MSc Geomatics Thesis P5

Dynamic energy simulations based on the 3D BAG 2.0

A collaboration between the 3D geoinformation research group and the Idiap Research Institute in Switzerland

Yuzhen Jin

1nd supervisor: Giorgio Agugiaro

2nd supervisor: Camilo León-Sánchez

External supervisor: Jérôme Kämpf (Idiap Research)

External supervisor: Giuseppe Peronato (Idiap Research)

Co-reader: Azarakhsh Rafiee Voermans

Delegate of the Board of Examiners: Leo van den Burg

TU Delft

Thursday, 16th June 2022

Content

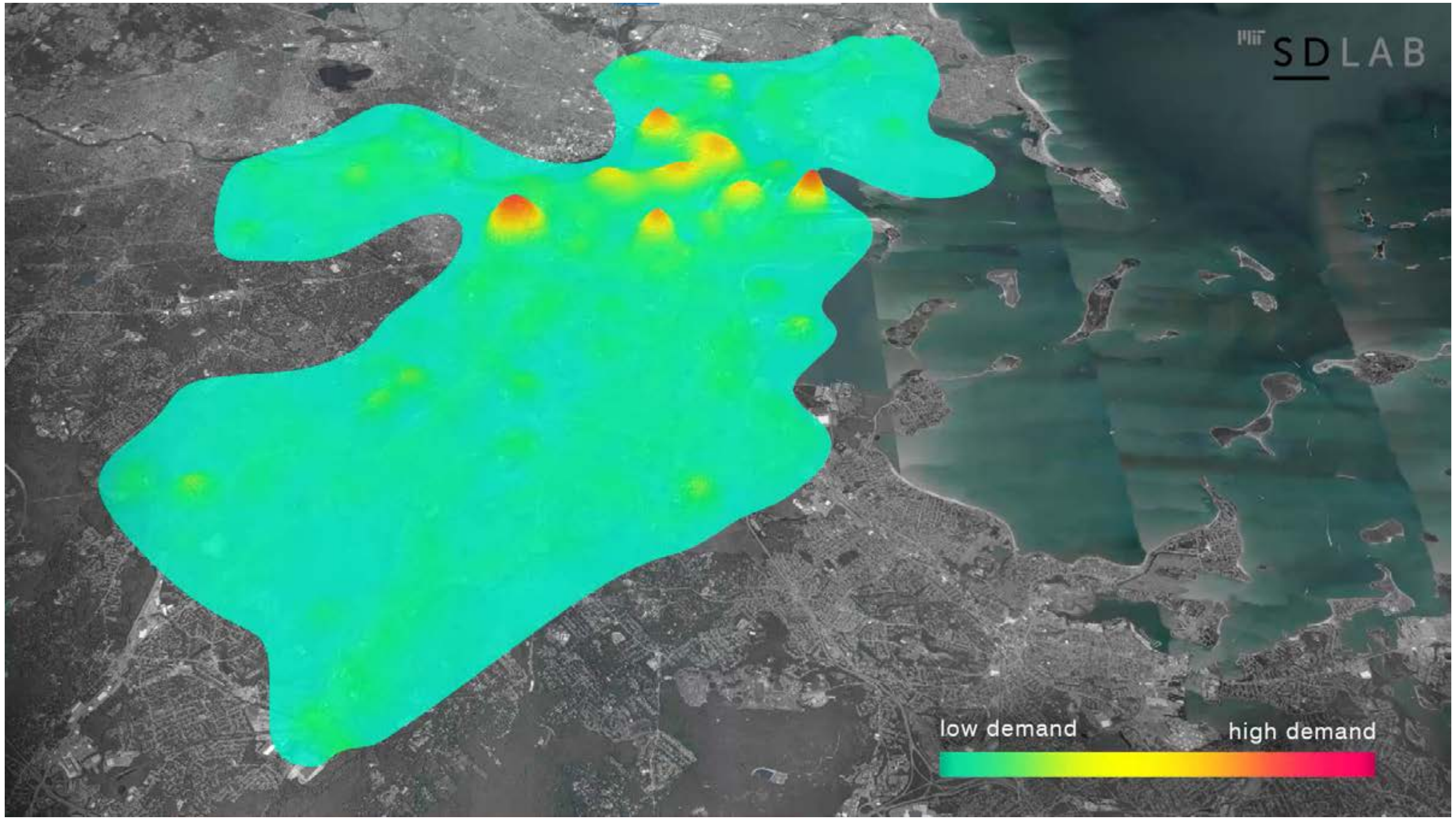
- Introduction
- Theory background and related work
- Methodology
- Data preparation
- Python implementation
- Result analysis, reflection and future work

TUDelft

Introduction

Introduction

Urban energy simulation



* From MIT Sustainable Design Lab

Introduction

Urban energy simulation

3D geometry model + Energy related parameters + Simulation tools

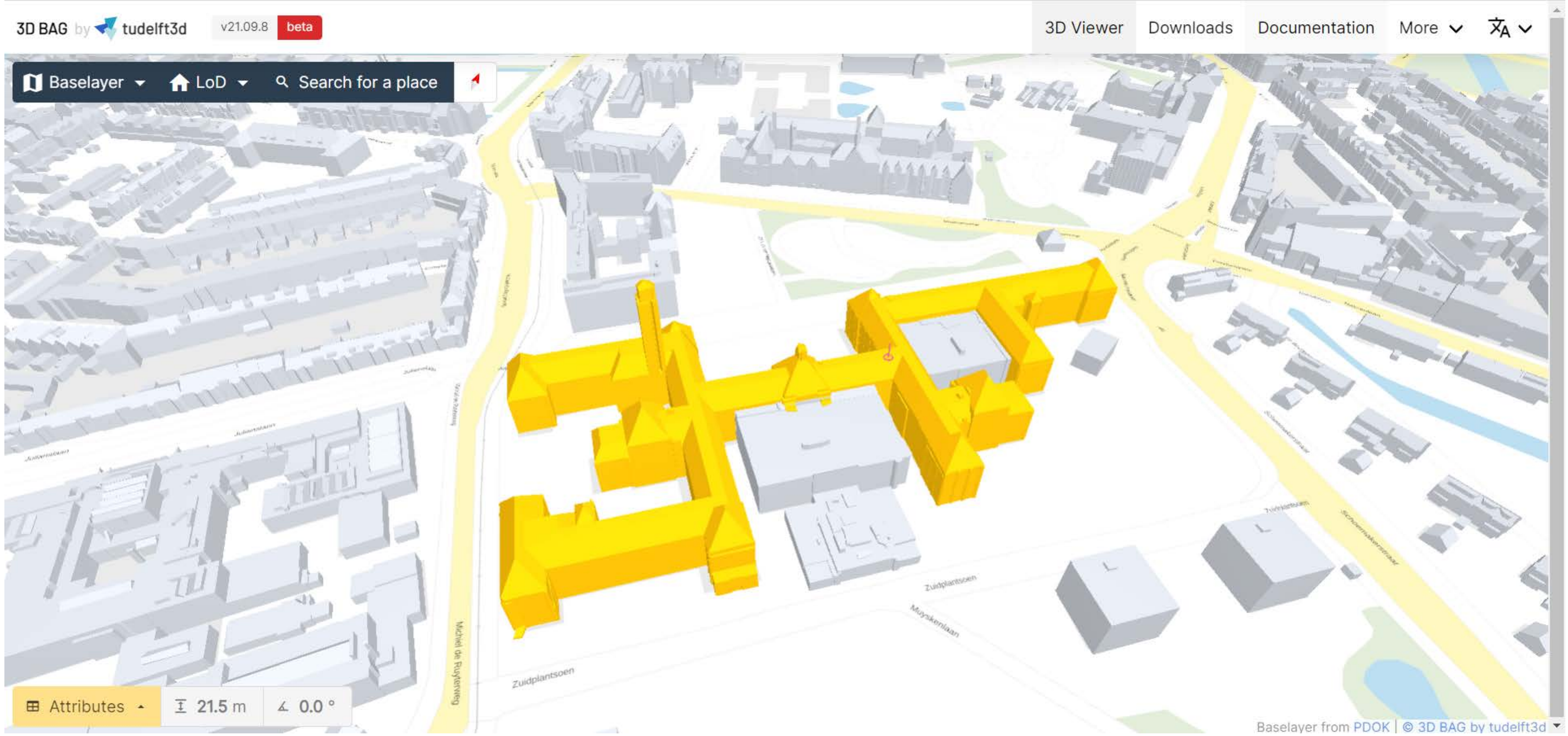
= Urban energy simulation result

e.g. heating and cooling demand, global solar irradiance, etc.

TU Delft

Introduction

3D building dataset – 3D BAG 2.0



* From 3D BAG 2.0

A dataset containing the 3D representation of buildings in several Level of Detail (LoD) of the whole Netherlands

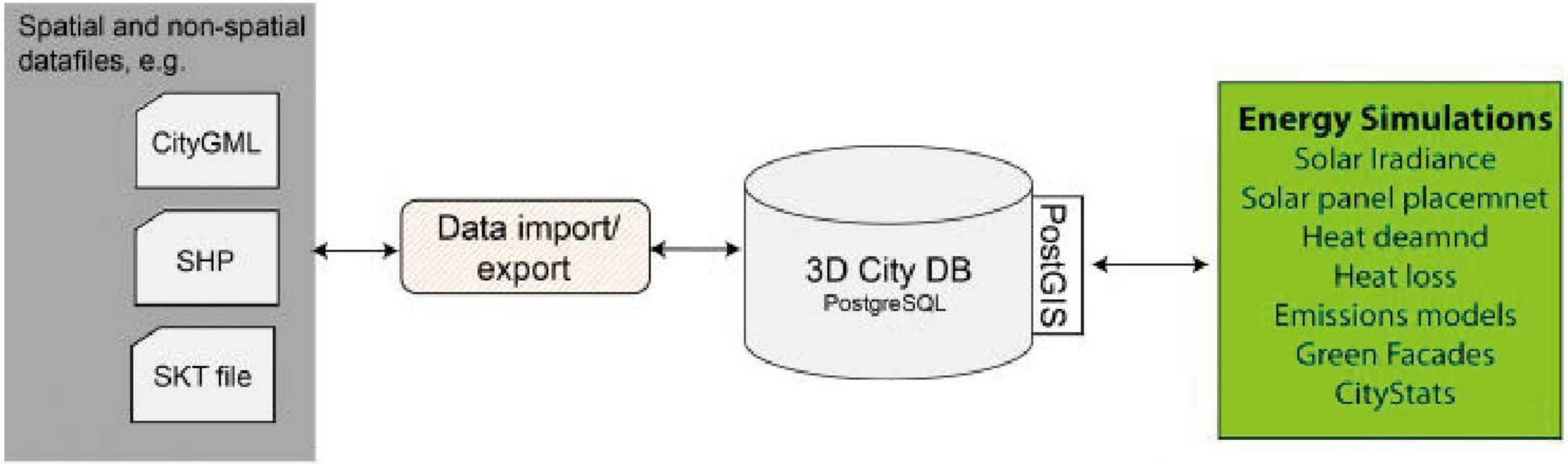
TU Delft

Introduction

3D City database – 3DCityDB

The CityGML Database
3D City DB

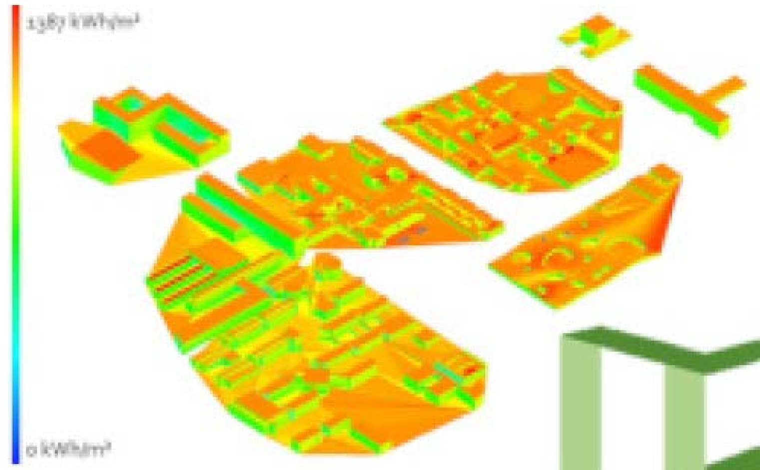
Storing, representing, and managing virtual 3D city models



TU Delft

Introduction

Varies energy simulation tool, focus on – CitySim



CitySim Pro

SimStadt

LakeSim

SynCity

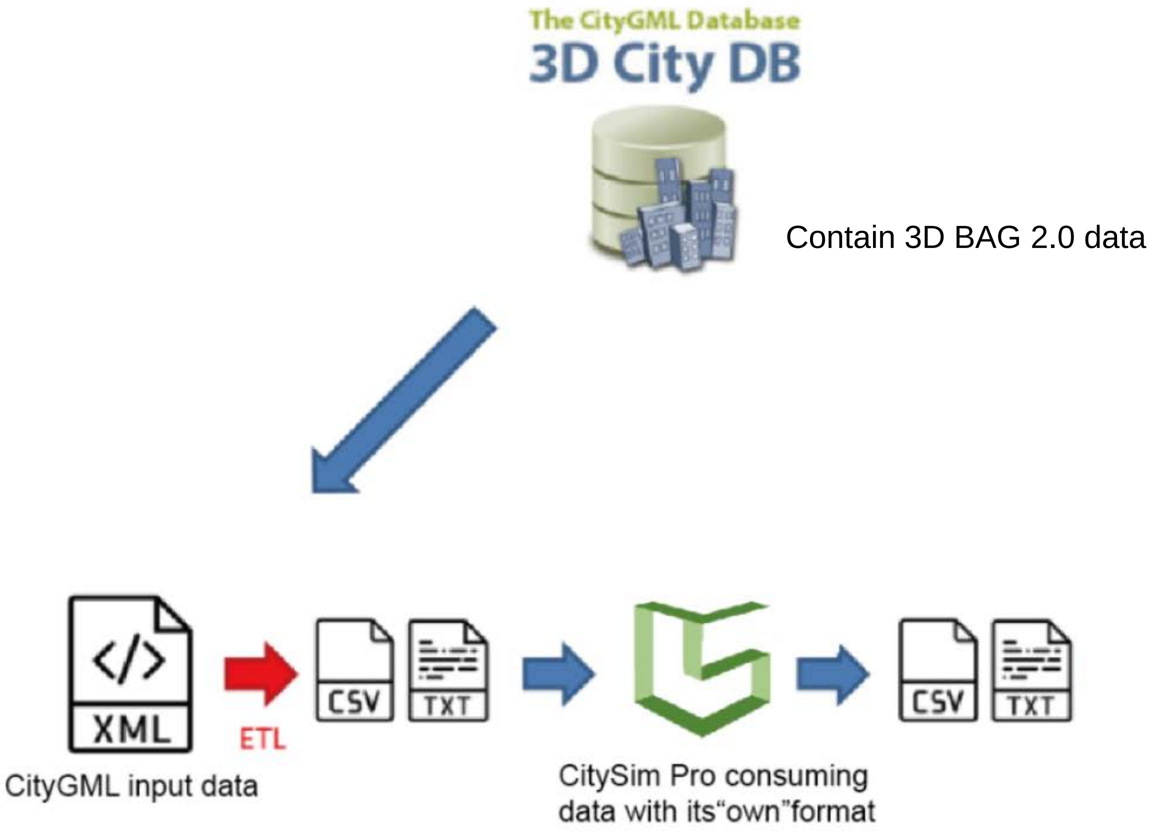
IDEAS

Simulate energy scenarios at an urban scale

TU Delft

Introduction

Motivation – Current workflow



- Too many data ETL
- Whole process not complete
- Database missing information
- No result storage
- High skill requirement for users

Introduction

Research objective

- Establish a complete database containing all urban energy related information
- Complete the whole energy simulation process, especially include the result storage
- Everything will be done in a python-based bidirectional interface

TU Delft

Introduction

Challenges

- **Database missing information**

  Physics and weather data (needed for CitySim simulation)

- **No storage schema in 3DCityDB**

  There is no storage schema to store physics and weather data library

- **The CitySim data format is special**

  Need to do several data transformation based on data formats

- **Special spatial data requirement of CitySim**

  Data pre-processing is required to fulfill the CitySim requirement

….. Varies things can be done to improve

TUDelft

Content

- Introduction
- Related work
- Methodology
- Data preparation
- Python implementation
- Result analysis, reflection and future work

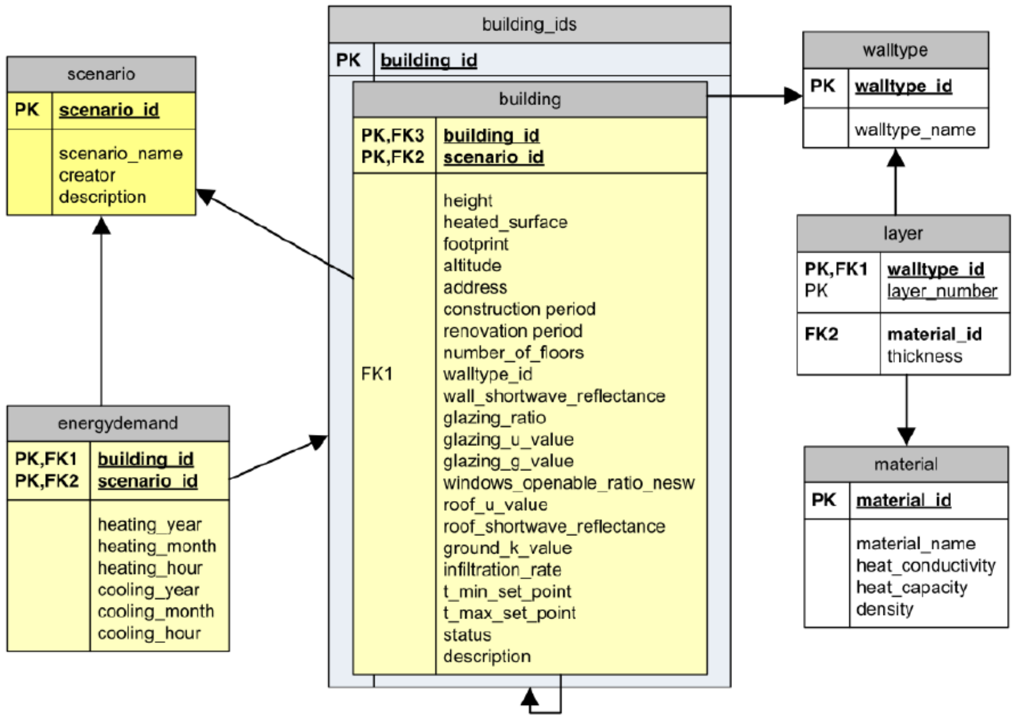
TU Delft

Related work

Case study:

CitySim simulation: a neighbourhood of Zürich city

- Storing and managing the data in a designed database
- Database and CitySim are linked via Java tool
- Database is specifically tailored to CitySim
- No spatial and non-spatial data processing functions



