Mapping the Via Appia in 3D

Via Appia 3D GIS: Software User Manual

O. Martinez-Rubi
Netherlands eScience Center,
Science Park 140 (Matrix 1), 1098 XG Amsterdam, the Netherlands

November 24, 2014

Contents

1 Introduction

In this project, an area full of funerary monuments or sites in the fifth and sixth miles of the Via Appia Antica is thoroughly studied. This is possible thanks to the new 3D GIS system which combines several types of data: point clouds in different resolutions, meshes, pictures, paintings and attributes information of the several monuments.

All the previous data can be visualized in the new eScience 3D GIS system that combines a 3D viewer with analysis tools of the University of Groningen and a database which contains the attribute data of multiple sites. The interaction between the viewer and the database allows the users to refine what data is visualized based on attributes selection. This system, which can be updated when new data is available, allows multiple researchers in different locations to analyse and study the area in 3D and aims to function as an example for other complex archaeological study areas.

This document is the user manual for the new Via Appia 3D GIS system. Section 2 gives an overview of the system. The Via Appia Linux server is described in Section 3. The different types of data are detailed in Section 4 while in Section 5 you can find information regarding the used database. The main API is a 3D viewer with access to the database and it is detailed in Section 6.

2 Via Appia 3D GIS overview

The developed 3D GIS system has a two-tier architecture. The server contains the master copy of the data and a PostgreSQL database called *viaappiadb*. The clients download the data required for visualization and run the 3D viewer which connects to the remote database when required.

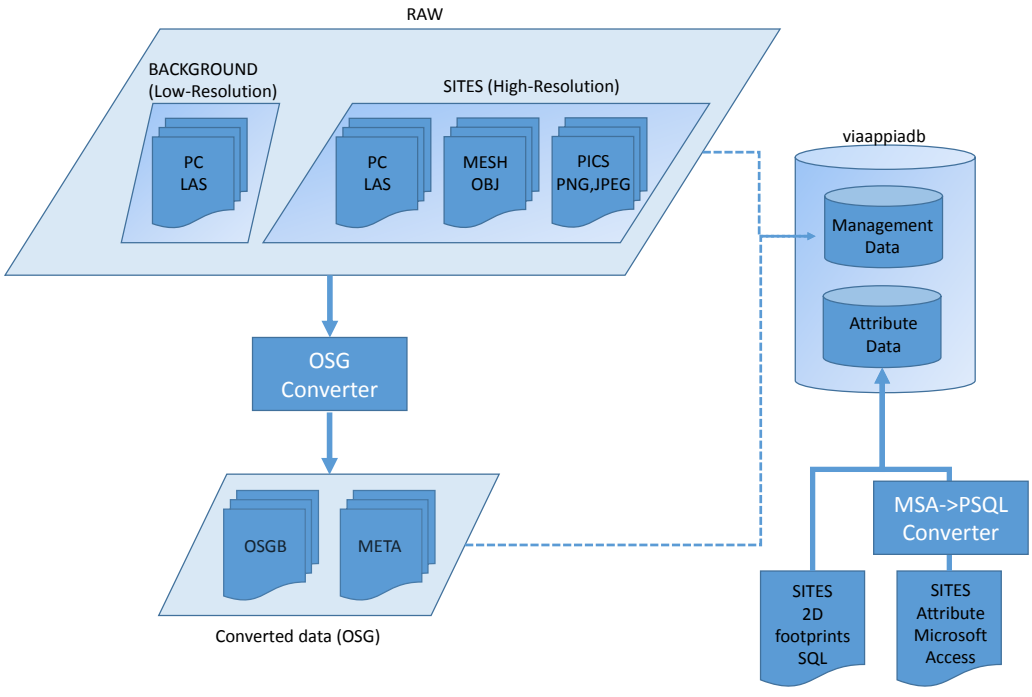

Figure 1: Data preparation framework executed in the Via Appia Linux server.

A diagram of the data preparation framework which is executed in the server is shown in Figure 1. The raw data is converted to the OpenSceneGraph (http://www.openscenegraph.org) binary format. The *viaappiadb* database is filled with meta-data information of the location of the raw data and the converted data. The archaeological information with attribute data for the several sites is provided in a Microsoft Access file. It needs to be converted to the PostgreSQL format before being imported into the main database. The footprints are provided in a PostgreSQL dump file and are imported into the *viaappiadb* database as well.

Once the data preparation framework has been executed in the server the several clients can start using the 3D GIS system. The clients (Windows desktops or laptops) need to download local copies of the data required for visualization. The *launcher* tool based on the Xenon library developed by the Netherlands eScience Center (NLeSC) is used for this purpose. The tool also downloads the configuration file required by the 3D viewer. Once both the data and the configuration file are locally available in the client (Windows computer) the *launcher* tool automatically starts the 3D viewer. Figure 2 illustrates the two-tier architecture and shows the steps required in the client done by the *launcher* tool.

