ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

GABRIEL FONSECA SILVA

MULTI-LEVEL MODELING OF URBAN ENVIRONMENTS

Porto Alegre

2021

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF TECHNOLOGY
GRADUATE PROGRAM IN COMPUTER SCIENCE

MULTI-LEVEL MODELING OF URBAN ENVIRONMENTS

GABRIEL FONSECA SILVA

Dissertation presented as partial requirement for obtaining the degree of Master in Computer Science at Pontifical Catholic University of Rio Grande do Sul.

Advisor: Prof. Dr. Soraia Raupp Musse
Co-Advisor: Prof. Dr. Vinícius Jurinic Cassol

Porto Alegre
2021

Ficha Catalográfica

S586m Silva, Gabriel Fonseca

Multi-level modeling of urban environments / Gabriel Fonseca Silva. – 2021.
88.
Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientadora: Profa. Dra. Soraia Raupp Musse.
Co-orientador: Prof. Dr. Vinícius Jurinic Cassol.

1. Urban Modeling. 2. Multi-Level Modeling. 3. Urban Environments. 4. City Layouts. I. Musse, Soraia Raupp. II. Cassol, Vinícius Jurinic. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecária responsável: Clarissa Jesinska Selbach CRB-10/2051

Gabriel Fonseca Silva

Multi-level modeling of urban environments

This Master Thesis has been submitted in partial fulfillment of the requirements for the degree of Doctor/Master of Computer Science, of the Graduate Program in Computer Science, School of Technology of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on March 31, 2021.

COMMITTEE MEMBERS:

MODELAGEM MULTINÍVEL DE AMBIENTES URBANOS

RESUMO

A modelagem e visualização de cidades virtuais tem sido um ponto de interesse nas áreas de pesquisa de simulações e computação gráfica, também apresentando contribuições para a indústria de filmes e jogos digitais. Cidades apresentam comportamentos populacionais complexos, com densas redes de estradas e um grande número de edificações com arquiteturas variadas. Nos últimos anos, diversos métodos baseados em dados geográficos e técnicas procedurais foram apresentados, permitindo a criação de conteúdos em larga escala. Ainda assim, esta área de pesquisa apresenta desafios a serem explorados, incluindo a falta de padrões de comunicação e troca de dados entre os diferentes níveis da hierarquia de uma cidade. Este trabalho apresenta um *framework* para modelagem multinível de ambientes urbanos. Em cada estágio do *pipeline* deste *framework*, novos elementos e áreas de uma cidade são representados em maior detalhe. Como contribuição, modelamos uma cidade do mundo real em um ambiente virtual com base em informações geográficas seguindo o modelo apresentado, permitindo sua representação em diferentes níveis de abstração e detalhamento de dados relacionados à população e aos elementos do ambiente. Realizamos uma análise dos dados de isolamento social da cidade, observando padrões de movimentação populacional durante a pandemia do COVID-19. Por fim, modelamos rotinas populacionais com base nos dados analisados, permitindo a visualização de grupos populacionais e indivíduos no ambiente virtual.

Palavras Chave: Modelagem Urbana, Modelagem Multinível, Ambientes Urbanos, Layouts Urbanos.

MULTI-LEVEL MODELING OF URBAN ENVIRONMENTS

ABSTRACT

The modeling and visualization of virtual cities is a point of interest in the research fields of simulations and computer graphics, also presenting contributions for the movies and games industries. Cities present complex population behaviors, with dense road networks and a huge number of buildings with varying architecture. In recent years, many methods based on geographical data and procedural techniques were presented, allowing the creation of content on a large scale. Still, there are known challenges to be explored in this field of research, including the lack of standards of communication and exchange of data between different levels in the hierarchy of a city. This work presents a framework for multi-level modeling of urban environments. At each stage of the framework's pipeline, various elements and areas of a city are represented with increasing degrees of information. We modeled a real-world city based on geographical information following the presented framework, allowing its representation in different levels of abstraction and detail of data related to the population and environmental elements. We performed an analysis of social isolation data of the city, observing patterns in the population movement during the COVID-19 pandemic. Also, we modeled population routines based on the analyzed data, allowing the visualization of population groups and individuals in the environment.

Keywords: Urban Modeling, Multi-Level Modeling, Urban Environments, City Layouts.

LIST OF FIGURES

2.1 City layout created through a set of input maps, as presented by Parish and Müller [34]. A Water map is presented in (A). White indicates the presence of water, and black the absence. An Elevation map is presented in (B). White indicates a higher elevation. A Desired Population Density map is presented in (C). White indicated a higher desired density. The final result is presented in (D). 22

2.2 Influence of multiple parameters in the procedural generation of an urban layout, as presented by Mustafa et al. [32]. 23

2.3 Procedural generation of road networks based on pattern masks, as presented by Müller [34]. Areas with an enforced "rectangular raster" pattern are presented in (A). Areas with an enforced "radial" pattern are presented in (B). A possible outcome of the method is presented in (C). 24

2.4 Process of procedural generation of roads and parcels in arbitrary terrains, as presented by Emilien et al. [14]. The input terrain is presented in (a). The creation of roads connecting the exit points of the map and the definition of settlement seed is presented in (b). The parcel subdivision is presented in (c). 25

2.5 Patch-based generation of roads, as presented by Teng and Bidarra [43]. The upper half of the road network was created using a parametric method. The lower half was created using road patches to fill areas between main roads. 26

2.6 GIS-based method for generation road networks, as presented by Gang and Guangshun [17]. A representation of the input data for a simple intersection is presented in (a). The smoothing process for road segments connected to the intersection is presented in (b). The result generating a sidewalk in an intersection is presented in (c). 26

2.7 The result of 5 iterations of the procedural generation of a building, as presented by Parish e Müller [34]. 27

2.8 Extrusion-based method presented by Greuter et al. [20]. At each iteration, a random primitive geometry is added to the base of the current model, being extruded downwards until the floor is reached. 28

2.9 Facade splitting process, as presented by Wonka et al. [47]. The set of rules that compose a Shape Grammar is presented in (a). The obtained result when applying the rules to an initial facade is presented in (b). 28

2.10 Buildings and houses procedurally generated using CGA Shape, as presented by Müller et al. [31]. 29

2.11 Adjustment of feature elements on irregular terrains, as presented by Elimien et al. [14]. A facade element being moved after colliding with the terrain and its kernel cost is presented in (a). The obtained result in a set of buildings is presented in (b). 29

2.12 BioCrowds model, as presented by Bicho et al. [6]. A larger square represents a single virtual agent (person). Each agent tries to capture markers (dots) inside their personal space to be able to move towards their goal. 31

2.13 BioClouds model, as presented by Antonitsch et al. [3]. A scenario with two clouds of agents moving towards each other is presented in (a). A visualization of a microscopic crowd simulation occurring within the clouds is presented in (b). This level of detail is obtained only when required, i.e., for the sections of clouds within the viewport. . . 31

3.1 LODUS architecture overview. 34

3.2 Multi-level Graph's hierarchy. 36

3.3 Overview of the multi-level map pipeline. In (a), the boundaries of the environment are defined within a 2D plane. In (b), a graph between Regions of interest is defined. In (c), edges receive higher detail. In (d), a Region is divided into Subregions. In (e), a Road Network is established for each Subregion. In (f), City Blocks are defined. In (g), a City Block is subdivided into land lots (Parcels). In (h), buildings are created in each Parcel. 38

3.4 Region Outlines. The environment boundaries are presented in (a). The *region outlines* are presented in (b). 39

3.5 Connections between a group of *regions*. Two-way segments are highlighted by a green arrow. One-way segments are highlight by a cyan arrow. Intersections are highlighted by an orange circle. 40

3.6 Parcel subdivision method based on OBB recursive cuts. The result of the first three iterations is presented in (a), (b), and (c). OBBs and their mid-points are represented by the dotted orange lines. Generated parcels at each iteration are highlighted in purple. 42

3.7 Population description step. An example template for a population being modeled is presented in (a). The composition of a Population Group using this template is presented in (b). The group is divided into subgroup types, each one with its population quantity and distribution. 44

3.8 Node structure. 45

3.9 Tool for creating population routines. A group of *regions* in an example environment is presented in (a). The selected *region* is highlighted by a green circle. Related regions are highlighted in blue. The remaining regions are highlighted in gray. The existing events in a selected node are presented in (b). The editing windows for different are presented in (c) and (d). 46

3.10 Visualization of social isolation data through hexagons. The isolation index ranges from 0 to 1. Lower values of isolation are presented in red (< 0.4). Medium values are presented in orange and yellow. Higher values are presented in green (> 0.6). . . 49

4.1 Multi-level Graph hierarchy and data available for each *level of abstraction*. "Individual Visualization" indicates which *elements* can be navigated in the visualization component. This does not apply to *elements* grouped in brackets (e.g. "Other 7 Intermediate Regions" in the second *level of abstraction*). "Data Distribution" indicates the starting *level of abstraction* in which that particular data is available. For example, Population Count is available at the first level and is inherited by lower levels. 52

4.2 Spatial structures of the Multi-level Graph represented in the visualization component. Each structure is divided into its respective *regions*, with *outlines* in different colors. *Region outlines* of an *element* in a higher *level of abstraction* are presented in gray-scale. 55

4.3 *Connections* of the "City of Porto Alegre". *Connections* between a group of 4 *regions* are presented in (a), containing: Vila Conceição (green); Sétimo Céu (yellow); Pedra Redonda (pink); and Jardim Isabel (blue). A visualization of *connections* between multiple *regions* is presented in (b). The center point of each *region* is highlighted by red squares. 57

4.4 Geometry creation. *Region outlines* and *connections* of Porto Alegre are presented in (a). Road geometries are added in (b). Building geometries are added in (c). 59

4.5 Additional structures of the "City of Porto Alegre". Bus routes of the city are presented in (a). Bus stops in the city are presented in (b). 60

4.6 Default interface of the visualization component. 61

4.7 Data of a selected *region* being displayed. 62

4.8 Visualization of the population groups in the city. 65

4.9 Visualization of the population groups with added geometry. 65

4.10 Simulation environment. Cells are represented by white quadrilaterals. Markers are represented as black points. Spawn areas are represented by blue cubes. 66

4.11 Virtual agents moving in the multi-level map environment. Green lines represent the agent-to-marker vectors. Red lines represent the navigation path of each agent towards their goal. 67

4.12 Social Isolation Index, from March 1, 2020, to January 20, 2021. The social index of the five neighborhoods with the largest populations is presented in (a). The index of the three neighborhoods with the smallest populations is presented in (b). The average index between all neighborhoods of the city is presented in (c). 70

4.13 Color gradient scale for the isolation index. 71

4.14 Social Isolation in the city of Porto Alegre, divided by neighborhoods. The isolation index for Saturday, Sunday, and Monday in a typical week (before restrictions were introduced by the government) are presented in (a), (b), and (c). The same days of the week in the first week of restrictions are presented in (d), (e), and (f). The isolation index months after the initial restrictions are presented in (g), (h), and (i). . 72

CONTENTS

1 INTRODUCTION . . . 17
1.1 OBJECTIVES . . . 18
1.2 MAIN CONTRIBUTIONS . . . 18
1.3 ORGANIZATION OF THIS DOCUMENT . . . 18

2 RELATED WORK . . . 21
2.1 URBAN LAYOUTS . . . 21
2.2 ROADS . . . 24
2.3 BUILDINGS . . . 26
2.4 POPULATION . . . 30

3 PROPOSED MODEL . . . 33
3.1 LODUS OVERVIEW . . . 33
3.2 MULTI-LEVEL MAP . . . 35
3.2.1 PIPELINE METHODOLOGY . . . 37
3.3 POPULATION SETUP . . . 43
3.3.1 POPULATION DESCRIPTION . . . 43
3.3.2 POPULATION DISTRIBUTION . . . 43
3.3.3 POPULATION EVENTS . . . 44
3.3.4 POPULATION SETUP TOOL . . . 45
3.4 MULTI-LEVEL SIMULATION . . . 47
3.5 SOCIAL ISOLATION PLUGIN . . . 47

4 RESULTS . . . 51
4.1 MULTI-LEVEL GRAPH . . . 51
4.2 ENVIRONMENT STRUCTURES . . . 56
4.2.1 CONNECTIONS . . . 56
4.2.2 ROADS AND BUILDINGS . . . 58
4.2.3 ADDITIONAL STRUCTURES AND POINTS OF INTEREST . . . 60
4.3 INTERFACE . . . 61
4.4 POPULATION . . . 62
4.4.1 POPULATION GROUPS . . . 63
4.4.2 VIRTUAL AGENTS . . . 65

4.5 SOCIAL ISOLATION 68

5 FINAL CONSIDERATIONS 73

5.1 LIMITATIONS 74

5.2 FUTURE WORK 75

REFERENCES 77

APPENDIX A – Papers Written During the Master's Degree **81**

APPENDIX B – Population Setup Files **83**

1. INTRODUCTION

The modeling of virtual cities and urban layouts has been a point of interest for different research fields in computer science, especially for simulations and computer graphics. Regarding urban planning, Eid and Eldin [13] discuss that simulation systems can be useful for simulating urban areas and their population, helping with decision making regarding public transportation, placement of residential locations, and evaluating the overall flow of a city. Human behavior is also a topic of interest for city planners and city managers [21], providing population studies, for instance, the impact of urban mobility in people's life [38].

Although smart cities have access to a multitude of electronic sensors to collect information about their population, one challenge that urban simulation models present is the level of detail of data and scenarios to be simulated. Supposing that a new set of public transportation routes are being planned, a simulation can be modeled to evaluate the impact on mobility for each route and possible alternatives. However, the input data required to describe the environment and population to be affected may vary, for example, if the routes cover small sections of a single city or multiple cities at an inner-state level. In these scenarios, having access to a framework that allows the user to define the level of detail of data to be used in simulations can be useful to city designers, helping them design and maintain various aspects of a real city and increase the population quality of life.

Regarding computer graphics, large virtual worlds are created to enhance the viewing experience in movies and the video games industry. In games, especially, the player can navigate through the city's road network and interact with buildings and elements placed in each area. The modeling of a virtual city can also be used to simulate traffic of large metropolitan areas [19, 33], pedestrian movement in smaller sections [29], and population growth for new and expanding areas [5]. New regions and neighborhoods can also be planned based on terrain elevation and desired population distributions [34]. Through an urban layout, it is also possible to calculate average sun exposure temperatures for certain regions, the impact of weather and rain [18], and estimate water flow in flood-sensitive areas [32].

Even with constant advancements in this field of research, there are still known challenges to be explored. According to Cogo et al. [10], the integration of different procedural generation methods is still an open question due to a lack of standards of communication between each level in the hierarchy of a city. Billen et al. [8] describes the need for a larger variety of simulations and analyses models for virtual cities. The authors also mention the need for better techniques for visualizing spatial data obtained through simulations and/or during the modeling process. According to Biljecki et al. [7], many openly available datasets used to model virtual cities contain geometric errors, such as duplicate vertices and self-intersecting volumes.

This work aims to present a framework able to tackle some of these mentioned issues, in particular having the goal to manage the multi-level data, as is described in the following sections. The work presented in this document is associated with the development of the LODUS Framework, which stands for *"Level Of Detail on Urban Simulation"* and was initially presented by Silva et al. [40]. LODUS is a framework for multi-level modeling of urban environments, where each of its stages allows for the representation of a virtual city in different levels of abstraction.

1.1 Objectives

The main objective of this work is to present a framework able to represent a virtual urban environment and its population in increasing levels of detail. We propose a multi-level approach, where the environment can be subdivided into more granular elements at each level until a microscopic representation can be achieved, for example, cities being divided into neighborhoods and subsequently into city blocks.

To achieve our main objective, we defined the following specific objectives:

- Establish a subdivision model for virtual environments that allows the distribution and exchange of data through multiple levels;
- Define a representation of the elements used in each level, from the boundaries of a city to environment structures such as roads and buildings that can be populated;
- Reproduce a real-world city in a virtual environment using our proposed framework;
- Visualize and analyze spatial data of the population and environment elements.

1.2 Main Contributions

The main contribution of this work is a framework for establishing a multi-level representation of virtual environments. We model a real-world city using our proposed framework, being able to represent it in increasing detail of data related to the population and a diverse set of environmental elements, such as neighborhoods, road networks, city blocks, and buildings. Regarding the population, we model routines executed through a multi-level simulation, being able to visualize the movement of population groups and individuals in the environment. Finally, we performed an analysis of social isolation data, observing patterns in the population movement during the COVID-19 pandemic and using the collected data to compose additional behaviors to our simulations.

1.3 Organization of this Document

This document is divided in the following manner: Chapter 2 presents important related work which contributed to understanding different methods for the generation of city layouts, road networks, building geometries, and the description and simulation of populations; Chapter 3 presents the methodology used for defining the multi-level map of the LODUS Framework and representing and simulating the population of a city in different levels of abstraction. Chapter 4 presents the results achieved in the modeling of a city in multiple levels of detail using our proposed framework. Finally, Chapter 5 presents the final considerations and future work of this project.

2. RELATED WORK

This chapter presents some work related to the modeling of urban environments and their multiple components, including the generation of road networks, buildings, and the representation of a population. The following methods apply a great variety of techniques to achieve their goals, with most of them using either real-world data [15, 24], procedurally generated content [32, 34], or a combination of both [4]. Although methods found in the literature present different steps to create urban content, many models follow a structure based on the seminal work of Parish and Müller [34]. A simplified pipeline to create a virtual city can be imagined as the following steps:

- Definition of the overall terrain of the environment, including terrain elevation and bodies of water that would possibly impede the placement of other elements;
- Definition of different sections of a city (e.g. industrial or residential areas) based on population descriptions;
- Creation of a road network, commonly in a hierarchical structure of primary (e.g. highways), secondary (e.g. arterials), and third-level roads (e.g. local access);
- Subdivision of city blocks into land parcels, with different sections of a city having a certain number of parcels assigned to parks, houses, and buildings;
- Creation of houses and building geometries for each parcel based on the section of the city that the parcel is located.

This chapter is divided into the following sections: Section 2.1 presents models related to the creation and modeling of urban layouts; Section 2.2 presents models related to the creation of road networks and road geometries; Section 2.3 presents models related to the creation of geometries and facades for buildings and houses; Section 2.4 presents models related to crowd simulations and population descriptions. Publications that present contributions for more than one of the sections described above will be referenced multiple times according to their proposed techniques and methods for that topic.

2.1 Urban Layouts

Parish and Müller [34] presented the first model for procedurally generating many elements of a virtual city, including streets and parcels. The authors proposed the use of extensions of L-Systems [25] to search for local optimal successors for each rule applied. The ideal successor is determined based on a set of global goals and local constraints, both defined by a set of texture maps and parameters given by the user. Global goals are established by a population density map or enforced by a road pattern for each section of the city. Local constraints represent terrain limitation, e.g., the presence of water or a terrain inclination higher than a defined threshold. Figure 2.1

presents a city layout procedurally generated using a set of texture maps including the presence or absence of water in (A); a heightmap in (B); and the desired population density in (C). The final result is presented in (D).

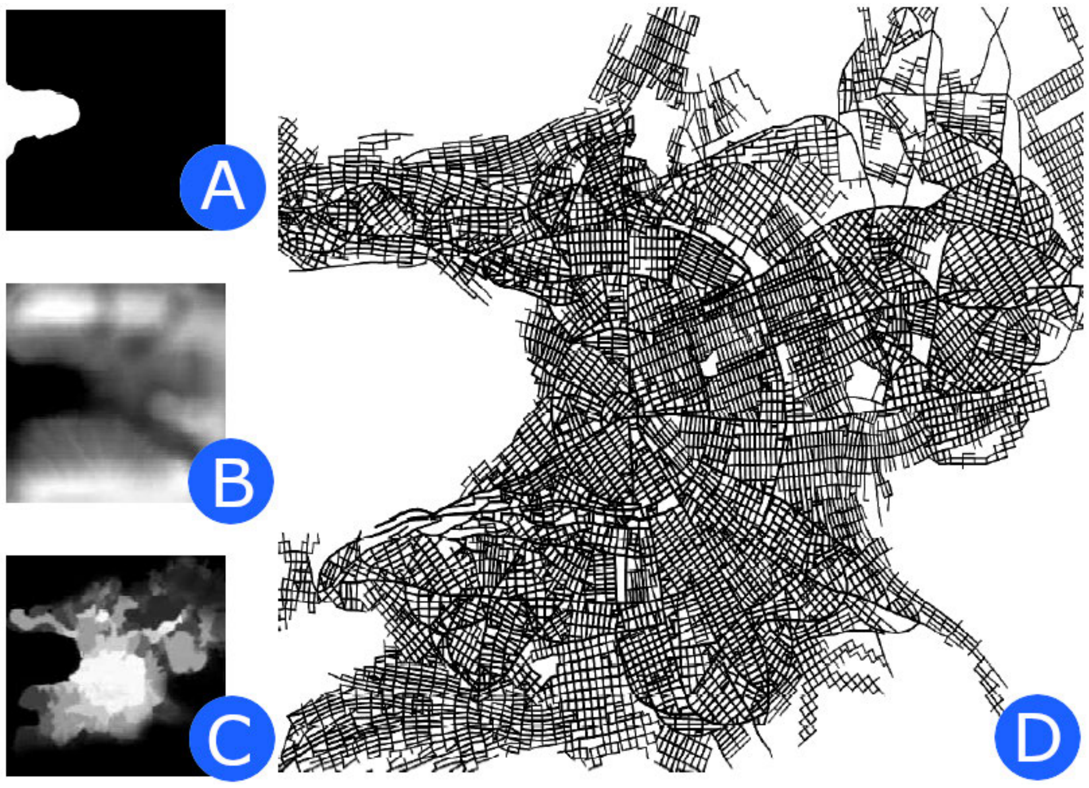


Figure 2.1: City layout created through a set of input maps, as presented by Parish and Müller [34]. A Water map is presented in (A). White indicates the presence of water, and black the absence. An Elevation map is presented in (B). White indicates a higher elevation. A Desired Population Density map is presented in (C). White indicated a higher desired density. The final result is presented in (D).

Talton et al. [42] presented an extension of this model to allow the parameterization of L-System rules and terminal symbols. Through these new additions and a set of user specifications, the authors use Bayesian inference modeling to define: a base geometric model; a derivation space for this model; and a probability function for fitness of the user specifications within the branch space. The model uses a Markov chain Monte Carlo (MCMC) method to approximate the ideal values for each parameter to create the desired environment.

Due to the limiting nature of L-Systems when modeling non-organic elements, different models have been presented to allow higher artistic control. Vanegas et al. [45] presented a model for combining behavioral modeling (e.g. population density distribution, job distribution, high access, land usage) and geometric modeling (e.g. road width and number of lanes, building area, number of floors), allowing the changes made in a certain parameter to have an impact on other parameters and, possibly, every element of the city layout. The authors also presented a method to estimate the accessibility and land value of sections of the city based on terrain slope, connectivity to roads, population, and nearby bodies of water. Therefore, if a change in accessibility is made (e.g. placement of a highway), nearby sections are also adjusted, altering the position of buildings and available jobs.

Vanegas et al. [46] presented an inverse design model where, given an initial procedurally generated city, adjustments are made using an MCMC method. These adjustments are applied to the procedural generation method, aiming to approximate a set of parameters (e.g. parcel average area, road curvature, building height) defined by the user. This allows for urban indicators (e.g. sun exposure or distance to parks) to be used as local or global goals, modifying the generation parameters of elements in the layout.

Garcia-Dorado et al. [18] presented a model that allows the user to draw the distribution of land usage (e.g. urban regions, agricultural regions, bodies of water) on the base terrain to create weather simulations. Roads and buildings for urban regions are adjusted using a Metropolis-Hastings algorithm [22, 30], an MCMC method, to explore the search space and find similar results to the user-specified weather. Specifications can include cloud coverage per region, humidity, rain distribution, and city temperature.

Mustafa et al. [32] presented a model for the generation of urban layouts that passively reduce water depth during flood scenarios. This work combines a procedural generation model with a hydraulic model to evaluate water flow characteristics for the created region. A neural network is used to identify relationships between the urban layout and the flow of water during floods, which are used as input to an MCMC model, adjusting the environment. The evaluation takes into consideration, e.g., the desired building coverage and average water depth. Figure 2.2 presents the influence of several parameters for the creation of an urban layout.

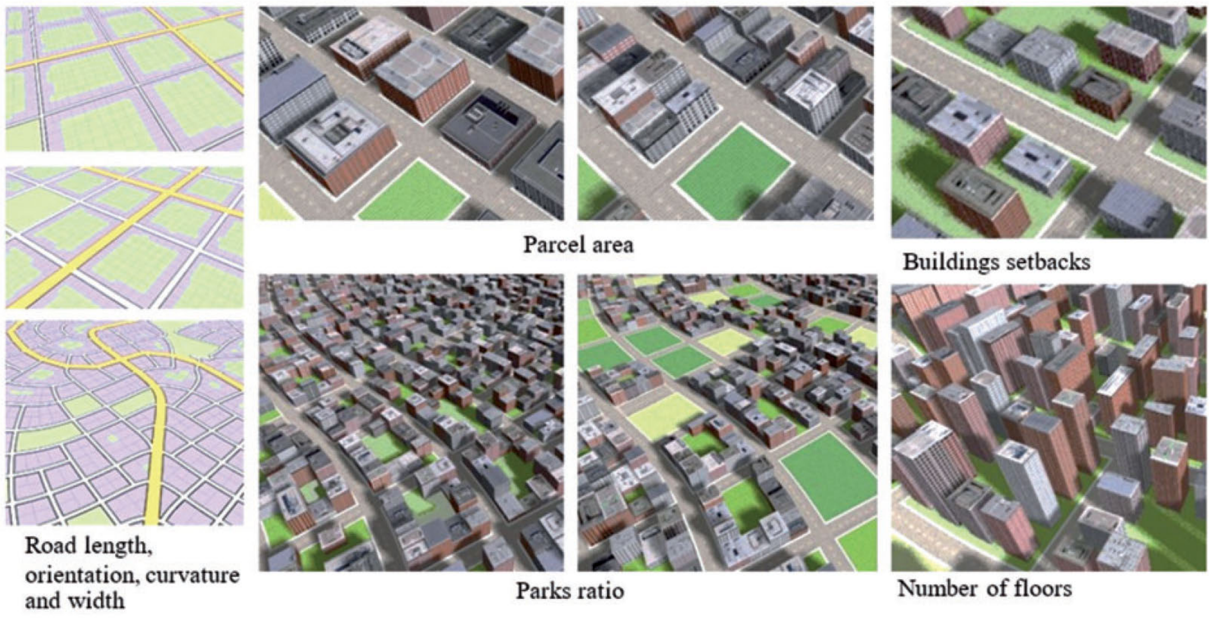


Figure 2.2: Influence of multiple parameters in the procedural generation of an urban layout, as presented by Mustafa et al. [32].

2.2 Roads

In their seminal work, Parish e Müller [34] used an L-System extension to create road networks. The ideal successor for new road segments is defined based on nearby population density peaks and a road pattern mask. The proposed patterns include a "rectangular raster" (based on New York City), a "radial to center" (based on Paris), and an "elevation min or max" (based on San Francisco). A road network can be formed from a combination of patterns for each region, with smaller streets having lesser fidelity due to L-Systems limitations. Figure 2.3 presents the combination of two pattern masks being applied to produce a road network.

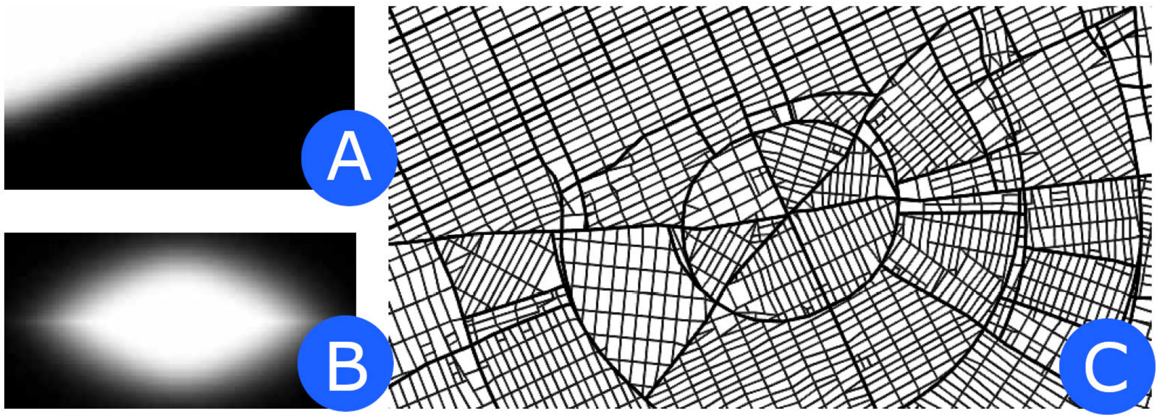


Figure 2.3: Procedural generation of road networks based on pattern masks, as presented by Müller [34]. Areas with an enforced "rectangular raster" pattern are presented in (A). Areas with an enforced "radial" pattern are presented in (B). A possible outcome of the method is presented in (C).

Dou et al. [12] proposed a hierarchical model for generating a road network based on templates for different areas of the environment. The model uses an image map or a free-hand drawing as input to establish primary roads (e.g. highways). These roads split the environment into areas, which receive a road template (similar to road patterns [34]) to generate secondary roads (e.g. arterial roads and avenues). City blocks contained within the network of secondary roads are subdivided into land parcels, and third-level roads are generated to provide local access. The authors also present a technique to adjust the positioning of secondary roads near boundaries, either by forcing an intersection or removing a group of road segments.

