Geographic Automata Systems: From The Paradigm to the Urban Modeling Software

Itzhak Benenson[$,§], Vlad Kharbash[§]
[$]Department of Geography and Human Environment,
[§]Environment Simulation Laboratory, Porter School of Environmental Studies,
Tel Aviv University
bennya@post.tau.ac.il, http://www.tau.ac.il/~bennya

SUMMARY

The concept of Geographic Automata System (GAS) formalizes an object-based view of city structure and functioning; OBEUS software implements this view on the operational level. The paper presents the basic components of the latest version of OBEUS, which is based on .NET technology and developed according to OODBMS logic. We claim that all urban Cellular Automata and Multi-Agent models we are aware of can be represented as GAS and implemented as OBEUS applications. GAS and OBEUS can thus serve as a universal framework for object-based urban simulations.

KEYWORDS: *Geographic Automata, urban simulation, cellular automata, multi-agent systems, Object-Oriented Programming*

FROM REGIONS AND CELLS TO SPATIALLY LOCATED OBJECTS

Urban and regional modeling started in 1960s (Forrester 1961; Lowry 1964; Chapin and Weiss 1965; Forrester 1969) with two lines of reasoning. The first operated with black-box *regions*, characterized by averaged characteristics such as fractions of the land-uses, population, jobs, transportation and services (Forrester 1969; Allen and Sanglier 1979). The second aimed at representation of a system at as high as possible resolution and operating with homogeneous spatial units (Chapin and Weiss 1965; Chapin and Weiss 1968). During first two decades of development, the concept of regional modeling took over, but its operational achievements were limited (Batty 2003). After the decade of frustration, the time came for the Cellular Automata (CA) model, which followed the second line and flourished during the 1990s (Batty 1997; Clarke, Hoppen et al. 1997; Portugali 2000; White and Engelen 2000; Benenson, Omer et al. 2002).

Nowadays we enter the third stage of urban modeling, which focuses on real-world urban objects, e.g. houses, cars, road segments, householders. Contrary to CA, the GAS framework does not demand division of space into cells in order to define the artificial skeleton that represents urban space. The object-based view of the cities is formalized with the recently introduced Geographic Automata Systems (Benenson and Torrens 2004; Torrens and Benenson 2005). Individual urban objects are directly represented in GAS by discrete *automata*. These automata can be of different nature, spatial extension, and hierarchical rank; they can also represent spatially fixed as well as non-fixed entities, the locations of which change in time. The structure of urban space is defined by *spatial relationships* between automata. Although the object approach to representation of urban reality has been mentioned in the literature (Erickson and Lloyd-Jones 1997; Semboloni 2000; Galton 2001), the idea itself remained inchoate up to date.

The recent boom in GIS data production and research provides strong empirical support for the GAS approach: Layers of urban GIS are nothing but collections of urban objects of the same kind. Spatial relationships between objects can be estimated within GIS based on adjacency, overlay, visibility, or accessibility. Moreover, the pattern recognition methods applied to objects of GIS layers can be utilized to grasp spatial emergence and self-organization.

To take the next step and portray *urban dynamics*, we have to 'animate' geographic features, that is, formulate the rules by which urban objects are created, relocated, changed, and destroyed. This step demands formulation of *geographic automata transition rules*; informally, these rules describe the *behavior* of urban objects. Benenson and Torrens (2004),who label the GAS view of urban model construction *Geosimulation*, claim that every high-resolution urban model developed to date can be reformulated in GAS terms.

The claim for GAS universality remains pretentious until a constructive procedure of transformation applicable, at least in principle, to any urban model, is provided. We propose that a software system capable of implementing the GAS framework can be considered as such a proof. This paper presents the most recent progress made in development of Object-Based Environment for Urban Simulation (OBEUS), a system that verily operationalizes the GAS concept.

