Interactive Design of Urban Spaces using Geometrical and Behavioral Modeling

Carlos A. Vanegas[1] Daniel G. Aliaga[1] Bedřich Beneš[1] Paul A. Waddell[2]
[1]Purdue University [2]University of California, Berkeley

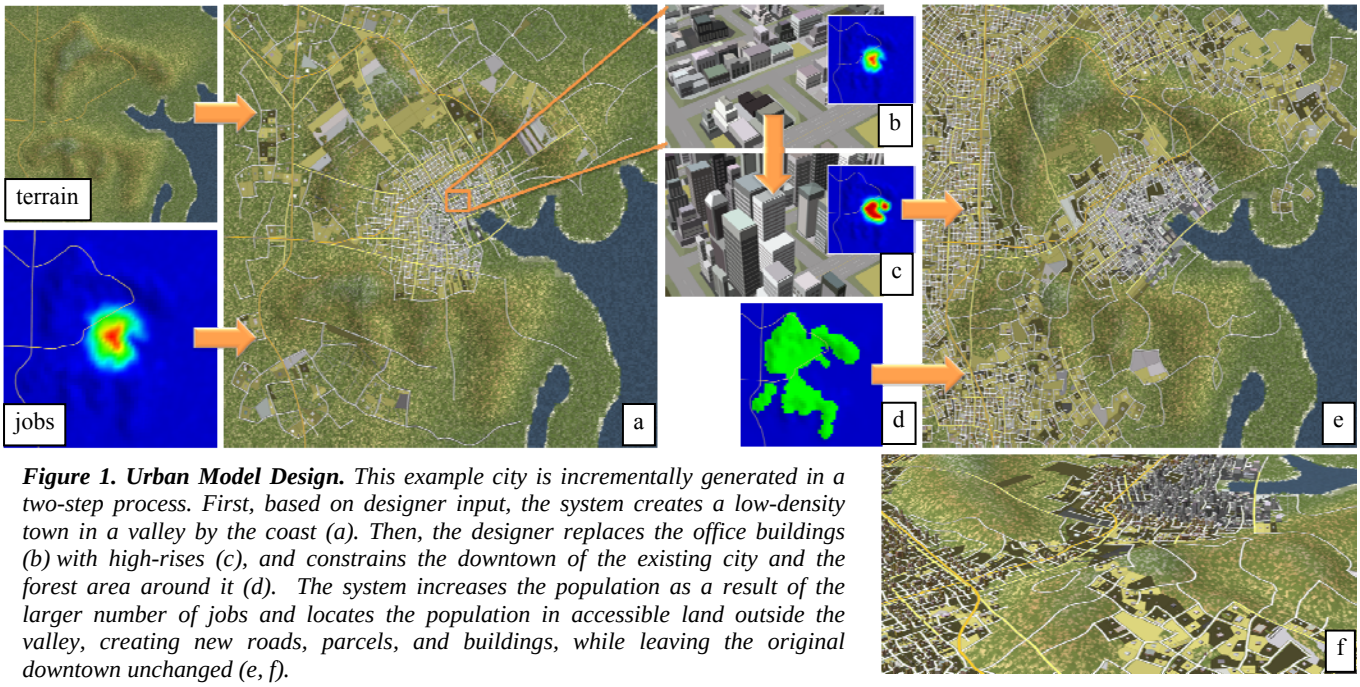

Figure 1. Urban Model Design. *This example city is incrementally generated in a two-step process. First, based on designer input, the system creates a low-density town in a valley by the coast (a). Then, the designer replaces the office buildings (b) with high-rises (c), and constrains the downtown of the existing city and the forest area around it (d). The system increases the population as a result of the larger number of jobs and locates the population in accessible land outside the valley, creating new roads, parcels, and buildings, while leaving the original downtown unchanged (e, f).*

Abstract

The main contribution of our work is in closing the loop between behavioral and geometrical modeling of cities. Editing of urban design variables is performed intuitively and visually using a graphical user interface. Any design variable can be constrained or changed. The design process uses an iterative dynamical system for reaching equilibrium: a state where the demands of behavioral modeling match those of geometrical modeling. 3D models are generated in a few seconds and conform to plausible urban behavior and urban geometry. Our framework includes an interactive agent-based behavioral modeling system as well as adaptive geometry generation algorithms. We demonstrate interactive and incremental design and editing for synthetic urban spaces spanning over 200 square kilometers.

Keywords: interactive, editing, 3D models, urban spaces.
CR Categories: I.3 [Computer Graphics], I.3.3 [Picture/Image Generation], I.3.5 [Computational Geometry and Object Modeling], I.3.6 [Methodology and Techniques].

1. INTRODUCTION

We present a framework for intuitive and interactive design of 3D geometric models of large, complex, and realistic urban spaces. An *urban space* is a collection of architectural structures arranged into buildings, parcels, blocks, and neighborhoods interconnected by roads. The key notion behind our approach is to close the loop between behavioral modeling and geometrical modeling of urban spaces. We model the design and editing process as a dynamical system using a set of functions that describe the change the variable values. Our system produces models resembling existing cities, and is useful for a variety of applications ranging from games and movies to urban planning and emergency management.

Previous research in urban modeling can be divided into the areas of *geometrical modeling* and *behavioral modeling*: the first is purely computer graphics oriented (e.g., [Parish and Muller 2001, Wonka et al. 2003, Mueller et al. 2006, Aliaga et al. 2008, Chen et al. 2008]), and the second lies outside this research domain (e.g., [Alkheder et al. 2008, Waddell 2002]). The results of urban behavioral modeling are intended for decision-making regarding urban policies in current and future urban areas. In general, however, behavioral simulation models use limited and fixed 2D geometric features (e.g., grid cells or parcels) and are computationally too expensive to run at interactive rates. Some research has been performed in feeding the output of a behavioral modeling system into a geometrical modeling system producing 2D layouts or 3D models that change over time (e.g., [Honda et al. 2004, Vanegas et al. 2009, Weber et al. 2009]). However, the focus is not on designing and editing a new urban model, but rather on computing changes (e.g., growth) over time to a provided model.

Geometrical and behavioral modeling, when applied in isolation on the same underlying urban space, yields a functional disconnection between the two resulting models. This is because behavioral modeling is not concerned with generating the geometry of the urban space and because geometrical modeling usually does not consider behavioral properties of real-world cities. This disconnection is particularly disadvantageous during the computer graphics design and editing process of 3D geometric urban models. Real-world urban spaces exhibit relationships between variables such as terrain, road networks, building shapes, distribution of population, jobs, and transportation routes. Existing geometrical modeling systems can be used to perform the changes to the 3D model that are implied by altering these variables, but the designer would have to be aware of the subtle interdependencies between the variables. The consequences of changing one variable can vary from local to global changes. Consider the following examples:

- a designer increases the height of the buildings in a downtown area; as a consequence, the number of jobs and the population increase; this should result in more local streets and residential buildings appearing in another part of the city that is conveniently accessible from the newly altered downtown;
- a designer inserts a highway into the model; as a consequence, the accessibility between different parts of the urban space may change, hence the location of jobs, houses, and buildings may have to be drastically altered; and
- a designer modifies a subset of the terrain or increases the local population; as a consequence, the geometric configuration of the local neighborhood of streets, city blocks, parcels, and buildings must be updated to accommodate either the modified terrain slopes or the increase in population.

Ignoring these changes may result in models that do not resemble real-world urban spaces such as tall buildings on top of mountains, cities with only skyscrapers and no residential areas, or an imbalance between road networks and buildings.

1.1 Key Inspiration

Our key inspiration is that we can reduce the design and editing time of 3D models of urban spaces and produce plausible urban models by tightly coupling behavioral modeling and geometrical modeling. This is in spirit similar to that performed for large crowd modeling (e.g., [Sung et al. 2004, Treuille et al. 2006]) and for flocking and animal behavior modeling (e.g., [Reynolds 1987, Tu and Terzopoulos 1994]). Using a dynamical system, we continually alter the model being interactively designed so as to conform to plausible urban behavior and urban geometry, enabling the production of large models in just a few seconds (Figure 1).

