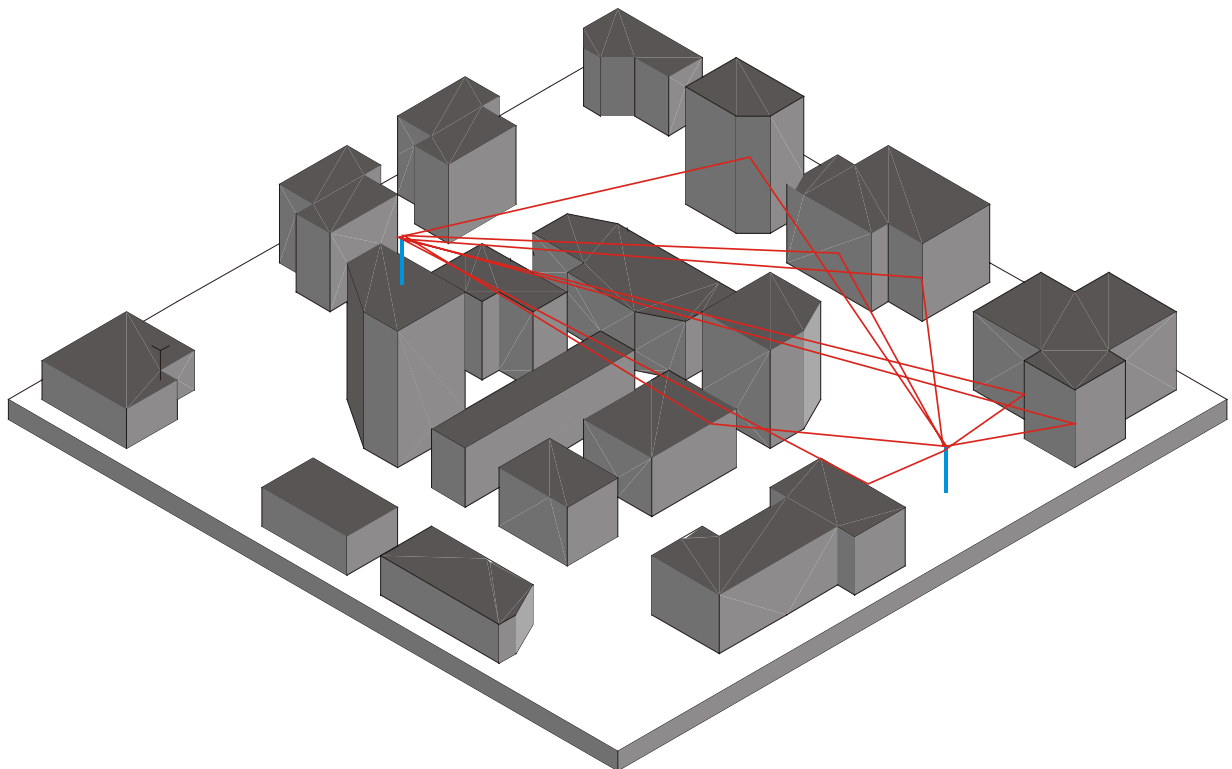




# ***WinProp***

## **Urban Database Requirements**

### **Database File Format Description**



# 1. Required Data

## 1.1. Definition of buildings

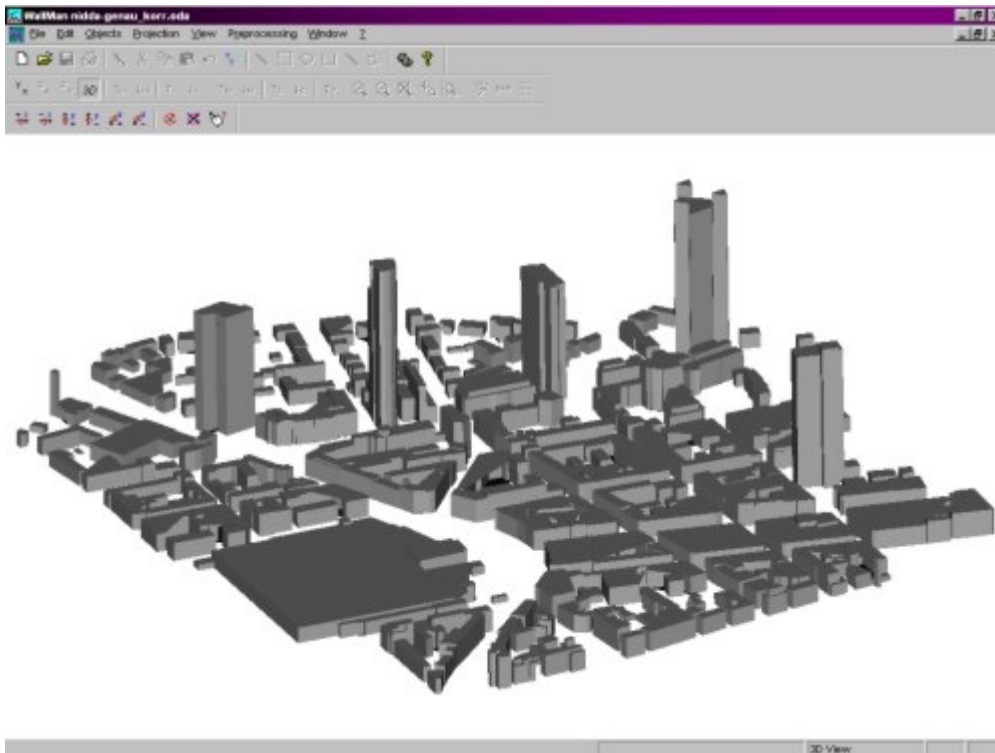


Figure 1: Urban database example (Frankfurt, Germany)

The buildings are described by polygonal cylinders, i.e. buildings with arbitrary forms can be used. The building database offers the following features:

- Each polygon can have an arbitrary number of corners.
- At least 3 corners are required to define a valid polygon.
- Each building has a uniform height (polygonal cylinder).
- Flat rooftops are used (horizontal planes)
- Only vertical walls (parallel to z-axis) are allowed
- Each building has a single set of material properties which are used for the whole building
- The polygon cannot intersect itself
- The polygon can intersect other polygons.

If angular roofs must be modeled, the indoor data base format and prediction tool must be used.

Buildings totally inside other buildings are removed because they do not influence the database at all. If the inner buildings are taller than the surrounding building it will be kept (and considered as tower).

Up to now the databases must be in UTM coordinates (meter) and they are therefore defined in an orthogonal 2D coordinate system (x is longitude, y is latitude).

Topography is not included in the vector database of the buildings. But an arbitrary topographical database can be considered additionally (more information can be found in section 1.4).