We assume that the reader is familiar with the background of Relational DBMS, the Entity-Relational Data Model (ERM) (Howe, 1983), and the Object-Oriented (OO) programming paradigm (Booch, 1994). In the following, only the main components OBEUS software are presented; other illustrations of the concept can be found in the literature (Benenson, Aronovich et al., 2004; Benenson and Torrens, 2004; Torrens and Benenson, 2005), and in the OBEUS manual. The OBEUS software and the manual can be downloaded from http://www.eslab.tau.ac.il/OBEUS/OBEUS.htm

A SHORT INTRODUCTION TO GEOGRAPHIC AUTOMATA SYSTEM (GAS)

The GAS concept treats urban infrastructure and social objects as spatially located automata - *Geographic Automata* (GA) (Benenson and Torrens 2004). To recapitulate the notions we have to incorporate into GA:

- An abstract automaton **A** is characterized by (vector) state **S**, which changes in time according to state transition rules **T**, depending on current input **I**.
- Cellular Automata theory considers **A**'s neighborhood **N(A)** and assumes that input **I** is defined by automata belonging to the **N(A)**.
- Agent **A** of a Multi-Agent System (MAS) can relocate in space, that is, its representation should be capable of managing location and neighborhood changing over time.

The minimal set of transition rules **T** for geographic automata **G** should thus specify changes in:

- Non-spatial attributes of G's state
- Location of G
- Relationships of G with other geographic automata

To formalize these demands, Benenson and Torrens (Benenson and Torrens 2004) consider Geographic Automata System G to consist of automata of different *types* (**K**). Automata of given type $\mathbf{k} \in \mathbf{K}$ are characterized by a non-spatial set of states - $\mathbf{S^k}$, *geo-referencing conventions* — $\mathbf{L^k}$ — that determine how automata are located in space , and their *relationships* to automata of the same and other types - $\mathbf{N^k}$ (we later omit the upper index **k**). Geo-referencing conventions and relationships usually include automata of different types.

State transition rules $\mathbf{T_S}$ determine how automata non-spatial states automata change in time, *movement rules* $\mathbf{M_L}$ govern changes of location whereas *relationship transition rules* $\mathbf{R_N}$ specify how automata relationships change in time.

Altogether, a Geographic Automata System **G** may be defined by seven components:

$$\mathbf{G} \sim \langle \mathbf{K}, \mathbf{S}, \mathbf{T_S}, \mathbf{L}, \mathbf{M_L}, \mathbf{N}, \mathbf{R_N} \rangle \qquad (1)$$

Unitary geographic automaton **G** is characterized at t by state S_t, location L_t, and relationship N_t; the state, location and relationships at t + 1 are determined by the transition rules $\mathbf{T_S}$, $\mathbf{M_L}$, $\mathbf{R_N}$:

$$\begin{aligned} &\mathbf{T_S}: (S_t, L_t, N_t) \rightarrow S_{t+1} \\ &\mathbf{M_L}: (S_t, L_t, N_t) \rightarrow L_{t+1} \\ &\mathbf{R_N}: (S_t, L_t, N_t) \rightarrow N_{t+1} \end{aligned} \qquad (2)$$

Exploration with GAS **G** involves the qualitative and quantitative investigation of its spatial and temporal dynamics, given all the components defined above. In this way, GAS models offer *Geosimulation* framework for spatially enabled collectives of geographic objects.

FROM GEOGRAPHIC AUTOMATA SYSTEM TO SOFTWARE

To interpret GAS paradigm, we have to translate all the components of (1) - (2) into software objects and methods.

Automata of a given type k∈K → Instances of population class

Urban objects always utilize information above the individual level. Objects belonging to an OBEUS **population** class represent "containers" for this meta-data. For instance, populations of cars can be restricted in their speed; the limitation is applicable to all cars. The methods of the population class include, in particular, procedures for estimating aggregate characteristics of the output.

