American University of Dubai, UAE
9th-11th November, 2014

EXPLORING CITYENGINE AS A VISUALISATION TOOL FOR 3D CADASTRE

Alexandra RIBEIRO
José-Paulo DUARTE de ALMEIDA
Claire ELLUL

Context

- Several visualisation prototypes have been proposed for 3D cadastre so far:
  - *e.g.* spatial databases with CAD and GIS front-ends.
- Technological advancements and the ease in the use of web-based platforms led to the construction of web technology based prototypes.
- However, prototypes above still require maturation and validation by users before being used in real life situations.

Motivation

- 3D visualisation systems are used in 3D modelling of urban environments. Thus, if such systems can somehow be reutilised in the 3D cadastre context, general costs may well be lower than those associate to systems entirely created from scratch.

- Shojaei, Kalantari, Bishop, Rajabifard & Aien (2013)[1]:
  - identified a series of 3D visualisation requirements for property cadastre,
  - undertook comparison tests between different applications: GoogleEarth, ArcGlobe, NASA World Wind, and Terra Explorer Viewer.

- CityEngine (by ESRI)…

[1] SHOJAEI, D.; KALANTARI, M.; BISHOP, I.D.; RAJABIFARD, A.; AIEN, A. (2013): Visualization Requirements for 3D Cadastral Systems. *Computers, Environment and Urban Systems*, 41: 39–54.

Main goal and objectives

- The evaluation of CityEngine’s suitability as a 3D cadastral visualisation tool.
  - *i.e.* main focus is on 3D visualisation, hence not on data management or data delivery.

- To test CityEngine’s performence against all requirements identified by Shojaei *et al.* (2013):
  - Cadastral features (except “massive data handling”);
  - Purely visualisation features;
  - Non-functional features.

CityEngine: general characteristics

- Stand-alone commercial desktop 3D modelling application developed by ESRI R&D Center Zurich for the purposes of generating 3D urban environments.

- A design-oriented application, hence not originally set up to deal with measured registered data.

- Main underlying features:
  - **Procedural modelling approach**

    A code-based procedure, consisting of a series of geometric modelling commands that is given to the system; *i.e.* instead of the classical intervention of the user, who manually interacts with the model and models 3D geometries, the whole task is described abstractly in a rule file.
  - **Computer generated architecture** (CGA) **shape grammar**

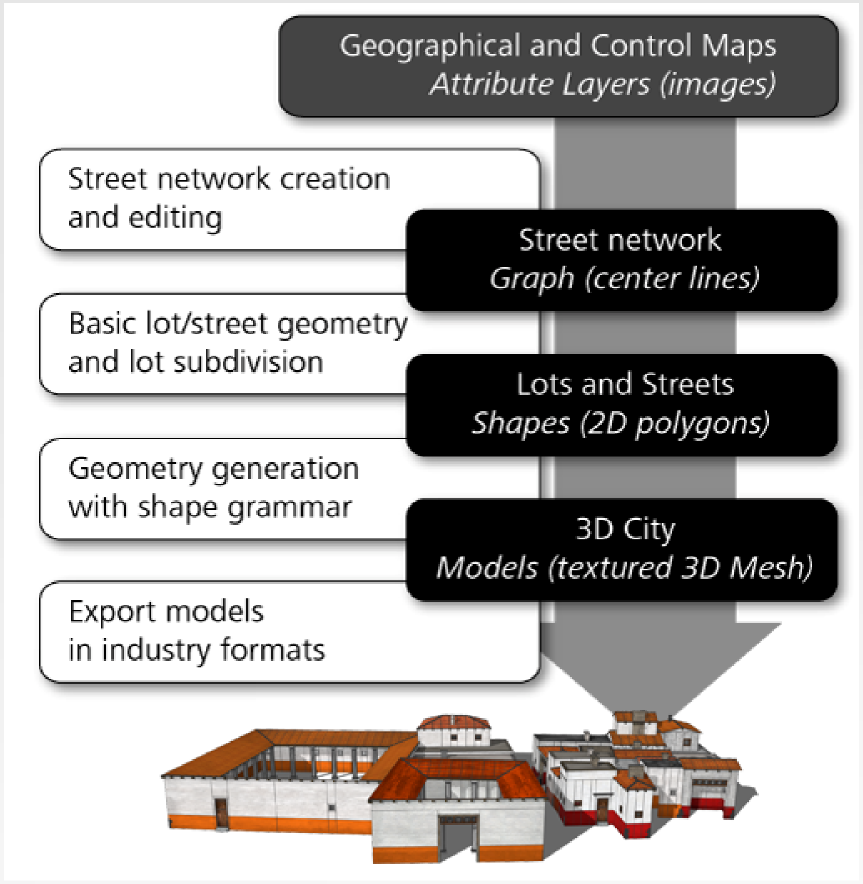
    Technical framework that supports all the aforementioned commands and rules; *i.e.* programming language indicated to generate architectural 3D content.