## 1.2. Types of Buildings

Four different types of buildings are possible. All buildings have the properties defined in section 1.1 (polygonal cylinders). The building types are:

- **Standard buildings**

Standard buildings can be used to model a building in an urban environment.

- **Courtyards & Towers**

This type of building is used to define a building which is totally inside a second building. If the inner building is higher this is called a tower and if it is lower it is called a court yard. Courtyards have the same material properties as standard buildings. But there is one big difference: While a standard building has no information about the situation above the building, the courtyard defines also the environment above the building. No standard building, vegetation building or virtual building is possible above a courtyard. Only courtyards above courtyards are possible. This is achieved with the priority list as given in section 1.3 of this document.

- Towers

For towers it is very simple, because always the highest building will be considered. If a tall building is inside another building, the tall building will be considered for all pixels inside the tall building. So single towers inside buildings can be modeled also with standard buildings.

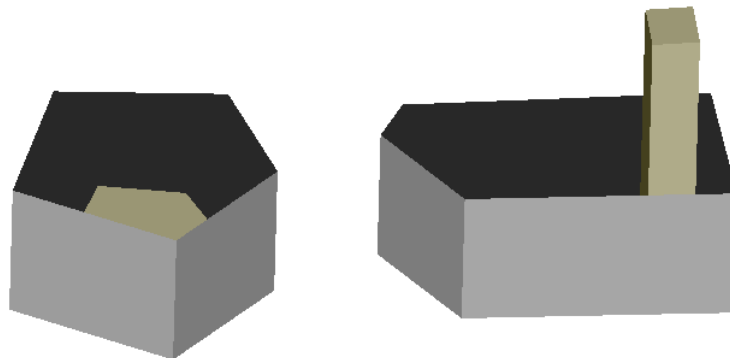


Figure 2: Towers

- Courtyards

If a court yard should to be modeled, it could be defined as an included building with the height zero (see Figure 4). This should be avoided, if further buildings or towers are placed inside the court yard because it could lead to undefined situations.