Traditional urban behavioral modeling simulates the temporal evolution of an urban space. The behaviors are the attempts to reach a state of dynamic equilibrium; i.e., a status of internal consistency between the demands of the population, job market, transportation routes, and building structures. Our goal is a computer graphics design process that maintains a state of static equilibrium between the variables of an urban space. We exploit the understood concepts in urban planning and simulation to create a behavior that is internally consistent with the current geometry of the urban model and, vice versa, we use adaptive geometric algorithms to generate geometry that satisfies behavioral demands.

1.2 Overview

Our generalized design framework consists of urban space design variables with spatially-varying values defined over an urban area. The variables control the distribution of population and jobs, land values, road network, parcel shape, and building geometry. Editing the variables is performed intuitively and visually using a graphical user interface (GUI) combined with a paint-brush style tool. Any variable can be globally or locally increased, decreased, or constrained. The ability to constrain variables is crucial to support incremental design and editing at various scales. Moreover, the designer can constrain several distinct parts of an existing model and let the system complete the rest of the model. The dependencies between variables are articulated via differential equations. After an edit operation, the dynamical system attempts to bring the urban model back into equilibrium (Figure 2).

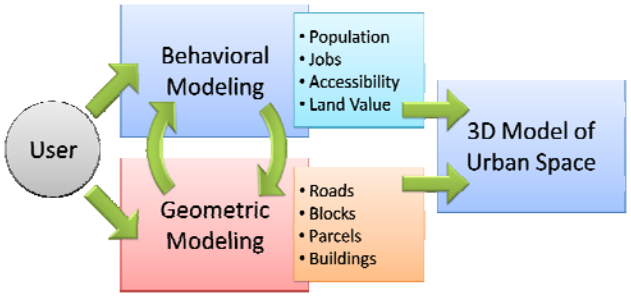

Figure 2. System Pipeline. *The user interactively modifies behavioral and/or geometrical variables of the urban space. A dynamical system continually recalculates the model to conform to plausible urban behavior and geometry.*

Our behavioral modeling is inspired by the urban simulation framework, UrbanSim, of Waddell [2002] in the sense of using an agent-based method and letting agents make individual choices. We reformulate their methodology with the objective of obtaining static equilibrium in the urban space. Their framework is a growth-oriented forward simulation for offline use that does not refresh its use of geometry during the simulation and does not produce detailed geometry as output. Our objective is not to capture the numerous behavioral nuances and interacting agents, nor to model the evolutionary processes that shape cities over time. Rather, we aim to develop a system that provides a designer with a very efficient and parsimonious means to create plausibly realistic city-scapes that are internally consistent. For this objective, we take an approach of simplifying the behavioral aspects that are well studied in urban planning, geography, economics and civil engineering, and provide instead a fast implementation that even creates large multi-city regions fully detailed in their composition of roads, parcels, buildings, jobs and population.

Our geometrical modeling engine consists of an adaptive road network generator, parcel generator, and building generator. Parcels are computed inside the blocks defined by a road network. Building envelopes are procedurally generated and their size is estimated from the population and jobs count that they are to contain. We use 3D models of intermediate complexity for interactive visual feedback during the design process, but support exporting our design variables for interfacing other software systems focused on creating photorealistic buildings and façades.

We demonstrate our system by interactively creating large models of urban spaces and modifying the model. Our system has designed urban models containing up to 50,000 buildings, 3,000 km of roads, and 200 km^2 of area. The total interactive design process time, including several iterations of variable changes and modeling alterations, is just a few minutes on a standard desktop computer.

1.3 Contributions

The main contributions of our work include

- a methodology for interactive urban model design that closes the loop between behavioral and geometrical modeling, by iteratively solving a dynamical system after any editing

operation to bring the design back to equilibrium (i.e., to internal consistency and to plausibility),

- an interactive agent-based behavioral modeling system which simulates population and job changes, lets agents make choices, and uses continually changing geometric data, and
- a road generation algorithm that adapts to the underlying population, jobs, terrain, and local transportation demand.

2. RELATED WORK

Our work relates to research within the areas of urban modeling and urban planning and simulation. One of the first systems to generate entire virtual cities was presented by Parish and Mueller [2001]. The user provides several types of input layers for describing an underlying environment (e.g., a fixed population and terrain map) and a set of custom-written grammars based on environmentally-sensitive L-systems [Měch et al. 1996] are used for generating street layouts and simple buildings. Their work does not include behavioral modeling, but indicates as desirable future work the inclusion of behavioral schemes in the road creation mechanism. Further, a single direction of communication is provided between input layers and generated results (i.e., the generated 3D model does not alter the input layers) and there is no inter-dependency between the input layers. Although layers can be manually changed for simple inter-dependencies, it must be known to the designer and does not scale well to large models.

Subsequent research has focused on improving particular aspects of city generation. For example, the process of generating a road network through the interactive manipulation of tensor fields was recently addressed by Chen et al. [2008]. In our previous work [Aliaga et al. 2008], we used an example-based approach to produce a layout of roads and to complete it with imagery synthesized from aerial photographs. While these methods can produce a variety of road configurations, the ad hoc design processes do not consider both the distribution of population and jobs and the accessibility between different parts of the model. Such consideration requires manual editing (for [Chen at al. 2008]) or many well-chosen example inputs (for [Aliaga et al. 2008]).

Various authors have focused on detailed modeling of either individual buildings or their parts (e.g., [Lipp et al. 2008, Merrell and Manocha 2008, Mueller et al. 2006, Mueller et al. 2007, Wonka et al. 2003]). CityEngine [Procedural 2009] implements some of the previous algorithms in a commercial software system.

Urban planning and simulation focuses on behavioral modeling (e.g., [Alkheder et al. 2008, Leonard et al. 1998, Portugali 2000, Waddell 2002]). As opposed to shorter term simulations of traffic flow and crowds in 3D environments, our use of the term behavioral simulation refers to modeling long term behaviors of an urban space based on socio-economic factors. The dominant methods for urban simulations are cellular automata [Leonard et al. 1998], agent-based models [Portugali 2000], and aggregate equilibrium models [Putman 2000]. In addition, micro-simulation models attempt to simulate decision-theoretic individual agents responding to both localized and broader environments [Waddell 2002] and leverage the Random Utility Maximization framework to provide the behavioral and statistical theory consistent with decision-theoretic agents [McFadden 1973, Train 2003]. In general, these simulation systems model an urban environment as a set of interconnected processes and are computationally expensive. Micro-simulation discrete choice models, such as UrbanSim [Waddell 2002], contain agents that make decisions to locate and move within the urban environment. This method works with very small cells or parcels, but it differs from others by integrating discrete choice models, an explicit representation of real estate markets, and statistical methods to estimate model parameters and to calibrate uncertainty in the model system. Our work leverages off UrbanSim (Section 3.2).

Some recent works model the change of an urban space over time. In Vanegas et al. [2009], we used the output of an offline behavioral simulation to make changes to current aerial imagery and produce tentative 2D aerial views of the urban space in the future – no feedback to the urban simulation is performed. Lechner et al. [2007] describe an offline agent-based temporal simulation to generate realistic 2D land-use maps based on a user-sketched global behavior, but their system does not provide tools for editing and designing 3D geometric models of urban spaces. Weber et al. [2009] presented a geometric simulation of a city over time (e.g., 25 years). Their interactive simulator focused on urban growth. However, their simulation is a forward computation without feedback. For instance, the alteration of building heights by the user does not modify the distribution of population and jobs, large changes to the road network based on designer input for altering agent behavior are not shown, and completion of urban models from a set of constrained design variables is not implemented. Furthermore, their typical total design and simulation time is between one hour and one day. In contrast, our system focuses on providing tools to design urban models from scratch and to yield complete models within minutes.

3. URBAN SPACE DESIGN

3.1 Design as a Dynamical System

Our system consists of N urban space variables defined over a 2D spatial domain. Each variable is sampled over a 2D spatial grid G of size $W \times H$. We use v_k to denote the values of the k-th state variable throughout the entire spatial grid, and $v_k(i,j)$ to denote the value of the k-th state variable at grid cell (i,j), for $k \in [1,N]$, $i \in [1,W]$, and $j \in [1,H]$. We represent the change of each variable by the differential equation $\dot{v}_k(i,j) = f_k(v_1, v_2, \ldots v_N)$. Since each variable is potentially dependent on the values of every other variable at all grid cells, each differential equation is generally a function of all state variables. While the derivatives of other variables could also be used as independent variables in each differential equation, in practice our current formulation is accurate enough for our interactive design purposes.

