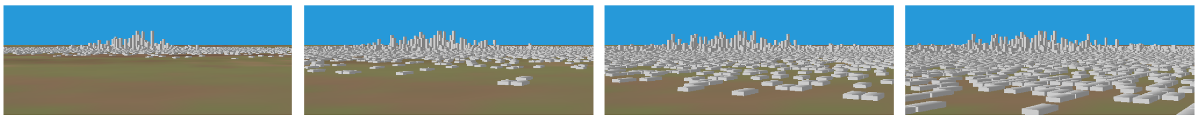
Interactive animation of cities over time

Paul C. DiLorenzo Victor B. Zordan Duong Tran

Riverside Graphics Lab
University of California, Riverside
pdiloren, vbz, dttran@cs.ucr.edu
http://www.cs.ucr.edu/rgl



Abstract

We present an interactive approach for animating the evolution of a city, adding to previous work in computer graphics related to automating the process of creating realistic-looking cities. Specifically, we combine the visual aspects of a graphical model with an optimization-driven solver to aid entertainers and urban developers alike in synthesizing the controlled growth and decay of a city over time. Our approach uses a hierarchical model based on primitive units of zoning and various heuristics taken from sociological and empirical data about real-world cities. We break down the city spatially into primitive units of zoning blocks, and the system performs optimization at this level to satisfy given constraints for growth (or decay.) Through this simplified model, animation for an entire city over a number of years can be performed quickly. To display the results, the system generates a visual representation of the city with buildings on the zoning blocks. To control the urban development of the generated city, the user selects parameters to control changes in the size of the population and land area used.

Keywords: Computer Graphics, Three-Dimensional Graphics and Realism - Animation, Simulation and Modeling

1 Introduction

In this paper, we explore the animation of a city over extended periods of time as it grows and evolves based on changes in population and land area use. To this end, we propose an interactive approach that uses a sociologically based model and a hierarchy of levels underlying it focused on faithfully capturing the visual artifacts as a city changes over time. By abstracting away unimportant details, we are able to create an evolving urban system that mimics many of the aspects apparent in real modern cities at interactive rates.

To model a growable city, our system takes advantage of two key features found in real urban settings. Nearly all urban areas have one or more city centers, steadfast business districts at the economic heart of the city, both metaphorically and physically. These city centers are filled with mostly commercial buildings and usually have the tallest buildings in the city, thus driving the look of the city to a large degree. Also, many cities are broken down into block-size, or larger, areas assigned based on zoning. These zones enable city planners to designate different areas for different uses, residential, commercial, municipal and so on. By looking at the city structure as being built up from these types of zoning units around city centers, we offer a representation that can easily be controlled by manipulating these two well-understood features and their attributes, as they change over time. We can also use these special 'building blocks' to describe a large amount about how urban cities develop fairly easily. For example combining multiple zones, we can create large parks or residential communities and breaking such zoning units into their smaller, constituent components yields buildings, yards, streets, and more - matching the real world.

With this model of the city, we cast the animation problem as a constrained, two-point boundary problem that must be solved in order to meet the expected growth (or shrinkage) based on the changes in population or land area use, as inputted by the software user. To solve this problem, we follow the approaches used to solve the thief's Knapsack problem, well-known in computer science, based on its likeness to our *evolution* problem. To show the power of our approach we reconstruct the development of the central basin in Los Angeles over an eighty-year span. With our system, we are able to capture the growth of the downtown city center of Los Angeles and, at the same time, recreate the development of *urban sprawl* indicative of growth in Southern California.

We anticipate approaches like the one presented will be useful in the area of urban development and sociology, since they can estimate the population and size of a city over a long time frame and include intuitive visual results. Social scientists and students alike can play out what-if scenarios with such a system and see how and what factors affect the growth and development of the synthesized city. Likewise, we hope this tool will find its utility in next-generation realistic environments for computer games and entertainment.

