5th INTERNATIONAL CONFERENCE
TRANSPORT SYSTEMS TELEMATICS TST'05

ZESZYTY NAUKOWE
POLITECHNIKI ŚLĄSKIEJ 2005
TRANSPORT z. 59, nr kol. 1691

Multi-agent, NetLogo, complex system, model, simulation, level crossing

Aleš JANOTA [1]
Karol RÁSTOČNÝ [1]
Jiří ZAHRADNÍK [1]

MULTI-AGENT APPROACH TO TRAFFIC SIMULATION IN NETLOGO ENVIRONMENT – LEVEL CROSSING MODEL

NetLogo [1] is a multi-agent programming language and cross platform modelling environment for simulation of complex phenomena and is designed for both research and education. It is used across a wide range of disciplines and education levels. In this paper, though, we focus on NetLogo as a tool for creating traffic models and their simulation and present a level crossing example.

WIELOCZYNNIKOWE PODEJŚCIE DO SYMULACJI RUCHU W ŚRODOWISKU NETLOGO – MODEL PRZEJAZDU

NetLogo [1] jest to wielo-czynnikowy język oprogramowania i międzyplatformowe środowisko modelowania pozwalające na modelowanie skomplikowanych zjawisk, przewidziane zarówno dla badań, jak i dla celów edukacyjnych. Wykorzystywany jest w szerokim zakresie dziedzin i poziomów edukacji. W niniejszym referacie jednak skupiamy się na NetLogo jako na narzędziu do tworzenia modeli ruchu drogowego i ich symulacji oraz prezentujemy przykład przejazdu.

1. INTRODUCTION

Modelling is a powerful tool that allows a programmer or engineer to observe cause-and-effect relationships in occurrences that happen too slowly or quickly to see; involve danger or safety concerns; occur on a scale too large or too small for study; are not a common occurrence or simply can hardly be realised in real environment with real entities. Multi-agent systems (or agent-based models) are composed of collections of synthetic, autonomous, interacting entities and have their origins in computer science, where they are used in the design of Artificial Intelligence, and in Information Communications Technology as bots and webcrawlers. Agent-based models can be used to simulate mobile entities (vehicles, pedestrians, migrating households) in spatial simulations. Agents are pieces of software code

[1] Department of Control and Information Systems, Faculty of Electrical Engineering, University of Žilina, Univerzitná 8215/1, SK-010 26 Žilina, Slovakia.
Phone: +421-41-513 3301, Email: {ales.janota | karol.rastocny | jiri.zahradnik}@fel.utc.sk

with attributes that describe their condition and characteristics that govern their behaviour (a very simplified characterization of agents).

NetLogo is a multi-agent programming language and cross platform modelling environment for simulation of complex natural and social phenomena. It is particularly well suited for modelling complex systems evolving over time. Modellers can give instructions to hundreds or thousands of independent "agents" all operating concurrently, in order to explore connections between micro-level behaviours of individuals and macro-level patterns that emerge from their interactions. NetLogo enables users to open simulations and "play" with them, exploring their behaviour under various conditions [2]. There are three types of agents: turtles, patches, and the observer. Turtles are agents that move around in the world. The world is two dimensional and is divided up into a grid of patches. Each patch is a square piece of "ground" over which turtles can move.

NetLogo was created by Northwestern University's Center for Connected Learning and Computer-Based Modeling and is freeware – anyone can download it for free and build models without restrictions. It comes packaged with extensive documentation and tutorials and a large collection of sample models (a library with over 150 sample models) for easy access. NetLogo has been under development since 1999. More on a history of NetLogo's origins including a tour of the NetLogo interface, an introduction to the NetLogo language and other details can be found in the less technical paper [3] recommended by the authors for background reading. The example discussed hereafter was created using the *NetLogo3.0 beta1* version available at the time of writing the paper (June 2005). As a language, NetLogo is a member of the Lisp family that supports agents and concurrency.