If a variable v_c, for some value $c \in [1,N]$, is changed by the user, the system iteratively updates the other variables in order to return to a state of equilibrium. We say an urban model is in equilibrium when for a small convergence threshold ε we have

$$|\dot{v}_k(i,j)| \leq \varepsilon. \quad (1)$$

for all k, and where the variables are assumed to be normalized.

The corresponding iterative system can be written as

$$v_k^{n+1}(i,j) = v_k^n(i,j) + \dot{v}_k(i,j) \quad (2)$$

where $n \geq 0$ is the iteration count, $k \neq c$, and v_k^0 are the initial values of the system assumed to already be at equilibrium..

Equations of the form of (2) can be solved using Euler's method, or other techniques for ordinary differential equations. But, in our problem domain v_k and $\dot{v}_k$ are difficult to express symbolically, thus computing v_k^{n+1} in symbolic form is hard. Furthermore, the total number of variables is very large; e.g., an urban area of only 10×10 kilometers typically consists of 100×100 cells, each of 100×100 meters, and amounts to $10000N$ variables with very intricate and wide spread dependencies.

Instead, our work proposes algorithms (or symbolic equations when possible) for efficiently computing v_k's and $\dot{v}_k$'s (Sections 4 and 5), thus enabling an iterative solution for the above dynamical system. Our framework can be extended to include any number of

variables as long as the equivalent of a differential equation is provided. The stochastic nature of some of our algorithms prevents the system from returning to the exact same equilibrium state. In our experiments, our approach shows stability in the sense of converging to a similar equilibrium state from nearby states with perturbed variable values (Figure 6). Oscillations, which are often present in high-dimensional dynamical systems, are not generated in practice because we ensure all state variables stay within their respective range of reasonable values and because the step size for all variables is small from iteration to iteration.

3.2 Behavioral Modeling

Waddell and Ulfarsson [2004] define as major components of an urban simulation a set of algorithms which are applied to a spatially distributed set of variables representing population and jobs. These components (algorithms) are i) *transition:* adds or removes population/jobs from the urban system based on interfaces with an exogenous macroeconomic model or an endogenous model, ii) *mobility:* predicts that a subset of the population/jobs will move from their current location due to a change in living conditions (e.g., marriage, new child, job change, etc.), iii) *location choice:* predicts the places to where population/jobs, that have chosen to move, will move, and iv) *real-estate developments:* imitates developers who change land use due to economic investments or changes in governmental regulations.

Based on Waddell and Ulfarsson [2004], we select a minimal set of design variables (i.e., specific instances of the $v_k(i,j)$'s):

- *population count* $p(i,j)$, corresponding to the number of people who live in the grid cell, and
- *job count* $b(i,j)$, referring to the number of jobs (total employment) in the grid cell.

In addition, we also use per grid cell values for accessibility $a(i,j)$ and land value $l(i,j)$. While these values could be user-specified parameters, we found them less intuitive and thus only provide algorithms to compute them automatically (Section 4.2).

The components of Waddell and Ulfarsson [2004] are mapped to our dynamical system and implemented as new algorithms with interactive update rates. Transitions and real-estate development are performed by the designer. Mobility and location choice are the primary way that our dynamical system alters its state in order to reach equilibrium. Consider a designer who directly changes the population and/or jobs count of one or more grid cells. The change will alter accessibility and land-value which then changes the desired spatial distribution of population and jobs and, consequently, the geometric model. If the change is sufficiently large, it will cause the difference in equation (1) to exceed ε. Then, the mobility and location choice algorithms will be executed by the dynamical system, and the population and jobs will be redistributed until again reaching a state of equilibrium. The condition in equation (1) terminates the iterations once the only change is due to random mobility changes which no longer significantly alter the spatial distribution of the variables. These two algorithms define the differential equations for either population or jobs (Section 4.1).

3.3 Geometrical Modeling

We define a set of geometric variables for which relations with behavioral variables can be established. Although variables are stored in grid cells, geometric structures are freely positioned and can cross cell boundaries. For a grid cell (i,j) we define

- *roads length* $r(i,j)$, the total length of the roads inside the grid cell,
- *average tortuosity* $\tau(i,j)$, the mean ratio between the road segment length and the distance between the road segment endpoints for the roads inside the grid cell,
- *building volume* $m(i,j)$, the total volume of interior space within the building structures of the grid cell, and
- *terrain elevation* $h(i,j)$, the average elevation of the terrain inside to the grid cell – in our current work, this variable is only under user control.

Furthermore, parcel size and land use are other design variables but are computed algorithmically (see Sections 5.3 and 5.4).

The differential equations for geometric variables can be succinctly expressed as symbolic expressions and such is given in Section 5. However, unlike behavioral modeling, calculating the value of the geometric variables requires generating the exact geometric model. In Sections 5.2-5.4, we describe our demand-based generating functions for geometrical modeling.

3.4 Variable Specification

The designer guides the editing process by interactively specifying variable values. Analogous to image painting, a brush tool enables increasing, decreasing, constraining, or setting the values of spatially-varying variables using mouse-driven strokes. Highways are sketched by the designer rather than produced by our dynamical system because their configuration is often subject to a very high-level decision process.

4. BEHAVIORAL MODELING

4.1 Differential Equations

The grid cell values for the behavioral differential equations are updated using an agent-based framework augmented with a discrete choice modeling approach. In our system, population or jobs are either changed directly by the designer or changed indirectly (e.g., road density is reduced, which then should cause a decrease in population). A directly modified variable will cause the accessibility and land values to change which in turn will alter the spatial probability distribution of the other behavioral variables; in contrast an indirectly changed variable will demand a new spatial distribution for population and for jobs. In both cases, our agent-based mobility and location choice algorithms will re-distribute the population and jobs according to the new desired probabilities.

4.1.1 Mobility Algorithm

Our mobility algorithm moves a subset of the agents from their current grid cell. The subset is of size $S = \alpha E$ where $\alpha \in [0,1]$ is a small fraction, $E = \sum_i \sum_j e(i,j)$, and $e(i,j)$ is either $p(i,j)$ or $b(i,j)$. The subset of agents is chosen at random and is placed in a pool Q of unlocated agents. If the designer wishes to increase the total number of agents, the count is simply added to Q. If instead the designer wishes to decrease the total number of agents, they are selected at random and discarded prior to the mobility algorithm.

4.1.2 Location Choice Algorithm

The agents in Q are placed in the grid by a location choice algorithm. The selection of the location depends on the current accessibility and land values. In Waddell [2002], multinomial logit models are used to implement discrete location choice. For our objective of interactive urban design, we found it sufficient, as well as efficient, to use a weighted attractiveness measure which is then translated to a probability by normalizing it with the sum of the attractiveness measure across all locations in a Monte Carlo sampled set. This approximates the discrete choice models by substituting in a simpler approximation of the utility function. In particular, for each agent $e_s \in Q$, where $s \in [1,S]$, we choose a small random subset of size $T = \alpha WH$ grid cells (where α is the same small fraction used by the mobility algorithm), namely

$\{(i_t, j_t)\}$ where $t \in [1, T]$. The probability q_{st} that e_s will be located at a cell (i_t, j_t) is given by

$$q_{st} = (w_a a_t + w_b l_t)/T_s \quad (3)$$

where w_a and w_b represent relative importance weights for the accessibility $a_t \in [0,1]$ and land value $l_t \in [0,1]$ of cell (i_t, j_t), and $T_s = \sum_t q_{st}$ is the sum of the q_{st}'s for the current agent. A random number generator with uniform distribution is used to determine the grid cell (i_t, j_t) to which e_s will be located.