Individual automata of a given type k → Class of objects of type k

At a conceptual level, OBEUS considers unitary **objects**, which are distinguished as either **fixed** and **non-fixed** (Benenson, Aronovich et al. 2004). Fixed objects — buildings, parks, road links, traffic lights, etc. — do not change their location once established whereas non-fixed objects — tenants, firms, pedestrians, vehicles, etc. — do. In the majority of situations, fixed objects can be referred to as *immobile* while non-fixed objects can be referred to as *mobile.* The dichotomy between fixed and non-fixed objects depends on the model: the parcels of a land partition can be considered as fixed in a model of urban land-use dynamics, and as non-fixed in a model of urban sprawl, where big agriculture parcels are divided into smaller ones before becoming a built-up area.

Two ways of automata locating

At a conceptual level, OBEUS considers two ways in which automata can be geo-referenced. Automata can be located in space **directly**, by means of a coordinate list, in a manner similar to vector GIS. For example, these can be automata, representing road segments in the model of residential dynamics. Another fundamental method of automata locating in OBEUS is indirect, by **pointing** to one or several other objects. For example, tenants in the model of residential dynamics are located by pointing to their habitats. In OBEUS we assume that all objects of a given type **k** ∈ **K** are located in the same manner. Indirect locating is expressed in OBEUS by means of relationships.

Relationships between automata → Class of *relationships*

OBEUS follows the Entity-Relationship Data Model (ERM) (Howe 1983); it therefore considers relationships between entities explicitly, as software objects. The compelling_dominance of the Entity-Relationship model in modern DBMS is the best proof available of the benefits reaped from the distinguishing between objects and relationships and the view of relationships as self-existing software objects.

To illustrate the use of relationships for geo-referencing, let us consider a hypothetical system consists of objects of the two types – **cars** and traffic **lanes**. Let us express the location of the car on a lane by means of the **car-lane** relationships, which has one attribute *FPosition* - the distance between an *FNode* node of a lane and a car along the lane (one can easily recognize standard dynamic segmentation in this view). Let us express adjacency of lanes by means of **lane-lane** relationship,

which has a Boolean attribute *Side*, equals *True* when **lane$_2$** in the **lane$_1$-line$_2$** instance of this relationship is to the right of **lane$_1$** and *False* otherwise.

If a certain **car$_1$** drives along the **lane$_1$** (i.e. is related to **lane$_1$**), the *FPosition* attribute is updated depending on car velocity. When the **car$_1$** changes the lane to the lane on the right, then the **car$_1$-lane$_1$** instance of relationship is destroyed, and the lane related to the **lane$_1$** and satisfying condition '*Side* = *True*' is retrieved from the table of the **lane-lane** relationship. Let the result of the query will be **lane$_2$** - the construction of the **car$_1$-lane$_2$** instance of the **car-lane** relationship concludes the procedure.

Relationships treated as separate software classes unify CA and explicit GIS-based land-use models, both of which are rooted in neighborhood relationships. For a regular square CA grid, von Neumann or Moore neighborhoods entail neighborhood relationship of 1:4 or 1:8 degrees. The popular 'constrained CA' (White and Engelen 2000) is based on neighborhoods of radius 6 around the cell, entailing neighborhood relationship of 1:113 degrees and accounts for a relationship's property – weight, which reflect the influence of neighboring cells on the land use of given cell. The only difference between a regular CA grid and an irregular land partition is the variation in degree of neighborhood relationship between the parcels (Flache and Hegselmann 2001; Benenson, Omer et al. 2002).

According to the Relational DBMS theory, relationship objects and their properties are stored in DBMS tables just as the objects representing geographic entities. The more complex the definition of a relationship and its properties, the greater the computational advantage of using a relationship class and a table presentation over on the fly evaluations as to whether two objects are related, made during simulation runs.

Limitations on relationships in OBEUS

GAS is a dynamic system and the properties of the unitary geographic automata as well as their relationships change over time. To avoid inconsistencies, essential limitations are imposed on the semantics of the relationships covered by OBEUS:

First, we assume that relationships between fixed objects are also fixed, that is, they do not change once established.

Second, no direct relationships between non-fixed objects are permitted. That is, non-fixed unitary objects can be related to fixed objects only; in particular, when locating by pointing, non-fixed objects can be related solely to fixed objects. Non-fixed objects maintain relationships in a transitive fashion, when non-fixed object is related to another non-fixed object only if they are related to fixed objects, which are themselves related. OBEUS *automatically constructs* methods for retrieving transitively related objects when the direct relationships are defined.