1.1. BRIEF SURVEY OF EXISTING TRAFFIC MODELS

Traffic domain is one of prospective domains where NetLogo can successfully be applied and bring novel views at solving everyday traffic problems. Several NetLogo examples devoted to traffic that have been published so far can be summarised as follows:

- "Traffic Basic" model [4] is included in the NetLogo's models library and models the movement of cars on a highway where each car follows a simple set of rules: it slows down if it sees a car close ahead and, speeds up if it doesn't see a car ahead (a newer version is also available in [5]);
- "Traffic 2 Lanes" model [6] is a more sophisticated two-lane version of the "Traffic Basic" model that demonstrates how traffic jams can form and gives drivers a new option (they can react by changing lanes);
- "Traffic" model [7] presents a one-lane traffic model that includes adjustments of speed, settings of police, a special driver ticking to the maximum official speed limit and several monitors useful for numerical experiments;
- "TrafficMode" model [8] uses 3 lanes and enable us to effectively study effect of different conditions (traffic lights, human curiosity) on the flow of traffic;
- "Gridlock" model [9] enables us to control traffic lights and overall variables, such as the speed limit and the number of cars, in a real-time traffic simulation and thus explore traffic dynamics;
- "HubNet Gridlock" model [10] represents extension of the previous model and solves traffic as an adaptation problem (since traffic flows and densities change constantly);
- Model based on the paper [11], which consists of a simulator to model traffic at intersections and three different intersection control policies: overpass, traffic light, and

reservation system. For certain the given survey is not complete, however, at least the main typical examples are mentioned.

2. LEVEL CROSSING EXAMPLE

Creation of a model that is presented in this paper has been inspired by some of models mentioned above. It enables us to model rules commonly accepted at crossings of roads and railway lines and study effects of different conditions on the flow of traffic. The NetLogo environment provides three main screens called Interface, Information and Procedures.

2.1 INTERFACE

Fig.1 shows NetLogo's user interface after opening and running a model. The graphic window makes the two-dimensional "world" of the model visible. It is divided up into a grid of patches that have coordinates pxcor and pycor. The patch in the centre of the world has coordinates (0,0) and the total number of patches is determined by the settings screen-edge-x and screen-edge-y. In our model the grid consists of 41 × 17 patches (horizontally from - screen-edge-x = - 20 to screen-edge-x = 20 with 0 in the middle; vertically from - screen-edge-y = - 8 to screen-edge-y = 8 with 0 in the middle).

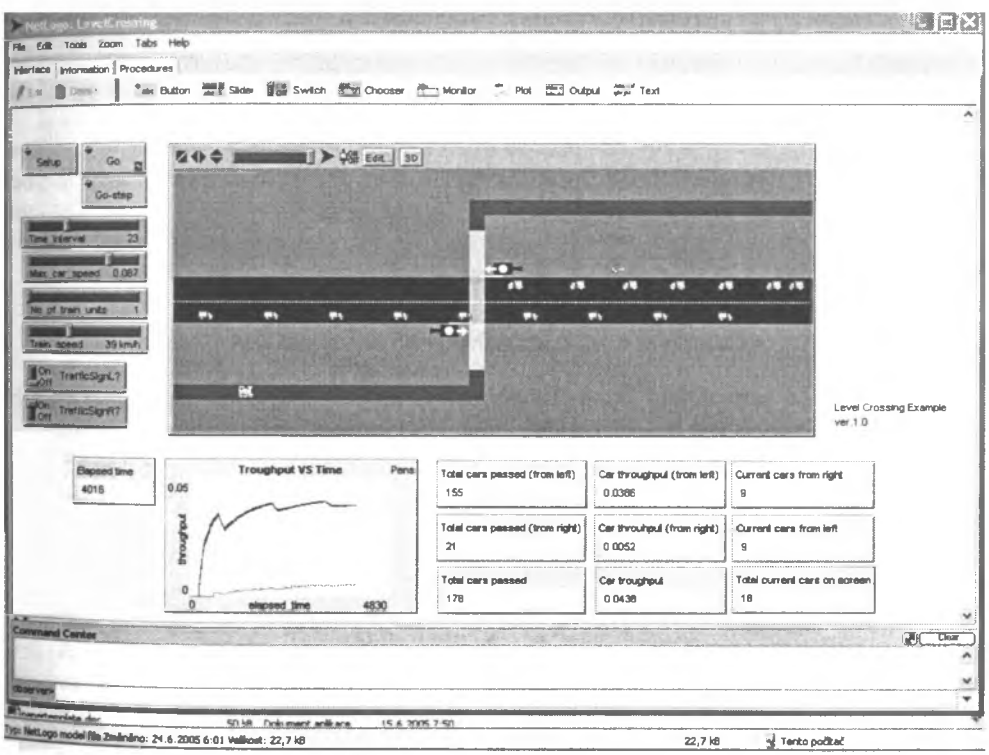

Fig.1. NetLogo's user interface, with model Level Crossing

Turtles (cars, train, traffic signs, barriers etc.) have coordinates too: xcor and ycor. Generally, when a turtle moves past the edge of the world, it disappears and reappears on the opposite edge. This way has been used for creating an agent "train", however, the "car" agents are "created" and "killed" in a more controlled way. Observer as the third kind of NetLogo's agents doesn't have a location and is as looking out over the world of turtles and patches.