4.2 Estimating Accessibility and Land Value

The accessibility $a(i,j)$ and land value $l(i,j)$ of grid cell (i,j) is calculated using a logistic function. Accessibility $a(i,j)$ is a measure of the access that grid cell (i,j) has to jobs and to the rest of the population. An intuitive behavior for accessibility is that it decreases with the slope of the terrain and increases with improved connectivity to roads, population and jobs, water fronts and river banks [Ortúzar and Willumsen 2001]. Good connectivity can be implied by geometric closeness or good access via the transportation network (e.g., highway).

To represent accessibility, we use the logistic function

$$a(i,j) = \frac{1}{1 + e^{-z(i,j)}} \quad (4)$$

where $z(i,j) = \beta_0 + \beta_1 u_1(i,j) + \beta_2 u_2(i,j) + \beta_3 u_3(i,j)$. The variable u_1 is an estimate of proximity of grid cell (i,j) to local roads weighted by their relative importance and distance,

$$u_1(i,j) = w_h dist(R_H) + w_r dist(R_A) + w_s dist(R_S) \quad (5)$$

where w_h, w_r, and w_s are the weights of relative importance of the normalized distance measures, $dist(X)$ returns distance from the grid cell center to the closest segment of X, where X can be one of highways R_H, avenues R_A, or streets R_S, u_2 represents the local slope of the terrain, and

$$u_3(i,j) = \sum_r \frac{D_{i_r j_r}}{d_{ij,i_r j_r}} \quad (6)$$

is a distance normalized measure of the activity level at (i,j) based on a sampling of R neighboring grid cells $\{(i_r, j_r)\}$ where $r \in [1, R]$. The value $d_{ij,i_r j_r}$ is the distance from grid cell (i,j) to grid cell (i_r, j_r). $D_{i_r j_r}$ is a measure of the activity level at grid cell (i_r, j_r) and is computed as the sum of $p(i_r, j_r) + b(i_r, j_r)$. The empirically determined constants $\beta_o, \beta_1, \beta_2,$ and β_3 provide the relative importance of the factors that affect accessibility. Land value $l(i,j)$ is calculated using a logistic function similar to (5).

Figure 3 shows how by quickly changing job distribution, the system automatically creates a new population center including its roads, parcels, and buildings connected to the original town.

5. GEOMETRICAL MODELING

5.1 Differential Equations

Our system reacts to per grid cell changes of geometric variables by using adaptive algorithms to generate geometry. The differential equations for determining per grid cell geometric variable changes are easy to write symbolically. However, obtaining the actual variables values for a just-changed set of geometric variables is challenging. Simply reusing the variable value computed in a previous iteration of the dynamical system is not appropriate because it does not necessarily represent the variable value of the current geometry - it may differ because the geometric model consists of discrete structures and its generation may be subjected to constraints imposed by the designer.

The differential equations in (1) can generically be written as

$$\dot{v}^{n+1}(i,j) = v^{n+1}(i,j) - v^n(i,j) \quad (7)$$

where $\in \{r, \tau, m\}$. For the $(n+1)$-th step, the target variable values v^{n+1} and the current variable values v^n are used.

The target variable values for r, τ, m are computed as follows: the target total length of the roads inside a grid cell (i,j) needed to accommodate its population $p^n(i,j)$ and jobs $j^n(i,j)$ is

$$r^{n+1}(i,j) = \min\left(w_{pr} p^n(i,j) + w_{br} b^n(i,j), r_{max}\right) \quad (8)$$

where w_{pr}, w_{br} represent the mean road length that one household and one job require for transportation, respectively, and r_{max} is the maximum road length inside a grid cell. The target total building volume in a grid cell (i,j) is

$$m^{n+1}(i,j) = w_{pm} p^n(i,j) + w_{bm} b^n(i,j) \quad (9)$$

where w_{pm} represents the mean building volume necessary to shelter one household, and w_{bm} is the mean building volume to allocate one job. The target tortuosity inside a grid cell (i,j) is

$$\tau^{n+1}(i,j) = 1 + k\left(1 - \frac{p^n(i,j) + b^n(i,j)}{p_{max} + b_{max}}\right) \quad (10)$$

where k is a user-controlled parameter that scales the global tortuosity, and p_{max} and b_{max} are respectively the maximum number of people and jobs per grid cell.

5.2 Generating Arterials and Streets

To generate the roads that correspond to per grid cell road length values $r(i,j)$, we create a network of roads that connect and span the main population clusters in the urban space. Adapting from [AASHTO 2004], we classify roads into three types based on their size: (i) highways, which are the highest capacity roads with limited access and carry inter-city traffic and, in large cities, intra-city traffic; (ii) arterials, which are lower capacity than highways, carry intra-city traffic and connect to local streets and to highways; and (iii) streets, which are local roads that route traffic from

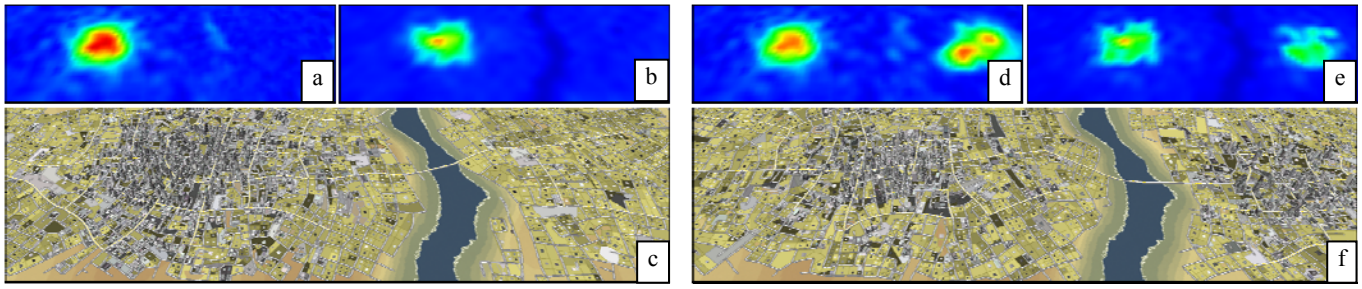

Figure 3. Example Behavioral Modeling. *Changing jobs distribution can be used to produce new plausible geometries. An initial city is located on the left side of a river (c) and has the job distribution shown in (a) and the population distribution shown in (b). A new employment area is sketched on the right side of the river (d). The system then increases and relocates population to meet the new job offer (e), and a second city is built on the right side of the river (f).*

arterial roads to individual parcels. First, in order to connect the main population clusters, a set of seeds are generated considering the population and jobs distribution and the location of designer-sketched highways. Each seed is converted to an intersection of the arterial roads network and is used to generate arterial road segments. Second, street seeds are generated along arterial road segments and used to create streets. The expansion of both streets and arterials is guided by the terrain, the spatial distribution of population and jobs, and user-specified style parameters.

Figure 4 shows how sketching a highway can produce a more widespread city. The designer draws a new highway and keeps the total population and jobs constant. The system determines that the new highway increases accessibility from the rural area to the downtown. Then, population moves to now accessible lower land-value areas and new roads and buildings are adaptively generated.

5.2.1 Observations and Assumptions

Our road generation method is based on the following observations about real-world roads. (a) Road networks are designed and built to meet a transportation demand by the population [Montes de Oca and Levinson 2006]. The capacity of a road, reflected by its width and the mean distance between its consecutive intersections, responds to such a demand. (b) Road networks exhibit a variety of styles which are difficult to be solely inferred from behavioral and geometrical parameters. While highways are usually designed to minimize travel distances, arterials and streets are more affected by historical and aesthetic factors.

We select a set of design parameters sufficiently expressive to represent a wide range of observed patterns (e.g., Figures 5 and 8). Our road generation algorithm uses the following key assumptions:

- the predominant patterns of arterials and streets are *grid* style and *radial* style with spurious occurrences of dead-ends,
- in the grid style, up to four nearly-perpendicular segments depart from each intersection point,
- in the radial style, three or more road segments depart from some intersection points at equally spaced angles, and
- the road pattern and its tortuosity is affected by the nearby population and jobs.