It is better to model such a court yard by using one polygonal line like shown in figure 3:

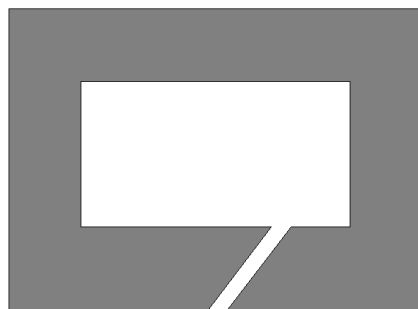


Figure 3: Courtyard (Version 1)

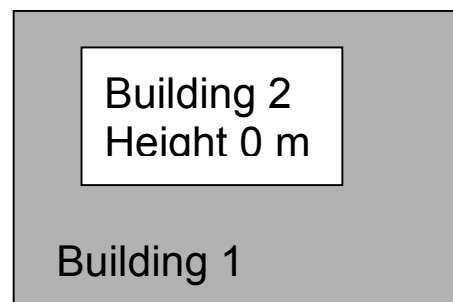


Figure 4: Courtyard (Version 2)

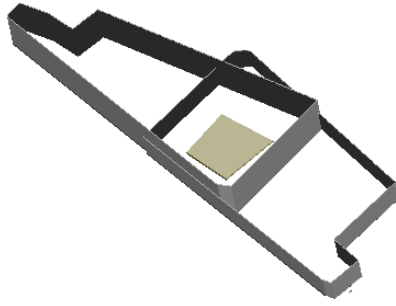


Figure 5. Courtyard definition (version 2) within overlapping buildings

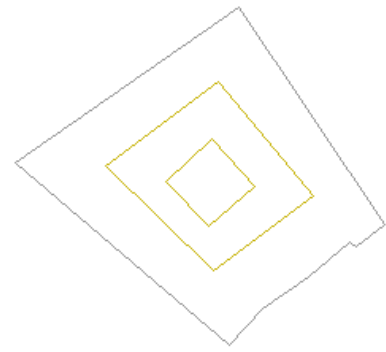


Figure 6: Multiple courtyards inside buildings are possible

It is possible to define a tower in a courtyard (see figure 6). But then the tower must be defined as tower/courtyard. If it is defined as standard building, the courtyard dominates over the standard building (see section 1.3).

- **Vegetation buildings**

This type of building is used to model vegetation (e.g. parks, trees,...). The “building” is transparent (the rays have no intersections with the “building”). The (part of the) ray inside the “building” can have an additional loss (per meter) and the pixels inside the “building” can also have an additional loss if they are predicted.

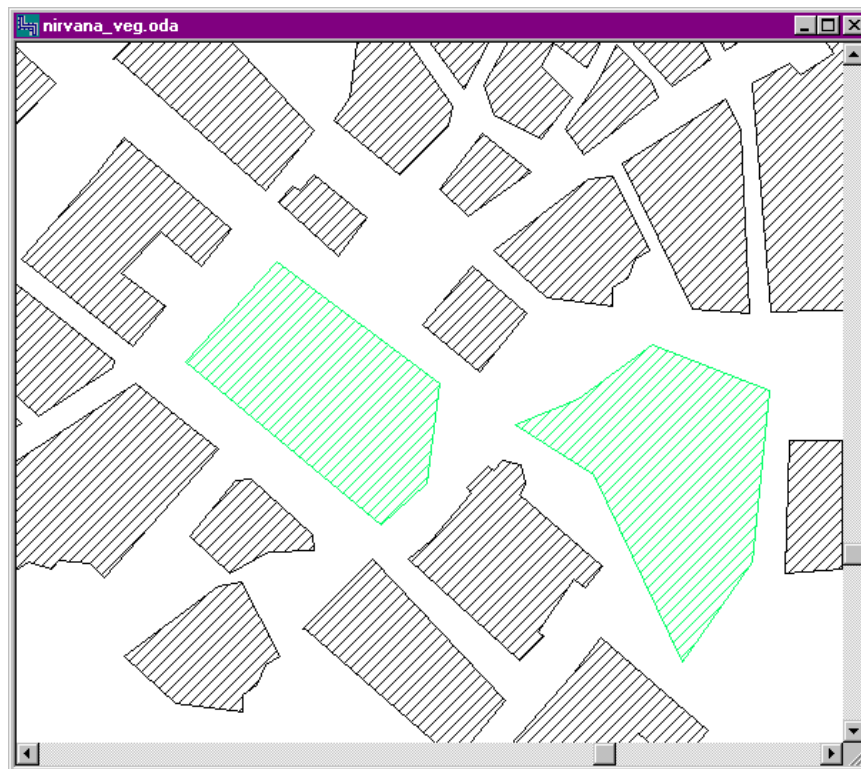


Figure 7: Vegetation “buildings” (drawn in green)

- **Virtual buildings**

With this type of building only pixels are excluded from the prediction (all pixels inside virtual buildings are not computed to accelerate the prediction and reduce the memory requirements). The virtual buildings are transparent for all prediction models. So the rays can pass the building without any additional loss. Only pixels inside the virtual buildings will not be predicted to save computation time.

