GoogleEarthTweetMapper: Documentation and User Manual

Written by: Lukas Graf, Stefan Mirkovic and Levente Papp

Winter Term 2018/2019 - Practice: Basics of Software Development

University of Salzburg – Department of Geoinformatics, Z_GIS

General Overview

GoogleEarthTweetMapper is a Java-based software to automatically display Twitter data stored in a CSV file in Google Earth using a user-defined ground overlay as background. Therefore, a keyhole-markup file (kml) is generated. The ground overlay itself is retrieved from a WMS-Server by sending a GetMap request for a selected bounding box. The GetMap request is stored locally on the computer and then used to generate the kml file.

The general structure of the program is shown in Figure 1 (the names in the boxes denote the names of Java classes). Firstly, the graphicalUserInterface class (in the code it is called GoogleEarthTweetMapper) is used to collect the user inputs. Secondly, a WMS connection is established to retrieve a rendered map for the ground overlay element in the kml file (WMSConnector and Img2GroundOverlay class). In meantime, the Twitter data is parsed from the CSV file and then inserted into a kml file structure (CSVParser and Tweets2KML, respectively). Finally, Google Earth is started to display the gernerated kml file.

A more detailed description of the single Java classes can be found in the following section of this document.

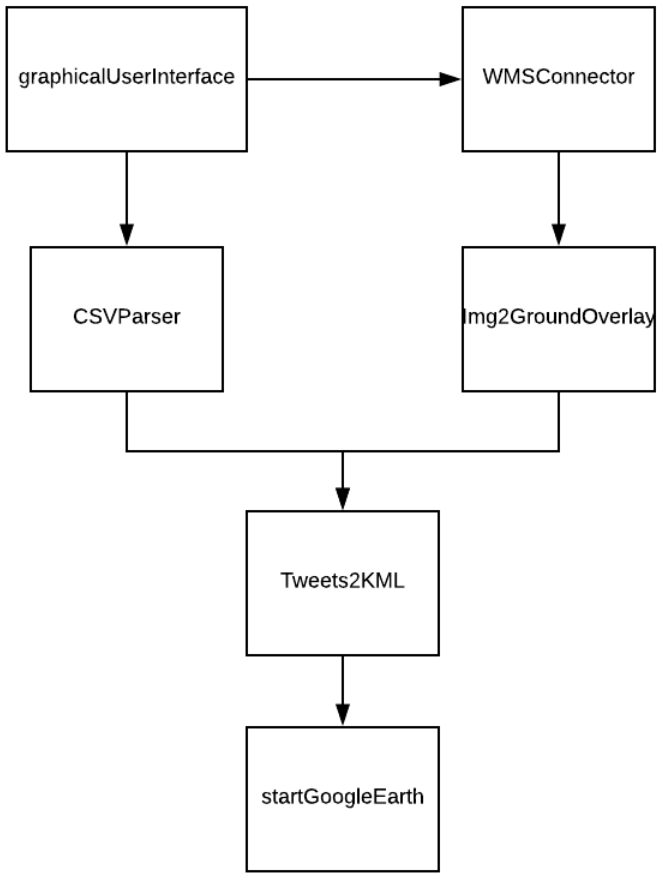


FIGURE 1: OVERVIEW OF THE JAVA CLASSES USED IN GOOGLEEARTHTWEETMAPPER AND THEIR CHRONOLOGICAL SEQUENCE IN THE PROGRAM CYCLE.

Documentation of Java Classes

googleEarthTweetMapper.java

This Java class is used to generate a graphical user interface (GUI) and contains the main method to collect all the required user inputs to run the program. The GUI is build using the java.swing package that provides a powerful collection of classes to generate and display graphical elements. For the list of user inputs please refer to the "User Manual" section. The interface itself is generated in the constructor of the class graphicalUserInterface(). As the class extends the java.swing.JDialog class it inherits all attributes and methods from the parent class.