- *The schema of the designed CitySim database*

Related work

Case study:

Machine learning for the energy of buildings on a GIS tool design project 2021

- Storing and managing the data in 3D City Database
- Database and CitySim are linked via Python tool
- Not finished yet
- Data collection is not complete
- No spatial and non-spatial data processing functions
- ……

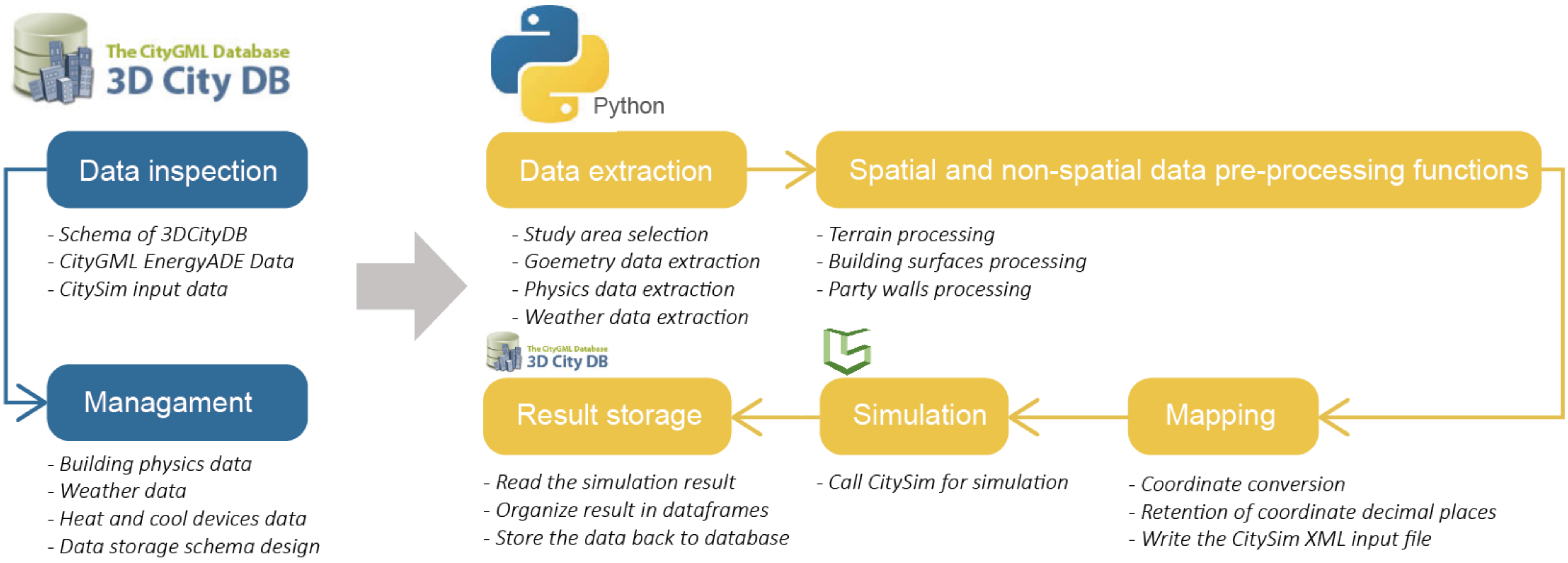
TUDelft

Content

- Introduction
- Theory background and related work
- Methodology
- Data preparation
- Python implementation
- Result analysis, reflection and future work

TUDelft

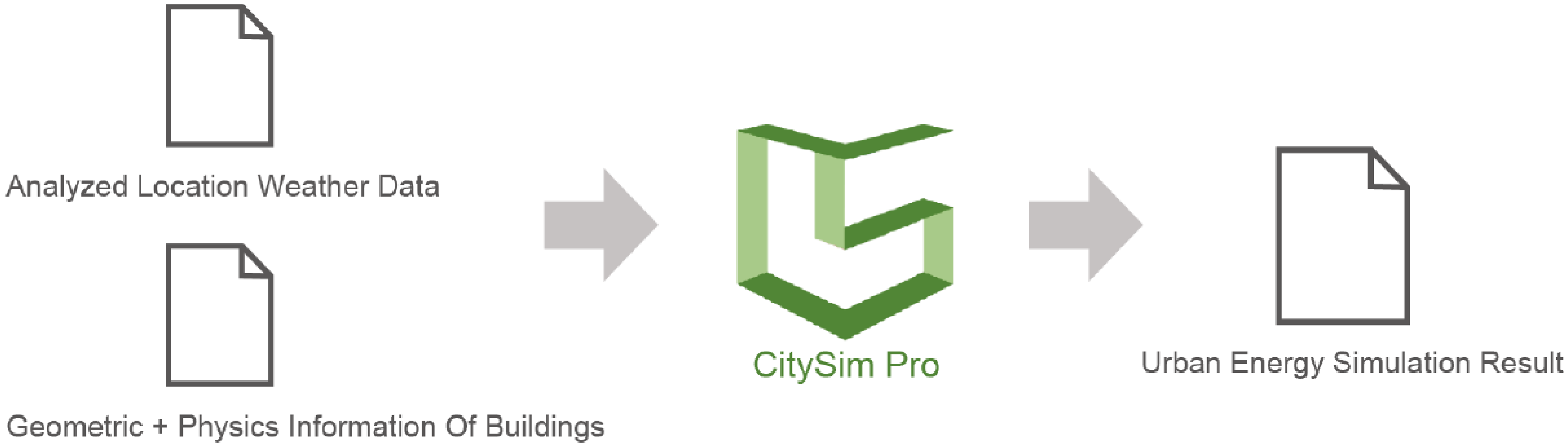
Methodology



TU Delft

Methodology

Management: Data storage structure design methodology

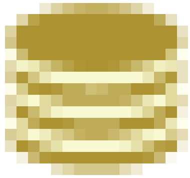


Methodology

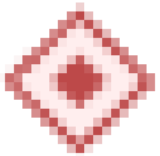
Management: Data storage structure design methodology

The CityGML Database 3D City DB

Include schemas for storing 3D city models



Database

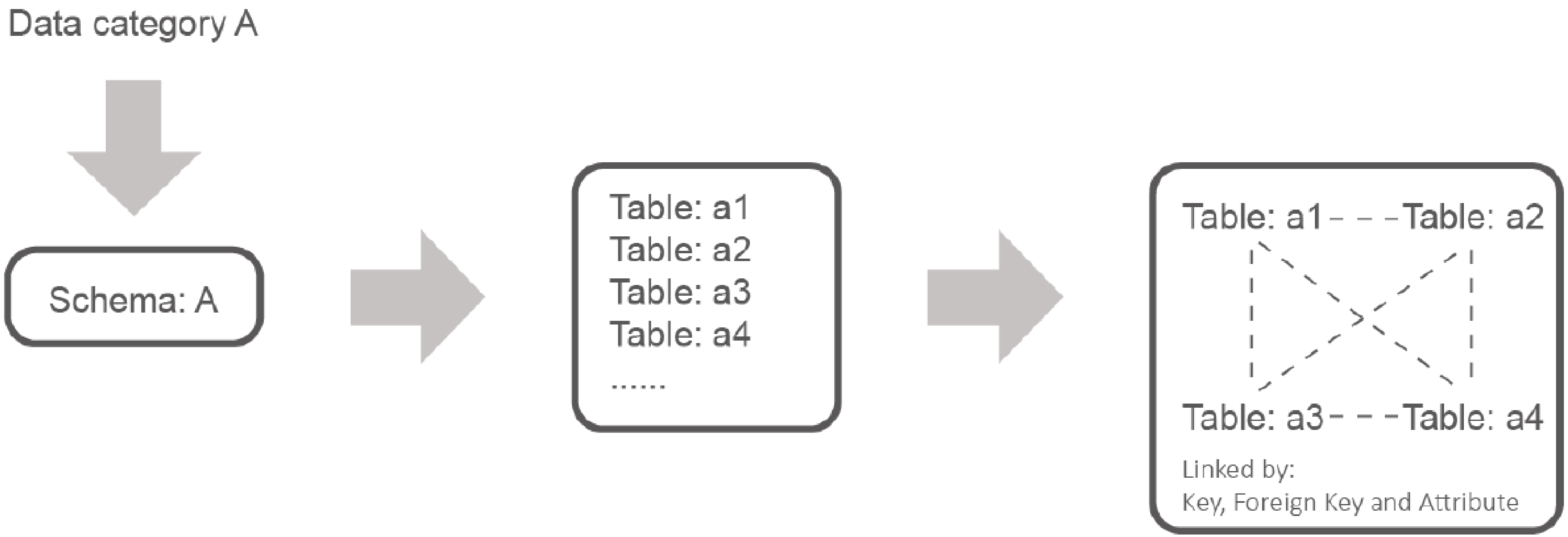


The other designed schemas for storing the library information (e.g. physics and weather library)

TU Delft

Methodology

Management: Data storage structure design methodology

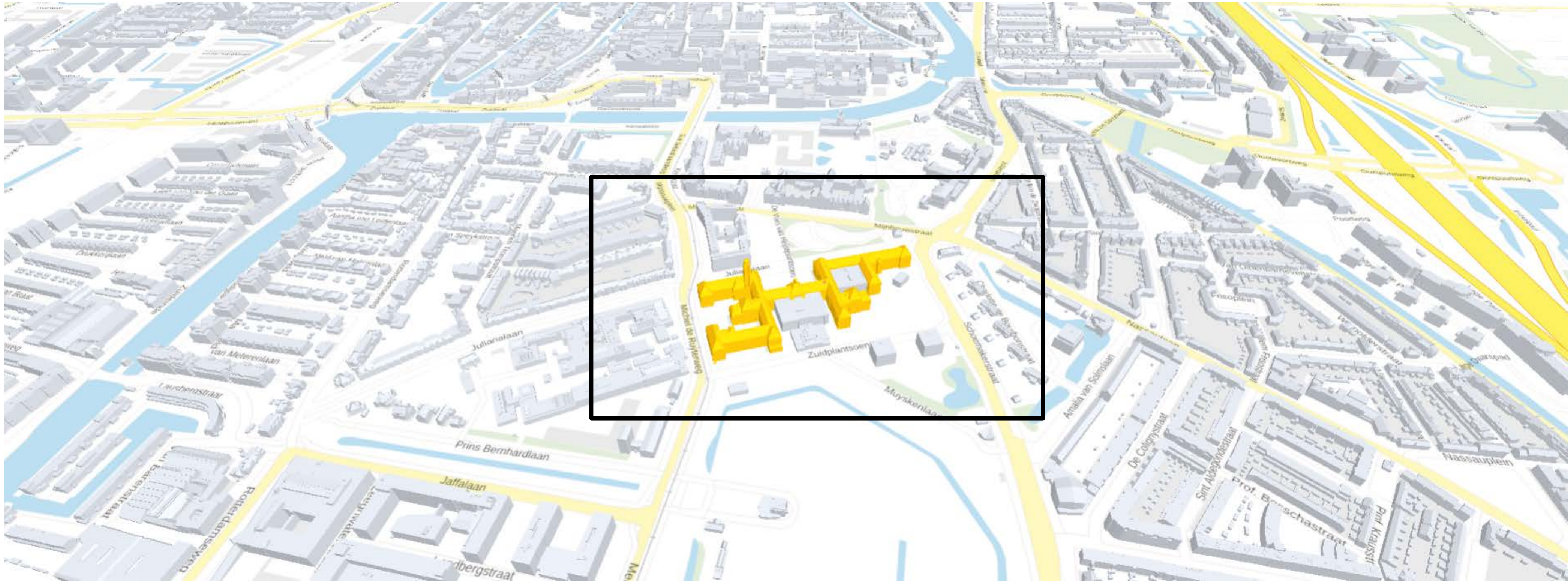


TUDelft

Methodology

Spatial and non-spatial data pre-processing functions:

Study area selection

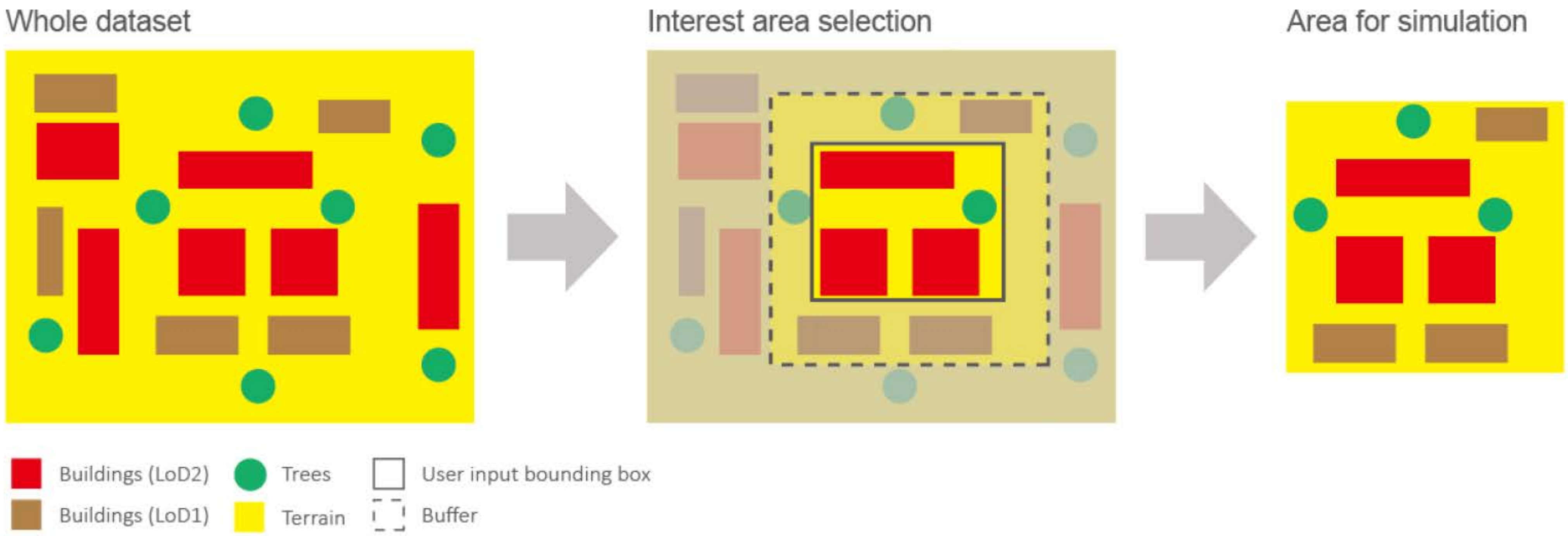


Drastically reduce the file size and simulation time

Methodology

Spatial and non-spatial data pre-processing functions:

Study area selection

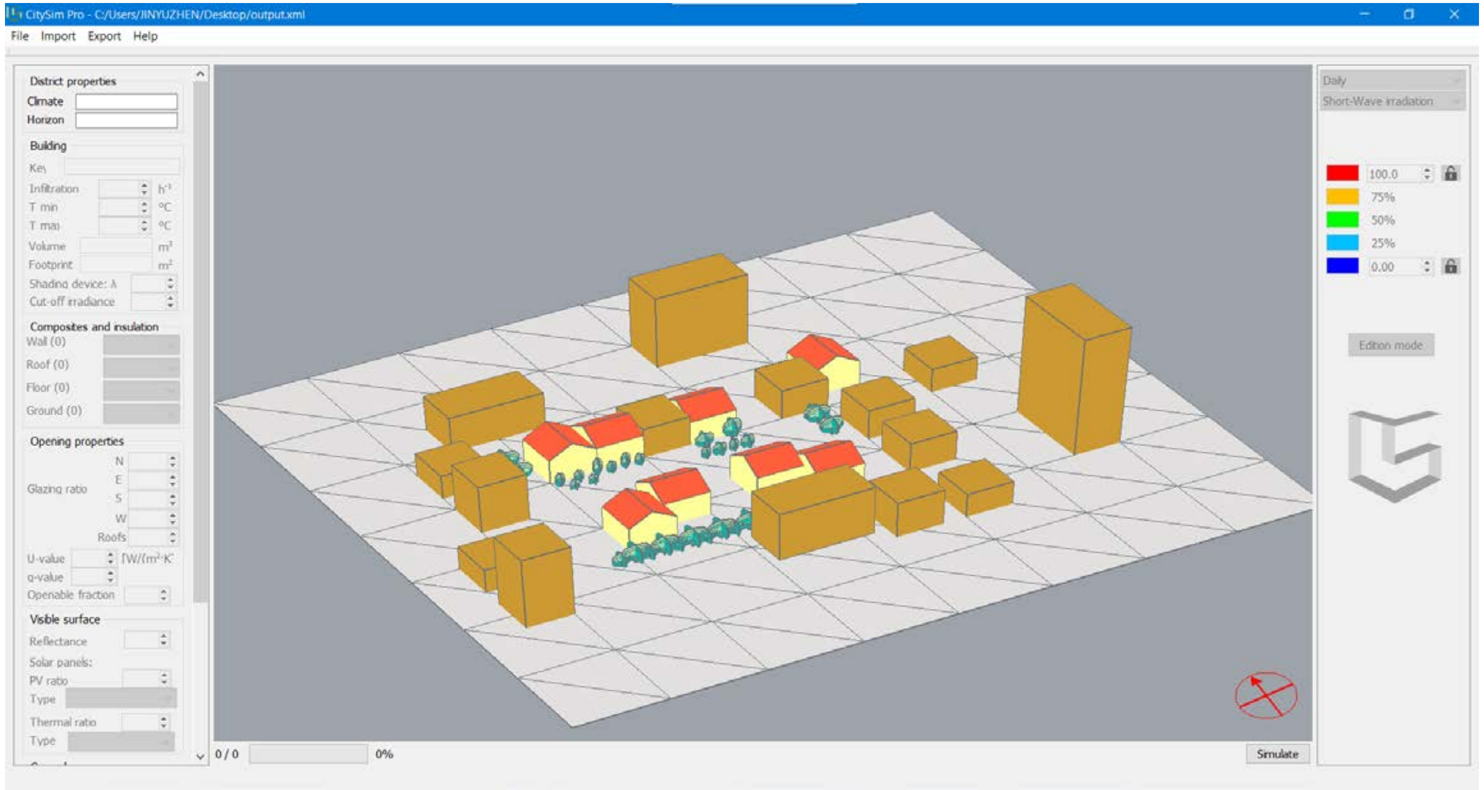


Users enter their preferred study area bounding box with a buffer for surrounding information in python interface

Methodology

Spatial and non-spatial data pre-processing functions:

Shading surfaces processing



Replacing the Non-simulated buildings from high level of detail to low level of detail