The work of Yang et al. [48] presented a model of hierarchical domain splitting. In this method, the user defines main roads within an input region, with secondary roads being created using either the streamlines of the main roads' crossfield or a template-based split. Both techniques are evaluated through quality metrics, such as land use, number of city blocks, and desired average parcel area. Fiser et al. [16] proposed a model utilizing Geometric Graph Grammars to generate road networks. The rule set can identify the street orientation and possible intersections, with adjustments being made after each iteration. The authors also present a technique to analyze and encode the Grammar Graph, allowing the model to learn and alter its ruleset.

Emilien et al. [14] presented a model for procedural road generation on arbitrary terrains, aiming the identify suitable locations for settlements. Given an input terrain, main roads are created aiming to connect all the "exit points" of the map. Settlement seeds are placed in areas with regular terrain elevation, nearby bodies of water, and no obstructing vegetation. Secondary roads are created to connect the seeds with primary roads. A skeleton of the village is defined by creating a set of building seeds that are connected to nearby roads. The area of these seeds is expanded, forming land parcels for future placement of buildings. Figure 2.4 presents the process of road and parcel generation.

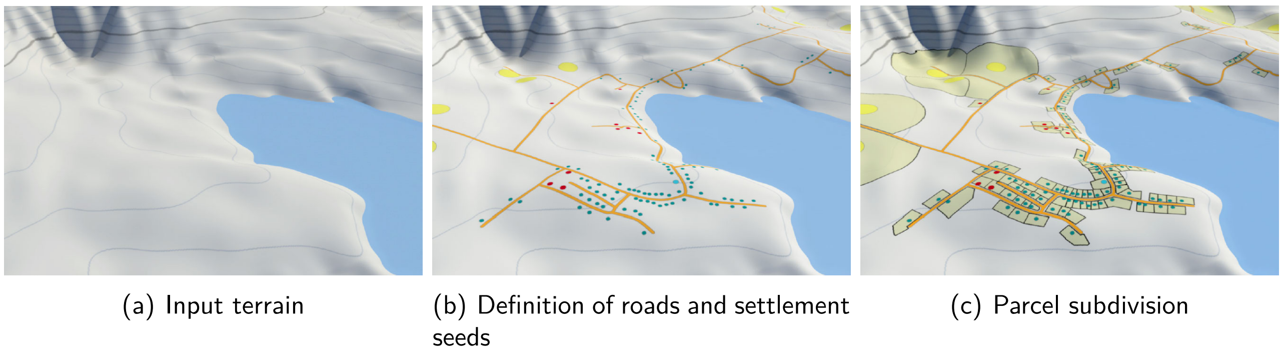

(a) Input terrain (b) Definition of roads and settlement seeds (c) Parcel subdivision

Figure 2.4: Process of procedural generation of roads and parcels in arbitrary terrains, as presented by Emilien et al. [14]. The input terrain is presented in (a). The creation of roads connecting the exit points of the map and the definition of settlement seed is presented in (b). The parcel subdivision is presented in (c).

Beneš et al. [5] presents a similar process, placing seed points near intersections of main roads and growing secondary roads based on traffic simulation. Chen et al. [9] uses tensor fields to create a street network. The user can modify the results by editing the tensor field or forcing the creation of main roads. Mustafa et al. [32] extends this model to automatically generate connection constraints with a set of parameters later controlled by an MCMC method, as described in Section 2.1.

Teng and Bidarra [43] presented a patch-based method for procedurally generating roads. This method aims to increase the user's creative control over the procedural content by using semantic patches associated with real-world road features, such as intersections, roundabouts, and curved segments. The model starts with main roads being given as input or being generated through a parametric method. Areas split by these roads are then filled with valid patches. Finally, when needed, third-level roads are generated to provide local access using the parametric method. Figure 2.5 presents two regions split by a body of water. The upper half was created entirely using the parametric method. The lower half was created with the use of road patches.

Several models in the literature utilize geographic information as input data. New networks are created to reproduce the input, identify patterns, and allow user adjustments. Gang and Guangshun [17] proposed a model to obtain road points and segments from a Geographic Information Systems (GIS) dataset, aerial and satellite images, and manual markings. A centerline for each road segment is identified and represented by a bezier curve. Different road segments can be connected to establish intersections, expanded sideways to form sidewalks, or combined to form sidewalks within intersections. All the elements are then united to establish a road network. This process can be observed in Figure 2.6. Aliaga et al. [1] presented a system for the reconfiguration of urban layouts. The base network is obtained via GIS. Regions and neighborhoods can be selected and edited by the user. This model also allows the user to distort and rearrange regions, with the results being adjusted according to previously established local constraints.

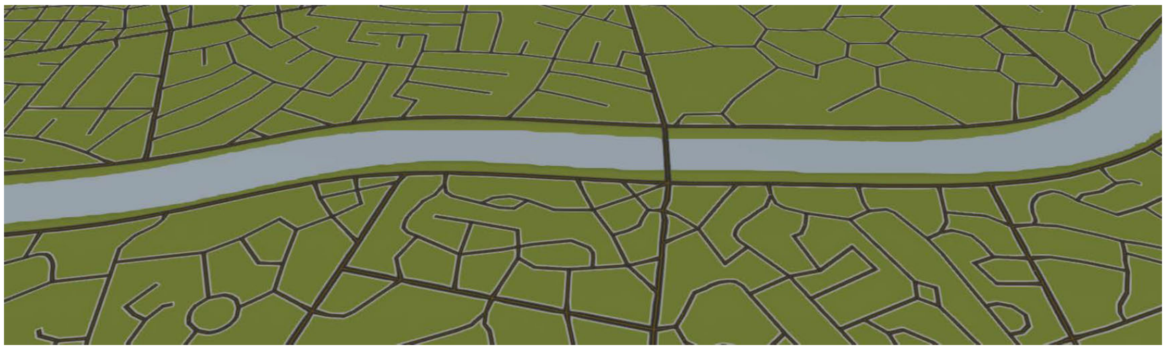
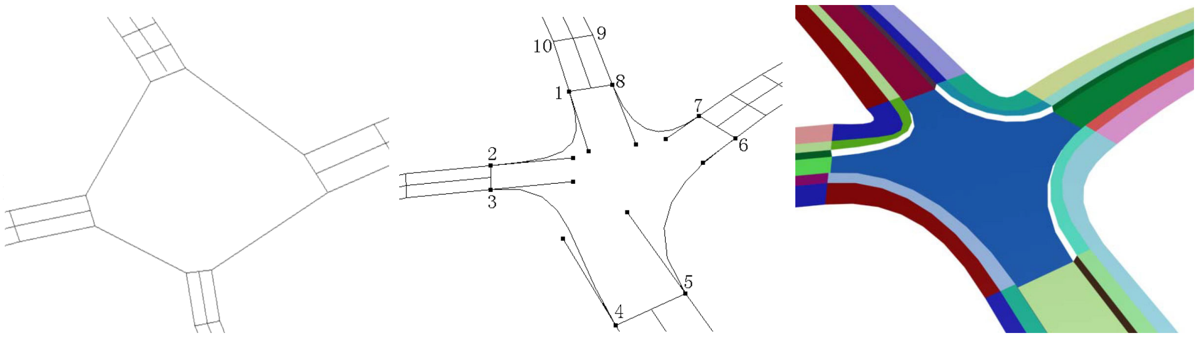


Figure 2.5: Patch-based generation of roads, as presented by Teng and Bidarra [43]. The upper half of the road network was created using a parametric method. The lower half was created using road patches to fill areas between main roads.



(a) Input data (b) Smoothing process for intersections (c) Generation of sidewalks

Figure 2.6: GIS-based method for generation road networks, as presented by Gang and Guangshun [17]. A representation of the input data for a simple intersection is presented in (a). The smoothing process for road segments connected to the intersection is presented in (b). The result generating a sidewalk in an intersection is presented in (c).

2.3 Buildings

Parish e Müller [34] also used L-Systems [25] to procedurally generate geometries for buildings. A geometric primitive is considered the initial axiom of the system, becoming more complex at each iteration. Because a rule is capable of generating new axioms, new iterations can introduce more complex partitions in the generated model. Figure 2.7 presents the result of multiple iterations using the method.

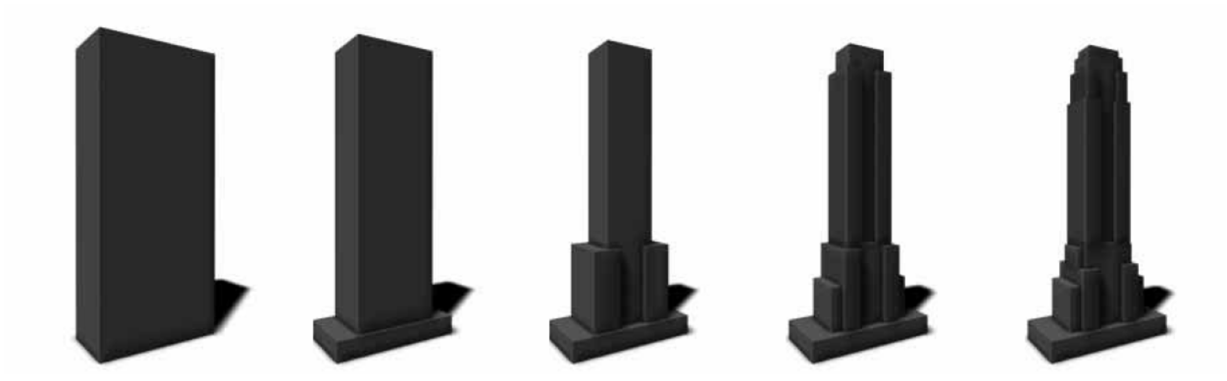


Figure 2.7: The result of 5 iterations of the procedural generation of a building, as presented by Parish e Müller [34].

Marvie et al. [28] proposed an extension for the use of L-systems named FL-system. In this model, the terminal symbols of the system are replaced by parameterized functions that can be activated at any time of their execution. This modification allows additional geometries to be created during the iteration process. Although buildings have different sets of partitions, such as floors and windows, they do not have a natural growth factor planned by L-systems. For this reason, several models based on grammars and parameterized blocks (i.e. primitives) have been proposed for the generation of these constructions.

Bekins and Aliaga [4] subdivide image-based buildings in features, enabling the creation of new geometries using these presets. The model also allows the filling of occluded areas in the image, with a shading equalization being applied based on similar subdivided features. Fan and Wonka [15] presented a model to represent exteriors of residential buildings through a set of attributes. The authors also proposed a model to encode relationships between attributes that are used to create a 3D representation of the buildings. Rodrigues et al. [37] proposed a model to extract information from textual descriptions, aiming to identify the optimal primitives to compose a geometry. Li et al. [24] uses geographic information to define building outlines. This process is followed by extrusion of the base lot and facade splitting.

Greuter et al. [20] presented an extrusion-based method that uses a set of 2D primitive geometries to compose buildings. A random primitive is selected as the top of the building. New primitives are placed around an existing vertex and extruded down. The process is repeated until the model reaches a defined height. Figure 2.8 presented the result of each iteration of this process.

More recent publications in the literature use a set of methods based on Shape Grammars, initially introduced by Stiny and Gips [41]. Shape Grammars allow for a more specialized set of rules, where labels can be used to filter the shapes affected by each operation, and axioms are represented by geometric shapes. Wonka et al. [47] presented a method called Split Grammars, to allow recursive axis-oriented cuts on the facades of buildings, creating new shapes (e.g. doors, windows, balconies, cornices). These shapes can receive different labels and be affected by new operations, like extrusions and rotations. Figure 2.9 presents a set of shape rules of the grammar in (a) and the obtained result when applying the rules to an initial facade in (b). Martinovik and Val

Gool [27] subdivide an image-based face in irregular lattices. Then, an MCMC method is used to identify the optimal grammar rule set to compose that facade. Zmugg et al. [49] uses Split Grammars to allow for geometry deformation, adjusting the cut parameters according to user interaction.

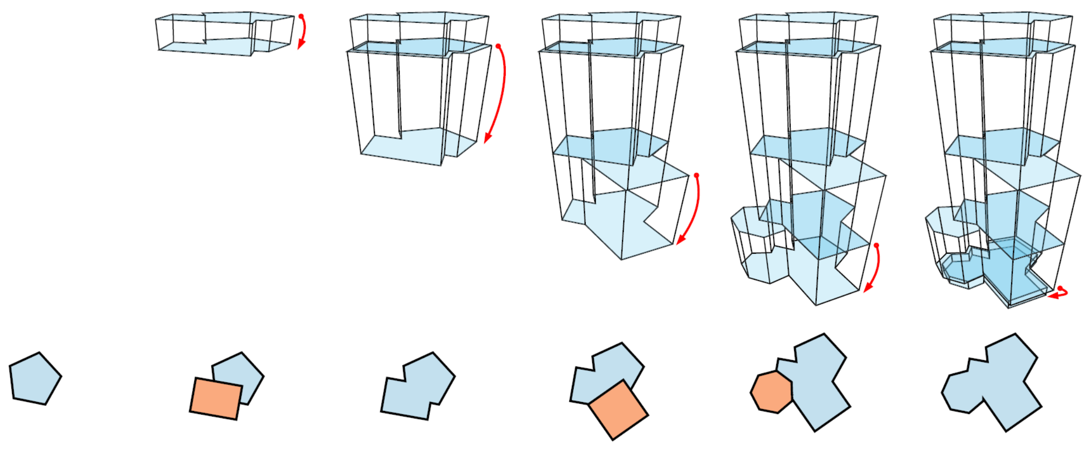
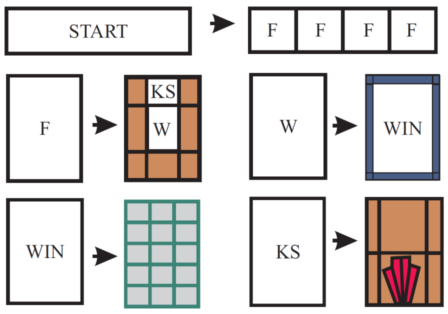
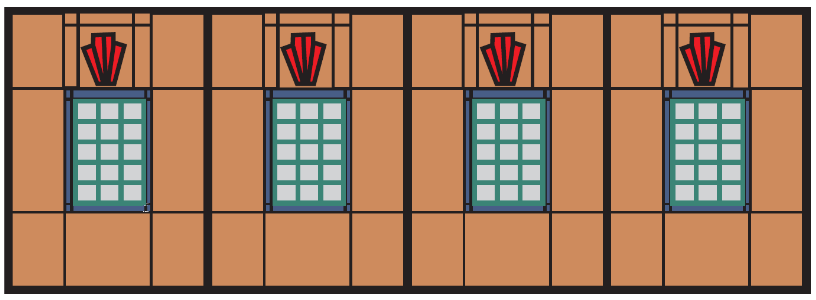


Figure 2.8: Extrusion-based method presented by Greuter et al. [20]. At each iteration, a random primitive geometry is added to the base of the current model, being extruded downwards until the floor is reached.



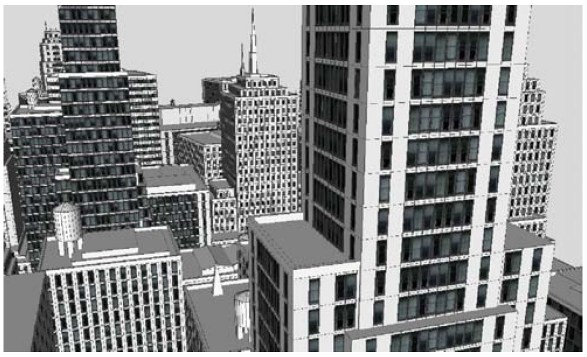
(a) Shape Rules



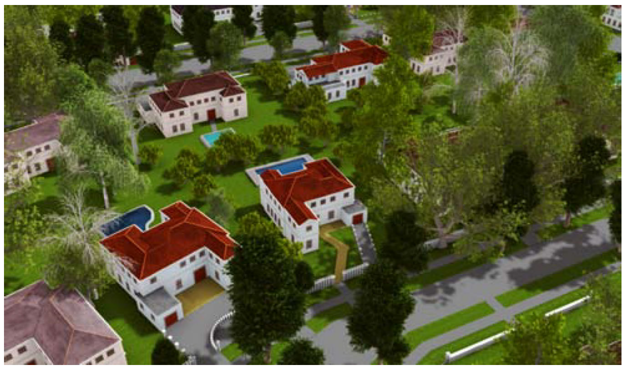
(b) Obtained result

Figure 2.9: Facade splitting process, as presented by Wonka et al. [47]. The set of rules that compose a Shape Grammar is presented in (a). The obtained result when applying the rules to an initial facade is presented in (b).

Müller et al. [31] presented the Computer Generated Architecture (CGA) shape grammar, where the modeling process is divided into two stages: mass modeling and facade modeling. In the first stage, a set of rules is applied to a different set of primitives, including 3D operations such as translation, rotation, scale, and extrusion. After the execution of the last rule, the 3D mesh is converted to a set of labeled 2D polygons. In this second step, a new set of rules is applied based on the label of each polygon, creating the building facade. Figure 2.10 presents buildings generated through CGA Shape. Schwarz and Müller [39] extend CGA Shape to allow boolean operations between shapes and the definition of instance variables. Demir and Koramaz [11] combine CGA Shape and geographic information to model buildings following city planning regulations. Liu et al. [26] combines CGA Shape and facade image segmentation to create geometries.



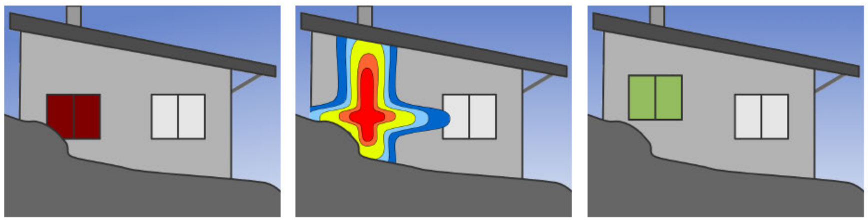
(a) Urban environment



(b) Suburban environment

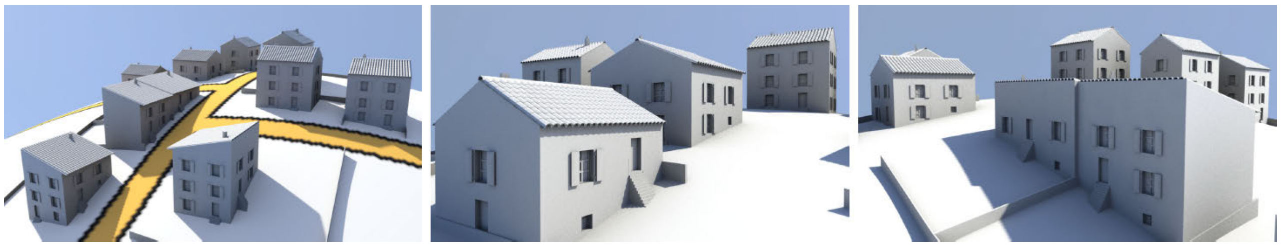
Figure 2.10: Buildings and houses procedurally generated using CGA Shape, as presented by Müller et al. [31].

Elimien et al. [14] presented a model called Open Shape Grammars to account for possible irregularities and inclinations in the terrain during the generation process of settlements. This variation allows both the position and the shape of new facade elements to be adjusted to achieve certain definitions. The result of a grammar rule can also be canceled if an external condition, such as a collision with the ground, is not reached within several attempts defined by the user. When a new facade element is created and its initial position collides with an external element, its position is adjusted based on a displacement cost kernel. Figure 2.11 presents the adjustment of facade features, an example of kernel cost, and the results achieved using an irregular terrain.



$$\mathcal{K} = \begin{pmatrix} 2.0 & 0.1 & 2.0 \\ 0.5 & \infty & 0.5 \\ 2.0 & 0.1 & 2.0 \end{pmatrix}$$

(a) Feature adjustment and kernel cost



(b) Obtained result

Figure 2.11: Adjustment of feature elements on irregular terrains, as presented by Elimien et al. [14]. A facade element being moved after colliding with the terrain and its kernel cost is presented in (a). The obtained result in a set of buildings is presented in (b).

2.4 Population

The modeling of a virtual city can go beyond the generation of geometries of its elements. One possible way to better represent a city is to establish a definition of the population that lives in it. In the work of Parish and Müller [34], the layout generation is influenced by a mapping of desired population densities. Higher densities tend to establish large groups of apartment buildings with a high number of floors. Low-density regions can be associated with suburban areas with a larger average parcel area. Vanegas et al. [45] explores a better representation of a population distribution with the inclusion of land value and job availability. These new parameters, when combined with the population density, allow the representation of industrial areas, for example, where there are a higher job availability and lower density, due to workers usually residing in different neighborhoods. Garcia-Dorado et al. [18] presents a similar concept to represent different sections of a city. The user has access to an interactive tool to paint the distribution of land, with residential, industrial, and commercial areas of low and high densities, alongside environmental descriptions such as deserts, mountains, and bodies of water.

Although these methods use descriptions of the population to establish sections of a city with particular geometric properties, it is possible to represent a population by simulating its movement and dynamics from microscopic and macroscopic points of view. Crowd simulation techniques are commonly used to represent the movement of a population in various levels of detail, both in interior and exterior environments. Models were presented for microscopic points of view [35, 36], macroscopic points of view [23, 44], and a combination of both [3].

Bicho et al. [6] presented a model for crowd simulations called BioCrowds, based on a space colonization algorithm designed to generate leaf venation patterns. In this model, a discrete space is populated by a set of markers. Virtual agents compete for these markers, effectively competing for the space in which they occupy and move. Agents have a marker capture range, representing their personal space, where all markers within this area are attempted to be captured. Because each marker can only be captured by one agent, it is possible to form both a Convex Hull and a Voronoi diagram to represent an area occupied by each agent. This also ensures that there are no collisions between agents, as they never occupy the same position in space as they move. Figure 2.12 presents an example of the BioCrowds model, where five agents compete for space.

Due to neighborhoods and cities containing large populations, simulations can be very computationally demanding for microscopic techniques. Macroscopic simulation methods allow for a higher granularity and smaller simulation cost with a trade-off of losing individual characteristics and behaviors for each agent. Antonitsch et al. [3] presented a model called BioClouds, based on the work of Bicho et al. [6]. BioClouds models crowds as a set of clouds, where each cloud is composed of a group of agents with a similar desire for a certain density, speed, and goal. Clouds compete for space amongst each other and occupy the environment in a manner that tries to keep their desired densities respected. Since a cloud is composed of a group of BioCrowds agents, it is possible to transition from a macroscopic simulation to a microscopic simulation when a certain level of detail is required. Figure 2.13 presents a simulation of the BioClouds model where it is possible to obverse both macroscopic and microscopic simulations.

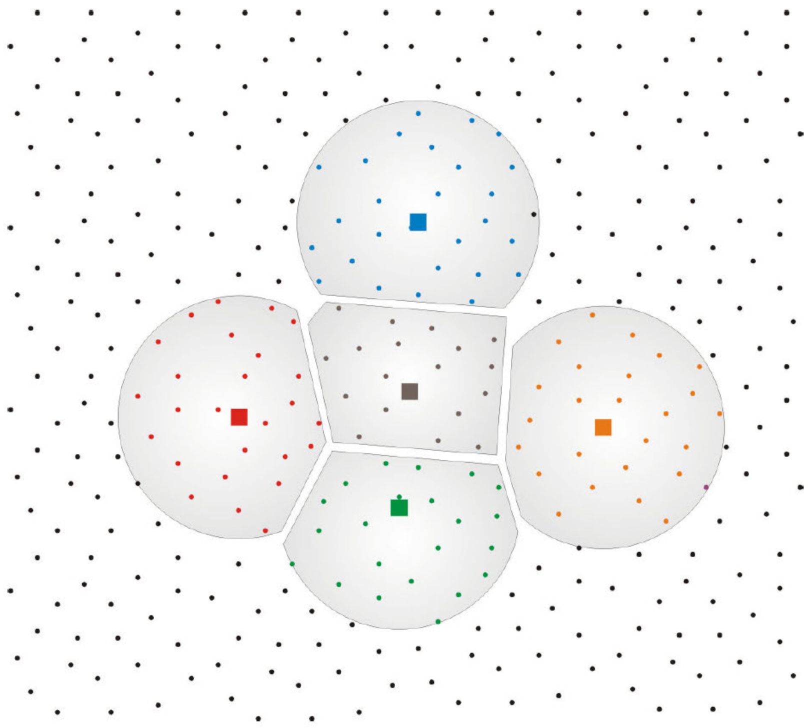
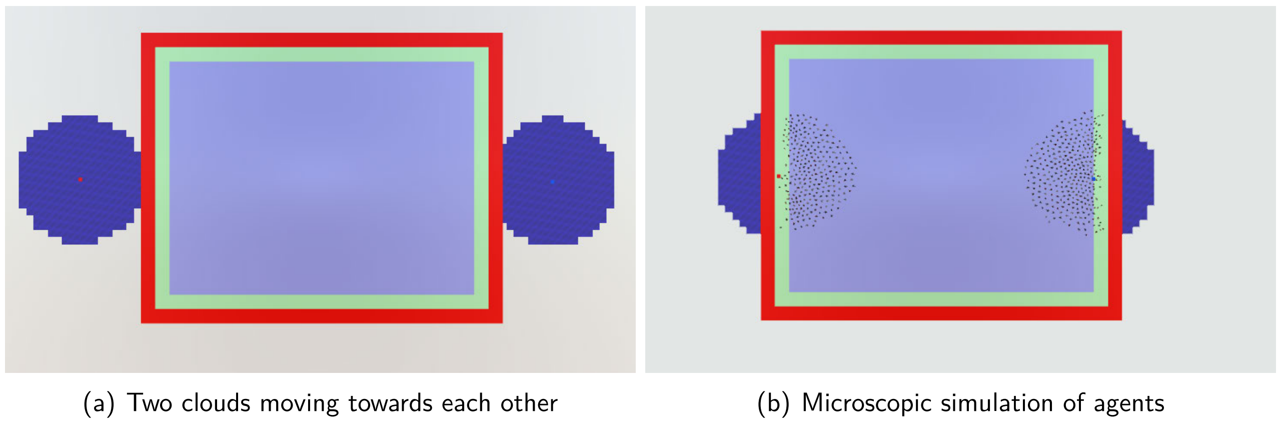


Figure 2.12: BioCrowds model, as presented by Bicho et al. [6]. A larger square represents a single virtual agent (person). Each agent tries to capture markers (dots) inside their personal space to be able to move towards their goal.



(a) Two clouds moving towards each other

(b) Microscopic simulation of agents

Figure 2.13: BioClouds model, as presented by Antonitsch et al. [3]. A scenario with two clouds of agents moving towards each other is presented in (a). A visualization of a microscopic crowd simulation occurring within the clouds is presented in (b). This level of detail is obtained only when required, i.e., for the sections of clouds within the viewport.

This chapter presented a variety of techniques to model and represent elements in a virtual city, from the overall layout down to individual buildings and streets. The next chapter presents our proposed methodology for modeling an urban environment through a multi-level map. Also, we propose a method to represent the population of a city in different levels of abstraction.

3. PROPOSED MODEL

This work is part of the LODUS Framework, which stands for *"Level Of Detail on Urban Simulation"* and was initially presented by Silva et al. [40]. LODUS is a framework for multi-level modeling of urban environments, where each of its stages allows for the representation of a virtual city in different levels of abstraction. The framework presents information in multi-levels of detail, so managers and urban planners can visualize spatial information of a city at the desired level to help their decision-making. LODUS is also able to simulate the movement and routines of the population through population dynamics models [1]. This document focuses on the description of the methodology responsible for the structured part of LODUS, i.e., the generation of the multi-level map, and the description of a population and its movement within the environment. Regarding the multi-level map, this thesis establishes a representation of the elements used at each stage, from a higher level of abstraction of a city to the definition of geometries for elements such as roads and buildings that can be populated.

This chapter is divided in the following manner: Section 3.1 presents an overview of the LODUS Framework [40] and its architecture; Section 3.2 presents our proposed pipeline for constructing a multi-level map; Section 3.3 presents our methodology for describing a population within LODUS; Section 3.4 presents an overview of the multi-level simulation;

3.1 LODUS Overview

LODUS [40] is a framework able to model an urban environment in different levels of abstraction, where each level presents a more detailed version of the environment (e.g. neighborhoods, roads, buildings) and its population. Through a multi-level simulation of the population movement in the environment, a variety of data can be obtained and presented, such as the current location of population groups at a certain time, occupancy of points of interest during the day, and the tracking of infected individuals/groups during contagion simulations. LODUS main goal is to provide a tool able to assist city managers and urban planners in their decision-making based on the simulation data and the visualization of spatial data. The multi-level aspect of the framework allows the user to evaluate the obtained data in different levels of abstraction, from a macro perspective (i.e. visualizing a country or groups of cities), all the way through a micro perspective (i.e. visualizing small areas such as a group of city blocks or individual buildings).

