Final report 2021

Methods for identification of free navigable space

Mitko Aleksandrov, Abdoulaye Diakité and Sisi Zlatanova

Introduction

The research within this project is a part of four-year (2018-2022) international project named iNous focusing on seamless indoor-outdoor navigation for emergency response. The project is funded by South Korea government and leaded by Pusan University, South Korea. The overall idea of the project is to develop a workflow for navigation for the purpose of disaster management system as illustrated below (Figure 1):

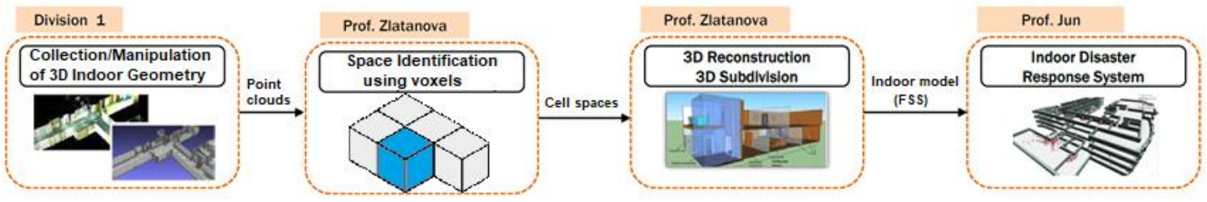

Division 1
Collection/Manipulation of 3D Indoor Geometry
Point clouds
Prof. Zlatanova
Space Identification using voxels
Cell spaces
Prof. Zlatanova
3D Reconstruction 3D Subdivision
Indoor model (FSS)
Prof. Jun
Indoor Disaster Response System

Figure 1: Workflow of the proposed Indoor disaster response system

The general scope of the four-year project is defined as follows:

- Strategies to progressively evaluate the level of semantic information of spatial data (point clouds) required to obtain a suitable input for the indoor disaster systems.
- Voxels classifications and automatic space subdivision based on it.
- Space subdivision according to FSS framework. Investigations will focus on the type of objects and their functional spaces to support creating IndoorGML model.
- Experimenting with different data sets and reporting.

Through the year some of the objectives were adapted with respect to new findings, results and new emerging directions for research. A lot of attention was given on processing, visualising and managing of voxels obtained from point clouds and vector (BIM) models, as well as, on completing the IndoorGML 2.0 documentation, examples and technical implementation.

In the fourth year of the project, the team of professor Zlatanova has concentrated on advancing the fee space identification from point clouds using voxels and completing IndoorGML 2.0. The general scope of the research and developments have been defined as:

- Developing algorithms for identification of free navigable spaces in complex buildings
- Developing functions for path planning at DBMS level
- Rounding-off the developments of IndoorGML and preparing examples

Administrative information

In the specified period the following researchers were involved: Mitko Aleksandrov, Abdoulaye Diakité, Abdullah Alattas and Sisi Zlatanova. In this period no dedicated meetings were organised with the other teams on this specific topic although several discussions were carried out regarding the IndoorGML 2.0.

Scientific project information

This year the research continued developing a voxel-based workflow for indoor modelling. Two major topics were considered 1) procedures for voxelisation and 2) pedestrian navigation in voxelised models. Last year several methods for management of voxels in PostGIS were identified and this year this research was extended towards investigating voxel data structures.

Parallel to the voxel-based workflow, we have continued the work on finalising IndoorGML 2.0. IndoorGML UML modifications was finalised and the first draft containing explanations to all classes, number of use cases and a GML technical implementation was submitted to the Standard Working Group on IndoorGML. SQL implementation is prepared as well but needs to be updated with respect to the last UML diagram. It will be added to the IndoorGML documentation in the first quarter of 2022.

Developing algorithms for identification of free navigable spaces for a complex building

To move from vector space (including point cloud) to voxel space, voxelisation methods are needed. The literature review on voxelisation approaches that was completed last year was extended with literature review on voxel data structures. A paper presenting both investigations was recently published (Aleksandrov et al 2021). Voxel-based approaches are used in many domains, but most of the algorithms and the data structures originate from computer graphics.

Voxelisation brings advantageous structures and neighbourhood information, which are beneficial for processing of point clouds and vector models. Within voxel space, navigable spaces are much easier to identify, especially in 3D. To derive navigable areas for 3D indoor environments and real-time update of it due to some changes, different approaches for a quick voxelisation have been also applied. One of the approaches is based on visibility analyses, in which case rays are cast against voxelised scene to identify occlusions

Voxelisation of geometric objects depends on several voxel properties like connectivity, separation, coverage and tunnelling. Since early seventies, a large number of voxelisation approaches have been reported in the literature, for points, lines, triangles, polygons, parametric surfaces, implicit surfaces, and constructive solid geometry. Naturally for point cloud, point algorithms are of importance. Voxelisation of points is very straightforward process and can be performed in several steps: 1) a translation according to a pivot point obtained from the bounding box of the points, 2) division of all coordinates considering a voxel size, 3) rounding the final values down to the first integer value, which can be a corner or centre point of the voxel and 4) recording a voxel in the voxel space if not existent.

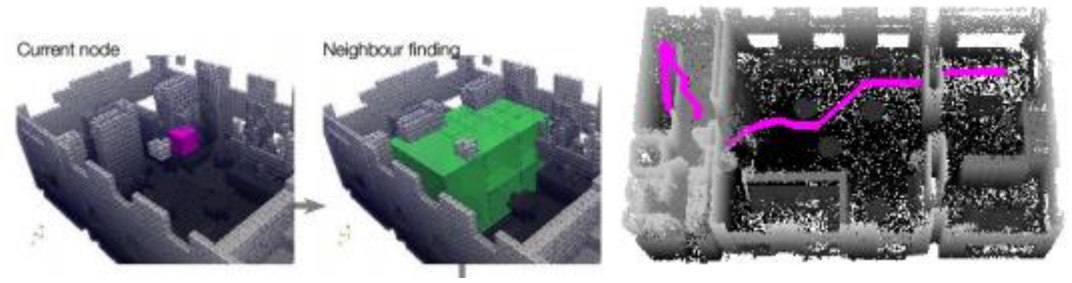


Figure 2: Pathfinding in empty cells in an octree of a point cloud (Rodenberg et al 2016)

In previous research we have reported algorithms for free space identification in point clouds using octrees (e.g. Rodenberg et al 2016) or directly empty and non-empty voxels (Gorte et al 2019). Figure 2 represents an example of path finding in empty octree cells. Using skeletonisation, detection of edges and nodes and creating, selection of surface voxels, a 3D voxel skeleton, navigation grid for egress modelling was presented (Figure 3)

