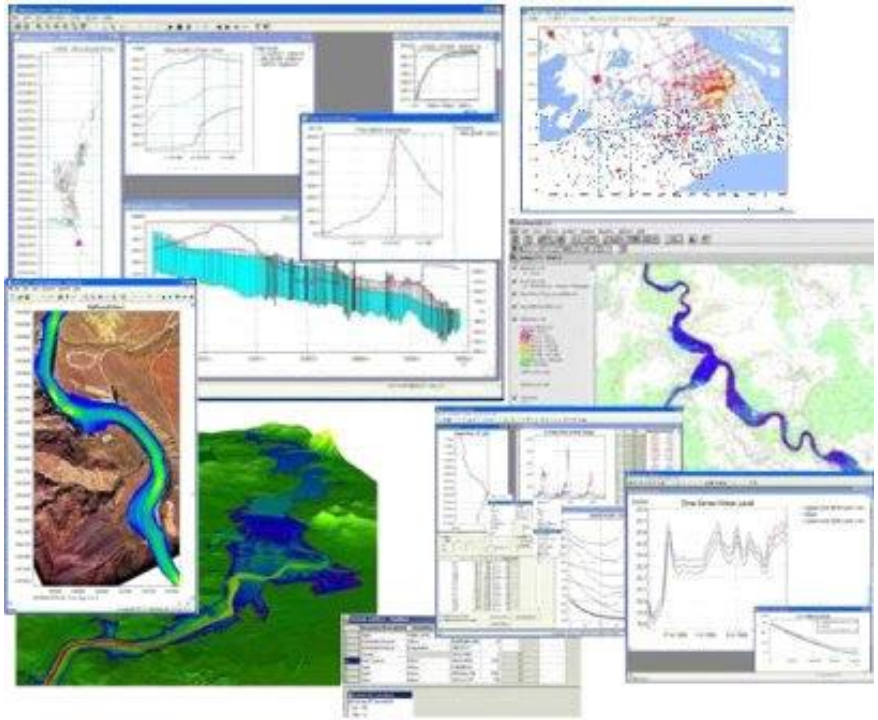
# River Modelling



## 2D Model River Rhine

# Post-Processing

## Result Analysis Telemac2D

# Result Post-Processing

## **Result Analysis / Visualization Options** (Examples)

3D view (2D space, 1D time)

| | |
|---|---|
| 2D view -> | selection of a time step (1D time)<br>2D map of scalar/vector values |
| 2D view -> | selection of a line (1D space)<br>profile (1D) animation (1D) of scalar/vector values |
| 1D view -> | selection of a point (2D space)<br>time series with scalar/vector values |
| 1D view -> | selection of a line (1D space) and a time step (1D time)<br>profile |
| 0D view -> | selection of a point (2D space) and a time step (1D time)<br>scalar/vector value |

…

mixture of **time series analysis** and **GIS functionality**

# Result Post-Processing

## Result Analysis / Visualization Options (Examples)

3D view (2D space, 1D time)

| | |
|---|---|
| 2D view -> | integration over time step (1D time)<br>2D map of scalar/vector values |
| 2D view -> | integration along line (1D space)<br>profile (1D) animation (1D) of scalar/vector values |
| 1D view -> | integration of a polygonal region (2D space)<br>time series with scalar/vector values |
| 1D view -> | integration vertical to a line (1D space) and over time (1D time)<br>profile |
| 0D view -> | integration over time and space<br>scalar/vector value |

…

mixture of **time series analysis** and **GIS functionality**

# Result Post-Processing

## Tools

- Blue Kenue
- **self-developed tools** e.g. using **R**, Matlab, Python, Java, C, C++, …
- **GIS interfaces** (shapefiles, KML)
- **PostTelemac plugin QGIS**
- Tecplot/Paraview interfaces

# Result Post-Processing

## Telemac2D Result Data File

- binary data, serafin/selafin file format
- 2D in space, 1D in time -> 3 coordinates
- several physical state variables
  (at every relevant spatial entity for every stored time step)
- scalar and vector variables
- water depth, velocity, Froude number, …

-> mass data

example: 24 days, 40133 nodes, 15898 time steps, 7+1 variables
result file size: 20 GB

https://svn.osgeo.org/gdal/trunk/gdal/ogr/ogrsf_frmts/selafin/drv_selafin.html

# Result Post-Processing

## Result File Format Selafin (see Telemac2D User Manual)

- for unclear historical reasons this format is also sometimes called SERAFIN
- binary file
- ‘SERAFIN ‘ for single precision storage, or
  ‘SERAFIND’ for double precision storage.
- double precision storage: for "cleaner" restarts,
  not compatible to all post-processors
- strings: utf-8 encoded.
- record-based (FORTRAN WRITE command: 80 characters - punched cards)
  1-72 data 73-80 comments
  e.g. record with title of the study (72 characters)
  8 characters file format (SERAFIN or SERAFIND)