2 Background

The automatic creation of graphical city models has received some attention in recent years. With the popularity of urban-set games like 'Sim-City' and 'Grand Theft Auto' as well as the creation of large synthetic environments for the 'Matrix,' 'Terminator III', and 'Spiderman' movies, there is no doubt that the need for automatically generated, believable cityscapes is on the rise. The techniques proposed for the generation of such realistic urban models have either sought to rely on pure data recorded from the real world as in the project described by Teller et al. [1], through direct hand-crafted efforts like those of Hamill and O'Sullivan [2], via procedural approaches such as Parish and Müller [3] or some combination there-in. One popular hybrid approach combines simple 3D models with real photographs to create virtual architecture, as described by [4, 5, 6]. Also, the creation of Instant Architecture [7] follows heuristic rules from actual architectural design plus procedural grammars created to mimic the look of real world buildings. While our work most closely models that of Parish and Müller in the generation of our initial, static city model, none of these efforts have addressed the evolution of a city as it changes over time.

Urban development and planning as well as the study of sociological aspects of city units have given way to both a plethora of models for describing the growth and decay of urban settings as well as the need for visualizing such environments. We conclude from this work and that of Palen [8], that the factors that drive the evolution of a city are both vast and difficult to separate into simple heuristics like 'the population of modern cities is directly correlated with its wealth' for example or other unifying rules. While Batty and Longley have shown that the gross changes of real urban environments over time share characteristics with models of urban systems generated using fractals [9], they shed little light on the practical visualization of such models for use in computer graphics or for the novice user interested in understanding the growth of a city. As such, in our work, we glean insights from the experts about urban development such as the Burgess zoning model described by Palen and combine them with a grounded approach focused on generating a realistic, graphical urban settings that are able to evolve over time based on user constraints.

2.1 Urban Geography

Consulting texts in the area of urban studies [8, 10, 11], we found three basic models for the use of land in urban settings: the Burgess' concentric zone model; Hoyt's sectoral model; and Harris and Ullman's multiple-nuclei model. According to Palen [8], one early model rising from urban "spatial-organizational concerns" was the concentric-zone hypothesis proposed by Burgess in 1924 [12]. This hypothesis explains the segregation of land into business, manufacturing, and residential usages, as seen in modern industrial cities and purports that these cities grow radially in a series of concentric *regions* surrounding a central business district (CBD), as seen in Figure 1. The sectoral model suggests that, instead of zonal rings, cities develop in wedge-like sectors moving radially outward from the CBD and it makes allowances for growth along main arterial routes like highways and railways. The third, multiple-nuclei model says that land usage depends on a number of separate centers of activity for industry, manufacturing, entertainment and more. Because of its simplicity and well-defined structure, we found the Burgess model most appropriate for our animation purposes, although the sectoral model and multiple-nuclei models do contain interesting features and attractive avenues for future exploration.

3 Building a city

Following urban land-use theories, we model the city by explicitly placing different types of *zoning units* in discrete regions that match those proposed by Burgess, surrounding one CBD and potentially a number of other city centers. For simplification, we consider only two different zoning types, residential, which encompasses low-income, middle and upper class housing, and commercial/municipal, which includes business, manufacturing, and government land usage. We define *density* for the two types based on population density for residential and the number of employees per square area, analogous to the amount or 'size' of commerce in the commercial zones. The individual density of each zoning unit varies and the maximum-allowed density in each zone is based on its distance to the nearby CBD, following a normal-distribution Gaussian curve. The ratio of residential-to-commercial zones within each region is created heuristically, based on the described regions found in the Burgess model. (See Figure 1)

3.1 Origination of a simple city

To create the starting model following the Burgess hypothesis, our system starts from a user-defined CBD and constrains the placement of zoning units based on the desired population of the city, the average number of individuals