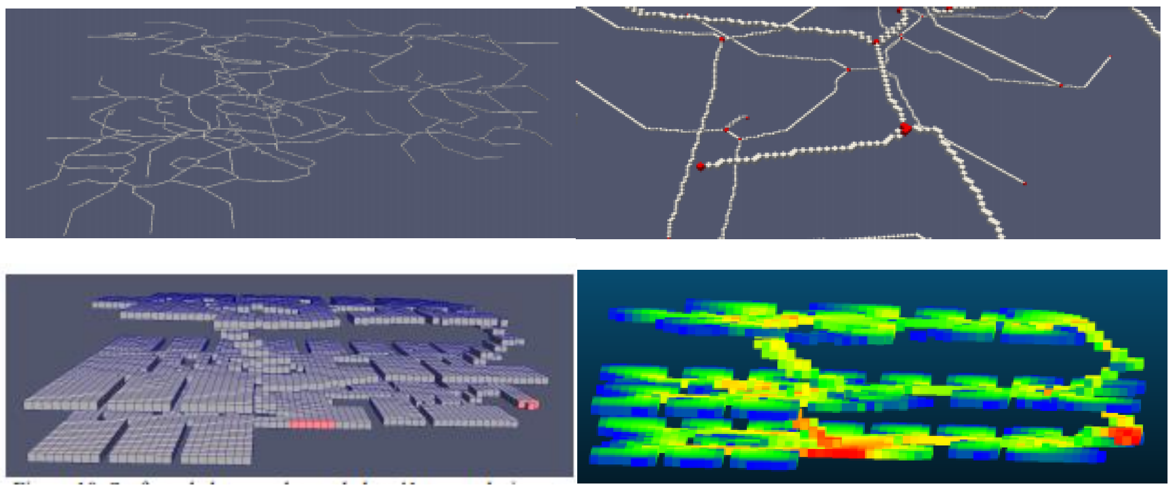
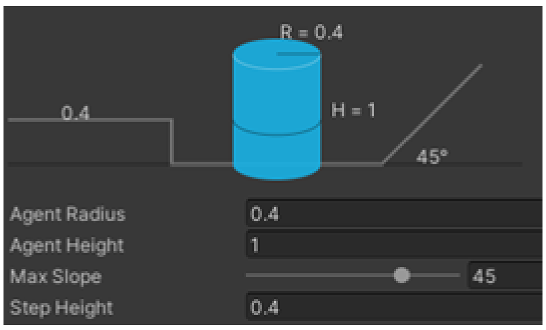


Figure 3:Egress according to Ohm-Kirchhoff method (clockwise): skeletonization, detection of edges and nodes, 3D voxel skeleton and navigation grid. (Gorte et al 2019)

Special attention was given on algorithms within Unity3D because navigable area can be defined with respect to characteristics of an agent. To identify such an area the user (or developer) needs to define the agent's characteristics such as agent's radius and height, maximum slope, and step height, which can be further adjusted (Figure 1). After that Unity identifies all navigable areas/meshes where an agent could walk and pass (Figure 5).



Agent Radius 0.4
Agent Height 1
Max Slope 45
Step Height 0.4

Figure 4: Defining capabilities of an agent to identify a navigable area

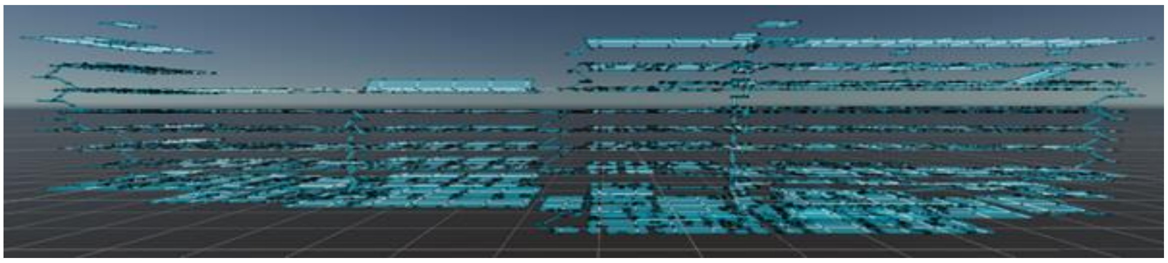


Figure 5:A navigation area in Unity

Unity provides triangles for pathfinding which match to a good extent the navigation mesh. Based on them, we can extract border edges which we can use to calculate distances (i.e., forces) to obstacles (Figure 6) used for crowd dynamics. Triangles can be used to understand if an agent is still within a navigable area, to initialise an agent on it, for pathfinding, and to identify navigable voxels.

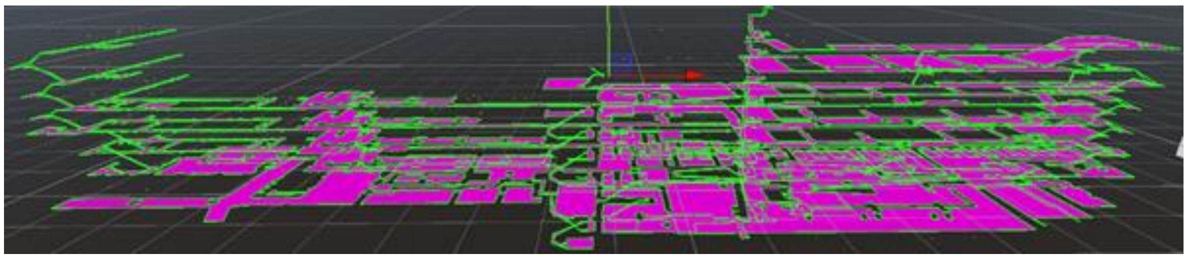


Figure 6: Triangles and border edges of a navigation mesh

Navigable triangles can be used to identify navigable voxels. This is extremely useful to store additional information and to perform some 3D spatial queries. The voxelisation that is required is a conservative voxelisation, which would guarantee having a voxel for any part belonging to the triangles (Figure 7).

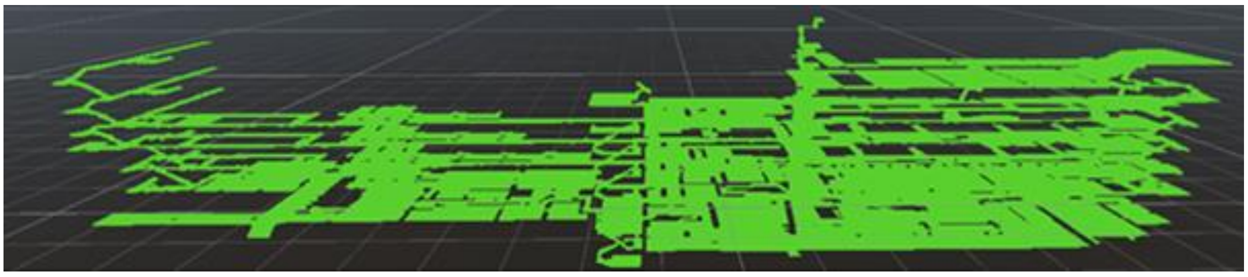


Figure 7: Conservative voxelisation based on navigable triangles

If we voxelise all IfcSpaces (Figure 8) and classify the navigable voxels, we can easily determine in which room each agent is.