Direct relationships between non-fixed objects are undoubtedly important in the real world. For example, a slow car can force other drivers to slow down. However, car-road relationship is sufficient to express this influence, while car-lane and lane-lane relationships are sufficient for representing details of the complex maneuvers. The models, where non-fixed objects cannot be located by pointing to the fixed ones – as models of flocks and fish schools (Reynolds 1987; Vicsek, Czirok et al. 1995; Brogan and Hodgins 1997; Toner and Tu 1998; Levine and Rappel 2001) can be realized in OBEUS in two ways. One is based on introduction of the artificial (grid) structure of space, the other on direct estimating of close objects. The performance of the second approach is essentially lower when the number of individual automata is large.

The relationships between non-fixed and fixed objects are formalized in OBEUS by means of the **leader-follower pattern** of relationship updating (Noble 2000). According to this pattern, non-fixed

objects have exclusive update rights whereas for the fixed objects relationships are read-only. An example of a car changing lanes illustrates this pattern.

In OBEUS, the leader and follower are defined separately for each relationship. Relationships between fixed objects are initialized when those objects are initiated; afterwards, they are retrieved only. Relationships between non-fixed and fixed objects are created or destroyed according to requests from the former, which are always the leaders.

Movement rules → Methods of relationship class

As noted, location agreements **L** of the GAS definition (1) become components of object state **S** in cases of direct location (fixed objects), or are represented by the relationship **N** in cases of indirect location (fixed and non-fixed objects). In the former case, the location does not change; in the latter, it is represented by the procedure, i.e. a method of relationship class. As a result, GAS can be formally restricted to

$$\mathbf{G} \sim \langle \mathbf{K}, \mathbf{S}, \mathbf{T_S}, \mathbf{N}, \mathbf{R_N} \rangle \qquad (3)$$

In this way, OBEUS becomes a temporal database.

State and relationship transition rules T_S and R_N → Automata behavior and assessment rules

Geographic objects 'behave.' In terms of GAS, geographic automata change states and relationships with other automata. Interpretation of the behavior of real-world objects in terms of the transition rules applied to software objects is the essence of any model; in OBEUS, this interpretation is 'coded' as methods of the software classes that represent automata of different types.

In OBEUS, we distinguish between **assessment rules**, aimed at estimating the parameters objects react to, and **automation rules,** which describe the behavioral act itself. We assume that the number of assessment rules is limitless but that only one (arbitrarily complex) automation rule can be active during any single simulation run.

Population time versus unitary objects time

In OBEUS, we distinguish between observer time and unitary object time. The student of GAS is interested in observing collective processes that occur at population and above levels in **population time**, which is measured in **iterations**. **Automata** have their own **time**, which is measured in **ticks**, depending on object type. Population characteristics are updated, aggregate criteria are tested, and the system state is stored in the course of iterations. Alternatively, objects change properties, locations, and relationships in ticks. Ticks and iterations can be variously interpreted and are model-specific although specified in part by the *synchronization scheme* applied.

Developments in Object-Oriented Databases indicate that there is no general, conceptual solution to the synchronization problem. Instead, existing methodologies provide software development patterns for specific classes of problems (Zeigler, Praehofer et al. 2000). OBEUS architecture utilizes two of those solutions when contending with synchronization problems.

Asynchronous mode: This is a basic OBEUS synchronization mode, when objects change in turn, with each observing the urban reality left by the previous object. To avoid the conflicts, caused by the order of updating, OBEUS demands that when applying the asynchronous mode, the modeler defines an order of object actions in advance. OBEUS supplies templates for temporal updating: for automata of a given class, random sequence and sequence in order of one of the automata characteristics are currently being implemented.