Figure 3.1 presents an overview of the LODUS architecture, with the highlighted modules being the main contributions of the authors of this document. These modules are described in higher detail in the following sections. The first module of the framework consists of the definition of a Multi-level Map, which is a representation of the environment in different levels of abstraction and is described in Section 3.2. Starting at a higher level of abstraction (i.e. less detail and lower density

[1]Thesis developed by André Antonitsch, at PPGCC, PUCRS supervised by Prof. Soraia Musse.

of information), the environment is described as a set of regions within its boundaries, representing areas of interest, such as the states of a country, or the neighborhood of a city. In the following levels, more information is added to the environment, including routes between neighboring regions; a road network that provides local access inside each region; and the definition of building areas.

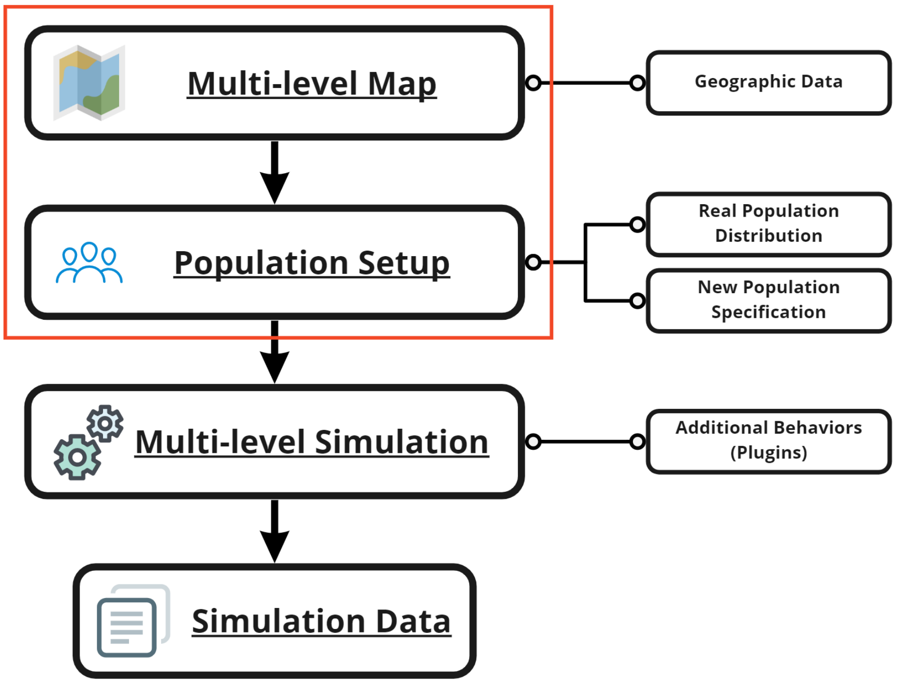

Multi-level Map
Geographic Data
Population Setup
Real Population Distribution
New Population Specification
Multi-level Simulation
Additional Behaviors (Plugins)
Simulation Data

Figure 3.1: LODUS architecture overview.

The Population Setup consists of three main steps: i) to describe the population, ii) to distribute the population around the simulated environment, and iii) to create routines for the population. In the first step, characteristics of the population are defined, such as age and occupation. Next, the population is distributed in the regions of the environment. Finally, the motions are defined, describing the movement of the population in groups of actions. For example, a city may send 100 people to work in another city every day. The characteristics defined in the first step can be used to filter people to go to a specific routine. The steps of the Population Setup module are described in Section 3.3.

The Multi-Level Simulation consists of the execution of the routines in the environment. The simulation is executed for a desired number of days, with each day being divided into time-steps (e.g. hours). At each time step: the simulator identifies the actions that should be executed; and executes the action, usually moving a population group between regions[2]. New behaviors and actions can be added to the simulation module through plugins. The simulation produces output data that contains information about the population, e.g., the position of population groups and the number of people in each area of the environment. Data is distributed by time-step and, when appropriate, by region.

[2]A more detailed description of the methodology of our multi-level simulation can be seen in the master's thesis of André Da Silva Antonitsch. His work will be made available at: http://tede2.pucrs.br/tede2/

The following sections present the methodology for the modules highlighted in red in Figure 3.1, which includes the creation of the multi-level map and the population setup, and are the contributions of this document.

3.2 Multi-Level Map

This section presents our proposed model for establishing a multi-level map. The map is a representation of a virtual environment (e.g. a city or a country) that can be depicted in increasing detail as more granular objects are modeled. In the first level, the whole environment is contained in a 2D plane that delimits the position of its objects (Environment Boundaries), which can be subdivided into regions. A Region (R) is formed by 2D points that represent a location, in the environment, with a high level of interest (e.g. a city, or a neighborhood). Given a set of *regions* distributed in the environment, a Regions Graph (G) is established, where nodes are the central points of the *regions* and edges represent access between *regions*. While the Regions Graph describes the spatial structure at each level of abstraction, e.g., neighborhoods and their connections, we also have the Multi-level Graph, which connects the *regions* from one higher level to lower levels, allowing a hierarchical representation of the environment. Both structures are next detailed.

The **Multi-level Graph** is a hierarchical representation of the environment. Each level of this hierarchy, called a *Level of Abstraction*, represents a different type of spatial structure and is composed of multiple elements. Each *Element* represents an instance of such spatial structure, for example, three distinct cities (City A, City B, and City C), represented by one *element* each, belong to the same *level of abstraction* and have the State as the higher level. Each *element* can be subdivided into a new type of spatial structure, represented by a new *level of abstraction*. For example, a city (*element* in the level Cities) may be divided into multiple neighborhoods (*elements* in a lower *level of abstraction*). Each time a new level is created, the lower level inherits information from the upper level. So, the cities know their states, that know their country, etc. While the Multi-level Graph connects the city with the neighborhoods (in the hierarchy among *levels of abstractions*), for instance, the **Regions Graph** connects the elements cities, in the level of a certain state, as another Regions graph connects elements neighborhoods, in the level of a certain city.

Our definition of Regions Graph is an undirected graph $G = (N, E)$, where N is a set of nodes (i.e. central points of *regions*) and E a set of edges (i.e routes) between nodes. A *region* (R) can represent different structures according to the *level of abstraction*. An edge is defined as a straight line segment between a pair of nodes. Figure 3.2 presents a typical example of a Multi-level Graph's hierarchy in a country. This hierarchy contains a diverse range of *levels of abstraction*, from a country divided into states, down to individual rooms of a building floor. The Multi-level Graph has a flexible structure, allowing the user to define the available *levels of abstraction* according to the environment being modeled. The only requirement is that the structure in a lower level must be restricted to the space of the higher level. The subdivision of all *elements* down to the lowest *level* *of abstraction* is not a requirement. Examples include City B and City D, which are not subdivided into neighborhoods. This allows the user to add *elements* according to only the available data, reducing the scope of experiments and the overall environment.

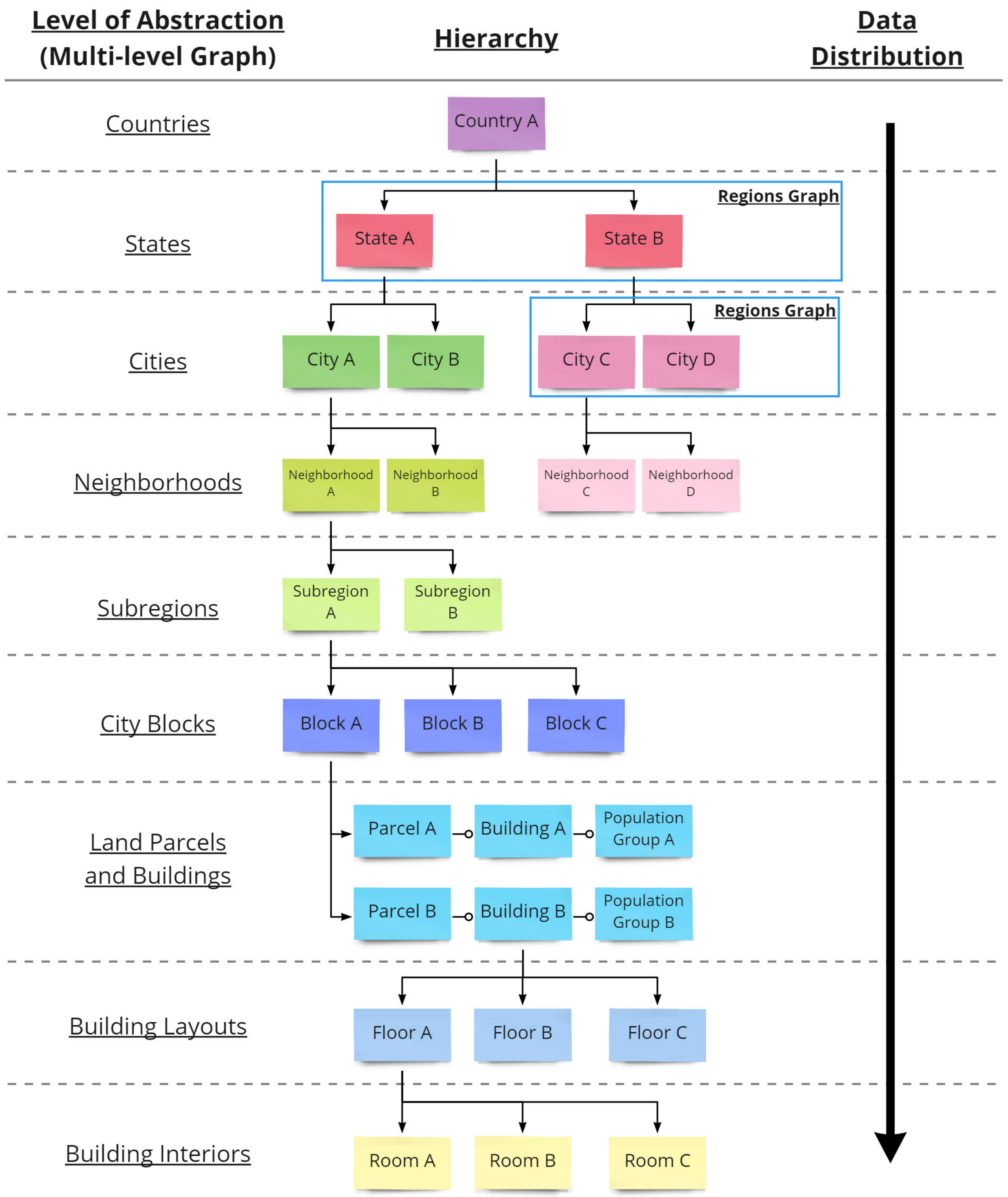


Figure 3.2: Multi-level Graph's hierarchy.

A *level of abstraction* can represent more than one type of spatial structure if desired by the user. In the example hierarchy (Figure 3.2), this occurs in the "Land Parcels and Buildings" level, where a land parcel contains a building within its boundaries, and the building contains a population group that, for example, describes its residents. In this relationship, the "building" structure type is used to model the next *level of abstraction*, where a building is divided into floors. Another possibility is to represent "Land Parcels" and "Buildings" as distinct *levels of abstraction* if desired by the user modeling the environment.

Although the multi-level map is designed to represent all the *levels of abstraction* mentioned, it is possible to restrict its construction to only the desired levels. For example, if the user desires to evaluate routes between cities of a single state, a Multi-level Graph can be constructed with only the levels of "States" and "Cities". If the user desires to evaluate spatial data regarding blocks in a city, a Multi-level Graph can be constructed with only the levels of "Cities" and "City Blocks", without requiring any intermediate levels. In both cases, our proposed method allows the user to model and navigate to lower *levels of abstraction*, given the required data. The next section presents details of the current implementation of our model.

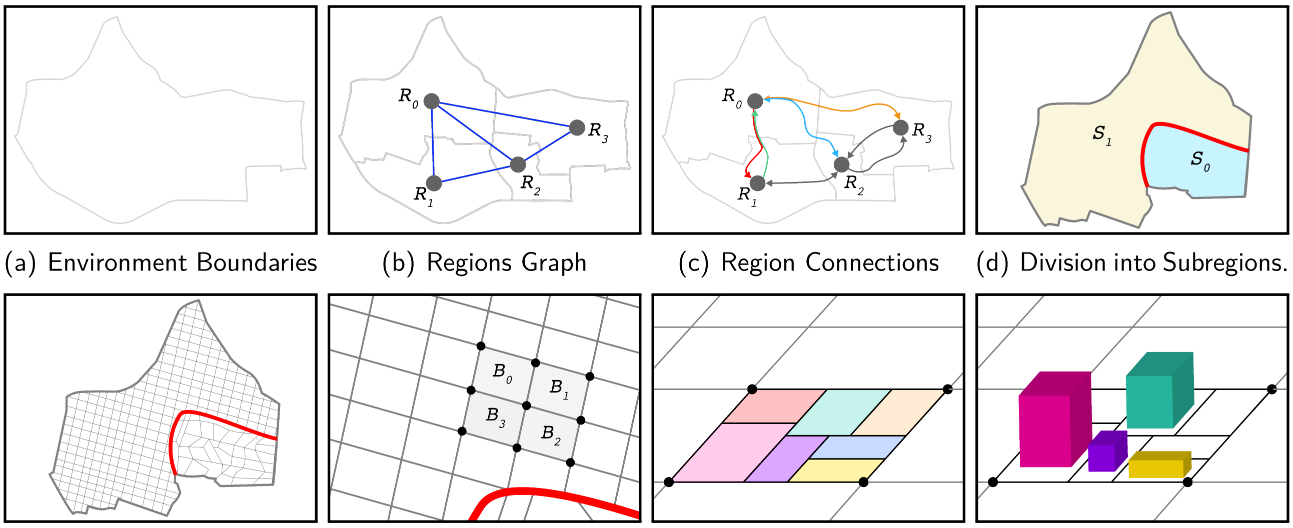
3.2.1 Pipeline Methodology

In this section, we detail the proposed methodology implemented in this work. The figure used as the basis for our explanation is Figure 3.3. Details about each stage of the multi-level pipeline and its elements are described in the following subsections. First, the environment boundaries are represented in a 2D plane (a). Then, links between a set of Regions are represented in a higher level of abstraction through a Regions Graph (b). Edges of the graph are replaced with detailed routes (c). A Region is divided into a set of Subregions (d). A Road Network is formed (e). City blocks are defined within the road network (f). Finally, a City Block is subdivided into a set of Parcels (g), which are used as the base for a building geometry (h).

The input data required to generate the multi-level map can be obtained through different methods. Publicly available Geographic Information Systems (GIS) databases, e.g. OpenStreetMap (OSM)[3], can provide information to describe the position of roads and buildings footprints of real cities. City managers may also have access to mappings of sections of the city, such as the boundaries of neighborhoods, which can be used as input for the definition of *regions*. Procedural generation methods, as presented in Chapter 2, can also be used to provide data for new areas or areas still being planned.

However, the required data to model the multi-level environment may not be available to the user in some specific levels of detail, e.g., data to define city blocks in a neighborhood. In this case, users can manually inform data following some concepts, as described in the next sections.

[3]OpenStreetMap (OSM) is available at: https://www.openstreetmap.org/



(a) Environment Boundaries (b) Regions Graph (c) Region Connections (d) Division into Subregions. (e) Definition of a Road Network (f) Definition of City Blocks. (g) Parcel Subdivision (h) Placement of Buildings

Figure 3.3: Overview of the multi-level map pipeline. In (a), the boundaries of the environment are defined within a 2D plane. In (b), a graph between Regions of interest is defined. In (c), edges receive higher detail. In (d), a Region is divided into Subregions. In (e), a Road Network is established for each Subregion. In (f), City Blocks are defined. In (g), a City Block is subdivided into land lots (Parcels). In (h), buildings are created in each Parcel.

Regions

In our method, the spatial elements are represented through *regions*, as described before, and must be restricted to the boundaries of the environment. The Outline of a Region (O_R) is the visual representation of the spatial elements. An *outline* is defined as a circular list of 2D points, forming a polygon that delimits a particular area. The line segments between the points of an *outline* are defined as Osg_R. Two constraints are established to ensure consistency in the placement of new structures in future stages of the pipeline: an *outline* polygon must not self-intersect, and the set of *outlines* must not contain overlapping sections.

The data to compose a *region outline* can be obtained from different sources. For example, data regarding administrative areas (e.g. countries and states) are publicly available through mapping datasets such as Natural Earth[4], OSM-Boundaries[5], and GADM[6]. Editing tools that allow the user to draw a closed 2D polygon may also be used to generate the data for an *outline*. In particular, GIS applications, such as QGIS[7], allow the user to draw and edit polygons in a geographical coordinate system (i.e. latitude-longitude). The constraints established before must be taken into account, independent of the data source. Figure 3.4 presents an example of environment boundaries in (a), and the *region outlines* it contains in (b).

[4]Natural Earth is available at: https://www.naturalearthdata.com/
[5]OSM-Boundaries is available at https://osm-boundaries.com/
[6]GADM is available at: https://gadm.org/
[7]QGIS is available at: https://qgis.org/

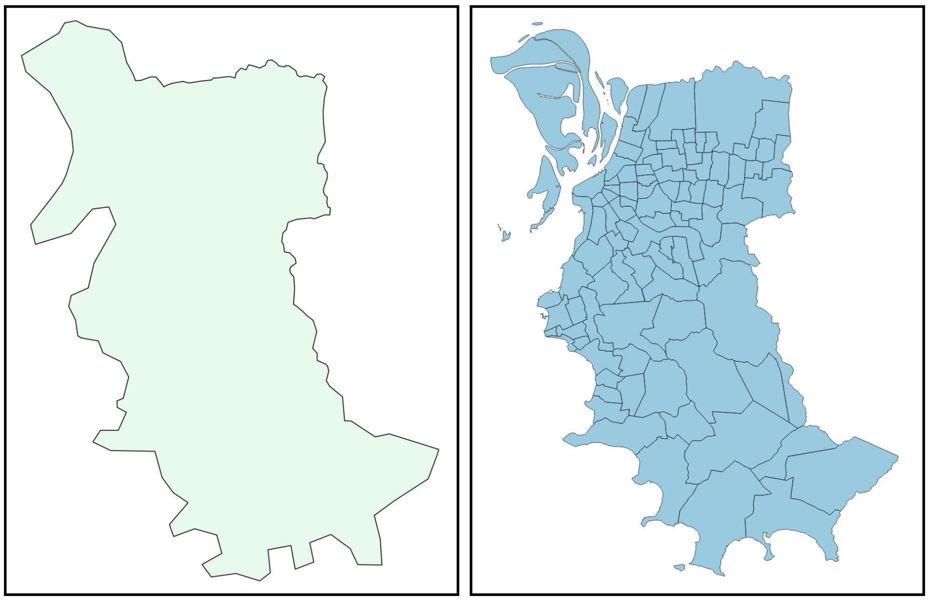

(a) Environment Boundaries. (b) Region Outlines.

Figure 3.4: Region Outlines. The environment boundaries are presented in (a). The *region outlines* are presented in (b).

Connections

A Connection represents an access route from R_a to R_b through an ordered list of 2D points in the environment. This allows a closer representation of real-world streets and roads, effectively replacing the straight edges between nodes of a *regions graph*. Since a road network may contain one-way streets, *connections* are directional. Opposing *connections* (from R_a to R_b and R_b to R_a) may share the same points (reverse order) if desired by the user and permitted by the constraints of the environment (i.e two-way street). Segments of a *connection* may also be shared between different *regions* to allow movement in similar routes. A $TotalTravelTime$ is defined for each *connection*, describing the time required to travel the entire route from start to end. This value is calculated based on an average travel speed for the entire environment, which can be defined by the user.

The data required to compose a *connection* can be obtained through routing APIs (Application Programming Interface), such as Open Source Routing Machine (OSRM)[8], OpenRouteService[9], and Valhalla[10]. These APIs provide a detailed route between two points in a latitude-longitude coordinate system, the same coordinate system of the databases presented previously.

Figure 3.5 presents an example scenario of *connections* between three related *regions*. Two-way street segments are highlighted with a double-headed green arrow. Intersections are highlighted by orange circles. Opposing routes may share all their segments (e.g. *connections* between *regions* 1 and 3), a portion of them, or even none. The *connections* can follow local transit laws, taking one-way roads, and intersections into consideration.

[8]Open Source Routing Machine (OSRM) is available at: http://project-osrm.org/
[9]OpenRouteService is available at: https://openrouteservice.org/
[10]Valhalla is available at: https://github.com/valhalla/valhalla

When modeling the presented structures, the environment can represent the movement of a population between a set of locations in a higher level of abstraction without the need of generating a high-detailed road network and building geometries. Each *region* may represent, e.g., a neighborhood, visually represented by its *outline*, and access between neighborhoods represented by *connections*. A *connection* may be represented through a simple straight line, a user defined route, or based on a real-world road network, e.g., the main avenue used by cars or the route of public transportation.

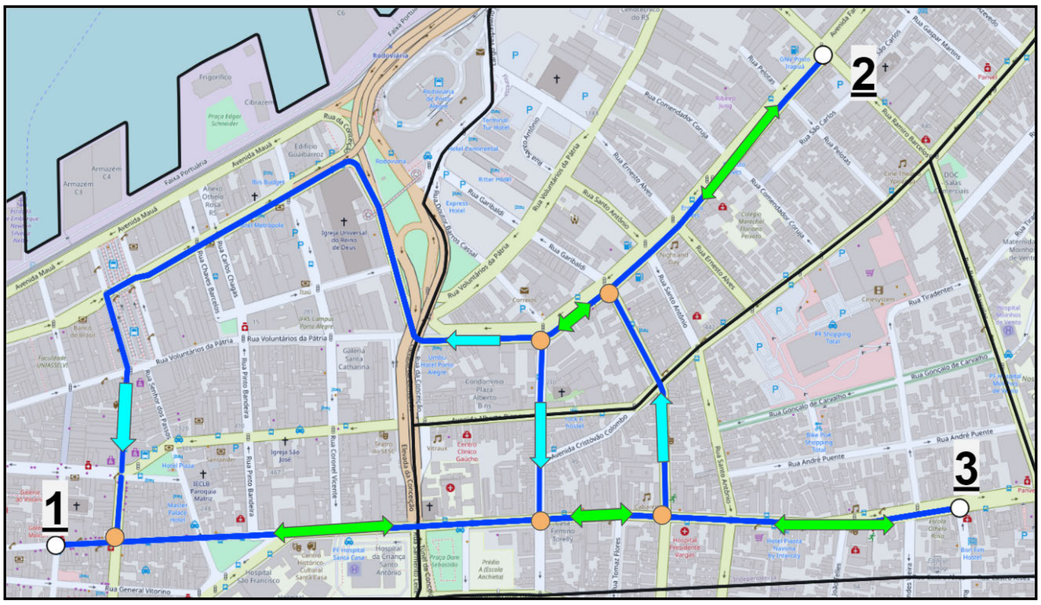


Figure 3.5: Connections between a group of *regions*. Two-way segments are highlighted by a green arrow. One-way segments are highlight by a cyan arrow. Intersections are highlighted by an orange circle.

Subregions

A Subregion is the result of a subdivision that occurred in a *region*, also defined as a polygon formed by a circular list of 2D points. Conceptually, each *subregion* represents an area that shares a profile of buildings, road infrastructure, and population distribution inherited from the *region* at the higher level. For example, a *subregion* that represents a suburban area may have a predominance of smaller houses over tall buildings and a higher number of parks; a city center may have a higher number of apartment buildings, therefore a higher population density. So, it affects the more detailed levels in the hierarchy.

In our proposed implementation, *subregions* can be informed by the user, using data extracted from geographical sources (as the regions, presented before). Also, users can manually define the *subregions* by entering with a list of points. If a *subregion* should be divided again, to build blocks from a neighborhood, for example, a similar process happens again, i.e., users can manually inform data or use extracted data to create the blocks from geographical data. The creation of *subregions* can be associated with the roads, depending on the level of detail the user is creating. Such roads are discussed in the next section.

Road Network

A Road Network can be established to achieve a more detailed representation of the environment. A Road is defined by a list of 2D points in space. Such information can be informed by the user using data from geographical systems, but also manually. For example, OpenStreetMap (OSM)[11] contains a large database describing real-world roads and streets. Services such as Geofabrik[12], Overpass API[13], and HOT Export[14] present a combination of both regularly updated OSM datasets and tools for extracting data for a specific area. This data contains a variety of attributes to describe roads in higher detail, e.g., width, surface material, and the presence of sidewalks. In our current implementation, we define the following set of parameters to represent characteristics of a *road*:

- $Width$: Total width of the road;
- $Lanes$: Number of lanes dividing the road;
- $IsOneWay$: Indicates if a road is a one-way street.

Real-world streets may present a change in their width in their extent, e.g., an increase in the number of lanes. In our proposed implementation, two distinct *roads* must be created to represent this transition. Also, a *road* may intersect with the polygon boundaries of a *subregion*. In this scenario, two distinct *roads* must also be created.

The user can define default values for the presented parameters. For example, all roads may have a default $Width$ of $6m$ unless the available data presents an overwriting value. The default values can be defined for the entire environment or specific *subregions*. This information, especially the $Width$ parameter, is used to create a 3D mesh to represent road segments. This process is described and presented in Section 4.2.2 in the following chapter.

City Blocks and Land Parcels

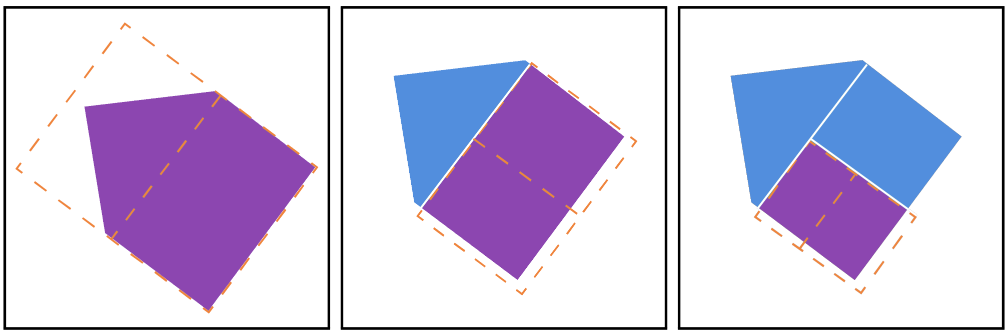
A City Block is also represented by a set of 2D points forming a polygon. Each *block* comprises a set of *land parcels*, which are used as a base for the generation of buildings and houses. *Blocks* within the same *subregion* may share a default set of properties to give them a similar look (described in the following subsection). In a similar process to the previously presented, the data to compose a *city block* and its *parcels* can be manually informed by the user or obtained from geographical data.

When not defined by the user, *land parcels* can be generated with a subdivision algorithm. The subdivision can be applied to only desired *blocks* with a large enough area. The process is

[11]OpenStreetMap (OSM) is available at: https://www.openstreetmap.org/
[12]Geofabrik is available at: https://www.geofabrik.de/
[13]Overpass API is available at: https://overpass-api.de/
[14]HOT Export is available at: https://export.hotosm.org/

based on the work of Parish and Müller [34], where a series of splits are applied to a geometry. Figure 3.6 presents a small number of steps of this algorithm. First, the larger axis of an Oriented Bounding Box (OBB) is identified. Then, a cut is made perpendicularly across the axis mid-point, creating two new *parcels*. This process is repeated recursively until the area of a new *parcel* is below a specified threshold. An offset value (in %) can be used to change the axis mid-point at each iteration, allowing a larger variety of outputs.

A cut can result in a *parcel* being created inside the geometry without street access. Following user preferences, different solutions can be applied in these scenarios: (a) the *parcel* is flagged as "non-building", being used to generate parks, playgrounds, and empty lots; (b) the cut is reverted and the previous *parcel* is validated, ending the recursion; (c) the cut is reverted and the smaller axis is selected for the split, creating two narrower lots, with both having street access; (d) a secondary street is created following one of the OBB's axis until the lot is reached; (e) local access (i.e alleyway) is established, not altering the *city block* geometry.



(a) First iteration. (b) Second iteration. (c) Third iteration.

Figure 3.6: Parcel subdivision method based on OBB recursive cuts. The result of the first three iterations is presented in (a), (b), and (c). OBBs and their mid-points are represented by the dotted orange lines. Generated parcels at each iteration are highlighted in purple.

Building Properties

The generation of complex geometries of buildings is out of the scope of this thesis and our proposed pipeline. To allow the creation of simple 3D representation for buildings in our current implementation, we defined the following set of parameters:

- $Setback_{front}$: Free area in the front of the building;
- $Setback_{side}$: Free area in the sides of the building;
- $BuildingHeight$: Total height of the building;
- $Floors$: Number of floors in the building.

Setbacks ($Setback_{front}$ and $Setback_{side}$) are applied to reduce the usable area of each *parcel*, creating a space for sidewalks, vegetation, and environment props. $BuildingHeight$ and

$Floors$ are randomly selected from a range specified for the entire environment, for each *subregion*, or individual *blocks*. In a similar process to *roads*, these values can be overwritten if the available data presents the required information. For example, the setback parameters may not be applied, if desired by the user, when data of the footprints of buildings are available.

In this final stage of our pipeline, a high-detail representation of the environment is presented, including roads and buildings. Lower *levels of abstraction*, such as buildings layouts and interiors (presented in Figure 3.2), are not covered in our current implementation. These extensions to our pipeline can be explored in future work. The next section presents our methodology for distributing a population in the multi-level map.

3.3 Population Setup

This section presents our methodology for the Population Setup module in LODUS. This module is responsible for: establishing a population description; distributing the population through the environment; and defining routines for this population to be executed during the multi-level simulation module.