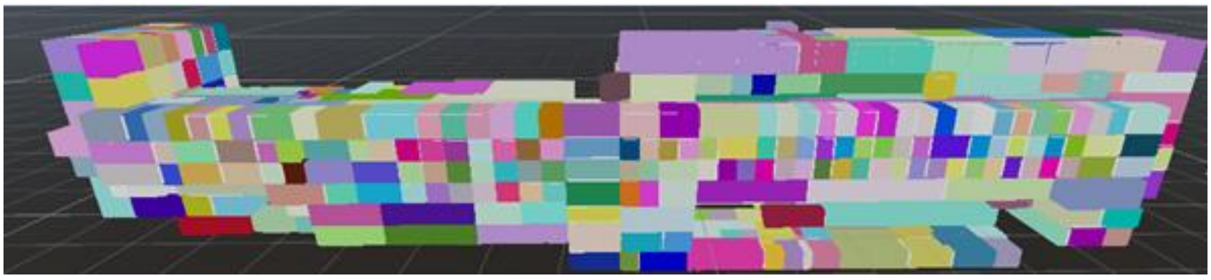


Figure 8: Voxelised IFC space

All objects in the indoor environments can be also classified into static (stairs, corridors furniture) and dynamic. Then the navigable mesh will consider all obstacles. (Figure 9).

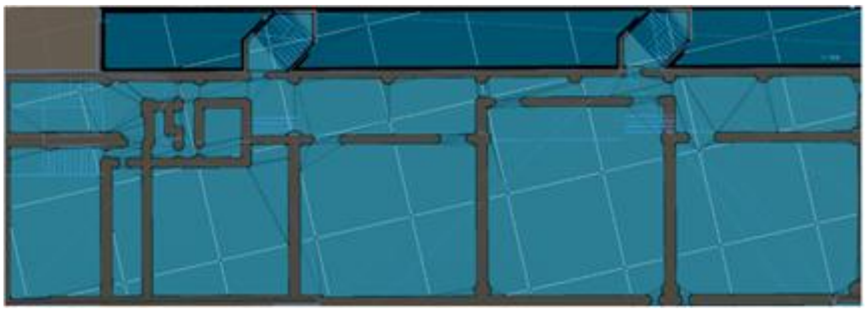


Figure 9: An example of a navigation mesh considering all physical objects in its calculation showed in blue. The grid is showing imaginary lines in white that represent the tessellated chunks

There is also an offset from walls, which roughly represents the pedestrian's radius. In this way, a pedestrian will not collide with any wall. However, this strategy may shorten the visible view of pedestrians and create narrow passages, which may create bottlenecks. Another important aspect is the minimum width allowed for a pedestrian. Therefore, we have followed an approach of maintaining two navigation meshes. The first one is used to extract border edges of navigable areas as close as possible to walls by using a value of 1cm for the agent's radius. The second one is used as a graph to determine paths for pedestrians, where the radius size is set up based on the maximum width that the crowd simulation model allows.

Space subdivision is important aspect of the navigation process because it supports the agents' activities. The subdivision can be performed on 2D and 3D, depending on the application. The main advantage of the 2D subdivision is that computations are in 2D but then vertical obstacles will be not taken into consideration. If the entire 3D space is used, the agents can freely travel in the environment, but it would be required to keep more information in memory. Therefore, we have explored a combination of the two approaches. a 2D space subdivision, but the navigable area is classified into horizontal and vertical areas (considering the height). More details can be found in Aleksandrov et al, 2021.

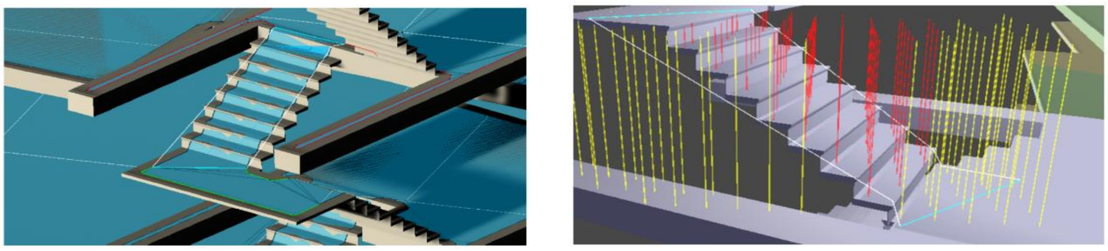


Figure 10: Mismatch between mesh and spaces (left) and raytracing to estimate the heights (right)

At the end, a simulation can be performed, showing the agents across the entire building, including stairs (Figure 11).

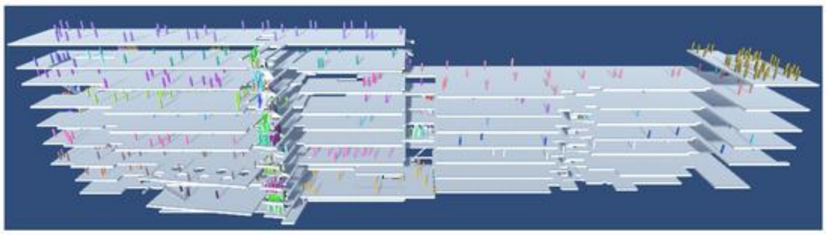


Figure 11: Agent navigation in the entire building. The colour of the agents indicates the space in which the agents are located.

Developing functions for path planning at DBMS level

For the path planning at DBMS, the PostGIS/PostgreSQL extension pgRouting was used (https://pgrouting.org/). Developing functions at a database level offer many advantages:

Data changes can be reflected immediately and would be valid for all users and the routing engine.

Data and attributes can also be dynamically changed by building owners or users. Via interfaces as JDBC, ODBC, geoJSON, clients like QGIS and Cesiusm can be linked to the database for direct access to different parameters of the navigation

The cost function can also be dynamically changed, using SQL scripts and the integrated PostGIS functions.

The current version of pgRouitng offers the following algorithms (https://pgrouting.org/):

- All Pairs Shortest Path, Johnson's Algorithm
- All Pairs Shortest Path, Floyd-Warshall Algorithm
- Shortest Path A*
- Shortest Path Dijkstra
- Bi-directional Dijkstra Shortest Path
- Bi-directional A* Shortest Path
- Driving Distance
- K-Shortest Path, Multiple Alternative Paths
- K-Dijkstra, One to Many Shortest Path
- Traveling Sales Person
- Turn Restriction Shortest Path (TRSP)

A number of experiments were performed in Unity3D and Cesium with pgRouting and other spatial functions.

pgRouting and Cesium

Detailed explanations of the data preparation and running the pgRouting Dijkstra Shortest path are reported in Alattas et al 2021. pgRoutin requires two specific tables Node and Edge. Node table contains coordinates of nodes (rooms). The Edge table provides the connectivity between the nodes and have two columns Target and Source . PgRouting provides a several options to import line string and create a topologically correct graph. Besides the compulsory columns, more information can be stored in both tables.

