Dissertation

Project report

3 Ba INF 2018-2019

ACED

This document contains an overview of all the work done for the dissertation and where to find the relevant changes.

1 Dataformats

1.1 HDF5 and JSON

Support for both of these formats has been added. The changes are as follows:

- `The libraries have been added in main/resources/lib`
- `Both are linked in CMakeCPP.cmake`
- `HDF5 is also linked separately in main/cpp/CMakeLists.txt`

1.2 GeoGrid

GeoGrids can be written to and read from files. Supported formats are JSON, HDF5 and Protobuf. Documentation is provided using Doxygen, and through the user manual. The involved files are:

- `main/cpp/geopop/io/GeoGridWriter.h`
- `main/cpp/geopop/io/GeoGridFileWriter.h`
- `main/cpp/geopop/io/GeoGridStreamWriter.h`
- `main/cpp/geopop/io/GeoGridWriterFactory [.h / .cpp]`
- `main/cpp/geopop/io/GeoGridHDF5Writer [.h / .cpp]`
- `main/cpp/geopop/io/GeoGridJSONWriter [.h / .cpp]`
- `main/cpp/geopop/io/GeoGridProtoWriter [.h / .cpp]`

The files concerning the readers are named analogously (where 'Writer' is replaced with 'Reader'). The files involved with testing this functionality are:

- test/cpp/gtester/geopop/io (Appropriately named tests can be found here)
- test/cpp/gtester/geopop/io/GeoGridCompare [.h / .cpp]

1.3 Household

Households can be read from JSON and CSV files. Documentation is provided using Doxygen. Changed files are:

- main/cpp/geopop/io/HouseholdJSONReader [.h / .cpp]
- main/cpp/geopop/io/ReaderFactory [.h / .cpp]
- test/cpp/gtester/geopop/io/HouseHoldJSONReaderTest.cpp

2 Daycare and preschool

Two new ContactTypes have been added. Documentation is available in the user manual. The involved files are:

- main/cpp/geopop/generators/DaycareGenerator.cpp
- main/cpp/geopop/populators/PreschoolPopulator.cpp
- main/cpp/GeoGridConfig [.h / .cpp]
- main/cpp/geopop/io/proto/geogrid.proto
- main/cpp/geopop/io/proto_pb/geogrid_pb [.h / .cpp]
- main/cpp/contact/AgeContactProfile.cpp
- main/cpp/contact/ContactType [.h / .cpp]
- main/cpp/contact/Infector.cpp
- main/cpp/pop/DefaultPopBuilder.cpp
- main/cpp/pop/GeoPopBuilder.cpp
- main/cpp/pop/Person [.h / .cpp]
- main/cpp/pop/Population [.h / .cpp]
- main/cpp/pop/SurveySeeder.cpp
- test/cpp/gtester (in the *generators* and *populators* subdirectories, tests can be found

3 Visualiser

A separate executable has been built in stride to allow the visualisation of epidemiological data. The user manual contains a section on this functionality.

3.1 Building and linking QT5

QT5 is added to the project with CMake by using the CMake utilities provided by QT5. The QT5-buildtool qmake was not used in the project. The following files are relevant to the QT5 build process:

- `CMakeCPP.cmake`
- `main/cpp/CMakeLists.txt`

3.2 Displaying locations on the Map

Locations can be displayed on a QML `Map` through the use of `MapCircle` objects. These circles are colored by using the `Qt.hsva()` function and are made transparent by taking `a = 0.5`. The size of the circles is determined in a non-linear way based on the population of the location. The following files are relevant for this functionality:

- `main/cpp/visualiser/view/main.qml`
- `main/cpp/visualiser/view/location_marker.qml`
- `main/cpp/visualiser/view/logic.js`

3.3 Location statistics in the sidebar

The locations that are displayed on the map are clickable. The click event is captured and used to update the sidebar with statistics about the location. The following files are relevant:

- `main/cpp/visualiser/view/main.qml`
- `main/cpp/visualiser/view/location_marker.qml`

3.4 Timestep selection

Selecting the timestep that is to be displayed can be done by using the slider in the toolbar and by using the autoplay functionality. Both the slider and the autoplay functionality are present in `main/cpp/visualiser/view/main.qml`.