Synchronous mode: In this mode, the properties of urban entities are assumed to change simultaneously. The calling order of the objects has no influence on the model's outcome. Note that the logic of synchronous updating can push conflicts further down the time path. If two mutually

avoiding householders occupy adjacent locations and simultaneously leave them at a given time-step, nothing in the synchronous mode can prevent occupation of these locations by yet another pair of avoiding agents.

An important feature of OBEUS is the *immediate updating of relationships* irrespective of the mode chosen for automata updating. To illustrate, **car-lane** relationship is updated immediately after a car changes a lane.

OBEUS USER'S INTERFACE

In accordance to the above formalization, OBEUS interface includes three basic components:

- Model Tree window, where user defines the classes of model automata;
- Model Flow window, where user defines the synchronization scheme;
- C# compiler, where user formulates automata behavior rules

Model output is displayed in Map and Graph windows and stored as DBMS tables.

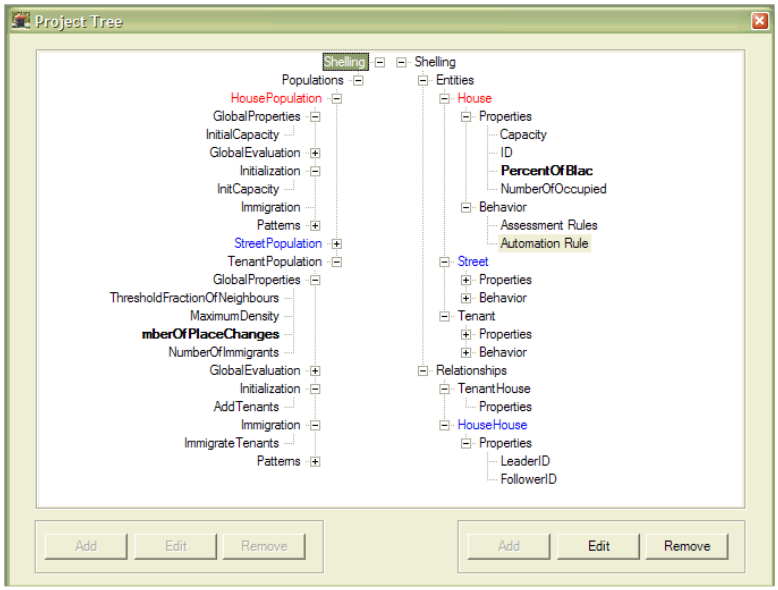
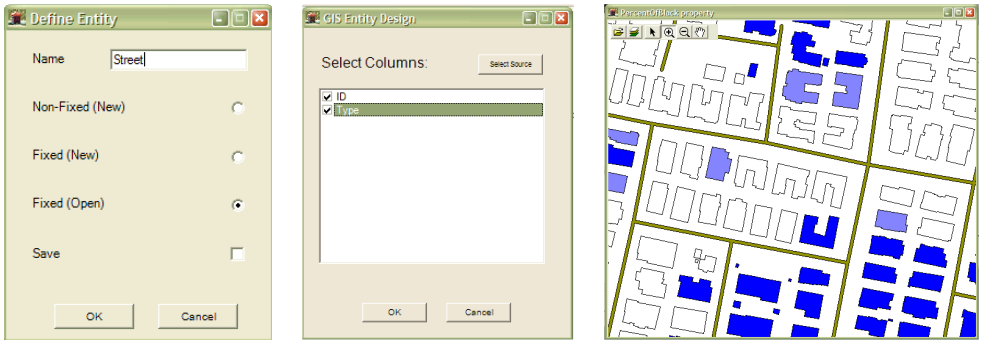


Figure 1. The model tree of the generalized Schelling residential dynamics model. The colors/bold fonts are used for marking various output options (save in DBMS, present by map, graph, etc.).

Building a model tree

By means of a model tree, the user defines classes of unitary automata, relationships and their properties. When a new type of automata is defined, the population of this type of automata is constructed automatically (Figure 1).