Procedural modelling *vs.* CGA shape grammar

- Basic primitive is a “shape” consisting of:
  - A name (so-called shape symbol or rule name),
  - A geometry (polygonal mesh & attributes, like colour, material and texture),
  - A pivot (describes shape's coordinate system - in object coordinates, relative to the initial shape's origin),
  - A locally oriented bounding-box (so-called scope) relatively to the pivot.
- Procedure: CGA rules operate on shapes
  - Starting point is a 2D shape,
  - Extrusion carried out according to the rule file,
  - Result consists of a 3D object.
- Advanced Edition comes with an integrated Python scripting interface.
  - To control repetitive tasks or to automate other specific actions,
  - To create formatted reports in file format,
  - To carry out attribute queries (only possible through Python scripting).
  - …

CityEngine's pipeline

- Sequential steps to generate an urban environment from scratch


Geographical and Control Maps
Attribute Layers (images)
Street network creation and editing
Street network
Graph (center lines)
Basic lot/street geometry and lot subdivision
Lots and Streets
Shapes (2D polygons)
Geometry generation with shape grammar
3D City
Models (textured 3D Mesh)
Export models in industry formats

Black boxes	illustrate data types (layers)
White boxes	illustrate the operations to create them

First step: the street network is created based on graph algorithms + 2D polygons representing street areas;

Second step: resulting blocks are subdivided into lots;

Third step: the 3D models of the buildings are generated using the CGA rules.