# Result Post-Processing

## Result File Format Selafin (see Telemac2D User Manual)

- 1 record containing the two integers NBV(1) and NBV(2)
  NBV(1) the number of variables, NBV(2) with the value of 0
- NBV(1) records: names and units of each variable (over 32 characters)
- 1 record containing the integers table IPARAM (10 integers, not all used)
- If IPARAM (3) is not 0: x-coordinate of the origin in the mesh
- If IPARAM (4) is not 0: y-coordinate of the origin in the mesh
- If IPARAM (7) is not 0: the number of planes on the vertical (inprisms.)
- If IPARAM (8) is not 0: the number of boundary points (in parallel).
- If IPARAM (9) is not 0: the number of interface points (in parallel).
- if IPARAM (10) = 1:
  a record containing the computation starting date in 6 integers:
  year, month, day, hour, minute, second

# Result Post-Processing

## Result File Format Selafin (see Telemac2D User Manual)

- 1 record containing the integers NELEM, NPOIN, NDP, 1
  (number of elements, number of points, points per element and the value 1)
- 1 record containing table IKLE
  (integer array of dimension (NDP,NELEM) which is the connectivity table.
  Beware: in TELEMAC-2D, the dimensions of this array are (NELEM,NDP))
- 1 record containing table IPOBO (integer array of dimension NPOIN);
  0 for an internal point, numbering of boundary points
- 1 record containing table X (real array of dimension NPOIN)
- 1 record containing table Y (real array of dimension NPOIN)
- each time step:
  1 record time T (real),
  NBV(1)+NBV(2) records for each results arrays for each variable at time T.

# Result Post-Processing

## Tools

- Blue Kenue
- **self-developed tools using R**, Matlab, Python, Java, C, C++, …
- GIS interfaces (shapefiles, KML)
- PostTelemac plugin QGIS
- Tecplot/Paraview interfaces

# Result Post-Processing

## Reading Binary Files

- creating file_connector:

  connector = **file**(“filename”,) or url(“url”, …) or …

- five elementary functions:

  **open**(connector, mode)  mode=“rb” -> read binary

  **close**(connector)

  **readBin**(connector, type, number, element_size, …)

  writeBin(object, connector, element_size, …)

  **seek**(connector, location, origin)

# Result Post-Processing

## Reading Binary Files

**readBin**(connector, type, number, element_size, …)

reading an integer array from selafin file

```
number = readBin(connector, integer(),1) / 4
array = readBin(connector, integer(), number)
```

reading a float array from selafin file

```
number = readBin(connector, integer(),1) / 4
array = readBin(connector, double(), number, size = 4)
```

reading a char array from selafin file

```
number = readBin(connector, integer(),1)
text = readChar(connector, number)
```

# Result Post-Processing

## Reading Binary Files

**seek**(connector, location, origin)

set the connector to the start of the file

```
seek(file_connector,0,"start")
```

move connector to a specific location

```
seek(file_connector,88,”start”)
```

move connector relative forward or backward

```
seek(connector,”current”,offset)
```

# Result Post-Processing

## Reading Selafin Result Files

### reading meta data of selafin result file

```
seek(file_connector,0,"start")
title <<- read_char_array()
values = read_integer_array();  variable_number <<- values[1];
variable_name <<- vector("integer",variable_number)
for(v in c(1:variable_number)) variable_name[v] <<- read_char_array()
values = read_integer_array();  x_origin <<- values[3]; y_origin <<- values[4];
if(values[10]==1) date <<- read_integer_array()
mesh = read_integer_array();  element_number <<- mesh[1]; node_number <<- mesh[2];
nodes = read_integer_array();
node_types <<- read_integer_array()
x <<- read_float_array()
y <<- read_float_array()
time_steps <<- vector("numeric",0);  t = 0; # time index
while(TRUE)
  { t = t +1
   tarray = read_float_array(); time = tarray[1]
   time_steps <<- append(time_steps, time)
   for(v in c(1:variable_number))seek(file_connector,origin="current",(node_number+2)*4)
  }
```

# Result Post-Processing

## Reading Selafin Result Files

### seek meta data to read result data

```
seekMetaData <- function()
{
 seek(file_connector,0,"start")
 seek(file_connector,88)                                    #title
 seek(file_connector,origin="current",16)                   #2 value array
 seek(file_connector,origin="current",variable_number*(32+8))
 seek(file_connector,origin="current",(10+2)*4)
 if(length(date)==6)seek(file_connector,origin="current",(6+2)*4) # date
 seek(file_connector,origin="current",(4+2)*4) # mesh metadata
 seek(file_connector,origin="current",(element_number*3+2)*4) # mesh metadata
 seek(file_connector,origin="current",(node_number+2)*4) # type
 seek(file_connector,origin="current",(node_number+2)*4) # x
 seek(file_connector,origin="current",(node_number+2)*4) # y
}
```

