THE ONE - Case2: Customizing The Movement of Nodes

Xu Liu *and Yuanzhu Chen

Department of Computer Science
Memorial University of Newfoundland

May 11, 2013

1 Introduction

In this report, we give a simple scenario to show some knowledge related to create customized path for each node (create a customized map for node to move). At the same time, we discuss some movement related class in THE ONE simulator. Finally, we talk about how to control the movement pattern of each node in order to adapt the real world situation.

1.1 Outline

This report is consisted of three sections

- Some knowledge related to Well-Known Text file
- Create a customized map for each node to move
- Specify the movement path for each node

2 Well-Known Text File

2.1 Introduction

Well-Known Text File is a kind of markup text file (Like HTML) which is used in THE ONE to decide the map for each node to move. The short name of Well-Known Text file is wkt. It is given by Open Geospatial Consortium (OGC) to render vector geometry objects. In THE ONE simulator, I mainly use the WKT Markup Language – "LineString' or "MultiLineString" to describe a path for node to move. We can find that in THE ONE, it originally contains some wkt files in folder "the_one_1.4.1/data/" to represent the default map. If we use the "default_settings.txt" to run THE ONE. It will load these wkt files "roads","main_roads","pedestrian_paths","shops" from the folder to create the map for nodes. All of these files are linked to each other. If we comment one of them or miss one of them, THE ONE will give us an error and THE ONE can not successfully run.

2.2 Geometry primitives

With Well-Known Text File, people can create many geometry objects. Such as point, line,polygon, circle and so on. In THE ONE, we only focus on line. Because we use lines to demonstrate the path (street, road, avenue and so on) for node (cars, bus, pedestrian) to move. Figure 1 shows a example how to define a single line or multiline. As Figure 1

*liuxu.mun@gmail.com

Type	Examples	
LineString	LINESTRING (30 10, 10 30, 40 40)	
MultiLineString	MULTILINESTRING ((10 10, 20 20, 10 40), (40 40, 30 30, 40 20, 30 10))	

Figure 1: Example of Creating Lines

shows that, the basic grammar in wkt file to define a line is that it starts with the keyword ("LINESTRING" or "MULTILINESTRING") Then use bracket to contain the points coordinate or lines (a set of points coordinate) for representing the object ("LINESTRING" or "MULTILINESTRING"). The first point shows in the bracket will be start point. The original point (0,0) is at the bottom left position. In THE ONE, the unit of the normalization is meter. (10,0) means the point is 10 meters away from the original point. Moreover, we need to pay more attention. In THE ONE simulator, if we want to define our customized wkt file, we need to guarantee that in our personal configuration file, we must keep the path coordinate value in wkt file small or equal to the boundaries value of THE ONE world (the property name in configuration file is "worldSize"). Otherwise, there will be an error thrown out by THE ONE. For example, if the worldSize in THE ONE configuration file is defined to 4500 x 3500, all the X coordinate value in wkt file should be larger than 0 and smaller or equal than 4500. all the Y coordinate value in wkt file should be larger than 0 and smaller or equal than 3500. If we user "MULTILINESTRING", we need to check all these lines should connect to each other. Otherwise, THE ONE can not run based on these coordinate value.

3 Use Customized Map in THE ONE

In this section, we create a scenario and show how we create a customized path for node to move in THE ONE by using wkt file.

3.1 Scenario

In order to make an easy scenario, we plan to use only five nodes in this scenario to walk on a five stars paths as Figure 2 shows (Figure 2 is not exact. The final path depends on the coordinate in Figure 2). Initially, these five nodes need to stay each angle of the five starts (node 1 at point A, node 2 at point D and so on). Then, node 1 to node 4 randomly move while we only force node 5 move as a clockwise order for each angle of the five stars. When node 5 finish a turn of traveling five angles of five stars, then it stop 5 seconds and continue do the same thing with a increasing speed while other nodes are always randomly and non-stop moving. Please pay more attend the red dots in Figure 2 indicates that this point is intersection. Every lines is connected at this point.

3.2 Generate WKT file

After we finish defining our scenario, now, based on the information in Figure 2. We need to generate the wkt file. Because we can draw this path one time without stop, jump and combination. So we select "LINESTRING" to describe the path in wkt file. Following the order and information of "ABCDEADBECA", we can generate a file in "the_one_1.4.1/data/customized.wkt" to save our coordinates. Finally, we can see that our "customized.wkt" file contain only one line and the value of that line is "LINESTRING (0 0, 3000 6000, 6000 0, 0 4000, 6000 4000, 0 0,0 4000,3000 6000, 6000 4000,6000 0, 0 0)". "LINESTRING" is the keyword in wkt file, It means we want to draw a line. Then each point is splited by comma. Furthermore X coordinate and Y coordinate of each point are splited by space. Finally, we can construct our five star path.