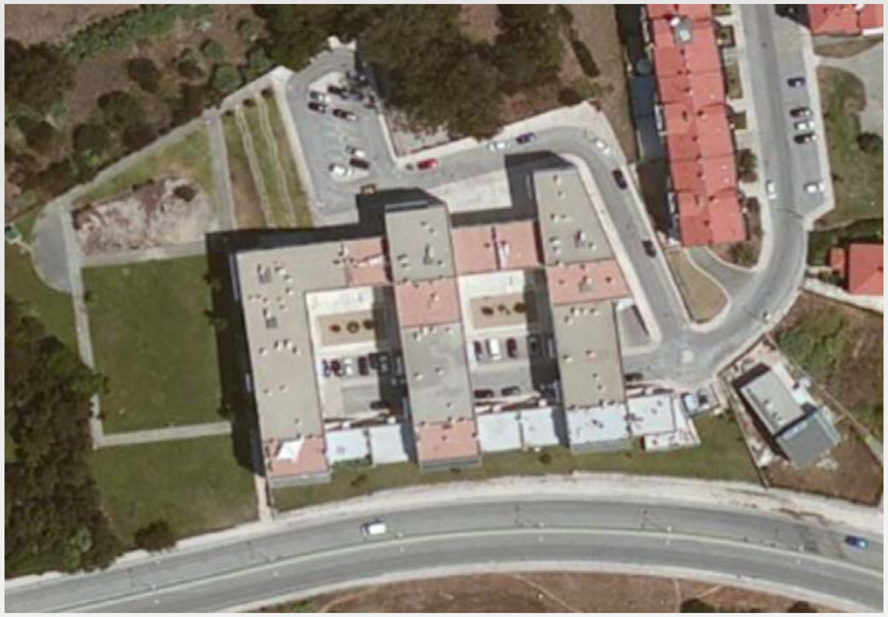
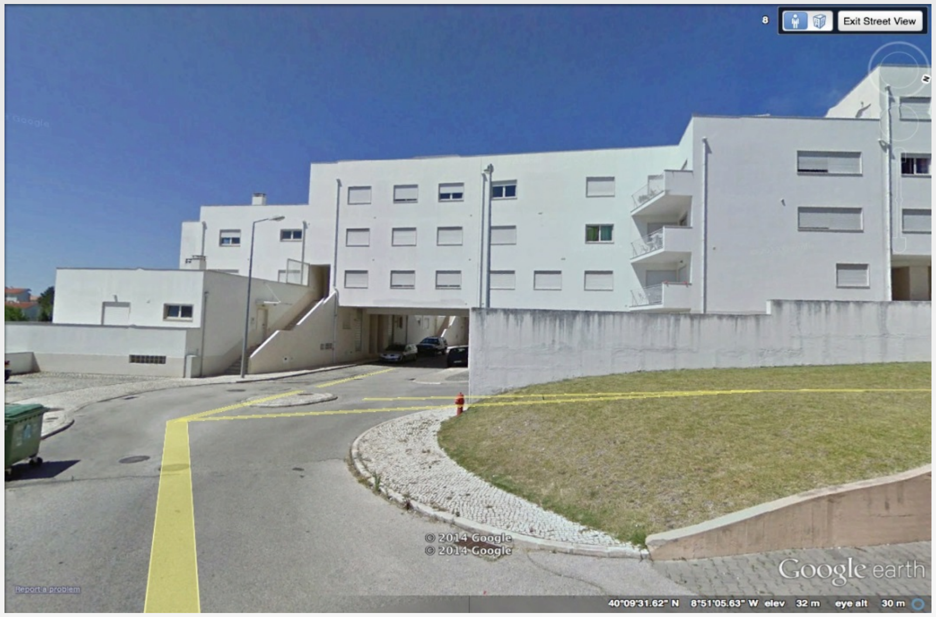
(Source: CityEngine help)

“3D Cadastral pipeline”

- Streets and lot units do not need to be generated, they can be imported from the cadastral system
  - 2D data must be converted into either ESRI Shapefile or File Geodatabase;
  - Feature geometry, attributes, and table relationships are imported;
    - Table relationships can be explored through code, CGA rules or Python.
- 3D objects extruded from 2D polygons
  - Extrusion CGA rules may be based for instance on attributes.
- Python programming
  - Attribute queries,
  - Creation of different layers corresponding to different types of data in order to be able to visualise them separately, *e.g.*:
    - different layers of legal objects within private *vs.* public spatial domains,
    - different layers of legal units per building floor,
    - …

Case study

Implementation & visualisation of a two-arch building of flats (private domain) over an urban road (public domain)

(source: GoogleEarth)

•Portuguese cadastral law: ownership rights over a given real estate on the ground also apply to open air space above and open space underground.

- Exception to the legal principle above: condominium ownership rights cannot be applied to the over ground areas underneath the arches as these represent public domain because of the road.