3.5 Exporting to image

The functionality to export the map to a PNG file is present in the following files:

- `main/cpp/visualiser/view/main.qml`
- `main/cpp/visualiser/controller/Controller.[h,cpp]`

3.6 Model-View-Controller

The C++ backend for the visualiser is structured as MVC. The following files are relevant:

- `main/cpp/visualiser/controller/Controller.[h,cpp]`
- `main/cpp/visualiser/model/Model.[h,cpp]`
- `main/cpp/visualiser/view/View.[h,cpp]`
- `main/cpp/visualiser/view/PopStatsView.[h,cpp]`

3.7 Selecting areas on the map

The visualisation tool also comes with both rectangle and radial selection functionalities. The following files are relevant to the selection functionality:

- `main/cpp/visualiser/view/main.qml`
- `main/cpp/visualiser/view/selection_dialog.qml`
- `main/cpp/visualiser/controller/Controller.[h,cpp]`
- `main/cpp/visualiser/model/Model.[h,cpp]`
- `main/cpp/visualiser/view/View.[h,cpp]`

3.8 Epi output reader

The visualiser supports three file formats for reading epidemiological data: JSON, HDF5 and ProtoBuf. The following files are relevant in providing this functionality:

- `main/cpp/visualiser/view/main.qml`
- `main/cpp/visualiser/controller/Controller.[h,cpp]`
- `main/cpp/visualiser/readers/JSONEpiReader.[h,cpp]`
- `main/cpp/visualiser/readers/HDF5EpiReader.[h,cpp]`
- `main/cpp/visualiser/readers/ProtobufEpiReader.[h,cpp]`
- `main/cpp/visualiser/readers/EpiReader.[h,cpp]`

3.9 Tests

The requirements stated that the GUI does not need to be tested. However, we decided to test the file readers and the `PopStatsViewer` class. The following files are relevant:

- `test/cpp/gtester/visualiser/TestEpiOutputHDF5Reader.cpp`
- `test/cpp/gtester/visualiser/TestEpiOutputJSONReader.cpp`
- `test/cpp/gtester/visualiser/TestEpiOutputProtobufReader.cpp`
- `test/cpp/gtester/visualiser/TestPopDataView.cpp`
- `test/cpp/gtester/visualiser/ModelPopDataChecker.cpp`
- `test/cpp/gtester/visualiser/ModelTimestepOrderChecker.cpp`
- `test/cpp/gtester/visualiser/VisualiserTestfileGetter.h`

3.10 Epi output writer

Epioutput files can be generated by stride to serve as input for the visualiser. This is also documented in the manual. The files involved in the generation are:

- `main/cpp/execs/ControlHelper.cpp`
- `main/cpp/viewers/EpiOutputFileViewer.[.h,cpp]`
- `main/cpp/viewers/EpiOutputFile.[h,cpp]`
- `main/cpp/viewers/EpiOutputHDF5.[h,cpp]`
- `main/cpp/viewers/EpiOutputJSON.[h,cpp]`
- `main/cpp/viewers/EpiOutputProto.[h,cpp]`

4 Geogrid & Location split

The `GeoGrid` and `Location` classes were split in order to separate the simulation layer and geographical layer. The `Location` class was split using inheritance instead of templates due to it being conceptually more correct, easier and safer. The `Location` was split into the `LocationBase`, `SimLocation` and `VisLocation` classes. The `GeoGrid` class was split into the `GeoGridBase`, `GeoGrid` and `VisGeoGrid` classes.

The following files are relevant to this refactoring:

- `main/cpp/geopop/GeoGridBase.[h,cpp]`
- `main/cpp/geopop/GeoGrid.[h,cpp]`
- `main/cpp/geopop/VisGeoGrid.[h,cpp]`
- `main/cpp/geopop/LocationBase.[h,cpp]`
- `main/cpp/geopop/SimLocation.[h,cpp]`
- `main/cpp/geopop/VisLocation.[h,cpp]`
- `main/cpp/geopop/PopStats.[h,cpp]`

5 Workplace size distribution

The simulation has been modified to account for different sizes of workplaces. This is described in the manual. Files involved are:

- `main/cpp/geopop/generators/WorkplaceGenerator.cpp` (and analogous populator)
- `main/cpp/geopop/io/WorkplaceReader.h`
- `test/cpp/gtester (contains appropriate tests)`

6 Demographic profiles

The simulation will be modified to account for different demographic profiles. This will be explained in the manual. Files involved are:

- `main/cpp/geopop/GeoGridConfig [.h / .cpp]`
- `main/cpp/geopop/populators/HouseholdPopulator.cpp`
- `test/cpp/gtester/geopop/io/HouseholdConfigTest.cpp`
- `test/cpp/gtester/geopop/populators/HouseholdPopulatorTest.cpp`