Model controls (on the left) quickly adjust the settings of the initial environment. They are represented by the following buttons, sliders and switches:

- "Setup" button is used to initialise the model, it is a "once-button" that runs its code once, then stops and pops back up;
- "Go" button is used to run the model, it is a "forever-button" that keeps running its code over and over again, until either the code hits the stop command, or we press the button again to stop it;
- "Go-step" button is a "once-button" that runs only one step of the code and then stops;
- "Time_interval" slider is used to set a time interval for generating new cars (the higher time the greater distances between cars);
- "Max_car_speed" slider is used to set the maximum speed for cars going in both directions;
- "No_of_train_units" slider is used to set a number of units (locomotives) forming the train – in our model only 1, 2 or 3 are available;
- "Train_speed" slider enables the user to set a current speed of the train (here from 0 to 100 $km \cdot h^{-1}$);
- "TrafficSignL?" switch enables the user to create a traffic sign for cars approaching the level crossing from the left; its existence forces all these cars to slow-down. Another switch "TrainSignR?" is used analogically and has the same effects on cars going in the opposed direction.

In the bottom part, above the Command Center box, there are "monitors" that allow the user to quickly examine variables (e.g. current number of cars in the world, total number of cars passed in either or both directions, throughput etc.), even as the simulation runs, and graphical output providing a quantitative way for the user to observe, record and compare data.

2.2 INFORMATION

According to conventions applied to NetLogo models, this informative part usually consists of nine sections and helps the user to find quick answers to the following answers:

- "What is it?" - a general understanding of what the model is trying to show or explain;
- "How it works" - what rules the agents use to create the overall behaviour of the model;
- "How to use it" - how to use the model, including a description of each of the items in the interface table;
- "Things to notice" - some ideas of things for the user to notice while running the model;

- "NetLogo features" - possible pointing out any especially interesting or unusual features of NetLogo that the model makes use of, particularly in the procedures table; it might also point out places where workarounds were needed because of missing features;
- "Related models" - the names of models in the NetLogo models library or elsewhere which are of related interest;
- "Credits and references" - a reference to the model's url on the web if it has one, as well as any other necessary credits or references.

The "Level Crossing" model enables the user to model different operation modes of level crossing installation and to see what are their effects on traffic. According to Fig.2 totally four different phases of a train movement across the level crossing can be identified:

Phase 1 – a train is approaching the level crossing, the equipment is in a default mode (no warning, possibly active signalling of vacated crossing and fault-free state of the equipment);

Phase 2 – a train has entered an approaching section what is detected by a proper track-side equipment (in reality a technical means such as track circuits, balises etc.); this results in activation of warning lights;

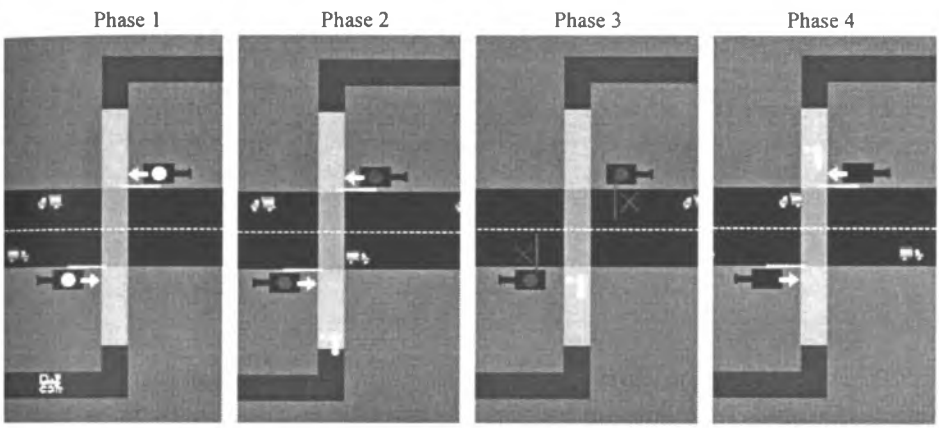

Fig.2. Train movement across the level crossing

Phase 3 – the warning period goes on, the barriers fall down to protect cars from entering the hazardous area and the train passes the level crossing what would be normally recorded by a proper technical means;

Phase 4 – the train has left the crossing and finds itself in an annulation's section (warning lights become of, active signalling is still off since the level crossing installation is in the annulation's state).