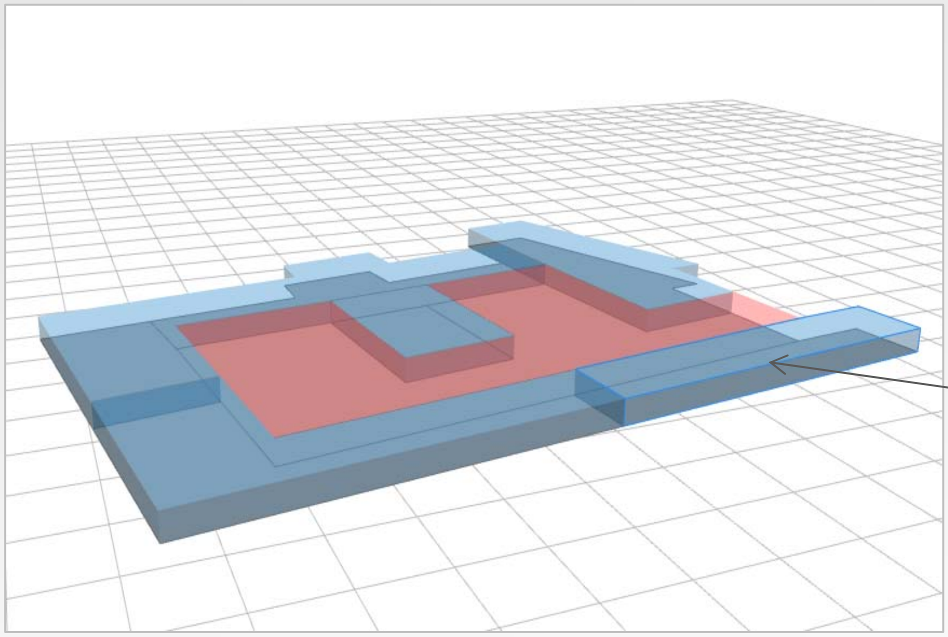
Implementation

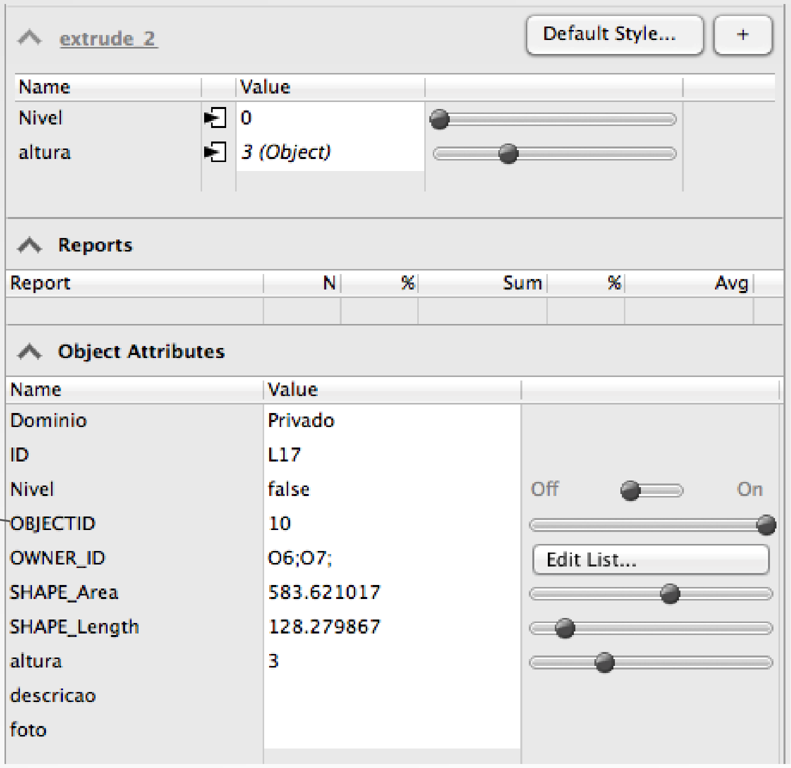
- Floor plans were outlined on a 2D map in ArcMap (ArcGIS):
  - Saved as a feature class in a File Geodatabase.
  - Each polygon feature in the feature class represents a flat or a public domain area.
  - Flats considered are not real and their spatial distribution per floor was simulated.
  - Each floor feature class was added additional attribute fields, *e.g.* floor ID, the legal entity ID, floor height (3m above ground, 4m underground), etc.
- An alphanumeric table listing all polygonal features was created:
  - Entity ID (regardless of the floor where they happen to be located):
  - Type of domain (private or public).
  - …
- An alphanumeric table listing all owners:
  - Owner ID.
  - …

3D visualisation (i)

Visualisation of a single layer corresponding to the 1st floor of the building: private domain objects in blue, public domain objects in red.

