ICT DM 2023

THE 8TH INTERNATIONAL CONFERENCE ON
INFORMATION AND COMMUNICATION TECHNOLOGIES
FOR DISASTER MANAGEMENT

Cosenza

September 13th - 15th | Cosenza, Italy

A simulation tool for crisis management and pre-disaster planning

Federico Eugeni, Sara Sacco, Donato Di Ludovico, Antinisca Di Marco

DICEAA - DISIM, Università degli Studi dell’Aquila

Track 2 | AI, Big data, ontologies and analytics for crisis management

Cosenza – September 13th - 15th, 2023

Introduction

The research described in this paper is part of three research projects in which the University of L'Aquila is involved as a partner: SICURA Project – "Intelligent house of technologies for safety - L'Aquila" - Emerging Technologies Support Programme (FSC 2014-2020) - Axis I "House of Emerging Technologies", Research Programme: Safe City: urban design and technologies for urban safety; the National Centre for HPC, Big Data and Quantum Computing - PNRR Project, funded by the European Union - Next Generation EU; Territori Aperti, a centre for documentation, training and research for the reconstruction and recovery of disaster-affected fragile territories, funded by Fondo Territori Lavoro Conoscenza - CGIL CISL UIL.

SICURA
5G
caSa Intelligente delle teCnologie per la sicURezza - L'Aquila

ICSC
Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

TERRITORI APERTI

Introduction

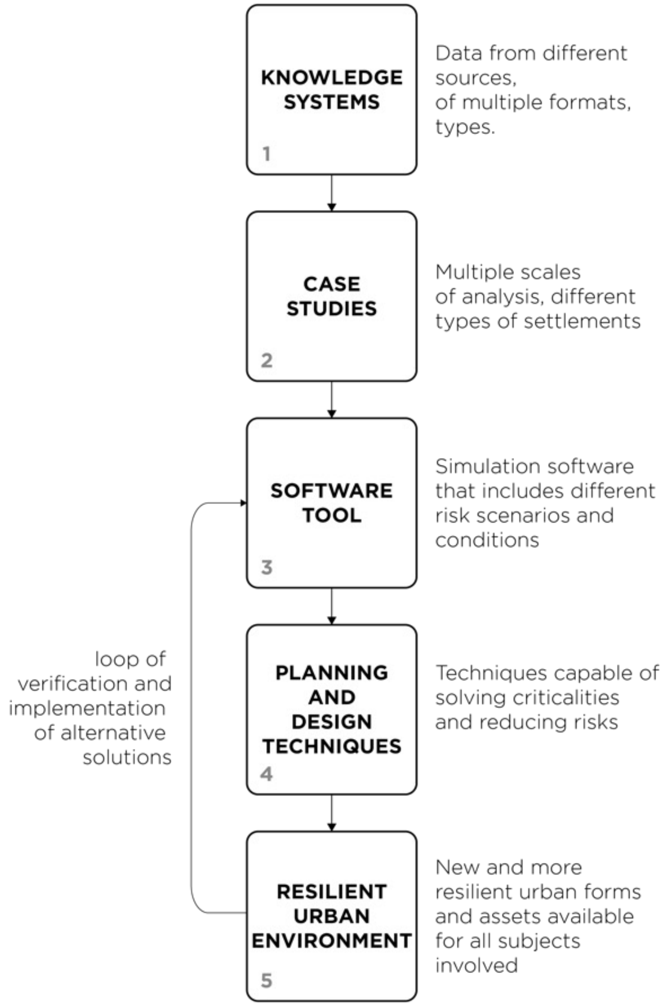
One of the issues on which today's urban planner is probably called upon to question himself is the ability to succeed in shaping a dynamic approach to face the challenges and events that contemporary life presents us with, placing himself in a position that is not always simple, trying to elaborate **methodologies and implement innovative tools that can support the planning practice**.

A fundamental part of this practice lies in the **virtual representation of conditions, behaviour and risk scenarios** in order to **plan, communicate and inform** citizens, public administrations and stakeholders living in the territories, regardless of their disaster history.

In this way, it will be possible to **define new urban planning and design practices** oriented towards the safety of the city's people and thus increase the cities' resilience.

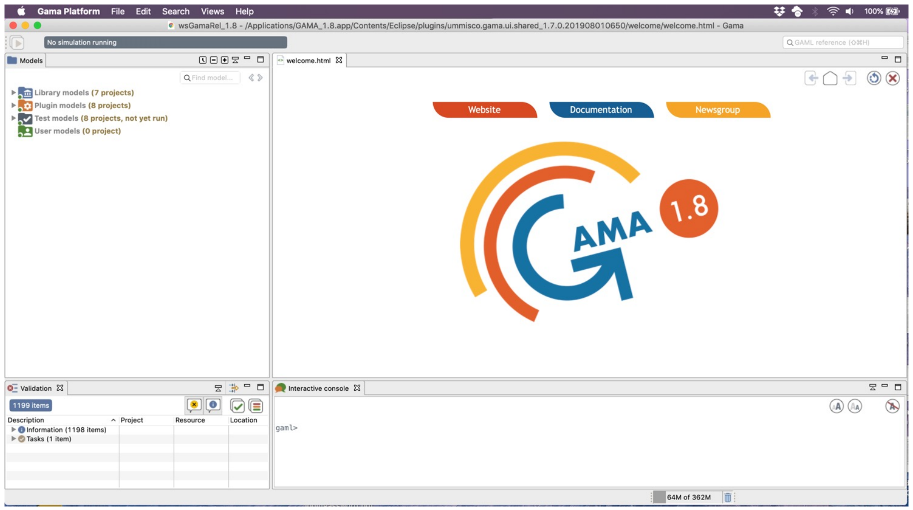
Methodology

The methodology, as depicted in figure is composed of five phases: (1) the construction of a knowledge system in the form of an Urban Digital Twin (UDT), (2) the selection of cross-scale case studies of the UDT on which to apply the output of phase 3, **(3) the realization of a software application with agent-based programming techniques aimed at simulating evacuation during a disaster**, (4) the definition of urban design techniques on the case studies for reducing risks during evacuation, and (5) the feedback application of phase 3 to verify the urban performance of the techniques defined in phase 4.