After the train has left the annulation's section the whole cycle is over and the equipment enters again the Phase 1.

Another function that can be modelled is setting a variable time period whose size depends on the current speed of the train approaching the level crossing. In our model this feature is modelled in a simplified way. We distinct only two categories of train speeds through different positions used to activate the red warning lights. Trains running with the current train_speed <= 40 $km.h^{-1}$ activate red warning lights when occupying the patch with

coordinates [-1,-4], and barriers at the patch with coordinates [-1,-3]. All trains running with the current train_speed > 40 km.h^{-1} activate both warning lights and barriers earlier (i.e. at the patch [-1,-6] and [-1,-5]).

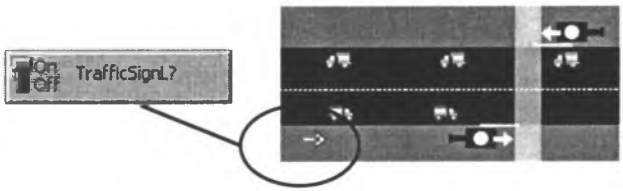

Fig.3. Setting of the Traffic Sign for cars going from left

The setting must enable the longest and slowest road vehicle, which is at the crossing just at the moment of activating warning lights, to leave the hazardous area before the train coming (falling the barriers down).

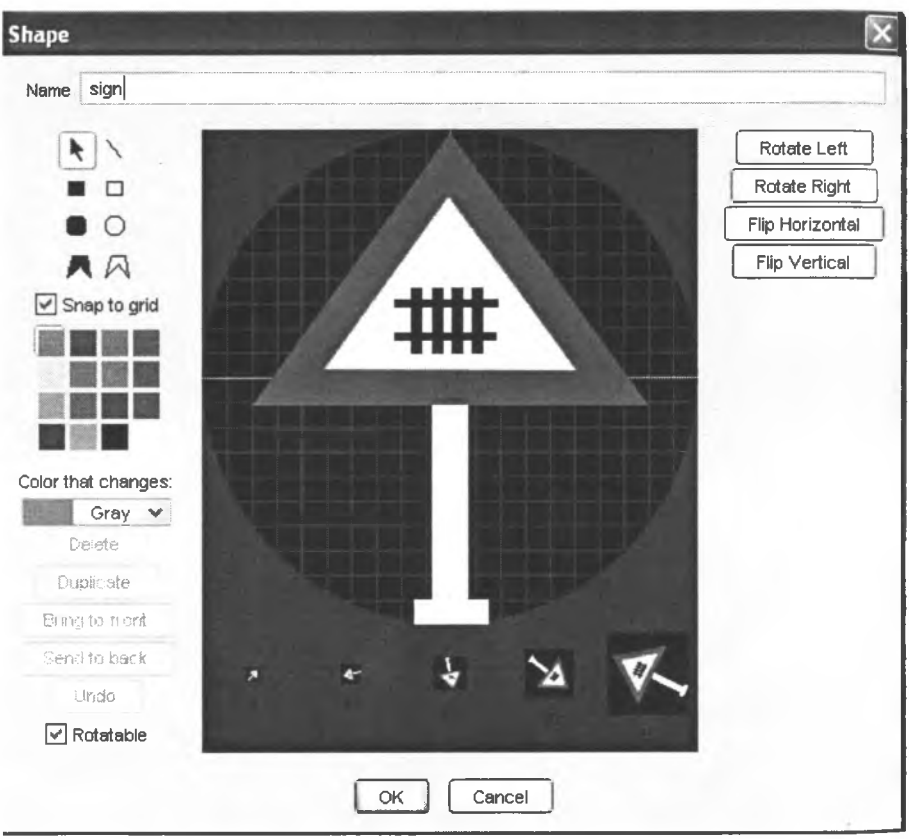

Fig.4. Creating a shape of the agent "sign"

Shapes of individual turtles-agents can be selected from a library, imported from another model or newly created (see Fig.4). Editor for turtle shapes makes it easier to customize how a model looks. This is important for data visualization.

Another feature respected in the model is a different length of the train. To be sure that a train left the crossing and the barriers may be taken up, different patches (patches with different coordinates) are used to finish the annulation's state for trains with a different length. A sample of the corresponding code is given in Fig.5.