5.2.2 Seed Generation Algorithm

To obtain a set of κ seeds for generating arterial roads, we group grid cells using a weighted k-means clustering algorithm. The value of κ is a user-specified constant set by default to $\gamma\sqrt{\sum_{ij}(p(i,j)+b(i,j))}$, where $\gamma \leq 1$ is a small constant. We let s_u, for $u \in [1,\kappa]$, be the center point of a cluster K_u to be determined. The clustering algorithm uses $w_u = p(i_u,j_u) + b(i_u,j_u)$ as grid cell weights in order to pull the cluster centers towards areas of larger population and jobs, though they will still be connected with sparsely populated clusters. Further, we denote the center of a grid cell (i,j) by x_{ij}. Thus, the algorithm searches for a set $S = \{s_1, s_2, \ldots, s_\kappa\}$ that minimizes the expression

$$\sum_{u=1}^{\kappa} \sum_{(i,j)\in K_u} w_u\left(\|x_{ij} - s_u\|\right)^2 \qquad (11)$$

The set S is augmented with seeds that are created on previously existing roads. When generating arterials, the seeds are created on highways (if they exist). In this manner, arterial roads are also connected to the highway network. After the arterial roads are generated, we create seeds along them for the street expansion. In both cases, the distance between two consecutive seeds along the highway/arterial is inversely proportional to the amount of population and jobs in nearby grid cells.

5.2.3 Expansion Algorithm

Starting at the previously computed seeds S, we generate road segments using a breadth-first expansion method. All pre-computed seeds are placed into a pool P. The first seed s_u is removed from P and an attempt is made to create road segments in several directions around the seed. A new seed s_v is created at the end of a newly created piecewise linear road segment C_{uv} provided no previously existing seed is nearby. The new seeds are added to P and the process repeats until the pool is empty. The set of resulting C_{uv} collectively form the road network R.

A seed s_u has m departing directions $\Theta_u = \{\theta_0, \theta_1, \ldots, \theta_{m-1}\}$ along which new road segments can be generated. The value of θ_k, for $k > 0$, is given by $\theta_k = \theta_0 + \frac{2\pi k}{m} + \epsilon$, where ϵ is a random variable with distribution $\epsilon \sim N(0, \sigma^2)$, σ is a small constant, and θ_0 is a reference angle. The reference angle is equal to the orientation of the road segment to which the seed is attached.

For an urban area, the user chooses either a grid style or a radial style road pattern. The choice affects the number of departing directions for the seeds: for grid style, $m = 4$ and for radial style, $m \geq 3$ for the initial seeds and $m = 4$ for all later seeds.

The road expansion for a seed s_u, in direction θ_k, consists of evaluating a piecewise linear curve integral from s_u to a point s_v, using numeric integration of a function $p(x)$. The function $p(x)$ measures the population and jobs in the grid cells located within a small distance of x. The integral is given by

$$\int_{C_{uv}} p(x)dx = \rho. \qquad (12)$$

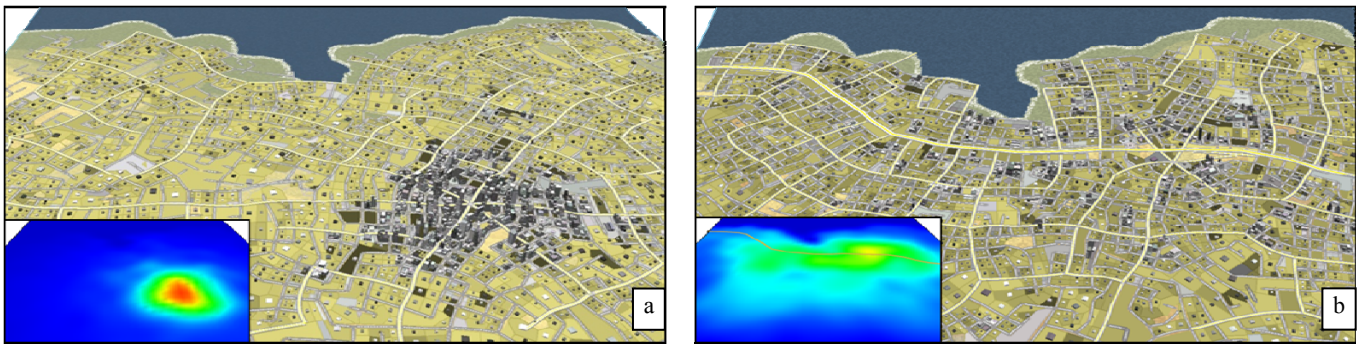

Figure 4. Example Geometrical Modeling. *The designer wishes to produce a more widespread city. The population is tightly gathered around a downtown (a). The user draws a new highway and, as a result, the population redistributes along the highway, the downtown density decreases, and new roads, parcels, and buildings are automatically created (b).*

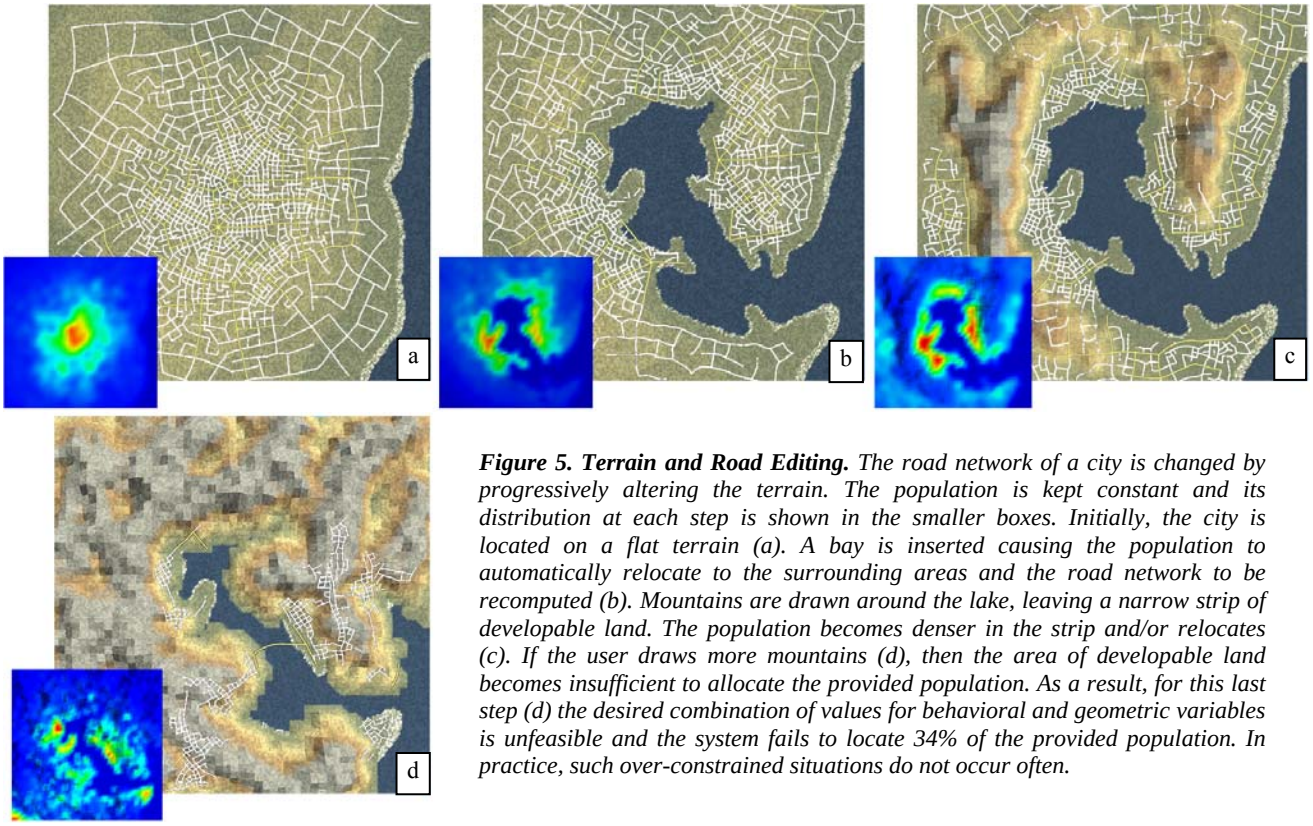

Figure 5. Terrain and Road Editing. *The road network of a city is changed by progressively altering the terrain. The population is kept constant and its distribution at each step is shown in the smaller boxes. Initially, the city is located on a flat terrain (a). A bay is inserted causing the population to automatically relocate to the surrounding areas and the road network to be recomputed (b). Mountains are drawn around the lake, leaving a narrow strip of developable land. The population becomes denser in the strip and/or relocates (c). If the user draws more mountains (d), then the area of developable land becomes insufficient to allocate the provided population. As a result, for this last step (d) the desired combination of values for behavioral and geometric variables is unfeasible and the system fails to locate 34% of the provided population. In practice, such over-constrained situations do not occur often.*

The integration stops when the total value of the integral is $\rho = \alpha r(i,j)$, where α is the desired population/job density to road density ratio. The location where the integral stops is denoted s_v.