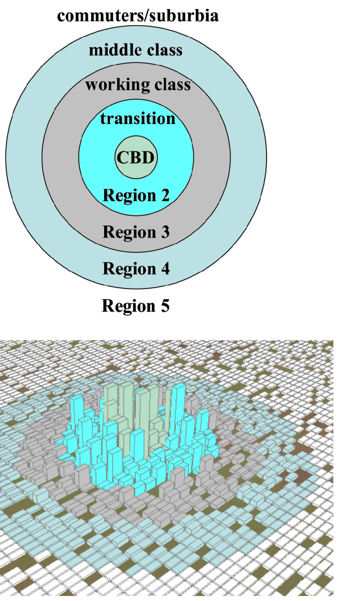


Figure 1: Burgess city model. Surrounding the central business district (CBD) are regions which move from highly commercial to commuter's suburbia. We approximate the ratios of residential to commercial land area for the regions outside of the CBD at 30, 50, 70,and 90 percent, respectively.

per household, and the overall span and developed area of the city. The model also relies on a number of fixed terms based on real cities related to the amount of commerce needed to sustain a given population and its difference between regions, the shape and size of the zoning units, and parameters associated with the population density and average size of domiciles within regions. For the simplest case, having a single CBD with no geographical obstruction, the system first creates empty (undeveloped) zones moving outward from the CBD until the city's span, area A_t, is met and then 'develops' zones randomly, weighted by their distance from the CBD, until the desired amount of developed area, A_d, is met. The relationship between A_d and A_t is controlled simply by the percentage of developed land in the city. To keep the development close to the CBD center, the distance is weighted by an exponential when selecting zones at random. During this initialization process, the system also randomly assigns zoning usages based on the percentages associated with the Burgess' region and density based on the fall-off curve:

$$\rho(r) = \rho_{max} e^{\frac{-r^2}{2\sigma^2 r_t}} \quad (1)$$

where the r is the distance from the zone to the CBD, r_t is the circular radius surrounding A_t and ρ_{max} and σ are the height and standard deviation of the Gaussian. A secondary pass through the developed zones makes small, uniform positive corrections in the densities of zones selected at random until the desired population is met. To do this, we compute the current population by summing up the number of people each residential zone holds based on density and the area size of the zone. This *origination* algorithm is outlined here.

```
originateCity(desired_pop,Ad,At){
    assignLandtoZones(At);
    while (Ad > 0){
        developZone(exp_random) ;
          //assigns density and type
        Ad --;
    }
    computePopulation();
    while (current_pop != desired_pop){
        adjustZoneDensity(random);
        computePopulation();
    }
}
```

3.2 *Additional features*

For more complex cities, we incorporate a number of additions to the simple city, to account for multiple centers and for geographic limitations by generating zones based on specific constraints. Since a real city with multiple city centers usually shows different size and growth rates among those centers, we allow the user to assign relative densities and rates of growth between different centers and the system uses these when assigning zones for each. Also, the zones of each center are given a unique orientation that may be assigned by the user or based on some geographic features, for example direction of the coastline.

Geographic features are specified through a coded *image map* of the terrain, where regions, such as bodies of water, represent areas where no urban zones may occupy and other, 'gray-scale' regions are used to specify changes in elevation. The system calculates slopes to determine the developable areas automatically based on a given steepness threshold. Once the set of assignable zones is determined for each center based on these various geographic constraints, the origination of the city follows along the method described above with the exception that ρ_i for a given zoning unit in center$_i$ may be replaced by the density from another center$_j$ if the computed density of Equation 1 is larger for that zone due to center$_j$. This adjustment avoids unwanted jumps in the density profile for the whole multi-centered urban region. The justification for this is straightforward when considering the *urban creep* across borders seen in areas surrounding metropolitan centers, for example the edges of New Jersey state near New York City.

To generate the appearance of the city from the zoning units described, we use heuristics to 'build' different numbers and sizes of structures appropriate according to the type of zone and its density. We describe this in the rendering section following the next section on the evolution of the city over time.