Selected object attributes (on the right).



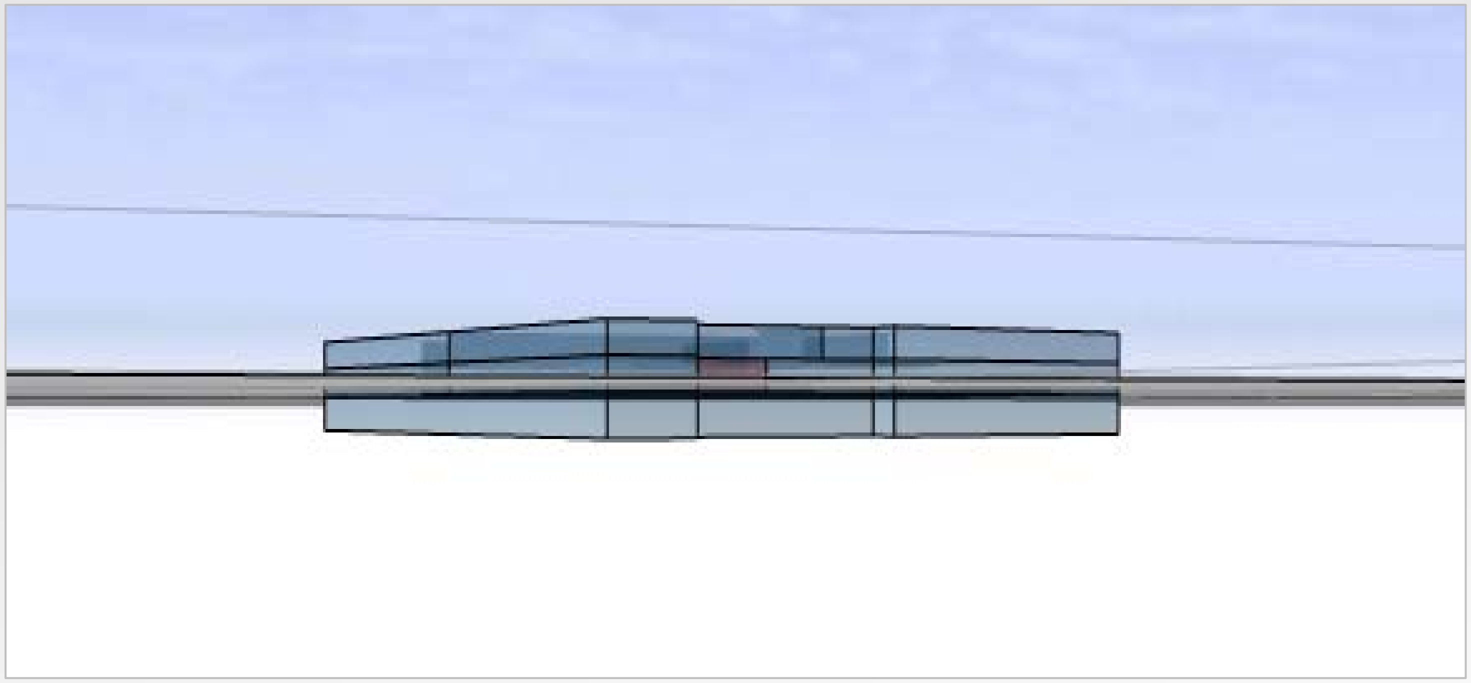


Name	Value
Nivel	0
altura	3 (Object)

