the simulation by the user (e.g. by closing the console window).	*outputting_time*	float	second	Any positive value. To disable the option assign 0.
To set a different time interval for outputting the results.	*outputting_time_interval*	float	second	Any positive value. To disable the option assign 0.
To change the 2D spatial dimensions for the study area.	*xmin; xmax; ymin; ymax*	float	metre	Any positive value. **Note:** *xmin* and *ymin* are both initially given zero; and *xmax* and *ymax* represent the length of the area in *x*- and *y*-axis directions respectively.
To change the time step of the pedestrian model. **Note:** this will be dominated by the *dt_flood* once the flood starts (see below).	*dt_ped*	float	second	Between 0.1 to 2.0 depending on the user's preference in preserving the realistic motion of pedestrians in real time.
To change the time step of the hydrodynamic model. **Note:** Any given initial value to this parameter will be updated automatically when '*auto_dt_on*' is enabled (see below).	*dt_flood*	float	second	Any positive value. **Note:** any given value should preserve the stability of the numerical solution depending on the initial

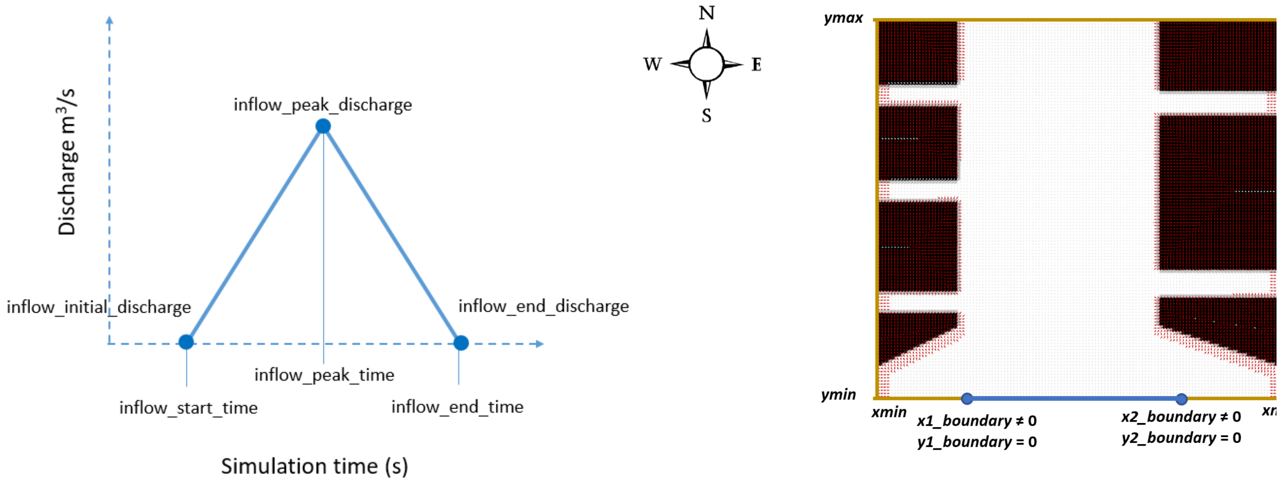
				condition of the flood inflow and geometry of the area (try varying it between 0.01 to 0.1).
To enable adaptive time stepping to maximise allowable *dt_flood* while keeping stability of the hydrodynamic solution. **Note:** when outputting at regular temporal intervals is considered, constant time stepping is usually preferable.	*auto_dt_on*	integer	none	1: Enable 0: Disable
To enable early evacuation of pedestrians prior to the start of flooding at a specific time (see further below).	*evacuation_on*	integer	none	1: Enable 0: Disable
To enable crowding of a certain number of pedestrians over the area before the start of flooding. **Note:** this option allows the user to study scenarios where a populated area is hit by flooding, like the shopping centre test case.	*preoccupying_on*	integer	none	1: Enable 0: Disable **Note:** the default scenario for the Hillsborough stadium case study does not require activation of this option, unless the user decides to study another scenario.
To select the shape of the inflow hydrograph. **Note:** the shape of the hydrograph is dependent on the parameters related to the time and discharge that are explained further below.	*poly_hydrograph_on*	integer	none	1: Polynomial 0: Triangular
To prevent pedestrians from entering the area after the flooding is started.	*stop_emission_on*	integer	none	1: Enable 0: Disable **Note:** by default it is enabled for the shopping center test case; but disabled for the Hillsborough stadium case study.
To enable the pedestrians to go to the user-defined emergency exit (explained further below) once the flooding is started.	*goto_emergency_exit_on*	integer	none	1: Enable 0: Disable **Note:** by default it is enabled for the shopping center test case as the emergency exit is made known to all the pedestrians