Besides the inherited attributes and methods, the class contains following additional functionality:

Method Name	Description
public static vouid main()	As the whole GUI is defined in the constructor of the class including all the in- and output fields as well as buttons, this main method is used to start up the GUI whenever the program is called. It generates a new thread using the "EventQueue" class

	that sets the GUI visible and allows for collecting all the inputs the user makes. This method calls the callWMSConnector() method.
private void callWMSConnector()	This private class is used to collect the user-inputs from the GUI and to call the WMSConnector class. After the WMSConnector class has sent a GetCapabilities request also the list of available layers is shown to the user who is then asked to graphically select one layer. This layer is then used to retrieve the rendered map as image file. The interactive layer selection allows for maximum flexibility of the whole program and is the reason why this class belongs to the graphicalUserInterface class. This method calls the private void retrieveImgFromWMS(WMSConnector wmsConn_, Layer layer_) method.
private void private void retrieveImgFromWMS(WMSConnector wmsConn_, Layer layer_)(WMSConnector wmsConn_, Layer layer_)	This private class is just a wrapper around the retrieveImageFromWMS method from the WMSConnector class. Its arguments are a WMSConnector object that holds the connection to a WMS server and a Layer object that specifies which layer should be used from the WMS.

WMSConnector.java

This class is used to connect to a user-defined WMS-Server, to send a GetCapabilities request, return a list of available layers and to retrieve a rendered map as image from the WMS GetMap response. The whole communication between the client and the server is OGC-compliant. Compliance is ensured by using geotools, an open-source library for geospatial data and web service access, manipulation and processing.

The advanced constructor of the class allows to setup all the required parameters for a WMS connection including the server URL, the spatial reference system (SRS), the extent of the bounding box (BBox), the height and width of the output image (imageDimensions) as well as the directory where the image should be stored (storageLocation) and whether the rendered map should be transparent or not:

```
WMSConnector

    (String URLString_, String bbox_, String SRS_, String storageLocation_, boolean transparent_,

        String[] imageDimensions_)
```

The class contains following methods:

Method Name	**Description**
public void getWMSConnectionParams()	Returns all the parameters set for establishing a WMS connection including the server URL, spatial reference system, chosen layer, bounding box and dimensions of the output image.
public void setWMSConnectionParams()	Allows to set the WMS parameters when the default instead of the advanced constructor was used. This requires the same input parameters as for the advanced constructor (see above).
private WebMapServer connectWMS()	This private method opens a connection to a WMS server and returns a WebMapServer object.
public Layer[] getLayerList()	This method returns a list of all layers that are available from a given WMS server. This list is used in the GUI for user-defined selection of one layer. Therefore, a GetCapabilities request is used.

public int retrieveImageFromWMS(Layer layerName_)	This method is used to send a GetMap request for a selected layer to the WMS server. The request also contains the bounding box, the SRS as well as the dimensions of the resulting image that is stored locally on the computer. If everything was successful, the method returns an integer equal zero.

CSVParser.java

The CSVParser class is used to extract the tweets stored in a csv file into a TweetData structure that is defined in the TweetData auxiliary class. Each TweetData element takes the coordinates of the tweet (latitude and longitude) as double, the actual message content and the creation time as strings and stores temporarily in the memory until the tweets can be written to a kml file structure using the Tweets2KML class.

The class has one method getTweets:

Method Name	**Description**
public static TweetData[] getTweets(String csvFile)	The method takes the name (including the absolute path) of the CSV file containing the tweets and extracts it into an array of TweetData structures as defined before. For reading from the file a file reader object is used. The program scans through the file line by line and splits each single line by the delimiter (semicolon). It takes those substrings (tweet, created_at, lat, lng) that are relevant for the further processing steps. If no file is found a FileNotFoundException is raised.

Img2GroundOverlay.java

This class takes the retrieved image file from the WMSConnector class and embeds it into a kml-file structure using a Ground-Overlay element. It inserts the image file via a “href” command and uses the extent of the bounding box that was given by the user. It is assumed that the projection of the image as well as of the coordinates of the bounding box are given in WGS84 (EPSG:4326). The class has one method (WriteToKML) to perform this task:

Method Name	**Description**
public static boolean WriteToKML(String FilePath, String ImageSource, String name, String description, double north, double south, double east, double west)	The method takes the storage location as well as the file name of the png file, the extent of the bounding box, and a description string to create a kml structure containing a ground overlay element with 30% transparency. The kml file is stored at the same user-defined location as the input png file. For writing content to a file, the Java.IO.FileWriter class is employed. If everything was successful, the method returns the Boolean value “True”, otherwise “False”.