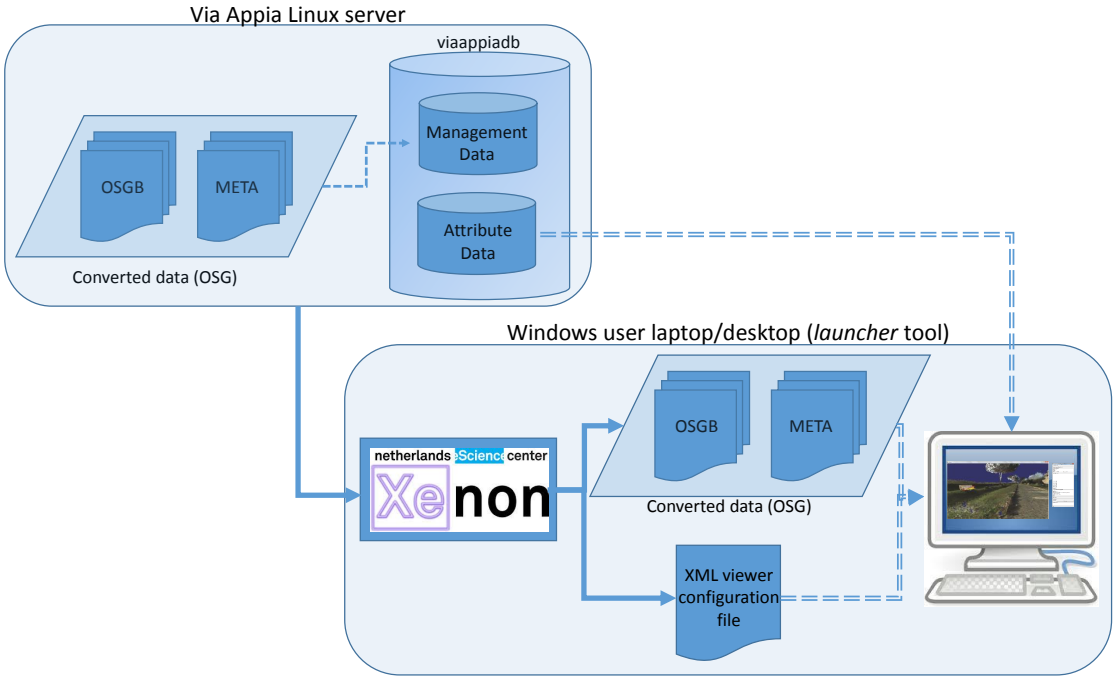

Figure 2: Two-tier architecture of the Via Appia 3D GIS and the steps executed in the clients (launcher tool).

Both the execution of the data preparation framework in the server and the data synchronization steps in the client (*launcher* tool) have been automated. In order to use the new 3D GIS system the end-user only needs to do:

- (a.) Obtain an account in the Via Appia Linux server and in the *viaappiadb* database and generate a SSH key pair (see Subsections 3.3 and 3.4).
- (b.) Download, install and configure the *launcher* tool in his Windows laptop or desktop (see Subsection 6.1). The *launcher* tool also contains the 3D viewer. It is also recommend to install two tools for the communication between the Linux server and the Windows computers: PuTTY and WinSCP (see Subsubsections 3.2.1 and 3.2.2).

3 Via Appia Linux server

The Via Appia Linux server contains the master copy of the data and the *viaappiadb* database. Any user attempting to use the Via Appia 3D GIS system must have access to the server and to the database.

3.1 Specifications

The Via Appia server is rented from Hetzner (http://www.hetzner.de). The specifications of the server are:

- OS: CentOS 6.5 64 bit
- 1 x CPU: Intel 8 Cores i7-4770 CPU @ 3.40GHz
- Memory: 32 GB
- Storage:
  - Western Digital RE WD2000FYYZ 2TB 64MB Cache SATA 6.0Gb/s (mounted in /home)
  - 2 x Intel SSDSC2BW240A4 530 Series 240GB in RAID 1 (mounted in /, /boot and SWAP)
- Network:
  - RealTek RTL-8169 Gigabit Ethernet driver
  - IP: 148.251.106.132

3.2 Linux - Windows communication

The server is a Linux system while the clients use Windows computers. There are many options in the market for the communication between Linux and Windows systems but for the Via Appia 3D GIS system we use the tools described in the following Subsubsections.

3.2.1 Windows SSH client

In order to have a command-line environment we connect to the server through ssh. From another Linux system the server can be reached with:

```
ssh userName@148.251.106.132
```

For Windows systems it is advisable to download and install PuTTY (http://www.putty.org) in order to get a terminal-like environment. The IP of the Linux server (148.251.106.132) and your user name in the server are required.

3.2.2 Windows SCP client

For simple data transfers between Windows systems and Linux systems and vice versa we recommend to download and install WinSCP (http://winscp.net). The IP of the Linux server (148.251.106.132) and your user name in the server are required.