				before the start of flooding (via assigning a value to *emergency_exit_number* parameter); but it is disabled for the Hillsborough stadium case study as the pedestrians are allowed to choose any destination upon enabling *escape_route_finder_on* option (see below).
To enable the 'autonomous change of direction' condition that allows pedestrians to autonomously navigate towards a destination while moving within a flooded zone. **Note:** more information about 'autonomous change of direction' condition is provided in Shirvani & Kesserwani (2021).	*escape_route_finder_on*	integer	none	1: Enable 0: Disable
To prevent pedestrians going back towards the exit/entrance through which they initially entered the area.	*no_return_on*	integer	none	1: Enable 0: Disable **Note:** the default scenario for the shopping centre and Hillsborough stadium test cases requires enabling this option, unless the user decides to study another scenario.
To enable pedestrians to go towards the most popular destination selected by most. **Note:** this option can only be enabled in scenarios where the 'autonomous change of direction' condition is enabled (see above).	*follow_popular_exit_on*	integer	none	1: Enable 0: Disable
To enable realistic moving speed of pedestrians in floodwater based on water depth and velocity. **Note:** more information about the realistic moving speed of pedestrians is provided in Shirvani & Kesserwani (2021).	*walking_speed_reduction_in_water_on*	integer	none	1: Enable 0: Disable **Note:** if this option is disabled, the pedestrians will maintain their in-dry walking speed that is randomly

				assigned to them once they are generated.
To immobilise pedestrians once they lose their stability in floodwater.	*freeze_while_instable_on*	integer	none	1: Enable 0: Disable
To enable the 'maximum excitement' condition that allows pedestrians to increase their walking speed under evacuation conditions. **Note:** more information about the 'maximum excitement' condition is provided in Shirvani & Kesserwani (2021).	*excitement_on*	integer	none	1: Enable 0: Disable
To enable the 'two-way interaction' condition for factoring in the effect of pedestrians' crowding on the bed roughness. **Note:** more information about the 'two-way interaction' condition is provided in Shirvani et al. (2021) and Shirvani & Kesserwani (2021).	*ped_roughness_effect_on*	integer	none	1: Enable 0: Disable
To switch between 'walking' and 'running' condition defining pedestrians moving speeds inside the floodwater. **Note:** this option is effective only when the realistic moving speed of pedestrians in floodwater is enabled (see above). More information about the 'walking' and 'running' conditions is provided in Shirvani & Kesserwani (2021).	*walk_run_switch*	integer	none	1: 'walking' condition 2: 'running' condition
To define the number of times that pedestrians can dynamically change their direction towards a new destination after which they follow the others towards the most popular destination. **Note:** this option is only enabled in scenarios where the 'autonomous change of direction' condition is also enabled (see above).	*dir_times*	integer	none	Any value. **Note:** assigning a value to this parameter is dependent on the time step of the simulation and the extent of flooding.
To change the threshold of floodwater depth to body height of	*wdepth_perc_thresh*	float	none	Between 0.0 and 1.