# Result Post-Processing

## Reading Selafin Result Files

### reading a time series for a node from a selafin result file

```
readNodeTimeVariables <- function(time_index=1, node_index=1)
{array <- vector("list",variable_number)
 seekMetaData()
 tryCatch(
    { t = 0 # time index
      while(TRUE)
      { t = t +1
      tarray = readFloatArray()
      # loop on variables
      for(v in c(1:variable_number))
      { if(t==time_index)
        { v_array = readFloatArray();
          array[v] = v_array[node_index]; }
        else seek(file_connector,origin="current",(node_number+2)*4);}
      if(t==time_index) break;
      }},
    error = function(e) { print(paste("end of time loop",e)); })
 return(array)
```

# Result Post-Processing

## Reading Selafin Result Files

### reading a time series for a node from a selafin result file

```
readTimeSeries <- function(node_index=0, variable_index=0)
{seekMetaData()
 time_series = vector("numeric",time_step_number)
 t = 0 # time index
 while(TRUE)
  { t = t +1
    tarray = readFloatArray()
    time = tarray[1]
    for(v in c(1:variable_number))
     { if(v==variable_index)
        { seek(file_connector,origin="current",(node_index)*4)
          time_series[t] = readBin(file_connector,double(),1,size=4, endian = "big")
          seek(file_connector,origin="current",(node_number-node+1)*4) }
        else seek(file_connector,origin="current",(node_number+2)*4)
     }
  }
 return(time_series)
}
```

# Result Post-Processing

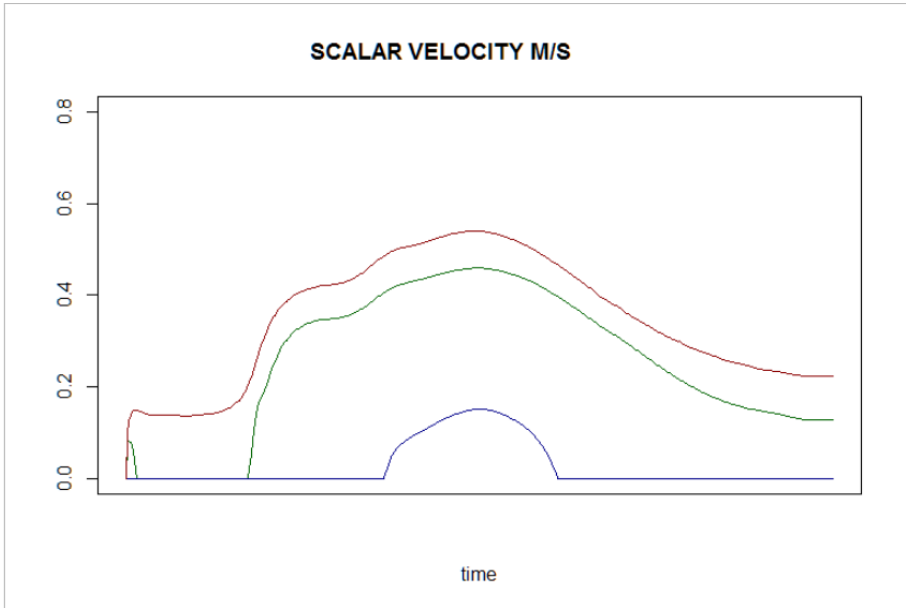
## Reading Selafin Result Files

### time series for several nodes from a selafin result file

```
velocity_index = 8;
openSerafinFile(telemac_file);
readMetaData();

t1 = readTimeSeries(1200, velocity_index);
t2 = readTimeSeries(1800, velocity_index);
t3 = readTimeSeries(4200, velocity_index);

plot(time,t1);
lines(time,t2);
lines(time,t3);
```

