Automated Drain Area Creation for Municipal Hydraulic Simulations

8. May 2018
GeoPython Conference 2018
Robin Dainton
Presentation Structure

- Drain Area Planning
- Input Data and Official Models
- System Design
- Workflow Examples and Techniques
- Summary
Drain Area Generator
Drain Area Generator

Drain Area Planning

11.01.2018 | Folie 5
Network Data

Legend

Node Function
- **Primary Node**
- **Secondary Node**
- **Gulley**
- **Primary Watercourse Discharge**
- **Secondary Watercourse Discharge**
- **Seepage Installation**

Sewer Utilisation
- **Primary Foulwater Sewer**
- **Primary Combined Sewer**
- **Primary Rainwater Sewer**
- **Secondary Foulwater Sewer**
- **Secondary Rainwater Sewer**
Drain Area Generator

Drain Area Model
Influence Diagram
Drain Area Generator

Topological Sorting

```python
from toposort import toposort

# Create dependency graph
dependencies = dict()
dependencies['Perimeter'] = ['Land Parcel', 'Building Zone']
dependencies['Built upon'] = ['Perimeter', 'Ground Cover', 'Building Entrance', 'Sewer Data']
dependencies['Impervious Total'] = ['Perimeter', 'Ground Cover']
dependencies['Primary Node Foul'] = ['Perimeter', 'DEM', 'Building Entrance', 'Ground Cover']
# Etc...

# Topological Sort
execution_order = list(toposort(dependencies))

print('Execution Order:
%s

' % execution_order)

""" Output:
Execution Order:
[{'Land Parcel', 'Sewer Data', 'Seepage Installation', 'Built upon', 'Watercourses', 'Build
{Perimeter'},
{'Inhabitant Density (future)', 'Primary Node Foul', 'Impervious Total', 'Primary Node Rain'
{'Primary Node Foul (future)', 'Retention', 'Seepage', 'Primary Node Rain (future)', 'Direct
{'Retention (future)', 'System Type', 'Seepage (future)', 'Direct Water Discharge (future)'}
{'Impervious Rainsystem', 'Impervious Foulssystem', 'System Type (future)'}],
{'Impervious Foulssystem (future)', 'Befestigungsgrad Rain (future)'}]"
```
Workflow
Drain Area Generator

System Architecture

1. Establish Database Connection
   \[db = Interface.PostGISDatabase()\]

2. Prepare Sewerage Network
   \[networker = drain_network.Networker(db)\]
   \[networker.do_everything()\]

3. Create drain areas perimeters
   \[perimeter_creator = drain_perimeter.PerimeterCreator(db)\]
   \[perimeter_creator.do_everything()\]

4. Characterize drain areas
   \[characterizer = drain_character.Characterizer(db)\]
   \[characterizer.do_everything()\]

5. Modify and characterize road drain areas
   \[road_drain = road_drainer.RoadDrainer(db)\]
   \[road_drain.do_everything()\]
Python Modules

- **data_loader**: import network and open data
- **repair_network**: network topology, establish primary network for simulation
- **drain_network**: establish connectivity of secondary drain network
- **drain_perimeter**: create drain area geometry and identify roads
- **drain_character**: establish connection points and drain area attributes
- **road_drainer**: create road drain areas from inlet points
Secondary Network Characterisation
Perimeter
Drain Area Characterisation
Use of Building Geometry
Drain Area Generator

Roads
Drain Area Generator

Roads
Drain Area Generator

Roads
Drain Area Generator

Roads
Drain Area Generator

Roads
Drain Area Generator

Roads
Roads
Drain Area Generator

Use and Documentation

Interfaces Module

Module to obtain and save data from different sources and formats.

**Example OGR Data Importer**

```python
class Interfaces_OgrDataImporter:
    def __init__(self):
        self.data = None

    def import_data(self, file):
        # Import data from file
        self.data = data

    def export_data(self):
        # Export data to file
        with open('output_file.txt', 'w') as f:
            f.write(str(self.data))
```

**Parameter:**
- `file` (str) - File path to the data file

**Example OGR Module**

```python
class OGRModule:
    def __init__(self, data):
        self.data = data

    def process_data(self):
        # Process data
        processed_data = [data for data in self.data]\n
    def save_data(self, path):
        # Save data to file
        with open(path, 'w') as f:
            f.write(str(processed_data))
```

**Parameter:**
- `path` (str) - Path to save the processed data file

11.01.2018 | Folie 29