The effect of this integration is to expand the road from s_u, with an initial direction θ_k, until the road has covered a sufficiently large amount of population and jobs. This implies that in an area of high population and/or jobs, the integration will terminate and generate an intersection (i.e., seed) at a shorter distance leading to a denser road network. Therefore, the road network is adaptively generated based on population and jobs.

5.2.4 Road Geometry Creation Algorithm

To determine the road geometry of C_{uv}, we choose the amount of smoothly varying curvature in C_{uv} by using the tortuosity parameter τ, perform adaptations based on the terrain, and produce several types of intersections. For a point s_u that is initially being expanded in direction θ_k, the next point along C_{uv} is calculated by advancing a step Δt in direction θ_k. The process is repeated until the integration completes. To introduce smoothly varying curvature, the direction θ_k is updated by $\theta_k = \theta_k + \Delta\theta$, where $\Delta\theta = \lambda \cdot \Delta\theta + \mu$. μ is a random sample from a uniform distribution $[-1, +1]$ and λ is a small fraction.

The expansion of the road geometry adapts to the terrain elevation and the presence of water bodies. When the vertical slope of a next step in the road integration process exceeds a threshold, the algorithm attempts to find a new direction of lesser slope. If such a direction does not exist, the road becomes a dead-end. Roads entering a water body are processed depending on the road type. Arterials may lie on the water, mimicking bridges. If the road segment corresponds to a street, it becomes a dead-end.

The tortuosity of the road patterns is determined by the underlying terrain or by style decisions related to land use (e.g., grid-style roads in dense commercial areas and organic/curvy roads in low-density residential areas). To attempt to imitate real-world tortuosity, our system automatically changes tortuosity based on the aforementioned criteria. Further, since real-world road networks exhibit less regular geometric features (e.g., sporadic dead-ends, "T" and "L" intersections), our algorithm introduces such features randomly by ignoring some expansion directions.

As an example, Figure 5 shows an initial city with radial-style roads (Figure 5a) being progressively transformed by large terrain changes. After each edit, our system automatically redistributes the population, based on changes in accessibility, and infers a new road network density and tortuosity to accommodate the population and jobs (Figures 5b-c). In particular, note the variation in tortuosity and road density as per the aforementioned guidelines. The figure also shows that upon specifying too many constraints (e.g., a terrain unable to house a given population), the system makes a good attempt to find a suitable urban configuration. Such a configuration is not always possible (Figure 5d) and, in this example, a subset of the population cannot be located in the city.

5.3 Generating Parcels

City blocks are defined as closed simple paths of the road network inside of which are parcels of land. The number of parcels inside each block is a function of the longest axis of the oriented bounding box of the block and the total count of population and jobs contained within the grid cells intersected by the block. For low-density areas the number of parcels increases approximately linearly with density; for higher-density areas, once a minimum parcel area has been reached, building height starts to increase. Given the desired number of parcels and the block contour, our method geometrically partitions the city block (e.g., as in Parish and Mueller [2001] or Aliaga et al. [2008]) into parcels.

5.4 Generating Building Envelopes

We procedurally generate building envelopes that correspond to per grid cell building volumes $m(i,j)$. To create a building in parcel k, we calculate its footprint F_k and its height h_k,. While the

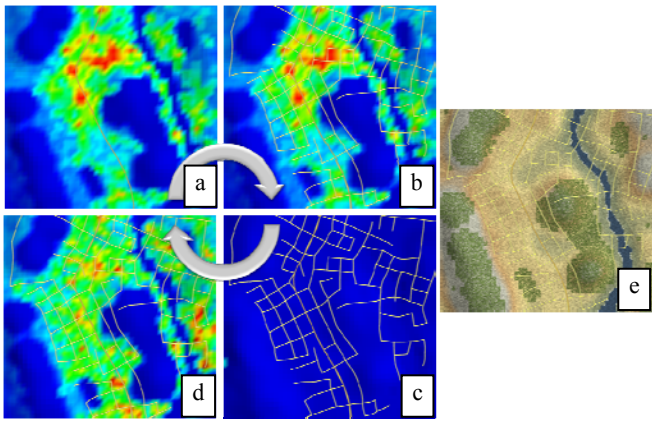

***Figure 6. Stability**. Starting from a specified population distribution (a), our system computes a model at equilibrium (b). Population is perturbed to a constant distribution (c) and then the system computes a similar equilibrium point (d) based on the roads and the terrain (e). Despite the perturbations, a similar equilibrium point is reached.*

latter step is done by our own interactive procedural method, the building type, the geometry of its footprint, and the building height can also be exported to an external application for offline photorealistic rendering (e.g., CityEngine, Mental Ray, etc).

The building type, which implicitly reflects land use, is determined by inspecting the ratio of number of jobs to the population size and a building type is chosen from a small database of procedural building styles. In particular, (i) a commercial building is assigned to a parcel with high job count and low population, (ii) a residential building is allocated to a parcel with nearly zero jobs and high population, (iii) an industrial building is placed on a parcel with a medium-level job count and nearly zero population, and (iv) a parcel with only a low population is assigned a house.

To calculate building footprint and height, the building volume v_k must accommodate a given amount of population and jobs. The volume v_k is proportional to the sum of the population and the jobs in each of the grid cells that are intersected by parcel k, multiplied by the fraction of area of each grid cell that the parcel occupies. To obtain a volume v_k, we first scale the building horizontally and then vertically. The horizontal expansion specifies the building footprint F_k; the vertical expansion yields the building height h_k.

To compute F_k, we assume a footprint is usually rectangular and is an approximate inset of the parcel that contains it. We first scale the footprint so its bounding box matches that of the maximum-contained box (MCB) of parcel k. Next, we let $h_k = 1$ be the initial height (in floors) of the building and A_k be the area of the MCB. If $A_k h_k < v_k$ then we let $h_k = \lceil v_k / A_k \rceil$. If $A_k h_k > v_k$, then we compute the inset of MCB such that $A_k h_k \approx v_k$. In the first case, a one-story building is not large enough to hold the required population and/or jobs, and thus more floors are necessary. In the second case, a one-story building occupying the area of the MCB is too large compared to the covered population and jobs. Thus, its footprint is scaled down. In some cases, our system randomly sets $h_k = 0$ so as to create an empty parcel that mimics a park.

6. RESULTS AND DISCUSSION

We have used our approach to interactively create and modify several synthetic urban spaces. Our system is implemented in C++ using OpenGL, a dual core 3.0Ghz PC, and a NVIDIA Quadro FX 1700 graphics card. The system is self-contained and no external software is required to reproduce our results. Behavioral modeling, parcel generation, and building generation use multiple CPU cores to perform computations in parallel. Grid sizes are determined by the designer. The processing time for the grid cells is very uniform thus no load balancing is performed. The update time during editing varies from a fraction of a second (for small grids) to a few seconds (for large grids, e.g., 15×15 kms). Given a terrain, the total design process for any of our examples takes five minutes or less. Most of the shown examples are also in our paper's video.

To test the stability of our urban design system, we alter a subset of the design variables and compare the states to which the system converges after each perturbation. In Figure 6a, we provide an initial population distribution and a simple sketched highway. Figure 6b shows the model at equilibrium as obtained by our system. In Figure 6c, we change the population distribution to a constant per-grid cell value and leave the road network unchanged. In Figure 6d, the system brings the model to an equilibrium point that is very similar to the one shown in Figure 6a. Since from two different perturbed states (Figure 6a-c) the system arrives at similar equilibrium points (Figure 6b-d), our system shows stability. The stochastic nature of our algorithms prevents the two equilibrium states from being identical yet their visual similarity is clear.