New classes of automata are introduced in the right-hand branch of the scheme and automatically cause construction of corresponding population classes in the left-hand branch (Figure 1). The class of automata and their properties can be defined from scratch according to templates (e.g., a cell grid) or acquired from a GIS layer (Figure 2). For fixed spatial automata, a map view is available. Information regarding non-fixed automata is presented by means of thematic maps.



a b c

Figure 2. The sequence of actions when defining a GIS-based **Street** object: (a) definition of an object type; (b) selection of the attributes from the list of the layers' attributes; (c) map view. The map presents the objects of the two types employed in Schelling model – **House** and **Street**, and thematic map representing the fraction of **Tenant** objects, which are related to a given **House**, and which *Color* attribute equals "Black"

To define a relationship, the choice of leader and follower is necessarily (Figure 3). According to OBEUS limitations, direct relationships between non-fixed automata are prohibited; in relationships between fixed and non-fixed automata, only the latter can be thus the leader in a relationship.

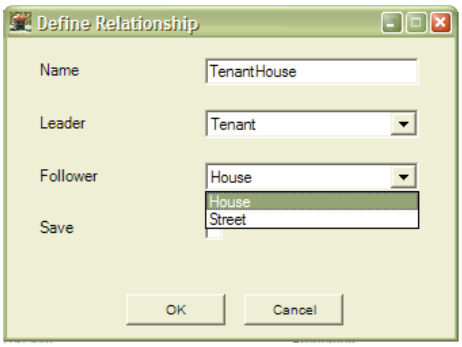


Figure 3. An example of a relationship definition for generalized Schelling model; **Tenant** entity does not appear as a possible Follower in the pull-down list.

Defining behavioral rules

The core of model design lies in defining automata behavior. As previously mentioned, the information necessary for behavior definition is collected via **Assessment Rules**, after which an **Automation Rule** is applied in order to alter automata state. According to the OBEUS concept, any limitations on this stage will constrain the modelers' ability to interpret their understanding of the modeled system. Assessment and behavior rules are consequently formulated as classes' methods via a C# compiler. We base currently on Borland C# Builder. When a C# compiler is activated, all the objects and previously defined assessment rules are available at prompts when constructing new models (Figure 4).

