Event reader: Reading Event from json and xml file format into network simulation

Pham Nguyen Quang

April 16, 2017

1 Introduction

1.1 Context

Event reader sub-project is part of the internship of the project from Bui Quang Minh, Pham Nguyen Quang, Nguyen Chi Thanh and Vo Huynh Quang Kiet at Frankfrut university of applied sciences. This project is about building a generic network optimizing system using evolutionary algorithm (EA). The event reader project enables the users to read the event list from the json or xml file format as an event input for the simulation without the restriction on the number of events that can be read from the file. The event-reader sub-project is an extension of the graph reader project that was documented earlier.

1.2 Requirements

- Programming Language: C++11
- Operating system: Linux 64 bit
- Compiler: Linux g++
- Dependencies: Tinyxml2 and jsoncpp library
- Recommended IDE: Eclipse IDE for C++, version Neon.2
  - To install Tinyxml2, open the terminal in linux and type:`sudo apt-get install libtinyxml2-dev`
  - To install jsoncpp, open the terminal in linux and type:`sudo apt-get install libjsoncpp-dev`

2 Structure

All header files (.h) and source code file (.cpp) for event readers are placed in the src/Event folder. The program reads events files in json or xml as the input and return a list of events as output, the class for list of events is declared in AnyEventList.h and its functions are defined in AnyEventList.cpp. The class for the event itself is declared in AnyEvent.h,its functions are defined in AnyEvent.cpp, AnyEvent encapsulates the attributes which is defined as Attributes.h in /src/Attributes. The attributes functionalities is defined in Attributes.cpp

3 Technical Implementation detail

3.1 Event list implementation

Event list is a list that contains individual events as an array. The event list class consists of the following functions and variables, as declared in header (AnyEvent.h) file.

- Event list (`vector <AnyEvent >event_list`): A list consists of individual events, implemented as a vector for the flexibility of the number of events, this variable is private.
- Constructor(`AnyEventList()`): The default constructor for the class.
- Get function (`AnyEvent get_event_at(int index)`): Get an event at specific index, return the object of type AnyEvent.
- Get function (`int get_number_of_event()`): Get total number of events that contained in the event list object;
- Add function (`void add_event(AnyEvent event)`: Allow the user to add an extra event of the type AnyEvent (if there are any).
- Read file function (`void read_json_event(string jsonpath)`): This function allow the user to read an event file in json format. All the event that are present will be added to the event list.
- Read file function (`void read_xml_event(string xmlpath)`): This function allow the user to read an event file in xml format. All the event that are present will be added to the event list.

3.2 Event Implementation

The event in this context is represented by the class AnyEvent. This class consist of the followiing variables and functions as written in header file (AnyEvent.h).

- Event attribute list (`vector <Attributes >event_attributes`): This is the attributes list for a single event, the attributes here are written according to the documentation on graph and event writing that was documented. The Attributes type that is used here is the same as the Attributes type that is being used for reading the graph (see below). However, for reading event, we only use attributes name and attribute data for event reading purpose, as the type of data will be string by default (see documentation on writing event and graph). The attributes represent what are in the events and "what will happen in the event". Event id (Event name) is excluded from the attributes.
- Event ID (`string event_id`): This represent the event ID, the name of the event will be written here.
- Constructor (`AnyEvent()`): This is a default constructor.
- Constructor (`AnyEvent(string id)`): This is a constructor to construct the event which the ID (event name) is known.
- Get function (`vector<Attributes >get_attributes_list()`): Get the attributes list of the event.
- Get function (`int get_attribute_size()`): Get the total number of attributes of the event.

- Get function (`string get_attribute_name_at(int index)`): Get the attribute name at the specific index location, this returns the attribute name with the type of string.
- Get function (`string get_attribute_data_at(int index)`): Get the attribute data at the specific index location, this returns the attribute data with the type of string.
- Add function (`void add_attribute(Attributes attr)`): Add a specific attribute of type Attributes to the event (if any).

3.3 Attribute implementation

The implementation for the attributes are based on how the attributes are written in thel files. The Attributes class declaration and definition can be found in Attributes.h and Attributes.cpp, the class contains:

- Attribute name (`string attribute_name`): Name of the attribute in string format, this is private variable.
- Attribute data (`string attribute_data`): Data of the attribute in string format, this is private variable.
- Attribute id (`string attribute_id`): Id of the attribute in string format, this is private variable. This is only relevant for graph reading
- Attribute type (`string attribute_type`): Type of the attribute in string format, this is private variable. This is only relevant for graph reading
- Constructor (`Attributes()`): Default constructor.
- Set function (`void set_name(string name_data)`): Set the attribute name.
- Set function (`void set_data(string data_data)`): Set the attribute data.
- Set function (`void set_id(string id_data)`): Set the attribute ID. This is only relevant for graph reading.
- Set function (`void set_type(string type)`): Set the attribute type. This is only relevant for graph reading.
- Get function (`string get_attribute_name()`): Get the attribute name.
- Get function (`string get_attribute_data()`): Get the attribute data.
- Get function (`string get_attribute_id()`): Get the attribute id.This is only relevant for graph reading.
- Get function (`string get_attribute_type()`): Get the attribute type.This is only relevant for graph reading.

For reading event, only attribute name and data are being used.

3.4 File reading implementation

Event reading utilize tinyxml2 and jsoncpp library in order to read the event file that is in either in xml or json format.

4 Usage

Event reader reads event file in either json or xml format in order to get the event list. To read the file format, first, the object of type AnyEventList must be initialized. Then use the read function according to what type of file is to read. Example:

- Initialize: `AnyEventList event_list list_of_event;`
- Read json file: `list_of_event.read_json_event(jsonpath);`: Where jsonpath is the path to the json file.
- Read xml file: `list_of_event.read_xml_event(xmlpath);`: Where xmlpath is the path to the xml file.
- Add an extra event: `list_of_event.addl_event(event);`
- Get Total number of events: `list_of_event.get_number_of_event();`
- Get attribute name of the first event: `list_of_event.get_attribute_list()[0].get_attribute_name();`
- Get attribute data of the first event: `list_of_event.get_attribute_list()[0].get_attribute_data();`