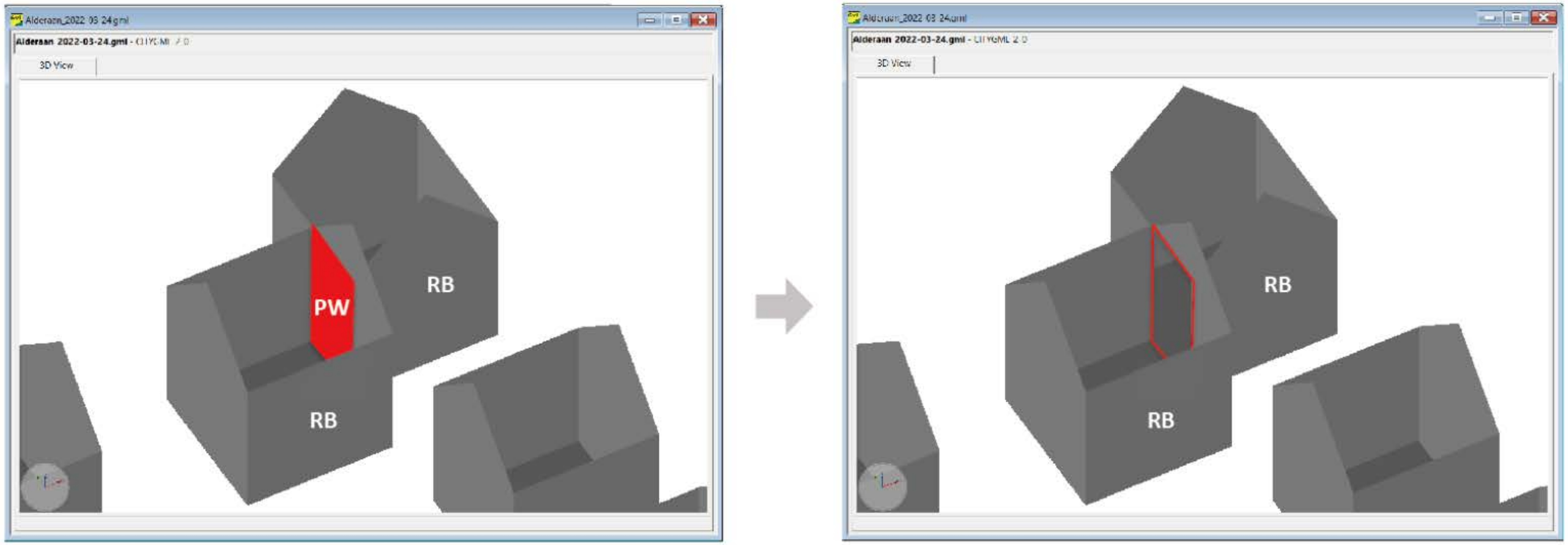
Drastically reduce the file size and simulation time

TU Delft

Methodology

Spatial and non-spatial data pre-processing functions:

Party wall processing



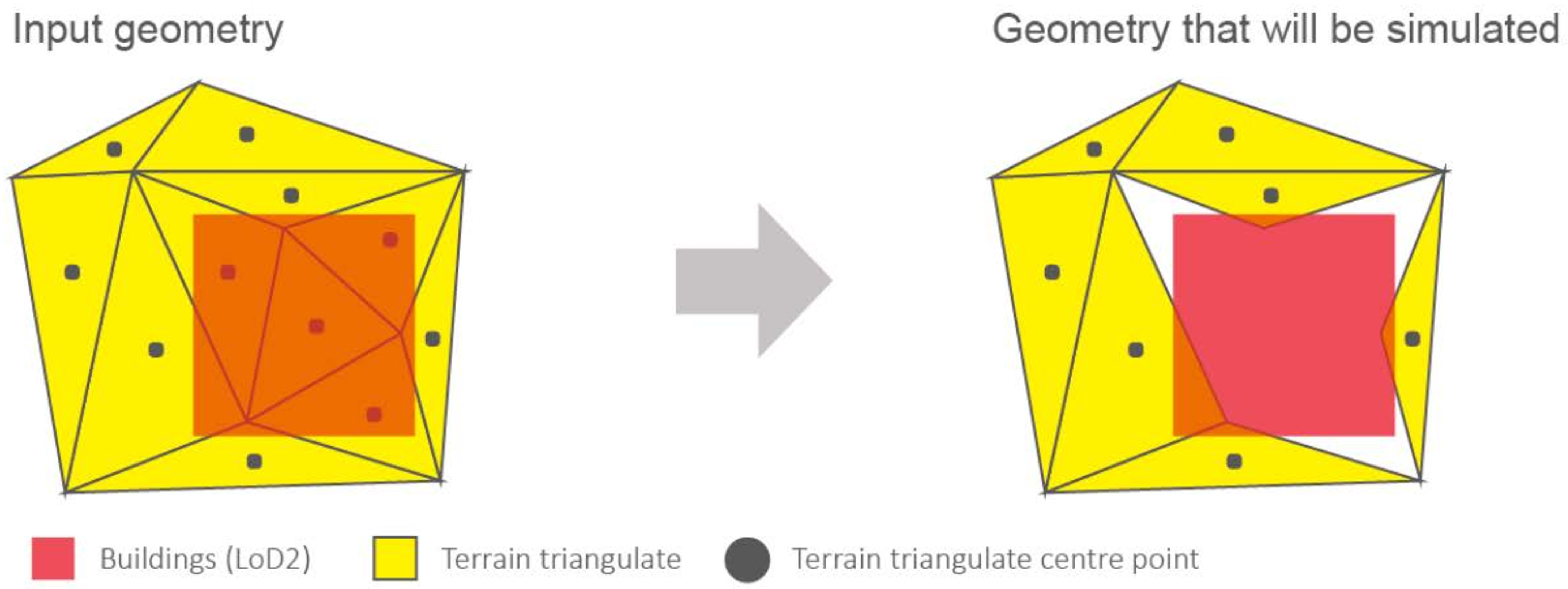
RB, residnetial building PW, party wall

Locate party walls that can be removed and remove them

Methodology

Spatial and non-spatial data pre-processing functions:

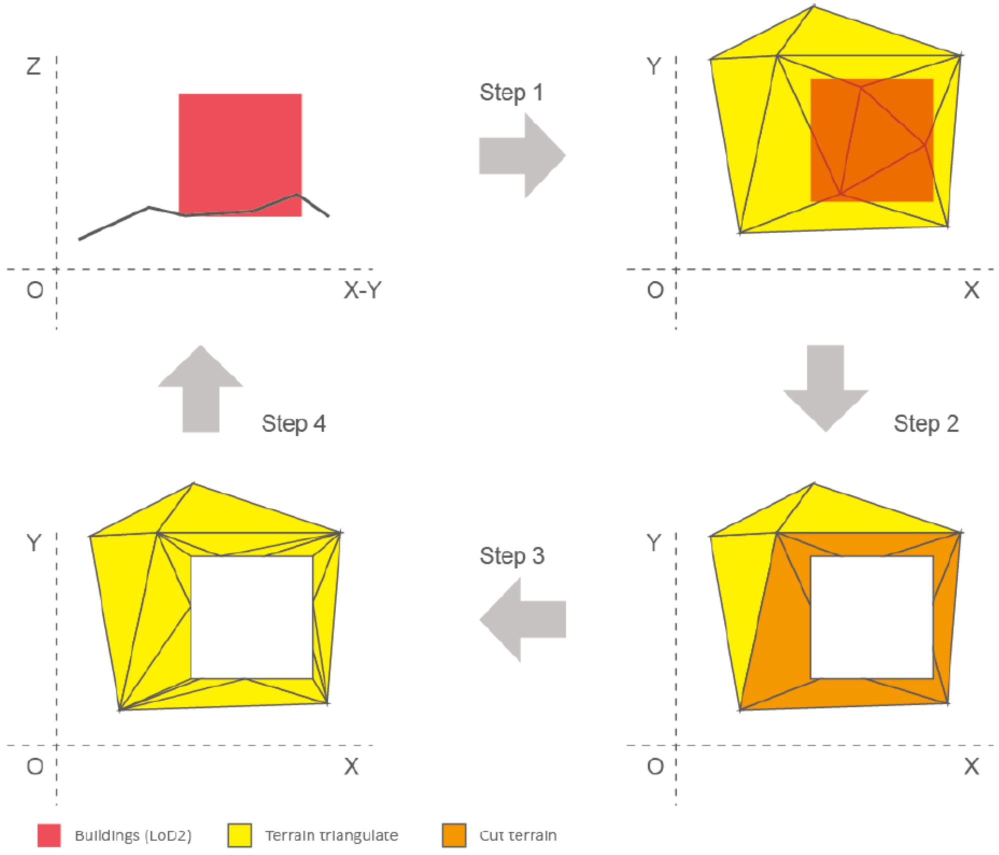
Terrain processing



Methodology

Spatial and non-spatial data pre-processing functions:

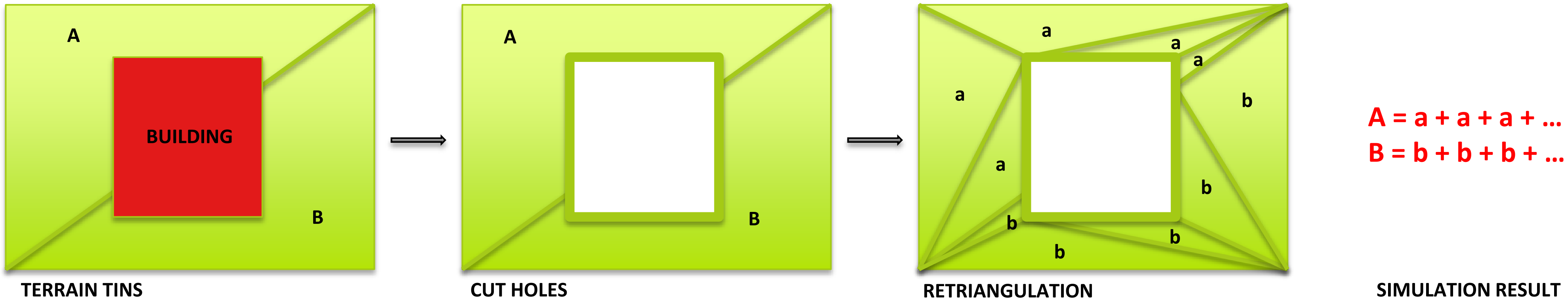
Terrain processing



Methodology

Spatial and non-spatial data pre-processing functions:

Terrain processing

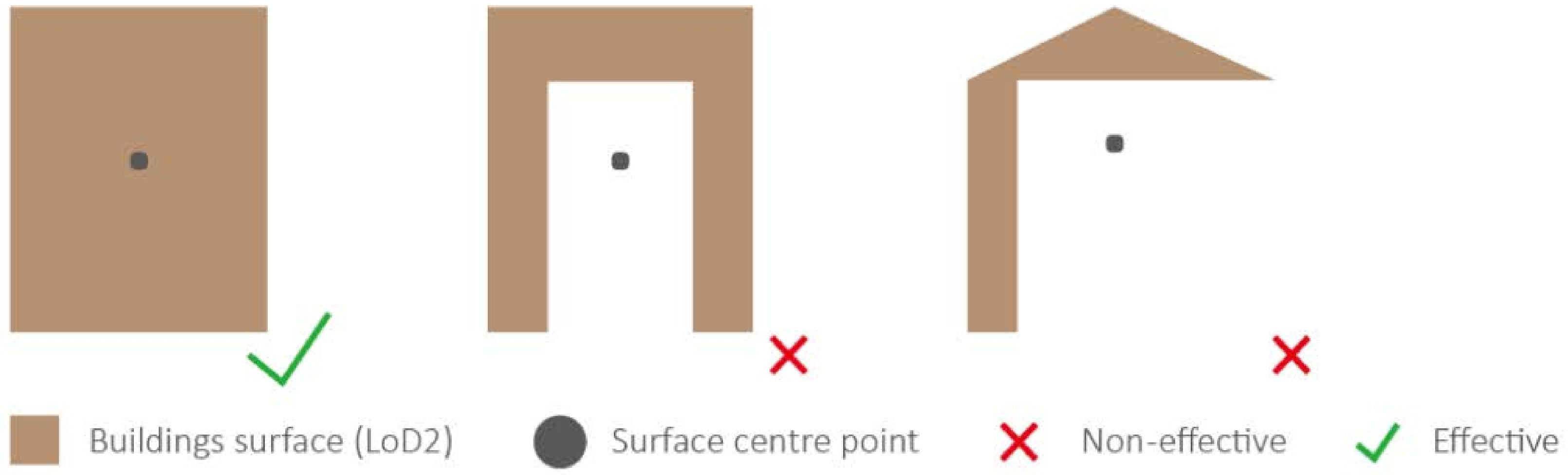


Terrain pieces are labeled with the same id as their 'parent' for result aggregation

Methodology

Spatial and non-spatial data pre-processing functions:

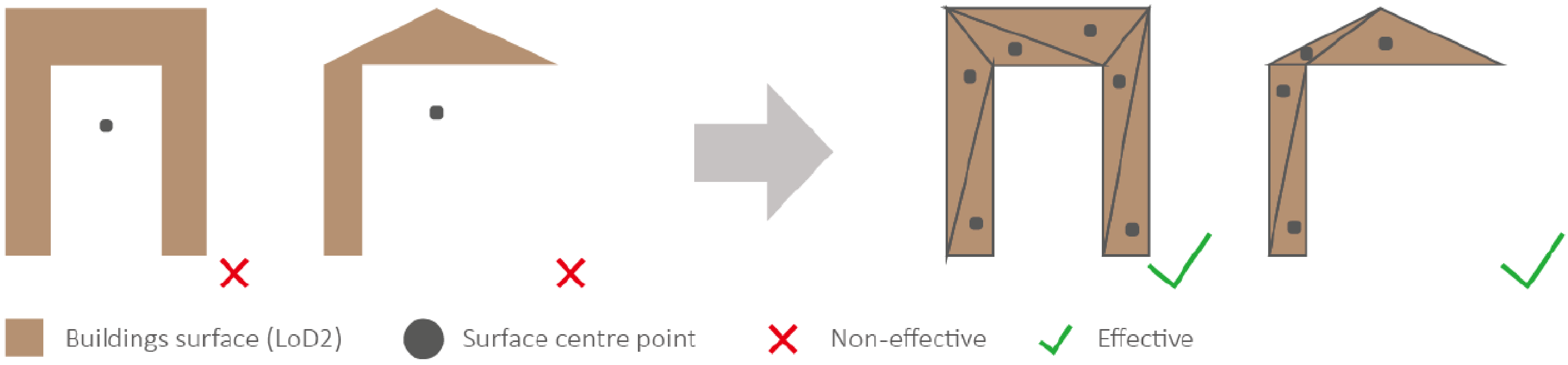
Building surface processing



Methodology

Spatial and non-spatial data pre-processing functions:

Building surface processing



Content

- Introduction
- Theory background and related work
- Methodology
- Data preparation
- Python implementation
- Result analysis, reflection and future work

TU Delft

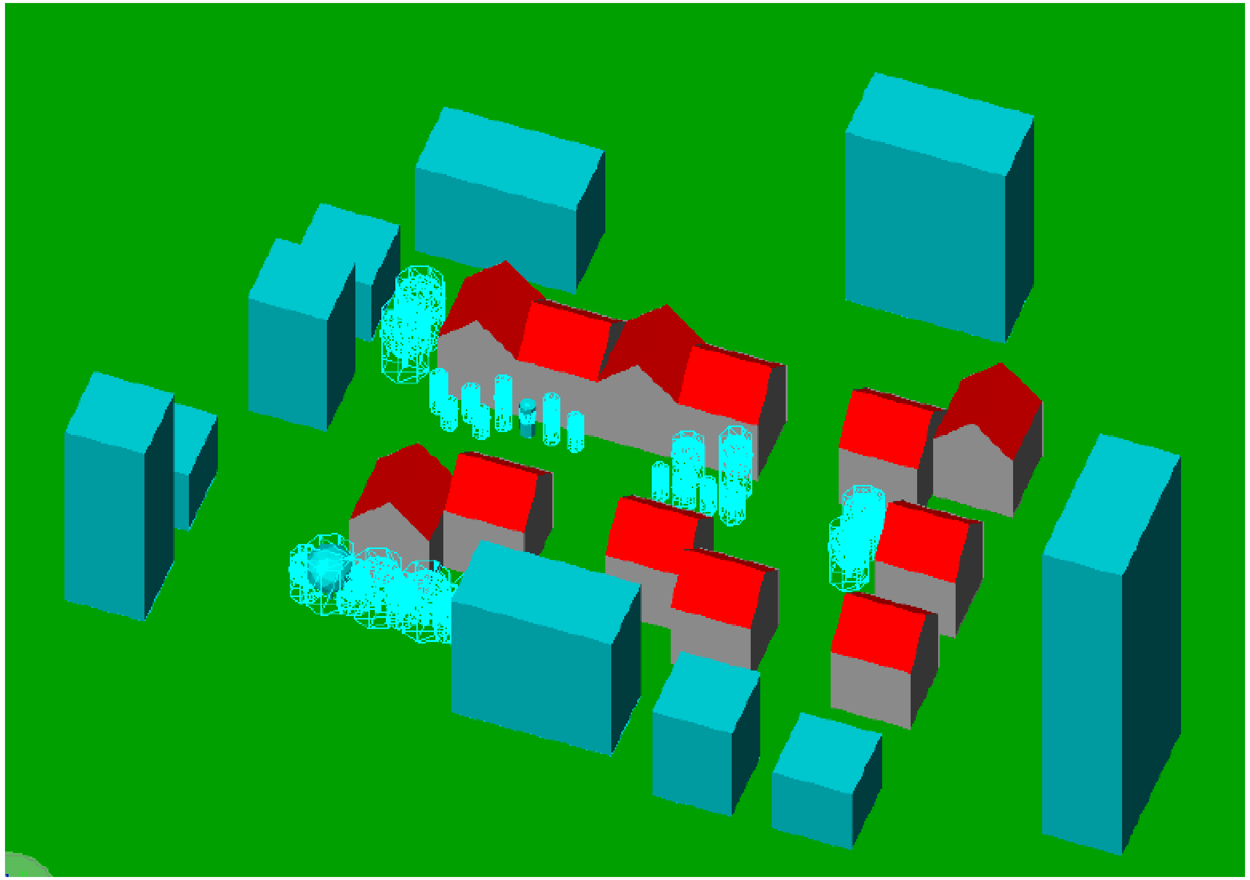
Data preparation

Required data	Existing data	Data source	Storage method
building geometry	Alderaan.gml	Testing model	Through 3DCityDB importer/exporter
building energy related information	Alderaan.gml	Testing model	Through 3DCityDB importer/exporter
building physics	Dutch_building _physics.xml	TABULA	Storage schema design and through pgAdmin
tree geometry	Alderaan_Trees.gml	Testing model	Through 3DCityDB importer/exporter
tree physics	NaN	CitySim example parameters	Storage schema design and through pgAdmin
terrain geometry	Alderaan_DTM.gml	Testing model	Through 3DCityDB importer/exporter
terrain physics	NaN	CitySim example parameters	Storage schema design and through pgAdmin
weather	NaN	Climate.OneBuilding.Org	Storage schema design and through pgAdmin
heat and cool information	NaN	CitySim example parameters	Storage schema design and through pgAdmin

