Fire Will Kill Us All

Created by:
Atanas Marchev
Filippo Nardocci
Nikolay Ganev
Yosif Kiradzhiev
Stef van den Tempel

Contents

PROCESS REPORT

This document will highlight the development and decision-making process of our application, how it works and why it does so in a certain way.

OVERVIEW

FireWillKillUsAll is an application which simulates the events that take place in a situation where a fire starts in closed quarters. Users can design their own floor layout: the shape and size of different rooms, the location of fire-doors and extinguishers and how many persons are taking part in the event.

This is done with a Winforms interface which communicates in real time with a Unity application through a WCF service. The Unity application is solely used to display what is happening during the simulation. The simulation uses two different timers (one master timer and a timer used to determine the speed of the fire propagation) which allow the program to change the state of the user-defined layout dynamically.

The team strived to achieve a somewhat high level of realism to depict the events during each simulation. Special attention was placed into how the fire can spread and propagate throughout the floor and how the people in the simulation react to it.

TILES and MAIN GRID

The main data-structure used by the application is a 2d array of Tile objects. A Tile can be considered as a node object, linked to its neighboring tiles in the array, used to store and change information over the state of the layout while the simulation is running.

The 2d array is created through the static class Grid, which is also responsible of assigning a color to the tiles according to their content.

Each tile has a color field. Color-coding is used to determine what kind of part of the layout the tile represents; different colors are assigned to tiles to represent their static content (such as walls), and their dynamic content (such as Person objects walking on the tile or fire spreading on those tiles).

Color coding of the tiles was chosen as it represented a simple and effective way for instantiating the initial grid, create the user defined layout and then determining the contents of the grid dynamically.

FLOORS, ROOMS and FIRE SPREADING

Each floor object contains a list of rooms. These can be created by the user before the start of the simulation. Rooms are in essence list of tiles objects separated from other rooms by tiles containing walls or doors.

The Room class main purpose is to manage the fire spreading logic; this includes determining where and how quickly to create new instances of Fire objects to be placed upon individual tiles.

Each room contains a fuel value representing how much flammable material the room has, together with a set of other fields related to how the fire can spread inside the room, such as air currents and temperature.

The Floor class main purpose is to manage the dynamic behavior of what happens in a room when a fire or a person are inside of it. This is the class which receives calls from the timer of the client and in turn sets the behavior of its rooms.

On every tick of the main timer from the client class, the Floor object interacts with the fire spread timer, which is used by a Room with instances of Fire on one or more of its tiles. The speed of the fire spread timer depends on several aspects; in real life, fire tends to spread more rapidly depending on the amount of fuel it can burn and on the amount of oxygen at its disposal. For this reason, the fire spread timer can change its speed depending of the contents of each particular room.

Under certain conditions, a room with enough fuel can reach an unsustainable temperature and create a real-life phenomenon called flashover.

During a flashover most combustible material of a room would ignite almost simultaneously; the Room class employs several methods to recreate this situation and spread fire to its tiles accordingly.

PEOPLE, EXTINGUISHERS and PATHFINDING

The Person class is responsible to represent the people interacting with the fire and manage their behavior.

To increase realism of the simulation the team tried to add some randomness to it by trying to recreate what can be referred to as human factor. Each Person object gets assigned a personality property (implemented with an enumeration) which will determine its behavior with respect to the fire. Some personalities will run straight to the exit if possible, some will try to actively extinguish the fire, while others will panic and do very little.

The movement of a Person object is performed through implementing either the A-star or Dijkstra (according to the situation) algorithms for shortest paths using Tile objects as nodes.

A Person object can interact with fire by picking one of the Extinguisher objects scattered around the floor and use it to decrease the intensity field of the particular Fire instance they are interacting with. A tile is considered extinguished if the intensity of the Fire instance on it reaches zero.

Extinguishers have their own fuel values and need to be changed if they get “empty”. They can also malfunction, leaving the Person object using it with no choice but to run to pick up another one.