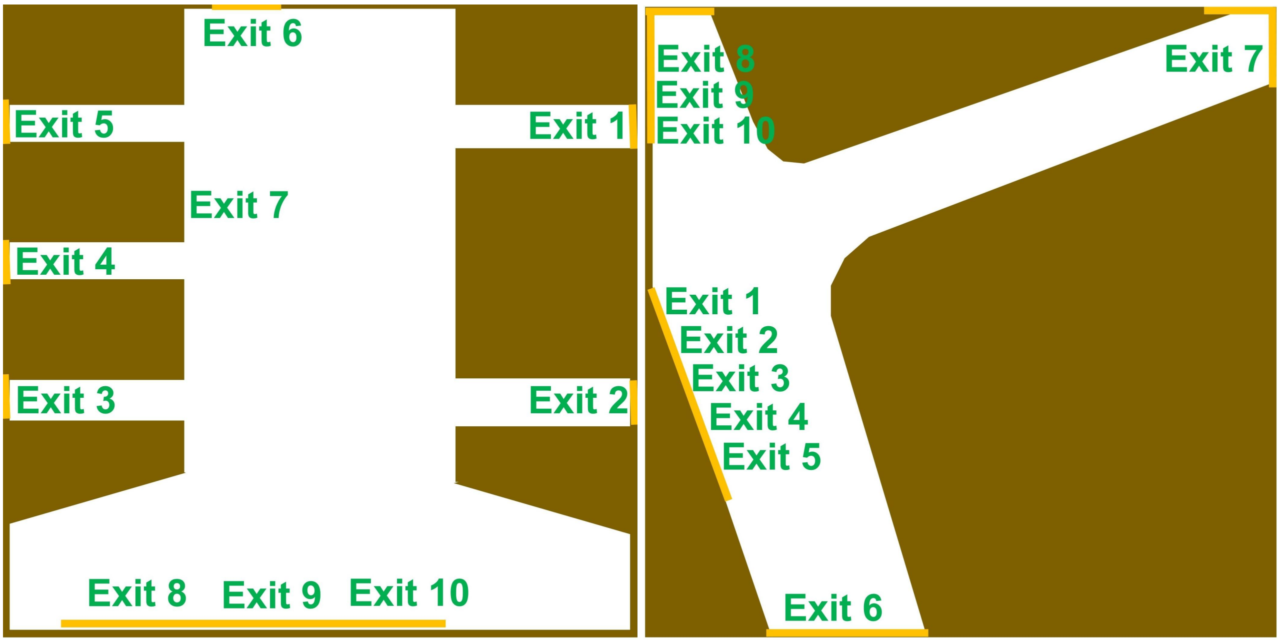
pedestrians required for enabling the 'autonomous change of direction' condition. **Note:** more information about how this threshold is effective in the 'autonomous change of direction' condition is provided in Shirvani & Kesserwani (2021).				
To define the initial number of pedestrians in the area.	*initial_population*	integer	none	Any positive value
To define the inflow hydrograph in terms of time and discharge (see Fig. 9a).	*inflow_start_time* *inflow_peak_time* *inflow_end_time*	float	second	Any positive value
	inflow_initial_discharge *inflow_peak_discharge* *inflow_end_discharge*		m^3/s	
To define the start and end time of the evacuation process.	*evacuation_start_time* *evacuation_end_time*	float	second	Any positive value
To specify the inflow boundary through which the floodwater starts to propagate.	*INFLOW_BOUNDARY*	integer	none	1: North boundary 2: East boundary 3: South boundary 4: West boundary
To define the boundary condition in the hydrodynamic model for the specified domain (Wang et al. 2011).	*BOUNDARY_EAST_STATUS* *BOUNDARY_WEST_STATUS* *BOUNDARY_NORTH_STATUS* *BOUNDARY_SOUTH_STATUS*	integer	none	1: Open boundary, allowing the floodwater to pass 2: Wall boundary, obstructing the floodwater
To define the location of the emergency exit. **Note:** the emergency exit is only applicable for evacuation from indoor areas where the emergency exit could be identified prior to the flooding, e.g. the shopping centre test case.	*emergency_exit_number*	integer	none	1 to 10 **Note:** the current version of the simulator can include up to 10 exits/entrances each of which is identified by a number. The location of Exit 1 to Exit 10 specified for the shopping centre and Hillsborough stadium test cases are presented in Fig. 10.
To define the length of the breach by specifying two points on the grid where floodwater is expected to propagate (see Fig. 9b).	*x1_boundary* *x2_boundary* *y1_boundary* *y2_boundary*	float	metre	Any positive value within the range of the domain length.
To specify the emergence rate of pedestrians per each iteration of the simulation at Exit 1 to Exit 10.	*EMMISION_RATE_EXIT1* *EMMISION_RATE_EXIT2* *EMMISION_RATE_EXIT3* *EMMISION_RATE_EXIT4* *EMMISION_RATE_EXIT5* *EMMISION_RATE_EXIT6* *EMMISION_RATE_EXIT7*	integer	none	Any positive value.