TU Delft

Data preparation

Geometry data, used for developing interface



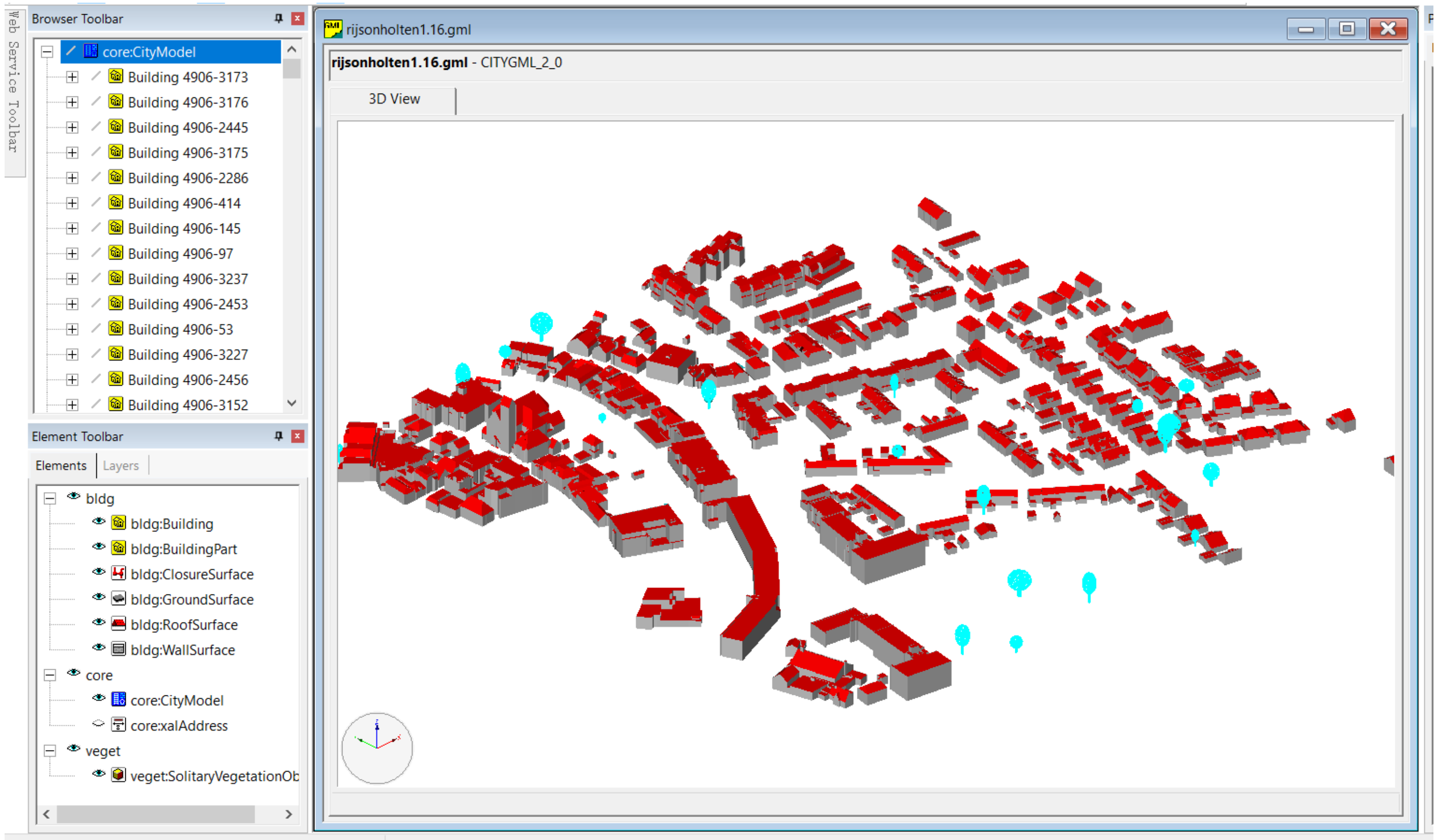
Alderaan.gml

- 11 LoD2 buildings (include 1 multi-part building)
- A group of LoD1 ancillary buildings
- Trees modelled in LoD1, LoD2 and LoD3
- Terrain are modelled as tiled TIN

attributes	UOM	description
lod2_volume	m^3	Involved in citysim energy simulation calculations
num_residents		Involved in citysim energy simulation calculations
building_type		Link the building physic. e.g. SFH (single family house)
function		Link the building physics and filter out non-residential buildings e.g. residential building
year_of_construction		Link the building physics, e.g. 1955

Data preparation

Geometry data, used for final testing



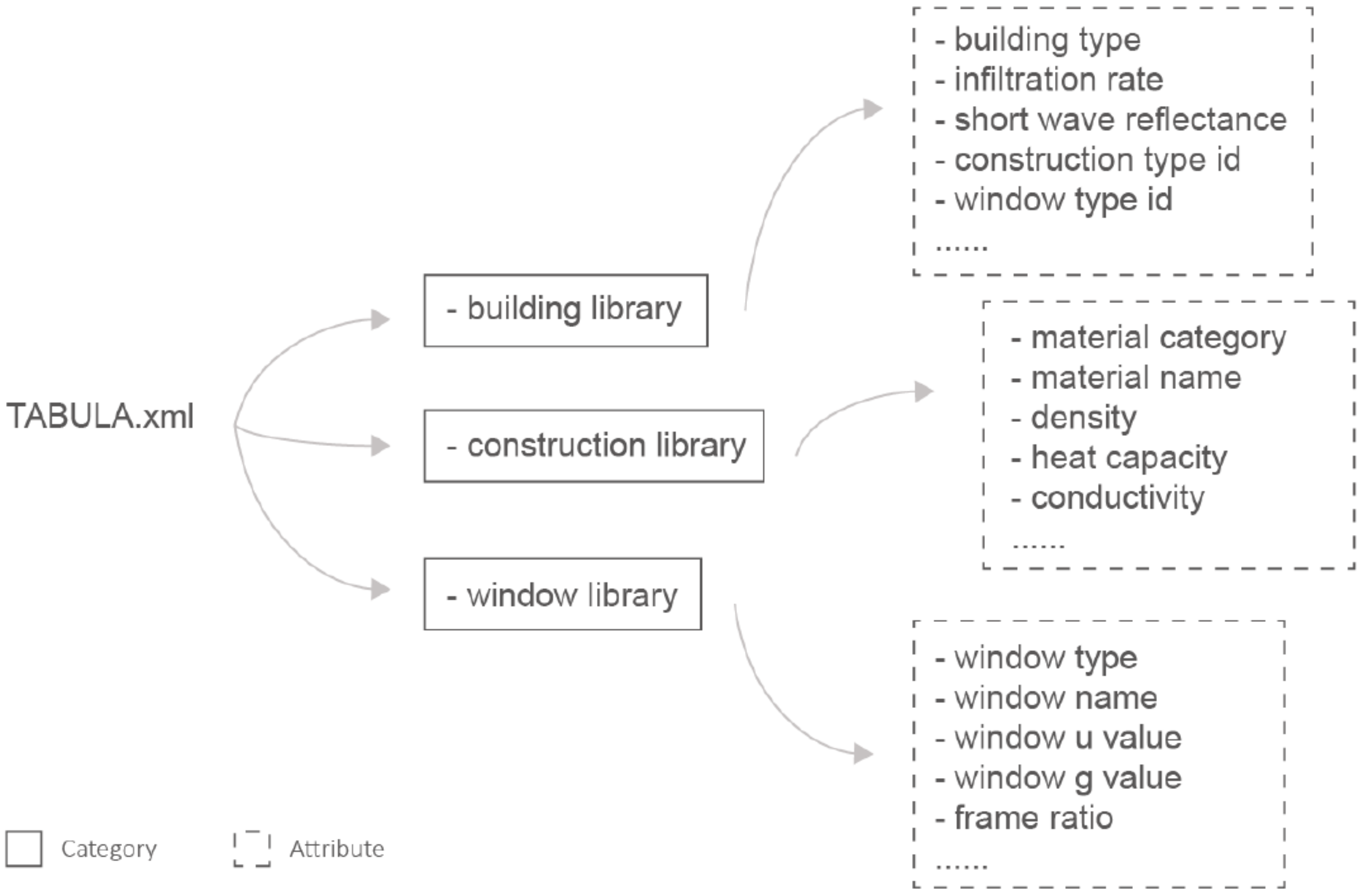
Real dataset from bag 2.0, RijssenHolten.gml (1/16 of the original)

Data preparation

Physics data library

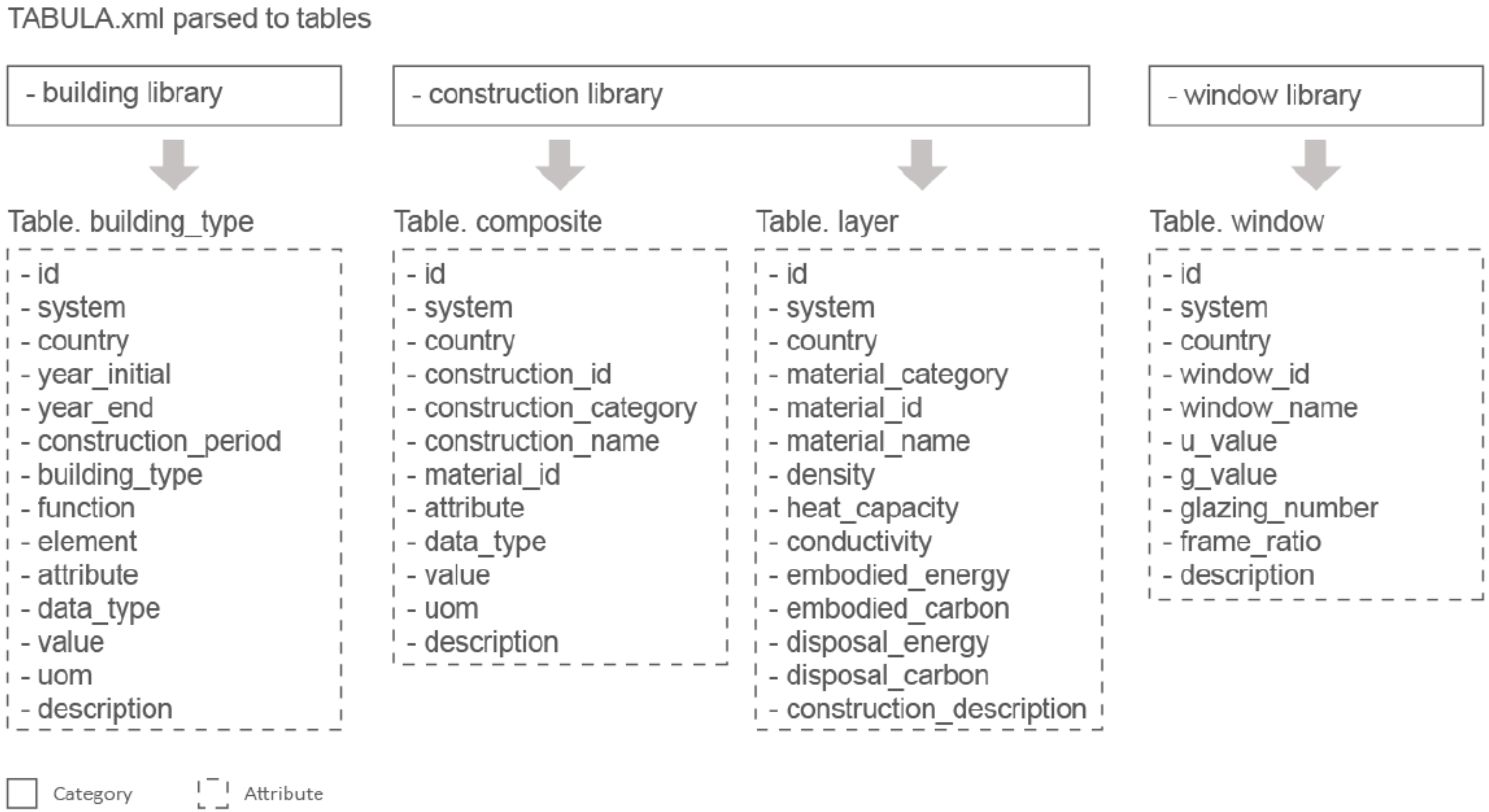
Data preparation

Physics data library



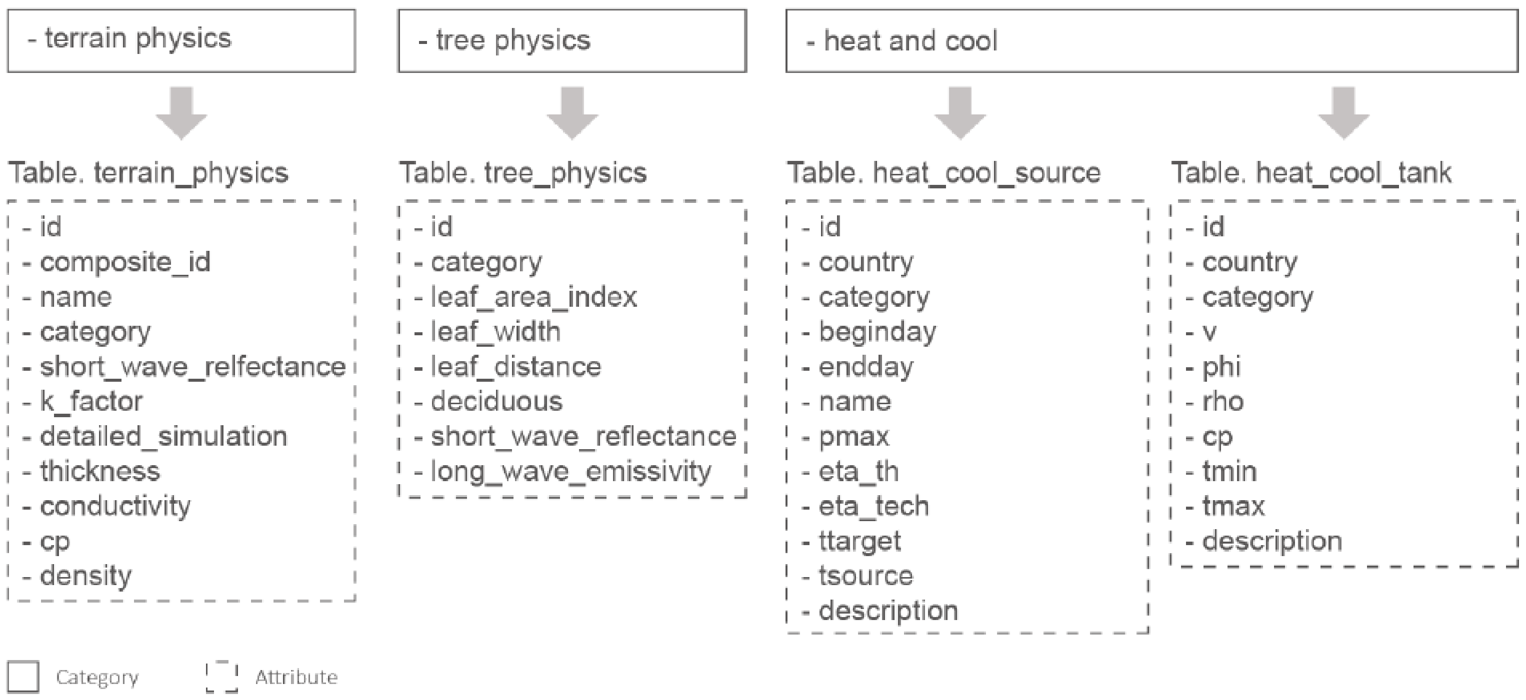
Data preparation

Physics data library



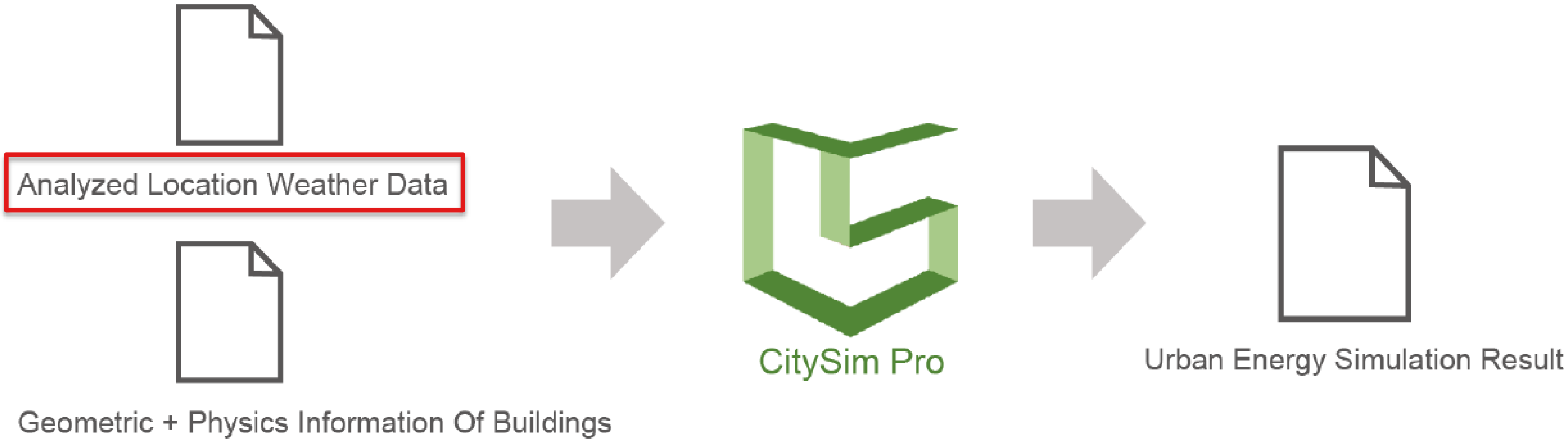
Data preparation

Physics data library



Data preparation

Weather data library



Data preparation

Weather data library

	CLI file requirement	UOM	EPW available	UOM
d	Day		Date – Day	
m	Month		Date – Month	
h	Hour		Time – Hour	
G_Dh	Diffuse horizontal irradiance	W/m2	Diffuse Horizontal Radiation	Wh/m2
G_Bn	Beam (solar) normal irradiance	W/m2	Direct Normal Radiation	Wh/m2
Ta	Air temperature	°C	Dry Bulb Temperature	°C
Ts	Ground surface temperature	°C	Dry Bulb Temperature	°C
FF	Wind Speed	m/s	Wind Speed	m/s
DD	Wind Direction	°	Wind Direction	°
RH	Relative Humidity	%	Relative Humidity	%
RR	Precipitation	mm	Liquid Precipitation Depth	mm
N	Nebulosity	Octas	Total Sky Cover	tenths