In the specific case described in Alattas et al 2021, the coordinates of the nodes are automatically derived from Revit and further encapsulated in spatial data type ST_point (Figure 12). The Edge Geometry is also automatically computed within Revit and imported in the Edge table as nodeID, nodeFrom and toNode. The geometry of the edge is computed with PostGIS function, which connects NodeFrom and toNode. gRouting does not support 3D network, but the derived connectivity using Revit is sufficient to populate the default fields.

To be able to compute a path, the edges must be enriched with information about the cost. The cost in this case in the length of the edges. This process is completely automatic and performed within PostGIS, using SQL statements.

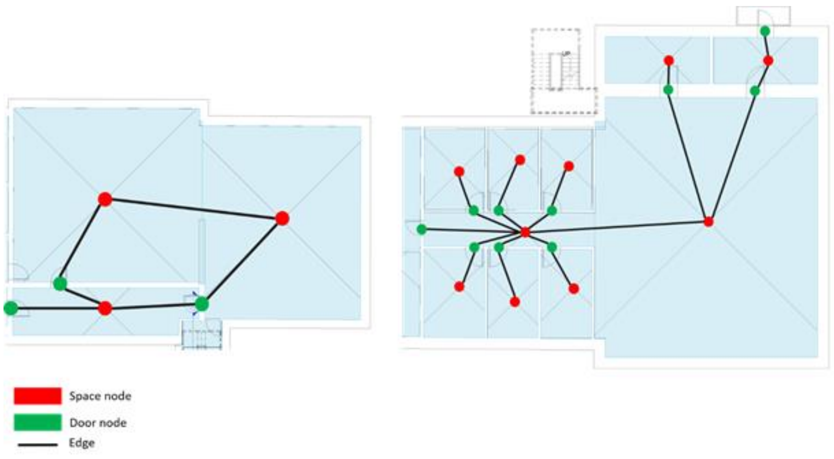
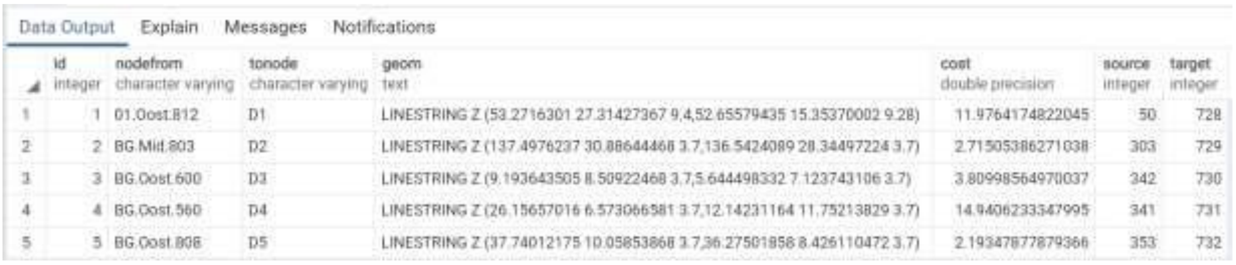


Figure 12: Extracting nodes and edges using Revit functions (Alattas et al 2021).

The final Edge table used for the path computation is given below:



Data Output
Explain
Messages
Notifications

	id integer	nodefrom character varying	tonode character varying	geom text	cost double precision	source integer	target integer
1	1	01.Oost.812	D1	LINESTRING Z (53.2716301 27.31427367 9,4,52.65579435 15.35370002 9.28)	11.9764174822045	50	728
2	2	BG.Mid.803	D2	LINESTRING Z (137.4976237 30.88644468 3.7,136.5424089 28.34497224 3.7)	2.71505386271038	303	729
3	3	BG.Oost.600	D3	LINESTRING Z (9.193643505 8.50922468 3.7,5.644498332 7.123743106 3.7)	3.80998564970037	342	730
4	4	BG.Oost.560	D4	LINESTRING Z (26.15657016 6.573066581 3.7,12.14231164 11.75213829 3.7)	14.9406233347995	341	731
5	5	BG.Oost.808	D5	LINESTRING Z (37.74012175 10.05853868 3.7,36.27501858 8.426110472 3.7)	2.19347877879366	353	732

Having all the information properly organised in pgRouting, a variety of paths can be computed. The paths are further visualised in Cesium, using GeoJSON. The geometry of the building is represented as Cesium tiles, which are created with FME. Figure 13 presents the system architecture and Figure 14 illustrates the graphic user interface to request a path between two rooms for a specific user group. The path is visualised in two ways: 1) highlighting the spaces that have to be visited and 2) the actual path (above the spaces).

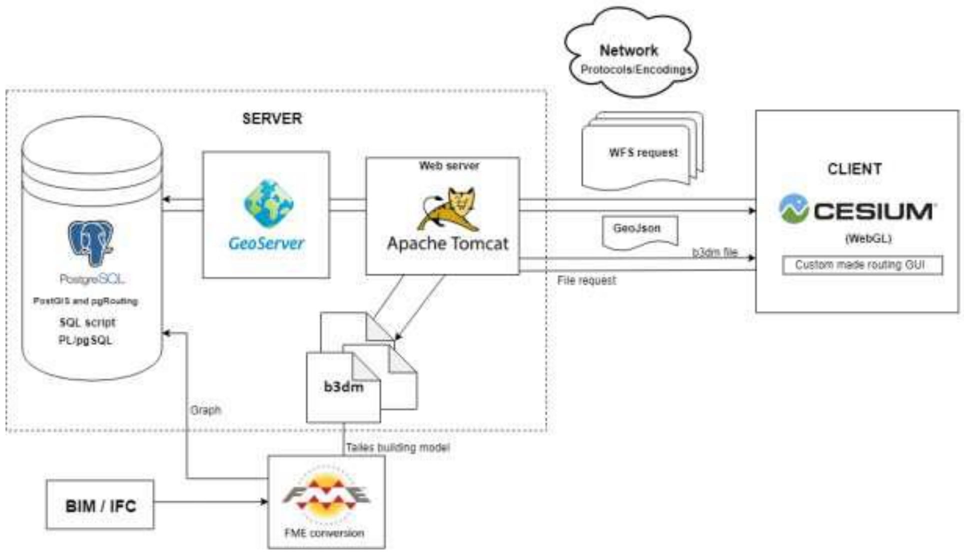
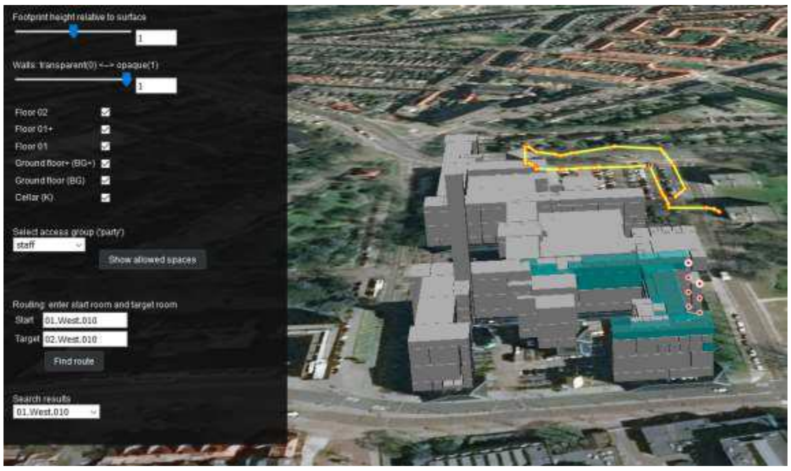