Our system completes an urban model from only a partial specification by the designer. This facilitates faster creation because the designer can focus solely on desired subsets of the model. In particular, Figures 7a-c show designer-specified landmarks consisting of the location of the downtown (specified by a single click of dropping a distribution of jobs) and sketched terrain, highways, and parks. Then, the system automatically computes a suitable distribution of the population (Figure 7d) and

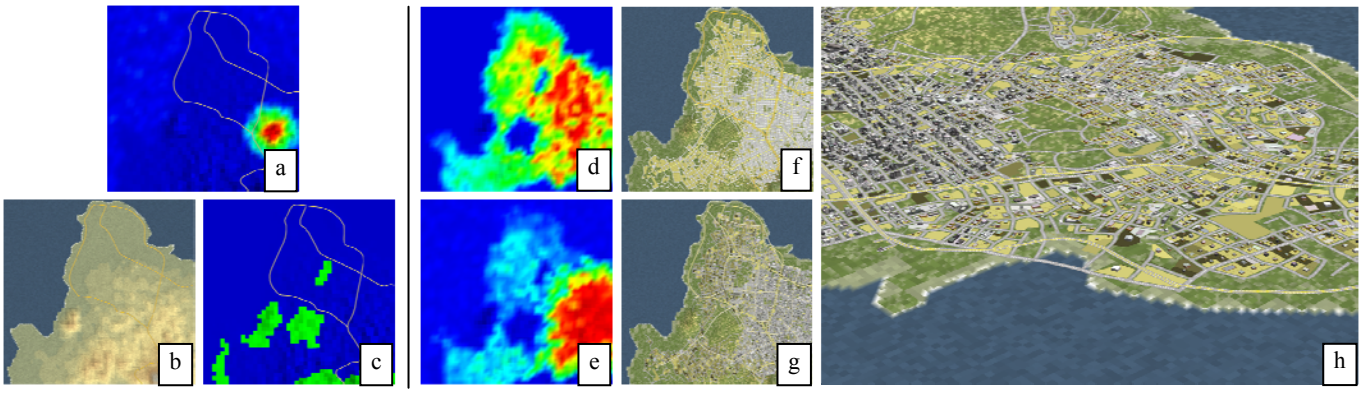

***Figure 7. Completion.** Taking as input a set of user-specified landmarks (downtown location (a), terrain and highways (b), and parks (c)), our system automatically completes the rest of the city. The process consists of first computing the behavioral variables (population (d) and jobs (e)) and then the geometric variables (blocks, parcels (f) and buildings (g)) while respecting the landmarks. A view of the resulting 3D model of the city is shown in (h).*

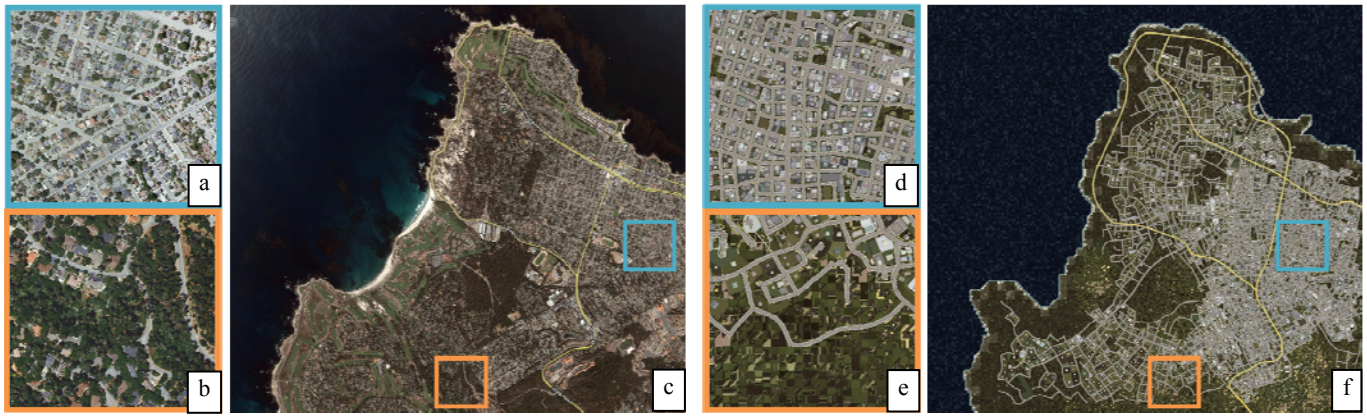

Figure 8. Validation. *Following the automatic completion process shown in Figure 7, our system generates a city that closely resembles a real-world city. In this case, a user is given a satellite view (c) of Pacific Grove, California, and is asked to draw the terrain and highways, click on a downtown location and highlight the parks. Our system generates the geometry of the roads and buildings (f) (colors are manually adjusted for comparison). Zoomed areas of the original (a, b) and synthetic (d, e) cities show how the density and style of the real city are reproduced at different locations, including high-density (a,d) and low-density (b, e) neighborhoods.*

jobs (Figure 7e) and generates a road network. The road network is filled-in with parcels (Figure 7f) and with buildings (Figure 7g) producing a final 3D model (Figure 7h). Once given the terrain, the entire design process took less than five minutes.

In Figure 8, we build upon the previous example and validate the result via a comparison to a real-world city. Figure 8c is a picture of a real city: Pacific Grove, California (USA). Figure 8f is the output from our system, showing that with a small amount of user input we are able to produce a plausible approximation of the actual city. The designer provided the same information as in the example of Figure 7 (i.e., sketched terrain, highways, parks, and downtown location), but also provided a similar total population and jobs count as in the real city. Clearly, the synthetic model will not match perfectly in all details, as shown in the insets for two small areas from approximately equal locations between the actual (Figure 8a-b) and the synthetic (Figure 8d-e) city. However, the overall patterns are quite similar visually and in terms of substantive attributes. We believe this to be a novel application of procedural generation of an entire urban area to approximate a real one, with very little input detail.

Our method also supports the use of constraints to generate complex urban models. Figure 1 shows an example two-step design sequence. The user specifies a terrain and low-density job distribution in the middle of a valley by a coast which results in an initial urban model (Figure 1a). As a design choice, the user wishes to create a larger city occupying the same terrain. More specifically, the designer wishes the downtown to have high-rise buildings and for the other developments of the larger city to be located outside the valley. Thus, the designer replaces the buildings with taller ones (Figure 1b-c) and constrains the downtown grid cells (Figure 1d). The system automatically increases the job count which in turn attracts more population. While new population would usually be located near the job center, the constraints force the additional developments to be elsewhere. Based on accessibility and land value, our system places the developments near the major roads connected to downtown and outside the valley (Figure 1e-f). The use of constraints enables closely controlling the modeling process and still benefitting from the automatic completion and assurance of producing plausible urban models.

Figure 9 shows an example of large multi-city urban space designed by our system and spanning over 200 km^2. The synthesis specifications can also be provided to existing tools to create more detailed buildings, highways, and photorealistic renderings (e.g., CityEngine [Procedural 2009], Mental Ray). For instance, Figures 9f-g were created by using a small set of parameterized building grammars and computing the parameters with our system.

7. CONCLUSIONS AND FUTURE WORK

We have presented an interactive system to design and edit 3D urban models. Our key inspiration is to close the loop between behavioral modeling and geometrical modeling producing a single dynamical system that assists a designer in creating urban models. After a user-specified change, the dynamical system attempts to bring the 3D urban model back into equilibrium, thus matching the demands of behavioral modeling with those of geometrical modeling. This ability enables the designer to only partially specify the design variables and have the system complete the rest. In addition, the designer can incrementally build the model and can explicitly constrain one or more design variables thus having both global and local editing control. Large and complex models can be produced in a few minutes or less.