Tweets2KML.java

This class is used to write the extracted tweets (including their geolocation) into a kml file structure. The structure of the tweet data is defined in a separate class called TweetData.java. The purpose of the TweetData merges the coordinates and the columns "tweet" & "created_at" as kind of container for one tweet. The method WriteToKML that is part of the Tweet2KML class, however, expects an array of TweetData that can be easily solved by using the CSVParser class (see above).

The only method of Tweets2KML is WriteToKML:

Method Name	Description
public static boolean WriteToKML(String FilePath, String ImageSource, TweetData[] TweetArray)	The method takes the storage location and an array of TweetData elements defined in TweetData.java. A for-loop is used to write down all the tweets extracted by the CSVParser class into a kml file where each tweet is handled as a placemark element. The content of the tweet and its creation time are shown together with its WGS-84 coordinates. The resulting kml file is stored at the user-defined location like the ground-overlay kml file and the response of the WMS-request. For performing all those tasks, a Java.IO.FileWriter object is used (see Img2GroundOverlay.java). After having finished the job, the method returns Boolean "True", if everything was as expected, and "False" if not.

StartGoogleEarth.java

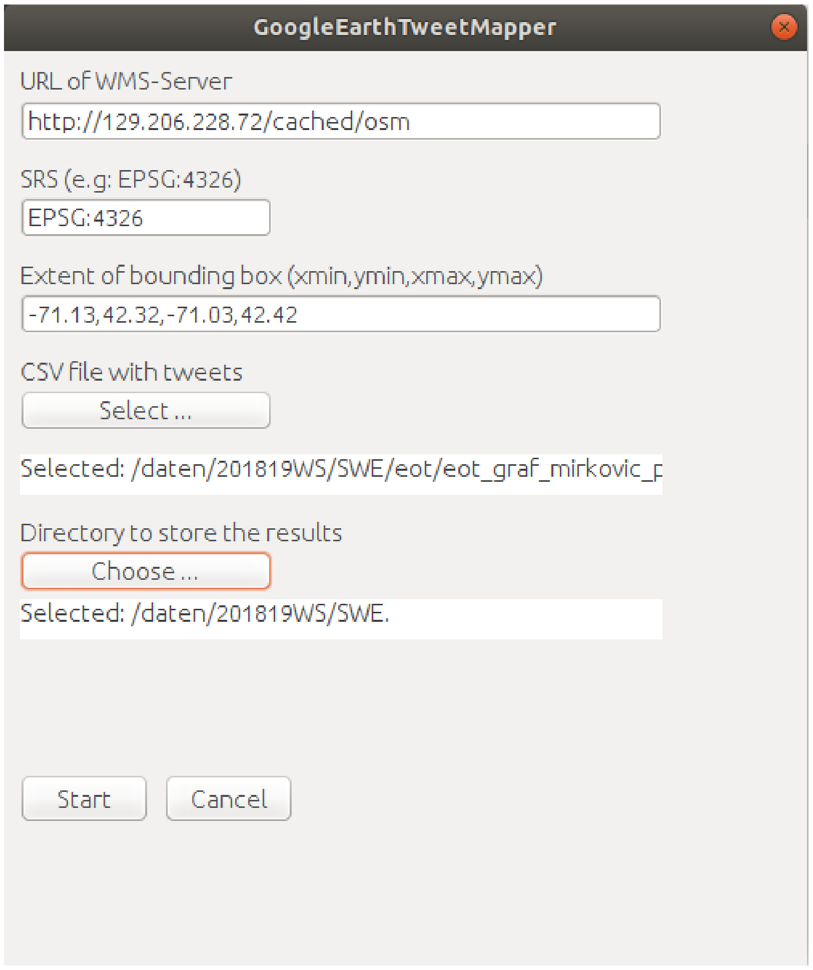
After the two kml files (one for the ground-overlay and one for the tweets) have been created Google Earth can be executed in order to display the content of the two files. Therefore, the install directory of Google Earth needs to be determined and then the program is called using the command line prompt from within this Java class using the Runtime class.