Figure 13: System architecture for path visualisation implemented by Alattas et al 2021


Footprint height relative to surface: 1

Walls: transparent(0) <--> opaque(1): 1

Floor 02 ✓
Floor 01+ ✓
Floor 01 ✓
Ground floor+ (BG+) ✓
Ground floor (BG) ✓
Cellar (K) ✓

Select access group ('party'): staff

Routing: enter start room and target room
Start: 01.West.010
Target: 02.West.010

Search results: 01.West.010

Figure 14: Visualisation in Cesium as implemented by Alattas et al 2021

pgRouting and Unity3D

PostGIS was also used to store the voxels from Unity3D as well. As mentioned previously, the main reason for selecting PostgreSQL is the fact that it supports storage of many different data types like points, lines, polygons, point clouds, raster and so on. Moreover, this allows to store different geometric data (vector and voxel) and be used when needed. Therefore, a decision has to be made which data type is most appropriate.

After experiments with different data types presented in the report of last year (Li et al 2020), to store the voxels along with their properties we identified two ways. The first one is to use PointZ data type for their storage based on PostGIS extension, where the properties are additionally attached as columns. The other option is to use the pgPointCloud extension, which is primarily built to manipulate large point clouds. While at the first glance the second option might be better, it does not support querying small areas or within a proximity. In contrast, storing the voxels as points gives more freedom to query quickly only one voxel or several voxels nearby an agent.

The edges representing borders of walkable area are stored as LineStringZ, whereas the triangles as PolygonZ. Once the data are in the database, we can visualise them in straightforward way with only a few clicks in QGIS (Figure 15). This is a quick way to examine if the data is correctly imported.

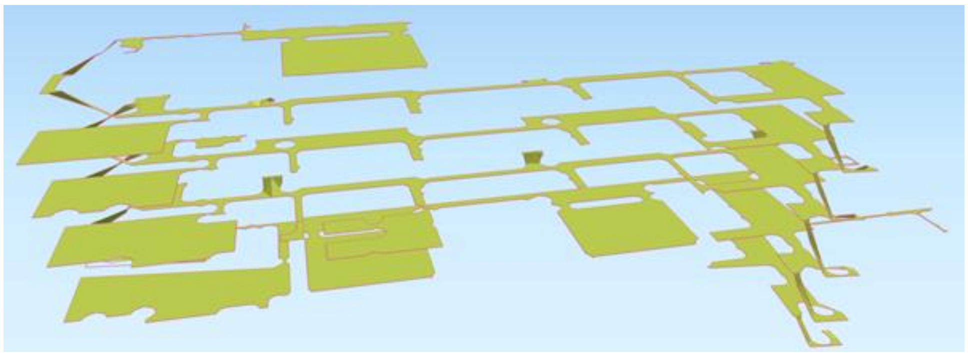


Figure 15: Visualisation of triangles and border edges in QGIS

The next step is creating Edge and Node tables of pgRouting. As explained above, it is important in the table with the edges to store ids of points representing edges as source and target. A route identified in this way will be the shortest between two points but requires joining the identified point sequence with the nodes table to identify route points in 3D (Figure 16).

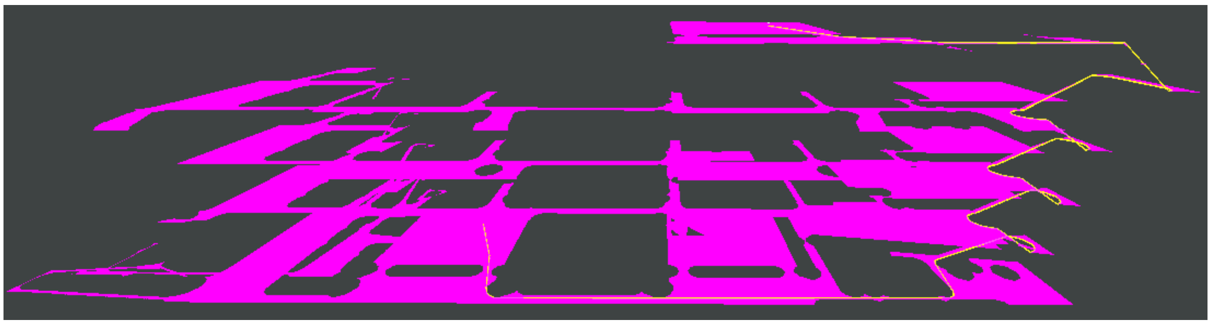


Figure 16: The shortest path (in yellow) between two points

To speed up the execution of queries, it is needed to add 3D spatial indexes for voxels represented as points, edges used for borders, edges used for path finding, and triangles roughly matching the navigation mesh representation. In this way, the queries will consider first the bounding boxes of the geometries. Only if a more precise calculations are needed, the actual geometry will be considered. Indexes are needed for the columns related to the sources and targets of the edges as well as for the ids of the nodes.

As mentioned above pgRouting allows to find shortest paths from one destination to many targeting destinations as well as from many destinations to one targeting destination. For instance, agents being in one room can quickly obtain their paths to desired locations as well as agents being located in many places can obtain their paths to one exit location. In this way, some network edges will be reused, and some agents will be grouped reducing the time to calculate all paths.

We did a stress test for the first scenario, where we compute paths for all agents to their designated destinations. Agents were placed in different locations in a building, not just in one room (Figure 17). We created a function which allows to take one target location or several of them and identify the quickest routes to those places. We did performance evaluation of the function to see how much time we need to instantiate all pedestrian parameters and find paths (Figure 19). The graph shows that for smaller number of pedestrians it takes more time on average to find paths than for 1000 or more. On average, initialising all parameters for a pedestrian including pathfinding process takes approximately 2.4 milliseconds. We need to mention that the configuration of our computer is Intel(R) Core(TM) i7-7600 CPU @ 2.8GHz, 2904Mhz having 2 cores and 16GB RAM.