Virtual buildings can be used at the border of the database where areas are free of buildings and which are of no interest (see figures 8 and 9). Or they can be used to exclude pixels in the middle of a river/lake in a city where predictions are not required.



Figure 8: Virtual buildings at the border of the database (dark grey in the left figure and blue in the right figure)

Table 1 summarizes the properties of the different types of buildings:

Type	Standard	Tower	Vegetation	Virtual
Interactions with rays (reflection, diffraction,...)	yes	yes	No, transparent	No, transparent
Prediction of pixels inside	Only with special indoor models	Only with special indoor models	Yes, but with additional attenuation (dB)	No.
Rays through the building	No	No	Yes, with additional attenuation (dB/m)	Yes. No additional attenuation
Material properties	Standard Set (Permittivity, Permeability, Conductance, Reflection Loss, Transm. Loss, Diffraction Loss)	Standard Set (Permittivity, Permeability, Conductance, Reflection Loss, Transm. Loss, Diffraction Loss)	Additional Attenuation for pixels [dB] Additional attenuation of rays [dB]	Properties not relevant
Building Type ID	0	3	2	1

Table 1: Types of buildings in urban databases

### 1.3. Intersections and priority of buildings

Buildings can intersect (if the polygonal groundplanes intersect). Therefore priorities must be defined to get a unique definition of an assignment of a given location (pixel) to a building. The following priorities are defined:

1. Courtyards (Towers)
  - Tallest courtyard (tower)
  - .....
  - Lowest courtyard
2. Standard Buildings
  - Tallest building
  - .....
  - Building with lowest height
3. Vegetation Buildings
  - Tallest building
  - .....
  - Building with lowest height
4. Virtual Buildings

This means that a courtyard is always more important and if a courtyard is defined, all other objects at this location are not relevant.

Buildings inside buildings are not considered at all because the priority is according to the height of the building.

## 1.4. Consideration of Topography

The topography can additionally be considered. Either the building heights in the vector database are defined inclusive topography or exclusive. This must be selected when preprocessing the database in *WallMan*. *WallMan* determines the topographical height of the center of the building and then either the relative height is obtained by subtracting the topo height from the absolute building height (if absolute height is given inclusive topography) or the absolute height is determined by adding the topo height to the relative building height.

Arbitrary topographical databases can be used and converted with WinProp. As topographical databases are pixel databases they are internally converted during the computation into a triangle approximation of the terrain profile (see figure 9).

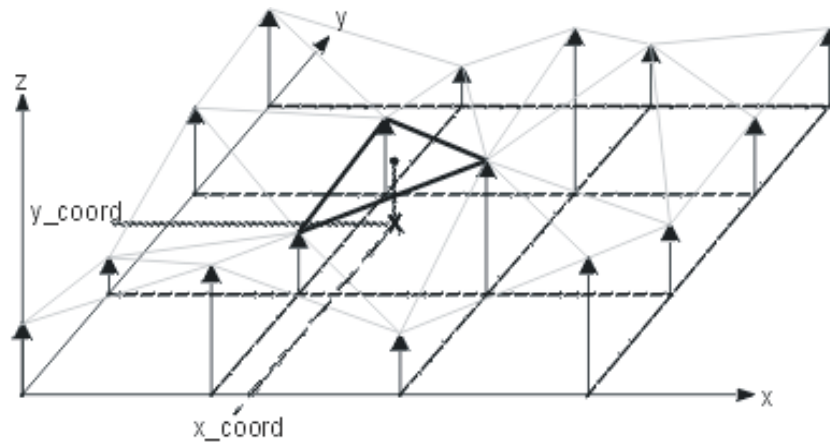


Figure 9: Terrain approximation with triangles

## 2. Description of the ASCII File Format (\*.uda)

The \*.uda database-files (“Urban Data ASCII”) are in a simple ASCII format. It is the raw format of the database. The \*.uda-files can be generated by the converters (see chapter 4) or by using *WallMan*