	EMMISION_RATE_EXIT7 *EMMISION_RATE_EXIT9* *EMMISION_RATE_EXIT10*			
To change the probability distribution of each exit to be selected by the pedestrians.	*EXIT1_PROBABILITY* *EXIT2_PROBABILITY* *EXIT3_PROBABILITY* *EXIT4_PROBABILITY* *EXIT5_PROBABILITY* *EXIT6_PROBABILITY* *EXIT7_PROBABILITY* *EXIT8_PROBABILITY* *EXIT9_PROBABILITY* *EXIT10_PROBABILITY*	integer	none	Any positive value. **Note:** in case of equal selection probability, assign the same value to all the exits/entrances.
To change the body height distribution of the pedestrians. **Note:** the distribution of the pedestrians' body height is set based on a cumulative probability for each range that is defined as a percentage of the total population.	*PedHeight_60_110_probability* *PedHeight_110_140_probability* *PedHeight_140_163_probability* *PedHeight_163_170_probability* *PedHeight_170_186_probability* *PedHeight_186_194_probability* *PedHeight_194_210_probability*	float	percent	0 to 100 **Note:** the current set up for the shopping centre and Hillsborough stadium test cases is based on the body height structure of the UK population (more information is provided in Shirvani et al. (2020)).
To change the age distribution of the pedestrians. **Note:** the distribution of the pedestrians' age is set based on a cumulative probability for each range that is defined as a percentage of the total population.	*PedAge_10_17_probability* *PedAge_18_29_probability* *PedAge_30_39_probability* *PedAge_40_49_probability* *PedAge_50_59_probability* *PedAge_60_69_probability* *PedAge_70_79_probability*	float	none	0 to 100 **Note:** the current set up for the shopping centre and Hillsborough stadium test cases is based on the seven age range structure of the UK population (more information is provided in Shirvani & Kesserwani (2021)).
To change the gender distribution of the pedestrians. **Note:** similar to the body height and age, the distribution of the pedestrians' age is set based on a cumulative probability for each range that is defined as a percentage of the total population.	*gender_female_probability* *gender_male_probability*	float	none	0 to 100



(a) (b)

Figure 9. Illustrative guidance to the users for (a) defining the inflow hydrograph through assigning values to the inflow parameters; and (b) defining the coordinates for specifying the location of the breach via assigning values to '*x1_boundary*', '*y1_boundary*', '*x2_boundary*' and '*y2_boundary*' in the *environment* element in the .xml input file.



(a) (b)

Figure 10. Schematic representation of the exits/entrances specified for: (a) the shopping centre and (b) the Hillsborough stadium test cases.

6. Instructions to create a new test case

The following steps will guide the users on how to create new test cases.

Step 1. Generating initial state of hydrodynamic and navigation agents.

Step 1.1: flood agents are required to be initialised by cartesian coordinates on a grid storing topography data of the subjected area. Users may use any method to generate the grid of flood agents, but it must be converted into XML format using the parent-child structure and naming shown below for each flood agent:

<xagent>	
*<name>*FloodCell*</name>*	'Floodcell' is name of flood agent
*<inDomain>*1*</inDomain>*	1 or 0 indicate agent is inside or outside the study area
<x>...</x>	a positive integer indicating the *x*-coordinate of the agent
<y>...</y>	a positive integer indicating the *y*-coordinate of the agent
<z0>...</z0>	Topography elevation at the location of the flood agent
</xagent>	