From: Climate.OneBuilding.Org

TU Delft

Data preparation

Weather data library

	dm	m	h	G_Dh	G_Bn	Ta	Ts	FF	DD	RH	RR	N
0	1	1	1	0	0	5.0	5.0	5.0	171	84	0.0	0
1	1	1	2	0	0	5.0	5.0	4.0	178	83	0.0	1
2	1	1	3	0	0	5.4	5.4	4.0	186	83	0.0	7
3	1	1	4	0	0	6.7	6.7	4.0	199	80	0.0	7
4	1	1	5	0	0	6.5	6.5	4.0	214	82	0.0	7
...	...	...	...	...	...	...	...	...	...	...	...	...
8755	365	12	20	0	0	6.9	6.9	6.0	220	86	0.0	8
8756	365	12	21	0	0	7.3	7.3	6.0	220	84	0.0	8
8757	365	12	22	0	0	7.8	7.8	6.0	220	84	0.0	8
8758	365	12	23	0	0	8.2	8.2	7.0	220	82	0.0	8
8759	365	12	24	0	0	8.4	8.4	7.0	220	83	0.0	8

8760 rows × 12 columns

.epw → .cli

TU Delft

Data preparation

Weather data library

- EPW climate file

Table. weather_location

- city
- state_province_region
- country
- wmocode
- latitude
- longitude
- altitude
- timezone

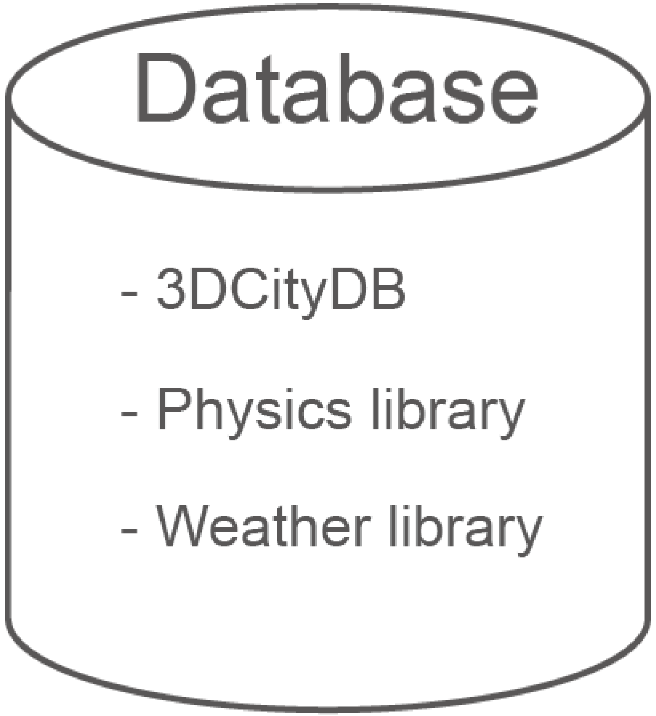
Table. weather_parameters

- reference
- country
- temporal_resolution
- attribute
- data_type
- length
- value

Category   Attribute

TU Delft

Data preparation



Content

- Introduction
- Theory background and related work
- Methodology
- Data preparation
- Python implementation
- Result analysis, reflection and future work

TU Delft

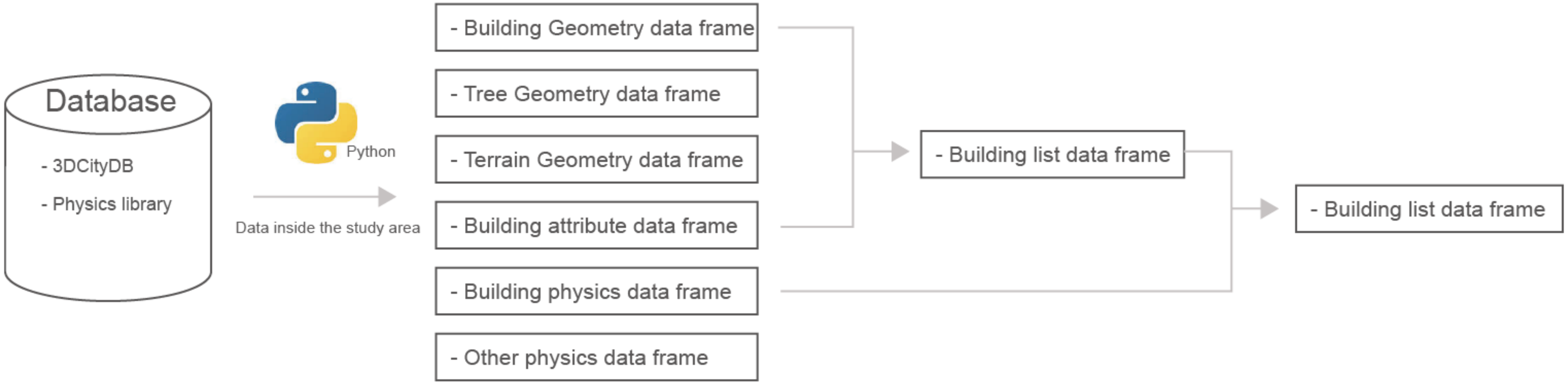
Python implementation

Connect python environment to the database

sqlalchemy create database engine for data extraction
User input: postgresql+psycopg2://postgres:123456@localhost:5432/GEO2020

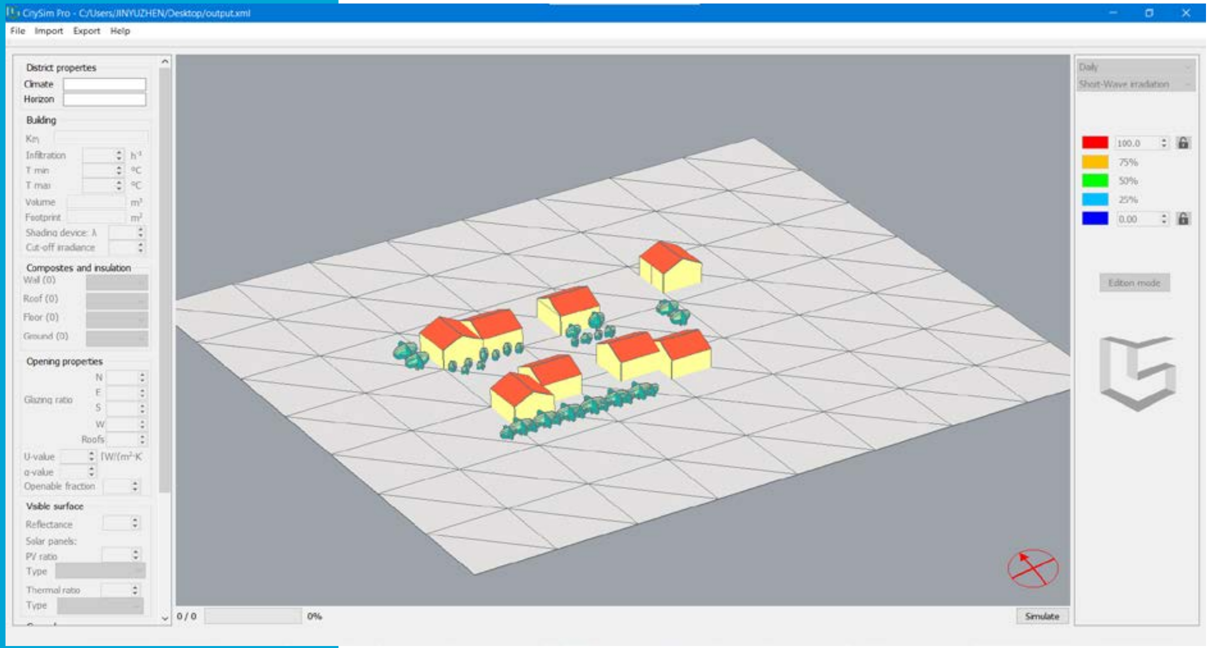
Python implementation

Data extraction and pre-processing

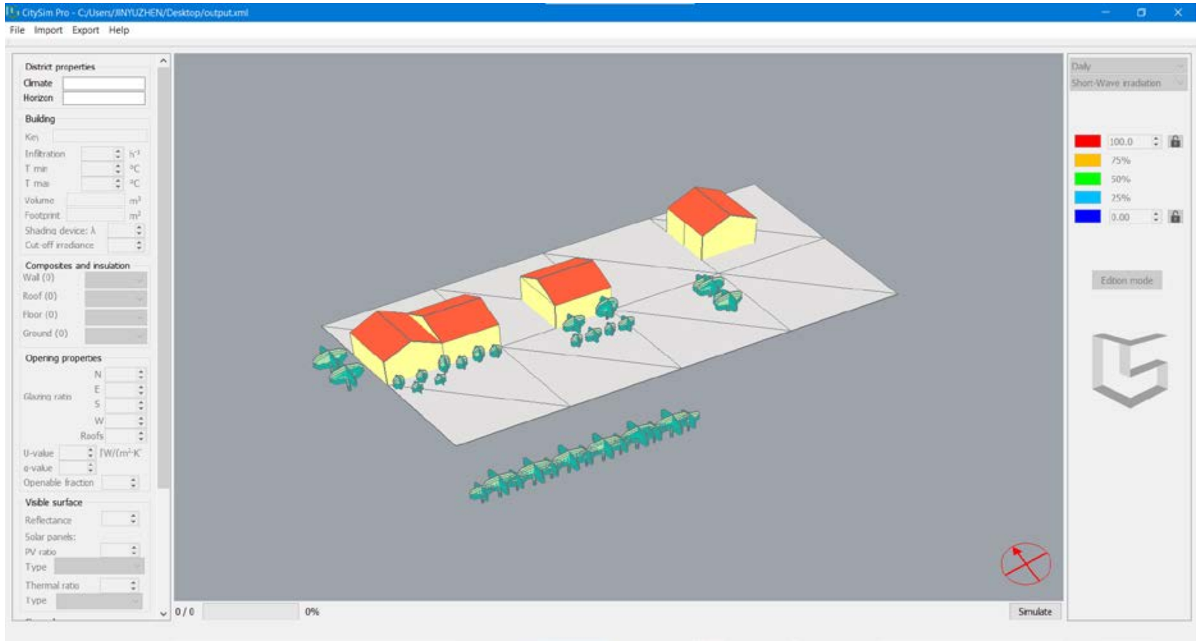


Python implementation

Data extraction and pre-processing - Study area selection



Original file, bounding box as xmin 0 ymin -30 xmax 70 ymax 15



Area selection, bounding box as xmin 0 ymin -20 xmax 70 ymax 15 buffer as 20m

TU Delft

Python implementation

Data extraction and pre-processing

Example of extraction of building geometry inside the study area:

```
geometry_envelope = gpd.read_postgis(
"SELECT thematic_surface.objectclass_id,
thematic_surface.building_id, thematic_surface.lod2_multi_surface_id,
surface_geometry.gmlid, surface_geometry.geometry, cityobject.gmlid as
parent_gmlid, surface_geometry.cityobject_id
FROM thematic_surface
LEFT JOIN surface_geometry ON
thematic_surface.lod2_multi_surface_id = surface_geometry.parent_id
LEFT JOIN cityobject ON
thematic_surface.building_id = cityobject.id
WHERE surface_geometry.geometry && st_makeenvelope(%s,%s,%s,%s,%s)
ORDER BY parent_gmlid",
%(x_min_selection, y_min_selection, x_max_selection, y_max_selection, EPSG),
db_engine, geom_col = "geometry")
```

	objectclass_id	building_id	lod2_multi_surface_id	gmlid	geometry	parent_gmlid	cityobject_id
0	34	41	281	id_building_1_polygon_4	POLYGON Z ((0.00000 10.00000 0.00000, 0.00000 ...	id_building_01	50
1	36	41	275	id_building_1_polygon_cs1	POLYGON Z ((10.00000 10.00000 10.00000, 10.000...	id_building_01	48
2	35	41	269	id_building_1_polygon_3	POLYGON Z ((0.00000 0.00000 0.00000, 0.00000 1...	id_building_01	46
3	33	41	263	id_building_1_polygon_2	POLYGON Z ((5.00000 0.00000 15.00000, 10.00000...	id_building_01	44
4	33	41	260	id_building_1_polygon_1	POLYGON Z ((0.00000 0.00000 10.00000, 5.00000 ...	id_building_01	43

The extracted data are organized and stored in python - panda dataframes

TU Delft

Python implementation

Data extraction and pre-processing

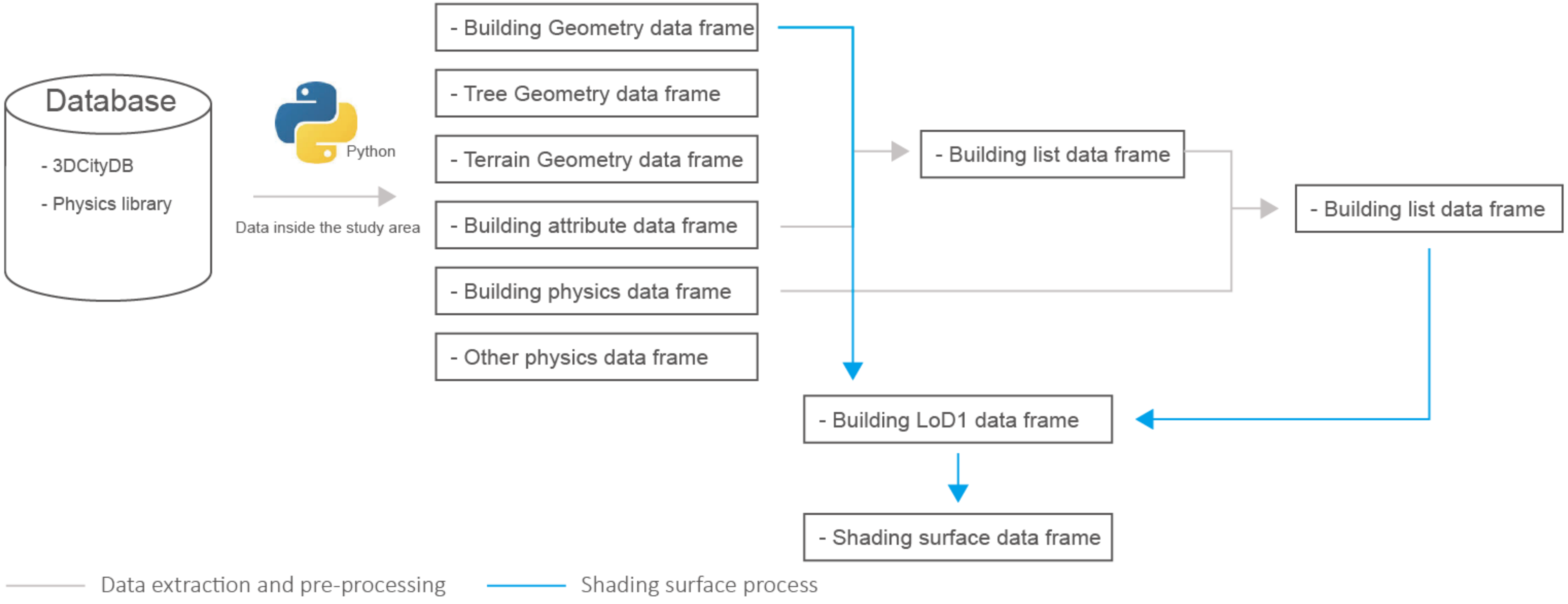
Extraction of physics parameters and merged with building geometry:

	index	building_id	parent_gmlid	num_residents	lod2_volume	function	year_of_construction	building_type	system	country	year_initial	year_end	outwalls_constructiontypeid	outwalls_windowtype
0	0	41	id_building_01	5.0	1250.0	residential building	1955.0	SFH	TABULA	NL	0	1964	204.0	
6	7	59	id_building_02	21.0	1250.0	residential building	1955.0	SFH	TABULA	NL	0	1964	204.0	
12	23	24	id_building_04	21.0	1250.0	residential building	1955.0	MFH	TABULA	NL	0	1964	204.0	
21	39	42	id_building_06	6.0	1250.0	residential building	1997.0	AB	TABULA	NL	1992	2005	201.0	
27	47	45	id_building_07	110.0	1250.0	residential building	2005.0	AB	TABULA	NL	1992	2005	201.0	
30	55	23	id_building_08	5.0	1250.0	residential building	1920.0	AB	TABULA	NL	0	1964	204.0	
36	63	92	id_building_11	12.0	1250.0	residential building	1920.0	MFH	TABULA	NL	0	1964	204.0	
42	70	83	id_building_12	34.0	1250.0	residential building	1964.0	MFH	TABULA	NL	0	1964	204.0	
49	77	101	id_buildingpart_09	20.0	1250.0	residential building	1965.0	AB	TABULA	NL	1965	1974	203.0	
54	85	110	id_buildingpart_10	25.0	1250.0	residential building	1940.0	AB	TABULA	NL	0	1964	204.0	