KNOWLEDGE SYSTEMS
1
Data from different sources, of multiple formats, types.
CASE STUDIES
2
Multiple scales of analysis, different types of settlements
SOFTWARE TOOL
3
Simulation software that includes different risk scenarios and conditions
PLANNING AND DESIGN TECHNIQUES
4
Techniques capable of solving criticalities and reducing risks
RESILIENT URBAN ENVIRONMENT
5
New and more resilient urban forms and assets available for all subjects involved
loop of verification and implementation of alternative solutions

The script

The chosen software is **Gama Platform**: a modelling and simulation development environment for the construction of spatially explicit agent-based simulations. It is possible to import and visualize **shapefiles** (which contain georeferenced data) with which to conduct analyses at different spatial levels and scales of representation. Although GAMA provides a scientific approach to constructing and exploring models, **it was also developed for use by researchers outside the field of data science**.



wsGamaRel_1.8 - /Applications/GAMA_1.8.app/Contents/Eclipse/plugins/ummisco.gama.ui.shared_1.7.0.201908010650/welcome/welcome.html - Gama

No simulation running

- Library models (7 projects)
- Plugin models (8 projects)
- Test models (8 projects, not yet run)
- User models (0 project)

1199 items

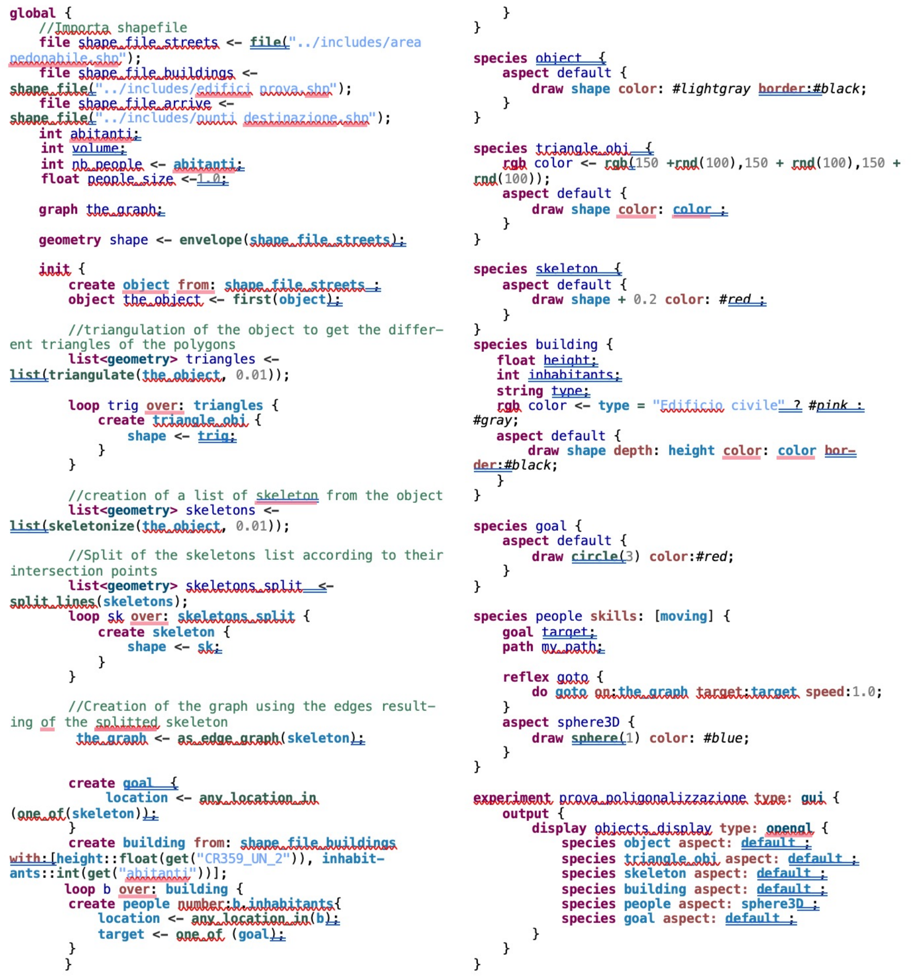
Description	Project	Resource	Location
Information (1198 items)			
Tasks (1 item)			

```
gaml>
```