Note:

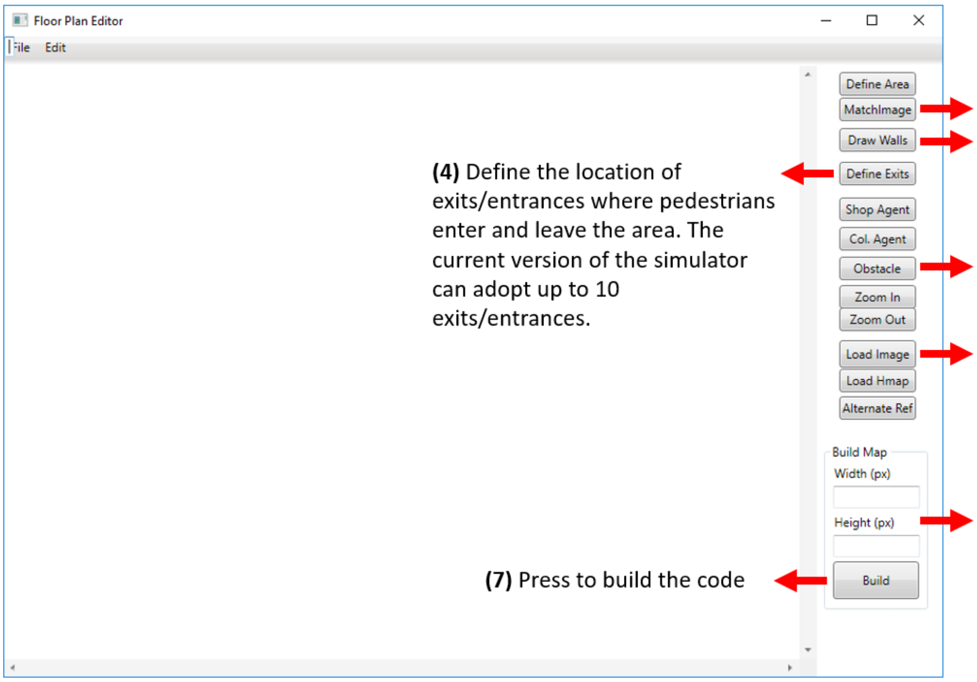
- In the current version of the simulator, the grid of flood agents must be defined as square with a length size in power of two (e.g. 256, 512, 1024, etc.).
- The code that was used to generate the grid of flood agents for the flooded shopping centre test case in Shirvani et al. (2021) and Shirvani et al. (2020), is accessible from another GitHub repository via this link: https://github.com/SahebSh/FLAMEGPU/tree/master/examples/FloodPedestrian_2018/Flood_XML_inpGe. This code is written in C++ and can be opened and executed from Visual Studio on Windows. Users can use this model to modify the shopping centre test case or take it as an example to produce another one.

Step 1.2: the grid of navigation agents are also required to be produced via using the FGPUGridNavPlanEditor package that is specifically designed for this purpose. A modified version of this model with the specific XML structure and component naming compatible with the present version of the simulator is also accessible for the user from another GitHub repository via this link: https://github.com/SahebSh/FLAMEGPU/tree/master/examples/FloodPedestrian_2018/FGPUGridNavPlanEditor.

After running FGPUGridNavPlanEditor via Visual Studio, a self-explanatory graphical user interface will pop up on the screen as shown in Figure 11. The user can take the steps shown in Figure 11 to generate the grid of navigation agents.

Note:

the pedestrian flow will be automatically generated over the grid of navigation agents during the simulations from where the exits are defined by the user; therefore, there is no need to take further steps for initialisation of pedestrian agents over the grid. The movement of pedestrian agents is driven by their internal interactions with each other and the information they receive from the navigation agents that steers and directs them towards their goal destination on the grid. More information about the interactions between the grid of navigation agents and pedestrian agents can be found in Karmakharm et al. (2010). It is also useful to note that, unlike the hydrodynamic and navigation agents, the pedestrian agents are not of discrete type; they are of continuous agents type that can dynamically change their coordinates.