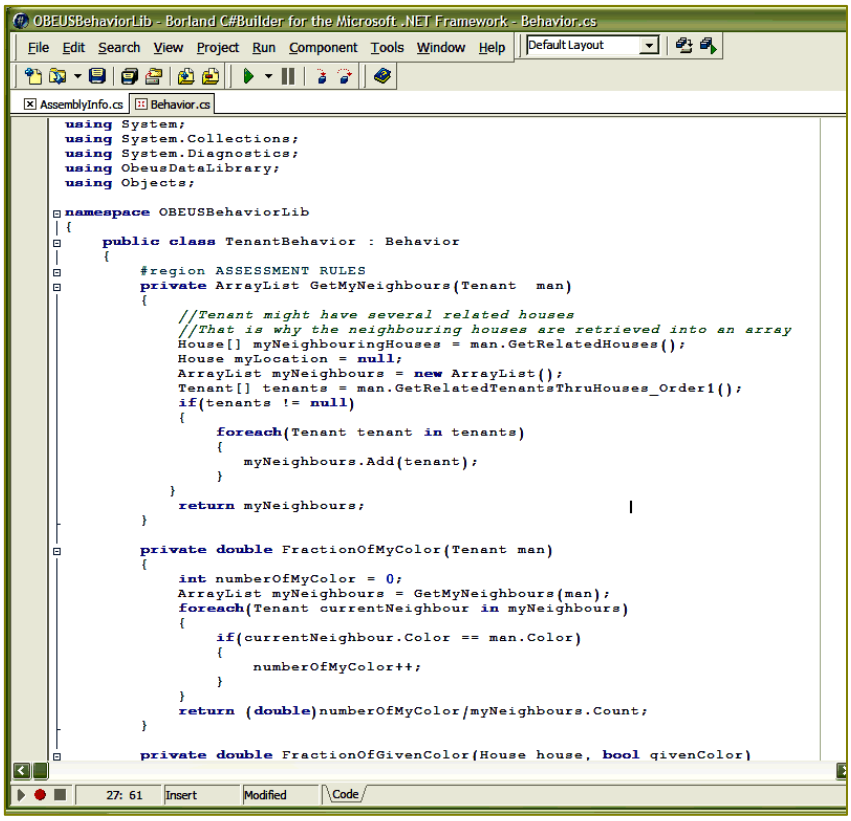


Figure 4. Two assessment rules of the generalized Schelling model as seen in Borland C# window; note examples of the OBEUS methods, automatically constructed when the classes of the objects and relationships are defined - **GetRelatedHouses()** and **GetRelatedTenantsThruHouses_Order1()**

Building a synchronization chart

At the final stage of the model construction, the synchronization mode and the order of updating (at the level of classes) that determines the temporal sequence for applying behavioral rules should be established (Figure 5). Recall that in OBEUS, relationships are always updated asynchronously.

In the sequential mode, either a random or user-defined order should be selected; in the latter case, the attribute that defines the order should be chosen. Populations are selected for updating in the order established by the user in the right-hand window of the dialog.

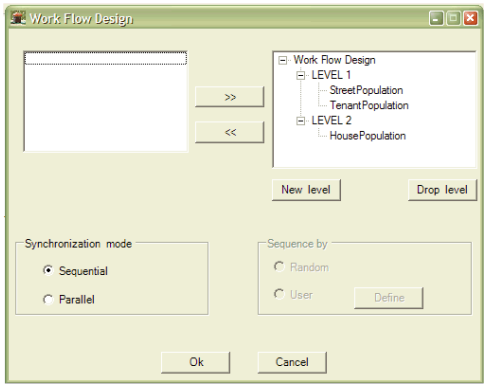


Figure 5. Dialog box for establishing model flow, example of generalized Schelling model.

GAS AND OBEUS AS A COMMON REFERENCE OF OBJECT-BASED MODELING

Several decades ago, urban modeling was born from a common reference point: the system of differential or difference equations that describe changes in aggregate state variables over time. One could distinguish between urban models of this kind by comparing their analytical presentations. With time, the regional framework was abandoned in favor of high-resolution simulations. This evolution, however, revived the problem of a sharable or common reference, which is compulsory if we want to advance urban modeling beyond art to engineering. We argue that the GAS concept together with an OBEUS environment provide this mutual reference: the GAS concept formalizes the view of the city as a system of interacting autonomous objects; OBEUS software provides operational tools for defining the objects, interactions (relationships) between them and the rules of objects' behavior.

The concept of GAS and the OBEUS software are very recent developments, and thorough investigation of their capability and limitations yet should to be done. Our view is that such an investigation should be merely operational – each published urban high-resolution Cellular Automata or Multi-Agent model should be tested whether it can be reformulated in GAS terms and further represented in OBEUS. To demonstrate the utility of OBEUS, we are rebuilding in this line various models, beginning from the basic models of residential dynamics (Schelling 1971) through the constrained CA (White and Engelen 1997) and on towards the deltatron model (Clarke 1997).

BIBLIOGRAPHY

Allen, P. M. and M. Sanglier (1979). "A dynamic model of growth in a central place system." Geographical Analysis **11**(3): 256-272.

Batty, M. (1997). "Cellular automata and urban form: A primer." Journal of the American Planning Association **63**(2): 266-274.

Batty, M. (2003). New Developments in Urban Modeling: Simulation, Representation, and Visualization. Integrated Land Use and Environmental Models. S. Guhathakurta. Berlin, Springer**:** 13-44.

Benenson, I., S. Aronovich, et al. (2004). "Let's Talk Objects: Generic Methodology for Urban High-Resolution Simulation." Computers, Environment and Urban Systems, **Forthcoming**.

Benenson, I., I. Omer, et al. (2002). "Entity-based modeling of urban residential dynamics - the case of Yaffo, Tel-Aviv." Environment and Planning B. **29**: 491-512.

Benenson, I. and P. M. Torrens (2004). Geosimulation: automata-based modeling of urban phenomena. London, Wiley.