64M of 362M

The script

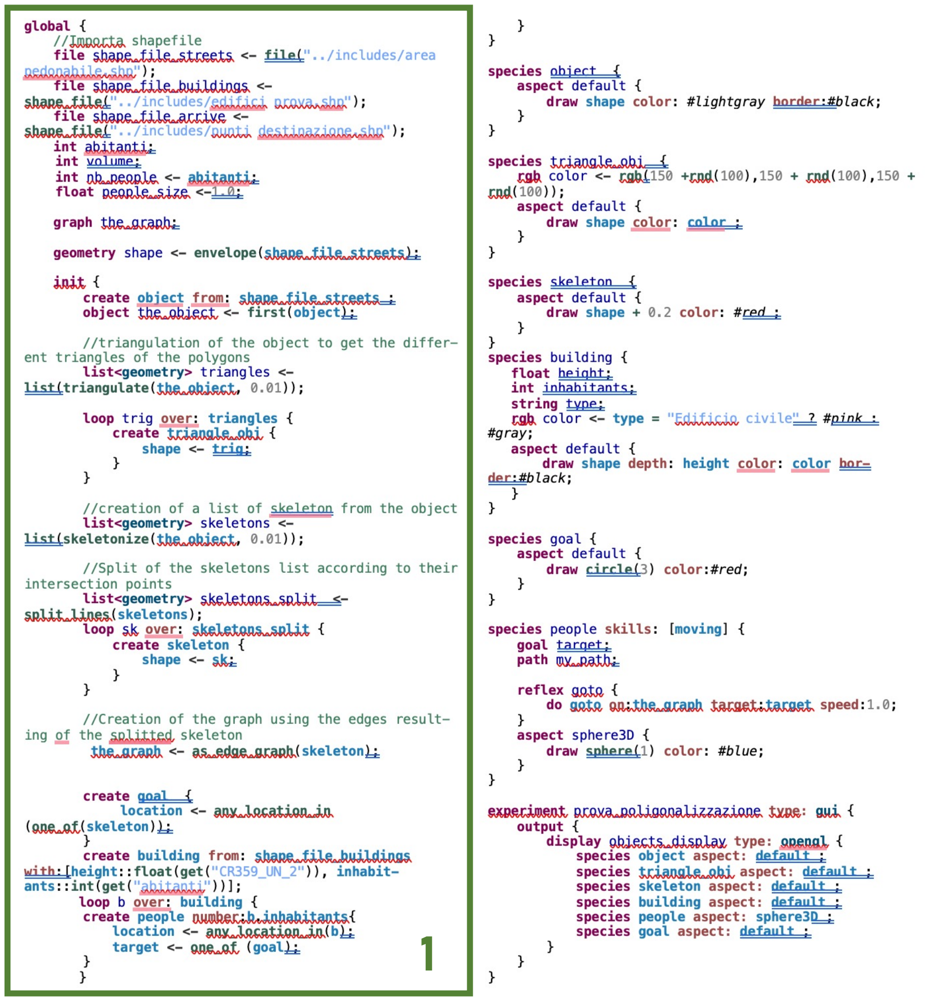
The choice was mainly oriented by the open-source nature of the software itself. As mentioned, in GAMA, processing can be implemented thanks to a programming language (based on Java) in which it is possible to define variables of different types related to each other. Variables defined represent the heart of the algorithm because they explain the fundamental components of the simulations. It is also possible to import data of different types, both geospatial and geometric (both two- and three-dimensional), which are displayed. The structure of the script, which is currently very lean and **affected by some limitations** that will be mentioned later, is presented in the picture.



```
global {
    //Importa shapefile
    file shape_file_streets <- file("../includes/area
pedonabile.shp");
    file shape_file_buildings <-
shape_file("../includes/edifici prova.shp");
    file shape_file_arrive <-
shape_file("../includes/punti destinazione.shp");
    int abitanti;
     int volume;
     int nb_people <- abitanti;
     float people_size <-1.0;

    graph the_graph;

    geometry shape <- envelope(shape_file_streets);

    init {
        create object from: shape_file_streets ;
        object the_object <- first(object);

        //triangulation of the object to get the differ-
ent triangles of the polygons
        list<geometry> triangles <-
list(triangulate(the_object, 0.01));

        loop trig over: triangles {
            create triangle_obj {
                shape <- trig;
            }
        }

        //creation of a list of skeleton from the object
        list<geometry> skeletons <-
list(skeletonize(the_object, 0.01));

        //Split of the skeletons list according to their
intersection points
        list<geometry> skeletons_split  <-
split_lines(skeletons);
        loop sk over: skeletons_split {
            create skeleton {
                shape <- sk;
            }
        }

        //Creation of the graph using the edges result-
ing of the splitted skeleton
         the_graph <- as_edge_graph(skeleton);


        create goal  {
             location <- any_location_in
(one_of(skeleton));
        }
        create building from: shape_file_buildings
with:[height::float(get("CR359_UN_2")), inhabit-
ants::int(get("abitanti"))];
       loop b over: building {
        create people number:b.inhabitants{
            location <- any_location_in(b);
            target <- one_of (goal);
        }
       }
    }
}

species object  {
    aspect default {
        draw shape color: #lightgray border:#black;
    }
}

species triangle_obj  {
    rgb color <- rgb(150 +rnd(100),150 + rnd(100),150 +
rnd(100));
    aspect default {
        draw shape color: color ;
    }
}

species skeleton  {
    aspect default {
        draw shape + 0.2 color: #red ;
    }
}
species building {
   float height;
   int inhabitants;
   string type;
   rgb color <- type = "Edificio civile" ? #pink :
#gray;
   aspect default {
       draw shape depth: height color: color bor-
der:#black;
   }
}

species goal {
    aspect default {
        draw circle(3) color:#red;
    }
}

species people skills: [moving] {
    goal target;
    path my_path;

    reflex goto {
        do goto on:the_graph target:target speed:1.0;
    }
    aspect sphere3D {
        draw sphere(1) color: #blue;
    }
}

experiment prova_poligonalizzazione type: gui {
    output {
        display objects_display type: opengl {
            species object aspect: default ;
            species triangle_obj aspect: default ;
            species skeleton aspect: default ;
            species building aspect: default ;
            species people aspect: sphere3D ;
            species goal aspect: default ;
        }
    }
}
```

1st Section

The first section, called '**global**', concerns the general parameters of the simulation, to which the contents of the other two refer. In fact, this section defines the first actions that, once the simulation is launched, are carried out by the software, such as the **import of databases and basic files**, in this case of a **geospatial type**, their processing and the strategy for using these data.