(1) Load the topography map as an image.

(2) Define the study area around the image.

(3) Draw a closed boundary to define where the pedestrians move.

(4) Define the location of exits/entrances where pedestrians enter and leave the area. The current version of the simulator can adopt up to 10 exits/entrances.

(5) Define areas where the obstacles are located.

(6) Specify the number of navigation agents in each direction. It must always be a power of two number (e.g. 256, 512, 1024, etc.) and it must be equal to the number of hydrodynamic agents.

(7) Press to build the code

Figure 11. Screenshot of the FGPUGridNavPlanEditor graphical user interface

The generated grids of flood and navigation agents from Step 1.1 and Step 1.2 need to be placed within the .xml input file later in Step 2 (below) in order to make the information accessible to the simulator. The new input file could be named 'map.xml' and located in '../iterations' folder containing the information of all the agents and the parameters enabling the users to set up other case studies.

Note:

Naming of the .xml input file should always be consistent with the address and name that is specified in the Command Arguments (see Sect. 3, Step 8) from the last build; otherwise, the executable file should be built again.

Step 2: setting up the .xml input file.

The .xml input file should always be structured based on a '<*states*>' parent and three childs in xml format, as outlined below:

<states>	
<itno>0<itno>	Indicate the iteration number. Must be initialised to 0 to load ‘map.xml’ in order to start a simulation.
<environment></environment>	Contains all the parameters that the user can modify to fit the simulator to a case study (see Table 4 for detailed description).
<xagents></xagents>	Contain the initial state of flood and navigation agents generated from Step 1. The user is not expected to change any of these variables directly from the .xml input file, but rather, S/he needs to produce them earlier to fit to the study area specification (recall Step 1 above and overwrite agent data in the .xml input file).
</states>	

Note:

- The users are allowed to use the contents of the already generated and provided input files (HilStad.xml and ShopCent.xml) and copy/paste the new flood and navigation agents information into the *xagents* element following the same template explained above.

- Steps 1-6 from Sect 3 should be repeated each time the user applies the simulator for a different grid size of navigation and flood agents. In this case, the user is also expected to adapt the simulator for the new grid size prior to the building procedure. To do so, click and open ‘**XMLModelFile.xml**’ from the ‘Solution Explorer’. Then find **gpu:bufferSize** tag where the total number of agents spanning over the generated grid is specified, e.g. for a grid of 128 × 128 navigation/flood agents, the **gpu:bufferSize** is set to 16384. The user may use Ctrl+f shortcut to find the old **gpu:bufferSize** value within the ‘**XMLModelFile.xml**’ file and replace it with a new one. Once the new value is assigned to **gpu:bufferSize**, save the changes via Ctrl+s (or clicking on 💾 below the top menu) and close the ‘**XMLModelFile.xml**’ tab. Also, to know more about the **XMLModelFile.xml** file and its contents visit FLAMEGPU Documentation and User guide.

References

Karmakharm, T., Richmond, P., & Romano, D. M. (2010). Agent-based Large Scale Simulation of Pedestrians With Adaptive Realistic Navigation Vector Fields. *TPCG*, 10, 67-74.

Shirvani, M., Kesserwani, G., & Richmond, P. (2020). Agent-based modelling of pedestrian responses during flood emergency: mobility behavioural rules and implications for flood risk analysis. *Journal of Hydroinformatics*, 22(5), 1078-1092.

Shirvani, M., Kesserwani, G., & Richmond, P. (2021). Agent‐based simulator of dynamic flood‐people interactions. *Journal of Flood Risk Management* [early view], e12695.

Shirvani, M. and Kesserwani, G. (2021). Flood-pedestrian simulator for modelling human response dynamics during flood-induced evacuation: Hillsborough stadium case study. *Journal of Natural Hazards and Earth System Sciences* [in review].

Wang, Y., Liang, Q., Kesserwani, G., & Hall, J. W. (2011). A 2D shallow flow model for practical dam-break simulations. *Journal of Hydraulic Research*, 49(3), 307-316.