Our framework has the following limitations: i) it includes a stochastic component which implies the exhibited behaviors and geometries are not exactly repeatable from session to session; ii) our dynamical system is subject to oscillations, though we ameliorate this by constraining parameter values and not straying far from equilibrium; and iii) over-constraining the system can lead to an unfeasible urban model (e.g., Figure 5d).

We are pursuing two items for future work. First, we are interested in including additional elements of behavioral modeling, such as a more sophisticated accessibility model. Second, we are seeking methods to generate more complex geometric structures using socioeconomic data such as generating additional building details.

8. Acknowledgements

This work is supported in part by NSF Grant 0753116, the Purdue Computing Research Institute, Purdue Research Foundation, and Adobe Inc. Also, we are very grateful to Aaron Link for his modeling help and to the reviewers for their valuable comments.

References

AASHTO, 2004. AASHTO Green Book - A Policy on Geometric Design of Highways and Streets, 5th Edition, *American Association of State and Highway Transportation Officials*.

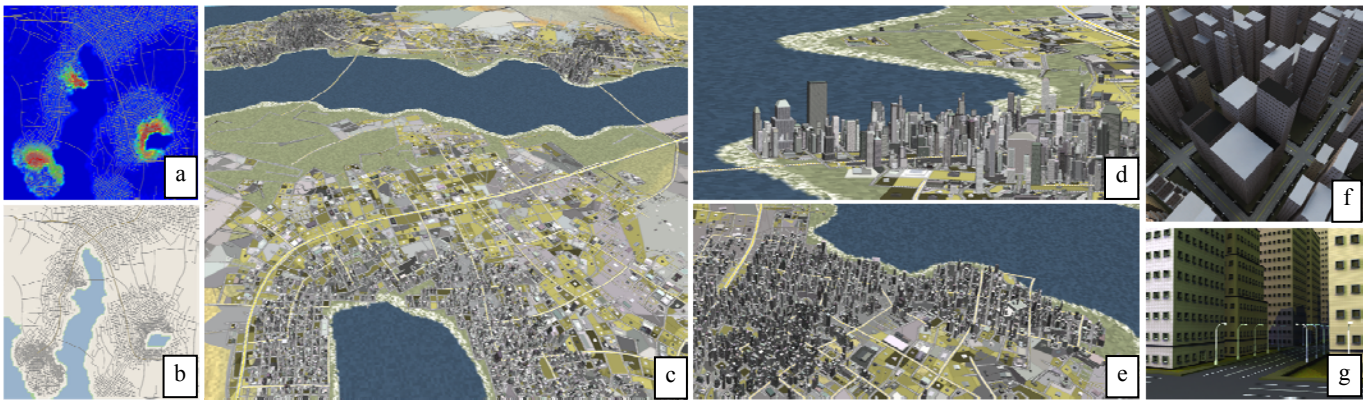

Figure 9. Design Example. *A designer draws a large terrain with mountains and water bodies, paints three main jobs clusters, draws two highways traversing the terrain, and specifies a population. Our dynamical system (a) moves the population to areas of high accessibility and forms three different cities. Our system (b) generates an adaptive road network. We show three different views of the road network, blocks, parcels, building footprints, and geometry (c, d, e). The 3D model is exported to CityEngine and rendered in Mental Ray for more detailed facades (f, g).*

ALIAGA, D. G., VANEGAS, C. A., AND BENEŠ, B. 2008. Interactive example-based urban layout synthesis. *ACM Trans. on Graphics,* 27(5), 160.

ALKHEDER, S., WANG, J., and SHAN, J. 2008. Fuzzy inference guided cellular automata urban-growth modeling using multi-temporal satellite images. *Int. J. of GIS* 22, 11-12, 1271-1293.

CHEN, G., ESCH, G., WONKA, P., MUELLER, P., AND ZHANG, E. 2008. Interactive procedural street modeling. *ACM Trans. on Graphics,* 27(3), 103.

HONDA M., MIZUNO K., FUKUI Y., NISHIHARA S., 2004. Generating Autonomous Time-Varying Virtual Cities. *International Conference on Cyberworlds*, 45-52.

LECHNER, T., WATSON, B.A., WILENSKY, U., TISUE, S., FELSEN, M., MODDRELL, A., REN, P., AND BROZEFSKY, C., 2007. Procedural modeling of urban land use. NC State Univ., CS-TR-2007-33.

LEONARD, K., CLARKE, C., GAYDOS, J. 1998. Loose-coupling a cellular automaton model and GIS: long-term urban growth prediction for San Francisco and Washington/Baltimore. *International Journal of GIS*, 12(7), 699-714.

LIPP, M., WONKA, P., AND WIMMER, M. 2008. Interactive visual editing of grammars for procedural architecture. *ACM Trans. on Graphics,* 27(3), 102.

MCFADDEN D., 1973. Conditional logic analysis of qualitative choice behavior. *Frontiers in Econometrics*, New York, Academic Press.

MĚCH, R. AND PRUSINKIEWICZ, P. 1996. Visual models of plants interacting with their environment. In *Proc of. SIGGRAPH 1996,* 397-410.

MERRELL, P. AND MANOCHA, D. 2008. Continuous model synthesis. *ACM Trans. on Graphics,* 27(5), 158.

MONTES DE OCA, N., LEVINSON D., 2006. Network Expansion Decision-making in the Twin Cities. *Journal of the Transportation Research Board: Transportation Research Record*, Volume 1981, 1-11.

MUELLER, P., WONKA, P., HAEGLER, S., ULMER, A., AND VAN GOOL, L. 2006. Procedural modeling of buildings. *ACM Trans. on Graphics,* 25(3), 614-623.

MUELLER, P., ZENG, G., WONKA, P., AND VAN GOOL, L. 2007. Image-based procedural modeling of facades. *ACM Trans. on Graphics,* 26(3), 85.

ORTÚZAR S. J. D. AND L. G. WILLUMSEN, 2001. Modelling transport, *Wiley*, 3rd edition.

PARISH, Y. I. AND MUELLER, P. 2001. Procedural modeling of cities. In *Proc. of SIGGRAPH 2001*, 301-308.

PORTUGALI J., 2000. Self-organization and the city. *Springer*.

PROCEDURAL INC. 2009. www.procedural.com.

PUTMAN, S., HASNOL ZAM ZAM, A., CHOI K., MCCARTHY W., YAN Y., 2000. Integrated Transportation and Land Use Policy Analysis for Sacramento, California, *Transp. Research Record,* Volume 1722, 38-47.

REYNOLDS, C. W. 1987. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics (Proceedings of SIGGRAPH 87),* 25–34.

SUNG, M., GLEICHER, M., AND CHENEY, S., 2004. Scalable behaviors for crowd simulation. *Computer Graphics Forum,* 23(3), 519–528.

TRAIN, K. 2003. Discrete Choice Models with Simulation. *Cambridge University Press*.

TREUILLE A., COOPER S., POPOVIĆ, Z., 2006. Continuum Crowds, *ACM Trans. on Graphics,* 25(3), 1160-1168.

TU, X., AND TERZOPOUOLOS, D. 1994. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of SIGGRAPH 1994,* 43–50.

VANEGAS, C.A., ALIAGA, D, BENEŠ, B, AND WADDELL, P., 2009 Visualization of Simulated Urban Spaces: Inferring Parameterized Generation of Streets, Parcels, and Aerial Imagery, *IEEE Trans. Vis. and Comp. Graphics,* 15(3), 424-435.

WADDELL, P., 2002. UrbanSim: Modeling Urban Development for Land Use, Transportation and Environmental Planning. *Journal of the American Planning Association*, 68(3), 297-314.

WADDELL, P., ULFARSSON, F., 2004. Introduction to Urban Simulation: Design and Development of Operational Models. *Handbook in Transport, Volume 5: Transport Geography and Spatial Systems*, Pergamon Press, 203-236.

WEBER, B., MULLER, P, WONKA, P, AND GROSS, M, 2009, Interactive Geometric Simulation of 4D Cities, *Computer Graphics Forum (Eurographics),* 28(2), 481-492.

WONKA, P., WIMMER, M., SILLION, F., AND RIBARSKY, W. 2003. Instant architecture. *ACM Trans. on Graphics,* 22(3), 669-677.