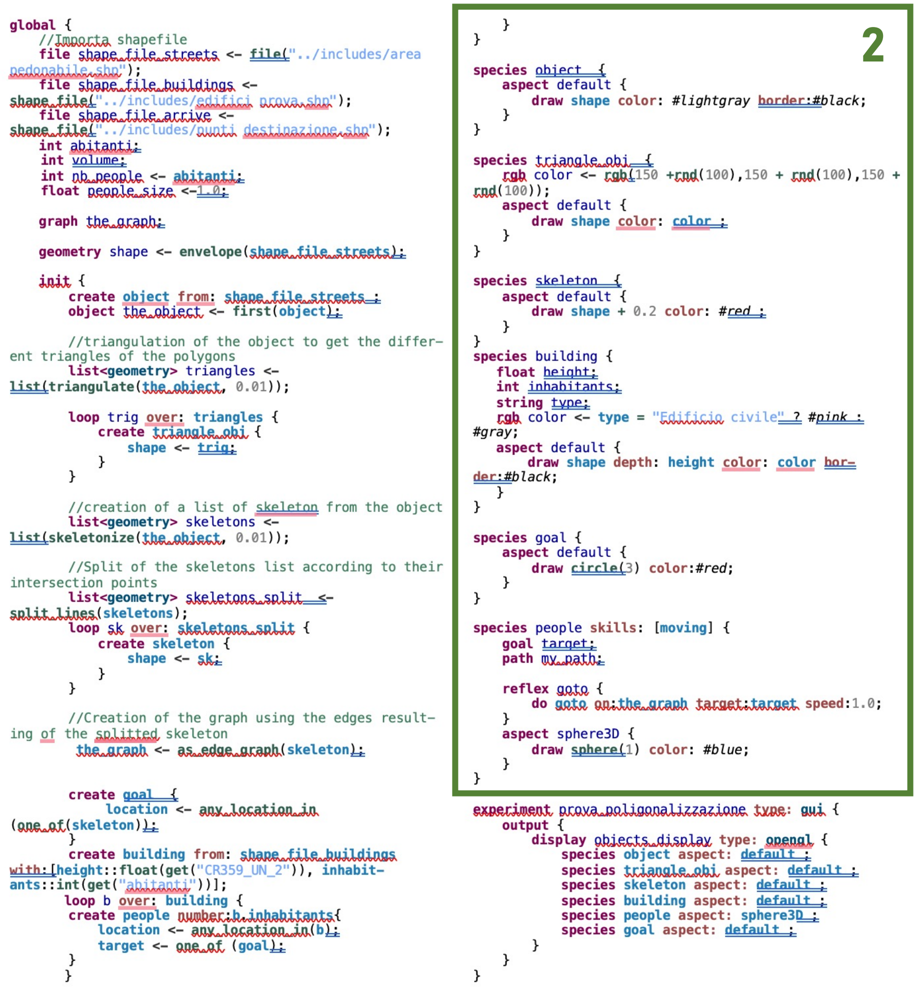
```
global {
    //Importa shapefile
    file shape_file_streets <- file("../includes/area
pedonabile.shp");
    file shape_file_buildings <-
shape_file("../includes/edifici prova.shp");
    file shape_file_arrive <-
shape_file("../includes/punti destinazione.shp");
    int abitanti;
    int volume;
    int nb_people <- abitanti;
    float people_size <-1.0;

    graph the_graph;

    geometry shape <- envelope(shape_file_streets);

    init {
        create object from: shape_file_streets ;
        object the_object <- first(object);

        //triangulation of the object to get the differ-
ent triangles of the polygons
        list<geometry> triangles <-
list(triangulate(the_object, 0.01));

        loop trig over: triangles {
            create triangle_obj {
                shape <- trig;
            }
        }

        //creation of a list of skeleton from the object
        list<geometry> skeletons <-
list(skeletonize(the_object, 0.01));

        //Split of the skeletons list according to their
intersection points
        list<geometry> skeletons_split  <-
split_lines(skeletons);
        loop sk over: skeletons_split {
            create skeleton {
                shape <- sk;
            }
        }

        //Creation of the graph using the edges result-
ing of the splitted skeleton
         the_graph <- as_edge_graph(skeleton);


        create goal  {
             location <- any_location_in
(one_of(skeleton));
        }
        create building from: shape_file_buildings
with:[height::float(get("CR359_UN_2")), inhabit-
ants::int(get("abitanti"))];
       loop b over: building {
        create people number:b.inhabitants{
            location <- any_location_in(b);
            target <- one_of (goal);
        }
       }
```

1

```
    }
}

species object  {
    aspect default {
        draw shape color: #lightgray border:#black;
    }
}

species triangle_obj  {
    rgb color <- rgb(150 +rnd(100),150 + rnd(100),150 +
rnd(100));
    aspect default {
        draw shape color: color ;
    }
}

species skeleton  {
    aspect default {
        draw shape + 0.2 color: #red ;
    }
}
species building {
   float height;
   int inhabitants;
   string type;
   rgb color <- type = "Edificio civile" ? #pink :
#gray;
   aspect default {
       draw shape depth: height color: color bor-
der:#black;
   }
}

species goal {
    aspect default {
        draw circle(3) color:#red;
    }
}

species people skills: [moving] {
    goal target;
    path my_path;

    reflex goto {
        do goto on:the_graph target:target speed:1.0;
    }
    aspect sphere3D {
        draw sphere(1) color: #blue;
    }
}

experiment prova_poligonalizzazione type: gui {
    output {
        display objects_display type: opengl {
            species object aspect: default ;
            species triangle_obj aspect: default ;
            species skeleton aspect: default ;
            species building aspect: default ;
            species people aspect: sphere3D ;
            species goal aspect: default ;
        }
    }
}
```