3.3.1 Population Description

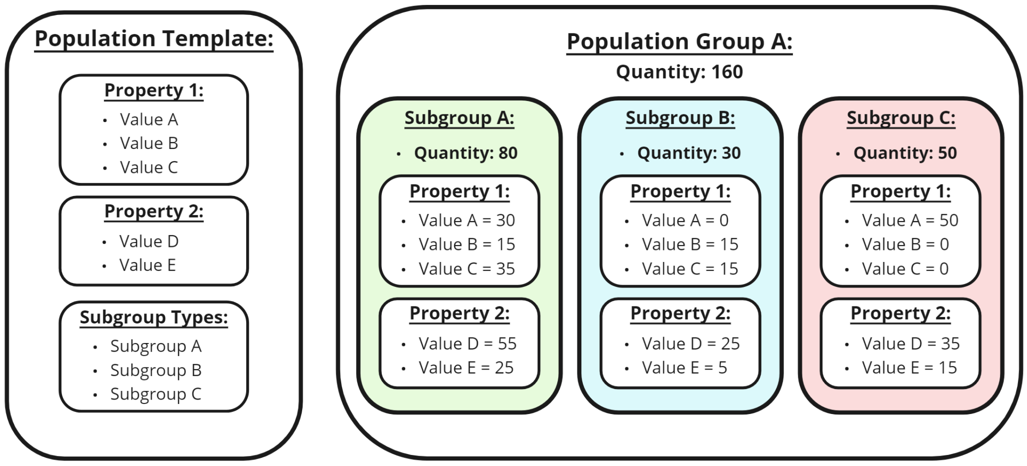
The first step in the Population Setup process consists of the creation of a population template. This template is defined by the user and presents the population profile that will be simulated. Example of profile properties and values are:

- Age: Child, Adolescent, Adult, and Elder;
- $Occupation$: Idle, Student, and Worker.

Additionally, the population template may contain a list of subgroup types, which are responsible for dividing a population group according to its value. Meaning that, when a group of people is moved in the simulation, they will be separated into their respective subgroups while still being a single group, with each subgroup containing its population based on the template. Figure 3.7(a) presents an example structure of a Population Template with two distinct properties and three subgroup types. Figure 3.7(b) presents an example structure of a population group using the template.

3.3.2 Population Distribution

In this step, the population is distributed through a set of *regions* of the environment, which are selected by the user according to the simulation being modeled. For example, the population may be distributed throughout a group of neighborhoods of a city. This process requires the information of a *regions graph*, describing the position of *regions* (i.e central points) and their relationships to other *regions*.



(a) Population Template. (b) Example Population Group.

Figure 3.7: Population description step. An example template for a population being modeled is presented in (a). The composition of a Population Group using this template is presented in (b). The group is divided into subgroup types, each one with its population quantity and distribution.

Each *region* receives a certain population for each subgroup type and each subgroup contains a population distribution. This distribution describes the number of people for each population profile. The sum of all values for every property must match the population count for the current subgroup, as presented in Figure 3.7(b).

During this step, each *region* may also receive a list of nodes. In the multi-level simulation, nodes represent a point in the environment that can be populated in a more detailed level of abstraction, instead of the population being distributed to, e.g., individual buildings. When the information of nodes is not available, a default node is defined in the same position as the *region*'s center point.

Suppose that the *regions* being modeled in an experiment represent neighborhoods, examples of nodes include: a pharmacy; a gas station; or a shopping mall. Nodes are not required in the multi-level map and do not impact the generation of structures in the next states of the pipeline. However, the use of nodes allows the addition of multiple locations for population groups without the need to model new *levels of abstraction* or create new geometries.

3.3.3 Population Events

The final step of the setup module consists of the definition of population routines, which are composed of events distributed in the nodes of the *regions*. Events are grouped by time-step inside the node structure. The time-step indicates the frame events will be executed at each cycle of the simulation (e.g. days, weeks). Each event contains an identifier, an event type, and a list of values. The identifier is only used to represent the goal of the event, such as moving adults to work or students to schools and colleges. The identifier does not affect the result of the event and is an additional resource to guide the user when they are designing large routines or analyzing the simulation log.

The event type indicates to the simulator which actions should be executed. An event type may be responsible for a single action, e.g., moving a group of people from *region* A to B, or multiple actions, e.g., moving groups of people from *region* A towards all other *regions*[15]. The list of values represents the parameters for that event in the simulation and varies according to the event type.

When the values of an event contain a population template, only people that fit the selected property values will be affected by the event. It is possible to not include one or more properties of the template. For example, if an event requests a group of people with "Worker" as their $Occupation$ (presented in Section 3.3.1), only workers will be selected when composing a population group. However, this group can be formed with all values of the Age properties, since a desired value was not specified. In this case, the distribution of the Age property values is based on the available population. Figure 3.8 presents an example of a node structure containing multiple events in different time-steps.

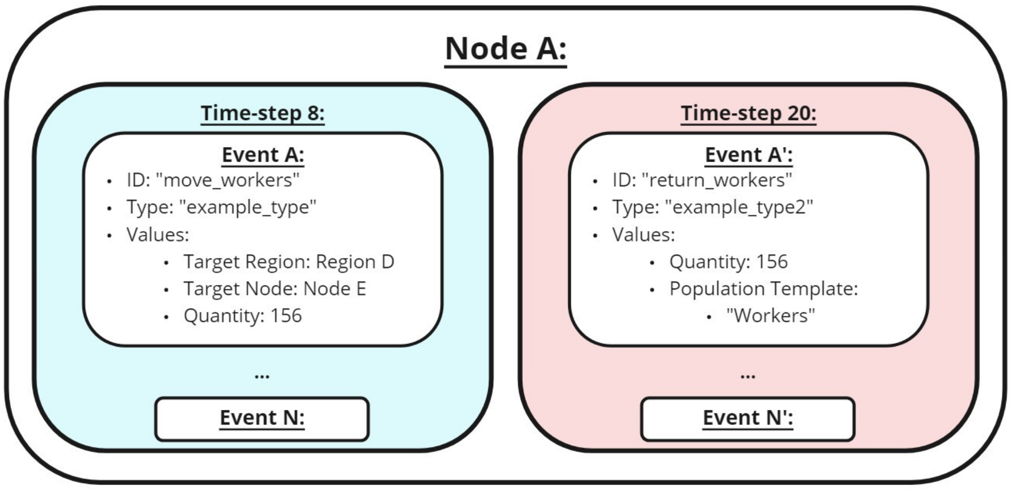


Figure 3.8: Node structure.

3.3.4 Population Setup Tool

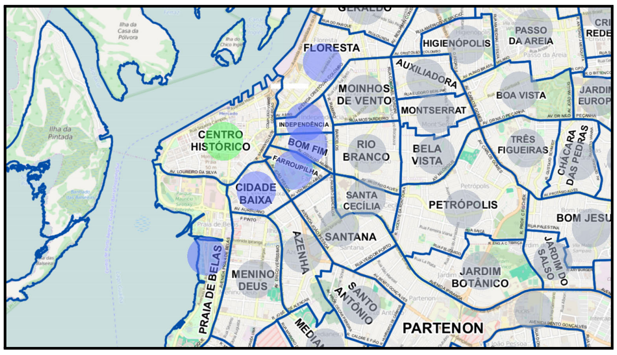
The Population Setup Tool, presented in Figure 3.9, was developed to facilitate the process of creating population routines, parsing the information described in previous sections, and generating

[15]A more detailed description of the methodology of the Multi-level Simulation module can be seen in the master's thesis of André Da Silva Antonitsch. His work will be made available at: http://tede2.pucrs.br/tede2/

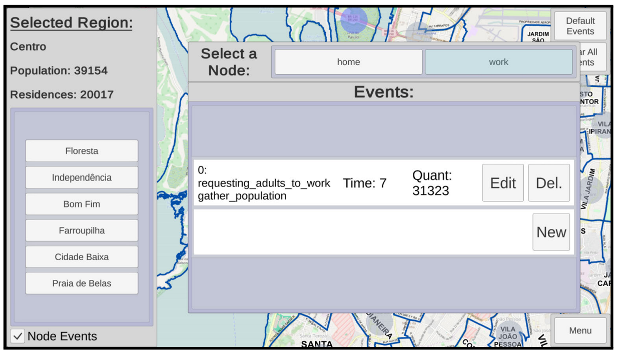
an output file to be used in the multi-level simulation. An example of this output file is presented in APPENDIX B.

Starting with the population description, a profile of possible characteristics of the population is established. These characteristics will become parameters for the routines, allowing certain actions to affect only or require only a population group with the selected values. Then, *regions* are added in the tool's environment based on their position in the *regions graph* (i.e. center points), and relationships between *regions* are established. Each *region* also receives its population distribution and available nodes. From that point, the user can select a *region* and node, and edit the desired events. Since the multi-level simulation module of LODUS can receive new types of actions through the use of plugins, the Population Setup Tool also allows the addition of event types by creating a description file.

Figure 3.9 present the Population Setup Tool. In (a), a group of *regions* is positioned in the environment, with relations between *regions* being established based on the input data. In (b), an overview of the selected *region* is presented in the left panel, containing data of the total population count, and related *regions*. Also, a list of nodes is presented in the central panel, with options to delete and edit existing events or add new ones. In (c) and (d), the editing window for an event is presented. The available options are based on the type of event being edited.



(a) Regions distributed in the environment.


Selected Region:
Centro
Population: 39154
Residences: 20017

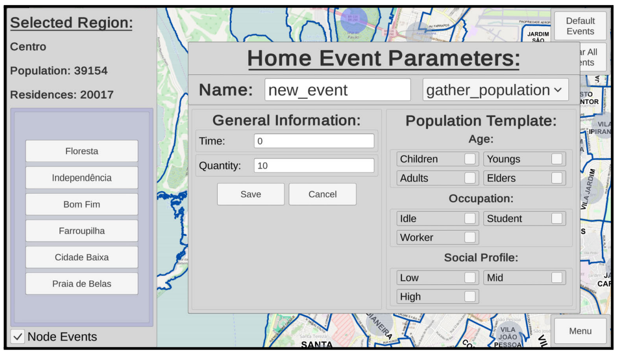
Select a Node: home | work

Events:

0: requesting_adults_to_work gather_population | Time: 7 | Quant: 31323 | Edit | Del.

New

(b) Events for a selected node in a region.

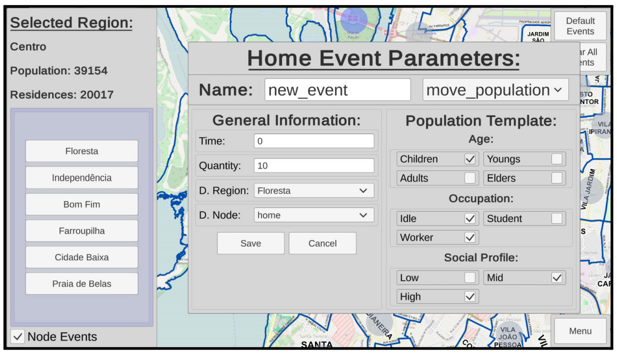

Selected Region:
Centro
Population: 39154
Residences: 20017

Home Event Parameters:
Name: new_event | gather_population

General Information:
Time: 0
Quantity: 10

Population Template:
Age: Children ☐ Youngs ☐ Adults ☐ Elders ☐
Occupation: Idle ☐ Student ☐ Worker ☐
Social Profile: Low ☐ Mid ☐ High ☐

(c) Editing window for a "gather_population" event.


Selected Region:
Centro
Population: 39154
Residences: 20017

Home Event Parameters:
Name: new_event | move_population

General Information:
Time: 0
Quantity: 10
D. Region: Floresta
D. Node: home

Population Template:
Age: Children ✓ Youngs ☐ Adults ☐ Elders ☐
Occupation: Idle ✓ Student ☐ Worker ✓
Social Profile: Low ☐ Mid ✓ High ✓

(d) Editing window for a "move_population" event.

Figure 3.9: Tool for creating population routines. A group of *regions* in an example environment is presented in (a). The selected *region* is highlighted by a green circle. Related regions are highlighted in blue. The remaining regions are highlighted in gray. The existing events in a selected node are presented in (b). The editing windows for different are presented in (c) and (d).

3.4 Multi-level Simulation

This section presents a brief description of the multi-level simulation module. It is important to highlight that this module is not a contribution of the author of this document and was designed and implemented by other members of the LODUS development team[16]. As input, this module requires both the environment description of the multi-level map module and the population distribution and routines of the population setup module.

The multi-level simulation consists of the execution of the population routines in the environment created. The simulation is executed for the desired number of days, with each day being divided into several time-steps (e.g. hours, quarter-hours). The population actions are associated with those time-steps. During the setup of the simulation, the population groups are placed in their initial positions, as described by the population distribution file. An example of a population distribution file is presented in APPENDIX B. At each time step: the simulator identifies the actions that should be executed; evaluates the parameters of the action; checks the availability of the requested population; and executes the action usually moving a population group between *regions* or nodes.

The multi-level simulation produces output data that are used to provide visualization of both the environment and data. Simulation data contains information about the population, i.e., the position of population groups, the number of people at each subgroup type in each group, the population profile, and the *region* of origin for each group. Data is distributed by time-step and, when appropriate, by *region*.

Also, the simulator allows the addition of new behaviors and types of actions through the use of plugins. The following section presents a proposed method to establish a Social Isolation Plugin, which can be used to restrict the movement of population groups during the simulations. Although the simulator was not developed in the scope of this thesis[17], we contributed with the mentioned plugin, explained in the next section, in the context of the LODUS development project.

3.5 Social Isolation Plugin

The population dynamics of a real-world city are not a constant factor. Different population groups may vary their routines from time to time. For example, not all workers have pre-established shifts from Monday to Friday. Also, activities may have inconsistent cycles, such as the need to go grocery shopping or the possibility of working from home a certain number of days in a week. During the COVID 19 pandemic, two terms became well known for the general public: social distancing and social isolation. Social Distancing consists in the act of maintaining a certain distance from

[16]This module was designed and implemented by André Da Silva Antonitsch.

[17]A more detailed description of the methodology of the multi-level simulation module can be seen in the master's thesis of André Da Silva Antonitsch. His work will be made available at: http://tede2.pucrs.br/tede2/

other people in the same location, avoiding physical contact aiming to reduce the spread of the virus. Social Isolation consists of, not only avoid contact with other people, but also the reduction of activities that require leaving a home.

In previous work [40], we proposed a method to estimate the probability of infection events occurring in the environment based on local densities. The method uses crowd simulation to control the movement of agents with and without a social distancing behavior. In that study, we concluded that the addition of social distancing behaviors reduces the probability of infection events. However, the proposed model focuses on a microscopic representation of the population, where each individual can be simulated. To evaluate the impact of social factors in a macroscopic representation of the population, following our multi-level approach, we propose a method to restrict the movement of population groups using a Social Isolation Index.

The social isolation factor represents the proportion of the population originating from a given location that does not leave that location. The index ranges from 0 to 1 and may vary every day (or a different time-step) according to the data available. A high value of the isolation index indicates that most of that population group stayed within the designated area. Once the Social Isolation Index for the *regions* of the environment is calculated, we are able the alter the number of people being affected by the population events during the multi-level simulation based on their origin *region*. This additional behavior is integrated into the multi-level simulation module through a social isolation plugin.

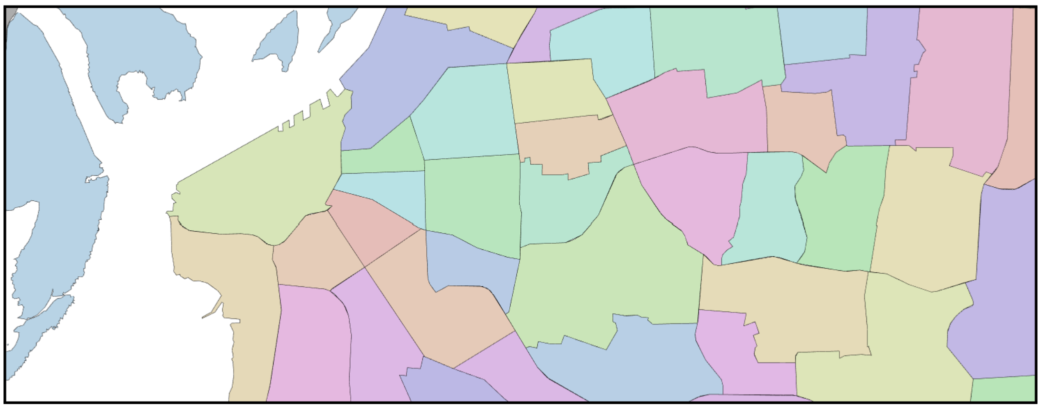
The obtained data[18] is modeled with the geospatial indexing system H3 (Hexagonal Hierarchical Geospatial Indexing System), where the space is hierarchically divided into hexagons (presented in Figure 3.10). Designated areas for each population group are composed of geospatial hexagons with approximately $450m$ radius. For any given day, each hexagon has six entries (one for each point) that shares a single isolation index.

To avoid high variations or inconsistencies in the isolation index due to small sample sizes, the data for a hexagon is only available for population groups of at least 50 people. To calculate the isolation data, the position of individuals is collected late at night, assuming they are asleep. These positions are used to determine which hexagon each person is associated with. If during the day they leave the associated area of their hexagon, the index is affected. It's important to highlight that the available data does not contain any information regarding the individuals, only the spatial position of hexagons and their isolation indexes.

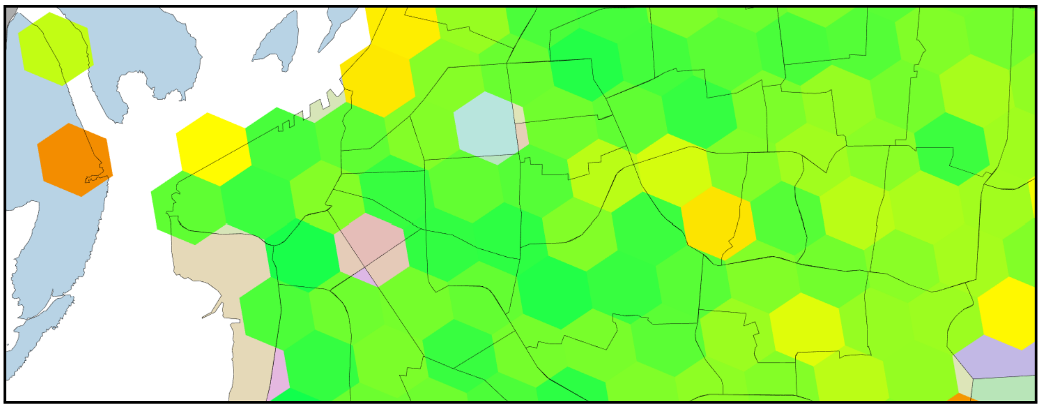
Figure 3.10 presents a visualization of the social isolation data. The isolation index ranges from 0 to 1, where lower values of isolation are presented in red and orange, average values are presented in yellow, and higher values are presented in green. In (a), an *element* divided into multiple *regions* is presented. In (b), a day with high social index values is presented, meaning that most of the population in the green hexagons did not leave their designated areas. In (c), a day with medium social index values is presented, meaning that approximately half of the population in the

[18]The isolation data was provided by Inloco (now called Incognia) in a partnership with PUCRS. Incognia website: https://www.incognia.com/

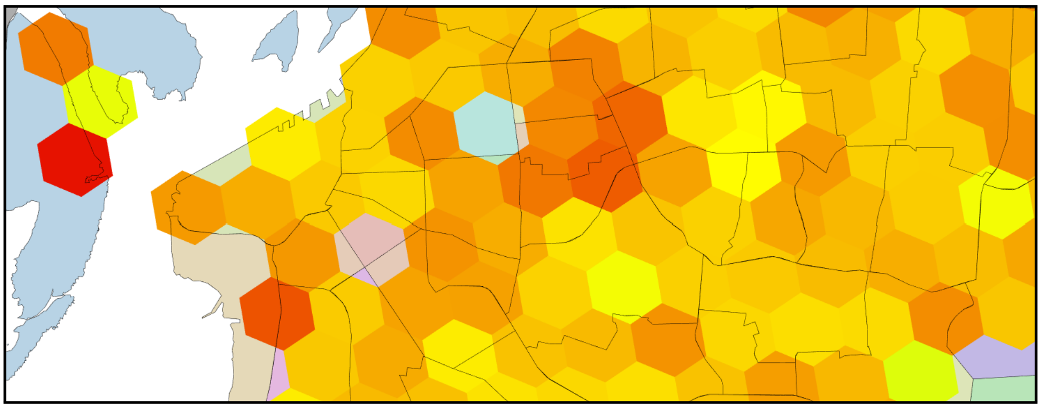
orange and yellow hexagons did not leave their designated areas. Section 4.5 presents our process for projecting the available data into an *element* modeled based on a real-world city, and an analysis of the evolution of the isolation index.



(a) Default visualization an *element* an its *regions*.



(b) Visualization of a day with high values of social isolation. Hexagons in green present a value of 0.6 or higher.



(c) Visualization of a day with medium values of social isolation. Hexagons in orange and yellow present a value between 0.4 and 0.6.

Figure 3.10: Visualization of social isolation data through hexagons. The isolation index ranges from 0 to 1. Lower values of isolation are presented in red (< 0.4). Medium values are presented in orange and yellow. Higher values are presented in green (> 0.6).

The next chapter presents the results achieved in modeling a real-world city using the described methodologies.

4. RESULTS

This chapter presents the process and results achieved in modeling a real-world city in a virtual environment using our proposed method. As proof of concept, we selected the city of Porto Alegre in Brazil to be modeled and presented in different levels of detail through a Multi-level Graph. Following the methodology for distributing data in the environment, *elements* at lower levels of the graph's hierarchy are represented in higher detail.

This virtual environment is presented through a visualization component for the multi-level map module of LODUS, responsible for creating and distributing objects, e.g., roads and buildings, according to the available data. The interface allows the user to navigate through the *elements* of the Multi-level Graph and enable or disable the visualization of different objects. Also, the user may select a visualization mode, to display spatial data in the environment.

Data regarding different structures of the modeled environment were collected from multiple sources, which are referenced in their respective sections. The geographic information system application QGIS[1] was used to align information from these different sources within the same coordinate system. The visualization component for the multi-level map was developed in Unity3D[2] using the C# programming language.

This chapter is divided into the following manner: Section 4.1 presents the hierarchy of the Multi-level Graph and the data utilized at every *level of abstraction*; Section 4.2 presents the modeling process and visualization of environment objects, such as roads and buildings; Section 4.3 presents the interface of the visualization component; Section 4.4 presents the results achieved in representing the population in the environment; Section 4.5 presents an analysis of the social isolation data of the city.

4.1 Multi-level Graph

This section presents the hierarchy and data distribution of the Multi-level Graph modeled in our proof of concept. Our main goal is to model the city of Porto Alegre in multiple levels of detail, from the city itself down to individual buildings. In addition to that, we included higher levels to demonstrate the capabilities of our proposed model.

Brazil's territorial subdivision process does not match with the example structure presented in Chapter 3, where states are directly subdivided into cities. Instead, the process includes two additional territorial (i.e. spatial) structures: Intermediate Geographical Regions (also referenced as Intermediate Regions), and Immediate Geographical Regions (also referenced as Immediate Regions). These additional steps were modeled into the Multi-level Graph as two distinct *levels of abstraction*

[1]QGIS is available at: https://www.qgis.org/
[2]Unity3D is available at: https://unity.com/

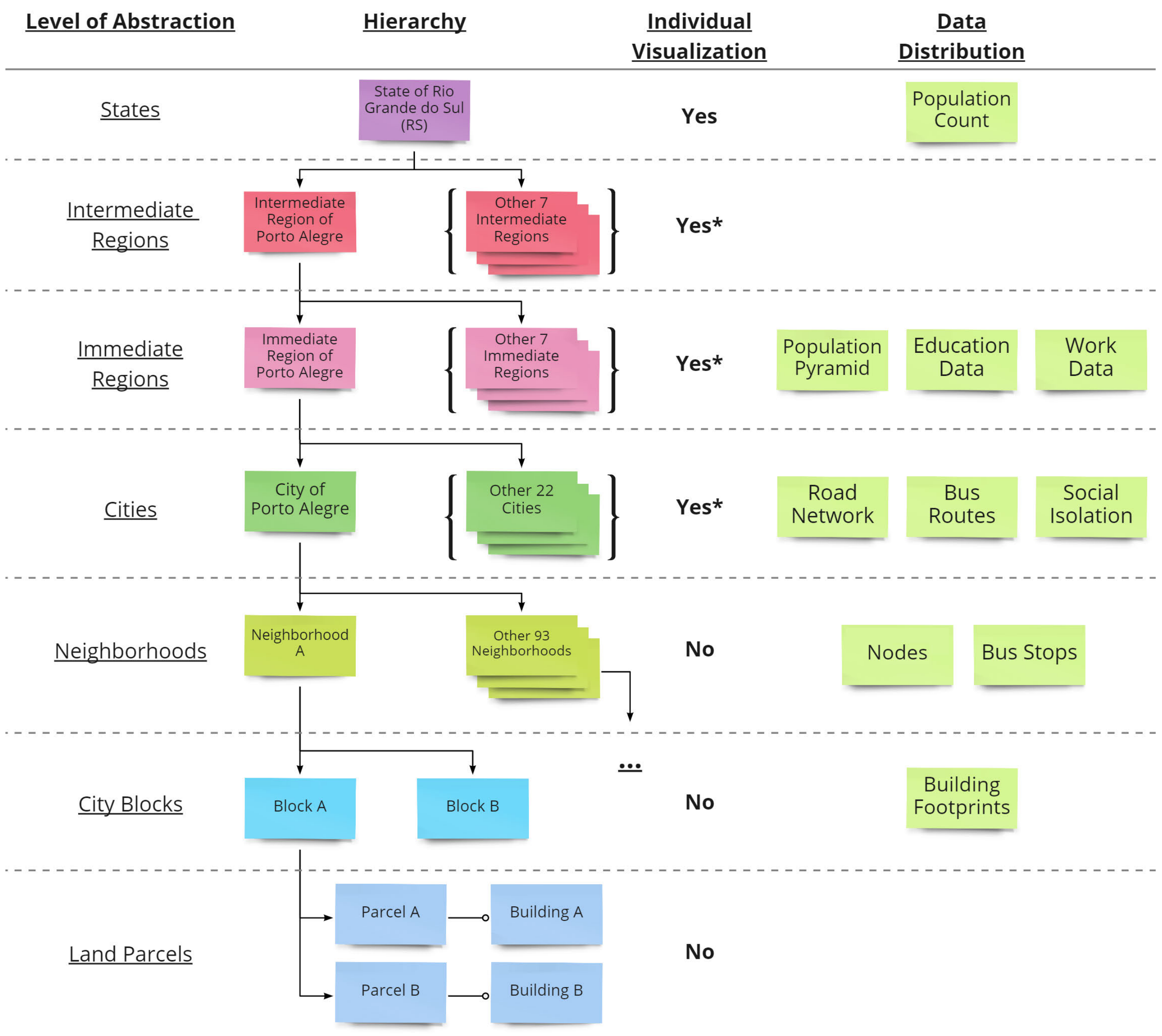


Figure 4.1: Multi-level Graph hierarchy and data available for each *level of abstraction*. "Individual Visualization" indicates which *elements* can be navigated in the visualization component. This does not apply to *elements* grouped in brackets (e.g. "Other 7 Intermediate Regions" in the second *level of abstraction*). "Data Distribution" indicates the starting *level of abstraction* in which that particular data is available. For example, Population Count is available at the first level and is inherited by lower levels.

to provide a more gradual transition from the "States" level to the "Cities" level in the visualization component. Our hierarchy, presented in Figure 4.1, is composed of the following *levels of abstraction*:

- A State is subdivided into multiple intermediate regions;
- An Intermediate Region is subdivided into multiple immediate regions;
- An Immediate Region is subdivided into multiple cities;
- A City is subdivided into multiple neighborhoods;

- A Neighborhood is subdivided into multiple city blocks;
- A City Block is subdivided into multiple parcels;

From the highest level (States) until the level of Porto Alegre (Cities), we selected a single *element* to continue the subdivision process into lower *levels of abstraction*. The environment was divided in the following manner: the "State of Rio Grande do Sul (RS)" was divided into 8 intermediate regions; the "Intermediate Region of Porto Alegre" was divided into 8 immediate regions; the "Immediate Region of Porto Alegre" was divided into 23 cities; the "City of Porto Alegre" was divided into 94 neighborhoods. The *elements* grouped in brackets in Figure 4.1 do not continue the hierarchy to lower levels, e.g., the "Other 7 Intermediate Regions" in the second *level of abstraction*.

The "Individual Visualization" column in Figure 4.1 indicates which spatial structures can be individually accessed and explored in the visualization component. For example, when selecting the "State of Rio Grande do Sul (RS)", information of its *regions* (i.e. 8 intermediate regions), *outlines*, and *connections* are displayed. *Elements* grouped in brackets do not receive an individual visualization, meaning that the user has access to the visualization of four spatial structures in total. Figure 4.2 presents a visualization of these four spatial structures and their respective subdivisions[3].

Each *region* receives a distinct color to highlight its boundaries. To give the user a better sense of space and position in the graph's hierarchy, *regions* of a higher *level of abstraction* are also presented in gray-scale. The user can select a *region* to display its data and, when available, access another *element* in a lower *level of abstraction*. Possible actions of the user and the interface of the visualization component are described in Section 4.3.