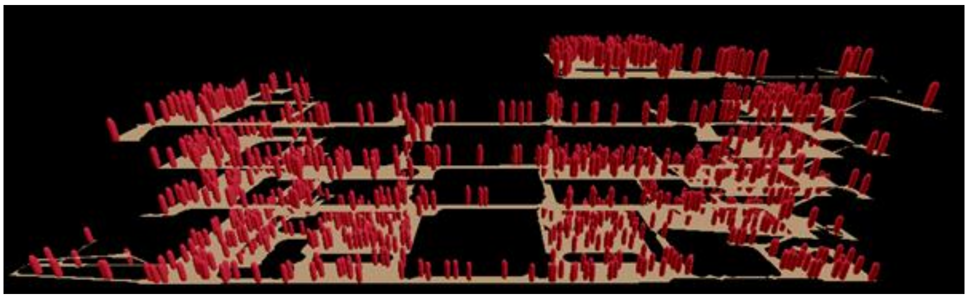


Figure 17: 1000 instantiated agents

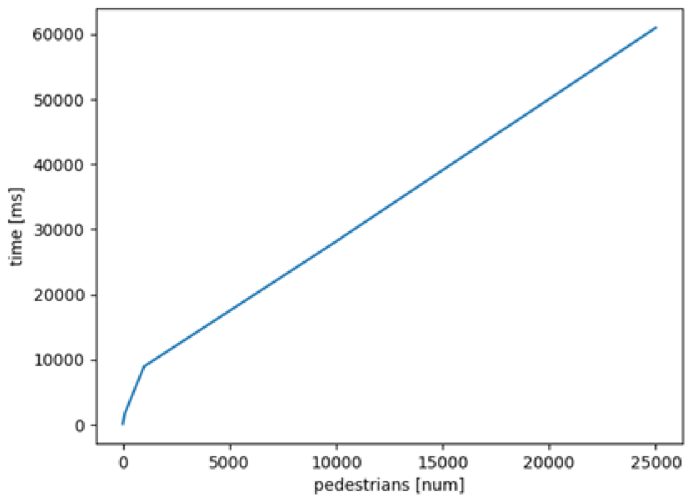


Figure 18: Time needed for finding paths and instantiation of pedestrians

This is to show that having voxels we can easily cover all the navigable space by agents, which is not the same if we use the triangles. At the same time, we can know in which IfcSpace they are located at any given time (Figure 19).

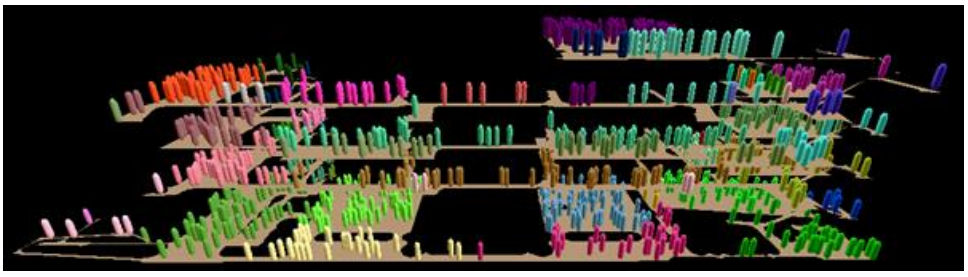


Figure 19: Identifying position of pedestrians per IfcSpace.

Rounding-off the developments of IndoorGML and preparing examples

Within this year the IndoorGML 2.0 was rounded off and the first complete version, including concepts presented in UML and the GML implementation was presented. As discussed in the document, IndoorGML has been designed to support Location-based services applications. IndoorGML provides simplified yet standardised notations for indoor spaces and networks, which can be used in different application contexts such as navigation, monitoring, asset and property management. IndoorGML can be linked to and derived from any geometric model that a building owner may have (floor plans, CAD models, BIM models, laser scans, measurements) (Figure 9). The semantics notations of IndoorGML are generic and therefore allowing to protect some sensitive building information.

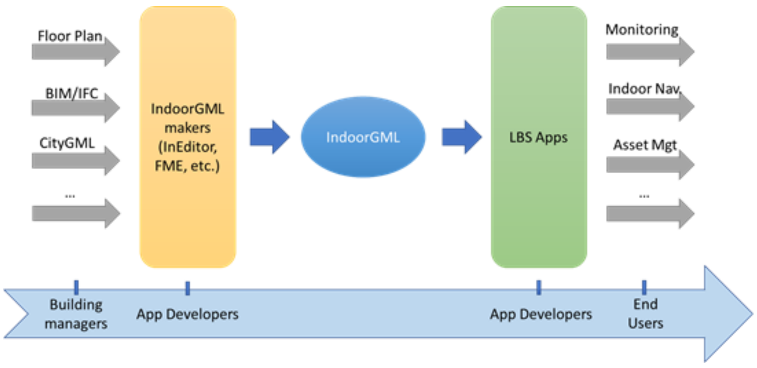
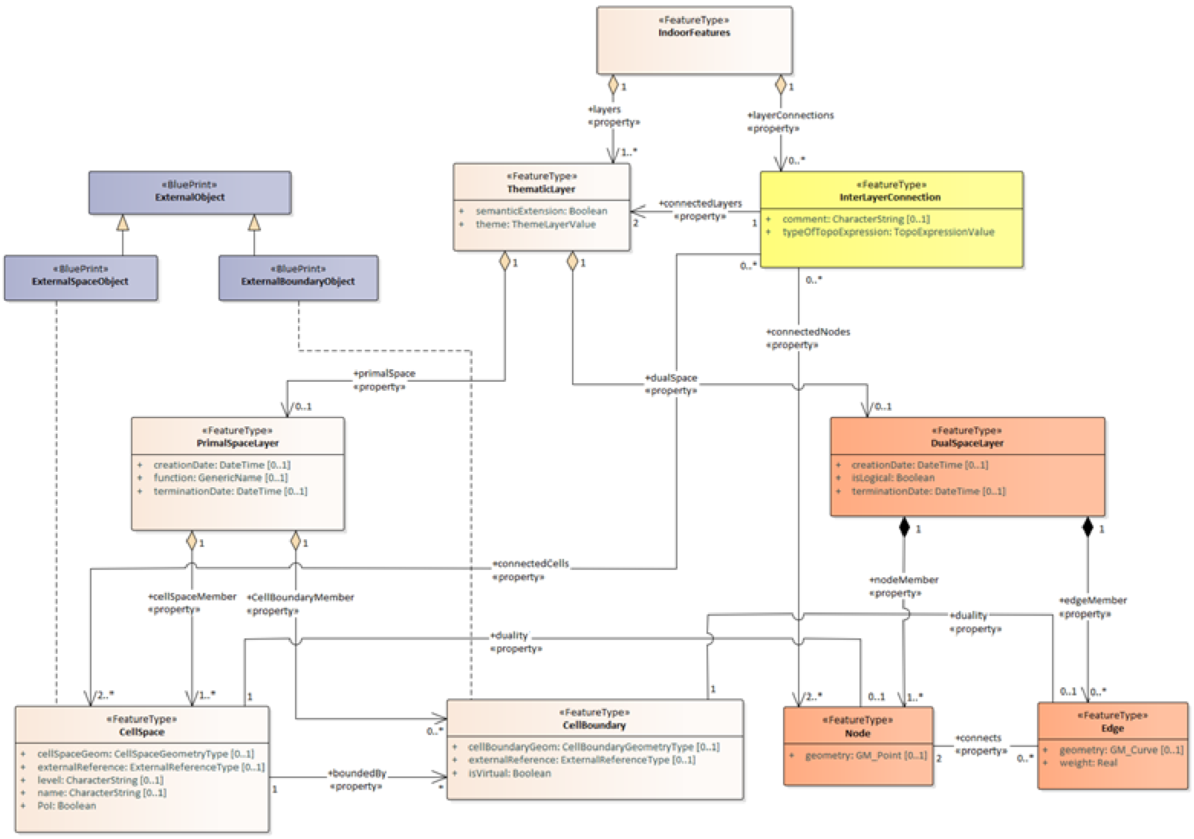


Figure 20: The place of IndoorGML in the overall application development ecosystem.