The way of writing the wkt file by ourselves is one approach to generate wkt file. Sometimes, calculating the value

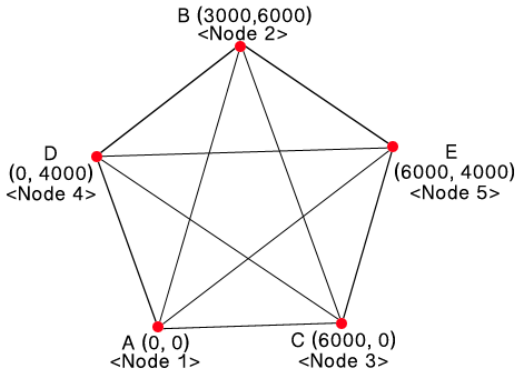

Figure 2: Final Scenario

for each coordinate and writing the wkt file by ourselves cost us too much time. There is another way to generate the coordinates for wkt files from the real world. We can generate our wkt file by using "osm2wkt" program. This program can convert the open source street information to wkt file. Namely, if we want to conduct some experiments based on an existed street or path in the real world. We can firstly use open source street xml file to describe the area that we want to conduct the experiment. Then use "osm2wkt" to convert the xml file to wkt file for THE ONE use.

3.3 Configurate WKT

When we finish creating our path in "the_one_1.4.1/data" folder. The next step is configure the wkt file in THE ONE so as to let THE ONE know, we want to load our customized path instead of the default path. Firstly, we open our configuration file "simple_scene.one" (the file is mentioned and created in the previous case 1 report). Secondly, we put some configuration commands in this file (it is better to put these commands behind the Group Setting (case 1 of report mentioned group setting)) as Figure 3. Finally, we can set up our simulator view proportion to "0.05". Then we can see our path as Figure 4 based on "simple_scene.one" configuration.

From Figure 3 we can clear see that before we start to load our wkt file. Firstly, we should define our movement model.

```
## Movement model settings
# seed for movement models' pseudo random number generator (default = 0)
MovementModel.rngSeed = 1
# World's size for Movement Models without implicit size (width, height; meters)
MovementModel.worldSize = 6100, 6100

## Map based movement -movement model specific settings
MapBasedMovement.nrofMapFiles = 1
MapBasedMovement.mapFile1 = data/customized.wkt
```

Figure 3: Configuration of WKT File

"rngSeed" is one of the property of movement model. It indicates that what is the initialization value for the random instance to generate random value for moving speed and waiting time. "worldSize" is what we have mentioned in the previous section. It is the range of the container that contain our paths. The coordinate value of the path should not over the size of the world. When we finish defining the world, we should give the value to "nrofMapFiles" in order to let THE ONE know, how many wkt files need to be loaded. In our configuration, we just need one wkt file. So here, we give value 1 to this property (the rules of multi-files are the same as Group Settings in Case 1 report). Then, we give the uri of our wkt file to THE ONE by assign value to property "mapFile1". Finally, we can find our customized path in Figure 4