4 Animating growth and demise

To synthesize evolution, our system solves the two point-boundary problem between a city's current state and the new, desired state using the origination algorithm to 'establish' the initial city and an incremental optimizer which manages its construction and destruction over time. Once the desired city is 'built' as described above, the user makes specifications to control the evolution of the city over some period of time. Specifically, the user may change the overall population and the physical size (land area) of the city and dictates the number of years over which the changes should occur. Then, an optimization routine manages the construction and re-development of the urban zones based on a uniform 'growth schedule'. This schedule is simply the equal distribution of the overall desired change(s) over the given number of one-year spans. Then, during each year, the incremental optimization problem is to meet the scheduled changes without any errors, although, if errors associated with unmet changes do arise, they are simply rolled-over and incorporated into the subsequent year's growth.

To solve the incremental optimization problem, as stated, we compare the desired goal to the classic "Knapsack" problem, which is described in any algorithms text (for example, see [13]). Mapping this algorithm to our city-development problem, we introduce "Citysack".

4.1 Citysack

We define Citysack as equivalent to the classic problem, KnapSack, which states “A thief robbing a store finds n items; the i-th item is worth v_i dollars and weighs w_i pounds. He wants to take as valuable a load as possible, but he can carry at most W pounds in his knapsack. What items should he take?” or

$$max \sum_{1}^{n} v_i \;\; such\ that \;\; \sum_{1}^{n} w_i \le W \qquad (2)$$

Citysack is stated as “A city needs to grow (or shrink) by modifying some subset of n zoning units; the i-th zone has a current density value of v_i and potential w_i $(\rho(r) - v_i)$. In order to meet the growth demands and minimize the people displaced, the overall change in density should be minimized while the total change potential, W, meets the new population requirements. What zones should be constructed or destroyed and redeveloped?” or

$$min \sum_{1}^{n} v_i \;\; such\ that \;\; \sum_{1}^{n} w_i \ge W \qquad (3)$$

By casting the development problem very similar to the Knapsack problem, we can take advantage of its known solutions and characteristics.

In order to apply Citysack to meet our urban-development goals, we make a few simplifying assumptions. We treat all zones based only on the *density*, the most critical aspect with respect to changes in population and land usage, so that it can be isolated and most directly managed by the optimizer. We associate a non-trivial cost, equivalent to the density, for the destruction of a unit. This prevents the optimizer from destroying more than is necessary. We also considered adding the ability for Citysack to use zone-by-zone vs. partial destruction and re-construction within a zone which stems from its likeness to Knapsack in which the thief is allowed to take whole or partial items in his looting. However, in practice, the former, so-called ‘0-1’ problem resulted in solutions that were easier to manage and showed little visual difference.

4.2 Solving the Citysack problem

To solve the stated Citysack problem, we mimic the origination algorithm by first generating new zones if the size of A_t has increased and again randomly build until the new desired developed amount, A_d, is met. At the end of this stage, if the population supported by the city is as large as or larger than the target population, the evolution algorithm halts. The assumption here is that any excess developed space will be left empty and abandoned but this will not necessarily prevent new development from occurring. (This step is analogous to the thief taking items that have value but weight nothing) If there is a deficit in the supported population, the evolution step must include some destruction and redevelopment.

The Knapsack problem can be solved in a number of ways. In [13], they propose using dynamic programming to find a globally optimal solution for the *‘0-1’ Knapsack* problem. We decided to use locally optimal methods because the computation constitutes several orders of magnitude speed-up over dynamic programming ($\mathbf{O(n\log(n))}$ **vs.** $\mathbf{O(nW)}$). We propose two algorithms and apply the best solution of the two. The first is a so-called *greedy* approach that picks the zone with the greatest potential improvement and redevelops that zone to reach its maximum potential (density.) This cycle repeats until the target population is met or all potential improvements are made. The second is a *conservative* approach where the zone with the smallest density-to-potential ratio is improved to its maximum potential, again repeating until the target population is met or all potential improvements are zero. While the conservative approach usually resulted in a lower cost neither scheme worked best in all cases. Thus, after calculating, the algorithm compares the solutions and applies the one with the lower cost. The schedule advances along with any resulting deficits or surpluses to the following year. The pseudocode below outlines this incremental animation algorithm.

```
SolveCitysack(desired_pop, newAd, newAt) {
  //run for each year in growth schedule
   assignLandtoZones(newAt);
   while (newAd > 0){
          developZone (exp_random);
          newAd--;
   }
   computePopulation();
   if (current_pop < desired_pop){
          runGreedy();
          runConservative();
          updateCity(); //based on lowest cost
   }
}
```