Although the 94 neighborhoods of Porto Alegre did not receive an individual representation in the visualization component, all of them continue the hierarchy to lower levels. Each neighborhood contains multiple city blocks, and each city block contains multiple land parcels. This process can be restricted by the user according to the experiment being modeled. For example, the user may select only a subset of neighborhoods to continue to lower levels, reducing the amount of data loaded into the environment.

The "Data Distribution" column in Figure 4.1 indicates the starting *level of abstraction* in which that particular data is available. For example, Population Count is available at the first *level of abstraction* and contains data regarding the "State of Rio Grande do Sul (RS)" and the 8 intermediate regions it is divided into. In addition to that, data can be distributed across multiple levels or be complemented by additional datasets. For example, Population Count is initially composed of data that described the population of the state down to individual cities, where is complemented by

[3]Territorial boundaries data was used to establish the area of individual region outlines. The dataset is publicly available at: https://www.ibge.gov.br/geociencias/organizacao-do-territorio/estrutura-territorial/15774-malhas.html?=&t=downloads

another dataset that describes the population of individual neighborhoods in Porto Alegre. The following list presents a summary of all data used to model the environment and its population[4]:

- Population Count: total population count for each territorial subdivision. The data is available from the entire state down to individual cities. The population of individual neighborhoods is only available for the city of Porto Alegre;

- Population Pyramid: population distribution regarding age and gender. Available for the Immediate Region of Porto Alegre and its 23 cities;

- Education Data: literacy rate, population distribution regarding education level, and the total number of students. Available for the Immediate Region of Porto Alegre and its 23 cities;

- Work Data: total number of people employed. Available for the Immediate Region of Porto Alegre and its 23 cities;

- Road Network: real-world road data, including position and number of lanes of individual segments. Available for the City of Porto Alegre and its 94 neighborhoods;

- Bus Routes: public transportation routes, including traveled streets and orientation. Available for the City of Porto Alegre and its 94 neighborhoods;

- Social Isolation: isolation index for spatial hexagons. The data was processed by the authors and is available in the City of Porto Alegre and its 94 neighborhoods;

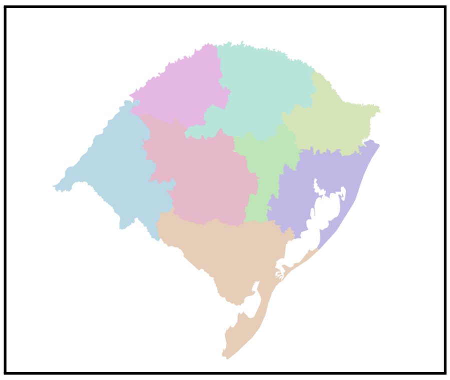
- Nodes: additional points of interest in a *region* to be used in the Population Setup module. Available for the 94 Neighborhoods of Porto Alegre;

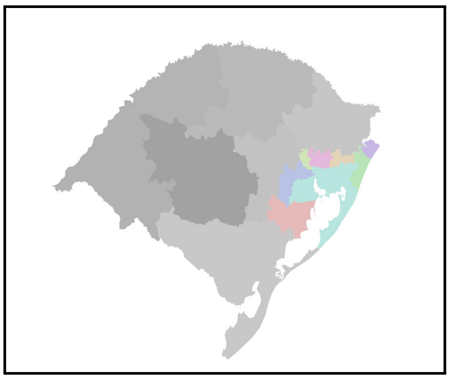
- Bus Stops: individual bus stops used by the public transport system. Available for the 94 Neighborhood of Porto Alegre and positioned in individual city blocks;

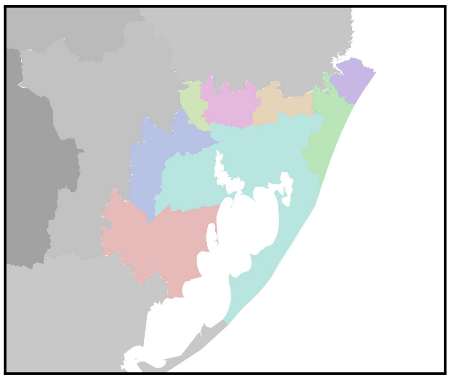
- Building Footprints: list of points outlining real-world buildings. Available for City Blocks and distributed to individual land parcels.

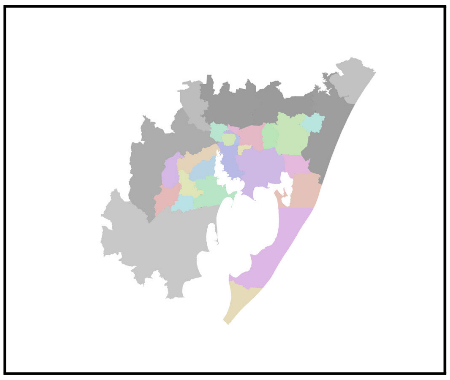
[4]The data distributed into the Multi-level Graph was provided by The Brazilian Institute of Geography and Statistics (IBGE) and The Observatory of the City of Porto Alegre (ObservaPOA). The data is publicly available at:

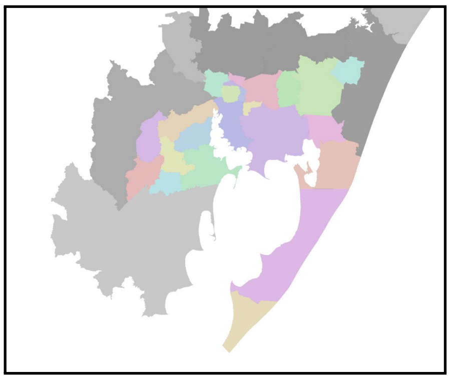
- https://www.ibge.gov.br/cidades-e-estados/rs.html
- https://cidades.ibge.gov.br/brasil/rs/panorama
- https://cidades.ibge.gov.br/brasil/rs/porto-alegre/panorama
- http://www.observapoa.com.br/

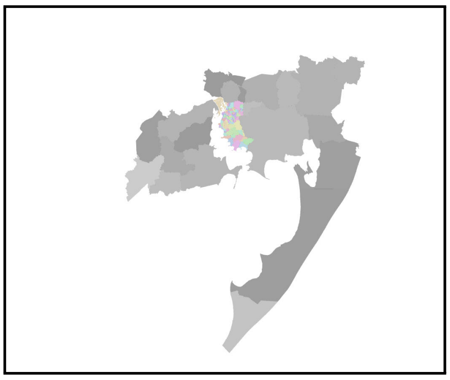

(a) "State of Rio Grande do Sul (RS)".


(b) Transition to "Intermediate Region of Porto Alegre".


(c) "Intermediate Region of Porto Alegre" zoomed-in.


(d) Transition to "Immediate Region of Porto Alegre".


(e) "Immediate Region of Porto Alegre" zoomed-in.


(f) Transition to "City of Porto Alegre".

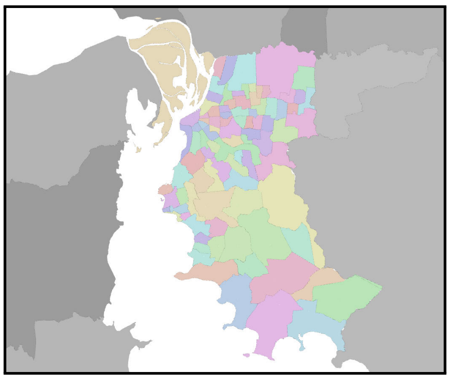

(g) "City of Porto Alegre" zoomed-in.

Figure 4.2: Spatial structures of the Multi-level Graph represented in the visualization component. Each structure is divided into its respective *regions*, with *outlines* in different colors. *Region outlines* of an *element* in a higher *level of abstraction* are presented in gray-scale.

4.2 Environment Structures

The visualization component of the multi-level map allows the user to explore and analyze the virtual environment being constructed. To achieve a more detailed representation, we model a variety of environment structures, such as roads and buildings. These structures were created following the proposed pipeline methodology, described in Section 3.2.1, based on the data available throughout the Multi-level Graph. Section 4.2.1 presents the process of creating detailed routes between *regions*. Section 4.2.2 presents the process of creating 3D geometries for roads and buildings. Section 4.2.3 presents the process of creating additional structures not presented in the proposed pipeline.

4.2.1 Connections

To allow a more detailed representation of access between *regions*, we modeled a set of *connections* using a routing API (application programming interface) called Open Source Routing Machine (OSRM)[5]. The OSRM uses the OpenStreetMap (OSM)[6] database to compose its routes, the same source we used to model roads and buildings, which are described in the following section. Since *connections* are directional, two routes were created between related *regions*: from *region* A towards B, and *region* B towards A. Also, *regions* received a custom position, replacing their center points. These new positions were used to represents points of high interest when modeling the *connections*, e.g., a high flux crossing in a neighborhood.

Through the OSRM API, we were able to obtain a list of 2D points to compose each *connection*. Routes account for local transit laws (e.g. one-way streets) and features of the real-world road network of the city (e.g. roundabouts). Additionally, the data contains the length (km) and travel-time (s) between each pair of points, allowing us to estimate the average travel-time between *regions* in a *regions graph*. Table 4.1 presents the data obtained when modeling the *connections*. Opposing routes are counted individually. The "State of Rio Grande do Sul (RS)" and "Intermediate Region of Porto Alegre" have a massive scale when compared to a single city, resulting in high average lengths. Average travel-times were calculated using the default values of driving speeds and are directly related to the length of a route. Variation may occur due to *elements* in higher *levels of abstraction*, containing a larger number of highways with higher maximum driving speeds when compared to the "City of Porto Alegre".

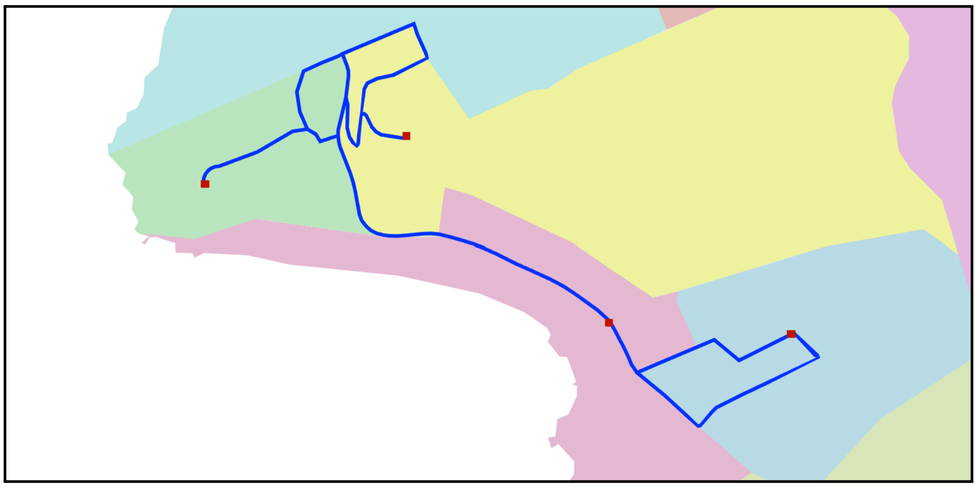
Figure 4.3 presents *connections*, represented by the blue lines, in the visualization component of the multi-level map. The position of each *region* (i.e. custom position replacing the center point) is highlighted by red squares and represent the start and endpoints for all *connections*.

[5]Open Source Routing Machine (OSRM) is available at: http://project-osrm.org/
[6]OpenStreetMap (OSM) is available at: https://www.openstreetmap.org/

Figure 4.3(a) presents the *connections* of a small subset of *regions* in the "City of Porto Alegre". Figure 4.3(b) presents a zoomed-out visualization, including multiple *regions*.

Table 4.1: Connection data by *element* of the Multi-level Graph

Element	**Number of Regions**	**Number of Connections**	**Average Length (km)**	**Average Travel-time (s)**
State of Rio Grande do Sul (RS)	8	30	252.288	13097.19
Intermediate Region of Porto Alegre	8	24	74.836	4349.054
Immediate Region of Porto Alegre	23	81	30.336	1988.459
City of Porto Alegre	94	532	3.302	361.339


(a) Small subset of connections.

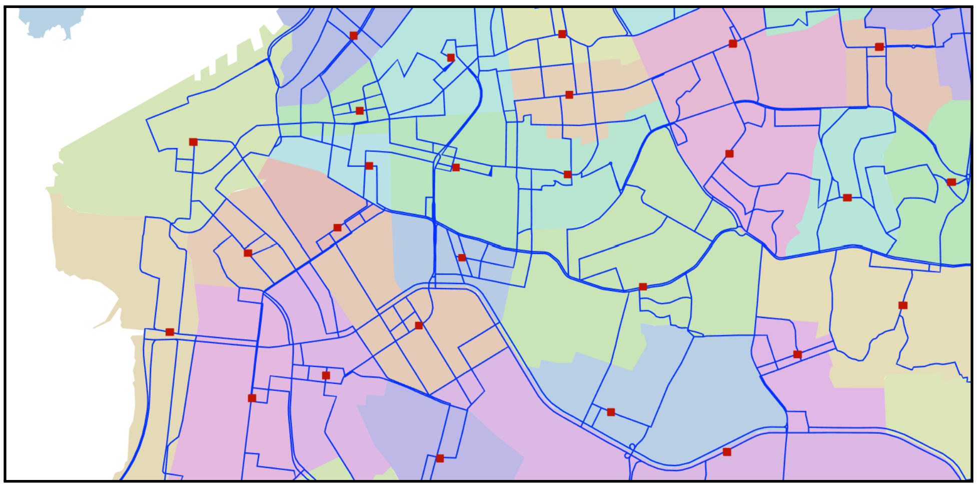

(b) Connections between multiple regions.

Figure 4.3: *Connections* of the "City of Porto Alegre". *Connections* between a group of 4 *regions* are presented in (a), containing: Vila Conceição (green); Sétimo Céu (yellow); Pedra Redonda (pink); and Jardim Isabel (blue). A visualization of *connections* between multiple *regions* is presented in (b). The center point of each *region* is highlighted by red squares.

4.2.2 Roads and Buildings

To approximate our virtual environment representation to the real city of Porto Alegre, we extracted geographic information from the OpenStreetMap (OSM) database, a collaborative mapping project[7], using the HOT Export tool[8]. Data collected was restricted to the boundaries of the city, with ranges of $[-30.2786, -29.9639]$ and $[-51.2691, -51.0123]$ for latitude and longitude respectively. The base structure of the OSM database is a Node (OsmNode), which represents a unique point in space and is defined as a unique ID, a Lat-Long coordinate, a set of tags, and a set of version-control parameters. Tags are a simple key-value pair and are used to describe what the node is representing (e.g. a bench or a building) and additional details (e.g. color, width, and height).

A Way (OsmWay) is a composition of basic data (unique ID and version-control), an oriented list of OsmNodes, and additional tags. Effectively, a Way represents a list of related nodes forming a line and representing, for example, roads and rivers. When the list contains the same node as the first and last index, the Way represents an Area (i.e. polygon), such as parks and building footprints. The data was processed to identify all Ways that represent roads, mainly by searching for the "highway" tag. The following tags were used to compose the *roads* of our proposed pipeline:

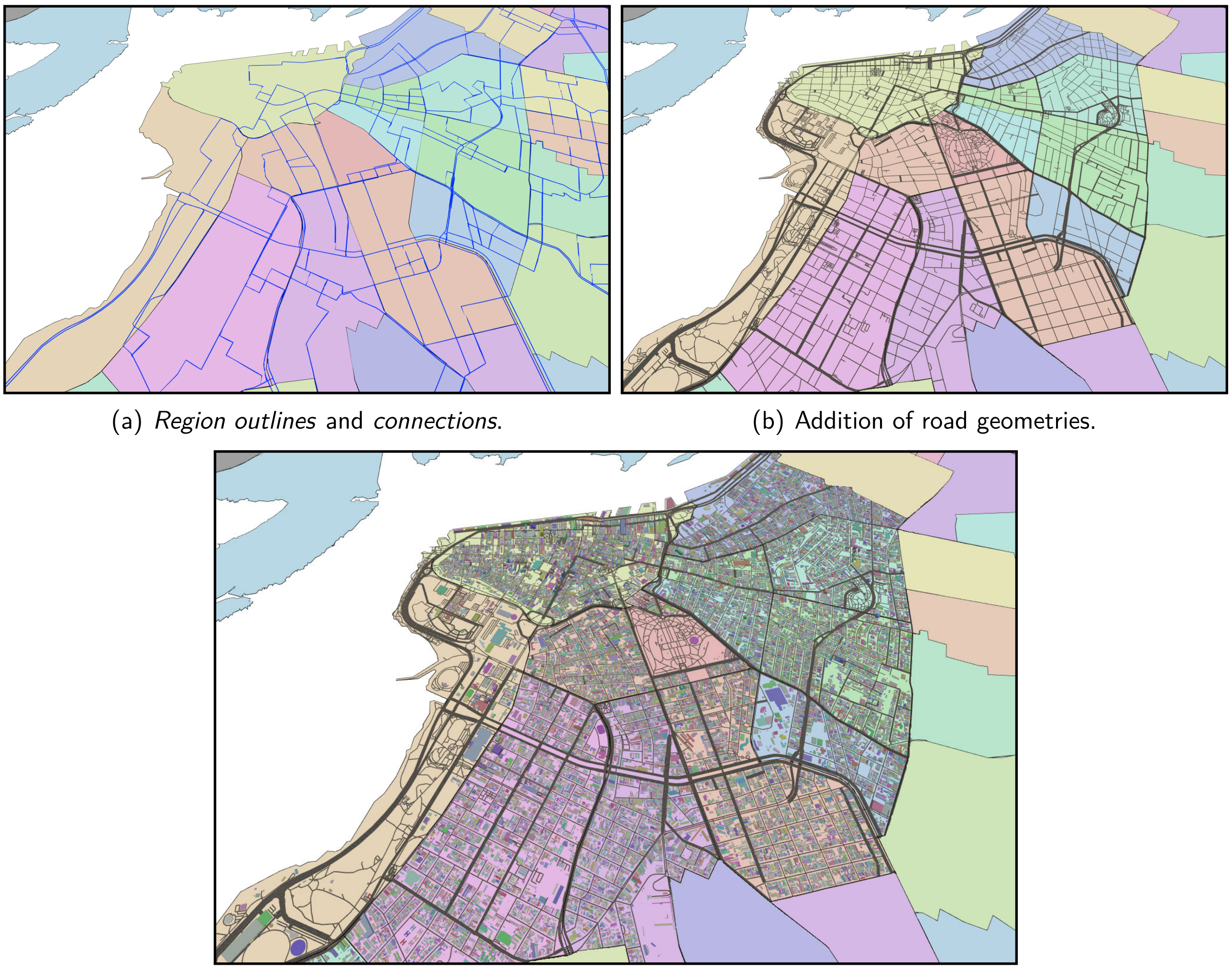
- $highway$: Indicates that the OsmWay is representing a road. The value of this tag indicates the type of road, such as primary, secondary and ternary;
- $width$: Indicates the width, in meters, of a road;
- $lanes$: Indicates the number of lanes of a road. We define a lane as a default value of 3.7 meters per lane. This value can be changed by the user.

The road network is assigned to the "Cities" *level of abstraction*, and points are subdivided throughout the *regions* of the city (i.e. neighborhoods). In the visualization component, roads are represented as oriented quadrilaterals connecting each *road segment*. When using the entire dataset of Porto Alegre, a total of 28,523 *roads* are created in the environment, composed of 82,046 individual points. The amount of data loaded in the environment can be restricted to a certain set of *regions* if desired by the user, as presented in Figure 4.4(b).

Figure 4.4(a) presents the default visualization of the *element* "City of Porto Alegre", containing only the *region outlines* (i.e. neighborhoods) and the *connections* between neighboring *regions*. Figure 4.4(b) presents the creation of road geometries for a subset of *regions*. Figure 4.4(c) presents the creation of building geometries for a subset of *regions*. This process is described below.

Building footprint data obtained through the OsmWays is distributed in the city. The footprint is used to compose the area of a land parcel. Finally, a simple building geometry is created based on the extrusion process presented by Greuter et al [20]. The overall footprint of the building

[7]OpenStreetMap (OSM) is available at: https://www.openstreetmap.org/
[8]HOT Export is available at: https://export.hotosm.org/



(a) *Region outlines* and *connections*.

(b) Addition of road geometries.

(c) Addition of building geometries.

Figure 4.4: Geometry creation. *Region outlines* and *connections* of Porto Alegre are presented in (a). Road geometries are added in (b). Building geometries are added in (c).

composes the floor and it is extruded upwards based on building height. The following tags were used to identify and compose the buildings:

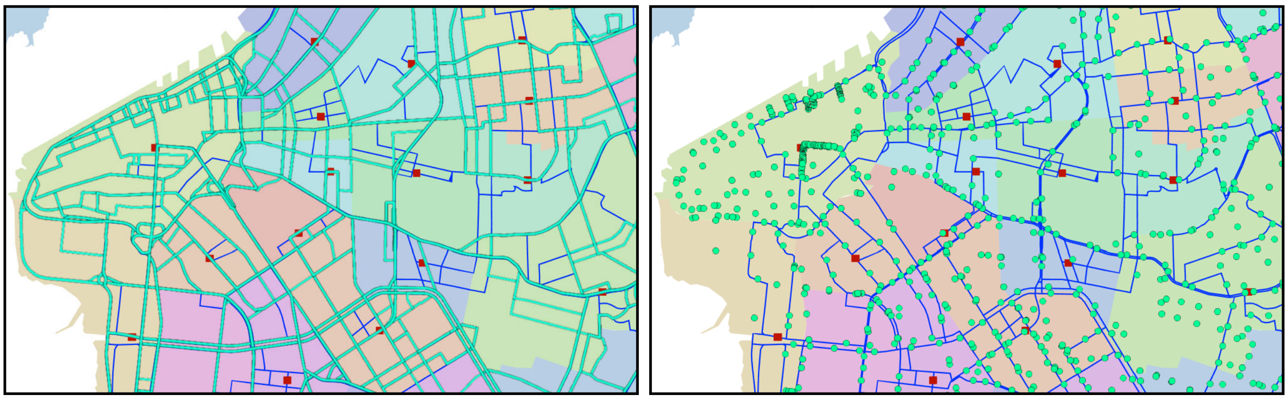
- $building$: Indicates that the OsmWay represents a building footprint;
- $height$: Indicates the height, in meters, of a building;
- $floors$: Indicates the number of floors of a building. Each floor is represented as 3 meters;
- $roof\_levels$: Indicates the number of floors contained within the roof, which are not already counted in $floors$.

Buildings without any information that describes height were created with a default value of 6 meters with an additional random offset between $[-1, 1]$ to add small variations and highlight their silhouettes. The available data about the city Porto Alegre contains a total of 492,723 buildings description. Following the hierarchical structure of the Multi-level Graph, buildings can be added to the environment according to a selected number of neighborhoods, as presented in Figure 4.4(c).

4.2.3 Additional Structures and Points of Interest

Besides roads and buildings, a variety of additional spatial structures can be modeled based on the Multi-level Graph's data and added to the virtual environment as 3D objects. We obtained a dataset of Porto Alegre's public transport system, containing information on bus routes and stops[9]. The data was filtered and adapted to our proposed methodology of hierarchical subdivisions. Bus routes were modeled to a similar structure of a *connection*, being composed by an ordered list of 2D points and represented by a colored line in the visualization component. Data regarding routes was assigned to the "Cities" level of abstraction, being distributed to its subdivisions (i.e. neighborhoods). Additionally, the points of a route were projected into the *region outlines* to identify all the neighborhoods it covers. A total of 1,087 routes were created, being composed of 292.413 points. Figure 4.5(a) presents the visual representation of bus routes in the "City of Porto Alegre".

The bus stops data is composed of a set of 2D points spread throughout the city of Porto Alegre. Additional information is available for each point, including an identifier, an address (street name and number individually), if it is a terminal point, a list of exclusive bus lines, and physical data, such as construction material, floor material, and roof/cover type. Stops are placed over city blocks close to the roads. The data was adapted to our proposed structure of the Multi-level Graph, being assigned to the "Neighborhoods" level of abstraction. Each Neighborhood element contains a group of bus stops, which are distributed to its subdivisions (i.e. city blocks). A total of 5,721 stops were created. Stops are shown as colored spheres in the visualization component. Figure 4.5(b) presents the visual representation of bus stops in the "City of Porto Alegre".



(a) Bus routes data.

(b) Bus stops data.

Figure 4.5: Additional structures of the "City of Porto Alegre". Bus routes of the city are presented in (a). Bus stops in the city are presented in (b).

[9]Data regarding Porto Alegre's public transport system is publicly available at: http://datapoa.com.br/dataset/stpoa-sistema-de-transporte-publico-de-porto-alegre

4.3 Interface

A user interface was developed for the visualization component with two objectives: i) to allow the user to navigate in the environment being modeled, and ii) to allow the visualization of spatial data. Figure 4.6 presented the default interface of the visualization component. The top area of the screen contains a breadcrumb-styled panel, presenting the current *element*'s position in the Multi-level Graph's hierarchy. The user may select another *element* to transition, with the camera position and rotation being adjusted to fit the size of the new *element*.

The bottom left corner presents a panel with most of the interface functionalities. The user may start a crowd simulation in the desired area of the city. A more detailed description of this mode and its interface is presented in Section 4.4.2. A checkbox is also available to enable region labels, containing the name of each *region* in the *element* being presented. The labels are placed directly over the *region*'s center point or when defined by the user, a custom position. The user may select a visualization mode to change the data being displayed in the environment. The following modes were added: Default; Hexagon Isolation; Region Isolation; and Population Groups.

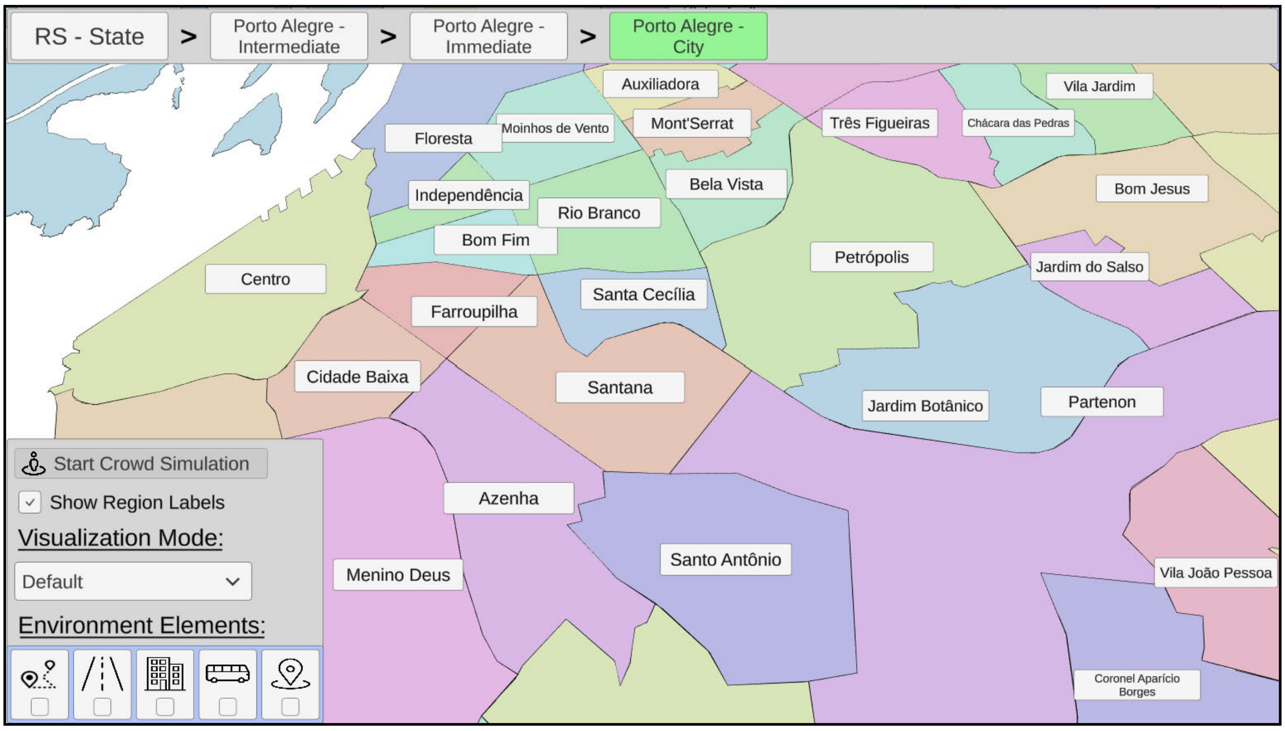
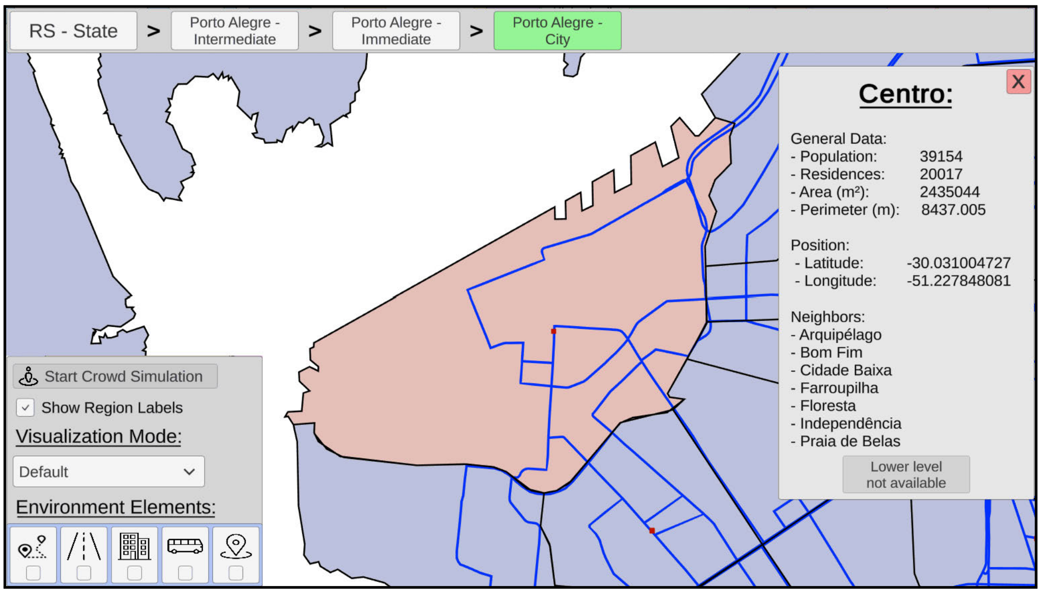


Figure 4.6: Default interface of the visualization component.