3.2.3 Xenon

When executing the 3D viewer in the Windows computer the required data is automatically downloaded from the Linux server. The *launcher* tool is in charge of this task and it is based on the NLeSC Xenon library (http://estep.esciencecenter.nl/index.php/component/k2/item/1-xenon).

3.3 Accounts

In order to obtain an account in the server the system administrator must be contacted (o.rubi@esciencecenter.nl). In addition, an account specific for the *viaappiadb* database should also be created. Usually both accounts have the same name.

After your account in the server has been created you can already connect to the server with ssh, through PuTTY in Windows systems.

Once the database account has been created and you are connected to the server you can access the DB locally using the *psql* client. You need to set your PostgreSQL password (which can be different to your operative system password). You can set your PostgreSQL password with (replacing *username* with your PostgreSQL user name and *password* with a password of your choice):

```
psql -c "alter user username WITH PASSWORD 'password'";
```

The *viaappiadb* database can also be accessed using *phpPgAdmin*. With an internet browser connect to *http://148.251.106.132/phpPgAdmin*. The generic user is *viaappia* and the generic password is *peileed0woh7AhY3*. Next you also need to specify your PostgreSQL user and password.

3.4 SSH keys

For the proper working of the 3D GIS system a SSK key pair has to be created in the Via Appia Linux Server. The steps are:

- Connect to the Via Appia Linux server. Use PuTTY in Windows systems.
- Create a key pair:

  ```
  ssh-keygen -t rsa
  ```

  Press *Enter* three times to use the default configuration (Important: Do not provide a passphrase).

- Add the public key to the authorized keys:

  ```
  cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
  ```

- Change permissions to the *.ssh* directory and its contents:

  ```
  chmod 600 ~/.ssh/
  chmod 600 ~/.ssh/*
  ```

After the key pair has been generated in the Via Appia Linux server you need to download a copy of the private key to your Windows computer. You need this key when configuring *launcher* tool. We recommend to use the clipboard together with PuTTY:

- Connect to the Via Appia Linux server from your Windows computer with PuTTY.
- Once your are connected open the private key file:

  ```
  less ~/.ssh/id_rsa
  ```

- Select all the text of the key and "copy" it (right button of the mouse and click on *Copy*). Be sure that you have selected all the key, we recommend to maximize the window.
- Create an empty text file in your Windows machine (for example in *Documents*) and "paste" the private key.
- Save the file and name it *id_rsa*.

4 Data

The types of data that the Via Appia 3D GIS system deal with are:

- A low resolution point cloud of the research area that has been generated making use of Fugros DRIVE-MAP services.
- High resolution point clouds and meshes of the different objects of interest (sites) generated using photogrammetric technologies.
- Sites reconstructions (meshes) for different epochs.
- Historical pictures or paintings.
- Attributes information for the sites and their parts, which are gathered by field observations, such as composition, condition, interpretation, description of the different elements, etc.

However, the data needs to be converted to the OpenSceneGraph binary format which is used by the 3D viewer. Hence, we do a distinction between raw data and OSG (converted) data. The master copy of the data is in the Via Appia Linux server where also the database required by the system is running. When new raw data is available, for example a new point cloud for a certain site or a new reconstruction of a site, it is required to add it to the raw data locations in the Via Appia Linux server. After adding the new data in the server location for the raw data we need to run the part of the data preparation framework that converts the data into the OSGB format and updates the *viaappiadb* database with the latest changes. For this purpose we use the Python script called *createosgdata.py*.

There are certain rules regarding the data that need to be followed for the correct functioning of the system. They are described in the following sections. Please, read them carefully.

All the data is stored in the directory ***/home/vadata/DATA***. Only the user ***vadata*** has permission to modify the content of the data directory. Therefore, before adding new raw data we have to change from our user to the *vadata* user. Once connected in the Via Appia Linux server we can become the *vadata* user with:

```
ssh vadata@localhost
```

Only certain users can currently use the *vadata* user: Oscar Martinez Rubi, Stefan Verhoeven, Maurice de Kleijn and Rens de Hond.

One usual use case is when we have new data in our Windows machine. In such case use WinSCP to transfer the data from your Windows systems to your home directory in the Via Appia Linux server. Then, close WinSCP and use PuTTY to connect to the server in a terminal-like environment. The first thing you need to do is become the *vadata* user. Next you have to copy the added data to the proper location in the directory */home/vadata/DATA/RAW*.

4.1 Raw data

The raw data is **only** stored in the Via Appia Linux server and there is Python script that needs to be used to generate the converted OSG data, the *createosgdata.py*. The raw data directory is */home/vadata/DATA/RAW*. In the next Subsubsections we give more information about the different types of raw data. All the data locations are relative to the directory ***/home/vadata/DATA/RAW***.