STATISTICS MODULE

The Statistics class is responsible of keeping track of specific circumstances that are affecting the simulation, such as which room is on fire, if a person is using an extinguisher, or if another has managed to run to safety.

It does so via a sequence of event calls from the objects taking part in the simulation and displaying them in real-time onto its own Winforms client.

GLOBAL STATISTICS MODULE

A feature that could not make it in the final version of the application, the global statistics module was supposed to generate sets of simulations with different states and running them sequentially in the background in order to suggest the user an optimal placement for fire extinguishers in the given layout.

Its desired implementation is described here for reference and in case someone in the future will like to develop it correctly.

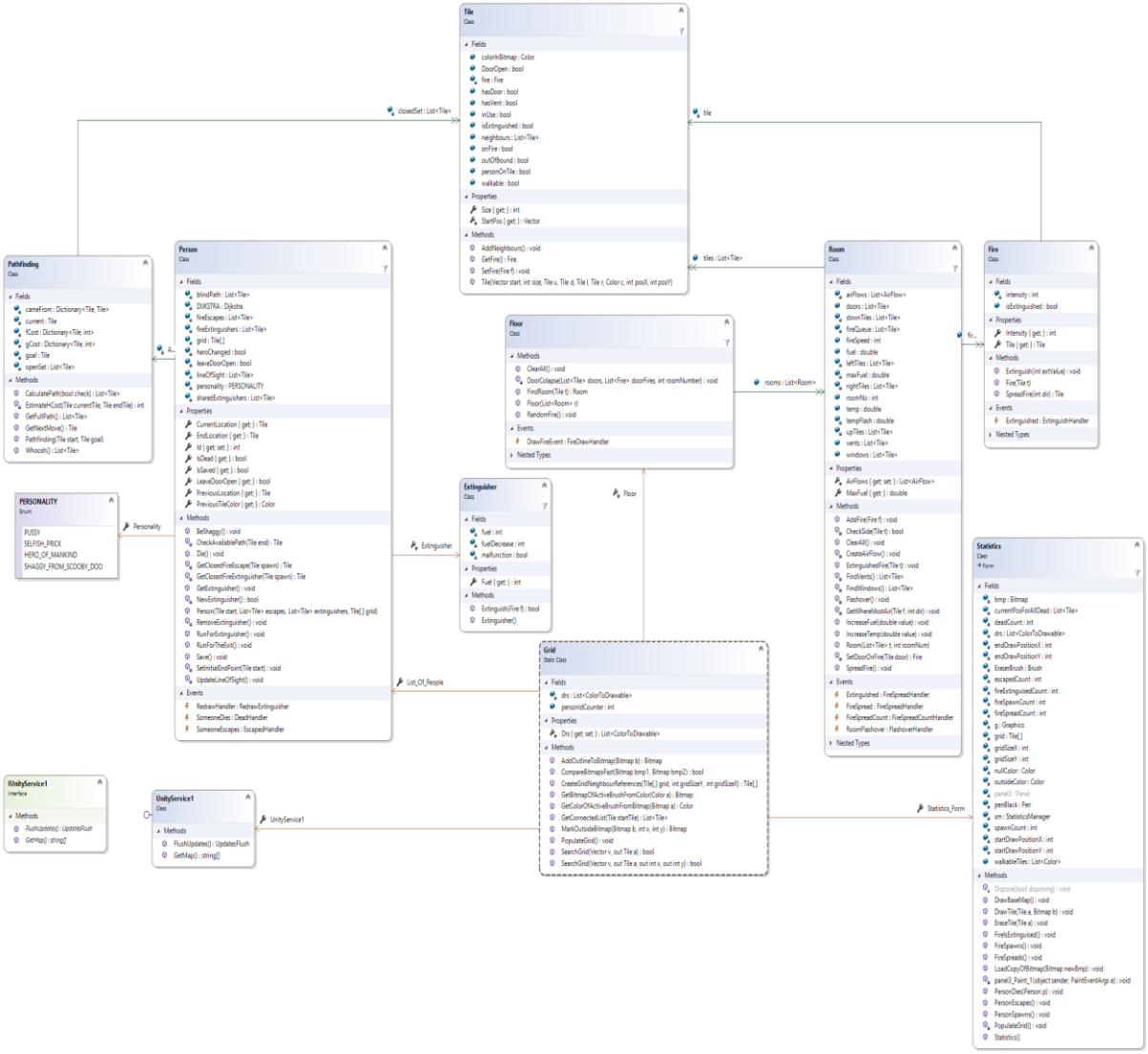
The module was supposed to receive the state of the simulation at its start, recreate it and run it without using the graphical component. After a generic number of runs, working in tandem with the statistics module, it would generate an average value of how many people could reach safety. After doing so the module would change the position of the extinguishers of the start layout and repeat the process. After covering all desired positionings of the extinguishers the module would have output the layout of the set of simulation which had the better ratio of deaths against people who got saved.