### 2.1 Header of File

A \*.uda-file starts with a header which is 6 lines long:

Test Buildings and header line	<i>Title, no effect on preprocessing</i>
Version: 1.0 ; Thu Feb 22 11:09:20	<i>Version and creation date as saved with WallMan. Only for information of user. Not read or considered during computation.</i>
Size of the area:	<i>Text for information. No effect on preprocessing</i>
Max. x = 950.00	<i>Size of the database [m] as saved by WallMan. No effect on preprocessing</i>
Max. y = 950.00	
86 100 100 950.0 950.0	<i>{Total number of buildings [integer]} {Offset (x and y coordinate) of database [integer or float]} {Size of the database in x- and y-direction [float]}</i>

Only line 6 is relevant and read. The first five lines are not considered and are just for information.

The *{Offset (x and y coordinate) of database [integer or float]}* is an offset which is added to all coordinates of the individual buildings.

The coordinates of the individual buildings are thus relative to the given offset and must be positive! The coordinates of the individual buildings should not exceed  $10^6$ .

The *{Size of the database in x- and y-direction [float]}* is the extension of the database relative to the lower left corner of the database, i.e. it is not the upper right corner in absolute coordinates but it is the vector pointing from the lower left corner to the upper right corner. This size can be larger defined in the file than the reality is. This is not important. Only if the size is smaller, all buildings outside the area defined by these parameters are not considered.

All values are in meter (because UTM coordinate systems are assumed) !!!!

The header consists always of these 6 lines.

The coordinates in the last line of the header are used in the preprocessing of the database. If they are changed, parts of the database might not be preprocessed or an error occurs if the selected preprocessing area is larger than given in the header.

## 2.2 Single building

After the header, there is one line for each building.

Each line is structured as follows:

```
{Building Index Number [integer]}
{Number of corners [integer]}

{x-coordinate of corner #1 [float]} ,
{y-coordinate of corner #1 [float]}
{x-coordinate of corner #2 [float]} ,
{y-coordinate of corner #2 [float]}
.....
{x-coordinate of corner #n [float]} ,
{y-coordinate of corner #n [float]}

{Building height [float]}
{Thickness of Walls [float]}
{Type of building [integer]}
```

This is a sample line for a building with 4 corners:

```
133 4 502.81, 666.00 540.75, 691.00 489.12, 791.75 475.50, 801.00 15.00 25.0 0
```

Table 2 shows the definition of the building parameters:

Parameter	Type	Description
Building Index Number	Index	Number to identify the building, e.g. during a prediction and in an error message. Number must be positive and greater than 0.
Number of corners	Index	Number of corners of the building/polygon (Only values greater than 3 are allowed).
x-coordinate of corner #n	Geometric	Coordinates [m] of each individual corner the building is described by. The values must be positive.
y-coordinate of corner #n	Geometric	
➤ <i>Note: The orientation of the corners should be counter-clockwise (top-view).</i>		
Building height	Geometric	Uniform building height [m] (in z-direction). Height can either be inclusive topography or exclusive (see section 1.4).
Thickness of Walls	Geometric	Thickness [cm] of the outer building walls (not used in current version of WinProp). Default value: 10
Type of building	Index	Code describing type of building: (0=Standard, 1=Virtual, 2=Vegetation, 3=Court Yard)

Table 2: Building parameters

### 3. Description of the binary (\*.odb) File Format

This file format is very complicated because it contains a lot of data used by ProMan and WallMan for internal communication. Therefore it is not published here.

Routines in DLLs for reading and writing the binary data format are available. Please ask for details.

### 4. Conversion of File Formats

For the conversion of databases and as an interface to the user, a raw ASCII data format is supported (\*.uda-file, see section 2).

Converters for different data formats are already available:

- DXF-format
- MIF-format (MapInfo)
- Arcview Shapefile
- Aircom Enterprise urban database format

The raw data (\*.uda-file) can also be generated by the user with an individual converter, which makes it possible to convert arbitrary databases to the **WinProp** format.

For conversion of databases the standard buildings are sufficient. Courtyards/towers can be determined automatically during the conversion in WallMan (further information is available in WallMan manual). Virtual buildings should be defined by the user to accelerate the predictions. But since they are not real buildings, they are not included in the databases provided by database vendors.