4.1.1 Background point cloud

Using the DRIVE-MAP service of Fugro the fifth and sixth miles of the Via Appia were scanned and a point cloud was produced. The spatial reference system of this data is EPSG:32633. The resolution of this point cloud is not enough to see detailed features and the sites were only scanned from the main road. Thus, the back of the sites is missing.

The background LAS files are stored in *PC/BACKGROUND/DRIVE_1_V3*. In the folder *PC/BACKGROUND* there must be only one sub-folder (currently DRIVE_1_V3). If a new version of the DRIVE-MAP is used you must delete the previous one (or moved it out of this directory).

About the names of the background sub-folders, they can not contain dots, so valid names are for example DRIVE_1_V3.

4.1.2 Sites

For the different sites (objects of interest in the area) several type of data are also used.

Point clouds

Higher resolution point clouds must be stored in the directory *PC/SITES*. This directory contains different folders for each site. For example the folder *PC/SITES/162* contains LAS files of different point clouds of site 162.

The LAS files contained in each site subfolder may have been pre-aligned (using CloudCompare for example). In that case the LAS file name must be ***_aligned_*BGNAME*** where *BGNAME* must be the background name (as contained in the folder *PC/BACKGROUND/*). Some point clouds generation tools store the color information in 8 bits instead of the usual 16 bits. In that case the file name must be ***_8bitcolor***. The effect of having an undeclared LAS file with 8 bit color is that the converted data will be black and white. Note that these properties are cumulative, for example *162_aligned_DRIVE_1_V3_8bitcolor.las* is a valid name for a LAS file with 8 bit color information and aligned points.

Meshes

The meshes are stored in the directory *MESHES*. There are two types of meshes: (a.) current mesh representations of the sites, and (b.) archeological reconstructions. They must be stored respectively in *MESHES/CURR* and *MESHES/ARCH_RECONS*. Each of these directories contains different folders for each site. For example the folder *MESHES/CURR/162* contains mesh representations of the current state of the site 162.

For each site there must be different folders for the several meshes. Inside these folders the files for the meshes are stored (OBJ, MTL, JPEG). For example *MESHES/CURR/162* could contain two folders called *162_curr_1* and *162_curr_2* and each of them contain a OBJ, a MTL and several JPEG files.

Similarly to the LAS files the meshes can also be aligned. In this case the folder name for a certain mesh must be ***_aligned_*BGNAME***.

Pictures

The pictures are stored in the directory *PICTURES*. There are two types of pictures: (a.) pictures of the current state of the sites, and (b.) historical pictures or even paintings. They must be stored respectively in *PICTURES/CURR* and *PICTURES/HIST*. Each of these directories contains different folders for each site. For example the folder *PICTURES/CURR/162* contains pictures of the current state of the site 162. Note that in this case these folders directly contain the images similarly to the point clouds but contrary to the meshes where the folder for each site contains subfolders.

Attributes data (Microsoft Access file)

The attribute information gathered in the field is stored in a Microsoft Access file. The content of this file has to be imported into the *viaappiadb* database. In order to do this conversion we recommend to use the tool *Bullzip Access to PostgreSQL* which is available for Windows machines. The steps for the conversion are:

- Install in a Windows system:
  - Microsoft Access.
  - Bullzip Access to PostgreSQL (http://www.bullzip.com/products/a2p/info.php).
- If the format of the file is not *.mdb* we recommend to reformat the file. If this is required follow these steps (example for Microsoft Access 2010):
  - Open the Access file.
  - Click Save and Publish.
  - Click on save database as Access 2002-2003 database.
- Execute the converter:
  - Select the Access file.
  - If there is an error message about a missing driver, just follow the recommended instructions.
  - Select create dump file.

  - Leave all the default values except creating indexes. Disable it because otherwise it would create indexes with erroneous names.
  - Run the converter.

- We still need to do some modifications to the generated SQL dump file:
  - Remove the *CREATE DATABASE* statement in the beginning of the file.
  - Remove any default values in all columns with type TIMESTAMP.
  - Remove all the double quotes (").
  - Be careful with possible table names with blank spaces (which should be removed).
  - Remove all index creations (if you did not de-select the index creation).
- Once the PostgreSQL dump file is generated, use WinSCP to transfer the file to the Via Appia Linux server.
- Use PuTTY to log in into the server and become the *vadata* user.
- Move the SQL dump file to the *ACCESS* folder.
- Still with the *vadata* user execute the Python script *mergeaccess.py* to update the Attribute information in the *vadata* database. Note that the attribute tables of the *viaappiadb* database will be overwritten.

Footprints

The footprints SQL dump file has to be put in the directory *FOOTPRINTS*. The table in this file has to be called *sites_geoms_temp* and contain (at least) the columns *site_id* and *geom*. To import a new footprints file use the Python script *mergefootprints.py*

4.2 OSG

The raw data needs to be converted to the OSGB format. For that purpose we have created the Python script *createosgdata.py*. This script should be run with the *vadata* user and should be run when some data has changed in the raw data directory.