The user may also decide if a traffic sign reducing a traffic speed for cars will be installed prior to the level crossing, individually for either of directions (see Fig.3). What's more, setting the time interval may influence frequency of creating cars (created together for both directions), and the "Max_car_speed" button may be used to set cars' maximum speed (their current speed depends on many conditions).

2.3 PROCEDURES

In the section "Procedures" the source code of the model can be seen and edited. To show how it looks like, a sample of the code corresponding to the described function of setting a variable time for warning lights activation as well as considerations of different lengths of train is given in Fig.5.

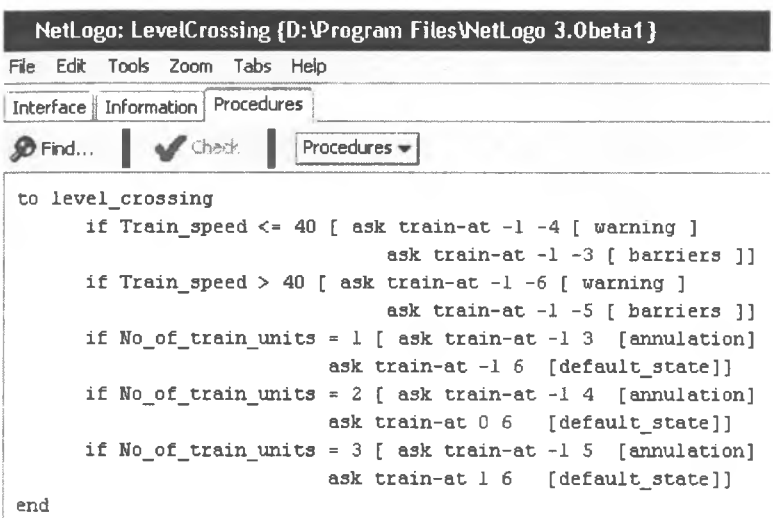

Fig.5. Control of level crossing states

3. CONCLUSIONS

The authors wrote this paper with intention to briefly introduce NetLogo environment, to give a survey of currently available NetLogo's traffic-based models and above all to present their model of level crossing. The model enables the user to give instructions to independent "agents" that are all operating concurrently creating the target behaviour of the whole system. NetLogo environment is under rapid development and able to support large,

ambitious modelling efforts. Its use is expected for both research and educational contexts since it significantly reduces complexity of the software development process.

This work has been partially supported by the Grant Agency of the Slovak Republic VEGA, grant No. 1/1044/04 "*Theoretical Foundations for Implementing e-Safety Principles into Intelligent Transportation Systems*" and partially by the institutional research project No. 07/604/2005 "*Transport telematics and tools to improve its quality*" solved at the Faculty of Electrical Engineering, University of Zilina.

BIBLIOGRAPHY

[1] WILENSKY U., NetLogo. http://ccl.northwestern.edu/netlogo. Center for Connected Learning and Computer-Based Modeling. Northwestern University, Evanston, IL, 1999.

[2] TISUE S., WILENSKY U., NetLogo: Design and Implementation of a Multi-Agent Modeling Environment, SwarmFest, Ann Arbor, 2004.

[3] TISUE S., WILENSKY U., NetLogo: A Simple Environment for Modeling Complexity. International Conference on Complex Systems, 2004.

[4] WILENSKY U., NetLogo Traffic Basic model. http://ccl.northwestern.edu/netlogo/models/TrafficBasic. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1997.

[5] http://www.cs.northwestern.edu/~ade285/Traffic%20Basicnew2.nlogo

[6] WILENSKY U., NetLogo Traffic2 Lanes model. http://ccl.northwestern.edu/netlogo/models/Traffic2Lanes. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1998.

[7] http://sps.nus.edu.sg/~marimuth/Lab2/traffic.nlogo

[8] http://web.cz3.nus.edu.sg/~chenk/gem2503_3/project/Group7/trafficMode.nlogo

[9] WILENSKY U., NetLogo Gridlock model. http://ccl.northwestern.edu/netlogo/models/Gridlock. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 2002.

[10] WILENSKY U., STROUP, W. NetLogo HubNet Gridlock model. http://ccl.northwestern.edu/netlogo/models/HubNetGridlock. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 2002.

[11] DRESNER K., STONE, P. Multiagent Traffic Management: A Reservation-Based Intersection Control Mechanism. In Proceedings of the Third International Joint Conference on Autonomous Agents and MultiAgent Systems, p. 530--537, ACM. 2004

Reviewer: Ph. D. Jerzy Mikulski