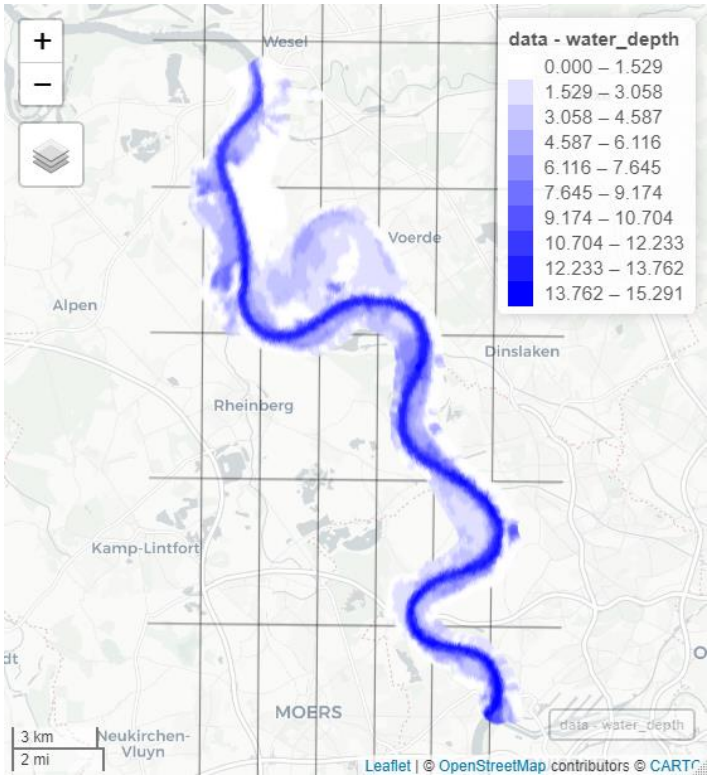


# Result Post-Processing

## Reading Selafin Result Files

### map a variable for a specific time read from a selafin result file

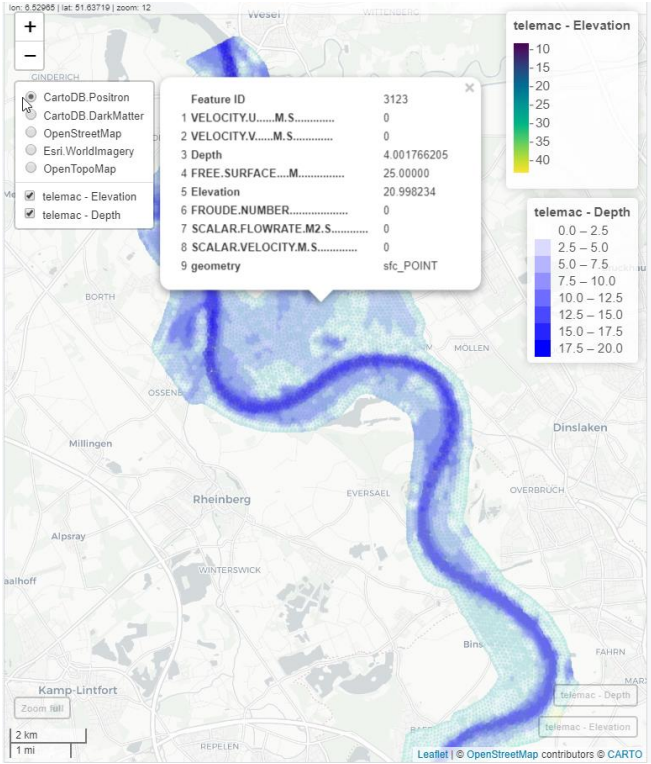
```
depth_index = 3;
openSerafinFile(telemac_file);
time_index = 100:
a = readAllVariables(time_index)
p = data.frame(x,y,a[depth_index])
hmax = max(a[depth_index])
colnames(p)[3] = "water_depth"
pal = colorRampPalette(c("white","blue"))
mapView(p,xcol="x",ycol="y",zcol="water_depth",
crs="EPSG:31466",col.regions = pal(101),
at = seq(0., hmax, ((hmax)/10.)),cex=3,lwd=0)
```



# Result Post-Processing

## Using Standard Libraries / packages

### interactive map




```
library(rgdal)
# file to read
telemac_file = "Results_Unsteady_industry.res"
# read the file
telemac = readOGR(telemac_file)
# first plot
plot(telemac)
# first interactive mapview
mapview(telemac)
# improved interactive mapview
telemac@proj4string = CRS("EPSG:31466")
colnames(telemac@data)[3] <- "Depth"
colnames(telemac@data)[5] <- "Elevation"
mapview(telemac, zcol="Elevation",cex=3,lwd=0)
# improved interactive mapview 2 layers
m1 = mapview(telemac,zcol="Elevation",cex=3,lwd=0,at = seq(5., 50., 5.))
m2 = mapview(telemac,zcol="Depth",cex=3,lwd=0, at = seq(0., 20., 2.5), col.regions =
colorRampPalette(c("white","blue")))
m1+m2
```

# Result Post-Processing

## Tools

- Blue Kenue
- self-developed tools using R, Matlab, Python, Java, C, C++, …
- **GIS interfaces (shapefiles, KML)**
- PostTelemac plugin QGIS
- Tecplot/Paraview interfaces