2nd Section

The second section, on the other hand, concerns the definition of the **different types of agents**, called '**species**', which play the role of protagonists in the simulation. In this case, four types of species are defined: street graphs, buildings, arrival points, people. **Actions, or reflexes**, that the agents representing them must have, once the simulation is launched, are defined in relation to the other species (e.g., here we define the type of **movement** they must perform following the road graph).



```
global {
    //Importa shapefile
    file shape_file_streets <- file("../includes/area
pedonabile.shp");
    file shape_file_buildings <-
shape_file("../includes/edifici prova.shp");
    file shape_file_arrive <-
shape_file("../includes/punti destinazione.shp");
    int abitanti;
     int volume;
     int nb_people <- abitanti;
     float people_size <-1.0;

    graph the_graph;

    geometry shape <- envelope(shape_file_streets);

    init {
        create object from: shape_file_streets ;
        object the_object <- first(object);

        //triangulation of the object to get the differ-
ent triangles of the polygons
        list<geometry> triangles <-
list(triangulate(the_object, 0.01));

        loop trig over: triangles {
            create triangle_obj {
                shape <- trig;
            }
        }

        //creation of a list of skeleton from the object
        list<geometry> skeletons <-
list(skeletonize(the_object, 0.01));

        //Split of the skeletons list according to their
intersection points
        list<geometry> skeletons_split  <-
split_lines(skeletons);
        loop sk over: skeletons_split {
            create skeleton {
                shape <- sk;
            }
        }

        //Creation of the graph using the edges result-
ing of the splitted skeleton
         the_graph <- as_edge_graph(skeleton);


        create goal  {
             location <- any_location_in
(one_of(skeleton));
        }
        create building from: shape_file_buildings
with:[height::float(get("CR359_UN_2")), inhabit-
ants::int(get("abitanti"))];
       loop b over: building {
        create people number:b.inhabitants{
            location <- any_location_in(b);
            target <- one_of (goal);
        }
       }
```

2

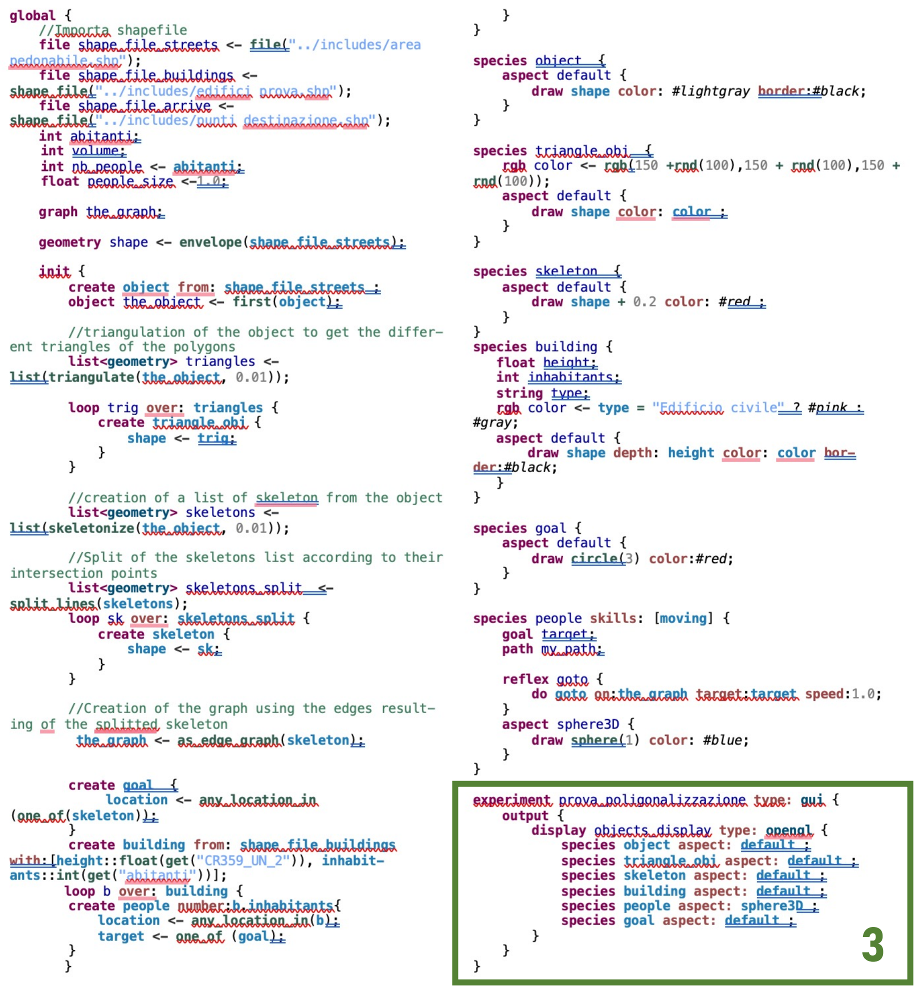
```
    }
}

species object  {
    aspect default {
        draw shape color: #lightgray border:#black;
    }
}

species triangle_obj  {
    rgb color <- rgb(150 +rnd(100),150 + rnd(100),150 +
rnd(100));
    aspect default {
        draw shape color: color ;
    }
}

species skeleton  {
    aspect default {
        draw shape + 0.2 color: #red ;
    }
}
species building {
   float height;
   int inhabitants;
   string type;
   rgb color <- type = "Edificio civile" ? #pink :
#gray;
   aspect default {
       draw shape depth: height color: color bor-
der:#black;
   }
}

species goal {
    aspect default {
        draw circle(3) color:#red;
    }
}

species people skills: [moving] {
    goal target;
    path my_path;

    reflex goto {
        do goto on:the_graph target:target speed:1.0;
    }
    aspect sphere3D {
        draw sphere(1) color: #blue;
    }
}
experiment prova_poligonalizzazione type: gui {
    output {
        display objects_display type: opengl {
            species object aspect: default ;
            species triangle_obj aspect: default ;
            species skeleton aspect: default ;
            species building aspect: default ;
            species people aspect: sphere3D ;
            species goal aspect: default ;
        }
    }
}
```

3rd Section

In the third section, currently the shortest, we define the **outputs of the experiment**, which will then be displayed graphically in real-time as the simulation runs its course.