The class has two methods:

Method Name	Description
public static void startUpGoogleEarth (String groundOverlayKML, String singleTweetsKML)	This method starts up the Google Earth binary in order to show the two kml files. The file paths of the two kml files are passed as additional command line arguments to Google Earth, thus, they are opened automatically.
private static String getGoogleEarthBinary()	This method determines the install directory of Google Earth in case of a Windows operating systems and the name of the executable binary (without path to install directory) in case of Ubuntu (Unix systems).

User Manual

When running the program, the user is asked to make some inputs via a graphical user interface (GUI). The structure of the GUI is shown in Figure 2. As visible in Figure 2 the user firstly must insert the URL of the WMS-Server. Secondly, the spatial reference system must be provided as EPSG-Code according to the example in Figure 2 (e.g. EPSG:4326 for WGS-84). The reference system should be the same as for the twitter data to be displayed in Google Earth as otherwise geolocation errors might occur. Next, a bounding box is required. The bounding box should be inserted as xmin, ymin, xmax, ymax with commas between the single values (see Figure 2). Finally, the CSV file containing the twitter data must be selected as well as one directory for storing the resulting rendered map as png image file and the kml-data. Therefore, a file chooser is used to allow comfortable searching for the file. For the storage location just the directory must be provided but no single filenames as the program applies its own internal file naming conventions.


GoogleEarthTweetMapper

URL of WMS-Server: http://129.206.228.72/cached/osm

SRS (e.g: EPSG:4326): EPSG:4326

Extent of bounding box (xmin,ymin,xmax,ymax): -71.13,42.32,-71.03,42.42

CSV file with tweets

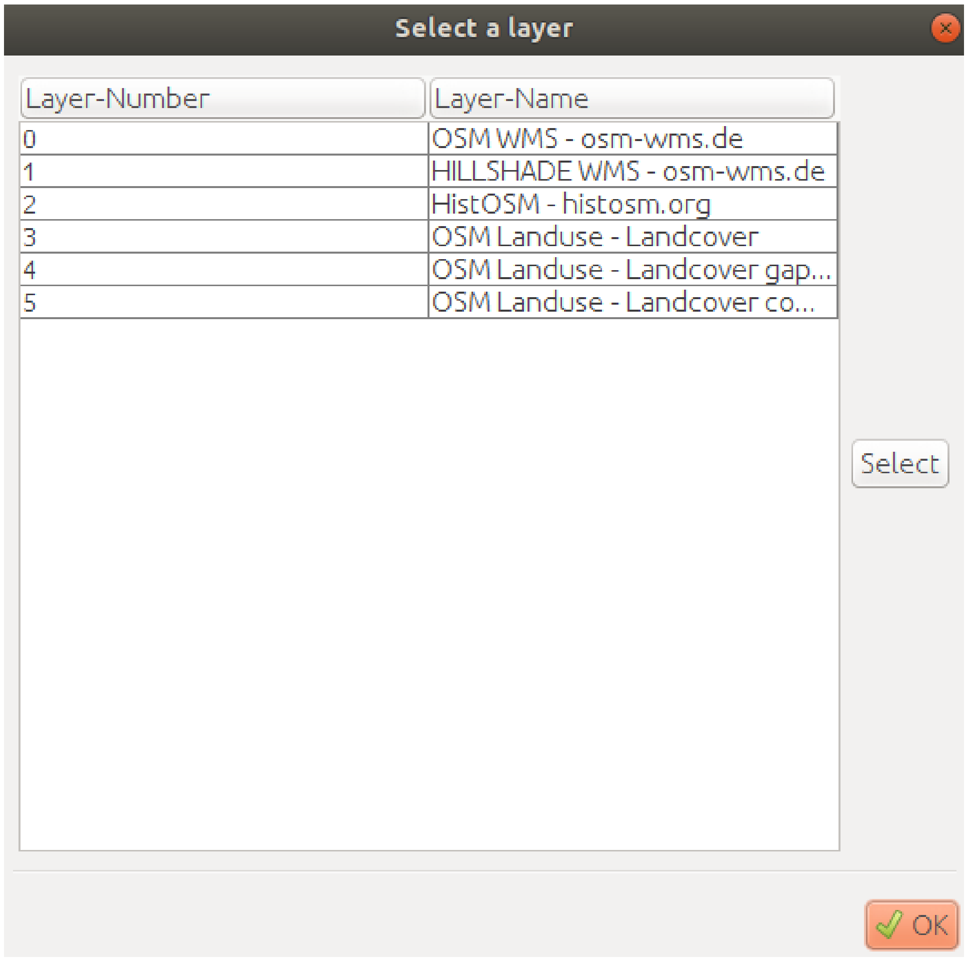
Selected: /daten/201819WS/SWE/eot/eot_graf_mirkovic_p

Directory to store the results

Selected: /daten/201819WS/SWE.