Report	N	%	Sum	%	Avg

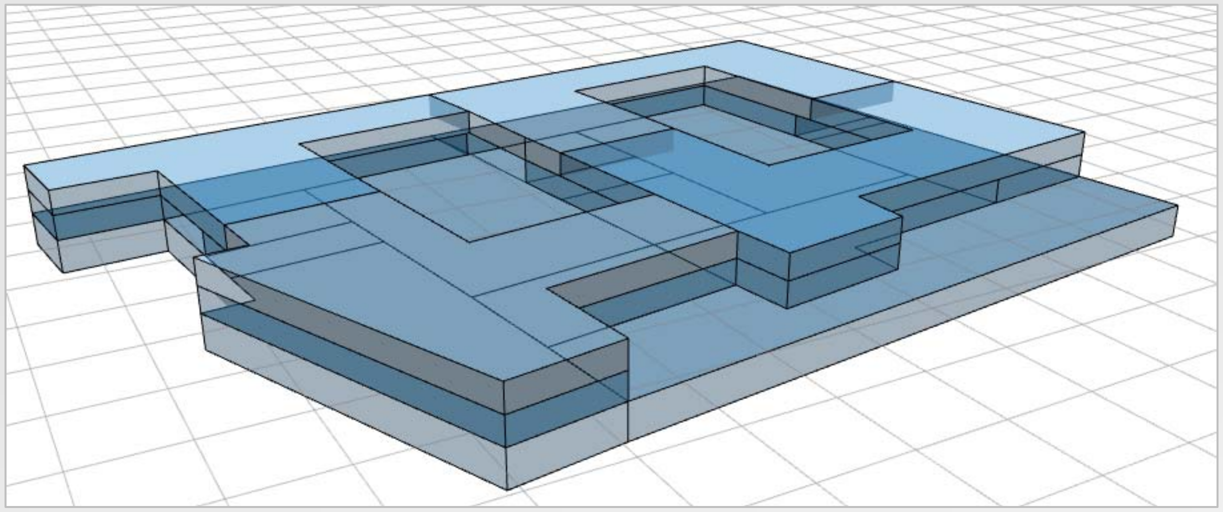
Name	Value
Dominio	Privado
ID	L17
Nivel	false
OBJECTID	10
OWNER_ID	06;07;
SHAPE_Area	583.621017
SHAPE_Length	128.279867
altura	3
descricao	
foto	

3D visualisation (ii)

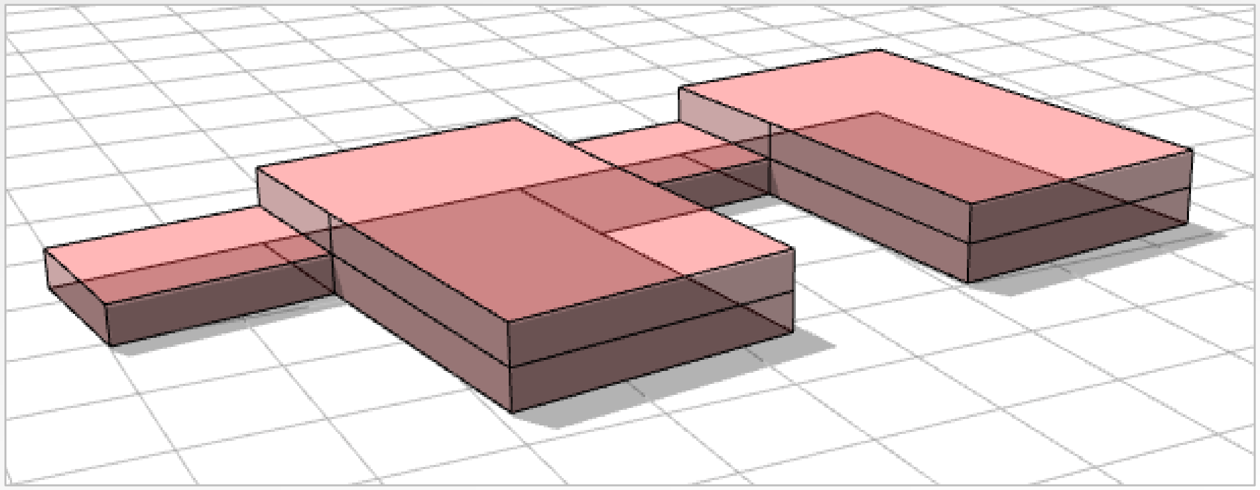


Visualisation of all floor layers: two floors above ground and one underground.