The simulation environment shows the following:

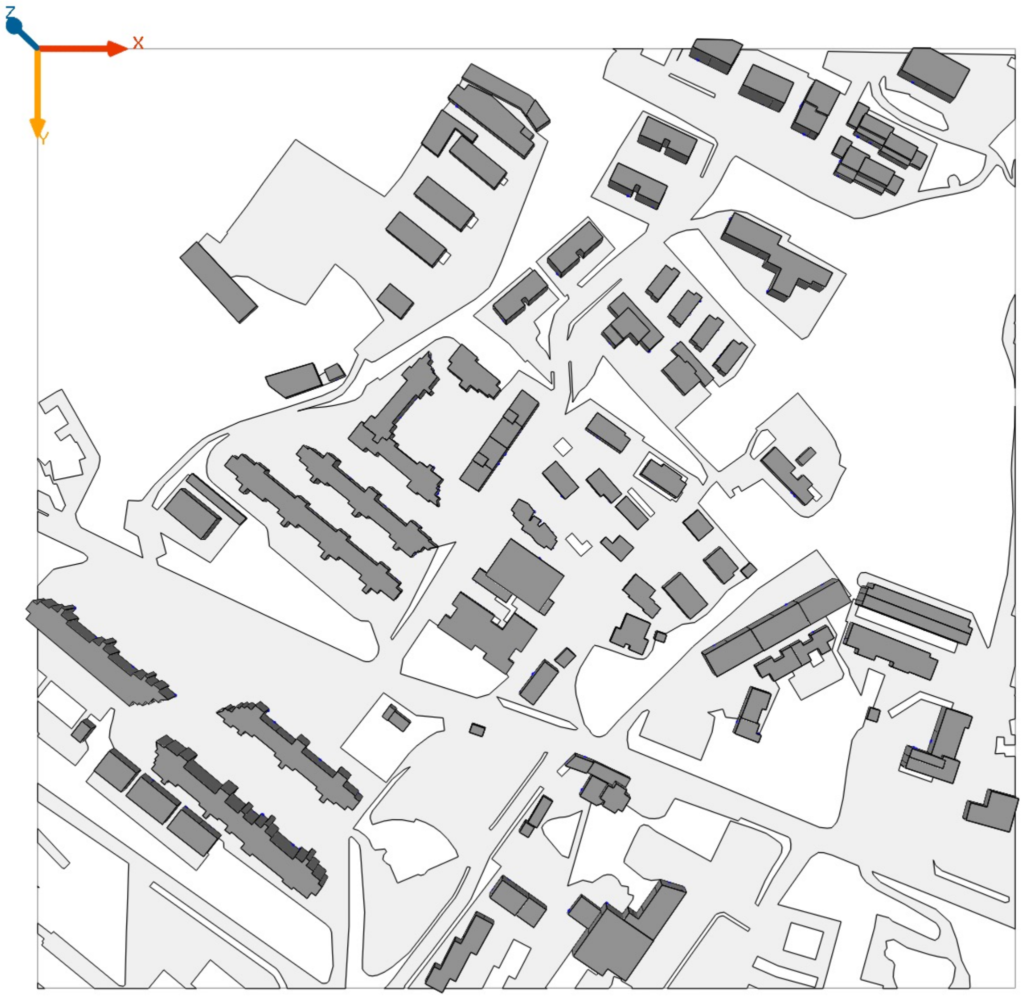
- Streets
- Triangulation
- Graph
- Buildings
- People
- Goal (safe area)



```
global {
    //Importa shapefile
    file shape_file_streets <- file("../includes/area
pedonabile.shp");
    file shape_file_buildings <-
shape_file("../includes/edifici prova.shp");
    file shape_file_arrive <-
shape_file("../includes/punti destinazione.shp");
    int abitanti;
    int volume;
    int nb_people <- abitanti;
    float people_size <-1.0;

    graph the_graph;

    geometry shape <- envelope(shape_file_streets);

    init {
        create object from: shape_file_streets ;
        object the_object <- first(object);

        //triangulation of the object to get the differ-
ent triangles of the polygons
        list<geometry> triangles <-
list(triangulate(the_object, 0.01));

        loop trig over: triangles {
            create triangle_obj {
                shape <- trig;
            }
        }

        //creation of a list of skeleton from the object
        list<geometry> skeletons <-
list(skeletonize(the_object, 0.01));

        //Split of the skeletons list according to their
intersection points
        list<geometry> skeletons_split  <-
split_lines(skeletons);
        loop sk over: skeletons_split {
            create skeleton {
                shape <- sk;
            }
        }

        //Creation of the graph using the edges result-
ing of the splitted skeleton
         the_graph <- as_edge_graph(skeleton);


        create goal  {
            location <- any_location_in
(one_of(skeleton));
        }
        create building from: shape_file_buildings
with:[height::float(get("CR359_UN_2")), inhabit-
ants::int(get("abitanti"))];
       loop b over: building {
        create people number:b.inhabitants{
            location <- any_location_in(b);
            target <- one_of (goal);
        }
       }
    }
}

species object  {
    aspect default {
        draw shape color: #lightgray border:#black;
    }
}

species triangle_obj  {
    rgb color <- rgb(150 +rnd(100),150 + rnd(100),150 +
rnd(100));
    aspect default {
        draw shape color: color ;
    }
}

species skeleton  {
    aspect default {
        draw shape + 0.2 color: #red ;
    }
}
species building {
   float height;
   int inhabitants;
   string type;
   rgb color <- type = "Edificio civile" ? #pink :
#gray;
   aspect default {
       draw shape depth: height color: color bor-
der:#black;
   }
}

species goal {
    aspect default {
        draw circle(3) color:#red;
    }
}

species people skills: [moving] {
    goal target;
    path my_path;

    reflex goto {
        do goto on:the_graph target:target speed:1.0;
    }
    aspect sphere3D {
        draw sphere(1) color: #blue;
    }
}

experiment prova_poligonalizzazione type: gui {
    output {
        display objects_display type: opengl {
            species object aspect: default ;
            species triangle_obj aspect: default ;
            species skeleton aspect: default ;
            species building aspect: default ;
            species people aspect: sphere3D ;
            species goal aspect: default ;
        }
    }
}
```