TU Delft

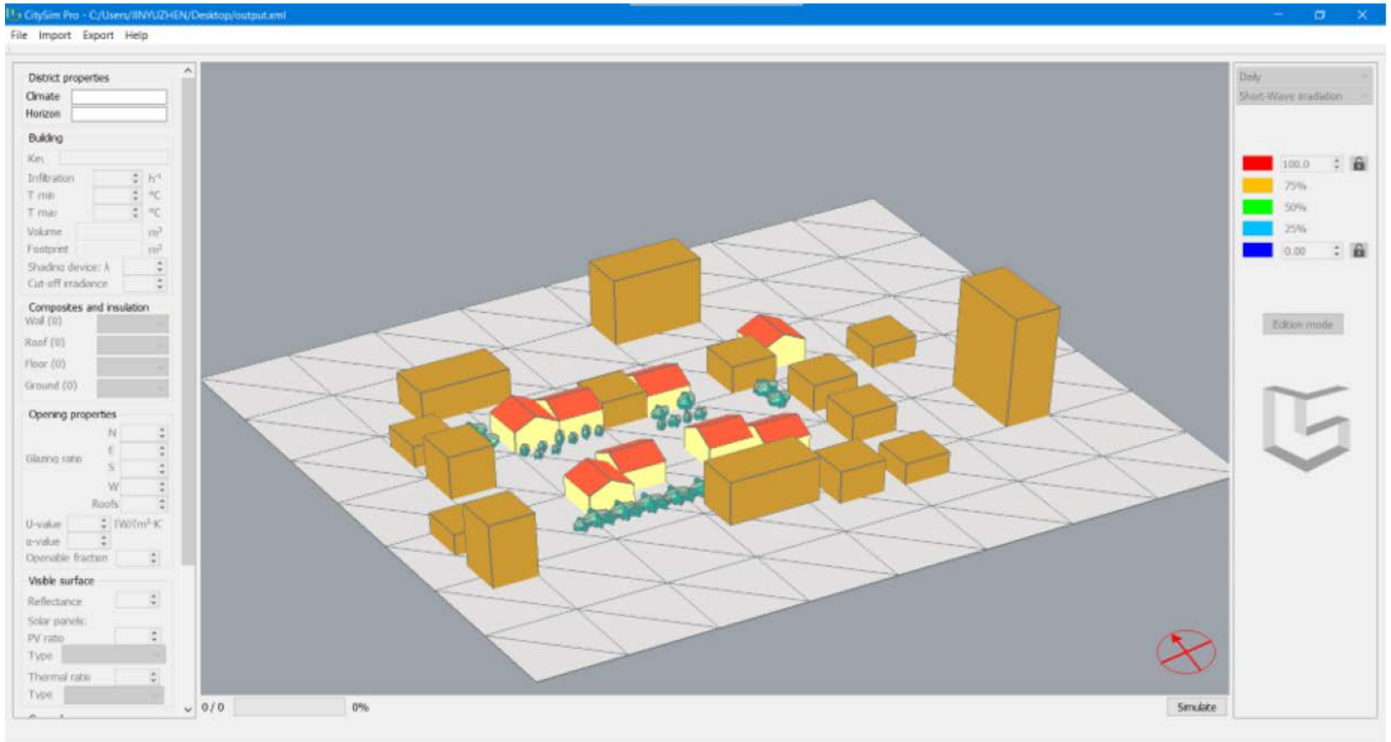
Python implementation

Shading surfaces



Python implementation

Shading surfaces



bounding box as xmin 0 ymin -30 xmax 70 ymax 15 buffer as 30m

Python implementation

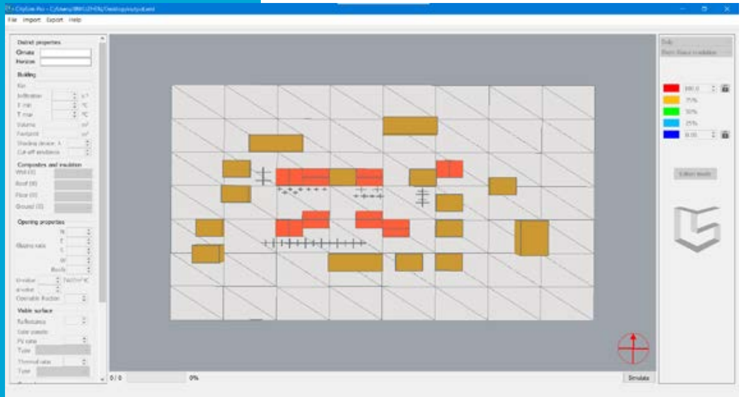
Terrain processing

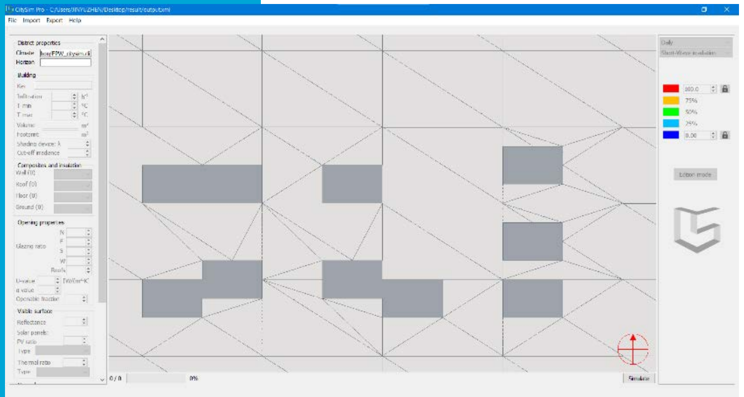
- Extract the building footprints
- Dissolve intersected building footprints into one
- Project 3D geometry to 2D
- Crop and re-triangulate the terrain
- Transform the 2D geometry to 3D

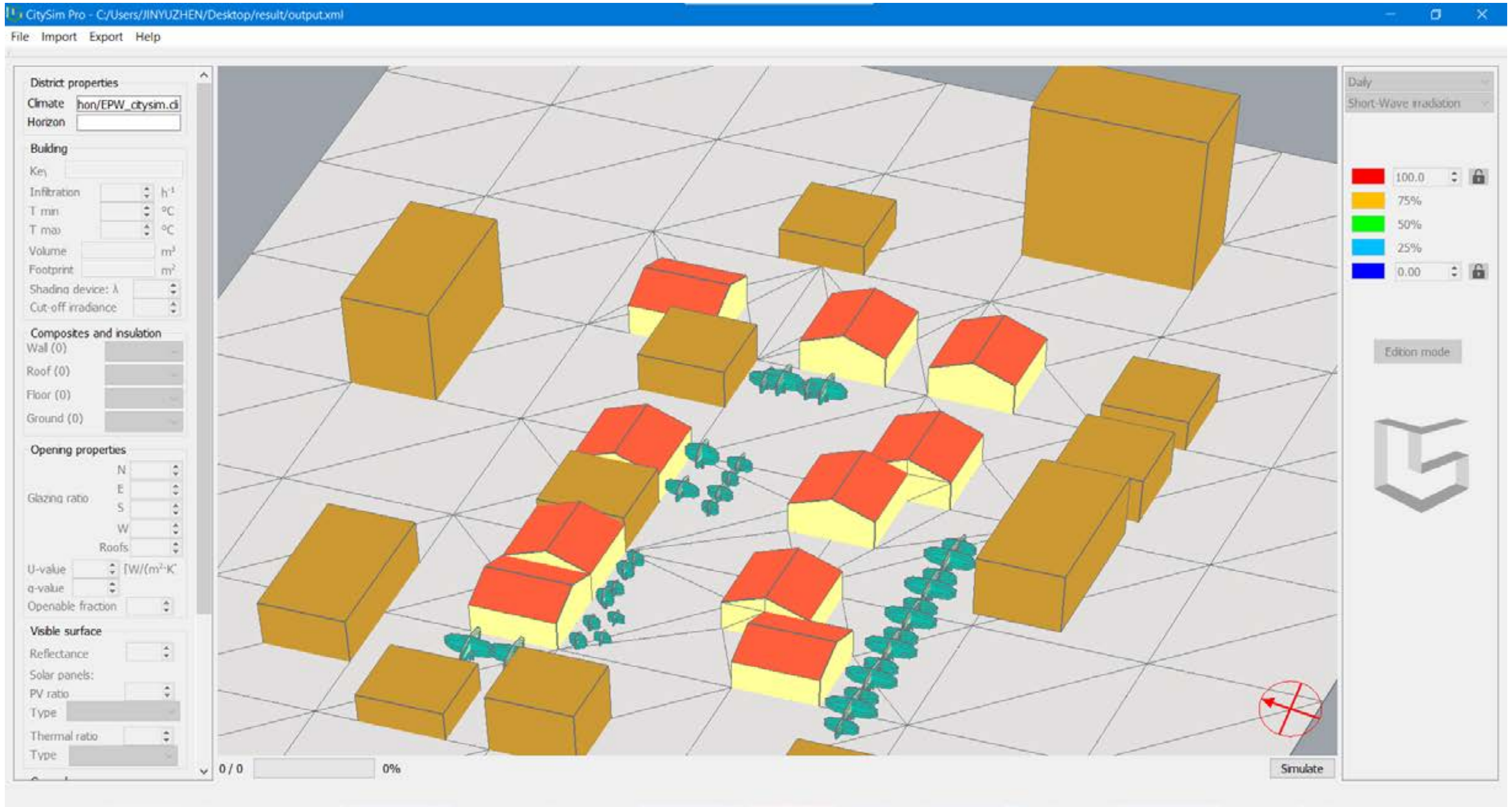
TU Delft

Python implementation

Terrain processing



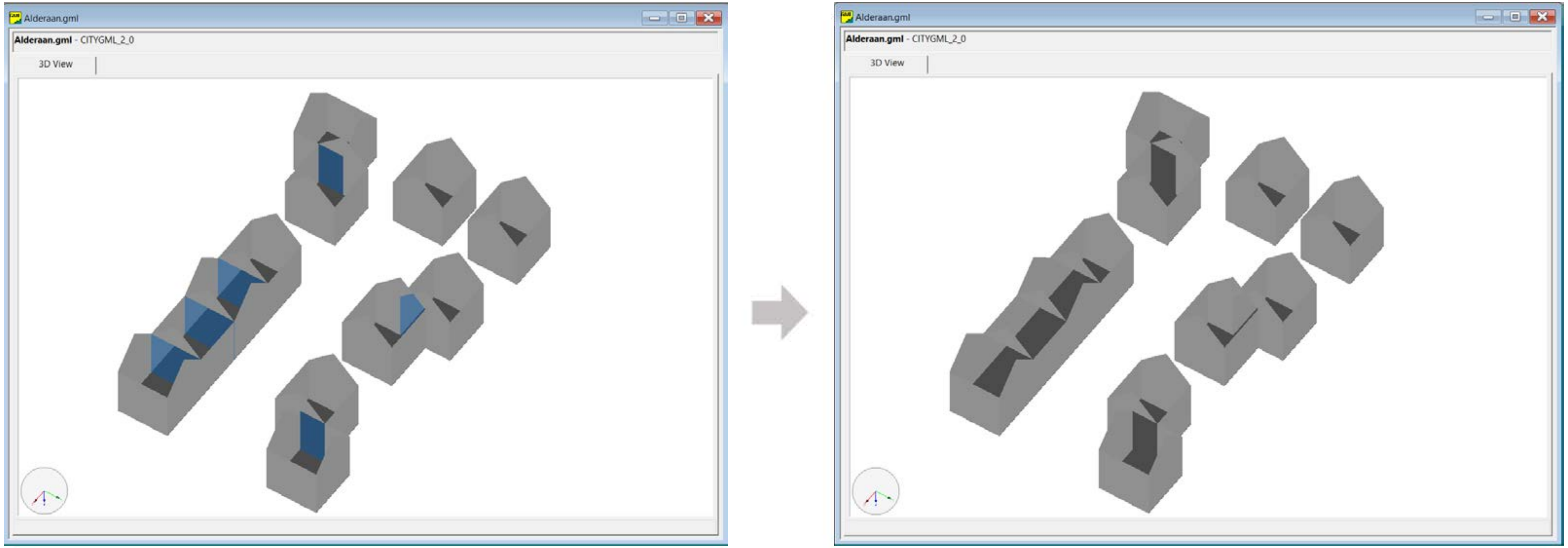




TUDelft

Python implementation

Party walls processing



RB, residnetial building PW, party wall

Filter the party walls information in table generalization, then remove those selected part walls

Python implementation

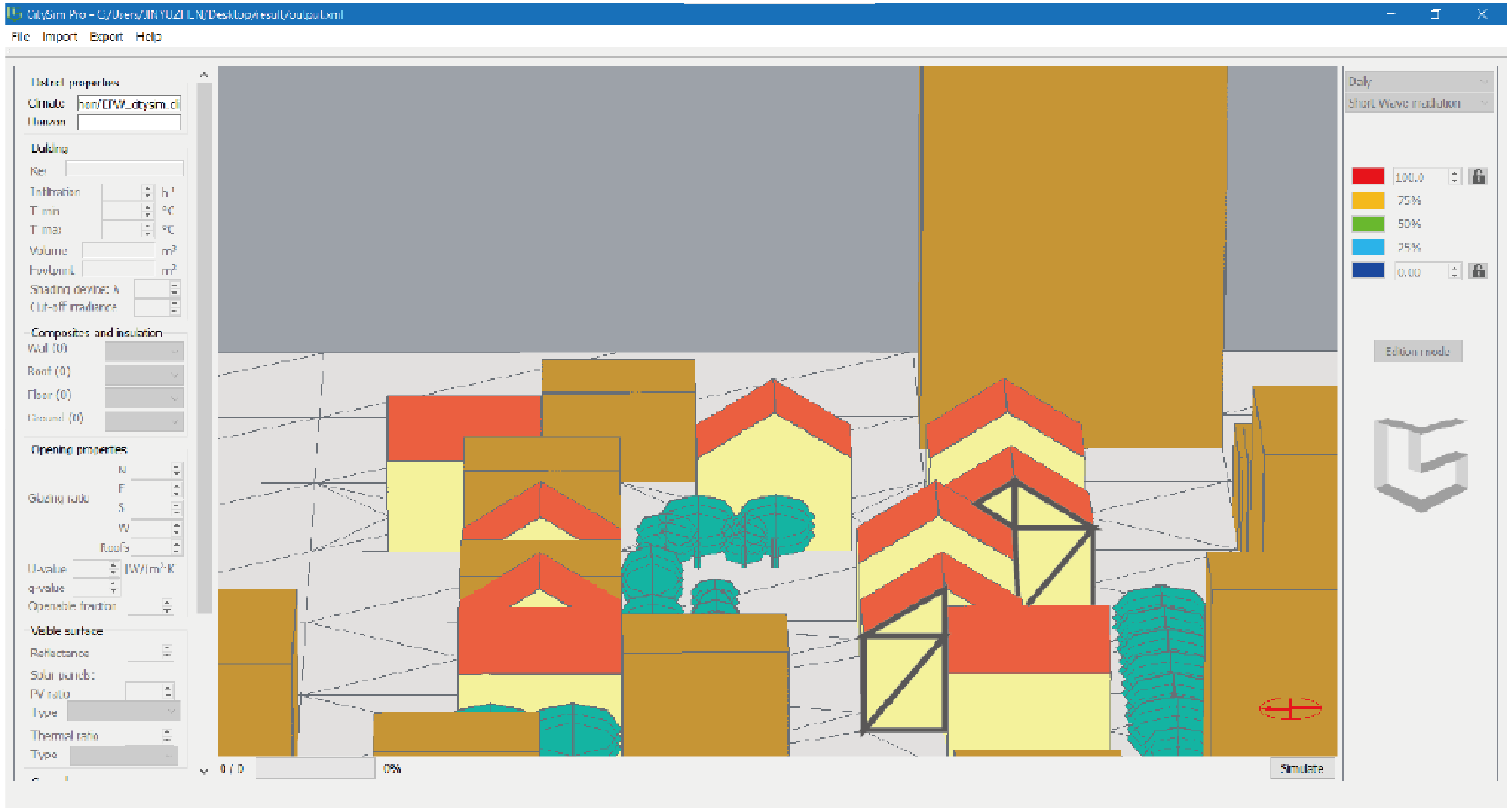
Building surfaces processing

- Check the geometry surfaces are convex or not
- Project the 3D concave surfaces to 2D
- Retriangulate the surfaces
- Transform the 2D geometry to 3D

TU Delft

Python implementation

Building surfaces processing



Re-triangulate concave building surfaces

Python implementation

Write CitySim input XML file

- Composite
- Profile
- Building
- Shading surface
- Tree
- Terrain

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<CitySim name="alderaan">
<Simulation beginMonth="1" endMonth="12" beginDay="1" endDay="31"/>
<Climate location="C:/Climate_file/alderaan.cli"/>

<Composite id="204.0" name="Wall_1,61" category="outWall">
<Layer Thickness="0.2" Conductivity="0.96" Cp="840.0" Density="2000.0"
NRE="0" GWP="0" UBP="0"/>
<Layer Thickness="0.011" Conductivity="0.045" Cp="1800.0" Density="105.0"
NRE="0" GWP="0" UBP="0"/>
<Layer Thickness="0.0" Conductivity="0.79" Cp="1014.0" Density="1329.0"
NRE="0" GWP="0" UBP="0"/>
</Composite>
```

Python implementation

Write CitySim input XML file

- Composite
- Profile
- Building
- Shading surface
- Tree
- Terrain

```
<OccupancyDayProfile id="1" name="occupancy_day_profile_of_residential_buildings"
p1="1.0" p2="1.0" p3="1.0" p4="1.0" p5="1.0"
p6="1.0" p7="0.8" p8="0.6" p9="0.4" p10="0.4"
p11="0.4" p12="0.6" p13="0.8" p14="0.6" p15="0.4"
p16="0.4" p17="0.6" p18="0.8" p19="0.8" p20="0.8"
p21="0.8" p22="1.0" p23="1.0" p24="1.0" />

<OccupancyYearProfile id="1" name="occupancy_year_profile_of_residential_buildings"
d1="1.0" d2="1.0" d3="1.0" d4="1.0" d5="1.0"
d6="1.0" d7="1.0" d8="1.0" d9="1.0" d10="1.0"
d11="1.0" d12="1.0" d13="1.0" d14="1.0" d15="1.0"
d16="1.0" d17="1.0" d18="1.0" d19="1.0" d20="1.0"
d21="1.0" d22="1.0" d23="1.0" ...... d365="1.0" />
```

TU Delft

Python implementation

Write CitySim input XML file

- Composite
- Profile
- Building
- Shading surface
- Tree
- Terrain

```
<Building id="0" key="id_building_01" Vi="1250.0" Ninf="0.1" BlindsLambda="0.2"
BlindsIrradianceCutOff="100" Simulate="true">
<HeatTank V="0.01" phi="20.0" rho="1000.0" Cp="4180.0" Tmin="20.0" Tmax="35.0"/>
<CoolTank V="0.01" phi="20.0" rho="1000.0" Cp="4180.0" Tmin="5.0" Tmax="20.0"/>
<HeatSource beginDay="258" endDay="135">
<Boiler name = "spaceX" Pmax="10000000.0" eta_th="0.95"/>
</HeatSource>
<CoolSource beginDay="136" endDay="257">
<HeatPump Pmax="10000000.0" eta_tech="0.3" Ttarget="5.0" Tsource="air"/>
</CoolSource>
<Zone id="0" volume="1000.0" psi="0" Tmin="20" Tmax="26" groundFloor="true"
nightVentilationBegin="0" nightVentilationEnd="0">
<Occupants n="5.0" sensibleHeat="90" sensibleHeatRadiantFraction="0.6"
latentHeat="0" type="1"/>
<Wall id="0" key="id_building_1_polygon_4" type="204.0" ShortWaveReflectance="0.3"

 GlazingRatio="0.17" GlazingGValue="0.6" GlazingUValue="3.7" OpenableRatio="0">
 <V0 x="-40.0" y="40.0" z="0.0"/>
 <V1 x="-40.0" y="40.0" z="10.0"/>
 <V2 x="-35.0" y="40.0" z="15.0"/>
 <V3 x="-30.0" y="40.0" z="10.0"/>
 <V4 x="-30.0" y="40.0" z="0.0"/>
 </Wall>
 ......
 <Roof id="3" key="id_building_1_polygon_2" type="166.0" ShortWaveReflectance="0.2"
 GlazingRatio="0.0" GlazingGValue="0.6" GlazingUValue="3.7" OpenableRatio="0" kFactor="0">
 <V0 x="-35.0" y="30.0" z="15.0"/>
 <V1 x="-30.0" y="30.0" z="10.0"/>
 <V2 x="-30.0" y="40.0" z="10.0"/>
 <V3 x="-35.0" y="40.0" z="15.0"/>
 </Roof>
 ......
 <Floor id="2" key="id_building_1_polygon_3" type="132.0" ShortWaveReflectance="0.0"
 GlazingRatio="0.0" GlazingGValue="0" GlazingUValue="0" OpenableRatio="0">
 <V0 x="-40.0" y="30.0" z="0.0"/>
 <V1 x="-40.0" y="40.0" z="0.0"/>
 <V2 x="-30.0" y="40.0" z="0.0"/>
 <V3 x="-30.0" y="30.0" z="0.0"/>
 </Floor>
 ......
 </Zone>
```

Including the option to retain decimal places and coordinate translation

TU Delft

Python implementation

Write CitySim input XML file

- Composite
- Profile
- Building
- Shading surface
- Tree
- Terrain