5 Rendering and Results

To generate the final look of the city, zones are decomposed into individual buildings. The decomposition from zone to building is dependent on the assigned density of the zone and on the equivalent building or set of buildings that would occupy such a density. To simplify the placement of buildings, all structures in a particular zoning unit are assigned the same height value. Obviously, this is not in general true, although there is often a high correlation between the heights of neighboring buildings. The height of a building, which drives the size of its footprint and subsequently the number of buildings per zone, is derived from the assumption that a uniform number of people will occupy each floor of a given building and so, the proper number of floors, therefore, may be computed based on the specified density of the zone or the integer number of floors is $round(\rho h)$ where factor h is normalized based on the maximum density ρ_{max} and the max number of floors allowed in the city. Thus, for example, if a particular zone is close to the maximum density, a single large skyscraper would be assigned to the zone. But, a zoning unit with a much lower density will be decomposed into several shorter buildings. For the high-resolution models shown in figure 3, we use the commercial libraries of Marlin Studios [14] for real-looking urban and suburban structures.

To show the progression of a city's evolution over time, we use simple edits that 'fill in' empty 'lots' with buildings or cross-dissolve between one building and its replacement. Again, this is a simplification, but it is not practical to animate actual brick-laying or demolition. At the rate of the animation, about a year every second or so, we assessed that the simplest animation gave way to the least amount of distractions and the most pleasing results.

LA Basin. In Figure 2, we show the development of the Los Angeles basin over an 80 year span. Urban sprawl, that is common in the southern California area, begins to become more prevalent as time goes on. In Figure 3, we show three close-up views of Central Los Angeles. The top close-up image is from our interactive software, decomposed into buildings, showing the residential and commercial buildings in red and blue, respectively. The middle close-up image is a rendered version of the model generated. The bottom image is a photograph of Central Los Angeles. In Figure 4, we show the Santa Monica area, the leftmost city center in Figure 2. Santa Monica is nestled between the ocean and the mountains and therefore is constrained by these geological obstacles, and forced to move inland to build. We colored the zoning units in four different ways (Burgess model, density, type, age) to show the depth of the model.

6 Conclusions

Animating the development of any urban environment requires serious concessions - in only modeling select, pertinent processes and ignoring many details to keep the computations fast. By simplifying the animation of cities to the incremental adjustment of 'zone' density and comparing the optimization of this to a well-known algorithm, we are able to find solutions that are easy to understand and implement but still produce complex graphical results because of a visual model that is interpreted from the underlying zone structure. Of course, these results are constrained to expose the particular artifacts important to the problem we chose as the focus of this paper, i.e. animating population and area changes over time.

We expect to improve this system and its implementation as our current system does get sluggish when animating the large city scenes, like the Los Angeles basin example, but should be able to find speed-up by exploiting repeated patterns both in the program calculations and in the city model itself. Even so, without speed-up, it runs at 2.5 years per second with full OpenGL graphics on a 2.8 Gz Pentium 4 processor with NVidia TI 4600 for a single-center type city with no geographical obstructions (see the image under the title). By adding main transportation networks, such as large roads or highways, we can use sociological models such as the sectoral model. Also, the addition of inner roads would increase the visual appeal of the zoning units. We anticipate that trees and parks, as well as people and traffic, would add immensely to the visual appeal of the final results. Although we do currently account for 'empty' undeveloped land, the overall results arguably appear stark. Also, our city system currently ignores factors like fire damage, demolition, and vandalism as well as many other known and documented artifacts that influence a city. With proper demographics, we could likely make the final data look more realistic and possibly convey more information through its animation.