Except for the default mode, all modes require that a specific dataset is available at the current *element*, otherwise, they will be disabled. When transitioning between *elements*, the default mode will be selected. A description of the Hexagon Isolation data was previously presented in Section 3.5. Data regarding the Region Isolation mode is presented in Section 4.5. Population Groups mode is presented in Section 4.4.1.

To avoid visual clutter and overlapping data, the user may enable and disable types of structures through a group of check-boxes in the lower-left of the screen. These changes are kept when transitioning between *elements*. If an *element* doesn't have the required dataset, the respective icon will be grayed-out. However, the checkbox will still be editable in case the user accesses a new *element* that contains the required data. The following environment structures can be enabled and disabled: *connections* between *regions*; road geometries; building geometries; bus routes; and bus stops.

When selecting a *region* in the environment, the camera is adjusted to focus on it and a panel is enabled on the right side of the screen, presented in Figure 4.7. This panel displays all available data for that *region*. Finally, the panel contains a button to transition to an *element* in the next lower *level of abstraction* when the data is available. When selected, the camera is adjusted to fit the new *element*, the visualization mode return to the default mode, and the *element* is added to the hierarchy on the top of the interface.



Centro:

General Data:
- Population: 39154
- Residences: 20017
- Area (m²): 2435044
- Perimeter (m): 8437.005

Position:
- Latitude: -30.031004727
- Longitude: -51.227848081

Neighbors:
- Arquipélago
- Bom Fim
- Cidade Baixa
- Farroupilha
- Floresta
- Independência
- Praia de Belas

Figure 4.7: Data of a selected *region* being displayed.

4.4 Population

This section presents the results achieved in visually representing the population in the modeled environment. We created representations in two levels of abstraction. At a higher level, the population is presented through groups moving in the environment, following routes between *regions* towards the locations of their activities (i.e. routines). At a lower level, individuals are represented through virtual agents, moving between buildings of small sections of the environment. Section 4.4.1 presents the composition of a population routine and the visualization of the movement of population groups through the environment. Section 4.4.2 presents the visualization of virtual agents in sections of the city.

4.4.1 Population Groups

We modeled routines for the population of Porto Alegre based on the data available throughout the Multi-level Graph and following the methodology on the Population Setup module, presented in Section 3.3. Files used to model the routines presented in this section are available in APPENDIX B. Our proposed methodology allows the user to model the environment's hierarchy according to the desired experiment. So, we selected a set of the available *regions* (i.e. neighborhoods), reducing the data required to create the environment to fit the scope of our routines. Table 4.2 presents the data of the selected *regions*. The "Related IDs" column indicates the IDs to which a *region* is related, i.e., edges in a *regions graph*.

Table 4.2: Data of the selected *regions* of Porto Alegre

ID	**Region**	**Population**	**Related IDs**
1	Azenha	13804	4, 5, 8, 13
2	Bom Fim	11593	3, 5, 7, 9, 11, 12, 13
3	Centro	39154	2, 4, 5, 6, 7, 10
4	Cidade Baixa	15379	1, 3, 5, 8, 10, 13
5	Farroupilha	961	1, 2, 3, 4, 11, 12, 13
6	Floresta	11596	3, 7, 9
7	Independência	8112	2, 3, 6, 9, 11
8	Menino Deus	31650	1, 4 , 10
9	Moinhos de Vento	11937	2, 6, 7, 11
10	Praia de Belas	2281	3, 4, 8
11	Rio Branco	17531	2, 5, 7, 9, 12, 13
12	Santa Cecília	5768	2, 5, 11, 13
13	Santana	20723	1, 2, 4, 5, 11, 12

The population template created is composed of two properties, "Age" and "Occupation". As presented in Section 4.1, data regarding the population pyramid, education (i.e. number of students), and workers are available for the "City of Porto Alegre", but not to its *regions*. The data were estimated for each neighborhood using the same proportions of the entire city[10]. The following property values and proportions were defined:

- Age: Children ($\approx 18.724\%$), Young ($\approx 15.706\%$); Adults ($\approx 50.507\%$); and Elders ($\approx 15.061\%$);
- $Occupation$: Students ($\approx 21.575\%$), Workers ($\approx 54.706\%$), and Idle ($\approx 23.718\%$).

[10]Data of the population is publicly available at:

- Population pyramid: https://cidades.ibge.gov.br/brasil/rs/porto-alegre/pesquisa/23/25888
- Number of students: https://cidades.ibge.gov.br/brasil/rs/porto-alegre/pesquisa/23/22469
- Number of workers: https://cidades.ibge.gov.br/brasil/rs/porto-alegre/pesquisa/23/22957

Regions received the nodes of "Home", "Work" and "School" to be populated during the simulation. The population was distributed to all 13 *regions* following the presented proportions, being placed initially in the node "Home". A population routine was composed using the population setup tool, presented in Section 3.3.4, in the following manner for all *regions*:

- Work:
  - Time-step 7: Requests "Workers" from neighboring *regions*;
- School:
  - Time-step 7: Requests "Young-Students" from neighboring *regions*;
  - Time-step 12: Requests "Children-Students" from neighboring *regions*;
- Home:
  - Time-step 12: "Young-Students" return to the node "Home" in their origin *region*;
  - Time-step 17: "Children-Students" return to the node "Home" in their origin *region*;
  - Time-step 18: "Workers" return to the node "Home" in their origin *region*.

The requests made from "Work" and "School" only affect people in the node "Home", meaning that people move from "Home" to the node making the request. When a node requests a population group, the *region* it is located in may also send people, i.e., population groups may move from the node "Home" to the node "Work" in the same *region*. The number of people requested is based on the population of the current region, i.e., neighborhoods with a larger population will request more people.

The presented routines were executed in the multi-level simulation module of LODUS, generating simulation data as output. This output describes the position of population groups in the environment and their respective distributions, e.g., a group containing 35 people of the profile "Adults-Workers" is located at the node "Work" of the *region* "Centro".

This data was used to represent the population in a higher level of abstraction through a visualization mode called "Population Groups". In this mode, groups are modeled as 2D metaballs in an organic visualization based on Antonitsch et al. [2][11]. The shapes of metaballs may merge when near each other and no obstacles are present. When a population group is moved between *regions*, the metaball will follow the route described by the *connection* between these *regions*.

Figure 4.8 presents the visualization mode of "Population Groups". In (a), population groups are at their starting positions in the node "Home". In (b), population groups are moving in the environment towards their goals, following the routes described by the *connections*. In (c), the groups are reaching their targets, returning the environment to a default state.

[11] The organic visualization was developed by the colleague André Da Silva Antonitsch, integrating the model of Antonitsch et al. [2] in the multi-level map visualization component.

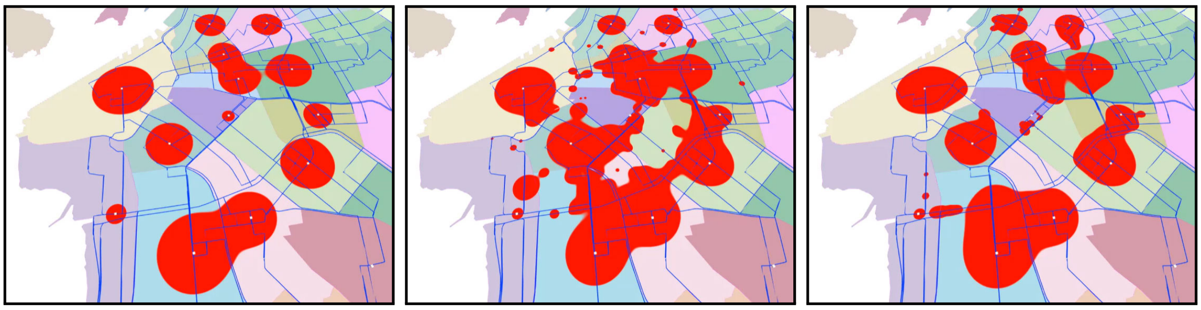

(a) Population groups at their starting positions. (b) Population groups moving along the *connections*. (c) Groups reaching their targets.

Figure 4.8: Visualization of the population groups in the city.

This process allows the visualization of the population in a higher level of abstraction without the need to create geometries for buildings and roads. If desired by the user, buildings can be added to the environment to increase the detail of the visualization. A distance map [2] is created to represent buildings as obstacles for the metaballs. Figure 4.9 presents an example distance map of buildings in (a) and the environment in (b).

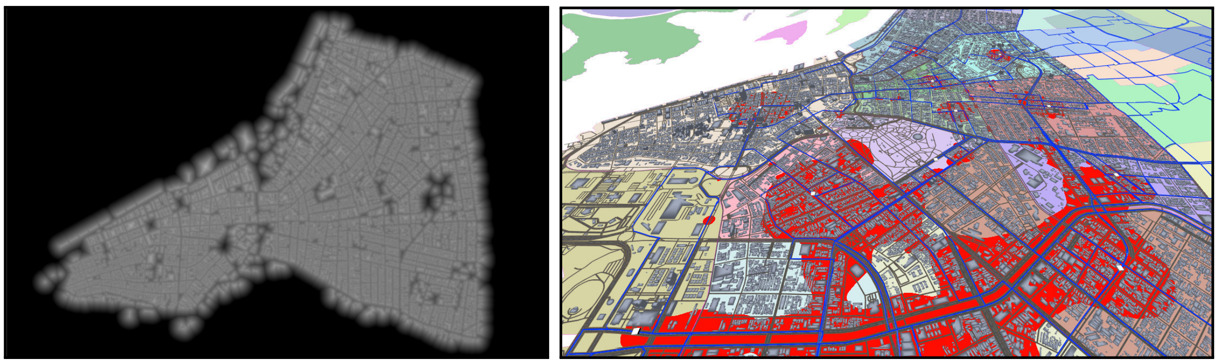

(a) Distance map of buildings. Lighter colors represent smaller distances. (b) Population groups moving in an environment with roads and buildings.

Figure 4.9: Visualization of the population groups with added geometry.

4.4.2 Virtual Agents

The population of the city can also be visually represented in a lower level of abstraction through virtual agents. To achieve that, we implemented the crowd simulation model BioCrowds, presented by Bicho et al. [6] and described in Section 2.4, within the multi-level map. In BioCrowds, a discrete space is divided into cells (i.e. a grid), which are populated by a set of markers. Virtual agents compete for these markers, effectively competing for the space in which they occupy and move.

In our current implementation, this process requires the addition of buildings in the environment, i.e., the lowest *level of abstraction* of the modeled Multi-level Graph. Buildings are used to determine obstacles and spawn areas for agents in the simulation. Also, the creation of a simulation environment is restricted to small sections of the city due to the processing power required. This section is limited to 200x150 units (i.e. meters) by default and may be changed by the user.

The creation of the virtual agents becomes available to the user when the camera of the visualization component is zoomed in the environment, enabling the "Start Crowd Simulation" button of the interface, previously presented in Figure 4.6. When selected, the camera will be adjusted to a top-down angle and orthographic view, as presented in Figure 4.10. Also, a plane, representing the world (i.e. boundaries) of the simulation is presented. Buildings that intersect the boundaries or are contained within the area of this plane will behave as obstacles and spawn areas for agents.

As obstacles, the buildings impede the placement of markers within their area. As spawn areas, agents will be instantiated in front of buildings during the simulation. The position of a spawn area is defined by identifying the center of the largest footprint segment (i.e. wall of the building) within the simulation world. A small offset is applied to this midpoint, moving it away from the building so agents are not instantiated inside it. In Figure 4.10, spawn areas are presented by small blue cubes. Spawn areas are also defined as the goals of the agents, i.e., agents will move between buildings during the simulation.

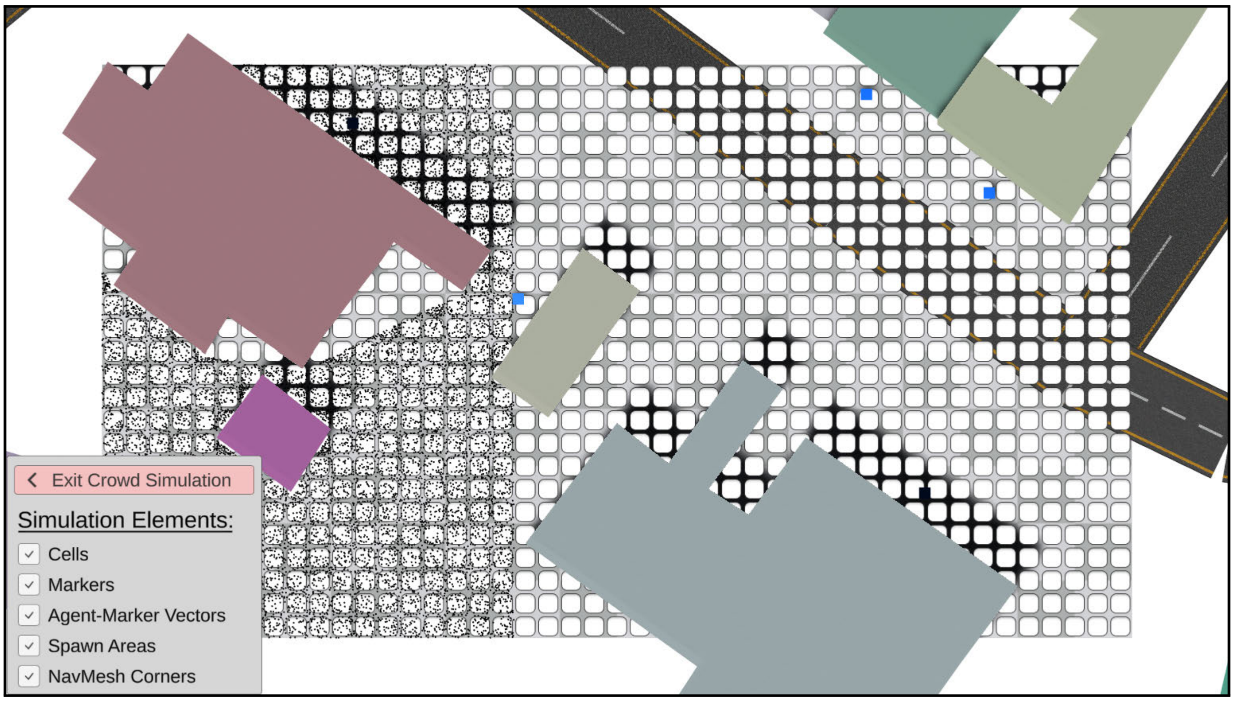
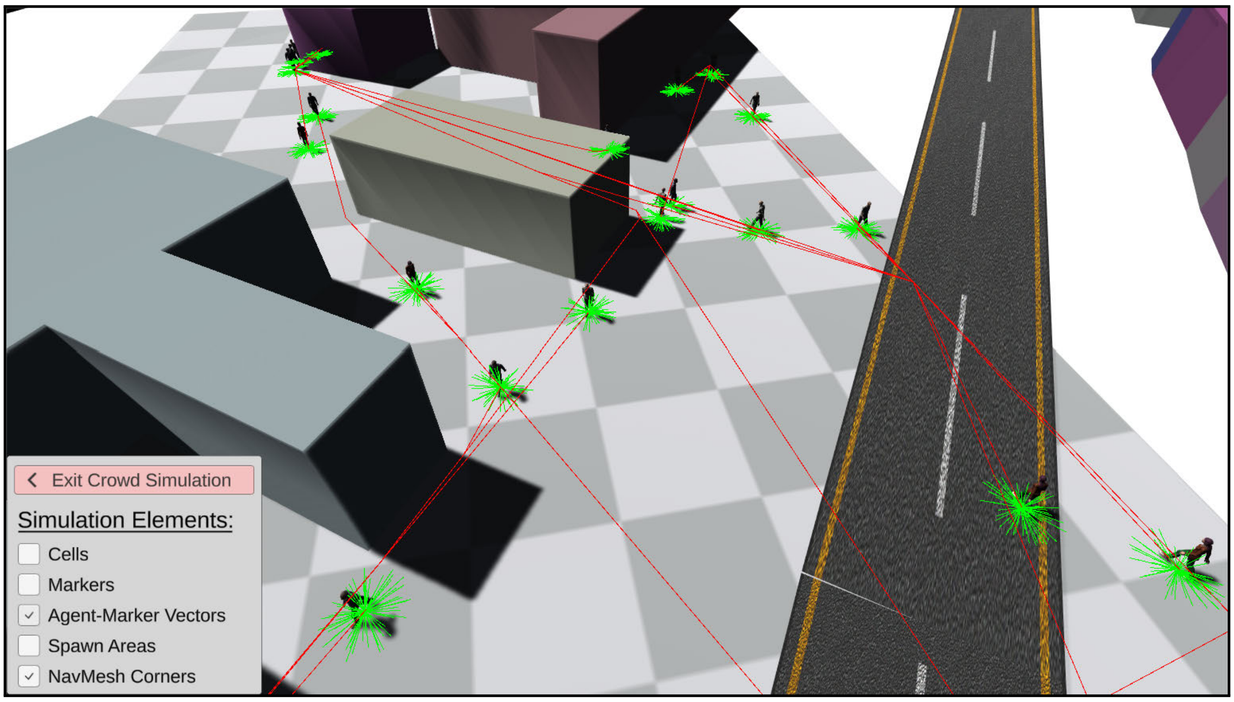
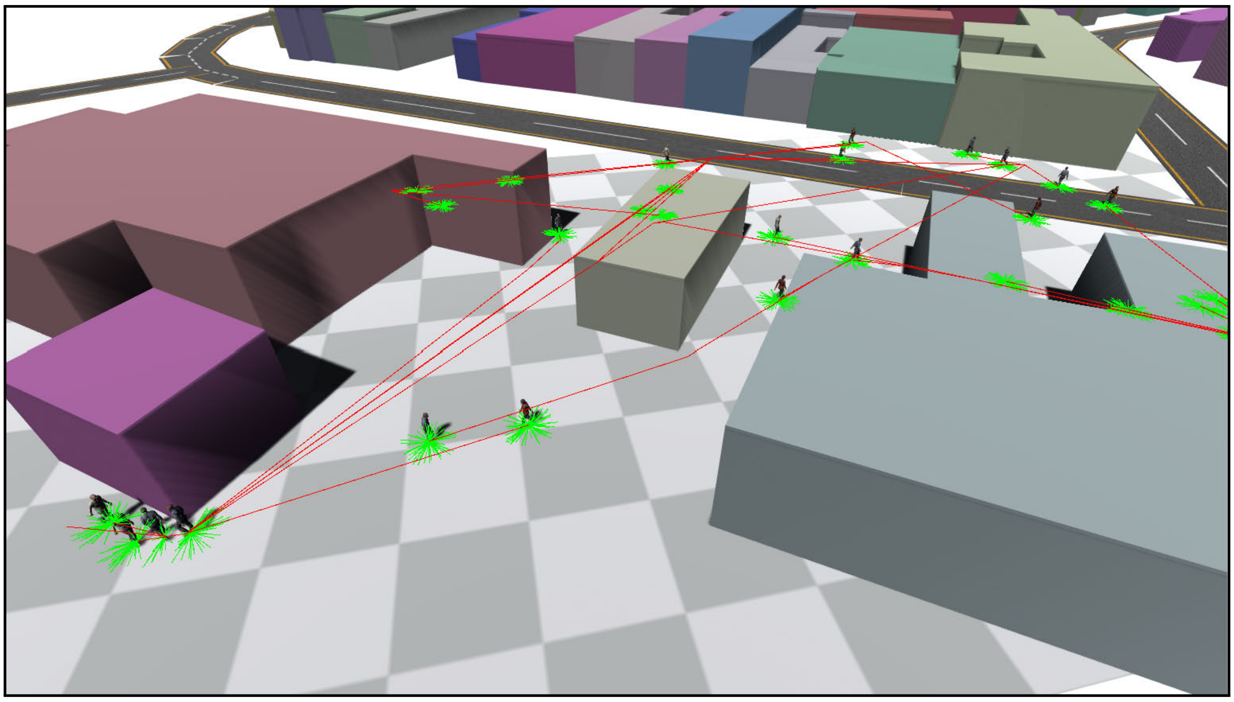


Figure 4.10: Simulation environment. Cells are represented by white quadrilaterals. Markers are represented as black points. Spawn areas are represented by blue cubes.

The user also has access to a small panel in the interface to exit the simulation at any time, returning the camera to its previous position before entering this mode. Also, the user may enable or disable the visualization of the following elements: cells; markers; spawn areas; agent-to-marker vectors; and navigation mesh corners. The "agent-to-marker vectors" are represented as line segments between the virtual agent and its markers in the current frame. The "navigation mesh corners" represent the navigation path the agents will traverse to achieve their goal. Cell, markers, and spawn areas are presented in Figure 4.10. Agent-marker vectors and navigation mesh corners are presented in Figure 4.11.



(a) View of the simulation with navigation mesh and agent-to-marker vectors enabled.



(b) Hidden interface.

Figure 4.11: Virtual agents moving in the multi-level map environment. Green lines represent the agent-to-marker vectors. Red lines represent the navigation path of each agent towards their goal.

4.5 Social Isolation

This section presents an analysis of the obtained data from social isolation, as presented in Section 3.5. The process was applied to the *element* "City of Porto Alegre" and its 94 *regions* (i.e. neighborhoods). The data available ranges from March 1, 2020, to January 20, 2021.

In the isolation data[12], each hexagon is composed of six points that share a single isolation index (ranging from 0 to 1) for a given day. So, the isolation index for a given *region* in a given day is defined as the average between the isolation index of all hexagon points within the *region*'s area. For example, if the *region* "Sarandi" contains three hexagon points within its area, the resulting isolation index is the average index of these three points.

It is possible for a *region* to not have an isolation index for a given date, since a hexagon may not have a valid entry due to a low population count. In these scenarios, we define the isolation index as an average between the index of all neighboring *regions*. In the 326 available dates and 94 neighborhoods, there were 30,644 possible entries. A total of 329 entries required this averaging process (1.073%), as briefly mentioned. Of these 329 cases, 8 occurred with the *region* "Extrema", and the remaining 321 with the *region* "São Caetano". The high frequency of missing data for "São Caetano" is due to the large area and a small population of the neighborhood, presented in Table 4.3.

To analyze the evolution of the isolation index, we selected a group of eight *regions*, listed in Table 4.3. This group contains the five *regions* with the largest population count and the three *regions* with the smallest population count. The selected *regions* present a variety of low and high population densities for both large and small population counts. Additionally, we evaluate the social isolation index of the city, which is defined by the average between the index of all *regions* of Porto Alegre.

Table 4.3: Population Count, Area, and Population Density for the selected neighborhoods.

Neighborhood	**Population Count**	**Area (ha)**	**Density (pop/ha)**
Sarandi	59711	2457.483	24.297
Lomba do Pinheiro	58106	2974.939	19.531
Restinga	53508	2010.277	26.617
Partenon	48160	633.854	75.979
Santa Tereza	39577	451.818	87.595
Farroupilha	961	59.310	16.203
São Caetano	757	830.314	0.911
Pedra Redonda	274	66.616	4.113

Figure 4.12 contains three charts presenting the isolation index of the five largest *regions* (a), the three smallest *regions* (b), and the city average (c). It's possible to observe a pattern in

[12]The isolation data was provided by Inloco (now called Incognia) in a partnership with PUCRS. Incognia website: https://www.incognia.com/

the behavior of the isolation index in Figure 4.12(a), where the value stays at a certain level, sharply increases (up to 0.3 in value), then returns to the previous level.

When analyzed individually, these cycles match the days of a week. Weekdays, where workers and students leave their houses during the day, present a lower isolation index. Weekends, where a larger portion of the population stays home for most of the day, present a higher isolation index. A small variation can be observed between weekdays of a single week, and between the same day in different weeks (e.g. Mondays). This occurs due to the routines of a real-world population not being constant.

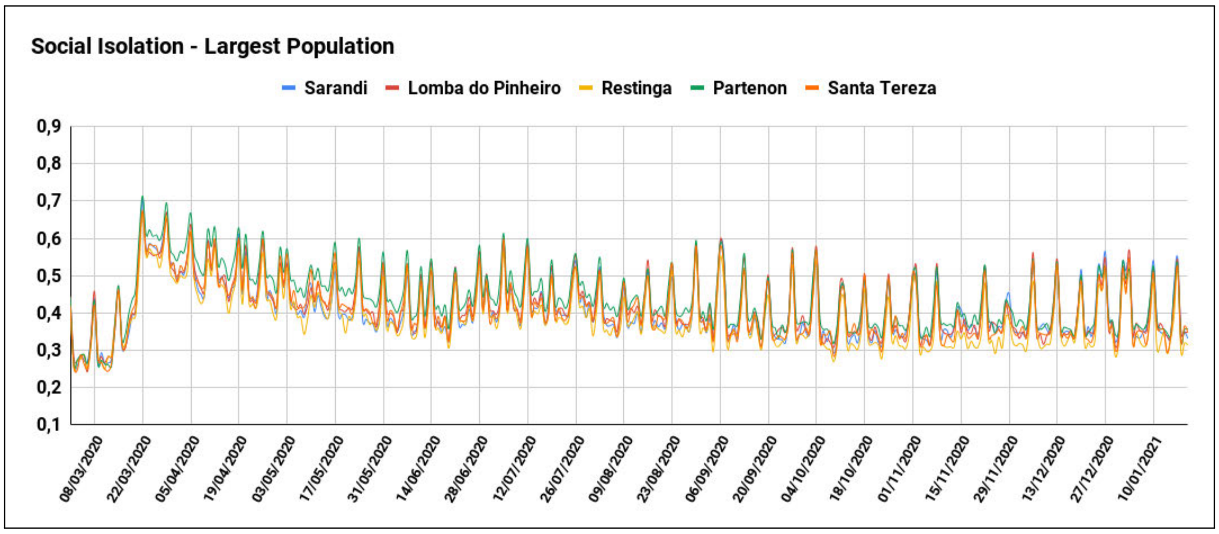
This pattern can also be observed in Figure 4.12(b), although with a higher variation between *regions* when compared to larger populations. The "São Caetano" *region* is an outlier due to the high frequency of unavailable data, as mentioned before. Most of the entries presented are an average of neighboring *regions*, which includes: Lomba do Pinheiro; Lageado; Boa Vista do Sul; Lami; Extrema; and Pitinga. Figure 4.12(c) presents the average isolation index of the city, with results similar to (a). This indicates that, even with the variations between each *region*, the overall city follows the week cycle of higher isolation indexes during weekends when compared to weekdays.

It's important to highlight the real-world events that occurred during the period when the data was collected. The year 2020 was marked by the COVID-19 pandemic, where governments across the world issued restrictions and guidelines to their citizens, aiming to contain the spread of the virus. In many cases, these restrictions included: different opening hours for essential services; temporary closure of schools, universities, and stores; and reduction of max capacity for many spaces there were still open. In Brazil, the first confirmed case occurred in the state of São Paulo on February 28[13]. In Porto Alegre, the first confirmed case occurred on March 11[14]. The Porto Alegre's city hall started enforcing its initial set of guidelines and restrictions on Monday, March 23.