3D visualisation (iii)

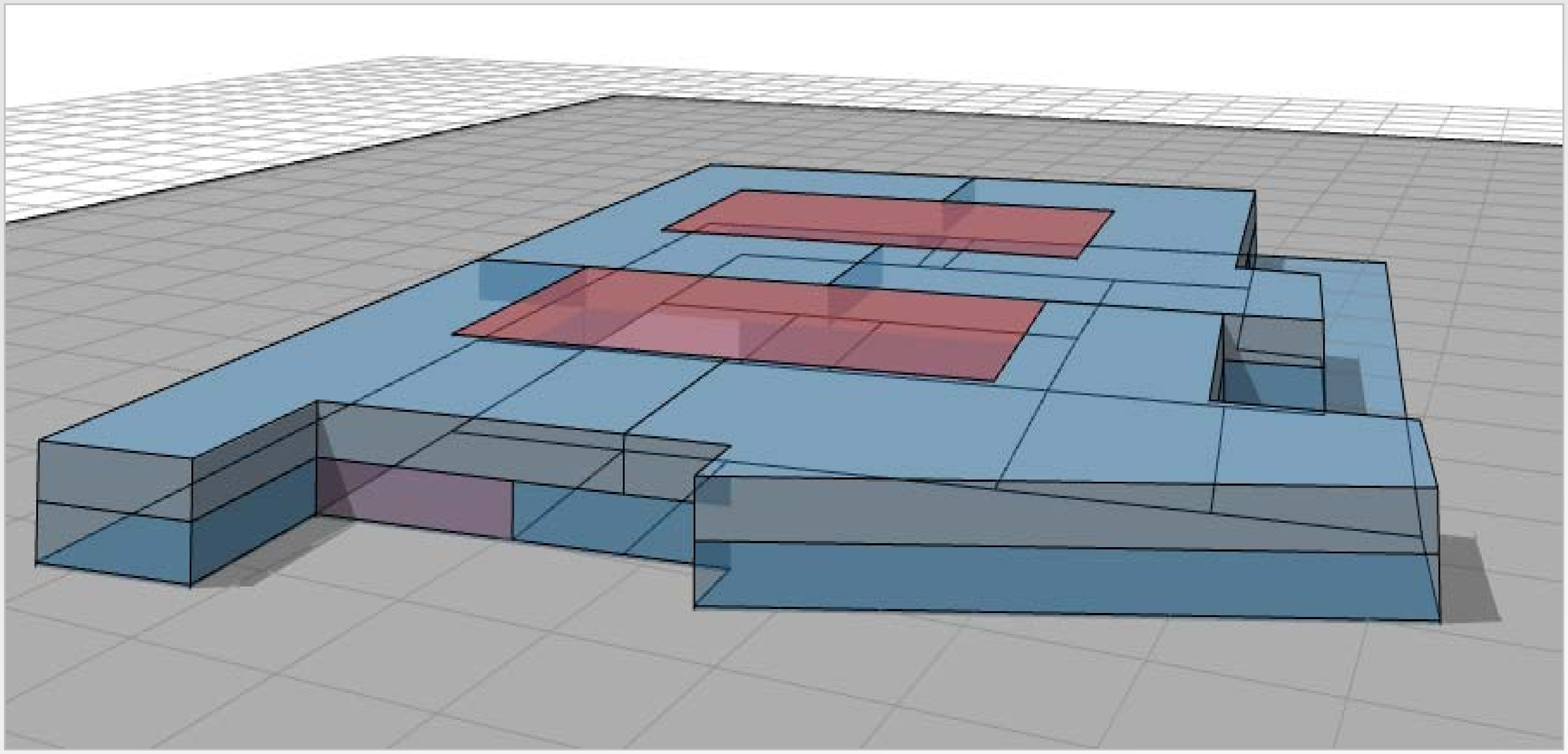


Visualisation of a single layer: “private domain” objects.