We see that such city development software could be used in number of scenarios. In the area of entertainment for computer animated movies that take place over time, in a city setting the software could be used to create a consistent urban world with a past, present, and future. With proper validation, such software could also be used in sociology - understanding how a city works, armed with measured data, a sociologist could play out careful scenarios to assess how different factors affect real cities. The software could aid in city planning when government officials want to visualize (or show) how their choices change things and what their potential consequences may entail. Note, we are not claiming this real-world problem-solving is validated in the models we present currently. But, by basing our system on a steadfast sociological theory like Burgess' and adding other known/proven constraints, this type of city animation could become a valuable tool for professionals. Through easy-to-control, interactive visualizations of city evolution, even simple animation models can allow novices to learn and understand the workings of a growing city in a natural and intuitive way.

Acknowledgements

The primary author's funding was provided by a UCR COE's Dean fellowship.

References

[1] Seth Teller. Mit city scanning project: Fully automated model acquisition in urban areas. http://city.lcs.mit.edu//city.html.

[2] John Hamill and Carol O'Sullivan. Virtual dublin - a framework for real-time urban simulation. Journal of WSCG, Vol.11, pp 221-225.

[3] Yoav I. H. Parish and Pascal Müller. Procedural modeling of cities. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 301–308, August 2001.

[4] William Jepson, Robin Liggett, and Scott Friedman. An environment for real-time urban simulation. In *1995 Symposium on Interactive 3D Graphics*, pages 165–166, April 1995.

[5] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proceedings of SIGGRAPH 96*, pages 11–20, August 1996.

[6] William Ribarsky, Tony Wasilewski, and Nickolas Faust. From urban terrain models to visible cities. *IEEE Computer Graphics and Applications*, 22(4):231–238, July 2002.

[7] Peter Wonka, Michael Wimmer, François X. Sillion, and William Ribarsky. Instant architecture. *ACM Transactions on Graphics*, 22(3):669–677, July 2003.

[8] J. J. Palen. *The Urban World*. The McGraw-Hill Co. Inc., 1997.

[9] M. Batty and P. Longley. *Fractal Cities*. Academic Press, 1994.

[10] R. Northam. Urban geography, 1975. ISBN 0-471-65135-4.

[11] E. B. Phillips and R. T. Legates. City lights: Introduction to urban studies, 1981. ISBN 0-19-502797-3.

[12] E. W. Burgess. The growth of the city: An introduction to a research project. *Publications of the American Sociological Society*, 18:85–97, 1924.

[13] T. H. Cormen, C. E. Leiserman, and R. L. Rivest. *Introduction to Algorithms*. The M.I.T. Press, 1994.

[14] T. Marlin. Marlin studios, premium 3d models. http://www.marlinstudios.com.

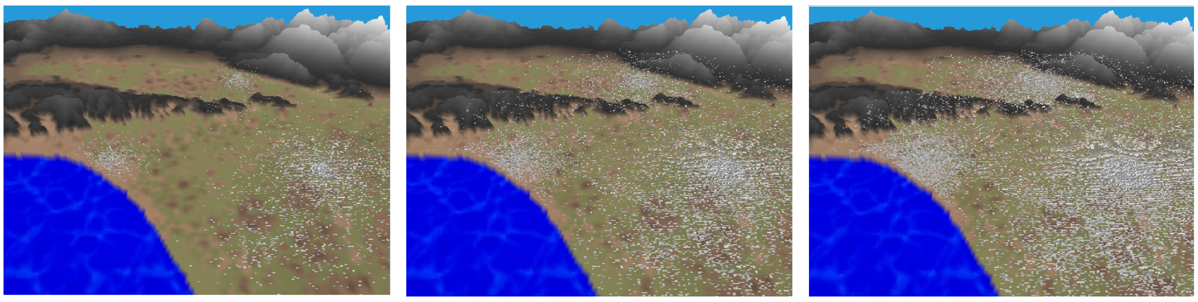


Figure 2: Los Angeles basin over 80 years.