# Result Post-Processing

## Exporting ESRI Shapefiles or KML (GDAL Driver)

R implementation:

```
variables = readAllVariables(time_index);
gispoints = SpatialPointsDataFrame(data.frame(x,y),variables,
                                   proj4string=CRS("EPSG:31466"))
writeOGR(gispoints,sprintf("telemac_nodes_%d.shp",as.integer(Sys.time())),
         layer="telemac_nodes",driver="ESRI Shapefile");
```

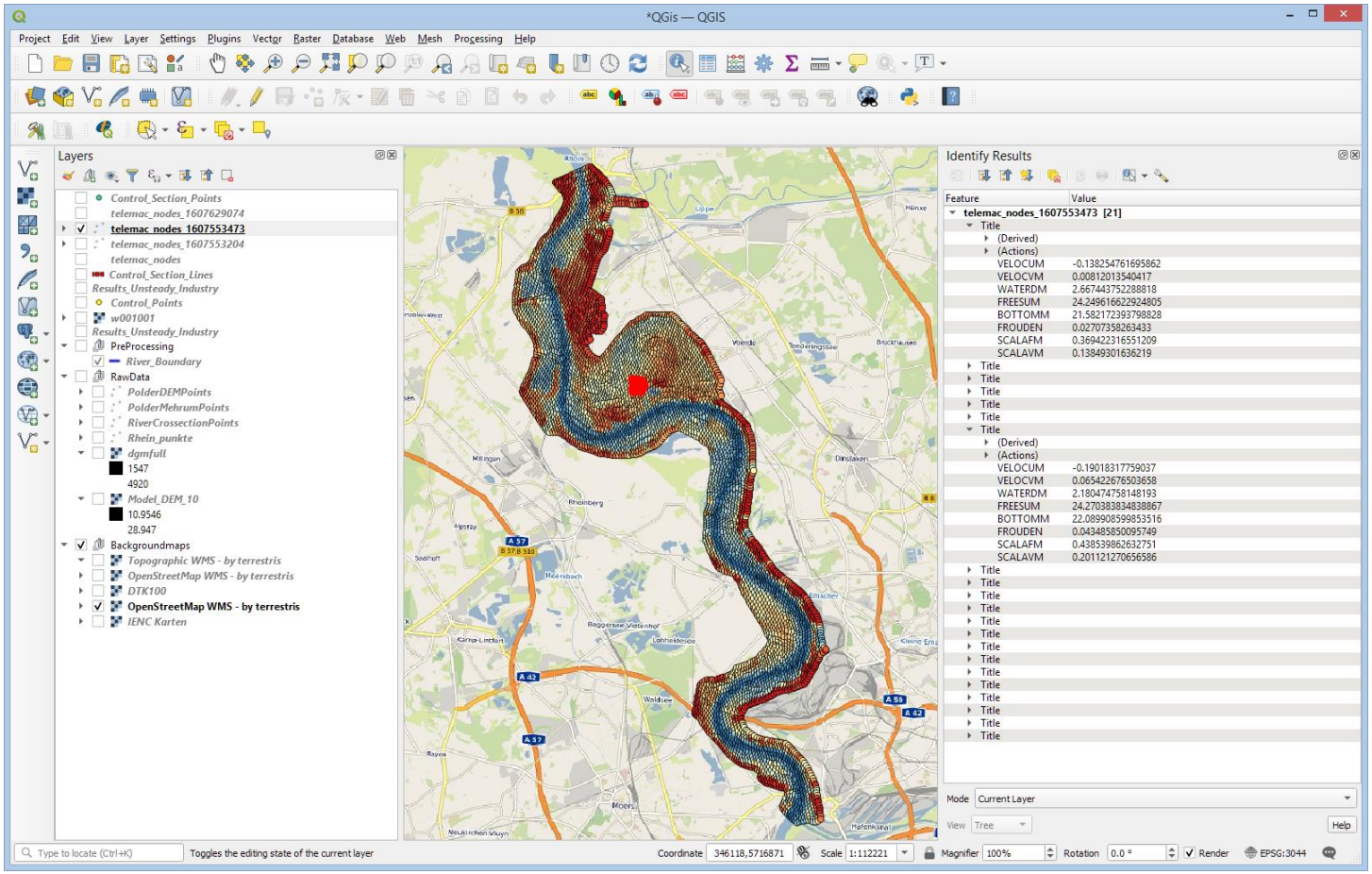
or simplified:

```
telemac = readOGR(telemac_file)
telemac@proj4string = CRS("EPSG:31466")
telemac <- spTransform(telemac, CRS("+proj=longlat +datum=WGS84"))

writeOGR(telemac,dsn="telemac.kml",layer="telemac",driver="KML")
writeOGR(telemac,sprintf("telemac_nodes_%d.shp",as.integer(Sys.time())),
         layer="telemac_nodes",driver="ESRI Shapefile");
```