Berec, L. (2002). "Techniques of spatially explicit individual-based models: construction, simulation, and mean-field analysis." Ecological Modelling **150**: 55-81.

Booch, G. (1994). Object-oriented analysis and design with applications. Menlo Park, CA, Addison-Wesley.

Brogan, D. C. and J. K. Hodgins (1997). "Group Behaviors for Systems with Significant Dynamics." Autonomous Robots **4**: 137-153.

Chapin, F. S. and S. F. Weiss (1965). Some Input Refinements for a Residential Model. An urban studies research monograph. Chapel Hill, Institute for Research in Social Science, University of North Carolina**:** 68.

Chapin, F. S. and S. F. Weiss (1968). "A Probabilistic Model for Residential Growth." Transportation Research **2**: 375-90.

Clarke, K. C. (1997). Land Transition Modeling With Deltatrons, http://www.geog.ucsb.edu/~kclarke/Papers/deltatron.html. **2002**.

Clarke, K. C., S. Hoppen, et al. (1997). "A self-modifying cellular automata model of historical urbanization in the San Francisco Bay area." Environment and Planning B: Planning and Design **24**(2): 247-261.

Erickson, B. and T. Lloyd-Jones (1997). "Experiments with settlement aggregation models." Environment and Planning B-Planning & Design **24**(6): 903-928.

Flache, A. and R. Hegselmann (2001). "Do Irregular Grids make a Difference? Relaxing the Spatial Regularity Assumption in Cellular Models of Social Dynamics." J. of Artificial Societies and Social Simulation **4**(4): <http://www.soc.surrey.ac.uk/JASSS/4/4/6.html>.

Forrester, J. W. (1961). Industrial Dynamics. Cambridge, Massachusetts and London, England, MIT.

Forrester, J. W. (1969). Urban Dynamics. Cambridge, MA, MIT.

Galton, A. (2001). "Space, Time, and the Representation of Geographical Reality." Topoi **20**: 173-87.

Howe, D. R. (1983). Data analysis for data base design. London, Edward Arnold.

Levine, H. and W. J. Rappel (2001). "Self organization in systems of self-propelled particles." Physical Review E **63**: 208-211.

Lowry, I. S. (1964). A model of metropolis Santa Monica, California, The RAND Corporation**:** 136 p.

Noble, J. (2000). Chapter 6. Basic Relationship Patterns. Pattern Languages of Program Design; In N. Harrison, B. Foote and H. Rohnert, Addison-Wesley**:** 73-89.

Portugali, J. (2000). Self-Organization and the City. Berlin, Springer.

Reynolds, C. (1987). "Flocks, birds, and schools: a distributed behavioral model." Computer Graphics **21**: 25-34.

Schelling, T. C. (1971). "Dynamic models of segregation." Journal of Mathematical Sociology **1**: 143-186.

Semboloni, F. (2000). "The growth of an urban cluster into a dynamic self-modifying spatial pattern." Environment and Planning B-Planning & Design **27**(4): 549-564.

Toner, J. and Y. Tu (1998). "Flocks, herds, and schools: A quantitative theory of flocking." Physical Review E **58**: 4828-4858.

Torrens, P. M. and I. Benenson (2005). "Geographic Automata Systems." International Journal of Geographic Information Science, **forthcoming**.

Vicsek, T., A. Czirok, et al. (1995). "Novel type of phase transitions in a system of self-driven particles." Physical Review Letters **75**: 1226-1229.

White, R. and G. Engelen (1997). "Cellular automata as the basis of integrated dynamic regional modelling." Environment and Planning B-Planning & Design **24**(2): 235-246.

White, R. and G. Engelen (2000). "High-resolution integrated modelling of the spatial dynamics of urban and regional systems." Computers, Environment and Urban Systems **24**(5): 383-400.

Zeigler, B. P., H. Praehofer, et al. (2000). Theory of modeling and simulation:Integrating Discrete Event and Continuous Complex Dynamic Systems. New York, Academic Press.