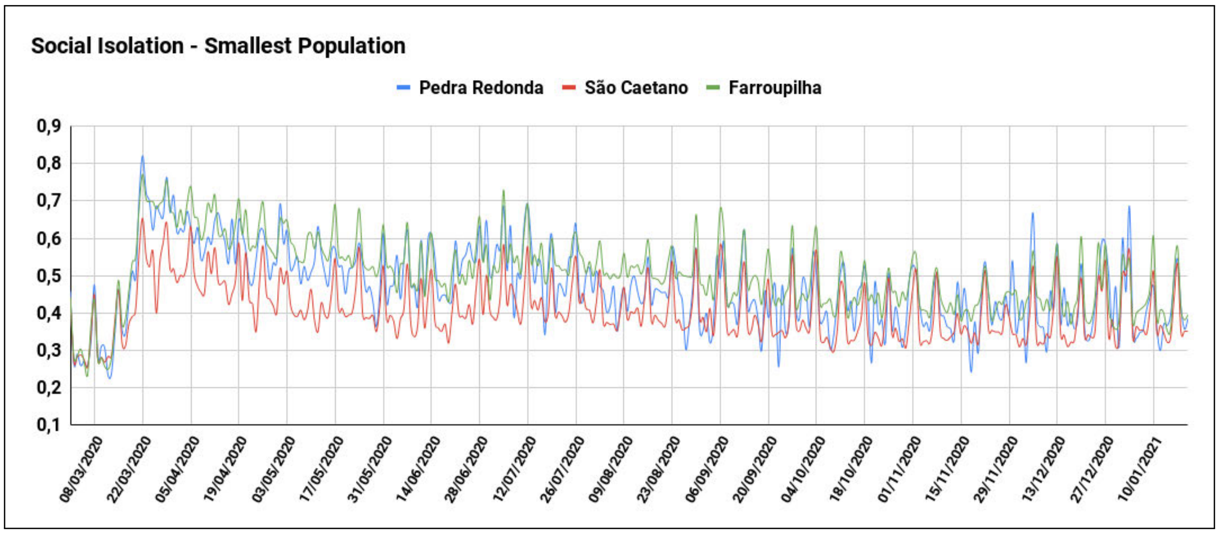
The impact of these events on the isolation index can be observed in the charts in Figure 4.12. The first couple of weeks, where the index is lower across all *regions*, contain data before the initial guidelines and represent a "typical week" of the city. A huge spike can be observed in all *regions* on the weekend before the enforcement of restrictions. The city average value changed from 0.441 on March 20, to 0.586 on March 21, to an all-time high of 0.697 on March 22. The following months presented a gradual reduction in the index baseline. This occurred mainly due to the reduction of restrictions and the reopening of local businesses and public spaces. Another possible factor is a larger part of the population becoming unable to self-quarantine. The index baseline of the last months is still higher than the typical week in March 2020. This indicates that online activities may be more prevalent in the population routine, such as home-office and online classes.

[13]Source of the first case in Brazil: https://www.sanarsaude.com/portal/residencias/artigos-noticias/confirmado-primeiro-caso-de-coronavirus-na-america-latina

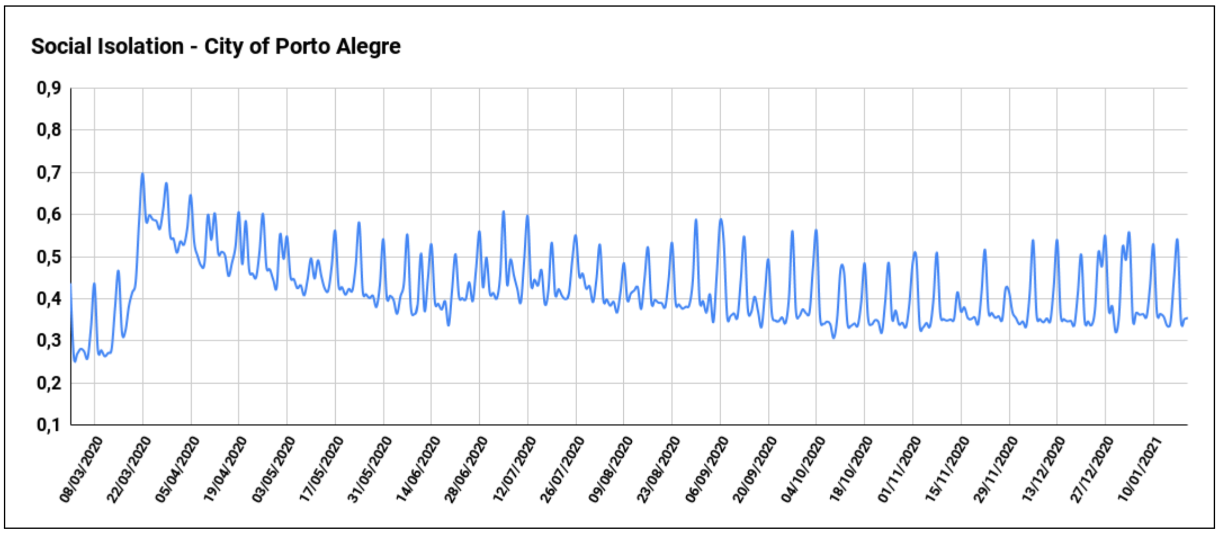
[14]Source of the first case in Porto Alegre: https://prefeitura.poa.br/sms/noticias/prefeitura-confirma-primeiro-caso-de-coronavirus-na-capital



(a) Five largest *regions*, population wise.



(b) Three smallest *regions*, population wise.



(c) Average social isolation in the city of Porto Alegre.

Figure 4.12: Social Isolation Index, from March 1, 2020, to January 20, 2021. The social index of the five neighborhoods with the largest populations is presented in (a). The index of the three neighborhoods with the smallest populations is presented in (b). The average index between all neighborhoods of the city is presented in (c).

The isolation index calculated for all neighborhoods was also used to model a Social Isolation plugin for the multi-level simulation module of LODUS[15]. This plugin affects the overall dynamic of the population being simulated, enforcing social isolation in all *regions*. This is achieved by altering the population being affected by movement actions during the simulation based on the isolation index of the current day. For example, if *region* A request a group of 150 people from *region* B, the total population sent will be reduced by the isolation index of *region* B. Supposing that the index equals 0.3 on the current day of the simulation, the group sent from *region* B towards *region* A will contain 105 people, assuming that *region* B contains the available population of the requested template.

The addition of this plugin allows the user to simulate the overall dynamics of a city, including the small variation found between weekdays, in the dates where data is available. A set of default values can also be used to simulate week cycles following the patterns presented in the charts of Figure 4.12, where the population is more active on a weekday than weekends. As future work, we aim to evaluate the impact of the social isolation index in simulations where a contagion model is also included. A population with a higher social isolation index presents a more restricted dynamic and should, therefore, reduce the spread of the contagion when compared to the isolation index of a typical week.

A visualization mode was added to the multi-level map interface to allow the user to explore and analyze the data of all *regions* for a selected date. This mode is accessible through the dropdown component presented in Figure 4.6. This visualization mode replaces the base color of the *region* based on the social isolation index of the target date. Figure 4.13 presents the color gradient scale used in this mode, while Figure 4.14 presents the Region Isolation visualization mode applied to the "City of Porto Alegre". A sequence of three days of the week (Saturday, Sunday, and Monday) are presented in three different moments of the year 2020. A typical week, before the introduction of any governmental guidelines, is presented in (a), (b), and (c). The first week of enforcement of the initial restrictions is presented in (d), (e), and (f). A week nine months after the initial restrictions is presented in (g), (h), and (i).

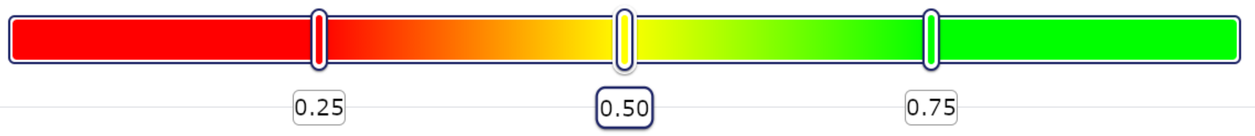
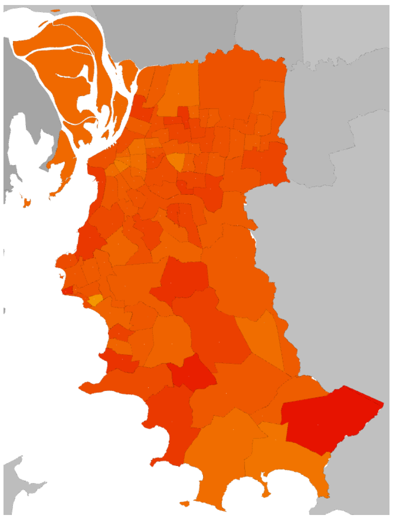
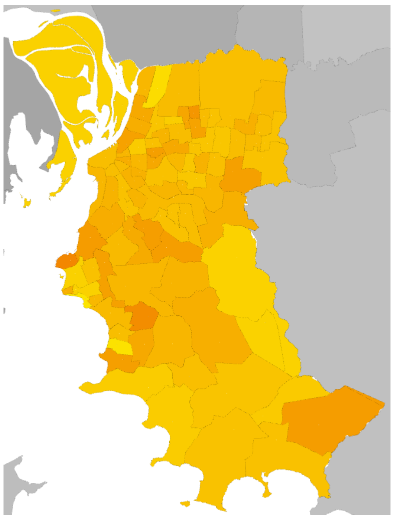
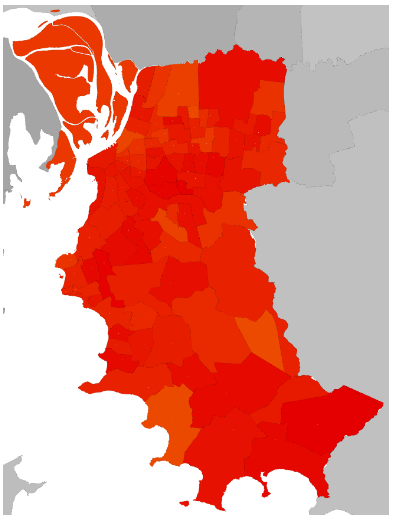


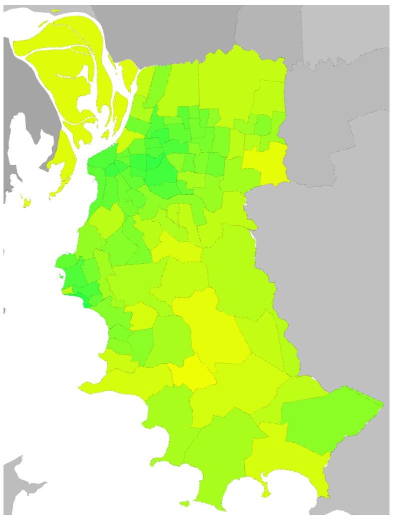
Figure 4.13: Color gradient scale for the isolation index.

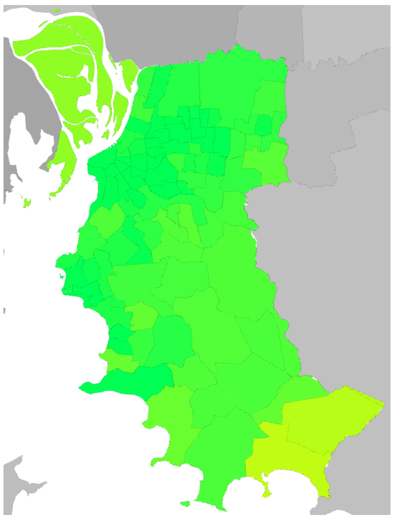
[15]The social isolation plugin was developed in collaboration with André Da Silva Antonitsch.

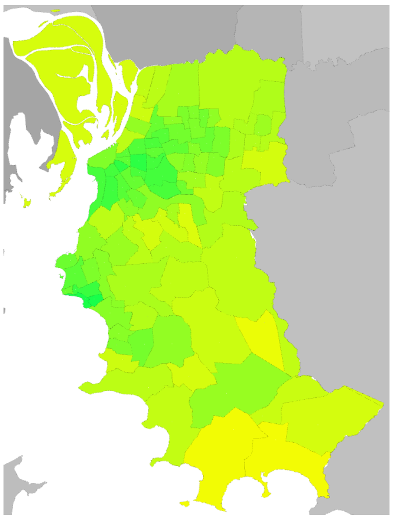

(a) Typical Saturday, March 7, 2020.

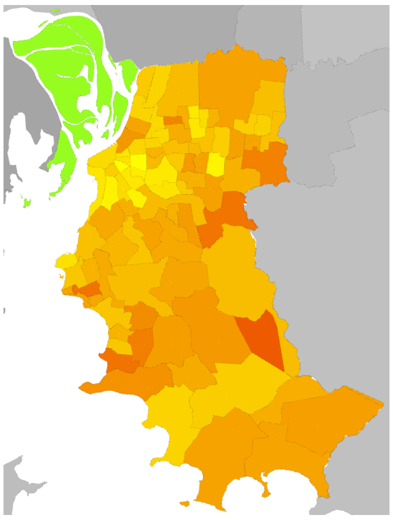

(b) Typical Sunday, March 8, 2020.

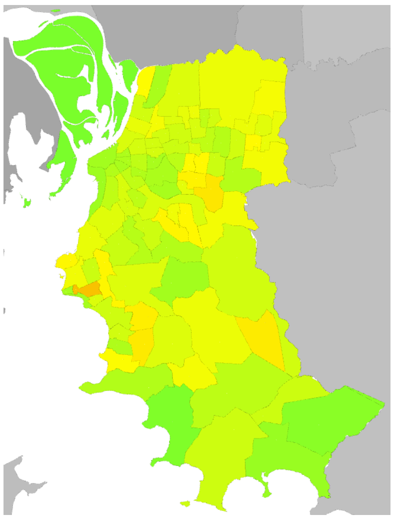

(c) Typical Monday,March 9, 2020.


(d) First Week's Saturday, March 21, 2020.


(e) First Week's Sunday, March 22, 2020.


(f) First Week's Monday, March 23. 2020.


(g) Saturday, December 12.


(h) Sunday, December 13.

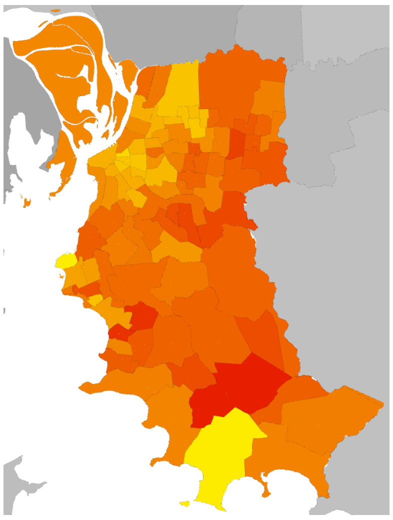

(i) Monday, December 14.

Figure 4.14: Social Isolation in the city of Porto Alegre, divided by neighborhoods. The isolation index for Saturday, Sunday, and Monday in a typical week (before restrictions were introduced by the government) are presented in (a), (b), and (c). The same days of the week in the first week of restrictions are presented in (d), (e), and (f). The isolation index months after the initial restrictions are presented in (g), (h), and (i).

5. FINAL CONSIDERATIONS

In this work, we presented a framework to represent a virtual environment and its population in increasing levels of detail. Our multi-level approach allows the user to hierarchically represent the environment, modeling new spatial structures at each level of this hierarchy. We also presented a method to establish a population template and create routines, allowing us to visualize the movement of groups and individuals in the modeled environment. Finally, an analysis of social isolation data was made, observing patterns in the population movement of a real city used to compose new behaviors for our simulations. Publications made during the progress of this thesis are presented in APPENDIX A, including the first publication of the LODUS Framework.

To validate the capabilities of our framework, we modeled a real-world city in a virtual environment following the proposed methodologies. The multi-level map was constructed using publicly available geographical data, allowing us to effectively represent a large number of spatial structures (e.g. boundaries of states, cities, and neighborhoods) and real-world elements (e.g. roads and buildings). The multi-level structure of the environment also allows us to only include new elements according to the simulations being modeled. This possibility was explored during the composition of a population routine for 13 neighborhoods (of 94 total) in Porto Alegre, where only the selected neighborhoods received a more detailed representation, reducing the scope of the environment.

The multi-level map presents a flexible structure that allows the user to model the levels of abstraction according to the desired experiments and the available data. Although the user has control over this structure, the main constraint of the map, where the elements in a lower level must be contained within the area of the one at a higher level, should be taken into account. In our proof of concept, two additional steps were added to the map's hierarchy between the level of a state and a city. This allowed us to compose a more gradual transition between these levels in the visualization component. Still, new experiments can be explored in future work using different hierarchies to further validate our proposed framework. For example, an experiment focusing on the movement of a population inside a building complex.

We developed a visualization component to allow the user to navigate in the modeled environment and to visualize spatial data. Different spatial structures can be explored, presenting information regarding their regions, connections, and any additional data available (e.g. bus routes, social isolation, and population count). This resource also allows the representation of the population in two levels of detail: as population groups following the routines created with the population setup tool; and as individuals represented by virtual agents moving through small sections of the environment. New functionalities can be explored to establish a more detailed representation of the environment and its population and to provide a better tool for the user when analyzing spatial data. Additionally, we contributed in the creation of a Social Isolation plugin for the multi-level simulation module of LODUS, allowing us to affect the overall movement of the population based on the isolation index of the modeled regions.

5.1 Limitations

The current implementation of LODUS presents a series of limitations. Regarding the multi-level map, we have implemented tools to obtain data from different file formats to compose our proposed structures, including obtaining spatial structures from GeoJSON files; relationships and population data of regions from CSV files; roads, blocks, and buildings from OSM files. However, this process of constructing the map needs to be better streamlined so the user can create new environments faster when following some guidelines. Additionally, other file formats can be supported in future implementation to provide more flexibility for the user.

The population setup module also presents limited capabilities when modeling routines for a large number of regions or complex behaviors containing many events and parameters for these events. New features can be explored to assist the user when composing such routines. Additionally, the multi-level simulation module is currently not fully integrated with the visualization component. This requires the user to execute the routines outside the visualization component and utilize the output data to visualize the movement of population groups, such as presented in Section 4.4.1. We aim to fully integrate the visualization component and the simulation module, so the user can model the environment, the population routines, execute the simulation and visualize the spatial data in the same application.

There are also limitations regarding the availability of data in many stages of LODUS. Governmental boundaries data (e.g. countries, cities, and neighborhoods) is available from multiple sources, which can be used to model the higher levels of abstraction. Data regarding city blocks, however, is more sparse since their boundaries are clearer mostly in urbanized areas with a defined road network. Data regarding the population can be obtained through a census and additional surveys. This data is commonly grouped by city or neighborhood, limiting the levels of abstraction which will contain information for individual regions. In our proof of concept, for example, the population data of an individual building can only be estimated based on the information available for the entire neighborhood.

The unavailability of data also presents a challenge when validating our framework and the results of simulations. Since data regarding the population is limited to a certain granularity, it is currently unfeasible to precisely model the routines of individuals in lower levels of abstraction, such as a neighborhood. Also, with increasing concern of data privacy and the protection of individuals, the information available in many datasets presents only averages and/or approximations. To validate the results in the movement of population groups in Porto Alegre, for example, the origin-destination matrix between the modeled regions is required. However, this data is not currently available to us.

5.2 Future Work

Many possibilities can be explored in future work. Regarding the multi-level map, new experiments can be modeled to explore lower levels of abstraction, currently not covered in our pipeline implementation. A further study on procedural generation methods for urban elements, e.g., the creation of road networks presented by Dou et al. [12] and the creation of building geometries presented by Li et al. [24], can be made aiming to implement the proposed techniques in our pipeline. This would allow the user to create and model the content required for new desired levels when the necessary information is not available or is not based on real-world geographic information. Regarding the population setup tool, improvements and new features can be explored to help the user when modeling larger and/or more complex routines. Also, we aim to fully integrate the functionalities of the visualization component and the simulation module, as described previously.

Usability tests should be conducted with users to validate our framework and the overall modeling process, evaluating its applicability when creating different environments. These tests would include the construction of the map based on geographical data; the definition of population routines; the execution of the simulation; and the visualization of spatial data, also evaluating the interface of the visualization component. Finally, considering the current contributions and applicability in real-world scenarios, we can explore the possibility to make LODUS available as a commercial tool, aiming to assist city planners in their decision-making when modeling new areas and analyzing spatial data.

BIBLIOGRAPHY

[1] Aliaga, D. G.; Beneš, B.; Vanegas, C. A.; Andrysco, N. "Interactive reconfiguration of urban layouts", *IEEE Computer Graphics and Applications*, vol. 28–3, Jun 2008, pp. 38–47.

[2] Antonitsch, A. D. S.; Musse, S. R.; Figueiredo, L. H. D. "Towards a legion of virtual humans: Steering behaviors and organic visualization". In: Proceedings of the Conference on Graphics, Patterns and Images (SIBGRAPI), 2020, pp. 31–38.

[3] Antonitsch, A. D. S.; Schaffer, D. H. M.; Rockenbach, G. W.; Knob, P.; Musse, S. R. "Bioclouds: A multi-level model to simulate and visualize large crowds". In: Proceedings of the Computer Graphics International Conference (CGI), 2019, pp. 15–27.

[4] Bekins, D.; Aliaga, D. G. "Build-by-number: Rearranging the real world to visualize novel architectural spaces". In: Proceedings of the IEEE Visualization Conference (VIS), 2005, pp. 143–150.

[5] Beneš, J.; Wilkie, A.; Křivánek, J. "Procedural modelling of urban road networks", *Computer Graphics Forum*, vol. 33–6, Sep 2014, pp. 132–142.

[6] Bicho, A. L. "Da modelagem de plantas à dinâmica de multidões: Um modelo de animação comportamental bio-inspirado", PhD thesis, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, 2009, 114p.

[7] Biljecki, F.; Ledoux, H.; Du, X.; Stoter, J.; Soon, K. H.; Khoo, V. H. S. "The most common geometric and semantic errors in CityGML datasets", *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. IV-2/W1, Oct 2016, pp. 13–22.

[8] Billen, R.; Cutting-Decelle, A.-F.; Métral, C.; Falquet, G.; Zlatanova, S.; Marina, O. "Challenges of semantic 3D city models: A contribution of the COST research action TU0801", *International Journal of 3-D Information Modeling (IJ3DIM)*, vol. 4–2, Apr 2015, pp. 68–76.

[9] Chen, G.; Esch, G.; Wonka, P.; Müller, P.; Zhang, E. "Interactive procedural street modeling", *ACM Transactions on Graphics*, vol. 27–3, Aug 2008, pp. 1–10.

[10] Cogo, E.; Prazina, I.; Hodzic, K.; Haseljic, H.; Rizvic, S. "Survey of integrability of procedural modeling techniques for generating a complete city". In: Proceedings of the International Conference on Information, Communication and Automation Technologies (ICAT), 2019, pp. 1–6.

[11] Demir, C.; Koramaz, T. K. "GIS-based procedural modeling in contemporary urban planning practice". In: Proceedings of the International Conference Information Visualisation (IV), 2018, pp. 553–560.

[12] Dou, X.; Qi, Y.; Hou, F.; Shen, X. "Interactive urban map design with template and parameterization". In: Proceedings of the International Congress on Image and Signal Processing (ICISP), 2009, pp. 1–5.

[13] Eid, M.; Eldin, H. "Simulation models as a tool for urban planning", *Computers & Industrial Engineering*, vol. 4–1, Dec 1980, pp. 53–66.

[14] Emilien, A.; Bernhardt, A.; Peytavie, A.; Cani, M.-P.; Galin, E. "Procedural generation of villages on arbitrary terrains", *The Visual Computer*, vol. 28, Jun 2012, pp. 809–818.

[15] Fan, L.; Wonka, P. "A probabilistic model for exteriors of residential buildings", *ACM Transactions on Graphics*, vol. 35–5, Jul 2016, pp. 1–13.

[16] Fiser, M.; Benes, B.; Galicia, J. G.; Abdul-Massih, M.; Aliaga, D. G.; Krs, V. "Learning geometric graph grammars". In: Proceedings of the Spring Conference on Computer Graphics (SCCG), 2016, pp. 7–15.

[17] Gang, L.; Guangshun, S. "Procedural modeling of urban road network". In: Proceedings of the International Forum on Information Technology and Applications (IFITA), 2010, pp. 75–79.

[18] Garcia-Dorado, I.; Aliaga, D. G.; Bhalachandran, S.; Schmid, P.; Niyogi, D. "Fast weather simulation for inverse procedural design of 3D urban models", *ACM Transactions on Graphics*, vol. 36–2, Apr 2017, pp. 1–19.

[19] Garg, D.; Chli, M.; Vogiatzis, G. "Traffic3D: A new traffic simulation paradigm". In: Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), 2019, pp. 2354–2356.

[20] Greuter, S.; Parker, J.; Stewart, N.; Leach, G. "Real-time procedural generation of 'pseudo infinite' cities". In: Proceedings of the International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia (GRAPHITE), 2003, pp. 87–ff.

[21] Handy, S. L.; Boarnet, M. G.; Ewing, R.; Killingsworth, R. E. "How the built environment affects physical activity: Views from urban planning", *American Journal of Preventive Medicine*, vol. 23–2, Supplement 1, Sep 2002, pp. 64–73.

[22] Hastings, W. D. "Monte carlo sampling methods using markov chains and their applications", *Biometrika*, vol. 57, Apr 1970, pp. 97–109.

[23] Hughes, R. L. "A continuum theory for the flow of pedestrians", *Transportation Research Part B: Methodological*, vol. 36–6, Jul 2002, pp. 507–535.

[24] Li, J.; Zhang, Y.; Kong, D. "Rule-based procedural modeling of buildings". In: Proceedings of the IEEE International Conference on Computer Science and Automation Engineering (CSAE), 2012, pp. 450–454.

[25] Lindenmayer, A. "Mathematical models for cellular interactions in development I. Filaments with one-sided inputs", *Journal of Theoretical Biology*, vol. 18–3, Apr 1968, pp. 280–299.

[26] Liu, K.; Chen, J.; Wang, S.; Zhu, X. "Procedural modeling of buildings based on façade image segmentation". In: Proceedings of the International Conference on Audio, Language and Image Processing (ICALIP), 2014, pp. 797–801.

[27] Martinovic, A.; Van Gool, L. "Bayesian grammar learning for inverse procedural modeling". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 201–208.

[28] Marvie, J.-E.; Perret, J.; Bouatouch, K. "FL-system : A functional L-system for procedural geometric modeling", *The Visual Computer*, vol. 21, Jun 2005, pp. 329–339.

[29] Mathew, C. D. T.; Knob, P. R.; Musse, S. R.; Aliaga, D. G. "Urban walkability design using virtual population simulation", *Computer Graphics Forum*, vol. 38–1, Oct 2019, pp. 455–469.

[30] Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M. N.; Teller, A. H.; Teller, E. "Equation of state calculations by fast computing machines", *The Journal of Chemical Physics*, vol. 21–6, Jan 1953, pp. 1087–1092.

[31] Müller, P.; Wonka, P.; Haegler, S.; Ulmer, A.; Van Gool, L. "Procedural modeling of buildings", *ACM Transactions of Graphics*, vol. 25–3, Jul 2006, pp. 614–623.

[32] Mustafa, A. M.; Zhang, X. W.; Aliaga, D. G.; Bruwier, M.; Nishida, G.; Dewals, B.; Erpicum, S.; Archambeau, P.; Pirotton, M.; Teller, J. "Procedural generation of flood-sensitive urban layouts", *Environment and Planning B: Urban Analytics and City Science*, vol. 47, Nov 2018, pp. 889–911.

[33] Ng, K. M.; Reaz, M. I. "Development and validation of the lwr-im traffic simulator". In: Proceedings of the IEEE International Conference on Control System, Computing and Engineering (ICCSCE), 2017, pp. 40–44.

[34] Parish, Y. I. H.; Müller, P. "Procedural modeling of cities". In: Proceedings of the ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), 2001, pp. 301–308.

[35] Pelechano, N.; Allbeck, J. M.; Badler, N. I. "Controlling individual agents in high-density crowd simulation". In: Proceedings of the Symposium on Computer Animation (SCA), 2007, pp. 99–108.

[36] Reynolds, C. W. "Flocks, herds and schools: A distributed behavioral model". In: Proceedings of the ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), 1987, pp. 25–34.

[37] Rodrigues, R.; Coelho, A.; Reis, L. P. “Data model for procedural modelling from textual descriptions”. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 2010, pp. 1–8.

[38] Sallis, J. F.; Frank, L. D.; Saelens, B. E.; Kraft, M. “Active transportation and physical activity: opportunities for collaboration on transportation and public health research”, *Transportation Research Part A: Policy and Practice*, vol. 38–4, Feb 2004, pp. 249–268.

[39] Schwarz, M.; Müller, P. “Advanced procedural modeling of architecture”, *ACM Transactions on Graphics*, vol. 34–4, Jul 2015, pp. 1–12.

[40] Silva, G. F.; Cassol, V.; Neto, A. B. F.; Antonitsch, A.; Schaffer, D.; Musse, S. R.; Linn, R. M. “Lodus: A multi-level framework for simulating environment and population - a contagion experiment on a pandemic world”. In: Proceedings of the IEEE International Smart Cities Conference (ISC2), 2020, pp. 1–8.

[41] Stiny, G.; Gips, J. “Shape grammars and the generative specification of painting and sculpture”. In: Proceedings of the International Federation for Information Processing Congress (IFIP), 1971, pp. 1460–1465.

[42] Talton, J. O.; Lou, Y.; Lesser, S.; Duke, J.; Měch, R.; Koltun, V. “Metropolis procedural modeling”, *ACM Transactions on Graphics*, vol. 30–2, Apr 2011, pp. 1–14.

[43] Teng, E.; Bidarra, R. “A semantic approach to patch-based procedural generation of urban road networks”. In: Proceedings of the International Conference on the Foundations of Digital Games (FDG), 2017, pp. 1–10.

[44] Treuille, A.; Cooper, S.; Popović, Z. “Continuum crowds”, *ACM Transactions on Graphics*, vol. 25–3, Jul 2006, pp. 1160–1168.

[45] Vanegas, C. A.; Aliaga, D. G.; Benes, B.; Waddell, P. A. “Interactive design of urban spaces using geometrical and behavioral modeling”, *ACM Transactions on Graphics*, vol. 28–5, Dec 2009, pp. 1–10.