In order to view the data in your Windows machine you need to download a version of the converted OSG data in your local machine. We use a NLeSC tool called Xenon.

4.3 Data structure overview

The data stored in the Via Appia Linux server follows a certain data structure which has been introduced in the previous Subsubsections. In Figure 3 we show the data structure overview, the folders and files in **bold** are within the defined structure while the ones in *italic* are examples. Only the *RAW* folder is analysed because it is in the raw locations where the users can make modifications, the *OSG* folder is automatically handled by the system.

```
DATA
  |- RAW
  |    |- PC
  |    |    |- BACKGROUND
  |    |    |      |- DRIVE_1_V3
  |    |    |           |- 1.las
  |    |    |- SITES
  |    |    |      |- 162
  |    |    |          |- 162_aligned_DRIVE_1_V3_8bitcolor.las
  |    |- MESHES
  |    |    |- CURR
  |    |    |      |- 162
  |    |    |          |- 162_aligned_DRIVE_1_V3
  |    |    |              |- 162_aligned_DRIVE_1_V3.obj
  |    |    |- ARCH_RECONS
  |    |- PICTURES
  |    |    |- CURR
  |    |    |      |- 162
  |    |    |          |- 162_current.png
  |    |    |- HIST
  |    |- ACCESS
  |    |    |- attributes.sql
  |    |- FOOTPRINTS
  |    |    |- footprints.sql
  |- OSG
       |- PC
       |    |- BACKGROUND
       |    |- SITES
       |- MESHES
       |    |- CURR
       |    |- ARCH_RECONS
       |- PICTURES
       |    |- CURR
       |    |- HIST
       |- DOMES
       |- BOUNDINGS
```

Figure 3: Data structure overview

5 Database

The *viaappiadb* database is running in the Via Appia Linux server. See Subsection 3.3 on how to get an account in the database. The database is a PostgreSQL 9.2.8 instance running with PostGIS 2.1.2.

This database has two categories: (a.) data management information and (b.) the attribute data. In next Subsections we describe each category. In Figure 4 we show the Entity-Relationship diagram of the *viaappiadb*.

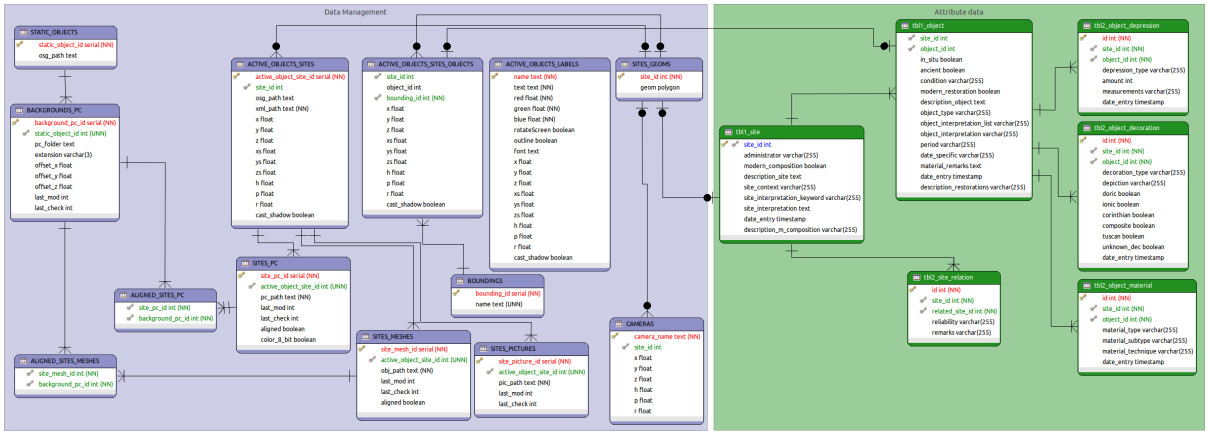

Figure 4: Entity-Relationship diagram of the *viaappiadb* with its two categories

Note that some of the nodes of the relationships are *0..n* or *0..1* (with black points) instead of the usual *1..n* or *1..1*. This is to illustrate that some sites and objects may have entries in some tables but not in others. For example it is possible to have a site in the attributes table (tbl1_site) that does not have any entry in the footprints table (sites_geoms).

5.1 Data management

This category contains tables with the locations of the raw and converted data, visualization-related data and the footprints of the sites.

- Raw data: BACKGROUNDS_PC, SITES_PC, SITES_MESHES and SITES_PICTURES contain the locations of the different types of raw data, concretely the low resolution point cloud of the background and the per-site high resolution point clouds, meshes and pictures. They also contain information about aligned point clouds and meshes (ALIGNED_SITES_PC and ALIGNED_SITES_MESHES) and 8-bit-colors in point clouds.