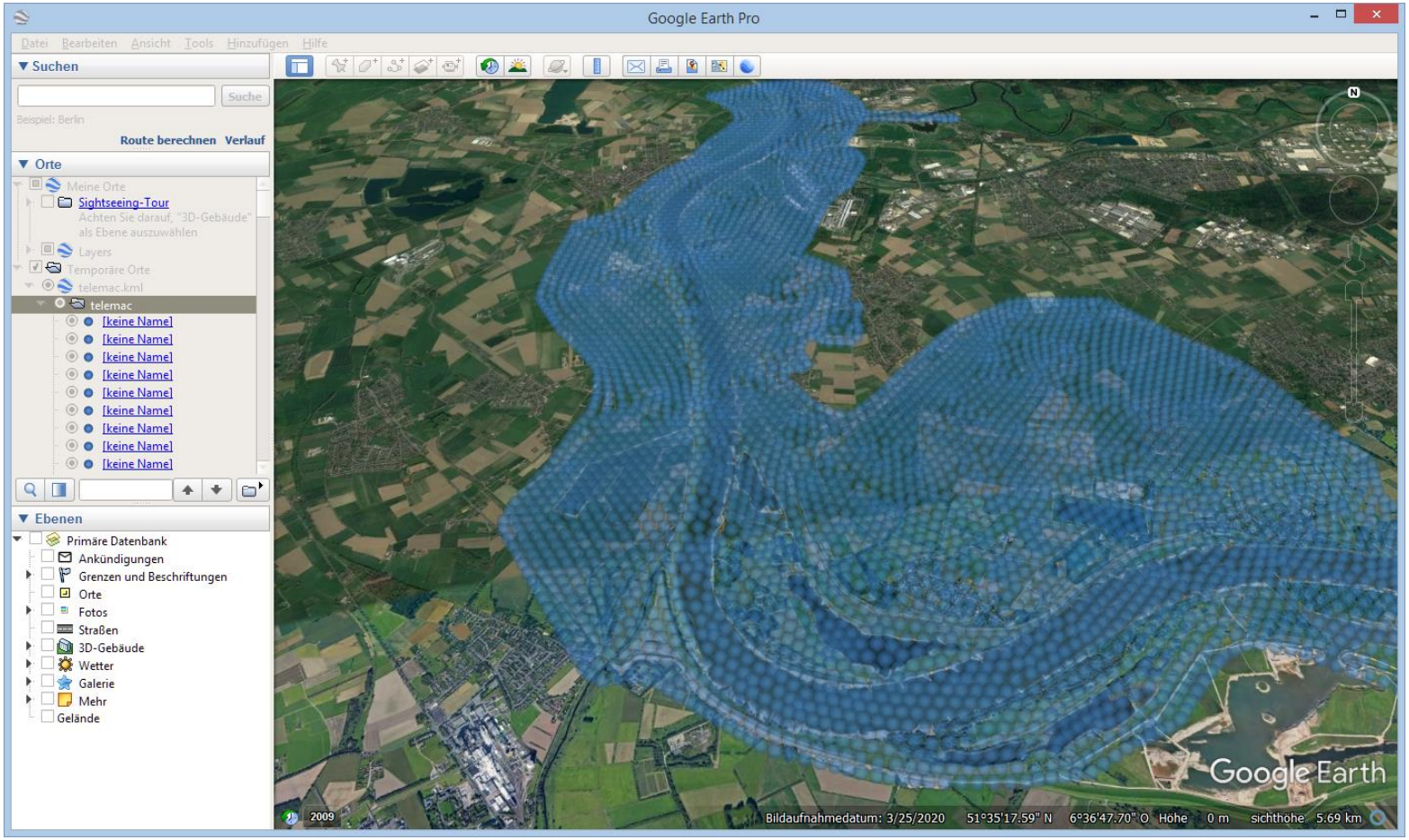
# Result Post-Processing

## Exporting ESRI Shapefiles



# Result Post-Processing

## Exporting KML (GDAL Driver)



# Result Post-Processing

## Tools

- Blue Kenue
- self-developed tools using R, Matlab, Python, Java, C, C++, …
- GIS interfaces (shapefiles, KML)
- **PostTelemac plugin QGIS**
- Tecplot/Paraview interfaces