```
<ShadingSurface>
<Surface id="0" ShortWaveReflectance="0.2">
<V0 x="-20.0" y="-20.0" z="5.0"/>
<V1 x="0.0" y="-20.0" z="5.0"/>
<V2 x="0.0" y="-10.0" z="5.0"/>
<V3 x="-20.0" y="-10.0" z="5.0"/>
</Surface>
......
</ShadingSurface>
```

Including the option to retain decimal places and coordinate translation

Python implementation

Write CitySim input XML file

- Composite
- Profile
- Building
- Shading surface
- Tree
- Terrain


```xml
<Trees>
<Tree id="240" name="Maple" leafAreaIndex="3.0" leafWidth="0.1" leafDistance="1.0"
deciduous="false">
<Leaf id="240" ShortWaveReflectance="0.3" LongWaveEmissivity="0.95">
<V0 x="14.791" y="20.0" z="3.283"/>
<V1 x="14.791" y="20.0" z="0.0"/>
<V2 x="15.209" y="20.0" z="0.0"/>
<V3 x="15.209" y="20.0" z="3.283"/>
<V4 x="15.0" y="20.0" z="3.2"/>
</Leaf>
<Leaf id="240" ShortWaveReflectance="0.3" LongWaveEmissivity="0.95">
<V0 x="15.0" y="19.791" z="3.283"/>
<V1 x="15.0" y="19.791" z="0.0"/>
<V2 x="15.0" y="20.209" z="0.0"/>
<V3 x="15.0" y="20.209" z="3.283"/>
<V4 x="15.0" y="20.0" z="3.2"/>
</Leaf>
<Leaf id="240" ShortWaveReflectance="0.3" LongWaveEmissivity="0.95">
<V0 x="15.0" y="19.791" z="3.283"/>
<V1 x="15.0" y="20.0" z="3.2"/>
<V2 x="15.0" y="20.209" z="3.283"/>
<V3 x="15.0" y="21.771" z="3.903"/>
<V4 x="15.0" y="22.505" z="5.6"/>
<V5 x="15.0" y="21.771" z="7.297"/>
<V6 x="15.0" y="20.0" z="8.0"/>
<V7 x="15.0" y="18.229" z="7.297"/>
<V8 x="15.0" y="17.495" z="5.6"/>
<V9 x="15.0" y="18.229" z="3.903"/>
</Leaf>
<Leaf id="240" ShortWaveReflectance="0.3" LongWaveEmissivity="0.95">
<V0 x="14.791" y="20.0" z="3.283"/>
<V1 x="15.0" y="20.0" z="3.2"/>
<V2 x="15.209" y="20.0" z="3.283"/>
<V3 x="16.771" y="20.0" z="3.903"/>
<V4 x="17.505" y="20.0" z="5.6"/>
<V5 x="16.771" y="20.0" z="7.297"/>
<V6 x="15.0" y="20.0" z="8.0"/>
<V7 x="13.229" y="20.0" z="7.297"/>
<V8 x="12.495" y="20.0" z="5.6"/>
<V9 x="13.229" y="20.0" z="3.903"/>
</Leaf>
</Tree>
......
</Trees>
```


Including the option to retain decimal places and coordinate translation

Python implementation

Write CitySim input XML file

- Composite
- Profile
- Building
- Shading surface
- Tree
- Terrain

```
<GroundSurface>
<Ground id="677" key="ID_d5a47bcc-398d-4d4a-8b28-a5d1bbbbadf1"
ShortWaveReflectance="0.3" type="999" kFactor="0.7" detailedSimulation="true">
<V0 x="20.0" y="30.0" z="0.0"/>
<V1 x="20.0" y="50.0" z="0.0"/>
<V2 x="0.0" y="50.0" z="0.0"/>
</Ground>
......
</GroundSurface>
```

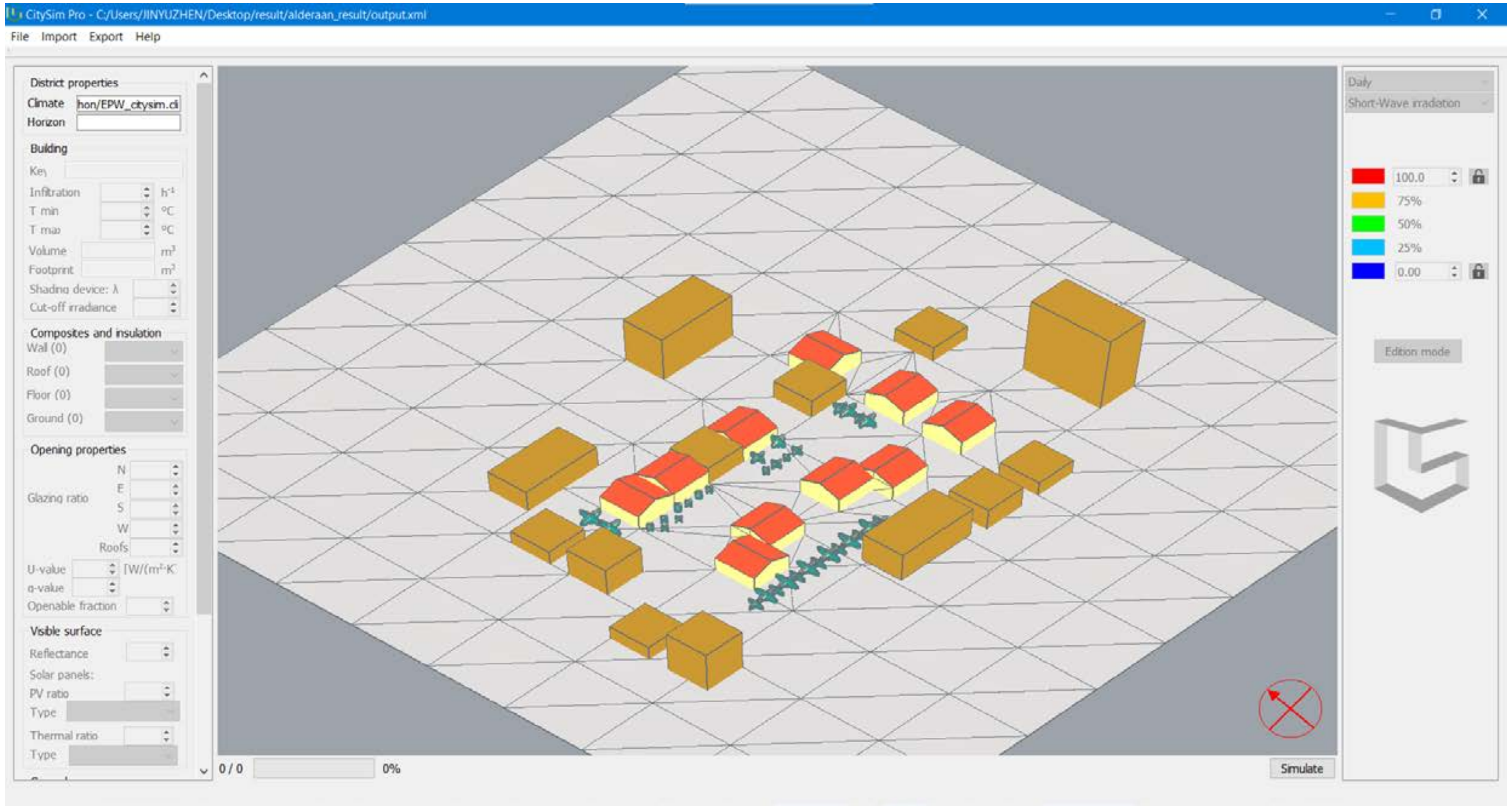
Including the option to retain decimal places and coordinate translation

Python implementation

Including the option to retain decimal places and coordinate translation

Python implementation

Load the input files in CitySim Pro GUI to run the simulation



The input files can be imported into the CitySim pro GUI to run the simulation

Python implementation

Call CitySim to run the simulation

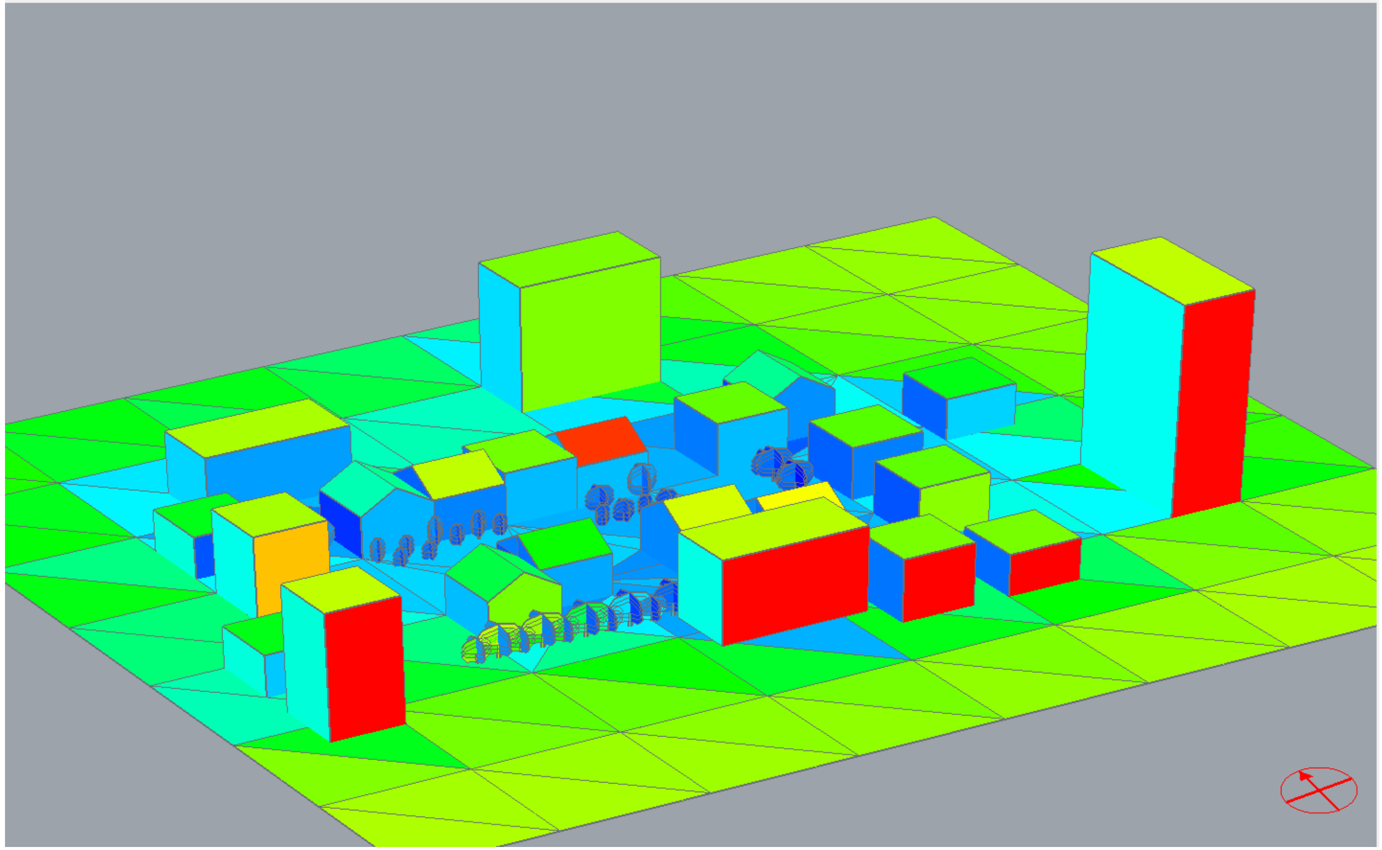
```
[168]: # Call CitySim ############################################################
print("Waking up CitySim...\n")
citysim_input = input ("What is the file path of CitySim solver?\n\n") #'C:/Users/JINYUZHEN/Desktop/citysim/CitySim.exe'
xml_input = input ("What is the file path of CitySim XML input file?\n\n") #'C:/Users/JINYUZHEN/Desktop/result/output.xml'

try:
    p = subprocess.Popen([citysim_input, xml_input], stderr=subprocess.PIPE)
    print("The simulation has begun!\n")
    # Print what CitySim solver is showing
    for line in p.stderr:
        print(line.decode())
        if line.decode() == 'Simulation ended.\r\n':
            print("Results are being written...")

except Exception:
    print("Couldn't wake up CitySim. Please, check if CitySim solver is in the script's directory")
    time.sleep(5)
    sys.exit()
```

```
Waking up CitySim...

What is the file path of CitySim solver?

 C:/Users/JINYUZHEN/Desktop/citysim/CitySim.exe
What is the file path of CitySim XML input file?

 C:/Users/JINYUZHEN/Desktop/result/output.xml
The simulation has begun!

XML description file: C:/Users/JINYUZHEN/Desktop/result/output.xml

Reading XML file...
```

TU Delft

Python implementation

Result files containing hourly resolution results are generated



Short-Wave irradiation result viewed in CitySim Pro

Simulation result files

- _TH.tsv
- _SW.tsv
- _TS.tsv
- _VF.tsv
- _LW.tsv
- _Ared.tsv
- ……

Python implementation

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	
1	#timeStep	0(id_buildi	0(id_buildi	0(id_buildi	0(id_buildi	0(id_buildi	0(id_buildi	0(id_buildi	0(id_buildi	0(id_buildi	0(id_buildi	0(id_buildi	0(id_buildi	0(id_buildi	6(id_buildi	6(id_buildi	6(id_buildi	6(id_buildi	6(id_buildi	6(id_buildi	6(id_buildi	6(id_buildi	6(id_buildi	6(i
2	1	20	11718	0	450	11718	0	35	18	12058	45.694	0	0	0	20	10614	0	1890	10614	0	35	18	10954	
3	2	20	11903	0	450	11903	0	35	18	12243	46.393	0	0	0	20	10819	0	1890	10819	0	35	18	11159	
4	3	20	11969	0	450	11969	0	35	18	12309	46.643	0	0	0	20	10903	0	1890	10903	0	35	18	11243	
5	4	20	11693	0	450	11693	0	35	18	12033	45.599	0	0	0	20	10631	0	1890	10631	0	35	18	10971	
6	5	20	11782	0	450	11782	0	35	18	12122	45.938	0	0	0	20	10720	0	1890	10720	0	35	18	11060	
7	6	20	12192	0	450	12192	0	35	18	12532	47.488	0	0	0	20	11125	0	1890	11125	0	35	18	11465	
8	7	20	12673	0	360	12673	0	35	18	13013	49.312	0	0	0	20	11818	0	1512	11818	0	35	18	12158	
9	8	20	13011	0	270	13011	0	35	18	13351	50.593	0	0	0	20	12378	0	1134	12378	0	35	18	12718	
10	9	20	13550	0	180	13550	0	35	18	13890	52.635	0	0	0	20	13143	0	756	13143	0	35	18	13483	
11	10	20	13338	0	552	13338	0	35	18	13678	51.831	0	0	0	20	12926	0	1169	12926	0	35	18	13266	
12	11	20	12728	0	1040	12728	0	35	18	13068	49.521	0	0	0	20	12272	0	1734	12272	0	35	18	12612	
13	12	20	12551	0	1193	12551	0	35	18	12891	48.85	0	0	0	20	11916	0	2126	11916	0	35	18	12256	
14	13	20	12563	0	982	12563	0	35	18	12903	48.896	0	0	0	20	11714	0	2206	11714	0	35	18	12054	
15	14	20	12587	0	841	12587	0	35	18	12927	48.986	0	0	0	20	11930	0	1767	11930	0	35	18	12270	
16	15	20	12544	0	572	12544	0	35	18	12884	48.824	0	0	0	20	12096	0	1193	12096	0	35	18	12436	
17	16	20	12680	0	633	12680	0	35	18	13020	49.341	0	0	0	20	12276	0	1155	12276	0	35	18	12616	
18	17	20	13718	0	270	13718	0	35	18	14058	53.273	0	0	0	20	13093	0	1134	13093	0	35	18	13433	
19	18	20	13155	0	360	13155	0	35	18	13495	51.139	0	0	0	20	12325	0	1512	12325	0	35	18	12665	
20	19	20	13402	0	360	13402	0	35	18	13742	52.073	0	0	0	20	12578	0	1512	12578	0	35	18	12918	
21	20	20	13578	0	360	13578	0	35	18	13918	52.741	0	0	0	20	12760	0	1512	12760	0	35	18	13100	
22	21	20	13638	0	360	13638	0	35	18	13978	52.971	0	0	0	20	12825	0	1512	12825	0	35	18	13165	
23	22	20	13681	0	450	13681	0	35	18	14021	53.133	0	0	0	20	12668	0	1890	12668	0	35	18	13008	
24	23	20	13981	0	450	13981	0	35	18	14321	54.269	0	0	0	20	12977	0	1890	12977	0	35	18	13317	
25	24	20	14095	0	450	14095	0	35	18	14435	54.703	0	0	0	20	13094	0	1890	13094	0	35	18	13434	
26	25	20	14170	0	450	14170	0	35	18	14510	54.987	0	0	0	20	13169	0	1890	13169	0	35	18	13509	
27	26	20	14256	0	450	14256	0	35	18	14596	55.311	0	0	0	20	13255	0	1890	13255	0	35	18	13595	

output_TH

CitySim simulation result TH.tsv in Excel window

TU Delft

Python implementation

Result storage – focus on _TH.tsv file for example

- Total heating and cooling demand (Qs) for each building
- 8760 hourly value