- Converted data: STATIC_OBJECTS and ACTIVE_OBJECTS_SITES contain the locations of the converted background point cloud and the converted per-site data, i.e. high resolution point clouds, meshes and pictures. The data of these tables is used in visualization.

- Visualization-related data: ACTIVE_OBJECTS_SITES_OBJECTS contains the different positions of the bounding boxes around the objects of the sites. Note that we consider the site itself to be an object as well, in this case it is the object with *object_id -1*. There are different types of bounding boxes listed in table BOUNDINGS.

ACTIVE_OBJECTS_LABELS and CAMERAS have information about labels added into the 3D scene and camera positions respectively.

- Footprints: The table SITES_GEOMS contains footprints of different sites.

The raw data is maintained and modified with the user *vadata* (see Section 4) while the converted data is generated with the *createosgdata.py* script. This tool also generates some visualization-related data which can be modified and extended through the 3D viewer. The footprints can be updated with the *mergefootprints.py* Python script.

5.2 Attribute data

The attribute data is a direct import from the Microsoft Access file that contains the attribute information for the sites. See the paragraph regarding the Attribute data in Subsection 4.1.2 to see how to import the Microsoft Access file into the *viaappiadb*. Note that every time that a new Attribute file is imported the tables are overwritten.

6 Via Appia Windows launcher and 3D viewer

Before being able to run the viewer, we need to download the data from the Via Appia Linux server to the user Windows computer. Not all the data is required, only the converted data is actually used for the visualization. The *launcher* tool takes care of (a.) synchronizing the data between the two systems, (b.) starting the 3D viewer, and (c.) updating the *viaappiadb* with possible changes done with the viewer (for example a repositioning of a mesh).

6.1 Installation

The GitHub repository in https://github.com/NLeSC/Via-Appia contains all the available software. Download the code either by cloning the repository with Git software or by downloading a ZIP (in the repository URL clink on "Download ZIP". If you do not have access to the repository contact Oscar Martinez Rubi, Stefan Verhoeven, Maurice de Kleijn or Rens de Hond.

Once you have acquired the ViaAppia software you can find the *launcher* tool in the directory launcher/binary/ViaAppia.bat.

6.2 Requirements

The launcher requires the JAVA SE development kit:

- Browse to http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Download the Java SDK.
- Install it (follow the instructions).

The 3D viewer requires the 2012 and 2013 Microsoft Visual Studio C++ Redistributable. You can either download them or use the installers which are already provided in the ViaAppia software in the directory viewer/EXT_LIB. In addition you should download the latest drivers for your graphics card.

6.3 Configuration

After solving the requirements both the *launcher* and the 3D viewer are ready to work. However, the *launcher* needs to be configured.

In launcher/binary directory you need to create a file called config.properties (Important: be careful with hidden extensions). You can use as template the provided config_template.properties. We recommend to "Copy-Paste" the file and rename the new file to config.properties. Open the file and fill in your details:

- Replace the tag <UserRemote> with your user name in the Via Appia Linux server.
- Replace the tag <UserLocal> with your user name in your Windows computer.
- Replace the tag <localPrivateKey> with the file name of the private key that you copied from the Linux server. If you have not done this yet see Subsection 3.4.
- In the template file we assume that you are storing all the Via Appia in:

  `C:\\Users\\<UserLocal>\\ViaAppia`

  If that is not the case you may need to explicitly change some other properties.
- In the template file we also assume that the root directory of the Via Appia software is in:

  `C:\\Users\\<UserLocal>\\ViaAppia\\Via-Appia-master`

  If that is not the case you need to change the *viewer* property.
- Note that in some properties the separation between folders is done with / and in others is done with \\. For each property use the same format as in the template file.

6.4 Execution

Once the configuration is properly filled in you can execute the *launcher* tool with the script in launcher/binary/ViaAppia.bat.

The execution of this tool will synchronize the data from the Via Appia server into your Windows computer and will download the latest configuration file required by the viewer. When the synchronization is done, the viewer is automatically started.

Once you are done with the viewer be sure to save the progress. After closing the several windows in the screen the *launcher* tool will ask you if you want to commit your changes to the *viaappiadb*.

6.5 3D Viewer API

The *launcher* tool starts the 3D viewer after the synchronization of the data is finished. Two windows are visible: one with the 3D scene and the other one with the GUI which contains four tabs. In the next Subsubsections we describe each of these tabs. By clicking on the Help button in the GUI main menu bar a more detailed help text can be displayed (only in Dutch).

In order to load the data into the scene we need to open the XML configuration file. Click on File and Open, navigate to the folder where the data has been downloaded and select the ViaAppia.conf.xml file (the default is /Users/<UserLocal>/ViaAppia/DATA/OSG/ViaAppia.conf.xml).

This will load the 3D scene and some data will already be visible in the visualization window. By default only the low resolution background point cloud is visible. This data is static and can not be moved, resized or rescaled. For each site there can be different pictures, meshes and point clouds. These are active objects which are not visible by default but when they become visible they can be moved, resized and rescaled.