Due to design and time constraints, the feature will not be part of the application. For reference a suggested class diagram is included in the design document section of this document.

DESIGN DOCUMENT

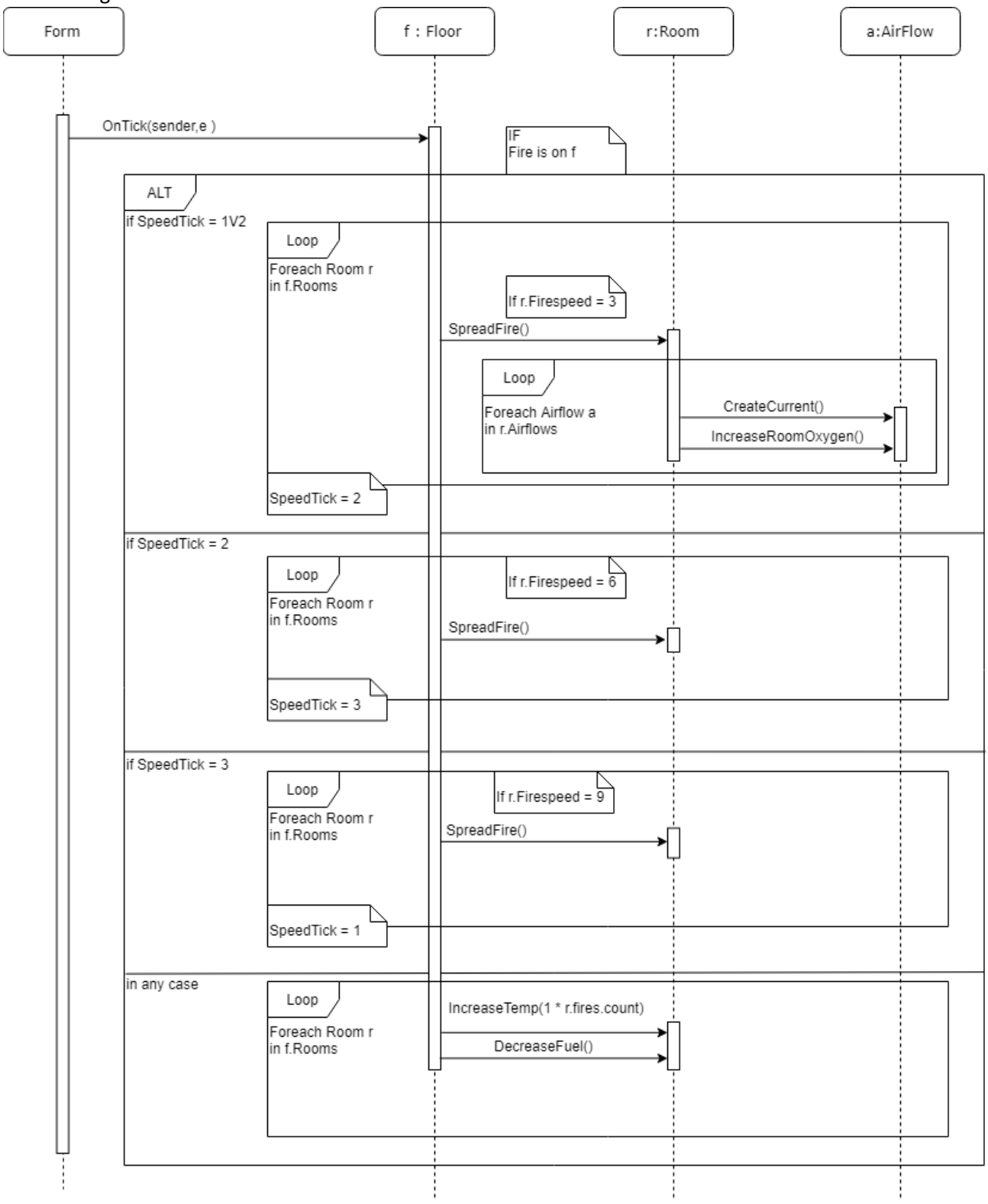
UML DIAGRAM

A clearer version of the diagram can be found at this link.



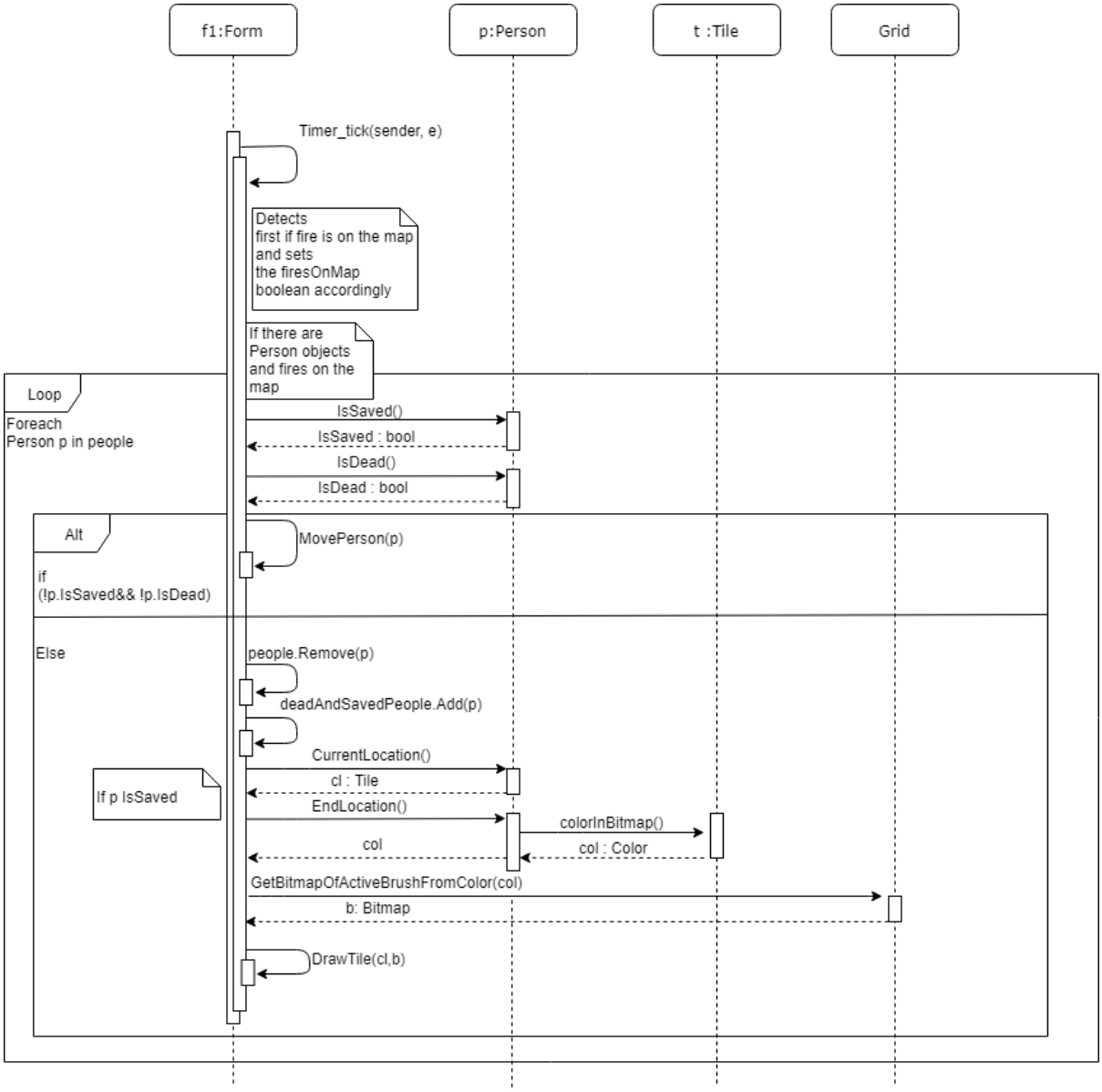
SEQUENCE DIAGRAM: Floor.OnTick(sender, e)