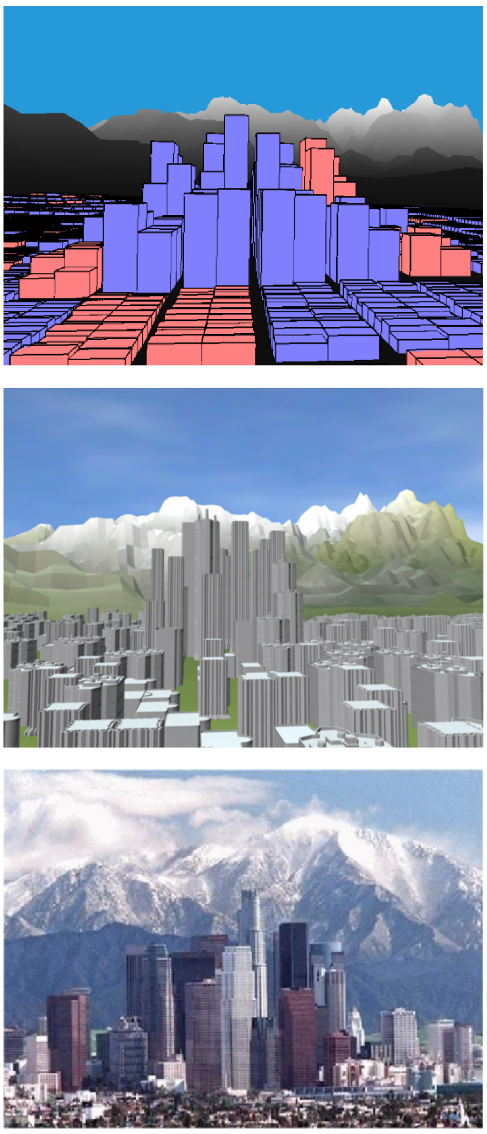


Figure 3: Top: Central Los Angeles Simulated. Middle: Central Los Angeles Rendered. Bottom: Central Los Angeles.

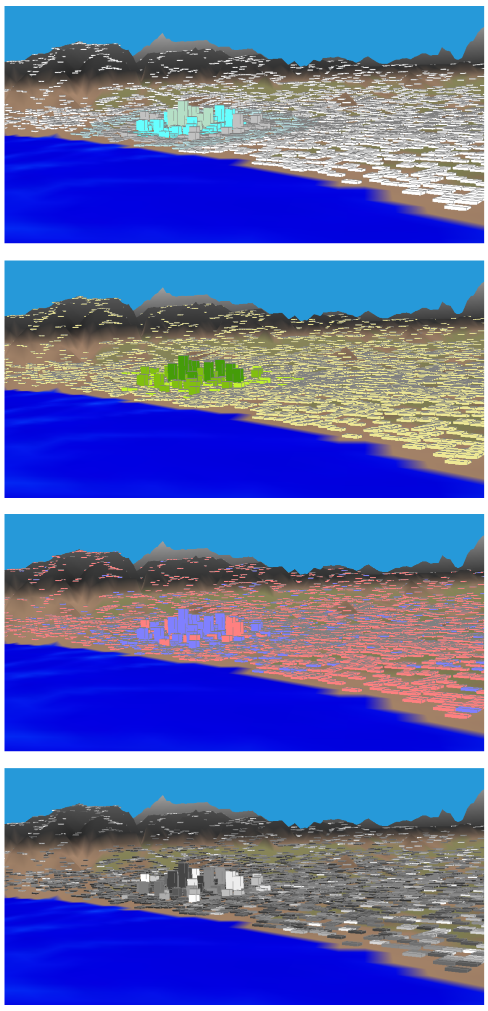


Figure 4: Close-up of Santa Monica area from the simulated growth of the LA basin. Top: Burgess model coloring as seen in Figure 1. Second: Density of zoning units, lighter colors are less dense. Third: Type of zoning unit, blue commercial and red residential. Bottom: Age of zoning unit, dark represents older construction