[46] Vanegas, C. A.; Garcia-Dorado, I.; Aliaga, D. G.; Benes, B.; Waddell, P. “Inverse design of urban procedural models”, *ACM Transactions on Graphics*, vol. 31–6, Nov 2012, pp. 1–11.

[47] Wonka, P.; Wimmer, M.; Sillion, F.; Ribarsky, W. “Instant architecture”, *ACM Transactions on Graphics*, vol. 22, Jul 2003, pp. 669–677.

[48] Yang, Y.-L.; Wang, J.; Vouga, E.; Wonka, P. “Urban pattern: Layout design by hierarchical domain splitting”, *ACM Transactions on Graphics*, vol. 32–6, Nov 2013, pp. 1–12.

[49] Zmugg, R.; Thaller, W.; Krispel, U.; Edelsbrunner, J.; Havemann, S.; Fellner, D. W. “Deformation-aware split grammars for architectural models”. In: Proceedings of the International Conference on Cyberworlds, 2013, pp. 4–11.

APPENDIX A – PAPERS WRITTEN DURING THE MASTER'S DEGREE

- SILVA, G. F.; CASSOL, V.; NETO, A. B. F.; ANTONITSCH, A.; SCHAFFER, D.; MUSSE, S. R.; LINN, R. M. "LODUS: A multi-level framework for simulating environment and population - a contagion experiment on a pandemic world". In: 2020 IEEE International Smart Cities Conference (ISC2), Piscataway, NJ, USA, 2020, pp. 1–8. https://doi.org/10.1109/ISC251055.2020.9239083 *(Published Paper)*
- SILVA, G. F.; KNOB, P.; SCHLATTER, D. A.; JOHANSSON, C. G.; MUSSE, S. R. "Moving virtual agents forward in space and time". In: 2020 19th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), Recife, Brazil, 2020, pp. 1–10. https://doi.org/10.1109/SBGames51465.2020.00026 *(Published Paper)*

APPENDIX B – POPULATION SETUP FILES

This appendix presents the files used in the Population Setup Tool to model the routines presented in Section 4.4.1. Only a part of the files is presented, focusing on the *region* "Azenha" as an example. The remaining *regions* follow the same structure, with numbers adjusted to their respective populations. The complete files were made available through GitHub[1]. All files use the JSON (JavaScript Object Notation) format.

Two files are required as input: one describing the *regions* of the environment (i.e. *regions graph* data), and one describing the population template to be simulated. Table B.1 presents the data used to describe the *regions*, including an ID; the *region*'s name; a list of neighbors' IDs; the total population of the *region*; a latitude-longitude position; and a list of available nodes. A JSON file was composed using this data, presented in Figure B.1.

Table B.1: Data of the simulated *regions*.

ID	RegionName	NeighborIDs	Population	Longitude	Latitude	Nodes
1	Azenha	4;5;13;8	13804	-51.2147436812232	-30.0510199525848	home,work,school
2	Bom Fim	7;11;12;13;5;3;9	11593	-51.2104807594223	-30.0330589939714	home,work,school
3	Centro	6;7;2;5;4;10	39154	-51.2278480810808	-30.031004727877	home,work,school
4	Cidade Baixa	3;5;13;1;8;10	15379	-51.2224525885919	-30.0404993243547	home,work,school
5	Farroupilha	2;13;1;4;3;11;12	961	-51.2136185372287	-30.038368061923	home,work,school
6	Floresta	9;7;3	11596	-51.2120255115732	-30.0218580655156	home,work,school
7	Independência	6;9;11;2;3	8112	-51.2114165227445	-30.0283138642156	home,work,school
8	Menino Deus	4;1;10	31650	-51.222051547168	-30.0529677373194	home,work,school
9	Moinhos de Vento	6;11;7;2	11937	-51.202415370788	-30.0237549822533	home,work,school
10	Praia de Belas	3;4;8	2281	-51.230172636	-30.0472881985923	home,work,school
11	Rio Branco	7;9;12;13;2;5	17531	-51.2019103556617	-30.0331907998885	home,work,school
12	Santa Cecília	2;11;13;5	5768	-51.201331073605	-30.0408947118911	home,work,school
13	Santana	2;12;1;4;5;11	20723	-51.2055865687131	-30.046728905517	home,work,school

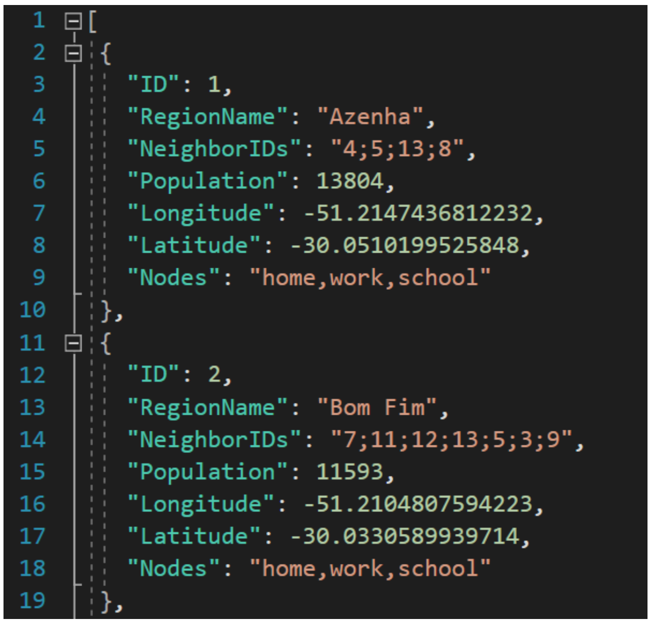
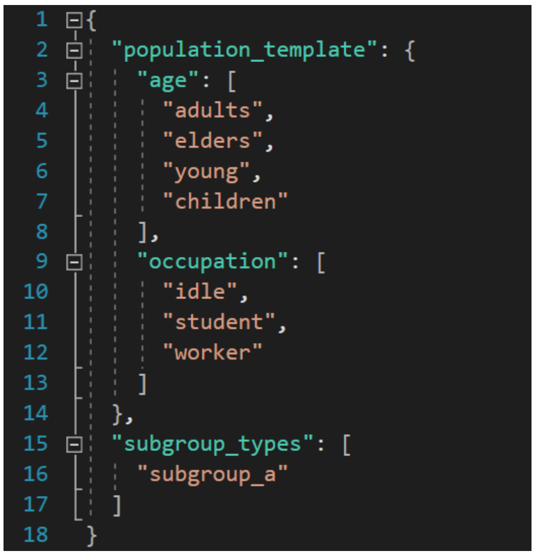

```
[
  {
    "ID": 1,
    "RegionName": "Azenha",
    "NeighborIDs": "4;5;13;8",
    "Population": 13804,
    "Longitude": -51.2147436812232,
    "Latitude": -30.0510199525848,
    "Nodes": "home,work,school"
  },
  {
    "ID": 2,
    "RegionName": "Bom Fim",
    "NeighborIDs": "7;11;12;13;5;3;9",
    "Population": 11593,
    "Longitude": -51.2104807594223,
    "Latitude": -30.0330589939714,
    "Nodes": "home,work,school"
  },
```

Figure B.1: Data of the simulated *regions* in JSON format. Data of the "Azenha" and "Bom Fim" *regions* are presented.

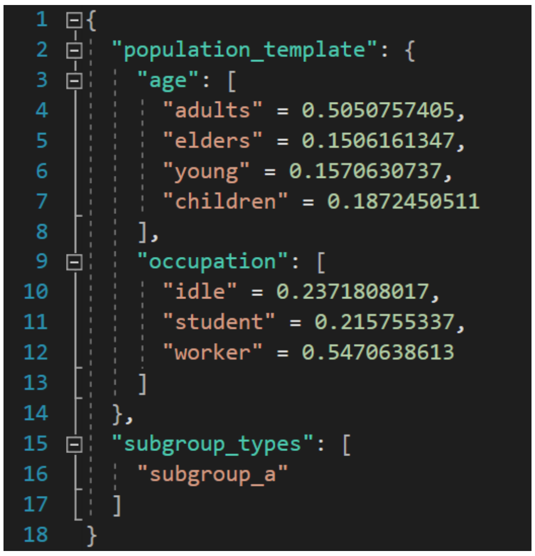
[1]The complete files are available at: https://github.com/GabrielFSilva/PopulationSetupFiles

Regarding the population template, the input file must contain the desired properties with their respective values, and the list of subgroup types. Additionally, the proportions of each value can be informed. These proportions are applied to all *regions* by default and can be individually (i.e. by *region*) edited when using the tool. When a proportion is not informed, they will be evenly distributed for the respective property. Figure B.2 presents two possible options for the population template file. Proportions presented in (b) were used to describe the population in the simulation. The data sources of these proportions were presented previously in Section 4.4.1.



```
{
  "population_template": {
    "age": [
      "adults",
      "elders",
      "young",
      "children"
    ],
    "occupation": [
      "idle",
      "student",
      "worker"
    ]
  },
  "subgroup_types": [
    "subgroup_a"
  ]
}
```

(a) Population template without proportions.



```
{
  "population_template": {
    "age": [
      "adults" = 0.5050757405,
      "elders" = 0.1506161347,
      "young" = 0.1570630737,
      "children" = 0.1872450511
    ],
    "occupation": [
      "idle" = 0.2371808017,
      "student" = 0.215755337,
      "worker" = 0.5470638613
    ]
  },
  "subgroup_types": [
    "subgroup_a"
  ]
}
```

(b) Population template with proportions.

Figure B.2: Population template files. Two options are presented: one without proportions in (a); and with proportions in (b). When informed, proportions are defined for all *regions*.

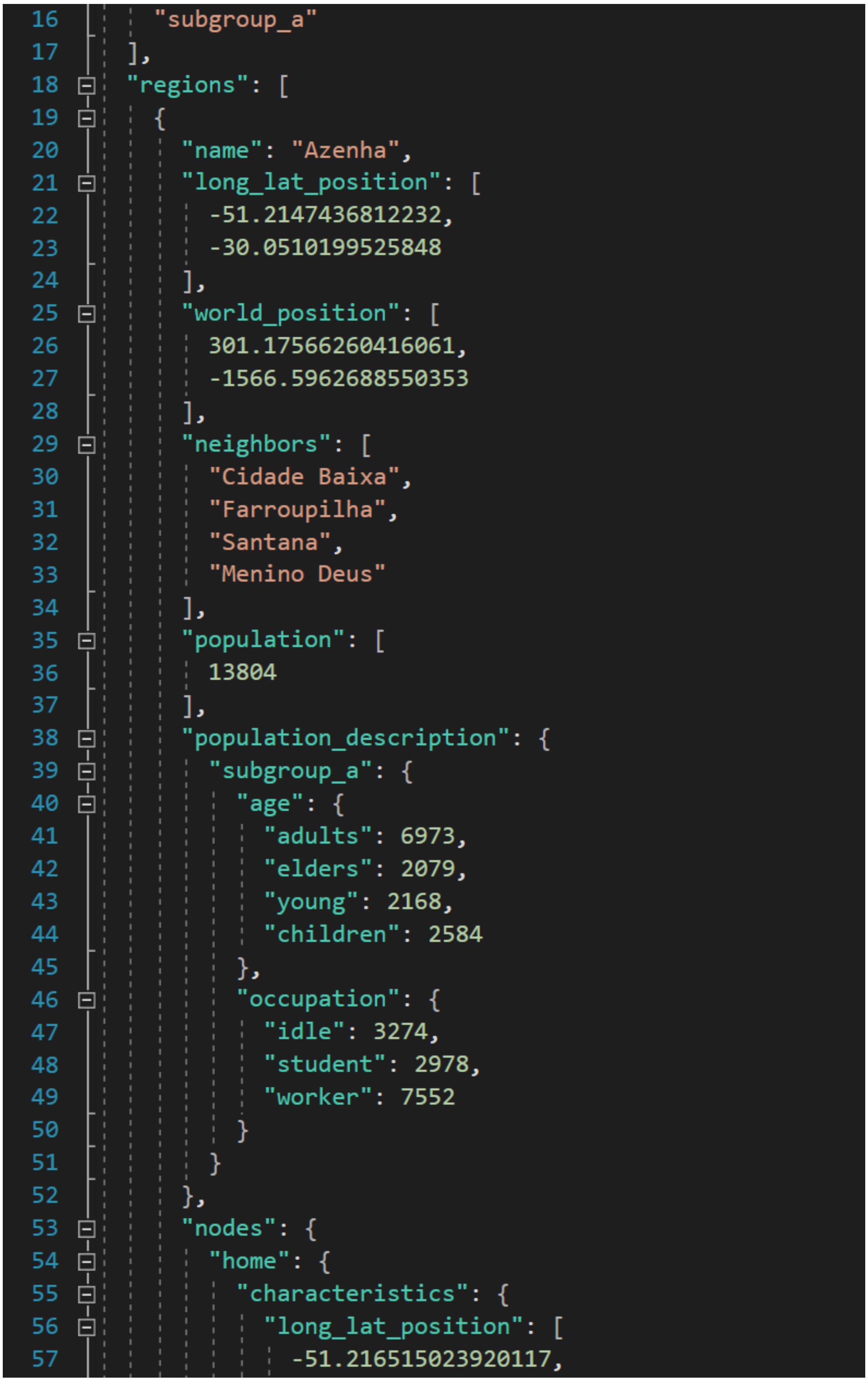
After the creation of the routines, a output file is generated to be used in the simulator. This file contains the population template without proportions and a list of *regions* and their respective events. The following figures present the entire description of the *region* "Azenha". Remaining *regions* follow the same structure and are available in the complete files at GitHub[2].

Figure B.3 presents the data describing the *region* "Azenha". The "population" attribute describes the total population count of each subgroup. Since our template only contains one type, a single value is presented. The "population_description" attribute describes the population distribution for all available subgroups.

Figure B.4 presents two events of the node "Home". The "return_population_home" type signals people of a given "population_template" to return to the "Home" node at their origin *region*. The value -1 in "quantity" indicates that all available population will be affected.

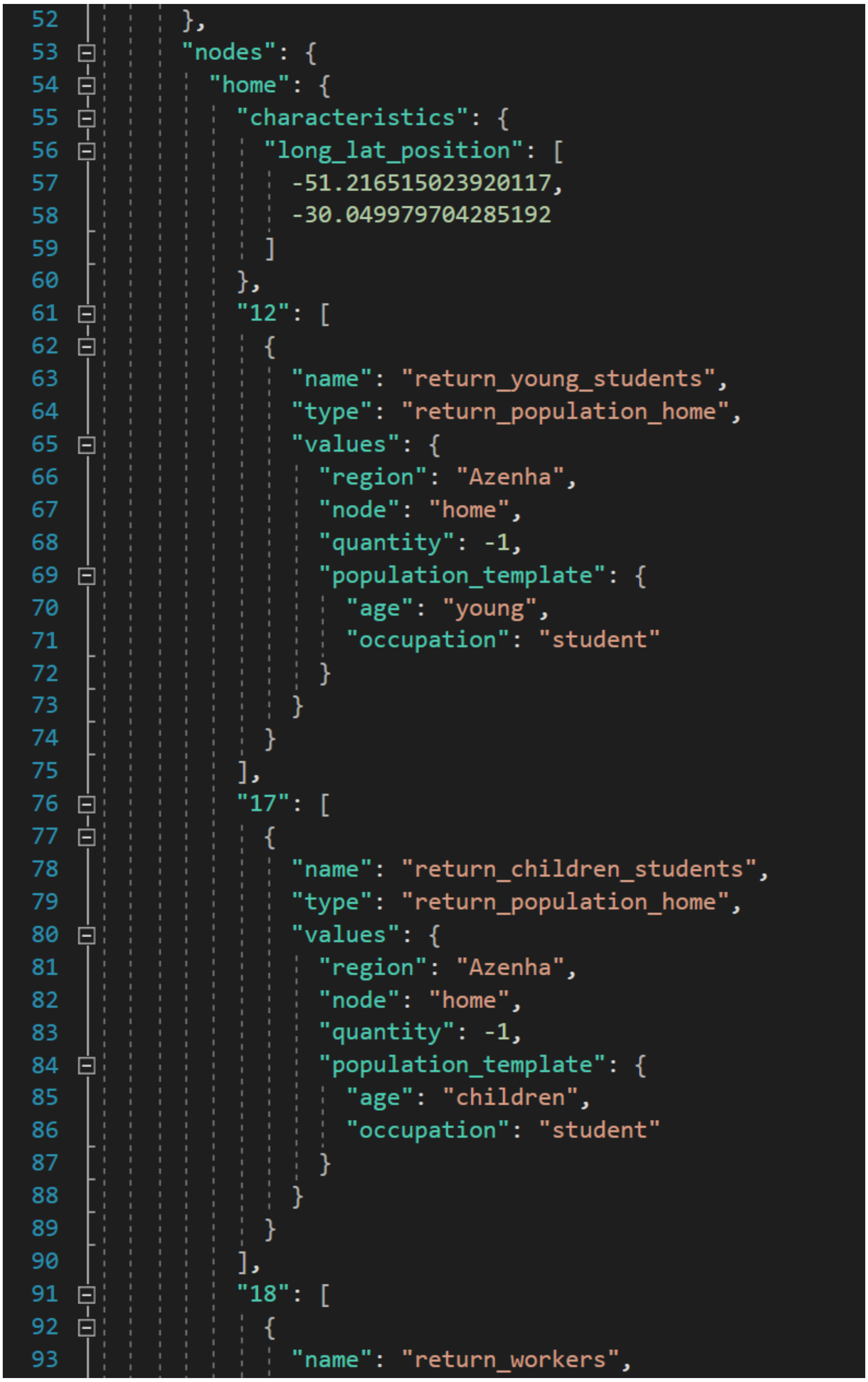
Figure B.5 presents the third event of the node "Home" and the only event of "Work". The "gather_population" event requests a population from neighboring *regions*. The *region* itself may also send people to the target node. Figure B.6 presents the two events of the node "School", requesting "Young-Students" and "Children-Students" respectively.

[2]The complete files are available at: https://github.com/GabrielFSilva/PopulationSetupFiles



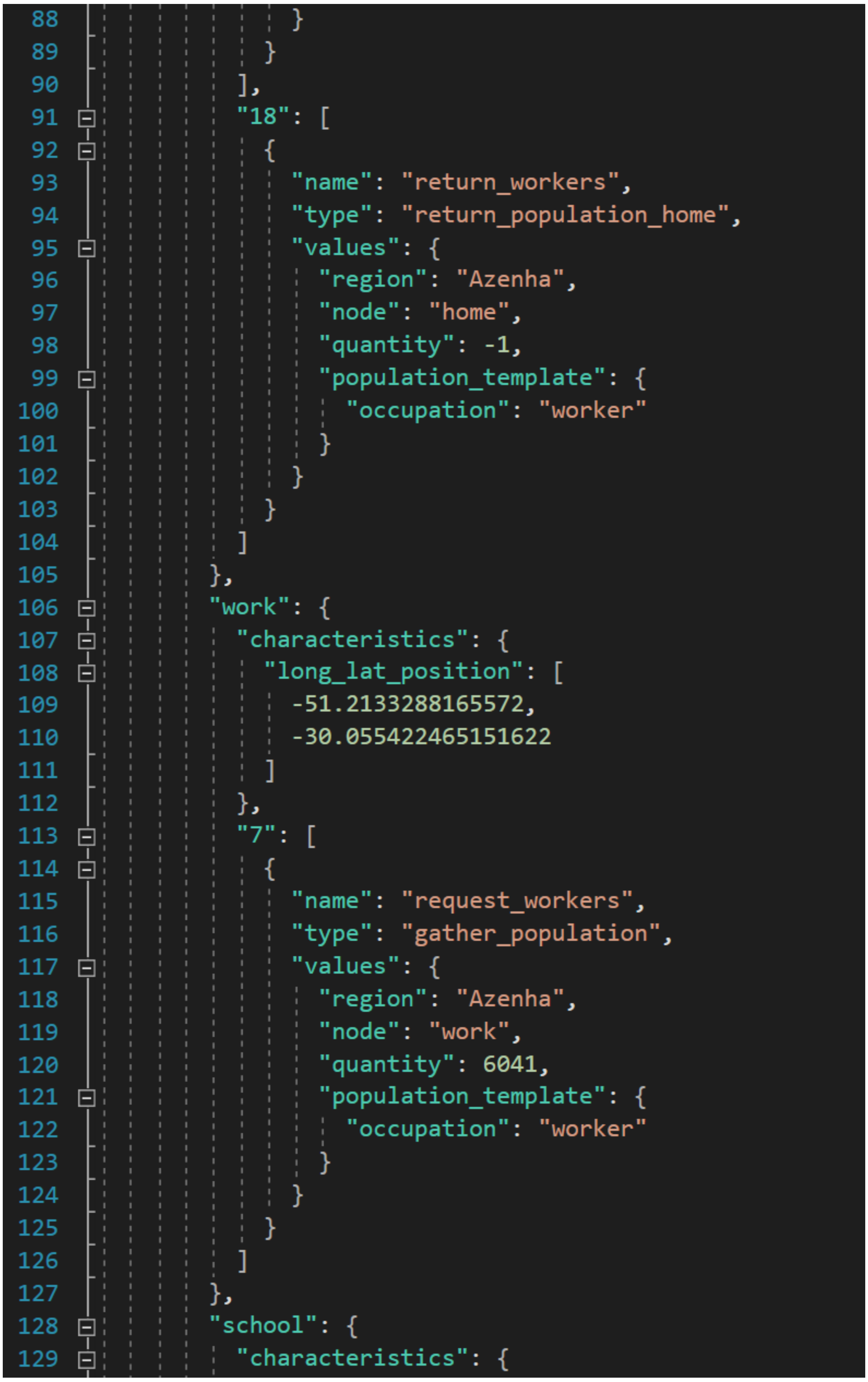
```
      "subgroup_a"
    ],
    "regions": [
      {
        "name": "Azenha",
        "long_lat_position": [
          -51.2147436812232,
          -30.0510199525848
        ],
        "world_position": [
          301.17566260416061,
          -1566.5962688550353
        ],
        "neighbors": [
          "Cidade Baixa",
          "Farroupilha",
          "Santana",
          "Menino Deus"
        ],
        "population": [
          13804
        ],
        "population_description": {
          "subgroup_a": {
            "age": {
              "adults": 6973,
              "elders": 2079,
              "young": 2168,
              "children": 2584
            },
            "occupation": {
              "idle": 3274,
              "student": 2978,
              "worker": 7552
            }
          }
        },
        "nodes": {
          "home": {
            "characteristics": {
              "long_lat_position": [
                -51.216515023920117,
```

Figure B.3: Data and population distribution of the *region* "Azenha". The "population" attribute describes the total population count of each subgroup. The "population_description" attribute describes the population distribution for all available subgroups.



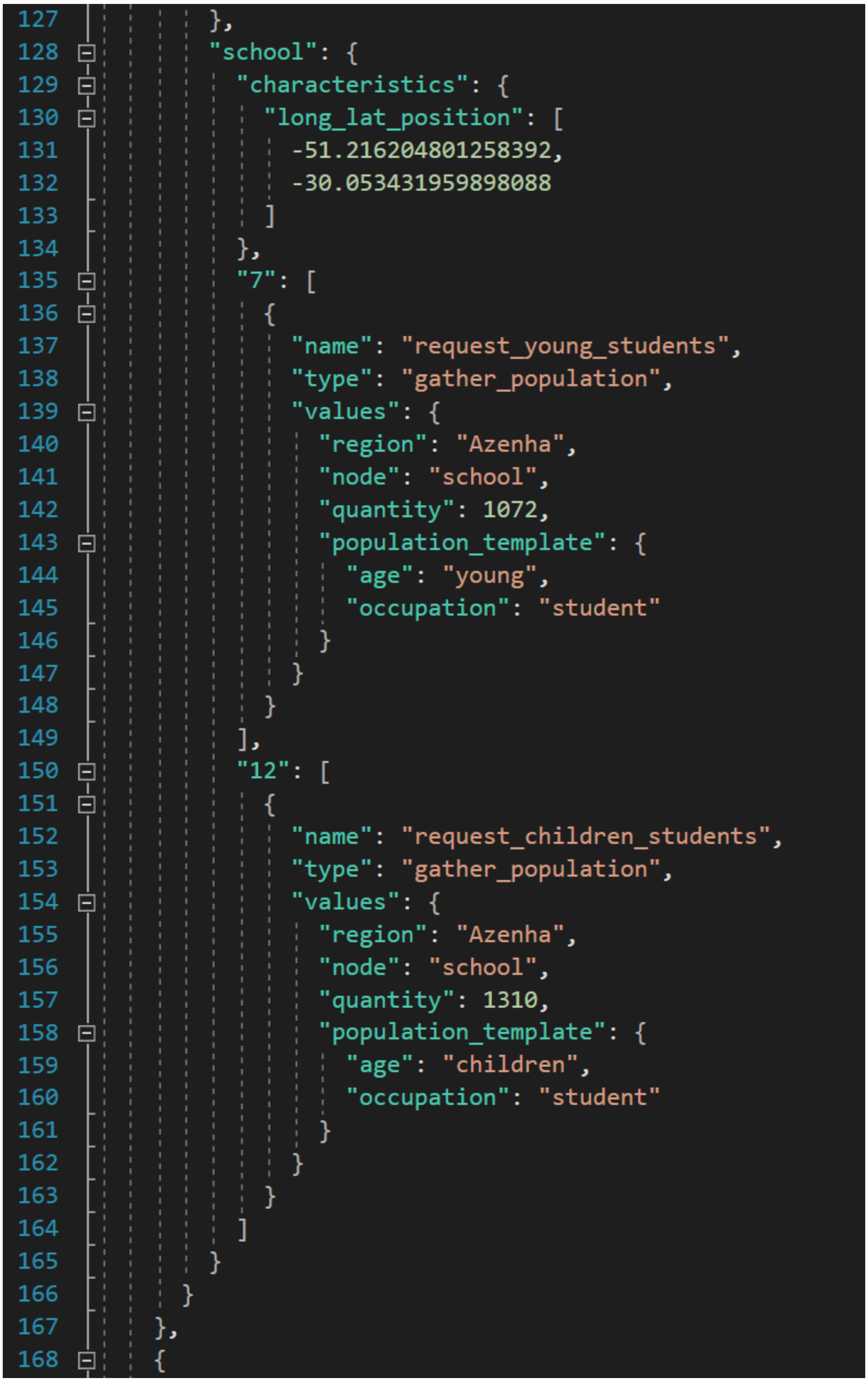
```
            },
            "nodes": {
                "home": {
                    "characteristics": {
                        "long_lat_position": [
                            -51.216515023920117,
                            -30.049979704285192
                        ]
                    },
                    "12": [
                        {
                            "name": "return_young_students",
                            "type": "return_population_home",
                            "values": {
                                "region": "Azenha",
                                "node": "home",
                                "quantity": -1,
                                "population_template": {
                                    "age": "young",
                                    "occupation": "student"
                                }
                            }
                        }
                    ],
                    "17": [
                        {
                            "name": "return_children_students",
                            "type": "return_population_home",
                            "values": {
                                "region": "Azenha",
                                "node": "home",
                                "quantity": -1,
                                "population_template": {
                                    "age": "children",
                                    "occupation": "student"
                                }
                            }
                        }
                    ],
                    "18": [
                        {
                            "name": "return_workers",
```

Figure B.4: Events of the node "Home". The "return_population_home" event type signals people of a given "population_template" to return to the "Home" node at their origin *region*. The value -1 in "quantity" indicates that all available population will be affected.



```
                }
            }
        ],
        "18": [
            {
                "name": "return_workers",
                "type": "return_population_home",
                "values": {
                    "region": "Azenha",
                    "node": "home",
                    "quantity": -1,
                    "population_template": {
                        "occupation": "worker"
                    }
                }
            }
        ]
    },
    "work": {
        "characteristics": {
            "long_lat_position": [
                -51.2133288165572,
                -30.055422465151622
            ]
        },
        "7": [
            {
                "name": "request_workers",
                "type": "gather_population",
                "values": {
                    "region": "Azenha",
                    "node": "work",
                    "quantity": 6041,
                    "population_template": {
                        "occupation": "worker"
                    }
                }
            }
        ]
    },
    "school": {
        "characteristics": {
```

Figure B.5: Third event of the node "Home" and single event of the node "Work". The "gather_population" event type requests a population from neighboring *regions*. The *region* itself may also send people to the target node.



```
                },
                "school": {
                    "characteristics": {
                        "long_lat_position": [
                            -51.216204801258392,
                            -30.053431959898088
                        ]
                    },
                    "7": [
                        {
                            "name": "request_young_students",
                            "type": "gather_population",
                            "values": {
                                "region": "Azenha",
                                "node": "school",
                                "quantity": 1072,
                                "population_template": {
                                    "age": "young",
                                    "occupation": "student"
                                }
                            }
                        }
                    ],
                    "12": [
                        {
                            "name": "request_children_students",
                            "type": "gather_population",
                            "values": {
                                "region": "Azenha",
                                "node": "school",
                                "quantity": 1310,
                                "population_template": {
                                    "age": "children",
                                    "occupation": "student"
                                }
                            }
                        }
                    ]
                }
            }
        },
        {
```

Figure B.6: Events of the node "School". The events requests "Young-Students" and "Children-Students" from neighboring *regions* at the 7th and 12th time-step, respectively.