The diagram depicts how the speed of the spreading of fire changes dynamically throughout the floor during the simulation.



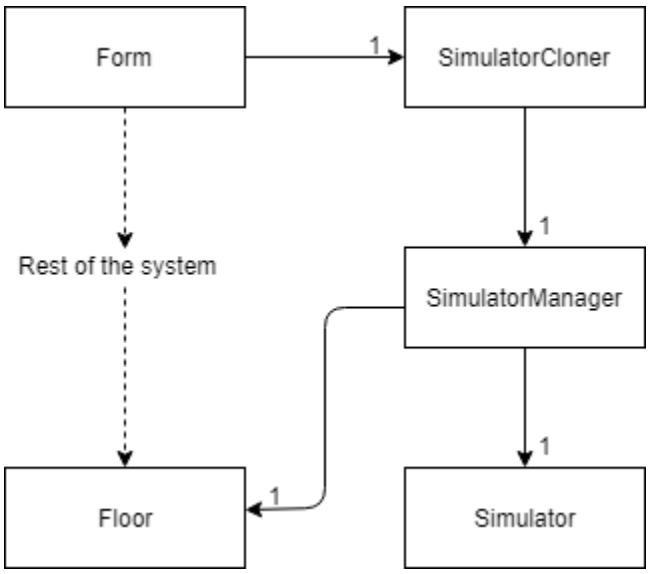
SEQUENCE DIAGRAM: Main timer tick

The diagram depicts the sequence activated at every tick of the main timer.



APPENDIX A: Proposed solutions to implement the Global Statistics Module

The class diagram depicted here shows the way the team tried to implement the module:



The idea behind It was to have a cloner class retrieving all relevant information on the user defined simulation at the start of the timer of the form class. The cloner in turn would send the cloned data to a manager class, in charge of running single instances of the simulator class (where the simulation would behave in the same manner as the one displayed in the app, only without a graphical component). These instances would be grouped into sets of a generic number of simulations, each with different placement of the extinguishers in the floor (change of layout would be performed in the manager class). These sets would in turn return statistical averages over the amount of people saved in every simulation of the set. After running all the constructed sets, the module would display the layout of the simulation set which had the best average to the user.

Unfortunately, due technical constraints the implementer of the feature (Nardocci) could not manage to find a way to decouple the logic of the rest of the application from its graphical components. This led to instances of the simulator running in a way which did not follow the behavior of the original system, thus had to be scrapped.