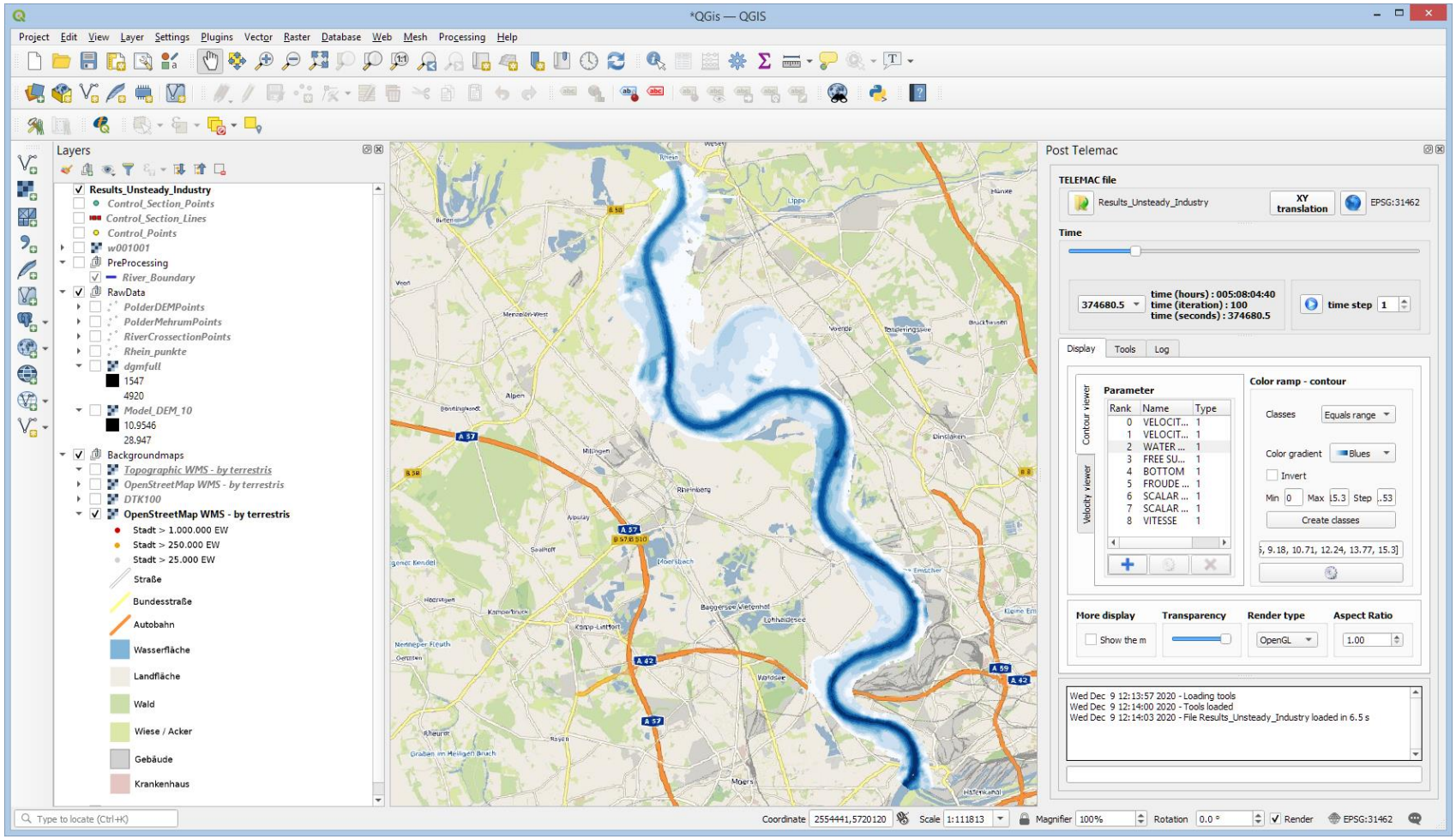
# Result Post-Processing

## Result Analysis using GIS

- GIS is useful for geospatial related visualization
- pre-proccessing and visualization of input data
- selafin result file contains geospatial information
- problem: GIS is not supporting selafin files
- solution: a) reading selafin file with R exporting to shape file
  b) using a plugin to read selafin files within GIS
  **PostTelemac plugin QGIS**