A very important characteristic of this GIS system is that each site is identified by its bounding cube. Within each site we can also identify objects or parts of the sites which are also defined by their bounding cubes. The bounding cubes (for a site or for the parts of a site) are also active objects, i.e. they can be moved, resized and rescaled. As the other types of active objects, they are not visible by default.

6.5.1 Main tab

After loading the configuration file the the main tab of the GUI looks like Figure 5.

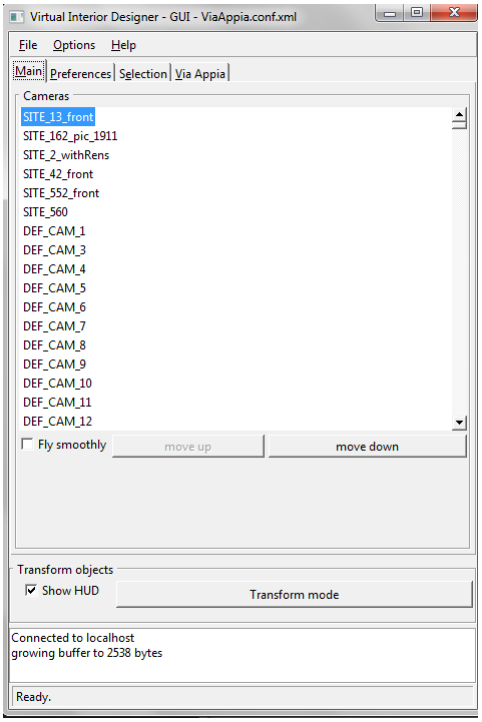

Figure 5: Via Appia 3D viewer GUI: main tab

This tab lists all the camera positions. The user can add new camera positions. Click on the mouse right-button on top of the cameras list and select "Create new". The cameras can be related to a certain site. If that is the case the camera name must start with *SITE_XXX* where *XXX* is the site identifier. For the sites without user-defined camera positions a default camera position is generated based on the sites footprints.

6.5.2 Preferences tab

The preferences tab looks like Figure 6 and it has some visualization options that can be modified. The most important one is the Level of detail. Computers with low visualization

resources (without powerful GPU cards) should run with lower Level of detail. Hence, if you experience a non smooth visualization try to decrease the Level of detail.

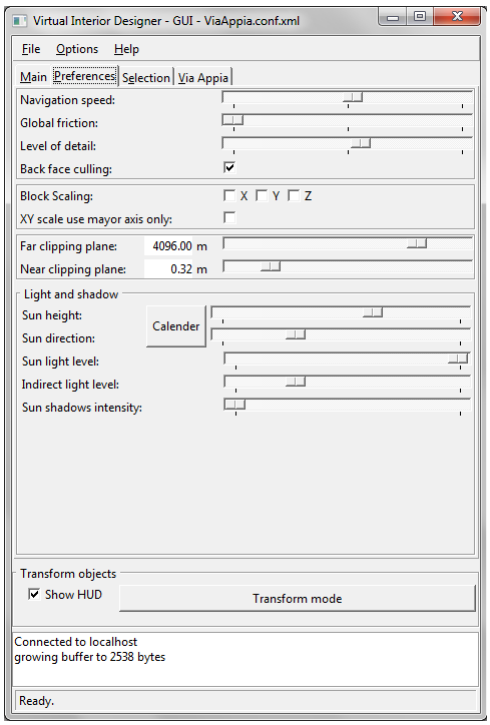

Figure 6: Via Appia 3D viewer GUI: preferences tab

The speed of the navigation is also configurable.

6.5.3 Selection tab

Clicking the "t" key activates the Transform mode. In this mode we can modify the position, size and rotation of active objects, i.e. points clouds, meshes and pictures related to the sites, and also their bounding cubes and bounding cubes of their parts / objects. In addition we can use the *ruler* tool. Moreover we can add labels into the 3D scene and define new bounding cubes for new parts or objects within sites. See section 6.5.5 for more information about adding new labels and bounding cubes.

When you select an active object in the 3D scene, its details appear in the Selection tab as for example shown in Figure 7. The active object can be moved, resized and rotated directly in the visualization of by specifying the values in the Selection tab (very useful when doing large changes).