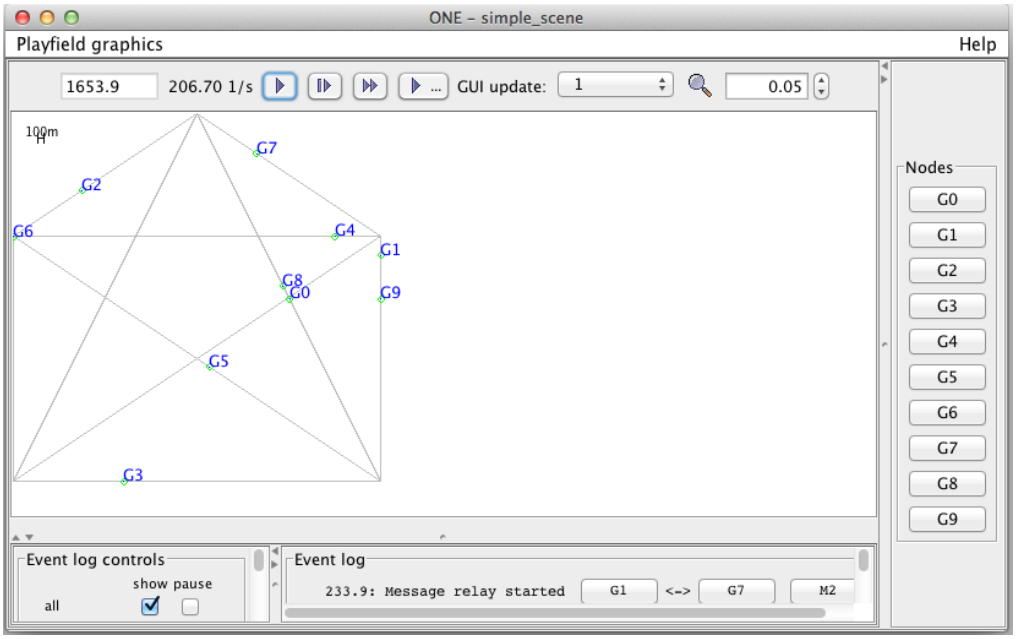

Figure 4: Final Effect in THE ONE

4 Specify Nodes Movement

4.1 Set up The Initialization Location

When we have the customized moving path (map). The next step is that we need to set up our node in a given position before they start to move. As I go through THE ONE's movement code, in my knowledge, there is no way for us to configure the initialization location for our moving node in configuration file. Therefore, we have to modify some codes in THE ONE so as to set up some initialization location for our moving node.

In order to set up the initialization location, we firstly need to change the total number of hosts from 10 (in previous Case 1 report we use 10 hosts) to 5. So we need to modify our configuration file "Group1.nrofHosts = 10" to "Group1.nrofHosts = 5". Also, due to the decrease of the number of hosts, we should modify the range of the number hosts who receive the message. Therefore, we need to modify the value in configuration file "Events1.hosts = 0,10" to "Events1.hosts = 0,5". If we do not change the value, 10 nodes can not fit the five conners of the path and message can not find some hosts to go. THE ONE will give us an error. Secondly, we go into function "getInitialLocation" of class "MapBasedMovement". Afterwards, we manually set up each node to stand on ABCDE point (because the coordinate of ABCDE we have known already). The code as Figure 5 shows, how we assign each node to a given point. In this code "iHostIndex" is the index of the map node that we create (Because THE ONE use a loop to create the node, so "iHostIndex" value is the node index and always increase). "iCurrentNode" is the variable to indicate and trace current node. Otherwise, we can node trace this node movement state later (when we create the moving path for this node). As we can see in the function, not only we give the exact position for each node, but also, we create another function called "getRestoredLocation" to convert the coordinate that we defined in array "initLocation". The reason for defining this function is that the coordinate system of Java (top left is 0,0) is different from our common coordinate system (bottom left is 0, 0). When THE ONE draw these points, it has some calculations. Therefore, when we pass the value to THE ONE, the real coordinate value is not the value which THE ONE can recognize. Therefore, we have to write a function to convert the common coordinate system value to the value that THE ONE can draw. Furthermore, after we finish give the initialization value to each node. We must pass the point of current host to "lastMapNode" in order to enable the movement of current node. If we do not pass the point to the variable. THE ONE will give us an error. Finally, we can see the result in Figure 6.

```
public static int iHostIndex = -1;
public int iCurrentNode = -1;
public Coord getRestoredLocation(Coord location)
{
    double yBound = map.getMaxBound().getY();
    return new Coord(location.getX(), -location.getY() + yBound);
}
 * Returns a (random) coordinate that is between two adjacent MapNodes
@Override
public Coord getInitialLocation() {
    Coord[] initLocation = {new Coord(0, 0),new Coord(3000, 6000),new Coord(6000, 0)
    ,new Coord(0, 4000),new Coord(6000, 4000)};
    iHostIndex += 1;
    iCurrentNode = iHostIndex;
    Coord coord = getRestoredLocation(initLocation[iHostIndex]);
    this.lastMapNode = map.getNodeByCoord(coord);
    return coord;
}
```