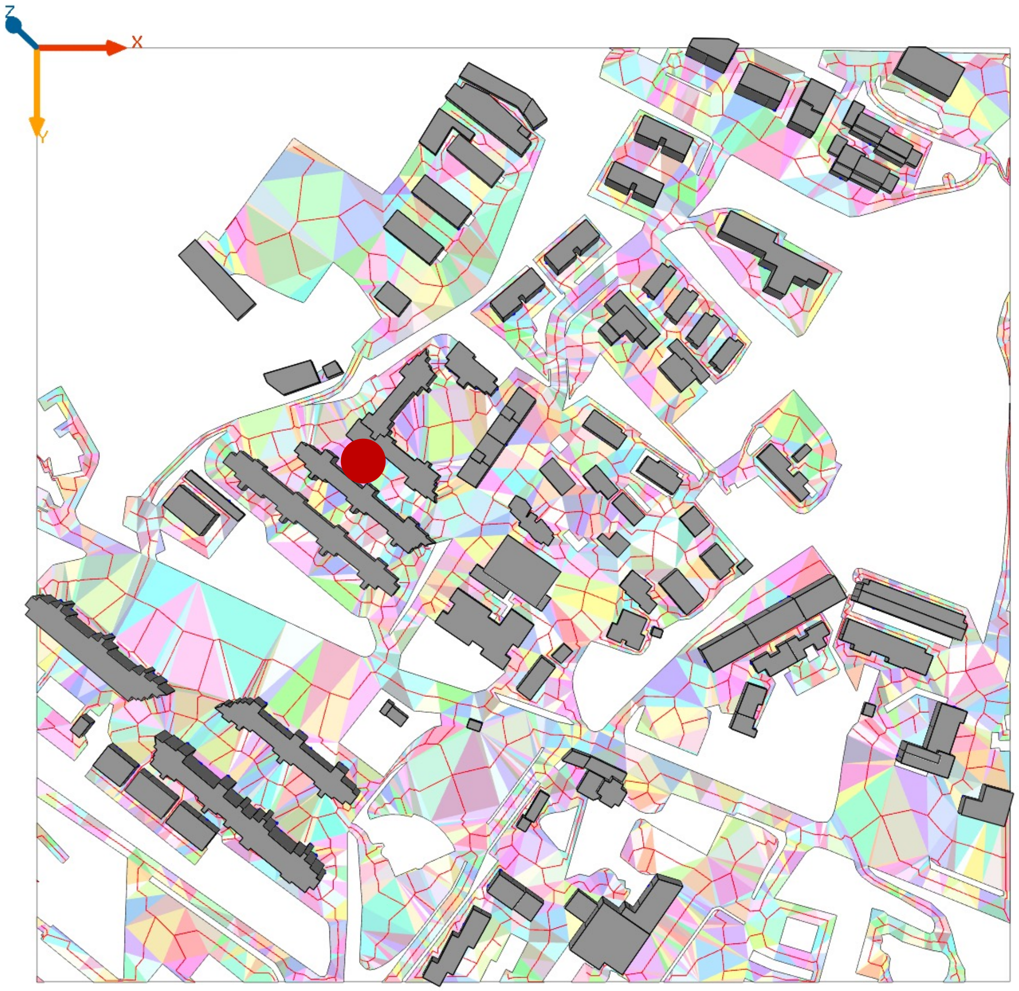
Results 1: Basic environment from geodata

The data useful for the development of the analysis are available on the open-data geoportal of the Abruzzo Region (http://opendata.regione.abruzzo.it/). In particular, the CTR available in the regional territorial database (DBTR) updated to 2007 (the most up-to-date official data available) was used, in which there is a shapefile containing geometries concerning the road system. Before using the shapefiles within GAMA, it is **necessary to prepare them in a GIS environment** (QGIS). To carry out crowd simulations, it is necessary to calculate the number of theoretical residents in each building. This operation was carried out by considering **1 inhabitant per 100 cubic meters** (law standards in Italy for residential buildings).



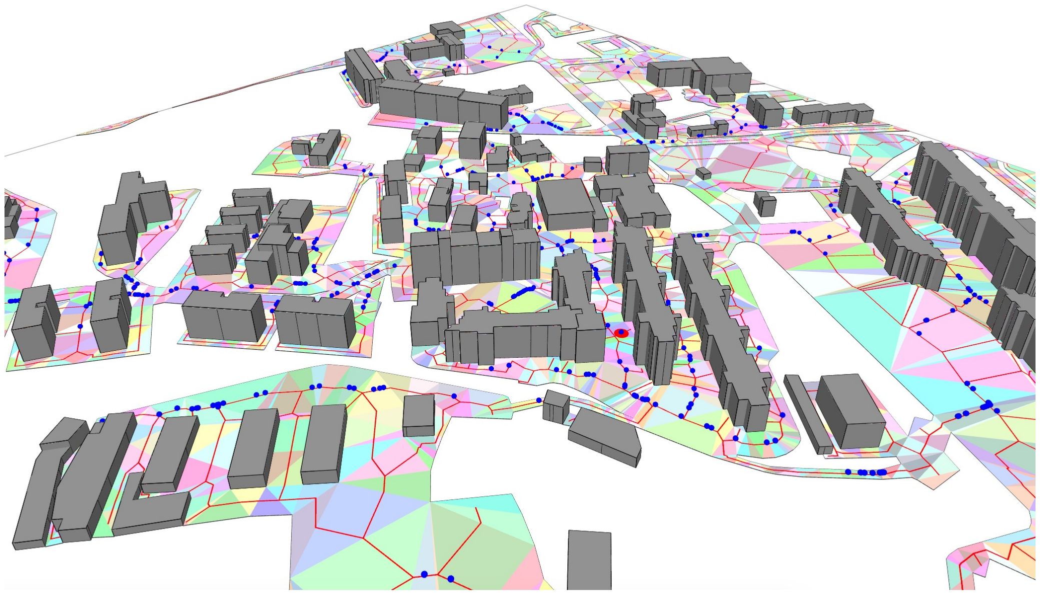
Results 2: From street layer to graph

To carry out a crowd simulation, it is necessary to have a **road graph** representing the flows that the agents representing the residents must follow to reach one or more specific safe points. Since these simulations are related to emergencies linked to natural disasters, the point that is considered indicates the location of a **hotspot where citizens can access and receive first aid**. To calculate the road graph from the polygonal geometries describing the road network, the latter were automatically triangulated in GAMA. **The red line** in figure represents the graph calculated in this way.