The IndoorGML 2.0 document consist of several sections discussing the motivation for IndoorGML, the basic concepts – Cellular space, Poincare Duality, Semantic extensions (modularisation) and Thematic layers – the UML diagrams of the core module, the UML of the Navigation extension and five GML implementation examples that illustrate the use of IndoorGML 2.0. The UML Core model was refined and the last version of it is given in Figure 21and the Navigation Module. The Navigation module has been also tuned and the last version is shown in Figure 22.



«FeatureType» IndoorFeatures
+layers «property»
+layerConnections «property»
«BluePrint» ExternalObject
«BluePrint» ExternalSpaceObject
«BluePrint» ExternalBoundaryObject
«FeatureType» ThematicLayer
+ semanticExtension: Boolean
+ theme: ThemeLayerValue
+connectedLayers «property»
«FeatureType» InterLayerConnection
+ comment: CharacterString [0..1]
+ typeOfTopoExpression: TopoExpressionValue
+connectedNodes «property»
+primalSpace «property»
+dualSpace «property»
«FeatureType» PrimalSpaceLayer
+ creationDate: DateTime [0..1]
+ function: GenericName [0..1]
+ terminationDate: DateTime [0..1]
«FeatureType» DualSpaceLayer
+ creationDate: DateTime [0..1]
+ isLogical: Boolean
+ terminationDate: DateTime [0..1]
+connectedCells «property»
+cellSpaceMember «property»
+CellBoundaryMember «property»
+nodeMember «property»
+edgeMember «property»
+duality «property»
+duality «property»
«FeatureType» CellSpace
+ cellSpaceGeom: CellSpaceGeometryType [0..1]
+ externalReference: ExternalReferenceType [0..1]
+ level: CharacterString [0..1]
+ name: CharacterString [0..1]
+ PoI: Boolean
+boundedBy «property»
«FeatureType» CellBoundary
+ cellBoundaryGeom: CellBoundaryGeometryType [0..1]
+ externalReference: ExternalReferenceType [0..1]
+ isVirtual: Boolean
«FeatureType» Node
+ geometry: GM_Point [0..1]
+connects «property»
«FeatureType» Edge
+ geometry: GM_Curve [0..1]
+ weight: Real

Figure 21:Figure 9. IndoorGML 2.0 core model.

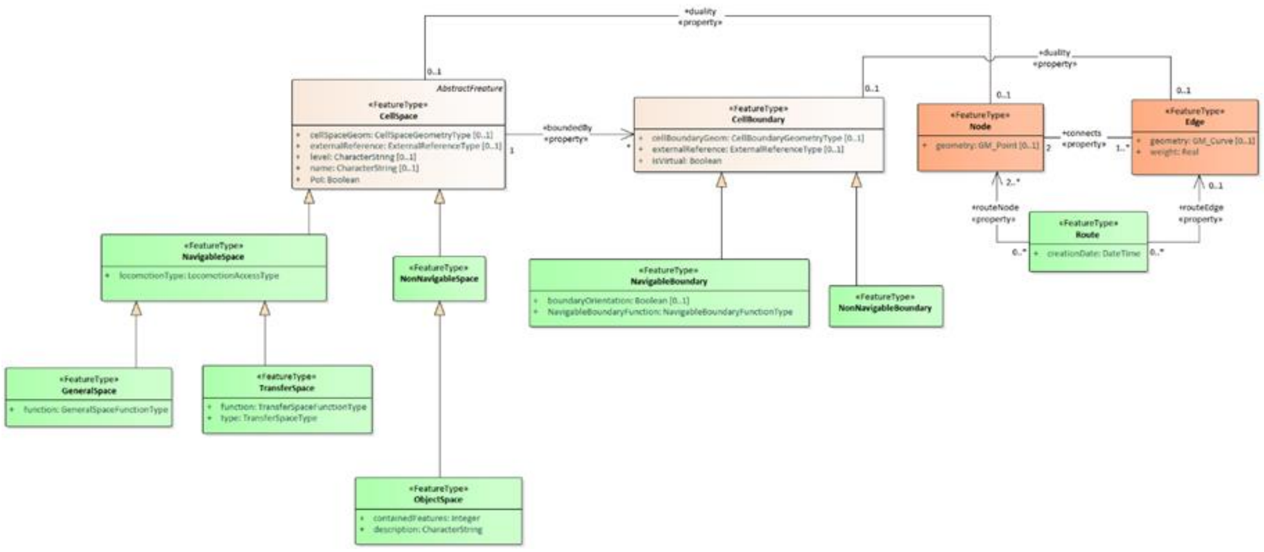

«duality «property»
«duality «property»
0..1
AbstractFreature
«FeatureType» CellSpace
+ cellSpaceGeom: CellSpaceGeometryType [0..1]
+ externalReference: ExternalReferenceType [0..1]
+ level: CharacterString [0..1]
+ name: CharacterString [0..1]
+ PoI: Boolean
«boundedBy «property»
1
*
«FeatureType» CellBoundary
+ cellBoundaryGeom: CellBoundaryGeometryType [0..1]
+ externalReference: ExternalReferenceType [0..1]
+ isVirtual: Boolean
«FeatureType» Node
+ geometry: GM_Point [0..1]
«connects «property»
2
1..*
«FeatureType» Edge
+ geometry: GM_Curve [0..1]
+ weight: Real
2..*
0..1
«routeNode «property»
«routeEdge «property»
«FeatureType» Route
+ creationDate: DateTime
0..*
«FeatureType» NavigableSpace
+ locomotionType: LocomotionAccessType
«FeatureType» NonNavigableSpace
«FeatureType» NavigableBoundary
+ boundaryOrientation: Boolean [0..1]
+ NavigableBoundaryFunction: NavigableBoundaryFunctionType
«FeatureType» NonNavigableBoundary
«FeatureType» GeneralSpace
+ function: GeneralSpaceFunctionType
«FeatureType» TransferSpace
+ function: TransferSpaceFunctionType
+ type: TransferSpaceType
«FeatureType» ObjectSpace
+ containedFeatures: Integer
+ description: CharacterString

Figure 22:Figure 9. Navigation module

IndoorGML 2.0 has the following differences in respect to IndoorGML 1.1

- The UML description is leading
- Text is amened, concepts/definitions are clarified
- There are planned three Appendices: GML,SQL, GeoJSON
- The UML is modified in several aspects
  - Geometry is attribute
  - Vocabulary is amended
  - Attributes are introduced
  - Interlayer connection between spaces is introduced

The basic concepts that have been leading in IndoorGML 1.1 are preserved and better described.