Visualisation of a single layer: “public domain” objects.

3D visualisation (iv)



Visualisation of the 1st and 2nd aboveground building floors – transparency, shadows and wireframe settings were applied.

Results (i)

Features	Visualisation requirements	Evaluation
Cadastral	Massive data handling	**Not tested**… however, actual capabilities of modelling foresee the possibility of handling massive cadastral data.
	Functions and queries	**Yes**, through programming though (Python).
	Underground view	**Yes.**
	Cross-section view	**Yes.**
	Measurements (3D)	**Yes**, through programming; there is no actual tool for direct measurements in a layer.
	Display non-spatial data	**Yes.**

Results (ii)

Features	Visualisation requirements	Evaluation
Visualisation	Interactivity	**Yes**; it is possible to set different viewing perspectives, zooming, set the scene light, view/hide shadows, view/hide textures.
	Levels of detail	**Yes**; schematic visualisation of parcels, buildings, streets, etc., object textures, indoor visualisation, vegetation, street furniture, etc.
	Symbology	**No.**
	Colour, line weight & style	**Yes**; any editable object or 3D model can be decomposed into faces (only 3D object), edges and nodes, whose display settings can be manipulated.
	Labelling	**No**; but possible to show measurements of objects in the x, y, and z directions through handles.
	Transparency	**Yes**; wire framing is also supported.
	Tooltips	**No**; properties of any selected object are displayed in the Inspector Window, including 3D coordinates and attributes.

Results (iii)

Features	Visualisation requirements	Evaluation
Non-functional	Technical diversity	**Yes**; import and export of most common 2D/3D formats, including GIS data.
	System integration and interoperability	**Yes.**
	Usability	**Moderate**. Easy to use graphical interface; however, the need of two scripting languages (CGA rules and Python) decreases usability.
	Platform independence	Windows, Mac OS X and Linux (Java based).
	Cost	Commercial solution; the learning curve is steep especially at the beginning. But, flexibility introduced by the procedural modelling and Python scripting facilitates maintenance – *i.e.* it is a matter of re-applying same CGA rules and Python code to other datasets.
	Web-based 3D visualisation	Not directly. Possibility of exporting 3D Web Scene format unto ArcGIS Online.

Results (iv)

- CityEngine is able to integrate and display
  - Aerial imagery,
  - Satellite imagery,
  - Digital terrain models.
- Connection to web services, such as WFS or WMS, is not possible.
- CityEngine is not a 3D analysis tool
  - Shadow or visibility analysis not possible to be carried out – unless content is exported and loaded unto ArcGIS 3D Analyst for visualisation, editing, and analysis purposes.
- 3D updating and manipulation
  - Possible if data stored in ESRI Shapefiles or File Geodatabases,
  - Otherwise, only rotate, scale and shift operations are allowed.

Conclusions

- Results obtained seem to be promising
  - 19 visualisation requirements:
    - 12 “yes” answers,
    - 3 “no” answers,
    - 3 “yes, but” / “no, but”,
    - One of them not tested…
  - However, more case studies have to be implemented and evaluated.
- Each individual legal unit
  - Visualisation and a certain sort of interaction is possible,
  - Visualisation of counterparts also possible.
- Combination of CGA shape grammar and Python programming
  - Sounds to be powerful and provides flexibility,
  - Facilitates maintenance, *i.e.* any block of code can be recycled.
  - But, learning curve is steep, *i.e.* may not suit all types of users.