Figure 5: The Code to Set up Initialization Location

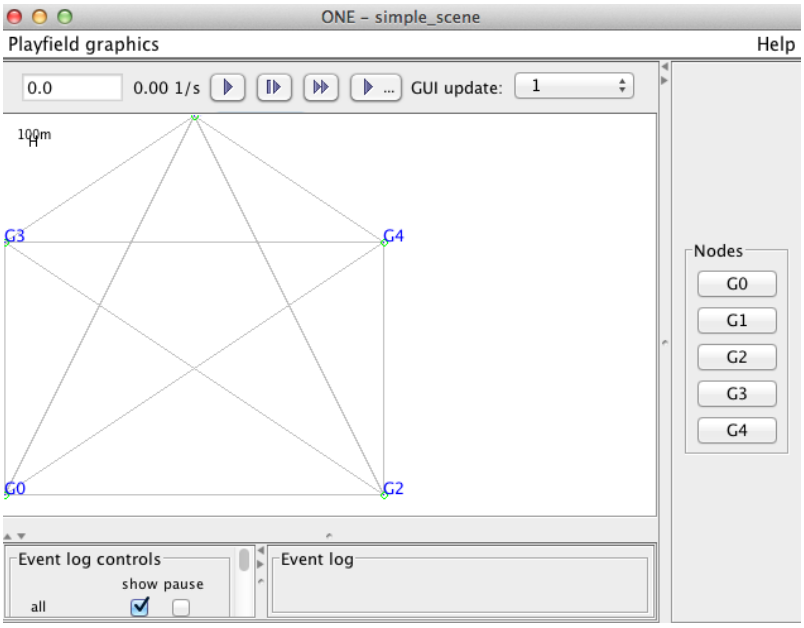

Figure 6: The Position of Nodes in THE ONE

4.2 Set up Moving Path

After we finish set up our nodes to the position. Now we need to do something for node 5 in order to control it to move based on the path we define. Then, set up the paths for node 1 - 4 to randomly move. In order to reach our goal, we need to go to function "getPath" of class "ShortestPathMapBasedMovement" (in fact, we can create our own class for creating a moving path to node). The reason why we select modify "ShortestPathMapBasedMovement" is that, we set up the movement of our nodes in configuration file to "ShortestPathMapBasedMovement" (Case 1 report describes how to set up it ["Group1.movementModel = ShortestPathMapBasedMovement"]).

The detailed code for setting up node 5 is given in Figure 7. As we can see in this Figure, there is a red rectangle area. this area is used to control the movement of node 5 (node 5's index is 4). Firstly, we define our five points coordinate (it is the same as we do the initialization work in Section 4.1). Then we define a path index array to indicate the path for node 5. Afterwards, we use a loop to set up the path based on the coordinate and the order array. Finally, we add all the

coordinate into a *Path* variable. Then, node 5 (Label:G4 in THE ONE) can follow the defined path to walk. Because we never consider other nodes, therefore, other nodes will pick up the random direction and use Dijkstra algorithm to create their own walking path.

```
@Override
public Path getPath() {
    Path p = new Path(generateSpeed());
    if(iCurrentNode == 4)
    {
        Coord[] initLocation = {new Coord(0, 0),new Coord(3000, 6000),new Coord(6000, 0)
        ,new Coord(0, 4000),new Coord(6000, 4000)};
        int[] iOrderNode5 = {2,0,3,1,4};
        for(int i = 0; i < iOrderNode5.length; i++)
        {
            p.addWaypoint(this.getRestoredLocation(initLocation[iOrderNode5[i]]));
        }
        return p;
    }
    MapNode to = pois.selectDestination();
    List<MapNode> nodePath = pathFinder.getShortestPath(lastMapNode, to);
    // this assertion should never fire if the map is checked in read phase
    assert nodePath.size() > 0 : "No path from " + lastMapNode + " to " +
        to + ". The simulation map isn't fully connected";
    // original one
    for (MapNode node : nodePath) { // create a Path from the shortest path
        p.addWaypoint(node.getLocation());}
    lastMapNode = to;
    return p;
}
```

Figure 7: Set up Node 5 to Walk in Clockwise Direction

4.3 Control node Moving Speed

As for control the speed of node 5. it is very simple. what we need to do is call another function when we want to add a speed between two point (one function is "addWaypoint(Cood wp)", another one is "addWaypoint(Cood wp, double speed)"). Because every time, when a node finish walking one path, it will call "getPath" function again and walk again. Therefore, if we want to increase the speed of node 5 at every turn of walking. We just need to define a variable for node moving speed out of "getPath" function. Then every time, after THE ONE finish using this variable in this function, we increase this value and reuse it in the next time. Finally, we can see that the speed of node 5 will increase when it finishes every walking (circle from ABCDE).

5 Conclusion

In this report, we describe how to use wkt file to define a customized map for nodes to move and shows up how to set up the initialization position of our nodes. Moreover, I shows how to customized a moving path for a special node. From what we can see in the previous description, it is clear that THE ONE is a very good DTN simulator for us to conduct experiments. Not only it is straightforward, but also it is very flexible for us to improve some special features for our work.