```
# Read the results - Qs value from TH file
out_df = pd.read_csv('C:/Users/JINYUZHEN/Desktop/result/output_TH.out',  sep='\t') # change file name
qs_cols = [col for col in out_df.columns if 'Qs' in col]
qs_df = pd.read_csv("C:/Users/JINYUZHEN/Desktop/result/output_TH.out",  sep='\t', usecols=qs_cols)
qs_df.head()
```

	0(id_building_01):0:Qs(Wh)	6(id_building_02):6:Qs(Wh)	12(id_building_04):12:Qs(Wh)	21(id_building_06):21:Qs(Wh)	27(id_building_07):27:Qs(Wh)	30(id_building_08):30:Qs(Wh)	39
0	11718	10609	13691	7130	0	13687	
1	11902	10814	13913	7173	0	13874	
2	11968	10898	13943	7066	0	13875	
3	11692	10626	13459	6604	0	13376	
4	11782	10714	13584	6694	0	13502	

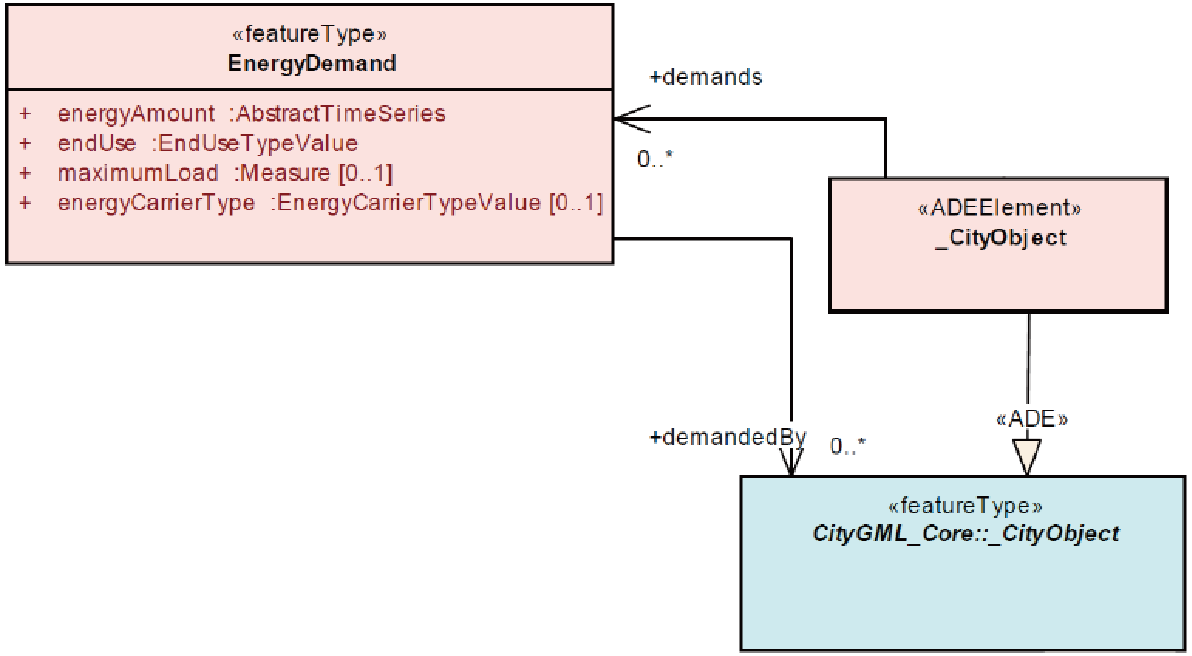
Extract Qs value from TH.tsv file into dataframe

TU Delft

Python implementation

Result storage – focus on _TH.tsv file for example

- Store in 3DCityDB extension Energy ADE (v. 2.0) structure



- Extract the hourly Qs parameters
- Calculate the monthly and yearly value
- Organize the data into dataframes same as the Energy ADE structure
- Insert the dataframes into the database

* Feature EnergyDemand used for storing the Qs value (From the UML diagram of the Energy ADE core)

Python implementation

Result storage – focus on _TH.tsv file for example

The simulation result Qs parameters stored in the database

	id [PK] bigint	timeinterval numeric	timeinterval_factor integer	timeinterval_radix integer	timeinterval_unit character varying (1000)	timeperiodprop_beginposition timestamp with time zone	timeperiodproper_endposition timestamp with time zone	values_ text
1	275	1	[null]	[null]	hour	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	11716 11900 11966 11691 11780 12189 12670 1
2	277	1	[null]	[null]	month	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	10878.657 8812.929 8924.511 4681.693 1639.64
3	279	1	[null]	[null]	year	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	61056.352
4	281	1	[null]	[null]	hour	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	10614 10819 10903 10631 10720 11124 11817 1
5	283	1	[null]	[null]	month	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	10327.964 8231.185 8260.245 4126.751 1369.68
6	285	1	[null]	[null]	year	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	56093.055
7	287	1	[null]	[null]	hour	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	13724 13946 13979 13496 13622 14250 15123 1
8	289	1	[null]	[null]	month	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	12867.419 10303.628 10352.846 5296.734 1799.
9	291	1	[null]	[null]	year	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	70396.426
10	293	1	[null]	[null]	hour	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	7145 7189 7083 6622 6712 7216 7725 7984 850

* Table ng_regulartimeseries in database contains the Alderaan simulation result

TU Delft

For other results they can also be stored in a way similar as Qs value

Content

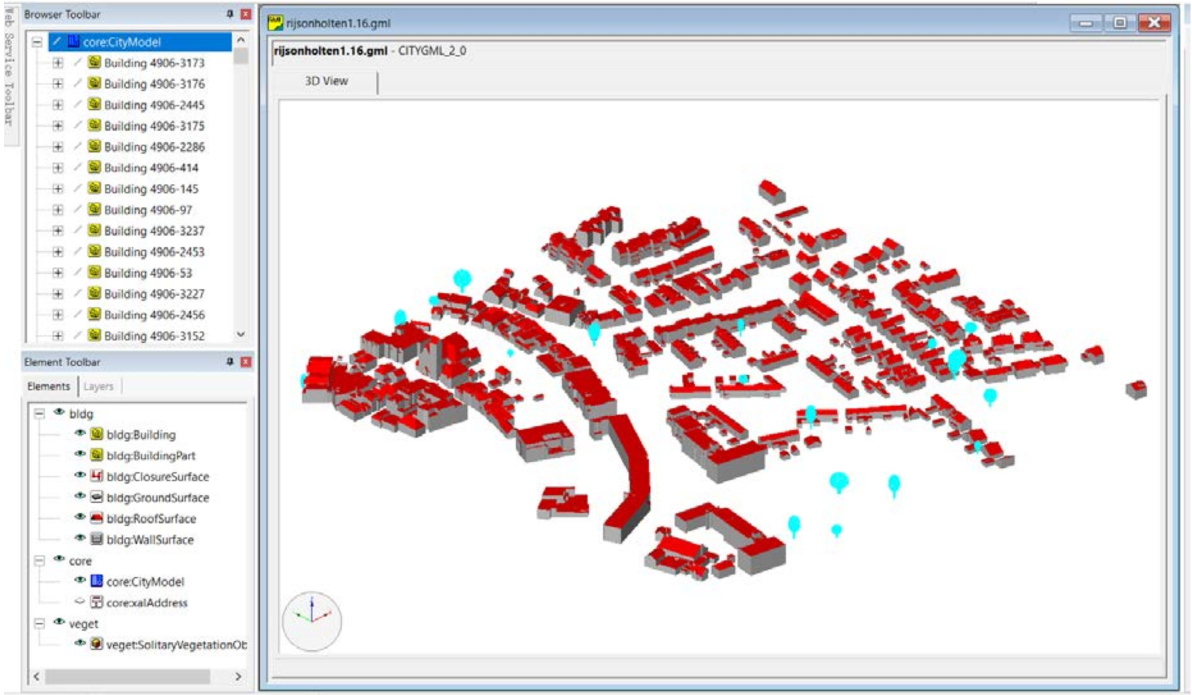
- Introduction
- Theory background and related work
- Methodology
- Data preparation
- Python implementation
- Result analysis, reflection and future work

TU Delft

Result analysis, reflection and future work

Python interface testing

- Real dataset RijssenHolten.gml from 3D BAG 2.0

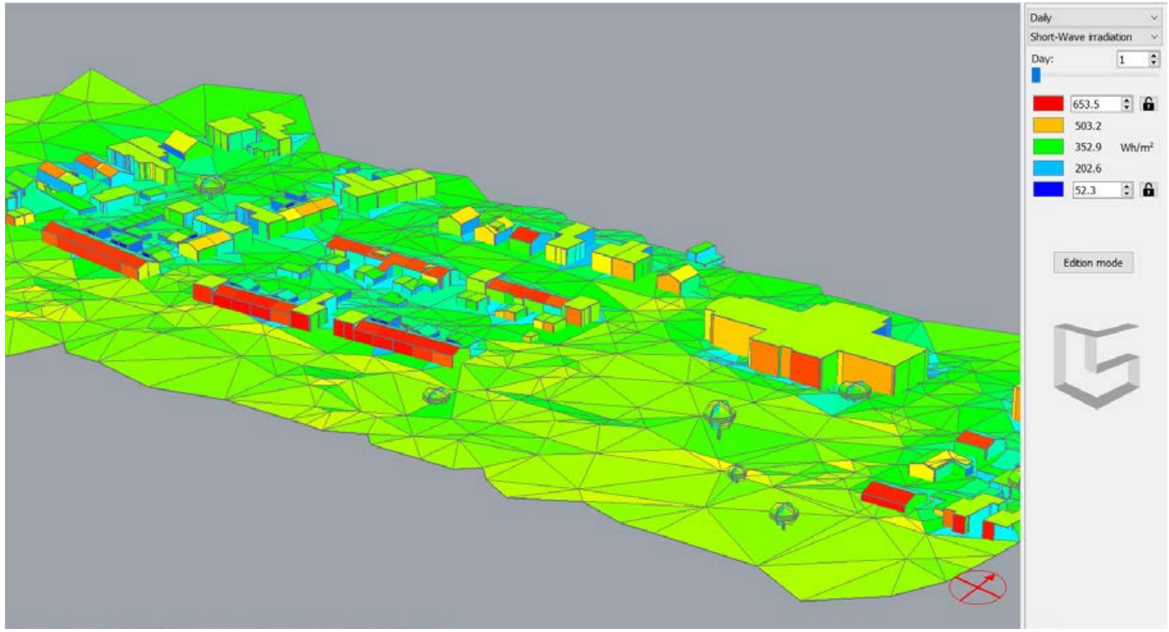


RijssenHolten.gml (1/16 of the original)

Result analysis, reflection and future work

Python interface testing

- Real dataset RijssenHolten.gml from 3D BAG 2.0



bounding box as xmin 231952 ymin 479844 xmax 232266 ymax 479944 buffer as 50m

Result analysis, reflection and future work

Python interface testing

- Real dataset RijssenHolten.gml from 3D BAG 2.0

	id [PK] bigint	timeinterval numeric	timeinterval_factor integer	timeinterval_radix integer	timeinterval_unit character varying (1000)	timeperiodprop_beginposition timestamp with time zone	timeperiodproper_endposition timestamp with time zone	values_ text
1	142833	1	[null]	[null]	hour	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	5814 5889 5867 5602 5662 5982 6334 6544 69
2	142835	1	[null]	[null]	month	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	5298.71 4289.198 4353.088 2347.889 817.633
3	142837	1	[null]	[null]	year	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	29514.317
4	142839	1	[null]	[null]	hour	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	5783 5857 5833 5570 5631 5948 6298 6507 68
5	142841	1	[null]	[null]	month	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	5276.545 4272.766 4335.081 2335.581 817.05
6	142843	1	[null]	[null]	year	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	29396.608
7	142845	1	[null]	[null]	hour	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	5821 5894 5869 5602 5663 5984 6336 6546 69
8	142847	1	[null]	[null]	month	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	5305.105 4296.968 4359.176 2352.559 824.42
9	142849	1	[null]	[null]	year	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	29575.321
10	142851	1	[null]	[null]	hour	2022-01-01 00:00:00+08	2022-12-31 11:59:59+08	5744 5819 5798 5536 5597 5912 6261 6472 68

* Table ng_regulartimeseries in database contains the rijssenholten simulation result

TU Delft

Result analysis, reflection and future work

Guidelines for the python interface

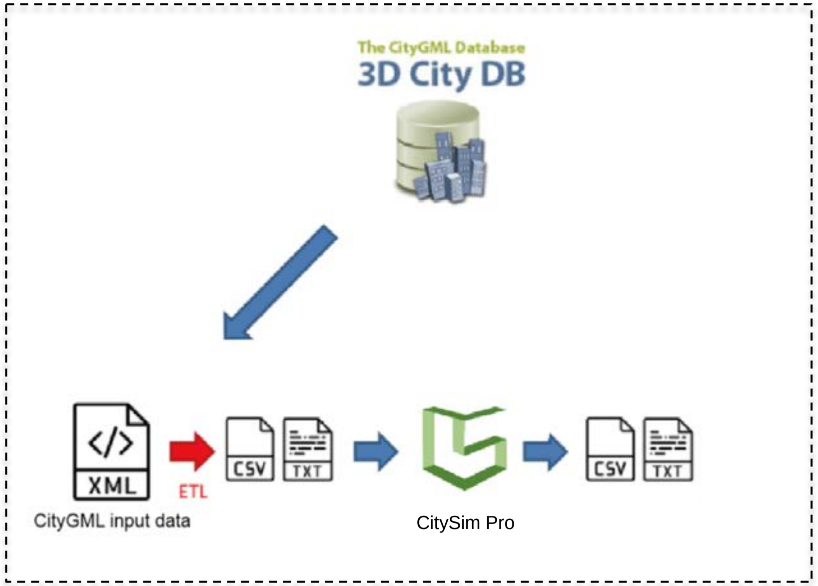
The script of the python interface along with physics library and weather library backup files are posted on GitHub:

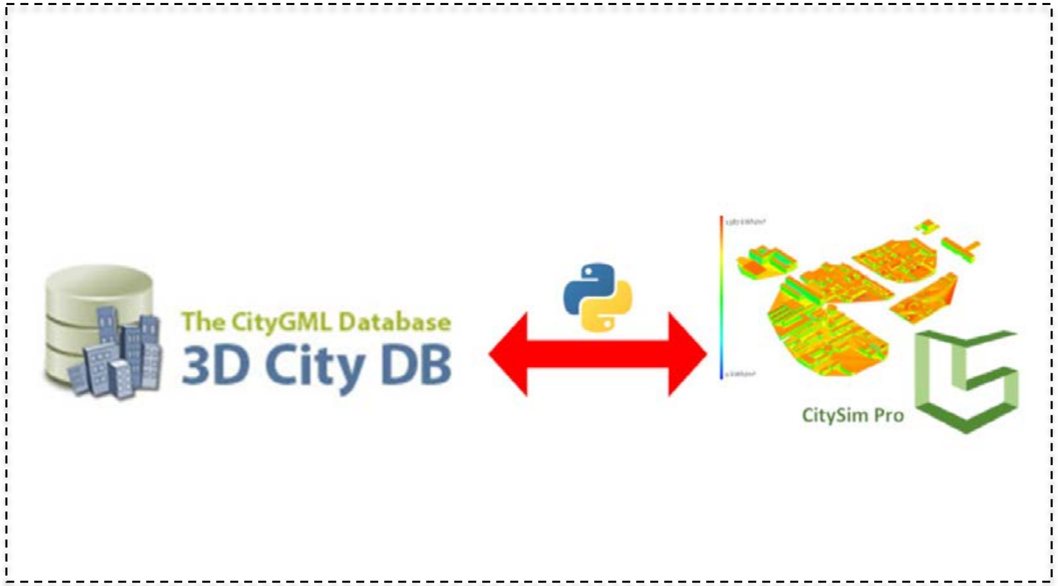
https://github.com/yuzhenjin3000/Dynamic-energy-simulations-based-on-the-3D-BAG-2.0.git

TU Delft

Result analysis, reflection and future work

Conclusion





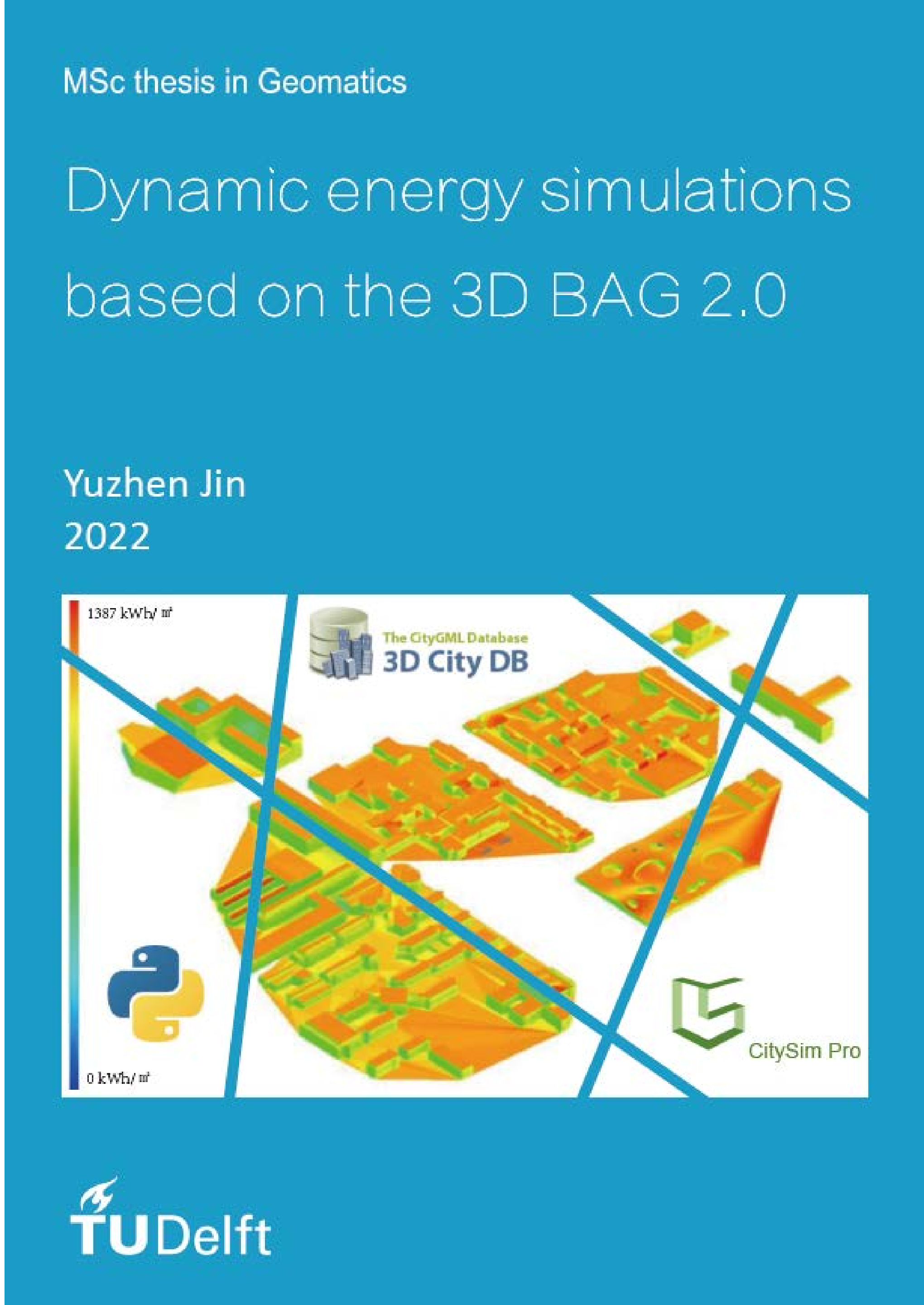
Result analysis, reflection and future work

Limitation and Future work

- Improve the interface's adaptability to real dataset
- Simplify the processing with more powerful python library
- Graphical user interface can be designed

TUDelft

Thanks!


MSc thesis in Geomatics
Dynamic energy simulations based on the 3D BAG 2.0
Yuzhen Jin
2022
1387 kWh/ m²
0 kWh/ m²
The CityGML Database
3D City DB
CitySim Pro
TUDelft