# Result Post-Processing

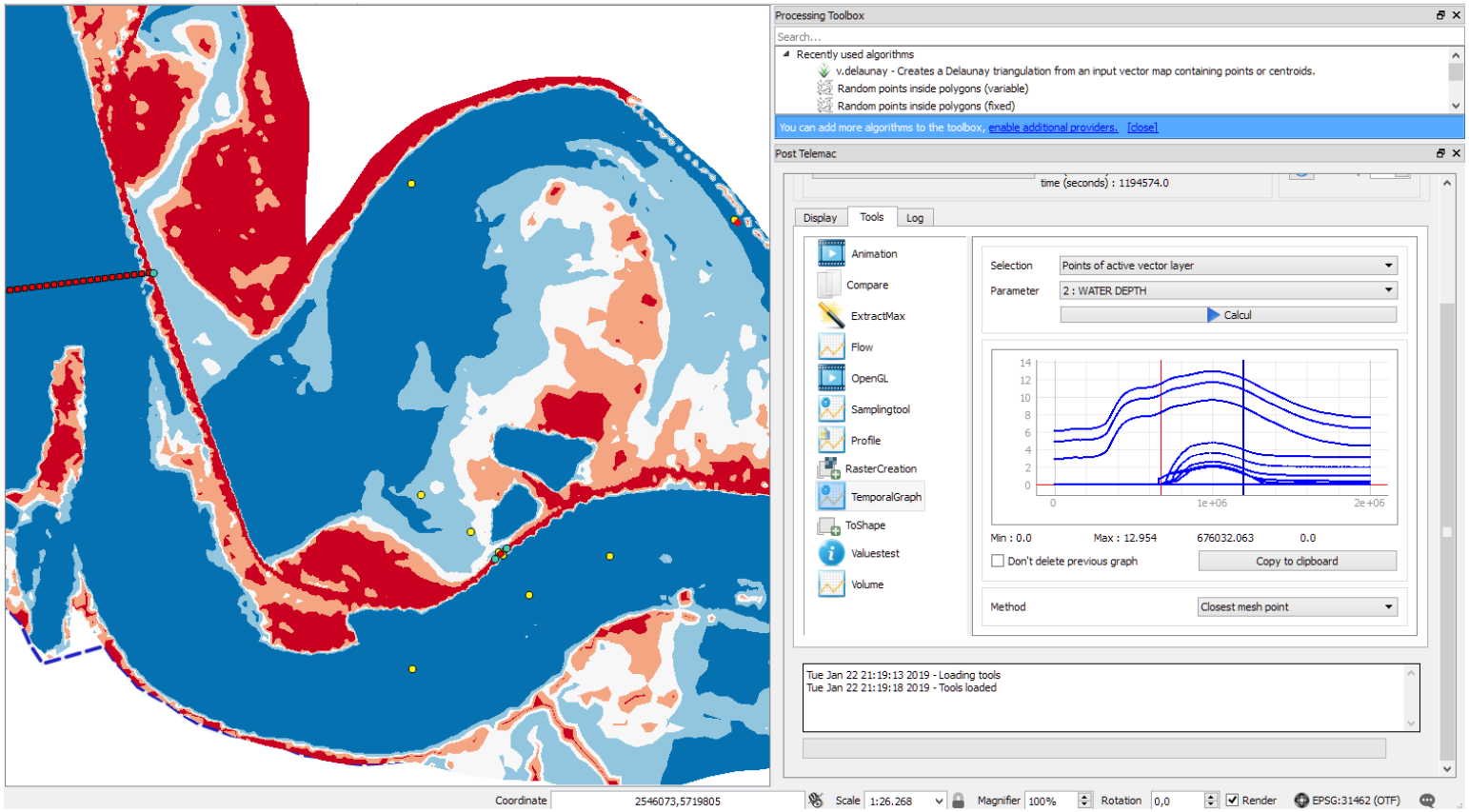
## Result Analysis using PostTelemac plugin QGIS



# Result Post-Processing

## Result Analysis / Visualization Options (Examples)

1D view -> selection of a point (2D space)

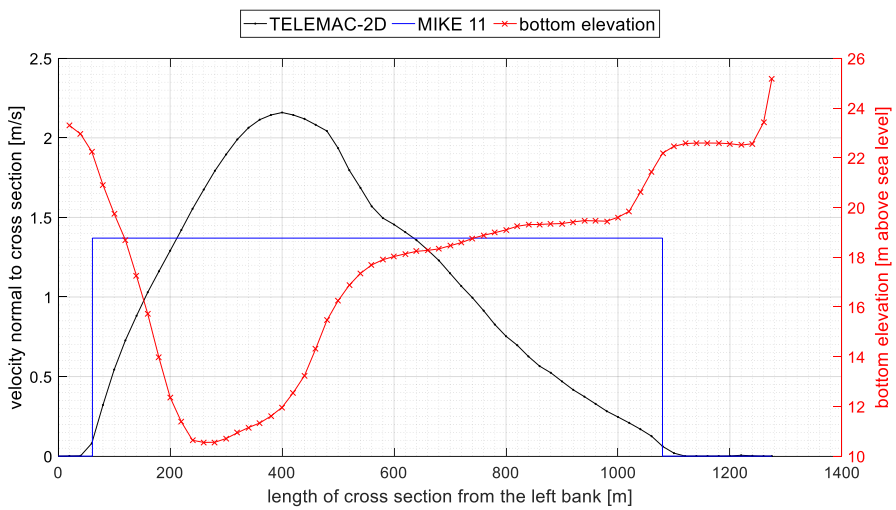


# Result Post-Processing

## Result Analysis / Visualization Options (Examples)

1D view -> selection of a line (1D space)
selection of a time step (1D time)

example cross section




example longitudal profile