FIGURE 2: GUI OF GOOGLEEARTHTWEETMAPPER SHOWING EXAMPLES OF USER-INPUTS REQUIRED FOR RUNNING THE PROGRAM.

After pressing “Start” the program connects to the given WMS Server. Using the “GetCapabilities” request a list of available layers is shown to the user in a second window that pops up. The layer-number (just used internally) as well as the layer name is shown as displayed in Figure 3. The user is now required to graphically select one layer by clicking the desired row (see Figure 3) and press “Select”.



Layer-Number	Layer-Name
0	OSM WMS - osm-wms.de
1	HILLSHADE WMS - osm-wms.de
2	HistOSM - histosm.org
3	OSM Landuse - Landcover
4	OSM Landuse - Landcover gap...
5	OSM Landuse - Landcover co...

FIGURE 3: WINDOW FOR SELECTING A DESIRED LAYER FROM THE SPECIFIED WMS SERVER.

Subsequently, the program will inform the user about all the following step via little message dialogs. Thus, it is ensured that the user is informed about the program's progress or potential errors occurring. Finally, Google-Earth will be opened and two kml files created during the program execution will be shown (see Figure 4).

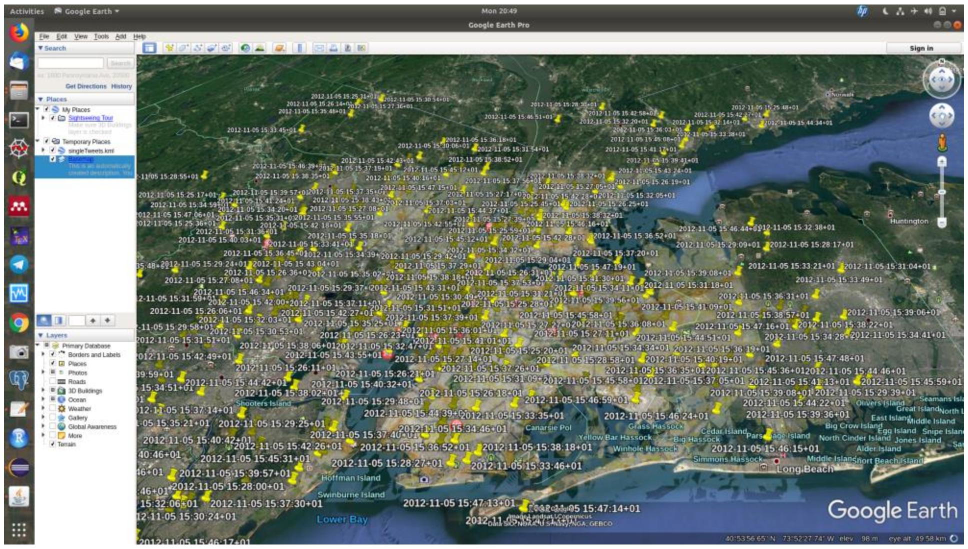


FIGURE 4: AFTER THE PROGRAM EXECUTION IS FINISHED GOOGLE-EARTH WILL OPEN DISPLAYING THE GROUND OVERLAY AS WELL AS THE TWEET FILES.

References

Parts of the functionality of GoogleEarthTweetMapper are archived by using external libraries:

- GeoTools (Version 18.4), http://geotools.org/about.html, for WMS-request handling and processing
- Google Earth Pro (Version 7.3.2.5495), https://www.google.com/intl/de/earth/, for displaying the kml files. KML is an OGC-compliant file format developed and maintained for and by Google Earth.

The whole program is coded in Java 8 using Java(TM) SE Runtime Environment (build 1.8.0_171-b11) and Java Software Development Kit 11 maintained by Oracle (https://www.oracle.com/technetwork/java/javase/overview/index.html).

Please note that starting Google-Earth from Java was tested only under Ubuntu even we believe that the specifications made for Windows should work as well.