6.5.4 Via Appia tab

The last tab is called Via Appia. It is used to toggle the visibility of the different active objects. It contains two parts:

- The *viaappiadb* connector. SQL statements can be constructed to access the Attributes data of the *viaappiadb*. The selected columns of the SQL statement must be *site_id* or *site_id,object_id*. After a SQL statement has been constructed and the connection details

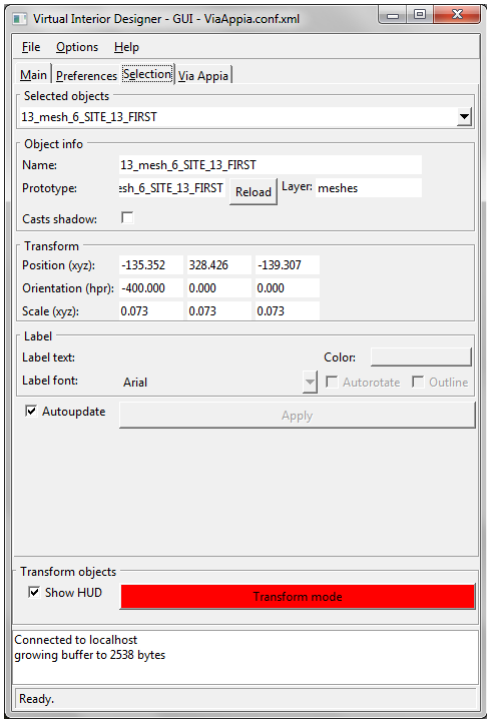

Figure 7: Via Appia 3D viewer GUI: selection tab

have been properly filled up it is executed in the remote DB. The bounding cubes related to the selected sites and objects (parts) will become visible in the 3D scene. If there is only one site or part selected in the DB and the *select* check-box is checked in the GUI, the bounding box is selected in the 3D scene and it can be modified (see Selection tab).

- The active objects tree. The rest of active objects (point clouds, meshes, pictures and labels) are listed and can be showed or hidden in the 3D scene.

Note that even if different active objects become visible the user will only see them if the camera position is pointing at them. To change the camera position use the main tab.

6.5.5 Naming rules

For the proper functioning of the system there are some rules that must be followed when adding new cameras, labels or bounding boxes:

Cameras

By default one camera is created per site and it is located in the center of the footprint of the site. The users can create new cameras but their names must start with *SITE_XXX* if they are related to a certain site.

Labels

When adding a new label the *Name* (in the Object info box) must contain be **label**. When a label is related to a site we suggest to use a name like *SITE_XXX_label*. The *Label text* (in the Label box) can contain any text.

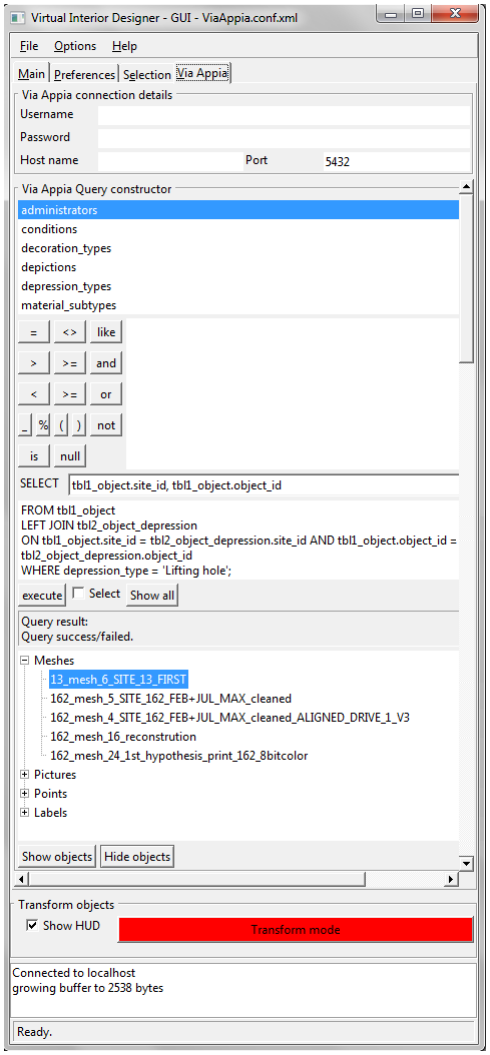

Figure 8: Via Appia 3D viewer GUI: Via Appia tab

Boundings

The different objects or parts of a site are identified by an *objectId*. We consider that the site itself is a object or part of the site, in this case it is the *objectId* -1. When adding a bounding for a new site use the "t" key to enable the Transform mode. Use the last option to add a bounding box and move it and rescale it to contain the site you want. The *Name* (in Object info in Selection tab) must be *[siteId]_obj_-1*.

When adding a new bounding box for a object or part of a site the *Name* must be *[siteId]_obj_[objectId]*. For example 162_obj_10. Please be sure that the *objectId* that you choose is not being already used. To see if it is used you can use the *viaappiadb* connector. For example by executing the SQL query *SELECT 162,10;* (and checking the Select check-box) you can see if there is already a object defined with id 10 for the site 162.

6.5.6 Unsynchronized 3D viewer

If you wish to start the viewer without data synchronization you can do it directly with the script in viewer/viewer/startViewer.bat. However, note that since the synchronization is deactivated this will start the viewer with probably older data and a older configuration file. In addition, in this mode the changes done in the scene are not automatically committed to the *viaappiadb*. Therefore, we discourage using the viewer/viewer/startViewer.bat.