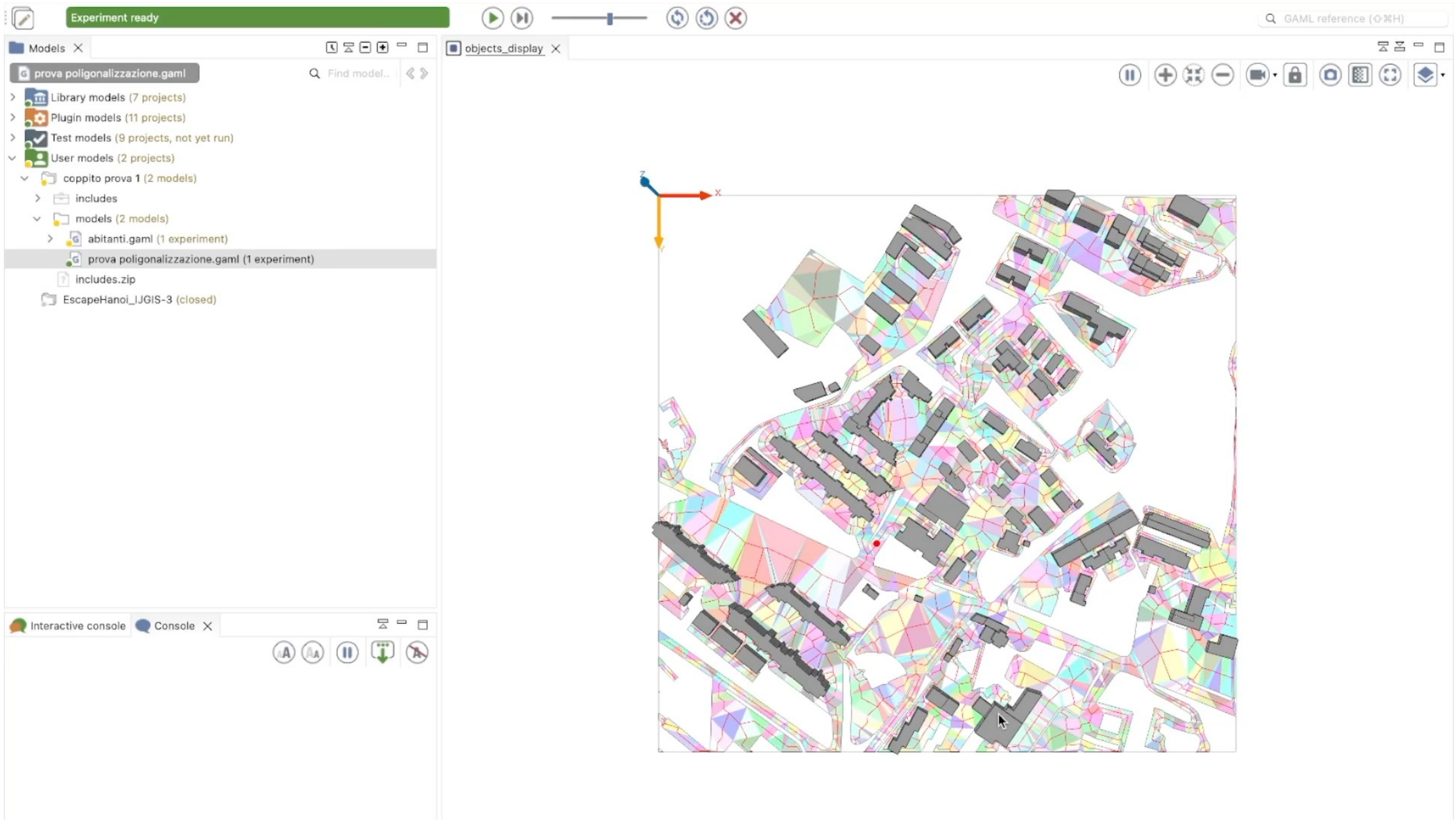


Results 3: Simulation in action

Agents representing the theoretical residents of each building are generated algorithmically. At the start of the simulation, the agents 'exit' the buildings and head towards the red point following the shortest path. Now, the agents, represented by the **small blue spheres** in figure, move one after the other on the red graph and converge towards the red point representing the hotspot.



Results 3: Simulation in action


Experiment ready
GAML reference
Models
prova poligonalizzazione.gaml
Find model...
Library models (7 projects)
Plugin models (11 projects)
Test models (9 projects, not yet run)
User models (2 projects)
coppito prova 1 (2 models)
includes
models (2 models)
abitanti.gaml (1 experiment)
prova poligonalizzazione.gaml (1 experiment)
includes.zip
EscapeHanoi_IJGIS-3 (closed)
Interactive console
Console
objects_display
z
x

Conclusions

Current limitations:

- Modelling of the behavioural aspects of agents are currently limited since only **one kind of individual** (the one that knows what to do in crisis situations) **is modeled**;
- The movement of people is currently aligned in a row so **chaos and panic situations are not taken into account**;
- Urban environment is flat so it does not take into account the **morphology** of the territory.

Future lines of research and implementation:

- Implementation of more kinds of **human-agents** relating to different types of users (with a focus on fragile people) – currently underway in collaboration with informatics and data science department of UnivAq;
- Construct **risk** (and multi-risk) **scenarios**;
- Implement urban design techniques that can follow along with simulations.

Thank you! ☺

Federico Eugeni, Sara Sacco, Donato Di Ludovico, Antinisca Di Marco

DICEAA - DISIM, Università degli Studi dell'Aquila

federico.eugeni@univaq.it
sara.sacco1@graduate.univaq.it
donato.diludovico@univaq.it
Antinisca.dimarco@univaq.it