- Poincare Duality (Spatial Networks)
- Thematic Layers
- Core /Extension Modules

We believe IndoorGML 2.0 is very flexible and can deal with all kinds of spaces (virtual and physical) for any kind of application that has the notion of space (navigation, sensor coverage, ownership, access spaces).

Each space subdivision can be organised as a separate Thematic layer. Each thematic layer can contain only non-overlapping spaces. Spaces that are accessible only for a group of users can also be organised as separate Thematic layers (e.g. Spaces for visitors, for Facility manages, for Employees). Allowing interlayer connection between Thematic layers ensures that spaces can be linked. The link can be any relationship as defined in the 9i model, including 'contains'. For example, a space subdivision that is created on the basis of floor subdivision can be linked with more detailed subdivision that includes room and corridor spaces. A relationship will be created indicating that a Floor X 'contains' spaces/rooms X1-X15. This mechanism allows to create arbitrary, application-dependent hierarchies of spaces subdivisions.

The concept of space extends to all parts of an indoor environments including construction components as walls and slabs and other indoor features such as furniture and equipment, even crowds. The geometric representation of complex indoor features (such as furniture) is recommended to be modelled as a minimum maximum box. Considering the semantics in the Navigation module, spaces can be classified into Navigable and Non-Navigable. According to this classification all spaces that are obstacles for navigation (walls, furniture, crowd) will be annotated a non-navigable. Following the Thematic layer mechanism, the non-navigable spaces can be organised in a separate layer. This will allow to specify a relationship between furniture or equipment and a specific room. Figure 11 illustrates the geometric modelling of furniture and the different paths that can be derived.

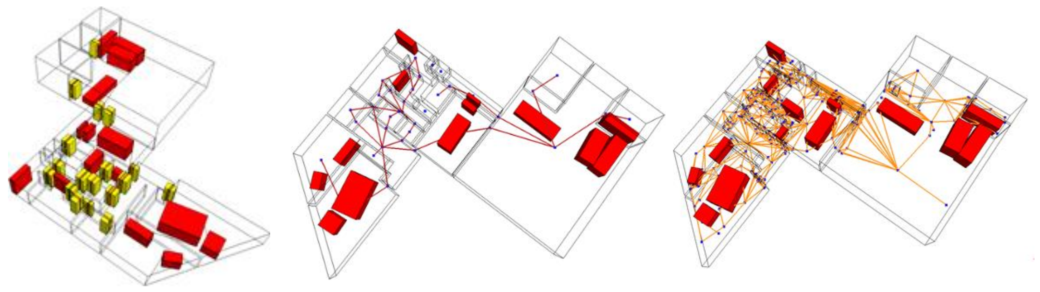


Figure 23: Furnished model with object represented with their minimum bounding box geometry (left) and two examples of path computation, one ignoring (middle) and one considering (right) the furnishing elements.

In the first months of 2022, the SQL implementation will be prepared and included in the IndoorGML 2.0 specification and later in the year GeoJSON implementation can be negotiated to be prepared.

Other related topics

Related to indoor navigation several other topics were simultaneously investigated, of which the most prominent is :

- Obtaining indoor spaces from Industry Foundation Classes. IFC contain notation of spaces and it seems to be the most natural source considering newly designed buildings and constructions. A literature review on the IFC for navigation has revealed that IfcSpace is extensively used to derive spaces for IndoorGML. Software packages such as Revit or Unity3D provide functions to derive a navigation network (Liu et al 2021)
- An Investigation was initiated on approaches for modelling of 3D aerial substances dispersion. The intention is to integrate agent modelling with hazard dispersion modelling to analyse the effect of various pollutants and hazards on humans. Some of the results within this project contributed to a successful grant and establishment of Research Centre of Excellence BREATHE, leaded by Faculty of Medicine, UNSW.
- A dedicated PhD research has started on 3D pedestrian evacuation modelling considering obstacles one can go under (like tables) or climb above (like small platforms).
- The importance of IndoorGML was recognised by the International Society for Photogrammetry and Remote Sensing (ISPRS) and a scientific initiative ‘Pushing forward the developments of software tools for IndoorGML’ was granted to Abdoulaye Diakite (https://www.isprs.org/society/si/default.aspx)

Conclusions

Whin this final year of the project several developments and experiments contributed to the following conclusions:

- Voxel space is a powerful mechanism for 3D indoor modelling of static and dynamic phenomena

- DBMS and more specifically PostgreSQL and its extensions PostGIS, pgRouting are favourable for management of voxels as well as integrated voxel and vector data. USING DBMS is a flexible approach of maintaining a dta according to a common spatila schema, which can be accessed by different packages such as QGIS, Unity, Cesium and Revit.
- pgRouting simplifies the work of developers by providing many navigation algorithms. It can be used to accommodate the IndoorGML information as well.
- IndooorGML is maturing and mentioned in international activities, e.g. https://www.ogc.org/projects/initiatives/idbepilot. The simplified UML diagrams as well as further developments of tools is increasing the interest to the standard.

Publications within 2021 project year

1. Alattas, A., M. de Vries, S. Zlatanova, P. van Oosterom, 3D pgRouting and visualization in Cesium JS using the integrated model of LADM and IndoorGML, In: Proceedings of the FIG Working Week 2021, Online, pp. 33, 2021.
2. Alexandrov, M., S. Zlatanova, and D. J. Heslop, 2021, Voxelisation Algorithms and Data Structures: A Review, Sensors, 2021 21(24), 8241 (*doi*)
3. Alexandrov, M., D. J. Heslop and S. Zlatanova, 2021, 3D Indoor Environment Abstraction for Crowd Simulations in Complex Buildings, Buildings, 2021, 11(10), 445 (*doi*)
4. Liu, L. B. Li, S. Zlatanova, P. van Oosterom, 2021, Indoor navigation supported by the Industry Foundation Classes (IFC): A survey, Automation in Construction, Vol 121, January 2021, 10436 (*doi*)
5. IndoorGML 2.0 specification (DRAFT version, submitted to OGC SWG IndoorGML)

Related publications:

1. Gorte, B, M. Aleksandrov and S. Zlatanova, 2019, Towards Egress modelling in voxel building models, ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., IV-4/W9, 43–47 (*doi*)
2. Li, W., S. Zlatanova, and B. Gorte, 2020, Voxel data management and analysis in PostgresSQL/PostGIS under different data layouts, ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., VI-3/W1-2020, 35–42 (*doi*)
3. Rodenberg, O. B. P. M., Verbree, E., and Zlatanova, S., 2016, Indoor A* pathfinding through and octree representation of a point cloud, ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., IV-2/W1, 249-255, doi:10.5194/isprs-annals-IV-2-W1-249-2016. (*pdf*)